Algorithms Qual Spring 2018

(provide proofs for all answers)

Problem 1: Dynamic Algorithm Design.
Suppose there you are input an array of n numbers (x_1, x_2,…,x_n). You are then input a sequence of commands each of which is either:
(a) An update of the form (i, v) indicating the value of x_i is updated to be v,
(b) A request for the sum of the values of the current array.
Describe a data structure which takes O(n) time to initially construct, and a dynamic algorithm which can execute each of these commands in O(log n) time.

Problem 2: Graph Algorithm Design.
You are given a complete graph on vertices V={1, 2,…, n} with edges of non-negative weights. For each subset of the vertices S, let P(S) be the monotonically increasing path that traverses all the vertices of S, and let W(S) be the total weight of the edges of P(S).

Give an $O(n^2)$ algorithm to partition the vertices V into two sets S_1 and S_2 that $W(S_1)+W(S_2)$ is minimized. In other words, find the partition of V that minimizes the sum of weights of the two monotonically increasing paths corresponding to the sets of the partition.

(For instance, if n = 5 and the partition is S_1 = {1, 3} and S_2 = {2, 4, 5}, then the edge (1, 3) forms one monotonically increasing path, and the edges (2, 4) and (4, 5) form the second monotonically increasing path.)

Problem 3: Greedy Algorithm Design.

You are driving to a city that is m miles away. The car can hold enough gas for t miles. There are n gas stations on the way and i-th gas station is at location x_i, and has a price of p_i. Give an efficient algorithm which given this information, finds a way to get to the destination with the minimum amount of money (the number of stops does not matter).

Hint: There is an O(n) greedy algorithm but it has two cases and is a bit tricky.

Problem 4: Optimization Algorithm Design.
You are given a sequence of n real numbers (some of which could be negative), and you need to add + and * signs in between consecutive numbers. Assume + is evaluated before *.
Give an efficient algorithm to find the largest number that you can get.

Problem 5: Probabilistic Analysis of Algorithms.

Suppose there are n seats of an airplane and n passengers for a flight, and the airline assigns a boarding pass to each of the n passengers specifying a distinct seat of the airplane. Also suppose the airline has the following boarding policy: when a passenger boards the airplane, she must occupy the seat on her boarding pass if it is empty, but if not empty, she must occupy a randomly chosen seat from the ones that are empty. Normally, this works out because every passenger takes her assigned seat. But, suppose the first passenger loses her boarding pass and occupies a randomly chosen seat. Then some number of passengers may not be seated in their assigned seat by the end of boarding.

What is the expected number of passengers who would not be seated in their assigned seat by the end of boarding?

Problem 6: Algorithms, NP-Hardness, and Approximation Algorithms.
There is a set of n politicians. Politician i makes m_i statements; politician i's j-th statement is denoted x_{ij}. Some pairs of statements are logically inconsistent; let $Inc(x_{ij}, x_{i'j'})$ denote that the j-th statement by politician i is inconsistent with the j'-th statement by politician i'. No two statements are the same. We try to make sense of it all.

A *hypothesis* assigns a truth value to every statement. A hypothesis is *feasible* if for every inconsistent pair, it labels at least one of the two hypotheses in that pair as false. That is, for a feasible hypothesis:

$Inc(x_{ij}, x_{i'j'})$ implies (NOT x_{ij}) OR (NOT $x_{i'j'}$).

For example, the hypothesis that all statements are false is always feasible. But we are more optimistic than that.

(a) Consider the problem of finding a feasible hypothesis that minimizes the number of statements that are labeled false. Can this problem be solved exactly in polynomial time? If so, give an efficient algorithm. If not, give a proof of NP-hardness, as well as a proof that there is a polynomial-time constant-factor approximation.

(b) Consider the problem of finding a feasible hypothesis that minimizes the number of lying politicians, i.e., that minimizes the number of politicians that have at least one of their statements labeled false. Can this problem be solved exactly in polynomial time? If so, give an efficient algorithm. If not, give a proof of NP-hardness, as well as a proof that there is a polynomial-time constant-factor approximation.