ALGORITHM DESIGN AND ANALYSIS

LAB 4, 2015

Instructions.

(1) Please submit a single file `WGraph.java` before the end of the lab session
(2) In each file, you MUST use the correct indentation, and comments.
(3) You may work in a group of no more than 3 people
(4) The total mark of your lab is 30. It is worth 5% of your final grade

Description. The file `WGraph.java` implements a simple weighted graph data structure where the nodes are integers $0, 1, 2, 3, n-1$. The main data structure is a two-dimensional array `data[0..n-1][0..n-1]` which represents the adjacency matrix of the graph. The value of `data[i][j]` indicates the length of the edge from node i to node j if there is an edge, and is -1 otherwise. The file implements the following methods:

- **load(String filename)**: Load a file representing a weighted graph to `data` to create an instance of the weighted graph object. The file would use `#` to indicate that no edge exists between two nodes.
- **print()**: Print out the adjacency matrix of the graph
- **isUndirected()**: Return true if the weighted graph is undirected. Return false otherwise.
- **generateRandomGraph(int num, double prob)**: Generate a graph with `num` nodes and density `prob`. All edges are assumed to have a positive weight
- **generateRandomGraphNegative(int num, double prob)**: Generate a graph with `num` nodes and density `prob`. All edges are assumed to have a positive or negative weight
- **generateUndirectedGraph(int num, double prob)**: Generate an undirected graph with `num` nodes and density `prob`. All edges are assumed to have a positive weight

Your tasks. Implement the following methods

(1) **mst()**: Compute a minimal spanning tree of the graph and print out all edges in the tree.
(2) **dijkstra(int i)**: Given a node i in the range $[1..n-1]$, run dijkstra's algorithm to compute the distance from i to all other nodes. Then print out the distances from i to all other nodes.
(3) **bellman(int i)**: Given a node i in the range $[1..n-1]$, run Bellman-Ford algorithm to compute the distance from i to all other nodes. Then print out the distance from i to all other nodes.
(4) **floyd()**: Run Floyd-Warshall algorithm to compute distance from any node to any node. Then print out the distances.