COMP 285 (NC A&T, Spr '22) Weekly Quiz 2

Reporting Issues If you find any issues with the solutions, reach out to Chi Wang (author) or Luis Perez (reviewer).

1

Which of the following is the correct recurrence relations for MergeSort?

Solution

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

For each step, MergeSort divides the original problem by two, recursively calling itself to solve these two smaller problems. This is where $2 \cdot T(n)$ comes from. Once it has the answers, it merges the results which takes an additional $O(n)$ time, giving the recurrence above.

2

What's the closed-form solution for the running time of the following recurrence relation $T(n) = 5 \cdot T\left(\frac{n}{3}\right) + O(n)$ (Hint: You might want to use the Master Theorem) (Aside: This is the actual recurrence relation of Strassen's Multiplication Algorithm, an improvement to Karatsuba's)

Solution

$$O(n^{\log_3 5})$$

We can see that $a = 5$, $b = 3$, $d = 1$, $a = 5 > b^d = 3$, so the result would be $O(n^{\log_b a})$ = $O(n^{\log_3 5})$ according to the Master Theorem.

3

What's the closed-form solution for the running time of the following recurrence relation $T(n) = T\left(\frac{999n}{1000}\right) + O(n)$ (Hint: You might want to use the Master Theorem).

Solution

$$O(n)$$

$a = 1, b = \frac{1000}{999}, d = 1, a = 1 < b^d = \frac{1000}{999}$, so the result would be $O(n)$ according to the Master Theorem.

4

Select the recurrence relations below for which you CANNOT directly apply the Master Theorem.

Solution

- $T(n) = 2T(n-1) + O(n^2)$ because were are creating smaller problems of size $n-1$. The Master Theore only works when the problems become a fraction of their original size.
- $T(n) = 4T\left(\frac{9n}{10}\right) + O(n \log n)$ because the additional work we do to combine the problems is not polynomial (eg, n^d) but $n \log n$.
- $T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + O(n)$ because we don't split the original problem into subproblems of equal size.

5

There is an $O(n)$ time algorithm for the k-Select Problem.

Solution

Yes. Use divide-and-conquer recursive-based solution as we covered in class.

6

What is the running time of a mergesort-based solution to the k-Select problem?

Solution

$$\Theta(nlogn)$$

MergeSort will take $\Theta(n \log n)$ time.

7

In our divide-and-conquer recursive-based solution to the k-Select problem, what is the running time if we always pick the minimum as the pivot.

Solution

$$\Theta(n^2)$$

If we always pick the minimum(worst-case pivot), we are unable to divide the problem into half each time. The recurrence relation will be $T(n) = T(n-1) + O(n)$ which will end-up with a running time of $O(n^2)$.

8

In our divide-and-conquer recursive-based solution to the k-Select problem, what is the running time if we always pick the median as the pivot.

Solution

$$\Theta(n)$$

If we always pick the median(best-case pivot), we can divide the problem into half each time. The recurrence relation will be $T(n) = T\left(\frac{n}{2}\right) + O(n)$ which is $O(n)$ by the Master Theorem.

9

The running time of our trivial implementation of k-Select using MergeSort is always slower than the running time of a divide-and-conquer solution that randomly selects the pivot element.

Solution

False

The running time of k-Select using MergeSort is $\Theta(n \log n)$, and in worst-case the divide-and-conquer solution would take $\Theta(n^2)$ time, which is slower than $\Theta(n \log n)$.

10

In practice, it's often best to simply pick the pivot randomly rather than implement a more sophisticated, deterministic pivot selection method.

Solution

True

If there is a bad guy who gets to see our pivot choices, that's just as bad as the worst-case pivot.