Definitions

- Spanning tree: given a connected, undirected graph \( G=(V,E) \), a spanning tree is a subgraph of \( G \) that is an undirected tree and contains all the vertices of \( G \).

- Minimum spanning tree (MST): is a spanning tree with minimum weight. The weight of a subgraph is the sum of the weights of the edges in the subgraph.
How to find MST?

- Using depth-first or breadth-first search to traverse the graph will yield a spanning tree, but the found spanning tree is not guaranteed to be a MST.

[Diagram of a graph with nodes U, V, W, and S and edges labeled with weights 1, 1, and 8]

- Need a different scheme to traverse the graph

Optimization problem & Greedy approach

- Pick a starting point randomly
- “grow” step by step, each step is the best among all possible choices
- Stop when a stopping criterion is satisfied

Note: being greedy in a short term may not lead to the overall best solution
Growing a MST

Definitions

- Tree vertices: vertices that are in the tree constructed so far
- Fringe vertices: not in the tree, but adjacent to some tree vertices
- Unseen vertices: all others.
primMST(G, n)
1. Initialize all vertices as unseen
2. Select an arbitrary vertex \( s \) to start the tree, mark \( s \) as tree vertex
3. Mark all vertices adjacent to \( s \) as fringe
4. While there are fringe vertices
5. select an edge of minimum weight between a tree vertex \( t \) and a fringe vertex \( v \)
6. mark \( v \) as tree and add edge \((tv)\) to the tree
7. mark all unseen vertices adjacent to \( v \) as fringe
Definitions

- Minimum spanning tree property: Let $T$ be any spanning tree of a connected, weighted graph $G$. For any edge $uv$ of $G$ that is not in $T$, if $uv$ is added to $T$ it creates a cycle and $uv$ is a maximum-weight edge on that cycle, then $T$ has the minimum spanning tree property.

Theorem 8.2
In a connected, weighted graph $G = (V,E,W)$, a spanning tree $T$ is a MST iff $T$ has the MST property.

Proof:
(Only if) A MST $T$ must have the MST property. Otherwise there exists an edge $uv$ not in $T$, and $uv$ is not the maximum weight edge in the cycle formed by adding $uv$ to $T$. The maximum weight edge, say $xy$, is in $T$. Removing $xy$ creates a new spanning tree $T'$ whose weight is less than $T$'s. This contradicts that $T$ is a MST.

(if) A spanning tree $T$ has the MST property, then $T$ must be a MST. Let $T_{\text{min}}$ be any minimum spanning tree of $G$. If $T_{\text{min}}$ and $T$ differ by only one edge; $uv$ is in $T$ but not in $T_{\text{min}}$, $xy$ is in $T_{\text{min}}$ but not in $T$.

If add $xy$ into $T$, a cycle $C$ is created and $uv$ must be on $C$. (why?)
Let's call $T + xy = TC$. TC becomes $T_{\text{min}}$ if $uv$ is removed, i.e., the cycle $C$ is broken.
Because $T$ has MST property, then $W(uv) \leq W(xy)$.
If add $uv$ into $T_{\text{min}}$, due to similar argument, we must have $W(uv) \geq W(xy)$.
Therefore, $W(uv) = W(xy)$. Since $T$ and $T_{\text{min}}$ just differ by edge $uv$ and $xy$, $T$ and $T_{\text{min}}$ must have the same weight. Therefore $T$ is also a minimum spanning tree. By induction, we can prove it is still true when $T$ and $T_{\text{min}}$ differs by more edges. (See Lemma 8.1)

Corollary: If all edge weights are distinct, the MST is unique.
Correctness of Prim’s MST algorithm

- Each time an edge from a tree vertex to a fringe vertex is added into the tree, so never is a cycle created.
- All vertices will be covered to the tree eventually.
- Therefore the final tree is a spanning tree.
- It is also a minimum spanning tree, because
  - At each step, the so far constructed tree has the MST property in its induced graph of G.

Proof by induction:
- Let $T_k$ be the tree in k-th step. Let $G_k$ be the subgraph of G induced by $T_k$ (i.e., $uv$ is an edge in $G_k$ if it is an edge in G and both $u$ and $v$ are in $T_k$)
- $T_1$ has MST property of $G_1$.
- Let assume it is true up to arbitrary $k>0$.
- At step $k+1$, vertex $v$ is added, and $v$ has edge with some vertices $u_1, ..., u_d$ in $T_k$. For definiteness, assume $vu_1$ is the edge of minimum weight among all possibilities. Now $T_{k+1} = T_k + vu_1$, and $G_{k+1} = G_k + vu_1 + ... + vu_d$.
  - To prove $T_{k+1}$ is MST of $G_{k+1}$, we need to prove, for any edge $xy$ in $G_{k+1}$ but not in $T_{k+1}$, if add $xy$ to $T_{k+1}$, $xy$ will be the maximum weight edge in the cycle thus created.
  - If neither of $x$ and $y$ is $v$, then $xy$ is in $G_k$ and $xy$ must be the maximum weight edge since $T_k$ has the MST property. If either $x$ or $y$ is $v$, $xy$ is still the maximum weight edge in the cycle, because no edge on the cycle can have a larger weight due to the way the tree is constructed.
- At step $|V|$, tree $T_{|V|}$ contains all vertices in $G$, and $G$ itself is the induced graph by $T_{|V|}$. Therefore, Spanning tree $T_{|V|}$ is the MST of G.
It is impossible to have edge $rt$ such that $W(rt) > W(vu)$.

Suppose $t$ is added in later than $r$, then Prim algorithm will add edge $(u_1v)$ or $(uv)$ instead of $(rt)$, since $v$ is in the fringe and has an edge of smaller weight.

**Managing the Fringe with a priority queue**

```
primMST(G, r)
1.   for each $u \in Q$
2.      key[u] ← $\infty$
3.      $P[u] \leftarrow \text{nil}$
4.      key[r] ← 0
5.      $Q \leftarrow V[G]$
6.      while $Q$ is not empty
7.         $u \leftarrow \text{Extract-Min}(Q)$
8.         for each $v \in \text{Adj}[u]$
9.             if $v \in Q$ and $w(u,v) < \text{key}[v]$
10.                then $p[v] \leftarrow u$
11.                   key[v] ← $w(u,v)$

Total running time: $O(V \log V + E \log V)$
```

$O(V)$ to build a binary heap

$O(\log(v))$ time to prioritize $v$ in $Q$

$O(E)$, combined with line 6

$O(\log(v))$ times

$O(V)$ to build a binary heap
Kruskal MST Algorithm

kruskalMST(G, n, F)
1. Initialize F // a forest of trees
2. Build a minimizing priority queue Q of edges of G, prioritized by weight
3. Initialize a union-find structure, sets, in which each vertex of G is in its own set.
4. while Q is not empty
5. vwEdge = Extract-Min(Q)
6. int vSet = find(sets, vwEdge.from)
7. int wSet = find(sets, vwEdge.to)
8. do if (vSet ≠ wSet) // make sure v and w not in the same tree, why?
9. then Add vwEdge to F
10. union(sets, vSet, wSet)
11. return F

NOTE: if there are degeneracy in edge weights, MST will not be unique, depending on how ties are resolved in the priority queue.

Kruskal algorithm: example

![Graph example](image-url)
Kruskal algorithm

Time analysis

- Initialization: $O(V)$
- Deleting all edges from the queue: $\Theta(E \log E)$
- Find called $2|E|$ times, union called $|V|$ times, cost is $O(E \log^*(E))$

Total worst-case time: $\Theta(E \log E)$