

Space Complexity → Rate of growth of space wrt i/p.

int → 4 Bytes

long → 8 Bytes

{ int x = 2; → 4B

? long y = x * 2; → 8B

Total memory = 12 B → O(1)

Find the Space Complexity [Big(O)] of the below program.

```
func(int N) { // 4 bytes → i/p
    int arr[10]; // 40 Bytes
    int x; // 4 bytes
    int y; // 4 bytes
    long z; // 8 bytes
    int[] arr = new int[N]; // 4 * N bytes
}
```

Total = $(54 + 4N)$ Bytes

SC = $O(N)$

Find the Space Complexity [Big(O)] of the below program.

```
func(int N) { // 4 bytes → i/p
    int x = N; // 4 bytes
    int y = x * x; // 4 bytes
    long z = x + y; // 8 bytes
    int[] arr = new int[N]; // 4 * N bytes
    long[][] l = new long[N][N]; // 8 * N * N bytes
}
```

Total = $(16 + 4N + 8N^2)$ B

SC = $O(N^2)$



Space apart from i/p & o/p.

Find the Space Complexity [Big(O)] of the below program.

```
int maxArr(int arr[], int N) {
    int ans = arr[0]; → 4B
    for(i from 1 to N-1) {
        ans = max(ans, arr[i]); → N * 4 Bytes
    }
    return ans; → 4B → o/p
}
```

↓
4B → SC = $O(1)$

Introduction to Arrays

$A = [\circled{0} \ 2 \ 8 \ 10 \ 12 \ \circled{4} \ 20]$ ← indices [0 (N-1)]

continuous memory allocation

$A[2] \rightarrow \text{(add. of } A) + (2 \times 4B) \rightarrow TC = \underline{O(1)}$

```
for i → 0 to (N-1) {  
    print (A[i])  
}
```

a → Reverse the given integer array.

$A = [\circled{1} \ 2 \ 3 \ 4 \ \circled{5}]$



2 Pointers
 $i = 0$ $j = N-1$

while ($i < j$) {

$t = A[i]$

$A[i] = A[j]$

$A[j] = t$

$i++$

$j--$

}

$TC = \underline{O(N)}$

swap x, y without 3rd variable

$x = x + y$

$y = x - y$

$x = x - y$



$SC = \underline{O(1)}$

Q → Reverse the array from index l to r .

$A = [\begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix}]$ $l = 1 \quad r = 2$

0 1 2 3 4
3 2

$i = l$ $j = r$

while ($i < j$) {

$t = A[i]$

$A[i] = A[j]$

$A[j] = t$

$i++$

$j--$

}

TC = $O(N)$

SC = $O(1)$

Q → Rotate the given array from right to left K times. (clockwise)

$A = [\begin{matrix} 9 & 5 & 8 & 2 & 3 & 1 & 7 & 8 \end{matrix}]$

$K = 1 \quad 8 \quad 9 \quad 5 \quad 8 \quad 2 \quad 3 \quad 1 \quad 7$

$2 \quad 7 \quad 8 \quad 9 \quad 5 \quad 8 \quad 2 \quad 3 \quad 1$

$3 \quad 1 \quad 7 \quad 8 \quad 9 \quad 5 \quad 8 \quad 2 \quad 3$

Bruteforce → $K = K \% N$

for $k \rightarrow 1$ to K {

$t = A[N-1]$

 for $i \rightarrow (N-2)$ to 0 { ←

$A[i+1] = A[i]$

}

$A[0] = t$

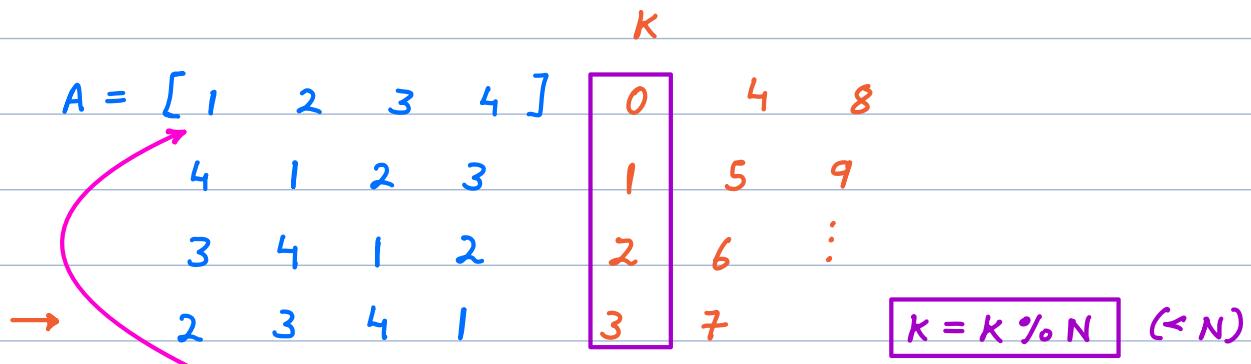
}

$$TC = \underline{O(K * N)} \quad SC = \underline{O(1)}$$

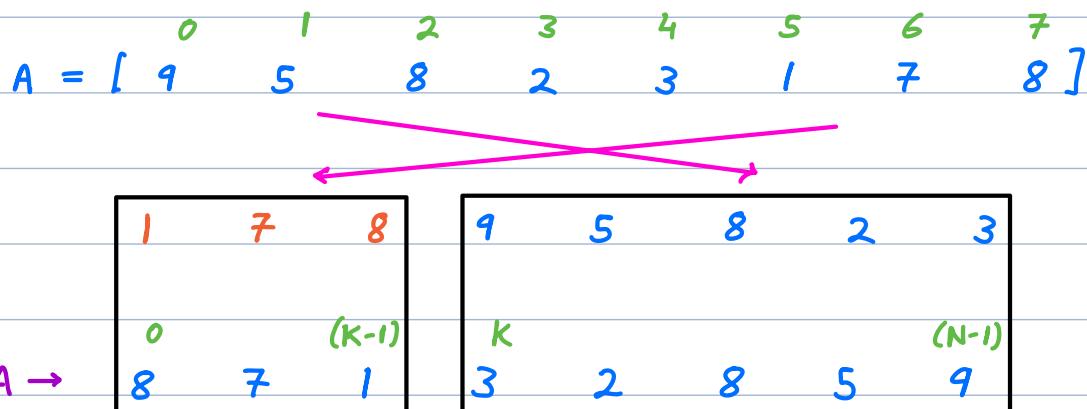
\nwarrow $\leftarrow \underline{O(N^2)}$

$TC / SC \rightarrow$ w.r.t. input (not always N)

$$K = 15 \quad N = 4 \quad \rightarrow \quad K = 15 \% 4 = \underline{3} \quad \checkmark$$



$$SC = \underline{O(1)}$$



Steps $\rightarrow \quad K = K \% N \quad \underline{TC}$

- 1) reverse $(A, 0, N-1) \rightarrow O(N)$
- 2) reverse $(A, 0, K-1) \rightarrow O(K)$
- 3) reverse $(A, K, N-1) \rightarrow O(N-K)$

$$\text{Total } TC = O(N + K + N - K) = O(2N) = \underline{O(N)}$$

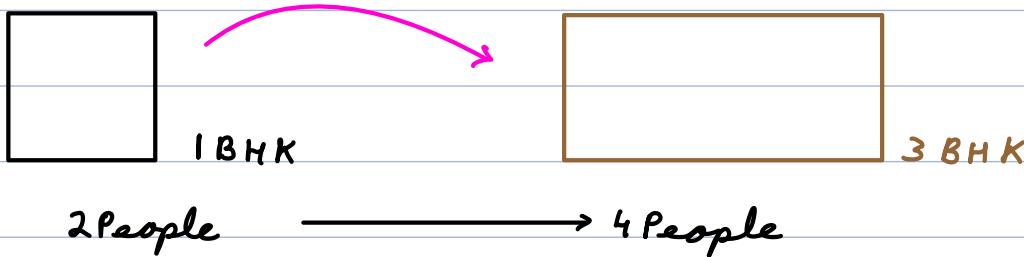
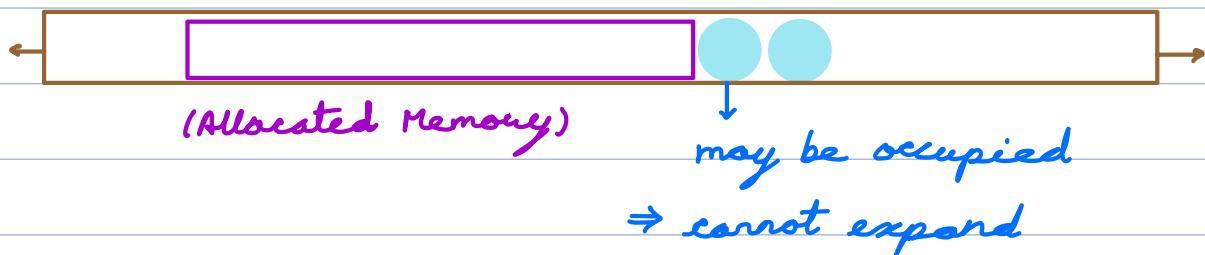
$$SC = \underline{O(1)}$$

Dynamic Array

```
int a[] = new int[10]; // 10 length integer array  
static array → pre-defined length
```

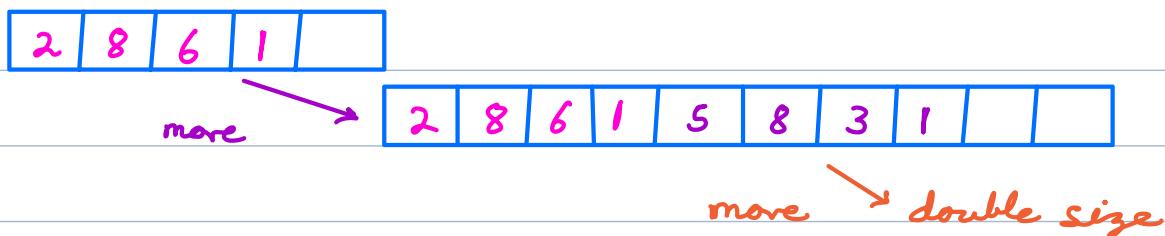
Sometimes defining length at the start
might not be possible → dynamic array
→ e.g. → Java `ArrayList`,
C++ `vector`, etc.
Access → $TC = O(1)$
→ continuous memory allocation

Memory



Load factor (0 1) = 0.8 (e.g.)

→ if 80% memory is occupied
find a double size memory.



if ((current size * load factor) is full)
find another location with double size.

if copy K elements \Rightarrow insert K more elements ^{copy + insert}
 \Rightarrow TC to insert K element = $O(2+K)$
 \Rightarrow TC to insert 1 element = $O(2) \rightarrow \underline{O(1)}$

$Q \rightarrow$ Given an array with daily profit/loss from a particular stock. calculate the total profit/loss over a given range of days.

The function should be able to handle multiple queries for different ranges.

$$A = [-5 \ 10 \ 20 \ 40 \ 50 \ -10 \ 8]$$

queries

start End

$$1 \ 4 \rightarrow 120$$

$$2 \ 5 \rightarrow 100$$

$$0 \ 0 \rightarrow -5$$

Given an integer array with N elements & Q queries, for every find sum of elements from index L to R .

$$A = [-5 \ 10 \ 20 \ 40 \ 50 \ -10 \ 8]$$

queries

L R

$$1 \ 4 \rightarrow 120$$

$$2 \ 5 \rightarrow 100$$

$$0 \ 0 \rightarrow -5$$

Range Sum Queries

Bruteforce \rightarrow Q queries, iterate from index L to R .

$$TC = O(N \times Q)$$

$$SC = O(1)$$

for $i \rightarrow 0$ to $(Q-1)$ {

$\| L, R$
 | for $i \rightarrow L$ to R {

Cricket

$$\sum_{i=0}^6 A[i] = \{10, 32, 6, 12, 20, 1\}$$

over →	0	1	2	3	4	5	6	7	8	9	10
Score →	2	8	14	29	31	49	65	79	88	97	

Runs scored in 7th over → 65 - 49 = 16

Runs scored from 6th to 10th over → 97 - 31 = 66

Runs scored in 10th over → 97 - 88 = 9

Runs scored from 3rd to 6th over → 49 - 8 = 41

Runs scored from 4th to 9th over → 88 - 14 = 74

Prefix Sum →

$$P[i] = A[0] + A[1] + \dots + A[i]$$

$$P[2] = A[0] + A[1] + A[2]$$

$$\begin{aligned} P[3] &= A[0] + A[1] + A[2] + A[3] \\ &= P[2] + A[3] \end{aligned}$$

$$P[i] = P[i-1] + A[i]$$

$$P[0] = A[0]$$

$$P[0] = A[0]$$

for $i \rightarrow 1$ to $(N-1)$ {

$$P[i] = P[i-1] + A[i]$$

}

$$TC = \underline{O(N)}$$

$$A = [\begin{smallmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 10 & 32 & 6 & 12 & 20 & 1 \end{smallmatrix}]$$

$$\checkmark P \rightarrow [\begin{smallmatrix} 10 & 42 & 48 & 60 & 80 & 81 \end{smallmatrix}]$$

sum (L to R)

$$P[R] - P[L-1]$$

$$\begin{aligned} & \text{sum (L to R)} \\ & P[R] - P[L-1] \end{aligned}$$

sum (0 — 3) \rightarrow $P[3] - P[0-1]$ index out of bound error

for $i \rightarrow 0$ to $(Q-1)$ {

 // L, R

 if ($L == 0$) print ($P[R]$)

 else print ($P[R] - P[L-1]$)

}

$TC = O(Q)$

$TC = O(N+Q)$

$SC = O(N)$

$O(1)$ \rightarrow Use $A[i]$ to store prefix sum.

~~$A[0] = A[0]$~~

for $i \rightarrow 1$ to $(N-1)$ {

$A[i] = A[i-1] + A[i]$

}

$A = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 10 & \cancel{32} & \cancel{6} & \cancel{12} & \cancel{20} & \cancel{1} \\ 42 & 48 & 60 & 80 & 81 \end{bmatrix}$

$A[i] = P[i] - P[i-1]$

$\alpha \rightarrow$ Given an integer array & α queries (L, R)
Find the sum of all even indexed elements.

$A = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 1 & \text{6} & 4 & 5 \end{bmatrix}$

queries

1 3 \rightarrow 1

Bruteforce \rightarrow $TC = O(N \times Q)$

2 5 \rightarrow 5

$SC = O(1)$

3 3 \rightarrow 0

$$p[i] \rightarrow A[0] + A[2] + A[4] \dots$$

$$A = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 1 & 6 & 4 & 5 \end{bmatrix}$$

$$p \rightarrow 2 \ 2 \ 3 \ 3 \ 7 \ 7$$

$$A = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 2 & 4 & 3 & 1 & 5 \end{bmatrix}$$

$$p \rightarrow 2 \ 2 \ 5 \ 5 \ 10$$

$$p[0] = A[0]$$

for $i \rightarrow 1$ to $(N-1)$ {

$$\begin{array}{|l} \text{if } (i \% 2 == 0) \quad p[i] = p[i-1] + A[i] \\ \text{else} \quad p[i] = p[i-1] \end{array}$$

}

for $i \rightarrow 0$ to $(Q-1)$ { // Queries $\rightarrow L = \text{query}[i][0]$

// L, R

$R = \text{query}[i][1]$

if ($L == 0$) print ($p[R]$)

else print ($p[R] - p[L-1]$)

}

$$TC = \underline{O(N+Q)} \quad SC = \underline{O(N)} \rightarrow \underline{O(1)}$$

Find the sum of all odd indexed elements.

$$p[0] = 0$$

for $i \rightarrow 1$ to $(N-1)$ {

$$\begin{array}{|l} \text{if } (i \% 2 == 1) \quad p[i] = p[i-1] + A[i] \\ \text{else} \quad p[i] = p[i-1] \end{array}$$

}

Q → Given an integer array, count the no. of special index i.e. index after removing which sum of all even indexed elements = sum of all odd indexed elements.

<u>i</u>	<u>A</u> = [3 2 7 6 -2]	<u>Even</u>	<u>Odd</u>
0	→ 2 7 6 -2	8	\neq 5
1	→ 3 7 6 -2	9	\neq 5
2	→ 3 2 6 -2	9	\neq 0
3	→ 3 2 7 -2	10	\neq 0
4	→ 3 2 7 6	10	\neq 8

Ans = 0

<u>A</u> = [4 1 3 7 10]	<u>7</u> 10	<u>odd sum = 1 + 10 = 11</u>
--------------------------------------	-------------	------------------------------

<u>A</u> = [2 3 1 4 0 -1 2 -2 10 8]	0 -1 2 -2 10 8
<u>odd sum = 3 + 0 + 2 + 10 = 15</u>	

elements $>$ i^{th} index → odd index move to even
even index move to odd

<u>A</u> = [2 3 1 4 0 -1 2 -2 10 8]	0 -1 2 -2 10 8
--	----------------

$$\text{even sum} = 2 + 1 + (-1) + (-2) + 8 = 8$$

check i^{th} index

$$S_{\text{even}} = S_{\text{even}}(0 - i-1) + S_{\text{odd}}(i+1 - N-1)$$

$$S_{\text{odd}} = S_{\text{odd}}(0 - i-1) + S_{\text{even}}(i+1 - N-1)$$

$$\text{Sum even} = \text{evenP}[i-1] + (\text{oddP}[N-1] - \text{oddP}[i])$$

$$\text{Sum odd} = \text{oddP}[i-1] + (\text{evenP}[N-1] - \text{evenP}[i])$$

// calculate oddP[] & evenP[]

crt = 0

for $i \rightarrow 0$ to $(N-1)$ {

 if ($i == 0$) {

$$se = \text{oddP}[N-1] - \text{oddP}[i]$$

$$so = \text{evenP}[N-1] - \text{evenP}[i]$$

 } else {

$$se = \text{evenP}[i-1] + (\text{oddP}[N-1] - \text{oddP}[i])$$

$$so = \text{oddP}[i-1] + (\text{evenP}[N-1] - \text{evenP}[i])$$

}

 if ($se == so$) crt++

}

return crt

TC = O(N)

SC = O(N)

Array - Carry forward & Subarrays

TABLE OF CONTENTS

1. Count 'a-g' pairs
2. Sub-arrays
3. Print a Subarray
4. Print all Subarrays
5. Min-Max





Count 'a-g' pairs

< Question > : Given a string s of lowercase characters, return the count of pairs (i, j) such that $i < j$ and $s[i]$ is 'a' and $s[j]$ is 'g'.

$s = "a b e g a g"$ $\begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \\ a & b & e & g & a & g \end{matrix}$ $\begin{matrix} i < j \\ 0 & 3 \end{matrix}$

Ans = 3 $\begin{matrix} 0 & 5 \end{matrix}$

$s = "a c g d g a g"$ $\begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ a & c & g & d & g & a & g \end{matrix}$ $\begin{matrix} i < j \\ 0 & 2 \end{matrix}$

Ans = 4 $\begin{matrix} 0 & 4 \end{matrix}$

$\begin{matrix} 0 & 6 \end{matrix}$

$s = "b c a g g a a g"$ $\begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ b & c & a & g & g & a & a & g \end{matrix}$ $\begin{matrix} i < j \\ 2 & 3 \end{matrix}$

$\begin{matrix} 2 & 4 \end{matrix}$

$\begin{matrix} 2 & 7 \end{matrix}$

$\begin{matrix} 5 & 7 \end{matrix}$

$\begin{matrix} 6 & 7 \end{matrix}$



$cnt = 0$

for $i \rightarrow 0$ to $(N-1)$ {

for $j \rightarrow 0$ to $(N-1)$ {

if ($i < j$ && $s[i] == 'a'$ && $s[j] == 'g'$)

$cnt++$

}

}

}

$TC = O(N^2)$

$SC = O(1)$

$cnt = 0$

for $i \rightarrow 0$ to $(N-2)$ {

if ($s[i] == 'a'$) {

for $j \rightarrow i+1$ to $(N-1)$ {
if ($s[j] == 'g'$)
 $cnt++$

}

}

}

count # 'g'
from $(i+1)$ to $(N-1)$.
 $ans += cnt - g[i+1]$

$TC = O(N^2)$

$O(N)$

$SC = O(1)$

$O(N)$

}

$s = "0 1 2 3 4 5 6 7"$
 $cnt_g \rightarrow [3 3 3 3 2 1 1 1]$

$L \leftarrow R$

$ans = 0 + 3 + 1 + 1 = 5$



$\forall i, \text{ if } (s[i] == 'g')$

$$\text{crt_g}[i] = \text{crt_g}[i+1] + 1$$

else $\text{if } s[i] \neq 'g'$

$$\text{crt_g}[i] = \text{crt_g}[i+1]$$



Idea

Carry Forward \rightarrow Calculate & Use

str \rightarrow

	0	1	2	3	4	5	6	7
b	c	a	g	g	a	a	g	

 $L \leftarrow R$

$$'g' \rightarrow \text{crt} \quad 3 \ 3 \ 3 \ 3 \ 2 \ 1 \ 1 \ 1$$

$$'a' \rightarrow \text{ans} = 0 + 1 + 1 + 3 = \underline{5}$$

$$\text{crt} = 0$$

$$\text{ans} = 0$$

for $i \rightarrow (N-1)$ to 0 {

 if ($s[i] == 'g'$) $\text{crt}++$ // calculate

 if ($s[i] == 'a'$) $\text{ans} += \text{crt}$ // use

$$\text{TC} = \underline{O(N)} \quad \text{SC} = \underline{O(1)}$$

$$i < j \quad s[i] = 'a' \quad s[j] = 'g'$$

count # 'a' from left to right

$$\text{crt} = 0 \quad \nearrow$$

$$0 \longrightarrow i$$

$$\text{ans} = 0$$

for $i \rightarrow 0$ to $N-1$ {

 if ($s[i] == 'a'$) $\text{crt}++$ // calculate

 if ($s[i] == 'g'$) $\text{ans} += \text{crt}$ // use

$$\text{TC} = \underline{O(N)}$$

$\checkmark \checkmark \checkmark \checkmark \checkmark \checkmark$

$$\text{crt} = +2$$

$$\text{ans} = 2 + 2 = \underline{4}$$

$$\text{SC} = \underline{O(1)}$$



Subarrays

→ continuous part of array

$A = [1 \ 2 \ 3 \ 4 \ 5]$

single element ✓

complete array ✓

Example: arr[] → [2 4 1 6 -3 7 8 4]

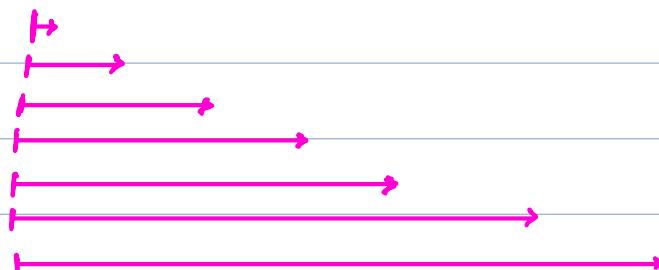
- a. [1, 6, 8]
- b. [1, 4]
- c. [6, 1, 4, 2]
- d. [7, 8, 4,] ✓



Representation of a subarray

\uparrow L R
start & end
 $\Rightarrow L$ (length) start & length

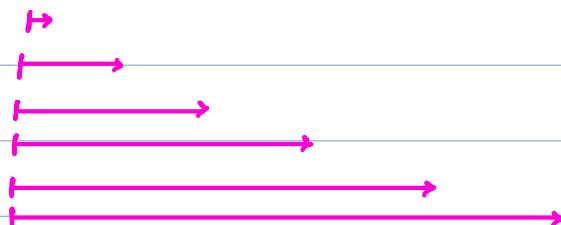
$$A = [0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6] \quad [4 \ 2 \ 10 \ 3 \ 12 \ -2 \ 15]$$



subarrays = 7

subarrays starting from index 0 = N

$$A = [0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6] \quad [4 \ 2 \ 10 \ 3 \ 12 \ -2 \ 15]$$



subarrays = 6

Total number of subarrays

subarrays starting from \downarrow = \downarrow

0 $\quad N$

1 $\quad N-1$

2 $\quad N-2$

:

$(N-1)$ 1

$N * (N+1) / 2$



< Question > : Given an array, si and ei. Print from si to ei. $si \leq ei$

arr \rightarrow [4 2 10 3 12 -2 15] $si = 2, ei = 5$
0 1 2 3 4 5 6

o/p \rightarrow 10 3 12 -2

- void printSubarray(arr, si, ei) {

for $i \rightarrow si$ to ei {
 print (arr[i])
}

}

print 1 subarray \rightarrow T.C $O(N)$

SC = $O(1)$



< Question > : Print all the possible sub-arrays of the given array.

[5, 7, 3, 2]

O/P - [5]

0 1 2 3

[5, 7]

[5, 7, 3]

$\frac{N \times (N+1)}{2} \times N \rightarrow O(N^3)$

[5, 7, 3, 2]

[7]

[7, 3]

[7, 3, 2]

[3]

8:30 AM

[3, 2]

[2]



Idea Consider all the subarrays & print Subarray()

```
for st → 0 to (N-1) {  
    for end → st to (N-1) {  
        for i → st to end {  
            print (A[i])  
        } print ('\n')  
    }  
}
```

$TC = O(N^3)$

$SC = O(1)$



Min Max

< Question > : Given an array of N integers, return the length of smallest subarray which contains both maximum and minimum elements of the array.

$1 \leq N \leq 10^6$

$\text{arr[]} \rightarrow [2 \ 2 \ 6 \ 4 \ 5 \ 1 \ 5 \ 2 \ 6 \ 4 \ 1]$

0 1 2 3 4 5 6 7 8 9 10

Ans = 3

$\text{arr[]} \rightarrow [1 \ 2 \ 3 \ 1 \ 3 \ 4 \ 6 \ 4 \ 6 \ 3]$

Ans = 4

0 1 2 3 4 5 6 7 8 9

$\text{arr[]} \rightarrow [8 \ 8 \ 8 \ 8 \ 8 \ 8]$

0 1 2 3 4 5

min = 8

max = 8

Ans = 1

Bruteforce \rightarrow Find min & max of the array $\rightarrow O(N)$

$O(N^2)$ $\xrightarrow{2}$ subarrays, check if it contains min & max and take smallest length subarray as answer. $\xrightarrow{O(N)}$

$TC = O(N + N^3) = O(N^3)$

$SC = O(1)$



Observation

1. There must be exactly one occurrence of min & max element.

[min min - - - - max]

2. Min and max elements should be the end point of subarray.

x min max

- ✓ $\forall \text{max check closest}$
min on left

3. case-1: [min - - - max] $\rightarrow \forall \text{min check closest}$
case-2: [max - - - min] max on right

✓ $\forall \text{min check closest}$
max on left

$\forall \text{max check closest}$
min on right

0 1 2 3 4 5 6 7 8 9 10
arr[] → [2 2 6 4 5 1 5 2 6 4 1] *min = 1*
 i i i i i i i i i i i *max = 6*

latest_mir_index = -1810

$$\text{latest_max_index} = 12 \quad \text{length} = 10 - 8 + 1 = 3$$
$$\text{ans} = 4 \ 3$$



</> Code

```
minA = A[0]      maxA = A[0]
for i → 1 to (N-1) {
    minA = min (minA, A[i])
    maxA = max (maxA, A[i])
}

l_min = -1      l_max = -1
ans = N

for i → 0 to (N-1) {
    if (A[i] == minA) {
        l_min = i      // calculating
        if (l_max != -1) {
            ans = min (ans, i - l_max + 1)
        }
    }
    if (A[i] == maxA) {
        l_max = i      [l_min ————— max
                           ————— i]
        if (l_min != -1) {
            ans = min (ans, i - l_min + 1)
        }
    }
}
return ans
```

TC = O(N) SC = O(1)

$\alpha \rightarrow$ Given an integer array,
find the total sum of all possible subarrays.

$$A = \begin{bmatrix} 0 & 1 & 2 \\ 3 & 2 & 5 \end{bmatrix}$$

$$3 \rightarrow 3$$

$$3 \ 2 \rightarrow 5$$

$$3 \ 2 \ 5 \rightarrow 10$$

$$A = \begin{bmatrix} 0 & 1 \\ 2 & 7 \end{bmatrix}$$

$$2 \rightarrow 2$$

$$2 \ 7 \rightarrow 9$$

$$7 \rightarrow 7$$

$$2 \rightarrow 2$$

$$2 \ 5 \rightarrow 7$$

$$5 \rightarrow 5$$

18 (Ans)

32 (Ans)

Bruteforce \rightarrow \forall subarrays, calculate sum & add it is the answer.

ans = 0

for $L \rightarrow 0$ to $(N-1)$ { $L - R$

for $R \rightarrow L$ to $(N-1)$ {

 sum = 0

 for $i \rightarrow L$ to R {

 sum += $A[i]$

 }

 subarray sum $L - R$

 ans += sum

$TC = O(N^3)$

$SC = O(1)$

}

 return ans

Prefix Sum →

$$P[0] = A[0]$$

for $i \rightarrow 1$ to $(N-1)$ {

$$P[i] = P[i-1] + A[i]$$

}

$$ans = 0$$

for $L \rightarrow 0$ to $(N-1)$ { $L = R$

for $R \rightarrow L$ to $(N-1)$ {

$$\text{if } (L == 0) \quad ans += P[R]$$

$$\text{else} \quad ans += P[R] - P[L-1]$$

}

} returns ans

$$TC = O(N + N^2) = \underline{O(N^2)}$$

$$SC = \underline{O(N)}$$

$$A = \begin{bmatrix} 3 & 2 & 5 \end{bmatrix}$$

$$s = \begin{cases} A[0] & 0-0 \\ A[0] + A[1] & 0-1 \\ \underline{A[0] + A[1] + A[2]} & 2 \end{cases} \quad ans += 5$$

$$\begin{cases} A[1] & ans += 5 \\ \underline{A[1] + A[2]} & ans += 5 \end{cases}$$

Carry Forward $L=2 \{ A[2] \}$

$$ans += 5$$

$$ans = 0$$

for $L \rightarrow 0$ to $(N-1)$ {

$$sum = 0$$

for $R \rightarrow L$ to $(N-1)$ {

$$sum += A[R]$$

$$ans += sum$$

}

} return ans

TC = O(N²) SC = O(1)

→ If one element is used multiple times to calculate the answer → contribution technique

$$\text{Ans} = \sum_{i=1}^n \text{contribution of } A[i]$$

$$A = [3^0, 2^1, 5^2]$$

$$3 \rightarrow 3$$

$$3, 2 \rightarrow 5 \quad 3 * 3 = 9$$

$$3, 2, 5 \rightarrow 10 \quad 2 * 4 = 8$$

$$2 \rightarrow 2 \quad 5 * 3 = 15$$

contribution of A[i]

A[i] * (# subarrays

$$2, 5 \rightarrow 7$$

32 (Ans)

A[i] is part of)

$$5 \rightarrow 5$$

32 (Ans)

$$A = [3^0, -2^1, 4^2, -1^3, 2^4, 5^5]$$

L R

L R

0 1

1 1

0 2

1 2

0 3

1 3

0 4

1 4

0 5

1 5

$$A = [3^0, -2^1, 4^2, -1^3, 2^4, 5^5]$$

L R

L R

L R

Ans = 12

0 2

1 2

2 2

0 3

1 3

2 3

0 4

1 4

2 4

0 5

1 5

2 5

subarrays with A[i]

starting index (L) = $[0 \ i]$ $\rightarrow i - 0 + 1 = \underline{i+1}$

ending index (R) = $[i \ N-1]$ $\rightarrow N-1-i+1 = \underline{N-i}$

subarrays having $A[i] = (i+1) * (N-i)$

$$\text{Ans} = \sum_{\forall i} A[i] * (i+1) * (N-i)$$

ans = 0

for $i \rightarrow 0$ to $(N-1)$ {

 |
 | $\text{ans} += A[i] * (i+1) * (N-i)$

} return ans

$$A = \begin{bmatrix} 3 & 2 & 5 \end{bmatrix}$$

i
 $0 \ 1 \ 2$

$TC = \underline{O(N)}$

$SC = \underline{O(1)}$

$$\text{Ans} = 3 * 1 * 3 +$$

$$2 * 2 * 2 +$$

$$5 * 3 * 1 = 9 + 8 + 15 = \underline{32} \checkmark$$

Q → Find the # of subarrays with length K.

$$A = \begin{bmatrix} 3 & -2 & 4 & -1 & 2 & 6 \end{bmatrix}$$

$0 \ 1 \ 2 \ 3 \ 4 \ 5$

$N = 6$

$$K = 3 \rightarrow \text{Ans} = \underline{4}$$

$$K = 5 \rightarrow \text{Ans} = \underline{2}$$

1st subarray $\rightarrow 0 \ \underline{\quad} \ (K-1)$

last subarray $\rightarrow (N-K) \ \underline{\quad} \ (N-1)$

$[st \ (N-1)]$

$$N-1 - st + 1 = K \Rightarrow st = \underline{N-K}$$

$$\# \text{values } [(k-1) \dots (N-1)] \rightarrow (N-1)-(k-1)+1 = \underline{N-k+1} \text{ (Ans)}$$

$$N=7 \quad k=4 \quad \text{Ans} = N-k+1 = 7-4+1 = \underline{4}$$

Q → Print all start & end indices of length k.

$$A = [\begin{smallmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 3 & -2 & 4 & -1 & 2 & 6 \end{smallmatrix}]$$

$$\begin{array}{ll} N=6 & K=3 \\ & \underline{L \quad R} \\ & \begin{array}{cc} 0 & 2 \\ 1 & 3 \\ 2 & 4 \\ 3 & 5 \end{array} \end{array}$$

for $L \rightarrow 0$ to $(N-K)$ {

$$R = L + (K-1)$$

print (L, R)

$$TC = \underline{O(N)} \quad SC = \underline{O(1)}$$

}

Q → Given an integer array,
find max subarray sum & subarrays
of length k.

$$A = [\begin{smallmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 3 & -2 & 4 & -1 & 2 & 6 \end{smallmatrix}] \quad N=6 \quad K=3$$

$$\begin{array}{ccccccc} & \overbrace{3 \quad -2 \quad 4}^{\text{subarray}} & \quad & \overbrace{-1 \quad 2 \quad 6}^{\text{subarray}} & & & \\ \downarrow & \downarrow & \downarrow & \downarrow & \underline{L \quad R} & & \\ 5 & 1 & 5 & 7 & 0 & 2 & [L \quad R] \end{array}$$

$$\text{Ans} = \underline{7}$$

$$1 \quad 3 \quad R-L+1 = K$$

$$2 \quad 4 \quad \Rightarrow \underline{R = K-1+L}$$

$$3 \quad 5$$

Bruteforce

ans = Int-Min

for $L \rightarrow 0$ to $(N-K)$ {

$R = L + (K-1)$

sum = 0

for $i \rightarrow L$ to R {

sum += A[i]

}

$\rightarrow TC = \underline{O(K)}$

ans = max(ans, sum)

}

$SC = \underline{O(1)}$

$TC = O((N-K+1) * K)$

$\rightarrow \underline{O(N^2)}$

$$K=1 \quad (N-1+1) * 1 = N$$

$$K=N \quad (N-N+1) * N = N$$

$$K=\frac{N}{2} \quad (N-\frac{N}{2}+1) * \frac{N}{2} = (\frac{N}{2}+1) * \frac{N}{2} = \frac{N^2}{4} + \frac{N}{2} \rightarrow \underline{O(N^2)}$$

Prefix Sum

$P[0] = A[0]$

for $i \rightarrow 1$ to $(N-1)$ {

$P[i] = P[i-1] + A[i]$

}

ans = Int-Min

for $L \rightarrow 0$ to $(N-K)$ {

$R = L + (K-1)$

if ($L == 0$) ans = max(ans, P[R])

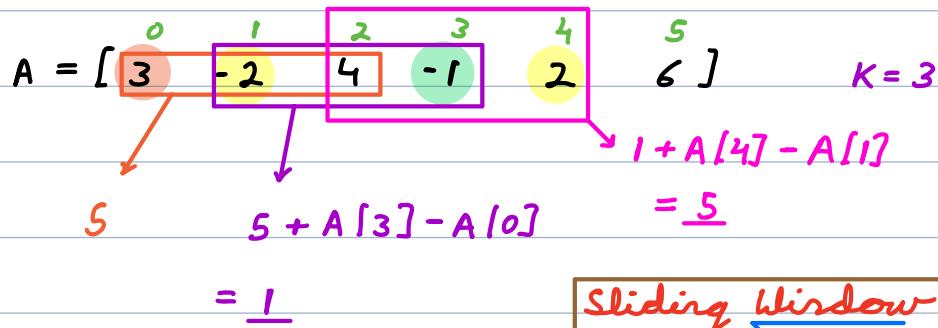
else ans = max(ans, P[R] - P[L-1])

}

return ans

$$TC = O(N + (N - K + 1)) \rightarrow O(N)$$

$$SC = O(1)$$



Sliding Window

Fixed length subarray

L	R	sum
0	2	$A[0] + A[1] + A[2] = 5$
1	3	$sum + A[3] - A[0] = 1$
2	4	$sum + A[4] - A[1] = 5$
3	5	$sum + A[5] - A[2] = 7$

max \rightarrow 7 (Ans)

$$sum = 0$$

```
for i → 0 to (K-1) {  
    sum += A[i]  
}
```

$$ans = sum$$

```
for R → K to (N-1) {  
    L1 = R - K  
    sum += A[R] - A[L1]  
    ans = max (ans, sum)  
}
```

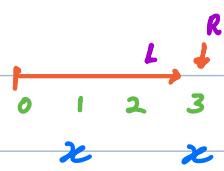
$$R - L + 1 = K \Rightarrow L = R - K + 1$$

L — (R)

return ans

$$TC = O(K + N - K) = O(N)$$

$$SC = O(1)$$



$$A = \begin{bmatrix} 3 & 2 & 4 & 2 & 3 & 4 \end{bmatrix}$$

L *R*

ret[3] = true

Ans = $\sum_{i=R}^R$ # subarrays ending at R

$$1 + 2 + 3 + 2 + 3 + 3 = \underline{14}$$

$$\underline{R-L+1}$$

	0	1	2
0	(0,0)	(0,1)	(0,2)
1	(1,0)	(1,1)	(1,2)
2	(2,0)	(2,1)	(2,2)

Column Index

$A[1][2]$

Row Index

$A[N][M]$

#rows #columns

Top right cell $\rightarrow A[0][M-1]$

Bottom right cell $\rightarrow A[N-1][M-1]$

Q \rightarrow Given a 2D matrix, print row-wise sum.

	0	1	2	3	O/P
0	1	2	3	4	$\rightarrow 10$
1	5	6	7	8	$\rightarrow 26$
2	9	10	11	12	$\rightarrow 42$

for $i \rightarrow 0$ to $(N-1)$ {

 sum = 0

 for $j \rightarrow 0$ to $(M-1)$ {

 sum += $A[i][j]$

 }

 print (sum)

TC = $O(N * M)$ SC = $O(1)$

}

Q \rightarrow Given a 2D matrix, find max of every column.

	0	1	2	3
0	5	12	3	1
1	5	6	7	8
2	10	8	2	19

0/p \rightarrow 10 12 7 19

for $i \rightarrow 0$ to $(M-1)$ { // columns

$m = A[0][i]$

 for $j \rightarrow 1$ to $(N-1)$ { // row

$m = \max(m, A[j][i])$

 }

 print(m)

}

$TC = \underline{\mathcal{O}(N \times M)}$ $SC = \underline{\mathcal{O}(1)}$

ArrayList<Integer> al[] = new ArrayList<>[N];

for $i \rightarrow 0$ to $(N-1)$ {

 al[i] = new ArrayList<>(); // al[i].get(j)

}

ArrayList<ArrayList<Integer>> al = new ArrayList<>();

Q \rightarrow Given a 2D square matrix, print diagonals.

1. **Principal Diagonal:** From top left to bottom right.

2. **Anti Diagonal:** From top right to bottom left.

1	5	8
4	3	1
6	5	2

Principal Diagonal

1	5	8
4	3	1
6	5	2

Anti-Diagonal

▷ Principal Diagonal

$(i == j)$



0	(0,0)	(0,1)	(0,2)
1	(1,0)	(1,1)	(1,2)
2	(2,0)	(2,1)	(2,2)

Column Index

Row Index

for $i \rightarrow 0$ to $(N-1)$ &
print $(A[i][i])$
}

TC = $O(N)$ SC = $O(1)$

3) Anti - Diagonal

$i++ \quad i--$ ←
(0, (n-1))

$$0 + 2 = 2$$

$$1 + 1 = 2$$

$$2 + 0 = 2 \quad i + j = N-1 \\ \Rightarrow j = N-1 - i$$

	0	1	2	←
0	(0,0)	(0,1)	(0,2)	Column Index
1	(1,0)	(1,1)	(1,2)	
2	(2,0)	(2,1)	(2,2)	

↑ Row Index

for $i \rightarrow 0$ to $(N-1)$ &
print $(A[i][N-1-i])$

}

TC = $O(N)$ SC = $O(1)$

Q → Given a 2D matrix, print all elements

diagonally from right to left.

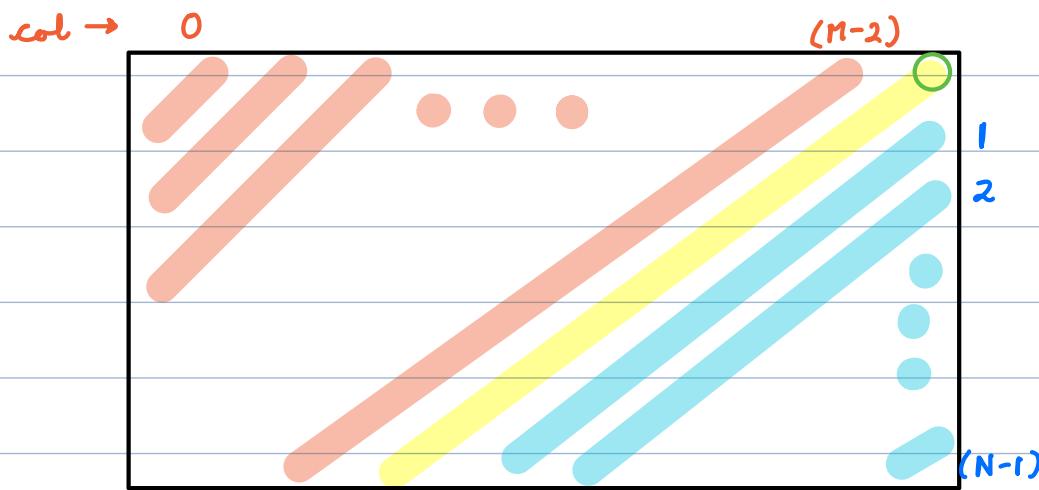
→ 0

0	1	2	3	
1	5	12	3	1
2	5	6	7	8
3	10	8	2	19

o/p → 5

12	5
3	6
10	8

8	2
19	



$$[0 \quad (M-2)] \rightarrow M-2-0+1 = \underline{M-1}$$

$$[1 \quad (N-1)] \rightarrow N-1-1+1 = \underline{N-1}$$

$$\text{Total diagonals} = M-1 + N-1 + 1 = \underline{M+N-1}$$

$$(4*5) \rightarrow 4+5-1 = \underline{8}$$

for $x \rightarrow 0$ to $(M-1)$ {

$i=0 \quad j=c$

 while ($i < N \quad \& \& \quad j \geq 0$) { $j = \cancel{z} z + 0$

 print (A[i][j])

$i++ \quad j--$

 } print ("\\n")

}

for $x \rightarrow 1$ to $(N-1)$ {

$i=r \quad j=M-1$

 while ($i < N \quad \& \& \quad j \geq 0$) { $r = + \quad \boxed{2}$

 print (A[i][j])

$i++ \quad j--$

 } print ("\\n")

}

$$c = \cancel{z} z \quad \boxed{3}$$

$$i = \cancel{z} z \quad 3$$

	0	1	2	3
0	5	12	3	1
1	5	6	7	8
2	10	8	2	19

A =

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

for $i \rightarrow 1$ to N {

 for $j \rightarrow 1$ to 10 {

 ...

$$TC = O(N \times 10) \rightarrow O(N)$$

}

for $c \rightarrow 0$ to $(M-1)$ {

$i=0$ $j=c$

 while ($i < N$ & $j >= 0$) {

 print (A[i][j])

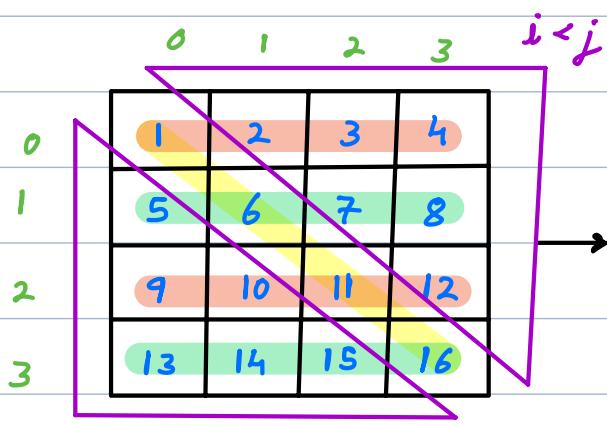
$i++$ $j-$

 } print ("\n")

}

c	while	# iterations
0	$0 \rightarrow N-1$ $0 \rightarrow 0$	1
1	$0 \rightarrow N-1$ $1 \rightarrow 0$	2
:		
3		

Q → Find transpose of given square matrix.



$i > j$

1) row \longleftrightarrow column

$i, j \longleftrightarrow j, i$

2) no change in principal diagonal elements.

$\text{swap}(x, y) \rightarrow x = x$

$x = y$

$y = x$

for $i \rightarrow 0$ to $(N-2)$ {

 for $j \rightarrow i+1$ to $(N-1)$ {

 swap $(A[i][j], A[j][i])$

}

}

$i=0 \quad j=1$

$(0, 1) \leftrightarrow (1, 0)$

$i=1 \quad j=0$

$(1, 0) \leftrightarrow (0, 1)$

#iterations

swap twice \Rightarrow no swap

$(N-1) + (N-2) + (N-3) + \dots + 1$

$= \frac{(N-1) \times (N)}{2}$

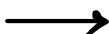
$TC = O(N^2)$

$SC = O(1)$

Q \rightarrow Rotate the given sq matrix 90° clockwise.

90°

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16



13	9	5	1
14	10	6	2
15	11	7	3
16	12	8	4

↓ transpose

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

↑ Reverse Row



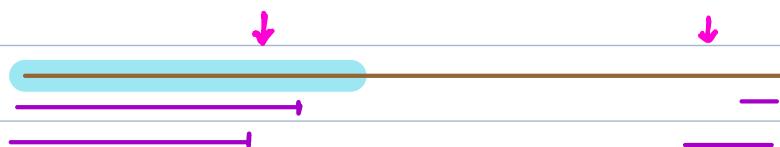
13	9	5	1
14	10	6	2
15	11	7	3
16	12	8	4

Sol → 1) transpose the matrix }
2) Reverse every row } 90° rotation

$$TC = O(N^2 + N^2) = \underline{O(N^2)} \quad SC = \underline{O(1)}$$

$a.length \rightarrow$ row length

$a[0].length \rightarrow$ col length



Q → Given a binary array of 1 & 0.

Find max # of consecutive 1's that can be obtained after replacing atmost one 0 with 1.

$$A = [\begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ \hline 1 & & & & & 1 & \\ \hline 5 & & & & 4 & & \end{array}] \quad \text{Ans} = \underline{5}$$

$$A = [\begin{array}{ccccccc} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ & & & & & 1 & & & \\ \hline & & & & & & & & \end{array}] \quad \text{Ans} = \underline{6}$$

$$A = [\begin{array}{cccccccccc} 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ \hline 4 & & & & 1 & & & 5 & & 1 & \\ & & & & & & & & & & \end{array}] \quad \text{Ans} = \underline{6}$$

Sol → ∀ 0's, check max consecutive 1's we can have by updating this 0 with 1.

count 1's on
left & right.

one = 0

```
for i → 0 to (N-1) {  
    if (A[i] == 1) one++  
}
```

if (one == N) return N

ans = 0

```
for i → 0 to (N-1) {  
    if (A[i] == 0) {  
        l = 0
```

```

for j → (i-1) to 0 { || for (j=i-1; j >= 0; j--)
    if (A[j] == 1) l++
    else break
}

r = 0

for j → (i+1) to (N-1) {
    if (A[j] == 1) r++
    else break
}

ans = max (ans, l + 1 + r)

```

$$A = [0 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \ 0]$$



Ans = 6

$\alpha \rightarrow$ Find max consecutive 1's by swapping almost a 0 with 1.

$$A = [0 \ 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 0]$$

Ans = 6

$$A = [0 \ 1 \ 2 \ 3 \ 4 \ 5]$$

Ans = 5

$$\text{one} = 0$$

for $i \rightarrow 0$ to $(N-1)$ {

if $[A[i]] == 1$ one + +

3

if (one == N) return N

ans = 0

for $i \rightarrow 0$ to $(N-1)$ {

if ($A[i] == 0$) {

 l = 0

 for $j \rightarrow (i-1)$ to 0 { // for $j = i-1$; $j \geq 0$; $j--$)

 if ($A[j] == 1$) l++

 else break

 }

 r = 0

 for $j \rightarrow (i+1)$ to $(N-1)$ {

 if ($A[j] == 1$) r++

 else break

 }

 ans = max (ans, l + 1 + r)

}

}

TC = $O(3N) = O(N)$ SC = $O(1)$

if (ans > one) ans --

return ans

Q → Given an integer array find majority element

i.e. element with frequency $> \frac{N}{2}$.

$A = [2 \ 1 \ 4]$ Ans = 1

$A = [2 \ 2 \ 4]$ Ans = 2

$A = [3 \ 4 \ 3 \ 2 \ 4 \ 4 \ 4 \ 4]$

$N = 8$ freq (4) = 5 $> \frac{N}{2}$ Ans = 4

$$A = [4 6 5 3 4 5 6 4 4 4] \quad \text{Ans} = 1$$

$$\text{freq}(4) = 5 \neq \frac{10}{2}$$

Max # majority elements = 1



Bruteforce \rightarrow $\forall A[i]$, calculate frequency & check if its $\geq \frac{N}{2}$.

$$TC = O(N^2) \quad SC = O(1) \rightarrow \text{Two loops}$$

$$TC = O(N) \quad SC = O(N) \rightarrow \text{Precompute & store freq.}$$

Observation \rightarrow If we remove 2 distinct elements from the input then majority element remains same.

$$\text{freq}(\text{majority element}) > \frac{N}{2}$$

$$\begin{array}{c|c|c} & -1 & 0 \end{array}$$

$$\text{freq}(\text{everything else}) < \frac{N}{2}$$

$$\begin{array}{c|c|c} & -1 & -2 \end{array}$$

$$A = [3 \checkmark 4 \checkmark 3 \checkmark 6 \checkmark 1 \checkmark 3 \checkmark 2 \checkmark 5 \checkmark 3 \checkmark 3 \checkmark 3] \rightarrow 3 \text{ (Ans)}$$

$$f = + \cancel{\sigma} + \cancel{\sigma} + \cancel{\sigma} + \cancel{\sigma} + \cancel{\sigma} + 2 \checkmark 3$$

$$m = 2 \cancel{\sigma} + 2 \checkmark 3$$

$$A = [3 \checkmark 4 \checkmark 3 \checkmark 0 \checkmark 3 \checkmark 1 \checkmark 3]$$

$$f = + \cancel{\sigma} + \cancel{\sigma} + \cancel{\sigma} + \cancel{\sigma} + 1$$

$$m = 3$$

check $\text{freq}(3) > N/2$

$m = -1$

$f = 0$

for $i \rightarrow 0$ to $(N-1)$ {

 if ($f == 0$) {

$m = A[i]$

$f++$

 } else if ($A[i] == m$) $f++$

 else $f--$

}

$A = [1 \ 3 \ 1 \ 8 \ 8 \ 8 \ 1 \ 3]$

0 1 2 3 4 5 6 7

$f = 0$

for $i \rightarrow 0$ to $(N-1)$ {

$f = f + 1$

$m = \underline{f + 8}$

}

 if ($A[i] == m$) $f++$

if ($f \geq N/2$) return m

return -1

$TC = \underline{O(N)}$

$SC = \underline{O(1)}$

Q → Given a 2D matrix of integers.

Update all elements in a row (i) & col (j) to 0

if $A[i][j] = 0$.

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 0 \\ 9 & 2 & 0 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 2 & 0 & 3 & 1 \\ 5 & 6 & 7 & 1 \\ 9 & 2 & 5 & 4 \end{bmatrix}$$

identify updated 0 vs original 0 ✓

- 1) Update all row / col with ($\neq 0$) -1 where current $\neq 0$.
- 2) Update all -1 to 0.

$$TC = \underline{O(N \times M)} \quad SC = \underline{O(1)}$$

One Dimensional Array

TABLE OF CONTENTS

1. Max Subarray Sum
2. Prefix Sum
3. Rain Water Trapped



Stock

$A = [-20 \ 30 \ 40 \ -10 \ 50 \ -100 \ 70]$

Max profit in one trade $\rightarrow 30 + 40 - 10 + 50 = \underline{110}$



< Question > : Given $\text{arr}[N]$. Find the maximum subarray sum out of all subarrays.

$$1 \leq N \leq 10^5$$

Example 1 : $\text{arr}[] \rightarrow [-3, 2, 4, -1, 3, -4, 3]$

0 1 2 3 4 5 6

Ans = 8

Example 2 : $\text{arr}[] \rightarrow [4, 5, 2, 1, 6]$

0 1 2 3 4

Ans = 18

Example 3 : $\text{arr}[] \rightarrow [-4, -3, -6, -9, -2]$

Ans = -2

$$\boxed{\begin{aligned} x &> y \\ -x &< -y \end{aligned}}$$

Bruteforce \rightarrow 1 subarray, calculate sum & take max.

$$\frac{N \times (N+1)}{2}$$

travel $\rightarrow O(N)$

$$\text{Total TC} = O(N^3)$$

$$SC = O(1)$$

$O(1)$ \rightarrow Prefix Sum ✓
Carry Forward ✓

```

ans = 0 INT-MIN // = A[0]
for i → 0 to (N-1) {
    sum = 0
    for j → i to (N-1) { // i → j
        sum += A[j]
        ans = max (ans, sum)
    }
}
return ans

```

TC = O(N²) SC = O(1)

Observations

Case 1

When all elements are positive

2	4	5	1	3
---	---	---	---	---

$$\text{Ans} = \sum A[i] \\ \forall i$$

Case 2

When all elements are negative

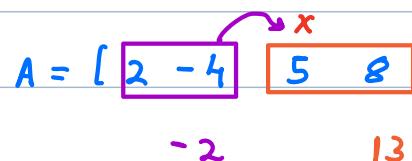
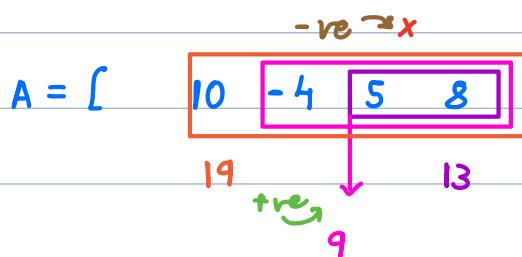
-7	-3	-11	-8	-15
----	----	-----	----	-----

$$\text{Ans} = \max [A[i]] \\ \forall i$$

Case 3

	+ve	
--	-----	--

Kadane's Algo





*Dry-Run

~~x~~ L i

arr[] → [5 , 6 , 7 , -3 , 2 , -10 , -12 , 8 , 12 , -4 , 7 , -2]
0 1 2 3 4 5 6 7 8 9 10 11

sum = 05 + 18 15 17 + ~~-3~~⁰ 8 20 16 23 21

ans = 8 + 18 20 23

</> Code

ans = INT. MIN

Kadane's Algo

sum = 0 l = 0

for i → 0 to (N-1) {

Subarray → L — R

 sum += A[i]

 if (ans < sum) {

 ans = sum

i
0 1 2
A = [-2 -1 -5]

 L = l R = i

sum = 0 - 2 - 0 = 1 - 0 = -80

}

ans = -2 -1

 if (sum < 0) {

 sum = 0 l = i + 1

}

}

TC = O(N) SC = O(1)



Continuous Sum Query

< Question > : There are A beggars sitting in a row outside a temple. Each beggar has an empty pot initially. There are N devotees. Each devotee gives some fixed amount to beggars from indices l to r.

Given a 2-D Array B[N][3], where B[i][0] represents l [left index] l

B[i][1] represents r [right index] r

B[i][2] represents amount

Give an integer array, where every element is 0 initially. Update the array with multiple queries.

Query (l, r, x) \rightarrow $\forall i \rightarrow l \text{ to } r \quad A[i] += x$

l r Amount(x)

Example : $A = 5, B[N][3] \rightarrow$

1	2	10
2	3	20
2	5	25

0	0	0	0	0
1	2	3	4	5

$+10 \quad +10$

$+20 \quad +20$

$+25 \quad +25 \quad +25 \quad +25$

10 55 45 25 25 (Ans)

Query (i, x) \rightarrow

Add x to all the

numbers from i to (N-1).

$A = [0 \quad 0 \quad 0 \quad 0 \quad 0]$

$+3 \quad +3 \quad +3 \quad +3$

$+2 \quad +2 \quad +2 \quad +2 \quad +2$

$+1$

2 5 5 5 6 (Ans)

$(N-1)$

\downarrow

I/P \rightarrow 1 4 3

0 4 2

4 4 1



BF Idea

query, travel from l to r & add x .length of array $\rightarrow A$ queries $\rightarrow N$

```
for i → 0 to (N-1) { // query  // For 1-based do -1
    l = B[i][0]           r = B[i][1]           x = B[i][2]
    for j → l to r {
        ans[j] += x
    }
}
return ans
TC = O(N * A)  SC = O(1)
```

$A = [0 \ 0 \ 0 \ 0 \ 0]$ $I/P \rightarrow 1 \ 4 \ 3$

$\begin{array}{r} 0 & 1 & 2 & 3 & 4 \\ +3 & +3 & +3 & +3 & +3 \\ +2 & +2 & +2 & +2 & +2 \\ \hline & & +1 & & \\ 2 & 5 & 5 & 5 & 6 \end{array}$ (Ans)

$\downarrow^{(N-1)}$

0 4 2
4 4 1

Add x from $i \rightarrow (N-1)$

$A = [0 \ 0 \ 0 \ 0 \ 0]$

$\begin{array}{r} +3 \\ +2 \\ \hline & +1 \\ 2 & 3 & 0 & 0 & 1 \end{array}$

$\underline{2 \ 5 \ 5 \ 5 \ 6}$ (Ans)

Prefix Sum

$$P[i] = A[i] + P[i-1]$$



```
for i → 0 to (N-1) { // query
    l = B[i][0]           x = B[i][1]
    ans[l] += x
}
```

```
for i → 1 to (A-1) {
    ans[i] = ans[i-1] + ans[i]
}
P[i]   P[i-1] + a[i]
```

return ans

TC = O(N + A) SC = O(1)

$$A = [\begin{smallmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{smallmatrix}]$$

$$\begin{array}{r} +2 \\ -2 \\ -5 \\ +3 \\ -3 \\ \hline 0 & 2 & 3 & 0 & -7 & 0 & -3 \\ \hline 0 & 2 & 5 & 5 & -2 & -2 & -5 \end{array}$$

queries

l	r	x
1	3	2
4	6	-5
2	5	3

```
for i → 0 to (N-1) { // query
```

$$l = B[i][0] \quad r = B[i][1] \quad x = B[i][2]$$

$$ans[l] += x$$

$$\text{if } (r \neq (N-1)) \quad ans[r+1] -= x$$

}

```
for i → 1 to (A-1) {
```

$$ans[i] = ans[i-1] + ans[i]$$

$$P[i] \quad P[i-1] + a[i]$$

TC = O(N+A)SC = O(1)

return ans



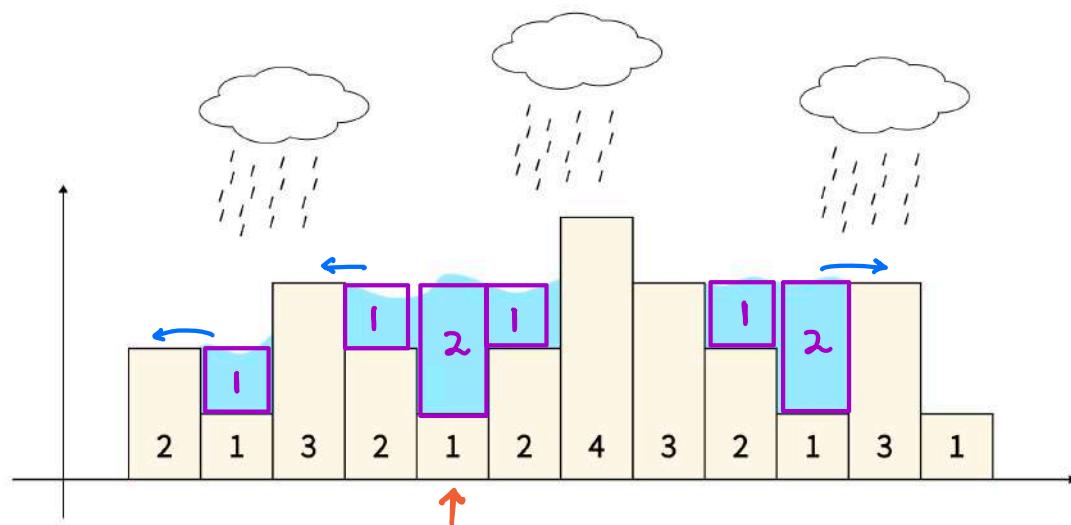
< Question > : Given $\text{arr}[N]$, where $\text{arr}[i] \rightarrow$ height of building. $\text{Base} = 1$ \forall buildings

Return amount of water trapped on all the buildings.

$$\text{Area} = B * H$$

$\text{arr} [2 \ 1 \ 3 \ 2 \ 1 \ 2 \ 4 \ 3 \ 2 \ 1 \ 3 \ 1]$

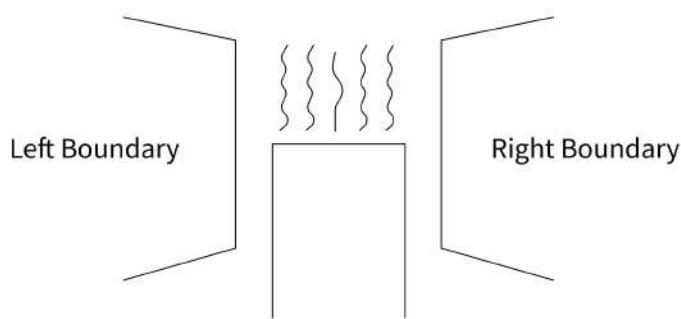
$1 \leq N \leq 10^5$



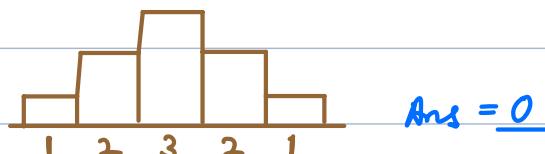
Ans = 8



Idea -1 Find the amount of water trapped on every building.



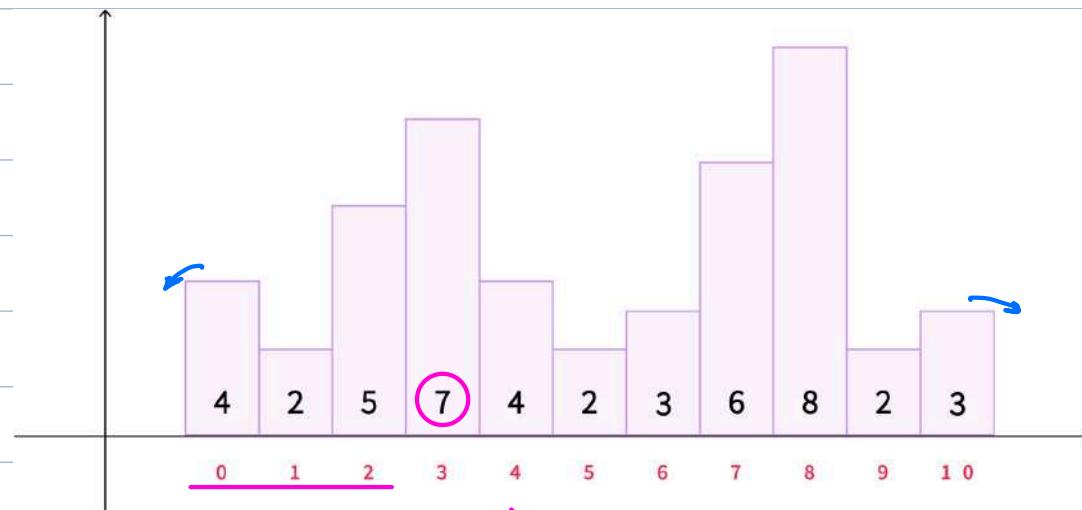
$$\min(\max A[i] \text{ on left}, \max A[i] \text{ on right}) - A[i]$$



Ans = 0

Bruteforce $\rightarrow \text{TC} = O(N^2)$

$\text{SC} = O(1)$



→ $lmax[0] \rightarrow [4 \ 4 \ 4 \ 5 \ 7 \ 7 \ 7 \ 7 \ 7 \ 8 \ 8 \ 8]$

→ $rmax[10] \rightarrow [8 \ 8 \ 8 \ 8 \ 8 \ 8 \ 8 \ 8 \ 8 \ 3 \ 3]$

$lmax[0] = A[0]$

for $i \rightarrow 1$ to $(N-1)$ { // →
| $lmax[i] = \max(lmax[i-1], A[i])$
}

$rmax[N-1] = A[N-1]$

for $i \rightarrow (N-2)$ to 0 { // ←
| $rmax[i] = \max(rmax[i+1], A[i])$
}

Prefix/Suffix → sum, max, min etc

$ans = 0$

for $i \rightarrow 0$ to $(N-1)$ {
| $ans += \min(lmax[i], rmax[i]) - A[i]$
} return ans



$$TC = \underline{\mathcal{O}(N)} \quad SC = \underline{\mathcal{O}(N)}$$

2D Matrices

TABLE OF CONTENTS

1. Search in row-wise & column-wise sorted 2D array
2. Row with maximum number of 1's
3. Spiral Matrix
4. Sum of all sub-matrices sum



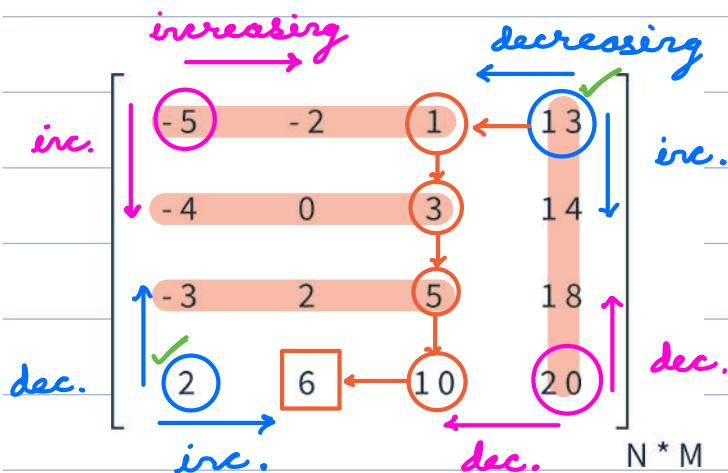
Search in row-wise & column-wise sorted 2D array

Search (6) → ✓ tree

Search (15) → ✗ false

Bruteforce → Travel all cells.

$$TC = \underline{O(N \times M)} \quad SC = \underline{O(1)}$$



travel from corners.

Search (6) \rightarrow true

Start → Top Right or
Bottom Left

$$g_1 = 0 \quad x_1 = M-1$$

while ($x < N$ && $x \geq 0$) {

```
if (A[r][c] == K)    return true
else if (A[r][c] < K)  r ++
else    c --
```

3

return false

$$TC = \underline{O(N + M)} \quad SC = \underline{O(1)}$$



Row with maximum number of 1's

Given a binary sorted matrix A of size $N \times N$. Find the row with the maximum number of 1's [Only rows are sorted]

0 0 ... 0 1 1 ... 1

$A = [\begin{matrix} 0 & 1 & 2 \\ 0 & [0, 1, 1] \\ 1 & [0, 0, 1] \\ 2 & [0, 1, 1] \end{matrix}]$

Ans = 0 if multiple rows are present, return the first row.

$A = [\begin{matrix} 0 & 1 & 2 & 3 \\ 0 & [0, 0, 0, 0] \\ 1 & [0, 0, 0, 1] \\ 2 & [0, 0, 1, 1] \\ 3 & [0, 1, 1, 1] \end{matrix}]$

Ans = 3

$$\begin{matrix} & 0 & 1 & 2 & 3 \\ 0 & \left[\begin{matrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{matrix} \right] \\ 1 & \\ 2 & \\ 3 & \end{matrix}$$

Ans = 2

Bruteforce \rightarrow \forall rows, count #1's.

$TC = O(N^2)$ $SC = O(1)$

Observation

1) Row with min 0's ✓

→ 2) First I can give #1's ✓

0	1	2	3	4	5
0	0	0	0	1	1
1	0	1	1	1	1
2	0	0	0	0	1
3	0	0	0	0	1
4	0	1	1	1	1
5	0	0	0	1	1



ans = 0

r = 0 c = M-1

while (r < N && c >= 0) {

 while (c >= 0 && A[r][c] == 1) {

 c--

 ans = r

}

 r++

}

return ans

	0	1	2	3
0	0	1	1	1
1	0	0	0	1
2	1	1	1	1
3	1	1	1	1

ans = 0
2

TC = O(N + N) = O(N) SC = O(1)

Print Boundary Elements

mat[N][N]

→ Square Matrix

mat[5][5]

	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20
4	21	22	23	24	25

o/p → [1 , 2 , 3 , 4 , 5 , 10 , 15 , 20 , 25 , 24 , 23 , 22 , 21 , 16 , 11 , 6]

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad o/p \rightarrow 1 \ 2 \ 3 \ 6 \ 9 \ 8 \ 7 \ 4$$

mat[N][N]

mat[5][5]

	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20
4	21	22	23	24	25



$$r=0 \quad c=0$$

while ($c < (N-1)$) { →

 print (A[r][c])

 c++

}

while ($r < (N-1)$) {

 print (A[r][c])

 r++

}

while ($c > 0$) { ←

 print (A[r][c])

 c--

}

while ($r > 0$) { ↑

 print (A[r][c])

 r--

}

$$TC = O(4 * (N-1)) \rightarrow \underline{O(N)}$$

$$SC = \underline{O(1)}$$



Spiral Matrix

mat[N][N]

	0	1	2	3	4	5
0	1	2	3	4	5	6
1	7	8	9	10	11	12
2	13	14	15	16	17	18
3	19	20	21	22	23	24
4	25	26	27	28	29	30
5	31	32	33	34	35	36

o/p → [1 , 2 , 3 , 4 , 5 , 6 , 12 , 18 , 24 , 30 , 36 , 35 , 34 , 33 , 32 , 31 , 25 , 19 ,

13 , 7 , 8 , 9 , 10 , 11 , 17 , 23 , 29 , 28 , 27 , 26 , 20 , 14 , 15 , 16 , 22 , 21]

Quiz :

13	14	12						
9								
0								
10								
	5	6	11					
				3	2	1		
					4			

o/p → 13 14 12 8 7 0

11 6 5 10 0 9

1 2 3 4



$r = 0$ $c = 0$

while ($N > 1$) {

 for ($i \rightarrow 1$ to $(N-1)$) { \rightarrow

 print ($A[r][c]$)

$c++$

 }

 for ($i \rightarrow 1$ to $(N-1)$) { \downarrow

 print ($A[r][c]$)

$r++$

 }

 for ($i \rightarrow 1$ to $(N-1)$) { \leftarrow

 print ($A[r][c]$)

$c--$

 }

 for ($i \rightarrow 1$ to $(N-1)$) { \uparrow

 print ($A[r][c]$)

$r--$

 }

$r++$

$c++$

$N -= 2$

}

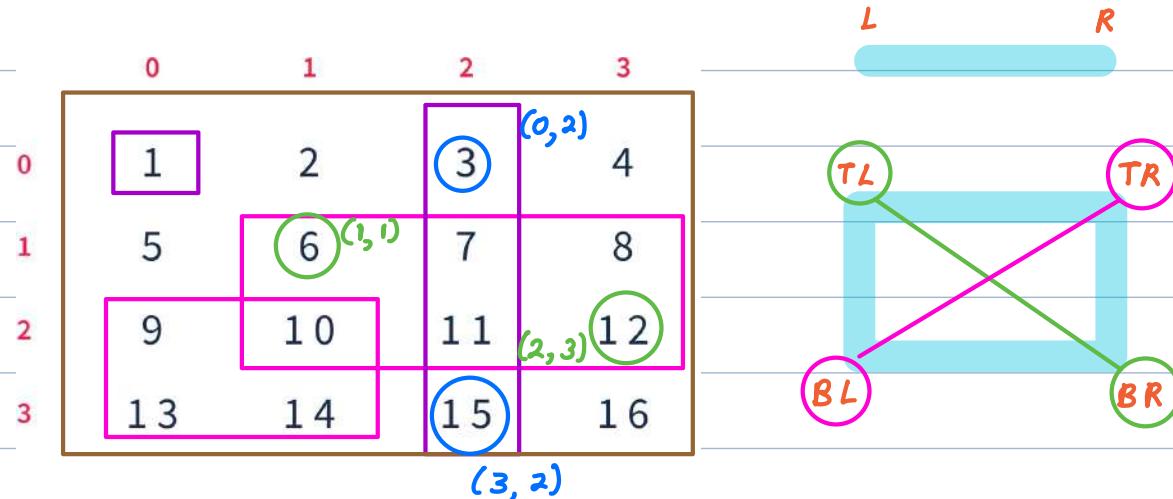
if ($N == 1$) print ($A[r][c]$)

$TC = \underline{O(N^2)}$

$SC = \underline{O(1)}$

Sub - Matrices

- Contiguous part of a matrix



< Question > : Given $\text{mat}[N][M]$. Find sum of all sub-matrix sums.

$$\begin{array}{cccc}
 & 0 & 1 & 2 \\
 \begin{matrix} 0 \\ 1 \end{matrix} & 4 & 9 & 6 \\
 & 5 & -1 & 2
 \end{array}
 \begin{array}{c}
 2 \times 3
 \end{array}
 \begin{array}{c}
 [4] 4 \quad [4 \ 9] 13 \quad [4] 9 \\
 [5] 5 \quad [9 \ 6] 15 \quad [5] 5 \\
 [9] 9 \quad [5 \ -1] 4 \quad [-1] 8 \\
 [-1] -1 \quad [-1 \ 2] 1 \quad [6] 6 \\
 [6] 6 \quad [4 \ 9 \ 6] 19 \quad [2] 8 \\
 [2] 2 \quad [5 \ -1 \ 2] 6
 \end{array}$$

$\# \text{subarray} = \frac{N \times (N+1)}{2}$

$$\begin{array}{c}
 \# \text{submatrix} = \frac{N \times (N+1) \times M \times (M+1)}{2 \ 2} \\
 \begin{array}{c}
 [4 \ 9] 17 \quad [9 \ 6] 16 \quad [4 \ 9 \ 6] 25 \\
 [5 \ -1] \quad [-1 \ 2] \quad [5 \ -1 \ 2]
 \end{array}
 \end{array}$$

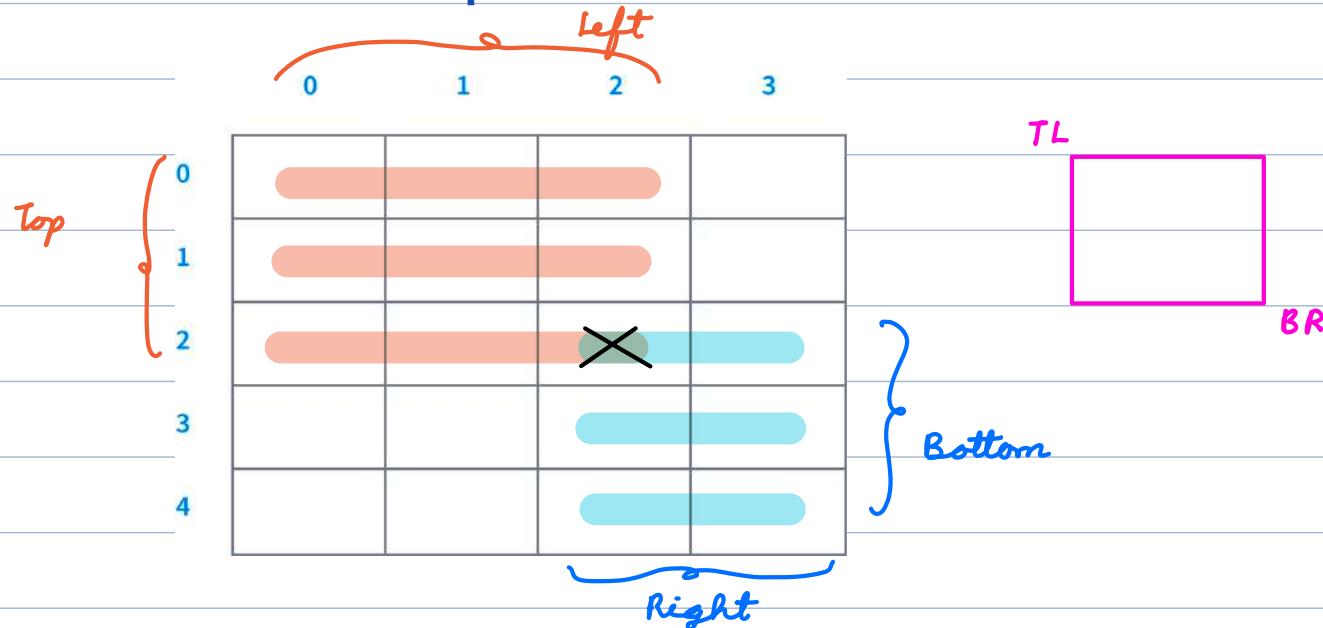
$\text{Ans} = 166$

$$2 \times 3 \rightarrow 3 \times 6 = 18$$

Contribution Technique \rightarrow $A_{ij} = \sum_{i,j} \text{contribution of } A[i:j, j]$

$A[i:j, j] * (\# \text{ submatrix}$
 $A[i:j, j] \text{ is part of})$

Contribution Technique



- In how many sub-matrices (2, 2) will be present?

$$9 \times 6 = 54$$

i, j

$$\text{Top} \rightarrow [0 \quad i] \rightarrow (i+1)$$

$$\text{Left} \rightarrow [0 \quad j] \rightarrow (j+1)$$

$$\text{Bottom} \rightarrow [i \quad N-1] \rightarrow (N-i)$$

$$\text{Right} \rightarrow [j \quad M-1] \rightarrow (M-j)$$

ans = 0

for $i \rightarrow 0$ to $(N-1)$ {

 for $j \rightarrow 0$ to $(M-1)$ {

 ans += $A[i][j] * (i+1) * (j+1) * (N-i) * (M-j)$

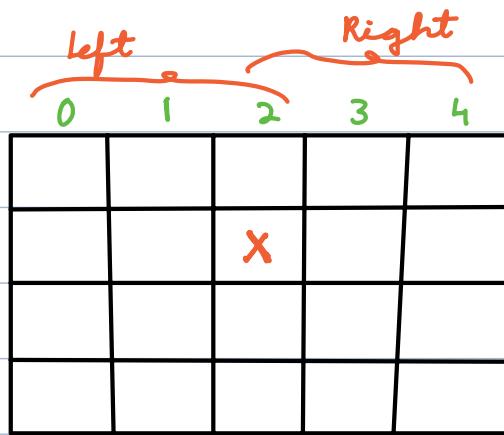
 }

}

return ans

$TC = O(N * M)$

$SC = O(1)$



$$\begin{aligned}
 & \text{Top} \rightarrow 2 \\
 & \text{Bottom} \rightarrow 3 \\
 & \text{Left} \rightarrow 3 \\
 & \text{Right} \rightarrow 3
 \end{aligned}
 \quad \left. \begin{aligned}
 & 2 * 3 * 3 * 3 \\
 & = 6 * 9 \\
 & = 54
 \end{aligned} \right\}$$

Today's Agenda

merge Intervals \rightarrow ②

find first +ve Integer
↑
missing

Intro

Microsoft → L60

Programming → Cofounder

Scalor → 2 years.
↳ Certification.

LND, Databases, H2D
Project

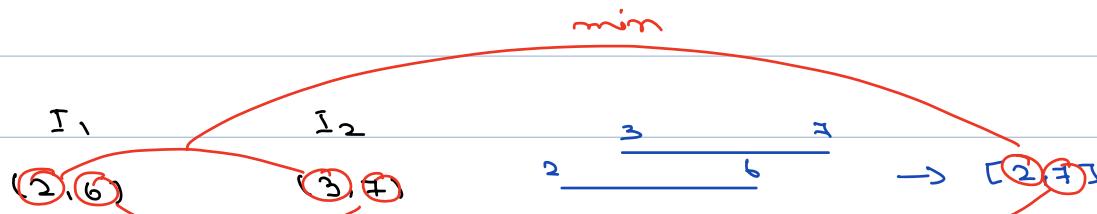
Merge Interval

2  5

Interval $[s, e] \rightarrow [2, 5]$

$[1, 5]$





$(2, 8)$ $(4, 6)$

$(3, 7)$ $(4, 10)$ $\frac{3}{4} \frac{7}{10} \rightarrow [3, 10]$

$(3, 6)$ $(6, 10)$ $\frac{3}{6} \frac{6}{10} \rightarrow [3, 10]$

$\frac{I_1}{(2, 5)}$ $\frac{I_2}{(8, 10)}$ $\frac{I_1}{2} \frac{I_2}{8} \frac{5}{10} \rightarrow \text{No overlap}$

$\frac{I_1}{(5, 8)}$ $\frac{I_2}{(1, 3)}$ $\frac{I_2}{1} \frac{I_1}{3} \frac{5}{8} \rightarrow \text{No overlap}$

I_1
 (s_1, e_1)

I_2
 (s_2, e_2)

$s_1 \quad I_1 \quad e_1 \quad s_2 \quad I_2 \quad e_2$

if $(s_2 > e_1)$

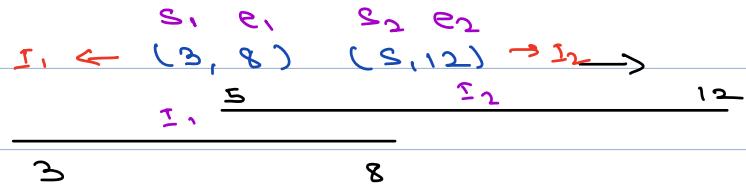
$s_2 \quad I_2 \quad e_2 \quad s_1 \quad I_1 \quad e_1$

if $(s_1 > e_2)$

if $(s_2 > e_1 \text{ || } s_1 > e_2) \rightarrow \text{no overlap}$

if overlap $\rightarrow \text{merged interval}$

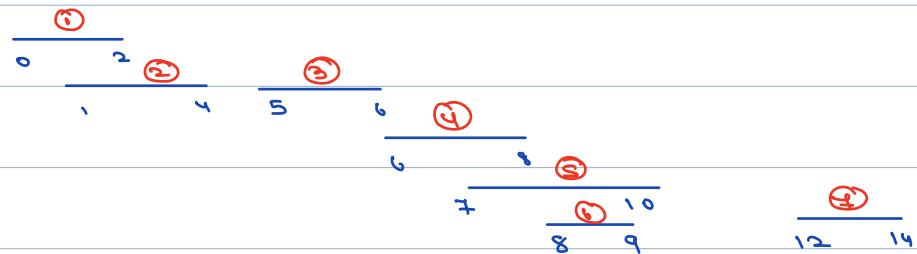
$[\min(s_1, s_2), \max(e_1, e_2)]$



Ques

You are given a collection of intervals A in a 2-D array format, where each interval is represented by a pair of integers $[start, end]$. The intervals are sorted based on their start values. Your task is to merge all overlapping intervals and return the resulting set of non-overlapping intervals.

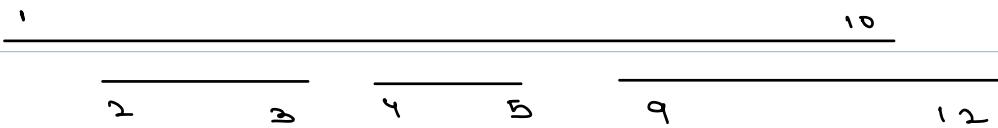
$$\text{Intervals } A = \{ (0, 2), (1, 4), (5, 6), (6, 8), (7, 10), (8, 9), (12, 14) \}$$



$$\text{Ans} = \{ (0, 4), (5, 10), (12, 14) \}$$

Ques 3

$$(1, 10), (2, 3), (4, 5), (9, 12)$$



\underline{I}_1
(s_1, e_1)

\underline{I}_2
(s_2, e_2)

$\underline{s_1 \quad \underline{I}_1 \quad e_1}$ $\underline{s_1 \quad \underline{I}_1 \quad e_1}$ $\underline{s_1 \quad \underline{I}_1 \quad e_1}$
 $\underline{s_2 \quad \underline{I}_2 \quad e_2}$ $\underline{s_2 \quad \underline{I}_2 \quad e_2}$ $\underline{s_2 \quad \underline{I}_2 \quad e_2}$

overlap $s_2 \leq e_1$

$\underline{s_1 \quad \underline{I}_1 \quad e_1}$

I.

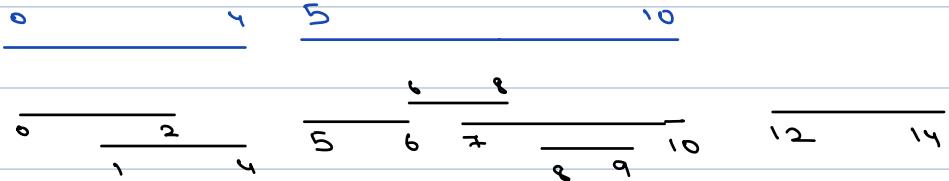
s.

e.

overlap $\rightarrow s_2 \leftarrow e_1$

arr[0] to

arr[0] to



Intervals $\Gamma_3 = \{ (0, 2), (1, 4), (5, 6), (6, 8), (7, 10), (8, 9), (12, 14) \}$

curr	next Available	After Merge	final ans
(0-2)	(1-4)	(0-4)	
(0-4)	(5-6)	—	(0-4)
(5-6)	(6-8)	(5-8)	(0-4)
(5-8)	(7-10)	(5-10)	(0-4)
(5-10)	(8,9)	(5-10)	(0-4)
(5-10)	(12-14)		$\begin{bmatrix} (0-4) \\ (5-10) \end{bmatrix}$
(12-14)			$\begin{bmatrix} (0-4) (5-10) \\ (12-14) \end{bmatrix}$

int[] arr \rightarrow $\underline{\text{int}}$. $\text{arr[0].l} \nearrow$
 Interval[] arr \rightarrow interval arr[0].l, arr[0].r,

Interval [] arr; // Given. $\frac{\text{curr}}{s} \frac{e}{e}$

list <Interval> ans;

curr_start = arr[0].s, curr_end = arr[0].e

for (i=1; i < m; i++) {

 if (arr[i].s <= curr_end) { // overlap

 // merge

 curr_end = max(curr_end, arr[i].e);

 } else {

 print [curr_start, curr_end]; // store it to ans;

 curr_start = arr[i].s;

 curr_end = arr[i].e;

}

print [curr_start, curr_end];

T.C $\rightarrow O(m)$

S.C $\rightarrow O(1)$



i = x[2]

Intervals \mathcal{I} = $\{ (0, 2), (1, 4), (5, 6), (6, 8), (7, 10), (8, 9) \}$

curr	most Available	After merge	final ans
(0, 2)	(1, 4)	(0, 4)	
(0, 4)	(5, 6)		(0, 4)
(5, 6)	(6, 8)	(5, 8)	
(5, 8)	(7, 10)	(5, 10)	
(5, 10)	(8, 9)	(5, 10)	

Ques

You have a set of non-overlapping intervals. You are given a new interval [start, end], insert this new interval into the set of intervals (merge if necessary). You may assume that the intervals were initially sorted according to their start times.

N = 9

(1, 3)



(4, 7)



(10, 14)



(16, 19)



(21, 24)



(27, 30)



(32, 35)



(38, 41)



(43, 50)

new interval = $[12, 22]$

N=9

(1,3) $(12, 22) \rightarrow (1, 3)$

(4,7) $(12, 22) \rightarrow (4, 7)$

(10,14) $(12, 22) ; (10, 22)$

(16,19) $(10, 22) ; (10, 22)$

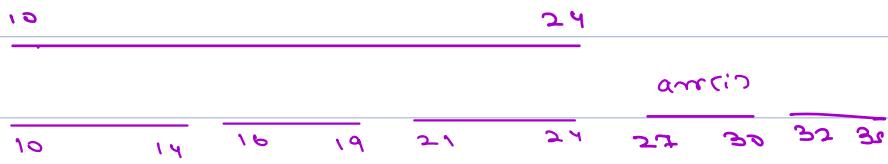
✓ (21,24) $(10, 22) ; (10, 24)$

✓ (27,30) $(10, 24) \rightarrow (10, 24)$

✓ (32,35)

✓ (38,41)

✓ (43,50)



Ex 2

(1,5) $(12, 24) \rightarrow (1, 5)$

(8,10) $(12, 24) \rightarrow (8, 10)$

(11,14) $(12, 24) ; (11, 24)$

(15,20) $(11, 24) ; (11, 24)$

(21,24) $(11, 24) \rightarrow (11, 24)$

11 new 24



new
me
am
cint
e

1. $C \rightarrow O(n)$
2. $C \rightarrow O(1)$

$(m_S, m_E) \rightarrow$ new interval.

for (i=0; i < n; i++) {

C_{Internal} = $\alpha n T_i J$;

if (ns > cInteval.e) {

Point (cInteval); }

else if (cintemal.s > me) {

```
print(m,me);
```

```
for(s=i; s<n; s++) {
```

```
|     point (arr [j]);
```

neben;

pine

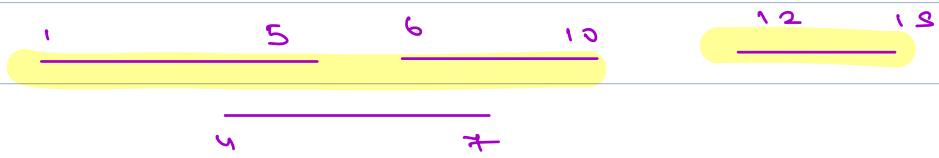
$ms = \min(C_{\text{Interval}}, ms);$

```
me = May (c[internal.e, mE]);
```

3

2

Point (m₁, m₂);



8:27 am → 8:35 am

Ques

→ positive Integer

Given an unsorted array of integers, Find first missing Natural Number.

$$\text{arr}[5] = \{3, -2, 1, 2, 7\} \rightarrow 4$$

$$\text{arr}[7] = \{-9, 2, 6, 4, -8, 1, 3\} \rightarrow 5$$

$$\{-2, 4, -1, -6, 3, 7, 8, 4, -3\} \rightarrow 1$$

$$\{1, 0, -5, -6, 4, 2\} \rightarrow 3$$

$$\{1, 2, 5, 6, 4, 3\} \rightarrow 7$$

$$\{-4, 8, 3, -1, 0\} \rightarrow -4, -1, 0, 3, 8$$

$$\{4, 2, 1, 3\} \rightarrow s$$

$$[7, 8, 3, 4, 6] \rightarrow 1$$

Observation $\rightarrow \underline{n} \rightarrow \frac{\text{to}}{\underline{n+1}}$

$n \rightarrow 5 \rightarrow [1, 2, 3, 4, 5]$

$n \rightarrow 6 \rightarrow [1, 6, 3, 2, 4]$

Stems $\rightarrow (\underline{1 \dots 5}) \rightarrow 5$

Do I m v :- check all the numbers from 1 to n+1

```
for (i=1;i<=m;i++) {
```

→ check if in array -

↳ i is missing \rightarrow return n.

1. C30m²)

$\Delta, C \supset O(1)$

een even $n+1$

Soln 2:-

Add all elements to the set and check if element(1 to N) is present or not.

T.C \rightarrow O(m)

$$\delta, c \rightarrow 0(n)$$

Soln :-

Sort & check

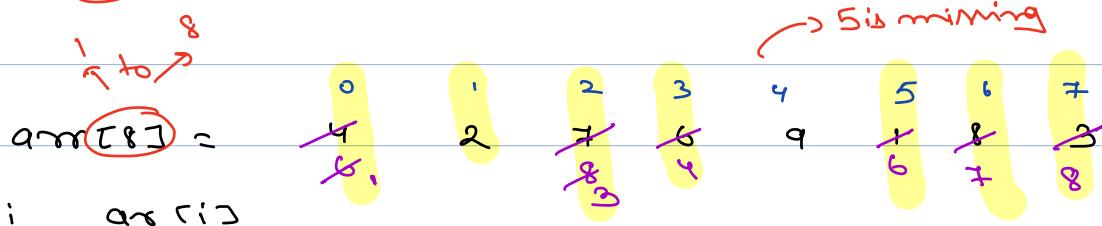
$$T \in \Theta(m \log n)$$

Solm 4:-

T.C \rightarrow O(m), S.C \rightarrow O(1)

<u>val</u>	<u>idx</u>	$n \rightarrow 0 \rightarrow 1$
1	0	$n \rightarrow 0 \rightarrow 1$
2	1	$n \rightarrow 1 \rightarrow n$
3	2	
4	3	
x	$x-1$	

valid.



0 4 $\Delta(0,3)$ $\Delta(0,5)$

1 2 Det

Ans \rightarrow 5

2 7 $\Delta(2,6)$ $\Delta(2,7)$

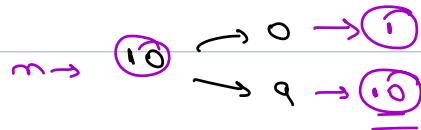
3 4

4 9 ignore

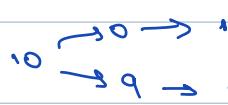
5 Det

6 Det

7 Det



1 to 10



$\text{arr}[10] =$	0	1	2	3	4	5	6	7	8	9
	5	-14	6	7	9	-10	5	6	7	8
	9	2	-20	2	-14					
	1	3								
	2									

i $\text{arr}[i] = \text{val}$

Ans

0 5 $\text{arr}[0, 4)$

$\text{arr}[i, \text{val}-1);$

1 ignore

2

3

for ($i=0; i < n; i++$) { valid

 while ($\text{arr}[i] > 0 \ \& \ \& \ \text{arr}[i] < n \ \& \ \&$

$\text{arr}[i] \neq i$ { not set

$\text{val} = \text{arr}[i];$

 if ($\text{arr}[i] == \text{arr}[\text{val}-1]$) { break }

$\text{arr}[i, \text{val}-1);$

 3

3

Generate & find missing no'.

for ($i=0; i < n; i++$) {

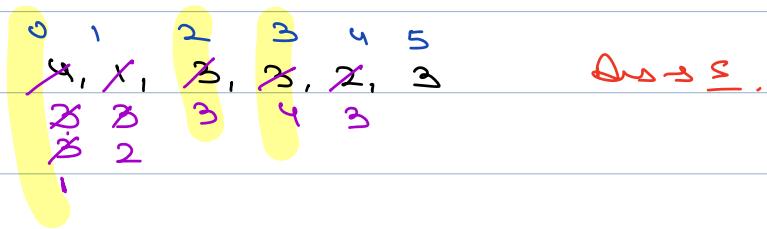
 if ($\text{arr}[i] \neq i+1$)

$\text{arr}[i, i+1];$

3

$\text{arr}[i+1, n+1];$

Duplicate in arr,



i arr[i]

0 4

T. $C \rightarrow O(n)$

S. $C \rightarrow O(1)$

$n = 10$

m^2

100

for (i = 0; i < n; i++) {

 while (arr[i] > 0 && arr[i] <= n &&

 arr[i] != i) {

 arr[i] = arr[i];

 val = arr[i];

 if (arr[i] == arr[val - 1]) { break; }

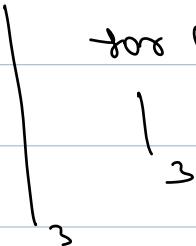
 arr[i] = arr[val - 1];

 }

}

for ($i=0$; $i < n$; $i++$) {

 for ($j=0$; $j < m$; $j++$) {



 m2
 $m=10$

