Constructing Knowledge in Graph Theory and Combinatorial Optimization

EVA MILKOVÁ
Department of Informatics and Quantitative Methods
University of Hradec Králové
Rokitanského 62
CZECH REPUBLIC
eva.milkova@uhk.cz http://lide.uhk.cz/fim/ucitel/milkoev1/

Abstract: - Graph theory and combinatorial optimization are powerful tools for teachers allowing them to develop logical thinking of students, increase their imagination and make them familiar with solutions to various practical problems. The paper offers some ideas how to make the teaching of these branches of applied mathematics and computer science more understandable and attractive. A case study is described which introduces the minimum spanning tree problem, its remarkable origin, and then its relation to the other problems. The approach used for teaching and learning graph theory and combinatorial optimization can serve as an inspiration for instruction of other subjects as well.

Key-Words: - Graph Theory, Graph Algorithms, Minimum Spanning Tree Problem, Breadth-First-Search, Education.

1 Introduction

The theory of graphs and combinatorial optimization are wonderful, practical disciplines. The main aim of the subject *Graph Theory and Combinatorial Optimization* is to develop and deepen students' capacity for logical thinking. Students should gain an appropriate level of competence in graph theory and graph algorithms.

On the one hand, usually, there are more methods which can be used for solving the same graph-problem, while on the other hand, using effective modifications of one algorithm, we can devise methods of solving various other tasks. Thus *it is important to make students familiar with certain algorithms in contexts* to be able to get deeper insight into each problem and entirely understand it. Well-prepared students should be able *to describe various practical situations with the aid of graphs*, solve the given problem expressed by the graph, and translate the solution back into the initial situation.

To make the subject *Graph Theory and Combinatorial Optimization* more attractive to the students, we can *use puzzles as motivation* to graph concepts and graph problems. Well, let us quote from the book Graph theory 1736-1936. "The origins of graph theory are humble, even frivolous. Whereas many branches of mathematics were motivated by fundamental problems of calculation, motion, and measurement, the problems which led to the development of graph theory were often little more than puzzles, designed to test the ingenuity rather than to stimulate the imagination. But despite the apparent triviality of such puzzles, they captured the interest of mathematicians, with the result that graph theory has become a subject rich in theoretical results of surprising variety and depth." [1]

In the subject *Graph Theory and Combinatorial Optimization* there is no problem in *illustrating the explained subject matter*. When preparing suitable illustrative graphs and using colors it is quite easy to emphasize the characteristics of the concepts. Colors also play a very important role in explaining graph-algorithms. Algorithms can be described as an edge-coloring or vertex-coloring process.

In this paper we present just a few ideas that have proved successful in constructing knowledge of students in these branches of applied mathematics and computer science.

2 Teaching in contexts

When teaching *Graph Theory and Combinatorial Optimization*, we always try to examine the given topic as thoroughly as possible and find a "bridge" to another topic. More precisely, our approach can be described in the following four steps:

1. When starting explanation of ***new subject matter***, we describe a particular problem with a *real life example and/or puzzle* to enable students to get an idea about its use.

2. If possible, we examine each concept and problem from more than one point of view and discuss *various approaches* to the given problem solution.
3. We let students practice and give their *own examples* describing the topic.
4. Using the constructed knowledge and *suitable modification* of the problem solution, we proceed to ***new subject matter***.

Let us illustrate the process in the following section 2.1 with a case study describing “bridges” between a topic dealing with the minimum spanning tree problem and a topic dealing with circles having the given properties.

Note:

A circle of the length k in the given graph G is a sequence $(v_0, e_1, v_1, ..., e_k, v_0)$, where $e_i = \{v_{i\text{-}1}, v_i\}$, $i = 1, ..., k\text{-}1$, $e_k = \{v_{k\text{-}1}, v_0\}$ and $v_i \neq v_j \; \forall_{i \neq j}$.

A tree is a connected graph without circles.

A spanning tree of the given connected graph $G = (V, E)$ is a tree $T = (V, E')$, where $E' \subseteq E$.

Examples

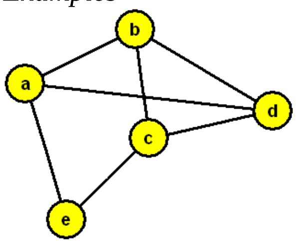


Fig. 1 Connected graph G

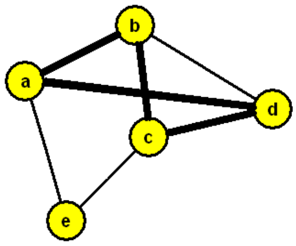


Fig. 2 Circle (a, b, c, d, a) of the length 4 in G

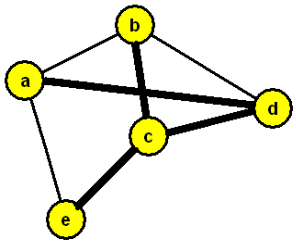


Fig. 3 A spanning tree of G

2.1 Case study

In the section we present a possible way of how to move from the Minimum Spanning Tree Problem (MSTP in short) to finding circles with the given properties

1st step: MSTP - introduction

Before we formulate this well-known problem using present terminology, we familiarize students with its remarkable historical background containing some interesting *practical applications*. Let us introduce the origins of the most important works connected with MSTP, the classical works, here as well (this part is almost the same as the part already published in [8]).

MSTP was formulated in 1926 by Czech mathematician Otakar Borůvka. He was introduced to the problem by his friend, Jindřich Saxel, an employee of the West_Moravian Powerplants. It was at that time that electrification of the south and west parts of Moravia was beginning (Remark. Moravia is a part of the Czech Republic), and Borůvka was asked for help in solving the problem. The challenge was *how and through which places to design the connection of several tens of municipalities in the Moravia region so that the solution was as short and consequently as low-cost as possible*.

Borůvka not only correctly stated this problem but also solved it. An adequate mathematical terminology in this area of mathematics had not been developed at that time, and thus the formulation and the proof of the correctness of the solution given in his paper [2] (in English *On a certain minimum problem*) was rather complicated. Otakar Borůvka formulated the problem in the following way:

“Given a matrix of numbers $r_{\alpha\beta}$ ($\alpha, \beta = 1, 2, \ldots n$; $n \geq 2$), all positive and pairwise different, with the exception of $r_{\alpha\alpha}=0$, $r_{\alpha\beta}=r_{\beta\alpha}$.

From that matrix a set of nonzero and pairwise different numbers should be chosen such that

1° for any p_1, p_2 mutually different natural numbers $\leq n$, it would be possible to choose a subset of the form $r_{p1c2}, r_{c2c3}, r_{c3c4}, \ldots, r_{cq\text{-}2cq\text{-}1}, r_{cq\text{-}1p2}$

2° the sum of its elements would be smaller than the sum of elements of any other subset of nonzero and pairwise different numbers, satisfying the condition 1°.”

In the theory not based on graph terminology it was really not easy to perform a correct formulation and proof of the procedure of the solution using a precise definition of the groups of numbers satisfying the above mentioned conditions 1° and 2°. Thus it is no wonder that Otakar Borůvka solved and proved the problem in 16 pages (5 pages solution, 11 pages proof).

However, he was convinced both about the importance of the work and about the essence of the algorithm. This is documented by the fact that Otakar Borůvka published, simultaneously with the paper [2], a short note [3] (*A contribution to the solution of a problem of economic construction of power-networks* in English), where he introduced a lucid description of the algorithm by means of geometric example of 40 cities (see Fig. 4 – Fig. 7). His formulation of the problem is written in a nearly contemporary style there:

"There are n points given on the plane (in the space) whose mutual distances are different. The problem is to join them through the net in such a way that

1. any two points are joined to each other directly or by the means of some other points,

2. the total length of the net would be the smallest."

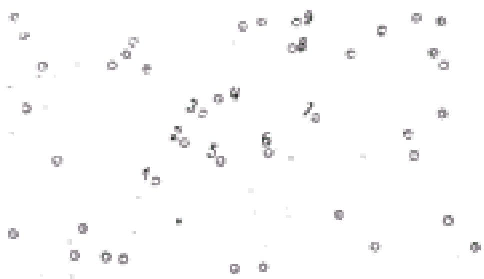


Fig. 4

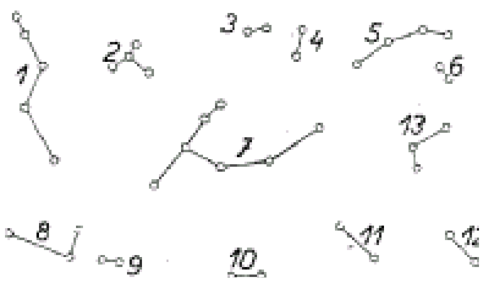


Fig. 5

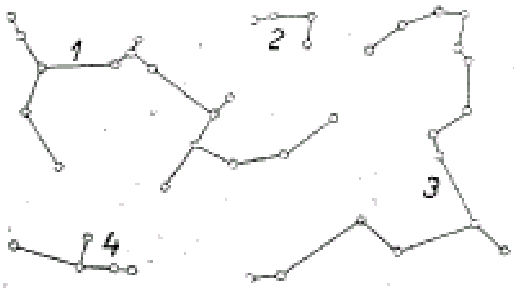


Fig. 6

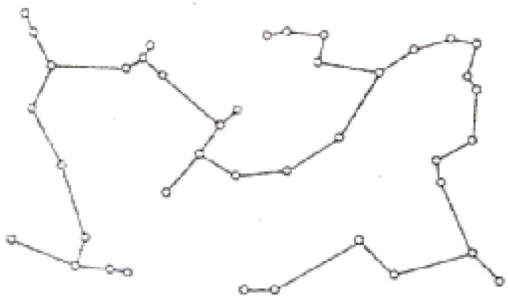


Fig. 7

Vojtěch Jarník, another Czech mathematician, quickly realized the novelty and importance of the problem after reading the Boruvka's paper. However the solution seemed to him very complicated. He started to think about another solution and very soon wrote a letter to Otakar Borůvka where he gave much easier elegant method of creating demanded construction. Consequently he published it in the article [5] (in English *On a certain minimum problem* with the subtitle *From the letter to Mr. Borůvka*).

Vojtěch Jarník added the following *geometric visual interpretation* at the end of his paper [5].

"We are given n balls numbered 1, 2, …, n which are joined pair wise by ½ n (n-1) sticks. Let $r_{a,b}$ be the mass of the stick joining a and b. Let the sticks be bent if necessary so that do not touch. From this system we want to remove some of the sticks so that the n balls hold together and the mass of the remaining sticks is small as possible."

Both Czech mathematicians preceded their fellow mathematicians by a quarter of a century. The enormous interest in this problem broke out with unusual vigour again after 1950 and at time was connected with the development of computers.

It is important to mention that all three above-mentioned articles were written in Czech, the Borůvka's first paper has a six page German summary fortunately (see thereinafter).

At that time Jarník's method was discovered independently several times more. Let us mention at least R.C. Prim who, just as the others, wasn't aware of Jarník's solution. Prim's solution is the same as Jarník's solution but he included a more detailed implementation suitable for computer processing.

The third solution of the problem different from the previous ones invented Joseph B. Kruskal in 1956 in his work *On the shortest spanning tree of a graph and the travelling salesman problem* (see [6]). Kruskal had opportunity to read the German Borůvka's summary. Let us quote a part from his reminiscence [7]:

"It happened at Princeton, in old Fine Hall, just outside the tea-room. I don't remember when, but it was probably a few months after June 1954.

Someone handed me two pages of very flimsy paper stapled together. He told me it was floating around the math department.

The pages were typewritten, carbon copy, and in German. They plunged right in to mathematics, and described a result about graphs, a subject which I found appealing. I didn't understand it very well at first reading, just got the general idea. I never found out who did the typing or why.

At the end, the document described itself as the German-language abstract of a 1926 paper by Otakar Borůvka."

Also Kruskal's algorithm has been discovered independently several times.

Note: The survey of the works devoted to the MSTP until 1985 is given in the article [4] and this historical paper is followed up in article [13] where also the first English translation of both Borůvka's papers is presented.

In the contemporary terminology the Minimum Spanning Tree problem can be formulated as follows (see e.g. [13]):

Given a connected undirected graph $G = (V, E)$ with n vertices, m edges and real weights assigned to its edges (i.e. $w: E \to \mathrm{R}$). Find among all spanning trees of G a spanning tree $T = (V, E')$ having minimum value $w(T) = \Sigma(w(e);\ e \in E')$, a so-called minimum spanning tree.

Example

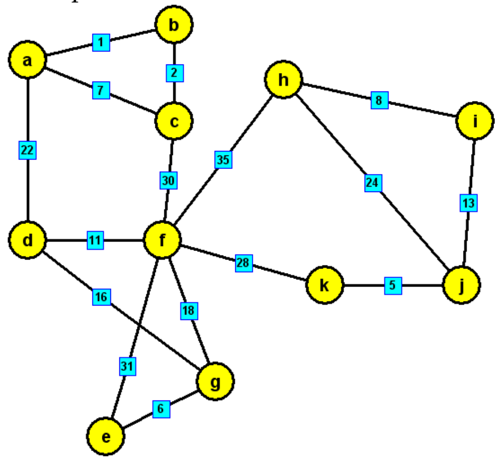


Fig. 8 Connected edge-weighted graph *G*

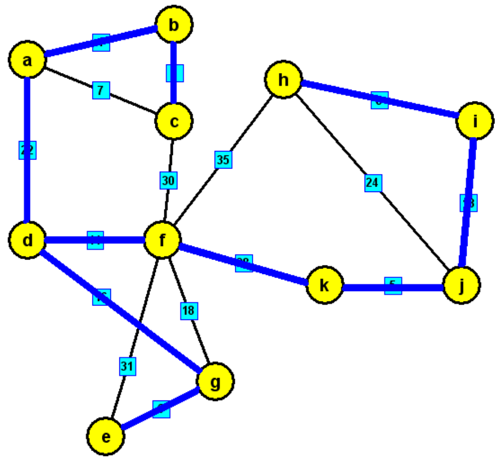


Fig. 9 Unique minimum spanning tree of *G*

2nd and 3rd steps: MSTP – various solutions

The Minimum Spanning Tree problem has wide applications; methods for its solution have produced important ideas in modern combinatorics and have played central role in the design of graph algorithms. Therefore MSTP is generally regarded as a cornerstone of Combinatorial Optimization.

We meet students familiar not only with Borůvka's and Jarník's algorithms but also with Prim's implementation of Jarník's solution and both Kruskal's algorithms solving the MSTP (see [9]). Each method brings an interesting approach to the problem and discussing the basic differences among solutions enables to get deeper insight into the problem.

We also discuss conditions when the shortest spanning tree is unique (note: the spanning tree on the figure 9 is unique because no two of the edge-weights are equal).

Students exercise all solutions not only on a graph given by figure but also on a graph represented by adjacency matrix (see [9]). They present their own practical applications of MSTP.

4th step: from MSTP to graph searching

Let us formulate here Jarník's solution of MSTP as an edge-coloring process.

<u>Jarník's algorithm</u>

1. Initially all vertices and edges of the given connected graph G, with n vertices and m edges, are uncoloured. Let us choose any single vertex and suppose it to be a trivial blue tree.
2. At each of $(n - 1)$ steps, colour the minimum-weight uncoloured edge, having one vertex in the blue tree and the other not, blue. (In case, there are more such edges, choose any of them.)
3. The blue coloured edges form a minimum spanning tree.

Example

On the graph G, represented by figure 10, let us illustrate Jarník's algorithm of the MSTP with initial vertex b (see Fig. 11 – Fig. 16). For clearer illustration of the consecutively obtained blue trees we also denote vertices by the color blue.

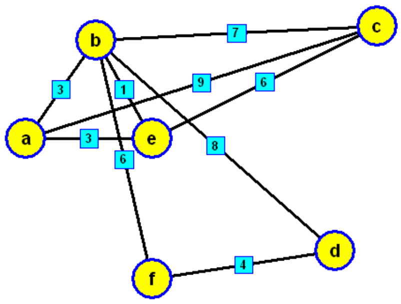


Fig. 10 Given graph G

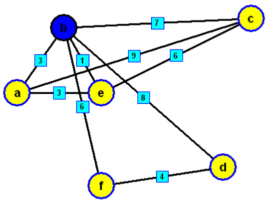


Fig. 11

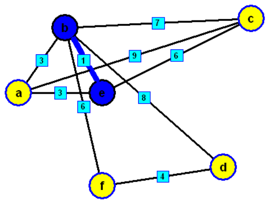


Fig. 12

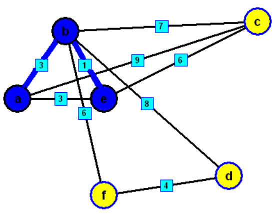


Fig. 13

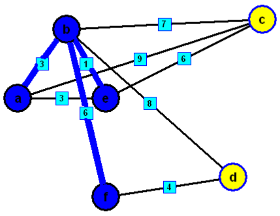


Fig. 14

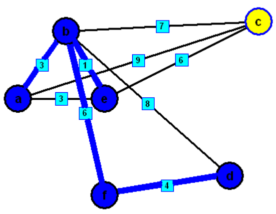


Fig. 15

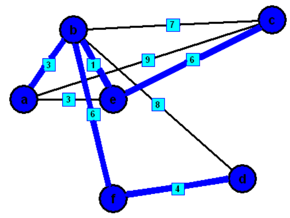


Fig. 16

It is evident that Jarník's solution spreads at each of $(n - 1)$ steps the only blue tree containing the initial vertex by the nearest vertex (we may assume $w(e)$ as length of e). Consecutive adding vertices into the blue tree can be understood as a consecutive search of them. Thus, using the following small *modification of Jarník's method*, we are able to easily describe the most used graph search algorithms, the Breadth-First-Search and Depth-First-Search.

<u>Breadth-First-Search</u>: At each of $(n - 1)$ steps we choose from the uncoloured edges, having one vertex in the blue tree and the other not, such an edge having the end-vertex being added to the blue tree as the first of all in blue tree vertices belonging to the mentioned uncoloured edges and colour it blue. (Remark: To identify the end-vertex we store vertices adding into the blue tree in the data structure queue (FIFO)).

<u>Depth-First-Search</u>: At each of $(n - 1)$ steps we choose from the uncoloured edges, having one vertex in the blue tree and the other not, such an edge having the end-vertex being added to the blue tree as the last of all in blue tree vertices belonging to the mentioned uncoloured edges and colour it blue. (Remark: To identify the end-vertex we store vertices adding into the blue tree in the data structure stack (LIFO)).

1st step - 4th step: From Breadth-First-Search to circles with the given properties

When solving a task where a consecutive searching of vertices and/or edges is demanded, we apply a searching algorithm. As for example: *we have a map of a town and want to visit subsequently all restaurants to find out where we can get the best beer*.

As we have already mentioned, the Breadth-First-Search algorithm next the Depth-First-Search algorithm belongs to the most used searching algorithms.

Let us formulate the Breadth-First-Search algorithm described at the end of the previous section more precisely.

Breadth-First-Search of vertices

1. Initially all vertices and edges of the given connected graph G, with n vertices and m edges, are uncoloured. Let us choose any single vertex, put it into FIFO, colour it blue and search it.

2. While FIFO is not empty do the following commands:

- choose the first vertex x in FIFO
- ***if*** there is an uncoloured edge $\{x, y\}$ ***then*** search and colour blue both the vertex y and the edge $\{x, y\}$, and put the vertex y into FIFO
  else remove the vertex x from FIFO

This description can easily be extended to get the algorithm consecutively searching not only vertices but also edges of a given connected graph (if the given graph is disconnected, we apply the algorithm on its components).

Breadth-First-Search of vertices and edges

1. Initially all vertices and edges of the given connected graph G, with n vertices and m edges, are uncoloured. Let us choose any single vertex, put it into FIFO, colour it blue and search it.

2. While FIFO is not empty do the following commands:

- choose the first vertex x in FIFO
- ***if*** there is an uncoloured edge $\{x, y\}$ ***then***
  if the vertex y is uncoloured, then search and colour blue both the vertex y and the edge $\{x, y\}$, and put the vertex y into FIFO
  else (i.e. if the ucoloured edge $\{x, y\}$ has both vertices already in the blue tree) search and colour the edge $\{x, y\}$ green
  else remove the vertex x from FIFO (i.e. remove x in the case that it isn‘t end-vertex of any uncoloured edge)

Applying the Breadth-First-Search algorithm on vertices of the graph represented by the figure 10 starting in the vertex b, the vertices can be searched in the order b, a, c, d, e, f and the blue spanning tree (see Fig. 17) will be obtained.

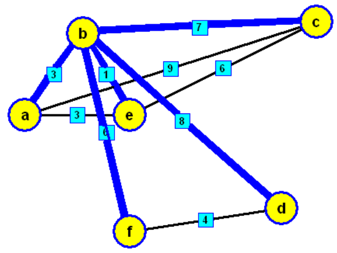


Fig. 17 Blue spanning tree obtained by the Breadth-First-Search starting in the vertex b

If we draw this blue tree as the rooted tree with the root b, we get so-called breadth-first-search tree appropriate to the Breadth-First-Search applied on the given graph starting with the vertex b.

Definition. Let G be a connected undirected graph, let v be a vertex of G, and let T_B be its spanning tree gained by the Breadth-First Search of G with the initial vertex v. The appropriate rooted tree (T_B, v) we call the **breadth-first-search tree** with the root v.

Applying the Breadth-First-Search algorithm on vertices and edges of the given graph the edges that do not appear in the breadth-first-search tree (i.e. the green edges) have special property.

Theorem. Let G be a connected undirected graph, let v be a vertex of G, and let (T_B, v) be the breadth-first-search tree with the root v. Then **the end-vertices of each green edge of G belong either to the same level or to the adjacent levels of (T_B, v)**.

Proof. Let $\{x, y\}$ be a green edge of G. We may assume that the vertex y was put into FIFO later than the vertex x (for otherwise we would go analogically). Let us denote the level of (T_B, v) where the vertex y lies by $h(y)$ supposing $h(v) = 0$.

Obviously, $h(z) \geq h(x)$ for each vertex z put into FIFO later than x and $h(z) \leq h(x) + 1$ for each vertex z put into FIFO before x. Hence, with regard to the assumption and because the vertices x and y are adjacent in G, $h(x) \leq h(y) \leq h(x) + 1$.

From the theorem we immediately get two important statements:

1) The length of the shortest path from the vertex v to the vertex y in G is equal $h(y)$.
2) There is a circle of odd length in G if and only if there is a green edge having both end-vertices in the same level of (T_B, v).

Example

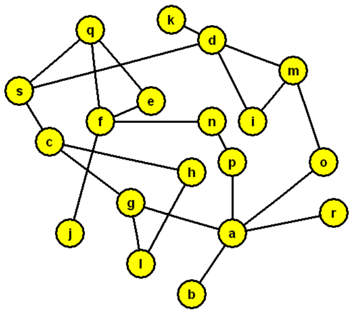


Fig. 18 Connected graph *G*

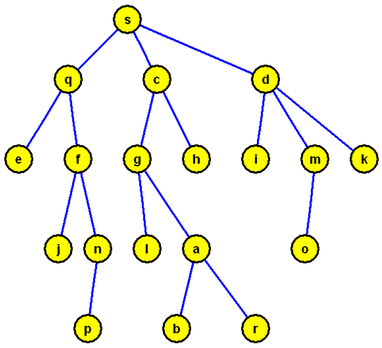


Fig. 19 A breadth-first-search tree of *G*

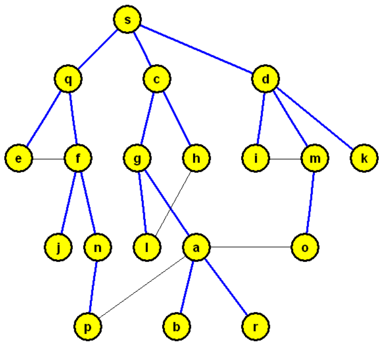


Fig. 20 The above breadth-first-search tree of *G* completed by green edges (thin lines)

If we carefully look at the figure 20, we can observe green edges not only regarding to levels of the tree, but also we can observe their end-vertices regarding to the sub-trees of the root *s* (i.e. the subtree with the root *q*, the subtree with the root *c* and the subtree with the root *d*) and obtain the other important statements concerning the root *s*. Generally,

3) There is a circle containing the vertex *v* in *G* if and only if there is a green edge having its end-vertices in different subtrees of (T_B, *v*).
4) There is a circle containing an edge {*v*, *w*} in *G* if and only if there is a green edge having one end-vertex in the subtree with the root *w* and the other end-vertex in another subtree of (T_B, *v*).

Using Breadth-First-Search and the mentioned statements, we are able to easily formulate appropriate algorithms concerning **circles with the given properties** as e.g. *algorithms determining if the given graph contains a circle of odd length (i.e. if it is bipartite or not); if there is a circle in the given graph containing the given vertex (edge respectively); the shortest circle containing the given vertex (edge respectively); determining the girth of the given graph (i.e. the shortest circle of the given graph) etc.*

Example

Let us have the graph represented by the following adjacency matrix.

	a	b	c	d	e	f	g	h	i
a			1	1					1
b				1	1	1			
c	1						1		1
d	1	1							
e		1					1		
f		1						1	
g			1		1			1	
h						1	1		
i	1		1						

a) Decide, if the graph is bipartite (i.e. if there does not exist a circle of odd length),

b) Decide, if there is a circle containing the vertex *g*.

c) If there are circles containing the vertex *g* determine one of the shortest ones.

Solution

a) We use the Breadth-First-Search starting with arbitrary vertex *r*, at each step we save by each vertex $v \neq r$ the information describing the ancestor of *v* and the level *h*(*v*), and when searching a green edge we determine if its both end-vertices are in the same level of the created breadth-first-search tree. If there is no such an edge, the graph is bipartite, otherwise it isn't.

FIFO	breadth-first-search tree	green edge
a	*a*	
a,*c*	*a*, *c*(*a*,1)	
a,*c*,*d*	*a*, *c*(*a*,1),*d*(*a*,1)	
a,*c*,*d*,*i*	*a*, *c*(*a*,1),*d*(*a*,1),*i*(*a*,1)	
c,*d*,*i*,*g*	*a*, *c*(*a*,1),*d*(*a*,1),*i*(*a*,1),*g*(*c*,2)	
c,*d*,*i*,*g*	*a*, *c*(*a*,**1**),*d*(*a*,1),*i*(*a*,**1**),*g*(*c*,2)	{***c***,***i***}

Both end-vertices of the edge {*c*,*i*} are in the same level, thus *the given graph is not bipartite*.

b) We use the Breadth-First-Search starting with the vertex g, at each step we save by each vertex $v \neq g$ the information describing the ancestor of v and the number of subtree in which the vertex v lies, and when searching green edge we determine if its both end-vertices are in different subtries of (T_B, g). If there is such an edge, there is a circle containing the vertex g in the given graph.

FIFO	breadth-first-search tree	green edge
$\boldsymbol{g}$	g	
$\boldsymbol{g},c$	$g,\ c(g,1)$	
$\boldsymbol{g},c,e$	$g,\ c(g,1),e(g,2)$	
$\boldsymbol{g},c,e,h$	$g,\ c(g,1),e(g,2),h(g,3)$	
$\boldsymbol{c},e,h,a$	$g,\ c(g,1),e(g,2),h(g,3),a(c,1)$	
$\boldsymbol{c},e,h,a,i$	$g,\ c(g,1),e(g,2),h(g,3),a(c,1),$ $i(c,1)$	
$\boldsymbol{e},h,a,i,b$	$g,\ c(g,1),e(g,2),h(g,3),a(c,1),$ $i(c,1),b(e,2)$	
$\boldsymbol{h},a,i,b,f$	$g,\ c(g,1),e(g,2),h(g,3),a(c,1),$ $i(c,1),b(e,2),f(h,3)$	
$\boldsymbol{a},i,b,f,d$	$g,\ c(g,1),e(g,2),h(g,3),a(c,1),$ $i(c,1),b(e,2),f(h,3),d(a,1)$	
$\boldsymbol{a},i,b,f,d$	$g,\ c(g,1),e(g,2),h(g,3),a(c,\mathbf{1}),$ $i(c,\mathbf{1}),b(e,2),f(h,3),d(a,1)$	$\{a,i\}$
$\boldsymbol{i},b,f,d$	$g,\ c(g,1),e(g,2),h(g,3),a(c,1),$ $i(c,1),b(e,2),f(h,3),d(a,1)$	
$\boldsymbol{b},f,d$	$g,\ c(g,1),e(g,2),h(g,3),a(c,1),$ $i(c,1),b(e,\mathbf{2}),f(h,3),d(a,\mathbf{1})$	$\{a,i\}$, $\{\boldsymbol{b},\boldsymbol{d}\}$

End-vertices of the edge $\{\boldsymbol{b},\boldsymbol{d}\}$ are in the different subtries, thus *there is the circle (g,e,b,d,a,c,g), of the length 6, containing the given vertex g*.

c) We use the Breadth-First-Search starting with the vertex g, at each step we save for each vertex $v \neq g$ the information describing the ancestor of v, the number of subtree in which the vertex v lies and the level $h(v)$. As soon as we find a green edge whose both end-vertices are at once in different subtries of (T_B, g) and in the same level of (T_B, g) we finish because we have found one of the shortest circles containing the vertex g.

However, if we find a green edge $\{v,w\}$ whose both end-vertices are in different subtries of (T_B, g) but $h(v) \neq h(w)$ (assume $h(v) < h(w)$) we mark this edge but continue searching vertices belonging to the level $h(v)$ to find out if there is an edge having both end-vertices at once in different subtries of (T_B, g) and in the same level of (T_B, g). If there is such an edge, then it determines one of the shortest circles containing the vertex g. Otherwise, if there is no such edge, than the edge $\{v,w\}$ determines one of the shortest circles containing the vertex g.

If there is no edge having both end-vertices in different subtries of (T_B, g), there is no circle containing the vertex g.

FIFO	breadth-first-search tree	green edge
$\boldsymbol{g}$	g	
$\boldsymbol{g},c$	$g,\ c(g,1,1)$	
$\boldsymbol{g},c,e$	$g,\ c(g,1,1),e(g,2,1)$	
$\boldsymbol{g},c,e,h$	$g,\ c(g,1,1),e(g,2,1),h(g,3,1)$	
$\boldsymbol{c},e,h,a$	$g,\ c(g,1,1),e(g,2,1),h(g,3,1),$ $a(c,1,2)$	
$\boldsymbol{c},e,h,a,i$	$g,\ c(g,1,1),e(g,2,1),h(g,3,1),$ $a(c,1,2),i(c,1,2)$	
$\boldsymbol{e},h,a,i,b$	$g,\ c(g,1,1),e(g,2,1),h(g,3,1),$ $a(c,1,2),i(c,1,2),b(e,2,2)$	
$\boldsymbol{h},a,i,b,f$	$g,\ c(g,1,1),e(g,2,1),h(g,3,1),$ $a(c,1,2),i(c,1,2),b(e,2,2),f(h,3,2)$	
$\boldsymbol{a},i,b,f,d$	$g,\ c(g,1,1),e(g,2,1),h(g,3,1),$ $a(c,1,2),i(c,1,2),b(e,2,2),f(h,3,2),d(a,1,3)$	
$\boldsymbol{a},i,b,f,d$	$g,\ c(g,1,1),e(g,2,1),h(g,3,1),$ $a(c,\mathbf{1},2),i(c,\mathbf{1},2),b(e,2,2),f(h,3,2),d(a,1,3)$	$\{a,i\}$
$\boldsymbol{i},b,f,d$	$g,\ c(g,1,1),e(g,2,1),h(g,3,1),$ $a(c,1,2),i(c,1,2),b(e,2,2),f(h,3,2),d(a,1,3)$	
$\boldsymbol{b},f,d$	$g,\ c(g,1,1),e(g,2,1),h(g,3,1),$ $a(c,1,2),i(c,1,2),b(e,\mathbf{2},\mathbf{2}),f(h,3,2),d(a,\mathbf{1},\mathbf{3})$	$\{a,i\},\underline{\{b,d\}}$
$\boldsymbol{b},f,d$	$g,\ c(g,1,1),e(g,2,1),h(g,3,1),$ $a(c,1,2),i(c,1,2),b(e,\mathbf{2},\boldsymbol{2}),f(h,\mathbf{3},\boldsymbol{2}),d(a,1,4)$	$\{a,i\},\underline{\{b,d\}}$, $\{\boldsymbol{b},\boldsymbol{f}\}$

The green edge $\{\boldsymbol{b},\boldsymbol{f}\}$ determines *the circle (g,e,b,f,h,g) of the length 5, which is the shortest circle in the given graph containing the vertex g.*

3 Amusing motivation to the topic

To find out if students have gained the needed ability to describe various situations with the aid of graphs, solve the given problem expressed by the graph, and translate the solution back into the initial situation, we also use puzzles. There are an endless number of enjoyable tasks, puzzles and logic problems in books like “Mathematics is Fun”, in riddles magazines and on the Internet. We have been looking for problems from these sources that can be efficiently solved with the help of graphs and introduce them in lectures devoted to the appropriate topic.

Let as illustrate two examples here (see also [10], [11], [12]). The first example illustrates a simple puzzle *Competition* used as a motivation to the concept graph and vertex degree. The second example is more interesting and also more difficult.

It describes the puzzle *Town* that we use when explaining the concept isomorphism.

Competition

An international company opened five positions for translators from the following languages: Russian, German, English, French and Spanish. Five candidates apply for the job.

- Mr. Smith can speak all 5 languages;
- Mr. Parker can speak English, French and Russian;
- Mr. Thomas can speak German and Russian;
- Mr. Brian can speak English and German;
- Mr. White can speak Russian and German.

Is the company able to provide all opened positions so that each of candidates would translate just from one language? If it is possible, propose the solution.

Solution

The situation can be easily represented by the following graph.

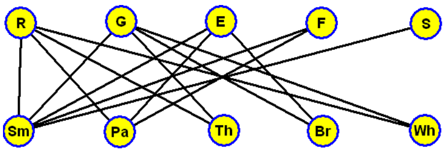


Fig. 21 Graph-representation of *Competition*

Looking at the graph on Figure 21, it is obvious that the vertex S with the degree 1 must be connected with the vertex Sm. Thus, at first, we color the edge $\{S, Sm\}$ blue and the other edges incident with the vertex Sm red to demonstrate that we have rejected them. Then, considering the graph with uncolored edges, it is obvious that the edge $\{F, Pa\}$ must be denoted by the color blue and hence the other edges incident with the vertex Pa by the color red, etc.

Finally, the blue edges represent two possible solutions (see the following figures).

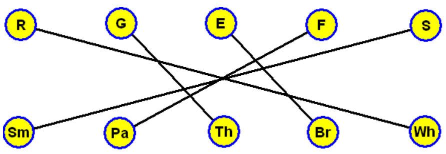


Fig. 22 The first solution of *Competition*

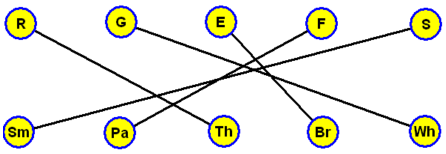
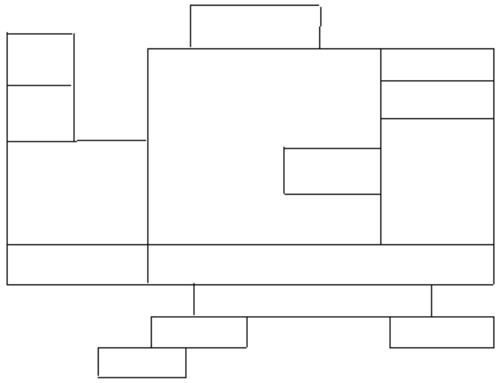


Fig. 23 The second solution of *Competition*

Towns

Try to place the names of towns Atlanta, Berlin, Caracas, Dallas, Lima, London, Metz, Nairobi, New York, Paris, Quito, Riga, Rome, Oslo, Tokyo into the frames of the given map so that no town shares any letter in its name with any towns in adjacent frames (neither horizontal nor vertical).



Solution

We advise students to describe both the map and also the relation between two towns that do not contain a same letter in their names, using graphs and find out the isomorphism between the graph representing the map and the subgraph of the graph representing the relation.

The given map can be represented by the following graph G_1 (see Figure 24).

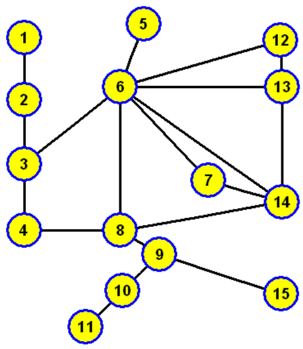


Fig. 24 Graph G_1, graph-representation of the map

The relation "be adjacent" defined as "town x is adjacent with town y if there is no same letter in their names" can be described by the graph G_2, whose vertices represent the towns and edges corresponding with the relation "be adjacent" (see Figure 25).

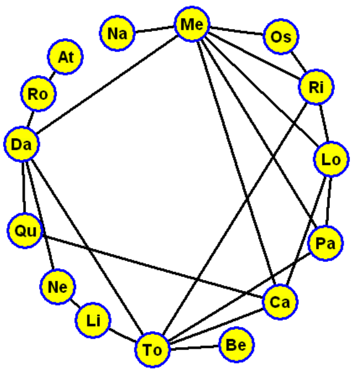


Fig. 25 Graph G_2, graph-representation of the above-described relation "be adjacent"

From both figures it is obvious that the vertex 6 with the degree 7 must correspond with the vertex *Me* (the only vertex with the degree bigger or equal 7). Let us draw the induced subgraph of the graph G_1 (of the graph G_2 respectively) containing the vertex 6 (the vertex *Me* respectively) and its neighbors (see Figure 26).

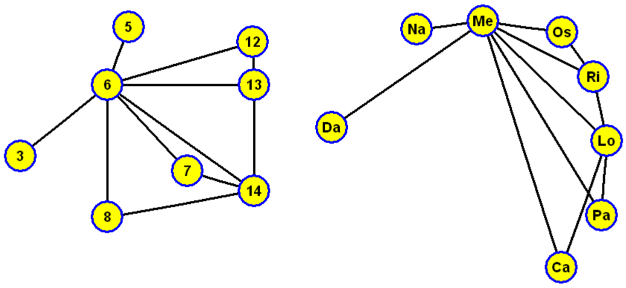


Fig. 26 Subgraph of the graph G_1 and the subgraph of the graph G_2

The given subgraphs are isomorphic. There exist the following two bijections characterizing isomorphism of the given subgraphs:

I) 3 → *Da*, 5 → *Na*, 6 → *Me*, **7 → *Ca*, 8 → *Pa***, 12 → *Os*, 13 → *Ri*, 14 → *Lo*

II) 3 → *Da*, 5 → *Na*, 6 → *Me*, **7 → *Pa*, 8 → *Ca***, 12 → *Os*, 13 → *Ri*, 14 → *Lo*

We choose the second bijection (compare degrees of vertices in the given graphs G_1 and G_2) for isomorphism of the graph G_1 and the subgraph of G_2 and, with help of the program *Graphs* (see thereinafter), we draw the graph G_2 using the picture similar to the figure of the graph G_1 (see Figure 27).

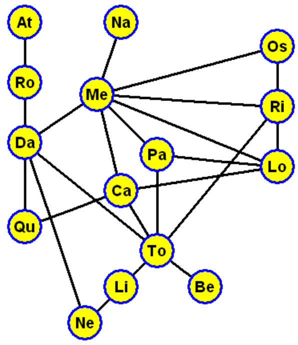


Fig. 27 Graph G_2 – another picture

Hence, the rest of the demanded isomorphism is obviously determined as follows: 1 → *At*, 2 → *Ro*, 4 → *Qu*, 9 → *To*, 10 → *Li*, 11 → *Ne*, 15 → *Be*

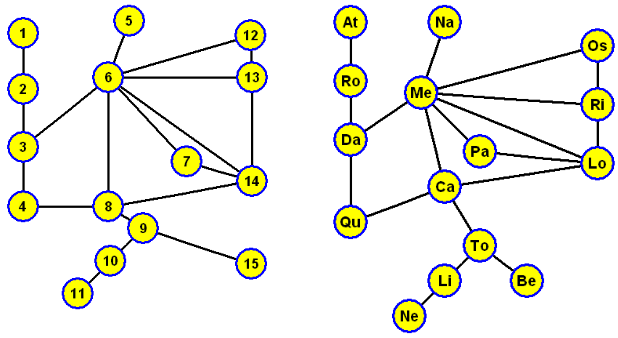


Fig. 28 Result of the puzzle *Towns*

4 Results and Conclusion

In the paper we offered some ideas how to construct knowledge of students in the subject *Graph Theory and Combinatorial Optimization*. As the most important elements in this process we consider the explanation of the subject matter in contexts and exercise it on practical examples.

All figures used in this paper were created within the program *Graphs*, the program created in the Delphi 5 environment by our student as a part of his thesis [14], the program, whose main purpose is a visual representation of basic graph-concepts and graph-algorithms using a colouring process on graphs created within the program. Short manual to the program *Graphs* can be found in [12], the link is http://edu.uhk.cz/~milkoev1/program_Graphs.

The program enables the creation of a new graph, editing it, working on it (moving, colouring vertices, edges, etc.), saving graph in the program, and saving the graph in bmp format.

There is also the possibility to open more than one window so that two (three) objects can be compared at once and thus a “bridge” between topics can be demonstrated.

Using the program there is no problem to illustrate the required concepts and processes. It is known that demonstration and visualization make the subject much clearer and comprehensible and make students’ self-preparation much more efficient (see e.g. [10], [11], [12], [15], [16], [17]). However, it is very important *to prepare suitable illustrative graphs* for students. Then they can easy *revise the subject-matter*. Students can not only use graphs prepared by the teacher but also graphs created by themselves and explore the properties of these graphs.

The ability to save graphs in bmp format allows them easy creation of needed graphs for their tasks (texts and/or presentations) where they describe various practical situations with the aid of graphs and solve the given problem.

References:

[1] Biggs, N. L., Lloyd, K. E., Wilson, R. J.:*Graph theory 1736-1936*, Clarendon Press, Oxford, 1976.

[2] Borůvka, O.: O jistém problému minimálním, *Práce Mor. Přírodověd. Spol v Brně*, 3, 1926, pp. 37-58.

[3] Borůvka, O.: Příspěvek k řešení otázky ekonomické stavby elektrovodních sítí. *Elektrotechnický obzor*, 15, 1926, 153-154.

[4] Graham, R. L., Hell, P.: On the History of the Minimum Spanning Tree Problem, *Annals of the History of Computing 7*, 1, 1985, 43-57.

[5] Jarník, V.: O jistém problému minimálním, *Práce Mor. Přírodověd. Spol.v Brně*, 6, 1930, pp. 57-63.

[6] Kruskal, J. B.: On the shortest spanning subtree of a graph and the travelling salesman problem, *Proc. Amer. Math. Soc.*, 7, 1956, 48-50.

[7] Kruskal, J.B.: A reminiscence about shortest spanning subtrees, *Archivum Mathematicum Brno*, 33, 1997, 13-14.

[8] Milková, E.: The Minimum Spanning Tree Problem in historical and present context, In: *Proceedings HPM 2004 & ESU 4*, Iraklion, Greece: University of Crete, 2006, pp. 261 – 267.

[9] Milková, E.: Combinatorial Optimization: Mutual Relations among Graph Algorithms, *WSEAS TRANSECTION on MATHEMATICS*, Issue 5, Volume 7, May 2008, pp. 869-879.

[10] Milková, E.: What can multimedia add to the optimization of teaching and learning at universities? In: *7th WSEAS International Conferences on Advances on Applied Computer & Applied Computational Science (ACACOS ’08)*, Hangzhou, China, April 6-8, 2008, WSEAS Press, pp. 704–709.

[11] Milková, E.: Multimedia Applications and their Benefit for Teaching and Learning at Universities, *WSEAS TRANSACTIONS on INF. SCIENCE & APPLICATIONS*, Issue 6, Volume 5, June 2008, pp. 869-879.

[12] Milková, Eva.: Enhancing Understanding and Constructing Knowledge in Graph Theory and Combinatorial Optimization within the ‘Graphs’ multimedia application, *Electronic Journal of Mathematics and Technology*, [On-line serial], 3 (1), 2009, pp. 57 – 72.

[13] Nešetřil, J., Milková, E., Nešetřilová, H., Otakar Borůvka on Minimum Spanning Tree Problem., *Discrete mathematics*, 233 (2001), pp. 3-36.

[14] Pozdílek M., *Grafové algoritmy: vizualizace*, Hradec Králové, diploma thesis, 2004.

[15] Sastre Rosa, M. A., Iglesias, C. E., Carbajo, A. G.: Interactive tools for Discrete Mathematics e-learning, *WSEAS TRANSACTIONS on ADVANCES in ENGINEERING EDUCATION*, Issue 2, Volume 5, February 2008, pp. 79-103.

[16] Slabý, A., Kozel, T., Malý, F.: Mobile approach, trends and technologies in modern information systems. In: *7th WSEAS International Conferences on Advances on Applied Computer & Applied Computational Science (ACACOS ’08)*, Hangzhou, China, 2008, WSEAS Press, pp. 716-720.

[17] Slabý, A., Kozel, T., Malý, F.: Mobile approach, trends and technologies in modern information systems, *WSEAS TRANSACTIONS on BUSSINES and ECONOMICS*, Issue 5, Volume 6, pp. 341-350.