Minimum Spanning Tree (MST) Deep Dive: Kruskal & Prim in DSA

Subtitle: Learn how to efficiently connect all nodes in a weighted graph with minimum total edge weight.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app

Introduction

A Minimum Spanning Tree (MST) of a weighted graph connects **all vertices with the minimum possible total edge weight**, without forming cycles. MSTs are fundamental in **network design, clustering, and graph optimization problems**.

Step 1: Core Concepts

- **MST** = subset of edges connecting all vertices **without cycles** and with **minimum total weight**
- Applicable only to **connected, undirected, weighted graphs**
- Two classic algorithms:
  1. **Kruskal's Algorithm** (edge-based, uses DSU)
  2. **Prim's Algorithm** (vertex-based, uses priority queue)

Step 2: Kruskal's Algorithm

Steps:

1. Sort all edges by **weight**
2. Initialize DSU for all vertices
3. Pick edges in order; add only if it **does not form a cycle**
4. Stop when **V-1 edges are added**

Implementation in JavaScript

```
class DSU {
 constructor(n){ this.parent=Array(n).fill(0).map((_,i)=>i); this.rank=Array(n).fill(0); }
 find(x){ if(this.parent[x]!==x) this.parent[x]=this.find(this.parent[x]); return this.parent[x]; }
 union(x,y){
  let px=this.find(x), py=this.find(y); if(px===py) return false;
  if(this.rank[px]<this.rank[py]) this.parent[px]=py;
  else if(this.rank[px]>this.rank[py]) this.parent[py]=px;
```

```
    else { this.parent[py]=px; this.rank[px]++; }
    return true;
  }
}

function kruskalMST(V, edges){
  edges.sort((a,b)=>a[2]-b[2]); // sort by weight
  let dsu = new DSU(V);
  let mst = [];
  for(let [u,v,w] of edges){
    if(dsu.union(u,v)) mst.push([u,v,w]);
  }
  return mst;
}

// Example usage
let edges = [[0,1,10],[0,2,6],[0,3,5],[1,3,15],[2,3,4]];
let mst = kruskalMST(4, edges);
console.log(mst); // [[2,3,4],[0,3,5],[0,1,10]]
```

Step 3: Prim's Algorithm

Steps:

1. Start with any vertex
2. Pick the **minimum weight edge** connecting MST to new vertex
3. Repeat until all vertices are included

Implementation in JavaScript

```
function primMST(graph){
  let V = graph.length;
  let key = Array(V).fill(Infinity);
```

```
  let parent = Array(V).fill(-1);
  let inMST = Array(V).fill(false);
  key[0] = 0;

  for(let count=0; count<V-1; count++){
    let u = -1;
    for(let i=0;i<V;i++) if(!inMST[i] && (u===-1||key[i]<key[u])) u=i;
    inMST[u] = true;

    for(let v=0;v<V;v++){
      if(graph[u][v] && !inMST[v] && graph[u][v]<key[v]){
        key[v] = graph[u][v];
        parent[v] = u;
      }
    }
  }

  let mst = [];
  for(let i=1;i<V;i++) mst.push([parent[i],i,graph[i][parent[i]]]);
  return mst;
}

// Example adjacency matrix
let graph = [
 [0,2,0,6,0],
 [2,0,3,8,5],
 [0,3,0,0,7],
```

```
 [6,8,0,0,9],

 [0,5,7,9,0]

];


console.log(primMST(graph));
```

Step 4: Applications

1. Designing **network wiring** with minimal cost
2. **Clustering** in data science
3. Optimizing **road or pipeline networks**
4. Competitive programming MST problems

Step 5: Mini Exercises

1. Implement Kruskal and Prim on sample graphs
2. Compare MST outputs for same graph
3. Solve MST problems with weighted edges given in edge list and adjacency matrix
4. Count number of distinct MSTs for a small graph

Step 6: Common Mistakes

- Forgetting to check cycles in Kruskal → invalid MST
- Using adjacency list incorrectly in Prim → inefficient
- Forgetting path compression in DSU → slower Kruskal
- Not handling disconnected graphs → MST cannot be formed

Step 7: Practice Plan for MST

Day 1:

- Implement Kruskal’s algorithm → 3–5 problems

Day 2:

- Implement Prim’s algorithm → 3–5 problems

Day 3:

- MST problem-solving + competitive programming → 5–7 problems

Key Takeaways

- MST connects all vertices **with minimum total weight**
- Kruskal → edge-based, DSU; Prim → vertex-based, priority queue

- Essential for **network design, clustering, and optimization problems**
- Practice strengthens **graph intuition, DSU usage, and edge selection logic**

Visit haas.dev for more MST guides, problem sets, and interview preparation resources.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app