CSE421: Design and Analysis of Algorithms Feb 1, 2018

Homework 5

Shayan Oveis Gharan Due: Feb 8, 2018 at 5:00 PM

Please see https://courses.cs.washington.edu/courses/cse421/18wi/grading.html for general guidelines about Homework problems.

Most of the problems only require one or two key ideas for their solution. It will help you a lot to spell out these main ideas so that you can get most of the credit for a problem even if you err on the finer details. Please justify all answers.

P1) Disprove the following claim: Suppose we are given an instance of the shortest s-t path problem on a directed graph G and assume that all edge weights are positive (and distinct). That is we want to find a minimum cost path from s to t. Let P be a minimum cost s-t path for this instance. Now, suppose we replace each edge weight c_e by its square c_e^2 thereby creating a new instance of the problem over the same graph but different costs. Then, P is still a minimum cost s-t path for this new instance.

P2) A small business - say, a photocopying service with a single large machine - faces the following scheduling problem: Each morning they get a set of jobs from customers. They want to do the jobs on their single machine in an order that keeps their customers happiest. Customer i's job will take t_i time to complete. Given a schedule (i.e., an ordering of jobs) let C_i denote the finishing time of job i. For example, if job j is the first to be done, we would have $C_j = t_j$; and if job j is done right after job i, we would have $C_j = C_i + t_j$. Each customer i also has a given weight w_i that represents his or her importance to the business. So, the company wants to order the jobs to minimize weighted sum of completion times, $\sum_{i=1}^{n} w_i C_i$.

a) Consider an ordering of jobs such that job j is done right after job i. Show that if $t_j/w_j < t_i/w_i$, then we can decrease the weighted sum of completion times by *exchanging* i, j.

b) Design a polynomial time algorithm that orders the jobs so as to minimize the weighted sum of the completion times (for simplicity, assume that for any two jobs i, j: $t_i/w_i \neq t_j/w_j$).

P3) Suppose you are choosing between the following three algorithms:

a) Algorithm A solves the problem by dividing it into five subproblems of half the size, recursively solves each subproblem, and then combines the solution in linear time.

b) Algorithm B solves problems of size n by recursively solving two subproblems of size $n-1$, and then combines the solution in constant time.

c) Algorithm C solves the problem by dividing it into nine subproblems of one third the size, recursively solves each subproblem, and then combines the solutions in quadratic time.

What are the running times of each of these algorithms? To receive full credit, it is enough to write down the running time.

P4) You are given two sorted lists of numbers of length m, n. Give an algorithm that finds the k'th smallest number in the union of the lists, in time $O(\log m + \log n)$ (for simplity you can assume that all numbers in the input are distinct).

P5) **Extra Credit** The spanning tree game is a 2-player game. Each player in turn selects an edge. Player 1 starts by deleting an edge, and then player 2 xes an edge (which has not been deleted yet); an edge xed cannot be deleted later on by the other player. Player 2 wins if he succeeds in constructing a spanning tree of the graph; otherwise, player 1 wins.

The question is which graphs admit a winning strategy for player 1 (no matter what the other player does), and which admit a winning strategy for player 2.

Show that player 1 has a winning strategy if and only if G does not have two edge-disjoint spanning trees. Otherwise, player 2 has a winning strategy.