Unit - 2

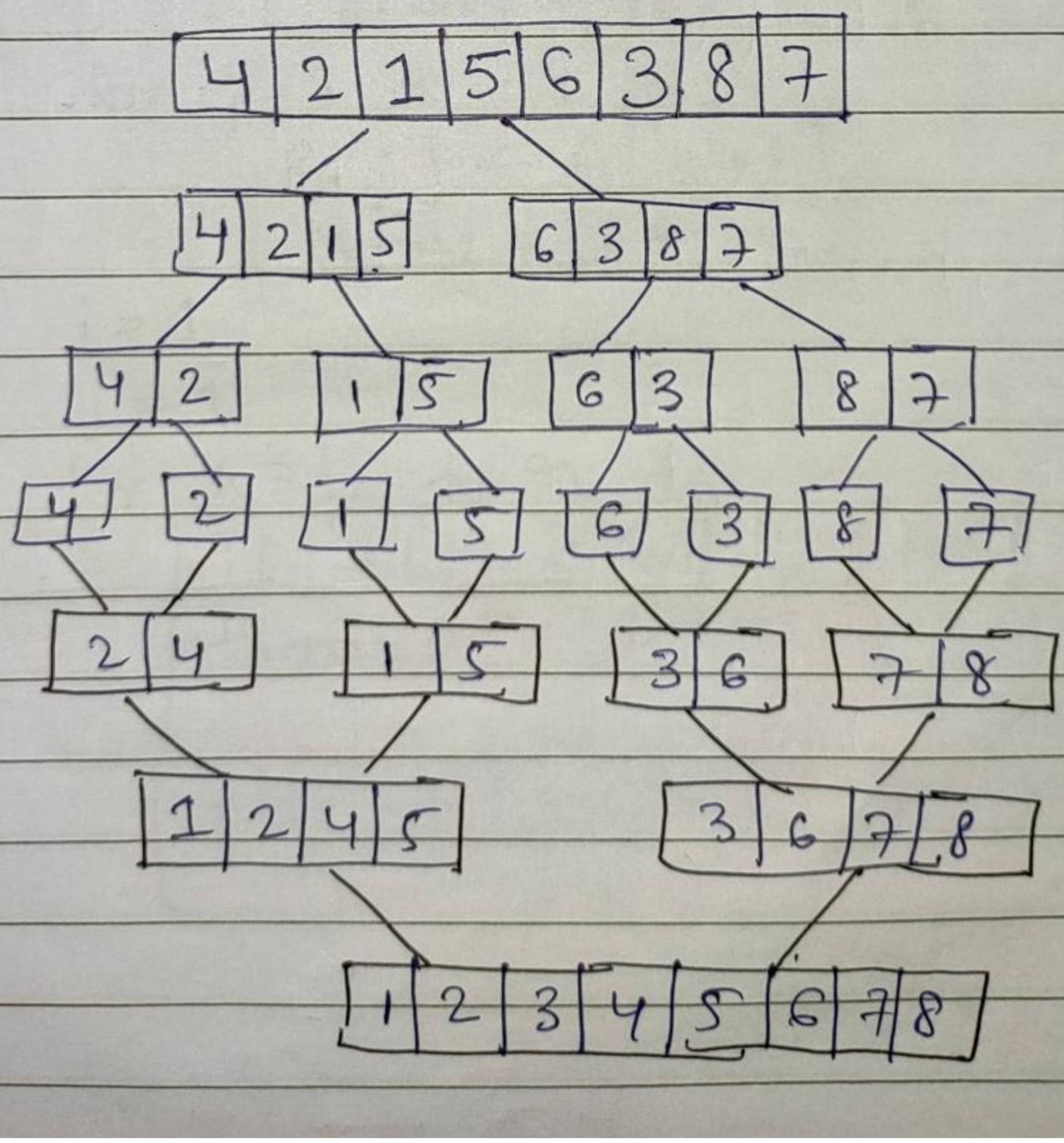
Divide & Conquer

Already studied in Unit-1, this algorithm design technique involves :-

1) Divide :- Break the problem into several subproblem
2) Conquer :- Solve subproblem recursively.
3) Combine :- Combine the solution of the subproblem to creat solution of original problem.

Example :- 1) Merge Sort
2) Quick Sort
3) Binary Search.

Merge Sort :-



Algorithm of Merge Sort :-

```
MERGE SORT (A, p, r)
    if p < r
    then q ← (p+r)/2
    MERGE SORT (A, p, q)
    MERGE SORT (A, q+1, r)
    MERGE (A, p, q, r)
    end of if
end MERGE SORT
```

```
MERGE (A, p, q, r)
1.      n1 ← q-p+1
2.      n2 ← r-q
3.      Create arrays L[1 -- n1+1] & R[1 -- n2+1]
4.      for i = 1 to n1 do
5.          L[i] ← A[p+i-1]
6.      L[n1+1] ← ∞
7.      for j = 1 to n2 do
8.          R[j] ← A[q+j]
9.      R[n2+1] ← ∞
10.     i = 1
11.     j = 1
12.     for k = p to r do
13.         if L[i] ≤ R[j]
14.         then A[k] = L[i]
15.             i ← i+1
16.         else A[k] ← R[j]
17.             j ← j+1
```

Date
Page

Recurrence Relation of Merge Sort :-

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

↳ Now, we can solve this by any method of our choice which we studied in unit-1.

Solution → $T(n) = O(n \log n)$

Quick Sort

Algorithm of Quick Sort :-

```
QUICK_SORT (A, p, r) {
1.      If (p<r)
2.      Then q ← PARTITION(A, p, r).
3.      QUICKSORT (A, p, q-1)
4.      QUICKSORT (A, q+1, r).
    }

PARTITION (A, p, r) {
1.          n ← A[r]
2.          i ← p-1
3.          for j ← p to n-1 do {
4.              if A[j] ≤ n
5.              Then i ← i+1
6.                  exchange A[i] ↔ A[j]
                }
7.          exchange A[i+1] ↔ A[r].
8.          return i+1
    }
```

Here, p → index of 1st element

r → index of last element

Example:- Perform Quick Sort on

9	2	7	5	1	4	10	6

Ans →

9	2	7	5	1	4	10	6
1	2	3	4	5	6	7	8
↑							↑
p							r

q ← Partition(A, 1, 8).

n ← A[8] i.e. 6.

i ← 0

j ← 1 to 7

j = 1 :-

A[1] ≤ n (false)

j = 2 :-

A[2] ≤ n (2 ≤ 6) (True)

i = 0 + 1 = 1

A[1] ⟷ A[2]

2	9	7	5	1	4	10	6
1	2	3	4	5	6	7	8

j = 3 :-

A[3] ≤ n

j = 4 :-

A[4] ≤ n

i = 1 + 1 = 2

A[2] ⟷ A[4]

2	5	7	9	1	4	10	6
1	2	3	4	5	6	7	8

j = 5 :-

A[5] ≤ n

i = 2 + 1 = 3

A[3] ⟷ A[5]

2	5	1	9	7	4	10	6
1	2	3	4	5	6	7	8

$j = 6$:-

$A[6] \leq n$

$i = 3+1 = 4$

$A[4] \longleftrightarrow A[6]$

2	5	1	4	7	9	10	6

$j = 7$:-

$A[7] \leq n$

$A[4+1] \longleftrightarrow A[8]$

2	5	1	4	6	9	10	7

return 5

$q \leftarrow 5$

Quicksort (A, 1, 4)

2	5	1	4
1	2	3	4

p ↑ r ↑

$q \leftarrow$ Partition (A, 1, 4).

$n \leftarrow A[4]$ i.e 4.

$i \leftarrow 0$

$j \leftarrow 1$ to 3.

$j = 1$:-

$A[1] \leq n$.

$i = 0+1 = 1$

$A[1] \longleftrightarrow A[1]$

2	5	1	4
1	2	3	4

j = 2 :-

A[2] ≤ n

j = 3 :-

A[3] ≤ n

i = 1 + 1 = 2

A[2] ↔ A[3]

2	1	5	4

A[2+1] ↔ A[4]

2	1	4	5

return 3.

q ← 3.

Quicksort (A, 1, 2).

2	1
1	2

q ← Partition (A, 1, 2).

n ← A[2] i.e. 2

i ← 0

j ← 1

j = 1 :-

A[1] ≤ n (false)

i ← 0 + 1 = 1

A[1] ↔ A[1].

2	1

A[0+1] ↔ A[2].

1	2

return 1.

QuickSort (A, 6, 8) :-

9	10	7

q ← partition (A, 6, 8)

n ← A[8] . i.e. 7

i ← 5

j ← 6 to 7

j = 6 :-

A[6] ≤ n

j = 7 :

A[7] ≤ n

A[5+1] ⟷ A[8]

7	10	9

return 6.

QuickSort (A, 7, 8).

10	9

q ← partition (A, 7, 8)

n ← A[8] ← 89

i ← 6

j ← 7

j = 7 :-

A[7] ≤ n

A[6+1] ⟷ A[8]

9	10

So, after combining all these we get

1	2	4	5	6	7	9	10

Analysis of Quick Sort :-

① Best Case :-

Best case occur where there is equal partition in two parts.

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

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

So, Case 2 will apply.

$$\Rightarrow T(n) = \theta(n^{\log_2 2} \cdot \log n)$$

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

② Worst Case :-

Worst case occur when pivot is at ~~uneven~~ unbalanced partition.

$$T(n) = T(0) + T(n-1) + cn$$

$$T(n) = T(n-1) + T(0) + cn.$$

$$T(n) = [T(n-2) + T(0) + c(n-1)] + T(0) + cn$$

$$T(n) = T(n-2) + 2T(0) + c(n-1) + cn$$

$$T(n) = [T(n-3) + T(0) + c(n-2)] + 2T(0) + c(n-1) + cn$$

$$T(n) = T(n-3) + 3T(0) + c(n-2) + c(n-1) + cn.$$

$$T(n) = T(n-3) + 3T(0) + cn((n-2) + (n-1) + n)$$

$$T(n-k) = T(n-k) + kT(0) + c\left[(n-k-1) + (n-(k-2)) + (n-(k-3)) \cdots n\right]$$

$$\Rightarrow T(n) = T(1) + kT(0) + c(2+3+4+\cdots n)$$

$$= 1 + k\cdot 1 + c[1+2+3+4+\cdots n-1]$$

$$= 1 + (n-1) + c\left[\frac{n(n+1)}{2} - 1\right]$$

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

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

Or

$$T(n) = T(n-k) + kT(0) + kcn - kc.$$

$$= T(1) + k\cdot 1 + c(kn-k)$$

$$= T(1) + (n-1) + c[(n-1)n - (n-1)]$$

$$= T(1) + (n-1) + c[n^2 - n - n + 1]$$

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

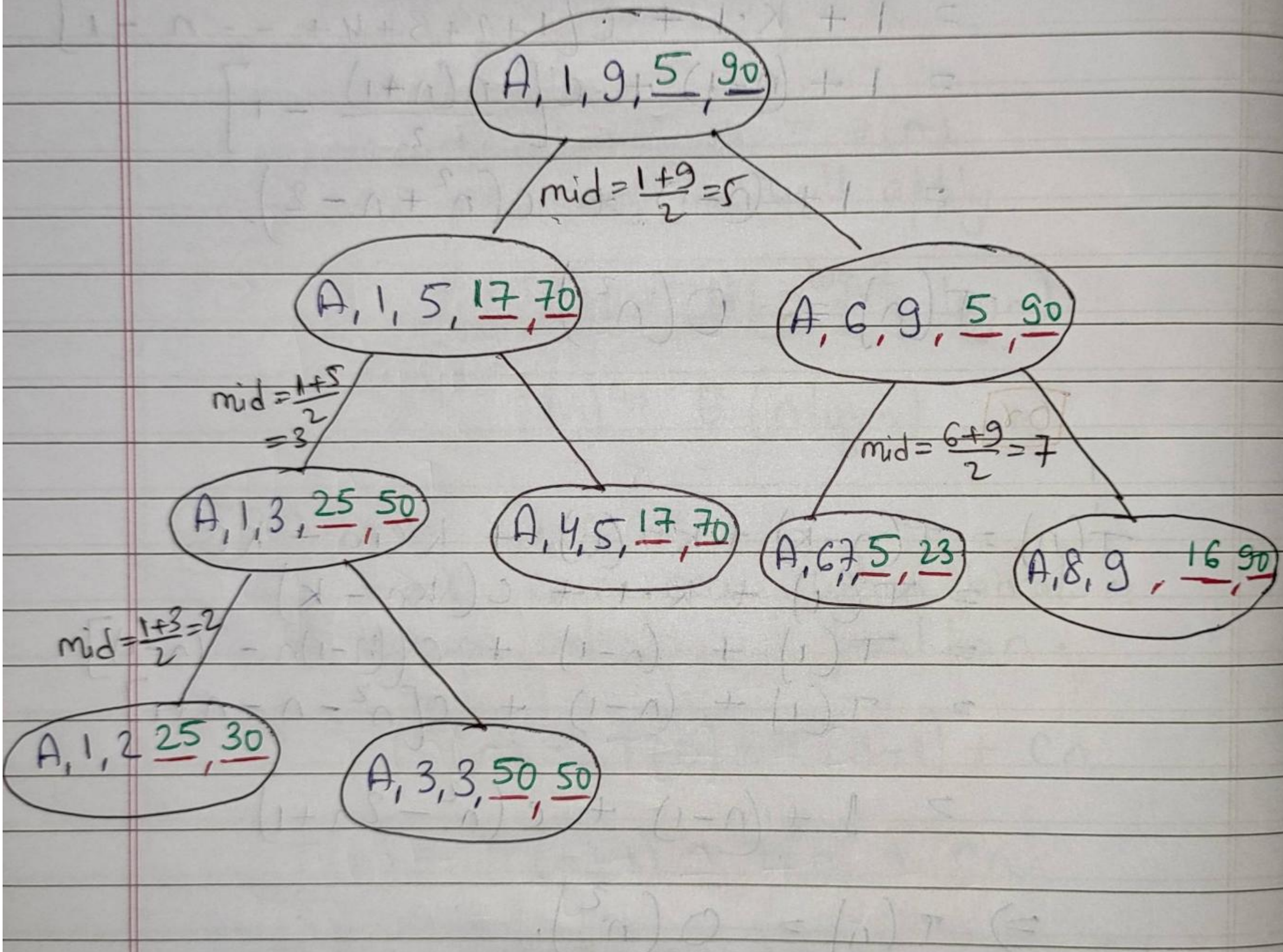
$$\Rightarrow T(n) = O(n^2).$$

(3) Average Case :-

$$O(n \log n)$$

Finding Maximum & Minimum using Divide & Conquer

$A = [25, 30, 50, 70, 17, 23, 5, 90, 16]$
$\quad\quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9$

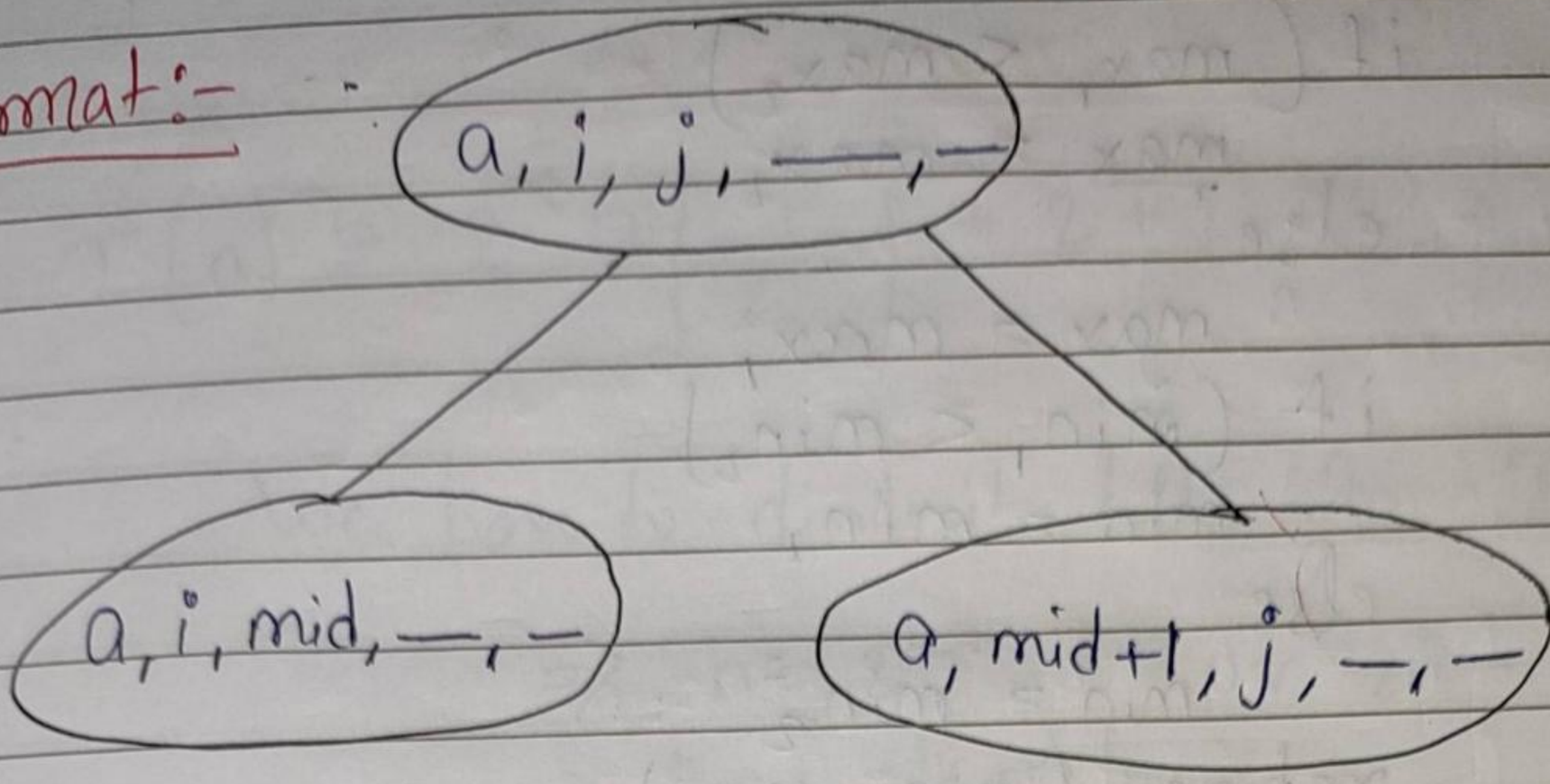


Format → A, i, j, min, max

Array, Start index, end index, To be filled

* $Mid = \left\lfloor \frac{i+j}{2} \right\rfloor$

Format:-



Algorithm :-

```
DACmaxmin (arr, i, j) {
    if (i == j)
    {
        max = min = arr[i];
        return (min, max)
    }
    if (i = j - 1) {
        if (arr[i] ≤ arr[j])
            max = arr[j], min = arr[i]
        else
            max = arr[i], min = arr[j].
        return (min, max).
    }
    else {
        mid = floor (i+j/2).
        max1, min1 = DACmaxmin (a, i, mid)
        max2, min2 = DACmaxmin (a, mid+1, j)
```

```
if (max1 < max2)
    max = max2
else
    max = max1
if (min1 < min2)
    min = min1
else
    min = min2.
return (min, max).
}
}
```

Analysis :-

If $n = 1$ $\quad T(n) = 0$

If $n = 2$ $\quad T(n) = 1$

If $n > 2$ $\quad T(n) = 2T\left(\frac{n}{2}\right) + 2$

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

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

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

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

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

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

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

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

We have to dinde it till $\frac{n}{2^k} = 2$

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

$\Rightarrow \log n = (k+1)\log_2 2$

$\Rightarrow k = \log_2 n - 1$.

$$T(n) = \frac{n}{2} T\left[2\right] + \sum_{k=1}^{\infty} 2^k.$$

$$= \frac{n}{2} T[2] + \frac{2(2^k - 1)}{2-1}$$

$$= \frac{n}{2} T[2] + \frac{2^1 (2^{\log_2 n - 1} - 1)}{1}$$

$$= \frac{n}{2} + 2^{\log_2 n} - 2$$

$$= \frac{n}{2} + n^{\log_2 2} - 2$$

$$= \frac{n}{2} + n - 2$$

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

Or

$$T(n) = \frac{n}{2} T[2] + \frac{2\left[\frac{n}{2} - 1\right]}{2-1}$$

$$= \frac{n}{2} \cdot 1 + n - 2 \quad = \frac{n}{2} + n - 2$$

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

Strassen's Matrix Multiplication :-

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

Normal :-

$$c_{11} = a_{11} * b_{11} + a_{12} * b_{21}$$
$$c_{12} = a_{11} * b_{12} + a_{12} * b_{22}$$
$$c_{21} = a_{21} * b_{11} + a_{22} * b_{21}$$
$$c_{22} = a_{21} * b_{12} + a_{22} * b_{22}$$

$\Rightarrow$ 8 Multiplication & 4 addition.

Complexity = $O(n^3)$.

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

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix}$$

4x4 4x4

divided into 2x2

N x N matrix is divided into $\frac{N}{2} \times \frac{N}{2}$ Matrix.

In 2x2 → each element is considered as matrix of 1x1.

As we can see, normal matrix multiplication require 8 Multiplication & 4 addition. ad andd Multiplication is one which greatly influence the Complexity.

Strassen's Came with method, such that it can be done in 7 Multiplication.

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \times \begin{bmatrix} E & F \\ G & H \end{bmatrix}$$

$$P_1 = A\ (F-H)$$
$$P_2 = H\ (A+B)$$
$$P_3 = E\ (C+D)$$
$$P_4 = D\ (G-E)$$
$$P_5 = (A+D) * (E+H)$$
$$P_6 = (B-D) * (G+H)$$
$$P_7 = (A-C) * (E+F).$$

$$C = \begin{bmatrix} P_6+P_5+P_4-P_2 & P_1+P_2 \\ P_3+P_4 & P_1+P_5-P_3-P_7 \end{bmatrix}$$

Trick To Remember :-

AHED → $P_1\ P_2\ P_3\ P_4$

P_5 → Go Diagonals

P_6 → Last Column Last Row

P_7 → First Column first Row.

* Colum → -ve, Rows = +ve

Q Find value of $\begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} \times \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix}$

Sol→ $X = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$ $Y = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix}$

On Comparing with $\begin{bmatrix} A & B \\ C & D \end{bmatrix} \times \begin{bmatrix} E & F \\ G & H \end{bmatrix}$

$A = 2,\ B = 2,\ C = 2,\ D = 2$
$E = 2,\ F = 3,\ G = 4,\ H = 5$

$P_1 = A(F-H) = 2(3-5) = -4$
$P_2 = H(A+B) = 5(2+2) = 20$
$P_3 = E(C+D) = 2(2+2) = 8$
$P_4 = D(G-E) = 2(4-2) = 4.$
$P_5 = (A+D)(E+H) = 4 \times 7 = 28$

$P_6 = (B-D) \times (G+H) = 0 \times 9 = 0$
$P_7 = (A-C) * (E+F) = 0 \times 5 = 0$

$$C = \begin{bmatrix} P_6+P_5+P_4-P_2 & P_1+P_2 \\ P_3+P_4 & P_1+P_5-P_3-P_7 \end{bmatrix}$$

$$= \begin{bmatrix} 0+28+4-20 & 20-4 \\ 8+4 & -4+28-8-0 \end{bmatrix}$$

$$= \begin{bmatrix} 12 & 16 \\ 12 & 16 \end{bmatrix}$$ Ans

Analysis of Strassen's Matrix Multiplication :-

$$T(n) = \begin{cases} 1 & \text{if } n \le 2 \\ 7T\left(\frac{n}{2}\right) + an^2 & \text{if } n > 2 \end{cases}$$

$$T(n) = 7T\left(\frac{n}{2}\right) + an^2$$

$$\Rightarrow T(n) = 7\left[7T\left(\frac{n}{4}\right) + a\left(\frac{n}{2}\right)^2\right] + an^2$$

$$T(n) = 7^2 T\left[\frac{n}{2^2}\right] + \left(\frac{7}{4}\right)an^2 + an^2$$

$$\Rightarrow T(n) = 7^2\left[7T\left(\frac{n}{2^3}\right) + a\frac{n^2}{16}\right] + \left(\frac{7}{4}\right)an^2 + an^2$$

$$T(n) = 7^3 T\left[\frac{n}{2^3}\right] + \frac{7^2}{4^2}an^2 + \left(\frac{7}{4}\right)an^2 + an^2$$

⋮

$$T(n) = 7^k T\left[\frac{n}{2^k}\right] + \left(\frac{7}{4}\right)^{k-1}an^2 + \left(\frac{7}{4}\right)^{k-2} + \cdots + an^2$$

$$\Rightarrow T(n) = 7^k T\left[\frac{n}{2^k}\right] + an^2 + \frac{7}{4}an^2 + \left(\frac{7}{4}\right)^2 + an^2 + \cdots + \left(\frac{7}{4}\right)^{k-1}an^2$$

$$= 7^k T\left[\frac{n}{2^k}\right] + an^2\left[1 + \frac{7}{4} + \left(\frac{7}{4}\right)^2 + \cdots + \left(\frac{7}{4}\right)^{k-1}\right]$$

$$= 7^k T\left[\frac{n}{2^k}\right] + an^2\left[1 \cdot \frac{\left[\left(\frac{7}{4}\right)^{k-1} - 1\right]}{\left[\frac{7}{4} - 1\right]}\right]$$

$$= 7^k T\left[\frac{n}{2^k}\right] + an^2\left[\frac{\left(\frac{7}{4}\right)^k / \frac{7}{4} - 1}{\frac{7}{4} - 1}\right]$$

$$= 7^k T\left[\frac{n}{2^k}\right] + an^2\left(\frac{7}{4}\right)^k \quad \text{(on neglecting constants)}$$

We want to reduce it to $T(1)$

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

$$\Rightarrow n = 2^k$$

$$\Rightarrow k = \log_2 n$$

$$7^k T(1) + an^2\left(\frac{7}{4}\right)^k$$

$$\Rightarrow 7^{\log_2 n} + an^2 \cdot \left(\frac{7}{4}\right)^{\log_2 n}$$

$$\Rightarrow 7^{\log_2 n} + an^{\log_2 4} \cdot \left(\frac{7}{4}\right)^{\log_2 n}$$

$$\Rightarrow n^{\log_2 7} + an^{\log_2 4} \cdot n^{\log_2 \frac{7}{4}}$$

$$\Rightarrow n^{\log_2 7} + an^{\log_2 4 + \log_2 7 - \log_2 4}$$

$$\Rightarrow n^{\log_2 7} + an^{\log_2 7}$$

$$T(n) \Rightarrow O\left(n^{\log_2 7}\right)$$

$$T(n) = O\left(n^{2.81}\right)$$

Slightly lower than n^3

Page

Q Perform Matrix Multiplication using Strassen's multiplication on $\begin{bmatrix} 3 & 2 \\ 1 & 5 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$.

Sol → On Comparing with $\begin{bmatrix} A & B \\ C & D \end{bmatrix} \times \begin{bmatrix} E & F \\ G & H \end{bmatrix}$

$A=3$, $B=2$, $C=1$, $D=5$

$E=1$, $F=2$, $G=3$, $H=4$.

$P_1 = A(F-H) = 3 \times -2 = -6$

$P_2 = H(A+B) = 4 \times 5 = 20$

$P_3 = E(C+D) = 1 \times 6 = 6$

$P_4 = D(G-E) = 5 \times 2 = 10$

$P_5 = (A+D)(E+H) = 8 \times 5 = 40$

$P_6 = (B-D)(G+H) = -3 \times 7 = -21$

$P_7 = (A-C)(E+F) = 2 \times 3 = 6$.

$$C = \begin{bmatrix} P_6+P_5+P_4-P_2 & P_1+P_2 \\ P_3+P_4 & P_1+P_5-P_3-P_7 \end{bmatrix}$$

$$= \begin{bmatrix} -21+40+10-20 & 20-6 \\ 6+10 & -6+40-6-6 \end{bmatrix}$$

$$C = \begin{bmatrix} 9 & 14 \\ 16 & 22 \end{bmatrix}$$ Ans

$$= \begin{bmatrix} 9 & 14 \\ 16 & 22 \end{bmatrix}$$ Ans

Largest Subarray Sum :-

Algorithm :-

Divide the array into two half.
Return the maximum of following three :-
1) Maximum Subarray sum of in left half.
2) Maximum Subarray sum in right half.
3) Maximum subarray sum such that the subarray crosses the mid-point.

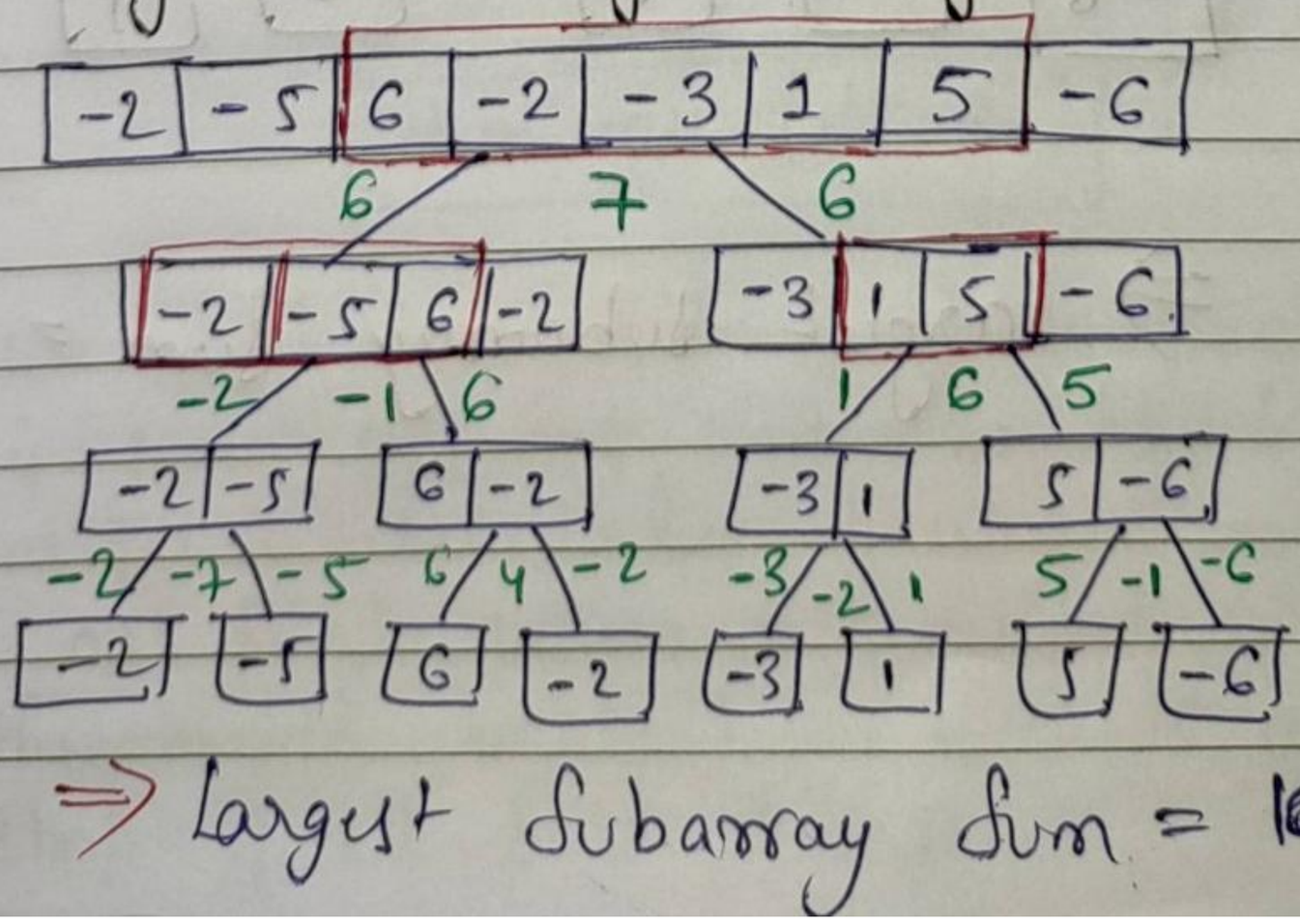
Analysis :-

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

We can solve it in any method we studied

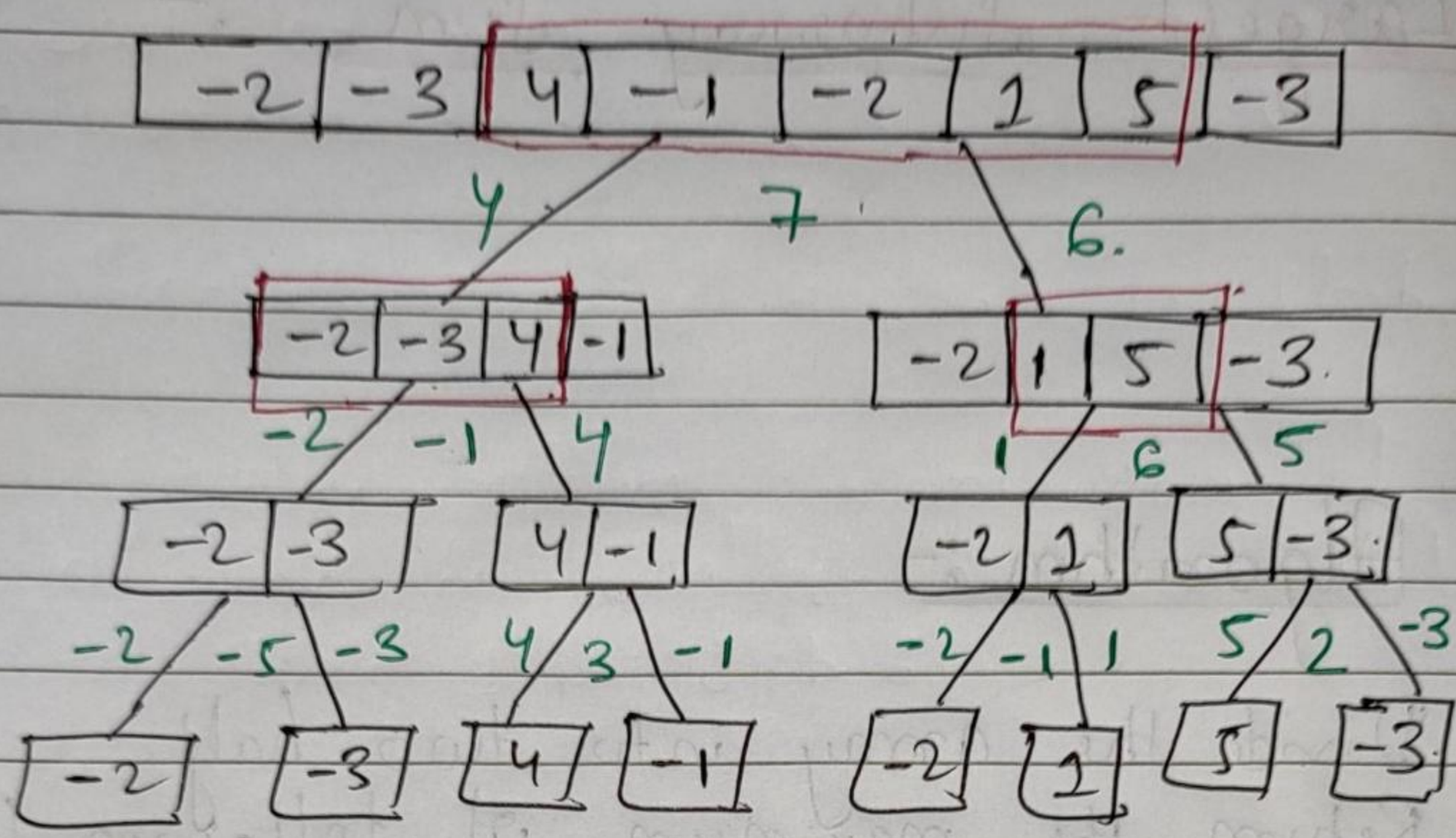
$$\Rightarrow T(n) = O(n \log n)$$

Q Find largest subarray sum of -2, -5, 6, -2, -3, 1, 5, -6



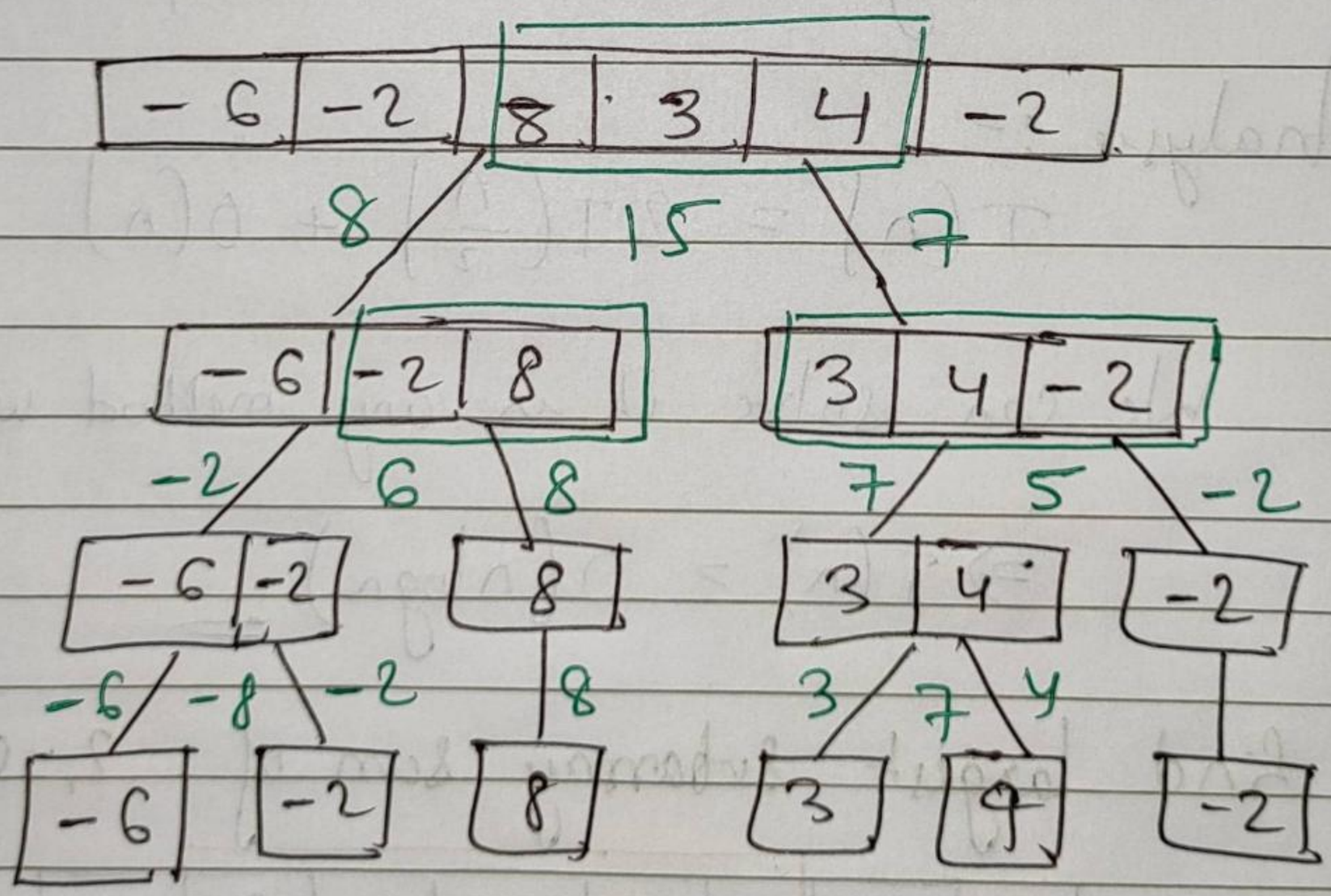
$\Rightarrow$ Largest Subarray Sum = 7

Q



⇒ largest subarray sum = 7.

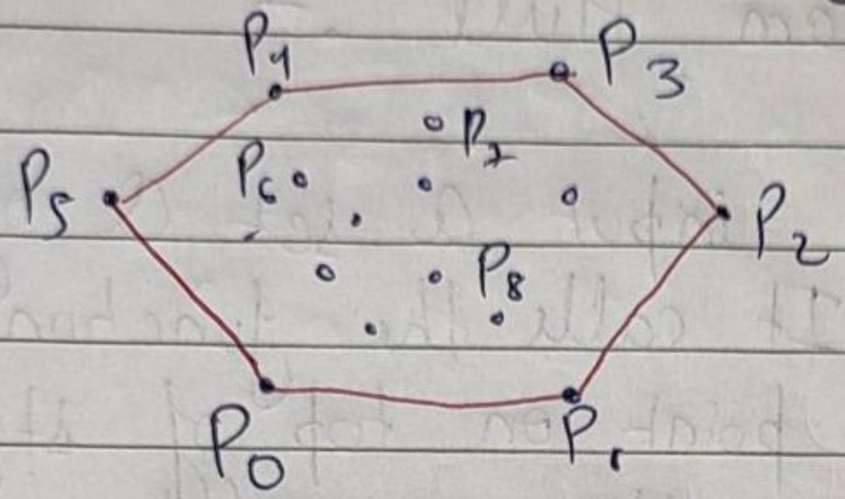
Q.



⇒ largest subarray sum = 15

Convex Hull

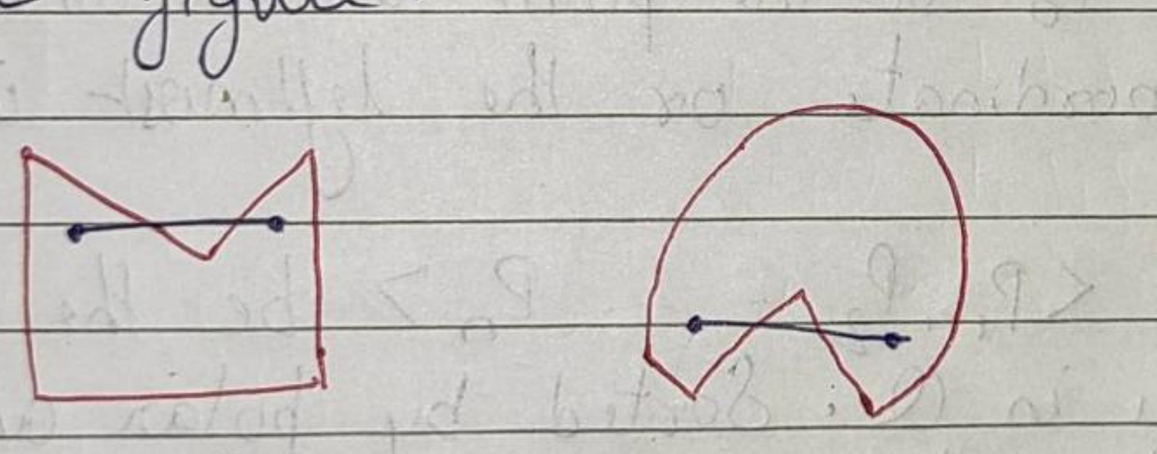
Convex Hull of set 'Q' of points is the smallest convex polygon 'P' for which each point of 'Q' is either on Boundary of 'P' or inside it.



A figure is convex if every line segment drawn between any two point inside the figure lies initially inside the figure.

A figure that is not convex is called a concave figure.

Concave →



Convex →

Properties :- 1) If we connect two points then it should not come outside (ie. remain inside)

2) Vertex will be convex. (i.e interior angle ≤ 180°)

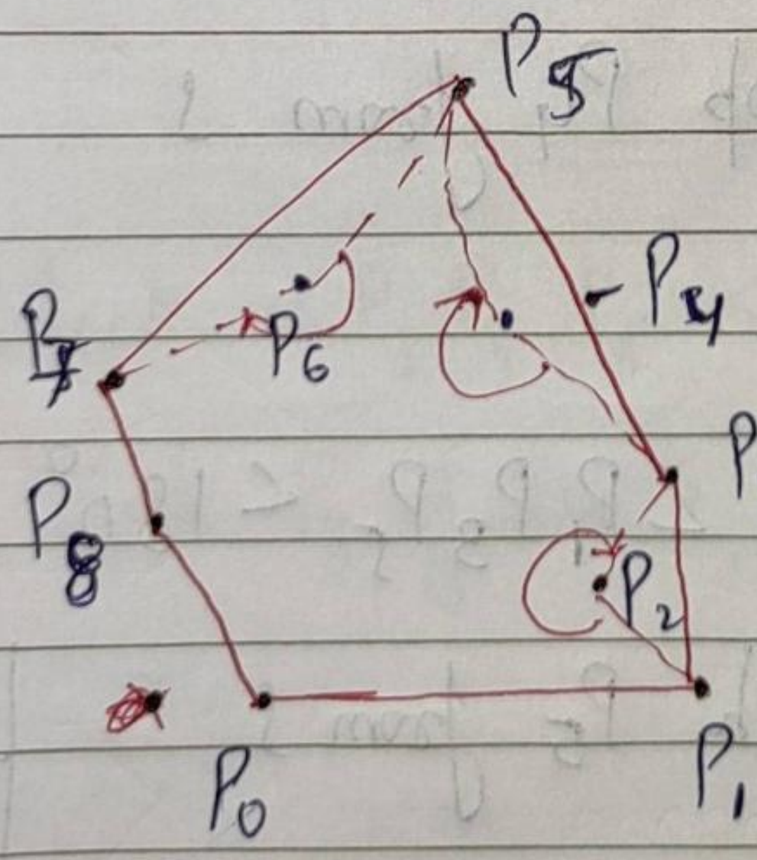
7. For $i = 3$ to n — $O(n-2) = O(n)$
{
do while the angle formed by points NEXT_TO_TOP(S), TOP(S) and P_i, makes a non-left turn. — $O(n)$
{
Pop(S) } — $O(n)$
Push(S, P_i)
}
return S. — $O(1)$

Aggregate Analysis :-

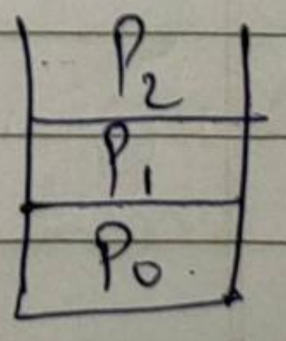
$T(n) = O(n) + O(n \log n) + O(1) + O(1) + O(1) + O(1) + O(n) + O(n) + O(1)$.

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

Q.

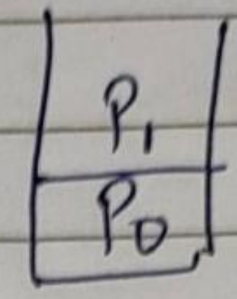


Ans → Push P_0, P_1, P_2 in S



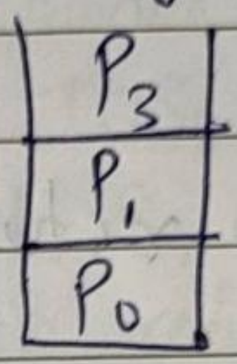
$P_i = P_3$:- check $\angle P_1 P_2 P_3 > 180^\circ$ (Right turn)

Pop P_2 from S.



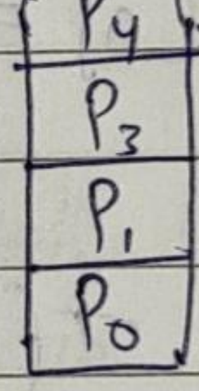
check $\angle P_0 P_1 P_3 < 180^\circ$ → left turn

Push P_3 in S



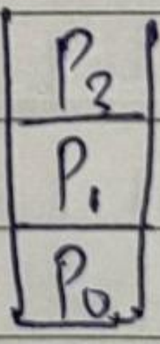
$P_i = P_4$:- check $\angle P_1 P_3 P_4 < 180^\circ$ → left turn

Push P_4 in S



$P_i = P_5$:- check $\angle P_3 P_4 P_5 > 180^\circ$ → Right turn

Pop P_4 from S

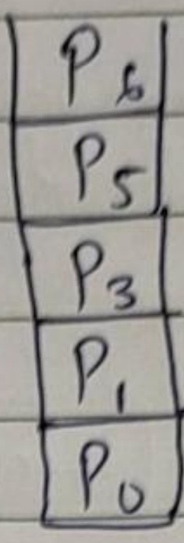


i check $\angle P_1 P_3 P_5 < 180^\circ$ → left turn.

Push P_5 from S

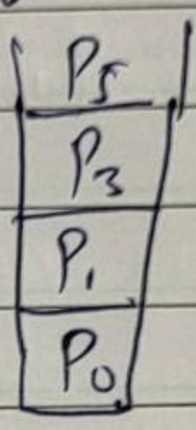
$P_i = P_6$:- Check angle $P_3 P_5 P_6 < 180° \rightarrow$ left turn

Push P_6 from S



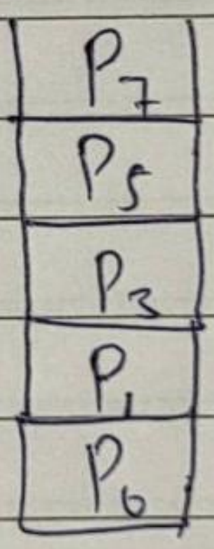
$P_i = P_7$:- check angle $P_5 P_6 P_7 > 180° \rightarrow$ Right turn

Pop P_6 from S



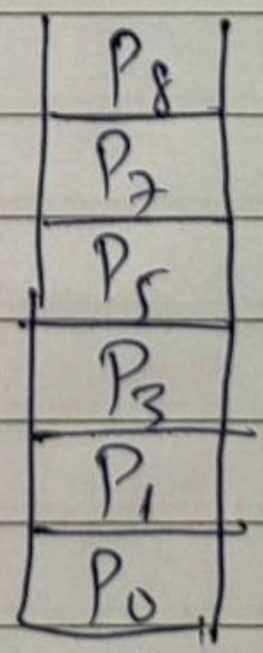
Check $\angle P_3 P_5 P_7 < 180° \rightarrow$ Left turn

Push P_7 from S



$P_i = P_8$:- Check $\angle P_5 P_7 P_8 < 180° \rightarrow$ Left turn

Push P_8 from S



Check angle of $P_7 P_8 P_0 < 180° \rightarrow$ Left turn.

Quick Hull Algorithm → based on Divide & Conquer.

Input = a set S of n Points. Assume that there are at least 2 points in the input set S of points.

Quick Hull (S)
{
1. Convex Hull = { }
   lowest x & highest x
2. Find left and right most points say A & B to convex Hull.
3. segment AB divides the remaining (n-2) points into 2 Groups.
   S_1 and S_2 where S_1 are points in S that are on the left side of the oriented line from A to B and S_2 are points on Right side of oriented line from B to A.
4. Find-Hull (S_1, A, B)
5. Find-Hull (S_2,)
}

FindHull (S_K, P, Q)
{
// Find Points on convex Hull from the set S_K of Points that are on the right side of the oriented line from P to Q.
(1.) If S_K has no points then return
(2.)
(3.) From, the given set of points in S_K, Find Farthest point, say C, from segment PQ.
(4.) Add point C to convex hull at the location between P & Q.
(5.) Three points P, Q & C partition the set of points into S_0', S_1', S_2' where S_0' are the points inside

PQ and S_1' are to the left of PC and right of CQ.

(1) Find-Hull (S_1', P, C)

(2) Find-Hull (S_2', C, Q)

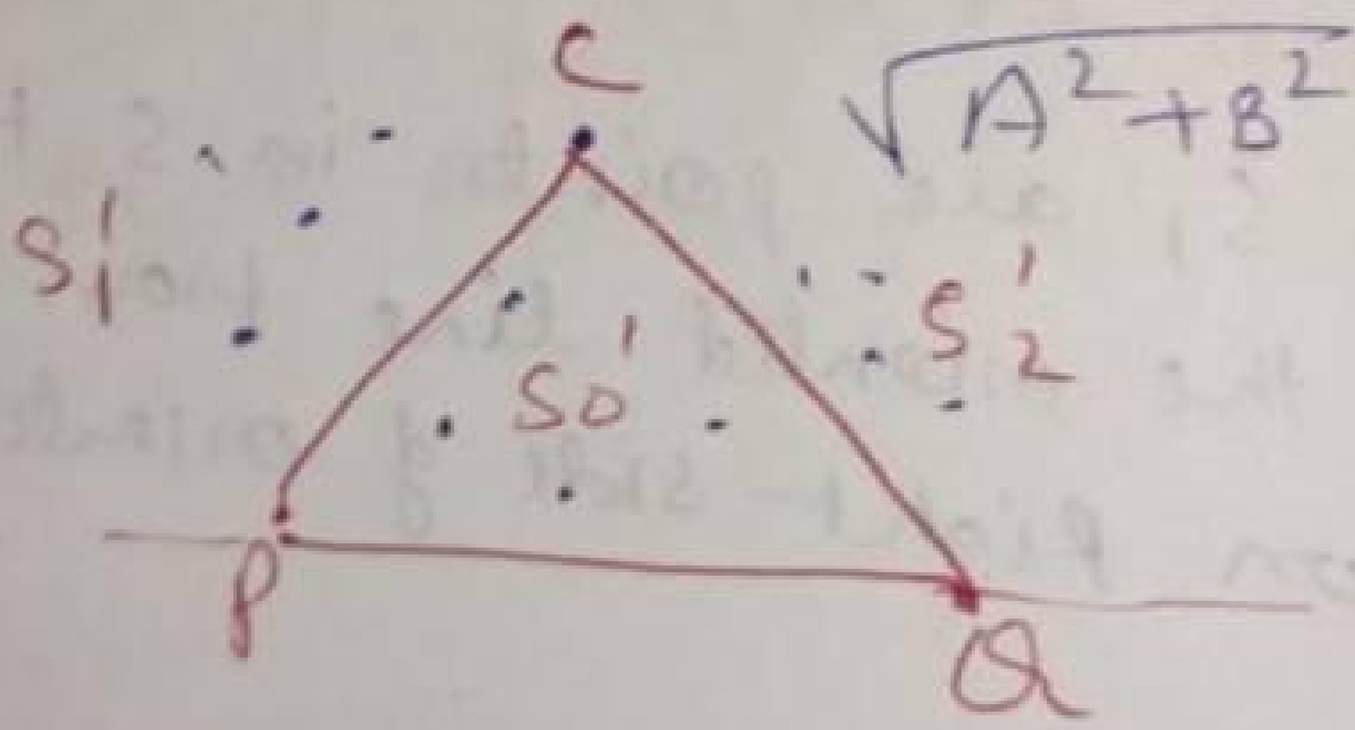
}

→ How we calculate Farthest point?

→ Formula is (m, n) is point.

$$Ax + By + C = 0$$

Farthest Point = $\frac{Am + Bn + C}{\sqrt{A^2 + B^2}}$ = distance



Merge Hull

This Algorithm works similar to merge sort in the following way -

1. Partition the given set of points into 2 sets S_1 & S_2.
2. Recursively Compute the Convex hulls of the two sets, say
   $CH(S_1)$ and $CH(S_2)$
3. Merge the two Small Convex Hull $CH(S_1)$ and $CH(S_2)$ to given the final Convex Hull $CH(S)$.

⇒ For step 1 compute the median on the basis of x - coordinate. This operation is $O(n)$.

→ For step 3

(1) First, find the lines that are upper tangent and lower tangent to the two hulls.

(2) Then, remove the points that are cut off (lying b/w the upper & lower tangents)

Finding tangent lines

Find the upper tangent in the following way -

(1) Start with the rightmost point of the left hull & the leftmost point of the right hull & join then assuming this line is the upper tangent.

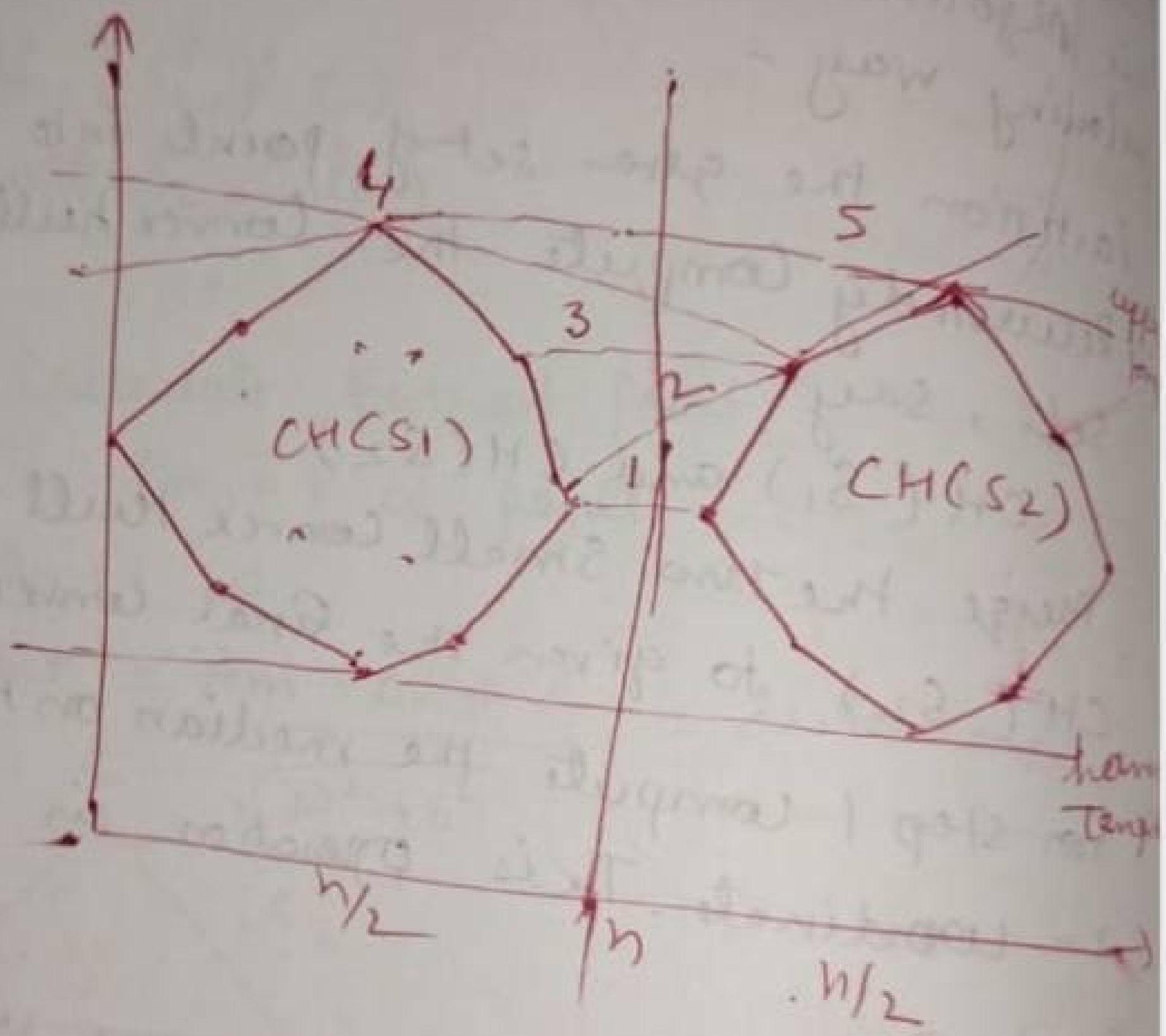
(2) While the line is not upper tangent to both the left & right halves do -
- If the line is not the upper Tangent to the left, move the next counter - clockwise point on the left convex hull and check.

⇒ Complexity

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n) + O(n) \rightarrow \text{for Finding upper \& lower Tangent}$$
(↑ For finding median)

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

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



4) Master's Method :-

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

$a \geq 1$, $b > 1$ are constant

Case 1 :- $n^{\log_b a - \epsilon} = f(n)$.

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

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

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

Case 3 :- $n^{\log_b a + \varepsilon} = f(n)$

$$\boxed{T(n) = \Theta(f(n))}$$

for Case 3 :-

Regulating Condition :-

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

where $c < 1$.

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

Sol → On Comparing it with

$aT\left(\frac{n}{b}\right) + f(n)$, we get

$a = 4, \quad b = 2, \quad f(n) = n$

$n^{\log_b a} = n^{\log_2 4} = n^2$

⇒ Case 1 will be applied

⇒ $T(n) = \Theta(n^{\log_2 4})$

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

Page

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

Sol → $a = 8, \quad b = 2, \quad f(n) = 3n^2$

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

$\Rightarrow$ Case #1 will be applied

$\Rightarrow T(n) = \theta\left(n^{\log_2 8}\right)$

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

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

Sol → $a = 4, \quad b = 2, \quad f(n) = n^2$

$$n^{\log_2 4} = n^2 = f(n)$$

$\Rightarrow$ Case 2 will be applied

$\Rightarrow T(n) = \theta(n^{\log_2 4} \cdot \log n)$

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

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

Sol → $a = 2, \quad b = 2, \quad f(n) = n$

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

$\Rightarrow$ Case 2 will be applied

$\Rightarrow T(n) = \theta(n^{\log_2 2} \cdot \log n) \Rightarrow T(n) = \theta(n \cdot \log n)$.

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

Sol → $a = 4,\ b = 2,\ f(n) = n^3$

$$n^{\log_b a} = n^{\log_2 4} = n^2$$

$\Rightarrow$ Case 3 will be applied

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

$$\Rightarrow 4 \cdot f\left(\frac{n}{2}\right) \leq c \cdot f(n)$$

$$\Rightarrow 4 \cdot \left(\frac{n}{2}\right)^3 \leq c \cdot n^3$$

$$= 4 \cdot \frac{n^3}{8} \leq c \cdot n^3$$

$$\Rightarrow c = \frac{1}{2} < 1$$

So, ~~rel~~ regulating condition is satisfied

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

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

Sol → $a = 9,\ b = 3,\ f(n) = n^3$

$$n^{\log_b a} = n^{\log_3 9} = n^2$$

$\Rightarrow$ Case 3 will be applied

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

$$\Rightarrow 9 \cdot f\left(\frac{n}{3}\right) \leq c \cdot f(n).$$

$$\Rightarrow 9 \cdot \left(\frac{n}{3}\right)^3 \leq c \cdot n^3$$

$$\Rightarrow 9 \cdot \frac{n^3}{27} \leq c \cdot n^3$$

$$\Rightarrow c = \frac{1}{3} < 1$$

So, regulating condition is satisfied.

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

Closest Pair of Points :-

Algorithm :-

Input :- An array of n points P[ ]
Output :- Smallest distance between 2 points in given array.

As a preprocessing step, Input array is sorted according to x-coordinate.

1. Find the middle point in the sorted array m can take $P[\frac{n}{2}]$ as middle point.

2. Divide the given array in two half. The first half contain points from P[0] to $P[\frac{n}{2}]$. The second half contain points from $P[\frac{n}{2}+1]$ to P[n-1].

3. Recursively find the smallest distance in both subarray ~~but~~ Let the distance be dl and dr. Find the minimum of dl and dr. Let minimum be d.

$$\boxed{d = \min(dl, dr)}$$

4. Now, we need to consider the pair such that the one point is from left half and other half is from right half. Consider a vertical line passing through $P[\frac{n}{2}]$ and find all points where x coordinate is closer than d to the middle vertical line. Build an array strip of all such point.

5. Sort the array strip[] according to the y-coordinate. This step take O(nlogn).

6. Find the smallest distance in strip.

7. Finally return the minimum of 'd' and the smallest distance calculated in above step (step 6).