Due April 18 (11:55 a.m.)

Instructions

- **Answer each question on a separate page.**
- **Honors questions are optional. They will not count towards your grade in the course. However you are encouraged to submit your solutions on these problems to receive feedback on your attempts. Our estimation of the difficulty level of these problems is expressed through an indicative number of stars** $('*' =$ **easiest**$)$ **to** $('*****' =$ **hardest**$)$**.**
- **You must enter the names of your collaborators or other sources as a response to Question** 0**. Do NOT leave this blank; if you worked on the homework entirely on your own, please write "None" here. Even though collaborations in groups of up to** 3 **people are encouraged, you are required to write your own solution.**

Question 0: List all your collaborators and sources: ($-\infty$ points if left blank)

Solution: None.

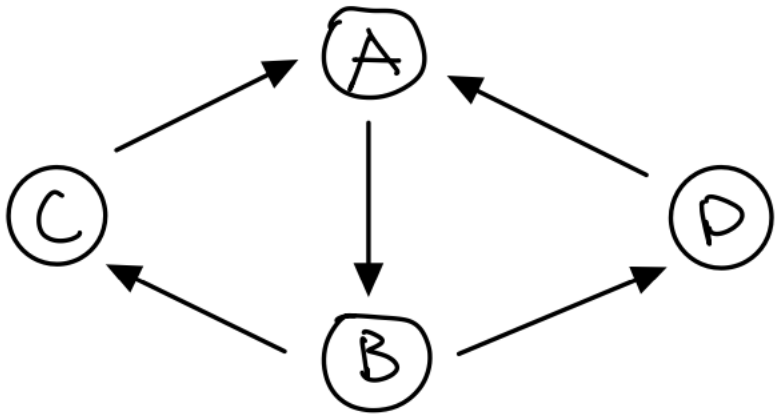
Question 1: (10 points)

We present the following (false) claim:

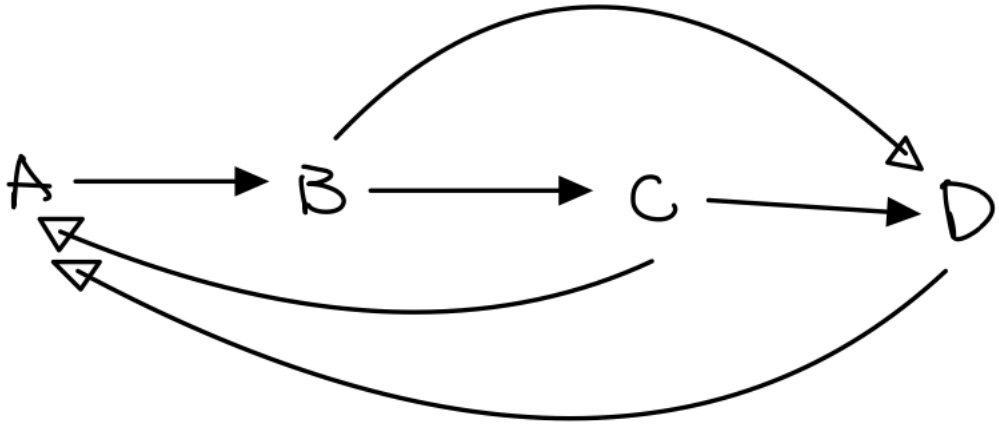
Claim 1. *If a directed graph G contains cycles (i.e., is not acyclic), then topological sort produces a vertex ordering that minimizes the number of "bad" edges that are inconsistent with the ordering produced. More precisely, a bad edge is one going from a vertex later in the ordering to an earlier vertex.*

Disprove this claim by providing a counterexample. Briefly justify why your graph fails the claim.

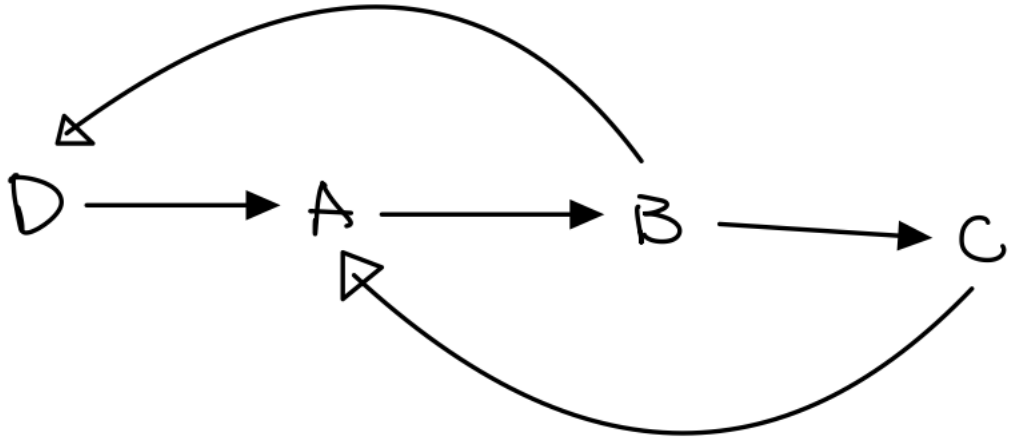
Solution:



If we begin topological sort on this graph from vertex A. This is the result that we will get



which has three backward edges. But the optional solution that minimizes bad edges is the following graph. It only has two bad edges.



Question 2: (5+5+10=20 points)

Suppose the New York Botanical Gardens are trying to drum up interest for their tree and shrub collections by offering tours through the arboretum, and you are tasked with reviewing their proposed routes. There are a set V of trees they want some tour to stop at, and a set E of routes from tree to tree their proposed tours would make. In order to make sure all the trees are visited, they want the following property:

Property 1. *For all pairs $v, u \in V$, we must have a path from u to v or a path from v to u (or both).*

They provide you with many proposed plans and a strict deadline for your feedback, and so you want to develop an efficient algorithm to automate this task.

1. Suppose $G = (V, E)$ is a directed acyclic graph. Design an $O(|V|+|E|)$ time algorithm to determine whether or not the given graph G satisfies the desired property.

Solution: Run a topological sort on G. We would have a sorted sequence of vertex sorted in decreasing order of finishing time. Then iterate through the sequence from the beginning to check whether there is a edge between every single adjacent vertex in the sequence. If yes, then return true. If not, return false. The running would be the runing time of DFS and traversing the output of the topologically sorted list. $T(n) = O(|V| + |E|) + O(|V|) = O(|V| + |E|)$

2. Suppose $G = (V, E)$ is any directed graph. Design a $O(|V| + |E|)$ time algorithm to determine whether or not the given graph G satisfies the desired property.

Solution: Run SCC on graph G. We will have a directed acyclic graph G^{SCC}. Then run topological sort on G^{SCC}. We would have a sorted sequence of vertex (strongly connected components of G) sorted in decreasing order of finishing time. Then iterate through the sequence from the beginning to check whether there is a edge between every single adjacent vertex in the sequence. If yes, then return true. If not, return false.

3. Prove that your algorithm in (2) is correct and that it runs in the required time.

Solution:
Correctness:
First we need to proof that if there is a edge connecting two strongly connected components of G, C and C' then there is a path between all vertexes in the C and C". For any two vertexes u, v in the two SCC, if they belong to the same SCC there is obvisously a path between them. If they belong to different SCC then, let's let the directed edge connecting C and C' be (c_u, c_v). Without the loss of generality, assume u and c_u are in both in SCC C and v and c_v are in both in C. Then going from u to c_u, then take the edge (c_u, c_v), and going from c_v to v would be a path from u to v. Secondly, we need to prove that there is a path between every single vertex in a directed acyclic graph G if and only iff there is an edge connecting every adjacent vertex in the sequence outputed by topological sort of G. First, prove the forward propostion by contraposition. Assume there are two vertex u, v are adjacent in the topological sort output and u has later finishing time then v. Since in output of topological sort of a acyclic graph, all the edge will be directing from a vertex with a higher finishing time to a vertex with lower finishing time. Then if there is not a edge from u to v, that means there is no path between u, v since it is impossible to have back edge in DFS of acyclic graph. And the backward propostion is very easy to prove since if there is an edge connecting every adjacent vertex, the path just go along the output of topological sort. With the above two propostion, the algorithm is automaticly correct since there must be a path between any two vertex in the same SCC. The last thing to prove is if the topological sort of G^{SCC} has two adjacent components C and C' in the output sequence there are no path between any vertex in component C to C'. Prove by contradiction. There are two adjacent components C (C has a later finishing time) and C' in the sequence of topological sort of G^SCC that they don't have a edge between them, assume there is a path between vertex $u \in C$ and vertex $v \in C'$. The path cannot be from v to u because topological sort of DAG does not have backward edges. Any edge coming out from C' could only go to components that

has a earlier than itself since there is no backward edge. Then it is impossible to reach C. The path cannot be from u to v also because there are no backward edge. And edge coming out from C could only go to components that has a earlier than C' since there is no edge between them. Then it is impossible to reach C'. Thus, if the G^{SCC} graph satisfies its topological sort has an edge connecting every adjacent vertex in the output sequence. Then we satisfies property 1. If not, property 1 is not satisfies as proved above.

Runtime:

The Runtime of the algorithm has the following components. Runing SCC on graph G takes two DFS running time. And Runing topological sort on G^{SCC} takes at most $O(|V|+|E|)$ since there could be at most $|E|$ edges and $|V|$ vertexes in G^{SCC}. The last step iterate through output of topological sort of G^SCC and there are at most $|V|$ vertexes we need to go over then we need to check at most $|V|-1$ edges. Then the total running time would be

$$T(n) = 2O(|V|+|E|) + O(|V|+|E|) + |V| - 1 = O(|V|+|E|)$$

(Hint: Recall that any directed graph G can be decomposed into a DAG of strongly connected components)

Question 3: (5+5=10 points)

1. How many valid topological sorts does the directed graph G in Figure 1 below have? List all the valid topological sorts in the following table. One of them has been listed as an example, where node A has the last finish time and D has the first.

1.	A	B	C	F	E	D
2.	A	E	B	C	F	D
3.	A	B	E	C	F	D
4.	A	B	C	E	F	D

Solution: The are 9 valid topological sorts.

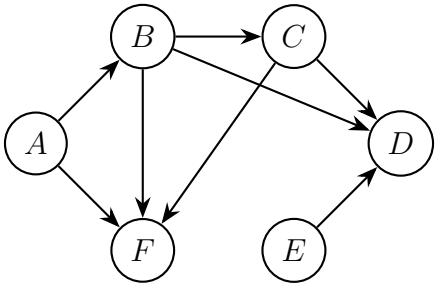
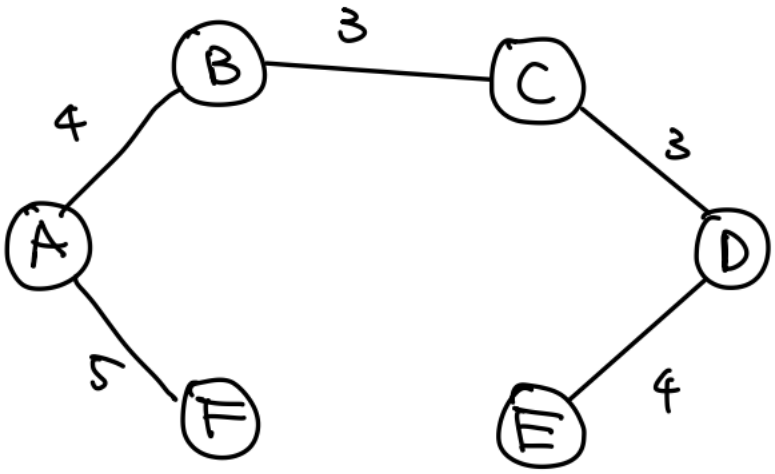


Figure 1: Directed G for topological sort.

2. Find a minimum spanning tree (MST) of the undirected graph G in Figure 2 below. Also, list the cost of your MST.

Solution:



The cost of my MST is $5 + 4 + 3 + 3 + 4 = 19$

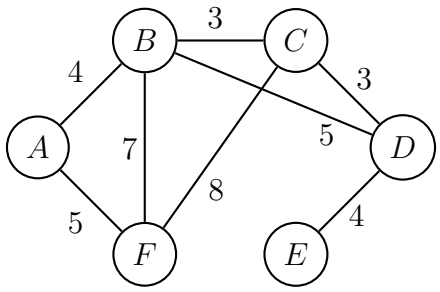


Figure 2: Undirected G for minimum spanning tree.

Question 4: (5+5=10 points)

Suppose we are given both an undirected graph G with weighted edges and a minimum spanning tree T of G. Further we are given an edge e of G and its new weight in the modified graph G'. In each of these cases, prove why T remains an MST of the modified graph G'.

1. The weight of one edge $e \in T$ is decreased.

Solution: Prove by contradiction. Assume that there exist another spanning tree T' that has a smaller weight after the weight of one edge $e \in T$ is decreased by $a > 0$. Let $*$ denote the state after weight change. Then the weight of T after decreasing weight is $w^*(T) = w(T) - a$. If $e \in T'$, then the weight of T' would be $w(T') = w^*(T') + a$. By assumption $w^*(T') < w^*(T)$, then $w^*(T') + a < w^*(T) + a$ which gives $w(T') < w(T)$. However this contradict with the condition that T is MST before the change. If $e \notin T'$, then the weight of T' would be $w(T') = w^*(T')$. Then $w^*(T') < w^*(T)$ and $w(T') \geq w(T)$ cannot be true at the same time swaping

$w^*(T) = w(T) - a$ and $w(T') = w^*(T')$ into the second inequality we have $w^*(T') \geq w^*(T) - a$ which we have another contradiction. Then we have proved T remains an MST of the modified graph G'.

2. The weight of one edge $e \notin T$ is increased.

Solution: Prove by contradiction. Assume that there exist another spanning tree T' that has a smaller weight after the weight of one edge $e \notin T$ is increased by $a > 0$. Then the weight of T after increasing weight is $w^*(T) = w(T)$. If $e \in T'$, then the weight of T' would be $w^*(T') = w(T') + a$. By assumption $w^*(T') < w^*(T)$, then $w^*(T') - a < w^*(T) - a$ which gives $w(T') < w(T) - a$. However this contradict with the condition that T is MST before the change. If $e \notin T'$, then the weight of T' would be $w(T') = w^*(T')$. Then $w^*(T') < w^*(T)$ and $w(T') \geq w(T)$ because there is no weight change. which we have another contradiction. Then we have proved T remains an MST of the modified graph G'.