University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn

# Algorithms and Data Structures
# Winter Term 2021/2022
# Sample Solution Exercise Sheet 9

## Exercise 1: Minimum Spanning Trees

Let $G = (V, E, w)$ be an *undirected, connected, weighted* graph with pairwise distinct edge weights.

(a) Show that $G$ has a *unique* minimum spanning tree.

(b) Show that the minimum spanning tree $T'$ of $G$ is obtained by the following construction:

*Start with $T' = \emptyset$. For each cut in $G$, add the lightest cut edge to $T'$.*

## Sample Solution

(a) Assume, for a contradiction, that there are two different MSTs, $T_1 = (V, E_1)$ and $T_2 = (V, E_2)$, where $E_1, E_2 \subset E$. Since $T_1$ and $T_2$ are different, then there are edges that are in $T_1$ but not in $T_2$, and similarly, there are edges in $T_2$ but not in $T_1$. Let $D_1 = E_1 \setminus E_2$, and let $D_2 = E_2 \setminus E_1$. Let $D = D_1 \cup D_2$ be the set containing all edges that are in one of the two trees, but not in both. Consider the edge with the smallest weight in $D$, and let's call it $e$ (note that, since $G$ has pairwise distinct edge weights, then $e$ is unique). By construction of $D$, either $e \in T_1$ or $e \in T_2$. Without loss of generality, assume that $e \in T_1$ (and hence $e \notin T_2$). In the following we show that, if we add $e$ to $T_2$ and then remove some other edge, then we get a new tree with smaller total weight, contradicting the fact that $T_2$ is an MST.

Let us add the edge $e$ to $T_2$. Since $e \notin T_2$, and since $T_2$ is a spanning tree, then, by adding $e$ to $T_2$, we must close a cycle. In this cycle, there must be an edge $e' \neq e$ that is not in $T_1$, otherwise $T_1$ would contain a cycle and hence it would not even be a tree. Therefore, we have that $e' \in T_2$ and $e' \notin T_1$, implying that $e' \in D_2$. Since edge $e$ has the minimum weight among all edges in $D$, then $w(e) < w(e')$. Starting from $T_2$ we create a new tree $T_2'$, where we remove $e'$ from $T_2$ and then we add $e$ to $T_2$, that is, we create the tree $T_2' = (V, E_2')$ where $E_2' = (E_2 \setminus \{e'\}) \cup \{e\}$. Notice that $T_2'$ is still a spanning tree: in fact, by adding $e$ to $T_2$ we created a cycle, and by removing $e'$ from $T_2$ we are breaking that cycle. But now, by construction, we get that $w(T_2') < w(T_2)$, but this is in contradiction with the fact that $T_2$ is an MST.

(b) Let $T$ be the MST of $G$ and $T'$ the set containing the lightest cut edges.

$T' \subseteq T$: Let $s \in T'$, i.e., $s$ is the lightest cut edge of a cut $(S, V \setminus S)$ in $G$. Let $e$ be the edge of $T$ connecting $S$ and $V \setminus S$. If $e \neq s$, then $w(s) < w(e)$ and the spanning tree $(T \setminus \{e\}) \cup \{s\}$ would have a smaller weight than $T$, contradicting that $T$ is an MST. Hence we have $e = s$ and thus $s \in T$.

$T \subseteq T'$: Let $e \in T$. The graph $T \setminus \{e\}$ has two connected components which define a cut in $G$. With an exchange argument as above one can show that $e$ is the (unique) lightest cut edge of this cut, i.e., we have $e \in T'$.

## Exercise 2: Travelling Salesperson Problem

Let $p_1, \ldots, p_n \in \mathbb{R}^2$ be points in the euclidean plane. Point $p_i$ represents the position of city $i$. The distance between cities $i$ and $j$ is defined as the euclidean distance between the points $p_i$ and $p_j$. A *tour* is a sequence of cities $(i_1, \ldots, i_n)$ such that each city is visited exactly once (formally, it is a permutation of $\{1, \ldots, n\}$). The task is to find a tour that minimizes the travelled distance. This problem is probably costly to solve.[1] We therefore aim for a tour that is at most twice as long as a minimal tour.

We can model this as a graph problem, using the graph $G = (V, E, w)$ with $V = \{p_1, \ldots, p_n\}$ and $w(p_i, p_j) := \|p_i - p_j\|_2$. Hence, $G$ is undirected and complete and fulfills the triangle inequality, i.e., for any nodes $x, y, z$ we have $w(\{x, z\}) \leq w(\{x, y\}) + w(\{y, z\})$. We aim for a tour $(i_1, \ldots, i_n)$ such that $w(p_{i_n}, p_{i_1}) + \sum_{j=1}^{n-1} w(p_{i_j}, p_{i_{j+1}})$ is small.

Let $G$ be a weighted, undirected, complete graph that fulfills the triangle inequality. Show that the sequence of nodes obtained by a pre-order traversal of a minimum spanning tree (starting at an arbitrary root) is a tour that is at most twice as long as a minimal tour.

## Sample Solution

Let $R = (i_1, \ldots, i_n)$ be a minimal tour and $w(R) := w(p_{i_n}, p_{i_1}) + \sum_{j=1}^{n-1} w(p_{i_j}, p_{i_{j+1}})$. Let $T$ be an MST, $w(T) := \sum_{e \in T} w(e)$ its weight and $\mathcal{P}_T$ its pre-order sequence of nodes. As the graph is complete, $\mathcal{P}_T$ is also a tour.

We add points to $\mathcal{P}_T$ as follows: If two subsequent nodes $u$ and $v$ are not connected in $T$ by a tree edge, we add between $u$ and $v$ all nodes on the shortest path from $u$ to $v$ in $T$ (these are all nodes from $u$ to the first common ancestor $w$ and from there to $v$). We write $\mathcal{P}'_T$ for the sequence that we obtain (this is formally not a tour as points are visited more than once).

In $\mathcal{P}'_T$, two subsequent nodes are neighbors in $T$, so we can consider this sequence as a sequence of edges in $T$. Each edge from $T$ is contained in $\mathcal{P}'_T$ exactly twice (if you go from the last point back to the root). Thus we have $w(\mathcal{P}'_T) = 2 \sum_{e \in T} w(e)$. The triangle inequality implies $w(\mathcal{P}_T) \leq w(\mathcal{P}'_T)$ and hence $w(\mathcal{P}_T) \leq 2 \sum_{e \in T} w(e)$.

The minimal tour $R$ defines a spanning tree $T_R$ by taking the edges between subsequent nodes in $R$. As $T$ is the minimum spanning tree we have $w(T) \leq w(T_R) \leq w(T_R) + w(p_{i_n}, p_{i_1}) = w(R)$ and hence $w(\mathcal{P}_T) \leq 2 \cdot w(R)$.

*Remark: The above argumentation also works for the post-order traversal. However, if you want the tour to start at a predefined point, it is easiest to use this point as the root of a pre-order traversal.*

[1] The Travelling Salesperson Problem is in the class of $\mathcal{NP}$-complete problems for which it is assumed that no algorithm with polynomial runtime exists. However, this has not been proven yet.