Annales Mathematicae et Informaticae
***33** (2006) pp. 151–165*
http://www.ektf.hu/tanszek/matematika/ami

Variations for spanning trees

László Zsakó

Faculty of Informatics ELTE
e-mail: Zsako@ludens.elte.hu

Submitted 20 June 2006; Accepted 8 December 2006

Abstract

Coursebooks discussing graph algorithms usually have a chapter on minimum spanning trees. It usually contains Prim's and Kruskal's algorithms [1, 2] but often lacks other applications. This type of problem is rarely present at informatics competitions or in tests in secondary or higher level informatics education This article is aimed at describing some competition tasks that help us prove that the application of the above algorithms are well-suited for both competition and evaluation purposes.

The Hungarian National Informatics Competition for Secondary School Students look back on a history of 20 years and so does the International Olympiad in Informatics. Basically, informatics competitions rely on algoritmization tasks [3, 4], the circle of which is continually developing though showing a surprisingly great constancy at the same time.

At both types of competitions there are often tasks connected to graphs or problems that can be reduced to representation of graphs. International competitions nearly lack tasks in connection with minimum spanning trees. On the other hand, at national competitions (Hungarian National Informatics Competitions for Secondary School Students and the Selecting Competition for the International Olympiad) the scientific committees deciding on tasks (mainly Gyula Horváth and László Zsakó) has set this kind of problem several times. You could also have met a similar one for instance at the 11th Lithuanian Olympiad in Informatics (Winter in The Kingdom of Fancy).

This article is about the experience gained so far: with the help of the tasks below, we would like to show that their solution is not a simple description of the two classical spanning tree algorithms (Prim's and Kruskal's) but their creative application [5, 6].

As being a methodologist, in this article I do not intend to invent fundamentally new algorithms but rather discover the applicability and limits of algorithms known so far.

When Hungarian secondary school students are being prepared for competitions, we rely on two fundamental books on algorithms [1, 2]. Therefore, our stepping stone here are two basic algorithms described in these books (one is taken from one of the books while the other is from the other one). This article is not aimed at describing further algorithms that are too complicated for this age group but the studying of the applicability of the two basic procedures.

Definition 1.1. Let $G = (V, E)$ be a connected undirected graph. An acyclic connected $F(V, E')$ subgraph of a G graph is a spanning tree of the graph where $E' \subseteq E$ (A spanning tree is a tree that contains all the nodes of G and its edges are taken from the edges of G.) Let a weight function $c : E \to \mathbb{R}$ be given. Then an F spanning tree of G is a minimum spanning tree if its cost (the sum of the weights of its edges) is the minimum of all the spanning trees of G.

Prim's procedure [1]: Let $G = (V, E)$ be a connected graph, $V = \{1, \ldots, N\}$. Starting from node 1, expand the tree containing this node until – expanded with all the nodes – you get the minimum spanning tree.

```
Procedure Prim(G: graph; var F: set of edges);
  var U: set of nodes;
    u,v: nodes;
  begin
    F:={};
    U:={1};
    while $U \neq G.V$ do begin
      let $(u, v)$ be a minimum weight edge, for which $u \in U$ and $v \in G.V \backslash U$;
      $F := F \cup \{(u, v)\}$;
      $U := U \cup \{v\}$;
    end;
  end;
```

Kruskal's procedure [2]: Sort the edges in increasing order by weight. Create an N-member spanning forest from node N. Check the edges and if there are any that connect various spanning trees, join the spanning trees.

```
Procedure Kruskal(G: graph; var A: set of edges)
  $A := \emptyset$;
  for all $v \in G.V$ do Create-a-set(v);
  sort the edges of $G.E$ in increasing order by weight $w$;
  for all $(u, v) \in G.E$
    do if $Find - set(u) \neq Find - set(v)$
        then begin $A := A \cup \{(u, v)\}$; Join$(u, v)$; end;
  end;
```

In this article we are only dealing with problems that can be solved by secondary school students participating at informatics competitions if they are familiar with the above basic graph algorithms. The essence of the method is the analysis of two starting situations:

- Let us modify the text of the task by adding or dropping conditions then examine whether the above algorithms are applicable and if so what modifications are necessary.
- Let us modify either of the above algorithms then find a sensible problem to match it, which can thus be solved.

First let us see some competition tasks from the past few years and the solutions we expected to receive. You can find examples for both strategies among them as well as among the ideas that follow them.

Problem 1.2 (Plant – spanning forest).[1] *A company manufactures goods in its plants based in K cities, which are then to be transported to N cities. The transport routes must be strengthened so that heavy lorries will be able to run along them. Therefore, the transport routes are meant to be arranged in such a way that the total length of the transport routes to be strengthened will be the smallest possible. Write a program that determines the minimum total length of the roads to be strengthened as well as from which city to which city the goods are to be transported.*

At this problem it is feasible to start with Prim's algorithm. The basic variant takes an arbitrary node (right node 1) as a starting tree. In this task take a starting forest with nodes $1, 2, \ldots, K$. Here you do not even need to assume that the graph is connected. The only thing is that in its every connected component there must be at least one plant.

In the solution you will need the spanning trees themselves. The starting nodes of the spanning trees are the nodes with plants. The task expects you to determine for all the nodes of the graph from which nodes of a given spanning tree they can be reached directly.

Therefore, you have to modify the algorithm at two parts–at the ones in italics:

```
Procedure Prim(G: graph; var F: set of edges);
   var U: set of nodes;
         u,v: nodes;
   begin
      $F := \{\}$;
      $U := \{1, 2, \ldots, K\}$;
      for all $u \in U$ do $w(u) := u$;
      while $U \neq G.V$ do begin
         let $(u, v)$ be the cheapest edge for which $u \in U$ and $v \in G.V \backslash U$;
```

[1]The Selecting Competition for the International Olympiad in 2002. Competition tasks are printed in italics. From the original tasks we have omitted the descriptions of the imput and the output as well as the limitations and the conditions.

```
        $F := F \cup \{(u, v)\}$; *$w(v) := u$;*
        $U := U \cup \{v\}$;
      end;
    end;
```

Here Kruskal's procedure is a bit awkward to use because you should work around so as not to join two sub-trees in both of which there is a plant.

Problem 1.3 (Mill – spanning forest).[2] *The Milling Company has flour milled and packed in K Mills. The flour should be transported to N cities in such a way that the total transportation cost be the lowest possible. Write a program that gives the minimum transportation cost, and for each town from which mill the flour is to be transported.*

The problem is fairly similar to the previous one: you have to construct a spanning forest here, as well [7]. As for time, it was set after that one at the competition but it is a bit tricky because in this problem it is not the total length of the edges that should be minimum but the total length of the paths deriving from the roots (mill) of trees. Therefore, it is possible that the shortest edge of the graph is not an element of the spanning forest as the figure above shows (Kruskal's procedure would first join these two nodes). According to the previous problem (Mill,A), (A,B) would be the minimum spanning tree, whereas for this problem the correct solution is (Mill,A), (Mill,B).

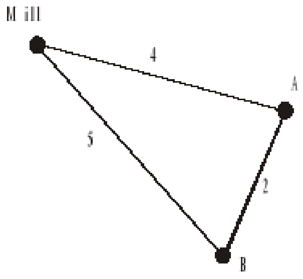

Figure 1:

Thus you need to store in the modified Prim's algorithm for each node of the spanning trees to-be-built, the length of the shortest path leading from the root (mill) to it, then you need to attach a node to one of its spanning trees to which the path leading from the root is the shortest. A further modification is that here for each node of the spanning tree you need to store its root (i.e. the starting node of the path leading there instead of the edge leading there).

Now you have to modify the algorithm at three places; the modifications here are in italics, as well:

[2]The Hungarian National Informatics Competition for Secondary School Students in 2005.

```
Procedure Prim(G: graph; var F: set of edges);
  var U: set of nodes;
      u,v: nodes;
  begin
    $F :=$ ;
    $U := \{1, 2, \ldots, K\}$; for all $u \in U$ do $w(u) := u$;
    while $U \neq G.V$ do begin
      Let $(u, v)$ be an edge where $u \in U$ and $v \in G.V \setminus U$ and
            $Min(u) + weight(u, v)$ are minimum;
      $F := F \cup \{(u, v)\}$; $w(v) := w(u)$;
      $U := U \cup \{v\}$; $Min(v) := Min(u) + weight(u, v)$;
    end;
  end;
```

Remark 1.4. If you started from one node, you would actually get a variant of Dijkstra's algorithm to determine the minimum paths starting from one node.

Problem 1.5 (Pipeline – a spanning forest for a complete graph).[3] *In a country oil refineries were built at K places, where petrol is produced from crude oil. There are N places with filling stations where the petrol should be transported through pipeline. Both of them are given with their co-ordinates. The pipeline must be designed in such a way that minimum length of pipes are to be laid. The pipeline can fork only at filling stations or refineries and can be led only horizontally or vertically in the map. The refineries are numbered from 1 to K while filling stations from K+1 to K+N. Write a program that determines between which nodes (oil refineries or filling stations) the pipeline should be constructed in a way that the lenght of the pipeline will be minimum.*

The problem is nearly identical with the "Plant" problem. There is just one little peculiarity here: there is an edge between any nodes of the graph and the length i.e. the weight of the edge is equal to the Manhattan-distance of its two extreme nodes.

Problem 1.6 (Road – a spanning tree with starting components).[4]
In a city there are several squares. Some of them are connected with roads while others are not and they should be constructed. Between the squares there are areas allocated by the local government for roads. Only after the construction of roads had started did it turn out that there would not be enough money to build all the roads. On the other hand, the reconversion of those that already exist is also terribly expensive. Write a program that determines which roads the local government should build so that it will cost the least and in the meantime you could travel from each square to each square.

What is peculiar about the problem is that there are starting connected components (not necessarily trees). The components can be defined relying on the

[3]The Selecting Competition for the International Olympiad in 2000.
[4]The Selecting Competition for the International Olympiad in 2003.

already strengthened roads. Therefore, it is not the minimum spanning tree of the starting graph that you should create but the minimum spanning tree of another graph that consists of the components. And from now on you can choose among three solutions:

A. Take a graph consisting of the components (i.e. combine the nodes of the components into the nodes of the new graph), then with the aid of the non-strengthened roads apply any algorithm that gives you a minimum spanning tree.

B. Applying Kruskal's procedure, first join the sub-trees between which there is a strengthened road without any further conditions then relying on the basic algorithm, join those between which there is a non-strengthened road.

Let F be the set of strengthened and E the set of not yet strengthened edges. The modifications are shown in italics:

```
Procedure Kruskal(G: graph;F: set of edges; var A: set of edges)
   $A := \{\}$;
   for all $v \in G.V$ do Make-a-set(v);
   for all $(u, v) \in F$ do if Find-a-set(u)$\neq$Find-a-set(v)
         then Join(u,v);
   sort the edges of $G.E$ in increasing order by weight $w$;
   for all $(u, v) \in G.E$
      do if Find-a-set(u)$\neq$Find-a-set(v)
            then begin $A := A \cup \{(u, v)\}$; Join(u,v); end;
   end;
```

C. As a third solution, the edges of set F could be added to set E with 0 weight. Then the basic algorithm could do nearly without any modifications. The only thing you need to take care about, however, is that you must not include the edges with 0 weight in the edge set A arising as a solution. The solution would be a bit slower since there is a larger edge set here to be sorted than in the case above.

Problem 1.7 (Network – online spanning tree).[5] *In Fairyland a network is to be constructed to connect N cities. During K weeks you keep receiving tenders for constructing a direct connection between two cities. For every week–if possible–give a plan, based on the tenders received so far, regarding which cities are to be connected directly so that every city will be accessible from every other city directly or via other cities but such that the total cost of construction will be minimum. The plan contains the pairs of cities to be connected and the total cost of constructing the network.*

The problem is the online version of creating a minimum spanning tree. The difficulty the competitors are to face is that they rarely, if ever, meet such algorithms at school or at the preparatory courses for the competitions [1].

The problem, however, can be deduced to the following case: if there is a minimum spanning forest and one single new graph edge, how can you make a

[5]The Selecting Competition for the International Olympiad in 2004.

minimum spanning forest? For example, a modification to Kruskal's algorithm could be the following:

```
Procedure online(G: graph; var A: set of edges)
  $A := \{\}$;
  for all v∈ G.V do Make-a-set(v);
  for all (u,v) ∈ G.E
    do if Find-a-set(u) ≠ Find-a-set(v)
          then begin $A := A \cup \{(u,v)\}$; Join(u,v); end
          else begin
               $(p,q)$:=the maximum weight edge of path $(u,v)$;
               if $weight(p,q) > weight(u,v)$
                  then $A := A \cup \{(u,v)\} - \{(p,q)\}$;
            end;
  end;
```

Further ideas

Below you will find problem ideas that the scientific committees of the competitions concerned have already dealt with and probably they will be set as competition tasks in the future.

Idea$_1$

The pipeline problem can also be formulated such that for the sake of safety there should be constructed a pipeline where every filling station can be supplied with petrol from at least two oil refineries.

A possible solution is: the solution of the original pipeline problem then connecting K components somehow.

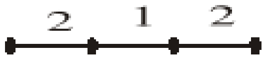

Figure 2:

It means that if you take the graph consisting of components, define the minimum spanning forest consisting of at least 2-element trees. It is easy to see that you cannot apply the greedy strategy here: you must not include the edge with weight 1 you can see in the figure into the spanning forest of the component-graph.

Therefore, this problem cannot be used as a variant of a greedy strategy minimum spanning tree, no matter how much it resembles the basic problem.

Idea$_2$

So far we have been dealing with undirected graphs. And what about directed ones? If you assume that a graph connected and directed i.e. it has at least one node from which there is a path to all other nodes then the problem can be solved.

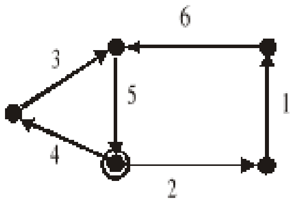

Figure 3:

If there is one starting node (i.e. one to which does not lead an edge), the solution is easy: start constructing the tree from that node following Prim's algorithm.

If there are several such nodes, they are definitely in a cycle (since starting from any node any of them can be accessed). In this case, create for each node their spanning trees, then choose the best one out of them. If you look at the figure, the good starting node is circled. Although all the nodes of the graph can be accessed from every node, if you start from any other nodes, you will get higher-cost spanning trees.

There is just one question left: to determine the nodes from which all the other nodes are accessible.

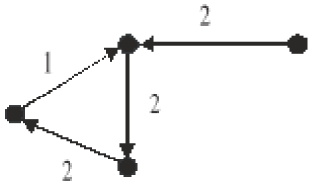

Figure 4:

If you take graph with several starting nodes and you are aware that all the nodes can be accessed from at least one node, you can apply Prim's algorithm (e.g. just like in the "Plant" problem), as well.

Interestingly, if you try to use Kruskal's algorithm, however, you may arrive in a dead-end street as its simple application can result in a wrong spanning tree. For example, as for the graph in the figure the edge of weight 1 is not in the spanning tree.

Idea$_3$

If you are unlucky, in the "Plant" problem you may even have to transport goods to cities with no plants from one single plant whereas from all the other plants nowhere i.e. tree K-1 of the minimum spanning forest consists of only one node whereas the Kth tree contains all the other nodes.

Modify the problem such that every sub-tree contains at least M nodes.

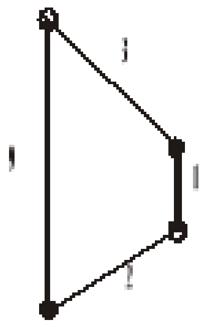

Figure 5:

It is easy to see that this problem cannot be solved with the so far applied greedy strategy. Let the black circles in the figure denote the starting nodes and let all the trees of the spanning forest contain 2 nodes. If you include the edge of weight 1 into any of the sub-trees, the cost of the spanning forest can be only 10 but the correct solution contains only the edges of weights 2 and 3.

For M⩾4 the problem is NP complete [10]. Therefore, it cannot be used as a variant of a greedy strategy minimum spanning tree although that is a natural extension of the original problem.

Idea$_4$

In an archipelago the islands are connected with bridges. The bridges of a maximum length of H can be re-newed. Write a program that determines the minimum total length of the bridges to be renewed in a way that you can travel from any island to any island. The problem is a modification of the "Plant" problem: the spanning forest must consist of maximum H long edges. Actually, you must determine a subgraph of a graph (excluding edges longer than H) then apply e.g. Kruskal's algorithm on them [8]. When sorting the edges, you can exclude the unnecessary ones.

Idea$_5$

An important feature of Kruskal's algorithm is that the spanning forest being built is always of the lowest cost. The number of the trees of the forest is being decreased by 1 at each step. Relying on this, you can create a K-element minimum spanning forest. The problem is similar to the "Plant" problem, just the nodes of the K-element spanning forest are not given in advance. In this case, the solution is to follow Kruskal's procedure until you have exactly K trees after the joining.

Idea$_6$

You can also modify Kruskal's algorithm in a way that in the meantime you subtract spanning trees of a certain size from further processing. Do this to trees that contain at least M nodes. (We do not expect, however, that the spanning forest created this way will be of minimum cost.)

Lemma 2.8. *Every tree subtracted from further processing can contain maximum* $2^{(M-1)}$ *nodes.*

Proof. According to the task, you will have to deal with trees that contain maximum $M-1$ nodes. The algorithm reduces two trees at each step so the number of elements of the tree can be maximum double the $M-1$. □

You should modify Kruskal's algorithm in a way that you exclude the edges leading out from reduced trees with minimum M nodes (the essence of the modification is in italics):

```
If *free(u) and free(v) and*
    Find-a-set(u)$\neq$ Find-a-set(v) then ...
```

Idea$_7$

The previous idea shows clearly that if you use Kruskal's procedure, it is far from being guaranteed that you will have an exactly M-element spanning tree in the course of progress.

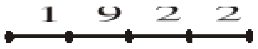

Figure 6:

Prim's procedure is suitable for this in principle but the result you get is not necessarily the minimum M-element spanning tree (the sub-graph containing M arbitrary nodes). If you take the left node in the figure as a starting point, you will get a 10-cost long spanning tree with the left 3 nodes but the cost of the spanning tree on the right containing 3 nodes is only 4. The figure shows that even if you started from the two ends of the shortest edge as a starting tree, it would not help, either.

Relying on Prim's algorithm, however, it may be a sensible task to create a minimum M-element spanning tree that starts from a given "starting node".

Idea$_8$

You can get the second best spanning treea from the minimum spanning tree. Take the minimum spanning tree. Take the edges not belonging to the tree in turn. If you add them to the tree, you will get a cycle. Omit the longest edge of the cycle. If you follow this procedure for each possible edge, from the trees obtained this way that one will be the second best spanning tree in which the difference in length of its new edge and omitted edge is minimum [1, 2].

Idea$_9$

You can formulate the minimum spanning tree problem by saying that you have to construct a spanning tree the longest edge of which is minimum. In this case you do not need a new algorithm as Kruskal's algorithm will give you the correct solution just like for the original problem [8].

Idea$_{10}$

The minimum spanning tree problem can easily be transformed into a maximum spanning tree one. An easy solution: the sum of the weights is maximal when/if the sum of their negatives is minimal.

In another formulation of the problem (previous idea): for the shortest edge to be maximal, the longest among the edges of the respective negative lengths should be minimal.

Another solution: sort the edges in Kruskal's algorithm in decreasing order [9].

Idea$_{11}$

An important feature of Kruskal's algorithm is that the spanning forest being constructed is always minimum and the number of trees in it is decreasing by one at each step. Thus it would be a sensible task to determine a spanning forest consisting of the fewest trees with less than a given cost.

Idea$_{12}$

We are going to connect big cities in Hungary with pipelines suitable for transporting petrol. A petrol pipeline can be built between two cities but the pipelines can be joined at any of their points. Every pipeline connects either two cities or joins a pipeline already constructed. The task is to construct the pipeline network so that you use the fewest pipes possible. Write a program that reads in the co-ordinates (real numbers) of big cities then displays the length of the minimum length pipeline network.

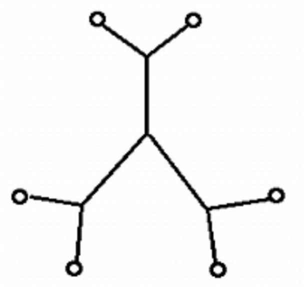

Figure 7:

The problem just sligthly differs from the "Pipeline" problem but its solution is completely different: you have to create a Steiner tree, which is far from being secondary school level.

Big cities of Hungary are to be connected with pipelines suitable for transporting petrol. First a petrol pipeline is constructed between two cities but a new city can be joined to the ready network not only at another city but at any point of the pipeline. Every pipeline connects either two cities or joins a pipeline already constructed. The task is to construct the pipeline network so that you use the least pipes possible. Write a program that reads in the co-ordinates (real numbers) of the cities then displays the length of the minimum length pipeline network.

It is the online version of the problem. It is greedy but it has nothing to do with classical spanning tree algorithms since in this case you have to determine in what order the cities are connected to the network. Undoubtedly, you have to construct the pipeline between the first two cities. Whereas regarding any other city, you

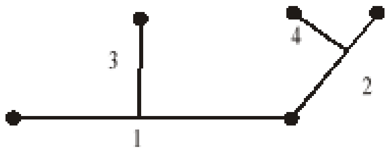

Figure 8:

have to determine the shortest length between the new city and the already existing pipes and then to connect the city to the closest node (city or pipeline). (In the figure you can see the connection sequence.)

Remark 2.9. the result you get this way will not be identical with the Steiner tree of the previous problem. What is more, you will get different spanning trees if the sequences of nodes are different.

Instead of a conclusion – survey experience

With the help of our survey experience, I would like to prove that these problems can be solved by the élite of secondary school students. The above shows that they can be varied highly therefore they are well-suited to be used at informatics competitions.

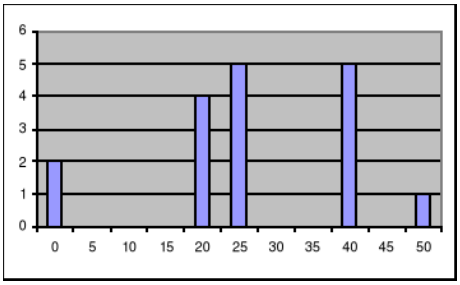

Figure 9: Pipeline

The slight deviation from the maximum score mostly derives from efficiency or from omitting the study of special cases.

This spanning tree task was the first of its kind set at a selecting competition for the student olympiad. The distribution of scores is interesting: about half of the competitors handed in a 50% solution and another significant percentage achieved an 80% performance. Since it was the first task of its kind, it took the competitors by surprise. The 80% solutions rose from good but not efficient algorithms i.e. as the diagram shows nobody but one student knew (or discovered) the efficient minimum spanning tree algorithm. Behind the 40-50% performance lie solutions of not full value.

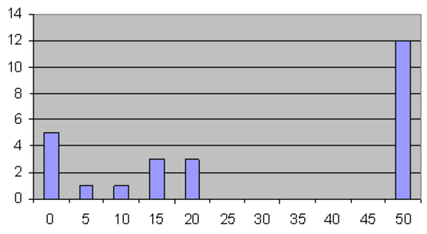

Figure 10: Plant

This diagram shows that in 2002 about 50% of the students participating at the selecting competition for the student olympiad provided a perfect solution while the other half did not deserve a score or just a little. When analysing their solutions, it turned out that they did not know either of the two original algorithm. They attempted at various solutions and in some special cases they got a right one. The students who received 15 to 20 scores, typically devised a correct algorithm, but with their minimum cubic or even worse algorithms they were not able to supply a correct result within the time limit of the competition. It was a definite progress within two years.

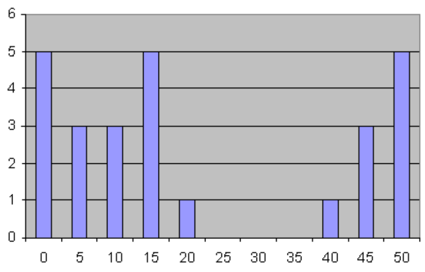

Figure 11: Road

This problem was set at a competition a year after the previous one and it shows a seemingly surprising distribution of scores. Compared to the previous one, many competitors fell into the 0-40% category. Analysing the solutions, you can see that about half of the 17 competitors belonging to this group did not take into account the ready components. However, they received a correct result for some test cases even this way. Most of those with 0-5 scores started working following Solution A described in the article and they either did not manage to finish work in time or their programs were correct only in a few cases. Half of those with high scores chose a solution similar to Variant B, while the other half to Variant C.

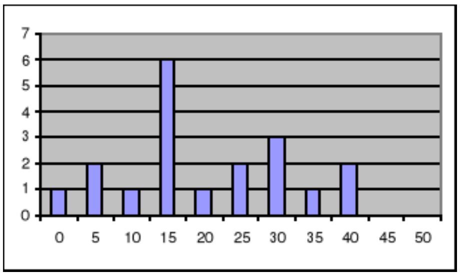

Figure 12: Network

As we mentioned above in the article, the online version took the competitors by surprise. In general, they are not familiar with online algorithms, which is clearly shown by the scores they achieved. Interestingly, about 50% of the competitors were the same as the participants of the competition a year before so they had already seen a spanning tree problem and some of them had given a good solution then.

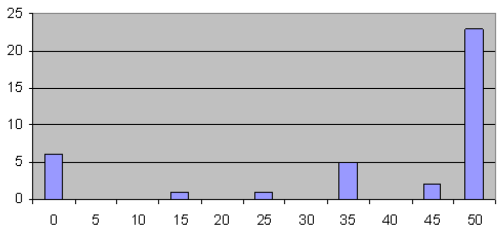

Figure 13: Mill

In 2005 this task was solved by a broader circle that could have seen the solutions of the previous tasks. Typically, here we also had about 50% with a maximum score and 10% with 0 score. Those with scores in between are different from the other task: typically, here they knew the original algorithms that give solutions to this group of problems but they applied them incorrectly.

References

[1] Rónyai Lajos, Ivanyos Gábor, Szabó Réka: Algoritmusok. Typotex, 1999.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest: Introduction to Algorithms, MIT, 1990.

[3] `http://olympiads.win.tue.nl/`

[4] `http://ceoi.inf.elte.hu/`

[5] Szlávi Péter, Zsakó László: Módszeres programozás: Gráfok, ELTE Informatikai Kar, 2004.

[6] Szlávi Péter, Zsakó László: Gráfokkal kapcsolatos algoritmusok, NJSZT – Neumann János Tehetséggondozó Program, 2004.

[7] Katona Gyula, Recski András, Szabó Csaba: A számítástudomány alapjai. Typotex, 2002.

[8] Steven Skiena: Analysis of Algorithms, SUNY Stony Brook distance learning program, 1996.

[9] R. Muhammad: Spanning Tree and Minimum Spanning Tree, web-publications 2005, `http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Greedy/ms.htm`

[10] Xiaoyun Ji, John E. Mitchell: Minimum Weight Constrained Forest Problem, 2005 Optimization Days, Montreal, Canada, 2005.

[11] `http://ims.mii.lt/olimp/?lang=en&sk=pasirengimas&id=0610` – Lithuanian Olympiad in Informatics, tasks

László Zsakó
Faculty of Informatics ELTE
1117 Budapest
Pázmány Péter sétány 1/C
Hungary