Kruskal’s algorithm

Reminder: Minimum Spanning Tree

Assume that $G(V, E)$ is a finite connected undirected graph, and each edge $e \in E$ has a positive real-valued weight $w(e)$:

$$w : E \to \mathbb{R}^+.$$

**Task:** find a connected subgraph $T(V, E_T)$ of $G(V, E)$, $E_T \subseteq E$, such that

$$\sum_{e \in E_T} w(e)$$

is minimum.

This $T$ must be a minimum spanning tree.

Kruskal’s Algorithm (1956)

| Input : Graph $G(V, E)$, weight function $w : E \to \mathbb{R}^+$ |
| Output: Minimum spanning tree $T(V, T)$ of $G(V, E)$ |

1. sort the set $E = \{e_1, e_2, \ldots, e_{|E|}\}$ such that if $i < j$ it holds $w(e_i) \leq w(e_j)$;
2. $T \leftarrow \varnothing$;
3. for $i = 1, 2, \ldots, |E|$ do
   4. if $(V, T \cup \{e_i\})$ is circuit-free then
   5. $T \leftarrow T \cup \{e_i\}$;
   6. end
7. end

Algorithm 1: Kruskal’s Algorithm

---

1Additional reading: Section 2.2 in the book of S. Even “Graph Algorithms”.
**Example.** Consider the following $G(V, E)$ with the corresponding weights given as input to Kruskal’s algorithm:

![Graph Image]

Sorting of the edges in non-decreasing order of their weights produces the following list:

$$\left(\{A,C\}, \{C,D\}, \{A,D\}, \{A,B\}, \{B,D\}, \{B,C\}, \{C,F\}, \{E,F\}, \{D,F\}\right).$$

The corresponding weights are:

$$\left(1, 2, 3, 4, 4, 4, 4, 5, 6\right).$$

Execution of Steps 3-7 in the algorithm produces the following results:

<table>
<thead>
<tr>
<th>$i$</th>
<th>Edge</th>
<th>Circuit-free?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>${A,C}$</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>${C,D}$</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>${A,D}$</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>${A,B}$</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>${B,D}$</td>
<td>No</td>
</tr>
<tr>
<td>6</td>
<td>${B,C}$</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>${C,F}$</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>${E,F}$</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>${D,F}$</td>
<td>No</td>
</tr>
</tbody>
</table>
The output is the following MST $T$ with $w(T) = 16$:

![Diagram of MST](image)

**Proof of correctness of the algorithm**

Next, we show that the algorithm does what is supposed to do – finds a minimum spanning tree. We show that step by step.

**Circuit-free.** The produced structure $T$ is circuit-free because any edge in Step 5 is added if it does not create a simple circuit.

**Connectivity.** The produced output $T$ is connected. If not, then there is an edge $u \overset{e}{\rightarrow} v$ in $G$ such that $u$ and $v$ are not connected in $T$. Adding this $e$ to $T$ would not create a simple circuit, and therefore the algorithm should have added $e$ to $T$ in Steps 4-5.

**Spanning tree.** Since $T$ is connected and circuit-free, it is a tree. Since it contains all vertices of $G$, it is a spanning tree.

**Minimality.** Assume by contrary that the produced $T(\mathcal{V}, T)$ is not MST. Let $e$ be the first edge added to $T$ such that there is no MST that contains $T \cup \{e\}$. For the rest of the proof, we assume that $T$ is as in the algorithm just before adding $e$.

Let $T_0(\mathcal{V}, E_0)$ be an MST that contains $T$. Consider the graph $\mathcal{H}(\mathcal{V}, E_0 \cup \{e\})$. There is a simple circuit in $\mathcal{H}$ that contains $e$. Denote this circuit as:

$$v_1 \overset{e_1}{\rightarrow} v_2 \overset{e_2}{\rightarrow} v_3 \ldots \overset{e_{k-1}}{\rightarrow} v_{k-1} \overset{e_k}{\rightarrow} v_k \overset{e=\epsilon_k}{\rightarrow} v_1.$$

Not all edges of this simple circuit are in $T$ (otherwise, we would not add $e$ to $T$).

Consider pairs of vertices $(v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k)$. If there was a path in $T$ between each pair of vertices, then we could not add $e$ to $T$. Without loss of generality (w.l.o.g.) there is no path between $v_i$ and $v_{i+1}$ in $T$ for some $i \in \{1, 2, \cdots, k - 1\}$. The edge $e_i$ has to appear after $e$ in the ordering of the algorithm (otherwise it would be added to $T$). Therefore, $w(e) \leq w(e_i)$.

Then $\mathcal{H}'(\mathcal{V}, E_0 \cup \{e\} \setminus \{e_i\})$ is a spanning tree (no simple circuits, $|\mathcal{V}| - 1$ edges). Its weight is smaller or equal to the weight of $T_0$. Therefore, $\mathcal{H}'$ is an MST that contains $T \cup \{e\}$. Contradiction to the assumption. $\blacksquare$
Time complexity

- Sorting the edges requires time $O(|E| \log |E|)$ (by using merge-sort).
- Checking whether an edge creates a simple circuit requires $O(\log |V|)$. Since there are $|E|$ edges, Steps 3-7 require time $O(|E| \log |V|)$ by using data structure union-and-find.
- Total complexity is $O(|E| \log |E|)$.

Union-and-find

When there are no edges in $T$, we have an empty array of pointers indexed by the vertices of the graph.

When we add an undirected edge between two vertices $v_i$ and $v_j$ in Step 5, we add a pointer from $v_i$ to $v_j$. For example, if we add edges $v_1-v_2$ and $v_3-v_4$ to $T$, the structure becomes.

When we add a new edge between $v_i$ and $v_j$, we check if they are not connected yet by going along the sequence of pointers that $v_i$ and $v_j$ belong to. For example, we can add an edge between $v_1$ and $v_3$, and the result is as follows:

If now we want to add an edge between $v_1$ and $v_4$, we can check that both $v_1$ and $v_4$ point to the same vertex $v_2$. Therefore, they belong to the same connected component, and the edge $v_1-v_4$ creates a simple circuit.

It is possible to ensure that the length of the longest sequence of pointers is $O(\log |V|)$.

Exercise

Let $G(V,E)$ be an undirected connected finite graph with weight function $w : E \to \mathbb{N}^+$ (i.e. the weights are positive integer numbers). Assume that each edge is colored in either green or black color. Propose an algorithm that finds the greenest MST (i.e. the MST has the maximum number of green edges among all MSTs).

Solution. Assume that the graph has no parallel edges and self-loops, otherwise first remove them. Define a new function $w' : E \to \mathbb{R}^+$ as follows:

$$w'(e) = \begin{cases} w(e) - \varepsilon & \text{if } e \text{ is green} \\ w(e) & \text{if } e \text{ is black} \end{cases}$$

Here $\varepsilon > 0$ is a very small real number, for example $\varepsilon = 1/n^2$, where $n = |V|$. Run Prim’s or Kruskal’s algorithm on $G$ with the modified weight function $w'$ (while ignoring the colors). Denote the resulting tree $T$.  

4
Denote by $g(T)$ the number of green edges in $T$. We have

$$w'(T) = w(T) - g(T)\varepsilon > w(T) - 1.$$  

(Note that the last transition is due to $g(T) \leq n - 1 < n^2$).

**Lemma 1** $T$ is the greenest MST with respect to $w$ if and only if $T$ is an MST with respect to $w'$.

**Proof of lemma.**

1. Take two spanning trees $T_1$ and $T_2$ such that $w(T_1) > w(T_2)$. Then,

$$w'(T_1) - w'(T_2) = w(T_1) - w(T_2) - \varepsilon \cdot (g(T_1) - g(T_2)) > 0.$$  

We obtain that $w(T_1) > w(T_2) \implies w'(T_1) > w'(T_2)$.

2. Take two spanning trees $T_1$ and $T_2$ such that $w(T_1) = w(T_2)$. Then,

$$w'(T_1) < w'(T_2) \quad \text{if and only if} \quad g(T_1) > g(T_2).$$  

MST with respect to $w'$ is greenest MST with respect to $w$.  

□

We have the following algorithm.

**Input**: Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, weight function $w : \mathcal{E} \rightarrow \mathbb{N}^+$, color function $c : \mathcal{E} \rightarrow \{g, b\}$  

**Output**: Greenest MST $T$ of $\mathcal{G}(\mathcal{V}, \mathcal{E})$

1. for all $e \in \mathcal{E}$ do
2.  \hspace{1em} compute $w'(e)$;
3. end
4. Find MST $T$ with respect to $w'$;

**Algorithm 2**: Green-black tree algorithm

**Time complexity.** Trivially, the time complexity of the proposed algorithm is $O(|\mathcal{V}|^2 + |\mathcal{E}|)$ or $O(|\mathcal{E}| \log |\mathcal{E}|)$, depending on whether Prim's or Kruskal's algorithm is used to find an MST.

**Note.** A similar approach works also if the weights are real-valued. What changes need to be done for the real-valued case?