Problem Set 3

CMSC 27230: Honors Theory of Algorithms

Assigned January 23, Due January 30

Problem 1: Recurrence Relations (15 points, 5 extra credit points available)

Solve the following recurrence relations. For your answers to parts a, b, and c, give the general solution to the recurrence relation as well as the solution when $T(1) = 1$. For part d, give the general solution to the recurrence relation as well as the solution when $T(1) = 1$ and $T(2) = 2$. Show your work.

(a) 5 points: $T(n) = 8T(\frac{n}{4}) + 5n$

(b) 5 points: $T(n) = 4T(\frac{n}{2}) + 3n^2$

(c) 5 points: $T(n) = 2T(\frac{n}{8}) + 6n - 1$

(d) 5 points extra credit: $T(n) = 6T(\frac{n}{2}) - 8T(\frac{n}{4}) + n + 9log_2(n)$

Problem 2: Modifying Edge Lengths (Variation of Chapter 4, Problem 2 in Kleinberg-Tardos, 20 points)

Decide whether each of the following statements is true or false. If the statement is true, explain why it is true. If the statement is false, give a counterexample.

(a) 10 points: Let G be an undirected, connected graph where each edge $e \in E(G)$ has a length l_e and let T be a minimum spanning tree of G. If we modify the edge lengths of G by changing the length of each edge $e \in E(G)$ from l_e to $2l_e + 3$ then in the resulting graph, T will still be a minimum spanning tree.

(b) 10 points: Let G be an directed graph where each edge $e \in E(G)$ has a length l_e, let $s, t \in V(G)$ be vertices of G, and let P be a path of minimum length from s to t in G. If we modify the edge lengths of G by changing the length of each edge $e \in E(G)$ from l_e to $2l_e + 3$ then in the resulting graph, P will still be a path of minimum length from s to t.

Problem 3: Batch Testing (20 points, 5 extra credit points available)

Let's say that you are testing n samples $s_1, \ldots, s_n$ for a disease and you need to identify all of the samples which have the disease. You can test the samples as follows.

1. Choose a subset $S \subseteq \{s_1, \ldots, s_n\}$ of the samples, take a small portion of each sample, and mix them together.
2. Test if there the disease is present in the combined sample, i.e., is there at least one $s_j \in S$ such that s_j has the disease?

In this problem, we analyze the following question. If there are k samples which have the disease (where you don't know k beforehand and k could be 0), how many tests are needed to identify all of the samples which have the disease?

(a) 20 points: Give an algorithm for solving this problem using $O(klogn+1)$ tests, explain why your algorithm is correct, and explain why your algorithm only requires $O(klogn+1)$ tests.

(b) 5 points extra credit: Give a tighter analysis of how many tests are needed (in terms of n and k). For full extra credit, you should show both an upper bound and a lower bound and these bounds should match up to a constant factor.

Hint is available.

Problem 4: Finding the Extra (20 points)

Consider the following problem. Given a sorted array A with $n+1$ distinct elements and a sorted array B with n distinct elements, find an element which is in A but is not in B (if there are multiple such elements, you just need to return one of them.) Give an $O(log(n))$ time algorithm to solve

A	2	5	6	8	11	15	18	20	21	24	30
B	2	5	6	8	11	15	18	21	24	30	

Figure 1: In this example, the extra element of A is 20.

this problem, explain why it is correct (you do not need to give a rigorous proof), and explain why it takes time $O(logn)$.

Note: Some (but not much) partial credit will be given if you provide a polynomial time algorithm which is significantly slower than $O(logn)$ and analyze its running time.

Problem 5: Nearly Undefeated Teams (25 points)

n teams are playing in a tournament T where each team plays every other team exactly once and there are no ties. Now consider the following problem. Which teams in the tournament were defeated at most k times?

Give an $O(kn)$ time algorithm to solve this problem, prove that your algorithm is correct, and explain why it has the claimed running time. Partial credit will be given for an $O(knlogn)$ algorithm. Hint is available.

Note: For this problem, the tournament is given to you as a two dimensional $n \times n$ array T where $T_{ij} = 1$ and $T_{ji} = 0$ if team i beats team j and $T_{ji} = 1$ and $T_{ij} = 0$ if team j beats team i (and the diagonal entries T_{ii} are 0).

Acknowledgement: This problem is due to Professor Andy Drucker.

Hints

3. In order to analyze the runtime for your algorithm, it may be useful to write a recurrence relation involving both n and k. Here it is sufficient to give an upper bound on the solution to the recurrence relation rather than solving it exactly.

5. It may be helpful to show that if $2k + 2$ teams all play each other, at least one of these teams must be defeated at least $k + 1$ times.