Test 3

Name: ______________________

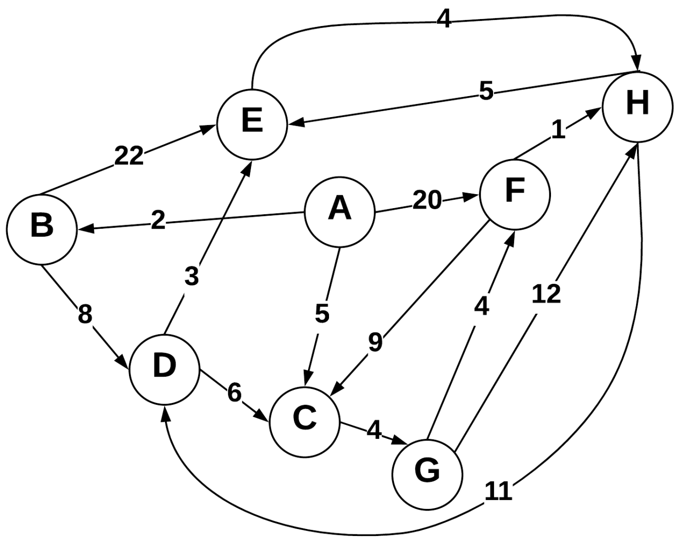
- Everything you turn in must be digitally created.
- No handwriting (except for signature below).
- You must work alone.
- Sharing of answers will result in a 0 on the exam, and possible F in the course.
- Send me your digitally created exam by Friday, May 4th by Midnight on a private slack message.
- Bring your printed signed copy by Monday Morning 10:00 am to my office.

Question	Possible	Score
1	10	
2	10	
3	10	
4	10	
5	10	
6	10	
7	10	
8	10	
9	10	
10	Bonus	
Total:		

By signing this, your saying "I worked alone and did not plagiarize":

____

1) Dijkstra's Algorithm

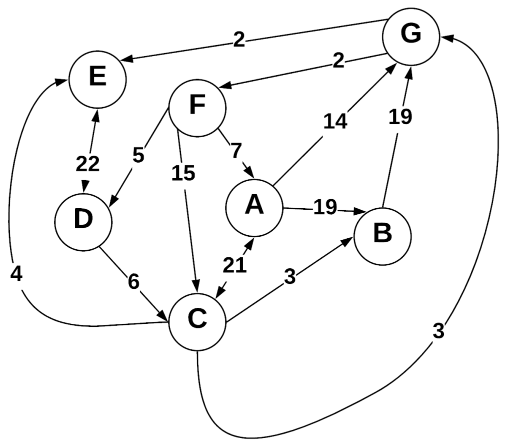


Use Dijkstra's algorithm to compute the shortest paths from vertex A to every other vertex. Show your work in the space provided below. As the algorithm proceeds, cross out old values and write in new ones, from left to right in each cell. If during your algorithm two unvisited vertices have the same distance, use alphabetical order to determine which one is selected first. Also list the vertices in the order which Dijkstra's algorithm marks them as discovered.

Vertices in Order of Discovery:

Vertex	Known	Cost	Previous
A			
B			
C			
D			
E			
F			
G			
H			

2) Prims Algorithm



Step through Prim's algorithm to calculate a minimum spanning tree starting from vertex *G.* Show your steps in the table below. As the algorithm proceeds, cross out old values and write in new ones, from left to right in each cell. If during your algorithm two unvisited vertices have the same distance, use alphabetical order to determine which one is selected first. Also list the vertices in the order which Prims algorithm discovers them.

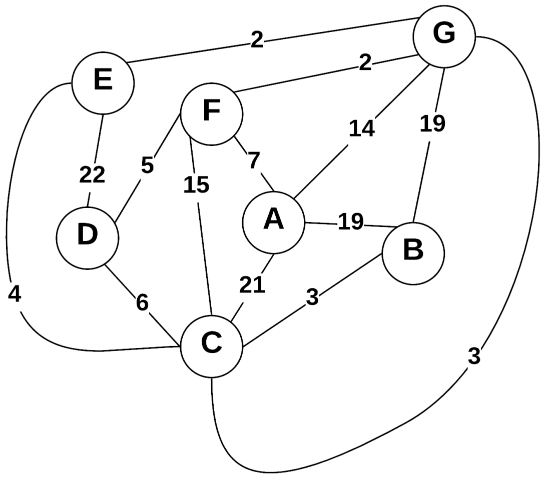
Vertices in Order of Discovery:

G								

- S = Vertices in spanning tree
- U = ! S (vertices not in S)
- Cut = edges going across cut listed alphabetically: (A B) , (C D) , etc.

S (spanning tree)	*U*	Cut (alphabetize)

3) Kruskel's Algorithm



Use Kruskal's algorithm to calculate a minimum spanning tree of the graph. Show your steps in the table below, including the disjoint sets at each iteration. If you can select two edges with the same weight, select the edge that would come alphabetically last (e.g., select E—F before B—C. Also, select A—F before A—B).

- Edge Added: put edges added to MST marked as (A B), (E G), etc.
- Edge Cost: weight of edge added
- Running cost is total weight of spanning tree at the point another edge is added.
- Disjoint sets starts as: (A) (B) (C) (D) (E) (F) (G) , and as edges are added => (A) (B C) (D) (E) (F) (G)

Edge Added	Edge Cost	Running Cost	Disjoint Sets

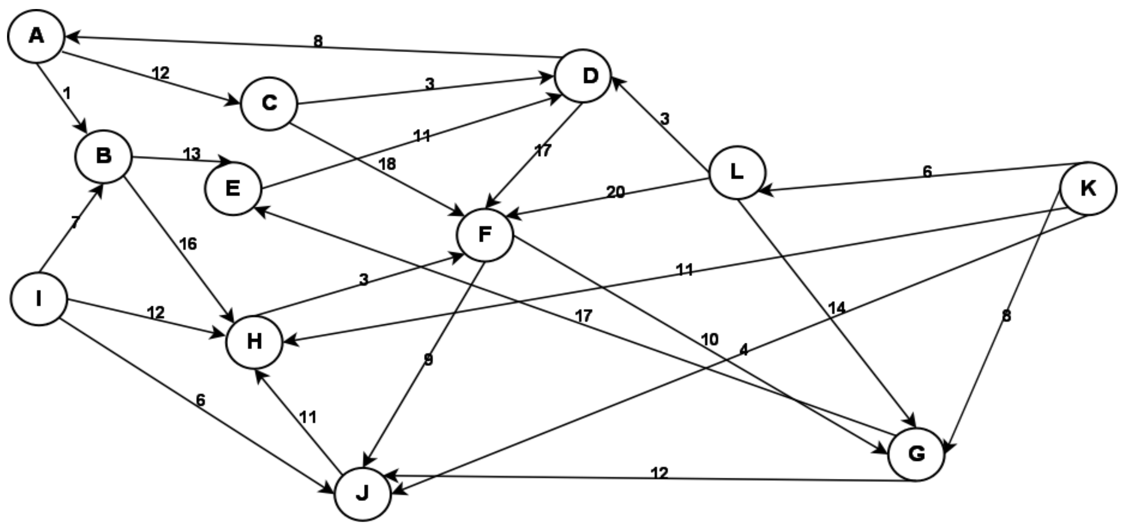
4) Prims Vs Kruskels

Explain why Prim’s algorithm is better for dense graphs, while Kruskal’s algorithm is better for sparse graphs. What data structures are used to represent each?

5) Greedy Algorithms

A. Define “Greedy Algorithm”
B. Give an example of a greedy algorithm with explanation of its greediness and performance.
C. Can greedy algorithms produce “optimal” solutions? Short explanation.

6) Graph Traversals



Given the above graph, provide the output of a breadth first and a depth first search. Make choices based on smallest edge weight, then alphabetical to break ties. Start at node A for both.

Depth First:

A											

Breadth First:

A											

7) Graph Storage / Manipulation

Given that a weighted directed graph is represented as an adjacency matrix called *adjM*, write a method that reverses all the edges of the graph. That is, for every edge (A , B) in the original graph, there will be an edge (B , A) in the reversed graph with the same weight. Your function should be called *reverse* .

8) Graph Traversal

Write a method that returns whether a graph is a tree. Your method takes a graph *G=(V,E)* as the input and outputs a boolean value. Your method should be called *isTree*().

9) Huffman Coding

(A) What is an optimal Huffman code for the following set of frequencies, based on the first 8 Fibonacci numbers:

a:1 b:1 c:2 d:3 e:5 f:8 g:13 h:21.

Show your answer as a tree. *Note:* assume that the ordering on the nodes is first by the frequency, and then by the alphabetic order of the node label, so that ab:2 precedes c:2; the node labels are alphabetized too, so that we have a node ab:2 but not ba:2.

(B) Use the code from part (a) to decode the string 11111111111001111101. (As a check: the result should be the name of something that is often yellow.)

10) Bellman Ford (Optional)

Using the graph from question 3, show a Bellman Ford solution.