

CIS606 ANALYSIS OF ALGORITHMS

Fall Semester, 2021

Assignment 3: Prune and Search

Due Date: 11:59 p.m., Sunday, Nov. 7, 2020

1. **(20 points)** Suppose you are given an array A with n entries, with each entry holding a distinct number (i.e., no two numbers of A are equal). You are told that the sequence of values $A[1], A[2], \dots, A[n]$ is *down-up ordered*: For some index p between 1 and n , the values in the array entries decrease down to position p in A and then increase the remainder of the way until position n . (So if you were to draw a plot with the array position j on the x -axis and the value of the entry $A[j]$ on the y -axis, the plotted points would decline until x -value p , where they'd achieve their minimum, and then rise from there on.)

You'd like to find the "peak entry" p without having to read the entire array – in fact, by reading as few entries of A as possible. Show how to find the entry p by reading at most $O(\log n)$ entries of A . In other words, design an $O(\log n)$ time algorithm to find the peak entry p .

For example, let $A = \{12, 11, 8, 6, 1, 10, 13, 19, 27\}$, which is be a down-up array. The peak entry of A is 1. So the output of your algorithm is either 1 or the index of 1 in A .

2. **(20 points)** Suppose you are consulting for site location of an emergency exit of a shopping street. There are n stores, represented by n points p_1, p_2, \dots, p_n on the line. Let the line be x -axis. We are given the x -coordinate of the n stores $p_i = (x_i, 0)$ for $i = 1, 2, \dots, n$. Note that the stores are not given in any sorted order. Our goal is to pick an optimal location for the emergency exit (i.e., find the x -coordinate of the exit) such that the **total sum of the distances** of stores to the exit is minimized. For simplicity, we assume no two stores have the same x -coordinate.

Design an $O(n)$ time algorithm to compute an optimal location for the emergency exit.

3. **(30 points)** Here is a generalized version of the selection problem, called *multiple selection*. Let $A[1 \dots n]$ be an array of n numbers. Given a sequence of m sorted integers k_1, k_2, \dots, k_m , with $1 \leq k_1 < k_2 < \dots < k_m \leq n$, the *multiple selection problem* is to find the k_i -th smallest number in A for all $i = 1, 2, \dots, m$. For simplicity, we assume that no two numbers of A are equal.

For example, let $A = \{1, 5, 9, 3, 7, 12, 15, 8, 21\}$, and $m = 3$ with $k_1 = 2$, $k_2 = 5$, and $k_3 = 7$. Hence, the goal is to find the 2nd, the 5-th, and the 7-th smallest numbers of A , which are 3, 8, and 12, respectively.

- (a) Design an $O(n \log n)$ time algorithm for the problem.

(5 points)

- (b) Design an $O(nm)$ time algorithm for the problem. Note that this is better than the $O(n \log n)$ time algorithm if $m < \log n$. **(5 points)**
- (c) Improve your algorithm to $O(n \log m)$ time, which is better than both the $O(n \log n)$ time and the $O(nm)$ time algorithms. **(20 points)**

Total Points: 70