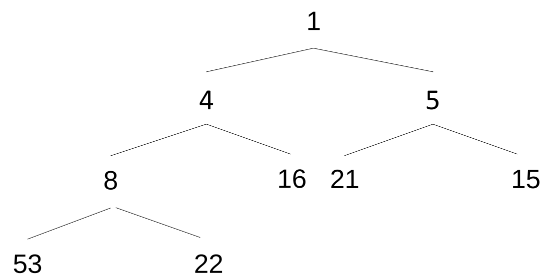
Spring 2019 CS372 Assignment #5 solution.

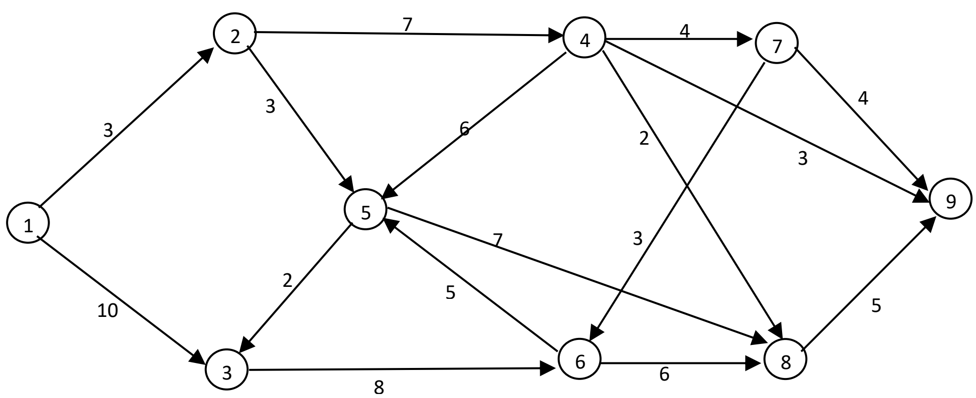
1. Build a min-heap heap by inserting elements one at a time in the following order 53, 8, 15, 16, 4, 21, 5, 22, 1. Show all your steps - draw a binary tree representing the heap after each insertion. Represent the resulting heap as a binary tree and as an array.

Solution: Only the resulting heap is shown:



Array representation: [1, 4, 5, 8, 16, 21, 15, 53, 22]

2. Suppose Dijkstra's algorithm is run on the following graph, starting at node 1.



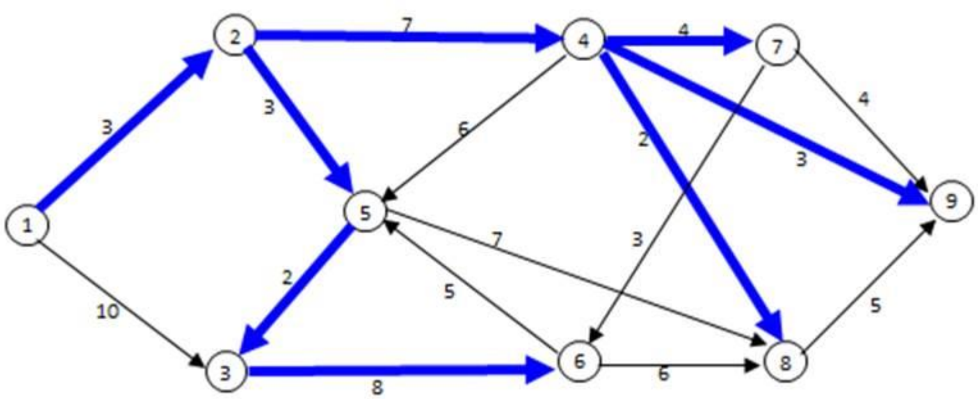
(a) Draw a table showing the intermediate distance values of all the nodes at each iteration of the algorithm.
(b) Show the final shortest-path graph.

Solution:

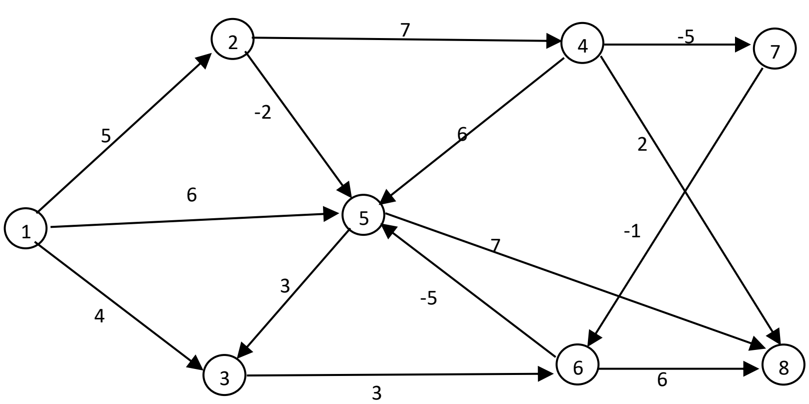
(a)

		Iteration									
		0	1	2	3	4	5	6	7	8	9
Vertices	1	0	0	0	0	0	0	0	0	0	0
	2	∞	3	3	3	3	3	3	3	3	3
	3	∞	10	10	8	8	8	8	8	8	8
	4	∞	∞	10	10	10	10	10	10	10	10
	5	∞	∞	6	6	6	6	6	6	6	6
	6	∞	∞	∞	∞	16	16	16	16	16	16
	7	∞	∞	∞	∞	∞	14	14	14	14	14
	8	∞	∞	∞	13	13	12	12	12	12	12
	9	∞	∞	∞	∞	∞	13	13	13	13	13

(b) The final shortest path is shown below in blue.



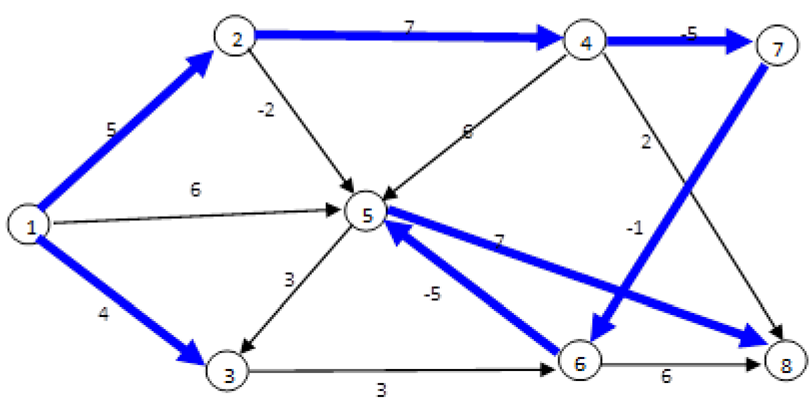
3. Suppose Bellman-Ford algorithm is run on the following graph, starting at node 1.



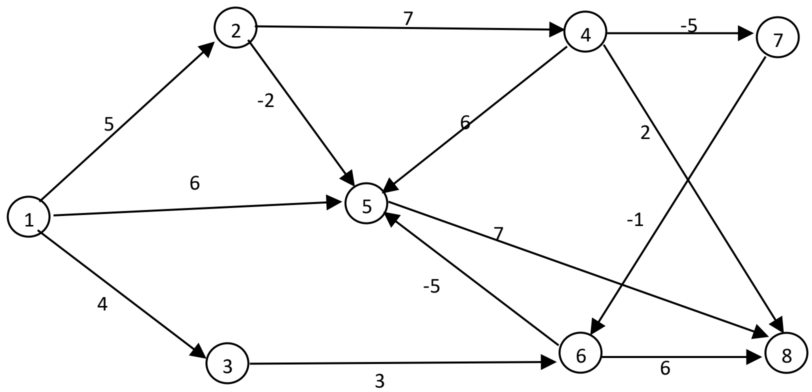
(a) Draw a table showing the intermediate distance values of all the nodes at each iteration of the algorithm.
(b) Show the final shortest-path graph.

Solution: (a) and (b)

		Iteration							
		0	1	2	3	4	5	6	7
Vertices	1	0	0	0	0	0	0	0	0
	2	∞	5	5	5	5	5	5	5
	3	∞	4	4	4	4	4	4	4
	4	∞	∞	12	12	12	12	12	12
	5	∞	6	3	2	2	1	1	1
	6	∞	∞	7	7	6	6	6	6
	7	∞	∞	∞	7	7	7	7	7
	8	∞	∞	13	10	9	9	8	8



4. Run the shortest paths in DAGs algorithm on the following DAG, starting at node 2.



(a) Show all your steps including

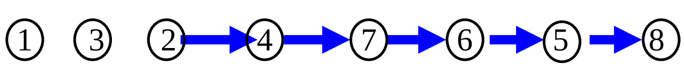
- the result of linearization and
- draw a table showing the intermediate distance values of all the nodes at each iteration of the algorithm.

(b) Show the final shortest-path graph.

Solution:

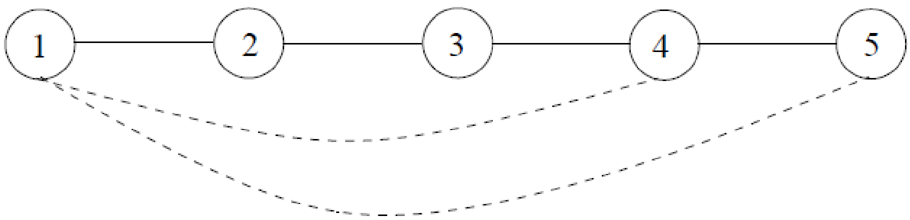
- Linearize the graph: run DFS and list the vertices in the order of decreasing post numbers. If the vertices are processed in increasing order then the following linearization is obtained: 1, 3, 2, 4, 7, 6, 5, 8.
- Note: vertex that is being processed at each iteration is listed in the parenthesis after the iteration number. For instance, 2(3) means that vertex 3 is being processed at iteration 2.

		0	1(1)	2(3)	3(2)	4(4)	5(7)	6(6)	7(5)	8(8)
Vertices	1	∞	∞	∞	∞	∞	∞	∞	∞	∞
	2	0	0	0	0	0	0	0	0	0
	3	∞	∞	∞	∞	∞	∞	∞	∞	∞
	4	∞	∞	∞	7_2	7_2	7_2	7_2	7_2	7_2
	5	∞	∞	∞	-2_2	-2_2	-2_2	-4_6	-4_6	-4_6
	6	∞	∞	∞	∞	∞	1_7	1_7	1_7	1_7
	7	∞	∞	∞	∞	2_4	2_4	2_4	2_4	2_4
	8	∞	∞	∞	∞	9_4	9_4	7_6	3_5	3_5

(b) 

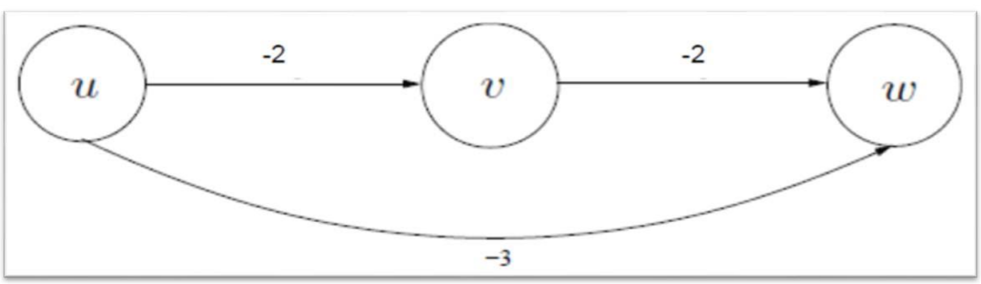
5. Exercise 4.4

Solution: The graph below is a counterexample: vertices are labeled with their level in the DFS tree, back edges are dashed. The shortest cycle consists of vertices 1−4−5, but the cycle found by the algorithm is 1−2−3−4. In general, the strategy will fail if the shortest cycle contains more than one back edge.



6. Exercise 4.8

Solution: The weighted graph below is a counterexample:



According to the algorithm proposed by Professor Lake we should add +4 to the weight of each edge. Then, the shortest path between u and w would be the edge (u,w) of weight 1. However, the shortest path in the original graph was u − v − w.