8·3·24

Quick Sort

→ Worst Case: The array is already sorted → either ascending or descending.

→ In-place Sorting Algorithm, as ~~not~~ no extra memory space or array is needed.

→ Although the worst case time complexity of Quick Sort is $O(n^2)$, the constant terms associated with it, are much less than ~~the~~ those for other sorting algorithms. Hence, it is considered to be the best sorting algorithm.

Pseudocode :-

Quicksort(A, p, r) // p → starting index, r → last index

1. if p < r
2. q = PARTITION(A, p, r)
3. Quicksort(A, p, q-1)
4. Quicksort(A, q+1, r)

PARTITION(A, p, r)

1. x = A[r]

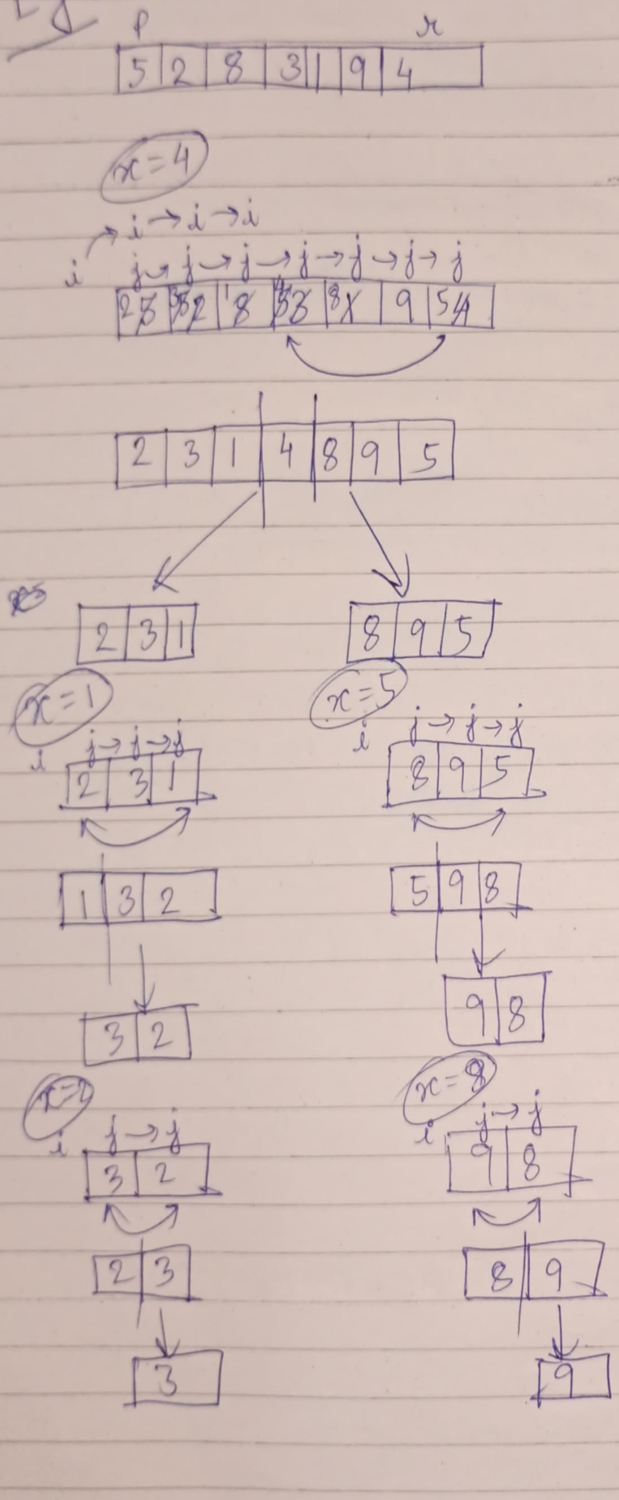
~~2. i = p + 1~~

2. i = p - 1
3. for j = p to r-1
4. if A[j] ≤ x
5. i = i + 1
6. exchange A[i] with A[j]
7. exchange A[i+1] with A[r].

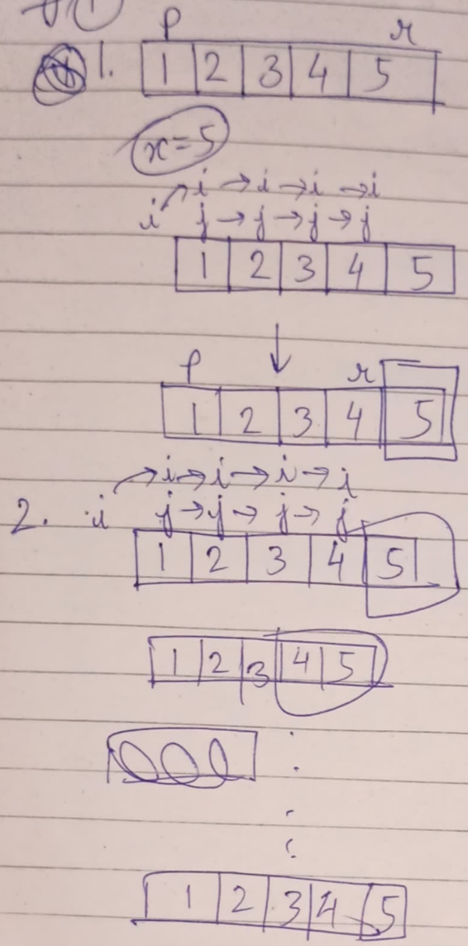
P.T.O

8. return i+1

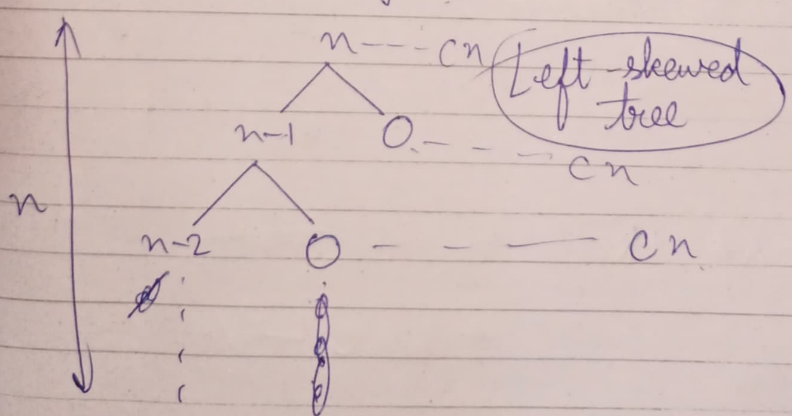
Eg:



Eg: 1

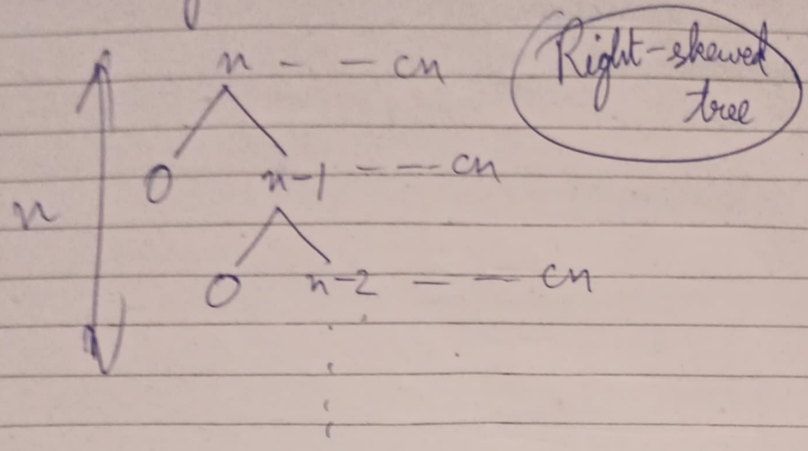


→ When array is sorted :-



PTO

:> When array is reverse-sorted :-



∴ ~~By~~

Recursion Tree Method :-

$$T(n) = n \cdot cn$$
$$= cn^2$$
$$\approx O(n^2).$$

Substitution Method :-

$$T(n) = T(n-1) + T(0) + nc.$$
$$= T(n-2) + \cancel{T(\ldots)} + 2T(0) + [(n-1)c + nc]$$

$2nc - c$

⋮

$$= T(n-k) + kT(0) + \cancel{(n-k)c + nc} + knc - (k+1)c$$

$n - k = 1$
$\Rightarrow k = n - 1$

$$\therefore T(n) = (n-1)nc \approx O(n^2).$$

$1, \frac{5}{3}, \frac{19}{9}.$

12.3.24

Revise

Recursion
- Direct
- Indirect
- Tail

Recurrences :-

1. Substitution Method
2. Recursion Tree Method
3. Master Theorem.

~~Substitution~~ Substitution Method

Ex-1

$$T(n) = \begin{cases} 1 & , \text{ if } n=1 \\ 2T(\lfloor n/3 \rfloor) + n & , \text{ otherwise} \end{cases}$$

$$T(n) = 2T(\lfloor n/3 \rfloor) + n$$

$$= 2\left[2T(\lfloor n/3^2 \rfloor) + n/3\right] + n = 2^2 T(\lfloor n/3^2 \rfloor) + \frac{5}{3}n.$$

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

$$= 2^3 T(\lfloor n/3^3 \rfloor) + \frac{4}{9}n + \frac{2}{3}n + n \; \cancel{= 2^3 T(\lfloor n/3^3 \rfloor) + \frac{19}{9}n.}$$

$\vdots$

$$= 2^k T(\lfloor n/3^k \rfloor) + \sum_{i=0}^{k-1} \left(\frac{2}{3}\right)^i n$$

P.T.O

Let $\frac{n}{3^k} = 1 \Rightarrow k = \log_3 n$

$$\therefore T(n) = 2^k \cdot T(1) + \sum_{i=0}^{k-1} \left(\frac{2}{3}\right)^i n.$$

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

$$\leq 2^k + \sum_{i=0}^{\infty} \left(\frac{2}{3}\right)^i \cdot n$$

$$\leq n + 3n. \quad \left[\because 3^k = n, \ \therefore 2^k \leq n\right].$$

$$\cancel{\leq 4n}$$

$$\therefore T(n) = O(n).$$

Ex-2

$$T(n) = 2T(\sqrt{n}) + \log_2 n.$$

Let $m = \log_2 n$.

$$\Rightarrow n = 2^m.$$

$$\Rightarrow \sqrt{n} = 2^{m/2}.$$

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

Let $S(m) = T(2^m)$.

$$S(m) = 2S(m/2) + m$$

$\therefore S(m) = O(m \log_2 m)$.

$\therefore T(n) = T(2^m) = S(m) = O(m \log_2 m)$.

$= O(\log_2 n \cdot \log_2 \log_2 n)$.

<u>Ex-3</u>

$$T(n) = \begin{cases} 1, & \text{if } n = 1 \\ 5T(n/5) + \dfrac{n}{\log_2 n}, & \text{otherwise.} \end{cases}$$

$$T(n) = 5T(n/5) + \frac{n}{\log_2 n}.$$

~~Let $m = \log_2 n$.~~

~~$\Rightarrow n = 2^m$.~~

$$= 5\left[5 \cdot T(n/5^2) + \frac{n/5}{\log_2 n/5}\right] + \frac{n}{\log_2 n}.$$

$$= 5^2 T(n/5^2) + \frac{n}{\log_2 n/5} + \frac{n}{\log_2 n}.$$

~~$= 5^2 [5T(n/5^3) +$~~

$$= 5^3 T(n/5^3) + \frac{n}{\log_2 n/5^2} + \frac{n}{\log_2 n/5} + \frac{n}{\log_2 n}$$

$\vdots$

$$= 5^k T(n/5^k) + \frac{n}{\log_2 n/5^{(k-1)}} + \frac{n}{\log_2 n/5^{(k-2)}} + \dots + \frac{n}{\log_2 n}$$

Let us assume, at k^{th} iteration, the problem size becomes 1.

$$\therefore \frac{n}{5^k} = 1$$

$$\Rightarrow k = \log_5 n.$$

$$\therefore T(n) = 5^k + n \sum_{i=0}^{k-1} \frac{1}{\log_2 n/5^i}$$

$$\leq 5^k + n. \sum_{i=0}^{\infty} \frac{1}{\log_2 n/5^i}$$

Ans
$O(\log_2 \log_5 n)$.

Recursion Tree Method :-

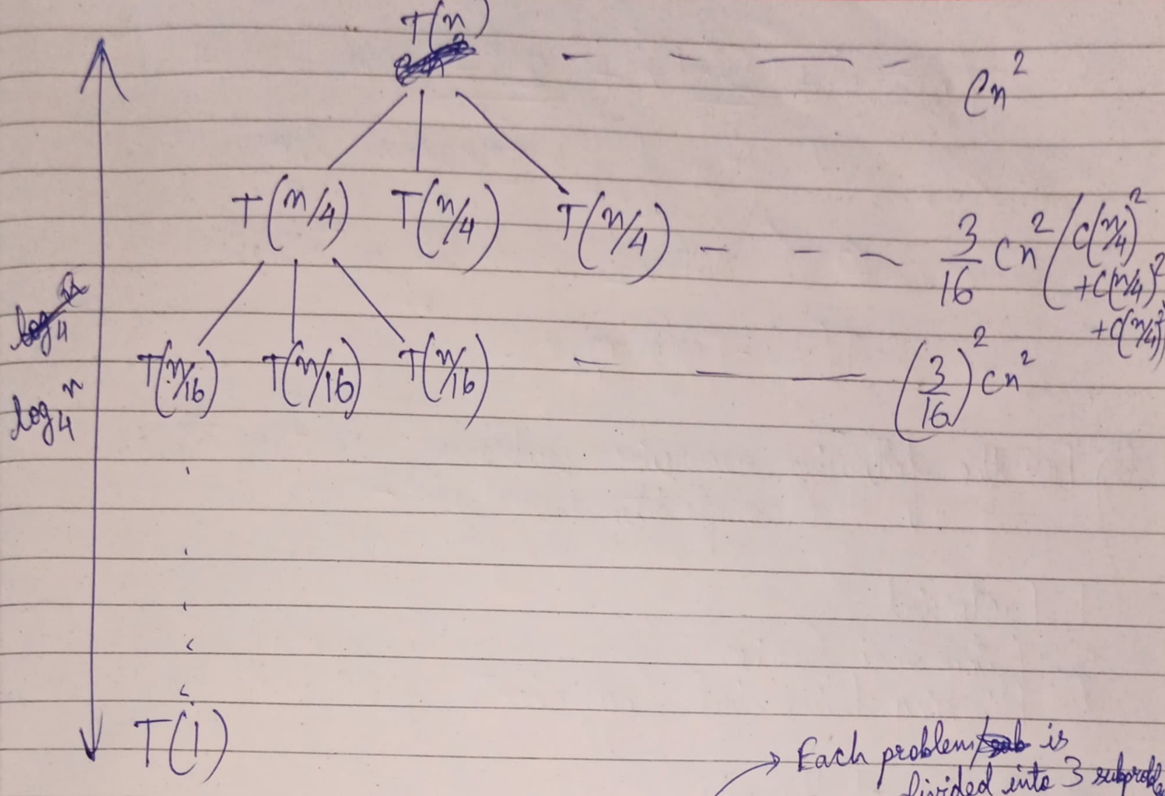
Ex-1

$$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2).$$

$$c > 0.$$

$$\Theta(n^2) = cn^2.$$

$n^{\log_4 3}$



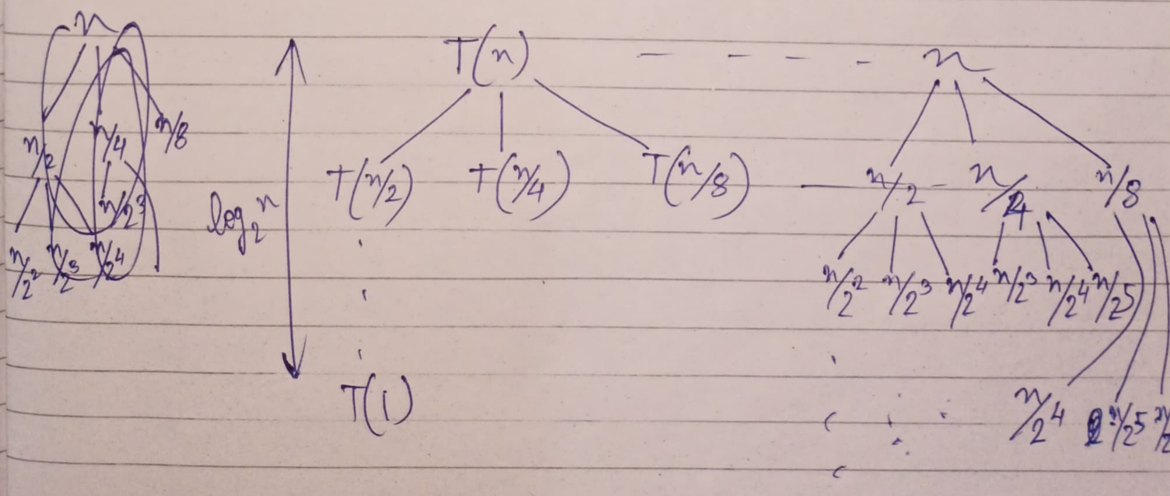
Each problem is divided into 3 subproblems

$$\therefore T(n) = \Theta\left(n^{\log_4 3}\right) \rightarrow \frac{1}{4}\text{th}$$

$$O(n^c)$$

Ex-2

$$T(n) = T(n/2) + T(n/4) + T(n/8) + n.$$



PTO

$$T(n) = n + \frac{2^3 - 1}{2^3} n + \frac{2^6 - 1}{2^6} n + \dots + \log_2 n \text{ times}$$

$$\leq n + n + \dots + \log_2 n \text{ times}$$

$$\therefore T(n) = O(n \log_2 n).$$

Q) For the following recursive problems, do the time complexity analysis:-

1. Factorial
2. Fibonacci Series
3. Finding Binomial Coefficient $\left({}^nC_r \, x^r\right)$.
4. Tower of Hanoi

Solution of Recurrence ~~Rea~~ Relation using Master Method

13.3.24

~~Let us ass~~

Assumption:-

Let $T(n) = aT(n/b) + f(n)$,

where $a \geq 1$ and $b > 1$ both are constants

and $f(n)$ is an asymptotically positive function.

Master Theorem:-

Let $a \geq 1$ and $b > 1$ be constants,

let $f(n)$ be a function, and let $T(n)$ be defined on non-negative integers by the recurrence,

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

where, we interpret $\frac{n}{b}$ ~~to mean~~ as $\lfloor \frac{n}{b} \rfloor$ or $\lceil \frac{n}{b} \rceil$.

Then, $T(n)$ has the following asymptotic bounds:-

1. If $f(n) = O(n^{\log_b (a - \epsilon)})$ for some constant $\epsilon > 0$,

   then, $T(n) = \Theta(n^{\log_b a})$. slightly below

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

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

   $\log_2 n$

PTO

slightly upper

3. If $f(n) = \Omega\left(n^{\log_b a + \epsilon}\right)$ for some constant $\epsilon > 0$, and if $a f(n/b) \leq c \cdot f(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$.

[a → No. of subproblems.
b → Size of subproblem.]

Ex - 1

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

$a = 2,\ b = 2,\ f(n) = cn.$

$n^{\log_b a} = n^{\log_2 2} = n^1 = n.$

Case - 2

$T(n) = \Theta(n \log n).$

Ex - 2

$T(n) = 16T(n/4) + n^3.$

$a = 16,\ b = 4,\ f(n) = n^3.$

$n^{\log_b a} = n^{\log_4 16} = n^2.$

$f(n) = \Omega(n^2)$.

Case-3

$\frac{n}{b} = \frac{n}{4}$

$f(n/4) = \frac{n^3}{64}$.

$a.f\left(\frac{n}{b}\right) = 16 \times \frac{n^3}{64} = \frac{n^3}{4}$

$\frac{n^3}{4} \leq c n^3$.

or, $c \geq \frac{1}{4}$

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

$\therefore T(n) = \theta(f(n))$

or, $T(n) = \theta(n^3)$.

$\Rightarrow$ If $a = \cancel{64}\ 256$, $b = 4$, $f(n) = n^3$.

$n^{\log_4 256} = n^4$.

$f(n) = O(n^4)$.

Case-1

$T(n) = \theta(n^4)$.

PTO

Factorial

Multiplication takes constant time

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

$$= T(n-2) + 2C.$$

$$= T(n-3) + 3C.$$

$\vdots$

$$= T(n-k) + kC.$$

$n-k = 0$ (or 1)

$n = k$

$\therefore T(n) = T(0) + nC = 1 + nC.$

$$= O(n).$$

Fibonacci Sequence

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

$$= T(n-2) + T(n-3) + C + T(n-3) + T(n-4) + C + C$$

$$= T(n-2) + 2T(n-3) + T(n-4) + 3C.$$

~~Using Cha~~

Characteristic equation: $x^2 = x+1$

$\Rightarrow x = (1+\sqrt{5})/2$

$x = (1-\sqrt{5})/2.$ } Golden Ratio

~~F(n)~~ = Solution of linear recursive function :-

$F(n) = (\alpha_1)^n + (\alpha_2)^n$, where α_1 and α_2 are the roots.

$\therefore$ Solution is: $F(n) = \left(\frac{1+\sqrt{5}}{2}\right)^n + \left(\frac{1-\sqrt{5}}{2}\right)^n.$

T(n) are F(n) are asymptotically same, as they are representing the same equation.

$$T(n) = O\left(\left(\frac{1+\sqrt{5}}{2}\right)^n + \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

$$= O\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right)$$ [~~Higher valu~~ Greater term is taken]

$$= O((1.6180)^n)$$

$$\approx O(2^n)$$

$1.6180 \Rightarrow$ Golden Ratio.

P.T.O

Tower of Hanoi

Algorithm:-

TowerOfHanoi(n, source, dest, spare)

<pre-cond.> : The n^{th} smallest disk is on pole ~~source~~ $_{source}$.

<post cond.> : They are moved to the pole ~~destination~~, $pole_{dest}$.

```
begin
    if (n ≤ 0)
        Nothing to do
    else
        TowerOfHanoi(n-1, source, spare, dest)
        Move the nth disk from pole_source to pole_dest.
        TowerOfHanoi(n-1, spare, dest, source)
    end if
end
```

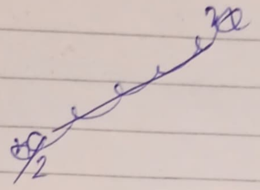
13.3.24 — Practice (Home)

① $T(n) = 4T(n/2) + cn^2$.

② $T(n) = 2T(n/2) + (n/\log n)$.

③ $T(1) = 1$, $T(n) = 3T(n/2) + 2n^{1.5}$.

Ans. 1)

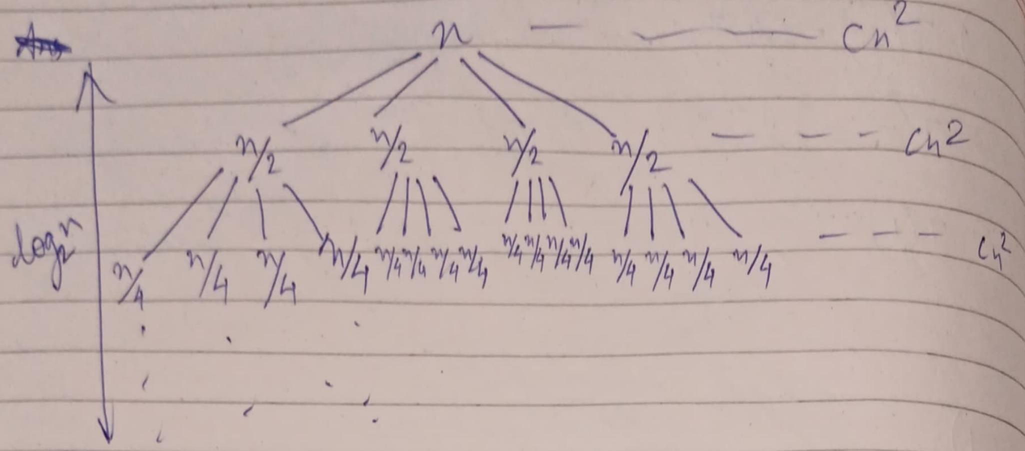


$$T(n) = 4T(n/2) + cn^2$$
$$= 4.\left\{4.T\left(\frac{n}{2^2}\right) + c.\left(\frac{n}{2}\right)^2\right\} + cn^2.$$
$$= 4^2.T\left(\frac{n}{2^2}\right) + 2cn^2.$$
$$\vdots$$
$$= 4^k.T\left(\frac{n}{2^k}\right) + kcn^2.$$

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

$$\therefore T(n) = 4^{\log n}.T(1) + \log n.cn^2.$$
$$= 2^{\log n^2} + cn^2 \log n.$$
$$= n^2 + cn^2 \log n.$$
$$\approx O(n^2 \log n).$$

P.T.O



$$\therefore T(n) = cn^2 \cdot \log n$$
$$= O(n^2 \log n).$$

~~Ans. 2) T(n) =~~

<u>Master Theorem</u>

$$a = 4,\ b = 2,\ f(n) = cn^2.$$
$$n^{\log_b a} = n^{\log_2 4} = n^2.$$

<u>Case-II</u> : Θ

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

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

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

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

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

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

$$\vdots$$

$$= 2^k T\left(\frac{n}{2^k}\right) + n\sum_{i=1}^{k} \frac{1}{\lg\left(\frac{n}{2^i}\right)}$$

$$\sum_{i=1}^{\infty} \frac{1}{\lg n - i}$$

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

$$\therefore T(n) = n.1 + n.\sum_{i=1}^{k} \frac{1}{\lg n/2^i}$$

$$\sum_{i=1}^{\infty} \frac{1}{\log n - i}$$

PTO

Ans. 3) $T(n) = 3T(n/2) + 2n^{1.5}$.

$$= 3\left\{3T(n/2^2) + 2\cdot\left(\frac{n}{2}\right)^{1.5}\right\} + 2n^{1.5}$$

$$= 3^2 T(n/2^2) + 6\cdot\left(\frac{n}{2}\right)^{1.5} + 2n^{1.5}$$

$$= 3^2\left\{3T(n/2^3) + 2\left(\frac{n}{2^2}\right)^{1.5}\right\} + 3.2\cdot\left(\frac{n}{2}\right)^{1.5} + 2n^{1.5}$$

$$= 3^3 T(n/2^3) + 3^2.2\left(\frac{n}{2^2}\right)^{1.5} + 3.2\left(\frac{n}{2}\right)^{1.5} + 2.n^{1.5}$$

⋮

$$= 3^k.T(n/2^k) + \sum_{i=0}^{k-1} 3^i.2\left(\frac{n}{2^i}\right)^{1.5}$$

$$= 3^k T(n/2^k) + n^{3/2}\sum_{i=0}^{k-1}\frac{3^i}{2^{i/2}}.$$

$n/2^k = 1 \Rightarrow k = \log n$. $\quad 2^k = n$

$\therefore 3^k > n$

$$\therefore T(n) > n.1 + n^{3/2}\sum_{i=0}^{\infty}\frac{3^i}{2^{i/2}}.$$

$$> n + n^{3/2}.\frac{1}{1-3/\sqrt{2}}$$

$\therefore T(n) = \Omega(n^{3/2})$.

14.3.24

Tower of Hanoi

$\cancel{no}$ $\text{move}(n, X, Z, Y) = \begin{cases} \text{move}(n-1, X, Y, Z) \\ \text{move } n^{th} \text{ disk from peg } X \text{ to peg } Z \\ \text{move}(n-1, Y, Z, X). \end{cases}$

X → source, Z → dest., Y → intermediate

If n=5

4, X, Y, Z move 5th disk X→Z

3, X, Z, Y

2, X, Y, Z

1, X, Z, Y.

Complexity: –

$T(n) = c$, if $n=1$ → Just ~~move~~ transfer 1 disk from source to dest.

$= T(n-1) + T(n-1) + c$, if $n>1$.

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

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

$= 2^2 T(n-2) + \cancel{3c}\ (1+2)c$

$= 2^3 T(n-3) + (1+2+2^2)c.$

⋮

$= \cancel{2^k(1} \; 2^k \cdot T(n-k) + c(1+2+2^2+ \dots + 2^{k-1} + \cancel{2^k})$

P.T.O

$n - k = 1$

$\Rightarrow k = n - 1$

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

$+ C\left(1 + 2 + 2^2 + \dots + 2^{n-2}\right).$

$\approx O(2^n).$

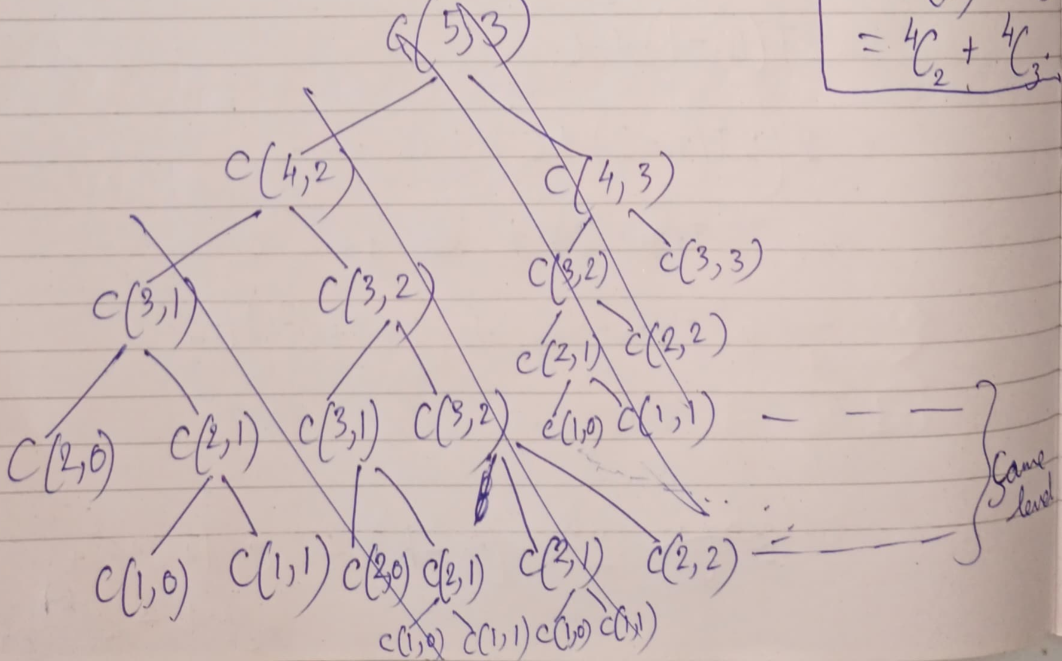
Binomial Coefficient

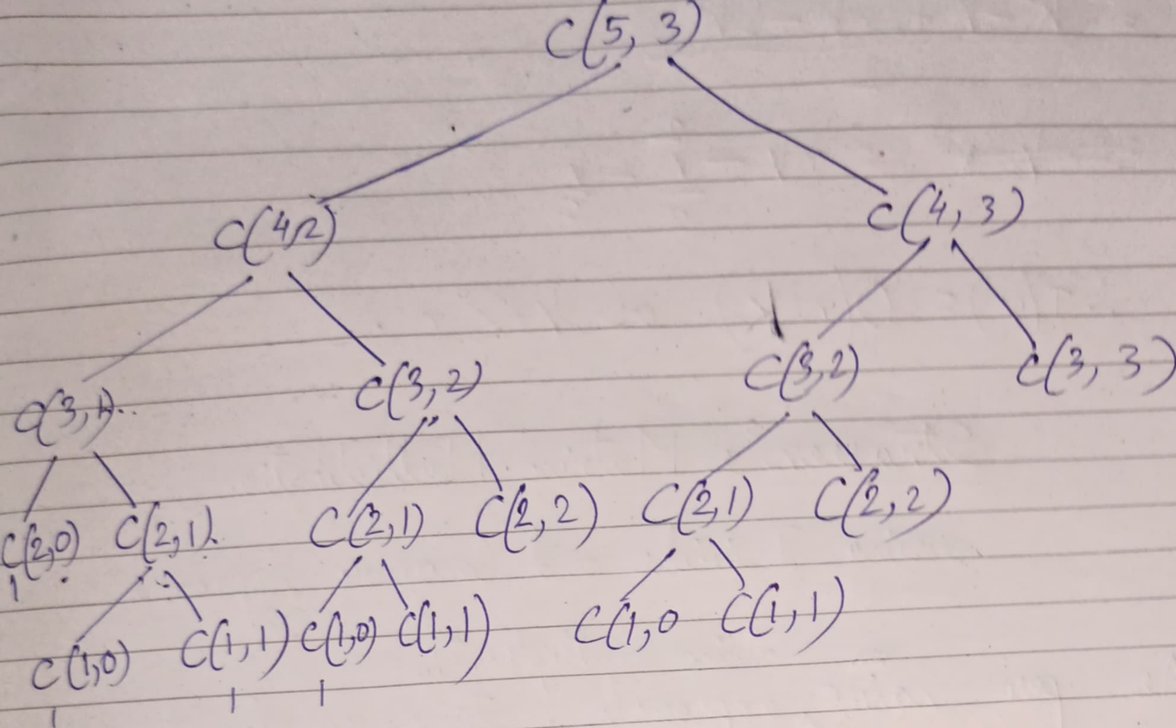
$C(n, k) = C\binom{n-1}{k-1} + C\binom{n-1}{k}$

$= C(n-1, k-1) + C(n-1, k)$, where $n > k > 0$

$C(n, 0) = C(n, n) = 1.$

$$\boxed{\begin{aligned} {}^5C_3 &= C(5,3) \\ &= C\binom{4}{2} + C\binom{4}{3} \\ &= {}^4C_2 + {}^4C_3. \end{aligned}}$$





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

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

$$= \sum_{i=1}^{k}\sum_{j=1}^{i-1} 1 + \sum_{i=k+1}^{n}\sum_{j=1}^{k} 1$$

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

$$= (1+2+3+\dots+(k-1)) + k(n-k)$$

$$= \frac{k(k-1)}{2} + k(n-k).$$

$$= \frac{k^2-k}{2} + nk - k^2$$

H/W for lab (Add at the end of the manual. Assignment 10) :-

1. Linear Search
2. Factorial
3. Fibonacci
4. Finding Binomial Coefficient
5. Tower of Hanoi

P.T.O

$$= \frac{k^2 - k + 2nk - 2k^2}{2}$$

$$= nk - \frac{k^2}{2} - \frac{k}{2}$$

$\therefore T(n) = O(nk)$ as $k \ll n$.

Strassen's Matrix Multiplication using Divide and Conquer Approach

- As per naive matrix multiplication approach, time complexity is $O(n^3)$.
- In Strassen's Matrix Multiplication using Divide and Conquer Approach, it has been improved to $O(n^{2.81})$.
- Order of the matrix must be in the order of 2.
- Best Case → 1 element.

→ Dimension of input matrix is $n \times n$.