**CSE 5331 – DESIGN AND ANALYIS OF ALGORITHM**

**FINAL PROJECT**

**TOPIC – MINIMUM SPANNING TREE**

**Instructor:**

**Mr. Md Hasanuzzaman Noor**

**Project by:**

**Vyash Natarajan (1002060598)**

**Shibani Prasath (1002060600)**

**ABSTRACT**

Minimum spanning tree (MST) is used to solve graph problem. By using MST, lowest cost of graph can be determined. We have Prim’s and Kruskals Algorithm are commonly used to find minimum spanning tree. This project discussed about comparison of Kruskal and Prim algorithm and analysis of it. Both algorithms was tested on three different samples of graph. Then, each result would be compared. By making a comparison, it will be easier to understand how each algorithm works.

**INTRODUCTION**

The purpose of this project is to design and analyse the minimum spanning tree algorithm that we have learned in class. The algorithm include Prims and Kruskals Algorithm. Even though we have many Algorithms and implementation techniques, we check the time complexity and performance of the Algorithm to get the best result and output. The total time for Prim’s Algorithm is O(m lg n) and the total time for Kruskals Algorithm is O(nm).

In this project we are going to implement and execute both the Prim’s and Kruskal’s Algorithm with three different Dataset which are small, medium and large. This will help us to compare both the algorithm.

The small Dataset contains;

1. Six Data with 2 vertices each and its weight.

The medium Dataset contains;

1. 812 Data with 2 vertices representing cities and its distance.

The large Dataset contains;

1. 50176 Data with 2 vertices representing country code and its distance.

**PRIMS ALGORITHM**

**Pseudocode**

Import necessary packages

readgraph(file,d)

for lines in file path:

line = [x.strip() for x in lines.split(split(f’{d}’)] #to split the data from the input

weight = round(float(line[2])) #we used decimal values

if e1 in glb.WGRAPH[e1][e2] = weight

glb.WGRAPH[e1][e2] = weight

else

glb.WGRAPH[e1] = {}

glb.WGRAPH[e1][e2] = weight

return glb.WGRAPH #return graph

add visited(Vertex):

if(Vertex in glb.Vr)

return false

else

glb.Vr.append(Vertex)

return true

extract minimum

prims()

cdist=0 #cummulative distsance

length = len(list(glb.WGRAPH.Keys()))

add visited

for 0 to length-1

get min

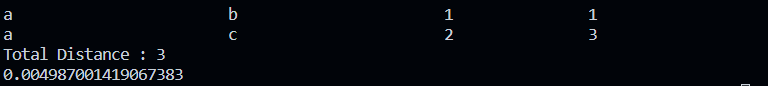
add v to MST

append v,v2,edist,cdist

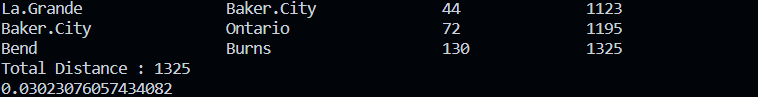
return MST

**Runtime (in sec) :**

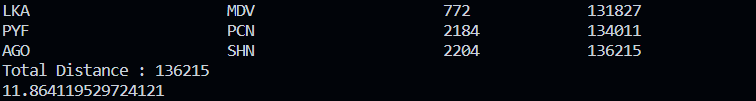
**For small dataset:**

****

**For medium dataset:**

****

**For large dataset:**

****

**Kruskal’s Algorithm**

Import necessary packages

readgraph(file,d)

for lines in file path:

line = [x.strip() for x in lines.split(split(f’{d}’)] #to split the data from the input

weight = round(float(line[2])) #we used decimal values

heappush edges,weight,v1,v2

return glb.edges

make set(v)

glb.parent[v]=v

glb.rank[v]=0

find v

return parent

union(v1,v2)

root1 = find v1

root2 = find v2

if root1!=root2

if rank[root1] > rank[root2]

parent[root2]=root1

else

parent[root1]=root2

rank[root2]=1

kruskals()

cdist=0

for v in glb.vertices

make set(v)

while glb.edges

e,v1,v2= heappop edges

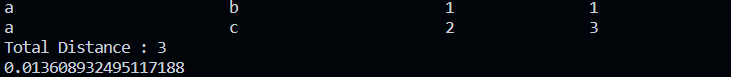
union v1 v2 if find v1!=findv2

append v1,v2,e,cdist

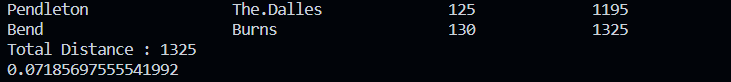
return MST

**Runtime (in sec) :**

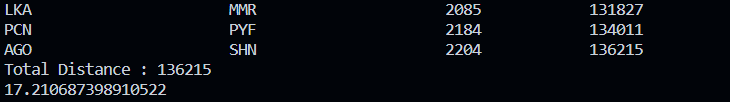
**For small data set;**

****

**For medium Data set;**

****

**For Large Data Set;**

****

**Result:**

The below graph shows the prims and kruskals runtime performance for small and medium dataset.

The below graph shows the prims and kruskals runtime performance for a large dataset.

**What we have learned?**

We have learned to implement and execute the minimum spanning tree algorithm using prims and kruskals. We have analysed the runtime and performance of this algorithm which will be useful for us in upcoming years as a developer. We came to know the realtime application of this algorithm which made us to research more about this project.

**What difficulties we encountered?**

We have encountered several issues while choosing the proper implementation part. The reference we took didn’t work out properly. First we tried to implement in java then we came to Python. Finding similar dataset and organizing it to a specific type took our time.

**Contribution:**

We almost did the project together. Particularly Vyash Natarajan (1002060598) did the kruskals implementation and the research behind it. Shibani Prasath did the Prims implementation and its research behind it. As the dataset is same and the approach is nearly same, we discussed what and how the approach is going to be. The project report work is shared by both. The testing, debugging is all done by both by looking at each other algorithms.