Minimum Spanning Tree

- The Cycle Property (Red Rule)
  - Reverse Delete Algorithm for MST
- Kruskal's Algorithm for MST
- Runtime and Implementation
  - Disjoint Sets Data Structure

Imdad ullah Khan

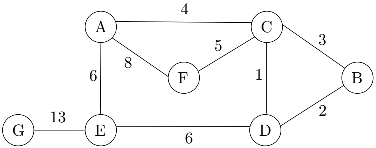
Minimum Spanning Tree: Review

- $T = (V', E')$ is a spanning tree of $G = (V, E)$ if
  - T is a spanning subgraph of G
  - T is a tree
- Weight of a tree T is sum of weights of its edges $w(T) = \sum_{e \in T} w(e)$
- A tree is a (minimally) connected graph with no cycles
- A tree on n vertices has $n - 1$ edges
- A MST is a spanning tree with minimum weight

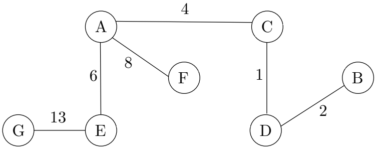
Computing MST is a classic optimization problem with many applications in graph analysis, combinatorial optimization, network formation,..

Minimum Spanning Tree Problem

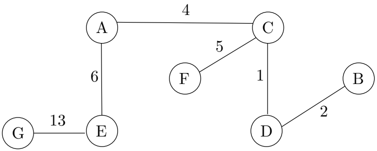
Input: A weighted graph $G = (V, E, w)$, $w : E \to \mathbb{R}$
Output: A spanning tree of G with minimum total weight



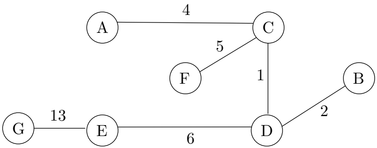
A weighted graph G



A spanning tree of G with weight 34



An MST of G with weight 31



An MST of G with weight 31

MST does not have to be unique

MST Algorithms

Input: An undirected weighted graph $G = (V, E, w)$, $w : E \rightarrow \mathbb{R}$
Output: A spanning tree of G with minimum total weight

We discuss two greedy algorithms to find MST in a graph

- Prim's Algorithm (1957) [also Dijkstra '59, Jarnik '30]
- Kruskal's Algorithm (1956)

We make the following assumptions

1. Input graph G is connected
   - Otherwise there is no spanning tree
   - Easy to check in preprocessing (e.g., BFS or DFS).
   - For disconnected graphs can find minimum spanning forest
2. Edge weights are distinct
   - Otherwise there can be more than one MSTs
   - Algorithms remain correct with arbitrarily breaking ties
   - Analysis is slightly complicated

Cuts in Graphs

A cut in G is a subset $S \subset V$

- Denoted as $[S, \overline{S}]$, $S = \emptyset$ and $S = V$ are trivial cuts
- An edge (u, v) is crossing the cut $[S, \overline{S}]$, if $u \in S$ and $v \in \overline{S}$
- Empty Cut Lemma:
  - A graph G is disconnected iff it has a cut with no crossing edge
- Double Crossing Lemma
  - If a cycle crosses a cut, then it has to cross at least twice
- Lonely Crossing Lemma
  - If e is the only edge crossing a cut $[S, \overline{S}]$, then it is not in any cycle
- The Blue Rule
  - If an edge $e \in E$ is the lightest edge crossing some cut $[S, \overline{S}]$, then e belongs to the MST of G

The cycle property (Red Rule)

If an edge $e \in E$ is the heaviest edge on some cycle C, then e does not belong to the MST of G

This statement assume edge weights are unique. More generally,

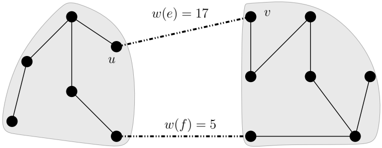
If an edge $e \in E$ is a heaviest edge on some cycle C, then e does not belong to some MST of G

The cycle property (Red Rule): Proof

If an edge $e \in E$ is the heaviest edge on some cycle C, then e does not belong to the MST of G

Proof by contradiction:

- Let $e = (u, v)$ be the heaviest edge across a cycle C
- Suppose $e \in T^*$, "the MST" of G
- Deleting e from T^* disconnects T^* $\quad\triangleright$ a tree is minimally connected
- Let $S = R(u)$ in $T^* \setminus \{e\}$, consider $[S, \overline{S}]$
- e and another edge $f \neq e \in C$ crosses $[S, \overline{S}]$ $\quad\triangleright$ double cut lemma
- $T' = T^* \setminus \{e\} \cup \{f\}$ and $w(T') < w(T^*)$



Reverse Delete Algorithm for MST

Input: An undirected weighted graph $G = (V, E, w)$, $w : E \to \mathbb{R}$
Output: A spanning tree of G with minimum total weight

Algorithm Reverse Delete Algorithm for MST

Sort edges in decreasing order of weights ▷ let $e_1, e_2, \ldots, e_m$ be the sorted order
$G' \leftarrow G$ ▷ Begin with the whole graph
for $i = 1$ to m **do**
  if $G' \setminus \{e_i\}$ is connected **then**
    $G' \leftarrow G' \setminus \{e_i\}$
return G'

Reverse Delete Algorithm

Algorithm Reverse Delete Algorithm for MST

Sort edges in decreasing order of weights ▷ $e_1, e_2, \ldots, e_m$ is sorted order
$G' \leftarrow G$ ▷ Begin with the whole graph
for $i = 1$ to m **do**
  if $G' \setminus \{e_i\}$ is connected **then**
    $G' \leftarrow G' \setminus \{e_i\}$
return G'

Removing an edge does not disconnect a graph iff it is on a cycle

- Since G is connected, by design the returned graph G' is connected
- G' is a tree, $\because$ an edge from a cycle wouldn't disconnect it
- Optimality follows from the **red rule**

If e is the heaviest edge on a cycle, then e is not in the MST

- Check connectivity of $G \setminus \{e_i\}$ by BFS/DFS