COMP 285 (NC A&T, Spr '22) Weekly Quiz 3

1

Assume we modify the k-select algorithm that we saw in previous lectures; instead of picking the pivot cleverly, we just pick a uniformly random element as the pivot each time. We call the resulting algorithm QuickSelect. What is the worst case runtime of QuickSelect?

Solution

$$\Theta(n^2)$$

If we get unlucky, it's just as bad as the worst-case pivot when we do k-select.

2

Recall our median-of-medians implementation of the Select-K algorithm (where we deterministically select a value close to the median). What is the correct recurrence relation for this algorithm?

Solution

$$T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + O(n)$$

Recall Lecture 8. The $T\left(\frac{n}{5}\right)$ comes from recursively finding the median of the medians array. The $T\left(\frac{7n}{10}\right)$ comes from an upper bound, where we know that the median-of-medians will make the array smaller by at least 30%. Finally, the $O(n)$ is the extra work needed to partition the input into left/right.

3

In the median-of-medians algorithm, we split our input array into chunks of at most size 5, took the median of each chunk, and then took the median of the medians. We argued that 3 out of 5 of the values in the bottom half of the chunks had values smaller than the median of medians. So at least $(3/5) * (1/2) = 3/10$ of the values in our array were smaller. This gave us the recurrence relation below. $T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + O(n)$ What is the recurrence relation if instead we split the input array into chunks of size 7.

Solution

$$T(n) = T\left(\frac{n}{7}\right) + T\left(\frac{5n}{7}\right) + O(n)$$

If we split the input array into chunks of size 7, we have the following:

$$L \leq \frac{5n}{7} + 7, R \leq \frac{5n}{7} + 7$$

Similarly with the size 5 chunks solution, we got $T(n) = T\left(\frac{n}{7}\right) + T\left(\frac{5n}{7}\right) + O(n)$. The argument is that 4 out of the 7 values in the bottom half of the chunks are smaller than the median of medians. So at least $(4/7) * (1/2) = 2/7$ of the values in the array were smaller.

4

In the median-of-medians algorithm, we split our input array into chunks of at most size 5, took the median of each chunk, and then took the median of the medians. We argued that 3 out of 5 of the values in the bottom half of the chunks had values smaller than the median of medians. So at least (3/5)*(1/2) = 3/10 of the values in our array were smaller. This gave us the recurrence relation below. $T(n) = T\left(\frac{n}{7}\right) + T\left(\frac{5n}{7}\right) + O(n)$ What is the recurrence relation if instead we split the input array into chunks of size 3.

Solution

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + O(n)$$

If we split the input array into chunks of size 3, we have the following:

$$L <= \frac{2n}{3} + 3, R <= \frac{2n}{3} + 3$$

Similarly with the size 5 chunks solution, we got $T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + O(n)$.

5

Recall from lecture that BogoSort is the algorithm where we randomly shuffle the array until its sorted. The worst-case running time and the expected running time of BogoSort are the same.

Solution

No

The worst-case running time is infinite and the expected running time is $O(n * n!)$

6

What is the expected running time of QuickSort?

Solution

$\Theta(n \log n)$ since in expectation, it is similar to MergeSort (divides the input into halfs).

7

What is the worst-case running time of QuickSort?

Solution

$\Theta(n^2)$ since in the worst-case, our pivots are bad and our inputs are not split well.

8

There are algorithms that sort elements in time faster that $O(n \log n)$ by not relying on doing comparisons between pairs of elements.

Solution

Yes

It is possible, for example Counting Sort and Radix Sort both of which use the fact that we're sorting integer/strings which can be "hashed".

9

In the implementation of QuickSort covered in lecture, all pairs of elements are always compared at least one.

Solution

False

Each pair of elements gets compared either 0 or 1 times.

10

There is no comparison-based algorithm that has a deterministic or expected running time better than $O(n \log n)$.

Solution

Yes. Any algorithm that relies on comparting elements must ask at least $n \log n$ questions to correcting partition the space. This has been mathematically proven.