CS5050 Advanced Algorithms

Fall 2023

Assignment 3: Algorithm Analysis

Due Date: 11:59:59 p.m., *Tuesday, October 17, 2023*

Total Points: 90 + (15 bonus)

1. **(20 points)** Suppose you are given an array $A[1 \cdots n]$ 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 $A[1], A[2], \ldots, A[n]$ is *unimodal:* For some index p between 1 and n, the values in the array entries increase up to position p in A and then decrease the remainder of the way until position n. (So if you were to draw a plot with the array position i on the x-axis and the value of the entry $A[i]$ on the y-axis, the plotted points would rise until x-value p, where they would achieve their maximum, and then fall from there on.)

   You would 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 $A = \{1, 4, 6, 8, 11, 12, 10, 9, 7\}$, which is be a unimodal array. The peak entry of A is 12. So the output of your algorithm may be either 12 or the index of 12 in A.

2. **(20 points)** In the SELECTION algorithm we studied in class, the input numbers are divided into groups of five. Will the algorithm still work in linear time if they are divided into groups of seven? Please justify your answer (i.e., explain why your answer is correct).

3. **(20 points)** Suppose you are consulting for an oil company, which is planning a large pipeline (called the *main pipeline*) running horizontally from east to west through an oil field of n wells. From each well, a *spur pipeline* is to be connected directly to the main pipeline along a shortest path (going to either the north or the south), as shown in Figure 1.

   Suppose there are n wells, represented by n points $p_1, p_2, \ldots, p_n$ in the plane. We are given the x- and y-coordinates of the n wells $p_i = (x_i, y_i)$ for $i = 1, 2, \ldots, n$. Note that the wells are not given in any sorted order. Our goal is to pick an optimal location for the main pipeline (i.e., find the y-coordinate of the main pipeline) such that the **total sum of the lengths** of the spur pipelines is minimized. For simplicity, we assume that no two wells have the same x-coordinate or y-coordinate.

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

4. **(30 points)** Here is a generalized version of the selection problem, called *multiple selection.* Let $A[1 \cdots n]$ be an array of n numbers. Given a sequence of m sorted integers $k_1, k_2, \ldots, k_m$, with $1 \leq k_1 < k_2 < \cdots < k_m \leq n$, the *multiple selection problem* is to find the k_i-th smallest number in A for all $i = 1, 2, \ldots, m$. 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)**

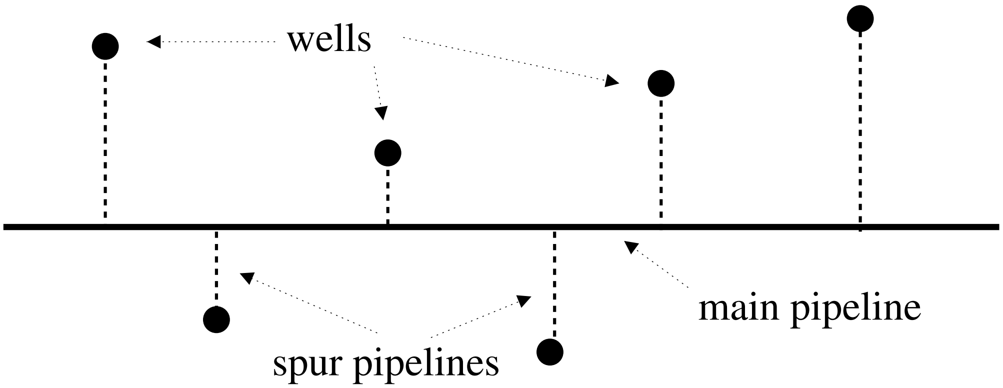

Figure 1: Illustrating Problem 3: the horizontal solid line is the main pipeline and the dashed vertical segments are the spur pipelines.

(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)**

5. (**Bonus**) Implement the SELECTION algorithm in Python, the one with the order of O(n), including the sophisticated pivot selection strategy. Your implementation should take an array A and k as the input and return k-th smallest element. A should be a general array of real numbers. A can have duplicates too. **(15 points)**. Test your algorithm's running time with various sizes of A and different ks.