Day 5

Done	☑
Topic	Disjoint Set Data Structure Kruskal's Algorithm
Languages	Java
Difficulty	⭐⭐
Date Started	@March 5, 2023 11:40 AM
Date Completed	@March 6, 2023 1:45 AM
Related to Progress (Days)	Your Progress

What I Learned Today

Whats Is Disjoint Set Data Structure And It's Implementation.

Kruskal's Algorithm and It's Implementation

Key Concepts

Disjoint Set Data Structure

Disjoint-set data structure is to efficiently maintain a collection of disjoint (non-overlapping) sets of elements, subject to two operations:

• Union: Merge two sets together into a single set.

• Find: Determine which set a given element belongs to.

Quick Links

Tutorial

Documentation

Tutorial

Documentation

The data structure provides efficient implementations of these operations, which are crucial for solving a variety of graph-related problems, such as clustering, connected components, and minimum spanning trees. The disjoint-set data structure is typically implemented using an array or tree-based structure, with optimizations such as path compression and union by rank to achieve optimal time complexity.

Kruskal's Algorithm

Kruskal's algorithm is a greedy algorithm that finds a minimum spanning tree in a connected, weighted graph. The key points of Kruskal's algorithm are:

1. Sort the edges of the graph by weight in non-decreasing order.Initialize an empty set of edges, which will eventually form the minimum spanning tree.
2. Iterate through the sorted edges and for each edge, check if adding it to the set of edges creates a cycle. If it does not create a cycle, add the edge to the set.
3. Continue until all edges have been considered or until the set of edges forms a tree that spans all vertices of the graph.

Kruskal's algorithm is an efficient algorithm with time complexity O(E log E), where E is the number of edges in the graph. It is easy to implement and has been widely used in various applications such as network design and clustering.

Code Snippets

```
public class DisjointSet {
    static int n = 7;
    static int par[] = new int[n];
    static int rank[] = new int[n];

    public static void init() {
        for (int i = 0; i < n; i++) {
            par[i] = i;
```

```
        }
    }

    public static int find(int x) {
        if (x == par[x]) {
            return x;
        }
        return par[x] = find(par[x]);
    }

    public static void union(int a, int b) {
        int parA = find(a);
        int parB = find(b);

        if (rank[parA] == rank[parB]) {
            par[parB] = parA;
            rank[parA]++;
        } else if (rank[parA] < rank[parB]) {
            par[parA] = parB;
        } else {
            par[parB] = parA;
        }
    }

    public static void main(String[] args) {
        init();
        System.out.println(find(3));
        union(1, 3);
        System.out.println(find(3));
        union(2, 4);
        union(3, 6);
        union(1, 4);
        System.out.println(find(3));
        System.out.println(find(4));
        union(1, 5);

    }
}
```

```
import java.util.ArrayList;
import java.util.Collection;
import java.util.Collections;

public class KruskalAlgorithm {
    static class Edge implements Comparable<Edge> {
        int src;
        int dest;
        int wt;

        public Edge(int src, int dest, int wt) {
            this.src = src;
            this.dest = dest;
            this.wt = wt;
        }
```

```
        @Override
        public int compareTo(Edge e2) {
            return this.wt - e2.wt;
        }
    }

    static void createGraph(ArrayList<Edge> edges) {
        edges.add(new Edge(0, 1, 10));
        edges.add(new Edge(0, 2, 15));
        edges.add(new Edge(0, 3, 30));
        edges.add(new Edge(1, 3, 40));
        edges.add(new Edge(2, 3, 50));

    }

    static int n = 7;
    static int par[] = new int[n];
    static int rank[] = new int[n];

    public static void init() {
        for (int i = 0; i < n; i++) {
            par[i] = i;
        }
    }

    public static int find(int x) {
        if (x == par[x]) {
            return x;
        }
        return par[x] = find(par[x]);
    }

    public static void union(int a, int b) {
        int parA = find(a);
        int parB = find(b);

        if (rank[parA] == rank[parB]) {
            par[parB] = parA;
            rank[parA]++;
        } else if (rank[parA] < rank[parB]) {
            par[parA] = parB;
        } else {
            par[parB] = parA;
        }
    }

    public static void kruskalMST(ArrayList<Edge> edges, int V) {
        init();
        Collections.sort(edges);
        int mstCost = 0;
        int count = 0;
        for (int i = 0; count<V-1;i++){
            Edge e = edges.get(i);
            //(src, dest, wt)
            int parA = find(e.src);
            int parB = find(e.dest);
            if (parA!=parB){
                union(e.src,e.dest);
```

```
                mstCost += e.wt;
                count++;
            }
        }
        System.out.println(mstCost);
    }

    public static void main(String[] args) {
        int V=4;
        ArrayList<Edge> edges = new ArrayList<>();
        createGraph(edges);
        kruskalMST(edges, V);
    }
}
```

Challenges Experienced

Nothing

Resources Used

Youtube, GeekForGeeks, Javapoint, ChatGPT