Minimum Spanning Trees

Yufei Tao

ITEE
University of Queensland

In this lecture, we will study another classic problem: finding a minimum spanning tree of an undirected weighted graph. Interestingly, even though the problem appears rather different from SSSP (single source shortest path), it can be solved by an algorithm that is reminiscent of Dijkstra's algorithm.

Undirected Weighted Graphs

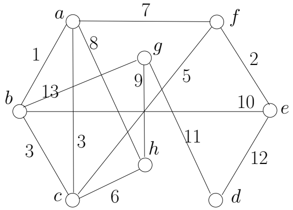
Let $G = (V, E)$ be an undirected graph. Let w be a function that maps each edge of G to a positive integer value. Specifically, for each edge e, $w(e)$ is a positive integer value, which we call the weight of e.

A undirected weighted graph is defined as the 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.

Example



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

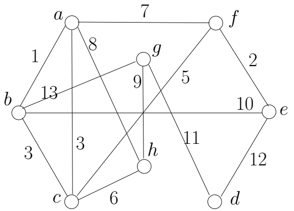
Remember that a tree is defined as a connected undirected graph with no cycles.

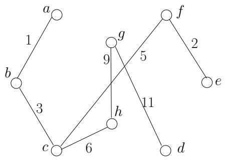
Given a connected undirected weighted graph (G, w) with $G = (V, E)$, a spanning tree T is a tree satisfying the following conditions:

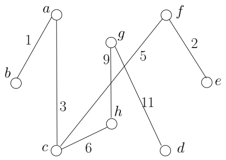
- The vertex set of T is V.
- Every edge of T is an edge in G.

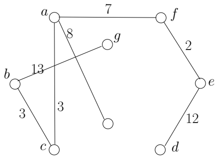
The cost of T is defined as the sum of the weights of all the edges in T (note that T must have $|V| - 1$ edges).

Example









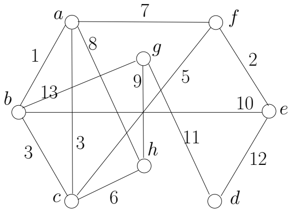
The second row shows three spanning trees (of the graph in the first row). The cost of the first two trees is 37, and that of the right tree is 48.

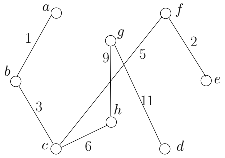
The Minimum Spanning Tree Problem

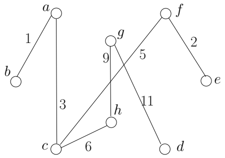
Given a connected undirected weighted graph (G, w) with $G = (V, E)$, the goal of the minimum spanning tree (MST) problem is to find a spanning tree of the smallest cost.

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

Example







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

Prim's Algorithm

Next, we will discuss an algorithm—called Prim's algorithm—for solving the MST problem.

We assume that G is stored in the adjacency list format. Recall that an edge $\{u, v\}$ is represented twice: once by placing u in the adjacency list of of v, and another time by placing v in the adjacency list of u. The weight of $\{u, v\}$ is stored in both places.

Prim's Algorithm

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: $V \setminus S$.

At the end of the algorithm, $S = V$.

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

At all times, the algorithm enforces the following **lightest extension principle**:

- For every vertex $v \in V \setminus S$, it remembers which extension edge of v has the smallest weight—referred to as the lightest extension edge of v, and denoted as $\textit{best-ext}(v)$.

Prim's Algorithm

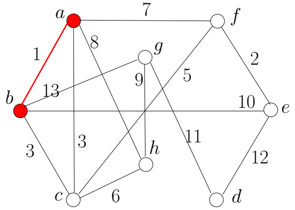
1. Let $\{u, v\}$ be an edge with the smallest weight among all edges.
2. Set $S = \{u, v\}$. Initialize a tree T_{mst} with only one edge $\{u, v\}$.
3. Enforce the lightest extension principle:
    - For every vertex z of $V \setminus S$
        - If z is a neighbor of u, but not of v
          $\textit{best-ext}(z) =$ edge $\{z, u\}$
        - If z is a neighbor of v, but not of u
          $\textit{best-ext}(z) =$ edge $\{z, v\}$
        - Otherwise
          $\textit{best-ext}(z) =$ the lighter edge between $\{z, u\}$ and $\{z, v\}$

Prim's Algorithm

```
4. Repeat the following until $S = V$:
    5. Get an extension edge $\{u, v\}$ with the smallest weight
       /* Without loss of generality, suppose $u \in S$ and $v \notin S$ */
    6. Add $v$ into $S$, and add edge $\{u, v\}$ into $T_{mst}$
       /* Next, we restore the lightest extension principle. */
    7. for every edge $\{v, z\}$ of $v$:
        • If $z \notin S$ then
              If $best\text{-}ext(z)$ is heavier than edge $\{v, z\}$ then
                  Set $best\text{-}ext(z) =$ edge $\{v, z\}$
```

Example

Edge $\{a, b\}$ is the lightest of all. So, at the beginning $S = \{a, b\}$. The MST we are growing now has one edge $\{a, b\}$.

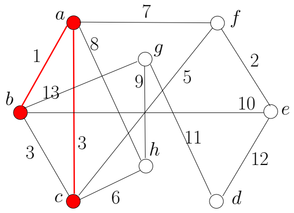


vertex v	*best-ext*(v) and weight
a	n/a
b	n/a
c	$\{c, a\}$, 3
d	nil, ∞
e	$\{e, b\}$, 10
f	$\{a, f\}$, 7
g	$\{g, b\}$, 13
h	$\{a, h\}$, 8

Note: Edges $\{c, a\}$ and $\{c, b\}$ have the same weight. Either of them can be *best-ext*(c).

Example

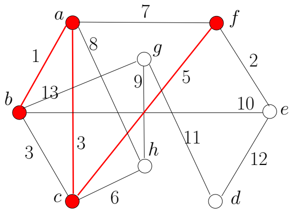
Edge $\{c, a\}$ is the lightest extension edge. So, we add c to S, which is now $S = \{a, b, c\}$. Add edge $\{c, a\}$ into the MST.



vertex v	*best-ext*(v) and weight
a	n/a
b	n/a
c	n/a
d	nil, ∞
e	$\{e, b\}$, 10
f	$\{c, f\}$, 5
g	$\{g, b\}$, 13
h	$\{c, h\}$, 6

Example

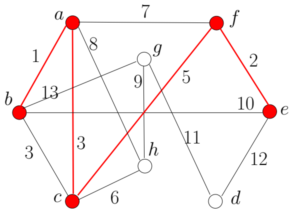
Edge $\{c, f\}$ is the lightest extension edge. So, we add f to S, which is now $S = \{a, b, c, f\}$. Add edge $\{c, f\}$ into the MST.



vertex v	*best-ext*(v) and weight
a	n/a
b	n/a
c	n/a
d	nil, ∞
e	$\{e, f\}$, 2
f	n/a
g	$\{g, b\}$, 13
h	$\{c, h\}$, 6

Example

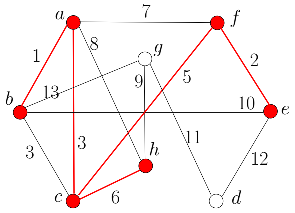
Edge $\{e, f\}$ is the lightest extension edge. So, we add e to S, which is now $S = \{a, b, c, f, e\}$. Add edge $\{e, f\}$ into the MST.



vertex v	*best-ext*(v) and weight
a	n/a
b	n/a
c	n/a
d	$\{e, d\}$, 12
e	n/a
f	n/a
g	$\{g, b\}$, 13
h	$\{c, h\}$, 6

Example

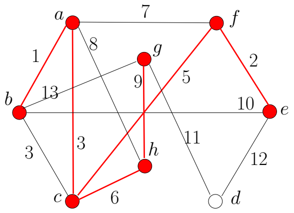
Edge $\{c, h\}$ is the lightest extension edge. So, we add h to S, which is now $S = \{a, b, c, f, e, h\}$. Add edge $\{c, h\}$ into the MST.



vertex v	*best-ext*(v) and weight
a	n/a
b	n/a
c	n/a
d	$\{e, d\}$, 12
e	n/a
f	n/a
g	$\{g, h\}$, 9
h	n/a

Example

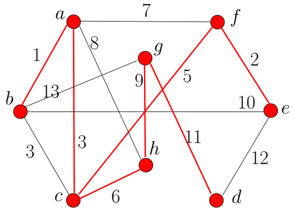
Edge $\{g, h\}$ is the lightest extension edge. So, we add h to S, which is now $S = \{a, b, c, f, e, h\}$. Add edge $\{g, h\}$ into the MST.



vertex v	*best-ext*(v) and weight
a	n/a
b	n/a
c	n/a
d	$\{d, g\}$, 11
e	n/a
f	n/a
g	n/a
h	n/a

Example

Finally, edge $\{d, g\}$ is the lightest extension edge. So, we add d to S, which is now $S = \{a, b, c, f, e, h, g, d\}$. Add edge $\{d, g\}$ into the MST.



vertex v	*best-ext*(v) and weight
a	n/a
b	n/a
c	n/a
d	n/a
e	n/a
f	n/a
g	n/a
h	n/a

We have obtained our 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 the edges added to the tree. Specifically, the claim to be proven is:

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

Then the algorithm's correctness follows from the above claim at $i = |V| - 1$.

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. $\square$

Correctness Proof

Now we proceed to prove the claim on Slide 20.

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 in T such that $e \neq \{u, v\}$, 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 $i = 1$.

Correctness Proof

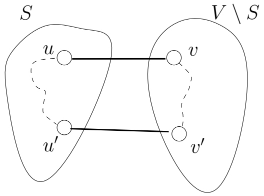
Inductive Case: Assuming that the claim holds for $i \leq k-1$ ($k \geq 2$), next we prove that it also holds for $i = k$. Let $\{u, v\}$ be the k-th edge added by our algorithm, and S be the set of vertices already in the algorithm's tree before the addition of $\{u, v\}$. Without loss of generality,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

Correctness Proof

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. $\square$

Running Time

It will be left as an exercise for you to apply the data structures you have learned to implement Prim's algorithm in $O((|V| + |E|) \cdot \log |V|)$ time.

Remark: Using again an advanced data structure (called the Fibonacci Heap) that will not be covered in this course, we can improve the running time to $O(|V| \log |V| + |E|)$.