1 (3') Minimum Spanning Tree and Topological Sort

Each question has one or more correct answer(s). Select all the correct answer(s). For each question, you get 0 point if you select one or more wrong answers, but you get 0.5 point if you select a non-empty subset of the correct answers.

Note that you should write you answers of section 1 in the table below.

Question 1	Question 2	Question 3
CD	D	ACD

Question 1. *Topological Sort can be carried out on what kinds of graphs:*

(A) All weight acyclic graphs

(B) All directed graphs

(C) All (Directed) Trees ✓

(D) All unweight directed acyclic graphs ✓

(E) All cyclic directed graphs

Question 2. *Consider the following algorithm that attempts to compute a minimum spanning tree of a connected undirected graph G with distinct edge costs. First, sort the edges in decreasing cost order (i.e., the opposite of Kruskal's algorithm). Initialize T to be all edges of G. Scan through the edges (in the sorted order), and remove the current edge from T if and only if it lies on a cycle of T.*
Which of the following statements is true?

(A) The output of the algorithm will never have a cycle, but it might not be connected. ✗

(B) The algorithm always outputs a spanning tree, but it might not be a minimum cost spanning tree.

(C) The output of the algorithm will always be connected, but it might have cycles.

(D) The algorithm always outputs a minimum spanning tree. ✓

Question 3. *You are given a connected undirected graph G with distinct edge costs, in adjacency list representation. You are also given the edges of a minimum spanning tree T of G. This question asks how quickly you can recompute the MST if we change the cost of a single edge. Which of the following are true? [RECALL: It is not known how to deterministically compute an MST from scratch in $O(m)$ time, where m is the number of edges of G.]*

(A) Suppose $e \in T$ and we increase the cost of e. Then, the new MST can be recomputed in $O(m)$ deterministic time.

(B) Suppose $e \notin T$ and we increase the cost of e. Then, the new MST can be recomputed in $O(m)$ deterministic time. ✗

(C) Suppose $e \in T$ and we decrease the cost of e. Then, the new MST can be recomputed in $O(m)$ deterministic time.

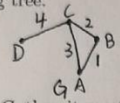
(D) Suppose $e \notin T$ and we decrease the cost of e. Then, the new MST can be recomputed in $O(m)$ deterministic time.

2 (4')True or False

The following statements may or may not be correct. In each case, give a counterexample (if it isn't correct). Always assume that the graph $G = (V, E)$ is undirected. Do not assume that edge weights are distinct unless this is specifically stated.

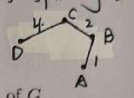
- If graph G has more than $|V| - 1$ edges, and there ia a unique heaviest edge, then this edge cannot bt part of a minimum spanning tree.

False. counterexample:



G has 4 vertices and 4 edges (which is larger than 4-1.

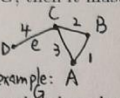
minimum
G's spanning tree:



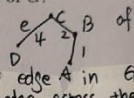
It contains the heaviest edge: CD

- If e is part of some MST of G, then it must be a lightest edge across some cut of G.

True. e can be the lightest edge in a cut of G if the cut contains e only. For example:



e is part of the MST:



of G

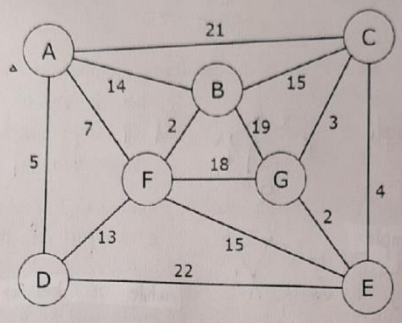
Even if e is the heaviest edge in G, it can be a lightest edge across the cut of G:

- Prim's algorithm works correctly when there are negative edges.

True. For a graph G with some negative edges, denote the lightest edge as e, and the weight of it is $x^{(x<0)}$. We can add $(-x+1)$ to the weight of every edge. After that, we can get a graph G' with all the weight of edges larger than 0. Then use Prim's algorithm to compute a minimum spanning tree. After that, we can just let the weight of every edge in the MST minus $(-x+1)$. Hence, Prim's algorithm works correctly when there are negative edges.

3 (18') Minimum Spanning Tree

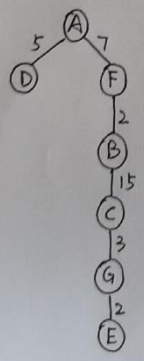
Given the following undirected, weighted graph, compute two minimum spanning trees.



(1)(5') Step through Prim's algorithm to calculate a minimum spanning tree starting from vertex **A**. 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. As an example, Row B is filled already.

	Vertex	Distance	From(Vertex)
①	A	-	-
⑤	B	~~14~~ 2	~~A~~ F
⑥	C	~~21~~ 15	~~A~~ B
②	D	5	A
	D		
⑦	E	~~22~~ ~~15~~ ~~4~~ 2	~~D~~ ~~F~~ ~~C~~ G
④	F	7	A
③	G	~~18~~ 3	~~F~~ C

(2)(2') Draw the MST you get by this algorithm:



(3)(9') Step through 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	Edge Cost	Disjoint Sets
-	-	(A)(B)(C)(D)(E)(F)(G)
E-G	2	{A}{B}{C}{D}{F}{E,G}
B-F	2	{A}{C}{D}{E,G}{B,F}
C-G	3	{A}{D}{B,F}{C,E,G}
A-D	5	{A,D}{B,F}{C,E,G}
A-F	7	{A,B,D,F}{C,E,G}
E-F	15	{A,B,C,D,E,F,G}

(4)(2') Draw the MST you get by this algorithm: