Problem Set 5

Data Structures and Algorithms, Fall 2021

Due: October 21, in class.

Problem 1

We say that an array $A[1 \cdots n]$ is *k-sorted* if it can be divided into k blocks, each of size n/k, such that the elements in each block are larger than the elements in earlier blocks, and smaller than elements in later blocks. (In this problem, you may assume n/k is always an integer.) The elements within each block need not be sorted. For example, the following array is 4-sorted:

1	2	4	3	7	6	8	5	10	11	9	12	15	13	16	14

(a) Describe an algorithm that k-sorts an arbitrary array in $O(n \log k)$ time. You need to briefly argue the correctness of your algorithm, and show it indeed has time complexity $O(n \log k)$.

(b) Prove that any comparison-based k-sorting algorithm requires $\Omega(n \log k)$ comparisons in the worst case.

Problem 2

You are applying to participate in this year's Trial of the Pyx, the annual ceremony where samples of all British coinage are tested, to ensure that they conform as strictly as possible to legal standards. As a test of your qualifications, your interviewer at the Worshipful Company of Goldsmiths has given you a bag of n commemorative Alan Turing half-guinea coins, exactly two of which are counterfeit. One counterfeit coin is very slightly lighter than a genuine Turing; the other is very slightly heavier. Together, the two counterfeit coins have *exactly* the same weight as two genuine coins. Your task is to identify the two counterfeit coins.

The weight difference between the real and fake coins is too small to be detected by anything other than the Royal Pyx Coin Balance. You can place any two disjoint sets of coins in each of the Balance's two pans; the Balance will then indicate which of the two subsets has larger total weight, or that the two subsets have the same total weight. Unfortunately, each use of the Balance requires completing a complicated authorization form (in triplicate), submitting a blood sample, and scheduling the Royal Bugle Corps, so you really want to use the Balance as few times as possible.

(a) Suppose you randomly choose $n/2$ of your n coins to put on one pan of the Balance, and put the remaining $n/2$ coins on the other pan. What is the probability that the two subsets have equal weight?

(b) Describe and analyze a randomized algorithm to identify the two fake coins. What is the *exact* expected number of times your algorithm uses the Balance? (You may assume n is some power of 2.[1])

[1]Can you solve this problem if n is an arbitrary integer that is at least two?

Problem 3

Suppose we are given a set S of n items, each with a value and a weight. For any element $x \in S$, we define two subsets:

- $S_{<x}$ is the set of elements of S whose value is less than the value of x.
- $S_{>x}$ is the set of elements of S whose value is more than the value of x.

For any subset $R \subseteq S$, let $w(R)$ denote the sum of the weights of elements in R. The *magical-mean* of S is any element x such that $w(S_{<x}) \leq w(S)/2$ and $w(S_{>x}) \leq w(S)/2$.

In this problem, your input consists of two unsorted arrays $S[1 \cdots n]$ and $W[1 \cdots n]$, where for each index i, the i^{th} element has value $S[i]$ and weight $W[i]$. You may assume that all values are distinct and all weights are positive.

(a) Describe and analyze an algorithm to compute the magical-mean of a weighted set in $O(n \log n)$ time. *(Hint: Use sorting.)*

(b) Describe and analyze an algorithm to compute the magical-mean of a weighted set in $O(n)$ time.

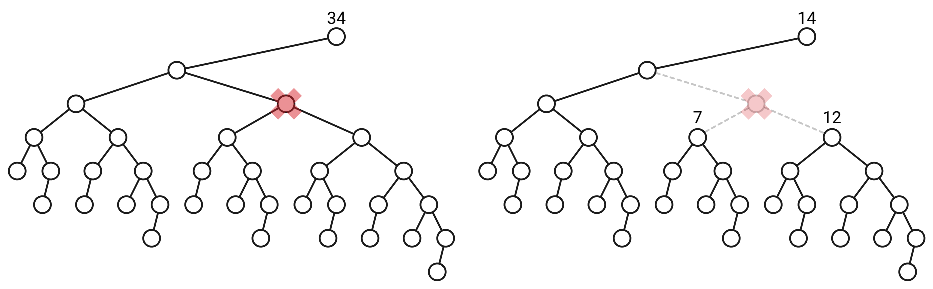
Problem 4

(a) Consider the problem of sorting a sequence of n 0's and 1's using comparisons. For each comparison of two values x and y, the algorithm learns which of $x < y$, $x = y$, or $x > y$ holds. Give an algorithm to sort in $n - 1$ comparisons in the worst case.

(b) You are given an array of strings, where different strings may have different numbers of characters, but the total number of characters over all the strings is n. Show how to sort the strings in $O(n)$ time. Note that the desired order here is the standard alphabetical order; for example, $a < ab < b$.

Problem 5

Let T be a binary tree with n vertices. Deleting any vertex v splits T into at most three subtrees, containing the left child of v (if any), the right child of v (if any), and the parent of v (if any). We call v a *central vertex* if each of these smaller trees has at most $n/2$ vertices. The following figure shows deleting a central vertex in a 34-node binary tree, leaving subtrees with 14, 7, and 12 nodes.



(a) Prove that every tree has a central vertex.

(b) Describe and analyze an algorithm to find a central vertex in an arbitrary given binary tree. (You are given a pointer to the root of that tree.)

Problem 6

[For this problem, you only need to solve one of the two parts. Pick the one you prefer! Nevertheless, you can get bonus points if you submit (correct) solutions for both.]

(a) Suppose we are given two sorted arrays $A[1 \cdots m]$ and $B[1 \cdots n]$ and an integer k. Describe an algorithm to find the k^{th} smallest element in $A \cup B$ in $O(\log(m+n))$ time. For example, if $k = 1$, your algorithm should return the smallest element of $A \cup B$. *(Hint: First come up with an algorithm to find the median of $A \cup B$ in $O(\log(m+n))$ time.)*

(b) Given the root of an n-node binary tree, devise an algorithm that traverses all nodes of this tree in $O(n)$ time, using $O(1)$ space.