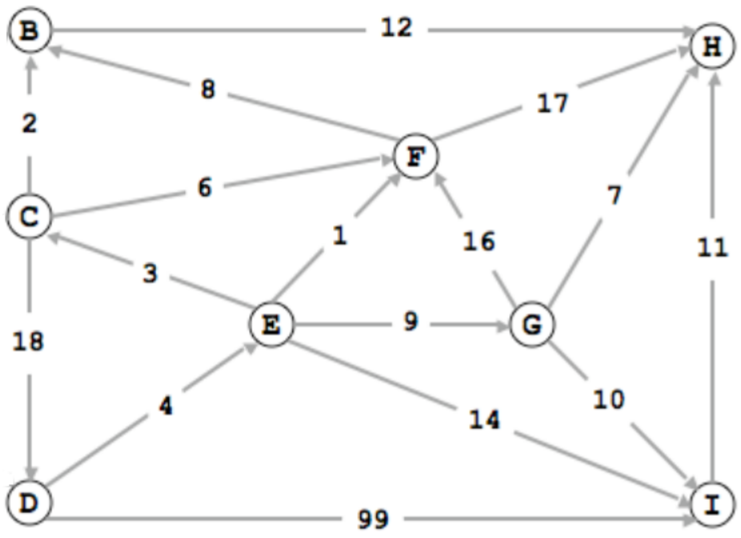
/*
Jason Wilson
Thursday, April 23rd 2020
CMPT435L 111 20S
*/

Assignment 9

Please read turn-in checklist at the end of this document before you start doing exercises.

Section 1: Pen-and-paper Exercises

1. Let vertex **C** be the source. Run Dijkstra's shortest path algorithm on the graph below.



a) Give the order in which the vertices are added to the shortest path tree (7 points).

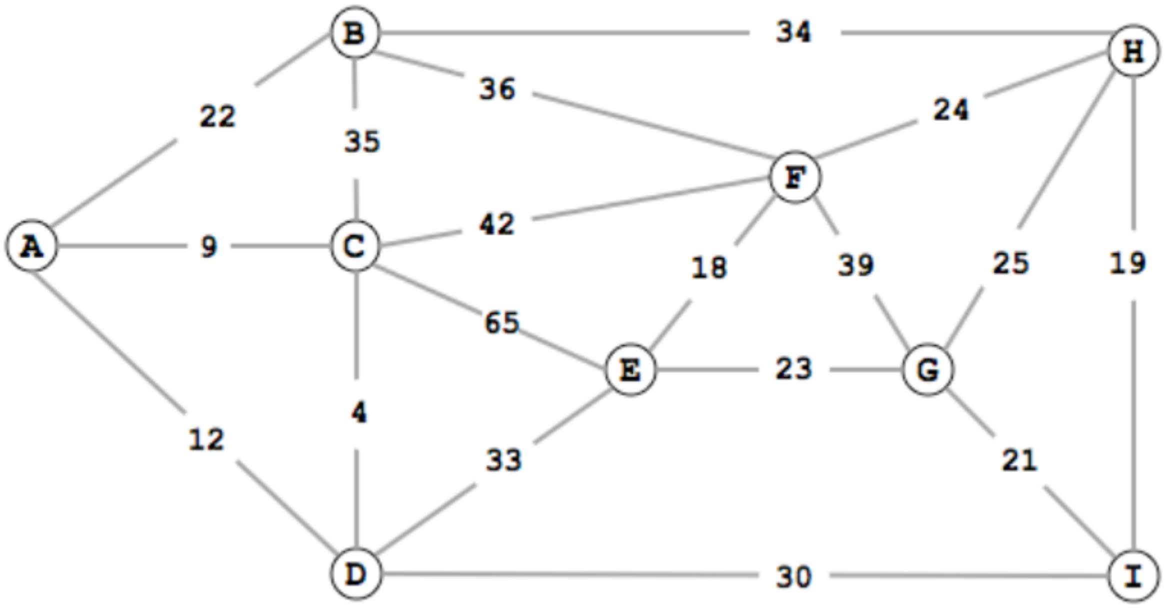
C, B, F, H, D, E, G, I

b) Give the distance of the shortest path from C to each vertex v (in other words, the final value of d[v] at the end of Dijkstra's shortest path algorithm, 7 points).

v		d[v]
B	-	2
C	-	0
D	-	18

E	-	22
F	-	6
G	-	31
H	-	14
I	-	36
Final	-	129

2. Given an undirected weighted graph, list the edges in the MST in the order in which they are discovered by Prim's algorithm, starting the search at vertex A. Since all edge weights are distinct, identify each edge by its weight (instead of its endpoints, 9 points).



A	C	D	B	I	H	G	E	F
1	9	4	22	30	19	21	23	18

3. Dijkstra's algorithm finds the shortest path from s to every other vertex in the graph. What if we just want to find the shortest path from s to some particular vertex (not all vertices) in the graph. What is your algorithm? (5 points)

To find the shortest path from one specific vertex to another, we can begin iterating through the graph following paths until the target is found, then back track on that path and explore new paths to the target and comparing their final values of d[v]. If taking another path takes longer at any point before reaching the target, then that path can be discarded and backtracked until the current search is less than the current established path. For example, starting from H with a target of C it can start iterating by traversing from H -> B at 34 then B -> C at 69. Here is our first path and we can be begin backtracking on it to find another possibly shorter path. Backtrack to B at 34 then to A at 56. 56 is less than 69 so we continue. A -> C (target) puts us at 65 and that is less than 69 so that is our new shorter path. H -> B -> A -> C. Then we can back track again and continue searching.

The algorithm will not search the entire graph because at every step it is checking to see if the current path is longer than the path already found. So the initial path value must be set to infinity. All of the edges that have been traversed will also be marked as visited once used, and will be backtracked on only when all edges on a vertex have been visited. Vertices being placed in a stack will allow for easy backtracking by popping out and continue on new paths with unvisited edges.

Section 2: Java Implementation

4. Implement the Dijkstra's Shortest Path Algorithm in Java.
   Note:
   Find a file called Dijkstra.java in assignment 9 folder.
   Complete the method of dijkstra().
   Test your method in the main method in WeightedGraph.java provided following the comments.

TURN-IN CHECKLIST:

1. **Answers to Section 1 (.doc/.txt/.pdf), and to Section 2 (all your source Code (.java files)). Remember to include your name, the date, and the course number in comments near the beginning of your code/report.**
2. **Create a folder and name it 'FirstName_LastName_assignment_9'. In the newly created folder copy and paste your files (.doc/.txt/.java files). Then compress the folder, and push it to iLearn.**