CSE101: Design and Analysis of Algorithms

Ragesh Jaiswal, CSE, UCSD
**Spanning Tree**: Given a strongly connected graph $G = (V, E)$, a spanning tree of $G$ is a subgraph $G' = (V, E')$ such that $G'$ is a tree.

**Minimum Spanning Tree (MST)**: Given a strongly connected weighted graph $G = (V, E)$, a Minimum Spanning Tree of $G$ is a spanning tree of $G$ of minimum total weight (i.e., sum of weight of edges in the tree).

![Graph Diagram]
Spanning Tree: Given a strongly connected graph $G = (V, E)$, a spanning tree of $G$ is a subgraph $G' = (V, E')$ such that $G'$ is a tree.

Minimum Spanning Tree (MST): Given a strongly connected weighted graph $G = (V, E)$, a Minimum Spanning Tree of $G$ is a spanning tree of $G$ of minimum total weight (i.e., sum of weight of edges in the tree).
Problem

Given a strongly connected weighted graph $G$ where all the edge weights are distinct, give an algorithm for finding the MST of $G$. 
Theorem

Cut property: Given a strongly connected weighted graph $G = (V, E)$ where all the edge weights are distinct. Consider a non-empty proper subset $S$ of $V$ and $S' = V \setminus S$. Let $e$ be the least weighted edge between any pair of vertices $(u, v)$, where $u$ is in $S$ and $v$ is in $S'$. Then $e$ is necessarily present in all MSTs of $G$. 

![Diagram showing the cut property with sets $S$ and $S'$ and edges $e$, $e'$, and $e''$.]
Greedy Algorithms
Minimum Spanning Tree

**Theorem**
Cut property: Given a strongly connected weighted graph $G = (V, E)$ where all the edge weights are distinct. Consider a non-empty proper subset $S$ of $V$ and $S' = V \setminus S$. Let $e$ be the least weighted edge between any pair of vertices $(u, v)$, where $u$ is in $S$ and $v$ is in $S'$. Then $e$ is necessarily present in all MSTs of $G$.

**Proof sketch**
- For the sake of contradiction, assume that there is a MST $T$ that does not contain edge $e = (u, v)$.
- **Claim 1**: Any path from $u$ to $v$ in tree $T$ will contain a cut-edge.
- Consider any path from $u$ to $v$ in tree $T$ and let $e'$ be the first cut-edge in this path. Consider graph $T'$ obtained by removing $e'$ from $T$ and adding $e$.
- **Claim 2**: The sum total weight of edges of $T'$ is smaller than that of $T$.
- **Claim 3**: $T'$ is strongly connected.
Greedy Algorithms
Minimum Spanning Tree

Algorithm

**Prim’s Algorithm(G)**
- \( S \leftarrow \{u\} \) //\( u \) is an arbitrary vertex in the graph
- \( T \leftarrow \{\} \)
- While \( S \) does not contain all vertices
  - Let \( e = (v, w) \) be the minimum weight edge between \( S \) and \( V \setminus S \)
  - \( T \leftarrow T \cup \{e\} \)
  - \( S \leftarrow S \cup \{w\} \)

Algorithm

**Kruskal’s Algorithm(G)**
- \( S \leftarrow E; \ T \leftarrow \{\} \)
- While the edge set \( T \) does not connect all the vertices
  - Let \( e \) be the minimum weight edge in the set \( S \)
  - If \( e \) does not create a cycle in \( T \)
    - \( T \leftarrow T \cup \{e\} \)
    - \( S \leftarrow S \setminus \{e\} \)
Greedy Algorithms
Minimum Spanning Tree

Algorithm

Prim’s Algorithm(G)
- \( S \leftarrow \{u\} \) // \( u \) is an arbitrary vertex in the graph
- \( T \leftarrow \{\} \)
- While \( S \) does not contain all vertices
  - Let \( e = (v, w) \) be the minimum weight edge between \( S \) and \( V \setminus S \)
  - \( T \leftarrow T \cup \{e\} \)
  - \( S \leftarrow S \cup \{w\} \)
Greedy Algorithms
Minimum Spanning Tree

Algorithm

Prim’s Algorithm (G)
- \( S \leftarrow \{u\} \) \(/u\ is\ an\ arbitrary\ vertex\ in\ the\ graph/\)
- \( T \leftarrow \{\} \)
- While \( S \) does not contain all vertices
  - Let \( e = (v, w) \) be the minimum weight edge between \( S \) and \( V \setminus S \)
  - \( T \leftarrow T \cup \{e\} \)
  - \( S \leftarrow S \cup \{w\} \)
Algorithm

Prim’s Algorithm \((G)\)
- \(S \leftarrow \{u\}\) // \(u\) is an arbitrary vertex in the graph
- \(T \leftarrow \{\}\)
- While \(S\) does not contain all vertices
  - Let \(e = (v, w)\) be the minimum weight edge between \(S\) and \(V \setminus S\)
  - \(T \leftarrow T \cup \{e\}\)
  - \(S \leftarrow S \cup \{w\}\)
Greedy Algorithms
Minimum Spanning Tree

Algorithm

Prim’s Algorithm(G)
- \( S \leftarrow \{u\} \) // \( u \) is an arbitrary vertex in the graph
- \( T \leftarrow \{\} \)
- While \( S \) does not contain all vertices
  - Let \( e = (v, w) \) be the minimum weight edge between \( S \) and \( V \setminus S \)
  - \( T \leftarrow T \cup \{e\} \)
  - \( S \leftarrow S \cup \{w\} \)
Greedy Algorithms
Minimum Spanning Tree

Algorithm

Prim’s Algorithm(G)

- $S \leftarrow \{u\}$  // $u$ is an arbitrary vertex in the graph
- $T \leftarrow \{\}$
- While $S$ does not contain all vertices
  - Let $e = (v, w)$ be the minimum weight edge between $S$ and $V \setminus S$
  - $T \leftarrow T \cup \{e\}$
  - $S \leftarrow S \cup \{w\}$
Greedy Algorithms
Minimum Spanning Tree

Algorithm

Prim’s Algorithm(G)
- $S \leftarrow \{u\}$ // $u$ is an arbitrary vertex in the graph
- $T \leftarrow \{\}$
- While $S$ does not contain all vertices
  - Let $e = (v, w)$ be the minimum weight edge between $S$ and $V \setminus S$
  - $T \leftarrow T \cup \{e\}$
  - $S \leftarrow S \cup \{w\}$
Algorithm

Kruskal’s Algorithm(G)
- \( S \leftarrow E; \ T \leftarrow \{\} \)
- While the edge set \( T \) does not connect all the vertices
  - Let \( e \) be the minimum weight edge in the set \( S \)
  - If \( e \) does not create a cycle in \( T \)
    - \( T \leftarrow T \cup \{e\} \)
  - \( S \leftarrow S \setminus \{e\} \)
Greedy Algorithms
Minimum Spanning Tree

Algorithm

Kruskal’s Algorithm(G)
- \( S \leftarrow E; \ T \leftarrow \{\} \)
- While the edge set \( T \) does not connect all the vertices
  - Let \( e \) be the minimum weight edge in the set \( S \)
  - If \( e \) does not create a cycle in \( T \)
    - \( T \leftarrow T \cup \{e\} \)
  - \( S \leftarrow S \setminus \{e\} \)
Greedy Algorithms
Minimum Spanning Tree

Algorithm

Kruskal’s Algorithm(G)
- \( S \leftarrow E; \ T \leftarrow \{\} \)
- While the edge set \( T \) does not connect all the vertices
  - Let \( e \) be the minimum weight edge in the set \( S \)
  - If \( e \) does not create a cycle in \( T \)
    - \( T \leftarrow T \cup \{e\} \)
  - \( S \leftarrow S \setminus \{e\} \)
Algorithm

Kruskal’s Algorithm(G)
- \( S \leftarrow E; \ T \leftarrow \{} \)
- While the edge set \( T \) does not connect all the vertices
  - Let \( e \) be the minimum weight edge in the set \( S \)
  - If \( e \) does not create a cycle in \( T \)
    - \( T \leftarrow T \cup \{e\} \)
  - \( S \leftarrow S \setminus \{e\} \)
Kruskal’s Algorithm (G)

- \( S \leftarrow E; \ T \leftarrow \{\} \)
- While the edge set \( T \) does not connect all the vertices
  - Let \( e \) be the minimum weight edge in the set \( S \)
  - If \( e \) does not create a cycle in \( T \)
    - \( T \leftarrow T \cup \{e\} \)
  - \( S \leftarrow S \setminus \{e\} \)
Greedy Algorithms
Minimum Spanning Tree

Algorithm

Kruskal’s Algorithm(G)
- $S \leftarrow E; \ T \leftarrow \{}$
- While the edge set $T$ does not connect all the vertices
  - Let $e$ be the minimum weight edge in the set $S$
  - If $e$ does not create a cycle in $T$
    - $T \leftarrow T \cup \{e\}$
  - $S \leftarrow S \setminus \{e\}$

---

Ragesh Jaiswal, CSE, UCSD

CSE101: Design and Analysis of Algorithms
Greedy Algorithms
Minimum Spanning Tree

Algorithm

Prim’s Algorithm(G)
- \( S \leftarrow \{u\} // u \text{ is an arbitrary vertex in the graph} \)
- \( T \leftarrow \{\} \)
- While \( S \) does not contain all vertices
  - Let \( e = (v, w) \) be the minimum weight edge between \( S \) and \( V \setminus S \)
  - \( T \leftarrow T \cup \{e\} \)
  - \( S \leftarrow S \cup \{w\} \)

What is the running time of Prim’s algorithm?
Greedy Algorithms
Minimum Spanning Tree

Algorithm

Prim’s Algorithm(G)
- $S \leftarrow \{u\}$ //u is an arbitrary vertex in the graph
- $T \leftarrow \{\}$
- While $S$ does not contain all vertices
  - Let $e = (v, w)$ be the minimum weight edge between $S$ and $V \setminus S$
  - $T \leftarrow T \cup \{e\}$
  - $S \leftarrow S \cup \{w\}$

What is the running time of Prim’s algorithm?
$O(|E| \cdot \log |V|)$
- Using a priority queue.
End