EECS3101 notes

Jerry Wu

2023-01-31

Example 1

Cauculate big O and big Ω.

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

We can combine both T functions into one by taking the bigger one, i.e. $T(\frac{2n}{3})$. So for O,

$$(T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + n) \leq (2T(\frac{2n}{3}) + n) \in \Theta(n^{1.7}) = O(n^{1.7})$$

For Ω, use the smaller one; $T(\frac{n}{3})$

$$T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + n \geq 2T(\frac{n}{3}) + n = \Theta(n) = \Omega(n)$$

Example 1 with recursion tree (slide 5)

The end result is $O(nlog(n))$ and $\Omega(nlog(n))$. Because it is lower bounded and upper bound by the same class $(nlog(n))$, We can conclude that the runtime of the function is $\Theta(nlog(n))$.

Exercise 4.3 (important, slide 6)

We want to design an algorithm that selects the k-th smallest element in a list using a pivot helper function. We can use divide and conquer to design this solution.

> *"This is the P O W E R of recursion and mathematical induction!"-Larry YL Zhang 2023*

We always want to partition into evenly sized partitions to split the problem efficiently. $\Theta(n^2)$ is no bueno.

Exercise 4.4: Search a matrix

> *"This shows the P O W E R of divide & conquer and master theorem"- Larry YL Zhang 2023*

We want to search for whether an element x exists in a 2D array of n integers $(\sqrt{n} \times \sqrt{n})$(true/false).

Assume the following are true::

- Each row is sorted in ascending order.
- Each column is also sorted in ascending order.

We want the algorithm to be faster than $\Theta(n)$
Divide into 2 subproblems
Define the runtime as $T(n) = 2T(\frac{n}{2}) + \sqrt{n} \implies \Theta(n)$ **NOT GOOD ENOUGH!**
If we use the middle element as a pivot, we can skip one of the quadrants completely.