

Design and implementation of teaching activities in Discrete Mathematics: an experience in the first cycle of education.

Aaron Gaio

Dipartimento di Matematica e Informatica

University of Palermo

A thesis submitted for the degree of

Philosophiæ Doctor, PhD

Corso di Dottorato in Matematica e Informatica, XXX ciclo

Università di Catania, Messina e Palermo

2018

Tutor: Dr. Cerroni, Cinzia

Co-tutor: Dr. Di Paola, Benedetto

Co-tutor: Dr. Ferrari, Pier Luigi

1. Reviewer: Dr. Ferrara, Francesca

2. Reviewer: Dr. Fontanari, Mauro

Day of the defense:

Signature from head of PhD committee:

Abstract

Within the subject of discrete mathematics, some of the topics and mathematical content can be presented, with interesting results even to younger children, in the first cycle of their education. Purpose of this document is to show how we designed and implemented some teaching tasks in various schools and grades, and the results we got in terms of enhancing some aspects, such as cognitive processes involved, task design, affection of students towards mathematics, connections with the curriculum.

After a presentation of the current situation, in Italy and some starting points from all around the World, some survey results are shown, to better understand the point of view of the actors of this, teachers. A whole chapter is dedicated to the mathematical content that lies at the base of these topics,. A description of the research methodology and theoretical framework that were used follows; this project uses design research, a mainly qualitative research method, to better embed the processes of task design in a more research-oriented vision, analyzing also many of the cognitive processes involved by the tasks we are designing. The teaching tasks that were used, and developed during the research, are presented, with connections between them and an effort on creating some sequences of tasks with a common learning object. After an overview of the whole project, some of the tasks, and worksheet used, are presented in more details. A chapter with the results we observed is following, trying to give some overview conclusions, still providing examples.

The author believes that this is a starting point for some future education research in the field; discrete mathematics proved to be appropriate and to create and interest towards important topics of mathematics, such as some aspects of algorithms, efficiency, complexity of a problem, abstraction, error correction, which are relevant in many field of mathematics, and not only mathematics. Moreover, the kind of activity proposed raised the attention on affective issues towards mathematics. We can

notice that problems posed as games, as storytelling and using manipulative objects, can enhance the conception children have of mathematics, which plays a central role in the teaching and learning of it.

Acknowledgements

Un primo ringraziamento va allo staff di didattica della matematica dell'Università di Palermo, con la professoressa Cinzia Cerroni che ha coordinato, come tutor disponibile e paziente, il mio percorso di dottorato. Benedetto di Paola è stato un po' il mio guru in questi 3 anni, ed è a lui che devo ogni contatto con altri ricercatori, esperienza all'estero, seminari, conferenze, presentazioni e corsi. Una pazienza e gentilezza da parte di entrambi, quando li cercavo con insistenza in Dipartimento a Palermo e quando invece, al nord-Italia, non mi facevo trovare.

Grazie a Riccardo ed Alice per i primi contatti nel mondo “didattico”, ad Andrea per avermi davvero passato tutte, e di più, le informazioni possibili su come e cosa fare in qualsiasi situazione, a Roberto e tutto il gruppo di Salerno per la sempre fantastica accoglienza, a Chiara, Federica, Laura, e tutto il gruppo bolognese per le innumerevoli trasferte in giro per l'Europa, oltre a tutti gli altri ricercatori e dottorandi conosciuti, italiani e non solo, partendo dall'esperienza in Svezia, alla Spring School di Wurzburg, fino alla scuola estiva YESS della Repubblica Ceca.

Grazie a tutti a casa in Trentino, Luana, Marco e nonno Giacomo, sempre di supporto anche quando, “albergo-style”, mi facevo vedere in casa solo per mangiare e dormire; al mio fratellino Iago e a zii e cugini con cui avrei potuto passare molto pi tempo.

Grazie alla disponibilità professori ed insegnanti dell'Istituto Comprensivo di Primiero e dell'Istituto Salesiano Santa Croce e un grazie a genitori e bambini (sono loro, alla fine, i protagonisti di questa tesi), che mi hanno supportato, sopportato e sperimentato con me questa matematica un po' pazza che alla fine tanto ci ha fatto divertire insieme.

Contents

List of Figures	vii
1 Introduction	1
1.1 Background	1
1.2 Motivation	2
2 Aims of the project	5
2.1 Research goal	5
3 Current Situation	7
3.1 Indicazioni Nazionali del MIUR: The Italian Ministry Guidelines . . .	7
Details about mathematics in grades 1 to 8	8
3.2 Key competences for lifelong learning - European Union	9
3.3 What has been done in Italy so far?	11
3.4 A preliminary survey: teachers' competences and difficulties	11
3.5 This survey and the research project	16
3.6 Survey's conclusion and some more motivation for the project	16
4 Mathematics content: Graph Theory and Algorithms	19
4.1 Algorithms and Computational Complexity	19
4.1.1 P vs. NP	22
4.2 Coloring maps and graphs	25
4.2.1 Introduction to graph theory	27
4.2.2 The two-colors problem	33
4.2.3 The three-colors problem	38
4.2.4 A Gröbner basis algorithm for the three-coloring problem . . .	39
4.2.5 Complexity of this problems	41
4.2.6 The four-colors problem	42
4.3 Cryptography	45
4.3.1 Cryptography in history	45

4.3.2	Basic Cryptography, private and public keys	47
4.4	Coding Theory	52
4.4.1	Error Correcting Codes	54
4.4.2	Perfect Codes and Hamming Codes	56
4.5	Topics in graph theory	62
4.5.1	Hamiltonian and Eulerian paths	62
4.5.2	Minimum weight spanning trees	64
4.5.3	Minimum dominating sets	69
4.5.4	Connection with one-way functions	71
5	Research Methodology and Framework	73
5.1	Design Research	73
5.1.1	An overview	73
5.1.2	Key Features of Design Research	74
5.1.3	Details of the methodology	76
5.1.4	Positive aspects	77
5.1.5	Critics	78
5.2	A constructivist framework	79
5.2.1	Cooperative learning and the ZPD in a social constructivist environment	80
5.2.2	The theory of didactical situation	81
5.3	A little about task design in education	83
5.4	Is design research appropriate for my research?	85
5.4.1	Design Research and a Socio-constructivist approach	85
5.4.2	Design Research towards the design of innovative tasks	85
5.5	Methods of video selection and analysis	86
6	Teaching Experiment	87
6.0	Pilot teaching experiment	87
6.0.1	Description	87
6.0.2	Development and preliminary results	90
6.1	Graph Theory and Cryptography	92
6.1.1	Map coloring	92
6.1.2	Graphs and dominating sets	99
6.1.3	Binary numbers and Error Correcting Codes	105
6.1.4	Public Cryptography from dominating sets	108
6.2	Computational Complexity of algorithms and Cooperative Learning .	112
6.2.1	Searching Algorithms in Primary School	112

6.2.2	Sorting Algorithms for 4 th , 5 th and 6 th grade students	113
6.2.3	Computational complexity in graph theory	116
6.2.3.1	Roads in the city and the Traveling Salesman problem	117
6.2.4	Cooperative Games	119
6.3	Programming and pre-programming for primary school, Scratch based or Unplugged?	121
6.3.1	Scratch-based teaching and learning	123
6.3.2	Unplugged approach, teaching and learning	125
7	Results reached and final discussion	129
7.1	Graph Theory and Cryptography	129
7.2	Computational Complexity of algorithms and Cooperative Learning .	143
7.3	Programming and pre-programming for primary school, Scratch based or Unplugged?	155
8	Students materials and appendices	163
	Bibliography	179
	Bibliography	179

List of Figures

3.1	Some screenshot of the survey proposed - part1.	12
3.2	Some screenshot of the survey proposed - part2.	13
3.3	Table 1. Knowledge about Cryptography, in their previous school/universities studies.	14
3.4	Table 2. Keywords to connect to Cryptography/Algorithm.	14
3.5	Part of the analysis of the answers received, divided into argument clusters.	15
3.6	Table 3. Interest in teaching these topics.	15
4.1	an illustration representing the Sieve of Eratosthenes. Working on a table with initially all natural numbers, we begin by taking out the even numbers (green), then the multiple of 3 (blue), then 5 (orange), 7 (purple), 11 (yellow) and so on, until obtaining just the prime numbers up to a given integer.	20
4.2	a diagram of the different complexity classes we are discussing.	24
4.3	an example of this map coloring, in a political map of Europe.	26
4.4	the bridges of Koenigsberg.	27
4.5	Euler's model of the town of Koenigsberg.	27
4.6	some examples of simple planar graphs.	28
4.7	examples of some different kind of graphs.	29
4.8	the pentagon and the star are not the same graph, however they are isomorphic, i.e. representing the same situation just with a relabelling of the edges.	30
4.9	a complete bipartite graph on the left and simple bipartite one on the right.	31
4.10	an example of the flights schedule planning: the colored lines are the schedules of flights and the the graph is drawn so that when two flights are conflicting with each other the vertices are connected.	31
4.11	the red graph is the dual of the black one and viceversa.	34

4.12	an example of how to start from a map, get the correspondent graph and then also the bipartite graph.	36
4.13	on the same map, we get two different colorings, depending on how we choose the order of the areas when applying the greedy algorithm. This algorithm is easy and works to find a solution but not the optimal one when trying to minimize the number of colors.	37
4.14	a possible way of coloring this example and its relative graph.	38
4.15	a possible way of coloring this example and its relative graph.	38
4.16	the graph we are considering in the example and its relative map.	40
4.17	up to permutations of the colors, the two different solutions of the example.	41
4.18	the vertex v and its other “surrounding” vertices.	43
4.19	the situation in which there is a cycle connecting v_1 and v_3 ends with making another cycle connecting v_2 and v_4 impossible	43
4.20	an example of a map of the USA colored with four colors.	44
4.21	Gardner’s proposed problem and the solution obtained by Wagon in 1998.	45
4.22	an example of a transposition cipher using a scytale.	45
4.23	an example of a Caesar’s cipher shifting every letter by 3.	46
4.24	an example of a Vigenère cipher with key “CRYPTO”.	46
4.25	the ADFGVX table.	47
4.26	In the first example 1-2-3-4-5-6-7-5-3-7-2-6-1 is a possible Eulerian path and circuit; in the second graph it is not possible to find such an example.	62
4.27	The edges in red show an example of a Hamiltonian path along the edges of a dodecahedron such that every vertex is visited a single time, and the ending point is the same as the starting point. This is the solution of the problem known as the “icosian game”, invented by W.R. Hamilton.	63
4.28	An example of a graph that is also a tree.	64
4.29	A graph G . Figure A is not a spanning tree (since it is not a tree for G); B and C are some possible spanning trees for G	65
4.30	The graph G	67
4.31	Finding the minimum spanning tree for G with Kruskal’s algorithm, after adding respectively one, three and all five edges needed for the final solution.	68

4.32 Finding the minimum spanning tree for G with Prim's algorithm, after adding respectively one, three and all five edges and vertices needed for the final solution.	68
4.33 A graph G . The set of the red vertices in Figure A is not a dominating set for G (since the vertex 7 is not dominated by any of the red vertices); B is a possible dominating set for G	69
4.34 A graph G and the attempts A and B as described in the example 4.21.	70
4.35 The graphs in the example 4.22.	71
4.36 Steps in the construction of a graph with a perfect dominating set solution.	72
5.1 Cycles of iteration in DR, following McKenney, 2001	76
6.1 The cards used for the game described, we have them printed in bigger size.	105
6.2 The cards used for counting with binary numbers; they were black on the other side. Usually just the first five or six were used.	106
6.3 Encryption of a numerical message, part 1.	108
6.4 Encryption of a numerical message, part 2.	109
6.5 A decryption key, which is a perfect dominating set for the graph. . .	110
6.6 Trying out the balance scale problem in practice.	114
6.7 Students involved in the ordering process involving the sorting network.	115
6.8 Discussing the sorting network problem.	115
6.9 The Colored Ball Game activity with colored spheres.	116
6.10 An overview of the city and the roads, with children in action to solve the problem.	117
6.11 A part of the TSP worksheet.	118
6.12 Reorganizing their place in the right chairs.	119
6.13 Playing in the garden with hula hoops circles.	120
6.14 Table 1: Classes involved in different grades, with students numbers and curriculum used.	122
6.15 Kids using MIT's Scratch.	123
6.16 Kids using code.org's online software.	124
6.17 Kids exploring the binary code.	125
6.18 Kids coloring with pixel art.	125
6.19 Setting for the kids to give instructions to the others.	126
6.20 Giving instruction to get the kid-robot out of the maze.	127

LIST OF FIGURES

7.1	A child involved in the different map coloring problems.	129
7.2	A child involved in the graph coloring problem.	130
7.3	An example of a possible coloring for the U.S.A. map, obtained by one of the students.	131
7.4	The comparison between a situation where it is possible to have just three colors and one where I can say that it is not possible.	132
7.5	Children involved in the “graph drawing” activity.	133
7.6	Examples of different figures. The green one works, while the red one does not.	134
7.7	Orienteering course to make the “follow the line” problem more real.	135
7.8	The minimum dominating set solution of the Firestation problem. . .	136
7.9	Constructing binary numbers.	137
7.10	Binary writing turns into a picture.	138
7.11	0 means empty and 1 is to color.	139
7.12	Younger children really need to be careful when following the right lines.	140
7.13	Children in the process of ordering themselves in a circle, following the alphabetic order, by first name.	145
7.14	Trying out the weight problem in practice resulted in an effective way to test the results.	146
7.15	After practicing with the mat, we made it more “mathematical” with a paper sheet.	147
7.16	Students discussing the properties of the sorting network.	148
7.17	The game situation from a graph theory perspective.	148
7.18	Stuck in a “greedy” configuration.	149
7.19	“Muddy city” problem, as presented in (BWF98).	151
7.20	A 23 pieces solution is the best result we can achieve.	151
7.21	The problem of paving the city get easier as they can walk on it. . . .	152
7.22	The Traveling Salesman Problem in its orienteering version, with one of the students’ solution.	153
7.23	Giorgia saying “Let’s go!” after she realizes where the best new step will lead her. Is it a first step towards the abstraction of a solution to the problem?	154
7.24	Showing the transition from sequential instruction (red) to iteration (blue).	155
7.25	Showing once more transition from sequential instruction (n.2) to iteration (n.3).	156

7.26	In a programming environment as Scratch, the passage between single instruction, multiple instruction and iteration to get, circa, the same movement.	157
7.27	Student here is following her own thoughts instead of the direction received (cannot be seen from the picture, but just got the instruction to turn right).	158
7.28	Trying to make the computer situation real, using the hand to impersonate the character.	159
7.29	Wrong direction for the white piece.	160
7.30	Here the mistake is in the word “on” or “over”, which in Italian has the same word for the two meanings.	160
7.31	Kids working on the Scratch storytelling.	161
7.32	“Programming” one of the scenes, before implementing it in Scratch.	161
8.1	Video analysis of one of the ordering games.	173
8.2	Video analysis of one of the instruction games.	173
8.3	The sorting network activity in one early stage of development.	176
8.4	Final mat for the sorting network activity.	176
8.5	After the unplugged tasks, also a lot of computer “pre-programming” was developed.	177
8.6	Video analysis of Scratch programming.	177

LIST OF FIGURES

1

Introduction

1.1 Background

Discrete Mathematics, with graph theory and cryptography, together with various algorithms found in computer science and computational complexity, can be a great teaching topic in lower school grades. I had this feeling after studying this kind of mathematics at university level, and seeing that it got attention also from a non-mathematical audience.

Looking at the past years, some projects about this have been tried around the World (works by Hart, 1990; Kenney, 1991; Rosenstein, 1997, Casey et al., 1992; Bell, Witten, Fellows, 1998-2015; (H⁺90), (KH91), (RFR97), (CFBW92), (BWF98)), but we feel that not much reached the Italian education system. Discrete mathematics is not clearly delimited in our curriculum and teachers are usually not aware that it actually could be.

Some new trends in teaching computer science in elementary education are recently emerging; elementary computer science is gaining attention both from the computer science education research community and from the elementary school teachers and heads (Franklin et al., 2015, (FHD⁺15)). The question about how young children learn computer science is still a new area of research; and providing effective learning opportunities to K-5 students is a big challenge (Gelderblom Kotze, 2009, (GK09)). While a good amount of examples are present in higher school grades (H⁺90), thinking about areas on the edge between mathematics and computer science such as discrete mathematics, graph theory and cryptography, together with various algorithms found in computer science, these can be great teaching topics also in lower school grades. We feel that not much did reach our national education system. Topics in computer science and discrete mathematics are not clearly delimited in our curriculum and teachers are usually not aware that they actually could. We are

thinking of our work as able to enhance the study of teaching and learning skills of mathematical practice through discrete mathematics problems, both general skills, such as reasoning and modeling, and skills particular to discrete mathematics, such as algorithmic and recursive thinking.

Topics difficult enough to be taught in college and university courses can be really dealt with in a primary school classroom? Well, that seemed enough a challenge to begin some research with.

1.2 Motivation

Briefly, the goal of this work is to study the learning of mathematical skills through the teaching of discrete mathematics. We are studying both the teaching of general skills, such as reasoning and modeling, and skills particular to discrete mathematics, such as algorithmic and recursive thinking.

The main research problem is to therefore support a proposal to alleviate the substantial lack of (sequences of) tasks in the Italian school curriculum about discrete mathematics, computer algorithms and cryptography, especially for primary and middle school. Both in the school programs and in textbooks, activities of this kind are missing almost entirely, despite much agreement that they can be really useful in improving the skills mentioned above.

Going back a little bit into “how did this develop”, the idea started just with the goal of creating some fun mathematics activity to present to young children this kind of topics. It then got more interest among some friends, teachers and educators to make so that young children could dive into the world of computer science, learning something about computer and computer functioning even before they start to program real programs and apps.

After some tryouts, still as just spots activity and not into the real curriculum, we noticed that this kind of games and tasks made children experience the same challenges that computer scientist and mathematicians do experience.

That sounded really good in a “let’s show the World what we are doing at the university” goal.

After that, we also became aware that this could have a bigger role into the national curriculum and started to get more formal with the activities, building longer sequences of tasks to achieve fixed goals, and also making the necessary connection with curriculum indications, which we will point out in the next chapters.

The other aspect of this research is more oriented towards the mathematics education research environment; i.e., the question that came to our mind is how will children react to this kind of activities? How will they improve their computational thinking skills? What about problem solving in mathematics education?

The final product is made of many different tasks and possible sequences made of those, which are presented in this thesis. The hope is that they could be used as a guideline for teachers, even curriculum developers, and, on the other side, a first step for researchers to see how the pedagogical approach to the teaching and learning of computational thinking, coding and so on will develop in the next few years.

2

Aims of the project

As we said in the introduction, our main research problem lays its groundwork in a proposal to implement activities in the national school curriculum about computer science, discrete mathematics and algorithms, with a focus on tasks for primary and middle school. Both in the school programs and in textbooks, activities of this kind are missing almost entirely, despite many agree that they can be really useful to improve the skills mentioned.

2.1 Research goal

The research project presented in this document has mainly two aims: RESEARCH and DESIGN.

Design

The design involves the creation of sequences of tasks, in the topics listed in the introduction and later dealt with in Chapter 4, to be used in the school curriculum, with some common and clear characteristics:

- Real Computer Science, and not “use of the machine only” involved;
- Learning by doing methodology is applied;
- No specialized props, nor particular knowledge required by teachers (not really even the computer is required);
- Problem solving based;
- Cooperative and inclusive.

Research

On the other side, by doing this in a research design paradigm (see 5.1), we have a goal in the mathematics education research environment.

Here we can analyze both the approach of design research on a curriculum perspective (MNvdA06), with its leading objective to improve the understanding of how to design for implementation, seeing design principles as important as curricular products in the result. As they write: “[...] insights are sought on how to build and implement consistent, harmonious and coherent components of a robust curriculum [...]”.

Over this, in the multiple cycle of research, we also have a goal to observe some psychological and pedagogical aspects in the learning of children, such as dynamics in cooperative games and learning of problem solving in mathematics.

3

Current Situation

3.1 Indicazioni Nazionali del MIUR: The Italian Ministry Guidelines

The Italian Ministry for Education, University and Research published in 2012 the current *Indicazioni Nazionali per il curriculum della scuola dell'infanzia e del primo ciclo di istruzione* (C⁺12), National Guidelines for the first cycle (kindergarden to 8th grade) of education.

This guidelines are not any longer a detailed description of school curriculum to follow, but provide guidelines from which the single schools and institutes, and teachers, can take the basic goals and competences to reach. Some general standards are set with objectives for the educational achievements and learning goals.

It is interesting to underline how this first cycle is given its due relevance in the learning process of each person:

La storia della scuola italiana [...] assegna alla scuola dell'infanzia e del primo ciclo d'istruzione un ruolo preminente in considerazione del rilievo che tale periodo assume nella biografia di ogni alunno. Entro tale ispirazione la scuola attribuisce grande importanza alla relazione educativa e ai metodi didattici capaci di attivare pienamente le energie e le potenzialità di ogni bambino e ragazzo.

Italian school history [...] gives kindergarden and the first education cycle a prominent role considering the importance of this time in every student's life. Within this, the school attributes great relevance to the education and teaching methods that can fully activate energies and potentialities of every kid.

and the importance of active involvement of student is to be considered a key point in the modern school world.

Multidisciplinary features play a leading role in a school environment where the borders between the different subject should fade more and more.

Moreover, other important aspects related to the research topic, underlined in the national guidelines, are connected to *problem solving* abilities:

Favorire l'esplorazione e la scoperta, al fine di promuovere il gusto per la ricerca di nuove conoscenze. In questa prospettiva, la problematizzazione svolge una funzione insostituibile: sollecita gli alunni a individuare problemi, a sollevare domande, a mettere in discussione le conoscenze già elaborate, a trovare appropriate piste d'indagine, a cercare soluzioni originali.

Support exploration and discovery as a mean to promote the curiosity towards new knowledge. In this perspective, the irreplaceable role of problems: students are stimulated to identify problems, pose questions, discuss these with knowledge which has already been developed and to finally look for original solutions.

and the use of *didactical activities as laboratories*:

Realizzare attività didattiche in forma di laboratorio, per favorire l'operatività e allo stesso tempo il dialogo e la riflessione su quello che si fa. Il laboratorio, se ben organizzato, è la modalità di lavoro che meglio incoraggia la ricerca e la progettualità, coinvolge gli alunni nel pensare, realizzare, valutare attività vissute in modo condiviso e partecipato con altri [...]

Didactic activities as laboratories, to promote practicality and dialog at the same time, also making the students reflect on what they are doing. A laboratory, if well administered, is the best way to encourage planning and search abilities, involve the students in thinking, realizing and evaluating processes, in a partecipative and shared with others. [...]

Details about mathematics in grades 1 to 8

The national guidelines gets into the details of mathematical competences. It is reaffirmed, in all grades, the importance of the laboratory activities, interpreted as not only the physical location but above all as an opportunity to have activities in first person and experimentation opportunities.

More specifically, there is a division between primary (in Italy, grade 1 to 5) and middle school (in Italy, grade 6 to 8), with effort on, for primary school:

- reading and understanding texts with logical content;
- build lines of reasoning, having own ideas, defending and comparing them with others;
- a positive attitude towards mathematics, realizing how mathematical tools are useful in the real world.

While for middle school we have a little more specific topics:

- logical aspects, thanks also to correct argumentations and possible changes of mind they can cause;
- importance of the relationship between mathematics and reality;
- a positive attitude towards mathematics, realizing how mathematical tools are useful in the real world.

Direct teaching requests are not expecting, for primary school, objectives that goes over basic arithmetic capacities and some geometry. Lot of freedom is left to the single school / teacher to choose the planning of the teaching program.

Algorithm and logical thinking as also referred to as important in the *technology* chapter of the guidelines, for all school grades.

3.2 Key competences for lifelong learning - European Union

In enhancing this, the Italian school system is connected to the European Union guidelines for learning, (PCotEU06), which explicit necessary competencies for students to reach. Among the eight key points by the EU, we are strongly interested in key point 3, *Mathematical competence and basic competences in science and technology*, from which:

Mathematical competence is the ability to develop and apply mathematical thinking in order to solve a range of problems in everyday situations. Building on a sound mastery of numeracy, the emphasis is on process and activity, as well as knowledge. [...]

An individual should have the skills to apply basic mathematical principles and processes in everyday contexts at home and work, and to follow and assess chains of arguments. An individual should be able to reason mathematically, understand mathematical proof and communicate in mathematical language, and to use appropriate aids.

and key point 4, *Digital competence*, from which:

Digital competence requires a sound understanding and knowledge of the nature, role and opportunities of IST in everyday contexts: in personal and social life as well as at work. This includes main computer applications such as word processing, spreadsheets, databases, information storage and management, and an understanding of the opportunities and potential risks of the Internet and communication via electronic media (e-mail, network tools) for work, leisure, information sharing and collaborative networking, learning and research.

3.3 What has been done in Italy so far?

Most of the work done around cryptography in the Italian school system is related to PLS project (Piano Lauree Scientifiche, project by universities to connect with high schools) or similar scientific divulgation projects and activities, still mainly on the last years of high school; some examples are (Alb) or (CGM07).

Carried out in many places all around the country, all projects have a quite similar structure. Topics and activities range from arithmetic in \mathbb{Z} , prime numbers, multiple and divisibility, Euclidean algorithm, divisibility rules, modular arithmetics with congruences and congruence classes, operations in \mathbb{Z}_n , Fermat's little theorem and so on ... All of this is then applied to build experiences in cryptography such as Caesar's cypher, Vigenère, up to RSA cryptosystem and its functioning as a main example of public key protocol.

University of Trieste developed a project for cryptography in primary school between the 90s and 2002-2003. The proposal was addressed at primary schools also as a mean to approach statistics; much relevance is given to frequency analysis as a tool to decypher substitution cyphers, or the maximum likelihood model. More details can be found in (Zuc92) e (BFSZ02). The idea of *il gioco dell'agente segreto*, *the secret agent game* can be entertaining and engaging for children, while having an impact on knowledges that school brings to the students. Cryptography is well used as a mean of presenting mathematical topics in a way that results fun for students. Some reports on students' reaction are published, but not much detail and research in math education is done among them.

Some more laboratory activities, single lesson proposals, applications to computer science or other directed to all school grades, but they are usually limited to divulgative proposals with no education background.

Giovanni Michele Bianco e Renzo Davoli did provide a translation of the Computer Science Unplugged (BWF98) project, but I didn't find any report on practical activities done in schools or with children after this.

3.4 A preliminary survey: teachers' competences and difficulties

What do teachers think? We had this question in mind when beginning our research. The goal was to have a clear starting point about the situation in our Country and,

at least some, teachers' thoughts about the topics described.

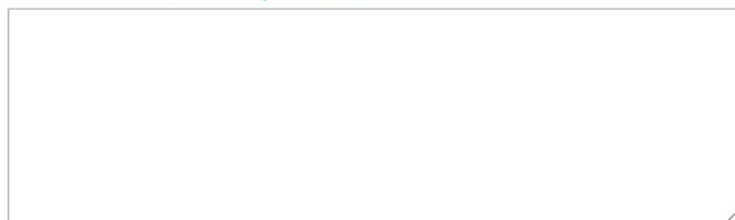
We had a first survey, with results collected from about 150 Italian teachers, mostly in-service and quite evenly divided between primary, middle and secondary school. The survey was done with an online platform; single answers to the questionnaire were given by the involved subjects and automatically registered in an online database.



Indagine statistica sull'insegnamento/apprendimento della Crittografia a Scuola

Crittografia a scuola

Prova ad associare delle parole al termine CRITTOGRAFIA.



Quali tra queste discipline di studio collegheresti alla Crittografia?

- Chimica
- Economia
- Educazione Artistica

Figure 3.1: Some screenshot of the survey proposed - part1.

We analysed the results in a .csv file, collecting them in a table for the multiple-choice answers and dividing the different answers using keywords for the open questions. The resulting analysis is mainly a quantitative approach: the qualitative analysis was the codification of some particular key words used by teachers.

In our preliminary survey, teachers, especially at lower levels, admit that they do not have the necessary knowledge to teach discrete mathematics topics in school.

We look at some details of some of the questions asked.

Prova ad associare delle parole al termine ALGORITMO.

Hai mai studiato Crittografia?

Sì

No

Se Sì:

alla Scuola Primaria

alla Scuola Secondaria di Primo Grado

alla Scuola Secondaria di Secondo Grado

all'Università

da autodidatta

Che cosa hai studiato?

Se No, pensi che ti sarebbe servito per gli studi successivi? Argomenta di seguito il perché.

Figure 3.2: Some screenshot of the survey proposed - part2.

Questions 6-7-8-9-10-11 were about their previous experience in learning cryptography and graph theory, asking about school levels and what they were taught at which level.

The results are shown in the Tables 1 and 2. We have a vast majority of primary and middle school teachers not having had any prior school knowledge about the topics, while secondary school teachers had. This is confirmed by their answers to the question: “write down some words that you think of when hearing the word cryptography/algorithm”. Key words were collected and grouped according to the kind of concepts they referred to. Generally, the primary teachers think about secret

3. Current Situation

codes and hidden things, while teachers who had a mathematics background and answered yes in the knowledge question used words connected with computer science, data and information security. More specifically, “hidden” and “mystery” were more frequent in a “secret codes” thinking, or word as “calculator” and “procedures” are more related to computer science, or, at least, give us some confidence that the compiler has something in mind about computers and concepts of informatics.

Teachers in:	Knowledge about Cryptography	No knowledge
Primary school (1-5)	3 (5.4%)	52 (94.6%)
Middle school (6-8)	5 (13.5%)	32 (86.5%)
Secondary school (9-13)	36 (63.1%)	21 (36.9%)
Total	29.53 %	70.47 %

Figure 3.3: Table 1. Knowledge about Cryptography, in their previous school/universities studies.

Teachers in:	related to Computer Science	Secret Codes/Mystery/Games	Both
Primary school (1-5)	14 (27.4%)	31 (60.8%)	6 (11.8%)
Middle school (6-8)	15 (44.1%)	16 (47.1%)	3 (8.8%)
Secondary school (9-13)	37 (67.2%)	8 (14.6%)	10 (18.2%)
Total	47.1 %	39.3 %	13.6 %

Figure 3.4: Table 2. Keywords to connect to Cryptography/Algorithm.

3.4. A preliminary survey: teachers' competences and difficulties

Prova ad associare delle parole al termine CRITTOGRAFIA.		Prova ad associare dell
Segreto Computer	COMPUTER - INFORMATICA	Sequenza di istruzioni
agente segreto, mistero, nascosto	MESSAGGI SEGRETI - MISTERI - GIOCHI	procedura standard
messaggio nascosto.; logica ; fantasia ; deduzione	MESSAGGI SEGRETI - MISTERI - GIOCHI	formula, soluzione
codice, messaggio, cifra, cifrario, decrittare, decifra	MESSAGGI SEGRETI - MISTERI - GIOCHI	arabo, procedimento, di
scrittura, messaggio	MESSAGGI SEGRETI - MISTERI - GIOCHI	operazioni
Segreto, protezione	MESSAGGI SEGRETI - MISTERI - GIOCHI	Procedura, calcolo
Internet Codici	COMPUTER - INFORMATICA	Sequenza di istruzioni
segreto, enigma, giocare a nascondino, rebus, codice	MESSAGGI SEGRETI - MISTERI - GIOCHI	sequenza, procedura, c
enigma segreto	MESSAGGI SEGRETI - MISTERI - GIOCHI	ComputerInformaticaOri
Codice, segreti, binario, numeri	MESSAGGI SEGRETI - MISTERI - GIOCHI	Procedura, sequenza, c
Ricerca, mistero, scoperta, simbologia, connessioni	MESSAGGI SEGRETI - MISTERI - GIOCHI	Percorso, strategia, sec
Nascondere, offuscare, segreto, proteggere, decifra	MESSAGGI SEGRETI - MISTERI - GIOCHI	Procedimento, calcolare
Codice segretoComputerMessaggio	ENTRAMBI	DatiOrdinamentoProced
informatica computer banca	COMPUTER - INFORMATICA	Informatica blocchi
messaggi segreti, computer	ENTRAMBI	procedura standard
protezione internet banca	COMPUTER - INFORMATICA	ordine
		procedimento
Egitto Codici Segreti	MESSAGGI SEGRETI - MISTERI - GIOCHI	calcoli a blocchi
curiosità, logica, sfida, intelligenza, intuito	MESSAGGI SEGRETI - MISTERI - GIOCHI	sequenza, operazioni, r
algoritmo	COMPUTER - INFORMATICA	ordinamento procedure
Codice segretoDatiNascosto	ENTRAMBI	ProceduraSequenzaDa
DATIPASSWORDCARTA DI CREDITO	COMPUTER - INFORMATICA	PROCEDIMENTOSEQUEN
enigma, cassaforte, computer, messaggio, algoritmo	ENTRAMBI	crittografia, procedimen
codice binariodati	COMPUTER - INFORMATICA	procedurasequenzalog
passwordcodicitrasmmissione dati sicura	COMPUTER - INFORMATICA	sequenzaprocedura
dati segreti cifratura	COMPUTER - INFORMATICA	Procedura calcolo

Figure 3.5: Part of the analysis of the answers received, divided into argument clusters.

Also, teachers were asked if they had any previous experience in teaching the topics or if ‘they would be interested in teaching some algorithm, cryptography and other discrete mathematics topic to students’, and their answers are quite promising, as seen in Table 3:

Teachers in:	related to Computer Science	Secret Codes/Mystery/Games	Both
Primary school (1-5)	14 (27.4%)	31 (60.8%)	6 (11.8%)
Middle school (6-8)	15 (44.1%)	16 (47.1%)	3 (8.8%)
Secondary school (9-13)	37 (67.2%)	8 (14.6%)	10 (18.2%)
Total	47.1 %	39.3 %	13.6 %

Figure 3.6: Table 3. Interest in teaching these topics.

From a qualitative point of view, we had answers which were quite encouraging: ”I’m not an expert in this field, but I really think that some innovative teaching methods could be appreciated in our school; contextualizing mathematical topics to

make them more appealing, and teach things that are both useful and feel real but at the same time make kids learn important mathematics is the way to go” (Math Science teacher, 8th grade).

”I have some basic knowledge from my personal interests, but a serious project which can provide some guidelines for teaching it is missing” (Math teacher, 10th grade). At the end, the data collected were giving us results as summarized above. Some qualitative reading of the meaningful answers is compatible with the quantitative results and confirm that we are on a path that can be appreciated by teachers, educators, schools and researchers.

3.5 This survey and the research project

As we said previously, a literature review on the subject of discrete mathematics, in text books, school curriculum and online resources, suggests that it is not a topic that we often deal with in the classroom. Pedagogical content available on this is, as a consequence, also quite poor. There are some documents about cryptography in secondary and high schools, but the problem we are facing is to bring some meaningful knowledge (or better, process of learning) into grades as low as the primary grades 3 to 5 and to the middle school.

From the questionnaire results and other contacts with teachers and educators in various Italian schools, we had encouragement to continue developing a project on discrete mathematics for various school levels in our country.

3.6 Survey’s conclusion and some more motivation for the project

We noticed that the subject chosen was quite engaging for both students and teachers, as it is seen as an innovative topic with the chances of increasing awareness of mathematics in everyday life and generating interest in all the subjects involved.

In a design research paradigm, we plan on having another cycle of refinement of the activities we have already developed and the development of those we have planned earlier.

We are focusing our didactical research activity on some of the points mentioned above, as computational thinking and problem solving general characteristics, or communication and creativity, applied to problem solving, among students; but also,

3.6. Survey's conclusion and some more motivation for the project

we did notice some implications coming from some tasks done in the classroom, in terms of group work and important consequences in cooperation or selfishness between students. In the last part of this project, we will focus on the collection of more focused results and a video-based analysis of the results, qualitative and fine-grained; in which both group activities and classroom discussions are recorded. We also have many of the transcripts, together with field notes, student's sheets, and interviews as other sources of evidence. As already said, the focus is put on students' learning and thinking, in reaction to the different tasks proposed.

4

Mathematics content: Graph Theory and Algorithms

4.1 Algorithms and Computational Complexity

In computer science and mathematics, many researchs are nowadays in the field of algorithms and computational complexity. Throughout this section we mainly refer to (GJ79), (FH03) and (For09)

Algorithms are mathematical processes treating some variables to define a required outcome. Euclid was the first user of algorithms and even the first one to separate “admissible” from “inadmissible” processes.

Example 4.1. Euclid’s Algorithm to calculate the greatest common divisor of two numbers a and b :

- Input: a and b positive integers.
- Calculations:
 1. if $a < b$, exchange a and b ;
 2. divide a by b and get the remainder r . If $r = 0$, b is the GCD of a and b ;
 3. otherwise, replace a by b and replace b by r . Return to the previous step until the GCD is found.
- Output: The greatest common divisor, GCD, of a and b .

Just to go on with some historical examples, the well-known Sieve of Eratosthenes (see the following figure) for finding prime numbers follows shortly after Euclid’s work.

The term algorithm itselfs comes from Arab and Persian mathematician Al-Khwarizmi

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
76	77	78	79	80	81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100	101	102	103	104	105
106	107	108	109	110	111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130	131	132	133	134	135
136	137	138	139	140	141	142	143	144	145	146	147	148	149	150
151	152	153	154	155	156	157	158	159	160	161	162	163	164	165
166	167	168	169	170	171	172	173	174	175	176	177	178	179	180
181	182	183	184	185	186	187	188	189	190	191	192	193	194	195
196	197	198	199	200	201	202	203	204	205	206	207	208	209	210
211	212	213	214	215	216	217	218	219	220	221	222	223	224	225

Figure 4.1: an illustration representing the Sieve of Eratosthenes. Working on a table with initially all natural numbers, we begin by taking out the even numbers (green), then the multiple of 3 (blue), then 5 (orange), 7 (purple), 11 (yellow) and so on, until obtaining just the prime numbers up to a given integer.

who is also famous for his book “Al-jabr wa’l muqabala”, whose title gives us the word algebra in English.

Going on with history, the use of algorithms grew quickly with the development of the engineering knowledge and machines and therefore got on a higher level in the last decades.

In modern times, since the introduction of the term NP-complete and of the P vs. NP discussion, the notoriety of this theme has been growing. After Turing (Tur36), modern algorithm history starts with the paper of Hartmanis and Stearns, “On the Computational Complexity of Algorithm” (HS65).

Our goal is to discuss and give a formal meaning to “difficulty”, and sometimes “intractability” of problems. Before talking about these mathematical facts (and especially present them to kids), we need to formally agree on and introduce some basics notions about them.

The famous $P = NP$ problem which is one of the key features in this subject, is one of the seven Millennium Prize Problems and solving it brings a 1,000,000 dollars prize from the Clay Mathematics Institute.

Definition 4.1. A *problem* is just a general question to be answered; each problem has some free variables, called *parameters*.

A *decision problem* is a problem for which the answer can only be *yes* or *no*.

To describe a problem, we will in general require a description of all the parameters and what properties the *solution* is required to satisfy.

An *instance* of a problem is obtained by specifying particular values for all its parameters.

Definition 4.2. An *algorithm* is a step-by-step procedure for solving problems.

An algorithm is said to *solve* a problem if it produces a solution when applied the given problem. Note therefore that the algorithm is required to always produce a solution for every possible instance of the problem we have.

Our goal is, in general, finding the most efficient algorithm to solve a particular problem. Most often “most efficient” is meaning the algorithm that gives us easily the solution, mainly referring to time requirements to produce a solution. There are some other computing resources needed for executing an algorithm, for example the number of computations required, but we will mainly deal time which is indeed the most relevant factor. Time requirements of an algorithm are usually referred to as the *size* of a problem instance, that is closely related to the amount of input data needed to describe the particular instance.

Definition 4.3. The *time complexity function* of an algorithm expresses the algorithm time requirements by giving the longest time needed to solve a problem instance, related to its size.

Example 4.2. An algorithm of time complexity $O(n)$ is linearly dependent from the size of the problem and its size n . An algorithm of complexity $O(n^2)$ has a quadratic dependence from the input and so on, if the complexity is $O(n^k)$ it will take a polynomial time. An algorithm of complexity $O(a^n)$, for some $a > 1$ integer, takes exponential time.

Distinguishing between “efficient” and “not efficient” algorithms is depending on the single instances and what we need the algorithm for; computer scientists however recognise a first difference between the two informal definitions.

To formalize this, recall from asymptotic analysis that a function $f(n)$ is said to be $O(g(n))$ if there exists a constant $k \in \mathbb{R}$ s.t. $|f(n)| \leq k \cdot |g(n)|$ for all values $n \geq 0$.

Definition 4.4. An algorithm is called a *polynomial time algorithm* if its time complexity function is $O(p(n))$ for some polynomial function p and input size n .

Definition 4.5. An algorithm whose time complexity function is not $O(p(n))$, i.e. cannot be bounded with a polynomial function, is called *exponential time algorithm*.

Especially when considering problems with large amount of datas, the difference between the two types of algorithm is clearly visible.

To go more in depth about this we can further divide problems (and algorithm to solve them) in complexity classes.

Definition 4.6. A *complexity class* is the set of all problem with related complexity.

We will give some examples and in depth explanation of these in the next subsection.

4.1.1 P vs. NP

The most well-known problem in computational complexity is the P vs. NP problem. To understand this, let's first define:

Definition 4.7. A *Turing machine* is an ideal model of a computing machine formed by a tape and a reading/writing instrument that can perform operations on this tape. It can formally be seen as a 5-tuple formed by

- the present state of the machine s ,
- the symbol i read at the present state,
- $S(s, i)$, state of the machine at the next step,
- $I(s, i)$, the symbol written at the next step,
- $V(s, i)$ direction in which the machine is moving.

A *deterministic Turing machine* (DTM) is a Turing machine which has a unique operation that can be performed as a consequence of the present state, while a *non-deterministic Turing machine* (NDTM) has the possibility of doing more than one thing at once, given the current step. The non-deterministic Turing machine has, in addition, a guessing module having its own write-only head, which provide the mean to writing down the guess characterizing the non-deterministic machine. The computation of an NDTM is therefore divided in 2 stages, the first one being a “guessing” stage and a “checking” stage after that.

Any non-deterministic Turing machine can be seen as a deterministic Turing machine by simply not splitting at any step.

Definition 4.8. The set P is the set of all decision problems which have an algorithm that can be successfully used in polynomial time by a deterministic Turing Machine.

Definition 4.9. The collection of problems that have efficiently verifiable solutions is known as NP . A NP problem is, in other words, a decision problem which has some instances for which the answer is positive and it is possible to verify this in polynomial time; i.e. we can check the correctness of this answer in polynomial time. Note that $P \subseteq NP$.

Example 4.3. Integer factorization decision problem is NP ; the problem we are referring to is: given an integer n , to find out if there is another integer, smaller than n , which divides n . It may be hard to compute such a number if asked so, but we can easily check the correctness of an answer if given one.

The question that still remains open is whether it is harder to find a solution to a problem than just checking a given solution for the same problem instance. In addition, it has never been proven that a non-deterministic Turing machine is more powerful than a deterministic one, even if it looks so. To prove that $P \neq NP$ we would need to prove that there exists a set of problems or at least a problem X s.t. X is in $NP \setminus P$, i.e. there is an algorithm to solve X in polynomial time by a non-deterministic Turing machine and X does not belong to P , so there is no algorithm being able to solve the problem in polynomial time by a deterministic Turing machine.

Example 4.4. In all this instances we are analyzing the decision problem relative to the actual problem we give as example.

Suppose we have a class of students and we want to put them in pairs. We know the students and compatibilities to work together among them; we can therefore try all possible combinations and find a solution which respects our requirements. In 1965, Jack Edmonds produced an efficient algorithm to solve the problem which now regarded as a P problem.

Anyway, we could have different request for slightly different problems which will require something more than polynomial time to be solved. Just think about making group of three people and the problem doesn't have a solution, yet, in polynomial time. We could want to divide the students in bigger groups (dividing all the class in three groups for example is connected to Map coloring which we will see in Chapter ??). We could even want to have all the class sit around a round table with no students sitting close to someone they don't like (Hamiltonian Cycle). Is it possible? The answer to these last problems is not yet known to be possible in polynomial time, but it is possible to verify the correctness of a solution for them once we are

given one.

$P = NP$ would mean that we could actually solve this advanced problems in polynomial time and making it become a relatively easy task. This is largely believed to be not true, but again, it hasn't been proven yet.

Definition 4.10. A decision problem X is said to be in the *NP-complete* complexity class if it is in the NP class and it is possible to reduce any other NP problem to X in polynomial time; i.e. if we can efficiently solve our problem X then we could also solve any other problem we reduce to it. Any NP-complete problem can be reduced to other NP-complete problems in polynomial time.

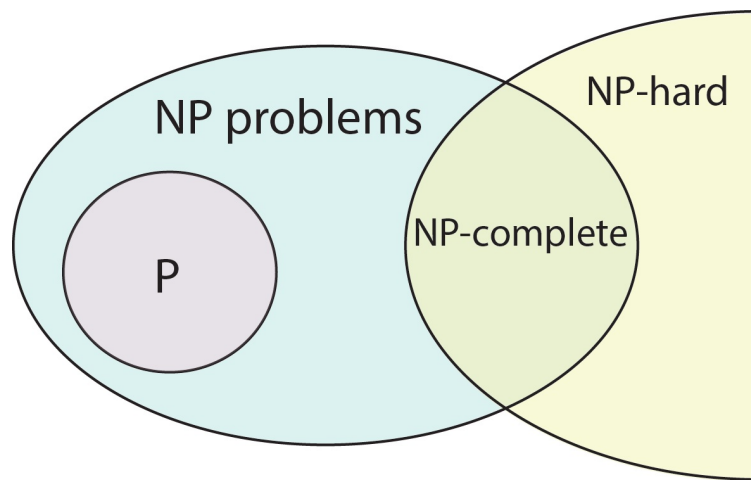


Figure 4.2: a diagram of the different complexity classes we are discussing.

Definition 4.11. A problem X is said to be *NP-hard* if there is an NP-complete problem which is reducible to X in polynomial time.

Any NP-complete problem can be reduced to NP-hard problems in polynomial time. We can conclude that if there is a solution to a NP-hard problem in polynomial time, then there will be solution to all NP problems still in polynomial time. To better understand this, note that an NP-hard problem does not need to be a decision problem. We therefore obtain a representation of this situation in a graph as in Fig. 4.2 .

Example 4.5. Many other actual problems (or their relative decision problem) are in the NP class:

- integer factorization, as seen above;
- finding a DNA sequence that best fits a collection of fragments of the sequence;

- finding Nash Equilibriums with specific properties in a number of environments;
- the graph isomorphism problem (i.e. can two graphs be drawn in the same way?);
- recall that any P problem is also NP.

Example 4.6. The Traveling Salesman.

This problem is the classic example of a NP-complete problem. Basically the salesman has to visit a number of cities in his sales tour and he knows the distances between each pair of cities. Which is the best route to take to visit all the cities travelling the shortest possible path?

The problem itself is actually in the class of NP-hard problem, but if we consider the decision version of it, i.e., given a possible solution path, determining whether the graph has any path shorter than the one given, is a NP-complete problem.

Translating it in mathematical language, we have a set of points in a space with a given distance and we know the value of the distance for each pair of points. The task is determining the shortest path connecting every point (exactly once in most versions of the problem, we are assuming this) and returning to the starting point. A first solution that seems easy is listing all the possibilities and choose the shortest. However, this is easy only if the number of points is very low. In general, for n points to connect, we will have $n!$ possibilities, making it a huge number without need to take n really big. So our first approach is actually running in *factorial time*. This means that for example, if our calculator can compute a solution for $n = 20$ in a short time like 1 second, it will take some million years just to compute the same problem with $n = 30$.

One of the best-known results so far in attacking this problem is the Held–Karp algorithm which solves, using dynamic programming, the problem in $O(n^2 2^n)$ time, and we see that the problem is not in P.

On the other hand, if we somehow already have a solution for the problem, it is possible to verify it in polynomial time.

4.2 Coloring maps and graphs

The problem of coloring maps begins from a basic and practical geographic problem: mapmakers want to draw maps which are easy to read; i.e., in this case, in such a way that when two regions border each other, they shouldn't be of the same color. Historically, it was Francis Guthrie, an english botanist and mathematician, who first pose this problem in 1852. The problem was posed to his brother Fredrick and then to professor De Morgan and quickly spread to the mathematics world.



Figure 4.3: an example of this map coloring, in a political map of Europe.

Our goal will be to understand from a mathematic viewpoint what this problem means and how many colors are required to color different maps.

Let us first define:

Definition 4.12. We call a *map* a division of a surface we are considering into distinct non-overlapping *regions*.

A *planar map* is a division of a surface which lies in the plan.

In this thesis, if not differently specified, we will refer to planar maps only.

Definition 4.13. Two regions of a map are *adjacent* when they border each other, i.e. when they share a border, other than a corner.

Definition 4.14. A map is *n-colorable* if it can be colored with n different colors, so that no pairs of adjacent regions have the same color.

The map coloring problem is difficult to handle as it has been presented right now. We can anyway translate it into a problem of graph theory.

We can associate every map to a graph just considering the vertices of a graph as the regions of the map. If two regions border each other, then they will have an edge connecting the vertices in the corresponding graph.

But, before going further, let's see more formally what we are talking about.

4.2.1 Introduction to graph theory

References for this general section will be (BM76), (CLZ10) and (W⁺01) as well as (Ros11) when applying graph theory to maps.

Graph theory was invented by Euler in the 18th century; history says that it began with the famous problem of the town of Koenigsberg, now Kaliningrad, Russia.



Figure 4.4: the bridges of Koenigsberg.

The problem was the following: is there a way to walk through the city, crossing each bridge once and only once?

To find an answer to this problem, Euler tried to simplify it, eliminating from the picture everything except for the pieces of land (indicated by dots) and the bridges (drawn as the lines connecting the dots). From this modeling, you can see that, if you reach a dot with one line, you need to leave by another, since you can't cross one bridge twice. The problem here is that each piece of land was connected with the other by an odd number of bridges, making the task impossible to do. The answer to the problem of finding such a way through the city was then no.

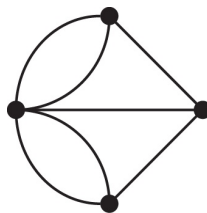


Figure 4.5: Euler's model of the town of Koenigsberg.

This was the first example of graph theory and led, in the following centuries, to many applications and development. We begin to formalize this mathematical area:

Definition 4.15. A *graph* is a pair $G = (V, E)$ where V is the finite set of the *vertices* or *nodes* of G and E is the finite set of the *edges* connecting the vertices; each edge is a pair of distinct elements in V , i.e. $(i, j) \in E$ s.t. $i, j \in V$.

Definition 4.16. A *simple graph (undirected)* is a graph which is undirected (edges have no direction), i.e. there is no difference between the pairs (i, j) and (j, i) in E , admits no loops (edges connecting a vertex with itself) and no pair is repeated, i.e. each pair (i, j) in E is such that $i < j$.

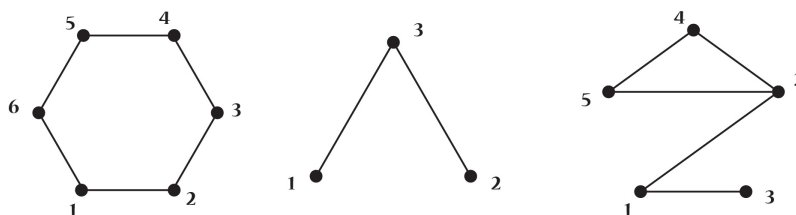


Figure 4.6: some examples of simple planar graphs.

Note: in the first graph definition, if we consider the edges of a graph as an ordered pair, we will need to define an order on vertices of V . In this case the graph is said to be directed, which we are not interested in for our purposes. We will consider mainly simple undirected graphs, where the edges don't have directions but simply connect two different vertices.

Definition 4.17. A *subgraph* of a graph $G = (V, E)$ is a graph $H = (V_H, E_H)$ such that $V_H \subseteq V$ and $E_H \subseteq E$.

Example 4.7. Back to the initial map coloring problem, consider:

- V as the set of regions on the map
- $E = \{(v, w) \text{ where } v \text{ and } w \text{ share a border}\}$

In this way, for every planar map we can obtain a simple graph $G = (V, E)$.

Definition 4.18. The *degree* of a vertex is the number of edges attached to it.

Definition 4.19. The *order* of a graph G is the number of vertices of G .

It is easy to see that, given a graph with n vertices and m edges,

$$\sum_{i=1}^n \deg(v_i) = 2m. \tag{4.1}$$

By equation 4.1, we obtain that every graph must have an even number of vertices with odd degree.

Definition 4.20. A *face* of a graph G is a region bounded by edges of G . We consider to be a face also the outer unbounded region.

Definition 4.21. A *cycle* in a graph is a sequence of vertices and edges which is closed, i.e. starting and ending in the same vertex.

Lemma 4.2.1 (Euler's Formula). *Given a graph $G = (V, E)$; call V the number of vertices, E the number of edges and F the number of faces in G . Then $V - E + F = 2$.*

We want now to give some examples of common graphs:

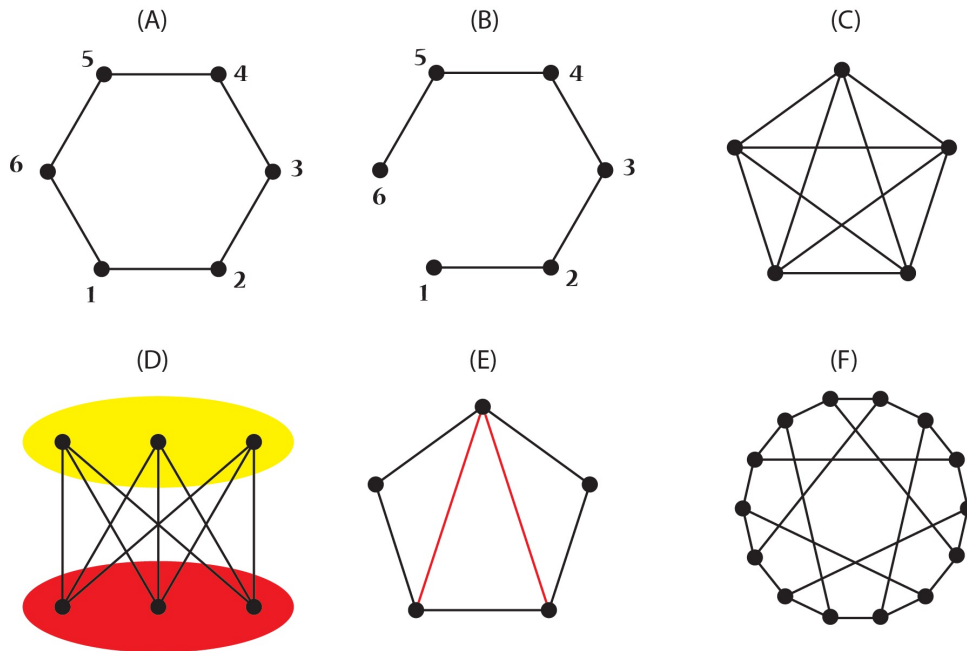


Figure 4.7: examples of some different kind of graphs.

- Cycle Graphs C_n , i.e. the simple graphs with $V = \{v_1, v_2, \dots, v_n\}$ and $E = \{(v_1, v_2), (v_2, v_3), \dots, (v_{n-1}, v_n), (v_n, v_1)\}$; e.g. the triangle, the square, ... , n -agons. See figure 4.7, A) ;
- Path Graphs P_n , i.e. the simple graphs with $V = \{v_1, v_2, \dots, v_n\}$ and $E = \{(v_1, v_2), (v_2, v_3), \dots, (v_{n-1}, v_n)\}$; where one edge is missing respect to the previous example. See figure 4.7, B) ;
- Complete Graphs K_n , i.e. the simple graphs with $V = \{v_1, v_2, \dots, v_n\}$ and $E = \{(v_i, v_j), i \neq j\}$; in other words the graphs with all possible edges. See figure 4.7, C) ;
- Complete Bipartite Graphs $K_{n,m}$ on $n + m$ vertices, i.e. the simple graphs with $V = \{v_1, \dots, v_n, w_1, \dots, w_m\}$ and $E = \{(v_i, w_j) : 1 \leq i \leq n, 1 \leq j \leq m\}$, with all the edges between one part $\{v_1, \dots, v_n\}$ and the other $\{w_1, \dots, w_m\}$. See figure 4.7, D) ;

- we will see later Chordal Graphs, where each cycle made of a number of vertices $n \geq 4$ has a chord, i.e. another edge not in the cycle connecting two not-consecutive vertices in the cycle. See figure 4.7, E) ;
- graphs are also classified by the degree of their vertices, e.g. a Cubic Graph is a graph in which each vertex has degree 3. For an example see figure 4.7, F) . We call Regular Graphs the graphs in which every vertex has the same degree; if every pair of adjacent vertices has the same number of other adjacent vertices in common, the graph is said to be Strongly Regular. An example of this is easy to find in the complete graph; in the complete graph with n vertices, each one has degree $n - 1$.

Definition 4.22. We say that two graphs G_1, G_2 are *isomorphic* if and only if there exist a bijective map $\sigma : V(G_1) \rightarrow V(G_2)$ (in this way we are relabelling the vertices) and (v_i, v_j) is an edge of G_1 if and only if $(\sigma(v_i), \sigma(v_j))$ is an edge in G_2 .

Two isomorphic graphs can be treated as the same for our purposes.

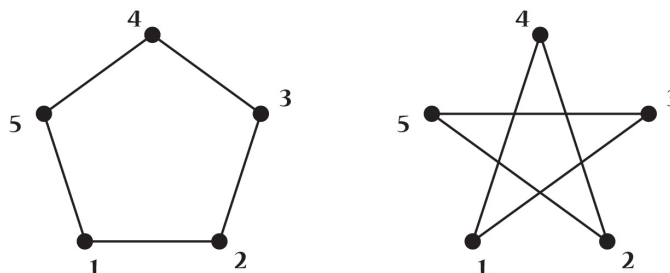


Figure 4.8: the pentagon and the star are not the same graph, however they are isomorphic, i.e. representing the same situation just with a relabelling of the edges.

Definition 4.23. A *vertex coloring* of a graph $G = (V, E)$ is a function $c : V \rightarrow S$ where S is the finite set of the colors and such that $c(v) \neq c(w)$ whenever v and w have an edge connecting each other.

The *chromatic number* $\chi(G)$ of a graph G is the smallest k such that $S = \{1, \dots, k\}$ and there exists a vertex coloring $c : V \rightarrow S$.

The graph G such that $\chi(G) = k$ is called *k-chromatic* and, whenever $\chi(G) \leq k$, G is called *k-colorable*.

Let's extend the definition of a complete bipartite graph in the following way:

Definition 4.24. A *bipartite graph* G is a graph in which we can break the set of the vertices V into two parts, such that every edge connects one vertex of the first part with one vertex of the second part.

Note that, from this definition, it has not necessarily to be complete.

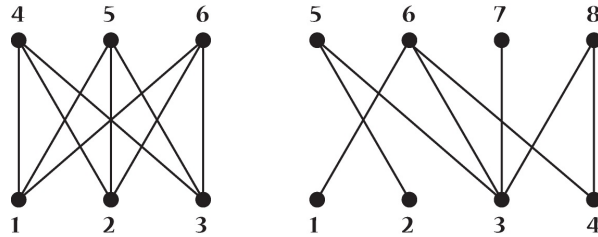


Figure 4.9: a complete bipartite graph on the left and simple bipartite one on the right.

Example 4.8. It is easy to find a graph that is NOT bipartite; for instance the cyclic graph C_3 is not. If we want to divide its vertices into two sets, then at least two vertices has to be in one of the sets of the partition; from this, there must be an edge which connect two vertices of the same set, so the partition does not make C_3 bipartite. In general, the cycle graph C_{2k+1} is not bipartite for each k .

Beyond this, graph/map coloring has many practical and theoretical applications. For example it has recently been used for new methods in Air Traffic Management conflict detection and resolution, in radio frequencies assignment and in time scheduling.

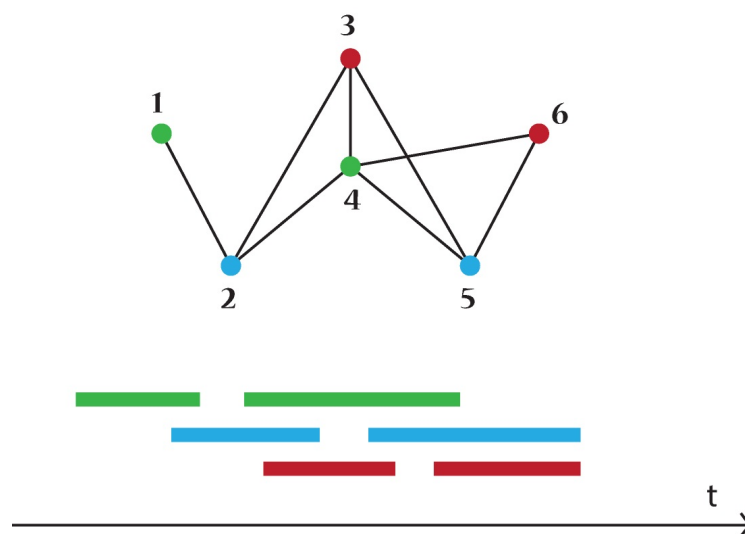


Figure 4.10: an example of the flights schedule planning: the colored lines are the schedules of flights and the the graph is drawn so that when two flights are conflicting with each other the vertices are connected.

An example of application of graph/map coloring in the last situation is the following: let's consider aircraft scheduling and we have n flight to assign to a certain number of aircrafts. Simplifying the problem, we need at least to know the interval of the flight, their time schedules. We consider a graph in which each vertex is a flight and we connect vertices when the schedules of two flights intersect. The goal will be to find a good coloring for the graph we built in that way. If we can find a solution with a number of colors less or equal to the number of aircrafts we have available then we can apply it to reality.

Such a graph is called an *interval graph*.

An interval graph is indeed a chordal graph and we can actually solve a graph coloring problem on this kind of graphs in polynomial time.

These and many other are the applications of map and graph coloring in modern mathematics and computer science; many references are easy to find about this subjects, see for example (Lei79), (Dán04) and (Hav07).

4.2.2 The two-colors problem

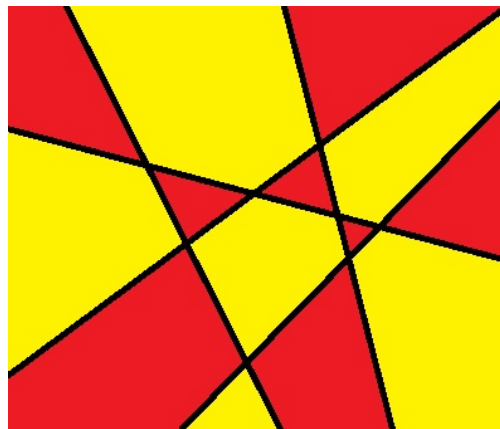
Reference for this section, as well as the next ones on map coloring, will be (BH83), together with other articles and book that will be cited through the text.

A first easy problem we can look at when considering map coloring is the following: is it possible to color a map with just two colors?

What we notice, by just trying with some different map examples, is that it is quite easy to realize whether two colors are enough or not. The algorithm that comes quite intuitive is to color one area of the map and then surround it with a second color; this makes immediately realize if it is possible to do the job with just two different colors. What we can see is that two-colorable maps are the ones drawn and obtained as intersections of closed curves.

A first result we need to show is the following:

Theorem 4.2.2. *Given a finite collection of straight lines in the plane, the map those line form in the plane is two-colorable.*



Proof We use induction on the number of lines n . If $n = 1$ it is easy to see that respectively 1 and 2 colors are enough. Now take $n > 1$ and suppose that every map obtained as a collection of up to $n - 1$ lines is two-colorable. Then we can consider a collection of n lines; choose a line l out of these and remove it from the plane. The resulting map is two-colorable by the inductive hypothesis; so it is possible to color it with two colors. Now we add back again the line l and reverse the colors of the different areas on one side of l . Each side of l will remain correctly colored and we can see that regions which border each other on l will be colored correctly as well as they were previously of the same color, and become now of different color. Finally, such a map is always colorable with two colors. \square

For the next step we need to introduce a new definition:

Definition 4.25. A graph G is said to be *connected* if for every vertices $v, w \in V$ there exists a *path*, i.e. a collection of edges, that connects them.

The *dual graph* of a graph G is a graph G' with a vertex for every face of G and an edge joining the vertices when the relative faces are bordering each other, for each edge in G . If a graph G has no cycles, then its dual graph will be made of only one vertex with loop vertices.

The dual of the dual G' is isomorphic to the graph G if G is a connected planar graph.

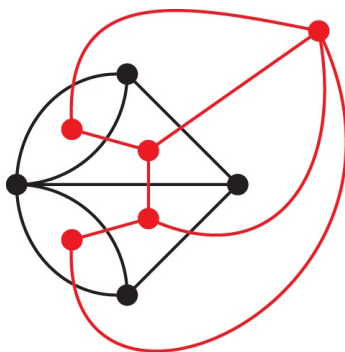


Figure 4.11: the red graph is the dual of the black one and viceversa.

Lemma 4.2.3. *If a graph G has every face with an even number of edges, then G is two-colorable.*

Proof We prove this by induction on the number of faces of the graph G . If $n = 0$, the graph has no faces (and no cycles) and we can just alternate the two colors to find a solution. Now take $n > 0$ and suppose that every graph with $n - 1$ faces, all with an even number of edges, is two-colorable. Consider then a graph G with n faces, again with an even number of edges surrounding each of them. We take an edge away from this (for example take the edge connecting V_1 and V_2 from the face border $V_1V_2 \dots V_{2k}$). What we obtain is a graph G^* with $n - 1$ faces and each of these faces have an even number of edges; also the new face we created is good since the number of its faces is the sum of the edges of the previous two faces we unified minus the edge we took away which count twice. By inductive hypothesis G^* is two colorable. Now notice that V_1 and V_2 have different colors; otherwise we would not be able to two-color for example the path $V_2V_3 \dots V_{2k}V_1$ which is in G^* as well. So we can just add back the edge connecting V_1 and V_2 and it will still be a good two-coloring for G . \square

Theorem 4.2.4. *Take a closed curve in the plane (topologically speaking, i.e. an immersion of S^1). Self-intersections are admitted but they have to be only points (retracing of part of a line that has already been drawn is not admitted). The map that this curve forms in the plane is two-colorable.*

Proof We can consider our map as if it is a graph G (i.e. a vertex for each self-intersection and edges following the curve; if the curve is not self-intersecting we will need to set one point of the curve to be a vertex) and find also the dual graph G' . Coloring the dual graph means automatically coloring the map corresponding to the graph. All the faces of G' , except for the single loops, have an even number of edges. In fact, each face of G' contains one crossing and the number of edges of that face are the different way to arrive at the self-intersection, which is an even number since the curve is closed and the intersections are only points.

By Lemma 4.2.3 G' is two-colorable and so is the map corresponding to G . If the map and the corresponding graph have a loop, we just find a coloring for the map without this loop and then add it back coloring it with the opposite color of the region surrounding it. \square

Finally, notice that we can draw such a map by drawing any curve on the paper without lifting the pencil we are using (as in Figure 4.12).

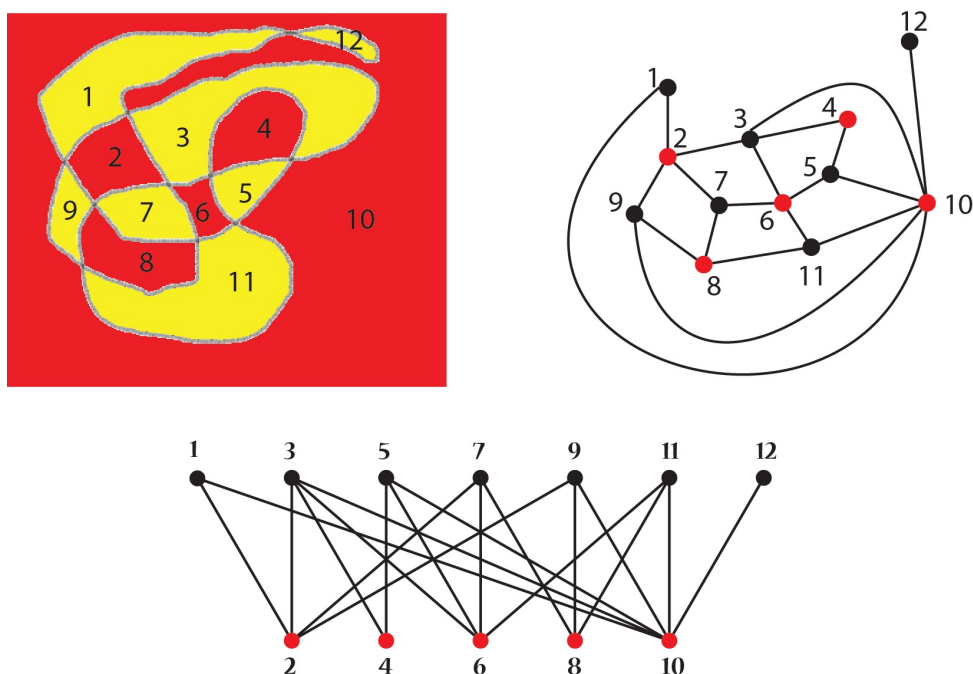


Figure 4.12: an example of how to start from a map, get the correspondent graph and then also the bipartite graph.

What we have seen above, introduces a nice connection between maps and graphs. If we go back to the definition (4.24) of bipartite graphs, we could break the set of the vertices V into two parts; this two sets are said to be *independent* as they contain no vertices connected with each other. Once we see these two sets, we can color one set in one color and the second in the other seeing immediately that *a graph is two-colorable if and only if it is bipartite*. It is again immediate to connect such a graph with the maps we built before just with a closed curve self-intersecting.

We can study algorithms to color this kind of maps.

If a map is two-colorable, the easiest way to color it is to start with one area, then color its surrounding areas of the other color and continue like this. This is a straight forward to apply algorithm as we have no choice on how to color the next area we get to.

Another possibility we want to introduce is the greedy coloring algorithm. To explain it better, give each color a number $(1, 2, 3, \dots)$, and start by giving an area color one. We set an order in which we will be analyzing the areas and proceed by giving the second area the lowest number that can be given (i.e. the lowest that is not assigned to a neighbouring area) and then just continue further on and do the same with the next number. We are sure that we will get a solution; what we are not sure about is if we are going to get the best solution (this is why the algorithm has

this name..we are not putting much effort in trying to lower the number of colors used). In the picture under we can see how a greedy algorithm can work or not with finding a two-coloring depending on the order we consider the areas (or equivalently, the vertices of the corresponding graph).

With two colors, we have a simple algorithm that works (described above) but when we get to more complicated maps, we will need to implement new method to color them.

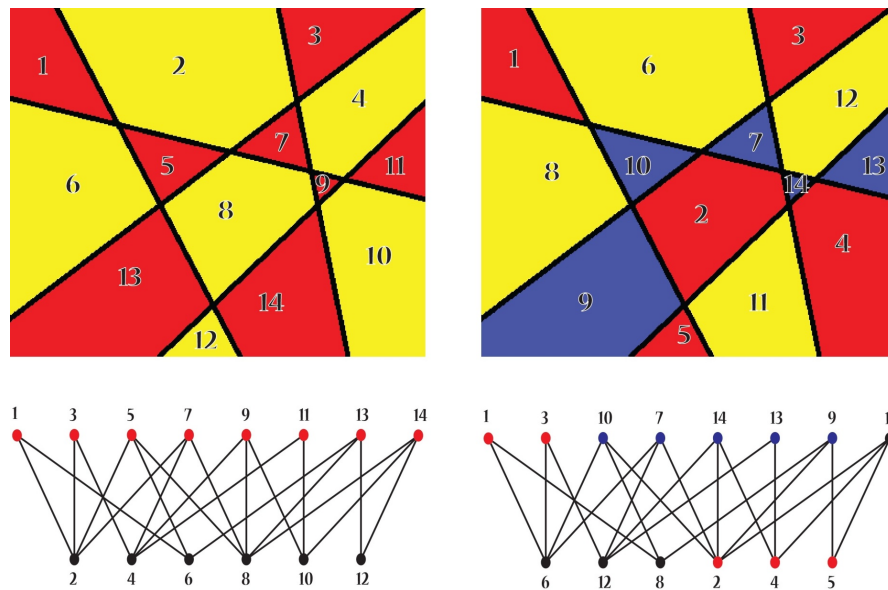


Figure 4.13: on the same map, we get two different colorings, depending on how we choose the order of the areas when applying the greedy algorithm. This algorithm is easy and works to find a solution but not the optimal one when trying to minimize the number of colors.

4.2.3 The three-colors problem

Main reference for this section is (Hen06). After the relatively easy two colors problem, we want to look at more complex problem in map coloring: is a map colorable with only three colors?

Here it is possible to find examples and counterexamples, too.

Example 4.9. A real example of a three-colorable map is the country of Switzerland and its border (with the exemption of Liechtenstein just for one moment). Switzerland has four countries at its border, all additionally bordering each other in pairs. This makes it possible to color these five countries with only three different colors.



Figure 4.14: a possible way of coloring this example and its relative graph.

Example 4.10. We don't need to move very far to find an example where it is not possible to do the same. Look at the country of Luxembourg, bordering France, Germany and Belgium. With only four countries, it is not possible to use three color but we need to have for different ones, since the countries all border each other.

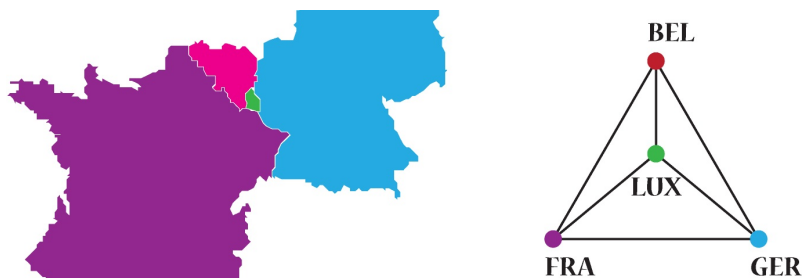


Figure 4.15: a possible way of coloring this example and its relative graph.

Try fixing, for example, Luxembourg as green; then Belgium will need a different color (e.g. pink), since they are adjacent. Then France could not be neither green nor pink, being it adjacent to both the previous countries; so let's say we color it purple. The last country we are considering, Germany, is adjacent to all the previous

ones, so it cannot be of one of the three other colors and we are forced to choose a fourth one.

More generally, a map requires at least four colors if its relative graph contains a piece in which four vertices are all connected with each other.

Is there a way to generalize this even more?

How can we know if three colors are enough to color a certain map (or graph)?

4.2.4 A Gröbner basis algorithm for the three-coloring problem

A way to decide if three colors are sufficient to color a map, is to analyse a polynomial system associated to the map. We represent each color with a complex cubic root of the unit ($1, \xi, \xi^2 \in \mathbb{C}$) and each region of the map with the variable x_i , with possible values for x_i in the set of the three colors, obtaining $x_i^3 - 1 = 0$ for each i .

For two regions x_i, x_j we obtain

$$x_i^3 - x_j^3 = (x_i - x_j)(x_i^2 + x_i x_j + x_j^2) = 0$$

If we analyze regions that border each other, then the values of their x 's has to be different, so $x_i \neq x_j$ and we obtain $(x_i^2 + x_i x_j + x_j^2) = 0$. We can therefore state that a map with n region is three-colorable if and only if the system of equations

$$\begin{cases} x_i^3 - 1 = 0 \\ x_j^2 + x_j x_k + x_k^2 = 0 \end{cases}$$

has at least one solution, where $i = 1, \dots, n$ and x_j, x_k represent adjacent regions.

In the case of polynomial systems, we can use the following solving method (which is analogous to the linear method using Gaussian elimination):

denote by $P(n)$ the set of polynomials with variables x_1, \dots, x_n and consider the ideal I generated by the polynomials f_1, \dots, f_r ;

$$I = \langle f_1, \dots, f_r \rangle = \{h_1 f_1 + \dots + h_r f_r \mid h_1, \dots, h_r \in P(n)\}$$

From Hilbert's Nullstellensatz we know that the system $f_1 = \dots = f_r = 0$ has a complex solution if and only if $1 \notin \langle f_1, \dots, f_r \rangle$; and here is the point where we need Gröbner bases. We will use Buchberger's algorithm to compute a Gröbner bases $G = \{g_1, \dots, g_s\}$ which will make it easier to solve our problem.

We will make clear how we use the Gröbner bases knowledge to solve this problem in a practical case.

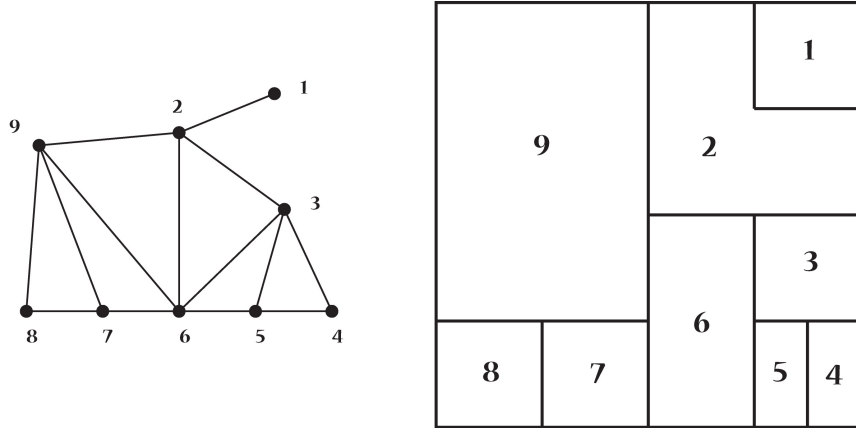


Figure 4.16: the graph we are considering in the example and its relative map.

Example 4.11. We consider the following graph (figure 4.16) and we want to find out if it is possible to color it with just three colors. So, let's go back to the system

$$\begin{cases} x_i^3 - 1 = 0 \\ x_j^2 + x_j x_k + x_k^2 = 0 \end{cases}$$

where we make i variable from 1 to 9 and $(j, k) \in \{(1, 2), (2, 3), (2, 6), (2, 9), (3, 4), (3, 5), (3, 6), (4, 5), (5, 6), (6, 7), (6, 9), (7, 8), (7, 9), (8, 9)\}$.

The system has for sure a finite number of solutions having each of the variables to take one of three possible values (the three different complex cubical roots of the unit). Our problem reduces to find if the system admit a solution at all, in which case the 3-colorability would be proved.

We use the lexicographical *lex* order and obtain the Gröbner basis $G = \{g_1, \dots, g_9\}$ with $g_1 = x_1^3 - 1$, $g_2 = x_2^2 + x_2 x_1 + x_1^2$, $g_3 = x_3^2 + x_3 x_3 - x_2 x_1 - x_1^2$, $g_4 = x_4 + x_3 + x_2$, $g_5 = x_5 - x_2$, $g_6 = x_6 + x_3 + x_2$, $g_7 = x_7 - x_2$, $g_8 = x_8 + x_3 + x_2$, $g_9 = x_9 - x_3$

As a consequence of the theorem previously stated, we can see that, as $1 \notin I$, the system admits at least one solution.

Solving $g_2 = g_5 = g_7 = g_9 = 0$ we obtain $x_1 \neq x_2, x_2 = x_5 = x_7, x_3 = x_9$ and from $g_1 = 0$ we see that we can give any color to x_1 ; $g_4 = g_6 = g_8 = 0$ leads us to have different colors for x_2, x_3, x_4 with $x_4 = x_6 = x_8$. The last equation $g_3 = 0$ can be written as $0 = g_3 = x_3^2 + x_3 x_3 - x_2 x_1 - x_1^2 = (x_3 - x_1)(x_3 + x_2 + x_1)$, so either $x_1 = x_3$ or x_1, x_2, x_3 need to have different values.

We end with finding two different solutions which are shown in figure 4.17.

We can deal with every situation in the same way as in the example; of course it gets harder as the map gets complicated, but still fall into a relatively simple linear system problem.

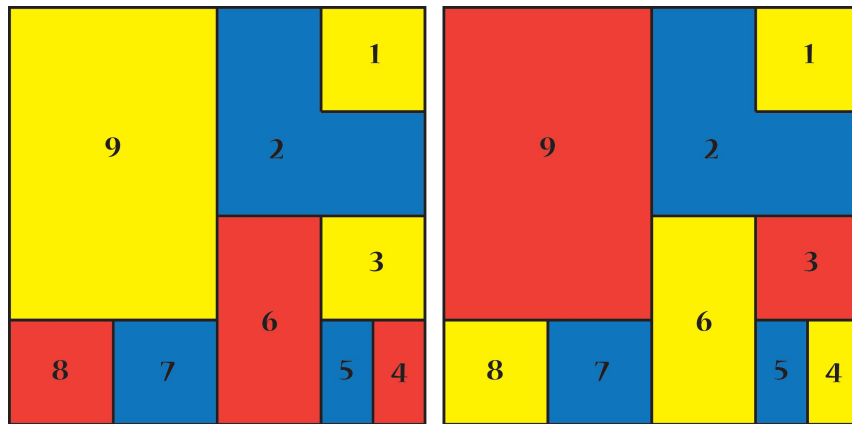


Figure 4.17: up to permutations of the colors, the two different solutions of the example.

4.2.5 Complexity of this problems

Let's begin with the complexity of the two-colors problem; we can see that this problem is solvable in polynomial time. Actually, as we saw, it is the same of determining whether a graph is bipartite or not and so it is computable in linear time.

The three-colors problem (determining if a map is three-colorable or not), on the other hand is a NP problem and so considered to be computationally hard as there are no known optimal polynomial-time solution. We can though, following the definition of NP, verify a particular solution quite easily (i.e. we just need to check if the coloring is working or not, respecting the minimum number of colors and the bordering condition).

The problem is in fact NP-complete, as is shown in (Dur02).

Considering a graph with degree ≤ 4 , finding if it is three-colorable, is even NP-hard. As we saw, we can easily check solutions but, as a consequence as the P vs. NP problem, it is still impossible to determine whether or not we can solve this problem in a quick way.

4.2.6 The four-colors problem

Allowing a fourth different color to solve the not-colorability with only three, we will get to an exceptional result: the 4-color theorem.

Theorem 4.2.5. *Every planar map is four-colorable.*

Before seeing some history of this, we will prove a weaker results, using five colors. First of all,

Lemma 4.2.6. *Every planar graph G contains a vertex of degree 5 or less.*

Proof Let $G = (V, E)$ be a planar graph. Suppose by contradiction that every vertex has 6 or more edges connecting it to other vertices and recall Euler's formula $V - E + F = 2$ where F is the number of faces of the graph. Moreover, each face is bounded by at least three edges and each edge is in touch with at most 2 faces, so $F \leq 2E/3$. $V - E + F = 2$ becomes $V - E + \frac{2}{3}E \geq 2$, then $3V - 3E + 2E \geq 6$ which is $E \leq 3V - 6$. If we double the number of edges we obtain $2E \leq 6V - 12$. If every vertex has at least 6 edges touching it, then $2E \geq 6V$ which leads to a contradiction. \square

This lemma leads to the following result:

Theorem 4.2.7. *Every planar map is five-colorable.*

Proof Let n be the order of the planar graph G corresponding to the map. We will prove the result by induction on n , noticing that it is clearly true for $n \leq 5$. So we will assume that every planar graph of order $n - 1$ is five-colorable, where $n \leq 6$. By Lemma 3.4.2, G contains a vertex v with $\deg(v) \leq 5$. Consider now the planar graph $G - \{v\}$, which has order $n - 1$ and, by hypothesis, is five-colorable. If we have a coloring of $G - \{v\}$ not using all five colors on the vertices connected (adjacent) to v then we can use one of the missing colors to color v and we are done. Let's assume that there are five vertices connected to v and they are colored in the five different colors. Call v_1, \dots, v_5 the vertices adjacent to v and suppose they are assigned colors 1 to 5 in clockwise order as in Fig. 4.18. Consider the subgraph $H_{1,3}$ of $G - \{v\}$, consisting of the vertices colored with either color 1 or color 3 and all edges connecting vertices of color 1 to vertices of color 3. If v_1 and v_3 are not connected in this subgraph (i.e. there is no sequence of edges between them) we can interchange colors 1 and 3 in the part of the subgraph containing v_1 . It will finally be possible to color the vertex v with color 1 since it is no longer connected to a vertex with color 1. Instead, if v_1 and v_3 are in the same component of the subgraph

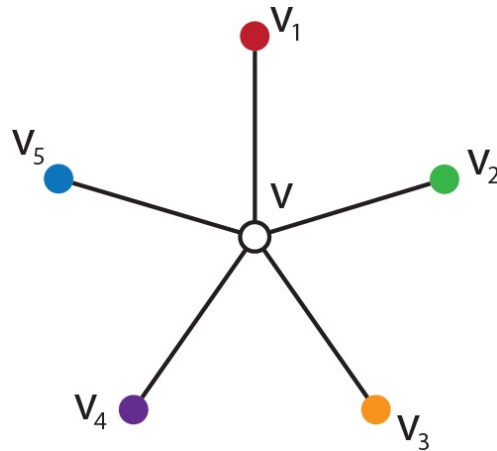


Figure 4.18: the vertex v and its other “surrounding” vertices.

$H_{1,3}$ of $G - \{v\}$ (i.e. there is a path connecting them in $G - \{v\}$), then we can see that the path connecting v_1 and v_3 in $G - \{v\}$ and the one connecting them through v form a cycle in the graph G . Therefore, either v_2 or both v_4 and v_5 will be enclosed

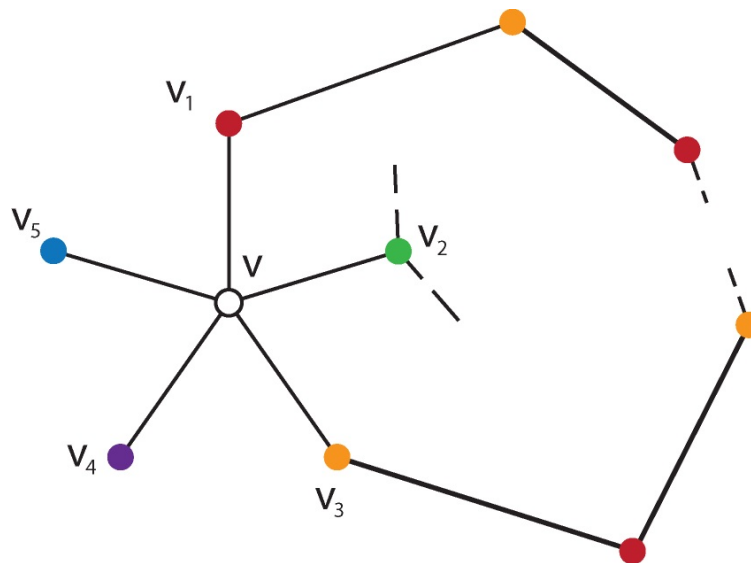


Figure 4.19: the situation in which there is a cycle connecting v_1 and v_3 ends with making another cycle connecting v_2 and v_4 impossible

in this cycle. If we proceed to form a subgraph $H_{2,4}$ of $G - \{v\}$ in the same way, we see that there cannot be a path connecting v_2 and v_4 in this subgraph.

We can proceed and switch colors 2 and 4 in the part of this last subgraph containing v_2 and we are then free to color v with color 2, obtaining a five-coloring of the graph G . \square

Back to the main result in map coloring, Appel and Haken in 1977 ((AH⁺77)) provided a computer-assisted proof that in fact four colors are enough to color every

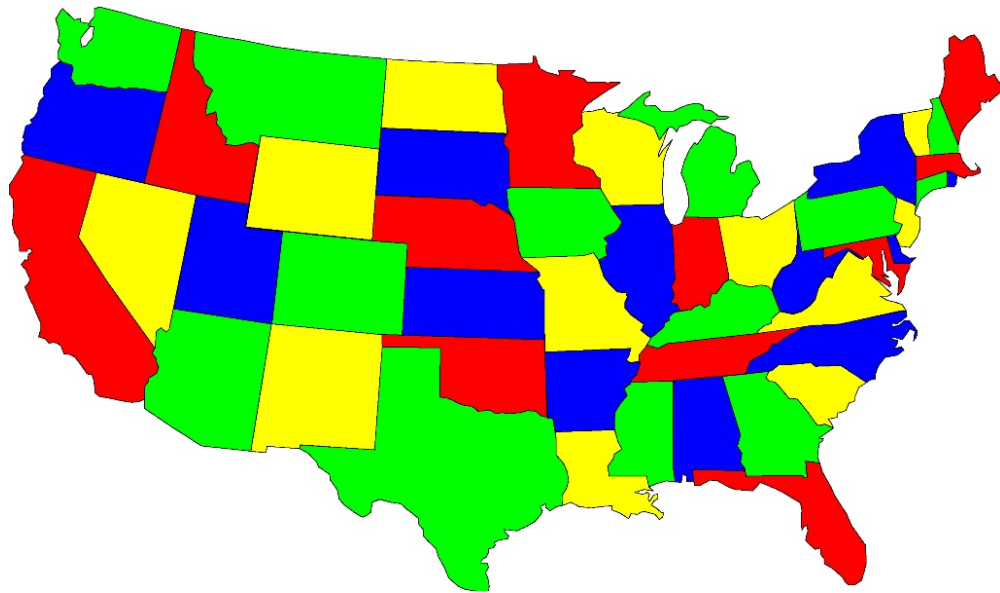


Figure 4.20: an example of a map of the USA colored with four colors.

planar map. The proof starts from the fact that the map that arise from a graph can always be drawn in such a way that no two edges cross eachother and then reduces all the possible cases to a finite number of configurations.

Before them, there are more then 100 years of tries between mathematicians and this proof. Guthrie and De Morgan were, as said, the first to pose this problem; in 1879 A.B. Kempe announced a proof of the theorem which lasted until, in 1890, Heawood ((Hea49)) showed that the proof was wrong proving in the meantime the five-colors theorem. The next major contribution was due to Birkhoff who allowed Franklin to prove, in 1922, that the four-color conjecture is true for maps with up to 25 regions. Other improvement of this result were provided in the following years and Heesch finally worked on reducibility and discharging, two last things to make the final proof possible.

In 1975, Martin Gardner in Scientific American even published a false counterexample to the theorem with a map which was supposed to need five colors, as seen in Fig. 4.21 where also a solution, which works according to the theorem is shown.

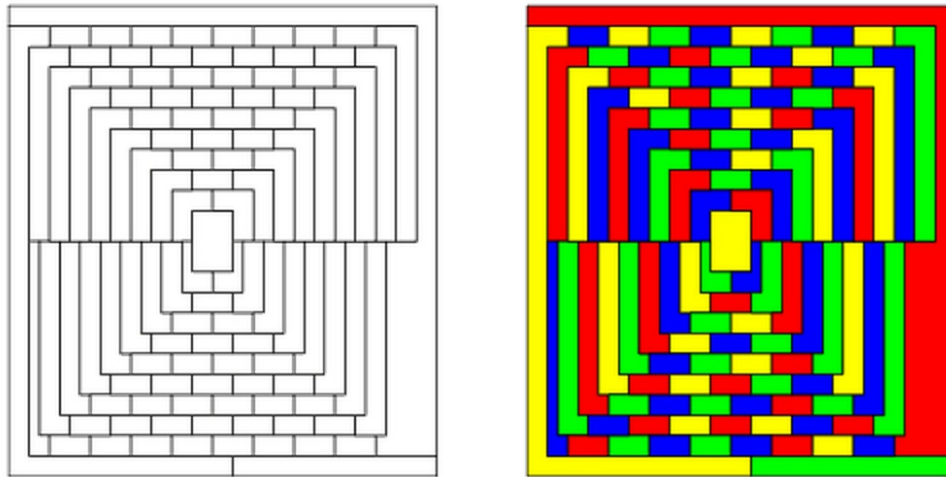


Figure 4.21: Gardner's proposed problem and the solution obtained by Wagon in 1998.

4.3 Cryptography

For this section the main references will be (Sti95), (Sch99) and (Sma03).

4.3.1 Cryptography in history

History of cryptography goes back in time to BC era. The need of humanity to transmit secret messages without being discovered has longed many years.

The first example found in history refers back to the Spartan Scytale Cipher, as seen in Fig. 4.22. This cypher consist of a (usually wooden) cylinder and a long thin



Figure 4.22: an example of a transposition cipher using a scytale.

strip of leather wrapped around the cylinder. Words were then written horizontally and, once unwrapped, could not be read by an intruder. The key to decrypt this was knowing the diameter of the cylinder used to encrypt the message.

A second example is the famous substitution cipher of Caesar. In this cypher you just need to substitute every letter with another one in the alphabet; usually simply

shifting them; e.g. $A \rightarrow D$, $B \rightarrow E$, $C \rightarrow F$, $D \rightarrow G$ and so on.

The disadvantage of these systems is that they are easy to break using the frequency

Original text	H	A	V	E	A	N	I	C	E	D	A	Y
	+	+	+	+	+	+	+	+	+	+	+	+
KEY	3	3	3	3	3	3	3	3	3	3	3	3
Ciphered text	K	D	Y	H	D	Q	L	F	H	G	D	B

Figure 4.23: an example of a Caesar’s cipher shifting every letter by 3.

of letter (i.e. letters which appear more frequently in the ciphertext are likely to be corresponding to letters which are more frequent in the alphabet) and finding the inverse substitution.

This remained the main problem in creating cyphers until the introduction of polyalphabetic substitution cyphers.

The most famous example of this is Vigenère cipher, probably the first cipher to use an encryption key which was not just a fixed number. Vigenère introduced a key which was used to change the transposition of each different letter. As we can see in the example (Fig. 4.24) the first letter of the message was substituted with the letter which is three places ahead in the alphabetical order, since the first letter of the key is C, the third letter of the english alphabet; in other words $H \rightarrow K$; in the same way the letter A becomes S, by effect of the key which is R in the second position and so on.

A more complex example in the beginning of the 20th century is the Germany

Original text	H	A	V	E	A	N	I	C	E	D	A	Y
	+	+	+	+	+	+	+	+	+	+	+	+
KEY	C	R	Y	P	T	O	C	R	Y	P	T	O
Ciphered text	K	S	U	U	U	C	L	U	D	T	U	N

Figure 4.24: an example of a Vigenère cipher with key “CRYPTO”.

Army’s ADFGVX cipher used during World War I, an example of product cipher. It works with a table which is fixed (as seen in Fig. 4.25) and used to encrypt the message. For every letter, substitute it with the pair (row, column) corresponding to it (e.g. $Q \rightarrow (F, A)$, $Y \rightarrow (G, F)$ and so on). After this there is a transposition part, using a fixed key, usually a word with all different letters, and reordering the part of the code obtained from the first passage, using the order of the letters of the key. This was improving the substitution cyphers but still not so difficult to decrypt

	<i>A</i>	<i>D</i>	<i>F</i>	<i>G</i>	<i>V</i>	<i>X</i>
<i>A</i>	<i>K</i>	<i>Z</i>	<i>W</i>	<i>R</i>	1	<i>F</i>
<i>D</i>	9	<i>B</i>	6	<i>C</i>	<i>L</i>	5
<i>F</i>	<i>Q</i>	7	<i>J</i>	<i>P</i>	<i>G</i>	<i>X</i>
<i>G</i>	<i>E</i>	<i>V</i>	<i>Y</i>	3	<i>A</i>	<i>N</i>
<i>V</i>	8	<i>O</i>	<i>D</i>	<i>H</i>	0	2
<i>X</i>	<i>U</i>	4	<i>I</i>	<i>S</i>	<i>T</i>	<i>M</i>

Figure 4.25: the ADFGVX table.

for a possible intruder since the substitution is fixed and easy to break.

During World War II, additional improvements were made regarding this. At the end of World War I, Arthur Scherbius invented the Enigma Machine, which used rotors to cipher and decrypt the messages. The same machine was used to both encrypt and decrypt messages, and the key (in this case the different rotors to use and their order) was changed every day. When it first was invented, Enigma was considered pretty safe and hard to break, but a group of Polish mathematicians already found a rule to break the system in the 30's.

This is how history bring us to the present; it is nowadays recognized that secrecy of a message depends on the secrecy of the key and not on the secrecy of the cryptosystem itself, as stated by Auguste Kerckhoffs von Nieuwenhof in 1883: “La sécurité d'un cryptosystème doit résider dans le secret de la clé. Les algorithmes utilisés doivent pouvoir être rendus publics” (Ker). In modern cryptography, the most established systems are public and quite spread among experts. The importance of exchanging keys and finding systems which makes it possible to keep keys secret are now the main task of modern cryptography experts.

4.3.2 Basic Cryptography, private and public keys

Every cryptographic situation begins with two parties, wanting to share a message, and a third part, usually called the intruder, willing to “overhear” them.

Definition 4.26. The *source* is the party (person or device) which wants to transmit the message. The *receiver* is the one the message is sent to.

Definition 4.27. The place where information is transmitted is called *channel*. A channel is *secure* if there is no one else than the two transmitting parties who is able to see the message. Otherwise the channel is *unsecure*.

Channels we will consider are usually unsecure. We want anyway to make it very difficult for untrusted listeners to understand the message transmitted. Moreover, we wish our data transmissions to be *safe* from errors in transmission such as corruption of data or interferences and possibly use a low bandwidth, transmitting the lowest number of bits. We will talk about this with more details in section 4.4.

We will continue seeing a few tools to improve the fulfillment of this wishes and therefore to keep our communications secure.

Definition 4.28. A *cryptosystem* is a 5-uple (P, C, K, E, D) where:

- P is a finite set of the possible *plaintexts*;
- C is a finite set of the possible *ciphertexts*;
- K is a finite set of possible keys;
- $E = \{e_k : P \rightarrow C\}_{k \in K}$ is the set of possible encryption functions. A *cipher* or *encryption algorithm* is a way to transform a plaintext m into ciphertext c , under the control of a secret *key* k .

$$c = e_k(m)$$

- $D = \{d_k : C \rightarrow P\}_{k \in K}$ is the set of the decryption functions. The process of *decryption* is the inverse process and use a *decryption algorithm* d :

$$m = d_k(c)$$

and such that

$$d_k \circ e_k = \text{Id}$$

The main goal we will use cryptography for is *confidentiality* of our messages. Among other goals cryptographers want to achieve the most important are *integrity* and *authentication*, usually through the use of a signature scheme, which makes the receiver sure that what he is receiving is coming from the source he thinks.

Definition 4.29. A cryptosystem is called *symmetric* or *private key cryptosystem*, if the same key is used for both encryption and decryption. In this process, the two parties have to share the key beforehand, and this key has to be kept secret from third parties.

All the historical example we saw above are cryptosystems with private key. As we pointed out above, the key is a fundamental parameter of the encryption and decryption algorithm. Even when the two algorithms are known for the attacker,

this knowledge is useless without knowing the key k (Kerckhoffs' principle). We can notice how Kerckhoffs' principle is working also when the encryption algorithm is kept secret (for military purposes for example); in fact, it does not state that the algorithm has to be kept secret, but how the message secrecy is preserved, due to secrecy of the key, even if the algorithm is discovered.

In private keys system, the number of possible keys must be very large. If the attacker knows or chooses some plaintexts and knows the encryption algorithm and the number of keys is quite small, then the attacker can, relatively easily, recover all keys and be able to decrypt every ciphertext.

We will briefly look at possible attacks to this ciphers:

- Passive Attacks where the intruder is only listening to encrypted messages. He can attempt to break the cryptosystem by recovering the key, either determining some secret that the communicating parties did not want leaked, or using some kind of frequency analysis if the cryptosystem allows it.
- Active Attacks where the adversary is allowed to insert, delete or replay messages between the two parties.

One of the main problems in symmetric key cryptography is finding a good way of sharing the key between the two parties. This is one of the reason that brought to the introduction of public cryptography.

Definition 4.30. A cryptosystem is called *asymmetric* or *public key cryptosystem* when each party involved in the communication has a pair of keys, a public key, which has to be made public, and a private key which is personal and secret.

Usually, one of the parties encrypt the message, using the public key, and the only one being able to decrypt it is the party which owns the corresponding private key.

There are many advantages of public key cryptography, for example there is no need to share the key between parties before communication can take place. Diffie and Hellman invented this kind of cryptography in (DH76).

Example 4.12. The RSA cryptosystem is one of the most used examples of public key cryptosystem. It was invented in 1978 by Rivest, Shamir and Adleman (ref. (RSA78)), and therefore takes their names. It works as follows:

- Alice chooses p and q distinct prime natural numbers;

- Alice computes

$$n = p \cdot q$$

and also the product

$$\varphi(n) = \varphi(p) \cdot \varphi(q) = (p - 1)(q - 1) = n - (p + q - 1)$$

where φ is Euler's totient function.

Recall that $\varphi(N)$ is counting the positive integers i , $1 \leq i \leq N$ which are relatively prime with N . In other words, for each $N \in \mathbb{N}$

$$\varphi(N) = |\{M \in \mathbb{N} \text{ s.t. } 1 \leq M \leq N \text{ and } \gcd(M, N) = 1\}|$$

- Alice chooses an integer e which is invertible in $\mathbb{Z}_{\varphi(n)}$, i.e. $\gcd(e, \varphi(n)) = 1$;
- Alice then computes d , multiplicative inverse of e , as: $d \equiv e^{-1} \pmod{\varphi(n)}$.
This is usually practically calculated using the extended Euclid's algorithm.

(n, e) is released as the public key while (p, q, d) is the private key.

When Bob wants to encrypt and send a message m to Alice, he computes the ciphertext as

$$c = m^e \pmod{n}$$

and he sends c to Alice.

Then Alice can then decrypt the message by

$$m = c^d = m^{ed} = m \pmod{n}$$

following from Euler-Fermat Theorem, since $d \cdot e \equiv 1 \pmod{\varphi(n)}$.

To be able to decrypt the message, an intruder should know the private key d , which is not easy to calculate knowing just n and e . The practical difficulty of factorizing the product of two large prime numbers p, q is lying at the base of the RSA system security.

As in RSA, what we require most in public key cryptography is a function or operation which is relatively easy to do in one way, but hard to do the other way (the inverse of the function corresponding to the decryption process here). Such mathematical tool is called one-way function.

Definition 4.31. A *one-way function* is a function which is easy to compute with every input, but it is hard to invert, i.e. find f^{-1} .

If f is a one-way function, then f^{-1} would be a problem whose output is hard to compute, but easy to check, just computing the value of f on it.

Definition 4.32. More formally, a function f is *one-way* if, given x it is easy to compute $f(x)$, but, given y in the codomain, it is hard to find x s.t. $f(x) = y$. In fact, any efficient algorithm solving a P-problem, succeded in inventing the function f with negligible probability.

Notice that a one-way function in sense of this second definition is not proven to exist. Its existence is connected to the P vs. NP problem and would imply $P \neq NP$.

4.4 Coding Theory

In addition to the previous references used for the cryptography section, see as well (Lin10) and (VL82), with the latter being the main reference. Coding Theory is the study of codes and their properties; we will use codes for cryptographic purposes, data compression and error correction.

In the trasmission situation we described in the previous sections, the main problem coding theory faces is that communications over these unreliable channels often result in errors in the transmitted message.

Definition 4.33. A code is based on *symbols*, which we will denote with a_i . Let $A = \{a_1, a_2, \dots, a_q\}$ be an *alphabet*, i.e. a collection of symbols.

We will mainly use the finite field with q elements \mathbb{F}_q and $(\mathbb{F}_q)^n$ as the space for our codes in the next pages, i.e. $A = \mathbb{F}_q$.

Definition 4.34. Let $k, n \in \mathbb{N}$, with $k \leq n$; a *linear block code* C is a k -dimensional vector subspace of $(\mathbb{F}_q)^n$. The *dimension* of the linear block code (which we will call just “code”) is k and its *length* is n .

An element of the code C is called *word*.

Note that we haven’t properly defined a *block* code, which gets the name from the fact that datas are encoded in blocks, nor a linear code. A *linear* code is simply a code in which linear combination of words has to be another word in the code.

The number of words of a code C of dimension k over $(\mathbb{F}_q)^n$ is

$$|C| = q^k$$

We usually refer to these codes as $[n, k, d]$ -codes.

The code is called *binary* if the field \mathbb{F}_q is \mathbb{F}_2 .

Definition 4.35. For any two vectors $x, y \in (\mathbb{F}_q)^n$, $x = (x_1, \dots, x_n)$, $y = (y_1, \dots, y_n)$ and for every i define:

$$d_i(x_i, y_i) = \begin{cases} 1 & \text{if } x_i \neq y_i \\ 0 & \text{if } x_i = y_i \end{cases}$$

and

$$d(x, y) = \sum_{i=1}^n d_i(x_i, y_i)$$

We call $d(x, y)$ the *Hamming distance* between x and y . It represents the number of coordinates where x and y differ.

The function d is a metric, that is, for every $x, y, z \in (\mathbb{F}_q)^n$,

- ▶ $0 \leq d(x, y) \leq n$;
- ▶ $d(x, y) = 0$ if and only if $x = y$;
- ▶ $d(x, y) = d(y, x)$;
- ▶ $d(x, z) \leq d(x, y) + d(y, z)$.

Definition 4.36. The Hamming distance between two words is usually referred to as just the “distance”, being the most used. The *distance of a code C* is the minimum distance between any two different words of the code.

$$d(C) = \min_{x, y \in C, x \neq y} \{d(x, y)\}$$

Definition 4.37. The *Hamming weight* of a word x is the number of non-zero coordinates of x ; $w(x) = d(x, 0)$. The weight of the code is the the minimum weight between non-zero words of the code.

$$w(C) = \min_{x \in C, x \neq 0} \{w(x)\}$$

Given a $[n, k, d]$ -code, the set $\{A_i\}$ is called the weight distribution of the code, where the single A_i is the number of words of weight i .

Example 4.13. Let’s consider the code generated by the words $c_1 = 1011100$, $c_2 = 0101110$, $c_3 = 0010111$).

To find all the codewords, just find all possible linear combinations of the generators getting:

$c_1 + c_2 = 1110010$, $c_1 + c_3 = 1001011$, $c_2 + c_3 = 0111001$ and $c_1 + c_2 + c_3 = 1100101$; the last word we are missing is 0000000.

The dimension of the code is 3 and its length is 7.

For example the distance between c_1 and c_2 is 4 (they differ in the 1st, 2nd, 3rd and 6th symbol), and we can notice that the distance between every pair of word is still 4. Therefore, the distance of the code is 4.

The weight of the code is 4, being the weight of all words, except the 0 one. The weight distribution is: $A_0 = 1$, $A_1 = A_2 = A_3 = 0$, $A_4 = 7$, $A_5 = A_6 = A_7 = 0$.

Definition 4.38. A *cyclic code* is a code in which, for every codeword $c = (c_1, \dots, c_n) \in C$ there exists also a codeword $c' = (c_n, c_1, \dots, c_{n-1}) \in C$. c' is a *cyclic shift* of c .

Definition 4.39. We can represent a code C using a matrix G formed by a minimum set of linear generators of the code itself, called *generator matrix* of C . We can also see the encoding process as an operation $c = m \cdot G$, where m is the message to transmit.

Example 4.14. The generator matrix for the code of the last example is:

$$G = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

4.4.1 Error Correcting Codes

There are usually four kinds of errors or interference:

- Errors in transmission, when a bit in the message is changed;
- Erasures, when a bit is not understood by the decoder (but it still knows how many bits were transmitted);
- Insertions, when the decoder understands a sequence which is longer than the one transmitted;
- Deletions, when the decoder understands a sequence which is shorter than the one transmitted;

Most codes will deal especially with detecting and correcting errors and erasures. In particular, we will now see how we can use a $[n, k, d]$ -code to recover from errors in data transmissions.

Definition 4.40. What we need is a *coding procedure*, an algorithm which computes an n -symbol vector for each k -symbol vector given as an input. With this tool we are coding blocks of the message into longer words, belonging to our code. The algorithm has to be invertible and there exists a *decoding procedure* which either gives us the original message or a warning status, saying that there were errors in the transmission.

Definition 4.41. The *error detection capability* of a code C is the maximum number of errors that a decoding procedure for C can always detect.

Definition 4.42. The *error correction capability* of a code C is the maximum number of errors that a decoding procedure for C can always correct.

Proposition 4.4.1. A $[n, k, d]$ -code C on \mathbb{F}_q has detection capability $d - 1$ and correction capability $t = \lfloor \frac{d-1}{2} \rfloor$, where $\lfloor x \rfloor := \{m \in \mathbb{Z} | x - 1 < m \leq x\}$.

Proof Detection. Suppose $d - 1$ errors occur. The transmitted word x becomes a vector e which is at distance $d - 1$ from x . Therefore e cannot be in the code C , since its distance is d . On the other side if there are d errors, it is possible that,

from one word of the code, we receive a different word of C , getting misled and not detecting the error.

Correction. If at most $\lfloor \frac{d-1}{2} \rfloor$ errors occur, the vector we receive is closer to the right transmitted word than any other word of C . So we can search for the, unique, word of C which is closer to the received one. If we have more than $\lfloor \frac{d-1}{2} \rfloor$ errors, it is impossible for the decoder to understand which word was sent. To see this, suppose $\tau = \lfloor \frac{d-1}{2} \rfloor + 1$.

If d is even, then $\tau = d/2$; take a word $c = (c_1, \dots, c_n)$ of weight d and let $I = \{i | c_i \neq 0\}$, the indexes of the coordinates of c that differ from 0. Let A and B s.t. $I = A \sqcup B$ and $|A| = |B| = d/2$. Construct a vector $a = (a_1, \dots, a_n)$ s.t. $A = \{i | a_i \neq 0\}$. In this way we forced $d(0, a) = d/2 = d(c, a)$. From this, C has two codewords which are “nearest” and therefore t must be smaller than τ .

If d is odd, then $\tau = \frac{d+1}{2}$; take a word $c = (c_1, \dots, c_n)$ of weight d and let $I = \{i | c_i \neq 0\}$, the indexes of the coordinates of c that differ from 0. Let A and B s.t. $I = A \sqcup B$ and $|A| = \frac{d+1}{2}$ and $|B| = \frac{d-1}{2}$. Construct a vector $a = (a_1, \dots, a_n)$ s.t. $A = \{i | a_i \neq 0\}$. In this way we forced $d(0, a) = \tau$ and $d(c, a) = \tau - 1$. From this, if we transmit 0 and receive the word a with τ errors, it would be closer to c than it is to 0, and not only we wouldn't be able to correct the error, but we would get into a correction mistake. \square

The problem of finding the distance of a code is the same of finding the minimum weight of a word (different from 0) of the code. The problem is generally difficult and no sub-exponential algorithms to do this are known yet.

Example 4.15. Back to the code of the previous section with generator matrix

$$G = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

This code can detect up to $4 - 1 = 3$ errors and correct $\lfloor \frac{4-1}{2} \rfloor = 1$ error.

Example 4.16. Consider the code C generated by $c_1 = (1, 1, 1)$. It is a $[3, 1, 3]$ -code, since its length is 3, its dimension 1 and the distance 3. This code can detect up to $3 - 1 = 2$ errors and correct $\lfloor \frac{3-1}{2} \rfloor = 1$ error.

Suppose we receive as a transmission $r = (0, 1, 1)$. The error is immediately detected, since $r \notin C$. We should then find the distances between r and the other codewords. We can compute $d(r, c_1) = 1$ and $d(r, (0, 0, 0)) = 2$ and we can correct the error deducing that the word that was meant to be transmitted is c_1 .

If, unluckily, the number of errors was > 1 , we wouldn't be able to correct it; actually, we would probably have corrected it wrong.

Definition 4.43. Let C be a $[n, k, d]$ -code. The *dual code* C^\perp of C is the set of all n -vectors which are orthogonal to all codewords. Recall the two vectors are orthogonal if their scalar product is 0.

There are also codes which are orthogonal to themselves, they are called *self-dual*, i.e. $C^\perp = C$.

The dual code of a $[n, k, d]$ -code is a $[n, n - k, d']$ -code; note also that

$$(V^\perp)^\perp = V$$

and

$$\dim V + \dim V^\perp = n$$

Definition 4.44. A generator matrix H for C^\perp is called a *parity-check matrix* for C .

To check if a vector x is a word of C , we just need to check that $xH^T = Hx^T = 0$.

4.4.2 Perfect Codes and Hamming Codes

We first note the following fact: the number of (binary) words (all vectors) of length n and distance i from a fixed word c is

$$\binom{n}{i} = \frac{n!}{i! \cdot (n - i)!}$$

If we fix a maximum distance d_m from a word c , the number of possible words within this distance (c included) is

$$\binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{d_m}$$

We can in this way consider the binary sphere of radius d_m with center in a codeword; its volume is the number of vectors in the sphere (i.e. with distance $\leq d_m$).

Theorem 4.4.2 (Hamming Bound). *Consider a binary $[n, k, d]$ code; then*

$$2^k \left[\binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{t} \right] \leq 2^n$$

or

$$\sum_{i=0}^t \binom{n}{i} \leq 2^{n-k}$$

Proof Consider the sphere of radius $\lfloor \frac{d-1}{2} \rfloor$, centered at the codewords. We saw above how the number of words in this sphere is

$$\sum_{i=0}^{\lfloor \frac{d-1}{2} \rfloor} \binom{n}{i}$$

None of these spheres intersect each other, being the minimum distance d , and therefore each word is covered at most once. So we have

$$|C| \cdot \sum_{i=0}^{\lfloor \frac{d-1}{2} \rfloor} \binom{n}{i} \leq 2^n$$

and the Bound follows, being $|C| = 2^k$. \square

Definition 4.45. A linear code $C \in (\mathbb{F}_2)^n$ is *perfect* if for every possible word $w \in (\mathbb{F}_2)^n$, there is a unique codeword $c \in C$ with distance at most $t = \lfloor \frac{d-1}{2} \rfloor$ from w .

In other words, the code is perfect if the spheres of radius t centered at the codewords partition the space $(\mathbb{F}_2)^n$.

A code is perfect if equality is holding in the Hamming bound expression.

A proof of this is straightforward, as it is the only possibility for the spheres to cover all the space $(\mathbb{F}_2)^n$. The Hamming bound is sometimes used as a definition of perfect code.

In other words, a code is said to be perfect if it does not exist a word in $(\mathbb{F}_2)^n$ which cannot be corrected into a codeword of C .

Theorem 4.4.3. *A perfect binary error-correcting code $C \in (\mathbb{F}_2)^n$, satisfies one of the following:*

- $|C| = 1, t = n;$
- $|C| = 2^n, t = 0;$
- $|C| = 2, n = 2t + 1;$
- $|C| = 2^{12}, t = 3, n = 23;$
- $|C| = 2^{n-r}, t = 1, n = (2^r - 1)$ with $r > 1;$

Proof A proof of this theorem is due to Tietavainen and van Lint ((VL71) and (Tie73)).

The first two are trivial codes and the third one repetition codes. The fourth point represents Golay Codes (which we will not treat further, see (Gol49)). The last point represents Hamming Codes.

Definition 4.46. A code C is a (binary) $[2^r - 1, 2^r - 1 - r, d]$ *Hamming code* if its parity-check matrix is an $r \times (2^r - 1)$ matrix, whose columns are all the binary vectors (non-zero) of length r .

Example 4.17. The most famous example of a Hamming code is the $[7, 4, 3]$ -code introduced by Hamming in 1950. The generator matrix of C is

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

and its parity-check matrix

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

We can see that the distance of the code is 3 (as in every Hamming code).

Looking at G the 3rd, 5th, 6th and 7th rows are used to encode a 4-bit message and the other rows are used to check on the parity and correct up to 1 error in transmission.

Groebner basis: some theory and definitions:

The name *Groebner basis* was first introduced by Bruno Buchberger in 1965 ((Buc65)). Similar notions in the field were introduced before and around that date and many applications have been found since then.

Main reference for this section is (AL94).

All along this section k will be a field and $k[x_1, \dots, x_n]$ the polynomial ring with coefficients in k and n variables. Also, x^α , where $\alpha = (\alpha_1, \dots, \alpha_n)$, will denote the monomial $x^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n}$.

Definition 4.47. Let R be a (commutative) ring. $I \subseteq R$ is called an *ideal of R* if:

- $0 \in I$;

- for each a, b in I , then $a + b \in I$;
- for each $a \in I$ and $r \in R$, $a \cdot r \in I$.

Definition 4.48. The *degree of a monomial* x^α is $\sum_i \alpha_i$. The *degree of a polynomial* f is the maximum degree of the monomials forming f .

Definition 4.49. An order \leq on the set of the monomials of $k[x_1, \dots, x_n]$ is a *monomial order* if:

- the order \leq is total, i.e. for every α, β we have either $x^\alpha \leq x^\beta$ or $x^\alpha \geq x^\beta$;
- the order \leq is multiplicative, i.e. if $x^\alpha \leq x^\beta$ then $x^\alpha x^\gamma \leq x^\beta x^\gamma$ for every γ ;
- there are no infinite descending chains $x^{\alpha(1)} > x^{\alpha(2)} > \dots$.

Straightforward from these, we have that, for every α , $x^\alpha > 1$.

Example 4.18. Lexicographical order: $\alpha <_{lex} \beta$ if $\alpha_i < \beta_i$ where i is the minimal index with $\alpha_i \neq \beta_i$. For example, in $k[x, y]$, $xy^{200} < x^2y$ and $xy^3 > x$.

Example 4.19. Degree lexicographic order: $\alpha <_{glex} \beta$ if $|\alpha| < |\beta|$ or $|\alpha| = |\beta|$ and $\alpha <_{lex} \beta$ (where $|\alpha| = \alpha_1 + \dots + \alpha_n$). In this order $1 < x_1 < x_2 < \dots < x_1^2 < x_1x_2 < \dots < x_2^2 < \dots$.

Definition 4.50. The *leading monomial* of f , denoted $\text{LM}(f)$ is the monomial (with non-zero coefficient) which is maximal with respect to the order used. The *leading coefficient* $\text{LC}(f)$ is the coefficient of $\text{LM}(f)$.

With this notation and a polynomial division algorithm, on a polynomial ring $R = k[x_1, \dots, x_n]$, given $f_1, \dots, f_s \in R, g \in R$, we can compute $r \in R$ such that $g = h_1f_1 + \dots + h_sf_s + r$ with all $h_i \in R$ and no monomial of r is divisible by $\text{LM}(f_i)$ for each $i \in \{1, \dots, s\}$.

Definition 4.51. Such r is called the *remainder of g modulo f_1, \dots, f_s* .

When repeating this process, dividing a fixed polynomial by a set of polynomials, we talk about *polynomial reduction*.

Definition 4.52. For $A \subseteq \mathbb{N}^n$, the ideal

$$I = \langle x^\alpha, \alpha \in A \rangle \subseteq R = k[x_1, \dots, x_n]$$

is called a *monomial ideal*.

Lemma 4.4.4. (Dickson) Let $A \subseteq \mathbb{N}^n$ and $I = \langle x^\alpha, \alpha \in A \rangle \subseteq R = k[x_1, \dots, x_n]$. Then there exists $A' \subseteq A$, with A' finite and $I = \langle x^\alpha, \alpha \in A' \rangle$.

Proof A proof of Dickson's Lemma can be found in (AL94), pag.23.

Theorem 4.4.5. (*Hilbert's Basis Theorem*) If I is an ideal of $R = k[x_1, \dots, x_n]$, then there exist polynomials $f_1, \dots, f_s \in k[x_1, \dots, x_n]$ such that $I = \langle f_1, \dots, f_s \rangle$

Proof A proof of the theorem can be found in (AL94).

Definition 4.53. Let $I \subseteq R = k[x_1, \dots, x_n]$ be an ideal and $\langle \text{LM}(I) \rangle := \langle \text{LM}(i) \mid i \in I \rangle$. $G \subseteq I$ such that $\langle \text{LM}(I) \rangle = \langle \text{LM}(g) \mid g \in G \rangle$ is called a *Gröbner basis* of I .

Lemma 4.4.6. Let $I \subseteq k[x_1, \dots, x_n]$ be an ideal. Then $G \subseteq I$ is a Gröbner basis of I if and only if for all $f \in I$ there is a $g \in G$ such that $\text{LM}(g)$ divides $\text{LM}(f)$.

Proof We have that G is a Gröbner basis of I if and only if $\langle \text{LM}(g) \mid g \in G \rangle = \langle \text{LM}(I) \rangle$, so if and only if for all $f \in I$ there exists $g \in G$ such that $\text{LM}(g)$ divides $\text{LM}(f)$. \square

Proposition 4.4.7. Let $I \subseteq k[x_1, \dots, x_n]$ be an ideal and $G \subseteq I$ a Gröbner basis. Let $F \in R$ and $r \in R$ be a remainder of f modulo G . Then r is uniquely determined, i.e. it does not depend on the choices made during the division algorithm. Moreover, $r = 0$ if and only if $f \in I$.

Proof A proof of this proposition can be found in (AL94).

From what stated above, every ideal has a finite Gröbner basis.

Definition 4.54. Let $f, g \in R$, $f, g \neq 0$ and $\text{LM}(f) = x^\alpha$, $\text{LM}(g) = x^\beta$; let $\gamma = (\gamma_1, \dots, \gamma_n)$ where $\gamma_i = \max(\alpha_i, \beta_i)$ for each i . We call x^γ the *least common multiple* of $\text{LM}(f)$ and $\text{LM}(g)$.

The *S-polynomial* of f_1 and f_2 is

$$S(f_1, f_2) = \frac{x^\gamma}{\text{LC}(f_1) \cdot \text{LT}(f_1)} \cdot f_1 - \frac{x^\gamma}{\text{LC}(f_2) \cdot \text{LT}(f_2)} \cdot f_2$$

Theorem 4.4.8. (*Buchberger's criterion*) Let $I \subseteq R = k[x_1, \dots, x_n]$ be an ideal generated by $G \subseteq I$. Then G is a Gröbner basis for I if and only if, for all $i \neq j$, the remainder of the division of $S(g_i, g_j)$ by G is 0; i.e. all $S(g_i, g_j)$ reduces to zero modulo G .

Proof See (Eis95) for a proof of the Criterion.

From the work of Buchberger, we also have the following algorithm that, given $\{g_1, \dots, g_s\}$ generating I , computes a Gröbner basis of I with respect to a given order. The main idea is to add up elements to the set we are starting with, until we are able to show that the set we obtain is a Gröbner basis of I .

- Set $G_0 := \{g_1, \dots, g_s\}$ and $i := 0$;
- if $S(g_i, g_j)$ reduces to zero modulo G_i for all $g_i, g_j \in G_i$, then G_i is a Gröbner basis of I ;
- if there are g_i, g_j such that the last step does not happen, we set $G_{i+1} = G_i \cup \{r\}$ and $i := i + 1$, and repeat the previous step.

A proof of the fact that Buchberger's algorithm terminates can be found for example in (Eis95).

Definition 4.55. Let $G = \{g_1, \dots, g_s\}$ be a Gröbner basis of an ideal $I \in k[x_1, \dots, x_n]$. The G is called a *reduced Gröbner basis* if and only if, for each $i = 1, \dots, s$, the leading coefficient of every polynomial in G is 1 and any monomial of a polynomial $f \in G$ is not in the ideal generated by the leading terms of the other polynomials in G .

An ideal $I \subseteq R = k[x_1, \dots, x_n]$ has a unique reduced Gröbner basis with respect to the given monomial order.

The application we will use Groebner bases for is the problem of solving polynomial equations. The problem is, given $f_1, \dots, f_s \in k[x_1, \dots, x_n]$, to determine whether there is a vector $(a_1, \dots, a_n) \in k^n$ with $f_1(a_1, \dots, a_n) = \dots = f_s(a_1, \dots, a_n) = 0$ or not.

Gröbner bases are very useful for solving systems of polynomial equations; in fact, given a finite set of polynomials F the complex zeros does not change if we replace F by another set of polynomials F' such that F and F' generate the same ideal in $k[x_1, \dots, x_n]$.

Let us formalize this concept. Recall that a field is *algebraically closed* if every polynomial in $k[x]$ has a zero in k .

Theorem 4.4.9. (*Hilbert's Nullstellensatz*) Let k be an algebraically closed field, $R = k[x_1, \dots, x_n]$ and $f_1, \dots, f_s \in R$. Then, $1 \in I = \langle f_1, \dots, f_s \rangle$ if and only if the polynomials f_1, \dots, f_s fail to have a common solution. In other words, $a = (a_1, \dots, a_n) \in k^n$ such that $f_1(a) = \dots = f_s(a) = 0$ does not exist.

Proof A proof of this, using Gröbner bases, can be found in (Gle12).

Conversely, $1 \notin I$ implies that the system admits at least one solution.

To determine whether 1 belongs to an ideal I or not, we can compute a Gröbner basis G of the ideal generated by the polynomials. Then 1 reduces to 0 modulo G if and only if $1 \in I$.

An application and some examples of this will be provided in section 4.2.4.

4.5 Topics in graph theory

For this section I will use (Eve11), as well as the main references on graphs used in the previous chapters and some specific references which will be cited throughout the text.

There are many different topics to introduce children to graph theory.

4.5.1 Hamiltonian and Eulerian paths

Recall that a *path* in a graph is a, finite or infinite, sequence of edges which connect a sequence of vertices.

Recall our example in section 4.2.1); we can now formally define an Eulerian path.

Definition 4.56. An *Eulerian path* in a graph G is a path that contains every edge of G exactly once. If the path is closed (i.e. it forms a cycle), we have an *Eulerian circuit*.

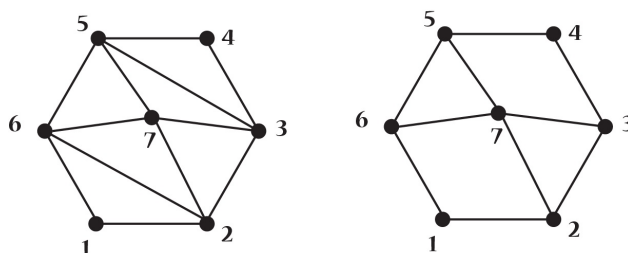


Figure 4.26: In the first example 1-2-3-4-5-6-7-5-3-7-2-6-1 is a possible Eulerian path and circuit; in the second graph it is not possible to find such an example.

The Eulerian path, i.e. the problem to find at least an Eulerian path in a graph, is quite easy even for kids to solve and provides a good start point to talk later about hard problems on graphs.

The following results hold:

Theorem 4.5.1. A graph G has at least one Eulerian path if and only if it is connected and has either zero or two vertices of odd degree.

Theorem 4.5.2. A graph G has at least one Eulerian circuit if and only if it is connected and has zero vertices of odd degree.

Proof A proof of these facts was conjectured by Euler and later proven by Hierholzer (see (BLW76)).

By the previous theorem, we mathematically justify the explanation of the Königsberg's bridges problem we saw in section 4.2.1.

Definition 4.57. A *Hamiltonian path* in a graph G is a path that contains every vertex of G exactly once. If the path forms a cycle, we have a *Hamiltonian circuit*.

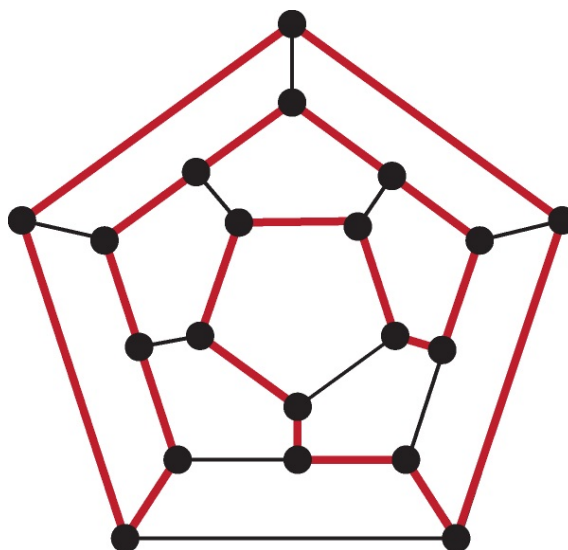


Figure 4.27: The edges in red show an example of a Hamiltonian path along the edges of a dodecahedron such that every vertex is visited a single time, and the ending point is the same as the starting point. This is the solution of the problem known as the “icosian game”, invented by W.R. Hamilton.

Opposite to the Eulerian path problem, the Hamiltonian path problem, i.e. the problem to find at least a Hamiltonian path in a graph, is generally hard to solve, and determining whether we can find this kind of paths and circuits in a graph is NP-complete.

4.5.2 Minimum weight spanning trees

Let's take a graph as in the previous chapter $G = (V, E)$.

The story usually associated with the problem we are introducing is the one of the road system in a region, i.e. the problem to pave some roads between different cities in such a way to connect all the possible cities with each other, but making the length of the total road network as short as possible. As we will see in Chapter 6, it will be possible to introduce the problem to kids in such a way.

Let's begin with some additional definitions to be able to understand mathematically this problem.

Definition 4.58. A undirected graph in which, taken any two vertices, they are connected by exactly one single path, is called a *tree*.

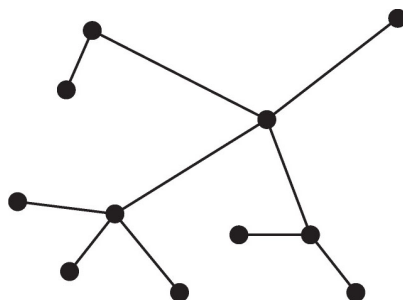


Figure 4.28: An example of a graph that is also a tree.

Recall that a connected graph is a graph with at least a path between each random two vertices we pick. A connected graph with no cycles is a tree.

Also, a tree is connected by definition, but it is not any more connected if we take out from E just one single edge.

Theorem 4.5.3. A tree T with n vertices has exactly $n - 1$ edges.

Proof First of all we show that at least one vertex of the tree has degree 1 (see definition 4.18). By contradiction, suppose T has no such vertices, then we can start from one vertex and follow a sequence of distinct edges until coming back to the same vertex; this is possible because every vertex has degree ≥ 2 . Since a tree has no cycles, there has to be at least one vertex with degree 1.

Now we proceed by induction. If a tree has $n = 1$ vertex, then it has 0 edges. We suppose that each tree with $n - 1$ vertices has $n - 2$ edges. Consider a tree T with n vertices. We can remove from T a vertex with degree 1, and the edge connecting it to the rest of the graph, obtaining a tree T' with $n - 1$ vertices. By the inductive hypothesis, T' has $n - 2$ edges and therefore T has $n - 1$ edges. \square

Definition 4.59. Given a undirected graph $G = (V, E)$, a subgraph $T = (V_T, E_T)$ of G is a *spanning tree* of G if it is a tree such that $V = V_T$, $E_T \subseteq E$.

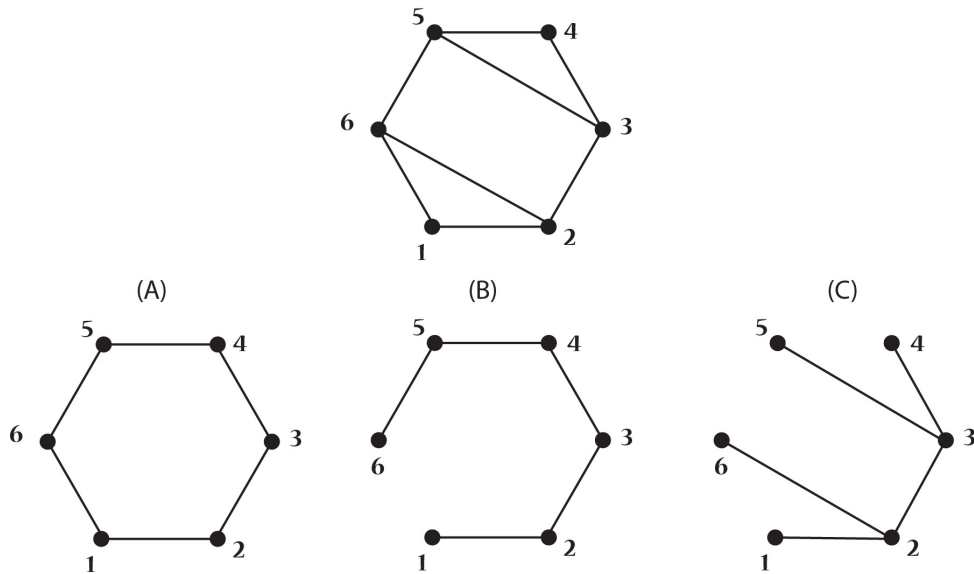


Figure 4.29: A graph G . Figure A is not a spanning tree (since it is not a tree for G); B and C are some possible spanning trees for G .

Examples of possible spanning trees are shown in Fig. 4.29.

Theorem 4.5.4. *Every connected graph has at least a spanning tree.*

Proof The intuitive idea is the following: if it is not possible to take off any of the edges, then the graph is already a spanning tree for itself, otherwise we proceed by elimination of “unnecessary” edges.

A formal proof is done by induction on the number of edges. If G is connected and has 0 edges, then it is a single vertex and therefore also a tree. Now suppose G has $n > 1$ edges; if G is a tree, the theorem is proved since it is its own spanning tree. If G is not a tree, it has at least one cycle; we can remove one edge from this cycle. The resulting graph G' is still connected and has $n - 1$ edges, therefore it has a spanning tree by inductive hypothesis; this spanning tree is also a spanning tree for G . \square

We can also introduce the following notion:

Definition 4.60. A graph G in which each edge has a weight, represented by a real number, is called a *weighted graph*. The *weight of a graph* is the sum of the weights for all edges.

Definition 4.61. Given a weighted graph G , a *minimum spanning tree* is a spanning tree of minimum weight.

This definition does not necessarily imply uniqueness.

Theorem 4.5.5. *If all edges have pairwise different weights, then the minimum spanning tree is also unique.*

Proof A proof of this fact using Kruskal's Algorithm, which we are going to see, is found in (Kru56).

The problem is now to find an efficient way to produce a minimum spanning tree, given a weighted graph G . in the following part we describe two algorithms to do this.

Kruskal's algorithm (Kru56) :

Let $G = (V, E)$ be a weighted graph.

1. Consider the subgraph $T = (V, \emptyset)$ of G which is just a collection of all vertices of G without edges;
2. order all edges by their weights in G and create a set S containing them all;
3. consider the edge with minimum weight of S ; remove it from S ;
4. if the edge we removed from S forms a cycle in T then discard it, otherwise add it to T ;
5. if $S \neq \emptyset$, go back to step 3;

The algorithm terminates correctly and can be shown to run, efficiently, in $O(n \cdot \log n)$ time, where n is the number of edges in G ((MS⁺94)).

Prim's algorithm (Pri57) :

Let $G = (V, E)$ be a weighted graph.

1. Start with a single vertex x of G and set $T = (\{x\}, \emptyset)$;
2. choose the edge e of minimum weight, of all the edges connecting T to the vertices in $G \setminus T$;
3. add the edge e to T together with the vertex which was not in T and was connected to T by e ;

4. if $G \setminus T$ has a number of vertices ≥ 1 , go back to step 2;

The algorithm terminates correctly and can be shown to run, efficiently, in $O(n \cdot \log n)$ time, where n is the number of edges in G ((MS+94)).

We saw that the algorithms work in polynomial time, therefore related decision problems (e.g. determining whether an edge is in the minimum spanning tree) are in the complexity class P.

Example 4.20. We will try to find a minimum spanning tree for G (in Fig. 4.30) with both Kruskal's and Prim's algorithms.

We know the solution will have 5 edges, given that G has 6 vertices. Following

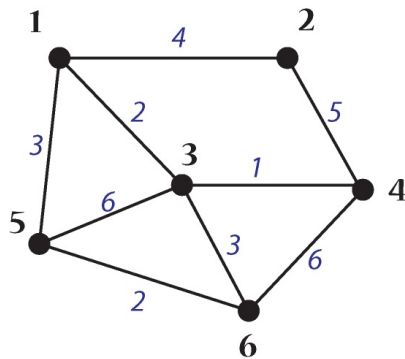


Figure 4.30: The graph G .

Kruskal's algorithm, the first edge we consider is $3 \leftrightarrow 4$, being the one of minimal weight 1. Then we can add both $1 \leftrightarrow 3$ and $5 \leftrightarrow 6$, of weight 2, since they do not form any cycle together with $3 \leftrightarrow 4$. We can further add $3 \leftrightarrow 6$, which has weight 3, since it does not form cycles with $3 \leftrightarrow 4$, $1 \leftrightarrow 3$ and $5 \leftrightarrow 6$, but we cannot take $1 \leftrightarrow 5$ because otherwise we will get a cycle $1 \leftrightarrow 5 \leftrightarrow 6 \leftrightarrow 3 \leftrightarrow 1$. Next one we add is $1 \leftrightarrow 2$ which has weight 4 and finally we get to the solution, which is seen in the last step of Fig. 4.31.

Following Prim's algorithm, starting from vertex 1, the edge leaving 1 with the minimal weight is $1 \leftrightarrow 3$; then we will choose $3 \leftrightarrow 4$ and subsequently $1 \leftrightarrow 5$; the next one is $5 \leftrightarrow 6$ and the last one has to be $1 \leftrightarrow 2$ which has weight 4, because the ones remaining with lower weight (e.g. $3 \leftrightarrow 6$) would create cycles with the edges previously considered. The final solution is shown in Fig. 4.32.

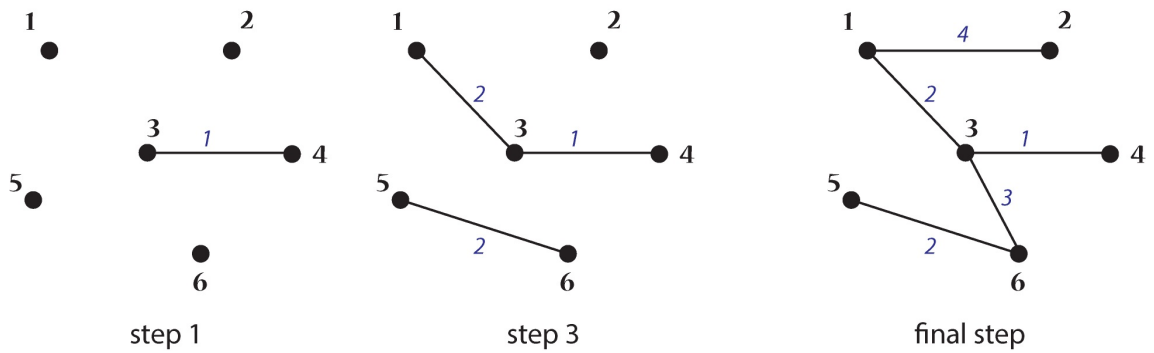


Figure 4.31: Finding the minimum spanning tree for G with Kruskal's algorithm, after adding respectively one, three and all five edges needed for the final solution.

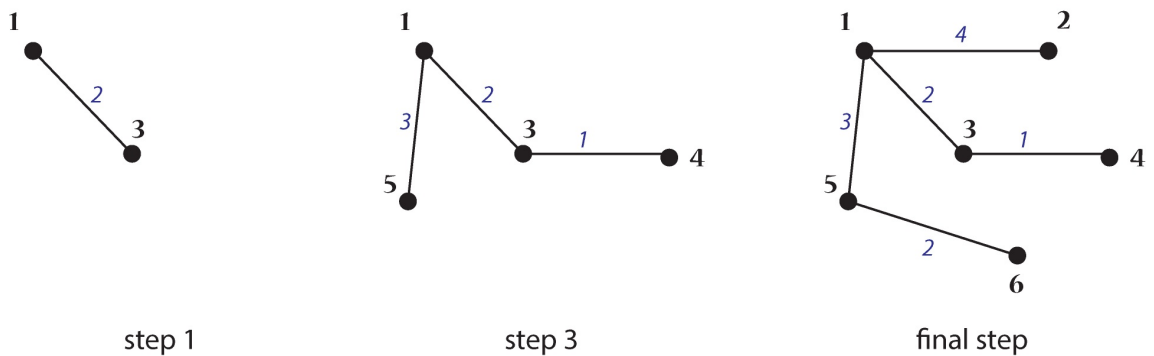


Figure 4.32: Finding the minimum spanning tree for G with Prim's algorithm, after adding respectively one, three and all five edges and vertices needed for the final solution.

Among the practical applications of this problem in graph theory we can find:

- computer networks and telecommunications;
- transportation networks;
- image registration and segmentation;
- taxonomy and other classifications.

These and many others are the possible applications of the minimum weight spanning tree problem. We will use it as an introduction to kids to problems which can be solved in polynomial time and greedy algorithms, algorithms which follows the problem and build a solution “piece by piece”.

4.5.3 Minimum dominating sets

In this section we will use (LS90) as main reference.

Another problem, still related to graphs, is the search of a minimum dominating set of a graph G .

As we will see in Ch. 6 the problem is easy to present to children by just introducing some practical examples of this, usually minimal facility location in a city or region works well.

Definition 4.62. Let $G = (V, E)$ be a graph. A vertex i is said to *dominate* a vertex j if $i = j$ or if E contains an edge from i to j .

A set S of vertices, $S \subseteq V$ is said to be a *dominating set* of G if every vertex of G is dominated by at least one element of S .

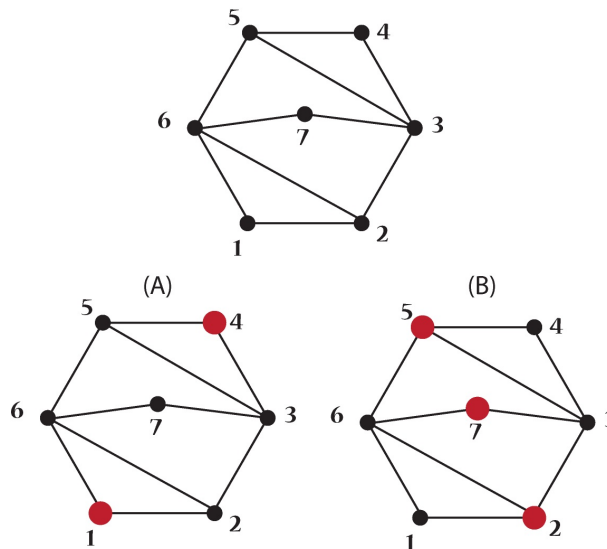


Figure 4.33: A graph G . The set of the red vertices in Figure A is not a dominating set for G (since the vertex 7 is not dominated by any of the red vertices); B is a possible dominating set for G .

Definition 4.63. The set S is called a *perfect dominating set (PDS)* of G if each vertex of G is dominated by at exactly one element of S .

We are then looking for the set of least cardinality among all dominating sets for a graph G .

Example 4.21. Given a graph G , find a perfect dominating set for it. We can try to produce a dominating set (subset of vertices in red) for G as in A in Fig. 4.34; we will notice how there are vertices that are dominated by more than one red

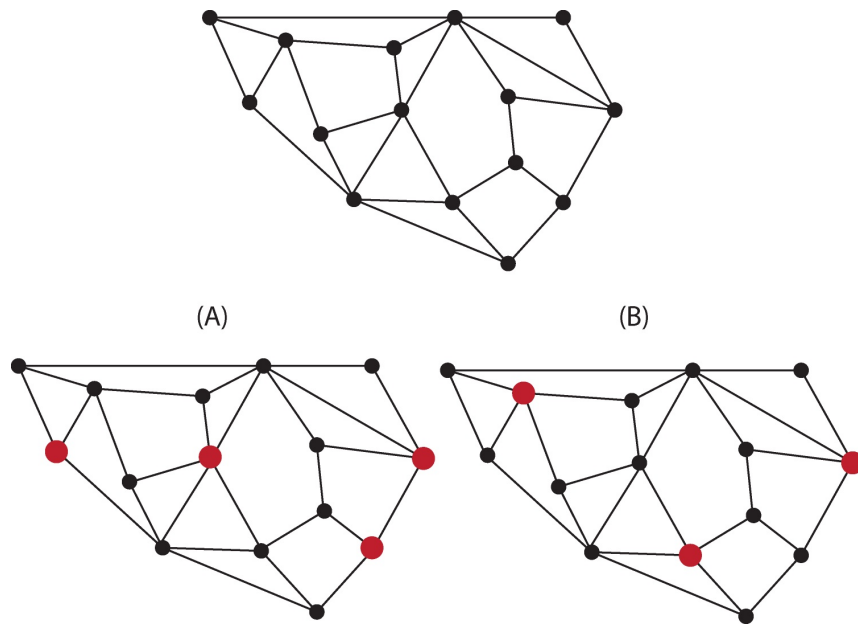


Figure 4.34: A graph G and the attempts A and B as described in the example 4.21.

vertex. For example, the two red vertices on the right of the graph are dominating each other, other than themselves; so the set of red vertices in A is not a perfect dominating set. In example B, each of the red vertices dominate five vertices (four other plus itself), covering exactly the fifteen vertices of G , without any vertex being dominated by two different red vertices. So B is a perfect dominating set for G .

Remark By definition, a dominating set for a graph can not be empty (unless the graph is $G = (\emptyset, \emptyset)$). So a minimal one exists.

Definition 4.64. The number of vertices of a graph G in a smallest dominating set for G is called *domination number of G* and it is denoted by $d(G)$.

Any dominating set which has cardinality equal to the domination number is called a *minimum dominating set* for the graph G .

By definition, it follows that a perfect dominating set for a graph G is also a minimum dominating set for G .

Viceversa, it is easy to produce an example of a minimum dominating set which is not perfect:

Example 4.22. In figure 4.35 (A) we can see an example of a minimum dominating set for a graph which is not perfect, since vertices 2, 5 and 7 are dominated by both the red dots in 3 and in 6.

In figure 4.35 (B) we have two different minimum dominating sets for the graph, the first one being not perfect while the second one is a perfect dominating set.

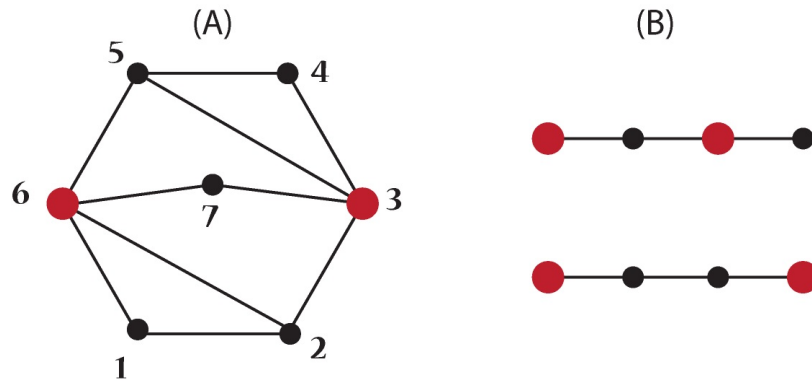


Figure 4.35: The graphs in the example 4.22.

The dominating set problem, i.e. finding out whether the domination number $d(G)$ is less or equal than a certain number k we fix is a classical NP-complete decision problem in computational complexity theory ((GJ79)).

It is therefore believed (if $P \neq NP$) that there is no efficient algorithm that finds a minimum dominating set for a given graph.

We will use this in connection with the perfect codes described in section 4.4.2. A perfect dominating set is equivalent to a perfect error-correcting code, with error correction capability of one error, where the elements of the dominating set are the codewords.

4.5.4 Connection with one-way functions

From what we saw above, the activity about the graphs in which we are using the notion of minimum dominating sets easily refers to the concept of a one-way function.

One-way functions (definition 4.31) is basically a concept that can be presented to kids, but yet it is representing a very difficult challenge and an always developing branch of computer science and mathematics.

As we saw the minimum dominating set problem is hard to solve; on the other hand, it is quite easy to produce a graph of which we know the perfect dominating set, constructing the graph starting from the solution itself.

Example 4.23. We begin by fixing some vertices (figure 4.36 (A)) which will be the final perfect dominating set solution.

From these vertices we connect some other vertices, forming many small “star-shape” graphs, as in figure 4.36 (C).

After doing this we can connect the different vertices (making sure we only add new

edges connecting the vertices we added in the second step) with many other edges (figure 4.36 (C)), obtaining a problem of which we know the solution before-hand, but which a third party will find very hard to solve, if presented as in figure 4.36 (D).

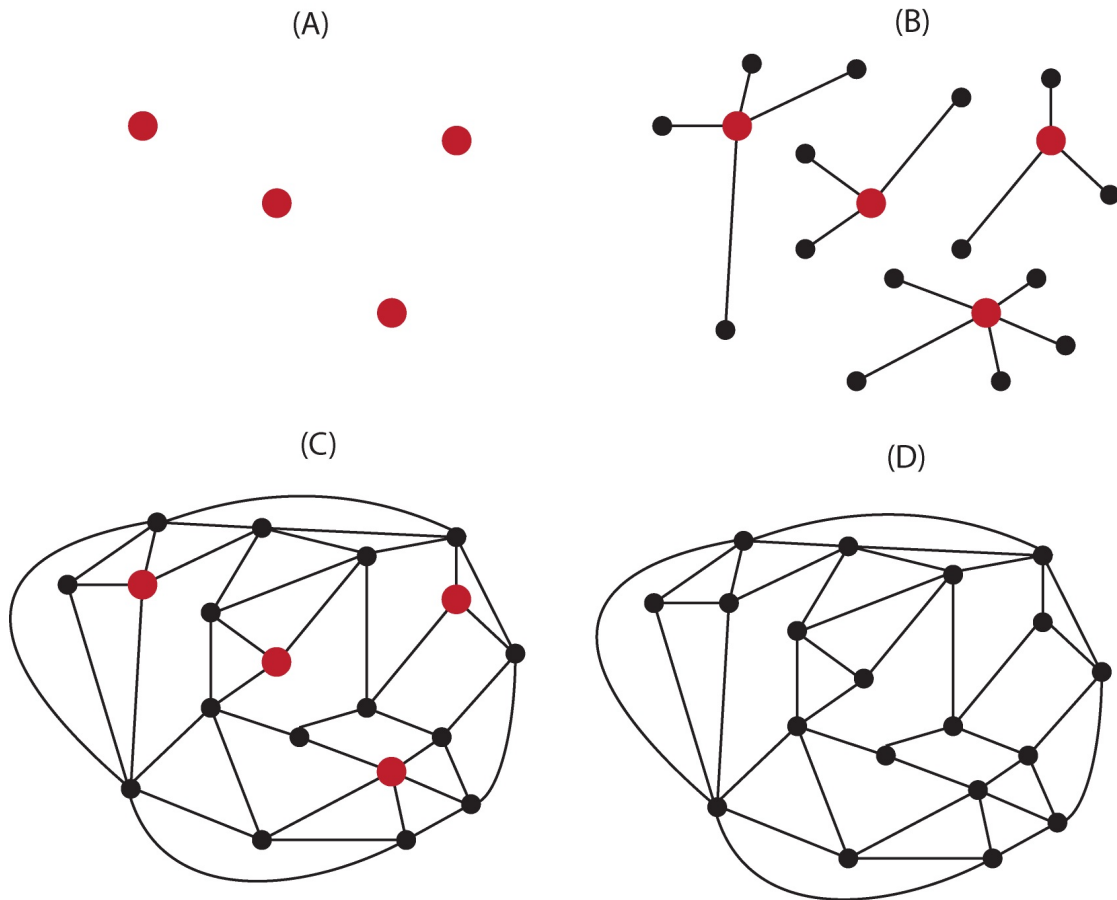


Figure 4.36: Steps in the construction of a graph with a perfect dominating set solution.

5

Research Methodology and Framework

5.1 Design Research

5.1.1 An overview

In 1992, A. Brown (Bro92) and A. Collins (Col92) introduced what, in cognitive and learning sciences, is now well-known as design-based research or design research. The base for this new view over educational approaches came probably from engineering and other applied sciences, where methodologies which intertwined both research and practice were already somehow used, due to the practical nature of the subjects (Cobb et al., 2003 (CCD⁺03)). The first reason to adopt this methodology is to bring the research into the practitioners' world and, vice versa, to have practitioners think about the importance of theory into practice.

In mathematics education, methodologies which used some principles of design research emerged many times (KL12); in fact, like in other scientific disciplines, researchers in mathematics education often are also teachers, or have been practitioners at some point of their career. For those people, the interaction between research and practice is something that cannot be avoided. On the other side, there are still researchers which are somehow “far” from the practice and school environment and therefore the need for a research design who takes into account this problem.

Design research came into use in mathematics education around the year 2000, and is found with many terms and definitions. The terms “design-based research”, “design research”, “educational design research”, “design experiments” are used by many authors. A good description of the new emerging methodology is given in Design Experiments in education research (CCD⁺03).

Many of the issues we are facing in mathematics education are complex problems, not depending by a few factors, but seen as the product of many different factors that are all relevant; we cannot design instructional activities without looking at theoretical models and we cannot do educational research without considering the social context we are working in. Educational research dealt always with two kind of problems (PGC15), understanding people learning and to develop instruction which are effective in practice on learning. The challenge is nowadays to be able to pursue both at the same time. Therefore a need for this new methodology and for more structured definition of it.

A definition of educational design research, as given in *An Introduction to Educational Design Research*, by Plomp and Nieveen (PN07), is that of the systematic study of designing, developing and evaluating educational interventions (such as programs, teaching-learning strategies and materials, products and systems) as solutions for complex problems in educational practice, which also aims at advancing our knowledge about the characteristics of these interventions and the processes of designing and developing them.

5.1.2 Key Features of Design Research

From what stated above, we can feel in some sense the main guidelines lying behind the design research methodology: to make needs of both theorists and practitioners coincide, using pre-designing the teaching interventions which are carefully designed thanks to the knowledge with have in the research world and then testing, evaluating and changing them according to the students' own reactions, practitioners needs and overall the interactions with the social and mathematical environment we are facing in each experiment. There is a double connection between changing and understanding things: we cannot change without understanding and we cannot understand without changing (Plo13). Following Cobb in (CCD⁺03), design experiments are, moreover, conducted to develop theories, not merely to empirically tune "what works". Cobb also talks about creating learning ecologies (complex and comprehending a lot of different aspects) to investigate the possibilities for educational improvement.

There are some really important features which are characteristics of design research:

- focus is not on the instructional product itself but more on the process, which results important for every person/group involved; with process we are mean-

ing both the way of getting new knowledge, but more relevant as the learning of new social practices, in the mathematical reasoning setting but also beyond;

- on the other hand, an interventionist approach, to show how research is set in the real world, documenting successes and failures of different stages and focusing more on the interaction with the learning environment; the setting of our experiments is usually complex and confuse, not to say unique;
- for this scope, the research follows an iterative process, with instructional cycles which are repeated in a dynamic way, with a strong emphasis on evaluating and changing things among these cycles (HLT); importance is given to both developing (always with a view on the research, not only on the practical activity) and revision; therefore fixed procedures and replicability are not the strongest in design research;
- Pragmatic, humble, theories developed during the process of the activity; with that we are meaning that they involve domain-specific learning processes on a local basis and also that they in some sense work for a design purpose in a real setting;
- different participants should all be involved in a successful “multitiered” model.

Cobb, P., Confrey, J., diSessa, A., Lehrer, R., Schauble (CCD⁺03)

There are two main different kinds of design research. We divide them into development studies and validation studies. Development studies function is to design and develop a, research based, intervention and constructing design principles in the process of developing it; validation studies, in an exploratory approach, design learning trajectories to develop and validate (or yet, improve) theories on learning or learning environments.

For the purpose of this thesis, the developmental approach is taken into consideration. The goal is to develop and explore new learning and teaching environments, to verify their effectiveness; to develop somehow new methods, instruments and teaching actions to further improve in the field of problem solving and logical thinking, using a somehow “new” topic as algorithms and cryptography are for primary school students. Doing this the goal is to contribute to the development of new teaching and learning theories, taking into consideration learning processes in the specific situation, with contents and goals clearly defined.

5.1.3 Details of the methodology

Going a little more into details, the methodology is quite well described in many publications; we take Design Experiments in education research (CCD⁺03) and An Introduction to Educational Design Research, Plomp and Nieveen (PN07) as good starting points for describing it.

First of all, one cannot start with designing the intervention as a first step. Before that some research work has to be done. What we want to do is clarifying the theoretical intent and research question of the study. Learning goals are usually not “traditional” results that we want to obtain, in terms of teaching some contents. A general theoretical framework is analyzed but there is also a need to get deeper into the both social and mathematical starting point of the local situation. In formulating the research question we are not only focusing on if a particular intervention works or not but more on the process and characteristics that we want the intervention to have to go towards a solution to our problem (Plo13). It is important also that this analysis of the problems is done together by researchers and teachers (CCD⁺03).

A local instruction theory is first developed in this phase, with the goal of improving it during the experimental and retrospective phase. In this, instructional starting points are considered, as well as the endpoints we want to reach.

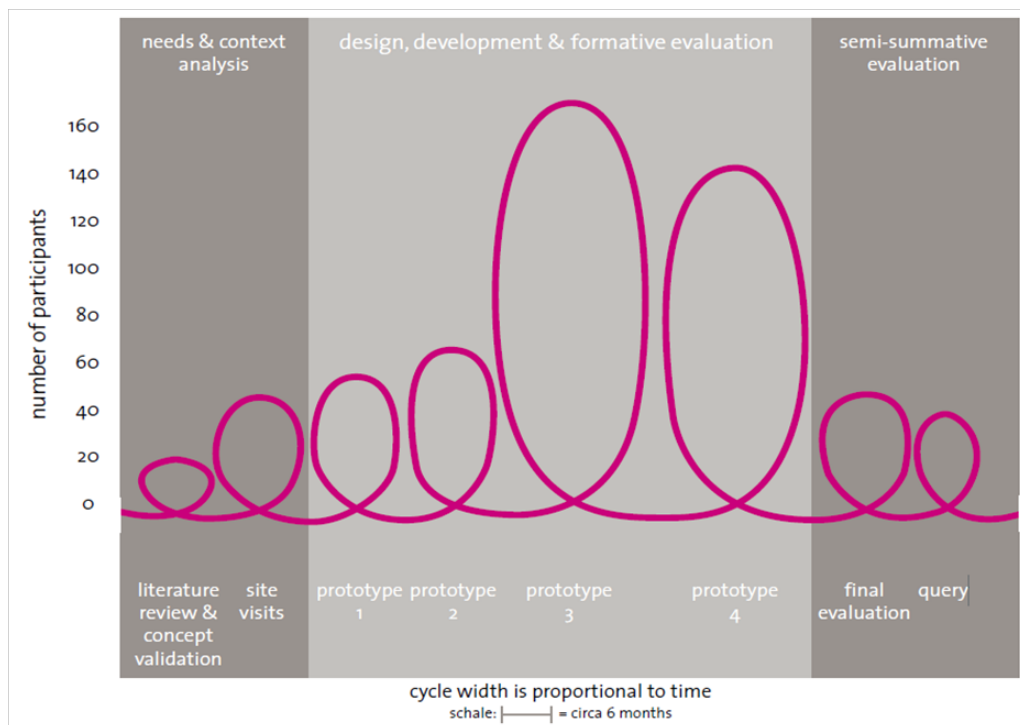


Figure 5.1: Cycles of iteration in DR, following McKenney, 2001

After this preliminary research about needs and context, the experiment phase itself begins with one (or more) prototyping phase. This is usually the first tryouts of designs to implement to see the validity of the design, the first of many cycles of designing and developing. We take as an example the image by McKenney ??, which is representing the research cycles of the whole project. Many different prototyping phases are considered here, with the goal of improving the intervention and to improve the understanding of the process itself. Among these cycles, other "micro-cycles" are considered; micro-design cycles take place within one design experiment, when the researches try and adapt both, the instructional activities and the theory that underpins them (PGC15), which could also last just one lesson in the classroom. The reason for this is a continue need of reviewing and re-designing the instructional trajectories we are planning, to better fit with the local situation needs. Practically speaking, we design the intervention, in cooperation between researchers and practitioners, trying to anticipate the mathematical thinking of the students (we talk about hypothetical learning trajectories in this) then it is tested to see if it correspond with what we were thinking and discussed later on, to reconsider, together with all actors in the process, the next hypothetical learning trajectory; there is a big flexibility in the adapting of the trajectory planned to the real setting. Focus is given therefore also to the follow-up activity of analysis and evaluation, to go to the next step with improvements, related to our goals. The single cycle of experiment will lead to develop a particular kind of reasoning and we will use this, together with others, will help us in re-organizing the past and future activities. In the different experiment phases, many factors are taken into consideration. In the preliminary step, validity, of both content and construct is wanted; we need both something that is related to the problem and which has a logical sense in doing in a classroom. Later on we look for interventions that are useful and usable in their local setting and finally we search for effectiveness of the interventions, i.e. does it help towards our goals? And can it be used by practitioners in a real setting?

5.1.4 Positive aspects

From what written above, we can summarize some of the plus side of choosing design research as methodology of research in mathematics education:

- design research brings research to the "real world" of students and teachers, making the possibilities of creating teaching models which are relevant and

meaningful; possibility of adapting the instructional activities to the local situation, considering all aspect of the so called learning ecology (by developing LIT, local instructional theories, from (Gra04), p.9. “In contrast, individual teachers can use a local instructional theory as a framework of reference for the design of hypothetical learning trajectories that fit the actual needs of their students.”);

- strong relationship among all participants of the study, in particular between researchers and teachers; gives a better chance to investigate the interacting development of several types of subjects (student and teachers, researchers and developers);
- as a result of this, the school and practitioners world can have a better idea and trust more the universities and researchers environment; a point which nowadays is critical for many, i.e. researchers are doing their own work and teachers are not finding it useful and, vice versa, teacher do not share experiences with the researchers who could benefit from it; moreover, many actors in this can play more than a single role; also, researchers can develop more trust into teachers and their professionalism.
- instruction designers could benefit by using knowledge and frameworks from the mathematics education research;

5.1.5 Critics

Design research has been subject, of course, of some critics as well.

- Generalizability of results in design research is an issue; since many factors in the ecology are to be considered, how can this be used in others similar environments? It is really difficult, or unrealistic to reproduce directly the results on a larger scale; we can anyway generalize the design principles or try to widen the local instruction theory to a broader domain;
- The teacher role is really dependent from one person to the other; again it get difficult (we might even say impossible) to reproduce the same experiment in different situations; goals, beliefs and intentions of both teachers and learners are heavily influencing both method and content.
- Long time needed to do the whole process, which isn't a suitable requirement for a profitable research project. On the other side we can argue that it is

possible to see results also during this process in this continuing with reviewing and adapting the instruction;

- ”Because the conceptual frameworks in design studies grow authentically and are grounded in the lived experience of teachers and students, they often appeal to researchers who work in similar contexts. Thus, a mathematics education researcher may find that the concept of socio-mathematical norms attractive because it may guide observation, suggest variables to study, or help make sense of data. Later studies, using different methodologies may extend generalizations across actors, behaviors, and context, and extend the work into the purview of studies of diffusion of innovations (Kel04).

5.2 A constructivist framework

Constructivist frameworks offer nowadays strong contrast to a view of education which is more based on instruction by transmission, or even absorption, where students passively absorb mathematical structures and algorithms invented by others. The milestones for this theory can be found in many articles, information here is from (VG12), (Bru09), (SC12), (Gla75). In the history of education, many important authors met with constructivism and developed it: from Dewey’s opinion on “Experience and Education”, to Piaget’s cognitive schema theory in “cognitive development”, and Montessori’s ideas on “Decentering the Teacher”.

The foundations of constructivism are:

- following the idea of Piaget that mathematical ideas are made, created and not only found somewhere, by the student; it is the result of something that the learner construct in his mind. Knowledge has to be then actively created by the child;
- students can create new mathematical knowledge by reflecting on their physical and mental action (CB90);
- as human being, we have no access to an objective reality which is independent of our way of knowing it; only individual interpretations of our world exist (Sim95);
- students should become autonomous and self-motivated in their mathematical activity (CB90); mathematics is then to be seen as a way of thinking about problems.

Therefore, following (CB90), constructivism gives importance to the development of students' personal mathematical ideas. On the other hand, the constructivist teacher guides the students' attention by offering appropriate tasks for dialogue, directing their learning in the right direction.

If we cite (PK09), an effective classroom, where teachers and students are communicating optimally, is dependent on using constructivist strategies, tools and practices. There are two major types of constructivism in the classroom: (1) Cognitive or individual constructivism depending on Piaget's theory, and (2) Social constructivism depending on Vygotsky's theory. Similarities include inquiry teaching methods and students creating concepts built on existing knowledge that are relevant and meaningful. Differences include language development theory where thinking precedes language for cognitive constructivism and language precedes thinking for the theory of social constructivism. Understanding communicative tools and strategies helps teachers to develop individual learning methods such as, discovery learning, and social interactive activities to develop peer collaboration.

5.2.1 Cooperative learning and the ZPD in a social constructivist environment

Social constructivism emphasizes the importance of culture and context in understanding what occurs in society and constructing knowledge based on this understanding (Derry, (Der99); McMahan, (McM97)). This perspective is closely associated with many contemporary theories, most notably the developmental theories of Vygotsky and Bruner.

As we saw in general constructivism, reality in the human mind is constructed with the human activity, and at the same time we need to see knowledge as a product of the students' mind. In this sense, learning become a (social) process, being something in between the individual and external presences and relation with others.

Cooperative Learning is a process of education that involves the students in group work to reach a common goal. An exercise of learning in group qualifies as CL if the following elements are present:

1. Positive interdependence. The members of the group trust each other to reach the purpose. If someone in the group does not make his/her own part, also the others suffer the consequences of it. Students must be responsible of their personal learning and of learning of the other members of the group.
2. Individual responsibility. All the students in a group have to contribute for his/her own part of the job and for what they have learned.

3. Interaction with peers. Although part of the group work can be divided and individually developed, it is necessary that the components the group works in interactive way, the chain of the reasoning verifying each other, drawing some common conclusions, difficulties that emerge and the feedback furnished. In this way another advantage is also that the students are teaching in some way to each other.
4. Adequate use of the abilities in the group. The students in the group are encouraged and helped to develop the trust in his/her own abilities, leadership, communication capacity, and to take some decisions.
5. Evaluation of the job. The members of a group, periodically appraise the effectiveness of their job and the operation of the work, and they identify the necessary changes to improve its efficiency.

In a constructivist environment, Vygotsky (Vyg87) sees the interaction with peers as the best way to develop new strategies, skills, and at the end, knowledge.

Following this point of view, when a student is in the Zone of Proximal Development (ZPD), if the educator manages to provide the assistance needed, the so called scaffolding, the student will be able to achieve the task he is almost managing to reach.

In the same way, this event can happen when learning among peers. The student is on the point of realizing something and, with the help of one peer student, he manages to realize the last step needed to fulfill the task proposed.

We can conclude by citing Vygotsky himself: “the distance between the actual developmental level as determined by independent problem solving and the level of potential development as determined through problem solving under adult guidance, or in collaboration with more capable peers” (Vygotsky, (Vyg78)).

5.2.2 The theory of didactical situation

One last theory we need to briefly describe is Brousseau’s Theory of Didactical Situation in mathematics (Bro06).

A situation is representing the circumstances in which the subject is set in and the relations that occurs in the environment he is in. In a more constructivist point of view, the situation is the environment of the learner moved by the teacher, who consider this situation the mean to reach something.

In Brousseau’s theory, we want to focus in particular on the a-didactical situations, which are the part of the bigger didactical situation in which the teachers’ intention

is not explicit for the learner; therefore it is the learner himself who has to find a solution to the problem, or how to get closer to one. The student gains responsibility in this situation and has to get more energy into the cognitive process. The learner is therefore the only active subject in the a-didactical situation, making himself some tryouts, errors, failures, interacting with the environmental characteristics and modifying his knowledges.

Knowledge gains space as the object to teach, in a process of finding new knowledge. In this document the teaching methods follow the model of Realistic Mathematics Education (Freudenthal (Str91); Gravemeijer (Gra94)) and Guided Reinvention of mathematics (Brousseau, (Bro06)).

Guided Reinvention of mathematics is based on Hans Freudenthal concept of mathematics as human activity. Education should give students the "guided" opportunity to "re-invent" mathematics by doing it. This means that in mathematics education, the focal point should not be on mathematics as a closed system but on the activity, on the process of mathematization (Freudenthal, (Fre73)). Realistic Mathematics Education (RME) is an instructional design theory which centers around the view of mathematics as a human activity (Freudenthal); "The idea is to allow learners to come to regard the knowledge that they acquire as their own private knowledge, knowledge for which they themselves are responsible." (Gra94). The main goal is to develop a local (i.e. domain-specific) instructional theory (LIT) that will allow students to "[invent] the mathematics themselves" (LZ08). This need two steps: Step 1, in which "students are engaged in activities designed to invoke powerful informal understandings" (WL08); Step 2, in which "students are engaged in activities designed to support reflection on these informal notions in order to promote the development of formal concepts" (WL08).

5.3 A little about task design in education

The discussion around the design of mathematical tasks has always been wide. The first question is really to understand what a “good” mathematical task is. Or, is there a bad and a good task? This will pretty much depend on what the goal of the task design, and of the educators, is. Then we can focus on, how different tasks can be designed to reach the kind of mathematical understanding we are looking for.

According to Schoenfeld (Sch82), one of the characteristics of a good problem is that it should serve as an introduction to important mathematical ideas. By definition, any use of mathematical tasks should access some mathematical idea. Somewhat more challenging is the use of mathematical tasks to access specific mathematical ideas (important ideas) (LCZ07).

Following Liljedahl et al., in (LCZ07), the process of designing mathematical tasks is a recursive process that applies as well to the creation of entirely new tasks as it does to the adaptation (or refinement) of already existing tasks. Design research and our research goes in this direction of refinements, and we did it in many of the tasks we will be presenting in the next chapter. This recursive process consists of four stages: predictive analysis, trial, reflective analysis, and adjustment.

1. Predictive analysis might be done with our experience, our mathematical knowledge and also comparing with similar tasks we are starting from, and already doing some adjustments from them. This will be shown in the first phase of the research presented 6.0.1;
2. Trial is the main teaching experiment (described in various stages, in all Chapter 6);
3. Reflective analysis for both the mathematical and pedagogical affordances of the task; this is when we may see either new affordances or realize the inaccessibility of the predicted affordances. Either one of these cases will require a rethinking of the tasks and/or its implementation (LCZ07); This is discussed in Chapter 7;
4. Adjustment phase is the rethinking of the previous stages. The adjustment phase, in particular, present a lot of thinking back about the tasks and the goals, and then the process start over. There is a lot about this again in Ch. 7.

Also, as said, we need to have clear in mind what our goals are (usage-goal framework); this provides the foundation upon which to design a good mathematical task

and how to later refine the design.

5.4 Is design research appropriate for my research?

5.4.1 Design Research and a Socio-constructivist approach

Design research connects well with the theory of Realistic Mathematics Education (RME) - The principles (guided-reinvention, didactical phenomenology, and emerging modeling) of Realistic Mathematics Education (Freudenthal (Str91); Gravemeijer (Gra94)) can provide ideas valuable in designing research, in the sense that they fit well to the design research requirements; also, with the work of Freudenthal and in some sense that of Guy Brousseau:

“The more the teacher gives in to her demands and reveals whatever the student wants, and the more she tells her precisely what she must do, the more she risks losing her chance of obtaining the learning which she is in fact aiming for.” (Bro97) p. 41.

All of the above helps motivating the use of constructivism (in our case socio-constructivism) as the background theory for our study. Human knowledge and experience are characterized by an active participation by the individual; this happens inside the socio-cultural context we are acting. In this theory, learning is considered as a construction process where we get to meaning shared with other and that are not external from us. Constructivism is therefore opposed to teaching by transmission.

This works quite well with the purpose of the research, which is not to create more content knowledge but is focusing on the process and on creating reasoning abilities in students.

5.4.2 Design Research towards the design of innovative tasks

As in (MKB03), design research seems appropriate when new research suggests a powerful innovation. For sure in the Italian school, there is not much older material about the algorithms, cryptography and graph theory topics and teachers feel a need for it. As shown in the problem overview chapter, as a result of a survey among school teachers, there is a lack of knowledge about this subject (total in primary school and still significant in the secondary school). The problem is quite open also about HOW to present these topics, and, while we have a feeling about what could be the desirable results, it is still quite open about that as well.

Analysing our preliminary survey, as we saw in the dedicated chapter, the starting point is quite promising.

Our goal and endpoint is to show how this kind of activities can help in improving the

students' problem solving capacities, logic ability and in bringing a more modern type of teaching and learning into the school system. It can further improve the computational thinking, seen as "algorithmic thinking", capability of students, a thing which is very much stressed by the new teaching guidelines all around the World. Finding a proper way of evaluation of "computational thinking" is an open problem, as it is not easy to evaluate with simple tests or surveys.

5.5 Methods of video selection and analysis

Analysis of the results is video-based, qualitative and fine-grained; both group activities and classroom discussion are recorded and we also have many of the transcripts, together with field notes, student's sheets, and interviews as other sources of evidence. As already said, focus is put on students' learning and thinking, in reaction to the different tasks proposed.

Following Zacks Tverski theory (ZT01), data is represented by events selected from the video recordings available. We used an inductive approach in video selecting, beginning with viewing the corpus in its entirety and focus on details later on. Indexing and summaries of videos, plus a content log, help in this process. Going on with the analysis some events which were particularly relevant were isolated. Although there are some recognizable recurring situation and choice of words we coded, our focus is more on a "play-by-play" description of these chosen events. We are, with this approach, analyzing selected episodes focusing on the same happening and constantly revising our finding and new hypothesis, as in Cobb and Whitenack (CW96) with the involvement of "constantly reconciling provisional analytic categories with subsequent data and newly formulated categories".

6

Teaching Experiment

6.0 Pilot teaching experiment

The first implementation step was a teaching activity, as a pilot evaluation, in 6th grade classrooms, chosen as a first preliminary tryout for the activity we plan to use. 6th grade is the central step in our project, lying between primary school and middle school and was therefore suitable for this preliminary experiment.

During the past school year many teachers showed interest in the projects that were made and were willing to have us go through these sequences of tasks with their classes. We developed many different tasks and activities which were meant to suit a particular age group.

The agreement reached with I.C. Primiero Institute and their middle school was to implement activities for 15 lessons of 1 hour, and to repeat these for 2 cycles, to better redefine some of the activities, with improvements and changes, when needed. The period for this has been the fall semester 2015 and the spring semester 2016.

6.0.1 Description

We had an initial plan for the whole cycle of 15 lessons, which was well followed, especially in the first cycle (fall semester 2015). For all the list, the activities will be described in details in the next sections, in their final form. When major changes did occur, we will present them, pointing out why and how they were done.

1. September 21st - introduction to binary codes

Following activity 1 of Computer Science unplugged 2015 book (BWF98), *Count the Dots - Binary numbers* and the more popular, at least among mathematicians, *Guess the number* trick.

2. September 28th - secret codes and private key cryptography

The basic concepts of Cryptography were presented, making the students create their own code and getting them familiar with ciphering and de-ciphering with Caesar's cipher.

3. October 5th - searching algorithms

Some common examples of searching algorithms, with a discussion with the students going from phone books to dictionaries; a practical example using a line bottle caps with numbers, previously ordered, hidden under them and the goal to find a given number and discovering a proper strategy to do it in a relatively fast way; finally, activity 6 of Computer Science unplugged 2015 book, *Battleships - Searching Algorithms*.

4. October 12th - sorting algorithms and sorting networks

Again, we used activity 7 of Computer Science unplugged 2015 book, *Lightest and Heaviest - Sorting Algorithms*, followed by a re-interpretation of the sorting network game, using a giant mat which was created from the optimal sorting network ordering 6 quantities.

5. October 19th - deadlock

Activity 10 of Computer Science unplugged 2015 book, *The Orange Game - Routing and Deadlock in Networks*, and a variant of it we came up with, using chairs and name tags. These activities have a very big impact on cooperative learning and will have a dedicated section later in the chapter.

6. October 26th - pixels and computer graphics

Slightly changing the subject, we discussed the functioning of monitors and image data transmission in computers; a brief activity with some sort of pixel art was developed, making the student think about how to make big quantity of data easy and fast to transmit.

7. November 9th - the Traveling Salesman Problem and Muddy City

Going more towards graph theory with this lesson, we presented students with the classical Traveling Salesman Problem, connecting four cities and then activity 9 of Computer Science unplugged 2015 book, *The Muddy City - Minimal Spanning Trees*.

8. November 16th - Euler's graphs and circuits - map coloring

Some more "classical" graph theory problems were presented, from Euler's

Koeningsberg bridges problem to some pencil-and-paper drawing to do, discovering some graph theory and Euler's circuits properties.

9. **November 23rd - minimum dominating sets and private key cryptography**

We introduced the concept (without giving a formal definition) of minimum dominating set and a feeling to the students of how difficult it can be to find one. We used the problem of roads and fire stations that will be presented in the next sections, also to introduce how to use minimum dominating set to create a public key cryptography algorithm.

10. **November 30th - algorithms and giving instructions - the language of computers**

After finishing the difficult topic from the previous lesson, we gave an introduction about algorithms inside a computer, recollecting all the important information we learned in the past lessons about them.

11. **December 14th - introduction to coding and programming**

Using the website code.org. This lesson was not in our original plans, but proved to be essential to give a first introduction to programming languages, before using Scratch.

12. **December 21st - Scratch pt.1**

We decided to use two lessons to practice something with Scratch. the main idea was to recall some of the concepts learned in the previous lessons and make use of those in practice while programming. The first lesson was used to get acquainted with the software and with creating a simple game with a character moving inside a labyrinth.

13. **January 11th Scratch pt.2**

Going on with our previous activity, we did some exercises involving iteration cycles and selection options.

14. **January 18th - algorithms and giving instructions**

After trying those on code.org and Scratch programming language, a reflection on the iteration and selection processes was due. We discussed with the class and then tried a game where they had to give instruction to each other while stacking cups in different compositions. Checking mistakes and trying to correct them also proved to be of great interest in this activity.

15. **January 25th - variables in programming**

Again using code.org activities, we focused on the concept of variable.

6.0.2 Development and preliminary results

The description in Section 6.0.1 was that of the first of the two cycles of activities tried out in I.C. Primiero middle school. Without repeating all the activities involved, we want to highlight the major changes done to the lesson plans, especially to show what we think should change and why.

- We made lesson number 2 longer, to allow the students to get more familiar with basic concepts of cryptography, especially to make it easier for lesson number 9 to be fruitful, since we noticed that probably the student need a bit longer time to get acquainted with some difficult concepts in this last lesson.
- We developed further the deadlock game (will be presented in details in 6.2), adapting it and making it possible to play only with chairs in a circle, instead of having to search for gym props or colored spheres.
- In general, we did try to make the activities from the Computer Science unplugged 2015 book a bit more adherent to the reality and socio-economic conditions we are in, trying to make examples that seemed to be more “real” to the students and therefore slightly changing the setting of some problems (e.g. the Traveling Salesman Problem) got a local setting with the region main cities and a local vendor having to go around, instead of the old fashioned traveler we presented in the first attempt.
- The map coloring activity was really appreciated by students, who could also conjecture more mathematics than we thought about it. Therefore we developed some more examples to make the mathematical concepts accessible for the majority of the students. The final version will be presented in details in 7.1.
- We spent a little bit more time (three lessons instead of two) on Scratch programming language, having already in mind a comparison between the use of the software and a more “unplugged” approach (see 6.3).
- Wanting to add some contents, as stated above, we had to cut some parts of our plan. The decision, even if suffered because the topic needs particular attention, has been to cut lesson 14 from the cycle.

In general, the first cycle of lessons was really meant to try everything that we thought would be in some sense useful, interesting or even just funny for the students. We had of course some goals in mind, but at first we just wanted to experiment this whole lot of new ideas and activities we had. Already with the second iteration of the same program, we could focus on some of the goals, change a little bit our plans and develop to activity to accomplish these goals.

the cooperation with the teacher(s) involved were optimal, and they both were fully available to let us try out new activities with their students and wanted to participate in the decision and development process, being of much help in connecting us with the classroom reality in that particular environment.

We present now all the different tasks that were developed, divided in four main different sections, considering the topics, the age of the students and the goals we wanted to reach. For each of the sections, results obtained will be analyzed in details in Chapter 7.

6.1 Graph Theory and Cryptography

As previously written, in addition to the activities which will be described in details, we used some well known cryptographic activities, such as:

- historic cryptographic examples, as described in section 4.3.1; the scytale was showed, with some groups we built a Caesar cipher wheel to better understand how it worked, we talked about Vigenere and so on;
- problems related to how to solve the key exchange problem and an example of the double lock encryption protocol, using a suitcase and two locks and making students try how to solve the problem;
- one-way functions, using, for example, a telephone book (where it is easy to find a number given the name, but not to find the corresponding name to a number given).

Important references and inspiration for the whole activity were (Fel91), (FK93), (BWF98) and (FK00).

6.1.1 Map coloring

The map coloring activity was initially proposed to two different groups, a 6th grade and a group of 4th and 5th graders in a primary school, during some afternoon extra classes. It was inserted in a longer project on algorithms and some basic cryptography, but could have been treated as well as a self-standing activity. It was, in the second and longer phase of this teaching experiment, tried out, with different level of challenge, from 1st grade to 6th.

The activity sheets which were used can be seen in the next pages. For using them in school during some lessons, original sheets are in Italian, the main parts will anyway be translated for this document.

Description of the activity

The first map presented is a first problem to introduce map coloring. The story described to the students is the story of the poor map colorer who wants to color his map, with no region of the same color bordering each other and using as few colors as possible. The students had no difficulty in understanding the delivery but the strategy to solve it was not immediate. The answer is three colors.

Once the students get the goal and some ideas to solve the problem, we can present them with the second map, which was the U.S.A. political map. How many colors do we, at least, have to use this time? The problem to solve is the same, though the regions to color are more, making it more difficult. Four colors are required to color this map.

Let us turn to the second page of the activity. Suppose now that an abstract painter wants to try to color his painting with, again, as few colors as possible. It is explained that regions touching only in one corner are not considered as bordering each other. This kind of “maps”, presented after the more classical geographic maps, are interesting to color, since they are two-colorable, as we saw in the previous chapter. Students were also asked, after finding the solution to this particular problem, to produce other examples of maps which are colorable with just two different colors. The last section of the second page is a “bonus” question, mainly for students which are fast in getting the solution of the problem and looking for something more engaging, after solving the other parts. It turns out not immediate to find out and to prove, that four colors are required for all the examples. This will lead to a possible interesting discussion about the four colors theorem: are four colors always enough or can we find a counterexample? For example, the students could try the impossible challenge (see Theorem 4.2.5) to produce a map for which four colors would not be enough, providing an interesting and intriguing activity.

In the third worksheet, the activities proposed to younger children are shown. What stated above is still valid, if adapted to the age of the pupils involved. Also, we noticed after some tryouts that also the “dot coloring” activity sheet is worth to try with younger students, in some cases it proved even more efficient than the standard map coloring one.

The worksheet presented for the dot coloring activity is also reported in the following pages.

Goals and mathematics behind the activity

- *Logical reasoning and problem solving* are topics which are in focus within this activity. Students were asked to think about the problem and understand the goal and find a way to reach it.
- *Algorithms and computational complexity* is the main goal of this particular activity.
Some of the examples proposed are quite easy to solve, e.g. the two-colorable maps, while some others are quite complicated and will require quite a lot of time to solve; this leads the students to a first approach to a P vs. NP way of thinking, discerning between easy (for a mathematician, solvable in polynomial time) and more difficult (related to NP problems) tasks.
- An *approach to formulating conjectures* is the first step to proof-oriented mathematics. Students were asked to think about the problem and search for a way to reach a general solution.

allegato scheda 1-a

allegato scheda 1-b

allegato scheda 2 easy

allegato 2 b

6.1.2 Graphs and dominating sets

With a similar goal to the map coloring activity sheets, we introduced another set of exercises on graph theory.

One or both the activities were proposed to many different groups, from 4th grade through middle school, up to 7th and 8th grade. It was inserted in a longer project on algorithms and some basic cryptography, but could, again, have been treated as well as a self-standing activity.

The activity sheets which were used can be seen in the next pages. For using them in school during some lessons, original sheets are in Italian, the main parts will anyway be translated for this document.

Description of the activity

First problem we show is an introduction to this activity.

Kids got a Tourist map of Trento city and a red path highlighted on it. The task is to find out a way to go along all of the path, without using the same route twice, starting and finishing at the train station (“stazione del treno”, south-west in the map you see in the next pages). Two different maps for the same problem were proposed. The first one is possible, and not too hard to solve, while the second will not work with the rules we set.

The first worksheet deals with the problem of Euler paths (cfr. section 4.5.1).

The question is whether it is possible to draw the figures or not, without lifting the pencil from the paper and without passing the same line twice. The solutions are that 1, 2, 4, 5, 7 and 10 are possible and the others are not. It further asks, once found which are possible and which ones are not, to try to find a reason.

The last problem of this sheet is Koenigsberg’s bridges problem, as presented in ??: is it possible to visit both shores and both islands, crossing every bridge exactly once? With which path? We now know it is not possible to find such a path, from easy graph theory reasoning.

In the second sheet the problem presented deals with the minimum dominating set problem (cfr. section 4.5.3).

The problem is presented to students as a practical problem: we have a number of different cities (represented by a building in the activity sheet) and we want to build a set of fire-stations to protect these cities. The first think to consider is that firemen have to get in action quickly, so we want every city to be at distance at most 1 from a fire-station (in other words distance one means that we can reach each city in at

most one black line); the other point is that we want anyway to save resources, so we want to built the minimum number of fire-stations, with the previous property. As we saw, the problem of finding the minimum dominating set is connected to NP decision problems.

allegato scheda 3

allegato scheda 4a

allegato scheda 4b

Goals and mathematics behind the activity

- *Logical reasoning and problem solving* are topics which are in focus within this activity. Students were asked to think about the problem and understand the goal and find a way to reach it.
- *Algorithms and computational complexity* is again the main goal of this particular activity.

The Euler path problems are easy to try and solve, while the minimum dominating set is quite hard to find even for a relatively small graph. Again we are confronting problems for which we know an easy solution (and even if we don't know it we can easily figure it out) versus a problem for which it is very difficult to find a resolute algorithm.

- An *approach to formulating conjectures* is the first step to proof-oriented mathematics. Students were asked to think about the problem and search for a way to reach a general solution.

6.1.3 Binary numbers and Error Correcting Codes

This section is about some activities to introduce students to binary numbers and codes.

It was presented to many classes, from 3rd grade up to 7th, proposing different and suitable activities for the age group we were teaching to. We add here an activity on error correcting codes which is more suitable for pupils in grades 7th through 12th. It was inserted in a longer project on algorithms and some basic cryptography and has the goal to introduce young people to codes and data transmission.

Description of the activity

The first activity is a game; in many occasions this was the first activity proposed to the students during the projects.

The teacher asks a student to think about a number between 1 and 30, without saying it out loud; then five big cards are shown to the student and she/he has to tell whether the number she/he is thinking of is written on the card or not.

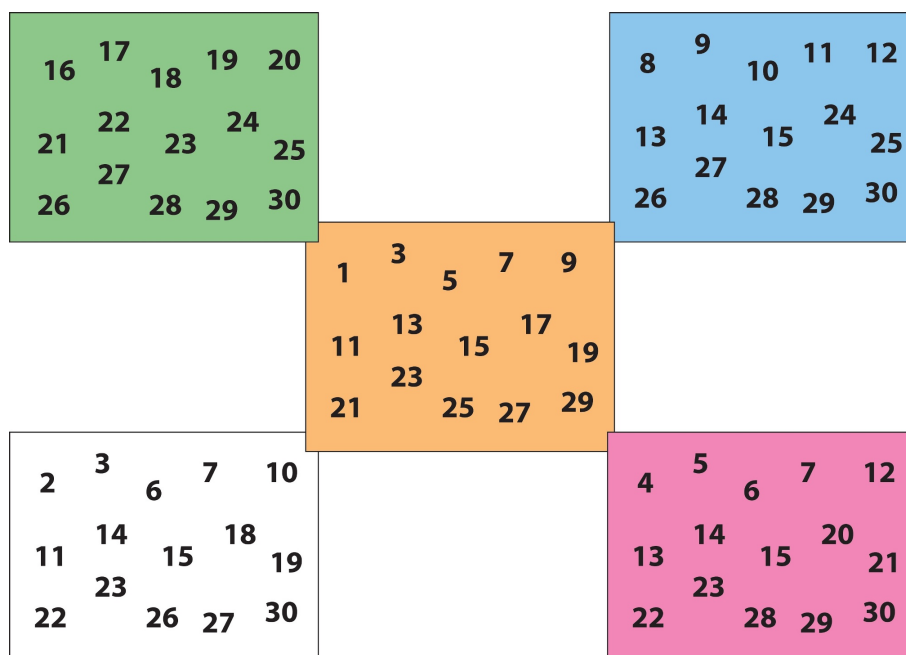


Figure 6.1: The cards used for the game described, we have them printed in bigger size.

Figure 6.1 shows how the cards are composed. After receiving the 5 answers (yes or no) the teacher is able to “guess” the right number the student is thinking about. From how the cards are printed, we know that what we are doing is reading the number in binary code. For example, looking at the figure 6.1, if the student is

thinking about number 19, what we get is:

YES to the green card

NO to the blue card

NO to the purple card

YES to the white card

YES to the orange card

This can be translated into binary numbers as 1 0 0 1 1, which is equal to $1+2+16 = 19$ reading the number in binary code.

Leaving the solution of the previous game for some minutes, we introduce binary number to students, especially in the lower grades, with an engaging activity. Five or six students are needed, and each will get a card, as represented in figure 6.2.

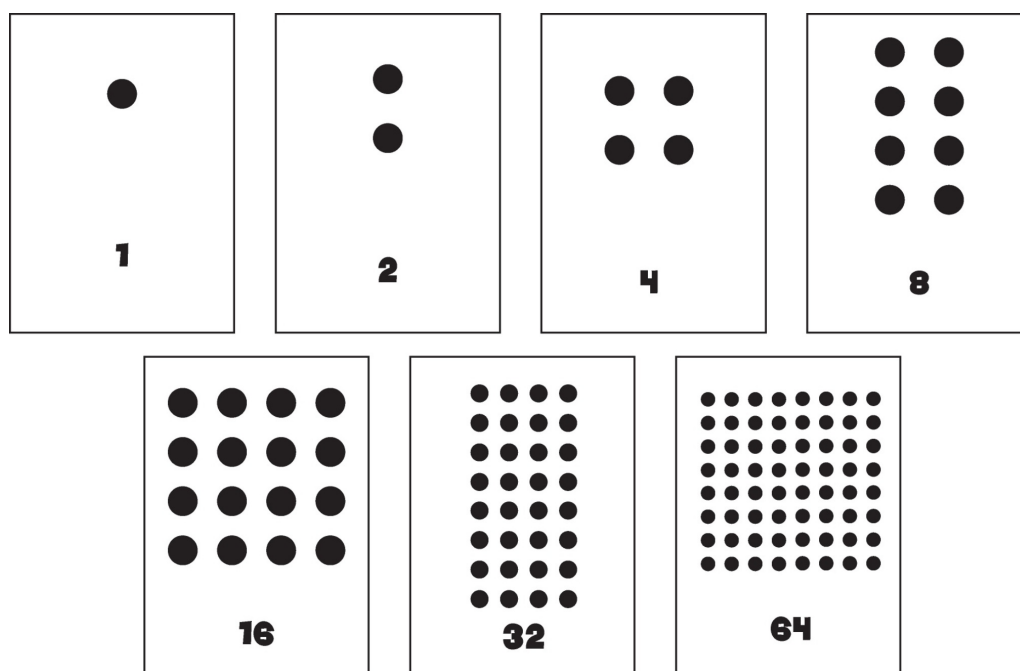


Figure 6.2: The cards used for counting with binary numbers; they were black on the other side. Usually just the first five or six were used.

Each one of the chosen students is now facing the rest of the class and holding a card; if the card is visible on the black side it will represent zero while if it is on the drawn side it will represent the number everybody can see. In other words, the students has to “turn on and off their cards” to form the right numbers.

The challenge for the students is now forming some number the teacher or some other student is calling out, while the others can help by looking at the cards, thinking at the solution and helping their classmates. Counting is very interesting in this way, as it gives some ideas on how binary numbers behave.

Once we introduce binary numbers and give an idea of codes and cryptography, we can also talk about error correction. This last activity in this section has the goal to introduce an idea of error detection and correction, again in an engaging way with a game.

We need for this activity some (at least 36) magnets which are black on one side and colored on the other. With the same purpose, to-sided cards, white on one side and black on the other will also work. Now we explain how the game works using the cards, in a similar way it works using the magnets. We ask a demonstrating student to lay out the cards in a 5×5 square, with a random mixture of sides showing. Once the student is done, we “casually” add an extra column and an extra row, saying that it is just to make it harder. The extra cards are in reality choosen to check the parity of each row and column, so there has to be an even number of black cards in each column and row.

Ask a student to flip over one of the cards while the teacher is not looking, once allowed to look at the square of cards the teacher will be able to identify the card which has been changed and do the trick.

After trying it some times, discuss with the student how the trick works.

Goals and mathematics behind the activity

- *Binary numbers and arithmetic*, which are not always part of the schools' programs.
- A different way of introducing *codes and data transmission*.
- *Error correction* is introduced using an engaging way the ensures the students attention.

6.1.4 Public Cryptography from dominating sets

The following activity is really interesting as it connects most of the topics handled in this thesis.

For this reason, it was presented to almost every classes I met during my experimentation, to students from 4th grade up to 11th and 12th grade. It was inserted in a longer project on algorithms and some basic cryptography and has the goal to introduce young people to public-key cryptography.

Description of the activity

The goal for student A is sending a message to student B in such a way that the conditions of a public key cryptosystem are satisfied, i.e. secrecy is kept, no previous key exchange is needed and only the receiver (the holder of his private key) can decrypt the message. The activity is best done if “student A” and “student B” are groups of students, to better check the computations.

The encryption process works as follows:

- student A receives a graph which will be used for encryption of the message; an example is showed in Fig. 6.3 (A);

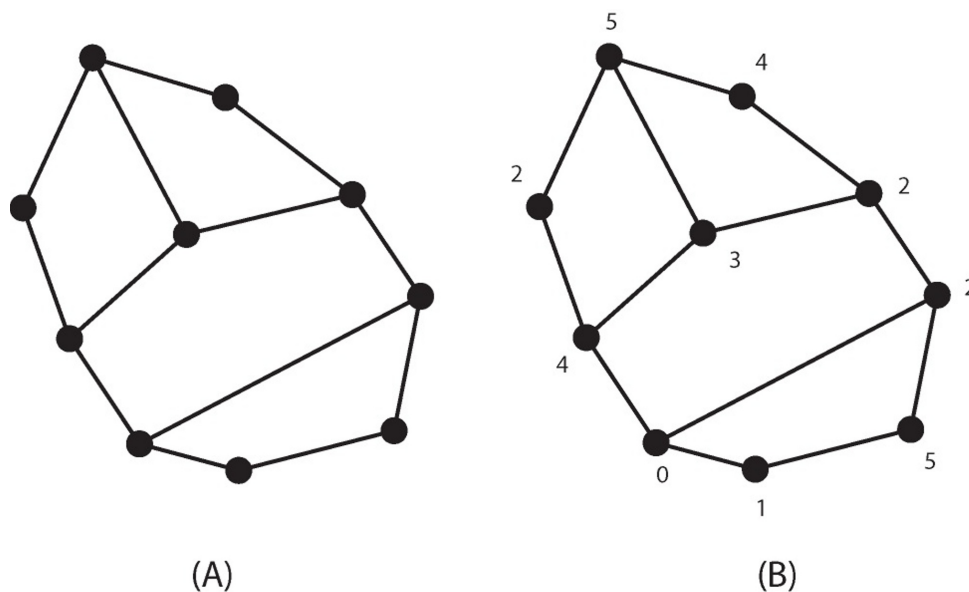


Figure 6.3: Encryption of a numerical message, part 1.

- student A wants to send a message to student B, he needs to encrypt it. For our example, the message we are sending is a natural number. So student A chooses a number that is going to be transmitted and write one number

for each vertex of the graph (see Fig. 6.3 (B)). The sum of these numbers is corresponding to the number to be transmitted. In the example the message is $5 + 4 + 2 + 2 + 5 + 1 + 0 + 4 + 2 + 3 = 28$;

- the next step is, for every vertex, adding up the numbers corresponding to the vertex itself, plus all the other vertices of the graph which are at distance at most 1 from the chosen vertex (we obtain the numbers in red in the example in Fig. 6.4 (A));

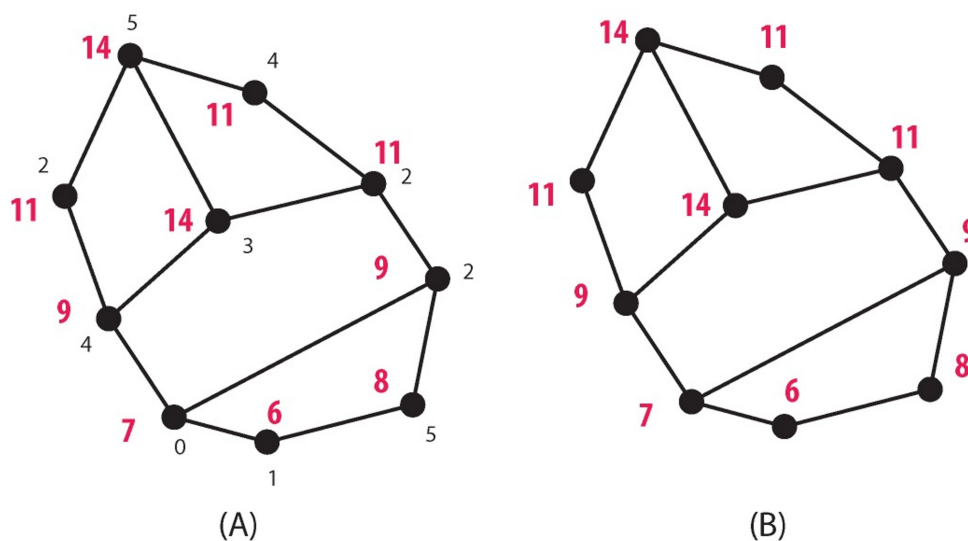


Figure 6.4: Encryption of a numerical message, part 2.

- the message is now ready to be sent, transmitting only the red number and not the small ones we started from.

After encrypting the message, the students can exchange their sheets with each other and try to see if they can figure out the others' messages.

To decrypt the message, we need a special private key, which only student B has. With this (see a solution for our example in Fig. 6.5) it is possible to read the message by just adding up the numbers corresponding to the red dots. So, in the example, we read $11 + 11 + 6$ which is indeed 28 as the message which was sent.

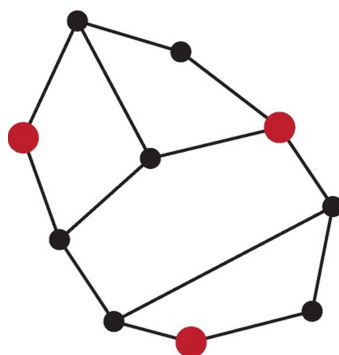


Figure 6.5: A decryption key, which is a perfect dominating set for the graph.

Summarising, student B managed to receive a message from student A, without need of key exchange and just by sending a public key (which is in our case the graph), which everyone can see. A possible intruder knows the public key and might even know the encryption process but still cannot access the information in the message.

It is worth noticing how this activity explains the concept of public-key cryptography without using, as in most cryptographic examples, notions of modular algebra which would require a lot of time and effort from the students.

This made it possible to present this concept to student as young as in primary schools, with good results in term of understanding of the relatively complex concept. After explaining that there is no known algorithm to solve the firestations problem (as in section 6.1.2), we showed the students how it is possible to “work backwards”, starting with a set of vertices which is going to become an efficient solution and generating a graph of the towns. Confront example 4.23 in the previous section for the mathematical details.

From these remarks, it follows that it is easy to create a public key from a key we already have which can be kept private.

From this example it is also possible to talk about the idea of a one-way function with the students which are now quite familiar, even if in a very informal way, to computational complexity.

Goals and mathematics behind the activity

- *Public-key cryptography and information security* presented in a way that is accessible even for young kids. Focus is on the actual problem of how to send a message without an intruder reading it and many problems connected to it.

- *Algorithms and computational complexity*, with the use of one-way functions, serves as an explanation of how this process works.

6.2 Computational Complexity of algorithms and Cooperative Learning

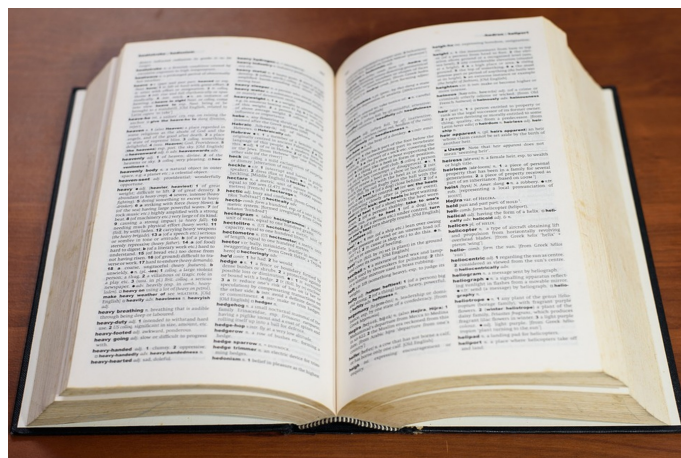
The tasks we are going to describe in this module are part of the sequences of tasks that were proposed to various schools and age groups during the 2015/2016 and 2016/2017 school years. In a design research paradigm (Plomp and Nieveen, 2007), the activities were tried out many times, always with an a priori analysis together with the teachers and with a retrospective look after each lesson. Students from two different schools were involved: 2 classes in 3rd grade (38 students), 5 classes in 4th grade (85 students), 4 classes in 5th grade (68 students), 9 classes in 6th grade (180 students) and 2 classes in 7th and 8th grade (41 students). The sequence of tasks we are focusing on is a particular sequence developed for 4th, 5th and 6th grade about sorting algorithms.

6.2.1 Searching Algorithms in Primary School

The main problem here is to search something inside a list of objects.

As we saw in the preliminary experiment, we presented the students some common examples of searching algorithms, with a discussion with the students going from phone books to dictionaries; a practical example using a line bottle caps with numbers, previously ordered, hidden under them and the goal to find a given number and discovering a proper strategy to do it in a relatively fast way.

The example of the dictionary is still pretty clear in the young students' minds. they learn in school how to find a word out of the list and already, unconsciously, developed some strategy to get faster in the task.



The phonebook gets one step further; we can use it to introduce the idea of one-way function, i.e. it is easy to find a number of a person whose name I know, but it is very difficult, almost impossible, in the real-world phone book, to find the name of an unknown number that called me.

Making a line of bottle caps with numbers, which were previously ordered, and trying to make them find the right number by picking the caps one at a time is even more interesting. It gets the student to develop an attitude towards getting to the shortest way. We can in fact easily count the number of rounds each strategy need to get to a positive end (finding a precise number).

All of these tasks are thought with the main goal of having kids realize that ordering things is very important, but, as we saw, it gets some really and insightful results in the mathematics area, making some important algorithms arise, from random search to a more complex binary search to get efficient.

This unit demonstrates two different search methods: sequential (sometimes called linear) search, and binary search. We will see that binary search is remarkably fast, and although there are other search algorithms that are can do even better (such as the hash table, which is covered in the unit on Data Structures for search algorithms), the step-up from sequential search to binary search demonstrates how much there is to gain there is to be made by applying the right algorithm to the job.

6.2.2 **Sorting Algorithms for 4th, 5th and 6th grade students**

The content goal of this sequence of tasks is to teach something about sorting algorithms, their mode of operation, their speed and the fact that there might be different algorithms leading to a solution. In some activities basic computational complexity is emerging, even if not formally introduced to children.

We start with some “easy to perform” tasks to better understand the problem we are facing with the students. As we said earlier, students are let free to try out on their own and find a way to a solution. We want them to re-discover the algorithm (or algorithms) which is good to solve a predetermined problem.

- Task 1 is done in groups. The students are divided into group of 8 to 10 people and they are arranged in a row. They are given the task to reorder themselves following a order which is given by the teacher/researcher. Also, they should try to be as fast as they can in doing this, usually being challenge to be faster than the other group(s).
- Task 2 is a more classic problem of mathematics and problem solving. Students, in smaller groups, are given a balance scale and 8 boxes. These boxes

6. Teaching Experiment

contain different weights, which cannot be seen from outside. The goal is to put the boxes in the right order, from the heaviest to the lightest, using only the balance scale (i.e. comparing only to boxes at a time) and trying to do it using the smaller number of comparison. Many different possibilities emerge from solution to this problem and they are later compared with the students.



Figure 6.6: Trying out the balance scale problem in practice.

- Task 3 is based on Computer Science Unplugged sorting networks activity. Students have a big carpet with a sorting network ordering 6 different quantities. The goal is that of making this sorting network work, i.e., starting in random order, they should be able to produce a result which is ordering themselves using the network. They are given a paper with a number and at each step they meet a partner. At each step they compare their number with the partner's and decide who should go left and who should go right in following the network path (left and right are predetermined by the group, and should always have bigger numbers on the same side and smaller number on the other).



Figure 6.7: Students involved in the ordering process involving the sorting network.

At the end a discussion follows. With some groups, we could even go further, making them think about how to draw their own simpler sorting network with 4 object; thinking about how it works (does it work, for example, backwards?) and other properties it might have.

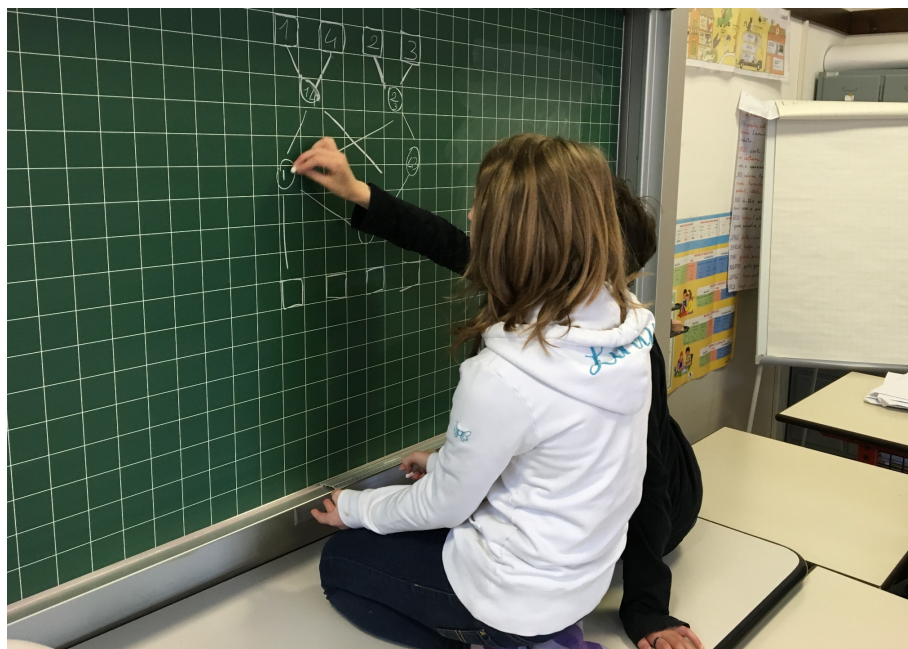


Figure 6.8: Discussing the sorting network problem.

- Task 4 is also based on a CSU activity. It is a slightly modified version of “the Orange Game”. Students are divided in groups of about 6 people and they are sitting in a circle. They are given 2 colored balls each (except for one in each group who only gets 1) of different colors; there are in total 2 balls for each of six colors. Students are also assigned one of the six colors. The goal is to pass the colored balls around, until each student gets in their hands the ones labeled with their assigned color.



Figure 6.9: The Colored Ball Game activity with colored spheres.

Two main rules: only one ball may be held in each hand and a ball can only be passed to an empty hand of an immediate neighbor in the circle (a student can pass either of the two they have to their neighbor). Kids are let free to try out their own strategies and “algorithms” to solve the game as fast as they can.

6.2.3 Computational complexity in graph theory

As we already saw, map coloring is a great example to introduce the topic of computational complexity to students.

Some of the examples proposed are quite easy to solve, e.g. the two-colorable maps, while some others are quite complicated and will require quite a lot of

time to solve; this leads the students to a first approach to a P vs. NP way of thinking, discerning between easy (for a mathematician, solvable in polynomial time) and more difficult (related to NP problems) tasks.

Also, the public cryptography activity on dominating sets is based on the different complexity of this particular one-way functions; functions that are really easy to solve in one sense, but very difficult in the other to get to the inverse.

6.2.3.1 Roads in the city and the Traveling Salesman problem

The setting of paving roads in a city can be presented in many different problems. In our scenario, we have a city with many roads that are dirty and muddy, and really needs to get paved. there are ten different neighborhoods in the city and they are connected by different roads of different length. Of course, otherwise it would get too easy, the people living in the city wish to have all the roads paved, but there is not enough money to do so. Somehow they need to reach an agreement. The agreement is such that the cost should be the lowest possible, but there has to be at least one road that connects every neighborhood with every other one.



Figure 6.10: An overview of the city and the roads, with children in action to solve the problem.

As you can see in the figure 6.10, we built a big mat to make the children try to solve this challenge in practice. White roads are unpaved and they have to try to pave them using the yellow pieces, of course using as few as possible, as they do not get pieces to cover the whole network; they are actually challenged to try to save as many as they can.

Kids are finding the minimal spanning tree for the graph that is represented by roads and neighborhoods, designing a network with the minimal total length, in term of pieces.

A last challenge in this topic is the more classic Traveling Salesman Problem. The problem here was presented in school on paper, in form of an exercise sheet. Kids were encouraged to try by themselves, and only later presented with some of the well-known algorithms for its solution, such as the Nearest Neighbor or Cheapest Link algorithms.

1. Nearest Neighbour

- dalla città di partenza (possiamo sceglierla noi), visita la città più vicina
- da questa città dove siamo arrivati, visita la più vicina che non hai già visitato
- quando abbiamo visitato tutte le città, ritorna a quella di partenza

Quanta strada hai percorso (come somma delle lunghezze)? _____

2. Cheapest Link

- ordina tutte le distanze tra le città in ordine crescente
- seleziona la più breve dalla lista
- seleziona la più breve delle rimanenti, a patto che non ci siano 3 strade che partono dalla stessa città e non che non si chiuda un giro (cioè non torni al punto di partenza) con meno di 4 città
- terminiamo quando otteniamo un giro (percorso chiuso) che tocca tutte le città

Quanta strada hai percorso (come somma delle lunghezze)? _____

3. C'è un modo più breve per farlo?

Se sì, come l'hai trovato?

Quanta strada hai percorso (come somma delle lunghezze)? _____

Figure 6.11: A part of the TSP worksheet.

6.2.4 Cooperative Games

If we want to underline the cooperative aspects of some games, the activity to begin with is for sure the game with colored ball to get back into their place (right color). 6.9

As said, the game was presented to students as a group task, giving them the rules, but with the goal that they find out by themselves the algorithm for the solution. Kids are let free to try out their own strategies and “algorithms” to solve the game as fast as they can. They usually get to some sort of good way to get to a solution in a reasonable time, but, as we will discuss in the following chapter, we noticed some interesting configuration occurring during the game.

As a variation activity, another similar team game can be played. Children get in circle, each with his own chair (or gym-circle on the ground); one extra place is added in the circle. Each of the players fix their own place among those in the circle and mark it with a name tag or with some object. To begin the game, the players have to randomly change places, ending up in a place that is not “theirs”.



Figure 6.12: Reorganizing their place in the right chairs.

The goal of the game is that everybody has to get back to their original spot. Of course, there are some rules to follow. People move one at a time, can move only to go to the empty place and only neighbors (or neighbor’s neighbors) can move to that place; it is so possible only to overpass one person.

6. Teaching Experiment

The dynamics involved are almost the same as we see in the colored ball game. This can though involve more people, being the best number of players between 10 and 15, and becoming a real challenge for groups up to 25-30 people.



Figure 6.13: Playing in the garden with hula hoops circles.

6.3 Programming and pre-programming for primary school, Scratch based or Unplugged?

In this part, we will describe a comparison between two different approaches to teach some algorithmic and computational thinking to children, mainly in 3rd and 4th grade. Children learning is taken into main consideration and we want to analyze the difficulties students encounter using the different approaches. We will then describe the tasks used and look at some examples of the difficulties children face, on one side dealing with the problem of abstract thinking while programming, and on the other having troubles relating more practical activity with what the calculator does. Some conclusions will be presented in Chapter 7.

The question is whether it is better to approach the subject with an unplugged approach and only later go on with computer-based coding or if it is ok to proceed using Scratch-based software and tools to serve the same purpose. We do this by describing two different approaches which have been used in the teaching of these computer science and discrete mathematics topics. We will, in the next chapter, analyze and focus on certain difficulties students encounter while using both.

As a first design step, based on theoretical framework and literature, two hypothetical learning trajectories were designed for the two different approaches. One approach is Scratch-based, and has been taken from the most popular book on curricular resources about Scratch programming in our country (Coding, DeAgostini publisher, Ferrareso, Colombini, Bonanome, 2014). The second approach was developed by our research team, taking idea and inspiration from the Computer Science Unplugged project (Bell, Witten, Fellows, 1998, 2015 review) and other related sources (Casey et al., 1992), with a development, after a preliminary teaching experiment, to better adapt the activities to the school level and local situation and norms. The two HLTs are taking into account the theoretical framework presented above, both in the choice of tasks (e.g. some tasks are chosen for their RME approach, others for the group and cooperative work students have to do, and so on) and in the way of presenting them to the classroom or students. The tasks we are going to describe are just some of the many sequences of tasks that were proposed to various schools and age groups during the 2015/2017 school years in the bigger research project. In a design research paradigm (Plomp and Nieven), the activities were tried

6. Teaching Experiment

out many times, always with an a priori analysis together with the teachers and with a retrospective look after each lesson.

Schools	Grades	n. of Classes	n. of Students	Approach
1,2,4	3	4	78	Unplugged
1,2,4	3,4	5	80	Scratch-based
3,5	3,4	3	54	Unplugged
3,5	3,4	2	39	Scratch-based

Figure 6.14: Table 1: Classes involved in different grades, with students numbers and curriculum used.

6.3.1 Scratch-based teaching and learning

As mentioned above, the sequence of tasks we called “Scratch-based” is taken from this Coding book, which is getting popular in our country’s schools. Also we did use the M.I.T. official Scratch guide (Creative Computing, Brennan, Balch and Chung, 2014). It is following a similar approach to many school text book and even M.I.T.’s own guidelines on Scratch use and we feel it is good material for teachers. We did choose the tasks that are most popular among teachers already doing this kind of activity in their classroom, at least investigating the most popular in our area. We did in particular choose tasks related to sequencing, selection and iteration. The goal is to have students learn basic ideas behind an algorithm (seen as a sequence of instructions), but also more complex concepts like selection instructions (i.e. do this only if something else happens) or iteration procedures. A sequence of tasks on those three topics were selected together with the classroom teacher and then tried out with the students during mathematics and technology lessons.



Figure 6.15: Kids using MIT’s Scratch.



Figure 6.16: Kids using code.org's online software.

6.3.2 Unplugged approach, teaching and learning

Our “unplugged” sequence of tasks occupies 3 or 4 lesson slots of about one and a half to two hours and follows a brief introduction given on how computer works and binary numbers, in form of games.



Figure 6.17: Kids exploring the binary code.

Briefly describing the tasks, task 1 was an activity on paper, about binary image representation. Students had to color a grid which was provided with 0s and 1s and produce a drawing following the numbers. This task goal was about following instructions and beginning to understand how a computer transmits information.



Figure 6.18: Kids coloring with pixel art.

6. Teaching Experiment

Task 2 was about giving and receiving instructions. Students were divided in pairs and given a series of shapes and objects they could move on their table. One student (1) for each pair was to create a composition on his table; without looking at each other (physical barrier between the two), student 1 had to explain to the other how to reproduce the same composition with the objects and shapes. Children were required to be as precise as possible while the game went on, and to try to find out compositions that were harder to form. Only oral communication were left them, not to make them “correct” the other mistakes or looking at the other composition. Slightly different versions of the game were tried out, e.g. with just one student giving instructions to all others, or with different kinds of objects, even with just drawing something instead of moving objects, and so on.



Figure 6.19: Setting for the kids to give instructions to the others.

The following tasks had the goal to make programming even more tangible for children. We wanted them to learn to give instruction as a calculator, through a path to walk on. One student (blinded) was the “robot” walking along this path on the ground and the others were the “programmers” having to give him instructions how to move to get to the end. We did this both speaking and then written. The written exercise does not give the possibility to correct the robot while you are actually giving the instructions, kind of how a real computer program works. Finally, with some of the classes we went on to

6.3. Programming and pre-programming for primary school, Scratch based or Unplugged?

construct some more complex sequence of instructions, posing different games to strengthen the concepts, but always with similar goals.



Figure 6.20: Giving instruction to get the kid-robot out of the maze.



Is this a base for programming and robotics in higher school grades?

7

Results reached and final discussion

The description of the results obtained in the various stages of the research will be divided, as in the presentation of the activities in Chapter 6, by topics and sequences of tasks.

7.1 Graph Theory and Cryptography

Map coloring



Figure 7.1: A child involved in the different map coloring problems.

The feedback obtained from the students has been incredibly varied and interesting. As supposed, the problem was easy to introduce and explain and most students got it in just a few tries.

The students were left to work on the problem by themselves but also free to cooperate with others; vivid discussions arose about the best procedure to fulfill the problem requirements.



Figure 7.2: A child involved in the graph coloring problem.

They soon realized the problem with a bigger map (e.g. the U.S.A. map, see figure 7.3) was not of the same difficulty as the first one, where, with a small number of regions, one could quickly try the different possibilities. Many didn't manage to color their map with just four colors, but some of them even found out some local properties, which we will see in some of the following examples.

The two-colorable map was, on the other hand, as predicted, easy to solve with just two colors, maybe also because they had some experience with handling the problem. The so called “have-to” algorithm (if a country is yellow, then the neighbors have to be red and their neighbors have to be yellow again, and so on, see Theorem 4.2.4 and the part following) was discovered by many of the students.

Even the task of producing other examples which only required two colors was positively solved by about half of the students.

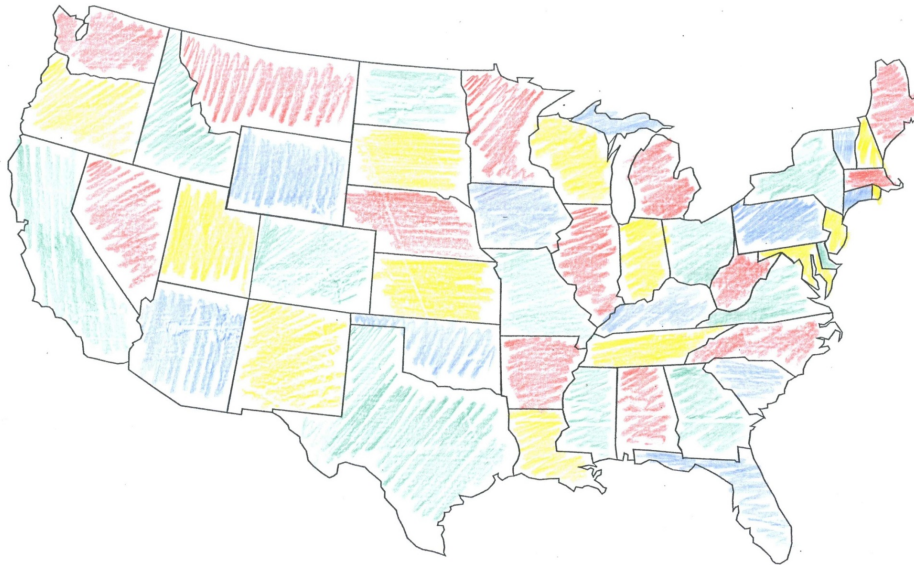


Figure 7.3: An example of a possible coloring for the U.S.A. map, obtained by one of the students.



Some properties from these worksheets, which are mathematically known, apart from the 4-colors Theorem, were discovered by some of the students.

Mappe e colori!

Il nostro cartografo vuole colorare le regioni di questi 2 riquadri, cercando di utilizzare meno colori possibile. Attenzione, regioni confinanti non possono essere però dello stesso colore! Di quanti colori avrà bisogno? Cerchiamo di utilizzarne il meno possibile!

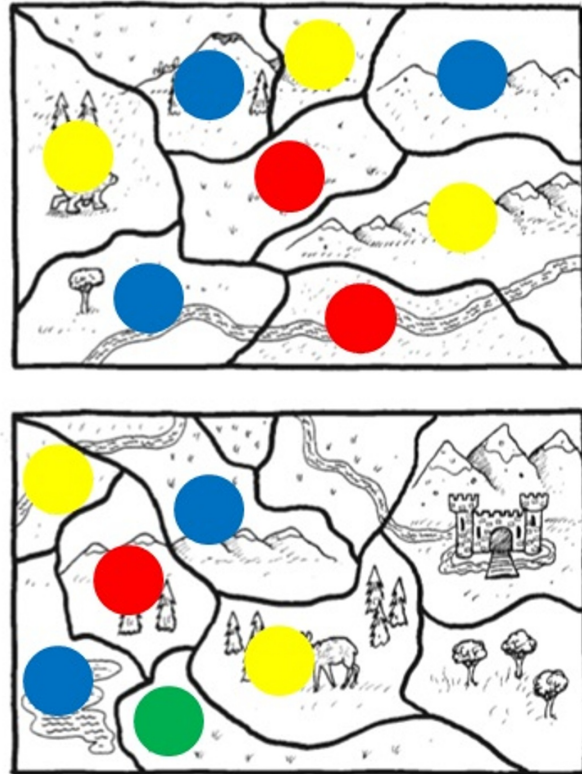


Figure 7.4: The comparison between a situation where it is possible to have just three colors and one where I can say that it is not possible.

For example, if a country is surrounded by, in total, an odd number of other countries, then it cannot be, even locally, three-colored). It is for sure not immediate to figure out this, without knowing any graph theory.

As you can see from the picture, when the red country in the second map, is surrounded by five different other areas, then we can conclude that three colors will never be enough to color the entire system, i.e. we need one color for the middle area and three others for the odd number of countries around it.

Graphs and dominating set

As in the map coloring activities, feedbacks were really interesting, also in a graph theory perspective. As supposed, the Euler paths problem is possible for students to solve just by trying some different possibilities.

They had some difficulties to notice that the solution to the Königsberg's bridges problem doesn't exist. The question on the worksheet is probably posed in a "tricky" way, asking to find which path is the correct one and letting to understand that it is like a solution is possible.



Figure 7.5: Children involved in the “graph drawing” activity.

The main challenge for the students was then trying to guess why some graphs work and some don't. Some guesses were really well thought and in the direction of the right solution. With the younger pupils, we had to guide them to find the right formulation of the answer (about number of vertices with odd number of edges, see Chapter 4), while some of the older ones almost got it totally right by themselves.

The discussion here is around even and odd numbers of edges starting from a given vertex. If all of the vertices have an “even number” of edges with the given vertex as an end point, then the configuration is easy to draw in this way.

Children need to be able to distinguish odd and even number and also to relate the parity of the number to the roads that they had in the first problem. If an intersection is crossed an even number of times, then it is plausible. If we have one with an odd number of roads going in, they soon realize it is difficult to get there and the go away, without crossing the same road more than once.

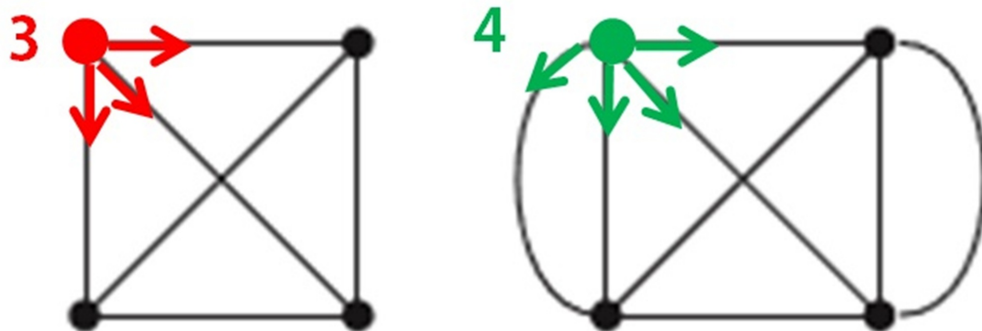


Figure 7.6: Examples of different figures. The green one works, while the red one does not.

It was clear in many circumstances that connecting the problem with the reality, as in the first one, make the student realize the fact way more easily. It is really difficult for them to see this solution in the paper setting of the graphs problem, while it is feasible to get it if put in practice in the “walk along” problem.

In some of the teaching experiments with younger children, we even set up a real path to follow among the roads of the villages and make them do it, as a real orienteering course.

As we saw in our introduction to RME, rich, “realistic” situations are given a prominent position in the learning process. These situations serve as a source for initiating the development of mathematical concepts, tools, and procedures and as a context in which students can in a later stage apply their mathematical knowledge, which then gradually has become more formal and general and less context specific. (VdHPD14)

As in the Dutch expression “zich realiseren”, which means to imagine, the word “realistic” has a broader meaning in this case, referring to the fact that students are offered situations which they can really imagine (in some case we have to say even put into practice with a real life exercise).

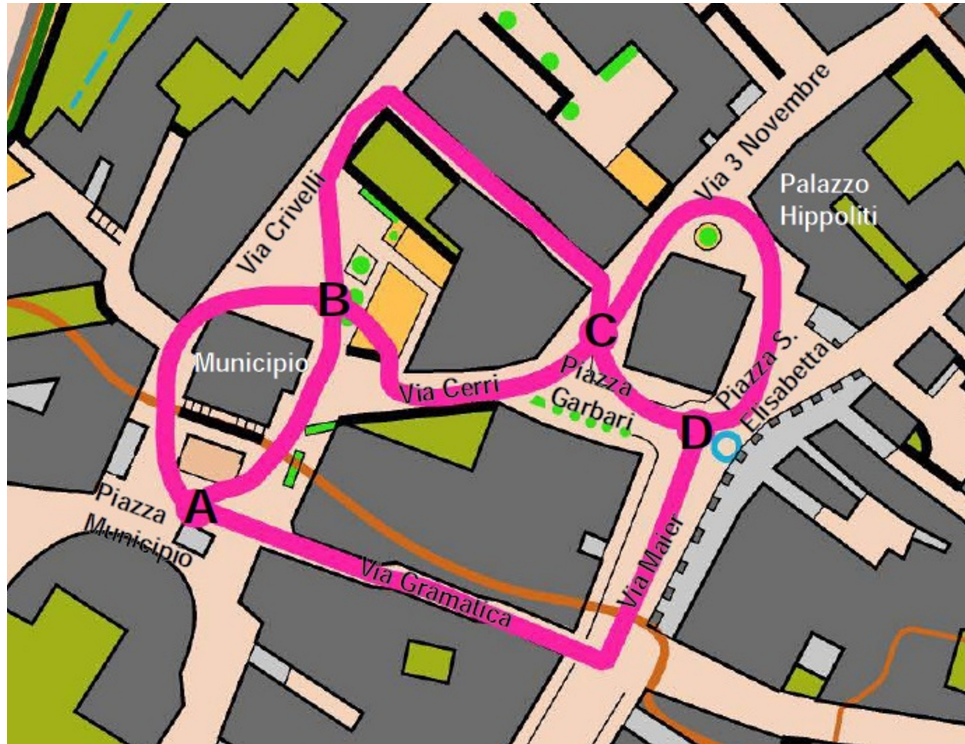


Figure 7.7: Orienteering course to make the “follow the line” problem more real.

In the minimum dominating set problem we saw in the students’ reaction that, as supposed, finding the right solution is difficult, i.e. it is not immediate to find it just by trying some possibilities. The approach was again to let them have some time to think about it and figure out some kind of algorithm (at least partially working) to try to solve this problem. We showed on the black-board with some magnets the best solution they were finding as time passed, to help motivating them. Kids usually get an intuitive sense that this problem is much more difficult than Euler paths one.

In a couple of occasions, a student found the right solution quickly but, when we asked her/him how, she/he realized that there wasn’t a fixed procedure and that it was probably in some sense a bit lucky to start from the right point. It was really impressive how students realize that the difficulty of problems is changed in relation to the previous exercises, and that they can notice the difference just by trying by hand. Once they are convinced about this, “translating” this into complexity and computer science gets possible, and giving the ideas of easy and difficult problems, still without talking about mathematics, is possible.

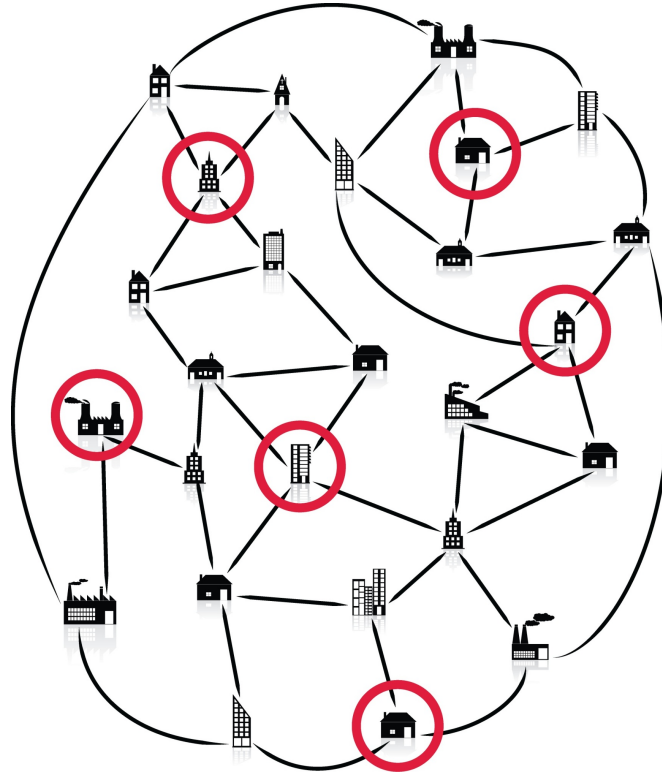


Figure 7.8: The minimum dominating set solution of the Firestation problem.

Binary numbers and error correcting codes

These activities were probably between the most captivating and engaging we proposed to the students in school, especially among those who are strongly related to mathematics.

The preliminary activity, “guessing” the number, captures a lot of attention and reactions, those of a magic trick. We left the suspense for a little bit and started the topic from the beginning before giving students the solution.

When explained that there is math behind it, students want to know how it works and learn the trick. In some of the classes the students even wanted to perform the game again with each other.

In the first activity about binary numbers, the problem is apparently easy to solve: adding up numbers to get a total number and its representation in binary code. Anyway, some students had troubles in deciding which way they should flip their cards and help from the rest of the group is needed.

Counting with these cards, i.e. making them represent numbers in order,



Figure 7.9: Constructing binary numbers.

makes students think about how binary numbers are written. The student with the card with number 1 on is the one who realizes things easier, as he/she

7. Results reached and final discussion

will flip the card each time, making some guessing about even and odd numbers; the student with the card with number 2 will flip every two numbers and so on. Discussing this while the activity is going on was very interesting and students had a good understanding of the situation. Especially when talking about even and odd numbers and how they form, we can use the activity to discover some beautiful properties of these numbers, e.g. if we want to add numbers to make an odd number, we need one of the addendum to be an odd number (more than one is not possible as we only have 1 which is odd).

Also, about data transmitting, it is easy to give the students an idea of how data is written and transmitted in a computer, with the examples of “small light” being easy enough to understand. We did not think it would be easy to deal with these topics, but results in the tests we had at the end of the lessons were quite good and demonstrated that children, even quite young ones, can grasp the hems of computer science and engineering without many troubles, even in a very simplified way.

To fix this concept of binary code even better, we did use some of the so called “pixel art”, with a more mathematical touch than what is usually presented in schools, and only with black and white colors.

And we see how a sequence of zeros and ones becomes a picture on the pa-

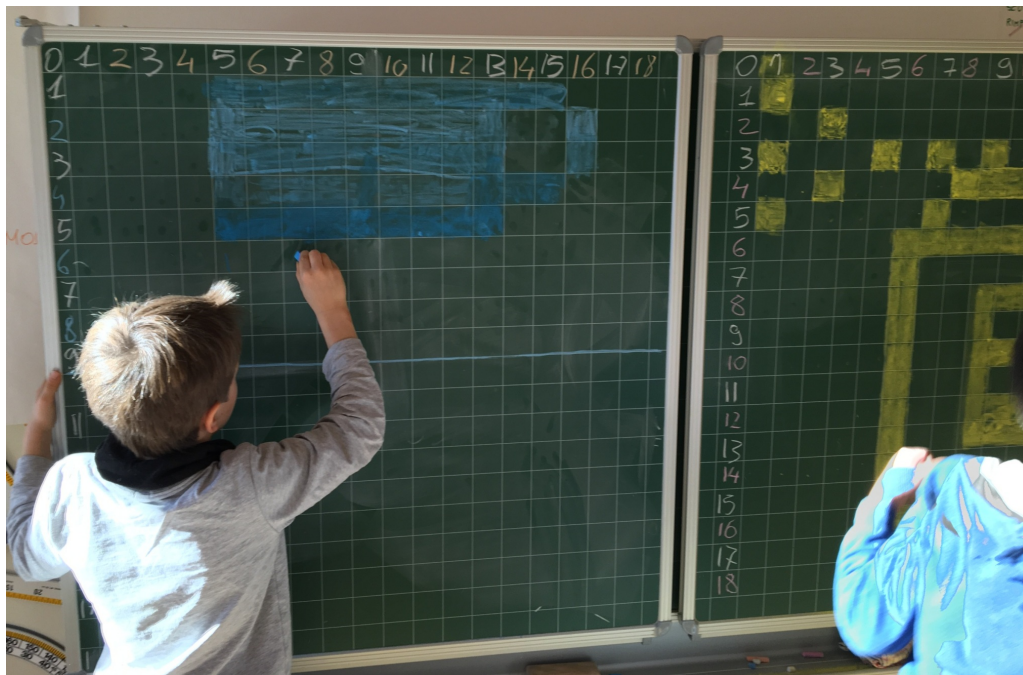


Figure 7.10: Binary writing turns into a picture.

per and grid that is provided to the students. This particular activity makes

a connection between the binary representation of pictures, with the computer science concept of pixel (picture element), mathematics, with sequence of numbers, art on the production of new drawings and also some competences in spatial perception, as following the right line of commands, do not skip any of them, counting the right number of spots, and so on.

The last activity, as said for the first one, often surprised the students as it



Figure 7.11: 0 means empty and 1 is to color.

looks like a magic trick, before realizing how it works.

When we asked their ideas on how the game works, many different possibilities came out. Someone conjectured about counting the numbers of black or white cards on each line, but still she/he didn't understand how so many numbers could be memorized. Just once a student found the right way of doing the game, and only after some discussion and various guesses.

I think that engaging the students and entertaining them with this kind of games is sometimes good to vary from the normal lesson and, for what this experience showed, it gets their attention really well. Even among the older pupils, the interest in this kind of activities were really high.

Connection with elementary number theory

Comparing our results with some statements from (LF01) we can notice that the possibility of mathematically controlling formulas, ordering them and connecting them to algorithms, help the students connect what they are doing



Figure 7.12: Younger children really need to be careful when following the right lines.

with the meaning of the mathematics they are dealing with. Some of the topics we are dealing with are posing the same challenges as, for example, prime decomposition might do in high school later on and offer “a number of pedagogical opportunities to help alleviate students’ improper attitudes towards mathematics [...]”.

About this, the binary number activity provided plenty of food for thought regarding the variety of representation of natural numbers, with a good reflection of unicity, and, as we saw, discussion for when a number is even or odd. The use of cards for doing this, especially in younger ages might help the “Semiotic control”, by (LF01) definition, the ability to proper interpret and handle symbolic expressions and statements involved in a mathematical task as texts to be interpreted, not as carriers of keywords or any sort of clues. So

our goal here is to have students focus on and recognizing the problem situation and the goal we want to achieve by the problem which is posed (this can be valid for every problem we are presenting in this document, not only this section).

As Ferrari is also showing, this connects with the constructivist interpretation by Dubinsky's (Dub91) with the action-process-object developmental framework. Dubinsky developed this framework starting from Piaget's ideas that an individual, disequibrated by a perceived problem situation into a particular context, will attempt to reequilibrate the situation by assimilating it to pre-existing schemas. Dubinsky holds that the constructions are of three kinds - actions, processes, and objects. An action is any repeatable physical or mental manipulation that transforms objects in some way. When the total action can take place inside the mind of an individual, without having to go through all the steps in the algorithm, the action becomes a process. New processes may always be constructed from actions but also from the modification of existing processes. An action, when conceptualized as a total process finally becomes encapsulated as a mental object (ZG97), (TTD⁺99).

Public Cryptography from dominating sets

The activity was inserted after a longer introduction of cryptography, so students already know about historic examples, the key exchange problem and so on.

The activity itself is not so easy to explain and usually kids need one example on the blackboard to fix better what they have to do. The summing process needs to be understood well and therefore we usually worked with pairs instead of single students doing the activity, so they can double check the computation.

When exchanging messages and trying to read the others' encrypted message, many students were surprised they actually could not work out how to understand the initial number.

The related activity of actually producing graphs to do the transmission process went smoothly, always with the aid of an example on the blackboard for everyone to see. Once they get how to build such a graph, some students wanted to repeat the whole process again, with a graph they invented.

I think we managed fairly well to make them understand, informally but with a good idea of the processes needed, the base of public-key crypto systems. At the end of our school experimentation, students were amused with the discovery of possible ways to transmit secret messages and thinking about computers doing the same things they just did on paper. Trying public cryptography “by hand” makes students more participative and the concept becomes easy to get, even without need of a lot of time to explain all the process in cryptography. The important concept of one-way function is at the end clear to the students. We could verify that in a direct way, seeing that they could produce new examples, and perfectly understand the dynamics of the cryptographic protocol we used in this task.

7.2 Computational Complexity of algorithms and Cooperative Learning

Searching Algorithms

Students, during these tasks, will learn, use, and describe the different algorithms that were showed in Ch. 6. They need to be able to follow clear instructions when doing the activities and while they are doing this they are carrying out algorithmic thinking.

As we anticipated, we are looking at two different kind of algorithms, sequential search and binary search, knowing (as teachers) that sequential search is way less efficient than binary search.

The amount of time sequential search takes isn't very predictable and can vary wildly, from finding something straight away, all the way to having to check every single item! If our luck goes wrong, it is possible that we might be looking at every single item before we get to the right answer! Binary search is instead more predictable and follows a given procedure which will lead us to a solution in a fixed (and limited to a small) number of steps, eliminating half the items we are searching every time we look at one.

Some other aspects the students are facing are abstraction capability, e.g. when searching through some data, having to understand that it is important to focus on the information that we need to find what we are looking for, and generalization, with the goal of the student understanding that they can repeat these algorithm they find out in other similar, but not equal, situations. In some after task tests, we could evaluate that almost 70 percent of the students (this in grade 5) were able to perform on their own the algorithm of binary search again on a new, but still equivalent, exercise.

Sorting Algorithms and Cooperative games

It is really interesting to observe and analyze the attempts and solutions children this age can find out, even without having prior knowledge of any of the algorithms used. As we said previously, we didn't provide any procedure to solve the problem, being the main task the re-discovering of algorithms. The analysis of algorithms used by students is one of our focuses, as is the cooperative aspects that these games involve, which are of broader interest and especially arise from these group games. We look at the cooperative aspects

of learning from a perspective close to that of Davidson (Dav90), also using Vygotsky's way to analyze situations (see for example as a reference (Doo97)). Analysis of the results is video-based, qualitative and fine-grained; both group activities and classroom discussion are recorded and we also have many of the transcripts, together with field notes, student's sheets, and interviews as other sources of evidence. As already said, focus is put on students' learning and thinking, in reaction to the different tasks proposed. Following Zacks' Tverski theory in event structure in perception and conception (ZT01), data is represented by events selected from the video recordings available. We used an inductive approach in video selecting, beginning with viewing the corpus in its entirety and focus on details later on. Indexing and summaries of videos, plus a content log, help in this process. Going on with the analysis some events which were particularly relevant were isolated. Although there are some recognizable recurring situation and choice of words we coded, our focus is more on a "play-by-play" description of these chosen events. We are, with this approach, analyzing selected episodes focusing on the same happening and constantly revising our finding and new hypothesis, as in Cobb and Whitenack (CW96) with the involvement of "constantly reconciling provisional analytic categories with subsequent data and newly formulated categories".

Task 1

Task 1 (see Fig. 7.13) was an experiment to introduce students to the topic and to see what could be a result, in a game where they are let totally free to find a strategy on their own to get to the results. Through video-analysis, we could observe that some of the groups involved actually are reconstructing well-known algorithms as for example selection sort or bubble sort. Looking at the collaboration between students, one can observe that these two algorithms translate into two totally different group dynamics. Selection sort usually emerges when there are one or two students acting as leaders and just giving direction to others in the group, while algorithms as bubble sort are usually used in cooperative oriented groups, or also when a leader is absent. Interesting to see how, on the other side, groups with just one pupil acting as a leader are usually more efficient in terms of time though.



Figure 7.13: Children in the process of ordering themselves in a circle, following the alphabetic order, by first name.

Task 2

In task 2 many algorithms which are known in computer science are easily emerging from children strategies. The fact that students can discover well-known algorithms as selection sort, bubble sort or even more complex algorithms as merge sort or the really efficient quicksort is quite interesting. It is then really interesting to discuss with them the speed of these algorithms, usually leading to a discussion to understand which one is more appropriate for our problem. Results give that quicksort is a lot faster than selection sort, particularly if we are able to make children abstract their process to larger lists. In fact, quicksort algorithms are in the best methods known. Having the children counting how many comparisons they need to make when they sort their objects using both algorithms, makes it easier to realize that one is significantly better than the other.

Computational complexity and efficiency are, with these games, brought to young children in an easy and understandable way. This task provides a great opportunity to be more formal about the procedure followed and makes it easier to make comparisons and class discussion about time efficiency.

Using a balance scale for sorting weights is an accurate model of what happens on computers, because the basic operation in sorting is to compare two



Figure 7.14: Trying out the weight problem in practice resulted in an effective way to test the results.

values to see which is the larger. Most sorting programs are based on this operation, and their efficiency is measured in terms of how many comparisons they require to sort a list, because that is a good indicator of how long they will take regardless of the computer they run on.

Both quicksort and mergesort algorithms introduce an important and powerful concept in computer science: recursion. This is where a method (algorithm) is described in terms of itself. Both methods work by dividing a list into parts, and then performing the same kind of sort on each of the parts. This approach is called divide-and-conquer. (BWF98)

Task 3

Again here, cooperation is a must. If “faster” students do not wait for the others to get to their place and to make the comparison working, the group will never be able to get to their final result. Also, the discussion afterwards, usually provide a great environment to make children think and work cooperatively.

In these lessons students were sorting things into order, performing an algorithm, a step-by-step process that will always give the right solution, as long

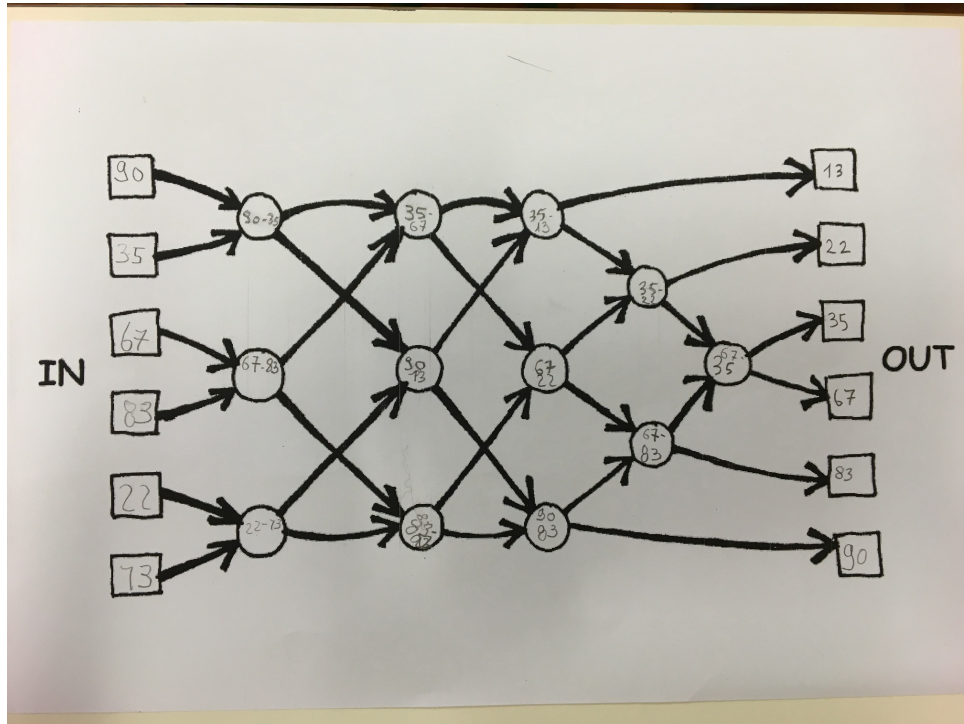


Figure 7.15: After practicing with the mat, we made it more “mathematical” with a paper sheet.

as it is followed exactly. In this case, the sorting network algorithm, is part of those so called parallel algorithms.

This sorting network is important in an abstraction process, as it is a really simple representation of a complex algorithms used in computer science.

We also discussed about the correctness of the algorithm: does it always work? Does it work if we go through it backwards? The answers to these questions were yes and no, as the sorting network, in its right version, is always working, no matter which the starting order is. On the other side, if we use it backwards, it does not always work, implying that a mathematical process is not always working, actually, we better throw it away, if we just can find one counterexample.

This proved a great task for team work and called for cooperation among the students on different levels: in fact, they need to cooperate with each other both in performing the “walk” on the sorting mat and later on in the class and group discussion.

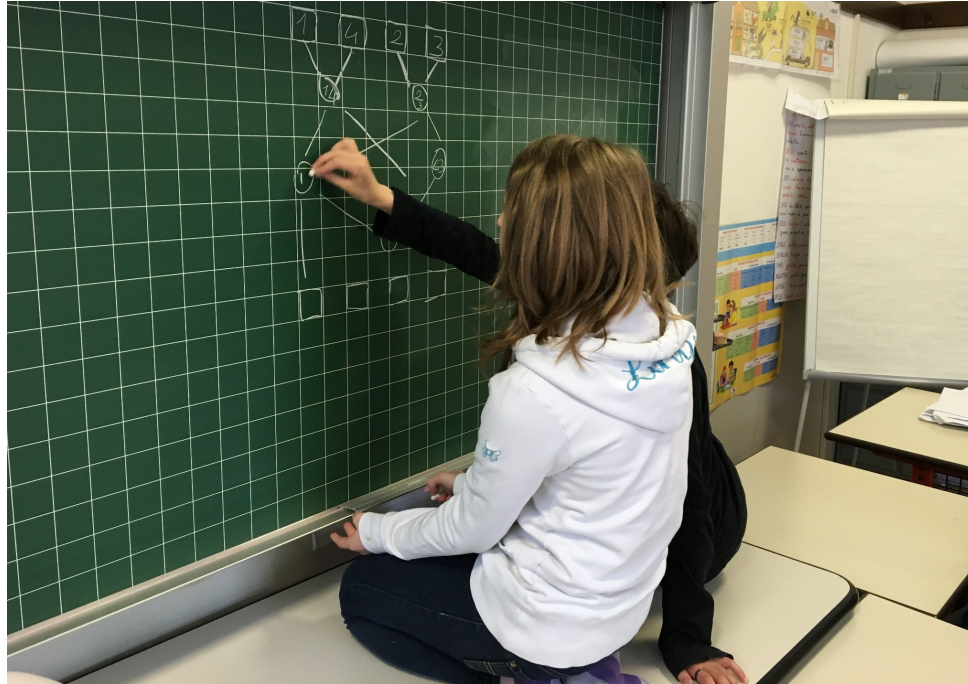


Figure 7.16: Students discussing the properties of the sorting network.

Task 4

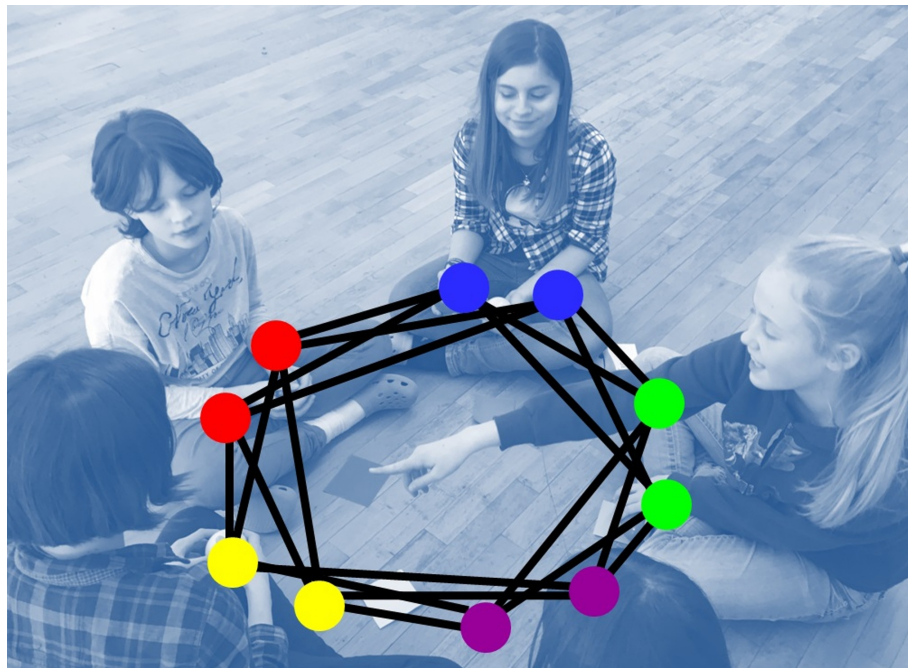


Figure 7.17: The game situation from a graph theory perspective.

In task 4 the algorithms are quite complicated (as based on graph theory) and students this age are not fully ready to make them formal. Anyways, it is

really easy to observe cooperative aspects and make kids reflect about them. While playing, they usually get to some good solution in a reasonable time, but some problematic situations arise.

A qualitative analysis of the results, through some videos recorded in the classroom, shows, according to Vygotsky's perspective on the zone of proximal development (Vyg87), the moment in which students find out that if they are greedy - or we might say use a greedy algorithm - (holding their own colored ball as soon as they grasp them) then the group might not success in the final goal.

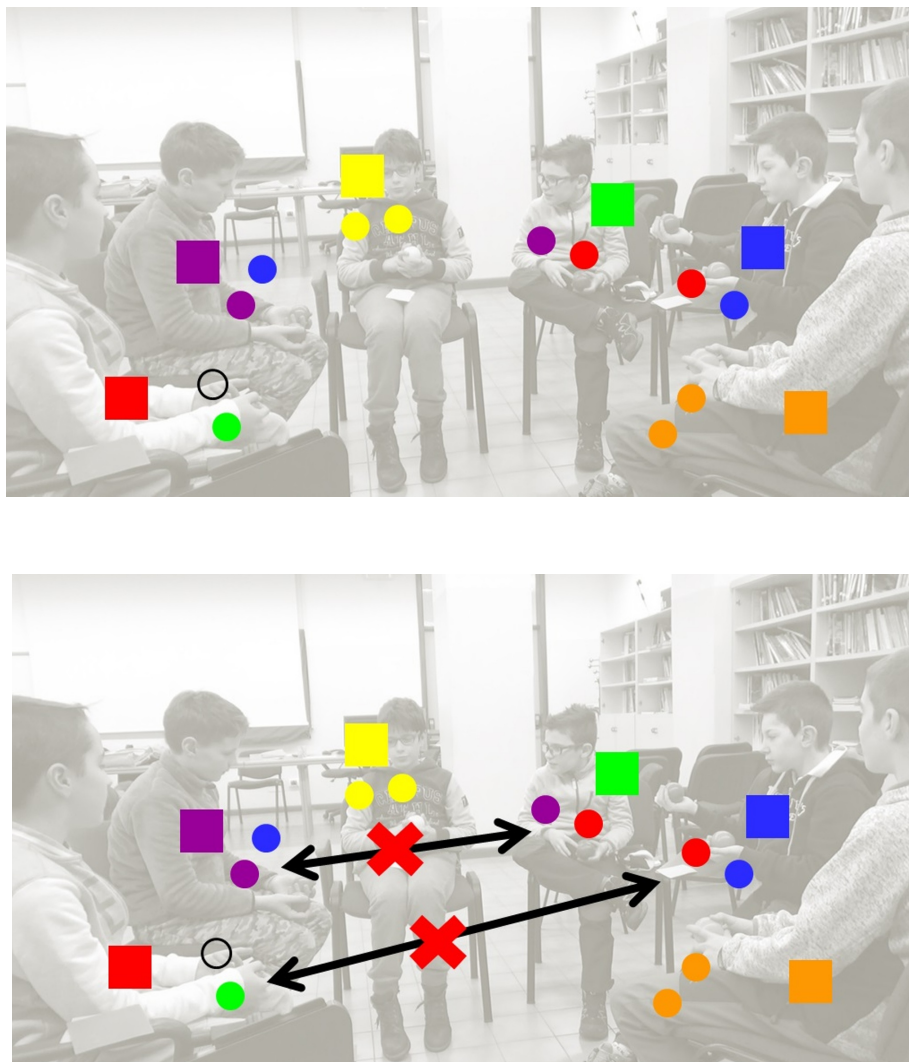


Figure 7.18: Stuck in a “greedy” configuration.

It may be necessary to emphasize that individuals do not win the game, but that the puzzle is solved when everyone has the correct color. They realize therefore that being greedy does not give them much advantage in the final.

They are somehow forced to work in a cooperative way. Also, they usually recognize this by themselves and they can easily realize that this is the fastest (and usually the only) way to get a solution to this game.

Something more on Cooperative Games

We already analyzed the dynamics of our cooperative games in the previous sections, and just wanted to underline the importance of cooperation in the modern world and especially in the everyday environment in the school.

One of the general antinomies frequently referred to by the Cooperative Learning scholars is that which put in contrast different models: competitive, cooperative and individualistic. These three different models of learning connects to three different models of values present in our school system as well as in global society. The change from a traditional school, essentially selective and competitive, to an innovative school, capable of producing in the new generations a cooperative culture.

The theoretical elements that demonstrate the superiority of the cooperative models (the students in interdependent groups towards a common project) compared to the competitive ones (the students against one another) and the individualistic ones (rigidly separated students) are, as already pointed out, numerous, valid and scientifically indisputable. All the works produced by the most important world researchers on Cooperative Learning, (Sha80) or as summed in (Chi11) demonstrate the superiority of a climate of cooperation with respect to a competitive or individualized situation in all cognitive, social and relational fields.

Computational complexity

The problem we presented about the roads to pave in the city (see Fig. 7.19), shows students the decision process involved in constructing a network (in this case, of public services). The main problem is optimization here. Also, in the Traveling Salesman Problem the reasoning should be quite the same, even if we get slightly different results.

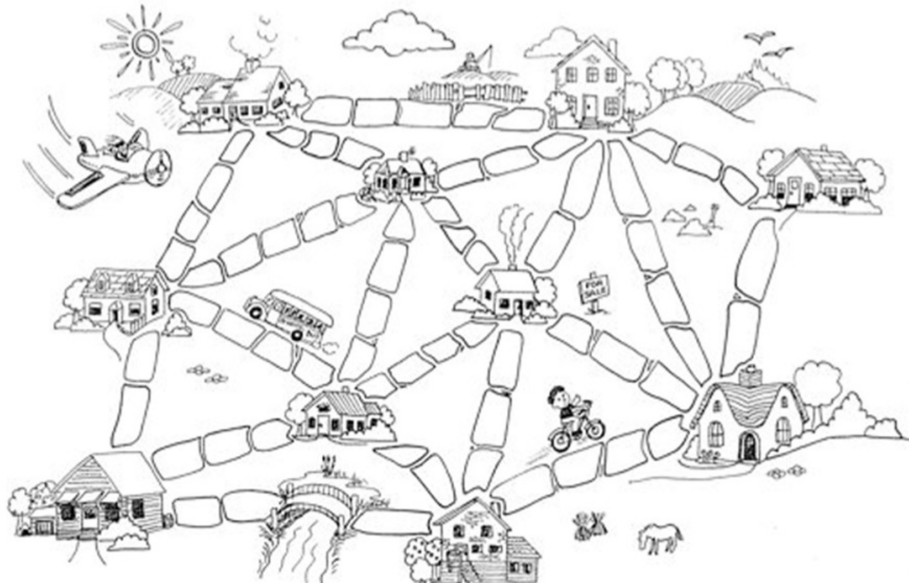


Figure 7.19: “Muddy city” problem, as presented in (BWF98).

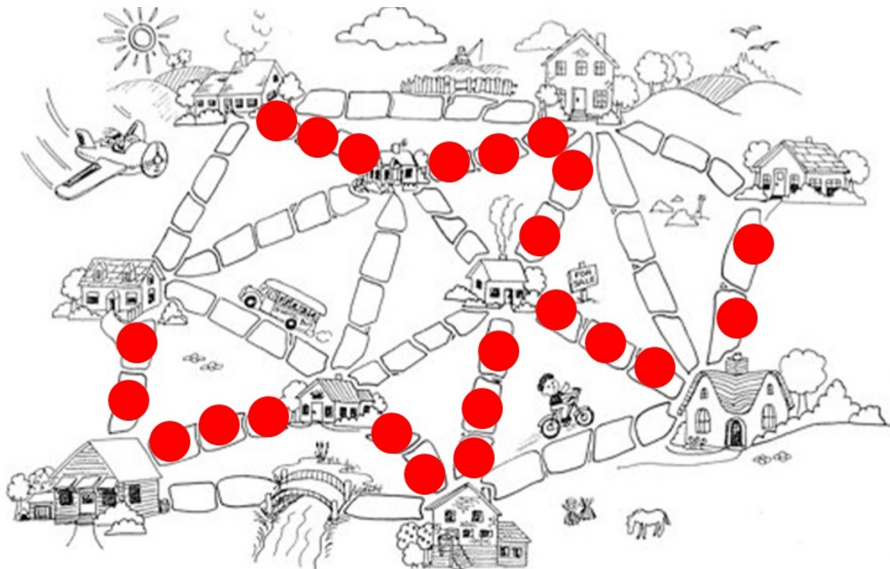


Figure 7.20: A 23 pieces solution is the best result we can achieve.

At a first tryout to solve this, learners usually get to a solution which works, but hardly ever is the best one in terms of efficiency. they manage to understand what they need to do pretty easily. Once again, the practical approach with the mat representing the city helps a lot in this abstraction capacity, making the problem more a “real world” one, as you can see in Fig. 7.21. Being able to move, put new ones and take some pieces of road away, makes the children feeling what they are doing and realizing they can change their project/solution pretty easily until they reach the best one.

Some of the conclusion children draw are that, in order for a calculator to



Figure 7.21: The problem of paving the city get easier as they can walk on it.

solve a problem of this kind in an efficient way, it must be given a precise system for doing it. While computers are good in following instruction we give them, they will not solve the problem by themselves.

We did try to hand to another class, acting as a control group, the worksheet of the problem on paper, and we did not get the same results. First of all, it takes way more time to make a single attempt, as they need to either write, or imagining it in their heads and usually need a longer time to understand what they need to practically do and so on. The mat and the storytelling around it makes the game and children immediately can work with their problem solving, without losing much time in actually understanding the problem. Also, trying to walk on the mat makes them realize way faster (in our timed attempt, only 2 students out of 23 in one group got the right answer, while all the groups in the “mat approach” could find this in under 4 minutes) that 9 roads are the right number to connect 10 different points, without being redundant, but still making the problem work. At last, also improving the 9-road solution to get the best possible is easier on the mat as, as said, it is quick to move and replace the pieces.

In this way, learners get to face different efficiency and different algorithms to get to the same solution, but not with the same effort/expense.

In the same way we could make the problem of the traveling salesman more tangible to learners by doing orienteering. This sport, involving navigation, is

quite suitable in the version where you have to visit a certain amount of control points (check points) and need to find out the best way, and order, to visit them, as you can see in Fig. 7.22. Orienteering is a sport, where competitors use maps to complete navigational challenges in given areas. Orienteering involves both physical and mental challenges and rewards competitors with a unique adventure on an ever changing playing field.

Here the sport approach, and setting it almost as a game, makes the trick and learners are super motivated to try to find the best possible solution (yes, being faster than the others in the group can sometimes be good motivation!).

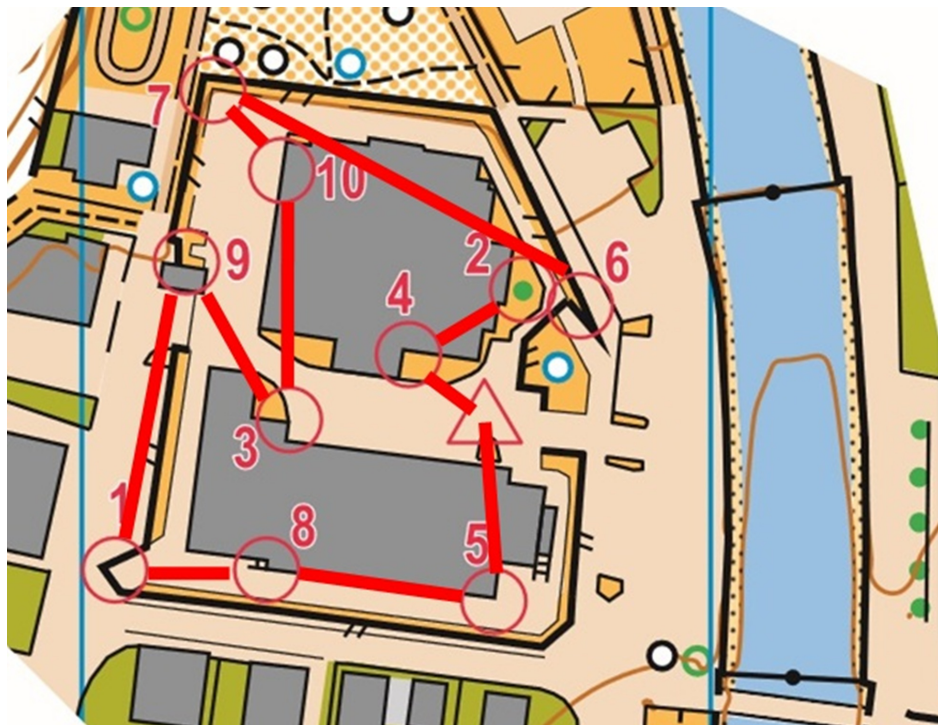


Figure 7.22: The Traveling Salesman Problem in its orienteering version, with one of the students' solution.



Figure 7.23: Giorgia saying “Let’s go!” after she realizes where the best new step will lead her. Is it a first step towards the abstraction of a solution to the problem?

7.3 Programming and pre-programming for primary school, Scratch based or Unplugged?

Different difficulties emerging

The events we are focusing on does not pretend to show that one approach is better than the other, but which kind of, different, difficulties each of them can create in the children learning and thinking using the different approaches. In the unplugged approach, children easily figure out what they are really and practically doing, drawing conclusions that they usually do not get in front of the computer. See for example the following figures where the transition from longer instruction in the first part to shorter instruction (switching to an iteration notation) in the second comes automatically. Almost every student quite naturally finds out that it takes a long time to write again and again the same instruction and is quickly asking himself if he might make it somehow shorter. This is probably due to the fact that they were left free to develop their own language with arrows, and they feel they can adapt it to what is more appropriate and efficient for the situation. Videos showing these moments when they realize this fact has been isolated from the data.

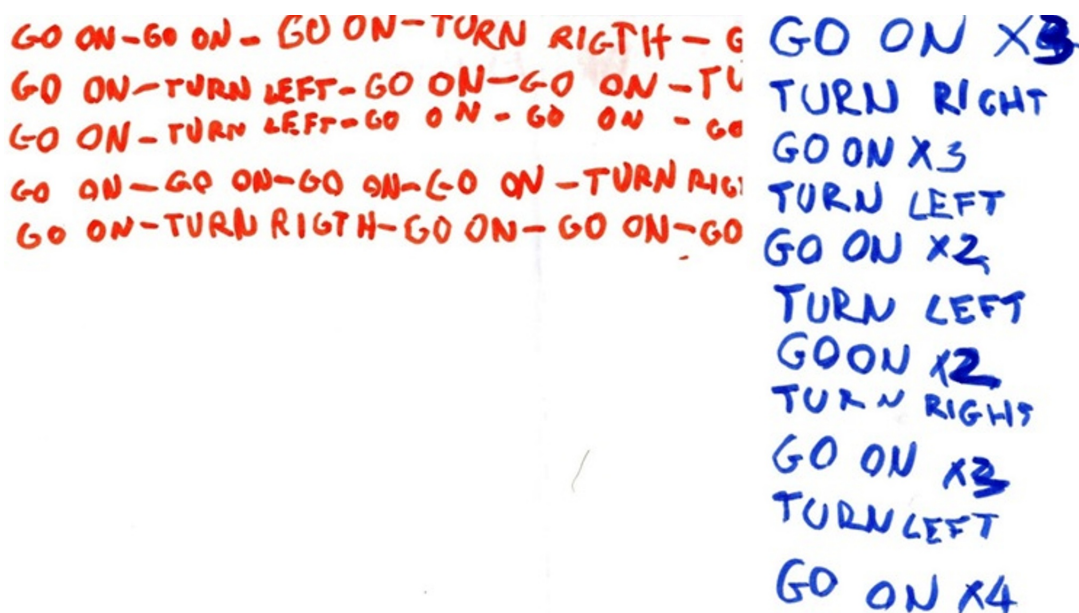


Figure 7.24: Showing the transition from sequential instruction (red) to iteration (blue).

7. Results reached and final discussion

The algorithmic thinking focuses on students learning to sequence a set of instructions to complete a task, to use specific types of instructions in their algorithms, and to find equivalent ways to achieve the same outcome. (BWF98)

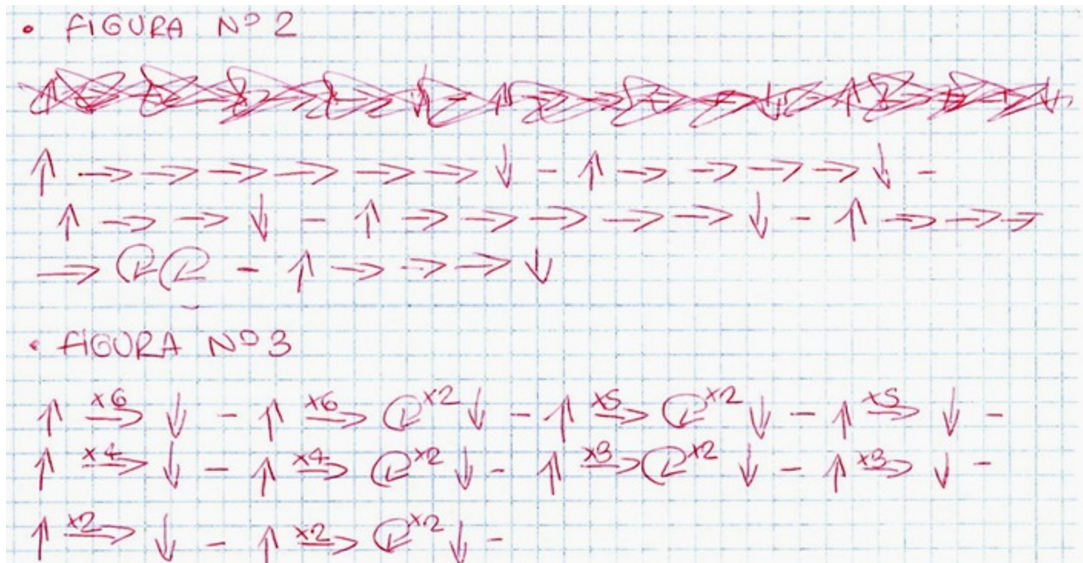


Figure 7.25: Showing once more transition from sequential instruction (n.2) to iteration (n.3).

Or, on the other hand, as an example, see the following short transcript from a video (in front of a Scratch set of instruction on the computer), where the students do not realize the usefulness of shortening a computer program to make it simple and more efficient:

Teacher: *Why aren't you writing it in a shorter way (more compact?). You don't need to write an instruction 6 times, you could write "do this 6 times".*

Student: *Well, but what's the need for it? The computer is doing it anyways.*

In these examples, students doing it unplugged quickly find out they are more efficient if they switch to an iterative mode of giving their instructions, while students doing it on the computer do not really realize this. On the contrary, they should learn one more command (or Scratch block) they do not already know to do it, so in the beginning it does not seem so appealing. From many events observed, quite surprisingly, this shortening is not immediate on the computer.

In detail, if we look at Fig. 7.26, we usually get to step 2 but not to the 3rd one without any help from the teacher. The mental passage to get to iteration is not clear, if not explained in a related way beforehand, of course.



Figure 7.26: In a programming environment as Scratch, the passage between single instruction, multiple instruction and iteration to get, circa, the same movement.

Following our qualitative video analysis, in general, it seems that both selection and iteration instructions do not come naturally in the Scratch environment as they do in an unplugged approach. Setting the activities and game in a real world scenario, especially at this young age, seems to give children a better idea of the advantages of iteration and the working principles of selection programming. Maybe creating some highly inefficient situation could force this process to happen in this case, too.

A second aspect to consider is errors done while writing a program. Analyzing error situation we can observe that it is actually easier for the students to spot the errors in a Computer-based environment. In the unplugged approach, sometimes, error fixing does not work at all, i.e. they cannot even spot the error if told that there is one. Our conclusion is that, as they are controlling their own game, they, more or less, unconsciously, get to a right solution even with a wrong set of instructions. See Fig. 7.27, in a path were the child went to the right direction, even if told to go on the other way.



Figure 7.27: Student here is following her own thoughts instead of the direction received (cannot be seen from the picture, but just got the instruction to turn right).

On the computer-based environment, trying the program and making it running, given that the computer executes exactly what it has been told, students easily spots where the mistake is.

Another aspect we are facing at this young age is abstraction capability. Recalling what abstraction is in mathematics education, from (Kra07), we focus on two particularly pertinent aspects. The first emphasizes the process of removing detail to simplify and focus attention based on the definitions: the act of withdrawing or removing something, and the act or process of leaving out of consideration one or more properties of a complex object so as to attend to others. The second emphasizes the process of generalization to identify the common core or essence based on the definitions: the process of formulating general concepts by abstracting common properties of instances, and a general concept formed by extracting common features from specific examples.

As many references states (e.g. see Kramer, (Kra07)), abstraction capability is the key to be a good programmer and to have future programming abilities. Abstraction is a very difficult process and Scratch helps a lot in this direction; on the other side, the unplugged approach makes activities somehow too distant from the abstraction of programming and makes it more difficult to children to relate what they are doing with what they will later do on the

7.3. Programming and pre-programming for primary school, Scratch based or Unplugged?

calculator, as some video excerpts from these moments show. Aspects of a real world mathematics surely can help the transition to the abstract world of programming (Futschek and Moschitz, (FM11)), but we have to be careful in the subtle connection between the two areas.

In Fig. 7.28 we can see that the learner is trying to imagine and in some sense making reality of the task proposed by the computer (she has to decide whether the character has to turn left or right in the game). Many of the children observed need to stand up or move to understand the right decision.

While observing all these factors, we noticed some interesting results in the



Figure 7.28: Trying to make the computer situation real, using the hand to impersonate the character.

“give and follow instructions” activity with the barrier between the students. As an example, in Fig. 7.29, the two children got the white paper in the wrong

7. Results reached and final discussion

direction, probably having skipped something in the oral description. As soon as we let them look (when they think they are done!), they realize it and spot the errors. If the instructions were written it would be easy to find where it is and correct it.



Figure 7.29: Wrong direction for the white piece.



Figure 7.30: Here the mistake is in the word “on” or “over”, which in Italian has the same word for the two meanings.

Something similar happens in Fig. 7.30, when we look to correct it. Here the mistake lies in the double meaning of the Italian word “sopra” which can lead to either “on” or “above” or “over”. One student obviously positioned

7.3. Programming and pre-programming for primary school, Scratch based or Unplugged?

the pink star misunderstanding this. To enhance the abstraction capability we mentioned above, we tried to embed the world of Scratch-programming into a multidisciplinary challenge, creating with a 4th and 5th grade class a theatre plot and making it animate in Scratch, after writing and drawing by hand in the literature and art classes in school.



Figure 7.31: Kids working on the Scratch storytelling.

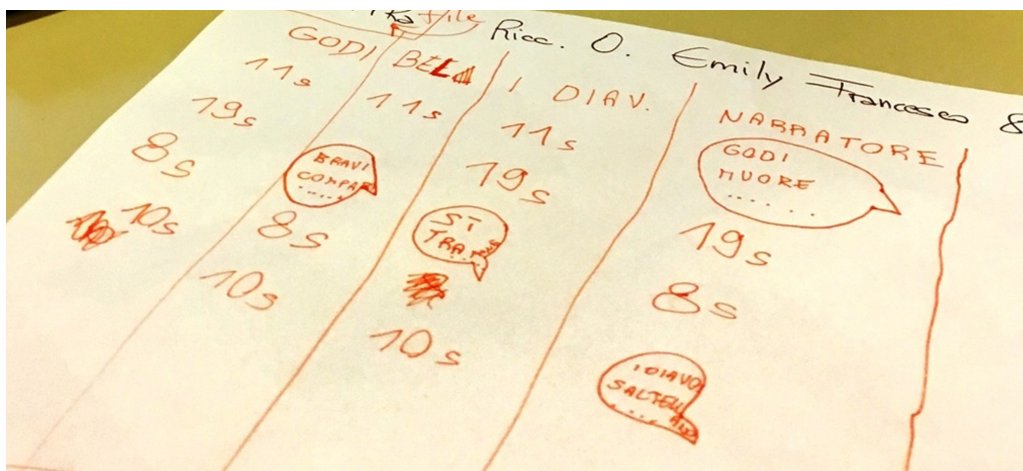


Figure 7.32: “Programming” one of the scenes, before implementing it in Scratch.

Some conclusions

As conclusions, we could see pros and cons of both approaches, and we feel that obsessing over using just one is not the correct decision. Video and other

data analysis show that there are aspects that ought to be dealt with in an unplugged way before writing them on the computer (algorithms, iteration processes and many others) and come really more natural to children if they use their own language, as can be seen in many “Aha! Moments” () in the recordings. On the other side, it is really difficult for young children to relate the more real-world-oriented tasks to computer science, as we can see example when they get stuck in finding the connections.

As we can see in (BWF98), giving sequential instructions are an important element of all programming languages. It also exercises the ability to predict what a program will do, reason about where any bugs are, and understand that there can be multiple correct ways to program a solution. Being able to give exact sequential instructions, work well together and understand how to break a big problem into small pieces and then tackling the small piece one at a time are all life skills that can be transferred from computer programming to other tasks that students need to do.

Future work will try to go into combining both approaches in a more comprehensive curriculum plan, creating new learning sequences that take both into account, which we will share with teachers and educators, in a relatively long developing process. Teachers willing to teach these topics are growing in number and they ought to be prepared for the challenge they will be facing.

8

Students materials and appendices

We report here some student's sheets and materials that have been collected throughout these three years of teaching experiments. Of course, it is just an exemplificative selection, among the tons of materials collected.

As we pointed out in the previous sections, a lot of video material, which cannot fit in this document but has been carefully analyzed for the various stages of the research project, is also available.

crypto

pixel art

mappe 1

mappe 2

mappe 3

istruzioni 1

istruzioni 2

istruzioni 3

istruzioni 4



Figure 8.1: Video analysis of one of the ordering games.



Figure 8.2: Video analysis of one of the instruction games.

order 1

order 2



Figure 8.3: The sorting network activity in one early stage of development.



Figure 8.4: Final mat for the sorting network activity.



Figure 8.5: After the unplugged tasks, also a lot of computer “pre-programming” was developed.

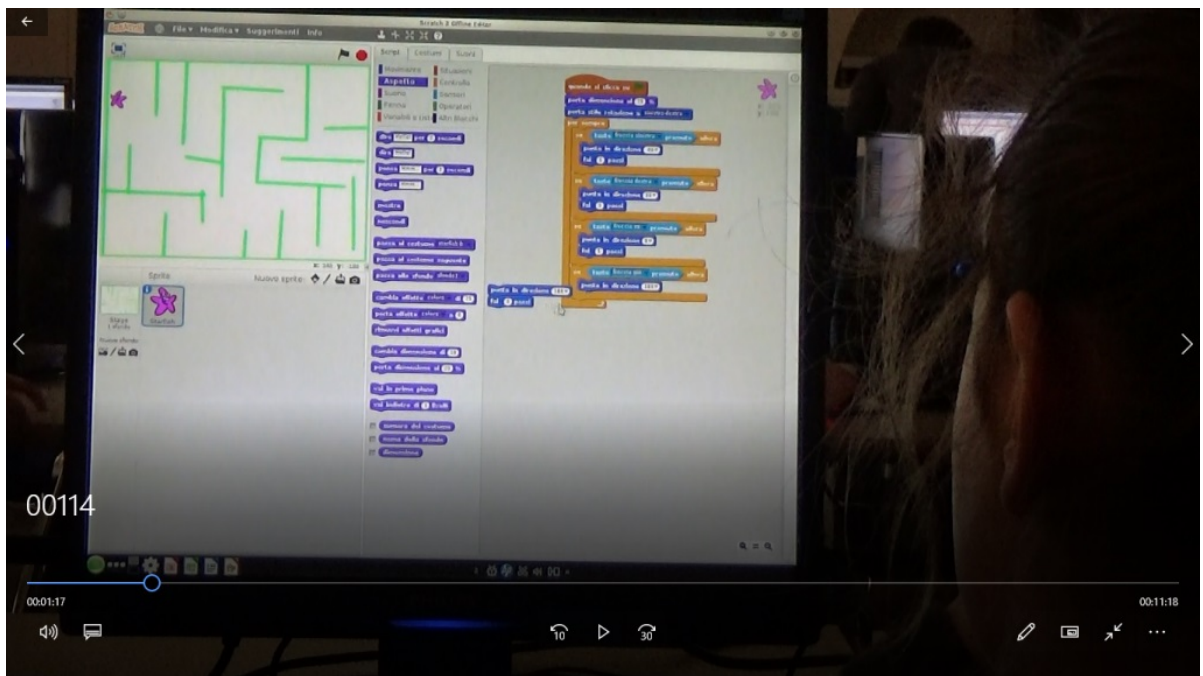


Figure 8.6: Video analysis of Scratch programming.

Bibliography

- [AH⁺77] K. Appel, W. Haken, et al., *Every planar map is four colorable. part i: Discharging*, Illinois Journal of Mathematics **21** (1977), no. 3, 429–490. 43
- [AL94] W.W. Adams and P. Lounstaunau, *An introduction to gröbner bases*, no. 3, American Mathematical Soc., 1994. 58, 60
- [Alb] G. Alberti, *Aritmetica finita e crittografia a chiave pubblica—un percorso didattico per gli studenti delle scuole medie superiori*, A. Abbondandolo, M. Giaquinta, F. Ricci (a cura di), Ricordando Franco Conti, 1–29. 11
- [BFSZ02] M. Borelli, A. Fioretto, A. Sgarro, and L. Zuccheri, *Cryptography and statistics: A didactical project*, International Conference on the Teaching of Mathematics 2. Proc, 2002. 11
- [BH83] D. Barnette and R. Honsberger, *Map coloring, polyhedra, and the four-color problem*, Mathematical Association of America, 1983. 33
- [BLW76] N.L. Biggs, E.K. Lloyd, and R.J. Wilson, *Graph theory 1736-1936*, 1976. 62
- [BM76] J.A. Bondy and U.S.R. Murty, *Graph theory with applications*, vol. 6, Macmillan London, 1976. 27
- [Bro92] A. L. Brown, *Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings*, The journal of the learning sciences **2** (1992), no. 2, 141–178. 73

- [Bro97] G. Brousseau, *The theory of didactic situations*, Edited and translated by N. Balacheff, M. Cooper, R. Sutherland & V. Warfield. Dordrecht: Kluwer Academic Publishers (1997). 85
- [Bro06] ———, *Theory of didactical situations in mathematics: Didactique des mathématiques, 1970–1990*, vol. 19, Springer Science & Business Media, 2006. 81, 82
- [Bru09] J. S. Bruner, *Actual minds, possible worlds*, Harvard University Press, 2009. 79
- [Buc65] Bruno Buchberger, *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*, Ph.D. thesis, Innsbruck, 1965. 58
- [BWF98] T. C. Bell, I. Witten, and M. Fellows, *Computer science unplugged: Off-line activities and games for all ages*, Citeseer, 1998. x, 1, 11, 87, 92, 146, 151, 156, 162
- [C⁺12] G. Cerini et al., *Indicazioni nazionali per il curricolo della scuola dell'infanzia e del primo ciclo d'istruzione*. 7
- [CB90] D. H. Clements and M. T. Battista, *Constructivist learning and teaching*, *Arithmetic Teacher* **38** (1990), no. 1, 34–35. 79, 80
- [CCD⁺03] P. Cobb, J. Confrey, A. DiSessa, R. Lehrer, and L. Schauble, *Design experiments in educational research*, *Educational researcher* **32** (2003), no. 1, 9–13. 73, 74, 75, 76
- [CFBW92] N. Casey, M. Fellows, T. Bell, and I. Witten, *This is megamathematics*, Proceedings of International Workshop on Parameterized and Exact Computation, IWPEC'09, vol. 955, Los Alamos National Labs, 1992, pp. 415–427. 1
- [CGM07] A. Centomo, E. Gregorio, and F. Mantese, *Crittografia per studenti*. 11
- [Chi11] G. Chiari, *Educazione interculturale e apprendimento cooperativo: teoria e pratica della educazione tra pari*, vol. 57, Trento: Università degli Studi di Trento, 2011. 150
- [CLZ10] G. Chartrand, L. Lesniak, and P. Zhang, *Graphs & digraphs*, CRC Press, 2010. 27

- [Col92] A. Collins, *Toward a design science of education*, New directions in educational technology, Springer, 1992, pp. 15–22. 73
- [CW96] Paul Cobb and Joy W Whitenack, *A method for conducting longitudinal analyses of classroom videorecordings and transcripts*, Educational studies in mathematics **30** (1996), no. 3, 213–228. 86, 144
- [Dán04] M. Dániel, *Graph colouring problems and their applications in scheduling*. 32
- [Dav90] Neil Davidson, *Cooperative learning in mathematics: A handbook for teachers.*, ERIC, 1990. 144
- [Der99] S. J. Derry, *A fish called peer learning: Searching for common themes*, Cognitive perspectives on peer learning **9** (1999), no. 1, 197–211. 80
- [DH76] W. Diffie and M.E. Hellman, *New directions in cryptography*, Information Theory, IEEE Transactions on **22** (1976), no. 6, 644–654. 49
- [Doo97] Peter E Doolittle, *Vygotsky's zone of proximal development as a theoretical foundation for cooperative learning*, Journal on Excellence in College Teaching **8** (1997), no. 1, 83–103. 144
- [Dub91] Ed Dubinsky, *Reflective abstraction in advanced mathematical thinking*, Advanced mathematical thinking, Springer, 1991, pp. 95–126. 141
- [Dur02] F. Duris, *Np-hard problems - univerzita komenského, bratislava, lecture 16, 2002*. 41
- [Eis95] D. Eisenbud, *Commutative algebra, volume 150 of graduate texts in mathematics*, 1995. 60, 61
- [Eve11] S. Even, *Graph algorithms*, Cambridge University Press, 2011. 62
- [Fel91] M. Fellows, *Computer science and mathematics in the elementary schools*, University of Victoria, Department of Computer Science, 1991. 92

[FH03] L. Fortnow and S. Homer, *A short history of computational complexity*, Bulletin of the EATCS **80** (2003), 95–133. 19

[FHD⁺15] D. Franklin, C. Hill, H. Dwyer, A. Iveland, A. Killian, and D. Harlow, *Getting started in teaching and researching computer science in the elementary classroom*, Proceedings of the 46th ACM Technical Symposium on Computer Science Education, ACM, 2015, pp. 552–557. 1

[FK93] M. Fellows and N. Koblitz, *Kid krypto*, Advances in Cryptology - CRYPTO'92, Springer, 1993, pp. 371–389. 92

[FK00] M. R. Fellows and N. Koblitz, *Combinatorially based cryptography for children (and adults)*, Congressus Numerantium (2000). 92

[FM11] G. Futschek and J. Moschitz, *Learning algorithmic thinking with tangible objects eases transition to computer programming*, International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, Springer, 2011, pp. 155–164. 159

[For09] L. Fortnow, *The status of the P versus NP Problem*, Communications of the ACM **52** (2009), no. 9, 78–86. 19

[Fre73] H. Freudenthal, *Mathematik als pädagogische aufgabe*, vol. 2, Klett Stuttgart, 1973. 82

[GJ79] M.R. Garey and D.S. Johnson, *Computer and Intractability*, A Guide to the NP-Completeness. Ney York, NY: WH Freeman and Company (1979). 19, 71

[GK09] H. Gelderblom and P. Kotzé, *Ten design lessons from the literature on child development and children's use of technology*, Proceedings of the 8th International Conference on Interaction Design and Children, ACM, 2009, pp. 52–60. 1

[Gla75] E. von Glasersfeld, *Radical constructivism and jean piaget's concept of knowledge*, The impact of Piagetian theory **30** (1975), 105–112. 79

[Gle12] L. Glebsky, *A proof of hilbert's nullstellensatz based on groebner bases*, arXiv preprint arXiv:1204.3128 (2012). 61

- [Gol49] M.J.E. Golay, *Notes on digital coding*, 1949, pp. 657–657. 58
- [Gra94] K. Gravemeijer, *Developing realistic mathematics education: Ontwikkelen van realistisch reken/wiskundeonderwijs*, CD-[beta] Press, 1994. 82, 85
- [Gra04] ———, *Local instruction theories as means of support for teachers in reform mathematics education*, *Mathematical thinking and learning* **6** (2004), no. 2, 105–128. 78
- [H⁺90] E.W. Hart et al., *Implementing the standards. teaching discrete mathematics in grades 7-12.*, *Mathematics Teacher* **83** (1990), no. 5, 362–67. 1
- [Hav07] F. Havet, *Graph colouring and applications*, Habilitationa diriger des recherches, Université de Nice-Sophia Antipolis (2007). 32
- [Hea49] P.J. Heawood, *Map-colour theorem*, *Proceedings of the London Mathematical Society* **2** (1949), no. 1, 161–175. 44
- [Hen06] A. Hennessy, *Grobner bases with applications in graph theory*, Ph.D. thesis, 2006. 38
- [HS65] J. Hartmanis and R.E. Stearns, *On the computational complexity of algorithms*, *Transactions of the American Mathematical Society* **117** (1965), no. 285-306, 1240. 20
- [Kel04] A. Kelly, *Design research in education: Yes, but is it methodological?*, *The journal of the learning sciences* **13** (2004), no. 1, 115–128. 79
- [Ker] A. Kerckhoffs, *La cryptographie militaire. journal des sciences militaires*, IX (38) **5**. 47
- [KH91] M.J. Kenney and C.R. Hirsch, *Discrete mathematics across the curriculum, k-12. 1991 yearbook.*, ERIC, 1991. 1
- [KL12] A.E. Kelly and R.A. Lesh, *Handbook of research design in mathematics and science education*, Routledge, 2012. 73
- [Kra07] J. Kramer, *Is abstraction the key to computing?*, *Communications of the ACM* **50** (2007), no. 4, 36–42. 158

- [Kru56] J.B. Kruskal, *On the shortest spanning subtree of a graph and the traveling salesman problem*, Proceedings of the American Mathematical society **7** (1956), no. 1, 48–50. 66
- [LCZ07] Peter Liljedahl, Egan Chernoff, and Rina Zazkis, *Interweaving mathematics and pedagogy in task design: A tale of one task*, Journal of Mathematics Teacher Education **10** (2007), no. 4-6, 239–249. 83
- [Lei79] F.T. Leighton, *A graph coloring algorithm for large scheduling problems*, Journal of research of the national bureau of standards **84** (1979), no. 6, 489–506. 32
- [LF01] Pier Luigi Ferrari, *Understanding elementary number theory at the undergraduate level: A semiotic approach*, Learning and teaching number theory: Research in cognition and instruction **2** (2001), 97–115. 139, 140
- [Lin10] Y. Lindell, *Introduction to coding theory, lecture notes*, 2010. 52
- [LS90] M. Livingston and Q.F. Stout, *Perfect dominating sets*, Citeseer, 1990. 69
- [LZ08] S. Larsen and M. Zandieh, *Proofs and refutations in the undergraduate mathematics classroom*, Educational Studies in Mathematics **67** (2008), no. 3, 205–216. 82
- [McM97] M. McMahon, *Social constructivism and the world wide web-a paradigm for learning*, ASCILITE conference. Perth, Australia, vol. 327, 1997. 80
- [MKB03] B. D. McCandliss, M. Kalchman, and P. Bryant, *Design experiments and laboratory approaches to learning: Steps toward collaborative exchange*, Educational Researcher **32** (2003), no. 1, 14–16. 85
- [MNvdA06] Susan McKenney, Nienke Nieveen, and Jan van den Akker, *Design research from a curriculum perspective*, Educational design research (2006), 67–90. 6

- [MS⁺94] B.M.E. Moret, H.D. Shapiro, et al., *An empirical assessment of algorithms for constructing a minimum spanning tree*, DIMACS Monographs in Discrete Mathematics and Theoretical Computer Science **15** (1994), 99–117. 66, 67
- [PCotEU06] Parliament and European Council of the European Union, *Recommendation of the european parliament and of the council, of 18 december 2006, on key competences for lifelong learning*, 2006. 9
- [PGC15] S. Prediger, K. Gravemeijer, and J. Confrey, *Design research with a focus on learning processes: an overview on achievements and challenges*, ZDM **47** (2015), no. 6, 877–891. 74, 77
- [PK09] K. Powell and C.J. Kalina, *Cognitive and social constructivism: Developing tools for an effective classroom*, Education **130** (2009), no. 2, 241–251. 80
- [Plo13] T. Plomp, *Educational design research: An introduction*, Educational design research (2013), 11–50. 74, 76
- [PN07] T. Plomp and N. Nieveen, *An introduction to educational design research*, Proceedings of the seminar conducted at the East China Normal University, Shanghai (PR China), 2007, pp. 23–26. 74, 76
- [Pri57] R.C. Prim, *Shortest connection networks and some generalizations*, Bell system technical journal **36** (1957), no. 6, 1389–1401. 66
- [RFR97] J.G. Rosenstein, D. Franzblau, and F. Roberts, *Dimacs series in discrete mathematics and theoretical computer science: Discrete mathematics in the schools. (vol. 36)*, American Mathematical Society and National Council of Teachers of Mathematics (1997). 1
- [Ros11] K. Rosen, *Discrete mathematics and its applications 7th edition*, McGraw-Hill Science, 2011. 27
- [RSA78] R. Rivest, A. Shamir, and L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM **21** (1978), no. 2, 120–126. 49

- [SC12] L. P. Steffe and P. Cobb, *Construction of arithmetical meanings and strategies*, Springer Science & Business Media, 2012. 79
- [Sch82] Alan H Schoenfeld, *Some thoughts on problem-solving research and mathematics education*, *Mathematical problem solving: Issues in research* (1982), 27–37. 83
- [Sch99] E. Schaefer, *An introduction to cryptography and cryptanalysis*, Santa Clara University, Available from <http://math.scu.edu/~eschaefe/crylec.pdf>, 1999. 45
- [Sha80] S. Sharan, *Cooperative learning in small groups: Recent methods and effects on achievement, attitudes, and ethnic relations*, *Review of educational research* **50** (1980), no. 2, 241–271. 150
- [Sim95] M. A. Simon, *Reconstructing mathematics pedagogy from a constructivist perspective*, *Journal for research in mathematics education* (1995), 114–145. 79
- [Sma03] N.P. Smart, *Cryptography: an introduction*, McGraw-Hill New York, 2003. 45
- [Sti95] D. R. Stinson, *Cryptography, Theory and Practice*, CRC Press, 1995. 45
- [Str91] L. Streefland, *Realistic mathematics education in primary school: On the occasion of the opening of the freudenthal institute*, 1991. 82, 85
- [Tie73] A. Tietäväinen, *On the nonexistence of perfect codes over finite fields*, *SIAM Journal on Applied Mathematics* **24** (1973), no. 1, 88–96. 57
- [TTD⁺99] David Tall, Michael Thomas, Gary Davis, Eddie Gray, and Adrian Simpson, *What is the object of the encapsulation of a process?*, *The Journal of Mathematical Behavior* **18** (1999), no. 2, 223–241. 141
- [Tur36] A.M. Turing, *On computable numbers, with an application to the entscheidungsproblem*, *J. of Math* **58** (1936), 345–363. 20

- [VdHPD14] M. Van den Heuvel-Panhuizen and P. D., *Realistic mathematics education*, Encyclopedia of mathematics education, Springer, 2014, pp. 521–525. 134
- [VG12] E. Von Glasersfeld, *A constructivist approach to teaching*, Constructivism in education, Routledge, 2012, pp. 21–34. 79
- [VL71] J.H Van Lint, *Nonexistence theorems for perfect error-correcting codes*, American Mathematical Society, 1971. 57
- [VL82] J.H. Van Lint, *Introduction to coding theory*, vol. 86, Springer, 1982. 52
- [Vyg78] L. Vygotsky, *Mind in society: The development of higher mental process*, 1978. 81
- [Vyg87] ———, *Zone of proximal development*, Mind in society: The development of higher psychological processes **5291** (1987), 157. 81, 149
- [W⁺01] D.B. West et al., *Introduction to graph theory*, vol. 2, Prentice hall Upper Saddle River, 2001. 27
- [WL08] K. Weber and S. Larsen, *Teaching and learning group theory*, Making the connection: Research and teaching in undergraduate mathematics education **73** (2008), 139. 82
- [ZG97] Rina Zazkis and Chris Gunn, *Sets, subsets, and the empty set: Students? constructions and mathematical conventions*, Journal of Computers in Mathematics and Science Teaching **16** (1997), no. 1, 133–169. 141
- [ZT01] Jeffrey M Zacks and Barbara Tversky, *Event structure in perception and conception.*, Psychological bulletin **127** (2001), no. 1, 3. 86, 144
- [Zuc92] L. Zuccheri, *Crittografia e statistica nella scuola elementare*, Insegnamento della Matematica e delle Scienze Integrate **15** (1992), 19–38. 11