Graph Algorithms: Spanning Trees, Kruskal's and Prim's Algorithm

Graph Theory Notes

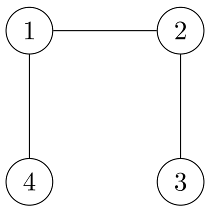
1 Spanning Tree

A **Spanning Tree** of a graph is a subgraph that includes all the vertices of the graph, connected with the minimal possible number of edges, i.e., *n - 1* edges for *n* vertices, and it contains no cycles.

- The spanning tree contains all the vertices of the graph.
- There must be exactly *n - 1* edges where *n* is the number of vertices.
- There should be no cycles.

Example: Here is a spanning tree example for a simple graph:

$$G(V, E) = (\{1, 2, 3, 4\}, \{(1, 2), (1, 4), (2, 3)\})$$



The above figure shows a spanning tree of a graph where all vertices are included and no cycles are formed.

2 Kruskal's Algorithm

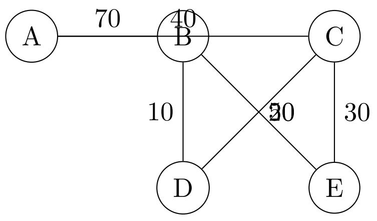
Kruskal's Algorithm is a **greedy** approach for finding the Minimum Spanning Tree (MST) of a graph. The steps involved are:

1. Sort all edges in non-decreasing order of their weights.
2. Pick the smallest edge. Check if it forms a cycle with the spanning tree formed so far. If a cycle is not formed, include this edge. Else, discard it.
3. Repeat step 2 until there are *n - 1* edges in the spanning tree.

Example:
For the following graph:

$$G(V, E) = (\{A, B, C, D, E\}, \{(A, B, 70), (A, C, 40), (B, D, 10), (B, E, 20), (C, D, 50), (C, E, 30)\})$$

We can apply Kruskal's Algorithm step by step:
1. Sort edges: $(B, D, 10), (B, E, 20), (C, E, 30), (A, C, 40), (C, D, 50), (A, B, 70)$
2. Add edges to the MST while avoiding cycles.



After applying Kruskal's Algorithm, the Minimum Spanning Tree is formed by selecting the edges (B, D), (B, E), (C, E), and (A, C) with a total cost of:

$$10 + 20 + 30 + 40 = 100$$

Minimum Spanning Tree (MST)

3 Minimum Spanning Tree (MST)

A **Minimum Spanning Tree (MST)** is a subgraph of a connected, weighted graph that:

- Includes all vertices of the original graph.
- Contains exactly $n - 1$ edges where n is the number of vertices in the graph.
- Has no cycles.
- Minimizes the total weight of the edges.

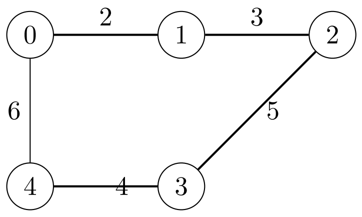
The MST can be defined as a subset of the original graph such that:

$$\text{MST} \subseteq G, \text{ and } \sum_{e \in \text{MST}} w(e) \text{ is minimized,}$$

where $w(e)$ denotes the weight of edge e.
Example
Let's consider a graph with the following vertices and edges:

$$V = \{0, 1, 2, 3, 4\}$$
$$E = \{(0, 1, 2), (1, 2, 3), (2, 3, 5), (3, 4, 4), (4, 0, 6)\}$$

The minimum spanning tree for this graph, based on Prim's Algorithm or Kruskal's Algorithm, would connect the vertices with the minimum sum of edge weights.



The edges included in the MST are:

$$\{(0, 1, 2), (1, 2, 3), (2, 3, 5), (3, 4, 4)\}$$

$$\text{Total Weight} = 2 + 3 + 5 + 4 = 14$$

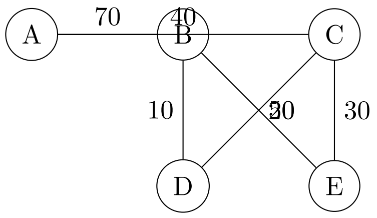
4 Prim's Algorithm

Prim's Algorithm is another greedy algorithm for finding the Minimum Spanning Tree. It starts with a single vertex and repeatedly adds the smallest edge connecting the spanning tree to a vertex outside of the tree.

The steps are:

1. Initialize the MST with a single vertex.
2. Find the minimum weight edge that connects a vertex in the MST to a vertex outside the MST.
3. Repeat step 2 until all vertices are included in the MST.

Example:
We will apply Prim's algorithm to the same graph. Starting with vertex B:



Step 1: Start with vertex B, choose the smallest edge $(B, D, 10)$.
Step 2: From D, choose the next smallest edge $(B, E, 20)$.
Step 3: From B and D, choose the next smallest edge $(C, E, 30)$.
Step 4: Finally, choose the edge $(A, C, 40)$.
The resulting MST has a total cost of:

$$10 + 20 + 30 + 40 = 100$$

5 Time Complexity

- **Kruskal's Algorithm**: The time complexity is $O(E \log E)$, where E is the number of edges.
- **Prim's Algorithm**: The time complexity is $O(E + V \log V)$, where V is the number of vertices and E is the number of edges.