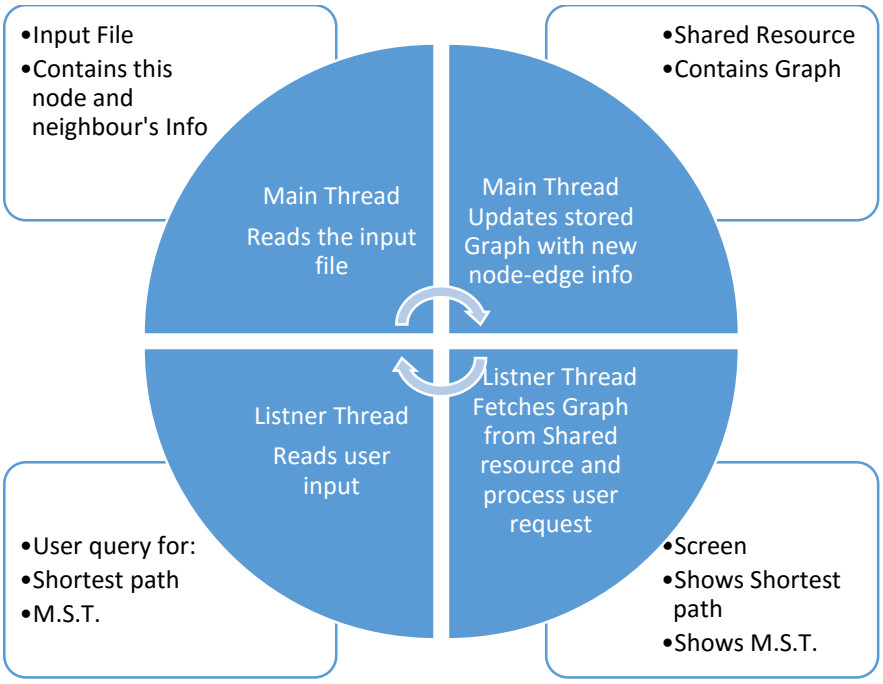
Design Document for Graph Algorithms

Revision – 0.1 (Initial Draft)

Author - Aditya Shrama

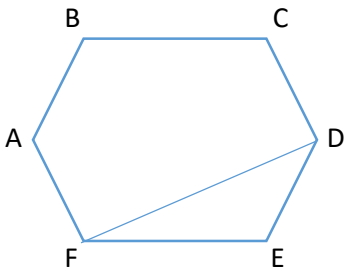
Design:

Below figure shows typical block diagram of the process.



Block Diagram

Above figure shows single instance of the process. Multiple instances of same process will run, each instance representing a node. So to create the sample graph shown below, total 6 instances (1 for each of the nodes A, B, C, D, E & F) will run.



Sample Graph

Process Explanation:

Each node information ie. Vertex name and adjacent (neighbour nodes) will be stored in a text file. This will be the input to process as a command line argument.

Upon execution of the process, it will read node info from the input file and update the stored graph in a shared memory location. This shared memory will be accessible by all the instances of this process.

After updating the graph in common shared memory, this process will spawn a listener thread which will wait for the user input. User will be presented with various choices like – Show the shortest path between any 2 arbitrary nodes or Show the minimum spanning tree.

As per the user input, it will fetch the latest info from shared memory segment, do the processing for finding out shortest path or minimum spanning tree and will output same on the screen.

Whenever any instance of this process dies, it's node info and connecting edges info will be deleted from the shared memory. This will be done inside the signal handler of the application. This all will be taken care by the same process, ie. no other central node management process is required. Having an idea of implementing graph in which nodes will auto expire and delete itself if it fails to query the process which created it.

Input:

Input to the program will be a text file containing this node and connecting nodes (edges) information.

Below is the sample input file.

Node = A

Connecting Nodes = B, F

Output:

Output of program will be either shortest path between any 2 arbitrary nodes or the minimum spanning tree as per the user query.

Note: This is initial draft. Actual product may differ as I proceed with the implementation.