University of Texas at Dallas—Department of Computer Science
CS 6380 Distributed Computing—Spring 2017
Project 1 Description

This is a group project: Form groups of three members.

You will develop a simple simulator that simulates a synchronous distributed system using multi threading. There are n+1 threads in the system: Each of the n processes will be simulated by one thread and there is one master thread. The master thread will "inform" all threads when one round starts. Thus, each thread simulating one process, before it can begin round x, must wait for the master thread for a "go ahead" signal for round x. Clearly, the master thread can give the signal to start round x to the threads only if the master thread is sure that all the n threads (simulating n processes) have completed their previous round (round x-1).

Your simulation will simulate BellmanFord algorithm for shortest paths in synchronous general networks. The code (algorithm) executed by all processes must be the same.

The input for this problem consists of (1) n (the number of processes of the distributed system which is equal to the number of threads to be created), (2) one array id[n] of size n; the i^{th} element of this array gives the unique id of the i^{th} process or i^{th} thread, (3) one n by n symmetric matrix Weight[n,n] that gives the connectivity and edge weight information, and (4) id of the leader from where the shortest paths tree is to be built. The master thread reads these inputs and then spawns n threads. Process i knows only about its connectivity. [Thus, process i knows only the information on the i^{th} row of the Weight matrix.] If processes x and y are connected by an edge between them then Weight[w,x] (which is equal to Weight[x,w]) is the (non-negative) weight of the link (w,x). If processes y and z are not connected by an edge (they are not neighbors), then Weight(y,z)=Weight(z,y)=infinity. Thus, all links are bidirectional.

No process knows n. Each process knows the number of neighbors it has, the ids of its neighbors and the edge weights for the edges incident on them. Termination is to be done through the convergecast procedure.

Output the MST by treating the tree as a graph and outputting the adjacency list.

Upload one tar file containing your source code, a README file that tells us how to compile and run, one sample input file and the result of running your program on your sample input file.

Due date: February 24, 11:55 pm.