Minimum Spanning Tree (MST)

- Undirected connected graph $G = (V, E)$
- Weight function $w : E \rightarrow \mathbb{R}$

- Spanning tree: a tree that connects all vertices
- Minimum Spanning Tree $T$:

$$w(T) = \sum_{(u,v)\in T} w(u,v)$$

- MST is not necessarily unique
Basic idea of “growing” a MST:

- construct the MST by successively select edges to include in the tree
- guarantee that after the inclusion of each new selected edge, it forms a subset of some MST.

*One of the most famous greedy algorithms, along with Huffman coding*
MST

Basic properties:

- **Optimal substructure**: optimal tree contains optimal subtrees.

Let $T$ be a MST of $G = (V, E)$. Removing $(u,v)$ of $T$ partitions $T$ into two trees $T_1$ and $T_2$. Then $T_1$ is a MST of $G_1 = (V_1, E_1)$ and $T_2$ is a MST of $G_2 = (V_2, E_2)$.$^1$

**Proof.** Note that

$$w(T) = w(T_1) + w(u, v) + w(T_2).$$

There cannot be a better subtree than $T_1$ or $T_2$, otherwise $T$ would be suboptimal.

---

$^1$The subgraph $G_1$ is induced by vertices in $T_1$, i.e., $V_1 = \{\text{vertices in } T_1\}$ and $E_1 = \{(x, y) \in E; x, y \in V_1\}$. Similarly for $G_2$. 

MST

Basic properties:

- **Greedy-choice property:**

  Let $T$ be a MST of $G = (V, E)$, $A \subseteq T$ be a subtree of $T$, and $(u, v)$ be min-weight edge in $G$ connecting $A$ and $V - A$. Then $(u, v) \in T$.

  **Proof.** If $(u, v) \notin T$, then

  - $(u, v) \cup T$ forms a cycle,
  - replace one of edges of $T$ by $(u, v)$ form a new tree $T$
  - this is contradiction to $T$ is MST

---

\(^2\)Note: there is an abuse of notation here that we will view $A$ as being both edges and vertices.
MST

Prim’s algorithm

- Basic idea:
  - builds one tree, so that $A$ is always a tree
  - starts from a root $r$
  - at each step, find the next lightest edge crossing cut $(A, V - A)$ and add this edge to $A$ ("greedy choice")

- How to find the next lightest edge quickly?

  Answer: use a priority queue
Review: Priority Queue

A priority queue maintains a set $S$ of elements, each with an associated value called a “key”, and supports the following operations:

- **Search($S, k$):** returns $x$ in $S$ with $\text{key}[x] = k$
- **Insert($S, x$)/Delete($S, x$):** inserts/deletes the element $x$ into the set $S$
- **Maximum($S$)/Minimum($S$):** returns $x$ in $S$ with largest/smallest key
- **Extract-max($S$)/Extract-min($S$):** removes and returns $x$ in $S$ with largest/smallest key
- **Increase-key($S, x, k$)/Decrease-key($S, x, k$):** increases/decreases the value of element $x$’s key to the new value $k$

*Recall that the priority queue has been used in Huffman coding.*
MST

MST-Prim(G, w, r)

Q = empty

for each vertex u in V
    key[u] = infty
    pi[u] = Nil
    Insert(Q, u)

endfor

Decrease-key(Q, r, 0)

while Q not empty
    u = Extract-Min(Q)
    for each v in Adj[u]
        if (v in Q) and (w(u,v) < key[v])
            Decrease-key(Q, v, w(u,v))
            pi[v] = u  // parent of v
        endif
    endfor

endwhile

return A = { (v, pi[v]): v in V-{r} }  // MST
MST

**Prim’s algorithm** – running time:

- depends on how the priority queue $Q$ is implemented
- Suppose $Q$ is a binary heap (see Section 6.1)
  - Initialize $Q$ and the first for loop: $O(|V| \lg |V|)$
  - Decrease key of root $r$: $O(\lg |V|)$
  - While-loop:
    - a) $|V|$ Extract-Min calls: $O(|V| \lg |V|)$
    - b) $\leq |E|$ Decrease-Key calls: $O(|E| \lg |E|)$
- Total: $O(|E| \lg |V|)$

**Note**: $G$ is connected, $\lg |E| = \Theta(\lg |V|)$
MST

**Kruskal’s algorithm**

- **Basic idea:**
  - scan edges in increasing of weight
  - put edge in if no loop created

- Why does this result in MST?
  Answer: min-weight edge is always in MST (the greedy-choice property).

- Implementation data structure: **disjoint-set**
Review: Disjoint-Set

Disjoint-Set maintains a collection of \( S = \{ S_1, S_2, \ldots S_k \} \) of disjoint dynamic sets. Each set is identified by a representative, which is some member of the set.

A disjoint-set data structure supports the following operations:

- **Make-set(\( x \)):**
  creates a new set whose only member (and thus representative) is \( x \).

- **Union(\( x, y \)):**
  unites the sets that contain \( x \) and \( y \), say \( S_x \) and \( S_y \), into a new set that is the union of these two sets: \( S_x \cup S_y \). The representative is any member of \( S_x \cup S_y \).

- **Find-set(\( x \)):**
  returns (a pointer to) the representative of the (unique) set containing \( x \).

*To learn more about the disjoint-set data structure, see Chapter 21.*
MST

MST-Kruskal(G, w)
A = empty
for each vertex v in V
    Make-set(v)
endfor
Sort the edges E in nondecreasing order by w
for each edge (u,v) in E, taken in nondecreasing order by w
    if Find-set(u) \= Find-set(v)
        A = A U {(u,v)}
        Union(u,v)
    endif
endfor
return A
**MST**

**Kruskal’s algorithm** – running time:

- depends on the implementation of the disjoint-set
- Sort: $\Theta(|E| \lg |E|)$
- $|V|$ Make-Set ops
- $2|E|$ Find-Set ops
- $|V| - 1$ Union ops
- Total: $O(|E| \lg |V|)$

**Note:** $G$ is connected, $\lg |E| = \Theta(\lg |V|)$