EX4: BFS, DFS, & Dijkstra's

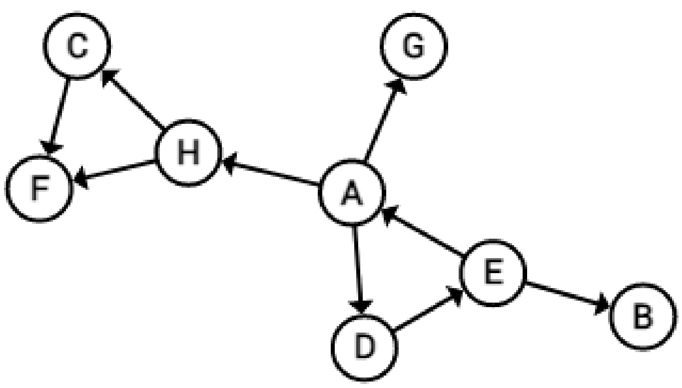
Due date: Friday May 13, 2022 at 11:59 pm
Latest turn-in date: Monday May 16, 2022 at 11:59 pm

Instructions:

High-level collaboration is allowed, but exercises are to be completed and submitted individually.
Submit your responses **digitally** through text box and image submission in the "EX4: BFS, DFS, & Dijkstra's" assignment on Gradescope here:
https://www.gradescope.com/courses/379339/assignments/1938570/.
Make sure to log in to your Gradescope account using your UW email to access our course.

1. Graph Question

Please look at the following graph and answer questions. For part (c) and (d), if a node has more than one neighbor, visit the neighbors in alphabetical order. For example, when you iterate over the neighbors {A, F, D, Z} of some node, it visits A, D, F, Z in order.



(a) Please draw an adjacency list representation of the graph above.

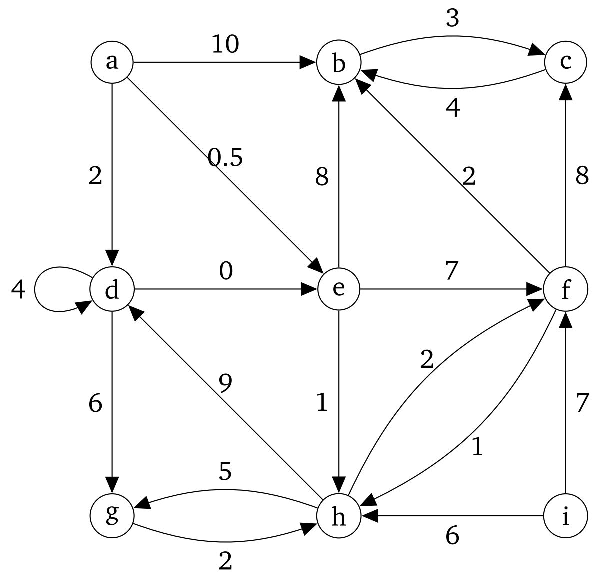
(b) Please draw an adjacency matrix representation of the graph above.

(c) Starting at node A, in which order will the nodes in the graph be visited by depth-first search? (separate your letters with commas and spaces, i.e. X, Y, Z)

(d) Starting at node A, in which order will the nodes in the graph be visited by breadth-first search? (separate your letters with commas and spaces, i.e. X, Y, Z)

2. Dijkstra's Algorithm

Consider the following graph:



Run Dijkstra's algorithm on this graph to compute the shortest path tree for this graph starting on node a. For each vertex in the graph (a - i), you should give the final result and show your work for data that gets stored by filling the following table.

Make sure to **show the intermediate values** you had during the algorithm. You'll list out all the proposed distances / predecessor edges as a list in chronological order as they occur during the algorithm, where the last item in the list is the actual best distance / predecessor edge.

Vertex	Distance	Predecessor	Processed
a			
b			
c			
d			
e			
f			
g			
h			
i			

3. Shortest Paths: T/F

Suppose we run Dijkstra's algorithm from a start node s in a **weighted undirected graph**. The distance to each reachable node t is given by `distTo.get(t)`.

Let u and v be two other nodes in the graph that are not the start node. For each of the following statements, state whether it is true or false.

If the statement is true, no further explanation is necessary. If the statement is false, draw a counterexample graph with 2-5 nodes (with real numbers for edge weights) that demonstrates why the statement is false. Write 1-2 sentences explaining your counterexample.

(a) There can only exist a single shortest path from u to v.

(b) A shortest path from u to v must include one or more edges in the shortest paths tree from s.

(c) If the shortest paths tree reaches u and v, then u and v are connected.

(d) The distance of a shortest path from u to v cannot be greater than `distTo.get(u) + distTo.get(v)`.

(e) In a graph that contains some negatively-weighted edges, but does not contain cycles with only negatively-weighted edges, Dijkstra's algorithm will always compute the correct shortest path from the start node s to any node in the graph.