TOTAL: 9/13

Homework W8 (15 pts)

1/5

1. [5 pts] The Bellman-Ford algorithm, when run on a graph with n nodes, relaxes edges n times. The first n-1 times it tries to find a shorter path from the start node to other nodes, and the last time is used to check for negative weight cycles. Under what circumstances, if any, can you run the algorithm fewer than n times and be sure you have the correct shortest path graph? Please briefly explain, using no more than the rest of this page (and probably a lot less).

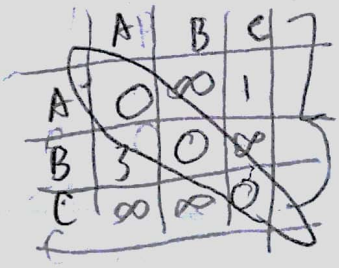
This doesn't answer the question, because I asked whether there was any time that you didn't need to relax the edges all n-1 times. (-4)

- If all the paths have n-1 edges then they will have all been relaxed n-1 times. Also if there is not a negative weight cycle. If these two conditions are met then we have the shortest path graph.

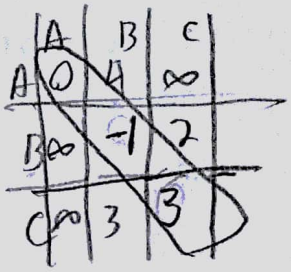
2. [5 pts] How would you use the Floyd-Warshall algorithm to determine whether or not a graph had a negative-weight cycle? Explain briefly.

4/5

For Floyd-Warshall when we make our adjacency matrix in our table the values for the diagnol entries would not all be zeros. This is the easiest way to detec a negative weight cycle.



This means no negative cycles



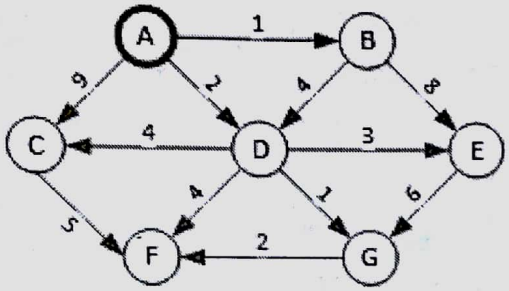
this means there is a negative cycle.

Not all 0s, and at least one negative (-1)

3. [5 pts] Apply the DAG shortest path algorithm as done in the lecture notes and text to find the shortest paths from source node A to every other node. As a reminder, here is pseudocode for the algorithm based on info from Database Zone:

4/5

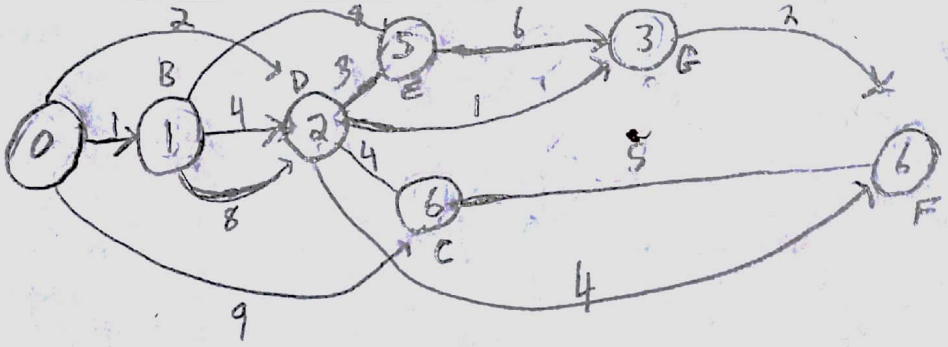
a. Topologically sort the graph, starting from the source node.
b. Set the distance to the source to 0, and set the distances to al other nodes to ∞.
c. For each node u in the sorted graph:
   i. Walk through all neighbors v of u
   ii. If dist(v) > dist(u) + w(u,v), dist(v) = dist(u) + w(u,v) and π(v) = u.



Part A: [2 pts] List the topological sort of this graph. That is, list the nodes left to right from the source in the order of a topological sort (there are multiple right answers).

F is last (-0.5)

#1: A ; #2: B ; #3: D ; #4: C ; #5: F ; #6 E ; #7: G



Part B: [3 pts] List the shortest path distances and the predecessors of each node. (Again, in case you have an error, show your work for partial credit.)

Node	A	B	C	D	E	F	G
Dist(A,node)	0	1	6	2	5	6	3
Π(node)	⊘	A	D	A	D	D	D

5
G
(-0.5)

TOTAL: 13/15

Homework W9 (15 pts)

1. [5 pts] Prove by mathematical induction that the complete recursion tree for computing the n^{th} Fibonacci number has Fib(n+1) leaves.

5/5

Case1: Fib(0) tree number of leaves = Fib(0+1) leaves

Case2: Fib(1) tree number of leaves = Fib(1+1) leaves

Inductive Hypothesis: Fib(n) tree number of leaves = Fib(n+1) leaves

: Fib(n+1) tree number of leaves = Fib((n+1)+1) leaves

Left side	Right side
Base Cost: Fib(0) tree number of leaves	= Fib(0+1) leaves
↳ = 0	⟹ = 1
by definition of the fibonacci sequence	by definition of the fibonacci sequence

therefore this statement is false

Extra Explanation:

Fib(0) or the 0th fibonacci number has only a zero which is not considered even a leaf node.

3/5

2. [5 pts] Consider the following three observations:

- "…lowest-cost-first search is typically exponential in both space and time. It generates *all* paths from the start that have a cost less than the cost of a solution." (Poole, 2nd ed., Section 3.5.4, last sentence)
- Dijkstra's algorithm runs in time $O(V^2)$ (Cormen, et. al., Ch. 24)
- Lowest-cost-first search is Dijkstra's algorithm where you terminate the search after you have found the shortest distance to the goal node.

These three statements seem contradictory. How do you resolve the contradictions?

The statements from poole and Cormen books are correct. The last statement is incorrect.

It is incorrect because it suggests that LCFS is at worst the same time complexity as Dijkstra's algorithm. Yet we know that time complexity for LCFS is exponetial while it is polynomial for Dijkstras Algorithm.

You basically said that the 3rd statement is incorrect because the first two statements claim different running times, which isn't really an explanation why. (-2)

5/5

3. [5 pts] Determine the h-values for the nodes 1, 2, 3, 4, and 5 in Johnson's Algorithm for this graph. You don't need to show your work, just get the correct answers.

h(1) = 0

h(2) = -1

h(3) = -5

h(4) = 0

h(5) = -4