CSE 431/531 (Fall 2021) Practice Exam

Lecturer: Prof. Shi Li

Student's Name: _____ UBITName: _____

Problem	1	2	3	4	5	6	7	8	9	10	Total
Maximum Score	20	5	10	10	10	10	10	5	10	10	100
Your Score											

Problem 1 (20 Points). Indicate if each of the following sentences is true or false. You do not need to give proofs for your answers.

(1a) $(\log_2 n)^2 = \Omega\big(5\log_2(n^3)\big)$. True or False?

(1b) If the recurrence of a running time T is $T(n) = 3T(n/3) + O(n)$, then solving the recurrence using the master theorem gives us $T(n) = O(n)$. True or False?

(1c) Let $G = (V, E)$ be a graph with n vertices and m edges. Assume G is connected so $m \geq n-1$. Then the adjacency-matrix representation of G takes $O(m)$ space. True or False?

(1d) Let T be a DFS tree of a connected graph $G = (V, E)$. Let $(u, v) \in E$ such that (u, v) is not in T. Then it is possible that u is not an ancestor of v in T, and v is not an ancestor of u in T. True or False?

(1e) In the interval scheduling problem, it is safe to schedule the job with the latest starting time. True or False?

(1f) Let $G = (V, E)$ be a connected graph and $w : E \to \mathbb{R}_{>0}$ be a weight function on edges. Then the minimum spanning tree of G w.r.t the weights w is unique if and only if the weights of edges in E are distinct. True or False?

(1g) Suppose we have 6 stones of distinct weights. There is an unlabelled balanced scale which, given two stones, can tell which one is heavier. To sort the 6 stones from the lightest to the heaviest, we need to use at least 10 comparisons in the worst case. True or False?

(1h) The Bellman-Ford algorithm can solve the shortest simple path problem on directed graphs with negative weights. (In this problem, we are given a graph with edge weights, which could be negative, two vertices s and t, and the goal is to find the shortest simple path from s to t in G). True or False?

(1i) It is possible that P $\neq$ NP, but P = Co-NP. True or False?

(1j) Let X be a NP-Complete problem. Then P = NP if and only if there is a polynomial time algorithm for solving X. True or False?

Problem 2 (5 Points). Using the definition of the O-notation to prove $\lceil |(n+10)\sin(n)| \rceil = O(n^2)$.

Problem 3 (10 Points). In the class, you learned how to check if a graph is bipartite or not. In this problem, we assume the input graph $G = (V, E)$, which is given using the linked-list representation, is *not* bipartite. Design an $O(n+m)$-time algorithm that outputs an odd cycle in G. You only need to write down the pseduo-code for your algorithm.

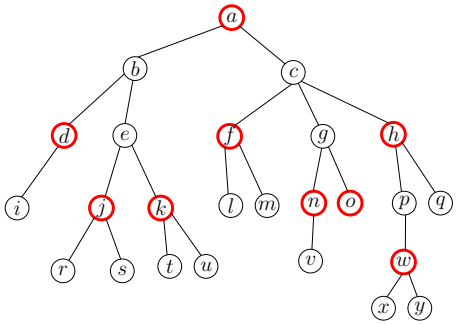

Figure 1: Dominating Set.

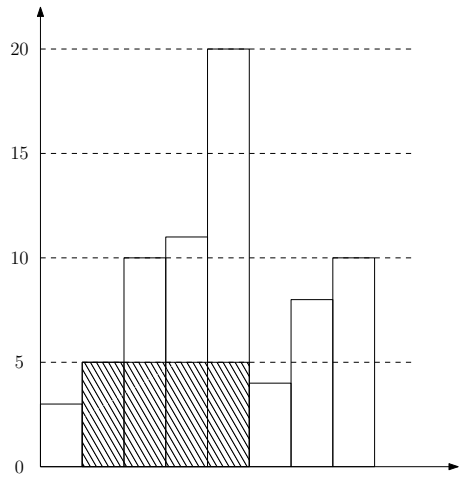

Figure 2: Largest Rectangle of Width k in a Histogram.

Problem 4 (10 Points). We consider the dominating set problem on trees. In this problem, we are given a tree $T = (V, E)$. A dominating set in T is a set $S \subseteq V$ of vertices such that for vertex u in T, either u is in S, or some neighbor of u is in S (or both). The problem asks for the smallest dominating set of T; that is, the dominating set S with the smallest $|S|$.

For example, in the tree T in Figure 1, the vertices denoted by thick red circles form a dominating set. It is the smallest. (There are many other dominating sets of size 9 as well.)

Design a greedy algorithm to solve the problem, following the steps below:

(4a) Given a tree T with at least two vertices, design a simple strategy to pick a vertex v^*, so that it is safe to include v^* in the dominating set.

(4b) Prove that it is indeed safe to include the v^* you choose in (4a) in the dominating set.

(4c) After we decided to include v^* in the dominating set, what is (are) the residual instance(s) we need to solve?

Problem 5 (10 Points). This problem asks for the highest rectangle of width k in a histogram for an array of n non-negative integers. For example, if the $n = 8$, and $k = 4$, and the array of size 8 is $(3, 5, 10, 11, 20, 4, 8, 10)$, then the highest rectangle of width 4 has height 5, spanning column 2 to column 5 (See Figure 2).

Design an $O(n \log n)$-time divide and conquer algorithm to solve the problem. You can use one of the following two techniques to solve the problem: (a) heap data structure, and (b) divide and conquer.

Problem 6 (10 Points). A palindrome is a string which reads the same backward or forward. For example, "racecar", "wasitacaroracatisaw", "putitup" are all palindromes.

In the longest palindrome subsequence problem, you are given a string A of length n, and you need to find the longest subsequence C of A that is a palindrome. Recall that a subsequence of A is a string formed by removing letters from A. For example, suppose A is acbcedeacab, then the

longest palindrome subsequence of A is acedeca, which has length 7. (The appearance of A in C is as follows: acbcedeacab.)

In this question, you only need to return the length of the longest palindrome subsequence of A. Design an efficient (i.e, polynomial-time) dynamic-programming algorithm to solve the problem, following the steps below:

(6a) Define the cells of the dynamic programming.

(6b) Show how to compute the cells using recursion.

(6c) Specify the order in which you compute the cells.

Problem 7 (10 Points). Consider the following job-selection problem. Suppose you can undertake one job in each of the following n weeks. The set of possible jobs is divided into those that are low-stress and that are high-stress. The basic question, each week, is whether to take on a low-stress job or a high-stress job.

If you select a low-stress job in week i, then you get a revenue of $l_i > 0$ dollars; if you select a high-stress job, you get a revenue of $h_i > 0$ dollars. The catch, however, is that in order for you to take on a high-stresss job in week i, it is required that you do no job (of either type) in week $i-1$. On the other hand, it's OK if you take a low-stress job in week i even if you have done a job (of either type) in week $i-1$.

So, given a sequence of n weeks, a plan is specified by a choice of "low", "high", or "none" for each of the n weeks, with the property that if "high" is chosen for week $i > 1$, then "none" has to be chosen for week $i-1$. (You can choose "high" for week 1.) The value of the plan is determined in the natural way: for each i, you add l_i to the value if you choose "low" in week i, and you add h_i to the value if you choose "high" in week i. (You add 0 if you choose "none".)

For example, suppose $n = 5$ and the h_i and l_i values are given in the following table:

i	1	2	3	4	5
h_i	50	20	20	10	60
l_i	30	15	10	5	50

Then the maximum revenue you can achieve is 135, achieved by the plan "high, low, low, none, high". In this plan, the revenue you get in the each of the 5 days are 50, 15, 10, 0 and 60.

Design an efficient (i.e, polynomial-time) dynamic-programming algorithm to solve the problem. For convenience you only need to output the maximum revenue you can achieve. Follow the steps below:

(7a) Define the cells of the dynamic programming.

(7b) Show how to compute the cells using recursion.

(7c) Specify the order in which you compute the cells.

Problem 8 (5 Points). Use Kruskal's algorithm to construct the minimum spanning tree of the graph in Figure 3, with edge weights indicated on their correspondent edges. You only need to output the set of edges in the minimimum spanning tree, in the order they are added into the tree, by filling the table to the right of the figure.

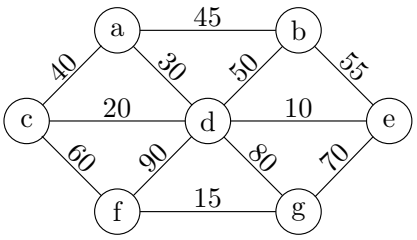

Figure 3: Instance for Kruskal's Algorithm.

i	1	2	3	4	5	6
i-th edge added in Kruskal's algorithm						

Problem 9 (10 Points). We are given a directed graph $G = (V, E)$ with positive weight function $w : E \to \mathbb{R}_{>0}$, and two vertices $s, t \in V$. Suppose we have already computed the d and π array using the Dijkastra's algorithm: $d[v]$ is the length of the shortest path from s to v, and $\pi[v]$ is the vertex before v in the path.

Given the graph $G = (V, E)$ (using the linked-list-representation), d and π, show how check if the shortest path from s to t in G is unique or not, in $O(n \log n + m)$ time. It suffices to give the pseudo-code of the algorithm.

Problem 10 (10 Points). For each of the following problems, state (1) whether the problem is known to be in NP, and (2) whether the problem is known to be in Co-NP. If your answer is yes, you should briefly describe the certifier and the certificate in the proof.

(10a) Given a graph $G = (V, E)$ and an integer $s \geq 1$, the problem asks whether there is an independent set of size s in the graph G or not.
Problem known to be in NP?_____ Problem known to be in Co-NP?_____

(10b) Given a tree $T = (V, E)$ and an integer $s \geq 1$, the problem asks whether there is an independent set of size s in the tree T or not.
Problem known to be in NP?_____ Problem known to be in Co-NP?_____

(10c) Given a connected graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}_{>0}$, and an integer $L \geq 0$, the problem asks whether there is a spanning tree of G with total weight at most L or not.
Problem known to be in NP?_____ Problem known to be in Co-NP?_____

(10d) A boolean formula is called a tautology if it evaluates to "true" for every assignment of "true/false" values to variables. For example, $x_1 \vee (\neg x_1 \wedge x_2) \vee (\neg\, x_1 \wedge x_3) \vee \neg(x_2 \vee x_3)$ is a tautology. Given a boolean formula with n variables, the problem asks whether it is a tautology or not.
Problem known to be in NP?_____ Problem known to be in Co-NP?_____