Homework #3
Introduction to Algorithms
601.433/633
Spring 2020

Due on: Tuesday, February 25th, 12pm
Format: Please start each problem on a new page.
Where to submit: On Gradescope, please mark the pages for each question

1 Problem 1 (24 points)

Recall that when using the QuickSort algorithm to sort an array A of length n, we picked an element $x \in A$ which we called the *pivot* and split the array A into two arrays A_S, A_L such that $\forall y \in A_S, y \leq x$ and $\forall y \in A_L, y > x$.

We will say that a pivot from an array A provides $t|n-t$ separation if t elements in A are smaller than or equal to the pivot, and $n-t$ elements are strictly 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 separation $\frac{n}{4}|\frac{3n}{4}$, her technique also works in constant time.

Recall that in the QuickSort algorithm, as per Section 7.1 in CLRS, the PARTITION subroutine picks a pivot x from A by simply picking the first element in the array. Alice and Bob's subroutines are subroutines for picking the pivot x in the PARTITION subroutine for QuickSort.

Alice and Bob applied their secret techniques as subroutines in the QuickSort algorithm to pick pivots. Whose algorithm works **asymptotically** faster? Or are the runtimes **asymptotically** the same? Prove your statement.

2 Problem 2 (13 points)

Resolve the **asymptotic complexity** of the following recurrences, i.e., solve them and give your answer in Big-Θ notation. Use Master theorem, if applicable. In all examples assume that $T(1) = 1$. To simplify your analysis, you can assume that $n = a^k$ for some a, k.

Your final answer should be as simple as possible, i.e., it should not contain any sums, recurrences, etc.

1. $T(n) = 2T(n/8) + n^{\frac{1}{5}} \log n \log\log n$
2. $T(n) = 8T(n/2) + n^3 - 8n\log n$
3. $T(n) = T(n/2) + \log n$
4. $T(n) = T(n-1) + T(n-2)$
5. $T(n) = 3T(n^{\frac{2}{3}}) + \log n$

3 Problem 3 (13 points)

Let A and B be two sorted arrays of n elements each. We can easily find the median element in A – it is just the element in the middle – and similarly we can easily find the median element in B. (Let us define the median of $2k$ elements as the element that is greater than $k-1$ elements and less than k elements.) However, suppose we want to find the median element overall – i.e., the nth smallest in the union of A and B.

Give an $O(\log n)$ time algorithm to compute the median of $A \cup B$. You may assume there are no duplicate elements.

As usual, prove correctness and the runtime of your algorithm.