Date 9/09/22
Page

Design & Analysis of Algorithms

↓ Space & Time Complexity

→ Divide & Conquer
- ↳ Binary Search
2. Merge Sort
3. Quick Sort
4. Strassen's Matrix Multiplication $O(n^{2.81})$
   (Reduce one multiplication operator)

→ Greedy Method
1. Task Scheduling problem
2. Travelling Sales Person problem
3. Knap Sack
4. Minimum Spanning Tree [Kruskal, Prim] UnDirected Graph.
5. Single Source Shortest Path [Dijkastra] → Directed Graph
6. Bellman Ford [-ve weights given]

→ Dynamic Programming. [Bottom Up Approach]

Subproblems dependent on each other

★★
1. Longest Common Subsequence LCS
2. Matrix Chain Multiplication MCM
3. 0/1 Knapsack
4. Travelling Sales Person problem
★ 5. All Pair Shortest Path → (Floyd Warshall)

the.hh.project

Date 12-07-22
Page

→ Backtracking
1. N Queen
2. Sum of subset
3. Hamiltonian Cycle
4. Graph Colouring

→ Branch and Bound
1. TSP
2. 0/1 KnapSack

UNIT5 → String Matching
1. KMP
2. Rabin Karp
3. Boyer Moore

What is an algorithm?
Algorithm is a finite set of instructions that if followd accomplish a particular task.

Various features of an algorithm
1. Input → Atleast one or more quantity is externally supplied.
2. Output → Atleast one quantity should produce
3. Effectiveness → Each instruction should be so basic that it can be carried by any person with help of pen & paper

4. Finiteness → For all cases algorithm should terminate after finite no of steps.

5. Definiteness → Each instruction should be clear and unambigious.

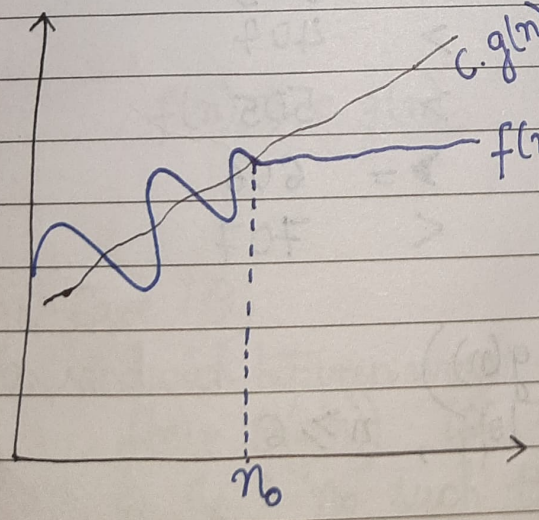
Asymptotic Notation →

1. Big Oh (O) → Worst Case
2. Omega (Ω) → Best Case
3. Theta (Θ) → Average Case

* Big Oh (O) → indicates max^m amount of time required to solve any problem. It describes the worst case of algorithm.

The function $f(n) = O(g(n))$ if & only if there exists +ve constants c and n_0 such that

$$\boxed{f(n) \leq c\, g(n)} \text{ for all } n,\ n \geq n_0$$



$f(n) = 3n+2$ $g(n) = n$ Prove that $f(n) = O(g(n))$

$f(n) \leq c\, g(n)$

$3n+2 \leq c \cdot n$, $n \geq n_0$

the.hh.project

n	$f(n) = 3n+2$		$g(n) = n$	$c.n$	$c=4$
1	5	$>$		$4n = 4$	
2	8	$=$		8	
3	11	$<$		12	

$f(n) = O(g(n))$
for $c = 4$, $n \geq 2$

$f(n) = 100n + 6$
$g(n) = n$
Prove $f(n) = O\,g(n)$

$f(n) \leq g(n)$
$100n + 6 \leq n$

n	$f(n)$		$c \cdot n$ $101n$	$c = 101$
1	106	$>$	101	
2	206	$>$	202	
3	306	$>$	303	
4	406	$>$	404	
5	506	$>$	505	
6	606	$=$	606	
7	706	$<$	707	

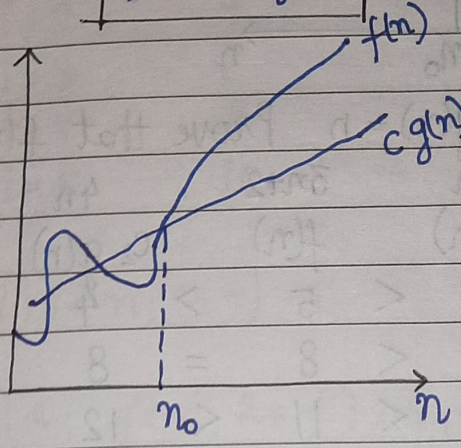
$f(n) = O(g(n))$
for $c = 101$, $n \geq 6$

Date
Page

* Omega Notation (Ω). → defines best case of an algorithm

The function $f(n) = \Omega g(n)$ iff there exists two +ve constants c and n_0 such that

$f(n) \geq c\,g(n)$ for all n, $n \geq n_0$



$f(n) = 3n+2$ $g(n) = n$ Prove that $f(n) = \Omega g(n)$.

$f(n) \geq c.g(n)$

n	3n+2		3n	Let c = 3
1	5	>	3	
2	8	>	6	

$f(n) \geq c.\,g(n)$ $\quad$ $f(n) = \Omega\, g(n)$

* Average Case (Θ)

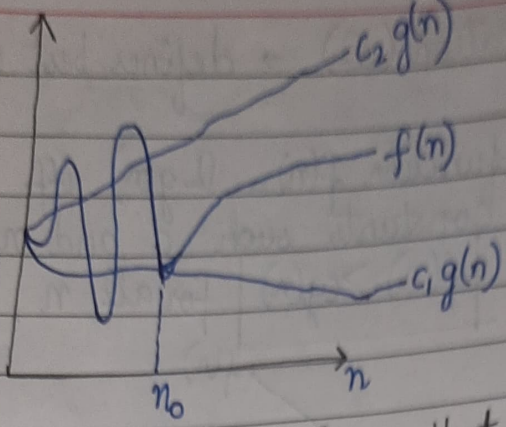
It is a sandwich between worst case & best case

$f(n) = \Theta(g(n))$ iff there exist +ve constants c_1 c_2 & n_0 such that

$c_1.\,g(n) \leq f(n) \leq c_2.\,g(n)$

Best Case ↓ Worst Case.

the.hh.project



$f(n) = 3n+2$ $g(n) = n$ Prove that $f(n) = O(g(n))$

n	$3n$ $c_1 g(n)$		$3n+2$ $f(n)$		$4n$ $c_2 g(n)$
1	3	<	5	>	4
2	6	<	8	=	8
3	9	<	11	<	12

$n_0 = 3$ $c_1 = 3$ $c_2 = 4$ $n \geq 2$

Type of Complexity

1. Constant $O(1)$
2. Logarithmic $O(\log n)$
3. Linear $O(n)$
4. Quadratic $O(n^2)$
5. Cubic $O(n^3)$
6. Exponential $O(2^n)$

(1.)
```
for(i=0; i<100; i++)
  {
    printf("*");
  }
```
100 times
$O(100)$ constant

(2)
```
for (i=0; i<n; i++)
{ for (j=0; j<n; j++)
  { printf(  );
  }
}
```
n^2

```
for(i=0; i<n; i++)
  {
  }
for (j=0; j<n; j++)
  {
  }
```
$n+n = 2n \Rightarrow O(n)$

the.hh.project

Q
```
for (i=1; i<n; 2*i)
{
  printf("  ")
}
```

i=1	2
i=2	4
i=3	6
i=4	8
i=5	10

$2^i < n$

$\log 2^i < \log n$

$i \log 2 < \log n$

$i < \log n$

Q
```
for (i=1; i<n; i++) n
{
  for (j=1; j<n; 2*j) log2 n
  {
    printf("  ")
  }
}
```

Independent loops

$O(n + \log n)$

$O(n)$

$O(n * \log n)$

Recurrence

1. Master's Method
2. Iterative Method
3. Recursion tree
4. Substitution Method.

Generalised form

$$T(n) = \begin{cases} O(1), & \text{if } n \le 1 \\ a.T\left(\frac{n}{b}\right) + D(n) + C(n), & \text{if } n > 1 \end{cases}$$

Imp

a → no of subproblems

n/b → size of subproblem.

D(n) → Time reqd for dividing any array

C(n) → Time reqd for combining sols

f(n) = D(n) + C(n)

the.hh.project

Date
Page

* **Binary Search.**
Right sublist discarded. At a time one

$$T(n) = 1.T\left(\frac{n}{2}\right) + O(1) + O(1)$$

$$T(n) = T\left(\frac{n}{2}\right) + 1.$$

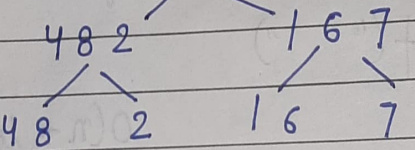
* **Linear Search**

$$T(n) = T(n-1) + O(1) + O(1)$$

Divide, combine

$$T(n) = T(n-1) + 1$$

* **Merge Sort** 4 8 2 1 6 7



$\frac{n}{2}$ → size of subproblems

$$T(n) = 2.T\left(\frac{n}{2}\right) + O(1) + O(n)$$

just cal mid, compare, swaps & combine

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

* **Master's Method**

$T(n) = \Theta\left(n^{\log_b a}\right)$ if $f(n) = O\left(n^{\log_b a - \epsilon}\right)$ for $\epsilon > 0$

2. $T(n) = \Theta\left(n^{\log_b a} \log n\right)$ if $f(n) = O\left(n^{\log_b a}\right)$

3. $T(n) = \Theta(f(n))$ if $f(n) = O\left(n^{\log_b a + \epsilon}\right)$ for $\epsilon > 0$ and if $a f\left(\frac{n}{b}\right) \leq f(n)$ for $c < 1$ and for large n

the.hh.project

Date 15/9/22
Page

Master's Method

Case 1: $T(n) = \Theta(n^{\log_b a})$ if $f(n) = O(n^{\log_b a - \epsilon})$ for $\epsilon > 0$

Case 2: $T(n) = \Theta(n^{\log_b a} \cdot \log n)$ if $f(n) = O(n^{\log_b a})$

Case 3: $T(n) = \Theta(f(n))$ $f(n) = O(n^{\log_b a + \epsilon})$ and $af(\frac{n}{b}) \leq cf(n)$

$c < 1$ and for large value of n

$\boxed{a, b \geq 1}$

Q $T(n) = 9T(\frac{n}{3}) + n$

$T(n) = aT(\frac{n}{b}) + f(n)$

$a = 9 \quad b = 3 \quad f(n) = n$

$n^{\log_b a} \quad n^{\log_3 9} \Rightarrow n^2$

Case 1: $n^{\log_b a} > f(n)$ Case 2: $n^{\log_b a} = f(n)$

Case 3: $n^{\log_b a} < f(n)$

$\because n^{\log_b a} > f(n)$ Case 1 holds

$T(n) = \Theta(n^{\log_b a}) = O(n^2)$

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

$n = O(n^{2-1})$, $\epsilon > 0 \quad \epsilon = 1$

$T(n) = \Theta(n^{\log_b a}) = \Theta(n^2)$

Q $T(n) = T\left(\frac{n}{2}\right) + 1$ // Binary Search

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$ $a=1$ $b=2$ $f(n)=1$

$n^{\log_b a} = n^{\log_2 1} = n^0 = 1$

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

$1 = 1$ Case 2 holds

$T(n) = \Theta\left(n^{\log_b a} \cdot \log n\right) = \Theta(n^0 \log n) = \underline{\Theta(\log n)}$

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

$a=1$ $b=2$ $f(n)=n^2$ $n^{\log_b a} = n^{\log_2 1} = n^0 = 1$

$n^{\log_b a} < f(n)$

Case 3 holds $f(n) = O(n^{0+2})$. $\epsilon = 2$

$n^2 = O(n^{0+2})$

Regularity condition $a f\left(\frac{n}{b}\right) \leq c f(n)$ $f(n) = n^2$

$a=1$ $b=2$ $f\left(\frac{n}{2}\right) = \left(\frac{n}{2}\right)^2$

$f(n) = n^2$

$1 \cdot f\left(\frac{n}{2}\right) \leq c f(n)$

$\left(\frac{n}{2}\right)^2 \leq c \cdot n^2$

$\frac{n^2}{4} \leq c \cdot n^2 \Rightarrow c = \frac{1}{4}$

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

Q $T(n) = 2T\left(\frac{n}{2}\right) + n$ → Merge sort $a=2$ $b=2$ $f(n)=n$

$n^{\log_b a} = n^{\log_2 2} = n = f(n)$ Case 2 holds

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

$= \Theta(n \log n)$

Date
Page

Q. $T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n}$

$a=2 \quad b=4 \quad f(n) = \sqrt{n}$

$n^{\log_b a} \Rightarrow n^{1/2}$ Case 2 holds

$\Theta(n^{1/2} \log n)$

Q $T(n) = T\left(\frac{2n}{3}\right) + 1$

$a=1 \quad b=3/2 \quad f(n)=1$

$n^{\log_b a} = n^{\log_{3/2} 1}$

Q $T(n) = T(\sqrt{n}) + 1$

$a=1 \quad b=1$

$= T(n^{1/2}) + 1$ ——— ①

By changing of variable method

Suppose $m = \log_2 n$

$n = 2^m$

Substitute value of n in eq^n ①

$T(2^m) = T(2^{m/2}) + 1$

$S(m) = S\left(\frac{m}{2}\right) + 1$

$a=1 \quad b=2 \quad f(m)=1$

$m^{\log_b a} = m^{\log_2 1} = m^0 = 1 = f(n)$ Case 2 holds

$S(m) = \Theta(\log m)$

$= \Theta(\log_2(\log_2 n))$

the.hh.project

$$T(n) = 2T(\sqrt{n}) + \log n$$

$$= 2T(n^{1/2}) + \log n \quad \text{——} (1)$$

By changing of variable

$$m = \log_2 n$$

$$n = 2^m$$

Substitute in (1)

$$T(2^m) = 2T(2^{m/2}) + m$$

$$S(m) = 2T\,S\left(\frac{m}{2}\right) + m.$$

$$m^{\log_b a} = m^{\log_2 2} = m^1 = m \qquad f(m) = m$$

Case 2 holds

$$\Theta(m^{\log_b a} \cdot \log m)$$

$$= (m \cdot \log(\log_2 n))$$

$$= \Theta(\log_2 n (\log(\log_2 n)))$$

Date 14-09-22

Master's Theorem
$f(n)$ is the cost of dividing solving a problem

$$T(n) = 3T\left(\frac{n}{4}\right) + n\log n$$

$a = 3 \quad b = 4$

$$n^{\log_b a} = n^{\log_4 3} = n^{0.793}$$

$$f(n) = n\log n$$

Case 3 holds

$$n^{\log_b a} < f(n)$$

$$\log\frac{n}{4} = \log n - \log_2 4$$

Regularity Condⁿ $\quad a f\left(\frac{n}{b}\right) \leq c \cdot f(n)$

$$3\left(\frac{n}{4}\log\frac{n}{4}\right) \leq c \cdot \log n$$

$$\left(\frac{3n}{4}\log n - \frac{3n}{4}\log_2 4\right) \leq c\log n$$

$$\left(\frac{3n}{4}\log n - \frac{3n}{4}2\right) \leq c\log n$$

$c = \frac{3}{4}$, $c < 1$

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

$a = 2 \quad b = 2$

$$n^{\log_b a} = n^1 = n$$

$$f(n) = n\log n$$

$n < n\log n$ Case 3 holds

the.hh.project

$$af\left(\frac{n}{b}\right) \leq c.\ f(n)$$

$$2\left(\frac{n}{2}\log\frac{n}{2}\right) \leq c.\ n\log n.$$

$$n\log n - n\log_2 2 \leq c.n\log n$$

$$n\log n - n \leq c.n\log n$$

$\boxed{c=1}$ } Case 3 does not hold as Regularity cond^n fails }

★ If Polynomial term is left, then we can apply Master's method after dividing

$T(n) = 3T\left(\frac{n}{4}\right) + n\log n$

$n^{0.793} < n\log n$

Bigger term up. $\frac{n\log n}{n^{0.793}} = n^{0.293}\log n$

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

$n < n\log n$

$\frac{n\log n}{n} = \log n.$

★. f(n) should grow asymptotically & ~~exponentially~~ polynomially. Only then Case 3.

$T(n) = 8T\left(\frac{n}{2}\right) + n^2\log n$

$a=8 \quad b=2$

$n^{\log_b a} = n^{\log_2 8} = n^3$

$n^3 > n^2\log n$ Case 3 holds

$\frac{n^3}{n^2\log n} = \frac{n}{\log n}.$

$T(n) = 7T(\frac{n}{2}) + n^2 \log n$

$T(n) = 6T(\frac{n}{2}) + n^2 \log n$

$T(n) = 5T(\frac{n}{2}) + n^2 \log n$ } Case 1:

$T(n) = 4T(\frac{n}{2}) + n^2 \log n$

$n^2 = n^2 \log n$ $\quad \frac{n^2 \log n}{n^2} = \log n$

Case 2. f_n grows asymptotically

There is a gap b/w Case 1 & Case 2

$T(n) = 3T(\frac{n}{2}) + n^2 \log n$

$n^{1} - \frac{n^2 \log n}{n^{0}} = n^{-}$

Case 3

$T(n) = 2T(\frac{n}{2}) + n^2 \log n$

$T(n) = 1T(\frac{n}{2}) + n^2 \log n$

* Iterative Method

$T(n) = 3T(\frac{n}{4}) + n$ — (1)

n → Cost

3 → No of subproblems

$\frac{n}{4}$ → size of each subproblem

$T(\frac{n}{4}) = 3T\left[\frac{n}{4 \cdot 4}\right] + \frac{n}{4}$

Putting value of $\frac{n}{4}$ in (1)

$T(n) = 3\left[3T(\frac{n}{4^2}) + \frac{n}{4}\right] + n$

$= 3^2 T\left[\frac{n}{4^2}\right] + \frac{3n}{4} + n$

$$3^2\left[3T\left[\frac{n}{4^3}\right]+\frac{n}{4^2}\right]+\frac{3n}{4}+n$$

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

$$3^kT\left[\frac{n}{4^k}\right]+\left(\frac{3}{4}\right)^{k-1}n+\cdots\left(\frac{3}{4}\right)^2 n+\frac{3n}{4}+n$$

Last Level

I

II

I $\quad 3^kT\left(\frac{n}{4^k}\right)$

Running Time $\quad T(1)=1 \qquad \frac{n}{4^k}=1$

$$T\left(\frac{n}{4^k}\right)=1 \Rightarrow \boxed{k=\log_4 n}$$

$$3^k(1) \Rightarrow 3^{\log_4 n} \Rightarrow n^{\log_4 3} \Rightarrow n<1$$

Changing variables

II $\quad \left(\frac{3}{4}\right)^{k-1}n \cdots \cdots \left(\frac{3}{4}\right)^2 n+\frac{3n}{4}+n \longrightarrow$ GP

↓

In this avg case is in form of θ

But We assume that we have ∞ terms.

$$\left(\frac{3}{4}\right)^0 n + \left(\frac{3}{4}\right)^1 n + \left(\frac{3}{4}\right)^2 n + \cdots\cdots$$

$$\frac{1}{1-a} = \frac{1}{1-\frac{3}{4}} = 4n$$

I + II $\Rightarrow n^{<1} + 4n = O(n)$ → Worst Case

Q. $T(n) = 2T\left(\frac{n}{2}\right) + n$ — ①

$$T\left(\frac{n}{2}\right) = 2T\left[\frac{n}{2\cdot 2}\right] + \frac{n}{2}$$

Putting the value of $\frac{n}{2}$ in ①

$$T\left(\frac{n}{2}\right) = 2\left[2\cdot T\left(\frac{n}{2^2}\right) + \frac{n}{2}\right] + n$$

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

$$= 2^2\left[2T\left(\frac{n}{2^3}\right) + \frac{n}{2}\right] + n + n$$

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

$$= 2^k T\left(\frac{n}{2^k}\right) + nk$$

I

$2^k T\left(\frac{n}{2^k}\right)$ $\quad T(1) = 1 \quad \frac{n}{2^k} = 1$

$k = \log_2 n$

$2^k = 2^{\log_2 n} \Rightarrow n^{\log_2 2} \Rightarrow n$

$n + n\log n \Rightarrow n\log n$. Merge Sort

the.hh.project

Date
Page

★ $T(n) = T(n-1) + n$ —— ① $O(n^2)$

Worst Case of QuickSort

$T(n-1) = T(n-1-1) + n-1$

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

Putting the value in ①

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

$T(n) = T(n-2) + 2n-1\ (n-1) + n$

$T(n) = T(n-3) + (n-2) + (n-1) + n$

⋮ ⋮ ⋮ ⋮

$T(n-k) + (n-(k-1)) + \ldots\ldots (n-2) + (n-1) + n$

Let $n-k=1$

$k = n-1$.

Sum of n natural nos

$\frac{n(n+1)}{2} = n^2+n$

$1 + n^2 + n = O(n^2)$

Imp ② $T(n) = 8T\left(\frac{n}{2}\right) + n^2$ —— ①.

$T\left(\frac{n}{2}\right) = 8T\left(\frac{n}{2^2}\right) + \left(\frac{n}{2}\right)^2$

Putting in ①

$T(n) = 8\left[8T\left(\frac{n}{2^3}\right)\right] + n + \left(\frac{n}{2}\right)^2$

$T(n) = 8.\left[8T\left(\frac{n}{2^2}\right) + \left(\frac{n}{2}\right)^2\right] + n^2$

$T(n) = 8^8[T$

$$8^2 T\left[\frac{n}{2^2}\right] + 8\frac{n^2}{4} + n^2$$

$$= 8^3 T\left[\frac{n}{2^3}\right] + 4n^2 + 2n^2 + n^2$$

$$= 8^k T\left[\frac{n}{2^k}\right] + \underbrace{(2)^{k-1} n^2 + \dots + 2n^2 + n^2}_{n^2}$$

$\underbrace{\qquad\qquad}_{I}$

$$\frac{n}{2^k} = 1 \Rightarrow k = \log_2 n$$

$$8^k = 8^{\log_2 n} \Rightarrow n^{\log_2 8} = n^3$$

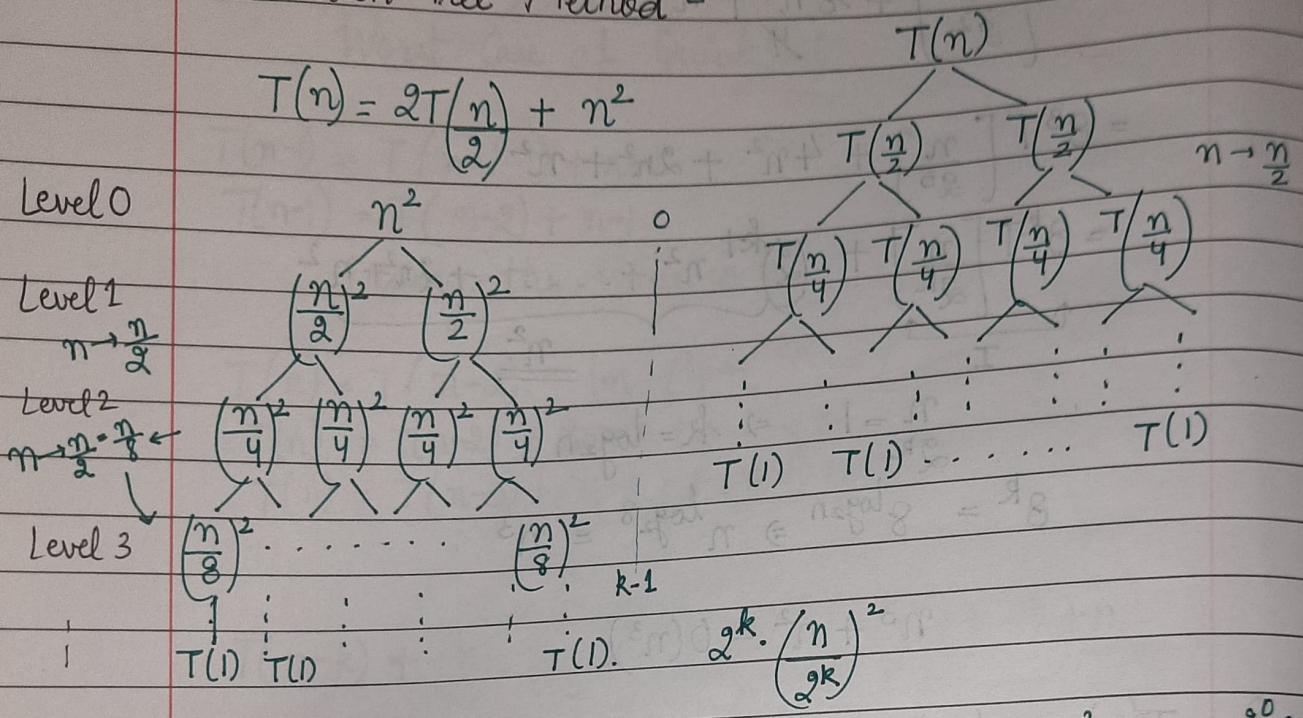
$$n^3 + n^2 = O(n^3)$$

8 $T(n) = 2T\left(\frac{n}{2}\right) + n\log n.$

Date 19/9/22
Page

* Recursion Tree Method -

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



Ht of tree = $\log n$

Level 0 Cost $\rightarrow n^2$ $\quad 2^0 n^2$

Level 1 Cost $\rightarrow \left(\frac{n}{2}\right)^2 + \left(\frac{n}{2}\right)^2$ $\quad 2^1 \cdot \left(\frac{n}{2}\right)^2 = 2 \cdot \frac{n^2}{2^2} = \frac{n^2}{2}$

Level 1 No of nodes x Cost

$2 \times \left(\frac{n}{2}\right)^2$

Level 2 $\rightarrow 2^2 \times \left(\frac{n}{4}\right)^2 \Rightarrow 2^2 \cdot \frac{n^2}{4^2} = \frac{n^2}{2^2}$

Level k $\rightarrow 2^k \cdot T(1)$

$\frac{n}{2^k} = 1 \quad \Rightarrow n = 2^k$

$k = \log_2 n$

Running time at last level = no of nodes at last level * Computing time at that level

$2^k \cdot 1 = 2^{\log_2 n} = n$

$$\left(n^2 + \frac{n^2}{2} + \frac{n^2}{4} + \frac{n^2}{8} + \cdots\right)$$

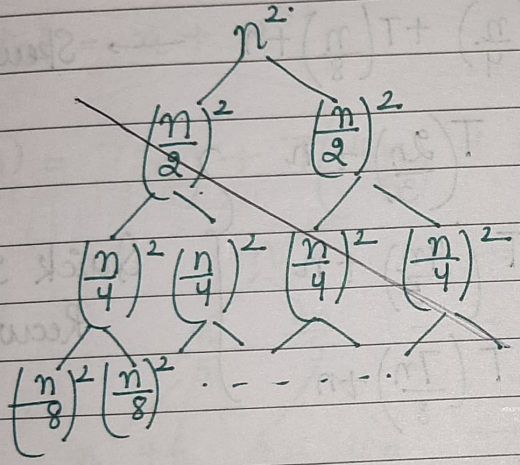
$$n^2\left(\frac{1}{2^0} + \frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} + \cdots\right)$$

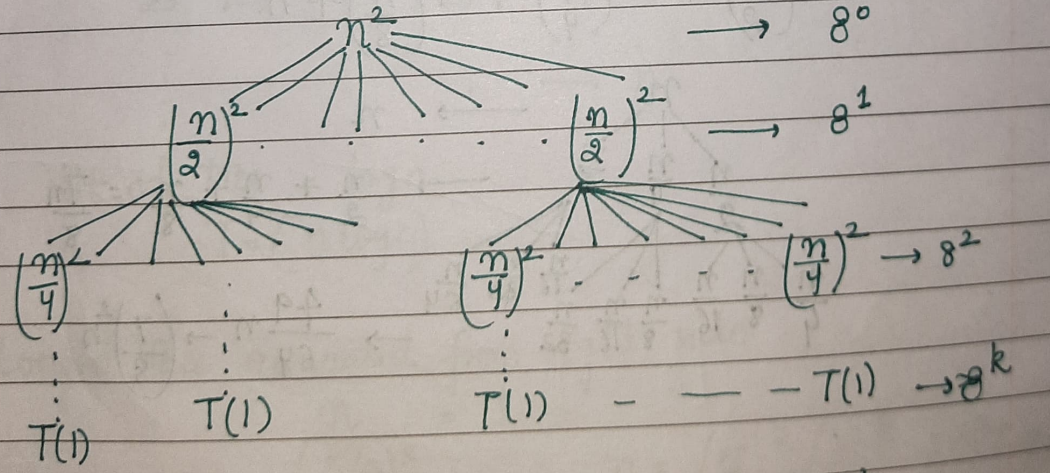
$$= n^2$$

Total complexity $= n^2 + n$

$$= O(n^2)$$

$$T(n) = 8T\left(\frac{n}{2}\right) + n^2$$





Level 0 Cost $\longrightarrow 8^0 \times n^2$ $\quad 8^0 \cdot \left(\frac{n}{2^0}\right)^2$

" 1 " $\longrightarrow 8^1 \times \left(\frac{n}{2}\right)^2$ $\quad \left(\frac{n}{2^1}\right)^2$

2 $\longrightarrow 8^2 \times \left(\frac{n}{4}\right)^2$ $\quad \left(\frac{n}{2^2}\right)^2$

$8^k \cdot \left(\frac{n}{2^k}\right)^2$

the.hh.project

Let $\frac{n}{2^k} = 1 \Rightarrow n = 2^k$

$k = \log_2 n$

Last level $8^k \cdot 1 = 8^{\log_2 n} \Rightarrow n^{\log_2 8} = n^3$

$n^2 + 2n^2 + 4n^2 + \dots$

$n^2(1 + 2 + 4 + \dots)$

n^2

Total complexity $= n^2 + n^3$

$= O(n^3)$

$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + T\left(\frac{n}{8}\right) + n$ ⟶ Skewed Tree

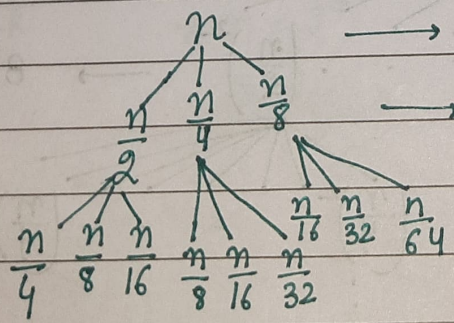
$T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{2n}{3}\right) + n$

$T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{4n}{5}\right) + n$

$T(n) = T\left(\frac{n}{8}\right) + T\left(\frac{7n}{8}\right) + n$

} Quick Sort Avg Case Recurrence

$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + T\left(\frac{n}{8}\right) + n.$



⟶ n

⟶ $\frac{n}{2} + \frac{n}{4} + \frac{n}{8} = \frac{7n}{8}$

$\rightarrow \frac{49}{64} n \rightarrow \left(\frac{7}{8}\right)^2 n$

$T(1) \dots \dots T(1) \rightarrow \left(\frac{7}{8}\right)^k \cdot 1$

$\frac{n}{2^k} = 1 \Rightarrow n = 2^k \Rightarrow k = \log_2 n$

Last level $\left(\frac{7}{8}\right)^k = \left(\frac{7}{8}\right)^{\log_2 n} \Rightarrow n^{\log_2 7/8} \Rightarrow n < 1 \quad n^{<1}$

Level 0 to k-1

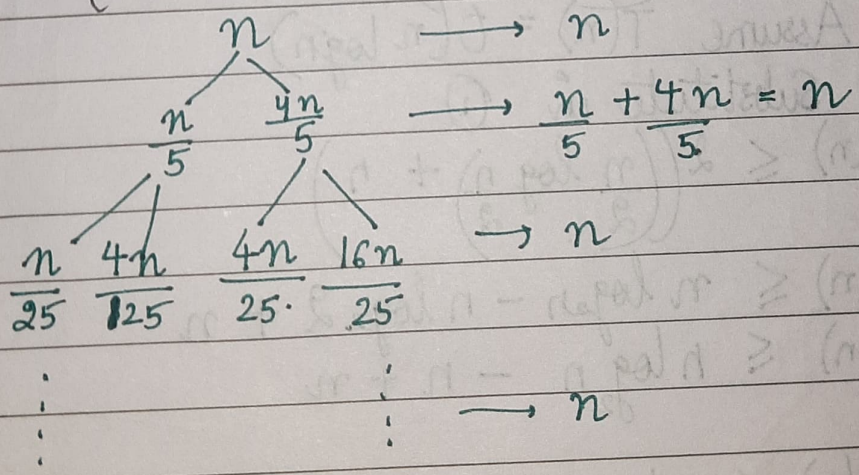
$$n + \frac{7}{8}n + \left(\frac{7}{8}\right)^2 n + \ldots$$

$$n\left[1 + \frac{7}{8} + \left(\frac{7}{8}\right)^2 + \ldots\right] \rightarrow GP$$

$$= n$$

Total $= n + n^{<1}$

Q. $T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{4n}{5}\right) + n$



Ht of tree = $\log n$

$T = (n \log n)$

Longest Tree generated from right side

$$\frac{n}{(5/4)^k} = 1 \qquad n = \left(\frac{5}{4}\right)^k \qquad k = \log_{5/4} n$$

$\theta = (n \log_{5/4} n)$ Complexity at each level * no of levels.

the.hh.project

Imp if $f(n)$ is $n, \log n, n\log n$ guess $T(n) = O(n\log n)$

if $f(n)$ is cons, 1, c $\rightarrow T(n) = O(\log n)$.

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

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

* Substitution Method

1. Make a guess.
2. Prove by Induction.

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

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

$$T(n) \leq \left(2T\left(\frac{n}{2}\right) + n\right) \quad \text{— (1)}$$

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

Substitute in (1)

$$T(n) \leq 2\left(\left(\frac{n}{2}\log\frac{n}{2}\right) + n\right)$$

$$T(n) \leq n\log_2 n - n\log_2 2 + n$$

$$T(n) \leq n\log_2 n - n + n$$

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

$T(n) = O(n\log n) \rightarrow$ Hence Proved

Q $T(n) = 2T\left(\left\lfloor\frac{n}{2} + 17\right\rfloor\right) + n$

$$T(n) = O\left(2T\left(\frac{n}{2} + 17\right) + n\right).$$

$T(n) \leq 2T\left(\frac{n}{2}\right) \left[2T\left(\left(\frac{n}{2}\right)+17\right)+n\right]$.

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

$T(n) \leq 2\left(\frac{n}{2}\log\left(\frac{n}{2}\right)+n\right)$.

$T(n) \leq n\log_2 n - n\log_2 2 + n$

$T(n) \leq n\log_2 n - n + n$

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

$T(n) = O(n\log n)$ Hence Proved.