Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Departement of Computer Science
Markus Püschel, David Steurer
Gleb Novikov, Tommaso d'Orsi, Ulysse Schaller, Rajai Nasser

13. December 2021

Algorithms & Data Structures Exercise sheet 12 HS 21

Exercise Class (Room & TA):	
Submitted by:	
Peer Feedback by:	
Points:	

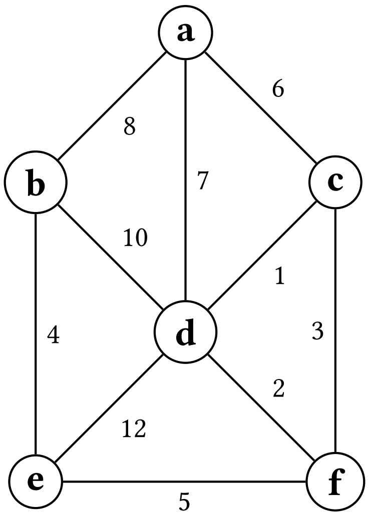
Submission: On Monday, 20. December 2021, hand in your solution to your TA *before* the exercise class starts. Exercises that are marked by * are challenge exercises. They do not count towards bonus points.

Remark. Let $G = (V, E)$ be a weighted graph with nonnegative weights ($w(e) \geq 0 \quad \forall e \in E$) such that all edge-weights are different ($\forall e \neq e'$ in E, $w(e) \neq w(e')$). Then the minimum spanning tree of G is unique.

You can use this fact without further justification for solving this exercise sheet.

Exercise 12.1 *MST practice **(1 point)**.*

Consider the following graph



a) Compute the minimum spanning tree (MST) using Boruvka's algorithm. For each step, provide the set of edges that are added to the MST.

Solution: At the first step we add edges $\{\mathbf{a}, \mathbf{c}\}, \{\mathbf{b}, \mathbf{e}\}, \{\mathbf{c}, \mathbf{d}\}, \{\mathbf{d}, \mathbf{f}\}$. At the second step we add $\{\mathbf{e}, \mathbf{f}\}$.

b) Provide the order in which Kruskal's algorithm adds the edges to the MST.

Solution: $\{\mathbf{c},\mathbf{d}\}, \{\mathbf{d},\mathbf{f}\}, \{\mathbf{b},\mathbf{e}\}, \{\mathbf{e},\mathbf{f}\}, \{\mathbf{a},\mathbf{c}\}$.

c) Provide the order in which Prim's algorithm (starting at vertex **d**) adds the edges to the MST.

Solution: $\{\mathbf{c},\mathbf{d}\}, \{\mathbf{d},\mathbf{f}\}, \{\mathbf{e},\mathbf{f}\}, \{\mathbf{b},\mathbf{e}\}, \{\mathbf{a},\mathbf{c}\}$.

Exercise 12.2 *Minimum Spanning Tree and Shortest Paths **(1 point)**.*

Let $G = (V, E)$ be a connected edge-weighted graph where all the weights are nonnegative and distinct. Let T be a minimum spanning tree of G.

Let $v \in V$ be some vertex and define T_v to be the tree of shortest paths that is obtained by applying Dijkstra's algorithm on G starting from the source v.

Is it possible that T and T_v do not have any edge in common? If the answer is yes, provide an example showing that it is possible. Otherwise, prove that it is impossible

Solution: The trivial graph with one vertex and no edges has T and T_v with no edges, so formally they do not have any edge in common.

However, if we consider only the case $|E| > 0$, then the answer is no: Since the edge-weights are nonnegative and distinct, there is a unique minimum spanning tree T. Therefore, T is the same as the tree that is obtained by applying Prim's algorithm starting from v.

Consider the edge $\{v, u\}$ that is incident to v and which has the minimum weight. The edge $\{v, u\}$ is the first edge to be added to T in Prim's algorithm. Similarly, (v, u) is a shortest path from v to u and $\{v, u\}$ is the first edge to be added to T_v is Dijkstra's algorithm. We conclude that T and T_v share the edge $\{v, u\}$.

Exercise 12.3 *Constructing a Fiber Optic Network **(1 point)**.*

The government of Atlantis put you in charge of installing a fiber optic network that connects all its n cities. There are two technologies of fibre optic that you can use:

- Fibre 1.0: It is a good reliable technology that is relatively cheap. There is a list of pairs of cities between which it is possible to install a direct Fibre 1.0 link. Furthermore, for each such pair, there is a corresponding positive integer cost.
- Fibre 2.0: It is an emerging technology that it extremely good and can directly connect any two cities. However, its cost is too high and the government cannot afford a single Fibre 2.0 link.

Note that all direct links are two-directional. The installed network should connect all the cities of Atlantis: Between any two cities, there should be a connected path of direct links in the network that connects them.

A philanthropist volunteered to donate the cost of exactly $k < n$ direct Fibre 2.0 links, and you can use them to connect any k pairs of cities. Your goal is to minimize the cost that is paid by the government for the Fibre 1.0 links that are needed to construct a connected network. Describe an algorithm that finds the network that costs the government the minimum amount of money.

Note that it is possible to construct a network connecting all the cities of Atlantis using only Fibre 1.0 links, but we would like to benefit from the k Fibre 2.0 links that were donated by the philantropist in order to minimize the cost that is paid by the government.

Hint: *Modify Kruskal's algorithm.*

Solution: Before stating the solution, it would be useful to introduce some terminology: An undirected graph is said to be a forest if it does not contain any cycle. In other words, a graph is a forest if and only if its connected components are trees. We say that it is an ℓ-forest if it has ℓ connected components. Note that a graph is a 1-forest if and only if it is a tree.

Note that the removal of k edges from a tree yields a $(k+1)$-forest. Conversely, if we have an ℓ-forest, we can convert it into a tree by adding $\ell - 1$ edges connecting its connecting components without creating cycles.

Let $G = (V, E)$ be the undirected graph where the set of vertices V corresponds to the cities of Antlantis and the set of edges E corresponds to the pairs of cities between which we can install direct Fibre 1.0 links. The edges in E are weighted by the cost of installing direct Fibre 1.0 links.

Let T be the tree corresponding to the network to be installed. Let $e_1, \ldots, e_k$ be the k edges corresponding to the Fibre 2.0 links that will be donated. Note that $e_1, \ldots, e_k$ may or may not be in E.

Let $F = T \setminus \{e_1, \ldots, e_k\}$ correspond to the Fibre 1.0 links for which the government has to pay. It is easy to see that F is a spanning $(k+1)$-forest of G: It is a subgraph of G (with the same set of vertices) which is also a $(k+1)$-forest.

We can now see that the problem is equivalent to finding a minimum spanning $(k+1)$-forest of G, i.e., one that has the minimum total weight. This can be done using a slight modification of Kruskal's algorithm. Let $n = |V|$ be the number of cities. Instead of completing Kruskal's procedure until adding $n-1$ edges, we stop after adding $n-k-1$ edges from E. The proof of correctness of this algorithm is very similar to that of Kruskal's algorithm for the minimum spanning tree problem.

Exercise 12.4 *Ancient Kingdom of Macedon.*

The ancient Kingdom of Macedon had n cities and m roads connecting them, such that from one city, you can reach all other $n-1$ cities. All roads were *roman roads* i.e. stone-paved roads that did not require any maintenance and no two roads were of the same length. With the technological developments in the Roman Kingdom, a new type of carriage was developed, called the *Tesla Carriage*, which was much faster than all the alternatives in the Ancient Macedon Kingdom. However, the Tesla Carriage required *asphalt roads* to operate, and such roads had to be maintained every year, or otherwise the asphalt would wear off, rendering the road unusable as if it was a roman road.

With the effort to modernize the kingdom, Phillip II promised the Ancient Macedonians that he will provide them with asphalt roads by paving some of the existing roman roads, such that every two cities can be reached through a Tesla Carriage. The price to pave a roman road or maintain an asphalt road is equal, and is proportional to the length of the road. To save money Phillip II decided to pave sufficient roman roads to fulfill his promise, while minimizing the overall yearly maintenance price.

Even in the first years, the new Tesla Carriages improved the lives of the average Ancient Macedonians, but at the same time, they also provided means for robbers to commit crimes and escape to another city. To resolve this, Phillip II decided to create checkpoints the second year, one at each asphalt road. Each of the checkpoint will have a fixed cost for both building and maintenance.

Assuming a fixed price k for each checkpoint, does Phillip II have to consider paving new roman roads, or he can maintain the same set of roads in order to make sure that the overall maintenance price of the roads and the checkpoints is still minimal? Prove your reasoning, or provide a counter example.

Note: For simplicity, assume that the roman roads were paved all at once, on the first day of the year, and maintenance will be done the same day next year, again all at once. Also assume that checkpoints can also be built at once for all roads, as well a they can be maintained all at once in a day.

Solution.

Let's think of all cities in the Ancient Macedon Kingdom as vertices in a graph G and all roads as edges. In order to minimize the overall maintenance price, Philip II must pave only roman roads that form a minimum spanning tree T in G. As a result, we can rephrase the problem as the following graph problem. Let T be a minimum spanning tree of a weighted graph G. Construct a new graph G' by increasing the weight of each edge in G by k. Do the edges of T form a minimum spanning tree of G'?

In a graph with n vertices, every spanning tree has $n-1$ edges. Thus the weight of every spanning tree is increased by exactly $(n-1)\cdot k$. Therefore, the minimum spanning trees remains the same. In other words, Phillip II does not have to consider paving new roman roads: keeping the current asphalt roads and building a checkpoint on each of them still guarantees minimal maintenance cost.

Exercise 12.5* *Spanning Forest with 2 components.*

Let $G = (V, E)$ be a connected edge-weighted graph in which all weights of the edges are different and positive. Consider the following two algorithms, which take G, the weights of all edges, and two different vertices $u, v \in V$ as input.

Algorithm 1

```
Run Kruskal's algorithm to get a minimum spanning tree $T = (V, E_T)$.
Find the unique path $\pi$ from $u$ to $v$ in $T$.
Find the edge $e$ of maximal weight among edges in $\pi$.
Remove $e$ from $T$ to get a graph $H = (V, E_T \setminus \{e\})$.
return $H$
```

Algorithm 2

```
$M \leftarrow \{u, v\}$
$E_M \leftarrow \emptyset$
while $M \neq V$ do
    $\Delta M = \{\{w_0, w_1\} \in E : w_0 \in M, w_1 \in V \setminus M\}$
    Find the edge $\{w_0, w_1\} \in \Delta M$ of minimal weight.
    $E_M \leftarrow E_M \cup \{\{w_0, w_1\}\}$
    $M \leftarrow M \cup \{w_1\}$
return $H = (M, E_M)$
```

Prove that the two algorithms return the same graph.

Hint: *Consider the graph G' which is obtained from G by adding an edge e_0 of weight 0 between u and v (if the edge $\{u, v\}$ already exists, then simply decrease its weight to 0). Try to relate the two given algorithms on G to algorithms that you know from the lecture on G'.*

Solution.

Note that G' is still a weighted graph with all edge-weights being different (since we assume that $w(e) > 0$ for all $e \in E$). Clearly the edges of G' all have nonnegative weights, so by the remark at the beginning of the sheet we know that the MST of G' is unique. In order to prove that the two algorithms return the same graph, we will show that they both yield the MST T' of G' minus the edge e_0 (note that since e_0 is the edge of minimal weight in G', it is clear that it is part of T').

Algorithm 1:

We claim that Algorithm 1 yields the same graph as running Kruskal's algorithm on G', minus e_0. Since e_0 has minimal weight in G', the first step of Kruskal's algorithm on G' is to add e_0 to T'. It remains to show that after this step, the only edge added to T in Algorithm 1 that is not added to T' is precisely e, and that every edge added to T' is also added to T. Let T_n denote the set of edges added to T by Kruskal's algorithm after n steps, and T'_n the set of edges added to T' by Kruskal's algorithm after $n+1$ steps (i.e. after going through e_0 and n additional edges). We will show by induction that $T_n \setminus \{e\} = T'_n \setminus \{e_0\}$, where e denotes the edge of maximal weight on the path from u to v in T.

The base case $n = 0$ clearly holds since $T_0 \setminus \{e\} = \emptyset \setminus \{e\} = \emptyset$ and $T'_0 \setminus \{e_0\} = \{e_0\} \setminus \{e_0\} = \emptyset$. Let $n \in \mathbb{N}$ and suppose by induction that $T_n \setminus \{e\} = T'_n \setminus \{e_0\}$. We will prove that $T_{n+1} \setminus \{e\} = T'_{n+1} \setminus \{e_0\}$. Let f be the $n+1$-th edge considered (i.e. in this step Kruskal's algorithm needs to decide if it adds f to T_n and/or to T'_n).

Suppose first that f is not added to T_n. This means that $T_n \cup \{f\}$ contains a cycle. If this cycle doesn't contain the edge $e := ww'$, then clearly there is also a cycle in $T'_n \cup \{f\}$ and thus f is also not added to T'_n. If the cycle contains e this means in particular that $e \in T_n$. Since e is the edge of maximal weight on the path π from u to v in T and Kruskal's algorithm considers edges in order of increasing weights, we must have $\pi \subset T_n$. Since $T_n \setminus \{e\} = T'_n \setminus \{e_0\}$, replacing e in the cycle by the path from w to u (on π), concatenated with e_0 and the path from v to w' (on π), we also obtain a closed walk in $T'_n \cup \{f\}$. Therefore f is also not added to T'_n. So in this case we have shown that indeed $T_{n+1} \setminus \{e\} = T_n \setminus \{e\} = T'_n \setminus \{e_0\} = T'_{n+1} \setminus \{e_0\}$.

Now suppose that f is added to T_n but not to T'_n. Since $T_n \setminus \{e\} = T'_n \setminus \{e_0\}$, this is only possible if the cycle in $T'_n \cup \{f\}$ contains e_0. In particular, there must be a path from u to v that uses f in

$$(T'_n \setminus \{e_0\}) \cup \{f\} = (T_n \setminus \{e\}) \cup \{f\} \subseteq T_n \cup \{f\}.$$

On the other hand, since f was added to T_n, there was no such path in T_n. This is only possible if f is the last edge on this path from u to v that was added by Kruskal's algorithm, i.e. if f has maximal weight on this path. Thus, we must have $f = e$, which implies

$$T_{n+1} \setminus \{e\} = (T_n \cup \{e\}) \setminus \{e\} = T_n \setminus \{e\} = T'_n \setminus \{e_0\} = T'_{n+1} \setminus \{e_0\}.$$

The only remaining case is when f is added both to T_n and to T'_n. But in this case it is clear that $T_n \setminus \{e\} = T'_n \setminus \{e_0\}$ implies $T_{n+1} \setminus \{e\} = T'_{n+1} \setminus \{e_0\}$, so the proof for Algorithm 1 is complete.

Algorithm 2:

Consider Prim's algorithm on the graph G' with starting vertex u. Since e_0 has minimal weight in the whole graph and is incident to u, this will be the first edge added by Prim's algorithm, creating a connected component $\{u, v\}$. After that, it is clear that Algorithm 2 and Prim's algorithm are doing the exact same operations (basically Algorithm 2 is simply Prim's algorithm in G' with the first step hidden in the initialization $M \leftarrow \{u, v\}$). So Algorithm 2 will also return T' minus the edge e_0. This concludes the whole proof.

Remark. Algorithm 1 and Algorithm 2 actually both return the optimal solution of the following problem. Given $u, v \in V$, find two trees $T_u = (V_u, E_u)$, $T_v = (V_v, E_v)$ that are subgraphs of G such that:

- they span the graph (i.e. $V_u \cup V_v = V$).
- $u \in V_u$ and $v \in V_v$.
- their total weight $\sum_{e \in E_u \cup E_v} w(e)$ is minimized subject to the above conditions.