Analysis of Algorithms

Chapter 2 : Recurrences

0xSaad Chapters

Introduction

Recurrences arise when an algorithm contains recursive calls to itself.

Analysis of Algorithms

Examples

- T(n) = T(n-1) + n is O(n^2)
- T(n) = T(n/2) + c is O(lg n)
- T(n)= T(n/2) + n is O(n)

Analysis of Algorithms

Methods to solving recurrences

- **Iteration method**
- **Substitution method**
- **Master method**

Analysis of Algorithms

Iteration method

Find the pattern and complete solving a recursive in algorithm.

Analysis of Algorithms

Iteration method

$T(n) = c + T(n/2)$

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

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

$$= c + c + T\left(\frac{n}{4}\right)$$

$$= c + c + c + T\left(\frac{n}{8}\right)$$

$$= Kc + T\left(\frac{n}{8}\right)$$

$$\lg n \cdot c + T(1)$$

$$\boxed{= O(\lg n)}$$

$$\frac{n}{2^K} \cancel{=} \frac{1}{1}$$

$$\boxed{n = 2^K}$$

$$\boxed{\log_2 n = K}$$

Analysis of Algorithms

Substitution method

$$T(n) = 2T(n/2) + n$$

Guess :

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

Induction Goal :

$T(n) \leq c n \log n$

Induction Hypothesis

$T(\frac{n}{2}) \leq c \cdot \frac{n}{2} \log \frac{n}{2}$

Prove :

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

$c \cdot \frac{n}{2} \log \frac{n}{2} + n \leq c n \log n$

Master method

Rule and cases :

$$T(n) = aT(n/b) + f(n).$$

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } f(n) < n^d \\ \Theta(n^d \log n) & \text{if } f(n) = n^d \\ \Theta(f(n)) & \text{if } f(n) > n^d \end{cases}$$

where $d = \log_b^a$

Analysis of Algorithms

Master method

$$T(n) = \underset{a}{3}\,T\left(\frac{n}{\underset{b}{2}}\right) + \underset{f(n)}{n}$$

$$n^{\log_b a} = n^{\log_2 3}$$

$$n^{1.5} \geq n$$

$$\boxed{\text{Case 1}}$$

$$f(n) = O(n^{1.5})$$

Analysis of Algorithms

Master method

$$T(n) = \underset{a}{2}T\left(\frac{n}{\underset{b}{2}}\right) + \underset{f(n)}{n^2}$$

$$n^d = n^{\log_b a} = n^{\log_2 2} = n$$

$$n \leq n^2$$

$$\boxed{\text{case 3}}$$

$$T(n) = \Theta(n^2)$$