Assignment 2

Distributed Systems

February 4, 2020

1 General Instructions

- **Deadline is 15th February 2020, 11:55 PM**
- The assignment has to be done in Java only.
- **Strict actions would be taken against anyone found involved in any kind of plagiarism either from the internet or from other students.**
- **Submission instructions given at the end of the assignment must be strictly followed. Since the evaluation will be automated, any deviation might result in loss of marks.**

2 Problem

For this assignment, you will be using RMI(Remote Method Invocation) in Java to implement a simple single server architecture with support for multiple clients. The details are as follows:

- The server maintains a list of graphs each associated with a distinct identifier.
- Clients can request to add a new graph, update an existing graph and query for the total weight of the minimum weight spanning tree of a given graph.
- Clients can request to add a new graph using '`add_graph ⟨graph_identifier⟩ n`'. This command will add a new graph on the server with the identifier graph_identifier and n number of nodes. The graph_identifier is a string with a maximum length of 10 and it won't already exist. n will be in the range: $1 <= n <= 100,000$.
- Clients can request to add a new edge in a graph using '`add_edge ⟨graph_identifier⟩ ⟨u⟩ ⟨v⟩ ⟨w⟩`'. This will add an undirected edge between the nodes u and v with weight w. u and v are the node numbers of the endpoints of the edge such that $1 <= u, v <= n$ and $0 <= w <= 10,000$. n is the number of nodes in the specified graph. A graph with identifier graph_identifier will already exist. There can be multiple edges and self-loops added to the graph.

- Clients can request for the total weight of the minimum weight spanning tree in a graph from the server using '`get_mst ⟨graph_identifier⟩`'. The client will print the solution the server returns. In case the graph does not have a spanning tree, -1 should be printed. A graph with identifier graph_identifier will already exist.

- All values should fit in 32-bit signed integers.

- The server should be able to handle multiple clients simultaneously and should also work with clients on other machines.

- You are free to use any algorithm for MST.

- A tutorial for RMI can be found at https://www.tutorialspoint.com/java_rmi/index.htm or https://www.geeksforgeeks.org/how-to-run-java-rmi-application/.

3 Submission Instructions

You need to submit a single file - **⟨RollNumber⟩.zip** containing a directory with the same name as your roll number that holds the following files:

- 2 Java source files - **Client.java** and **Server.java**. Your code will be run using:
  - The files would be compiled with '`javac *.java`'.
  - The server is run using with '`java Server ⟨server_port⟩`'.
  - The clients are run using '`java Client ⟨server_ip⟩ ⟨server_port⟩`'.

- A report for the problem - **README.md**. The report should contain details of the architecture, algorithm implementation, results and observations.