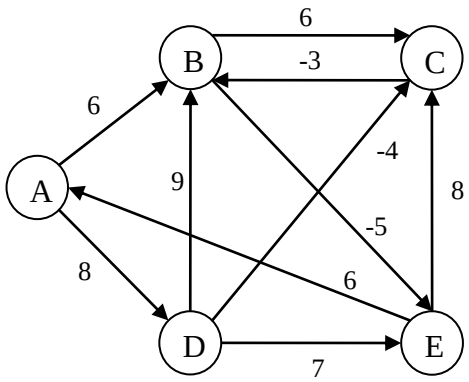
CSC 464/564 Homework #5 Instructor: Jeff Ward
Assigned: Wednesday, September 26, 2018 Due: 11:59pm, Wednesday, October 10
Covers: Paths in Graphs
Worth 25 points

#1 (7 points) Do Exercise 4.1 from the textbook. Here is the table that you can fill in for part (a):

nodes	initially	dequeue A	dequeue	dequeue	dequeue	dequeue	dequeue	dequeue
A	0,nil							
B	∞,nil							
C	∞,nil							
D	∞,nil							
E	∞,nil							
F	∞,nil							
G	∞,nil							
H	∞,nil							

For part (b) the problem is asking you to draw a tree rooted at A that has an edge from node u to node v if and only if u is the predecessor of v on the shortest path from A to v that was found by the algorithm.

#2 (7 points) Repeat the previous problem, but use the Bellman-Ford algorithm on the following graph. Again, use A as the starting point. Do not forget to draw the final shortest-path tree.



Also, answer the following questions:

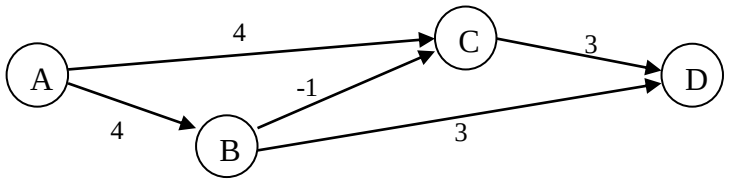
Which edges above, if any, were used more than once to update the distance value of a node?

If we run an additional iteration of the algorithm's main loop, does it detect a negative weight cycle?

Here is the table that you can fill in for part (a):

edges	nodes	initially	Iteration 1	Iteration 2	Iteration 3	Iteration 4
EA 6	A	0,nil				
AB 6 CB -3 DB 9	B	∞,nil				
BC 6 DC -4 EC 8	C	∞,nil				
AD 8	D	∞,nil				
BE -5 DE 7	E	∞,nil				

#3 (4 points) Repeat Problem #1 (parts a and b), yet again. This time use the dag-shortest-paths algorithm from Section 4.7, applied to the following graph:



Here is the table that you can fill in for part (a):

	initially	A	B	C	D
A	0,nil				
B	∞,nil				
C	∞,nil				
D	∞,nil				

#4 (7 points) Consider an empty 3-ary min-heap. (Recall that a "min-heap" is one where the uppermost, i.e. highest priority, elements have the lowest values. We use a min-heap for Dijkstra's algorithm.)

(a) Draw the heap that would result if you inserted the values 1, …, 10 in increasing order. (You are not required to show the intermediate states of the heap for this part.)

(b) Trace the states of the heap that would result if you inserted an additional copy of the value 1 into the heap that you drew for part (a). ("Trace" here includes showing the underlying tree when the element has first been added, and then showing the results of each element swap.)

(c) On the heap that resulted from part (b), trace the results of performing a decreasekey operation that changes the element with value 7 to a value of 0.

(d) On the heap that resulted from part (c), trace the results of performing a single deletemin operation.

#5 (4 bonus points) Consider the runtime analysis of Dijkstra's algorithm where the priority queue is implemented with a d-ary heap (see "Which heap is best?", Section 4.5, page 114). Assume that each time we run the algorithm, we will set the variable $d = |E| / |V|$. For the purposes of this exercise, you may assume that d is always an integer ≥ 2. (In practice, you could just round to the nearest integer and take the maximum with 2.) If we will run the algorithm only on a family of graphs for which $|E| = |V|^2 / c$, where c is a fixed constant, then show that the big-O runtime is as good as for Dijkstra's algorithm with the priority queue implemented by a simple unordered array. Hint: Find an equation that expresses d in terms of |V| and c.