Recurrences and The Master Theorem

Note Title 9/23/2019

Recall: Merge sort

1] Time Complexity of merge sort:

$$T(n) = 2 \times T\left(\frac{n}{2}\right) + \theta(n) \quad \leftarrow \text{recurrence relation}$$

$$= \theta(n \log n) \quad \leftarrow \text{by solving the recurrence}$$

2] Optimal Algorithm:

"Algorithm A is optimal for problem Π, if it has the "best" time complexity among all algorithms that solve Π."

e.g. Merge sort is optimal for sorting

3] Recurrence Relations:

e.g.

$$f(n) = c_1 f(n-1) + c_2 f(n-2) + \cdots$$

How to solve the recurrence relations?

1. Master theorem ✓
2. Expansion / substitution
3. Change of variable

} textbook (Reading)

4] e.g. Hanoi Tower

by expansion:

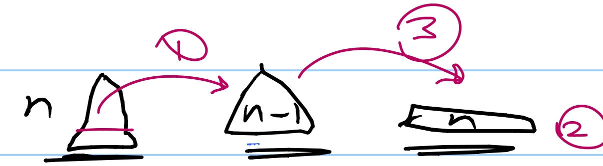
$$h(n) = 2h(n-1) + 1$$

$$= 2[2h(n-2)+1]+1$$

$$=$$

$$\vdots$$

$$= 2^n - 1$$



$$h(n-1) + 1 + h(n-1)$$

$$= 2h(n-1) + 1$$

5] The Master Theorem

Let $a \geq 1$ $b > 1$, $f(n) = a f(\frac{n}{b}) + g(n)$

then :

① if $g(n) = O(n^{\log_b a - \varepsilon})$

then $f(n) = \Theta(n^{\log_b a})$

e.g. Merge sort $f(n) = 2f(\frac{n}{2}) + cn$

look at $\log_b a$ ang $g(n)$

$\Longrightarrow n^{\log_b a}$

② if $g(n) = \Theta(n^{\log_b a})$

then $f(n) = \Theta(n^{\log_b a} \cdot \log n)$

③ if $g(n) = \Omega(n^{\log_b a + \varepsilon})$ and $\begin{cases} \text{if } \exists\, c < 1 \text{ s.t.} \\ a g(n/b) \leq c\, g(n) \quad \forall n > n_0 \end{cases}$

then

$f(n) = \Theta(g(n))$

6] e.g. Binary search : $f(n) = f(\frac{n}{2}) + 1$

$\log_2 1 = 0 \Rightarrow n^{\log_b a} = 1$; $g(n) = 1 = \Theta(1)$

$\therefore$ case 2 : $f(n) = \Theta(n^{\log_b a} \cdot \log n) = \Theta(1 \cdot \log n)$

7] e.g. $f(n) = 4f(\frac{n}{2}) + n$

$n^{\log_2 4} = n^2$, $g(n) = n = O(n^{2-\varepsilon})$

$\therefore$ case 1 $f(n) = \Theta(n^2)$

8]

e.g. $f(n) = 4f\left(\frac{n}{2}\right) + n^2$

$n^{\log_2^4} = n^2$,

$g(n) = n^2 = \theta(n^2)$

$\Rightarrow$ Case 2 :

$f(n) = \Theta\left(n^{\log_b^a} \cdot \log n\right)$

$= \theta(n^2 \cdot \log n)$