Problem Set 9

Data Structures and Algorithms, Fall 2020

Due: November 26, in class.

Problem 1

Suppose that all edge weights in a graph are integers in the range from 1 to $|V|$. How fast can you make Kruskal's algorithm run? What if the edge weights are integers in the range from 1 to W for some constant W? You need to describe your algorithms and analyze their time complexities.

Problem 2

You are given a graph $G = (V, E)$ with positive edge weights, and an MST $T = (V, E')$ with respect to these weights. You may assume G and T are given as adjacency lists. Now suppose the weight of a particular edge $e \in E$ is modified from $w(e)$ to a new value $\hat{w}(e)$. You wish to quickly update the MST T to reflect this change, without recomputing the entire tree from scratch. There are four cases. In each case give an $O(|V| + |E|)$ time algorithm for updating the tree.

(a) $e \notin E'$ and $\hat{w}(e) > w(e)$.

(b) $e \notin E'$ and $\hat{w}(e) < w(e)$.

(c) $e \in E'$ and $\hat{w}(e) < w(e)$.

(d) $e \in E'$ and $\hat{w}(e) > w(e)$. *[For this part, you will get full credit if you can find an $O((|V| + |E|) \lg |V|)$ time algorithm; and you will get bonus credit for finding an $O(|V| + |E|)$ time algorithm.]*

Problem 3

Not just any greedy approach to the activity-selection problem produces a maximum-size set of mutually compatible activities. Give an example to show that the approach of selecting the activity of least duration from among those that are compatible with previously selected activities does not work. Do the same for the approaches of always selecting the compatible activity that overlaps the fewest other remaining activities and always selecting the compatible remaining activity with the earliest start time.

Problem 4

Suppose that in a 0-1 knapsack problem, the order of the items when sorted by increasing weight is the same as their order when sorted by decreasing value. Give an efficient greedy algorithm to find an optimal solution to this variant of the knapsack problem, and prove that your algorithm is correct.

Problem 5

Suppose that a data file contains a sequence of 8-bit characters such that all 256 characters are about equally common: the maximum character frequency is less than twice the minimum character frequency. Prove that Huffman coding in this case is no more efficient than using an ordinary 8-bit fixed-length code.

Problem 6

Let X be a set of n intervals on the real line. A proper coloring of X assigns a color to each interval, so that any two overlapping intervals are assigned different colors. Devise a greedy algorithm to compute the minimum number of colors needed to properly color X. Assume that your input consists of two arrays $L[1 \cdots n]$ and $R[1 \cdots n]$, representing the left and right endpoints of the intervals in X. You need to argue the correctness of your algorithm and analyze its time complexity.

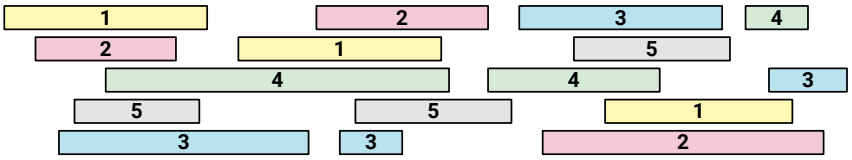

Figure 1: A proper coloring of a set of intervals using five colors.

Problem 7

One day Alex got tired of climbing in a gym and decided to take a large group of climber friends outside to climb. They went to a climbing area with a huge wide boulder, not very tall, with several marked hand and foot holds. Alex quickly determined an "allowed" set of moves that her group of friends can perform to get from one hold to another.

The overall system of holds can be described by a rooted tree T with n vertices, where each vertex corresponds to a hold and each edge corresponds to an allowed move between holds. The climbing paths converge as they go up the boulder, leading to a unique hold at the summit, represented by the root of T.

Alex and her friends decided to play a game, where as many climbers as possible are simultaneously on the boulder and each climber needs to perform a sequence of exactly k moves. Each climber can choose an arbitrary hold to start from, and all moves must move away from the ground. Thus, each climber traces out a path of k edges in the tree T, all directed toward the root. However, no two climbers are allowed to touch the same hold; the paths followed by different climbers cannot intersect at all.

Describe a greedy algorithm to compute the maximum number of climbers that can play this game, and analyze the correctness and time complexity of your algorithm. Your algorithm is given a rooted tree T and an integer k as input, and it should compute the largest possible number of disjoint paths in T, where each path has length k. Do not assume that T is a binary tree. For example, given the tree below as input, your algorithm should return the integer 8.

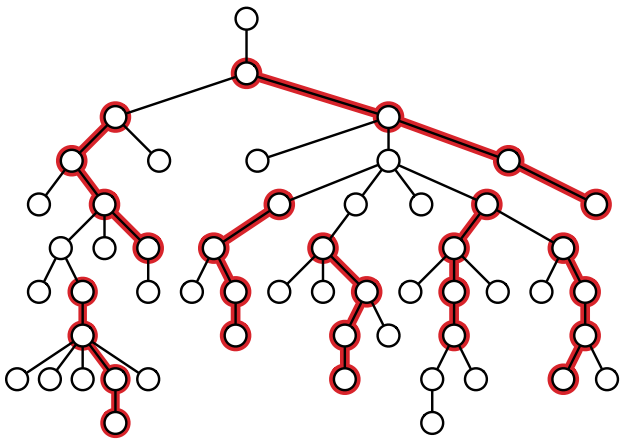

Figure 2: Seven disjoint paths of length $k = 3$. This is *not* the largest such set of paths in this tree.