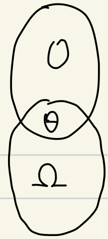


Time complexities

upper bound ← Big - OH

$f, g \to 2\ f^n s$

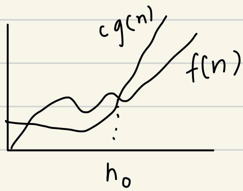
$f(n) : N \to R^+ \qquad g(n) : N \to R^+$

if there exist positive constants c and n_0 such that
$f(n) <= c\,g(n) \quad \forall\ n > n_0$

then $f(n) = O(g(n))$

if $\lim_{n \to \infty} \frac{f(n)}{g(n)} = c < \infty$ then $f(n) = O(g(n))$

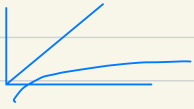
(c can = 0)



here, g(n) gives the asymptotic upper bound for f(n)

$f(n) = \lg(n) \quad g(n) = n$

$\lim_{n \to \infty} \frac{\lg n}{n} = 0 < \infty$



∴ f(n) will increase slower than g(n) as n ↑

lower bound ← Big-Omega

· $f, g \to 2$ f^n

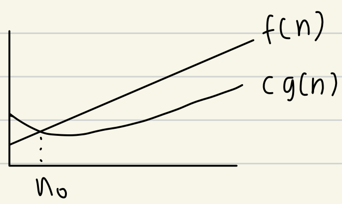
$f(n) : N \to R^+$, $g(n) : N \to R^+$

if there exist positive constants c and n_0 such that
$f(n) \geqslant c(g(n))$ $\forall$ $n > n_0$

then $f(n) = \Omega(g(n))$

· if $\lim_{n \to \infty} \frac{f(n)}{g(n)} = c > 0$, $f(n) = \Omega(g(n))$

(c can = ∞)



here g(n) gives the asymptotic lower bound for f(n)

$f(n) = n^2$ $g(n) = 4n+3$

$\lim_{n \to \infty} \frac{n^2}{4n+3} = \lim_{n \to \infty} \frac{n}{4+3/n} = \infty > 0$

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

" f(n) will increase faster than g(n) as n ↑ "

Big - Theta

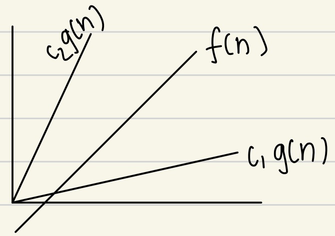
- f, g : $f(n) : N \to R^+$, $f(g) : N \to R^+$

if there exist positive constants c_1, c_2, n_0 :

$c_1 g(n) \leq f(n) \leq c_2 g(n) \quad \forall \; n > n_0$

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

- if $\lim\limits_{n \to \infty} \frac{f(n)}{g(n)} = c$ $\quad (0 < c < \infty)$



here g(n) is the asymptotic tight bound.
f(n) will increase as fast as g(n)

- these notations establish the relative growth between functions (f, g)
- we compare their relative growths

Recursive Algorithms & Recurrence Relations

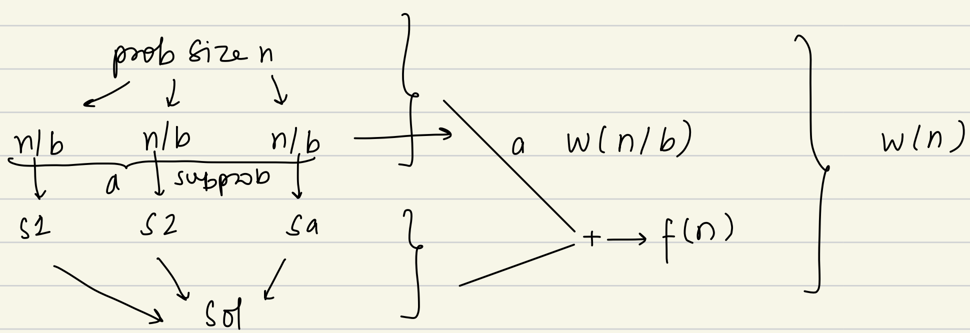
recurrence relation : an equation or inequality that describes a function in terms of its value on smaller inputs

$W(n) = a\,W(n/b) + f(n)$ $\quad a, b > 1$

↑ defines computational cost of an algo using divide and conquer

$f(n)$ = cost of division and recombination at each stage

n is divided into sub problems of size $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$



recognisable recurrence relations

① $W(n) = 2W(n/2) + 2$
finding min or max from sequence

② $W(n) = W(n/2) + 2$
binary search

③ $W(n) = 3W(n/2) + cn$
multiplying 2 2n-bit integers

④ $W(n) = 2W(n/2) + n - 1$
merge sort

⑤ $W(n) = 7W(n/2) + 15n^2/4$
multiplying 2 nxn matrices

read purdue notes

most powerful

Solving

1. Substitution

- guess and check
  1. guess the form of the solution
  2. use mathematical induction to prove it.

- easier to prove than calculate

eg. worst case mergesort $(n = 2^k)$

$W(2) = 1$

$W(n) = 2W(n/2) + (n-1)$

steps
1. guess $W(n) = O(f(n))$
2. show $W(2) <= f(2)$
3. assume for int $k \geq 2$, $W(n) = O(f(n))$ for $n \leq 2^k$
4. prove $W(2n) <= f(2n)$ then
5. $W(n) = O(f(n)) \ \forall \ n \geq 2$

1st guess $W(n) = O(n^2)$

$W(2) = 1 \leq 2^2$ holds

$W(2^k) \leq 2^{2k}$ $\quad n \leq 2^k$

consider $n = 2^{k+1}$

$W(2^{k+1}) = 2W(2^k) + 2^{k+1} - 1$

$W(2^{k+1}) \leq 2 \cdot 2^{2k} + 2 \cdot 2^k - 1$

$W(2^{k+1}) \leq 2 \cdot 2^{2k} + 2 \cdot 2^k - 1 \leq 4 \cdot 2^{2k} - 1$

$W(2^{k+1}) \leq 4 \cdot 2^{2k}$

$W(2^{k+1}) \leq 2^{2(k+1)}$

hence true. best guess?

2^{nd} guess $W(n) = O(n)$

base case : $W(2) = 1 \leq 2c$ ✓

assume : $W(2^k) = O(2^k)$ for $n \leq 2^k$

for $n = 2^{k+1}$

$$W(2^{k+1}) = 2W(2^k) + 2^{k+1} - 1$$
$$\leq 2 \cdot c \cdot 2^k + 2^{k+1} - 1$$
$$= c \cdot 2^{k+1} + 2^{k+1} - 1$$

$\therefore W(2^{k+1}) \leq (c+1) \cdot 2^{k+1}$

but not $\leq c \cdot 2^{k+1}$

$\therefore W(n) \neq O(n)$

3^{rd} guess $W(n) = O(n\log n)$

$W(2) = 1 \leq 2\log 2$ ✓

$W(n) = O(n\log n) \ \forall \ n \leq 2^k$ (assume)

for $n = 2^{k+1}$

$$W(2^{k+1}) = 2W(2^k) + 2^{k+1} - 1$$
$$\leq 2 \cdot 2^k \log 2^k + 2^{k+1} - 1$$
$$\leq 2^{k+1} \log 2^k + 2^{k+1} - 1$$
$$\leq 2^{k+1}(k+1) - 1$$
$$\leq 2^{k+1} \log 2^{k+1}$$

true. nlogn is a very close upper bound.

- if base case doesnt hold for any c, change it.
- what about general case?

if n : $2^k < n < 2^{k+1}$

$\because$ $w(n)$ is monotonically increasing

$w(2^k) < w(n) < w(2^{k+1})$

proven $w(n) = O(n\log n)$ for powers of 2 so

$w(2^{k+1}) \le c \cdot (k+1) \cdot (2^{k+1})$

for any $n < 2^{k+1}$ for some k, $w(n) \le w(2^{k+1})$

$\therefore$ $w(n) \le c * (k+1) * 2^{k+1} < c \cdot \lg(2n) \cdot (2n) < 4cn\lg n$

$\therefore$ $w(n) = O(n\lg n)$

formulating a guess: calculate work at each level, sum it up

$T(n) = 4T(n/2) + n$, $n = 2^k$

n 4^0	n
$2n$ $n/2 \times 4^1$	$T(n/2) \times 4$ 4^1
	$\times 4$
	$T(n/4) \times 16$ 4^2
$4n$ $n/4 \times 16$ 4^2	
	$T(n/8) \times 64$ 4^3
$n/8 \times 4^3$ $8n$	$T(n/16) \times 256$ 4^4

at lvl i $\rightarrow$ $T(n/2^i) \times 4^i + \underline{\underline{2^i n}}$

work at each level

$$\sum_{i=0}^{k} n \cdot 2^i = n(2^{k+1} - 1) = n(2n-1) = 2n^2 - n$$

so try $O(n^2)$

important identities

$\gamma = 1 \quad : \quad 1 + \gamma^1 + \gamma^2 + \ldots + \gamma^k = k + 1 \quad \in \Theta(k)$

$0 < \gamma < 1 \quad : \quad 1 + \gamma^1 + \gamma^2 + \ldots + \gamma^k \approx \dfrac{1}{1-\gamma} \quad \in \Theta(1)$

$1 < \gamma \quad : \quad 1 + \gamma^1 + \gamma^2 + \ldots + \gamma^k \approx \dfrac{\gamma^{k+1}}{\gamma - 1} \quad \in \Theta(\gamma^k)$

$k = \log_b n$ $\qquad (T(n) = aT(n/b) + f(n))$

$\gamma = \left(\dfrac{a}{b^c}\right)$ $\qquad n^c = f(n)$

$$T(n) = n^c \left(\frac{1 - \gamma^{k+1}}{1 - \gamma} \right)$$

2. Iteration Method.

- expand + express as sum
- get sequence
- iterate until boundary is found

finding ending :

① bounding $4^3 < \underline{n} < 4^4$ $\left\lfloor \frac{n}{4^i} \right\rfloor = x$ ← range = ?

go to wikipedia. ← **3. The master method** ← tight bound.

- gives manual for solving recurrence of type

$$W(n) = a\,W(n/b) + \underline{\underline{f(n)}}$$

$a \geq 1$, $b > 1$ ← constants , $W(n)$ ← monotonic, $f(n)$ ← poly.

① If $f(n) = O(n^{\log_b a - \varepsilon})$ for constant $\varepsilon > 0$
$W(n) = \Theta(n^{\log_b a})$

② If $f(n) = \Theta(n^{\log_b a})$ then $W(n) = \Theta(n^{\log_b a}\log n)$

general If $f(n) = \Theta(n^{\log_b a}\log^k n)$ $k \geq 0$
$w(n) = \Theta(n^{\log_b a}\log^{k+1} n)$

③ If $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$ and if $a f(n/b) \leq c f(n)$ for some constant $c < 1$ and a sufficiently large n, then $W(n) = \Theta(f(n))$

work done at each level = $\begin{pmatrix} \text{no of nodes} \\ \text{at that level} \end{pmatrix} \times \begin{pmatrix} \text{size of} \\ \text{nodes} \end{pmatrix}$

work to split + recombine : $f(n)$

n : size of input problem
a : number of subproblems
b : factor by which subproblem size is reduced in each recursive call

- assume $T(n) = \Theta(1)$ → where n is a base case

let $f^n = n^c$
we want : small c, small a, big b

small a ⇑ faster

big b ⇑ greater reduction reach base case faster

critical exponent = $\log_b a$

- compare $f(n)$ w/ $n^{\log_b a}$; whichever is bigger → solution

n ↑ measures num of leaves

pg 96 (117) Intro to Algo.

Case 1 work to split/recombine a problem is dwarfed by subproblems $\rightarrow$ recursion tree is leaf heavy

$f(n) = O(f(n)) = O(n^c) \rightarrow \underbrace{c < \log_b a}$

polynomially $\therefore$ $n^{\log_b a - \varepsilon}$

here, f(n) should be asymptotically smaller than $n^{\log_b a}$ by a factor of n^{ε}

Case 2 work to split/recombine a problem is comparable to subproblems

$f(n) = \theta(n^{\log_b a} \log^k n) \quad k \geq 0$

$W(n) = \theta(n^{\log_b a} \cdot \log^{k+1} n)$

Case 3 Work to split/recombine a problem dominates subproblems $\rightarrow$ recursion tree is root-heavy

$f(n) = \Omega(n^c) \quad c > \log_b a \quad \therefore + \varepsilon$

polynomially

here, f(n) should be asymptotically larger than $n^{\log_b a}$ and

regularity condition $\rightarrow$ $af(n/b) \leq cf(n) \quad c < 1$, large n

$W(n) = \theta(f(n))$

eg 1

$W(n) = 3W(n/3) + 2$

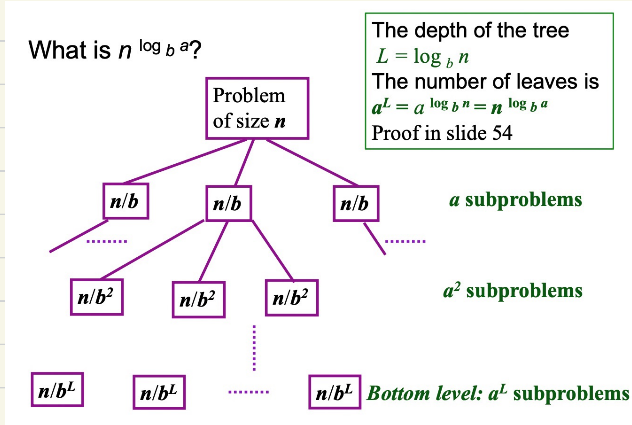
$a = 3,\ b = 3$

no of levels $= \log_3 n$

no of leaves $= a^{\log_3 n}$

$= n^{\log_3 a}$

$= n^{\underbrace{\log_3 3}_{1}} = n$



$f(n) = 2 = \Theta(1) = O(n^0)$

$\therefore f(n) = O(n^{1-\varepsilon})$

ε can be 0.5.

we can use case 1

$\therefore\ W(n) = \Theta(n^{\log_b a}) = \Theta(n)$

eg 2

$W(n) = 4W(n/4) + n - 1$

$a = 4,\ b = 4$ crit $= \log_4 4 = 1$

$f(n) = n - 1 = \Theta(n) = O(n)$ $\varepsilon = 0$ here

$\therefore$ case 2

$f(n) = \Theta(n^{\log_4 4})$

$\therefore\ W(n) = \Theta(n \lg n)$

eg 3 $W(n) = 2W(n/2) + n\log n$

$a = 2$, $b = 2$, $f(n) = n\log n$ $\quad c_{crit} = \log_2 2 = 1$

$f(n) = n\log n = \Theta(n\log n) = \Theta(n^{\log_2 2}\log n)$

$W(n) = \Theta(n^{\log_2 2} \cdot \log^2 n)$

eg 4. $W(n) = 2W(n/4) + n$

$a = 2$, $b = 4$, $f(n) = n$ $\quad c_{crit} = \log_4 2 = 1/2$

$f(n) = n = \Theta(n) = \Omega(n^{1/2 + 1/2})$ $\quad \varepsilon = 1/2$

$a \cdot f(n/b) \leq c\, f(n)$

$2f(n/4) = \frac{n}{2} \leq \frac{3}{4}n \quad \forall n.$

$\therefore W(n) = \Theta(f(n)) = \Theta(n)$

CANT APPLY MASTERS

eg 1.. $W(n) = 3W(n/3) + n/\lg n$

$a = 3$, $b = 3$, $f(n) = n/\lg n$

$f(n) = \frac{n}{\lg n} = O(n)$ $\quad \lim_{n\to\infty} \frac{n/\lg n}{n} = \lim_{n\to\infty} \frac{1}{\lg n} = 0$

$c_{crit} = 1$

$f(n) = O(n) \neq O(n^{1-\varepsilon})$ $\quad \therefore$ no case 1

$\because f(n) \neq \Theta(n)$ $\leftarrow$ fall into gap b/w case 1 & 2

eg 2. $W(n) = W(n/3) + f(n)$

$f(n) = \begin{cases} 3n + 2^{3n} & n = 2^i \\ 3n & \text{else} \end{cases}$, $a = 1$, $b = 3$, $c_{crit} = 0$

$f(n) = O(2^{3n}) = \Omega(n^{0+1})$ $\quad \varepsilon = 1$

case 3 ?

$a f(n/b) \le c f(n)$?

if $n = 3 \cdot 2^i$

LHS $f(2^i) = 3 \cdot 2^i + 2^{3 \cdot 2^i} = n + 2^n$

RHS $c \cdot f(3 \cdot 2^i) = c \cdot 3n$

$\therefore a f(n/b) > c$ for $n = 6$ or greater

Linear Homogeneous recurrence relation

- degree k, constant coeff :

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \dots + c_k a_{n-k}$$

$c_1, c_2, \dots, c_k$ ← real constants , $c_k \neq 0$

- linear : $a_{n-1}, a_{n-2} \dots a_{n-k}$ appear in separate terms and to the 1st power

- Homogeneous : total degree of each term is the same (no constant term, no squared term)

- Constant coefficients : $c_1, c_2, \dots, c_k$ are fixed real constants that do not depend on n

- degree k : expression for a_n contains previous k terms : $a_{n-1}, a_{n-2}, a_{n-3} \dots a_{n-k}$
  $c_k \neq 0$
  but $c_i = 0$ is okay $n-k < i \leq n-1$

- k can be systematically solved → find explicit expression for a_n of the form

$a_n = t^n$ where t is a constant

If $a_n = t^n$ is a soln

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \dots + c_k a_{n-k}$$

then

$$t^n = c_1 t^{n-1} + c_2 t^{n-2} + \dots + c_k t^{n-k}$$

$$t^k = c_1 t^{k-1} + c_2 t^{k-2} + \dots + c_k \quad \text{(divide by } t^{n-k}\text{)}$$

$$t^k - c_1 t^{k-1} - c_2 t^{k-2} - \dots - c_k = 0$$

recurrence relation

$$a_n = c_1 a_{n-1} + c_2 a_{\underline{\underline{n-2}}}$$

characteristic equation

$$x^{\text{degree}} = c_1 x^{\text{deg}-1} + c_2 x^{\text{deg}-2}$$

degree = 2

$$t^2 - c_1 t - c_2 = 0 \leftarrow$$

1. 2 distinct roots
2. 1 root

c_1, c_2 are real

generalised to $a_n = c_1 a_{n-1} + c_2 a_{n-2} \dots c_k a_{n-k}$'s characteristic eqn with k distinct roots

Theorem 1 - Distinct Roots Theorem

- $t^2 - c_1 t - c_2 = 0 \longrightarrow r, s$ 2 distinct roots

a_0, a_1, a_2 are given by the explicit formula:

$a_n = C r^n + D s^n$ (C, D are calced by a_1, a_0)

generalise to less than k roots

Theorem 2 → Single Root Theorem

- r is the single real

then a_n is given by

$a_n = C r^n + D \cdot n \cdot r^n$ (C, D are calced by a_1, a_0)