CS 344: Design and Analysis of Computer Algorithms **Rutgers: Spring 2022**

Practice Final Exam

Name: ______________________ *NetID:* ____________

Instructions

1. Do not forget to write your name and NetID above, and to sign Rutgers honor pledge below.
2. The exam contains 5 problems worth 100 points in total *plus* one extra credit problem worth 10 points.
3. The exam should be done **individually** and you are not allowed to discuss these questions with anyone else. This includes asking any questions or clarifications regarding the exam from other students or posting them publicly on Piazza (**any inquiry should be posted privately on Piazza**). You may however consult all the materials used in this course (video lectures, notes, textbook, etc.) while writing your solution, but **no other resources are allowed**.
4. Remember that you can leave a problem (or parts of it) entirely blank and receive 25% of the grade for that problem (or part). However, this should not discourage you from attempting a problem if you think you know how to approach it as you will receive partial credit more than 25% if you are on the right track. But keep in mind that if you simply do not know the answer, writing a very wrong answer may lead to 0% credit.

   The only **exception** to this rule is the extra credit problem: you do not get any credit for leaving the extra credit problem blank, and there is almost no partial credit on that problem.
5. **You should always prove the correctness of your algorithm and analyze its runtime.** Also, as a general rule, avoid using complicated pseudo-code and instead explain your algorithm in English.
6. You may use any algorithm presented in the class or homeworks as a building block for your solutions.

Rutgers honor pledge:

On my honor, I have neither received nor given any unauthorized assistance on this examination.

Signature: ______________________

Problem. #	Points	Score
1	20	
2	20	
3	20	
4	20	
5	20	
6	+10	
Total	$100 + 10$	

Problem 1.

(a) Mark each of the assertions below as True or False and provide a short justification for your answer.

(i) If $f(n) = n^{\log n}$ and $g(n) = 2^{\sqrt{n}}$, then $f(n) = \Omega(g(n))$. **(2.5 points)**

(ii) If $T(n) = 2T(n/2) + O(n^2)$, then $T(n) = O(n^2)$. **(2.5 points)**

(iii) If a problem in NP can be solved in polynomial time, then all problems in NP can be solved in polynomial time. **(2.5 points)**

(iv) If P = NP, then all NP-complete problems can be solved in polynomial time. **(2.5 points)**

(b) Prove the following statement: Consider a flow network G and a maximum flow f in G. There is always an edge e with $f(e) = c_e$, i.e., there is at least one edge that carries the same amount of flow as its capacity. **(10 points)**

Problem 2. You are given a two arrays $score[1 : n]$ and $wait[1 : n]$, each consisting of n positive integers. You are playing a game with the following rules:

- For any integer $1 \leq i \leq n$, you are allowed to pick number i and obtain $score[i]$ points;
- Whenever you pick a number i, you are no longer allowed to pick the previous $wait[i]$ numbers, i.e., any of the numbers $i - 1, \ldots, i - wait[i]$.

Design an $O(n)$ time dynamic programming algorithm to compute the maximum number of points you can obtain in this game. It is sufficient to write your specification and the recursive formula for computing the solution (i.e., you do *not* need to write the final algorithm using either memoization or bottom-up dynamic programming). **(20 points)**

Remember to *separately* write your specification of recursive formula in plain English (**7.5 points**), your recursive solution for the formula and its proof of correctness (**7.5 points**), and runtime analysis (**5 points**).

Problem 3. You are given a directed graph $G = (V, E)$ such that every *edge* is colored red, blue, or green. We say that a path P_1 is *lighter* than a path P_2 if one of the conditions below holds:

- P_1 has fewer number of red edges;
- P_1, P_2 have the same number of red edges and P_1 has fewer blue edges;
- P_1, P_2 have the same number of red and blue edges and P_1 has no more green edges than P_2.

Design an $O(n + m \log m)$ time algorithm that given G and vertices s, t, finds one the lightest paths from s to t in G, namely, a path which is lighter than any other s-t path. **(20 points)**

Remember to *separately* write your algorithm (**7.5 points**), the proof of correctness (**7.5 points**), and runtime analysis (**5 points**).

Problem 4. You are given an undirected *bipartite* graph G where V can be partitioned into $L \cup R$ and every edge in G is between a vertex in L and a vertex in R. For any integers $p, q \geq 1$, a (p,q)**-factor** in G is any subset of edges $M \subseteq E$ such that no vertex in L is shared in more than p edges of M and no vertex in R is shared in more than q edges of M. As an example, notice that a *matching* introduced in the class is a $(1,1)$-factor.

Design an $O((m+n) \cdot n \cdot (p+q))$ time algorithm that given a graph G and two integers p and q, outputs the size of the *largest* (p,q)-factor of G. **(20 points)**

Remember to *separately* write your algorithm (**7.5 points**), the proof of correctness (**7.5 points**), and runtime analysis (**5 points**).

Problem 5. Prove that the following problems are NP-hard. For each problem, you are only allowed to use a reduction from the problem specified.

(a) **Two-Third 3-SAT Problem:** Given a 3-CNF formula Φ (in which size of each clause is *at most* 3), is there an assignment to the variables that satisfies at least $2/3$ of the clauses? **(10 points)**

For this problem, use a reduction from the *3-SAT problem.* Recall that in the 3-SAT problem, we are given a 3-CNF formula Φ and the goal is to output whether there exist an assignment that satisfies *all* clauses.

(b) **Maximum Pyramid Problem:** Given an undirected graph $G = (V, E)$, what is the size of the largest **pyramid** in G? A pyramid is a set of vertices $v_1, \ldots, v_k$ plus another vertex u such that there are no edges between $v_1, \ldots, v_k$ in G and every vertex v_i is neighbor to u. **(10 points)**

For this problem, use a reduction from the *maximum independent set problem.* Recall that in the maximum independent set problem, you are given an undirected graph $G = (V, E)$ and the goal is to find the largest independent set in G, namely, the largest set of vertices with no edges between them.

Problem 6. [**Extra credit**] You are given a set C of p courses that a student has taken. You are also given q subsets $S_1, \ldots, S_q$ of C, and q integers $r_1, \ldots, r_q$. In order to graduate, the student has to meet the following q requirements:

- For every $1 \leq i \leq q$, the student needs to have taken at least r_i courses from the subset S_i.

The goal is to determine whether or not the courses taken by the student can be used to satisfy all q requirements. The tricky part is that any given course *cannot* be used towards satisfying multiple requirements.

For example, if one requirement states that the student should have taken 2 courses from $S_1 = \{A, B, C\}$ and another requirement asks for taking 2 courses from $S_2 = \{C, D, E\}$, then a student who had taken just $\{B, C, D\}$ *cannot* graduate.

Design a polynomial time algorithm that given a set C of p courses a student has taken, q subsets $S_1, \ldots, S_q$ of C, and q integers $r_1, \ldots, r_q$, determines whether or not the student can graduate.

Extra Workspace