Design Document for Graph Algorithms

Author: Sandeep Kumar Verma

=--=-=-=-=-=-=-=-=-=-

CONTENTS:

=--=-=-=-=-=-=-==-=-=-

1.0 PROBLEM STATEMENT:

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:

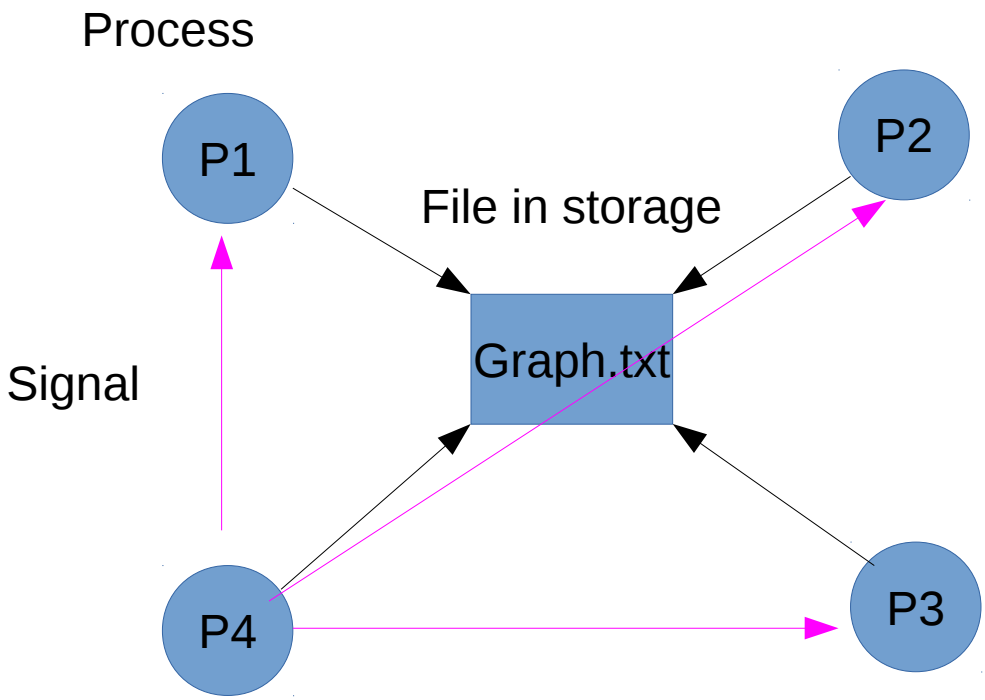
1. C/C++

2.0 OVERVIEW:

Each process will act as a node and it will be having a graph data structure in RAM as well as in a file (in persistance storage).
Same process of multiple instances will be accessing same file in the storage which will be having graph data of each running process.
Once process runs, it reads graphs of all running process from a file and updates it's datastructure with exact depth from itself. It updates the file with its own graph data and send signal to all process to update their own structures.
In the same way before termination of the process, it will update the file by removing its own graph and send signal to all the process to update their own.

Any running process can take input from terminal (stdin) as a query and respond for it.

3.0 BLOCK DIAGRAM:



As clearly mentioned in the block diagram, all process will be accessing the same file.
When process comes up or before termination it sends signal to all the process to read the file and update their data structure.

4.0 PREREQUISITES

4.1 PC running linux on it
4.2 Make, GCC, GIT utilitie, GIT repository

5.0 USEFUL COMMAND

5.1 To run a process as a node, we have to tell which node it is and which node it is connected with. It can be done while executing the process via command line.
Ex:

$ ./node 23 12 20 13

* here “node” is exwcutable file name, 23 is a node identity, 12 20 13 are nodes with 23 is attached.

5.2 Query can be given to any running process, through stdin
Ex:

-> **s_path 12 13**

***** It will give a shortest path between node 12 and node 13

-> **mini_spi_tree**

***** It will show result of minimum spinning tree for graph

6.0 COMPILATION

6.1 makefile has been written for the project, just need to give command “make” to compile and generate the executable file.

$ make

Note: Actual implementation may vary and so design document.