1 Reversals (33 pts)

Given an array \([a_1, a_2, \ldots, a_n]\), a reversal is a pair \((i, j)\) such that \(i < j\) but \(a_i > a_j\). For example, in the array \([5, 3, 2, 10]\) there are three reversals \(((1, 2), (1, 3), (2, 3))\). Note that the array has no reversals if and only if it is sorted, so the number of reversals can be thought of as a measure of how well-sorted an array is.

(a) What is the expected number of reversals in a random array? More formally, consider a random permutation of \(n\) distinct elements \(a_1, \ldots, a_n\): what is the expected number of reversals?

(b) Recall the insertion sort algorithm:

```plaintext
for i = 1 to n
    j = i
    while j > 0 and A[j-1] > A[j]
        swap A[j] and A[j-1]
        j = j - 1
```

Suppose that our array has \(d\) reversals. Prove that the running time of insertion sort is \(O(n + d)\).

(c) What does this imply about the average-case running time of insertion sort as a function only of \(n\)? That is, if we draw a permutation uniformly at random, what is the expected running time of insertion sort? Note that this is not a randomized algorithm; this is a deterministic algorithm on a random input.

2 Costly Median (33 points)

Suppose that you are given \(n\) distinct numbers \(x_1, x_2, \ldots, x_n \in \mathbb{R}^+\), each of which also has a cost \(c_i \in \mathbb{R}^+\) so that \(\sum_{i=1}^{n} c_i = 1\). The costly median is defined to be the number \(x_k\) such that

\[
\sum_{i : x_i < x_k} c_i < \frac{1}{2} \quad \text{and} \quad \sum_{i : x_i > x_k} c_i \leq \frac{1}{2}.
\]

Give a deterministic algorithm which finds the costly median and has \(O(n)\) worst-case running time (and prove correctness and running time).
3 Dumbbell Matching (34 points)

You belong to a gym which has two sets of dumbbells $A$ and $B$, each of which has $n$ dumbbells. You know that there is a correspondence between the sets: for every dumbbell in set $A$ there is exactly one dumbbell in set $B$ that has the same weight, and similarly for every dumbbell in set $B$ there is exactly one dumbbell in set $A$ that has the same weight. You want to perform exercises that require two dumbbells of the same weight. So you want to pair up the dumbbells by weight, i.e., for every dumbbell you want to know which dumbbell from the other set has the exact same weight.

Unfortunately the dumbbells are unsorted and unlabeled, and you can’t tell their weights by looking at them. The only way to compare two dumbbells is to pick them both up simultaneously (one in each hand) and perform a curl. By comparing the strain on your arms, you can tell whether the two dumbbells are the same weight, and if not, which one is heavier. Even more unfortunately, the owner of the gym has a rule that two dumbbells from the same set cannot be used at the same time. So you can compare a dumbbell from set $A$ to a dumbbell from set $B$, but cannot compare two dumbbells from the same set.

Design a randomized algorithm which correctly determines the pairing between the sets and uses only $O(n \log n)$ comparisons in expectation. As always, prove correctness and running time.