

# Foundations of Algorithms, Spring 2022: Homework 5

Due: Wednesday, April 6, 11:59pm

## Problem 1 (5 points)

Suppose you are given a directed graph,  $G$ . Consider the following algorithm to determine the minimum number of edges that must be removed from  $G$  in order to make it acyclic:

- Use the DFS with finish times algorithm and arrange the nodes by decreasing finish time. We know that if  $G$  is acyclic, all of the edges will point in a common direction, from a higher finish time vertex to a lower finish time vertex.
- Count the number of edges that are ‘pointed the wrong way’ (from a lower finish time vertex to a higher finish time vertex). Return this number as the minimum number of edges that must be removed to make  $G$  acyclic.

Either prove that this algorithm is correct, or provide a counterexample in which the algorithm fails to correctly identify the minimum number of edges that must be removed from the directed graph to make it acyclic.

## Problem 2 (14 points: 10 for implementation / 4 for writeup)

You have a friend that is terrible with directions. When your friend asks you for directions on getting from point  $A$  to point  $B$ , you decide that the best response is to provide the fewest steps of directions possible that allows your friend to arrive at point  $B$ , even if it is not the most direct path (i.e. the path of fewest steps of directions may involve more intermediate vertices than other paths). Assume you are given a graph,  $G$ , containing  $n$  vertices and  $m$  edges, one vertex of which is point  $A$ , and another vertex of which is point  $B$ . Assume that  $G$  is connected. Give an  $O(m + n)$  algorithm that determines the fewest steps of directions needed to guide your friend from point  $A$  to point  $B$ . A step of directions is required any time your friend arrives at a vertex such that there is more than one possible edge that could be traversed next (not counting the edge just taken; your friend is smart enough to know you’d never send them back and forth on the same edge!).

## Problem 3 (20 points: 14 for implementation / 6 for writeup)

In a town full of one-way streets, the town council suddenly realized that it is not possible to get from every location to every other location. They want to fix this but their budget is limited - they can build a single new road. Help them! Given is a directed graph  $G = (V, E)$  (where vertices represent crossings and edges one-way streets). Determine if there exists a pair of vertices  $u$  and  $v$  such that adding an edge from  $u$  to  $v$  makes the graph strongly connected. Your algorithm should run in time  $O(n + m)$ , where  $n = |V|$  and  $m = |E|$ .

## Problem 4 (20 points: 14 for implementation / 6 for writeup)

Given is a weighted graph  $G = (V, E)$ , such that the vertices are partitioned into two non-empty sets,  $A$  and  $B$ . That is,  $A \cup B = V$ ,  $A \cap B = \emptyset$ ,  $|A| > 0$ , and  $|B| > 0$ .

Determine the minimum possible cost of a spanning tree for  $G$  satisfying the constraint that at most two edges can be included that cross from  $A$  to  $B$ . Your algorithm should run in  $O(m \log n)$  or  $O(n^2)$  time.