mathematics

MDPI

Article

An Analysis of the Influence of Graph Theory When Preparing for Programming Contests

Cristina Jordán [1], Jon Ander Gómez [2] and J. Alberto Conejero [3,*]

[1] Instituto Universitario de Matemática Multidisciplinar & ETS Ingeniería Informática, Universitat Politècnica de València, 46022 València, Spain; cjordan@mat.upv.es
[2] PRHLT Research Centre & ETS Ingeniería Informática, Universitat Politècnica de València, 46022 València, Spain; jon@dsic.upv.es
[3] Instituto Universitario de Matemática Pura y Aplicada & ETS Ingeniería Informática, Universitat Politècnica de València, 46022 València, Spain
* Correspondence: aconejero@upv.es

Academic Editor: Lokenath Debnath
Received: 27 October 2016; Accepted: 13 January 2017; Published: 20 January 2017

Abstract: The subject known as Programming Contests in the Bachelor's Degree in Computer Engineering course focuses on solving programming problems frequently met within contests such as the Southwest Europe Regional Contest (SWERC). In order to solve these problems one first needs to model the problem correctly, find the ideal solution, and then be able to program it without making any mistakes in a very short period of time. Leading multinationals such as Google, Apple, IBM, Facebook and Microsoft place a very high value on these abilities when selecting candidates for posts in their companies. In this communication we present some preliminary results of an analysis of the interaction between two optional subjects in the Computer Science Degree course: Programming Contests (PC) and Graphs, Models and Applications (GMA). The results of this analysis enabled us to make changes to some of the contents in GMA in order to better prepare the students to deal with the challenges they have to face in programming contests.

Keywords: algorithmic; graph theory; programming; programming challenges; the Southwest Europe Regional Contest (SWERC)

1. Introduction-Context

The companies involved in Information and Communication Technologies have now begun to give greater importance to the results obtained by students in programming contests than the average marks they obtain throughout their university careers. It is obvious that the results obtained in these contests are generally considered good evidence of the level of the students' skills in relation to their classmates, and the contests themselves are good training for future job interviews. Readers interested in obtaining information about the selection processes employed by large companies such as Google, Yahoo, Apple and Microsoft, are referred to [1,2], while more detailed information on training for programming job interviews can be found in [3,4]. We can also recommend Poundstone's excellent books on unusual questions that may be set as a challenge to candidates during selection processes [5,6].

The subject of Programming Contests (PC) in the School of Informatics (ETSINF) of the Universitat Politècnica de València is designed to help graduates to master the programming problems they will be expected to solve during the process of job selection for software companies. Some of the largest multinationals, Google, Apple, Facebook, IBM and Microsoft are behind the most important international programming contests. Not only these companies, but many others also select their future software engineers from among the participants in these programming contests. The syllabus of

this subject covers the main aspects of algorithms, which the students must be able to use and combine appropriately for solving problems. The book *Programming Challenges* [7] is used as the guideline in the subject PC, and books *Introduction to Algorithms* [8] and *The Algorithm Design Manual* [9], both are used as the main references in this subject.

The subject of Graphs, Models and Applications (GMA) focuses on modelling the problems that can be solved by algorithms, some of whose implementations are dealt with in PC. This means that some problems can be studied within both subjects; in GMA from the modelling standpoint and in PC by finding the solution using an appropriately programmed and applied algorithm.

This paper describes preliminary results of an ongoing study based on the assessments of students who had taken both subjects and had also participated in programming contests. These assessments are compared with those of another group of students who had only taken one of the subjects in the same year.

In Section 2 we review the structure of the SWERC contest, which is regarded in ETSINF as the model for programming contests. In Section 3 both subjects are placed within the framework of the Bachelor's Degree in Computer Engineering course syllabus. The description of the data collection used for the subsequent analysis is described in Section 4, and the results of the analysis are given in Section 5. Finally, our conclusions are given in Section 6, together with some future lines of research.

Thanks to the analysis carried out, we were able to identify the contents of Graph Theory that the students considered to be most useful for preparing them to participate in programming contests.

2. The Southwest Europe Regional Contest

Programming contests are usually either organized by the Computer Science faculties themselves or by large multinationals such as those cited above. For some years now, the ETSINF has assiduously taken part in the International Collegiate Programming Contest, the most senior of these competitions, organised by the Association for Computer Machinery (ACM) and sponsored by IBM.

The structure of this contest is similar at each of the different levels; the participants can be either teams or individual competitors who have to solve a number of problems. The winner is the one who comes up with the best solution, or the one who took the shortest time in the case of a draw. It should be remembered that contestants that offer incorrect solutions to a problem for which a satisfactory solution is eventually found will suffer penalties for the time used in the process.

The generally acknowledged basic text books required to train for these contests include the manual [7], and the above mentioned main references [8,9]. A crucial role is played by the the train sites freely available in Internet. The most used in the subject PC is the problem repository of the University of Valladolid [10], which is also one of the first on-line judges and problem repositories. Other train sites are CodeForces [11], Topcoder [12], and USACO Gate [13]. The first one is getting an increased popularity during recent years, and the USACO Gate is recommendable for beginners because the problems are shown to each user in an increasing difficulty as he/she solves new problems. For further information on Graph Theory and the algorithms cited in the present paper see, among others, [8,14–25].

To solve the problems, the contestants are given a PC with the Linux operating system installed and configured for basic programming in Java and C++. All the PCs are connected to a central computer that acts as the judge. The contestants have five hours to solve the problems, and when they consider that they have found the right solution, they send it to the judge, who decides whether or not it is correct and assesses the efficiency of the algorithm proposed by the contestants (within the available resources and time limits). Any incorrect solutions are given a 20-min time penalty, but only when the correct solution has been found. Further details on the structure and organisation of this event can be found at [26].

We outline an overview of the types of problems related with Graph Theory (and the algortihms that can contribute to arrive to the solution after a suitable modeling):

- Shortest path problem (Dijkstra and Floyd algorithms) [17,23–25].
- Short Spanning tree (Kruskal algorithm) [17,18,24,25].
- Orientability (Hopcroft-Tarjan algorithm) [17].
- Accesibility (BFS and DFS algorithms) [17,23–25].
- Finding a Eulerian tour (Hierholzer algorithm) [17,24,25].
- Finding a Hamiltonian tour (Dirac algorithm, closure algorithm) [17,24].
- The travelling salesperson problem (algorithms to find hamiltonian cycles of low weight).
- The matching problem in a bipartite weighted graph (Kuhn Munkres algorithm) [16–18,25].
- Maximum flow/minimum capacity on directed graphs (with restrictions on the nodes, with low capacity in each arc, with more than one source/ sink) [14,21,22].
- Coloration (the greedy coloration algorithm) [23,24].
- The timetable problem [16].

3. Syllabus and Elective Subjects

When we are faced with a real problem and use the Graph Theory approach to solve it, the first step is to decide which of the concepts in the formulation is to be represented by a vertex and which by an edge or arc. This decision is of course related to the context of the problem, since in situations that appear to be similar, the nodes and edges can represent different concepts, according to the objective required. When the problem has been modelled in the form of a graph, each of the individual pieces of information, conditions, circumstances, etc. that it contains is transformed into concepts pertaining to the theory. Once it has been converted into an abstract problem, finding a solution depends on the solver's expertise in using Graph Theory. When the solution has been found, the concepts are changed back into the real terms of the initial context.

The process of transforming a real problem into an abstract one for the purpose of finding a solution using a specific mathematical theory is known as modelling and can either be very simple or very complicated, since the result of the modelling can be an already studied standard problem (so that it will be enough to apply the known solution), similar to a standard problem (so that an already known method will have to be modified to fit the circumstances), or simply a new problem that will require certain degree of ingenuity in order to achieve an imaginative solution. It is of course evident that the greater the student's knowledge of modelling, the greater his chances of finding an appropriate solution.

Obtaining a solution to these problems mainly depends on the individual's mastery of the contents of the subjects of Programming and Algorithms. In the present Bachelor's Degree in Computer Engineering course in the ETSINF, these are covered in the subjects known as Programming (12 ECTS in the first year), Data Structures and Algorithms (4.5 ECTS in 2nd year), Algorithms (4.5 ECTS in 3rd year, Computation itinerary), and an elective subject worth 4.5 ECTS in the 4th year known as Programming Contests (PC). The latter subject is used to revise and go deeper into previously learned concepts as well as to teach specific new algorithms. The evaluation of this subject is based on solving problems in programming contests from a list of progressively difficult problems given to the students at the beginning of each year. As in real contests, the solutions they propose are judged automatically.

In addition to mastering the above mentioned subjects, students must also be adept in certain mathematical procedures in order to successfully solve the problems, including Discrete Mathematics (1.5 ECTS—1st year) oriented to Graph Theory, Statistics (6 ECTS—1st year) and an elective subject on mathematical modelling known as Graphs, Models and Applications (4.5 ECTS—4th year).

The basic objective of PC is to learn how to program efficiently in a limited period of time (when solving a wide range of problems), mainly using the Graph Theory. In GMA, one of the objectives is to use algorithms to solve practical problems by transforming them into abstract problems within the context of Graph Theory. Even though both these subjects deal with very different concepts, the focus given to GMA is found to be a helpful support for solving the problems met within PC.

The content of PC can be grouped in the following blocks (in many cases already seen in other subjects): data structure and sorting variants; arithmetic, algebra and number theory; backtracking;

graphs (in only two sessions of one and a half hours); dynamic programming, and lastly reticles, grids and geometry. Similarly, GMA can be grouped as follows: generalities, matchings, flows and networks, Hamiltonian graphs, Eulerian graphs, and colouring.

Since it is quite frequent that problems related with Graph Theory appeared in the competition, some of the students who took part in contests recently proposed to PC and GMA teachers that those who studied both subjects should be encouraged to consider the same contents from the different points of view of both mathematical modelling and programming. Taking this into consideration, some of the lecturers in the subjects that approached problem solving from both directions scheduled a number of sessions, after which the students were given a range of problems suitable for treatment in this way and whose solution could contribute to evaluating both subjects simultaneously.

For fostering the participation of the students taken only one of the subjects, they were also allowed to do exercises of the subject in which they were not enrolled, as part of the evaluation of the subject they were taken. This last option enable teachers to increase the scope of the curriculum of each subject with complimentary resources. Further information on this innovation can be obtained from [27].

4. Methods and Materials

After completing both subjects and holding a local programming contest in the Faculty, an online questionnaire was drawn up to obtain the students' opinion of the usefulness of studying both subjects in order to prepare them for programming contests. Responses were obtained from 5 of the 8 students who took GMA only, 10 of the 24 who took PC only and 6 of those who took both subjects. Each group received a different version with the appropriate questions. The students gave their responses on a Likert scale of 1–5: 1—Strongly Disagree, 2—Disagree, 3—Neither Agree or Disagree, 4—Agree, and 5—Strongly Agree.

5. Results

Below we give the results of the questionnaires for each subject, grouped by those who studied only one subject and those who took both subjects.

5.1. Programming Contests

Firstly, all the students generally agreed that the time given to Dynamic Programming should be increased, and selected Backtracking as the second most important subject for in-depth studies. The next subject in importance was Data Structures and Sorting Variants, especially in the opinion of those who took both subjects, probably due to the improvement in the Graphs contents seen in GMA, see Table 1. Those who took both subjects were also fairly satisfied with the results they obtained in the contest, even if they had not won. The average score was 3.4 out of 5 with $\sigma = 1.14$, although they gave a score of 4.2, with $\sigma = 0.83$, to the value of PC for preparing students for local contests.

Table 1. Average assessment values and standard deviation of the importance of the contents of Programming Contests (PC) by students who took PC only and those who took both PC and Graphs, Models and Applications (GMA).

Contents	Students' Assessment (Only PC) Avg./σ	Students' Assessment (PC-GMA) Avg./σ
Data structures and sorting variants	3.4/0.54	3/0.89
Arithmetic, algebra and number theory	3/0.70	2.83/0.75
Backtracking	3.2/0.44	3.33/0.52
Graphs	3/0.77	3.5/1.22
Dynamic programming	2.8/0.44	3.66/0.52
Reticles, grids and geometry	2/0.70	3.16/0.75

Those who only took PC were less satisfied with their performance in the contest, with an average of 2.66 out of 5 and $\sigma = 1.15$. Most of them thought that if they had also taken GMA they would have

done better (average of 4.6 out of 5 with $\sigma = 0.55$). The reasons given for not taking GMA included the fact of already having enough credits or that they had to take other subjects related to their Final Degree Project.

5.2. Graphs, Models and Applications

Students who only took GMA chose this subject as they found it interesting, although they were not greatly interested in programming. None of these finally took part in the local contest but generally had an increased interest in maths and appreciated their usefulness after having taken the subject.

Those who took both PC and GMA were deeply interested in programming. In fact, half of them would have liked to have gone deeper into the algorithms they had seen in GMA, for which they would have needed more credits.

After completing the GMA subject, the students had an enhanced perception of it. In spite of the fact that it was inherently oriented towards solving real problems, those who had only taken this subject were of the opinion that it had given them a greater appreciation of the usefulness of maths, with a score of 3.8 and $\sigma = 1.30$, while those who had done both subjects gave it a much higher score of 4.75 out of 5, with $\sigma = 0.52$. In the latter group, the students were mostly in agreement with the usefulness of this subject as training for taking part in programming contests, which they scored at 4.83 out of 5 with $\sigma = 0.41$. The mean scores of the students as regards the contents that should be given more class time can be seen in Table 2. Among those who only took GMA, there was no general agreement about the contents that needed more in-depth treatment, which can be interpreted as meaning that the subject is already adequately structured.

Table 2. Average and standard deviation values of the importance of GMA contents for students who took GMA only and those who studied both PC and GMA.

Contents	Students' Assessment (only GMA) Avg./σ	Students' Assessment (CP-GMA) Avg./σ
Generalities	2.8/1.30	2.66/1.03
Matchings	3/0.71	3.16/0.75
Networks	3/0.71	3.33/0.82
Eulerian graphs	3.6/0.55	2.83/0.41
Hamiltonian graphs	3.4/0.89	3.16/0.75
Colouring	3.2/0.45	3.16/0.98

However, those who took both subjects and entered for the programming contest thought it was necessary to devote more time to matchings and networks. The students were also questioned about which maths content they missed most throughout the course and most were in agreement that the least attention had been given to combinatorics, which in the current syllabus is reduced to practically half an ECTS in Discrete Mathematics in the first year.

6. Conclusions

This paper has shown some preliminary results on the usefulness of combining one eminently practical subject and another more theoretical one in the second semester of the fourth year of the Bachelor's Degree in Computer Engineering course with a view to better preparing students to compete in the work market. These first results are, by now, of limited significance according to the variances, but provide a little insight on how effective could be to connect both subjects in order to facilitate the quest of a job.

It should be pointed out that even though the contents of both subjects may not be directly applicable to many of their normal work activities, they have been found to be critical for gaining access to the job market, especially in the case of the better known multinational companies. It should also be remembered that one of the main aims of any degree course is to train future graduates to be able to continue learning by themselves for the rest of their lives, and that this is one of the abilities that companies look for when assessing the results of programming contests. In this regard, in spite of

the growing interest of firms in programming skills and the use of modeling to solve problems, their credits have been reduced in compulsory subjects in the present syllabus as compared to previous study plans for the Bachelor's Degree in Computer Science.

Finally, we wish to emphasize that the collaboration between both subjects consisted of considering various problems from the perspective of each one and of increasing the number of problems in which both could be applied. In fact, this collaboration has been extended beyond the classroom, as the lecturers of both subjects have cooperated in preparing the students selected to represent the Computer Science Faculty in the annual SWERC contest.

We believe that this experience is a good example of how collaboration among the lecturers of different departments can considerably enrich the quality of teaching in the subjects involved while at the same time improving the students' training and reducing the dispersion of the contents.

Acknowledgments: The authors are grateful to the University Administration for supporting this initiative and sponsoring the teams that took part in the SWERC in recent years, also to the students of both subjects who suggested this collaboration.

This project was funded by the Vicerrectorado de Estudios y Calidad Académica of the Universitat Politècnica de València. PIME-B08: *Modelos de la Teoría de Grafos aplicados a problemas de competiciones de programación*.

Author Contributions: Cristina Jordán and J. Alberto Conejero developed the work in the subject of GMA. Jon Ander Gómez did it in the subject of CP. All the authors contribute to the design of the study, its analysis, and to write the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bock, L. *Work Rules!*; Hachette Book Group: New York, NY, USA, 2015.
2. Laakman McDowell, G. *The Google Résumé: How to Prepare for a Career and Land a Job at Apple, Google or Any Top Tech Company*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2010.
3. Aziz, A.; Tsung-Hien, L.; Amit, P. Elements of Programming Interviews: The Insider'S Guide; Avaliable online: https://books.google.com/books?hl=zh-CN&lr=&id=y6FLBQAAQBAJ&oi=fnd&pg=PP2&dq=Elements+of+Programming+Interviews:+The+Insider%E2%80%99S+Guide&ots=AsPqkQAazl&sig=8C6L-unD9bo-PXz-5CRy_d_xFnw#v=onepage&q=Elements%20of%20Programming%20Interviews%3A%20The%20Insider%E2%80%99S%20Guide&f=false (accessed on 18 January 2017).
4. Laakman McDowell, G. *Cracking the Coding Interviw: 189 Programming Questions & Solutions*, 6th ed.; CareerCup: Palo Alto, CA, USA, 2015.
5. Poundstone, W. *How Would You Move Mount Fuji?: Microsoft's Cult of the Puzzle: Microsoft's Cult of the Puzzle—How the World's Smartest Companies Select the Most Creative Thinkers*; Little, Brown and Company: New York, NY, USA; Boston, MA, USA, 2005.
6. Poundstone, W. *Are You Smart Enough to Work at Google? Fiendish Puzzles and Impossible Interview Questions From the World'S Top Companies*; Oneworld Publications: London, UK, 2012.
7. Skiena, S.S.; Revilla, M. *Programming Challenges: The Programming Contest Training Manual*; Springer: New York, NY, USA, 2003.
8. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. *Introduction to Algorithms*; MIT Press: Cambridge, MA, USA, 2009.
9. Skiena, S.S. *The Algorithm Design Manual*; Springer: London, UK, 2008.
10. ACM Contest Problems Archive. Universidad de Valladolid. Available online: https://uva.onlinejudge.org/ (accessed on 12 December 2016).
11. Codeforces. Available online: http://codeforces.com/ (accessed on 12 December 2016).
12. Topcoder. Available online: https://www.topcoder.com/what-can-you-do/algorithms-analytics/ (accessed on 12 December 2016).
13. USA Computing Olympiad. Available online: http://train.usaco.org/usacogate (accessed on 12 December 2016).
14. Ahuja, R.K.; Magnanti, T.L.; Orlin, A.B. *Network Flows*; Prentice Hall: Upper Saddle River, NJ, USA, 1993.
15. Bang-Jensen, J.; Gutin, G. *Digraphs: Theory, Algorithms and Applications*; Springer: London, UK, 2002.
16. Bondy, J.A.; Murty, U.S.R. *Graph Theory*; Springer: London, UK, 2008.

17. Chartrand, G.; Oellermann, O.P. *Applied and Algorithmic Graph Theory*; McGraw Hill: New York, NY, USA, 1993.
18. Christofides, N. *Graph Theory: An Algorithmic Approach (Computer Science and Applied Mathematics)*; Academic Press Inc.: Orlando, FL, USA, 1975.
19. Conejero, J.A.; Jordán, C. Aplicaciones de la Teoría de Grafos a la vida Real I. 2015. Available online: www.edx.org/course/aplicaciones-de-la-teoria-de-grafos-la-upvalenciax-tgv201x-1-0 (accessed on 12 December 2016).
20. Conejero, J.A.; Jordán, C. Aplicaciones de la Teoría de Grafos a la vida Real II. 2015. Available online: www.edx.org/course/aplicaciones-de-la-teoria-de-grafos-la-upvalenciax-tgv201x-2-0 (accessed on 12 December 2016).
21. Dolan, A.; Aldous, J. *Networks and Algorithms*; Wiley: Hoboken, NJ, USA, 1993.
22. Evans, J.R.; Minieka, E. *Optimization Algorithms for Networks and Graphs*; Dekker: Toronto, ON, Canada, 1992.
23. Foulds, L.R. *Graph Theory Applications*; Springer: New York, NY, USA, 1992.
24. Gross, J.L.; Yellen, J.J. *Handbook of Graph Theory. Discrete Mathematics and Its Applications*; CRC Press: Boca Raton, FL, USA, 2004.
25. Jordán, C.; Torregrosa, J.R. *Introducción a la Teoría de Grafos y sus Algoritmos*; Universitat Politècnica de València: València, Spain, 1996.
26. Gómez, J.A.; Planells, J.; Casanova, A.; Galiano, M.; Llorens, M.; Moltó, G.; Marqués, F.; Prieto, N.; Álvaro, F.; Barella, A.; et al. Competiciones de programación. Estímulo y salida laboral para los alumnos. In Proceedings of the XIX JENUI, Castellón de la Plana, Spain, 10–12 July 2013; Universitat Jaume I: Castellón de la Plana, Spain, 2013; pp. 161–166.
27. Jordán, C.; Gómez, J.; Calvo, M.; Conejero, J.A. Modelos de la teoría de grafos aplicados a problemas de competiciones de programación. In Proceedings of the In-Red 2016, València, Spain, 7–8 July 2016; Universitat Politècnica de València: València, Spain, 2016.