CS 146 (Taylor)	**Fall 2018**
Data Structures & Algorithms	**Rote Exam 2 Template (?? pages)**

Cell phones, pagers, etc. should all be off! If such a device goes off, you might be officially finished with your test. No tools (calculators, computers, sliderules, screwdrivers, friends, books, or notes) should be used for the test, other than a pen/pencil, your wits and knowledge. **You should try to answer all questions for the rote exams.** Good luck.

1. (10 points) You run the strongly connected components algorithm on the following graph. Assume that all ties are broken alphabetically (in DFS, for instance). Show the components you get, in the order you get them, along with **all relevant work**. Show the final underlying component graph. The graph is given in adjacency list format.

   THE GRAPH, AS AN ADJACENCY LIST, WILL BE GIVEN HERE

   - First DFS discovery/finish values:
     $a =$ __/__, $b =$ __/__, $c =$ __/__, $d =$ __/__, $e =$ __/__, $f =$ __/__, $g =$ __/__, $h =$ __/__.
   - Vertex order: _____.
   - New graph to search, as an adjacency list:
   - Second DFS discovery/finish values:
     $a =$ __/__, $b =$ __/__, $c =$ __/__, $d =$ __/__, $e =$ __/__, $f =$ __/__, $g =$ __/__, $h =$ __/__.
   - Component graph, with each component (and its vertices) given in the order the algorithm finds it:

2. (10 points) Simulate the DAG Single Source Shortest Path algorithm on the graph below, using vertex s as a source. Use Kahn's topological sort algorithm, assuming that it initially traverses all vertices alphabetically to initialize it's FIFO Queue. Also assume that within each vertex, its edge list is traversed alphabetically. **Note:** the graph has a cycle. Like in our 3rd programming assignment, you should just use the ordering you get from Kahn's algorithm of vertices not waiting on the cycle, to compute your shortest paths. Because there is a cycle in the graph, your resulting shortest path tree might not actually be correct for the given graph.

   The assumption for the algorithm is that the graph is represented by a list of outgoing edges for each vertex, but for the purpose of this question, I will give the graph in adjacency matrix format, with weights, so that you can practice with that format as well. (Blank means no edge.) Show all work: in the vertical blanks of the table, you will fill in the values of your Queue in order, with the earlier values in the top blank. Within the table fill in the initial values, and any update that changes a value. **(Only fill in updates that change values.)**

   To the right of that work, show your final tree.

   THE GRAPH, AS AN ADJACENCY MATRIX, WILL BE GIVEN HERE.

dist/π	a	b	c	d	e	f	g	s
init	/	/	/	/	/	/	/	/
__	/	/	/	/	/	/	/	/
__	/	/	/	/	/	/	/	/
__	/	/	/	/	/	/	/	/
__	/	/	/	/	/	/	/	/
__	/	/	/	/	/	/	/	/
__	/	/	/	/	/	/	/	/
__	/	/	/	/	/	/	/	/
__	/	/	/	/	/	/	/	/

3. Consider the following graphs for each of the two subproblems. Each subproblem can be answered (or left blank) independently of the other (subject to the 4 total blank for partial credit rule).

   [a] (10 points) You are running Prim's MST algorithm on the graph below and left, starting with vertex (some vertex named here). You are about to take vertex (some other vertex named here) out of the min-heap,

but have not done so yet. Show all work done so far: the order that vertices were considered, and weight and π values if they were ever stored with the vertex. Also, for that instant in time, categorize each edge (every one of them) of the graph from the following list:

(a) Already known to be in the MST.

(b) The edge has not been seen yet.

(c) The edge has been seen, and is known to not be in the MST.

(d) The edge has been seen, and is stored with a vertex in the priority queue.

Finally, highlight the edges of the tree that has been found so far.

GIVE GRAPHS HERE

[b] (10 points) You are in the middle of running Kruskal's Minimum Weight Spanning Tree algorithm, using the Disjoint Set data structure, *with union by size and path compression.* The graph is given above to the right. You are about to consider the edge with weight X. Draw the corresponding disjoint set data structure for the graph, *before and after the edge with weight X.* Assume that, for each edge of the graph, the edge is called with the alphabetically first vertex first, and that for sets of the same size, the first set joins to the second.

4. (10 points) Run Dijkstra's Single Source Shortest Path algorithm, from vertex s, on the graph below. Show all work: in the vertical blanks of the table, fill in vertices in the order that Dijkstra's considers them, with the earliest value in the top blank. Within the table fill in the initial values, and any update that changes a value. **(Only fill in updates that change values.)**

   To the right of that work, show your final tree.

GRAPH HERE

dist/π	a	b	c	d	f	s
init	/	/	/	/	/	/
__	/	/	/	/	/	/
__	/	/	/	/	/	/
__	/	/	/	/	/	/
__	/	/	/	/	/	/
__	/	/	/	/	/	/
__	/	/	/	/	/	/

5. (10 points) You are in the middle of running the Floyd-Warshal algorithm. You have just completed calculating D^3, Π^3, and Φ^3. (THESE WILL BE FILLED IN.) Follow the algorithm to calculate values for D^4, Π^4, and Φ^4. Fill in those values below. (You only need to do it once, I just give the table twice to help you make it legible if it comes out poorly once. Your answer should be legible.)

D^3	1	2	3	4	5		D^4	1	2	3	4	5		D^4	1	2	3	4	5
1	0						1							1					
2		0					2							2					
3			0				3							3					
4				0			4							4					
5					0		5							5					

Π^3	1	2	3	4	5		Π^4	1	2	3	4	5		Π^4	1	2	3	4	5
1	-						1							1					
2		-					2							2					
3			-				3							3					
4				-			4							4					
5					-		5							5					

Φ^3	1	2	3	4	5		Φ^4	1	2	3	4	5		Φ^4	1	2	3	4	5
1	-						1							1					
2		-					2							2					
3			-				3							3					
4				-			4							4					
5					-		5							5					

6. (10 points) For the following graph, you are in the midst of running the Bellman Ford algorithm, from source vertex s. Relax all edges in alphabetical order. (Edge $a \to d$ would precede (go before) $b \to c$, and edge $a \to b$ would precede $a \to c$.) For each vertex, you should show every distance and π value that is *ever* stored as a distance estimate for the vertex. *Do not show estimates that are never stored with the vertex.* After one more iteration through all edges, show the shortest path tree that has been found so far.

   GIVE GRAPH HERE

7. (10 points) I will give you some kind of dynamic program similar to the memoized Fibonacci program that we saw. You will have to specify two orders: the order that recursive calls are *made* to the program, and the order that the table gets filled in. The assumption is that evaluation happens left to right: if in the Fibonacci program, you have a recursive call like "return f(n-1) + f(n-2)", that f(n-1) will be called on that line before f(n-2). However, anything that f(n-1) calls will return an answer before it does.