# Design Document

# Problem

**1. Each process reads neighbour information and cost from a file**

1. **Each process builds a network graph from all nodes using this neighbor information**

**Given this information answer the following questions:**

1. **Find the shortest path from any to any node given.**
2. **Find the minimum spanning tree.**

Input File Format:

Input to the file should be as follow

Src\_node,(dest1,weght1), (dest2, Weight2), (dest3, Weight3)...... (destn, Weightn)

Where,

Src\_node: this node is the starting node.

Destn: end node from src\_node.

Weightn: this is the weight/Cost from Src\_node to Destn

Example:

0, (1,4) ,(7,8)

1, (0,4),(7,11),(2,8)

7,(0,8),(6,1)

2,(1,8),(8,2),(5,4),(3,7)

8,(6,6),(7,7),(2,2)

6,(7,1),(5,2),(8,6)

3,(2,7),(4,9),(5,14)

5,(4,10),(6,2)(3,14),(4,2)

4,(3,9),(5,10)

Graph created for above input file as shown below



Shortest Path algorithm:

Finding the shortest path between a nodes

To find shortest path maintail all nodes with short distance from source after each time when we visited to node .

Make sure to not visite if the node is already visited .

After finding shortest path for above graph the table as shown below

|  |  |  |  |
| --- | --- | --- | --- |
| Source Node | Previous Node | Destination Node | Shortest Path |
| 0 | 0 | 1 | 4 |
| 0 | 0 | 7 | 8 |
| 0 | 2 | 2 | 12 |
| 0 | 7 | 8 | 14 |
| 0 | 7 | 6 | 9 |
| 0 | 2 | 3 | 19 |
| 0 | 6 | 5 | 11 |
| 0 | 5 | 4 | 21 |

Algorithm:

1. Initialise weight of the source node to zero and all other nodes to infinity.

Map[src]=0;

map[src +1........dst]=infinity;

visited[src ....dst]=false;

1. Find the weight of nodes from source and keep these weights in the map for feature use.
2. Repeat step 2 for all nodes and save smallest weight in the map till the last node.

For (i=src to all other nodes)

Visited[i]=true

For(j=0 to no of adjacent nodes)

If(visited[j]!=false)

Weight=nodes[i].adj[j]+map[i];

If(weight<map[j])

Map[j]=weight;

# Minimum Spanning tree:

Maintain two sets of vertices one is to maintain already included in spanning tree and other one is not included.

Algorithm:

1. Create a set that keeps track of vertices already included in MST.

Included [v]=false;

**2)** Assign a key value to all vertices in the input graph. Initialize all key values as INFINITE. Assign key value as 0 for the first vertex so that it is picked first.

Key[0 to v]=infinity;

Key[0]=0;

Mst[0]=0;

Included[v]=true;

**3)** While doesn’t include all vertices, Pick a vertex which is not there in Setand has minimum key value. Include uto Set. Update key value of all adjacent vertices of u. To update the key values, iterate through all adjacent vertices. For every adjacent vertex v, if weight of edge u-v is less than the previous key value of v, update the key value as weight of u-v.

For (i=src to all other nodes)

U=Find minKey(key, Included);

For(j=0 to no of adjacent nodes)

If(included[j]==false&&graph[u,j]<key[j])

{

Key [j] =nodes[i].adj[j];

Mst[j]=u;

}

Minimum spanning tree of above graph is as shown below.

