Ananya
590013832
B-33
DAA

1. Using all 3 approaches (Tree, Substitution, Master's Theorem)

$$T(n) = 2T(n/4) + \log n$$

① Masters Theorem:

- Parameters: $a = 2$
  $b = 4$
  $f(n) = \log n$
- Calculate $n^{\log_b a}$:

$$n^{\log_b a} = n^{\log_4 2}$$

Since $4^{1/2} = 2$ $(\log_4 2 = 1/2)$

So,

$$n^{\log_4 2} = \sqrt{n}$$

- Applying Case:

Compare $f(n) = \log n$ with $n^{\log_b a} = \sqrt{n}$

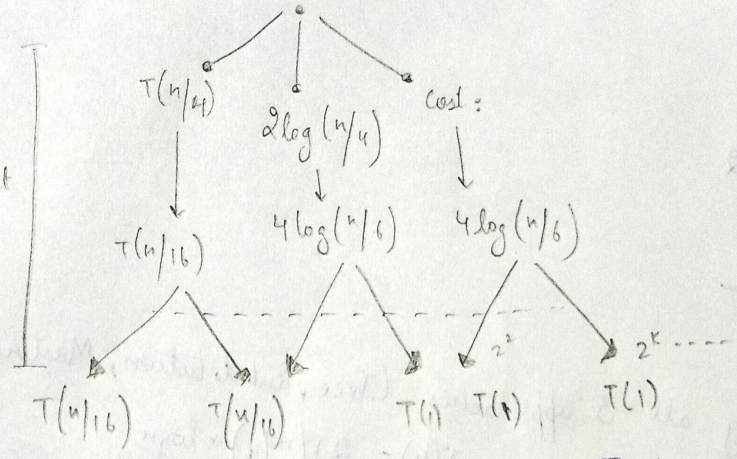
Since $\log n = O(\sqrt{n}^{1-\epsilon})$ for any $\epsilon > 0$, $f(n)$ is polynomially smaller than $n^{\log_b a}$.

This falls under Case 1.

- $\boxed{T(n) = \Theta(n^{\log_b a}) = \Theta(\sqrt{n})}$

② Recursion Tree approach:

We



Total cost (leaves) = $\Theta(\sqrt{n})$

∴ * Height $k = \log_4 n$ ∵ recursion stop when problem size is 1: $n/4^k = 1 \Rightarrow 4^k = n$

- Total Cost: (leaves)

Total no. of leaves (base cases) is 2^k.

Substitute k & no. of leaves = $2^{\log_4 n}$

Using Property:

$$a^{\log_b c} = c^{\log_b a},$$

$$2^{\log_4 n} = \sqrt{n}$$

- Total Cost: (Internal nodes)

↳ Its sum of geometric series but it grows slower than leaves cost.

Since Cost (leaves) is asymptotically larger than root / logn) it's dominate runtime

$$T(n) = \Theta(\sqrt{n})$$

③ Substitution

Inductive Hypothesis:

Assume $T(k) \leq c\sqrt{k}$ for all $k < n$

Substitute

$$T(n) = 2T(n/4) + \log n$$

$$T(n) \leq 2 \cdot c\sqrt{n}/2 + \log n$$

We need to show that $c\sqrt{n} + \log n \leq c\sqrt{n}$ which is impossible since $\log n > 0$ for $n > 1$.

- Fix: $\log n = O(\sqrt{n})$ ($\log n$ lower order, gets easily absorbed.

Therefore upper bound holds.

$$T(n) = O(\sqrt{n})$$

Q2. Explain 'merge sort' & analyse time complexity (Tree):

↓ merge Procedure

Void merge( ):

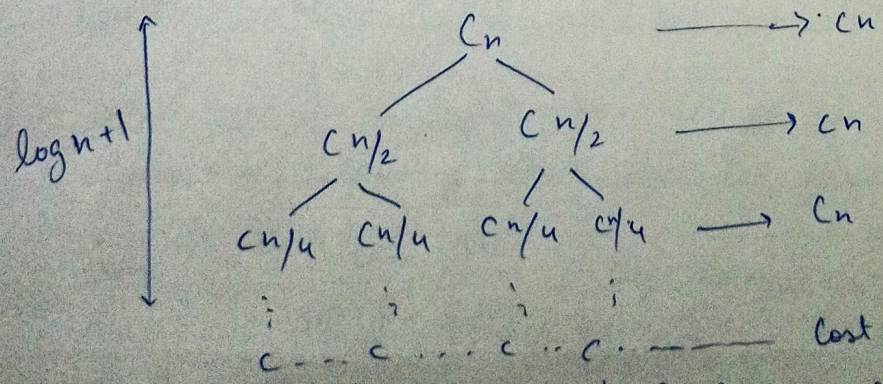
It's key to merge sort. assumes two input subarrays, L & R are already sorted.

Goal: combine those two list into one final.

- Algo uses 2 Pointers to compare smallest remaining element in left subarray with smallest rem. el. in right.
- The smaller element is copied to final array & pointer is advanced.
- Sentinal values manage cleanup, ensure remaining are copied over end.
- This process guarantees sorted list $O(n)$

Time Complexity:

$$T(n) = 2T(n/2) + cn \text{ if } n > 1$$



Total cost = $c(\log n) * cn$, Cost per level × no. levels

$$T(n) = O(n \log n)$$

merge sort has time complexity

$O(n \log n)$ in best, worst, average case