LECTURE 3

GOWTHAM
WLG07812

INSERTION SORT

- Naive solution
- Compare - right to left element by element

```
insertion_sort (A, n):
    for (i=1 ; i<n; i++)
        k = A[i]
        j = i-1
        while j ≥ 0 and A[j] > k
            A[j+1] = A[j]
            j = j-1
        A[j+1] = k
```

LOOP INVARIANT - INNER LOOP

k is smaller than the elements in the subarray

A[j+1 .... i-1]

OUTER LOOP - LI

All elements in subarray A[0...i-1] are sorted

RUNNING TIME

Represent in summations

$$T(n) = \sum_{i=1}^{n-1} \left(3 + \sum_{j=i-1}^{0} 2\right)$$

↗ since no swaps → 0

$(B \cdot 1) \Rightarrow (n-1)\cdot 3 \Rightarrow \Omega(n)$

↓

Prob = $1/n!$

$= 1/120$

(very less chance)

$$= \sum_{i=1}^{n-1} \left(3 + (i-1+1)2\right) \Rightarrow 3 + 2i$$

$$= (n-1+1)3 + 2\sum_{i=1}^{n-1} i$$

$$T(n) = O(n^2)$$

MERGE SORT - DAC

```
merge_sort(A, n, i, j)
    if (i == j)
        return
    m = ⌊(i+j)/2⌋
    merge_sort(A, n, i, m)
    merge_sort(A, n, m+1, j)
    merge(A, n, i, m, j)

merge(A, n, i, m, j)
    s_L = m - i + 1
    s_R = j - m
    create 2 arrays
    fill it & sentinel
```

(YOU KNOW IT)

LOOP INVARIANTS

1st 2 for loops — true by construction (just copies)

② → from i to k-1, list is sorted

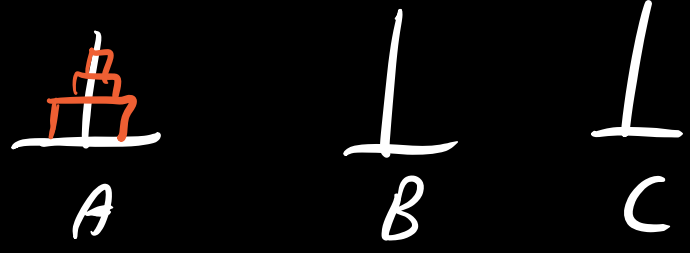
RUN TIME

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

Generally, D&C → $TC = D(n) + C(n) + a\,T(n/b)$

- a: sub prob
- n/b: inp size
- $D(n)$: divide
- $C(n)$: combine
- $T(n/b)$: conquer

TOWER OF HANOI



PATTERN

Move (n-1) tiles into the non-target pile

Move biggest to target pile

Bring others to target

D&C ALGORITHM

ALGORITHM

```
TOH(N, source, target, temp)
    if N!=0
        TOH(N-1, source, temp, target)
        move(N, source, target)
        TOH(N-1, temp, target, source)
```

RUNNING TIME

$$T(N) = 2\,T(N-1) + O(1) + O(1)$$

$$= 2\,T(N-1)$$

$$= O(2^n)$$ EXPONENTIAL RUN-TIME

See it from recursion tree

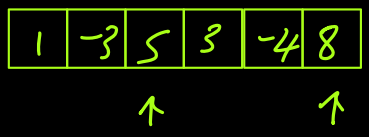
There is no better solution than this ★

(PROVEN)

LECTURE 4

SEP 29

MAXIMUM SUB-ARRAY



BRUTE FORCE

```
def max_subarray(A, n)
  max = -∞, max_i, max_j = 0, 0
  for i = 0 ... n-1
      sum = 0
      for j = i ... n-1
          sum += A[j]
          max_sum = max(sum, max_sum)
          max_i = i
          max_j = j
  return (max_i, max_j, max_sum)
```

TC: $\Theta(n^2)$

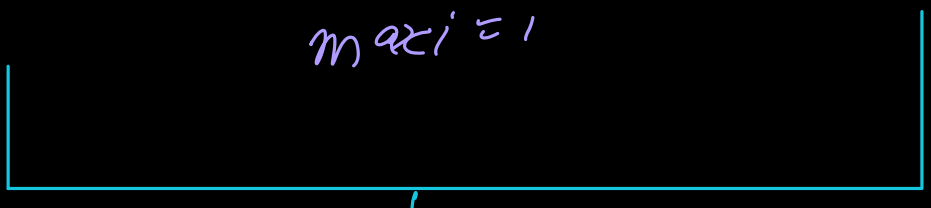
DIVIDE & CONQUER — can solve many problems

Here divide problems into 3 (2 rec & 1 cross-combine)

CROSS ALGORITHM

```
def cross_max(A, l, m, h):
    sumtoL = A[m], maxL = A[m], maxi = m
    for i = m-1 .. 0:
        sumtoL += A[i]
        if sumL > maxL:
            maxL = sumL
            maxi = i
```

TC: O(n)



Same for right

return (maxi, maxj, max sum)

DIVIDE AND CONQUER ALGORITHM

```
d_c_max_sub(A, n, x, y)
    if x = y:
        return (A[x], x, x)
    m = x + y // 2
a = max_sub (A, n, x, m)
b = max_sub (A, n, m+1, y)
c = cross (A, n, x, m, y)
    return max(a, b, c)
```

$TC = 2T(n/2) + \theta(n)$

$TC = \theta(n \log n)$ //

$\theta(n)$ solution ⟶ KADANE'S ALGORITHM

MULTIPLICATION – DIV & CON

```
multiplication (A, B, n)
    if n == 1
        return A[0] * B[0]
    partition A into Ah and Al → Ah * 10^(n/2) + Al = A
    partition B into Bh and Bl → Bh * 10^(n/2) + Bl = B
    C1 = multiplication (Al, Bl, n/2)
    C2 = multiplication (Ah, Bh, n/2)
    C3 = multiplication (Ah, Bl, n/2) ×
    C4 = multiplication (Bh, Al, n/2) ×
    return C2 · 10^n + (C3 + C4) · 10^(n/2) + C1
```

$$TC = 4T(n/2) + \theta(n) + \theta(n) + \theta(1)$$

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

BETTER! →

FROM GUASS (next page)

z = multiplication (Ah+Al, Bh+Bl, n/2)

↓

KARATSUBA ALGORITHM

$$T(n) = 3T(n/2) + \theta(n)$$

$$= \theta(n^{\log_2 3})$$

GUASS

$$(a+b)\cdot(c+d) = a\cdot c + a\cdot d + b\cdot c + b\cdot d$$

$$(A_h + A_l)\cdot(B_h \cdot B_l) = A_h\cdot B_h + A_h\cdot B_l + A_l\cdot B_h + A_l\cdot B_l$$

$$z = (x, y, m/2) \quad c_2 + c_3 + c_4 + c_1$$

$$z - c_2 - c_1 = c_3 + c_4$$

MATRIX MULTIPLICATION

REAL LIFE ⇒ DIFF EQS → FOURIER TRANS → MATRIX MULTIPLICATION

$$\begin{bmatrix} a & c \\ b & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae+cg & af+ch \\ be+dg & bf+dh \end{bmatrix}$$

$$\left[\begin{array}{cc|cc} a_1 & a_2 & a_3 & a_4 \\ a_5 & A & a_6 & C \\ \hline a_9 & a_{10} & a_{11} & a_{12} \\ & B & & D \\ a_{13} & a_{14} & a_{15} & a_{16} \end{array}\right] \left[\begin{array}{cc|cc} b_1 & b_2 & b_3 & b_4 \\ b_5 & E & b_6 & F \\ \hline b_9 & b_{10} & b_{11} & b_{12} \\ & G & & H \\ b_{13} & b_{14} & b_{15} & b_{16} \end{array}\right]$$

DIVIDE & CONQUER - APPROACH

4x2 / 8

$C_1 =$ matrix Mul (AE) + matrixMul (CA)

$C_2 =$ "

$C_3 =$ "

$C_4 =$ "

$$TC = 8T(n/2) + \theta(n^2)$$
$$= \theta(n^{\lg 8})$$
$$= \theta(n^3)$$

STRASSEN'S MATRIX MULTIPLICATION

$$\begin{aligned}
S_1 &= B_{12} - B_{22}, \\
S_2 &= A_{11} + A_{12}, \\
S_3 &= A_{21} + A_{22}, \\
S_4 &= B_{21} - B_{11}, \\
S_5 &= A_{11} + A_{22}, \\
S_6 &= B_{11} + B_{22}, \\
S_7 &= A_{12} - A_{22}, \\
S_8 &= B_{21} + B_{22}, \\
S_9 &= A_{11} - A_{21}, \\
S_{10} &= B_{11} + B_{12}.
\end{aligned}$$

10 MATRICES

$$\begin{aligned}
P_1 &= A_{11} \cdot S_1 \\
P_2 &= S_2 \cdot B_{22} \\
P_3 &= S_3 \cdot B_{11} \\
P_4 &= A_{22} \cdot S_4 \\
P_5 &= S_5 \cdot S_6 \\
P_6 &= S_7 \cdot S_8 \\
P_7 &= S_9 \cdot S_{10}
\end{aligned}$$

7 MULTIPLICATIONS

$$TC = 7T(n/2) + \theta(n^2)$$
$$= \theta(n^{2.81})$$

$\theta(n^{2.38})$ → BEST NOW

TAKE HOMES

1. Karatsuba Algorithm
2. Chapter 3 fully and Chapter 4 (3 methods)

WEEK 3 - LECTURE 5

OCT 4

ASYMPTOTIC NOTATIONS

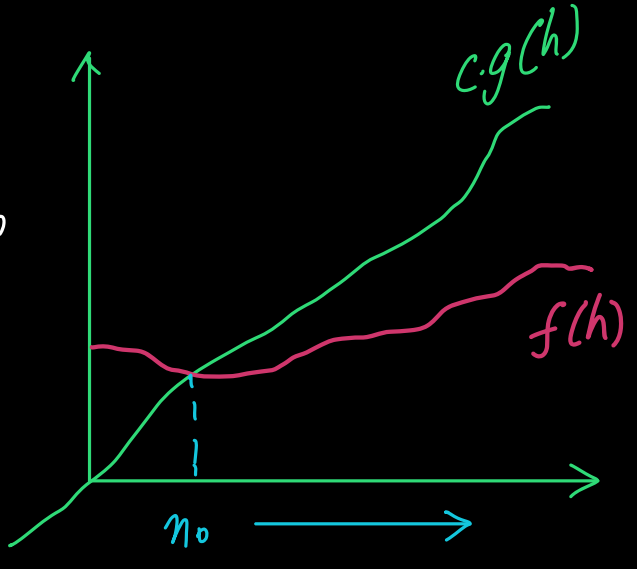
Notations to represent the running time of characteristics of algorithmic function for large enough values of n.

O-NOTATION (Big-oh)

$O(g(n)) = \{ f(n) :$ there exists positive consts. n_0, c such that

$0 \leq f(n) \leq c \cdot g(n)$ for all $n \geq n_0 \}$

* To prove it, pick some values of c and n_0 such that the equation holds true.



EXAMPLE:

$T(n) = \underbrace{n^2+4}_{f(n)} = \underbrace{O(n^2)}_{g(n)}$ — ①

$0 \leq n^2+4 \leq c \cdot n^2$

$n^2+4 \leq c \cdot n^2 \Rightarrow 1 + \frac{4}{n^2} \leq c$ — ∵ by n^2 — ②

$n_0 > 0$ because $n_0 = 0$ doesn't satisfy equation ① — ③

Based on equation ② and ③, we can take

$n_0 = 1$ and $C = 5$

$0 \leq 1 + 4 \leq 5$

Hence Proved

---

① Can $T(n) = n^2 + 4 = O(n^3)$? Yes

$0 \leq n^2 + 4 \leq C \cdot n^3$ for all $n \geq n_0$

$n^2 + 4 \leq C \cdot n^3 \Rightarrow C \geq \frac{1}{n} + \frac{4}{n^3}$

Taking values as $n_0 \geq 3$ and $C = 1$

This equation holds true

---

② Can $T(n) = n^2 + 4 = O(n)$? NO

$0 \leq n^2 + 4 \leq C \cdot n$ for all $n \geq n_0$

$n + \frac{4}{n} \leq C$ → a constant C can never be greater than 'n' for large values of input size (n).

④ $n^3 - n^2 + 5n + 6 = \Theta(n^3 + n^2 - 100n + 10000000000)$

BY DEF,

$$0 \leq n^3 - n^2 + 5n + 6 \leq C \cdot (n^3 + n^2 - 100n + 10000000000)$$

for all $n \geq n_0$

For such big functions, solve separately big term and compare it with lower order terms. *

PROOF:

$$n^3 - n^2 + 5n + 6 \leq C \cdot n^3$$

For $n = 0$ $\quad 6 \leq 0$ ✗

For $n = 1$ $\quad 11 \leq C$ ✓ $\quad \therefore n_0 = 1 \Rightarrow n \geq 1$ — ①

$$1 - \frac{1}{n} + \frac{5}{n^2} + \frac{6}{n^3} \leq C \quad \text{— ②}$$

$n_0 \geq 3$

$C = 3$ [∵ makes calc easier]

SHOW THE WORK
① Partial Credit
② Alg. mistake ignored

$$0 \leq n^3 + n^2 - 100n + 10000000000 \leq C \cdot n^3$$

$n_0 > 0 \quad 1 + \frac{1}{n} - \frac{100}{n^2} + \frac{10\ldots0}{n^3} \leq C$

$n_0 = 1$

$C = 10000000000$ //

⑤ $T(n) = n^2+4 = O(n \log n)$ – NO

$0 \leq n^2+4 \leq c \cdot n \log n$

$n_0 > 0$ (with 0 log becomes ∞) ★

$n_0 = 1 \Rightarrow 4 \leq 0$ ×

$n_0 \geq 2, \quad \frac{n}{\log n} + \frac{4}{n \log n} \leq c$ (n increases faster than log n)

∴ c can never be greater than $\frac{n}{\log n}$. Hence not possible

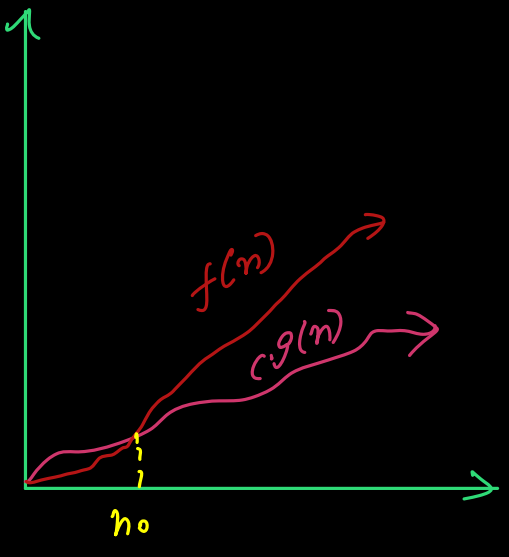
---

Ω NOTATION (Big-Omega)

$\Omega(g(n)) = \{ f(n)$: there exists constants c and n_0 such that

$0 \leq c \cdot g(n) \leq f(n)$ for all $n \geq n_0 \}$

* Gives asymptotic lower bound
* Function cannot go lesser than this



① $T(n) = n^2+4 = \Omega(n)$

$c \cdot n \leq n^2+4$ for all $n > n_0$

$n_0 > 0$

$n_0 = 1$, $c \cdot n \leq n^2+4$ $\quad c = 1$ holds true

$c \leq n + \frac{4}{n}$ $\quad n_0 = 1$

$c \leq 5$

---

② $T(n) = n^2+4 = \Omega(n \log n)$

$c \cdot n \log n \leq n^2+4$

$n_0 > 1$ (you know, see before example)

$n_0 = 2$, $c \log n \leq n + \frac{4}{n}$

$c \leq \frac{n}{\log n} + \frac{4}{n \log n}$ $\quad c = 1$, $n_0 = 2$ holds true

$c \leq \frac{2}{1} + \frac{4}{2}$

$c \leq 4$

Θ - NOTATION (BIG-THETA)

$\Theta(g(n)) = \{ f(n)$: there exists 3 consts. c_1, c_2, n_0 such that ②

$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0$ ①

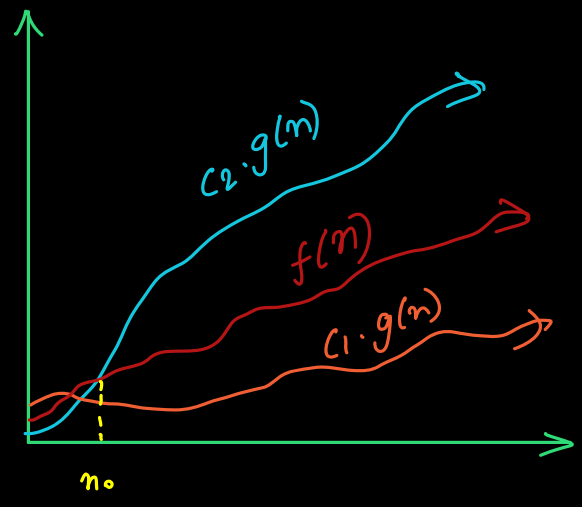
* Always lies between 2 bounds after n_0
* Gives a tight bound.

① & ② prove separately to prove Θ functions

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

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

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



① $T(n) = n^2 + n\log n + 6 = \Theta(n^2)$

$0 \leq c_1 n^2 \leq n^2 + n\log n + 6 \leq c_2 n^2$

$n_0 \geq 1$

$c_1 \leq \underbrace{1}_{\text{fixed}} + \underbrace{\frac{\log n}{n} + \frac{6}{n^2}}_{\text{eventually becomes 2 for large } n} \leq c_2$

worst case $c_1 \leq 1 \leq c_2$

best case $4 \leq 3 \leq c_2$

$\therefore c_1 = 1$, $c_2 = 10$ holds true

$c_1 \leq 1 + 0 + 6 \leq c_2$

$c_1 \leq 7 \leq c_2 \rightarrow$ eventually $c_1 \leq 3 \leq c_2$

PROPERTIES – Use these to prove function bounds

① $f(n) = \Theta(g(n))$ iff $g(n) = \Theta(f(n))$

② $f(n) = O(g(n))$ iff $g(n) = \Omega(f(n))$ (TRANSPOSE SYMM)

③ $f(n) = O(f(n))$
$f(n) = \Omega(f(n))$
$f(n) = \Theta(f(n))$
[REFLECTIVITY]

④ $f(n) = O(g(n))$ and $g(n) = O(h(n))$
then $f(n) = O(h(n))$ (TRANSITIVITY)

Applicable for Ω & Θ as well

LECTURE 6 OCT 6

① $100n + \log n = \Theta(n + \sqrt{n})$

$$0 \le c(n+\sqrt{n}) \le 100n + \log n \le d(n+\sqrt{n})$$

Find d, n_0, c $\qquad n_0 > 1$

÷ by $n+\sqrt{n}$,

$$c \le \frac{100n+\log n}{n+\sqrt{n}} \le d \qquad c \le \frac{100n+\log n}{n+\sqrt{n}} \le d$$

$$\frac{100 + \frac{\log n}{n}}{1+\frac{\sqrt{n}}{n}} \quad \begin{matrix} n \neq 0 \\ n = 1 \end{matrix}$$

$$\frac{100 + \log n/n}{1 + \sqrt{n}/n}$$

$c \le \frac{(60)}{2} \le d$ so

$c \le 50 \le d$

$$\Rightarrow \frac{100}{1+1/\sqrt{n}} + \frac{\log n/n}{1+1/\sqrt{n}}$$

$\downarrow$ $\qquad$ $\downarrow$

val. bet 60-100 $\qquad$ 1 0

$$c \le 60-100 \le d$$

$c = 1$, $d = 200$, $n_0 = 4$ (using 4 in calculation)

The previous method (direct compar.) is time consuming,

We can use properties to find it faster. *

For previous question

$100n + \log n = \theta(n) \iff n = \theta(n + \sqrt{n})$

prove this (easy) — then prove this (easy)

then by property $a < b,\ b < c \Rightarrow a < c$ (transitive?)

$100n + \log n = \theta(n + \sqrt{n})$

MEM. THE PROPERTIES *

LOOSER BOUNDS $(0, \infty)$

$o(g(n)) = \{ f(n) :$ for <u>any</u> $c > 0$, there exists a constant n_0 such that $0 \leq f(n) < c \cdot g(n)$ for all $n \geq n_0 \}$

other way to prove $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$ (easier)

$n^b = O(a^n)$ → poly → expo

$$\lim_{n\to\infty} \frac{n^b}{a^n} = \frac{\infty}{\infty}$$

(review calc. given in book)

Apply log,

$$\log_a\left(\lim_{n\to\infty} \frac{n^b}{a^n}\right) = \lim_{n\to\infty} \log_a \frac{n^b}{a^n}$$

$\lim_{n\to\infty} \frac{n^b}{a^n} = y$

$$= \lim_{n\to\infty} (\log_a n^b - n)$$

$$= -\infty$$

$$\log_a(y) = -\infty$$

$$a^{\log_a(y)} = -a^{\infty}$$

(MULTIPLE STEPS INVOLVED)

$$y = 0$$

$$\lim_{n\to\infty} \frac{n^b}{a^n} = 0 \;//\; [H.P]$$

ω (little omega)

$$\omega(g(n)) = \{ f(n) : \text{for any } c>0, \text{ there exists a constant } n_0 > 0 \text{ such that } 0 \le c \cdot g(n) < f(n) \text{ for all } n \ge n_0 \}$$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \quad \left[\text{You can prove Using this}\right]$$

$f = o(g)$

$g = \omega(f)$

REVIEW THE MATH REQ FOR THIS CHAPTER *

---

①

$n! = \omega(2^n)$

$$\lim_{n\to\infty} \frac{n!}{2^n} \to \frac{\infty}{\infty}$$ (not exists, we need to do something)

USE STERLING APPROXIMATION FOR $n!$

$$\lim_{n\to\infty} \frac{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1+\theta(1/n)\right)}{2^n} = \lim_{n\to\infty} \sqrt{2\pi n}\left(\frac{n}{2e}\right)^n \left(1+\theta\left(\frac{1}{n}\right)\right)$$

$= \infty$ [Hence Proved]

② $c^n = \omega(n^{\sqrt{n}})$

$$\lim_{n\to\infty} \frac{c^n}{n^{\sqrt{n}}} = \frac{\infty}{\infty} \text{ (not exists)}$$

don't use = (transformation)

$$\lim_{n\to\infty} \frac{c^n}{n^{\sqrt{n}}} \xrightarrow{\log_c} \lim_{n\to\infty} \frac{n \log_c c}{\sqrt{n} \log_c n}$$

$$= \lim_{n\to\infty} \frac{\sqrt{n}}{\log_c n}$$

$$\log_c n = \frac{\ln(n)}{\ln(c)}$$

Apply L'Hopital Rule,

$$\lim_{n\to\infty} \frac{\frac{1}{2} n^{-1/2}}{1/n \left(\frac{1}{\ln(c)}\right)}$$

$$= \lim_{n\to\infty} \left[(\ln c) \frac{1}{2} \sqrt{n} \right]$$

(remember to undo log, apply c (whole thing))

$= \infty$ [Hence Proved]

$c^{\infty} = \infty$

PROVING RECURRENCE (MATH INDUCTION)

① $$T(n) = \begin{cases} 1 & n = 1 \\ 2T(n-1)+1, & n > 1 \end{cases} = 2^n$$

BASE CASE

$n = 1$ $\qquad 2^1 - 1 = 1$

$T(1) = 1$

(true)

INDUCTIVE HYPOTHESIS

Suppose $T(k) = 2^k - 1$

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

INDUCTIVE CASE:

$$T(k+1) = 2T(k) + 1 = 2(2^k - 1) + 1 = 2^{k+1} - 2 + 1$$
$$= 2^{k+1} - 1 \;//$$

* By law of Math Ind, the given recurrence is true

(2)

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T(n/2) + n & n > 1 \end{cases} = n \log n + n$$

BASE CASE:

$n = 1$ $\quad\quad$ $1 \log 1 + 1 = 1$

$T(1) = 1$ $\quad\quad$ (True)

IND. HYPO:

Suppose, $T(k) = k \log k + k$ ——— (1)

Prove $T(2k) = [2k(\log 2k)] + 2k$ ——— (2)

IND. CASE

$$T(2k) = 2T(k) + 2k$$

$$= 2[k \log k + k] + 2k$$

$$= 2k \log k + 4k \quad \text{——— (3)}$$

$[2k(\log 2k)] + 2k$ [FROM ②, RHS]

$= 2k(\log 2 + \log k) + 2k$

$= 2k \log k + 2k \log_2 2 + 2k$

$= 2k \log k + 4k$ — ④

③ = ④

∴ By law of math. Induction, the eq. holds true

Lec/HW/Pre. Qs/CLRS ← [TEST UPTO THIS] + Rec. Tree + Sub

oct 20

③ SUBSTITUTION METHOD

$T(n) = 2T(n/2) + \Theta(n) = O(n \log n)$

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

IND HYPO:

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

i.e. $\Rightarrow T(k/2) = (\frac{k}{2} \log \frac{k}{2}$

$T(n) = 2\left[c\frac{n}{2} \log\left(\frac{n}{2}\right)\right] + n \leq cn \log n$

$cn(\log n - \log 2) + n \leq cn \log n$

$cn \log n - cn \log 2 + n \leq cn \log n$

$-cn \log 2 + n \leq 0$

$n \leq cn \log 2$

$1 \leq c \log 2$ (c can be any value)

$\therefore \log 2 \leq c \cdot \log 2$ c is +ve

$\therefore$ this inequality holds true

LECTURE 7 SUBSTITUTION

OCT 11

① $T(n) = 2T(n/2) + \Theta(n) = \Theta(n \log n)$

BY DEF, $0 \leq d n \cdot \log n \leq 2T(n/2) + \Theta(n) \leq C \cdot n \log n$

UPPER BOUND ↓

LOWER BOUND ⟶ Do them separately

* Final constants and imply if it holds true or not
* Can introduce lower order terms.

② $T(n) = 2T(n/2) + \Theta(n) = O(n^2)$

BY DEF, $2T(n/2) + cn \leq dn^2$

$$2\left[d\left(n/2\right)^2\right] + cn \leq d \cdot n^2$$

$$\frac{dn^2}{2} + cn \leq d \cdot n^2$$

$$cn \leq \frac{1}{2} dn^2$$

$$c \leq \frac{dn}{2} \qquad n_0 = 1$$

Can find c and that satisfy the equation

$c = 1, d = 10$ (safe)

∴ The recurrence holds true

③ $T(n) = 2T(n/2) + \Theta n = \Omega(n)^2$

$0 \leq dn^2 \leq 2T(n/2) + cn$

$2\left[\frac{dn^2}{4}\right] + cn \geq dn^2$

$\frac{dn^2}{2} + cn \geq dn^2$

$cn \geq \frac{1}{2} dn^2$

$c \geq \frac{1}{2} dn$ cannot bound c by n (can get very large)

∴ The equation doesn't hold true.

---

④ $T(n) = 2T(n/2) + \Theta n = O(n)$

$0 \leq 2T(n/2) + cn \leq dn$

$2\left[d\frac{n}{2}\right] + cn \leq dn$

$dn + cn \leq dn$

$cn \leq 0$

c and n cannot be < 0

∴ The equation doesn't hold true.

⑤ $T(n) = T(n-1) + c = O(n)$

$0 \leq T(n-1)+c \leq dn$

$d(n-1) + c \leq dn$

$dn - d + c \leq dn$

$c - d \leq 0$

$c=1$
$d=2$

n not there, can pick any n (independent)

∴ The equation holds true.

---

⑤ $T(n) = T(n-1) + c = O(\log n)$

$0 \leq T(n-1)+c \leq d \log n$

$d \log(n-1) + c \leq d \log n$

→ at inf this becomes 0

$c \leq d(\log n - \log(n-1))$

BUT ↑ It works for these values

$c=1$

✗ $d=10$

$n_0 = 4$

$c \leq 0$ (not possible ⇒ $c > 0$)

∴ The equation doesn't hold true.

NOTE:

After proving upper bound, prove lower bound (approach other side) to know the right bound (the tight bound)

⑥ $T(n) = 8\,T(n/2) + cn^2 = O(n^3)$

$8T(n/2) + cn^2 \leq dn^3 \quad (-en^2)$ invisible lower order term to make it work

↳ should be (-)

$8\left[d\left(\frac{n}{2}\right)^3 + e\left(\frac{n}{2}\right)^2\right] + cn^2 \leq dn^3 - en^2$

$-2en^2 + dn^3 + cn^2 \leq dn^3 - en^2$

$-en^2 + cn^2 \leq 0 \quad \because c > 0$, the above equation fails, but it should

It failed because of lower order term ⇒ include that

$-en^2 + cn^2 \leq 0$

$cn^2 \leq en^2$

LEARNING:

If it doesn't hold but looks correct,

you can sub a magic lower order term.

↳ works because $O(g(n))$ is a function with lower order terms.

---

⑦ $T(n) = 7\,T(n/2) + cn^2 = O(n^{\log_2 7})$

$7T(n/2) + cn^2 \leq d\,n^{\log_2 7} - en^2$

$7\left[d\,(n/2)^{\log_2 7} - e\left(\frac{n}{2}\right)^2\right] + cn^2 \leq d\,n^{\log_2 7} - en^2$

$\frac{-7}{4}en^2 + \cancel{d\,n^{\log_2 7}} + cn^2 \leq \cancel{d\,n^{\log_2 7}} - en^2$

$-\frac{3}{4}cn^2 + cn^2 \leq 0$

$n^2\left(c - \frac{3}{4}e\right) \leq 0$ → $n_0 = 1$, $c = 1$, $e = 4$

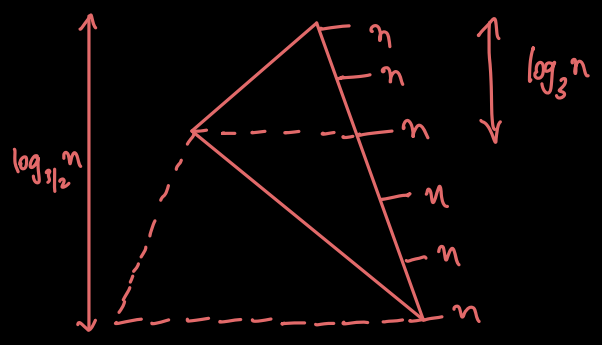
$\therefore$ The equation holds true.

$$T(n) = T(n/3) + T\left(\frac{2n}{3}\right) + n = ?$$

RECURSION TREE

GUESS FROM Δ^{le}:

- $\Omega(n \log_3 n)$
- $\Theta(n \log_{3/2} n)$



$h = \min(\log_3 n, \log_{3/2} n)$

$T(n/3) + T\left(\frac{2n}{3}\right) + n \leq cn \log_{3/2} n$

$\leq cn \log n$ ←

(to simplify)

$O\left(\frac{n \log n}{\log 3/2}\right) = \Theta(n \log n)$

$$\left[c\frac{n}{3}\log\left(\frac{n}{3}\right)\right] + \left[c\frac{2n}{3}\log\left(\frac{2n}{3}\right)\right] + n \leq cn \log n$$

$$\left[c\frac{n}{3}(\log n - \log 3)\right] + \left[\frac{c2n}{3}(\log n + \log 2 - \log 3)\right] + n \leq cn \log n$$

$$-cn \log 3 + \frac{2}{3} cn \log 2 + n \leq 0$$

$$-c \log 3 + \frac{2}{3} c \log 2 + 1 \leq 0$$

$$c\left(\frac{2}{3}\log 2 - \log 3\right) + 1 \leq 0$$

$c = 100, n_0 = 1$

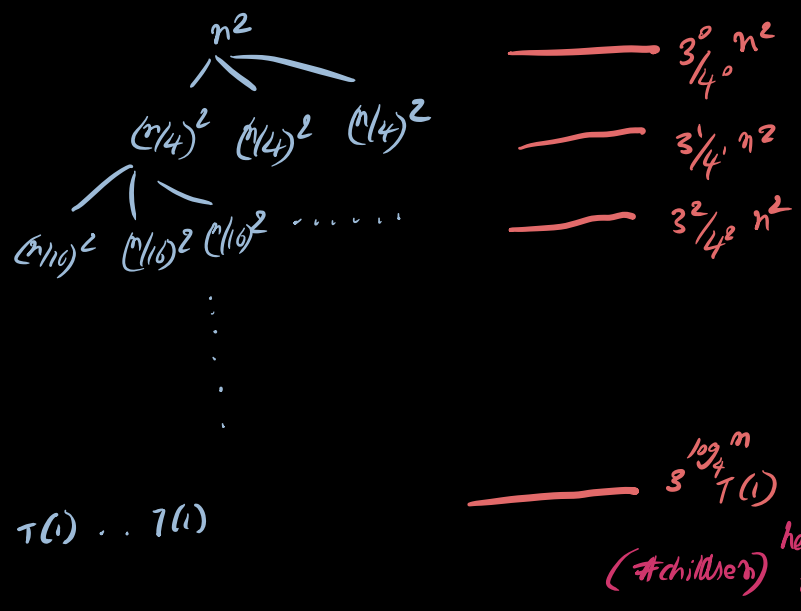
The eq. holds.

Ily, it holds for $O(n \log n)$ [try on your own]

$\therefore T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + n = \Theta(n \log n)$

LECTURE 8 MASTER THEOREM

① $T(n) = 3\,T(n/4) + n^2$



$T(1)$ .. $T(1)$ — $3^{\log_4 n}\, T(1)$

(#children)$^{\text{height}}$ $\times T(1)$

$$T(n) = \sum_{i=0}^{(\log_4 n)-1} 3^i \left(\frac{n}{4^i}\right)^2 + 3^{\log_4 n}$$

$$= n^2 \sum_{i=0}^{(\log_4 n)-1} \left(\frac{3}{16}\right)^i + c \cdot n^{\log_4 3}$$

$\left(\frac{3}{16}\right)^i$ → constant

$= O(n^2)$

$= \Omega(n^2)$ → because this least is constant not 0.

PROOF:

$3\,T(n/4) + n^2 \le cn^2$

$3\left[c\left(\frac{n}{4}\right)^2\right] + n^2 \le cn^2$

$\frac{3}{16} cn^2 + n^2 \le cn^2$

$1 \le \frac{13}{16} c$ $\quad n_0 = 1$ $\quad c = 16$

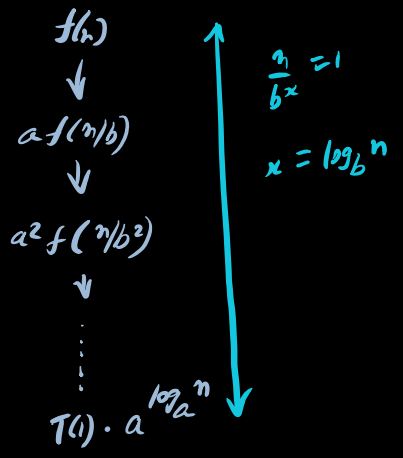
the equation holds true

can prove lower bound (Ω) as well by flipping ($\ge$)

$n_0 = 1,\ c = 1$

$\therefore \Theta(n^2)$

② $T(n) = a\,T(n/b) + f(n)$



$$= \sum_{i=0}^{\log_b n - 1} a^i f(n/b^i) + n^{\log_b a} \cdot T(1)$$

$$= f(n) + a f(n/b) + \cdots$$

$$= c \cdot n^{\log_b a}$$

$= O(\ ) \rightarrow$ 3 cases

Base Behind Master Theorem

MASTER THEOREM

$T(n) = a\,T(n/b) + f(n)$

$a \geq 1$

$b > 1$

$f(n) > 0$

① $f(n) = O\left(n^{\log_b a - \varepsilon}\right)$ $\varepsilon > 0$, then $T(n) = \Theta\left(n^{\log_b a}\right)$

② $f(n) = \Theta\left(n^{\log_b^k a}\right)$, then $T(n) = \Theta\left(n^{\log_b a} \log^{k+1} n\right)$

③ $f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right)$ & $a f(n/b) < c\, f(n)$ $\Big\}$ $T(n) = \Theta(f(n))$

$c < 1$, then

① $f(n) \leq c \cdot n^{\log_b a - \varepsilon}$

$\leq c \dfrac{n^{\log_b a}}{n^{\varepsilon}}$ (polynomially smaller)

② $f(n) \geq c \cdot n^{\log_b a + \varepsilon}$

$\geq c\, n^{\log_b a} \times n^{\varepsilon}$ (polynomially larger)

---

① $T(n) = 3\,T(n/4) + n^2$

$a = 3$

$b = 4$

$f(n) = n^2$ ③

$n^2 = \Omega\left(n^{\log_4 3 + \varepsilon}\right)$

$2 = \log_4 3 + \varepsilon$

$\varepsilon = 2 - 0.2$

$= 1.9 > 0$

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

$af(n/b) < cf(n/b)$

$3\left(\frac{n}{4}\right)^2 < cn^2$

$\frac{3}{16} < c$

(PROVED)

② $T(n) = 8T(n/2) + dn^2$

$dn^2 = O(n^{3-\varepsilon})$ ①

$\varepsilon = 1$

$\therefore T(n) = \theta(n^3)$

③ $T(n) = 2T(n/2) + n$

$a = 2$

$b = 2$

$f(n) = n$ $\quad n^{\log_2 2} = n$ ②

$\therefore T(n) = \theta(n \log n)$ (k=0, use k as log power)

④ $T(n) = 2T(n/2) + n \log n$

$n \log n = \Omega(n^{\log_2 2 + \varepsilon})$ [REMEMBER REFLEXIVITY]

$n \log n \geq c \cdot n \cdot n^{\varepsilon}$

$\log n \geq c \cdot n^{\varepsilon}$

Cannot bound a logarithmic function by polynomial.

Trying case ②

$n \log n = \theta(n^{\log_2 2} \log^k n)$

$k = 1$

$\therefore T(n) = \theta(n \log^2 n)$

⑤ $T(n) = T(n/2) + 1$

$a = 1 \geq 1$

$b = 2 > 1$

$1 > 0$

$1 = \theta(n^{\log_2 1} \log^k n)$

$k = 0$

$\therefore$ Ans $= \theta(1 \cdot \log^{k+1} n) \Rightarrow k =$

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

⑥ $T(n) = 27\, T(n/3) + n^3 \log n$

CASE 3 will fail

Try case 2

$n^3 \log n = \theta(n^3 \log^k n)$

$k = 1 \geq 0$

$\therefore T(n) = \theta(n^3 \log^2 n)$

[When there is $\log n$ in $f(n)$ always go for general case ②]

⑦ $T(n) = 27\, T(n/3) + n^3/\log n$

Fails case ① $\Rightarrow$ cannot find ε

$\frac{n^3}{\log n} = \theta(n^3 \log^k n)$

$k = -1 \quad ! \neq 0$

$\therefore$ None of the cases can be used

(8) $T(n) = T(n/8) + T(n/4) + T(n/3) + n$

$T(n/8)$ → goes down fast → Ω

$T(n/3)$ → goes down slow → 0

Cannot use M.T directly but use it for guessing

approx. from 3 calls above

(3) $T(n/8) + n = \Omega(n)$

(3) $T(n/3) + n = O(n \log n)$ → This will fail when you try to prove.

$\therefore T(n) = \Theta(n)$

CHAP 8 FROM HERE