Homework #2
Introduction to Algorithms/Algorithms 1
601.433/633
Spring 2018

Due on: Thursday, February 15th, 5:00pm
Late submissions: will NOT be accepted
Format: Please start each problem on a new page.
Where to submit: Gradescope.

Please type your answers in LaTeX; handwritten assignments will not be accepted.
To get full credit, your answers must be explained clearly,
with enough details and rigorous proofs.

February 6, 2018

1 Problem 1 (10 points)

You are given an unsorted integer array $A$ of size $n$. You know that $A$ contains at most $m$ pairs of indices $(i, j)$ such that $i < j$ and $A[i] > A[j]$. To sort array $A$ (in ascending order by default for any sorting problems) you applied algorithm Insertion Sort. Prove that it will take at most $O(n + m)$ steps.

2 Problem 2 (10 points)

Given two unsorted integer arrays, $A$ and $B$, of size $n$, where $A$ has no repeated elements and $B$ has no repeated elements, give an algorithm that finds $k$-th smallest entry of their intersection, $A \cap B$. For full credit, you need to provide an algorithm that runs in $O(n \log n)$ time with correctness proof and running time analysis.
3 Problem 3 (10 points)

An array of numbers $A$ is almost sorted if for every $1 \leq i \leq \sqrt{n} \leq j$, we have $A[i] \leq A[j]$ and for every $\sqrt{n} \leq j \leq k \leq n$ we have $A[j] \leq A[k]$. Give an algorithm that takes an almost sorted array $A$ as input and sorts $A$ in $o(n)$ time. Prove the correctness of your algorithm and analyze the running time.

4 Problem 4 (10 points)

A sequence $a_1, a_2, \ldots, a_n$ has a dominant element if more than half of the elements in the sequence are the same. For example, 3 is a dominant element in the sequence 7, 3, 3, 3, 1, 3, 3, 4, 5, 3. On the other hand, the sequence 54, 1, 1, 2, 3, 2, 3, 6 has no dominant element. Devise a divide-and-conquer algorithm that runs in $O(n \log n)$ time and returns a dominant element in a sequence of $n$ numbers or returns None if no such element exists. Prove the correctness of your algorithm and show that its running time is $O(n \log n)$. (Note: there exists an $O(n)$ algorithm to solve this problem that doesn’t make use of divide-and-conquer if you figure it out, you may prove its correctness and running time instead.)

5 Problem 5 (10 points)

Let’s say that a pivot provides $x|n - x$ separation if $x$ elements in an array are smaller than the pivot, and $n - x$ elements are larger than the pivot. Suppose Bob knows a secret way to find a good pivot with $\frac{n}{3}|\frac{2n}{3}$ separation in constant time. But at the same time, Alice knows her own secret technique, which provides $\frac{n}{4}|\frac{3n}{4}$ separation and works in constant time. Alice and Bob applied their secret techniques as a subroutine in QuickSort algorithm. Whose algorithm works asymptotically faster? Prove your statement.