Some Practice Problems for Final

Problem 1. Derive an asymptotically tight bound for the following recurrence. Assume that $T(n)$ is $\Theta(1)$ for $n \leq 4$.

$$T(n) = T(n/3) + T(2n/3) + n^2.$$

Problem 2. We are given an array A with n distinct integers, which is partially sorted; each element in A is within distance k of its position in the sorted order (increasing order). Formally, the input array A satisfies the following condition: For $i \in [1, n]$, the index of the element $A[i]$ in *sorted* order is in the range $[\max\{1, i - k\}, \min\{n, i + k\}]$.

(a) Suppose we know k. Design an $O(n \lg k)$ time algorithm to solve the problem. Briefly justify the worst-case running time of your algorithm.

(b) Now suppose we *do not know* k. Using the algorithm of part (a) as a black box, design an algorithm that takes $O(n \lg^2 k)$ time to sort the array.

Problem 3. We are given a set S of n activites that we would like to schedule among two lecture halls. For each activity, we have a start time s_i and a finish time f_i. Two activities that overlap in time may not be scheduled in the same lecture hall; they may, however, be scheduled separately in the two lecture halls.

Design a greedy algorithm to select a maximum-size set of activities from S that may be scheduled among the two lecture halls. Briefly justify the correctness of your algorithm.

Problem 4. Given a directed graph $G = (V, E)$ with positi ve weights on each edge, design an efficient algorithm to determine a directed cycle in G of minimum total weight. If no directed cycle exists in G, then your algorithm must indicate so.

Analyze the worst-case running time of your algorithm. The more efficient your algorithm is in terms of its worst-case running time, the more credit you will get.

Problem 5. Let $G = (V, E)$ be a directed graph in which each vertex has been assigned a color, either red or blue. A directed path in G is called an *alternating red-blue path* if and only if no two consecutive vertices on the path have the same color. Give an efficient algorithm that determines for *all* pairs of vertices u, v in V whether v is reachable from u via an alternating red-blue path.

Analyze the worst-case running time of your algorithm. The more efficient your algorithm is in terms of its worst-case running time, the more credit you will get.

Problem 6. Give an efficient algorithm to determine the longest common subsequence of three given sequences of length m, n, and p, respectively. Analyze the worst-case running time of your algorithm.

Problem 7. A set of n clients has jobs that may be assigned to a set of m servers. Each client i has j_i jobs, each job being of the same size, and an associated set S_i of servers on which any of these jobs may be executed. Each server i can service at most s_i jobs. We would like to determine an assignment of jobs to servers such that the total number of jobs serviced is maximized. The assignment needs to only specify how many jobs of client i is assigned to server j, for all i and j. (Note that the j_i jobs of client i need not be assigned to the same server from S_i; they may be distributed among multiple servers.)

- Write an integer linear program to solve the above optimization problem.
- Give a polynomial-time algorithm for solving the above problem using network flows.

Problem 8. Given a directed flow network $G = (V, E)$ and a maximum flow f, determine an $O(V + E)$-time algorithm to update the flow when the capacity of an edge decreases by 1. Assume that the flow f is specified by providing the value $f(u, v)$ for each edge (u, v). Also assume that all capacities are integers.

Problem 9. Clustering of similar documents is an important technique often used in information retrieval applications. It is common to define "similarity" by means of a distance function. Let S denote a set of documents. For any two documents X and Y, let $d(X, Y)$ denote the distance between them. (We will assume that $d(X, Y)$ is a nonnegative integer.) The smaller the distance, the more similar the two documents are.

Consider the LargestCluster problem: given a set S of documents, a distance function d, and an integer r, determine the largest subset T of S such that $d(X, Y) \leq r$ for any two documents X and Y in T.

Formulate a decision version of LargestCluster and prove that it is NP-complete. (*Hint:* You may use a reduction from the decision version of the Clique problem.)