CSCI3160 Design and Analysis of Algorithms (2025 Fall)

Greedy 2: Minimum Spanning Trees

Instructor: Xiao Liang[1]

Department of Computer Science and Engineering
Chinese University of Hong Kong

[1] These slides are primarily based on materials prepared by Prof. Yufei Tao (please refer to Prof. Tao's version from 2024 Fall for the original content). Some modifications have been made to better align with this year's teaching progress, incorporating student feedback, in-class interactions, and my own teaching style and research perspective.

Undirected Weighted Graphs

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). Note that the ordering of u, v does not matter (i.e., G is undirected).

We consider that G is **connected**, namely, there is a path between **any** two vertices in V.

Motivation: Cheapest Network

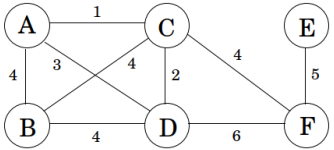


Figure: Picture taken from [Section 5.1, DPV]

Suppose you must link a collection of computers/sites (as shown above):

- Nodes are computers (or cities); edges are potential links.
- Each link has a maintenance cost (edge weight).
- Goal: connect everyone while minimizing total cost.

What structure should the optimal network have?

A Running Example

After some initial thoughts, we realize that the solution must be:

- **Connected**: to ensure all the sites are connected
- **Acyclic**: the optimal set of edges cannot contain a cycle, because removing an edge from this cycle would reduce the cost without compromising connectivity!

Undirected graphs of this kind are called **trees**. The particular tree we want is the one with minimum total weight, known as the **Minimum Spanning Tree**.

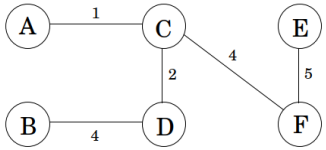


Figure: A MST for the Previous Graph

Formal Definition

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.

That is, it is a tree (i.e., connected and contains no cycles) obtained by deleting some edges from G while ensuring that all the vertices of G are retained (i.e., do not become isolated).

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).

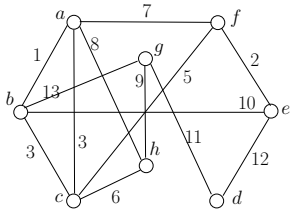
More Motivating Examples for MST

- Networking: Connect data centers or office sites with cables (links have costs), where the goal is to connect everyone at minimum total cost.
- Transportation: Build roads, pipelines, or power lines to reach all cities while minimizing construction cost.
- Clustering in ML: Use MST to reveal structure; cut the largest edges to form clusters (single-linkage).

An important shared pattern: We must connect all points while keeping total cost small.

A More Sophisticated Example

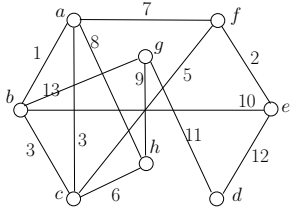
Example

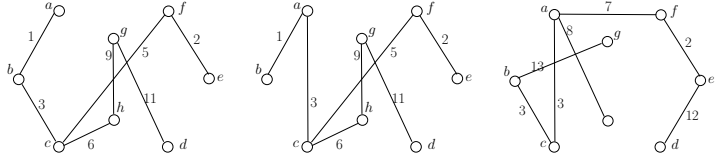


The integer on each edge indicates its weight. For example, the weight of (g, h) is 9, and that of (d, g) is 11.

A More Sophisticated Example

Example





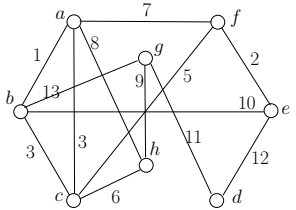
The second row shows three spanning trees (of the graph in the first row). The cost of the first two trees is 37, and that of the right tree is 48.

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 (of cost 37). This means that MSTs may not be unique.

Prim's Algorithm for MST

Choose an arbitrary vertex as your starting point (Think: why doesn't it matter where you start?)

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: $V \setminus S$.

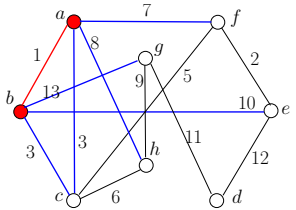
If an edge connects a vertex in S and a vertex in $V \setminus S$, we call it a **cross edge**.

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

Example

Say, we start from vertex a. So, $S = \{a\}$ in the beginning.

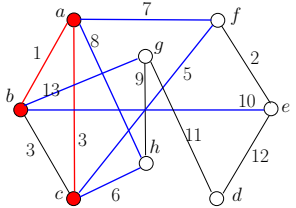
After the first iteration, $S = \{a, b\}$. The MST we are growing now contains the only red edge in the figure. Cross edges are shown in blue.



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).

Example

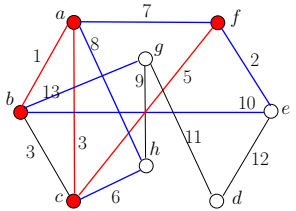
Now $S = \{a, b, c\}$. The MST we are growing now contains the red edges in the figure. Cross edges are shown in blue.



Note that (b, c) is not a cross edge.
The lightest cross edge is (c, f).

Example

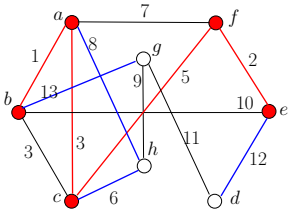
Now $S = \{a, b, c, f\}$. The MST we are growing now contains the red edges in the figure. Cross edges are shown in blue.



The lightest cross edge is (e, f).

Example

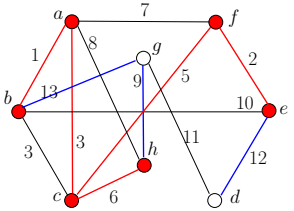
Now $S = \{a, b, c, f, e\}$. The MST we are growing now contains the red edges in the figure. Cross edges are shown in blue.



The lightest cross edge is (c, h).

Example

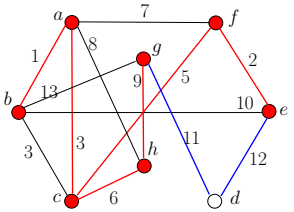
Now $S = \{a, b, c, f, e, h\}$. The MST we are growing now contains the red edges in the figure. Cross edges are shown in blue.



The lightest cross edge is (g, h).

Example

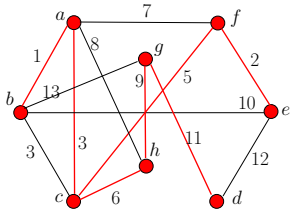
Now $S = \{a, b, c, f, e, h, g\}$. The MST we are growing now contains the red edges in the figure. Cross edges are shown in blue.



The lightest cross edge is (g, d).

Example

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



Running Time

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

Proving the Correctness of Prim

We will prove correctness by induction:

- Maintain an invariant condition through the induction steps:
  - **Invariant Condition:** the edges selected so far by Prim are contained in **some** MST.
- At each induction step, we use an **exchange argument** to update the tree. Here, we will rely on three easy-yet-crucial graph-theoretic facts.
- At the end of the induction: the invariant condition implies that we have a spanning tree that is also minimal (aka a MST). (This step does not trivially follow from the invariant condition. It requires some extra efforts.)

In the following, we first state the three graph-theoretic facts, and then present the induction-based proof.

Graph Fact 1: Adding Edge to Tree Forms Cycle

Fact (Fact 1)

If $T = (E_T, V_T)$ is a tree and an edgee $\notin E_T$ connects two vertices in V_T, then the following graph

$$T' := (E_T \cup \{e\}, V_T)$$

contains exactly one cycle.

Proof.

- Since T is connected, there is already a unique path between the endpoints of e.
- Adding e creates a cycle by joining the ends of this path.
- As trees are acyclic, this must be the only cycle. [Think: how to formally prove this statement?]

□

Graph Fact 2: Removing Edge from Cycle Keeps Connectivity

Fact (Fact 2)

Let C be a set of edges that form a cycle in a connected graph. Removing any edge e from C keeps the graph connected.

Proof.

- Since the removed edge was part of a cycle, there remains an alternative path between its endpoints.
- Thus, all vertices remain reachable from one another.

□

Graph Fact 3

Fact (Fact 3)

Let $G = (V, E)$ be an undirected graph, and let T be a subgraph of G. Then the following are equivalent:

1. *T is a* **spanning tree** *of G* *(Note that we didn't say "minimum" spanning tree here.)*
2. *T is* **connected** *and has exactly $|V| - 1$ edges*

We prove this fact in the following two slides.

Direction 1: Spanning Tree $\Rightarrow$ Connectivity and $|V| - 1$ Edges

Assume: T is a spanning tree of G

Then by definition:

- T is connected
- T is acyclic
- T spans all vertices in V

Since any tree on $|V|$ vertices has exactly $|V| - 1$ edges, thus we know that T is connected and has exactly $|V| - 1$ edges.

Direction 2: Connectivity and $|V| - 1$ Edges $\Rightarrow$ Spanning Tree

Assume: T (being a subgraph of G) is connected and has exactly $|V| - 1$ edges.

1. First, note that T must be acyclic.
   Proof: Otherwise, removing one edge from the cycle does not break connectivity; however, it is impossible to have a connected graph of $|V|$ vertices but with only $|V| - 2$ edges.
2. Also, T must span all vertices.
   Proof: Otherwise, T can contain at most $|V| - 1$ vertices. However, recall that T has only $|V| - 1$ edges. These two conditions imply that T must contains a cycle, contradicting to the acyclic property we just established for T above.

In summary: T is connected, acyclic, and spans all vertices $\Rightarrow$ T is a spanning tree.

Starting the Proof of the Correctness of Prim

Summarizing the three facts:

1. **Fact 1:** If $T = (E_T, V_T)$ is a tree and an edgee $\notin E_T$ connects two vertices in V_T, then the graph $T' := (E_T \cup \{e\}, V_T)$ contains exactly one cycle.
2. **Fact 2:** Let C be a set of edges that form a cycle in a connected graph. Removing any edge e from C keeps the graph connected.
3. **Fact 3:** Let $G = (V, E)$ be an undirected graph, and let T be a subgraph of G. Then the following are equivalent:
   - T is a **spanning tree** of G
   - T is **connected** and has exactly $|V| - 1$ edges

With these three facts, we proceed to prove the correctness of Prim, using mathematical induction.

Inductive Proof Setup (for the Correctness of Prim)

Let:

- $G = (V, E)$: connected, undirected graph with edge weights.
- T: tree built by Prim's algorithm.
- T^*: any MST of G.
- E_k: first k edges selected by Prim.
- $S_k \subset V$: vertices included so far.

Through out the steps of the induction, we will maintain an **invariant condition**:

- in the k-th step, the tree formed by E_k is a sub-tree of some MST.

Base Case

- Initially, $E_0 = \emptyset$
- Any MST trivially contains E_0
- Invariant holds: E_0 is a subgraph of T^*.

[Some students are not comfortable with using the "corner case" E_0 as the induction basis. You could simply run the algorithm for one step and use E_1 as your induction basis.]

Inductive Hypothesis

Suppose after k steps:

- Prim has selected E_k
- **Invariant Condition** is true: There exists an MST T^* such that $E_k \subseteq T^*$

Next: Show that after adding edge e in the $k+1$-th step, the new set $E_{k+1} = E_k \cup \{e\}$ is also a sub-tree of some MST (that might be different from T^*).

Step $k+1$: Two Cases

Let $e = (u, v)$ be the next edge chosen by Prim:

- $u \in S_k$, $v \notin S_k$
- e is the minimum-weight edge crossing the cut

There are two possibilities:

- **Case 1:** $e \in T^* \Rightarrow E_{k+1}$ is a sub-tree of T^*
- **Case 2:** $e \notin T^*$: We will use an exchange argument to construct another MST T' such that E_{k+1} is a sub-tree of T'.

Exchange Argument (Case 2)

Since T^* is a spanning tree:

- There is a unique path in T^* between u and v (recall that they are the two end points of the edge e)
- This path must contain some edge f crossing the cut $(S_k, V \setminus S_k)$

Construct:

$$T' := T^* + e - f$$

That is, T' is obtained by adding edge e to T^* , and then removing edge f from it.

It is obvious that $E_{k+1} (= E_k \cup \{e\})$ is a sub-graph of T'.

We next prove that T' is also a MST for G.

We first prove that T' is a spanning tree:

- By Fact 1: $T^* + e$ contains a cycle. (Also note that f is a part of this cycle.)
- By Fact 2: T' $(= T^* + e - f)$ is connected.
- Since T^* is a MST, it follows from by Fact 3 that T^* has exactly $|V| - 1$ edges. This implies that T' must also has $|V| - 1$ edges (as it is obtained by $T' = T^* + e - f$).

The above shows that T' is connected and has exactly $|V| - 1$ edges. Then, by Fact 3, T' is a spanning tree.

We now prove that T' is MST:

- e is the lightest edge across the cut
- So $w(e) \leq w(f)$
- T' $(= T^* + e - f)$ has weight $\leq$ weight of T^*
- Therefore, T' is an MST

Termination

(Think: convince yourself that Prim's algorithm terminates after $|V| - 1$ steps)

After $|V| - 1$ steps:

- Prim has selected $|V| - 1$ (distinct) edges. Denote this set of edges by P.
- By induction, P must form a sub-tree of some MST. Denote this MST by T_{final}.
- By Fact 3, T_{final} must have exactly $|V| - 1$ edges.
- Thus, $P = T_{final}$.

$$\Rightarrow \text{Prim's algorithm outputs an MST } (i.e., T_{final})$$