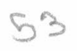
CS 372 HW 5 Zachary Neeley

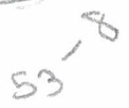
Question 1

a)

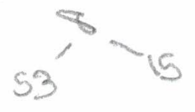
Step 1: Insert(53)



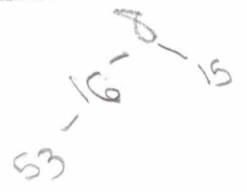
Step 2: Insert(8)



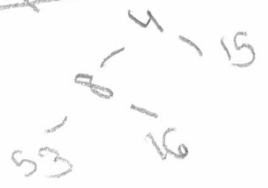
Step 3: Insert(15)



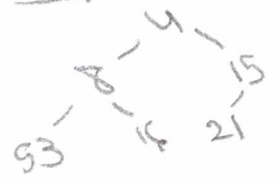
Step 4: Insert(16)



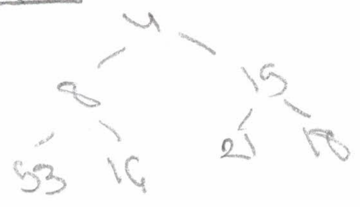
Step 5: Insert(4)



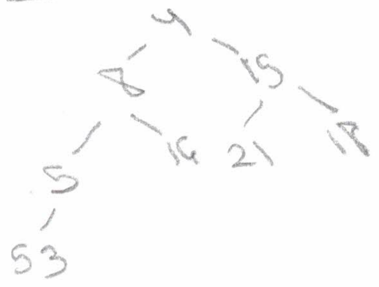
Step 6: Insert(21)



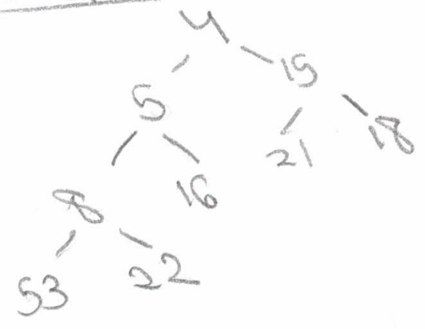
Step 7: Insert(18)



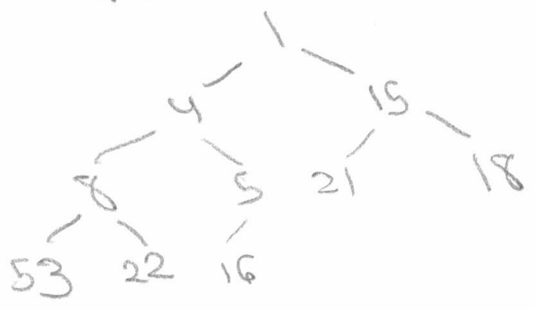
Step 8: Insert(5)



Step 9: Insert(22)



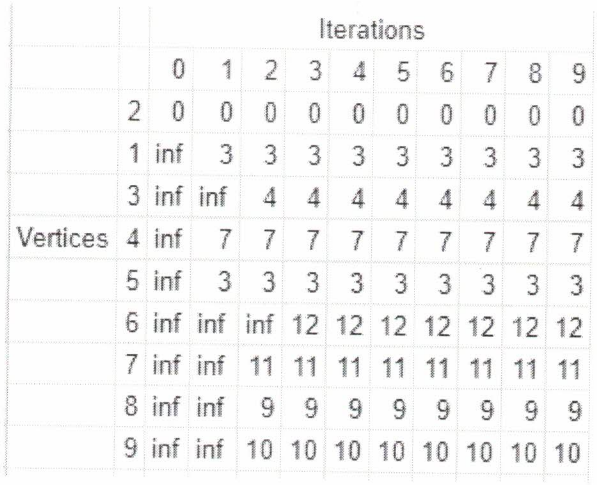
Step 10: Insert(1)



B) Array[1,4,15,8,5,21,18,53,22,16]

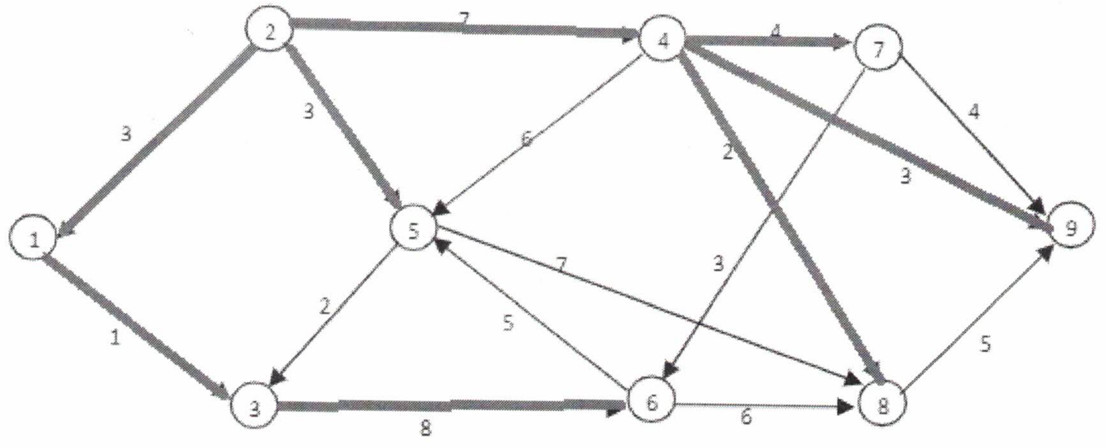
2. Suppose Dijkstra's algorithm is run on the following graph, starting at node 2. Note: We do NOT start at node 1, we start at node 2.

a)



		Iterations									
		0	1	2	3	4	5	6	7	8	9
	2	0	0	0	0	0	0	0	0	0	0
	1	inf	3	3	3	3	3	3	3	3	3
	3	inf	inf	4	4	4	4	4	4	4	4
Vertices	4	inf	7	7	7	7	7	7	7	7	7
	5	inf	3	3	3	3	3	3	3	3	3
	6	inf	inf	inf	12	12	12	12	12	12	12
	7	inf	inf	11	11	11	11	11	11	11	11
	8	inf	inf	9	9	9	9	9	9	9	9
	9	inf	inf	10	10	10	10	10	10	10	10

b)

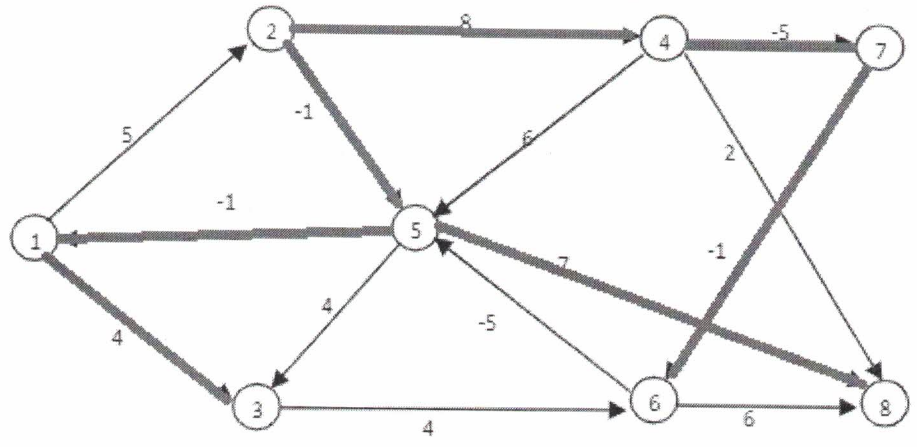


3. Suppose Bellman-Ford algorithm is run on the following graph, starting at node 2. Note: We do NOT start at node 1, we start at node 2.

a)

		Iterations							
		0	1	2	3	4	5	6	7
	2	0	0	0	0	0	0	0	0
Vertices	1	inf	inf	-1	-1	-1	-1	-1	-1
	3	inf	inf	3	2	2	2	2	2
	4	inf	8	8	8	8	8	8	8
	5	inf	-1	-1	-1	-1	-1	-1	-1
	6	inf	inf	inf	2	2	2	2	2
	7	inf	inf	3	3	3	3	3	3
	8	inf	inf	6	6	6	6	6	6

b)



4. Run the shortest paths in DAGs algorithm on the following DAG, starting at node 1.

a)

		Iterations								
		0	1	2	3	4	5	6	7	8
	1	0	0	0	0	0	0	0	0	0
Vertices	2	inf	inf	inf	inf	inf	inf	inf	inf	inf
	3	inf	4	4	4	4	4	4	4	4
	4	inf	inf	inf	inf	inf	inf	inf	inf	inf
	5	inf	6	6	6	2	2	2	2	2
	6	inf	inf	7	7	7	7	7	7	7
	7	inf	inf	inf	inf	inf	inf	inf	inf	inf
	8	inf	inf	inf	inf	13	9	9	9	9

b)
$$\begin{tikzcd}
2 & 1 \arrow[r, "4"'] & 3 \arrow[rrr, bend right] & 4 & 7 & 6 \arrow[r, "-5"] & 5 \arrow[r, "2"] & 8
\end{tikzcd}$$

4.4. Here's a proposal for how to find the length of the shortest cycle in an undirected graph with unit edge lengths.

> When a back edge, say (v, w), is encountered during a depth-first search, it forms a cycle with the tree edges from w to v. The length of the cycle is level$[v]$ − level$[w]$ + 1, where the level of a vertex is its distance in the DFS tree from the root vertex. This suggests the following algorithm:
>
> - Do a depth-first search, keeping track of the level of each vertex.
> - Each time a back edge is encountered, compute the cycle length and save it if it is smaller than the shortest one previously seen.

Show that this strategy does not always work by providing a counterexample as well as a brief (one or two sentence) explanation.

$$\begin{tikzcd}
A \arrow[r] & B \arrow[r] & C \arrow[r] & D \arrow[r] \arrow[lll, bend right, dashed] & E
\end{tikzcd}$$

The shortest cycle will contain A - D - E but with the algorithm explained in the question above it would find that A - B - C - D is the best. But this will fail once the cycle contains more than 1 back edge.

4.8. Professor F. Lake suggests the following algorithm for finding the shortest path from node s to node t in a directed graph with some negative edges: add a large constant to each edge weight so that all the weights become positive, then run Dijkstra's algorithm starting at node s, and return the shortest path found to node t.

Is this a valid method? Either prove that it works correctly, or give a counterexample.

$$\begin{tikzcd}
S \arrow[r, "-10"] \arrow[rrr, bend right, "15"'] & C \arrow[r, "-10"] & D \arrow[r, "25"] & T
\end{tikzcd}$$

No this is not a valid method. With the counterexample above the shortest path would be S - C - D - T unless a value of 15 or more is given where the shortest path will become S - T.