Assignment I for AI2615 (Spring 2022)

March 5, 2022

Warning: You **don't** need to write down the answer for those questions in the margin (but you are encouraged to think about them).

Problem 1 (20 Points)

Prove the following generalization of the master theorem. Given constants $a \geq 1, b > 1, d \geq 0$, and $w \geq 0$, if $T(n) = 1$ for $n < b$ and $T(n) = aT(n/b) + n^d \log^w n$, we have

$$T(n) = \begin{cases} O(n^d \log^w n) & \text{if } a < b^d \\ O(n^{\log_b a}) & \text{if } a > b^d \\ O(n^d \log^{w+1} n) & \text{if } a = b^d \end{cases}.$$

Problem 2 (20 points)

Recall that Merge Sort divides the sequence into two subsequences with nearly the same size and sort them recursively. What if we want the two subsequences to have different sizes?

Please use the *one third dividing approach* (dividing the sequence $a_1, a_2, \ldots, a_n$ into $a_1, a_2, \ldots, a_{\lceil n/3 \rceil}$ and $a_{\lceil n/3 \rceil + 1}, a_{\lceil n/3 \rceil + 2}, \ldots, a_n$) to complete the *One Third Merge Sort Algorithm*, and analyze its time complexity.

Think: How the complexity depneds on $\alpha \in (0,1)$ if one uses α *dividing approach*?

Problem 3 (30 points + 5 points)

For two vectors $\mathbf{a} = (a_1, \ldots, a_d), \mathbf{b} = (b_1, \ldots, b_d) \in \mathbb{R}^d$, we say $\mathbf{a}$ is *greater than* $\mathbf{b}$ if $a_k > b_k$ for each $k = 1, \ldots, d$. You are given two collections of vectors $A, B \subseteq \mathbb{R}^d$. The objective is to count the number of pairs $(\mathbf{a}, \mathbf{b}) \in (A, B)$ such that $\mathbf{a}$ is greater than $\mathbf{b}$. You can assume all the entries in all the vectors are distinct. Let $n = |A| + |B|$ and d be the dimension of the vectors.

We literally write down every entry of every vector consecutively in the input file.

1. (10 points) Design an $O(n \log n)$ time algorithm for this problem with $d = 1$.
2. (20 points) Design an algorithm for this problem with $d = 2$. Your algorithm must run in $o(n^{1.1})$ time.
3. (5 points, bonus) Generalize the algorithm in the last question so that it works for general d. Analyze its running time. The running time must be in terms of n and d.

The are many possible algorithms for question 2. I encourage you to use *divide and conquer* and reduce to the question 1.

Problem 4 (30 points)

Recall that we have learned how to find the k-th element in a list with a randomized algorithm (randomly choose a pivot), can we do it deterministically? In this exercise, we will develop one called the *Median of Medians* algorithm, invented by Blum, Floyd, Pratt, Rivest, and Tarjan.

Why we need to pick the *pivot* x randomly in our randomized algorithm? This is to guarantee, at least in expectation, that the numbers less than x and the numbers greater than x are in close proportion. In fact, this task is quite similar to the task of "finding the k-th largest number" itself, and therefore we can bootstrap and solve it recursively!

Assume we have an array A of n distinct numbers and would like to find its k-th largest number.

1. Consider that we line up elements in groups of three and find the median of each group. Let x be the median of these $n/3$ medians. Show that x is close to the median of A, in the sense that a constant fraction of numbers in a is less than x and a constant fraction of numbers is greater than x as well.

2. Design a recursive algorithm using the above idea and analyze the running time.

Can we improve the running time by increasing the number of elements (e.g. 4,5,6?) in each group? What is the best choice?

Problem 5

How long does it take you to finish the assignment (including thinking and discussion)?

Give a rating (1,2,3,4,5) to the difficulty (the higher the more difficult) for each problem.

Do you have any collaborators? Please write down their names here.