Homework 2

CS41, Spring 2023 — Due Sunday, February 5

1. You have a sorted array of distinct integers $A[0..n-1]$. You want to determine whether or not there is an index i such that $A[i] = i$. For example, if the input array is $[-9, -5, 2, 6, 20]$, then your algorithm should return `true`, because $A[2] = 2$.

   Describe a recursive algorithm for this problem that runs in $O(\log n)$ time. Prove its correctness and running time.

2. Recall that in the linear-time selection algorithm, we used the median of medians as the pivot in each iteration. That is, we chose the third-smallest element from each of the $n/5$ groups of five, and then we took the median of these medians—that is, the $(n/10)^{\text{th}}$-smallest median—as our pivot. This yielded the recurrence relation

   $$T(n) \leq T(n/5) + T(7n/10) + O(n).$$

   (As in Lecture 4, we simplify the analysis here by ignoring issues of divisibility; for example, we assume that $n/10$ is an integer.)

   Suppose that instead of choosing the third-smallest element from each group of five, we had chosen the **second**-smallest element from each group. Let's call these second-smallest elements the "sub-medians" of their groups. Taking the $(n/10)^{\text{th}}$-smallest sub-median as our pivot would be a bad idea, but this algorithm can still be salvaged!

   Find a value α such that choosing the $(\alpha \cdot n)^{\text{th}}$-smallest of the $n/5$ sub-medians as the pivot would still yield a linear-time selection algorithm. What is the resulting recurrence relation? Briefly explain why your recurrence relation is correct and why its solution is $O(n)$.

3. Let A and B be sorted arrays of length m and n, respectively, with $m < n$. Our goal is to determine whether or not there is any element that appears in both A and B. The optimal approach depends on *how much* shorter A is than B. To compare the lengths asymptotically, we can treat m as a function of n; for example, we might have $m = n/2$ or $m = \sqrt{n}$.

   (a) Describe an algorithm that runs in $O(n)$ time when $m = \Theta(n)$.

   (b) Describe an algorithm that runs in $o(n)$ time when $m = o(n/\log n)$.

   You don't need to prove the correctness or running time of these two algorithms.