COM S 311 SPRING 2022
HOMEWORK 5

Due: April 26, 11:59 p.m. (note that the 26th is a Tuesday, not a Thursday)
Late Submission Due: April 27, 11:59 p.m. (25% penalty)

(1) (Exercise 22.4-3 of CLRS) Give an algorithm that determines whether or not a given undirected graph $G = (V, E)$ contains a cycle. Your algorithm should run in $O(V)$ time, independent of $|E|$.

(2) The following problem arises in automatic program analysis. For a set of variables $x_1, \ldots, x_n$, you are given some equality constraints, of the form "$x_i = x_j$" and some *disequality* constraints, of the form "$x_i \neq x_j$." The question is whether it is possible to satisfy all of the constraints. For instance, the constraints

$$x_1 = x_2, x_2 = x_3, x_3 = x_4, x_1 \neq x_4$$

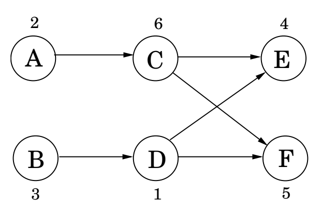
cannot be satisfied.

Give an efficient algorithm that takes as input m constraints over n variables and decides whether the constraints can be satisfied.

(3) You are given a directed graph $G = (V, E)$ in which each node $u \in V$ has an associated *price* p_u, which is a positive integer. Define the array `cost` as follows: for each $u \in V$,

`cost`$[u]$ = price of the cheapest node reachable from u (including u itself).

For instance, in the graph below (with prices shown for each vertex), the `cost` values of the nodes A, B, C, D, E, F are 2, 1, 4, 1, 4, 5, respectively.



Your goal is to design an algorithm that fills in the entire `cost` array (i.e., for all vertices).

(a) Give an $O(V + E)$ algorithm that works for directed *acyclic* graphs. (*Hint:* Handle the vertices in a particular order.)
(b) Extend this to an $O(V + E)$ algorithm that works for *all* directed graphs, not just acyclic ones. (*Hint:* First compute the strongly connected components of G.)

(4) You are given a connected undirected graph $G = (V, E)$ with positive edge weights, and a minimum spanning tree $T = (V, F)$, where $F \subseteq E$, with respect to these weights. Assume that G and T are given as adjacency lists. Now suppose the weight of a particular edge $e \in E$ is modified from $w(e)$ to a new value $\tilde{w}(e)$. You wish to quickly update the minimum spanning tree T to reflect this change, without recomputing the entire tree from scratch. There are four cases. In each case give a linear-time algorithm for updating the tree.

(a) $e \notin F$ and $\tilde{w}(e) > w(e)$.
(b) $e \notin F$ and $\tilde{w}(e) < w(e)$.
(c) $e \in F$ and $\tilde{w}(e) < w(e)$.
(d) $e \in F$ and $\tilde{w}(e) > w(e)$.

(5) Suppose you are given a timetable, which consists of:

- A set $\mathcal{A}$ of n airports, and for each airport a in $\mathcal{A}$, a minimum connecting time $c(a)$.
- A set $\mathcal{F}$ of m flights, and the following, for each flight f in $\mathcal{F}$:
  - Origin airport $a_1(f)$ in $\mathcal{A}$
  - Destination airport $a_2(f)$ in $\mathcal{A}$
  - Departure time $t_1(f)$
  - Arrival time $t_2(f)$

In the ***flight scheduling problem***, we are given airports a and b, and a time t; the goal is to compute a sequence of flights that allows one to arrive at the earliest possible time in b when departing from a at or after time t. Minimum connecting times at intermediate airports must be observed. Describe and justify an efficient algorithm for the flight scheduling problem. What is the running time of your algorithm as a function of n and m?

(6) (15 extra credit points) Let $G = (V, E)$ be a directed graph where every edge $e \in E$ has a positive weight $w(e)$ and let $s \in V$ be a specified source node of G. The ***bottleneck weight*** of a path P from s to node $u \in V$ is the minimum weight of an edge in P. The ***bottleneck shortest path problem*** is to find, for every node $u \in V$, an $s \rightsquigarrow u$ path P of maximum bottleneck weight.

  (a) Show that a shortest $s \rightsquigarrow u$ path (i.e., an $s \rightsquigarrow u$ path of minimum total weight) is not necessarily an $s \rightsquigarrow u$ path of minimum bottleneck weight and vice versa.
  (b) Give an $O(V + E)$-time algorithm that solves the bottleneck shortest path problem in a DAG. Justify your algorithm and analyze its running time.
  (c) Give an $O(E \log V)$-time algorithm that solves the bottleneck shortest path problem in an arbitrary directed graph. Justify your algorithm and analyze its running time.

Guidelines

- For each problem except problem 6, if you write the statement "I do not know how to solve this problem" (and nothing else), you will receive 20% credit for that problem. If you do write a solution, then your grade could be anywhere between 0% to 100%. To receive this 20% credit, you must explicitly state that you do not know how to solve the problem.
- You may discuss homework with classmates, but you must write the final solutions alone, without consulting anyone. Your writing should demonstrate that you completely understand the solution you present.
- Remember that, often, the clearest way to describe an algorithm is to give a brief overview of the algorithm, followed by pseudocode.
- When presenting an algorithm, always argue its correctness and analyze its running time.
- When writing a proof, make sure it clear and rigorous.
- *Homework solutions must be typed.* We can make exceptions for certain diagrams, which can be hand-drawn and scanned. We reserve the right not to grade homework that does not follow the formatting requirements.
- Submit a pdf version of your assignment via Canvas. Please make sure that the file you submit is not corrupted and that its size is reasonable (no more than, say, 10-11 MB).

  If we cannot open your file, your homework will not be graded.

- Any concerns about grading should be expressed within one week of returning the homework.

Note. We reserve the right to grade only a subset of the problems assigned. Which problems will be graded will be decided after the submission deadline.