CS 430 Introduction to Algorithms **Spring Semester, 2021**

First Extra Homework for Remote Students Version 1.2

Assigned: Feb. 22 *Due: March 15*

.

Please respect the following guidelines for writing pseudocode:

1. C/Java/Python instructions are fine. But do not write object-oriented additions. Do not declare or use any class. Declare only procedures (if necessary) and explain in words what each procedure does, and what is the use of each parameter.
2. One instruction per line
3. Match the brackets with a horizontal line
4. Number your lines
5. Write down if your array is indexed $0 \ldots n-1$ or $1 \ldots n$.
6. If you decide to use procedures/functions/methods from the class (such as, say, DELETE-MAX(S) for S a max-heap), explain clearly what they do and in what running time. Explain the data structures, if any.

Problem 1 Augment the **stack** data structure to do in constant time, besides push/insert and pop/remove, findmin. Describe the data structure, and present pseudocode for push/insert, pop/remove, and findmin. Argue that each operation takes constant time. Argue that findmin correctly returns the element with minimum key.

Problem 2 Suppose we are given a n-element sequence S such that each element in S represents a different vote in an election, where each vote is given as an integer representing the ID of the chosen candidate (these IDs come from the range $1 \ldots n^5$). Suppose you know the number $k < n$ of candidates running. Describe a $O(n \log k)$ worst-case algorithm for determining who wins the election. (Thus hashing will not work, as it does not have a good enough worst-case guarantee). The space requirement is $O(n)$, so a table of size n^5 cannot be used.

Argue that your running time is correct. Note that k could be much smaller than n and $\Theta(n \log n)$ is only a partial solution.

Problem 3 You are given a tree represented by the following data structure. Nodes have three fields: element, parent, and auxiliary. The auxiliary field is 0 and is there to help during traversals.

In this problem your algorithm should leave the tree unchanged. You are given the location (or pointer to) two nodes x and y.

The goal is to print the elements on the path from x to y in tree, in time proportional to the length of the path. That is, if the tree has n nodes and this path has length k, the running time of the algorithm should be $O(k)$ ($O(n)$ is not good enough, and the tree does not have to be balanced). See Figure 1 for an example, where the path from x to y is $\{x, l, f, c, y\}$ and has length $k = 4$.

Argue that your algorithm algorithm indeed has running time $O(k)$. Don't forget to bring back the tree to its original state. Present pseudocode.

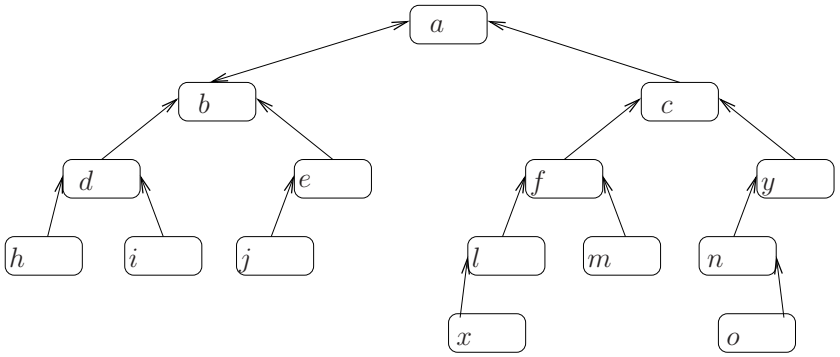

Figure 1: Example for Problem 3.

Problem 4 We are given two arrays of size n A and B, of integers, Consider the n closed intervals $[A[i], B[i]]$, which may not be disjoint. The union of these n intervals is another set of intervals, which can be made disjoint and maximal. Present an $O(n \log n)$ algorithm to output this new set of disjoint and maximal intervals into arrays C and D. Present pseudocode and analyze the running time.

Two sets are *disjoint* if they have no common elements. Consider, for example, A={6,1,4,9,12} and B={9,3,7,11,14}. The intervals are [6,9], [1,3], [4,7], and [9,11], and [12,14]. Here [6,9] and [4,7] are not disjoint. Also, with closed intervals, 9 belongs to both [4,9] and [9,11], and thus [4,9] and [9,11] are not disjoint. Then, one of the solutions can be C={4,1,12} and

D={11,3,14}, which represent disjoint and maximal intervals. Another valid solution would be C={1,4,12} and D={3,11,14}.

Problem 5 Suppose we are given two sorted arrays S and T, each with n items. Describe an $O(\log n)$-time algorithm for finding the kth smallest key in the union of the keys from S and T (assuming no duplicates). Present pseudocode, argue correctness and analyze the running time.

Problem 6 Suppose that you have a "black-box" worst-case linear-time median subroutine. (the median of n numbers is the $\lceil n/2 \rceil^{th}$ smallest element, i.e. the median of $\{1, 2, 4, 7, 11, 94\}$ is 4).

Give a simple, linear-time algorithm that, given an array $A[1..n]$ and a positive integer $i \leq n$ finds the i^{th} smallest element of A. Present pseudocode using procedures from the textbook or notes; give complete specifications and state the running time of the procedure in terms of its parameters.