5.2 Greedy Algorithms for the Minimum Spanning Tree Problem [4.5, 4.6]

Given a connected graph $G$ where each edge $e$ has a weight $w(e)$. The Minimum Spanning Tree (MST) problem is to find a spanning tree of $G$ that has minimum total weight.

The idea of Kruskal algorithm is to keep adding minimal-weight edges as long as they do not form a cycle. In other words, viewing the set of vertices as a collection of $n$ connected components (where each component consists of only one vertex). Then in each step reduce the number of connected components by 1 by joining two components by an edge that has smallest weight.

- Sort the edges in increasing order of weight:
  
  $$w(e_1) \leq w(e_2) \leq \ldots \leq w(e_m)$$

- $E' \leftarrow \emptyset$
- For $i = 1$ to $m$ do
  - If $E' \cup \{e_i\}$ does not contain a cycle then $E' \leftarrow E' \cup \{e_i\}$

The sorting (line 1) takes time $\Theta(m \log m)$. The main loop leaves room for different implementations. With careful chosen data structures, it can be done in time $\Theta(m \log n)$. (Note that since $m = O(n^2)$, $\log m = O(\log n)$. Section 4.6 in the text has a detailed discussion of implementations of this algorithm.

Correctness of the Kruskal’s algorithm

Let $S_i$ be the the set of edges in $E'$ after the $i$-th iteration. In particular, $S_0 = \emptyset$ and $S_m$ is the output, where $m$ is the number of edges in $G$.

The notion of “promising” partial solution is defined below:

**Definition:** $S_i$ is said to be “promising” if it can be extended to an optimal solution using only edges from $\{e_{i+1}, e_{i+2}, \ldots, e_m\}$. In other words, $S_i$ is promising if there is a minimum spanning tree $OPT_i$ of $G$ so that

$$S_i \subseteq OPT_i \subseteq S_i \cup \{e_{i+1}, e_{i+2}, \ldots, e_m\}$$

Note that by this definition, the optimal solution $OPT_i$ may be distinct.

We will prove the following claim by induction on $i$:

**Claim:** For $0 \leq i \leq n$, $S_i$ is promising.

**Base case:** $i = 0$: $S_0 = \emptyset$, and any minimum spanning tree extends $\emptyset$ by the set of all edges $\{e_1, e_2, \ldots, e_m\}$. Thus the base case is trivially true.

**Induction step:** Assume that the Claim is true for some $i < n$. We prove it for $i + 1$. Note that either $S_{i+1} = S_i$ or $S_{i+1} = S_i \cup \{e_{i+1}\}$. We consider these two cases.
Case I: $S_{i+1} = S_i$. This means that the edge $e_{i+1}$ is not selected by our algorithm, i.e., adding $e_{i+1}$ to $S_i$ creates a cycle. Let $OPT$ be the minimum spanning tree that extends $S_i$ by the edges from $\{e_{i+1}, e_{i+2}, \ldots, e_m\}$. Then $OPT$ cannot contain $e_{i+1}$. So $OPT$ extends $S_{i+1}$ using only edges from $\{e_{i+2}, \ldots, e_m\}$. Hence $S_{i+1}$ is promising.

Case II: $S_{i+1} = S_i \cup \{e_{i+1}\}$. This means that the edge $e_{i+1}$ is selected by our algorithm, i.e., adding $e_{i+1}$ to $S_i$ does not create a cycle.

Let $OPT$ be the minimum spanning tree that extends $S_i$ by the edges from $\{e_{i+1}, e_{i+2}, \ldots, e_m\}$. Then $OPT$ cannot contain $e_{i+1}$. So $OPT$ extends $S_{i+1}$ using only edges from $\{e_{i+2}, \ldots, e_m\}$. Hence $S_{i+1}$ is promising.

Subcase IIA: $OPT$ contains $e_{i+1}$. Then $OPT$ also extends $S_{i+1}$, and it extends $S_{i+1}$ by edges only from $\{e_{i+2}, \ldots, e_m\}$. So we are done.

Subcase IIB: $OPT$ does not contain $e_{i+1}$. Then $OPT$ does not extend $S_{i+1}$. This is the most interesting case. We will show that $OPT$ can be modified to give another minimum spanning tree $OPT'$ that extends $S_{i+1}$ using only edges from $\{e_{i+2}, \ldots, e_m\}$.

Since $OPT$ does not contain $e_{i+1}$, adding $e_{i+1}$ to $OPT$ creates exactly one cycle $C$ that contains $e_{i+1}$. Since $S_{i+1}$ is a tree, the cycle $C$ must contain some edges $e_j$ not from $S_{i+1}$. Since OPT agrees with $S_i$ on all edges $e_1, e_2, \ldots, e_i$ the edge $e_j$ must be from $\{e_{i+2}, \ldots, e_m\}$. So $w(e_j) \geq w(e_{i+1})$.

Now modify $OPT$ by exchange $e_j$ and $e_{i+1}$. The result is still a spanning tree $OPT'$ of $G$, and it has total weight less than or equal the total weight of $OPT$. Since $OPT$ is already a minimum spanning tree, it follows that $OPT'$ has the same total weight as $OPT$. Therefore $OPT'$ is also a minimum spanning tree of $G$. Clearly $OPT'$ extends $S_{i+1}$ by edges only from $\{e_{i+2}, \ldots, e_m\}$. □

Other algorithms for the MST problem:

Prim’s algorithm starts with a set $X$ that consists of a single vertex $v_1$. At each step it extends $X$ by one vertex which is connected to some vertex in $X$ by an edge of smallest weight:

1. $X \leftarrow \{v_1\}$, $E' \leftarrow \emptyset$
2. While $X \neq V$ do
3. Let $(u, v)$ be the edge of smallest weight so that $u \in X$ and $v \in V - X$.
4. $E' \leftarrow E' \cup \{(u, v)\}$
5. $X \leftarrow X \cup \{v\}$
6. End While
7. Output $E'$

Exercise: Prove that Prim’s algorithm returns a minimum spanning tree by defining an appropriate notion of promising partial solution.

Another algorithm (called Reverse-Delete in the text) differs from Kruskal’s and Prim’s algorithms in that it starts with the full graph $G$ and then keep deleting maximum-weight edges as long as connectivity is maintained. All three algorithms discussed here can be seen as special instance of a more generic algorithm for the MST problem which we do not present here. The correctness of this generic algorithm can also be proved using some appropriate notion of “promising” partial solution.