0.1 Optimisation Example: Kruskal's Algorithm

Example 0.1. Suppose you are from a country called Xzghtyta which contains n cities that are currently all isolated from each other. As the new minister for transport it is your idea to correct this transport issue and to lay a railroad which should connect all the cities of your country. However, you have a budget. Each railway line will cost a certain predefined amount to lay (with no difficulties or unforeseen costs). How will you decide which city-links are the optimal ones to lay railtracks on?

We will soon see that this can be done by finding a minimal spanning tree. Which can be found through a greedy algorithm process. A suitable example of this kind of algorithm which should solve our problem is Kruskal's Algorithm, which is detailed below.

Algorithm 1 Kruskal's algorithm

Let G be a connected graph with vertex set $V = \{1, ..., n\}$ and $\omega : E \longrightarrow \mathbb{R}^+$ a weight function. The edges of G are ordered according to their weight, that is, $E = \{e_1, ..., e_m\}$ and $\omega(e_1) \leq ... \leq \omega(e_m)$.

```
1: procedure KRUSKAL($G, \omega, T$)
2:     $T \leftarrow \emptyset$
3:     for $k = 1$ to $m$ do
4:         if ACYCLIC($T \cup \{e_k\}$) then
5:             append $e_k$ to $T$
```

This process is very similar to our prior description of the generic greedy algorithm. Although this time, we sort the elements of our edge set from lightest to heaviest. And then we iterate through. At each iteration, we check if the created set created by $T \cup \{e_k\}$ at each iteration remains independent and if so we add e_k to the generated set and if not we ignore that element and move to the next in the list E.

For graphs, the process above is equivalent to this semi-pseudocode. That you make a forest containing just the vertices of G. And then we iteratively add the desired elements to that forest as long as no cycles are induced in the resulting subgraph of G.

```
1) Create a graph $F$ containing just the vertices of $G$.
2) Create a set $S = E(G)$; the edge set of $G$.
3) While $S$ is non-empty and $F$ is not yet spanning
3($a$) Remove an edge with minimum weight from S.
3($b$) If the removed edge introduces no cycles to $F$
then add the edge to $F$
```

In the next sub-section we will prove the correctness of algorithms such as this. It is also notable to mention that Kruskal's algorithm will terminate after exactly $n - 1$ iterations, where n is the number of vertices in a graph. This corresponds to *lemma 3.7* which says a spanning tree has exactly $n - 1$ edges. Any more edges added at that point will induce a cycle and so kruskal's algorithm terminates.