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
François Hublet, Goran Zuzic, Tommaso d'Orsi, Jingqiu Ding

12 December 2022

Algorithms & Data Structures   Exercise sheet 12   HS 22

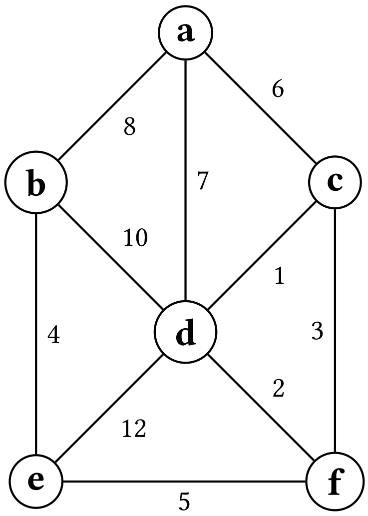
The solutions for this sheet are submitted at the beginning of the exercise class on 19 December 2022.

Exercises that are marked by * are "challenge exercises". They do not count towards bonus points.

You can use results from previous parts without solving those parts.

Exercise 12.1 *MST practice.*

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 *Maximum Spanning Trees and Trucking* **(2 points)**.

We start with a few questions about **maximum spanning trees**.

(a) How would you find the **maximum** spanning tree in a weighted graph G? Briefly explain an algorithm with runtime $O((|V| + |E|) \log |V|)$.

Solution:

We simply take any MST algorithm (e.g., Boruvka, Prim, or Kruskal) and replace all the $\min$s with $\max$s. Specifically: in Boruvka, we will find the maximum-weight outgoing edge from each connected component ("ZHK" from the lecture); in Prim, we will extract-max (instead of extract-min), use $\max$ to update weights, and use increase-key; in Kruskal, we will sort in decreasing order. The correctness arguments do not change (except for replacing "minimum" with "maximum"); the same $O((|V| + |E|) \log |V|)$ bound holds for runtime.

(b) Given a weighted graph $G = (V, E)$ with weights $w : E \to \mathbb{R}$, let $G_{\geq x} = (V, \{e \in E \mid w(e) \geq x\})$ be the subgraph where we only preserve edges of weight x or more. Prove that for every $s \in V, t \in V, x \in \mathbb{R}$, if s and t are connected in $G_{\geq x}$ then they will also be connected in $T_{\geq x}$, where T is the maximum spanning tree of G.

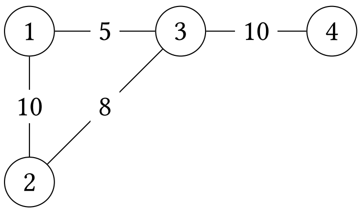
Hint: *Use Kruskal's algorithm as inspiration for the proof.*
Hint: *If it helps, you can assume all edges have distinct weight and only prove the claim for that case.*

Solution:

As argued in class, the maximum spanning tree is obtained by running Kruskal's algorithm that sorts the edges by decreasing weight, hence edges of $G_{\geq x}$ will be processed strictly before all of $G_{<x} := G \setminus G_{\geq x}$. Furthermore, Kruskal's algorithm only removes an edge if it would create a cycle, which does not affect connectivity. Hence, any pair $s, t \in V$ that was connected in $G_{\geq x}$ will still be connected in the maximum spanning tree using edges of weight at least x. In other words, s and t will be connected in $T_{\geq x}$, as needed.

Problem: You are starting a truck company in a graph $G = (V, E)$ with $V = \{1, 2, \ldots, n\}$. Your headquarters are in vertex 1 and your goal is to deliver the maximum amount of cargo to a destination $t \in V$ in a single trip. Due to local laws, each road $e \in E$ has a maximum amount of cargo your truck can be loaded with while traversing e. Find the maximum amount of cargo you can deliver for each $t \in V$ with an algorithm that runs in $O((|V| + |E|) \log |V|)$ time.

Example:




```
Output:
Max cargo to 1 is ∞
Max cargo to 2 is 10
Max cargo to 3 is 8
Max cargo to 4 is 8
```

Explanation:
The best path from the headquarters to 4 is $1 \to 2 \to 3 \to 4$, and the maximum cargo the truck can carry is $\min(10, 8, 10) = 8$.

(c) Prove that for every $t \in V$, the optimal route is to take the unique path in the **maximum** spanning tree of G.

Hint: *Suppose that the largest amount of cargo we can carry from 1 to t in G (i.e., the correct result) is OPT and let ALG be the largest amount of cargo from 1 to t in the maximum spanning tree. We need to prove two directions: $OPT \leq ALG$ and $OPT \geq ALG$.*

Hint: *One direction holds trivially as any spanning tree is a subgraph. For the other direction, use part (b).*

Solution:

Suppose that the largest amount of cargo we can carry from 1 to t in G (i.e., the correct result) is OPT and let ALG be the largest amount of cargo from 1 to t in the maximum spanning tree.

Direction $ALG \geq OPT$. By definition of OPT, there exists a path from 1 to t where all edges have weight $w(e) \geq OPT$. In other words, 1 and t are connected via $G_{\geq OPT}$. By part (b), they will also be connected in $T_{\geq OPT}$, where T is the maximum spanning tree of G. Hence, there is a path in T between 1 and t where all edges have weight $w(e) \geq OPT$. We conclude that $ALG \geq OPT$.

Direction $ALG \leq OPT$. Since any spanning tree is a subgraph of the original graph and no solution in a subgraph can be larger than in G, we conclude that $ALG \leq OPT$.

(d) Write the pseudocode of the algorithm that computes the output for all $t \in V$ and runs in $O((|V| + |E|) \log |V|)$. You can assume that you have access to a function that computes the maximum spanning tree from G and outputs it in any standard format. Briefly explain why the runtime bound holds.

Solution:

Algorithm 1

```
Input: graph G, given as n ≥ 1 and an adjacency list adj of (neighbor, weight) pairs.
Global variable: marked[1...n], initialized to [False, False, ..., False].

function DFS(u, capacity)                      ▷ we can reach u with a truck of capacity
    Print("Max cargo to ", u, " is ", capacity)
    marked[u] ← True
    for each neighbor (v, w) ∈ adj[u] do       ▷ edge u → v has weight w
        if not marked[v] then
            DFS(v, min(capacity, w))

adj ← MaximumSpanningTree(G)                   ▷ We replace G with its maximum spanning tree.
DFS(1, ∞)
```

The runtime of maximum spanning tree is $O((|V| + |E|) \log |V|)$ and the DFS runtime is $O(|V| + |E|)$. In total, we have a runtime of $O((|V| + |E|) \log |V|)$.

Exercise 12.3 *Counting Minimum Spanning Trees With Identical Edge Weights* **(1 point)**.

Let $G = (V, E)$ be an undirected, weighted graph with weight function w.

It can be proven that, if G is connected and all its edge weights are pairwise distinct[1], then its Minimum Spanning Tree is unique. You can use this fact without proof in the rest of this exercise.

For $k \geq 0$, we say that G is k*-redundant* if k of G's edge weights are non-unique, e.g.

$$|\{e \in E \mid \exists e' \in E.\, e \neq e' \wedge w(e) = w(e')\}| = k.$$

[1] I.e., for all $e \neq e' \in E$, $w(e) \neq w(e')$.

In particular, if G's edge weights are all distinct, then G is 0-redundant, and if its edge weights are all identical, it is $|E|$-redundant.

(a) Given a weighted graph $G = (V, E)$ with weight function c and $e = \{v, w\} \in E$, we say that we *contract* e when we perform the following operations:

(i) Replace v and w by a single vertex vw in V, i.e., $V' \leftarrow V - \{v, w\} \cup \{vw\}$.

(ii) Replace any edge $\{v, x\}$ or $\{w, x\}$ by an edge $\{vw, x\}$ in E, i.e.,

$$E' \leftarrow E - \{\{v, x\} \mid x \in V\} - \{\{w, x\} \mid x \in V\} \cup \{\{vw, x\} \mid \{v, x\} \in E \vee \{w, x\} \in E\}.$$

(iii) Set the weight of the new edges to the weight of the original edges, taking the minimum of the two weights if two edges are merged, i.e.

$$\begin{aligned}
c'(\{x, y\}) &= c(\{x, y\}) && x, y \notin \{v, w\} \\
c'(\{vw, x\}) &= c(\{v, x\}) && \{v, x\} \in E, \{w, x\} \notin E \\
c'(\{vw, x\}) &= c(\{w, x\}) && \{v, x\} \notin E, \{w, x\} \in E \\
c'(\{vw, x\}) &= \min(c(\{v, x\}), c(\{w, x\})) && \{v, x\} \in E, \{w, x\} \in E.
\end{aligned}$$

For all $G = (V, E)$ and $e \in E$, we denote by G_e the graph obtained by contracting e in G. Explain why if T is an MST of G and $e \in T$, then T_e must be an MST of G_e.

Solution:

Assume that T_e is not an MST of $G_e = (V_e, E_e)$. Then there exists a spanning tree (V_e, T') of G_e with total cost $w(T') < w(T_e)$. Based on T', we will construct a spanning tree in the original graph G with smaller total cost.

Consider the following set of edges of the original graph G:

$$\begin{aligned}
T'' = \ & \{e\} \cup \{\{x, y\} \mid \{x, y\} \in T' \wedge x, y \neq vw\} \\
& \cup \{\{v, x\} \mid \{vw, x\} \in T' \wedge \{v, x\} \in E \wedge (\{w, x\} \notin E \vee c(\{w, x\}) > c(\{v, x\})\} \\
& \cup \{\{w, x\} \mid \{vw, x\} \in T' \wedge \{w, x\} \in E \wedge (\{v, x\} \notin E \vee c(\{v, x\}) > c(\{w, x\})\}
\end{aligned}$$

Let us show that (V, T'') is a tree, using the following characterization: a tree is a connected graph on n vertices with $n - 1$ edges. First, T'' has $|T''| = |T'| + 1 = |V_e| - 1 + 1 = |V_e| = |V| - 1$ edges. Moreover, there is a path between every pair of vertices of G in T''. To show this, consider $x, y \in V$. If $\{x, y\} = \{v, w\}$, then e is a path between x and y in T''. If $\{x, y\} \neq \{v, w\}$, let p be a path between x and y in T'. There are two cases:

- Either p does not go through vw, and it is also a path in T'';
- Or it contains vw, and we can replace the (at most two) edges adjacent to vw in p by their preimage in T''. If the path p is transformed into two disjoint paths ending at v and w in the process, then the edge e can be used to reconnect them in T''.

Therefore, (V, T'') is a tree. As it covers all vertices of G, (V, T'') is also a *spanning tree* of G.

Now, $w(T'') = w(T') + w(e) < w(T_e) + w(e) = w(T)$, contradicting the minimality of T. We conclude that T_e is an MST of G_e.

(b) Let $k > 0$. Show that for all k-redundant $G = (V, E)$ and $e \neq e' \in E$ with $w(e) = w(e')$, then G_e is k'-redundant for some $k' \leq k - 1$.

Solution:

Let V_e, E_e such that $G_e = (V_e, E_e)$. Denote by w_e the weight function of G_e. For each $a \neq b \in E_e$ such that $w_e(a) = w_e(b)$, we can find $a' \neq b' \in E$ such that a' and b' are contracted to a and b respectively, and $w(a') = w(b')$. However, a' and b' can never be e, since e is removed from the graph through the contraction operation. Therefore,

$$|\{a \in E \mid \exists b \in E_e.\, a \neq b \wedge w_e(a) = w_e(b)\}| \leq |\{a' \in E \mid \exists b' \in E.\, a' \neq b' \wedge w(a') = w(b')\}| - 1,$$

and G_e is k'-redundant for some $k' \leq k - 1$.

(c) Show that if G is connected and k-redundant, it has at most 2^k distinct MSTs.

Hint: *By induction over k, using (a) and (b).*

Solution:

We prove, by induction over $k \geq 0$: $P(k)$: "Any k-redundant graph has at most 2^k distinct MSTs."

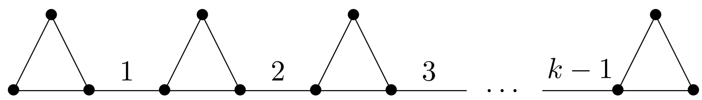
Base case. For $k = 0$, this is exactly the lemma from the lecture: a graph whose edge weights are all pairwise distinct has $2^0 = 1$ MSTs.

Induction hypothesis. Let $k \geq 0$ such that $P(k')$ holds for all $k' \leq k$, i.e., any k'-redundant graph has at most $2^{k'}$ distinct MSTs.

Induction step. Let $G = (V, E)$ be a $k+1$-redundant graph. Let e be an edge whose weight $w(e)$ is not unique among the weights of edges in E. Let us consider the sets M_1 of MSTs of G that contain e and M_2 of MSTs of G that do not contain e. Clearly, the total number of MSTs of G is $|M_1| + |M_2|$. By (a), for any MST $T \in M_1$, T_e is an MST of G_e. Moreover, G_e is k'-redundant for some $k' \leq k$. Now, $|M_1|$ is at most the number of MSTs of G_e, which is at most 2^k by $P(k)$. Every MST $T \in M_2$ is also an MST of $G - \{e\}$, and therefore $|M_2| \leq 2^k$ by $P(k)$. We get $|M_1| + |M_2| \leq 2^k + 2^k = 2^{k+1}k$, which proves $P(k+1)$.

(d) Show that for all large enough n, there exists a graph G such that G is n-redundant and has at least $2^{\frac{n}{2}}$ distinct MSTs.

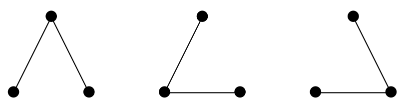
Hint: *First assume that $n = 3k$ for some k. Consider graphs of the following form, where all unmarked edges have weight* 0*. When $n = 3k+1$ or $n = 3k+2$, you can add one or two edges with cost k and $k+1$ at either end.*



Solution:

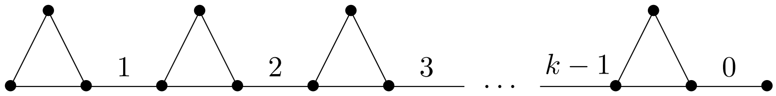
For $k \geq 0$, denote by G_k the graph of the above form, with k connected triangles. This graph has $3k + (k-1) = 4k - 1$ edges and redundancy $3k$, since there are $3k$ edges with weight 0 (the triangle edges) and all other edges have distinct weights $1..k-1$.

For any $k \geq 0$, the MSTs of G_k contain all non-zero edges, while in each triangle, one can choose independently between the following three pairs of edges:



Hence, the $3k$-redundant graph has $3^k = 3^{\frac{3k}{3}} = 2^{\log_2 3 \cdot \frac{3k}{3}}$ distinct MSTs. Since $\frac{\log_2 3}{3} \approx 0.53 > \frac{1}{2}$, this is more that $2^{\frac{3k}{2}}$ MSTs. This proves the result when $n = 3k$.

When $n = 3k + 1$ or $n = 3k + 2$, we can add one or two additional edges at either end of G_k to obtain an n-redundant graph, e.g., for $n = 3k + 1$:



The graph has $2^{\log_2 3 \cdot \frac{n-1}{3}}$ or $2^{\log_2 3 \cdot \frac{n-2}{3}}$ MSTs, which is at least $2^{\frac{n}{2}}$ as soon as $\log_2 3 \cdot \frac{n-2}{3} \geq \frac{n}{2}$, which is $n(\frac{\log_2 3}{3} - \frac{1}{2}) \geq \frac{2 \log_2 3}{3}$ or $n \geq \frac{2 \log_2 3}{\log_2 3 - \frac{3}{2}} = \frac{2}{1 - \frac{3}{2 \log_2 3}} \approx 37.3$. Hence, for $n \geq 38$, there exists an n-redundant graph with at least $2^{\frac{n}{2}}$ distinct MSTs.