CSE421: Design and Analysis of Algorithms	April 17, 2024

Homework 4

Shayan Oveis Gharan Due: April 24th, 2024 at 23:59 PM

P1) (10 points) Construct an example to show that the following Greedy algorithm for the interval partitioning problem can allocate more than depth many classrooms (so it is not optimum): Sort the lectures based on their **finishing time**. When considering the next job, allocate it to the available classroom with the smallest index. If no classroom is available, allocate a new classroom.

P2) Given a sequence of n real numbers $a_1, \ldots, a_n$ where n is even, design a polynomial time algorithm to partition these numbers into $n/2$ pairs in the following way: For each pair we compute the sum of its numbers. Denote $s_1, ..., s_{n/2}$ these $n/2$ sums. Your algorithm should find the partition which minimizes the maximum sum. For example, given numbers $3, -1, 2, 7$ you should output the following partition: $\{3, 2\}, \{-1, 7\}$. In such a case the maximum sum is $7 + (-1) = 6$.

P3) (20points) Given two edge disjoint spanning trees T_1, T_2 on n vertices, prove that for every edge $e \in T_1$ there exists an edge $f \in T_2$ that satisfies both of the following criteria:

- $T_1 - e + f$ is a spanning tree (on n vertices).
- $T_2 - f + e$ is a spanning tree (on n vertices).

P4) (10 points) Suppose you are choosing between the following three algorithms:

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

b) Algorithm B solves the problem by dividing it into twenty five subproblems of one fifth the size, recursively solves each subproblem, and then combines the solutions in quadratic time.

c) Algorithm C solves problems of size n by recursively solving four subproblems of size $n-4$, and then combines the solution in constant time.

In all cases you can assume it takes $O(1)$ time to solve instances of size 1. What are the running times of each of these algorithms? To receive full credit, it is enough to write down the running time.

P5) Given n integers $a_1, \ldots, a_n$ and two integers $\ell < u$, design an algorithm that runs in time $O(n \log^c n)$ (for any constant integer $c \geq 0$) and returns the number of interval-sums that lie in the interval $[\ell, u]$ inclusive. For example it is ok if your algorithms runs in $O(n \log^{10} n)$.

Interval sum $I(i, j) = a_i + a_{i+1} + \cdots + a_j$ is defined as the sum of the numbers $a_i, \ldots, a_j$ where $1 \leq i \leq j \leq n$.

For example, suppose $n = 3$ and we are given the sequence $-2, 5, -1$ with $\ell = -2, u = 2$. Then, you should output 3. This corresponds to three interval sums $I(1, 1), I(3, 3)$, and $I(1, 3)$ with their respective sums: $-2, -1, 2$.

P6) **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 fixes an edge (which has not been deleted yet); an edge fixed 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.