Greedy 2: Minimum Spanning Trees

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong

Let $G = (V, E)$ be an undirected graph. Let w be a function that maps each edge e of G to a positive integer value $w(e)$, which we call the **weight** of e.

An **undirected weighted graph** is defined as a pair (G, w).

We will denote an edge between vertices u and v in G as $\{u, v\}$ — instead of (u, v) — to emphasize that the ordering of u, v does not matter.

We consider that G is **connected**, namely, there is a path between any two vertices in V.
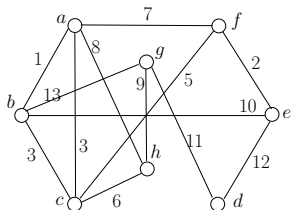
The integer on each edge indicates its weight. For example, the weight of $\{g, h\}$ is 9, and that of $\{d, g\}$ is 11.
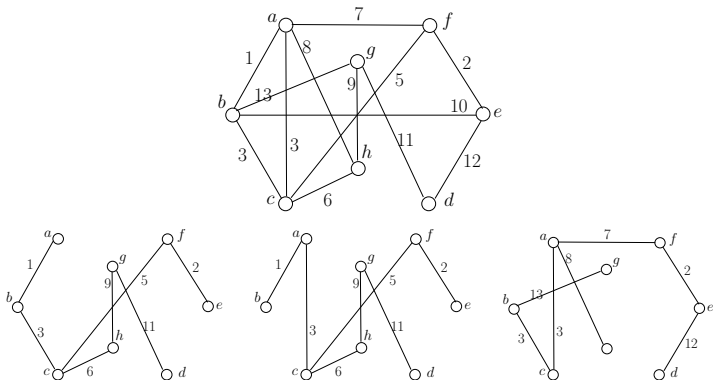
Spanning Trees

Recall that a **tree** is defined as a connected undirected graph with no cycles.

Given a connected undirected weighted graph (G, w) with $G = (V, E)$, a **spanning tree** T is a tree satisfying the following conditions:

- The vertex set of T is V.

- Every edge of T is an edge in G.

The **cost** of T is defined as the sum of the weights of all the edges in T (note that T must have $|V| - 1$ edges).
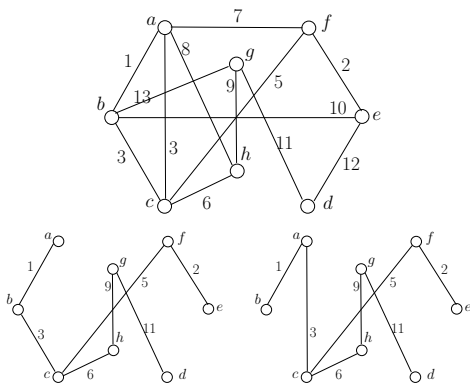
The second row shows three spanning trees (of the graph in the first row). The cost of the first two trees is 37, and that of the right tree is 48.

Given a connected undirected weighted graph (G, w) with $G = (V, E)$, the goal of the **minimum spanning tree (MST) problem** is to find a spanning tree of the smallest cost.

Such a tree is called an MST of (G, w).

Both trees in the second row are MSTs. This means that MSTs may not be unique.

Prim's Algorithm

Next, we will introduce **Prim's algorithm** for solving the MST problem.

The algorithm grows a tree T_{mst} by including one vertex at a time. At any moment, it divides the vertex set V into two parts:

- The set S of vertices that are already in T_{mst}.
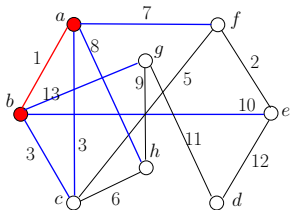
- The set of other vertices: $V \setminus S$.

If an edge connects a vertex in S and a vertex in $V \setminus S$, we call it a **cross edge**.

Greedy: The algorithm works by repeatedly taking the **lightest** cross edge.

Edge $\{a, b\}$ is the lightest of all. So, $S = \{a, b\}$ in the beginning. The MST we are growing now contains the only red edge in the figure. Cross edges are shown in blue.

Cross edges $\{c, a\}$ and $\{c, b\}$ are both the lightest. Either one can be taken into the tree. Without of loss generality, suppose we take $\{c, a\}$.

Now $S = \{a, b, c\}$. The MST we are growing now contains the red edges in the figure. Cross edges are shown in blue.
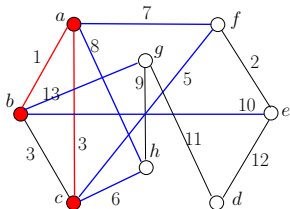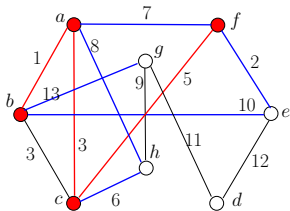

Note that $\{b, c\}$ is not a cross edge.
The lightest cross edge is $\{c, f\}$.

(Example)

Now $S = \{a, b, c, f\}$. The MST we are growing now contains the red edges in the figure. Cross edges are shown in blue.

The lightest cross edge is $\{e, f\}$.
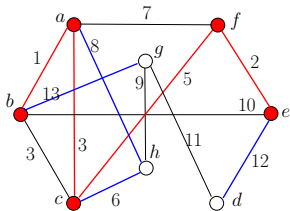
Now $S = \{a, b, c, f, e\}$. The MST we are growing now contains the red edges in the figure. Cross edges are shown in blue.

The lightest cross edge is $\{c, h\}$.
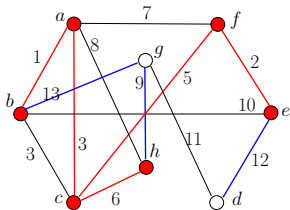
Now $S = \{a, b, c, f, e, h\}$. The MST we are growing now contains the red edges in the figure. Cross edges are shown in blue.

The lightest cross edge is $\{g, h\}$.
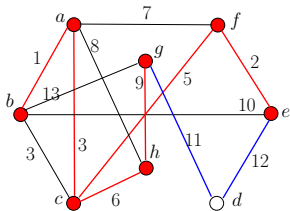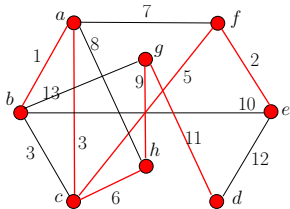
Now $S = \{a, b, c, f, e, h, g\}$. The MST we are growing now contains the red edges in the figure. Cross edges are shown in blue.

The lightest cross edge is $\{g, d\}$.

Now $S = \{a, b, c, f, e, h, g, d\}$. We now have the final MST.

Next we will prove that the algorithm is correct, namely, the tree output is indeed an MST. We will do so by induction on the sequence of edges added to the tree. Specifically, the claim to be proven is:

Claim: For any $i \in [1, |V| - 1]$, there is an MST containing all the first i edges chosen by our algorithm.

Correctness Proof

Let us first recall a fundamental property of undirected graphs:

Lemma: Let T be a tree of n vertices. Adding an arbitrary edge between two vertices in T introduces a cycle.

Proof: Suppose that the edge is added between u and v. Before the edge was added, there is already a path allowing us to go from u to v in T. Therefore, the edge $\{u, v\}$ allows us to move from v back to u, thus witnessing a cycle. □

Now we proceed to prove the claim on Slide 16.

Base Case: $i = 1$. Let $\{u, v\}$ be an edge with the smallest weight in the graph. We will prove that the edge must exist in some MST.

Take any MST T that does not contain edge $\{u, v\}$. Add the edge to T, which creates a cycle. Remove an arbitrary edge $e \neq \{u, v\}$ in the cycle, which gives a new tree T'. Since $\{u, v\}$ has the smallest weight, the cost of T' is smaller than or equal to that of T. This means that T' is also an MST.

Hence, the claim holds for $i = 1$.

$\boxed{\text{Correctness Proof}}$

Inductive Case: Assuming that the claim holds for $i \leq k - 1$ ($k \geq 2$), next we prove that it also holds for $i = k$. Let $\{u, v\}$ be the k-th edge added by our algorithm, and S be the set of vertices already in the algorithm's tree before the addition of $\{u, v\}$. Without loss of generality, suppose that $u \in S$, and $v \notin S$.

By the inductive assumption, we know that there is some MST T that includes all the first $k - 1$ edges. If T also includes edge $\{u, v\}$, then the claim already holds.

Next, we consider the case where T does not have $\{u, v\}$.

We add $\{u, v\}$ to T, which creates a cycle. Let us walk on this cycle starting from v, cross $\{u, v\}$ into S, keep walking within S until traveling out of S for the first time. Let the edge that brought us out of S be $\{u', v'\}$.
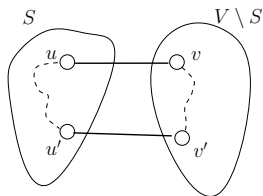
Note that both $\{u, v\}$ and $\{u', v'\}$ are extension edges right before the moment our algorithm picks the k-th edge. Since $\{u, v\}$ has the smallest weight among all the extension edges, we know that the weight of $\{u, v\}$ is smaller than or equal to that of $\{u', v'\}$.

Correctness Proof

Now, remove edge $\{u', v'\}$ from T, which gives another tree T'. The cost of T' cannot be more than that of T. This means that T' must also be an MST.

We thus have proved that the claim holds for $i = k$ as well. $\qquad\square$

Running Time

Think: How to implement Prim's algorithm in $O((|V| + |E|) \cdot \log |V|)$ time? (We will discuss this in a tutorial.)