**Q8: Minimum Spanning Tree for a Power Grid**: Implement **Kruskal’s algorithm** to find the **minimum spanning tree (MST)** for a power grid system connecting cities. Each city is a node, and each connection between cities has a cost.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Minimum Spanning Tree - Kruskal's Algorithm</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 20px;

background-color: #f5f5f5;

}

h1 {

text-align: center;

}

#graphInput {

margin: 20px auto;

max-width: 500px;

padding: 20px;

border: 1px solid #ddd;

border-radius: 8px;

background-color: #fff;

}

textarea {

width: 100%;

height: 100px;

margin-bottom: 10px;

font-family: monospace;

}

button {

padding: 10px 20px;

background-color: #007bff;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

#output {

margin-top: 20px;

padding: 20px;

border: 1px solid #ddd;

border-radius: 8px;

background-color: #fff;

}

pre {

white-space: pre-wrap;

word-wrap: break-word;

}

</style>

</head>

<body>

<h1>Minimum Spanning Tree - Kruskal's Algorithm</h1>

<div id="graphInput">

<h2>Enter Graph Edges</h2>

<p>Format: city1 city2 cost (one edge per line)</p>

<textarea id="edgesInput"></textarea>

<button onclick="findMST()">Find MST</button>

</div>

<div id="output">

<h2>Output</h2>

<pre id="result"></pre>

</div>

<script>

function findMST() {

const edgesInput = document.getElementById('edgesInput').value.trim();

const edges = edgesInput.split('\n').map(line => {

const [city1, city2, cost] = line.split(' ');

return { city1, city2, cost: parseInt(cost) };

});

edges.sort((a, b) => a.cost - b.cost); // Sort edges by cost

const parent = {};

const rank = {};

// Helper functions for union-find

function find(city) {

if (parent[city] !== city) {

parent[city] = find(parent[city]); // Path compression

}

return parent[city];

}

function union(city1, city2) {

const root1 = find(city1);

const root2 = find(city2);

if (root1 !== root2) {

if (rank[root1] > rank[root2]) {

parent[root2] = root1;

} else if (rank[root1] < rank[root2]) {

parent[root1] = root2;

} else {

parent[root2] = root1;

rank[root1]++;

}

}

}

// Initialize union-find structures

edges.forEach(edge => {

if (!parent[edge.city1]) {

parent[edge.city1] = edge.city1;

rank[edge.city1] = 0;

}

if (!parent[edge.city2]) {

parent[edge.city2] = edge.city2;

rank[edge.city2] = 0;

}

});

const mst = [];

let totalCost = 0;

// Kruskal's algorithm

for (const edge of edges) {

if (find(edge.city1) !== find(edge.city2)) {

mst.push(edge);

totalCost += edge.cost;

union(edge.city1, edge.city2);

}

}

// Output the MST

const resultElement = document.getElementById('result');

resultElement.textContent = `Minimum Spanning Tree:\n${mst.map(edge => `${edge.city1} - ${edge.city2} : ${edge.cost}`).join('\n')}\n\nTotal Cost: ${totalCost}`;

}

</script>

</body>

</html>