# Project: Minimum Spanning Tree

Group 25 – Keerthi Bala Sundram, Maitri Jatakia, Monish Muthuvijayan, Aastha Dixit

# Executive summary or abstract

This project provides the implementation of three algorithms – Prim1, Prim2, and Kruskal’s algorithm for finding the Minimum Spanning Trees.

# Introduction

A minimum spanning tree is a spanning tree which is always connected and an undirected graph that connects every one of the vertices collectively with the minimal total weighting for its edges.

A single graph that is taken for finding MST can possess many different spanning trees. The edges are assigned with a weight, which represents how far/distant they are or how heavy they are. The algorithm basically finds the tree which comprises of the edges that possess minimum weight and that connects all the vertices.

More generally, any undirected graph contains a minimum amount of spanning forest, which can easily be a union of minimum spanning trees pertaining to its components.

# Methodology

We use 3 methodologies: Prim’s algorithm with Priority Queue of Edges, Prim’s algorithm with Priority Queue of Vertices, and Kruskal’s algorithm. Prims algorithm is a type of greedy algorithm that helps in finding an MST for an undirected graph that has weights. This algorithm finds a set of edges that would form a minimum spanning tree, which includes every vertex, and where the total weight is the sum of total edges in the tree is minimized. Prim basically uses priority queue to store the edges/vertices. Prim1 uses the traditional priority queue, which gives minimum edge from all the edges observed. Prim2 uses indexed priority queue, which stores vertices based on its distance from the parent. Each time we find a way to reach any vertex using a smaller edge we update the distance of that node (decrease key) and its parent. The Kruskals algorithm finds the minimum spanning tree by using a concept called disjoint sets. The algorithm finds a subset of the edges that forms a tree that includes all the vertex where the total weight of all the edges in the tree is minimized.

# Development platform/experimental setup

Eclipse IDE running on 8GB RAM PC with i5 processor.

# Test results

Sample Input

5 7

1 5 8

1 4 7

1 3 6

4 3 3

3 5 6

5 3 2

5 2 1

Sample Output

12

(5,2)

(5,3)

(4,3)

(1,3)

Sample Input

7 12

1 2 4

1 3 8

2 3 9

2 4 8

3 4 2

3 6 1

2 5 10

4 5 7

5 6 5

4 6 9

5 7 6

6 7 2

Sample Output

(1,2)

(1,3)

(3,6)

(3,4)

(6,7)

(6,5)

Min Weight 22

Sample Input

6 9

1 2 3

1 4 1

2 4 3

2 3 1

3 4 1

3 5 5

3 6 4

4 5 6

5 6 2

Sample Output

(1,4)

(4,3)

(3,2)

(3,6)

(6,5)

Min Weight 9

# Discussion of results

Kruskal’s algorithm: It runs in O(E log V) . It requires O(V) time to perform each iteration to connect vertex to the spanning tree including the find operation and the union.

Prims : Each edge is processed exactly once and may cause a decrease in the weight of some non-tree vertex ( not seen Vertex). Decreasing the weight takes O(log V) time. So the overall running time of the algorithm is

Running Time: O(E log V)

# References

Class Notes

Youtube.com videos