Minimum Spanning Tree

- Minimum Spanning Tree
- Prim's Algorithm for MST
- Cuts in Graphs
- Correctness and Optimality of Prim's Algorithm
- Runtime
  - Basic Implementation
  - Vertex-Centric Implementation
  - Heap Based Implementation

Imdad ullah Khan

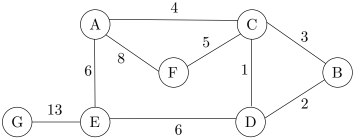
Minimum Spanning Tree: Review

- $T = (V', E')$ 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 connected graph with no cycles
- A tree on n vertices has $n - 1$ edges
- A MST is a spanning tree with minimum weight

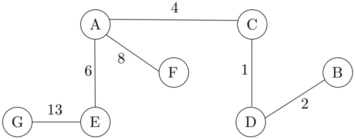
Computing MST is a classic optimization problem with many applications in graph analysis, combinatorial optimization, network formation,..

Minimum Spanning Tree Problem

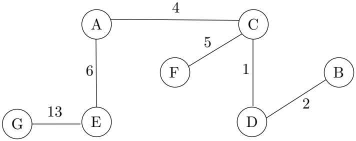
Input: A weighted graph $G = (V, E, w)$, $w : E \to \mathbb{R}$
Output: A spanning tree of G with minimum total weight



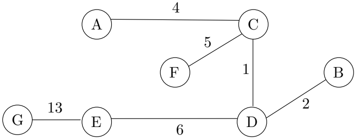
A weighted graph G



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 $G = (V, E, w)$, $w : E \to \mathbb{R}$
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

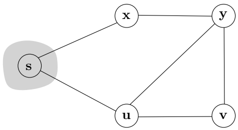
1. Input graph G is connected
   - Otherwise there is no spanning tree
   - Easy to check in preprocessing (e.g., BFS or DFS).
   - For disconnected graphs can find minimum spanning forest
2. Edge weights are distinct
   - Otherwise there can be more than one MSTs
   - Algorithms remain correct with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

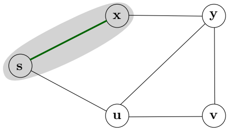
Prim's Algorithm

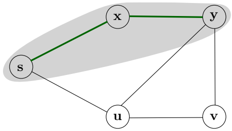
- Maintains a set $R \subset V$ and a tree T spanning vertices in R
- $V(T) = R$ $\triangleright$ vertices spanned by T
- Grow R by adding one vertex v in every iteration
- Grow T by adding an edge connecting v to some vertex in current R
- Initially $R = \{s\}$, an arbitrary vertex and $T = \emptyset$
- Select a minimum crossing edge from R to $\overline{R}$ $\triangleright$ (greedy criteria)

$$\arg\min_{e=(u,v),u\in R,v\notin R} w(e)$$

- Add v to R and e to T



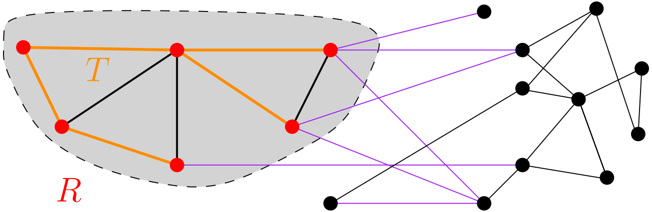




Prim's Algorithm

Algorithm Prim's Algorithm for MST in $G = (V, E, w)$

$R \leftarrow \{s\}$ $\triangleright$ $s \in V$ an arbitrary vertex
$T \leftarrow \emptyset$ $\triangleright$ Begin with an empty tree
while $R \neq V$ **do**
  Get $e = (u, v)$, $u \in R, v \notin R$ with minimum $w(uv)$
  $T \leftarrow T \cup \{e\}$
  $R \leftarrow R \cup \{v\}$



Prim's Algorithm: Example