Design Document:

Section-1.Problem statement:

## Design the below graph algorithm.

## Graph Algorithms

You are given a set of interacting processes that exchange neighbor adjacency data. The process works like this:

1. Each process reads neighbor information and cost from a file
2. 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.

Also, reconfigure the network connectivity graph on a node when a node (process) dies or an edge is lost

#### Issues to be handled:

1. Nodes (process) can startup and die at any point of time
2. No central manager for the set of nodes (process)

#### Some aspects of evaluation:

1. Write-up on approach, design, unit testing, debugging hooks
2. Choice of data structures, threads, processes, IPC
3. Simplicity of implementation
4. Efficiency of implementation
5. Bugs/corner cases handling

#### Language to be used:

1. C/c++.

**Section -2. Evolution of Problem:**

**Platform**:

Linux -Ubuntu.

**Technologies will be used:**

c,c++,socket programming

multiprocessing.

**Modules**:

generate.cpp

main.cpp

file – datafile

**Design Steps:**

we have with following thing with us as in the problem mentioned:

->Each process reads neighbor information and cost from a file

->Each process builds a network graph from all nodes using this neighbor information

→ This graph information I am storing it into a data file.

->and this file I will use to create a graph.

→ There are many ways of doing this like client server model, multiprocessing or parallel programming using **mvapich and mpich library.**

**→ also we can use openmpi and pthread to create a light weight process.**

→ so will be using the simplest way of doing it using process and thread algorithms.

→ main process will take the data from the file about neighbor information and start building graph using it.

→ file will be containing adjacency matrix data.

→ main process that take the graph data as a input and create a adjacency matrix graph using below algorithm.

**→ Algorithm will take care about two things,**

**find the shortest path between nodes.**

**find the minimum span tree.**

**Algorithm:**

1)Create a setptSet (shortest path tree set) that keeps track of vertices included in shortest path tree, i.e., whose minimum distance from source is calculated and finalized. Initially, this set is empty.

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

1. While sptSet oesn’t include all vertices  
   ….a)Pick a vertex u which is not there insptSetand has minimum distance value.  
   ….b)include u tosptSet.  
   ….c)Update distance value of all adjacent vertices of u. To update the distance values, iterate through all adjacent vertices. For every adjacent vertex v, if sum of distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

**Section3:Debugging:**

**→ As we are using platform as a Linux. We can use the below debugging skills,**

1. **GDB:**

**gnu debugger**

→ in this there the multiple commands for the debugging using gdb available here please go through it:[**http://www.tutorialspoint.com/gnu\_debugger/**](http://www.tutorialspoint.com/gnu_debugger/)

→ put the breakpoints when some error occurs .

→ you can step inside the function and check the return value also using gdb.

→ gdb helpful in the case of thread also. We can **attach a thread** using thread id here.

→ also in case of **core dump the core dump file will be created .**

→ so we can debug easily.

**Section 4: Test cases:**

**Pass test cases:**

1. Try the algorithm using the different inputs:

→ I am going to use 2d array and change the value and test if it always returns the shortest distance or not.

1. Try to change the vertex information increase and decrease it and test.
2. Close the running process it should die and in memory graph should be destroyed.
3. Try with multiple files if scanning happing properly or not.
4. At a time if two connection established then check it should be implemented using **semaphores** or **mutex**.

Fail test cases:

1. Try to pass string as a input it should fail because node information we are using in the format of int.