Minimum Spanning Tree (MST)

Kruskal's Algorithm

Kruskal’s Algorithm is a greedy algorithm used to find the Minimum Spanning Tree (MST) of a connected, undirected, and weighted graph.

Key Objectives:

- Connect all vertices with the **minimum total edge weight**
- **Avoid forming cycles**
- The resulting structure is a **tree** that spans all vertices

Real-World Analogy

Imagine you're connecting cities with roads, and each road has a cost.
You want to **connect all cities using roads** such that:

- **Total cost is minimized**
- There are **no loops (cycles)**

Minimum Spanning Tree (MST)

Properties:

- Contains **V – 1 edges** (if the graph has V vertices)
- No cycles
- Spans all vertices (connected)

Steps of Kruskal’s Algorithm

1. **Sort all edges** in the graph by their weight (ascending).
2. Create a **Disjoint Set** (Union-Find) to keep track of connected components.
3. Initialize MST as empty.
4. Traverse edges in sorted order:
    - If the current edge **does not form a cycle**, include it in the MST.
    - If it forms a cycle, **skip** it.
5. Repeat until MST has **V - 1 edges**.

Disjoint Set (Union-Find)

Kruskal’s algorithm uses **Disjoint Set Union (DSU)** to keep track of which nodes are connected.

It supports:

- **Find(u):** Return the representative (parent) of the set containing u
- **Union(u, v):** Merge the sets containing u and v
  Optimized with:
- **Path Compression**
- **Union by Rank**

Advantages

- Simple and intuitive
- Works well when the graph has **less number of edges (sparse graph)**
- Easy to implement with Union-Find

Limitations

- Slower on **dense graphs** compared to Prim's (because of sorting all edges)
- Requires the graph to be **connected** to form a full MST