Lab 10
Minimum Spanning Tree

Quang D. C.
dcquang@it.tdt.edu.vn

November 02, 2020

In the previous lab, we learned graph and how to traverse it. In this lab, we will learn a problem of graph is Minimum Spanning Tree. To find the Minimum Spanning Tree, we have two basic algorithms:

1. Prim's algorithm
2. Kruskal's algorithm

1. Minimum Spanning Tree

Tree **T** is a connected graph that has V vertices and V-1 edges, only one unique path between any two pair of vertices in T.

Given connected graph **G** with positive edge weights, find a min weight set of edges that connects all of the vertices.

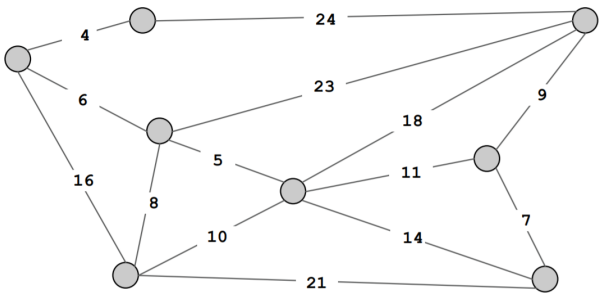


Figure 1: Graph

A spanning tree of a graph G is a subgraph T that is connected and acyclic.

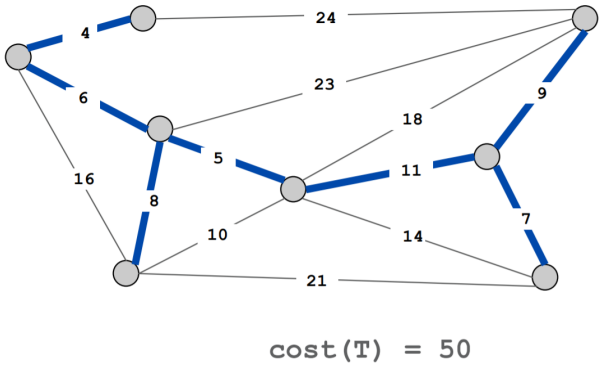


Figure 2: Minimum Spanning Tree

2. Prim's Algorithm

The way Prim's algorithm works is as follows :

- Initialize the minimum spanning tree with a initial vertex.
- Find all the edges that connect the tree to new vertices, find the minimum, and add it to the tree.
- Keep repeating step 2 until we get a minimum spanning tree (until all vertices are reached).

You can follow this instruction to step by step implement Prim algorithm:

1. Create a boolean array (visited[]) and two integer arrays (parent[], cost[])
2. Choose the initial vertex.
3. Find the vertex which has the minimum cost from the latest vertex and hasn't been visited (we can call this vertex is min_id).
4. Set visited[min_id] = true
5. Update parent[] from min_id to the vertices which haven't visited. Update cost[] if the cost from new vertex is smaller than from the previous.
6. Repeat step 3 until all vertices are visited.

3. Kruskal's Algorithm

To simplify Kruskal's implementation, we will you **Edge List** to store the graph. The steps for implementing Kruskal's algorithm are as follows:

1. Sort all the edges from low weight to high.
2. Take the edge with the lowest weight and add it to the spanning tree. If adding the edge created a cycle, then reject this edge.
3. Keep adding edges until we reach all vertices.

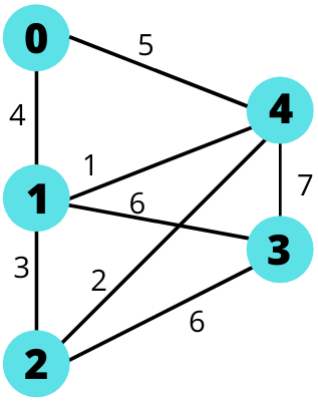
To check the edge created a cycle or not, you can **Union-Find Disjoint Sets**.

This is Union-Find code, you can init an instance and use isSameSet(int i, int j) method to check two vertices i, j create a cycle or not.

```
class UnionFind
    {
    private Vector<Integer> p, rank, setSize;

    public UnionFind(int N) {
        p = new Vector<Integer>(N);
        rank = new Vector<Integer>(N);
        setSize = new Vector<Integer>(N);
        for (int i = 0; i < N; i++) {
            p.add(i);
            rank.add(0);
            setSize.add(1);
        }
    }

    public int findSet(int i) {
        if (p.get(i) == i)
            return i;
        else {
            int ret = findSet(p.get(i));
            p.set(i, ret);
            return ret;
        }
    }

    public void unionSet(int i, int j) {
        if (!isSameSet(i, j)) {
            int x = findSet(i), y = findSet(j);
            if (rank.get(x) > rank.get(y)) {
                p.set(y, x);
                setSize.set(x, setSize.get(x) + setSize.get(y));
            }
            else{
                p.set(x, y);
                setSize.set(y, setSize.get(y) + setSize.get(x));
                if (rank.get(x) == rank.get(y))
                rank.set(y, rank.get(y) + 1);
            }
        }
    }

    public boolean isSameSet(int i, int j){
        return findSet(i) == findSet(j);
    }
}
```

4. Exercise

Given a graph:



Read this graph from a text file with Adjacency Matrix.

(a) Write the Kruskal function for the given graph. Print MST result on the screen.

(b) Write the Prim function for the given graph starting from vertex 0. Print MST result on the screen.

5. Reference

1. https://www.cs.princeton.edu/courses/archive/spr07/cos226/lectures/mst.pdf
2. https://www.journaldev.com/43746/prims-algorithm-minimum-spanning-tree-java

THE END