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. (5 Points)

(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 $T'$.

(5 Points)

Sample Solution

(a) Let $T_1$ and $T_2$ be two minimum spanning trees with edges $e_1, \ldots, e_{n-1}$ and $e'_1, \ldots, e'_{n-1}$, sorted by increasing weight. Assume we have $T_1 \neq T_2$. Let $j$ be the largest index for which $e_j \neq e'_j$. As the weights are pairwise distinct, we also have $w(e_j) \neq w(e'_j)$. W.l.o.g. let $w(e_j) < w(e'_j)$. The graph $T' \setminus \{e'_j\}$ has two connected components with nodes $S$ and $V \setminus S$. Let $e_k$ be an edge in $T$ connecting $S$ and $V \setminus S$. As $T'$ contains only one edge between $S$ and $V \setminus S$, it must hold $k \leq j$ (as $e_k = e'_k$ for $k > j$). As $(T' \setminus \{e'_j\}) \cup \{e_k\}$ is a spanning tree and $w(e'_j) > w(e_j) \geq w(e_k)$, it has a smaller weight than $T'$, contradicting the fact that $T'$ is minimal.

(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 $G$. Assume $s \notin T$. In $T$ there is a (unique) path from $x$ to $y$. Let $e$ be an edge on that path which is a cut edge of $(S, V \setminus S)$. According to the assumption we have $e \neq s$ and $s$ is the (unique) lightest cut edge of $(S, V \setminus S)$, so we have $w(s) < w(e)$. It follows that the spanning tree $(T \setminus \{e\}) \cup \{s\}$ is lighter than $T$, contradicting that $T$ is an MST.

$T \subseteq T'$: 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
The Travelling Salesperson Problem is in the class of \(NP\)-complete problems for which it is assumed that no algorithm with polynomial runtime exists. However, this has not been proven yet.

E.g., \texttt{heapq} and \texttt{networkx.utils.union_find}. In \texttt{heapq} the function \texttt{heappush} corresponds to the \texttt{insert} operation and \texttt{heappop} to the \texttt{delete-min} operation from the lecture. You can also use \texttt{heappush} and \texttt{heappop} on Python-lists (more details here). If you instantiated an object \texttt{uf} of the class \texttt{UnionFind}, the command \texttt{uf[i]} creates a new set \(\{i\}\) if \(i\) does not exist in \texttt{uf} yet, and else returns the representative of the set containing \(i\) (this combines the functions \texttt{make-set} and \texttt{find} from the lecture. More details here).
Figure 1: The approximated tour.