Olympiads in Informatics, 2017, Vol. 11, 41–53

DOI: 10.15388/ioi.2017.04

Teaching Graphs for Contestants in Lower-Secondary-School-Age

Ágnes ERDŐSNÉ NÉMETH
Batthyány High School, Nagykanizsa, Hungary
Doctoral School, Faculty of Informatics, Eötvös Loránd University, Budapest, Hungary
e-mail: erdosne@blg.hu

Abstract. A didactically interesting question is how to familiarize lower secondary school children with abstract modelling tools – especially graphs – and how problem-solving paradigms can be developed in their minds while using the structure of graphs and the operations defined on them. In this paper, we make an overview of introducing graph models and basic graph algorithms to lower-secondary-school-age pupils and to older students, who are new to programming.

Keywords: graph theory, teaching informatics in primary and secondary schools, preparing for algorithmic contests.

1. Overview

A didactically and methodologically interesting question is how problem solving can be developed in children's minds while teaching computational thinking (CT) for all or computer science (CS) for competitors; what steps and tasks lead through from understanding the idea to its professional usage; at what age and abstraction level they can use specific tools and methods.

In this paper, we examine these questions in connection with maybe the most important modelling tool: graphs and also the operations defined on them.

There are many problems in real life where we can draw graphs naturally: like networks of roads, social networks, family tree through relationships, kinship relations and supply chains. There are problems where drawing graphs doesn't appear so naturally, but after simplification we use them for modelling and understanding the core of the problem, like map-colouring, counting the number of elements with certain properties in a specific set, packing and sorting efficiently, describing the states of games. Using graphs is a very simple and intuitive tool for modelling ideas in the solution of various mathematical and CS problems.

The aim of this paper is to examine graphs from a methodologist's perspective, so it is not intended to introduce new algorithms but rather to discover the applicability and limits of using graphs as a modelling tool and data structure for students in lower-secondary-school-age (K 6–10).

2. Place of Graphs in Algorithms Textbooks

There are many textbooks about algorithms. They discuss all relevant algorithms sequentially and directly as they are written for university students. The structure of these books is wide-ranging. There are not two textbooks, in which the place of graphs is the same in the sequence of algorithms. However the important role of graphs – as a modelling tools, data structure and problem-solving strategy – agrees in them:

- Cormen: separate chapters about graph data structures and algorithms.
- Dasgupta: two whole chapters plus three subchapters.
- Halim: one chapter out of nine is about graph algorithms and many subchapters are about modelling something with graph structures.
- Kleinberg: almost each chapters contains a subchapter about graphs.
- Rónyai: two whole chapters out of overall ten are exclusively about graphs.
- Sedgewick: one chapter out of overall six is about graph algorithms and 2 subchapters are about trees.
- Skiena: four whole chapters about graph algorithms and four subchapters about graphs as data structures.

These textbooks are almost useless for primary and secondary school students because of their advanced mathematical and informatical contents. Some parts of them may be useful, but just in upper secondary, for contestants preparing for IOI. However, they are very good resources for teachers about some important themes: data structures, specific procedures and different wording of practice tasks.

3. Teaching Graphs as a Part of Teaching CT for All – Facing to Teaching Graphs for Competitors

Teaching graphs ideally begins in primary school in mathematics and informatics lessons through using the idea without labelling it.

3.1. *Mathematics Lessons*

Teaching graphs in primary and secondary school begins in mathematics lessons in most of the countries, while the pupils create a simple model for combinatorial tasks, like permutations, variations and combinations. In these cases, they just draw an appropriate figure without labelling it as a graph.

3.2. *CSUnlugged Activities*

CSUnplugged − CS without a computer is a collection of activities, which can be used to spark interest in and give motivation to learn CT and CS in primary schools. There are four different games among these activities based on graphs.
So the next four games can be used for modelling a problem with a graph without naming the parts of the graph – and solving a small problem on the model. These games can be altered to use a slightly larger but still manageable number of elements in a related model. In this case, the elements of the graph can be named and a specific algorithm should be given to solve the problem on the specific graph model. They can be used to help develop a deep understanding of the whole idea through games for contestants later, in secondary school.

The first example in a book about graphs is an undirected and weighted graph in *Activity 9: The Muddy City (Minimal Spanning Tree)* Fig. 1. The task is:

- At first finding the shortest route between two given point.
- Second time to find the road network with minimum cost.

This is an excellent exercise in the primary school for counting and summarizing numbers systematically on the model. For the secondary-school-age children on a larger example it is a good model for understanding the concept of finding the minimal spanning tree algorithm.

The second one is an example of directed and unweighted graph in the *Activity 12: Treasure Hunt (Finite State Automata)* Fig. 2. The task is to find the way to the treasure

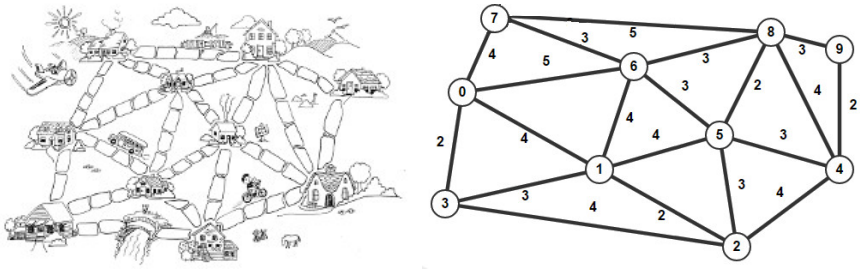


Fig. 1. Muddy City task and its model.

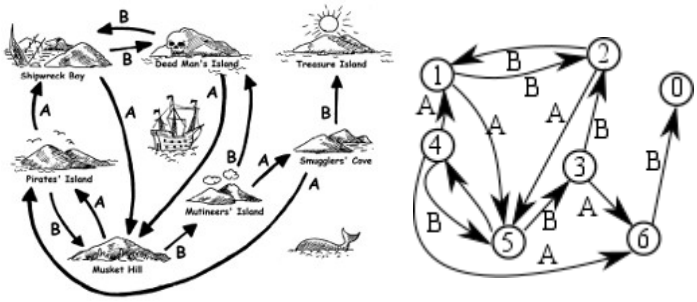


Fig. 2. Treasure Hunt task and its graph model.

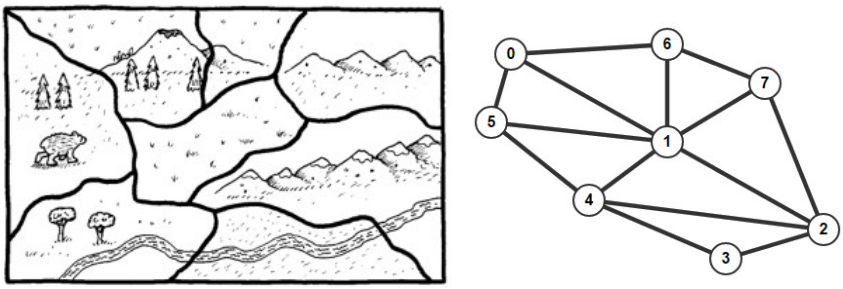


Fig. 3. The Poor Cartographer task and its graph model.

on a Treasure Island and trying to find more than one route. It is a very good example of simplification via model – the vertices (the stages of a journey) are simple numbers without drawing complex island's maps and the cruises are directed edges between these numbers.

There is another type of graph, undirected and unweighted in *Activity 14: The Poor Cartographer (Graph Coloring)* Fig. 3. The task is to colour the countries on this map with as few colours as possible, but make sure that no two bordering countries are the same colour. This task is nearly a simple colouring book with an additional special condition. It is worth to give more than one copy to the children to attempt the appropriate colouring.

For older pupils colouring is not such an interested task, but finding the smallest number of different colours needed for a specific graph model could be exciting activity.

There is also an undirected and unweighted graph model in *Activity 15: Tourist Town (Dominating Set)* Fig. 4.

The task is to mark some of the vertices in a graph model in such a way that all other vertices are at most one edge away from the marked ones. It is a good example of a mind-breaker, which is hard to solve, but it is easy to check if the answer is correct.

There may be more than one correct solution – it is a new idea in informatics tasks while learning CT. This activity leads on to many extensions and variations.

It is an appropriate task to think over the algorithm's execution time when the size of a modelling graph is increasing. It is very important for children to see problems with more than one correct solution and with more elements of the modelling graphs.

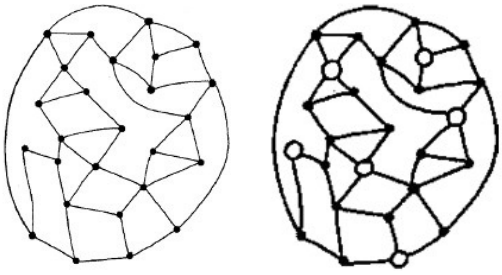


Fig. 4. Tourist Town task's model and its solution**.**

3.3. *BEBRAS*

The tasks of the BEBRAS competition is used to develop CT and to check thinking skills and abstraction level. In K 5–6 all the tasks are marked difficult about graphs, indicating that children have to use non-standard and not commonly used tools (*Shortest paths, Beaver logs, Beaver the Alchemist, Irrigation system, Many friends*). These tasks are about counting something on a task that can be described with an undirected graph model.

All the tasks using a graph model in K 7–8 are marked either medium or hard (*Street stones, Super Power Family, Pirate Hunters, Word chains, Ceremony, Storm proof network, Dice, Hobbit beaver, Cinema, Necklace, Mr. Beaver's shopping*) on undirected and sometimes weighted graphs. The children have to count or summarize a specific property of a graph. In K 5–8 the figure of the model is included in the task's description.

The upper-secondary-school-age pupils have to draw the model itself either it is undirected/directed and unweighted/weighted. In the medium and hard tasks the elements of the graphs are expressively without the exact definitions (*The magician, Word Jumble, Encounter, Expensive bridges, Control of rivers, Neighbourhood*).

The BEBRAS competition's tasks create interest and give motivation to appropriate children to learn CS.

4. Teaching CS for Contestants

While learning programming (and on programming contests), lower-secondary-school-age children use basic data structures (integer, boolean, one- and two-dimensional array of integers, simple strings) and basic algorithms can be applied for various problems with various wordings. Choosing, selecting, counting, searching, summarizing, selecting maximum/minimum, sorting, separating into groups are basic algorithms. Children must be proficient on these algorithms.

In the textbooks the basic tasks' wording are very easy and boring. It is possible to make them more interesting with different wording and by making the modelling part interesting as well.

Stages of solving algorithmic tasks are as follows:

- Understand the problem.
- Create an appropriate model.
- Choose the right data structure.
- Select the right algorithm.
- Verify constraints.
- Implement and test.

In addition to the basic tasks, there are many problems where children can apply advanced algorithms creatively and do the above mentioned steps over and over.

In lower-secondary-school-age the tasks can be made more complicated by phrasing them in a way that hides the underlying model, in order to make the problem in-

teresting. Using graphs for simplification and modelling makes these difficult-looking problems easy to understand. The difficulty is supposed to be finding the right model and deriving the correct algorithm. After drawing the model, the corresponding data structure can be as simple as a one- or two-dimensional array, on which a simple, basic algorithm can be used. This method is useful for contestants, whose mind's abstraction level is higher than that of other talented students of their age. In this way graphs can be introduced as a new concept through basic algorithms, to make the comprehension of more complicated algorithms easier at a later time.

4.1. *Tasks*

In informatics contests in Hungary, classic graph tasks can be in every round, so the contestants have to be ready to solve them. New tasks are usually worded in a way that highly resembles the wording of former tasks. These tasks come out with another question circularly, as follows. (N is the number of vertices and M is the number of edges.)

Each of the following chapters describe a teaching block. Each block is a unit of teaching, designed specially for primary school pupils, described in the intended order. At this level trees are to be handled as general graphs, no special property of them is used.

4.2. *Using Graph as a Model, Implementing with Edge List*

In this stage of learning the concept of vertex, edge, directed and undirected graphs can be introduced, we use unweighted graphs for modelling. Storing the graph structure is in a two-dimensional matrix as an adjacency list. The basic algorithms, like counting, searching, selecting can be used on an array.

If the modelling graph is undirected (Fig. 5), then in some cases the edge list has to be duplicate for easier handling (each edge is stored twice, and ordered by the endpoints).

If the modelling graph is directed one (Fig. 6), the order of pairs is important.

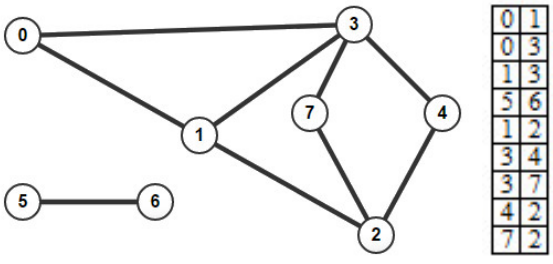


Fig. 5. Undirected, unweighted graph and edge list.

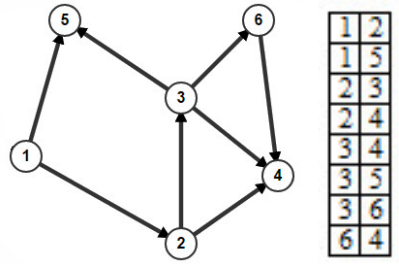


Fig. 6. Directed, unweighted graph described with edge list.

Example tasks are as follows:

- **Family-tree** – *The parents of some people – identified with numbers between 1 and N – are given with an ordered pair of numbers. (1 ≤ N, M ≤ 100 000)*
  - Who are the parents of the people with the given number?
  - Who are the brothers and sisters of the person with the given number? (Somebody is one's brother or sister if they have minimum one common parent.)
- **Acquaintances** – *On a social portal there are N people. The acquaintances between the people are given and always mutual. (1 ≤ N, M ≤ 100 000)*
  - Who are acquaintances of a given people?
  - Are two given people acquainted?
- **Rumour** – *Rumours are formed by some people telling something to others and then those people passing on the rumour. People – identified by numbers between 1 and N – were asked about which person told them the rumour. It is described by an ordered pair of numbers: from to.*
  - Who passed the rumour to a given person?
  - To whom did a given person pass on the rumour?
- **Customs** – *In the middle ages certain towns made merchants who travelled through them pay customs. Of course going through towns was unavoidable, as major roads went through them. For all towns we know a list of neighbouring towns, to which merchants can directly travel, and the amount of customs that has to be paid. (1 ≤ N, M ≤ 100 000)*
  - Which city's customs is the most/less?
  - Which cities are in a one-way-connection of a city with the given number?
- **Gangs –** *The police of Johannesburg knows every criminals in the city. They know that two criminals belongs to the same gang, if and only if they have committed a crime together at least once. Gang members knows each other if they have committed a crime together. The criminals are identified by numbers from 1 to N. The police stores the identifiers each pair of criminals, who have committed a crime together. There may be lonely criminals too. (1 ≤ N, M ≤ 100 000):*
  - Who didn't commit any crime?
  - How many gangs are there?
  - Who are in the same gang with a given person?
  - Who committed a crime together with a given person?

- **Transportation** – *A company has three types of yards: production, warehousing and sales. It has N yards and none of them does two different activities. We know the transportation routes between these yards: routes can be from the production yards to a warehouse or a sales yard, or from the warehouse to another warehouse or to a sales yard. (1 ≤ N, M ≤ 100 000):*
  - Which are the production yards? (Production yard have only outgoing edges).
  - Which are the warehousing yards? (Warehousing yard has incoming and outgoing edges also).
  - How many sales yards are there? (Sales yards only have incoming edges).

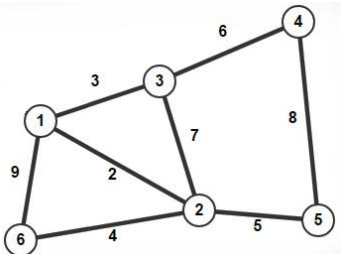
4.3. *Using Graph as a Model, Implementing with Adjacency Matrix*

Sometimes the graph model is weighted and easier algorithms can be coded on adjacency matrix. In this stage of studying programing, the transformation between the adjacency matrix and edge list is a challenge, regardless of the modelling graph being directed/undirected or weighted/unweighted (Fig. 7, Fig. 8).

In the next examples the children have to decide, which storage method is more efficient and which basic algorithm has to be chosen:

Villages – *The lengths of the roads in a county are given. The villages are identified by numbers between 1 and N. The roads are defined with a list of the numbers of the villages, which are connected by that road. (1 ≤ N, M ≤ 10 000):*

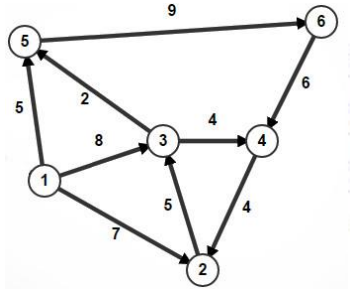
- How long is the longest/shortest road between villages?



	1	2	3	4	5	6
1	0	2	3	0	0	9
2	2	0	7	0	5	4
3	3	7	0	6	0	0
4	0	0	6	0	8	0
5	0	5	0	8	0	0
6	9	4	0	0	0	0

1	2	2
1	3	3
1	6	9
2	3	7
2	5	5
2	6	4
3	4	6
4	5	8

Fig. 7. Undirected, weighted graph described with adjacency matrix and edge list.



	1	2	3	4	5	6
1	0	7	8	0	5	0
2	0	0	5	0	0	0
3	0	0	0	4	2	0
4	0	4	0	0	0	0
5	0	0	0	0	0	9
6	0	0	0	6	0	0

1	2	7
1	3	8
1	5	5
2	3	5
3	5	2
3	4	4
4	2	4
5	6	9
6	4	6

Fig. 8. Directed, weighted graph described with adjacency matrix and edge list.

- Which villages are connected with the longest/shortest road(s)?
- Which village is farthest away from the nearest neighbour?
- Which village is the nearest to a given village?

4.4. *Counting the Degree of Vertices*

There are a lot of difficult tasks, in which after the simplification and modelling, the task is to count, how many edges lead to a vertex, or in directed graphs how many edges goes to and from a vertex. The algorithm is easy, but the children have to think about how the edges can be counted.

Example tasks are as follows:

- **Zoo** – *In the zoo the walking paths between animals are known. The entrance is numbered with 0 and the animals are numbered from 1 to N. The walking paths are defined with two animals' number which cages are linked by a walking path. ($1 \leq N, M \leq 100\,000$):*
  - How many of the cages are in a dead end of walking paths?
  - Which animals can be visited directly from the most others?
- **Villages** – *As before. A dead-end village is a village, which is connected with other villages by just one road. ($1 \leq N, M \leq 100\,000$):*
  - Which villages leads the most number of roads to? (If there is more than one, then list all!)
  - How many dead-end villages are there?
  - Which villages are dead-end villages?
- **Family-tree:** *($1 \leq N, M \leq 100\,000$):*
  - Who has the most/less children? (If there is more than one, then list all!)
  - Who hasn't got any children? (If there is more than one, then list all!)
- **Rumour:** *($1 \leq N, M \leq 100\,000$):*
  - Who didn't pass on the rumour?
  - Who started the rumour?
  - Who passed on the rumour to the most others?
- **Acquaintances:** *($1 \leq N, M \leq 100\,000$):*
  - Who has the most/less acquaintances?
- **Gangs:** *($1 \leq N, M \leq 100\,000$):*
  - How many lonely criminals exist?
  - Who is a lonely criminal?
- **Towns:** *($1 \leq$ No. of towns $\leq 10\,000$, $1 \leq$ No. of triangles $\leq 100\,000$)*
  There are N towns in a convex polygon shape country. The country was divided to triangle counties, the vertices of which are towns. The towns are identified by numbers from 1 to N. The number of triangles is known. Every triangle is defined by giving the three corresponding towns.
  - How many cities are on the border of the country?
  - How many neighbours does each city have cities have?

4.5. *Tasks on Special Graphs*

There are a lot of special cases, when solving the problem is based on a specific attribution of a graph, like:

- In a tree the root and levels.
- Complete graph.
- Complementer graph.
- In a directed graph the super source and the super sink.

Some examples:

- **Villages**: *(1 ≤ N, M ≤ 100 000):*
  - What is the length of the longest way to the dead-end villages, for which there is no edge to get off the path ie. the only options at any intermediate village are continuing on the path and turning back?
- **Family-tree:** *(1 ≤ N, M ≤ 100 000):*
  - Who has the most grandchildren? (If there is more than one, then list all!)
  - Who hasn't got any grandchildren? (If there is more than one, then list all!)
- **Acquaintances:** *(1 ≤ N, M ≤ 10 000):*
  - Which pairs have common acquaintances?
  - Who knows each other without any other common acquaintances?
  - Which pairs have common acquaintances without knowing each other?
- **Gangs:** *(1 ≤ N ≤ 50, 1 ≤ M ≤ 100):*
  - How many gangs are there?
  - Who has the most connection in a certain gang?
  - What is the number of members of the largest gang?
  - Who are the member of the largest gang?
  - Who are the members of a gang, where just the head knows everybody else?
  - How many "totally ordered" gangs are, where everybody knows everybody?
- **Trip** – *On a map (of size N*M) the altitude of each gridpoint above the sea level is known. The time of a hiking between two adjacent points is (1+absolut value of the height's difference). A point cannot be reached directly from another adjacent point if the difference of their high is more than given number H.*
  - What is the minimum time to reach a given Q point from a given Ppoint?
  - Give a shortest path between a given Q and P points?
- **Castle** – *There are octagonal rooms in a castle, connected with square shaped hidden doors. The visitor has to pay different amounts when entering an octagonal room. Using the hidden doors cost the same each time.*
  - What is the minimum time to reach a given point from another given point?
  - What is a way with minimum time to reach a given point from another given point?

4.6. *Floyd-Warshall Algorithm and its Variations*

After teaching recursion and dynamic programing concept, we can make a detour into the world of graph algorithms with a classical Floyd-Warshall algorithm and its variations. It is used to solve:

- General case of the path existence on an unweighted graphs.
- General case of shortest paths problems on weighted graphs.

It is frequently used in other problems, as long as the input graph is small. It needs to store the graph with adjacency matrix, but the code is only four lines: three nested loop. It is understandable after learning dynamic programing concept, the concept of Floyd-Warshall is on the Fig. 9.

It has many variations: maximin, minimax, safest path, minimin, maximax [Horváth2].

- **Yard:** *A company wants to sell its product in different towns (numbered from 1 to N) and looks for one town, where it is optimal to build a warehouse. The distances of the roads between towns are given. (1 ≤ N, M ≤ 400):*
  - What is the shortest distances to all the towns from a given town?
  - Which is the best place to plant a yard? (The distance of the farthest town is the shortest).
- **Trip** – *(1 ≤ N ≤ 400):*
  - What is the minimum time to reach from a given point P to a given point Q?
  - What is the path with minimum time from a given point P to a given point Q?
- **Castle:** *(1 ≤ N ≤ 400):*
  - What is the minimum cost to reach B from A? (for every pair).
  - What is the way with the minimum cost from A to B?
- **Duty** – *(1 ≤ N ≤ 400):*
  - What is the minimum cost to reach a given town from another given town?
  - What is a way with minimum cost to a given town from another given town?
- **Acquaintances:** *(1 ≤ N ≤ 1000):*
  - Which pairs have only common acquaintances? (at least all of them have one acquaintance).

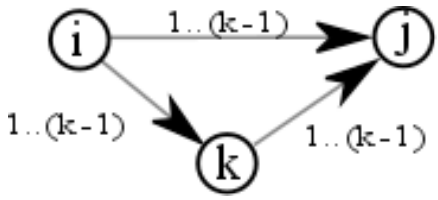


Fig. 9. i » j ⇔ i » k & k » j.

4.7. *Further Studies*

With these simple tasks the children can be familiar with the concept of a graph and its components, as a modelling tool and as a thinking tool as well. Next level of their studies is more complicated data structures, like list and stacks, following by more complex algorithms from this topic, like graph traversal, spanning trees and more advanced algorithms.

5. Conclusions

Some antecedents of the graph concept might come up in earlier mathematical and informatical studies. If you are aware of this, introducing graphs as a new modelling tool and using graph algorithms as a problem-solving strategy is much easier.

We think, if you want to teach graph algorithms you could start the whole process in the lower-secondary-school-age and circularly, and then when returning to it in higher and higher levels your students would be familiar with this concept.

Finally, graphs and graph algorithms would not be magic for the contestants, just a useful modelling tool and problem-solving strategy.

References

Bebras–International Contest on Informatics and Computer Fluency (2007–2017). `http://bebras.org; http://www.beaver-comp.org.uk/; http://informatik-biber.de/archiv/`

CSUnplugged_OS_2015_v3.1. `http://csunplugged.org`

Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C. (2001). *Introduction to Algorithms*. MIT Press, 2nd edition.

Dagienė, V., Stupurienė, G. (2016). Bebras – a sustainable community building model for the concept based learning of informatics and computational thinking. *Informatics in Education,* 15(1), 25–44.

Dasgupta, S., Papadimitriou, C.H., Vazirani, U.V. (2006). *Algorithms*. McGraw-Hill.

Halim, S., Halim, F. (2014). Competitive Programming 3. The New Lower Bound of Programming Contests. `http://cpbook.net/`

Horváth, Gy. (2004). A programozási versenyek szerepe az oktatásban. In: *INFOÉRA Konferencia.* `http://www.infoera.hu/infoera2004/eaok/horvathgyula.pdf`

Horváth, Gy., Horváth, Gy., Zsakó, L. (2016). Variations on a classic task. In: *XXIXth DIDMATTECH*. 72–78.

Kleinberg, J., Tardos, É. (2006). *Algorithm Design*. Addison-Wesley.

MESTER online task archive and judge system. `https://mester.inf.elte.hu`

Rónyai, L., Ivanyos, G., Szabó, R. (1999). *Algoritmusok*. Typotex.

Sedgewick, R., Wayne, K. (2011). *Algorithms*, Fourth Edition. Addison-Wesley.

Skiena, S.S. (2008). *The Algorithm Design Manual*. Springer-Verlag, 2nd edition.

Szlávi, P., Zsakó, L. (2012). Informatika oktatása. *TÁMOP-4.1.2 A1 és A2 könyvei, ELTE IK*.

Zsakó, L. (2012). Variations for spanning trees. *Annales Mathematicae et Informaticae*, 33, 151–165.

Á. Erdősné Németh teaches mathematics and informatics at Batthyány Lajos High School in Nagykanizsa, Hungary. A lot of her students are in the final rounds of the national programming competitions, some on CEOI and IOI. She is a PhD student in the Doctoral School of Faculty of Informatics, Eötvös Loránd University in Hungary. Her current research interest is teaching computer science for talented pupils in primary and secondary school.