CS4102 - ALGORITHMS - SPRING 2022

DUE FRIDAY, FEBRUARY 18, 2022 AT 11:30 PM VIA GRADESCOPE

A3 BASIC(HZ9XS)

Collaboration Policy: You are encouraged to collaborate with up to 3 other students, but all work submitted must be your own *independently* written solution. List the computing ids of all of your collaborators in the `collabs` command at the top of the tex file. Do not share written notes, documents (including Google docs, Overleaf docs, discussion notes, PDFs), or code. Do not seek published or online solutions for any assignments. If you use any published or online resources (which may not include solutions) when completing this assignment, be sure to cite by naming the book etc. or listing a website's URL. Do not submit a solution that you are unable to explain orally to a member of the course staff. Any solutions that share similar text/code will be considered in breach of this policy. Please refer to the syllabus for a complete description of the collaboration policy.

Collaborators: hl8zd xs7tng

Sources: Cormen, et al, Introduction to Algorithms. *(add others here)*

PROBLEM 1 *QuickSort*

1. Briefly describe a scenario when Quicksort runs in $O(n \log n)$ time.

   Solution: When the list is in reverse order, where the pivot is always the median of the list and sublists.

2. For Quicksort to be a stable sort when sorting a list with non-unique values, the Partition algorithm it uses would have to have a certain property (or would have to behave a certain way). In a sentence or two, explain what would have to be true of Partition for it to result in a stable Quicksort. (Note: we're not asking you to analyze or explain a particular *implementation* of Partition, but to describe a general behavior or property.)

   Solution: The partition has to keep the raw order for equal elements. One strategy is to swap the pointed value and the value at pivot if the value at pointer equals to the pivot value.

PROBLEM 2 *QuickSelect and Median of Medians*

1. When we add the median-of-medians method to QuickSelect in order to find a good pivot for QuickSelect, name the algorithm we use to find the median value in the list of medians from the 5-element "chunks".

   Solution: QuickSelect

2. Let's say we used the median-of-medians method to find a "pretty good" pivot and used that value for the Partition we use for Quicksort. (We're *n*ot using that value with QuickSelect to find the real median, but instead we'll just use this "pretty good" value for the pivot value before we call QuickSort recursively.) Fill in the blanks in this recurrence to show the time-complexity Quicksort if the size of the two sub-lists on either side of the pivot were as uneven as possible in this situation:

$$T(n) \approx T(0.3n) + T(0.7n) + \Theta(n)$$

   Replace each "??" with some fraction of n, such as $0.5n$ or $0.95n$ etc.

PROBLEM 3 *Other Divide and Conquer Problems*

1. What trade-off did the arithmetic "trick" of both Karatsuba's algorithm allow us to make, compared with the initial divide and conquer solutions for the problem that we first discussed? Why did making that change reduce the overall run-time of the algorithm?

   Solution: Karatsuba's algorithm is faster while it requires more calculation. Karatsuba's algorithm is faster because it only requires 3 multiplications while divide and conquer requires 4 multiplications.

2. Would it be feasible (without reducing the time complexity) to implement the closest pair of points algorithm from class by handling the points in the runway first, and then recursively solving the left and right sub-problems? If your answer is ”no”, briefly explain the reason why.

   Solution: No. If you handle the points in the runway first, there will be no reference limits for the comparison of runway points, which will make time complexity $O(n^2)$.

3. In the closest pair of points algorithm, when processing points in the runway, which of the following are true?

   (a) It's possible that the pair of points we're seeking could be in the runway and both points could be on the same side of the midpoint.

   (b) The algorithm will have a worse time-complexity if we needed to check 50 points above a given point instead of 7 (as we did in class).

   (c) The algorithm will have a worse time-complexity if we needed to check $\sqrt{n}$ points above a given point instead of 7 (as we did in class).
   Solution: (c) is true

PROBLEM 4 *Lower Bounds Proof for Comparison Sorts*

In class, we saw a lower-bounds proof that general comparison sorts are always $\Omega(n \log n)$. Answer the following questions about the decision tree proof that we did.

1. What did the internal nodes in the decision tree represent?

   Solution: They represent each individual comparison

2. What did leaf nodes of the decision tree represent?

   Solution: They represent permutation of sorted list

PROBLEM 5 *Gradescope Submission*

Submit a version of this `.tex` file to Gradescope with your solutions added. You should only submit your `.pdf` and `.tex` files.