CS 2443 : Examination 2

Department of Computer Science, IIT Hyderabad

Due date: 17 March 2022, 05:00 AM IST

- Read the questions carefully and answer only to the questions.
- When asked to give an algorithm you need to prove correctness as well as analyse running time of your algorithm.
- Upload the answers and mark them towards the questions in gradescope.
- Maintain academic honesty.

---

1. Decide whether you think the following statement is true of false. If it is true, give a short explanation. If it false, give a counterexample.

   In every instance of the Stable Matching Problem, there is a stable matching containing a pair (m, w) such that m is ranked first on the preference list of w and w is ranked first on the preference list of m.

   (3 marks)

2. Decide whether you think the following statement is true of false. If it is true, give a short explanation. If it false, give a counterexample.

   Consider an instance of the Stable Matching Problem in which there exists a man m and a woman w such that m is ranked first on the preference list of w and w is ranked first on the preference list of m. Then in every stable matching S for this instance, the pair (m, w) belongs to S.

   (3 marks)

3. Suppose we have two television networks, whom we'll call $\mathcal{A}$ and $\mathcal{B}$. There are n prime-time programming slots, and each network has n TV shows. Each network wants to devise a *schedule*—an assignment of each show to a distinct slot—so as to attract as much market share as possible.

   Here is the way we determine how well the two networks perform relative to each other, given their schedules. Each show has a fixed *rating*, which is based on the number of

people who watched it last year; we'll assume that no two shows have exactly the same rating. A network *wins* a given time slot if the show that it schedules for the time slot has a larger rating than the show the other network schedules for that time slot. The goal of each network is to win as many time slots as possible.

Suppose in the opening week of the fall season, Network $\mathcal{A}$ reveals a schedule S and Network $\mathcal{B}$ reveals a schedule T. On the basis of this pair of schedules, each network wins certain time slots, according to the rule above. We'll say that the pair of schedules (S, T) is *stable* if neither network can unilaterally change its own schedule and win more time slots. That is, there is no schedule S' such that Network A wins more slots with the pair (S', T) than it did with the pair (S, T); and symmetrically, there is no schedule T' such that Network B wins more slots with the pair (S, T') than it did with the pair (S, T).

The analogue of Gale and Shapley's question for this kind of stability is the following: For every set of TV shows and ratings, is there always a stable pair of schedules? Resolve this question by doing one of the following two things:

(a) give an algorithm that, for any set of TV shows and associated ratings, produces a stable pair of schedules; or

(b) give an example of a set of TV shows and associated ratings for which there is no stable pair of schedules.

(4 marks)

4. The greedy algorithm for Interval Scheduling that we described in the class is not the only greedy strategy we could have tried. For each of the following alternative greedy strategies, either prove that the resulting algorithm always constructs an optimal schedule, or describe a small example for which the algorithm does not produce an optimal schedule. Assume that all algorithms break ties arbitrarily (that is, in a manner that is completely out of your control).

   (a) Choose an interval x that *ends last*, discard intervals that conflict with x, and recurse.

   (b) Choose an interval x that *starts first*, discard all intervals that conflict with x, and recurse.

   (c) Choose an interval x that *starts last*, discard all intervals that conflict with x, and recurse.

   (d) Choose an interval x with *shortest duration*, discard all intervals that conflict with x, and recurse.

   (e) Choose an interval x that *conflicts with the fewest other intervals*, discard all intervals that conflict with x, and recurse.

   (f) If no intervals conflict, choose them all. Otherwise, discard an interval with the *longest duration* and recurse.

(g) If no intervals conflict, choose them all. Otherwise, discard an interval that conflicts *with the most other courses* and recurse.

(7 marks)

5. Let X be a set of n closed intervals on the real line. We say that a subset of intervals $Y \subseteq X$ *covers* X if the union of all intervals in Y is equal to the union of all intervals in X. The *size* of a cover is just the number of intervals.

   Give an efficient algorithm to compute the *smallest* cover of X. Assume that your input consists of two arrays $L[1, \ldots, n]$ and $R[1, \ldots, n]$, representing the left and right endpoints of the intervals in X. **(5 marks)**

6. For every integer $n \geq 2$,

   (a) Find frequencies for a set of n letters, say $\{a_1, a_2, \ldots, a_n\}$, whose Huffman code tree has depth $n - 1$.

   (b) Find frequencies for a set of n letters whose Huffman code tree has depth $n - 1$ and the largest frequency is as small as possible.

   (3 + 2 marks)

7. Suppose you are given an array $A[1, \ldots, n]$ of integers, each of which may be positive, negative, or zero. A contiguous subarray $A[i, \ldots, j]$ is called a **positive-interval** if the sum of its entries is greater than zero. Give an efficient algorithm to compute the minimum number of positive-intervals that cover every positive entry in A. For example, given the following array as input, your algorithm should output 3. If every entry in the input array is negative, your algorithm should output 0.

3	−5	7	−4	1	−8	3	−7	5	−9	5	−2	4

   The three positive-intervals in the above array are $A[1, 2, 3, 4, 5]$, $A[7, 8, 9]$, and $A[11, 12, 13]$. **(5 marks)**

8. Consider the following process. At all times you have a single positive integer x, which is initially equal to 1. In each step, you can either *increment* x or *double* x. Your goal is to produce a target value n. For example, you can produce the integer 10 in four steps as follows:

$$1 \xrightarrow{+1} 2 \xrightarrow{\times 2} 4 \xrightarrow{+1} 5 \xrightarrow{\times 2} 10.$$

   Obviously you can produce any integer n using exactly $n-1$ increments, but for almost all values of n, this is horribly inefficient. Give an efficient algorithm to compute the minimum number of steps required to produce any given integer n. **(5 marks)**

9. Consider the directed acyclic graph G show below (Figure 1). How many topological orderings does it have? **(3 marks)**

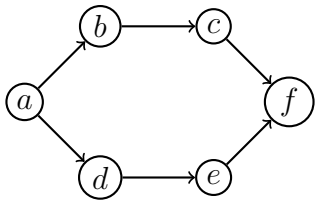

Figure 1: Directed acyclic graph G

10. Give an algorithm to detect whether a given undirected graph contains a cycle. If the graph contains a cycle, then your algorithm should output one of them. (It should not output all cycles in the graph, just one of them.) The running time of your algorithm should be $O(m+n)$ for a graph with n nodes and m edges. **(5 marks)**

11. The algorithm described in the class for computing a topological ordering of a DAG repeatedly finds a node with no incoming edges and deletes it. This will eventually produce a topological ordering, provided that the input graph really is a DAG.

    But suppose that we're given an arbitrary directed graph that may or may not be a DAG. Extend the topological ordering algorithm so that, given an input directed graph G, it outputs one of two things: (a) a topological ordering, thus establishing that G is a DAG; or (b) a directed cycle in G, thus establishing that G is not a DAG. The running time of your algorithm should be $O(m+n)$ for a directed graph with n nodes and m edges. **(5 marks)**

12. We have a connected undirected graph $G = (V, E)$, and a specific vertex $u \in V$. Suppose we compute a depth-first search tree rooted at u, and obtain a tree T that includes all nodes of G. Suppose we then compute a breadth-first search tree rooted at u, and obtain the same tree T. Prove that $G = T$. (In other words, if T is both a depth-first search tree and a breadth-first search tree rooted at u, then G cannot contain any edges that do not belong to T.) **(5 marks)**

13. Decide whether you think the following claim is true or false, and give a proof of either the claim or its negation.

    Claim: Let G be an undirected graph on n nodes, where n is an even number. If every node of G has degree at least $n/2$, then G is connected.

    (5 marks)

14. Suppose that an n-node undirected graph $G = (V, E)$ contains two nodes s and t such that the distance between s and t is strictly greater than $n/2$. Show that there must exist some node v, not equal to either s or t, such that deleting v from G destroys all s-t paths. (In other words, the graph obtained from G by deleting v contains no path from s to t.) Give an algorithm with running time $O(m+n)$ to find such a node v. **(5 marks)**

15. Suppose we are given an undirected graph $G = (V, E)$, and we identify two nodes s and t in G. Give an algorithm that computes the number of shortest s-t paths in G. (The algorithm should not list all the paths; just the number suffices.) The running time of your algorithm should be $O(m + n)$ for a graph with n nodes and m edges.

(5 marks)

16. You're helping a group of ethnographers analyze some oral history data they've collected by interviewing members of a village to learn about the lives of people who've lived there over the past two hundred years.

    From these interviews, they've learned about a set of n people (all of them now deceased), whom we'll denote $P_1, P_2, \ldots, P_n$. They've also collected facts about when these people lived relative to one another. Each fact has one of the following two forms:

    (a) For some i and j, person P_i died before person P_j was born; or

    (b) For some i and j, the life spans of P_i and P_j overlapped at least partially.

    Naturally, they're not sure that all these facts are correct; memories are not so good, and a lot of this was passed down by word of mouth. So what they'd like you to determine is whether the data they've collected is at least internally consistent, in the sense that there could have existed a set of people for which all the facts they've learned simultaneously hold.

    Give an efficient algorithm to do this: either it should report 'Consistent' if there could exist dates of birth and death for each of the n people so that all the facts hold true, or it should report 'Inconsistent' in case no such dates can exist—that is, the facts collected by the ethnographers are not internally consistent. **(5 marks)**

17. You're helping some security analysts monitor a collection of networked computers, tracking the spread of an online virus. There are n computers in the system, labeled $C_1, C_2, \ldots, C_n$, and as input you're given a collection of trace data indicating the times at which pairs of computers communicated. Thus the data is a sequence of ordered triples (C_i, C_j, t_k); such a triple indicates that C_i and C_j exchanged bits at time t_k. There are m triples total.

    We'll assume that the triples are presented to you in sorted order of time. For purposes of simplicity, we'll assume that each pair of computers communicates at most once during the interval you're observing.

    The security analysts you're working with would like to be able to answer questions of the following form: If the virus was inserted into computer C_a at time x, could it possibly have infected computer C_b by time y? The mechanics of infection are simple: if an infected computer C_i communicates with an uninfected computer C_j at time t_k (in other words, if one of the triples (C_i, C_j, t_k) or (C_j, C_i, t_k) appears in the trace data), then computer C_j becomes infected as well, starting at time t_k. Infection can thus

spread from one machine to another across a sequence of communications, provided that no step in this sequence involves a move backward in time. Thus, for example, if C_i is infected by time t_k, and the trace data contains triples (C_i, C_j, t_k) and (C_j, C_q, t_r), where $t_k \leq t_r$, then C_q will become infected via C_j. (Note that it is okay for t_k to be equal to t_r; this would mean that C_j had open connections to both C_i and C_q at the same time, and so a virus could move from C_i to C_q.)

For example, suppose $n = 4$, the trace data consists of the triples

$$(C_1, C_2, 4), (C_2, C_4, 8), (C_3, C_4, 8), (C_1, C_4, 12),$$

and the virus was inserted into computer C_1 at time 2. Then C_3 would be infected at time 8 by a sequence of three steps: first C_2 becomes infected at time 4, then C_4 gets the virus from C_2 at time 8, and then C_3 gets the virus from C_4 at time 8. On the other hand, if the trace data were

$$(C_2, C_3, 8), (C_1, C_4, 12), (C_1, C_2, 14),$$

and again the virus was inserted into computer C_1 at time 2, then C_3 would not become infected during the period of observation: although C_2 becomes infected at time 14, we see that C_3 only communicates with C_2 before C_2 was infected. There is no sequence of communications moving forward in time by which the virus could get from C_1 to C_3 in this second example.

Design an algorithm that answers questions of this type: given a collection of trace data, the algorithm should decide whether a virus introduced at computer C_a at time x could have infected computer C_b by time y. The algorithm should run in time $O(m+n)$.

(5 marks)