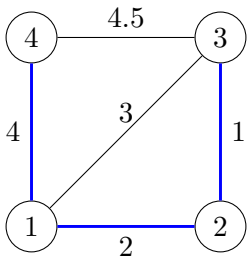
CSE421: Design and Analysis of Algorithms April 22, 2021

Homework 4

Shayan Oveis Gharan Due: April 29th, 2021 at 23:59 PM

P1) (20 points) In class we discussed an algorithm for interval schedule problem where we sorted the jobs based on their finishing times and we greedily scheduled the largest possible set of compatible jobs. Now, suppose that we sort the jobs based on their length in ascending order. Construct an example where Greedy algorithm does not give the maximum number of jobs to schedule. Justify your example by analyzing greedy and optimum on the example.

P2) (20 points) You are given a connected undirected weighted graph $G = (V, E)$ with n vertices and m edges, their weights, w_e for all edges $e \in E$, a minimum spanning tree $T \subseteq E$ together with an edge $f \in E$ and a new weight w'_f for f, design an algorithm that runs in time $O(n+m)$ to test if T still remains the minimum spanning tree of the graph if we change the weight of f to w'_f. Your algorithm should output "yes" if T is still the MST and "no" otherwise. Note that the edge f may or may not belong to T and we may decrease or increase weight of f. For simplicity, assume that all edge weights are distinct both before and after the update. For example in the following picture the MST is colored in blue. If we update the weight of the edge $(3, 1)$ to 1.5 the blue tree is no longer the MST.



P3) (10 points) Suppose you are choosing between the following three algorithms:

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

b) Algorithm B solves the problem by dividing it into sixteen subproblems of one fourth 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 three subproblems of size $n-5$, and then combines the solution in constant 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) (20 points) Given n distinct numbers $a_1, \ldots, a_n$ we say a violation has occurred if a smaller number comes after a larger number. Design an $O(n \log n)$ time algorithm that outputs the

number of violations. For example, given the sequence $2, 5, 3, 1, 4$ it has 5 violations because 3 comes after 5, 1 comes afters $2, 5, 3$ and 4 comes after 5.

If you design an algorithm that runs in $O(n \log^c n)$ for some arbitrary constant $c > 1$ you receive 18 points.

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 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.