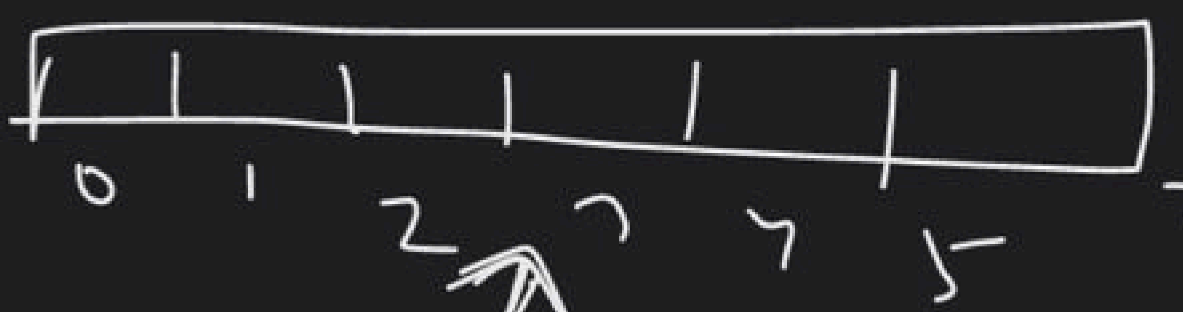
Two - Dimensional Arrays

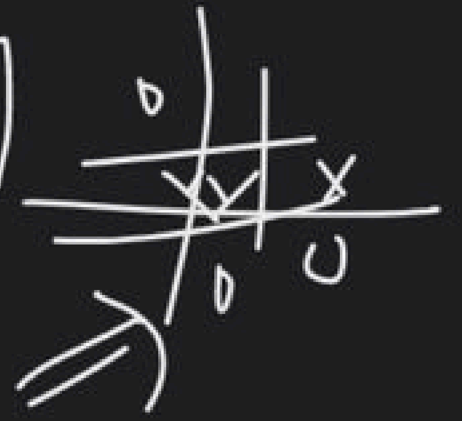
Foundation Course on Data Structures & Algorithm - Part I

Love Babbar • Lesson 9 • Apr 30, 2022

→ 2D Arrays :-

1D array

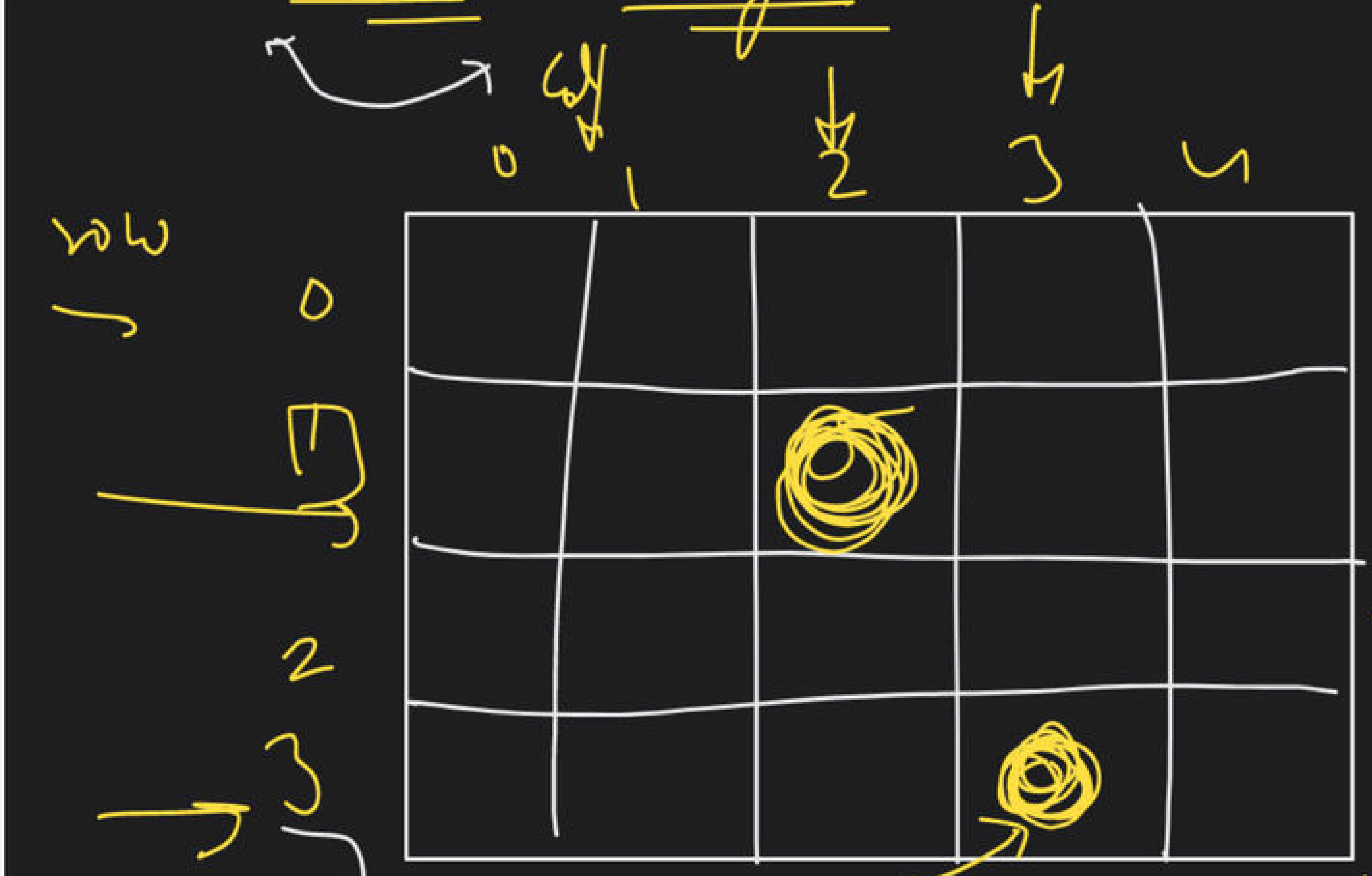




discord

O/X

Ti..



arr[1][2]

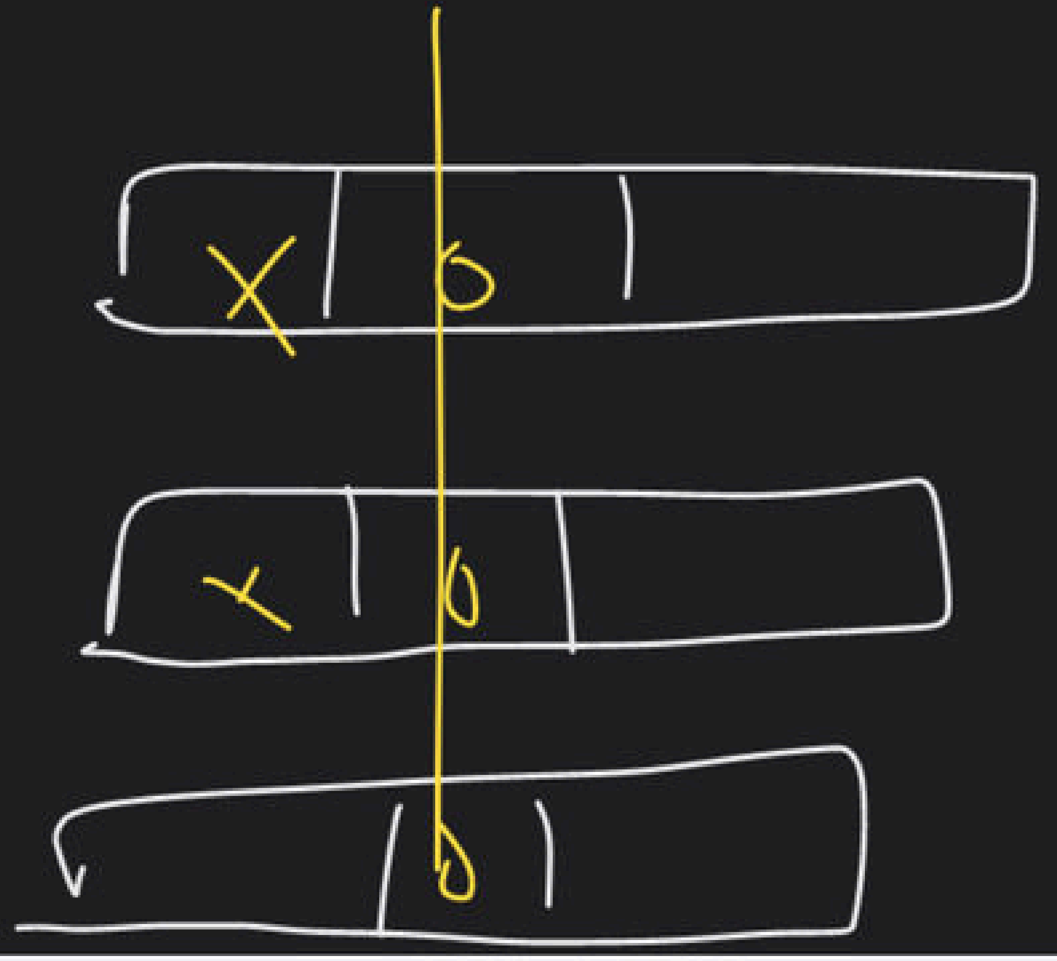
row col

arr[3][3]

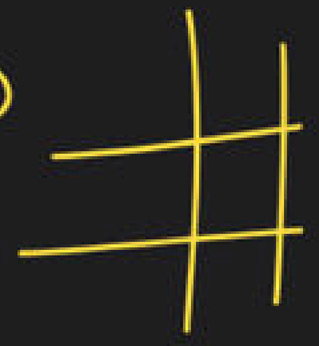
row colsize

→ int arr [5] [10] ;

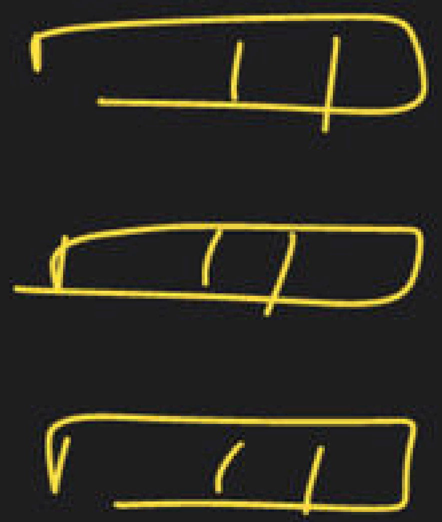
→ 1000 r
1000 col → int babbar[1000][1000]



3x3



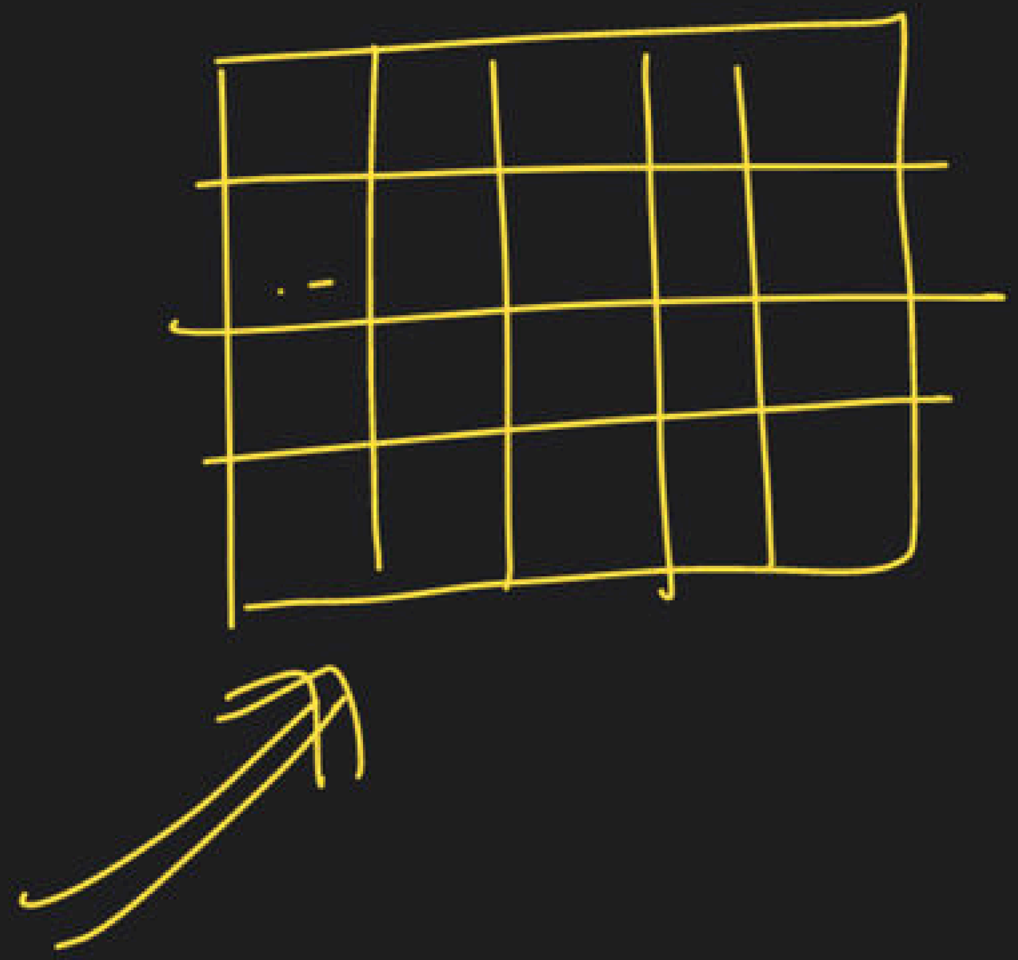
3 row, 3 col —

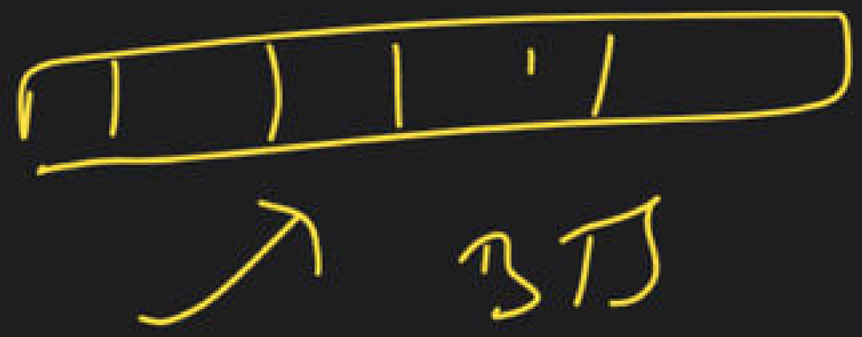


1000 row

1000 kol → Accha

2D





3D

→ Code

- → creation ✓
- → access ✓
- → fuction → o/p ✓

2D array is stored in memory ?

total no of column → $\boxed{c \times i + j}$

Memory $= c \times i + j$

$= 3 \times 1 + 2$

$= 5$

i j

arr[1][2]

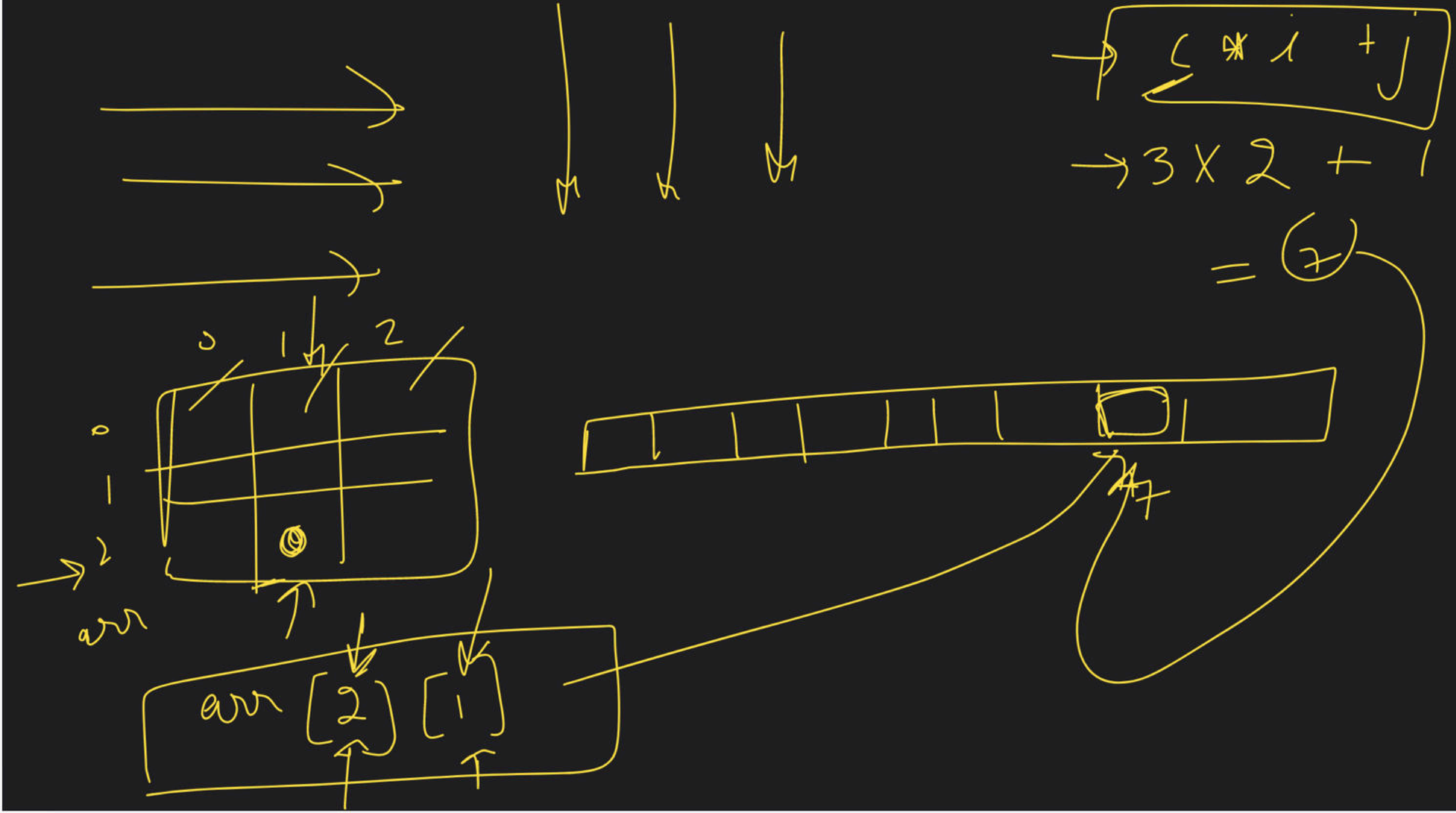
3 col

3 row

	0	1	2
0	3	4	5
1	6	7	8
2	9	10	11

3	4	5	6	7	8	9	10	11
0	1	2	3	4	5	6	7	8

1D Array


c * i + j
→ 3 X 2 + 1
= 7
0
1
2
0
1
2
arr
arr [2] [1]
7

→ i/p → 2D array

5	1	3
4	6	7
10	9	11

why?

target = 13 → Present or not

o/p → T/F

T.C → ?

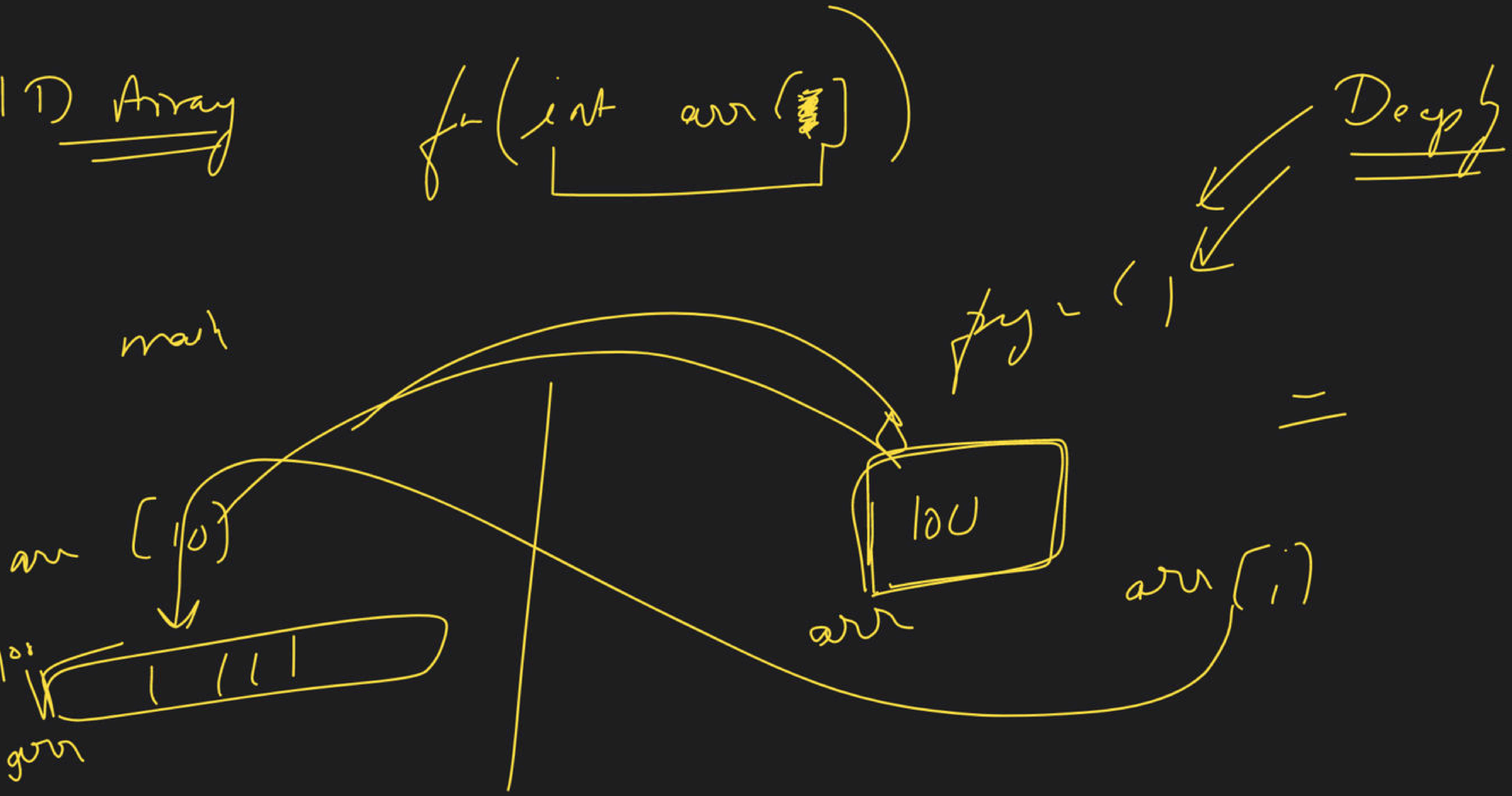
→ Bo rec → break → code → ?

→ T.C → ?

why

int arr[] [size];

1D Array



→ creation

→ array / input

→ function

⇒ Yes or No

(I) print column wise sum

(II) max column wise sum

i/p

$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 3 & 5 & 11 & 2 \\ 1 & 7 & 9 & 8 & 0 \\ 2 & 5 & 0 & 1 & 3 \end{array}$$

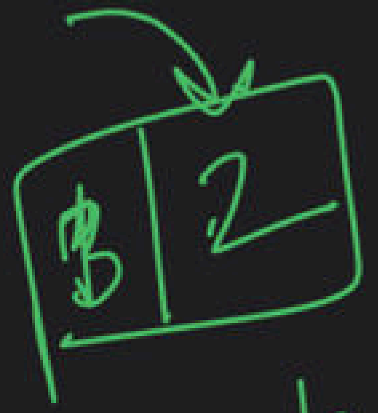
o/p → 15 14 20 5 → maxi → 20

→ fun (int arr[3][5])

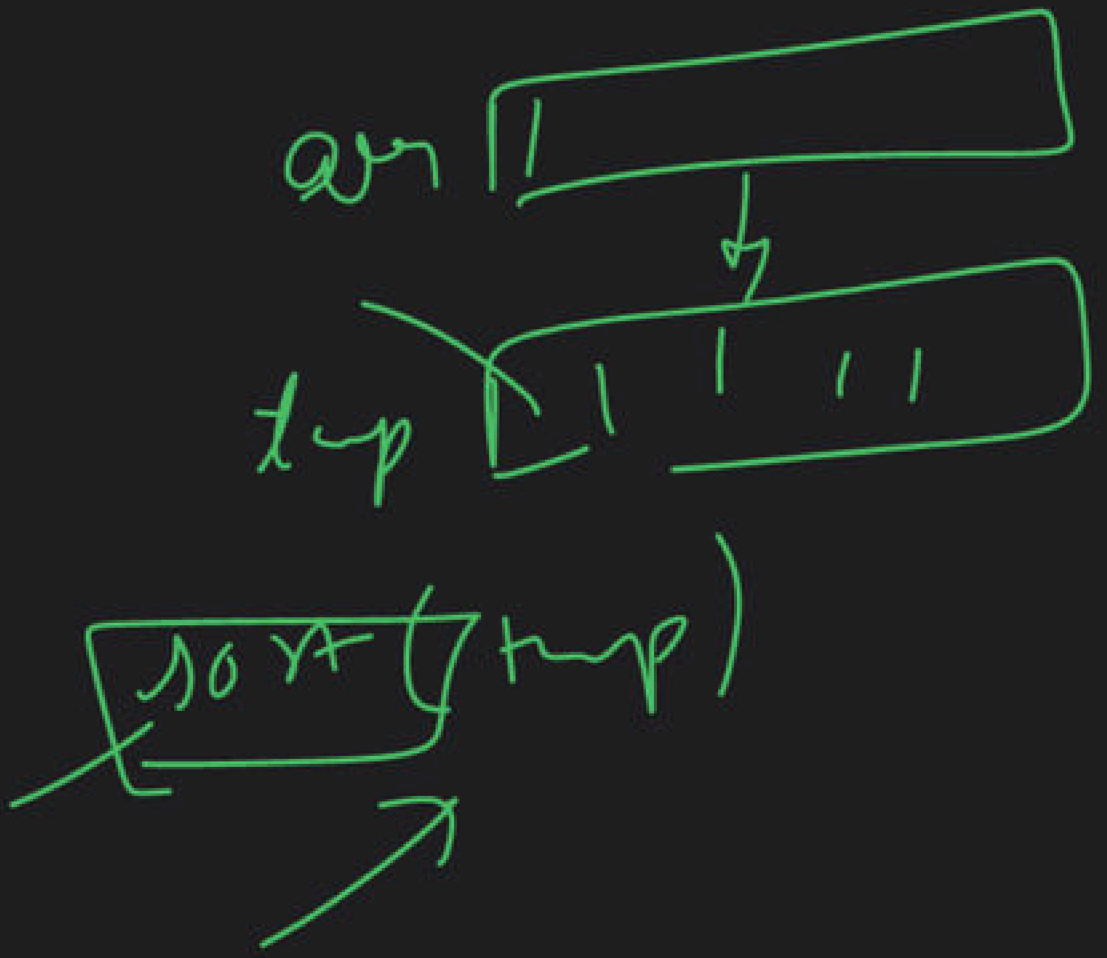
→ f (int arr[ ][5])

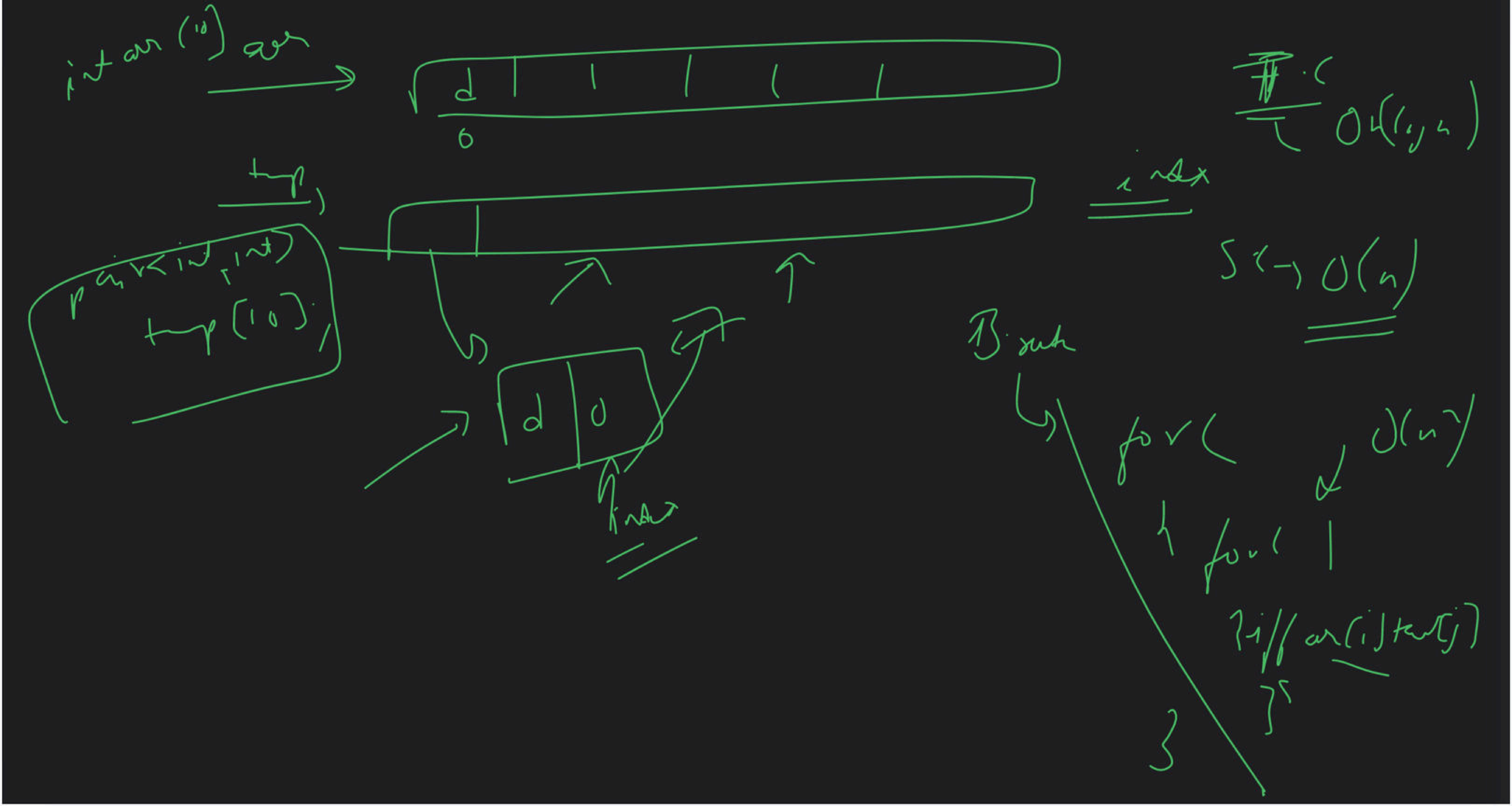
Doubt ?

pairSum:- → sort
└→ 2 ptr



nlogn




int arr[10] asc
2
0
temp,
pair<int,int> temp[10];
2
0
index
II: C
O(n log n)
index
S.C → O(n)
B: brute
for(
{ for(
}if(arr[i]==temp[j]
}}
}
O(n^2)

→ i/p → 2D array → Rotate by 90 degree

OA
Int
↓
[illegible]
repeat

arr

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

90

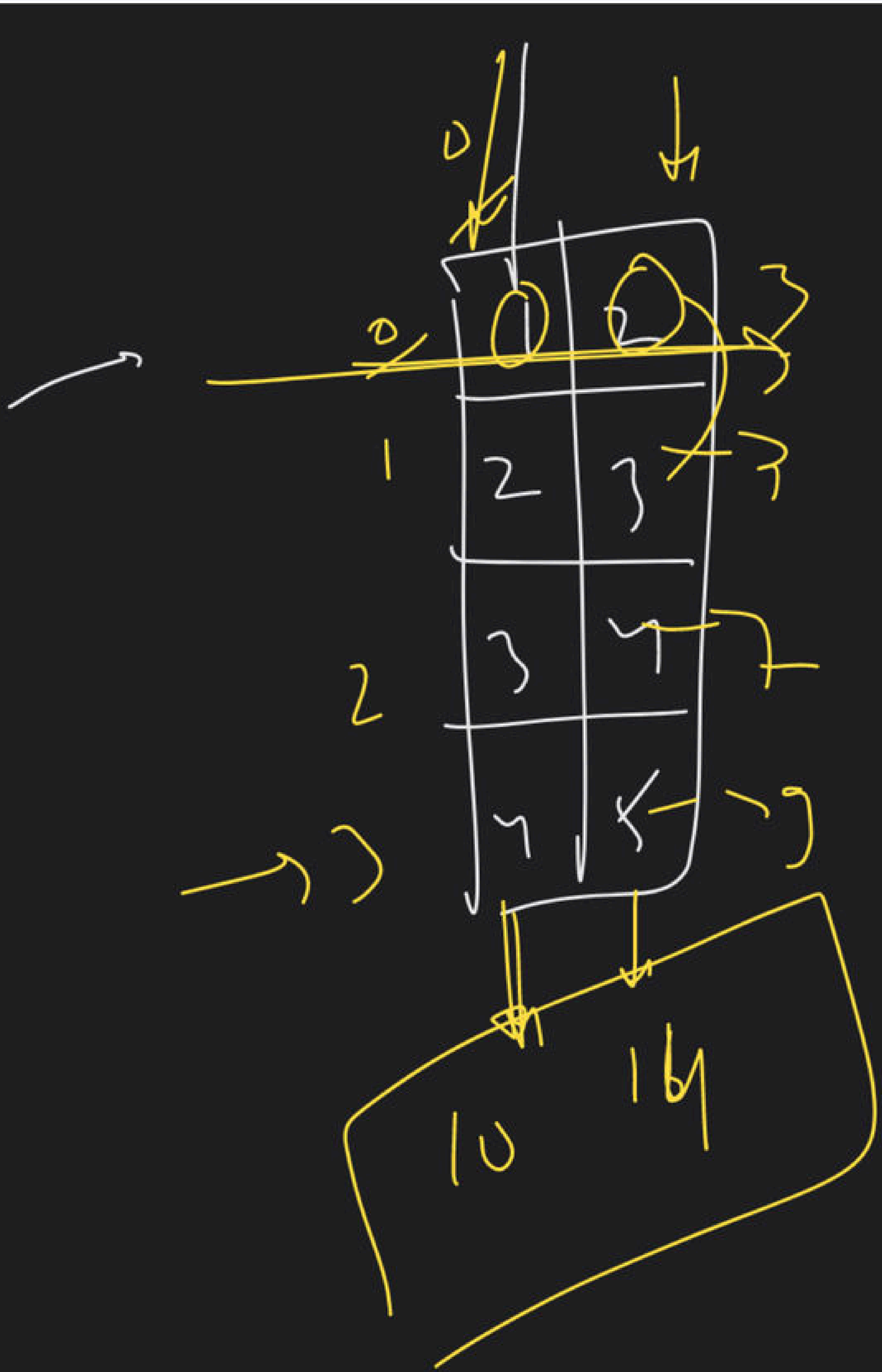
90
Rotate

o/p

7	4	1
8	5	2
9	6	3

1	4	7
2	5	8
3	6	9

transpose → row ⟷ col



arr[0][0]

+

arr[1][0]

+

arr[2][0]

+

arr[3][0]

$\downarrow$

10

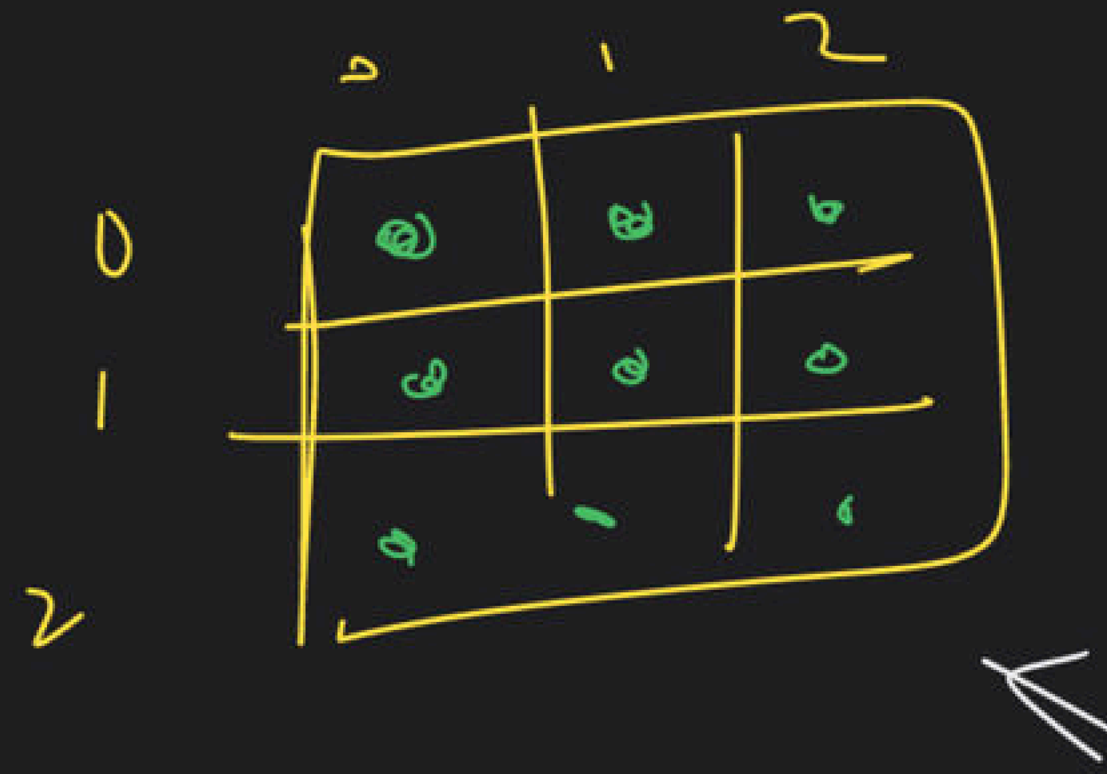
row wise

sum

arr[0][0]

+ arr[0][1] → $\boxed{3}$

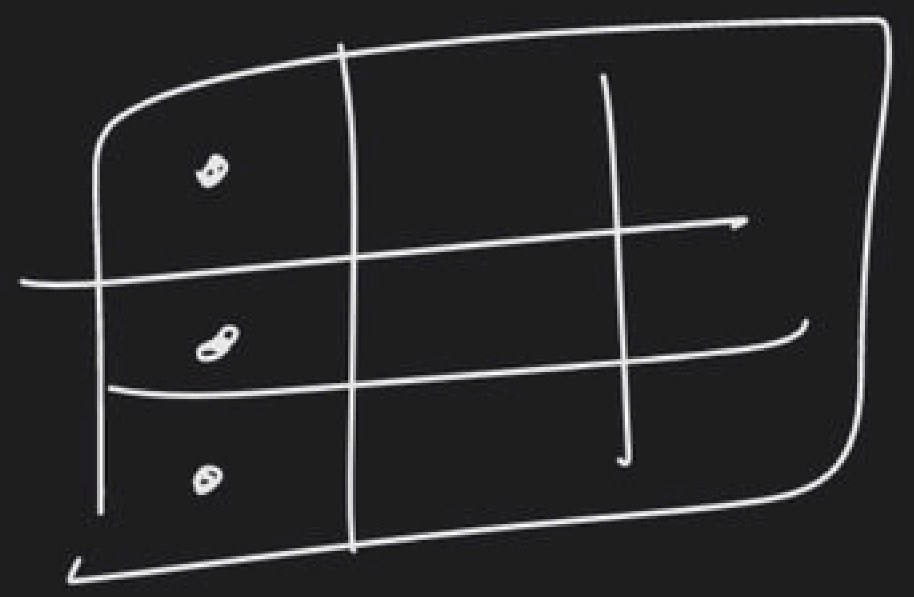
DRY RUN



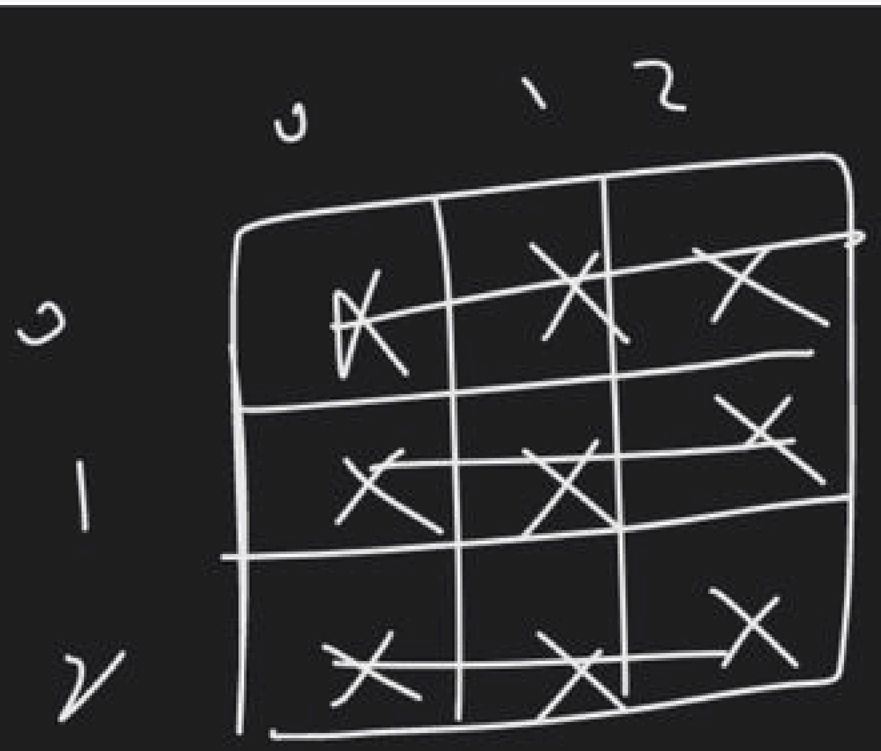
i = 0 → n

j → 0 → n

arr[i][j]



arr[j][i]



n = 3

```
for (int i -> 0 -> n)
{
    for(int j -> 0 -> n)
    {
        - arr[i][j]
    }
}
```

i = 0 (0,0) (0,1) (0,2)

↳ j = 0, 1, 2

i = 1 (1,0) (1,1) (1,2)

↳ j = 0, 1, 2

i = 2 (2,0) (2,1) (2,2

↳ j = 0, 1, 2

Transpose

(0, 1)   (1, 0)

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & \boxed{2} & 3 \\ 1 & 4 & 5 & \boxed{6} \\ 2 & 7 & 8 & 9 \end{array}$$

(1, 2)

transpose

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 4 & 7 \\ 1 & \boxed{2} & 5 & 8 \\ 2 & 3 & 6 & 9 \end{array}$$

(2, 1)

transpose ( )

```
~~tras~~
{ for(0 -> 4
    for(0 -> i
        { swap
    }
}
```

back to original

swap (arr[i][j] , arr[j][i])

→ Rotate by 90°

→ transpose ( ) → O(n²)

2 loop

→ reverse ( ) → O(n²)

↓

$\boxed{o/p}$

→ main → $\boxed{X}$ Ci (

1%

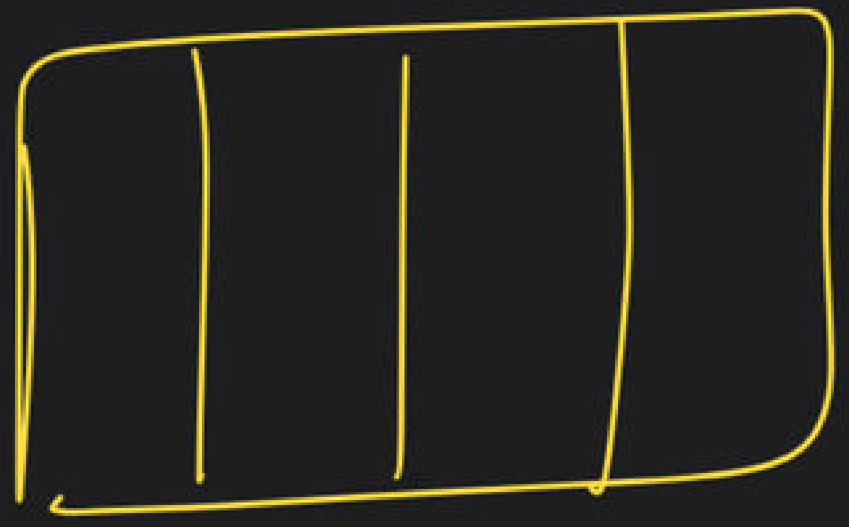
T.C → O(n²) + O(n²)

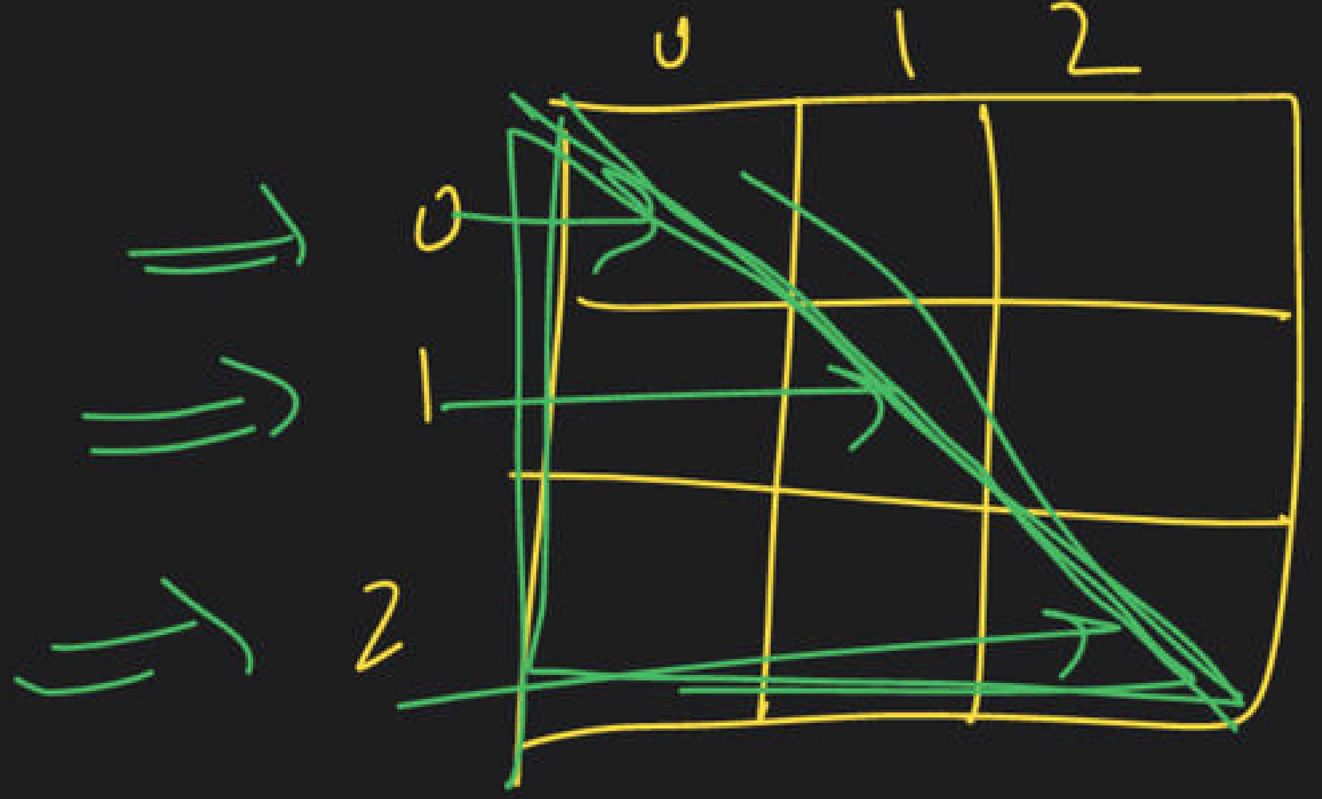
→ O(n²)

ar .25 → Q

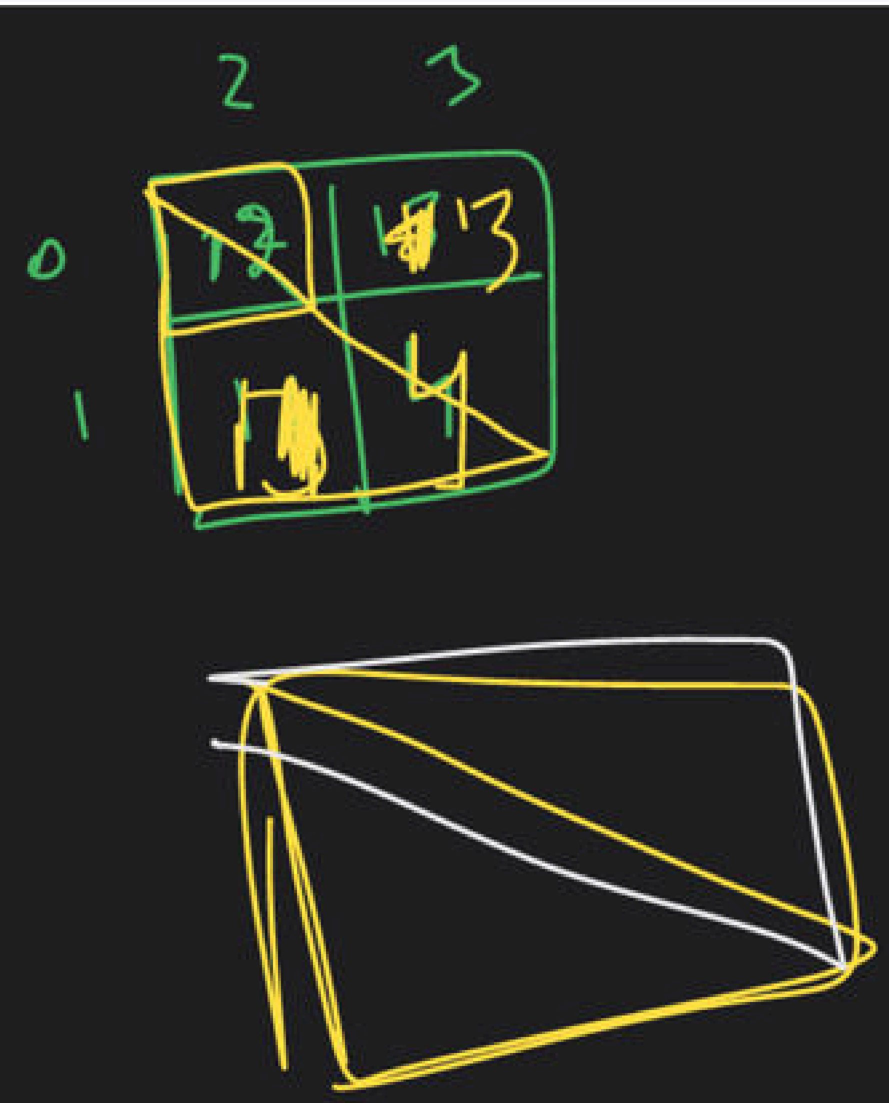
an 2D → 3.25

Please





```
for ( i -> 0 -> i<n)
{
    for (j = 0 ; j < i
```



```
for (i -> 0 -> n)
{
    for (j -> 0 -> i)
    {
        swap (arr[i][j], arr[j][i])
    }
}
```

i = 0, j = 0

arr[0][0], arr[0][0]

i = 0, j = 1

arr[0][1], arr[1][0]

i = 1, j = 0

arr[1][0] arr[0][1]

i = 1, j = 1

arr[1][1] arr[1][1]

rotate

transpose ( )

Reverse ( ) → your zimmedari - 7 lines

HW

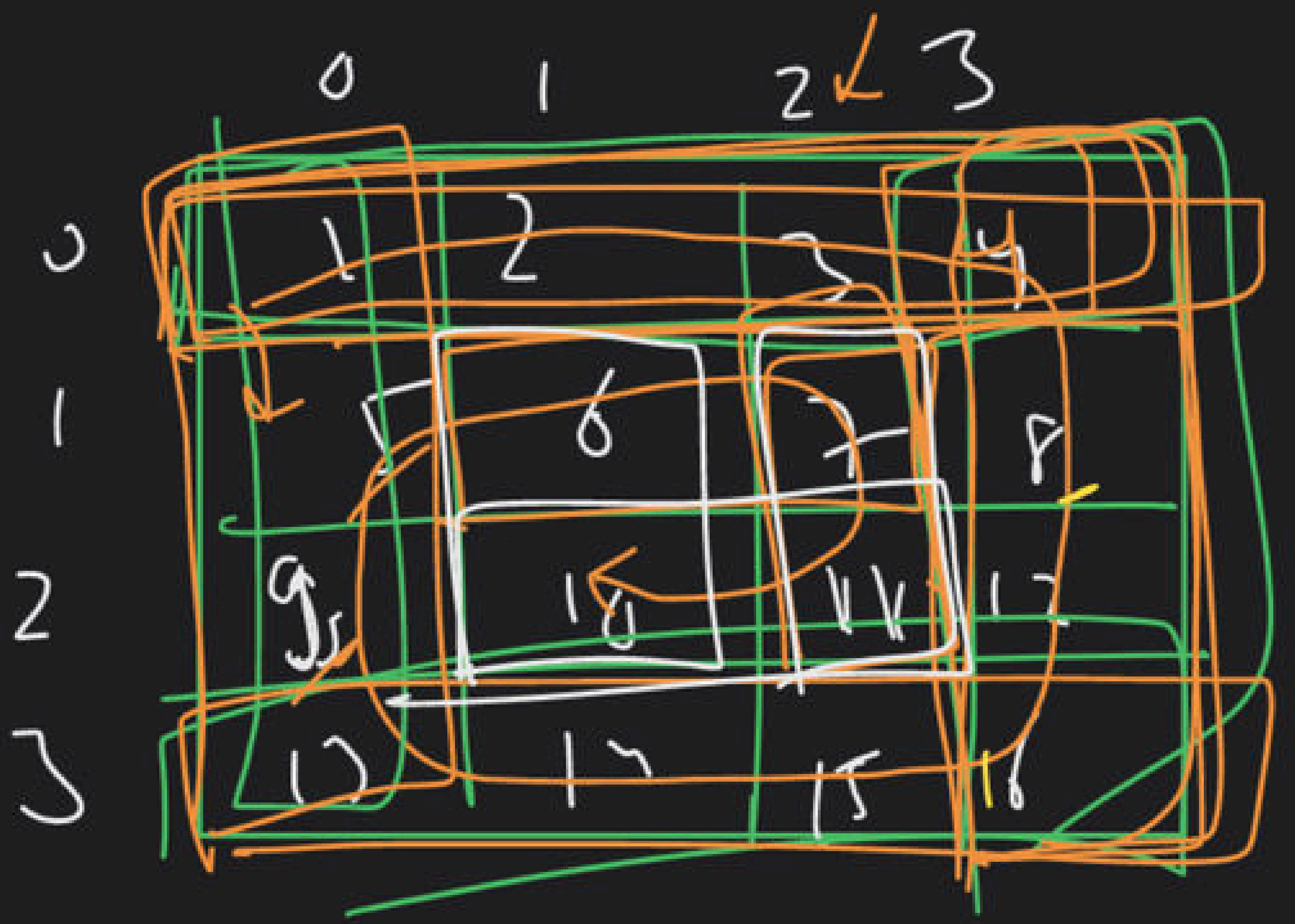
180°

270

→ Spiral Matrix

Ways to avoid duplicat

adobe
amazon
oyo
expedia



	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12
3	13	14	15	16

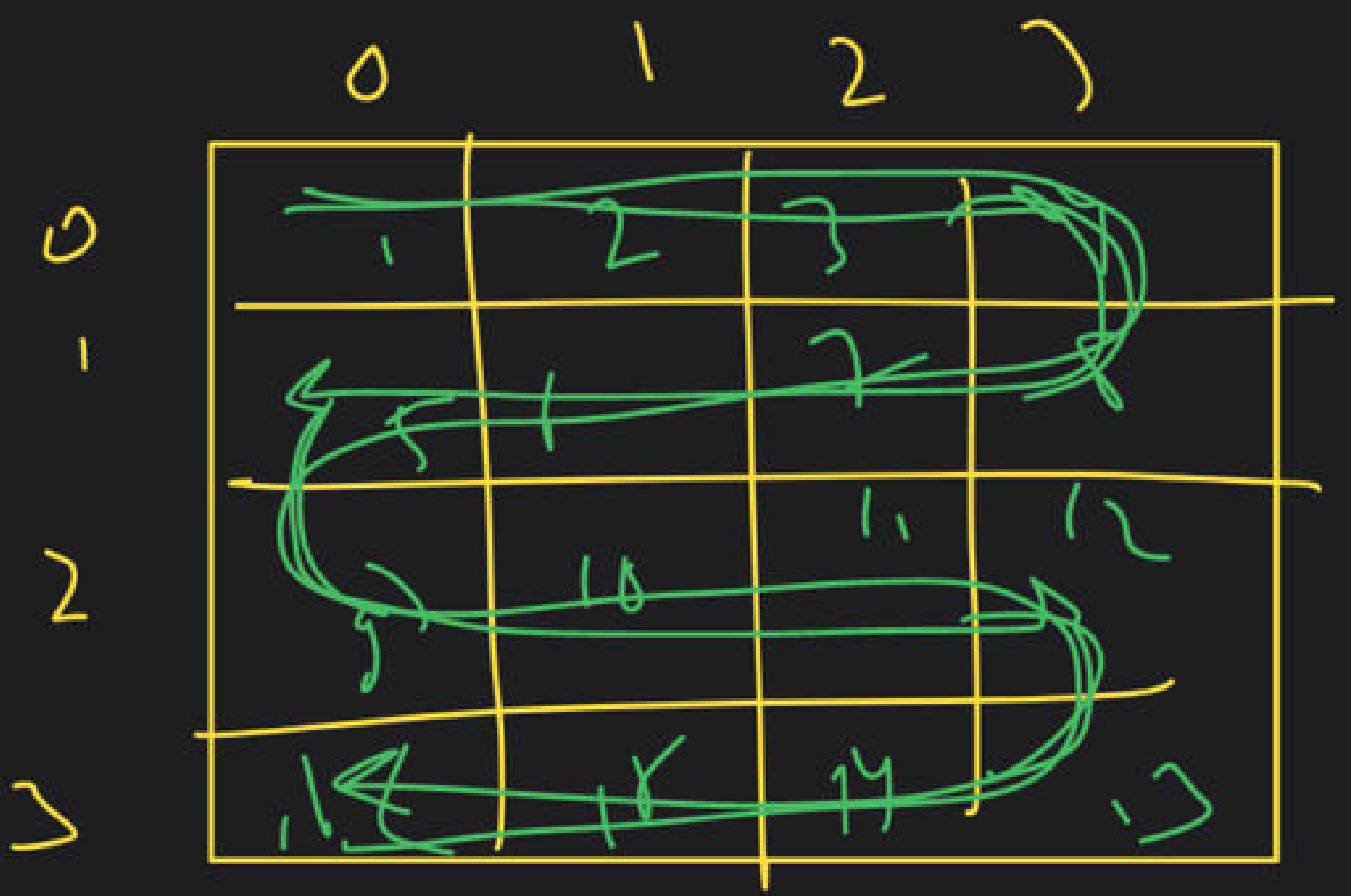
1 2 3 4 8 12 16
15 14 13 9 5
6 7 11 10

Logic:-

startingRow = 0
+1
startingCol = 0
+1
endingRow = n - 1
endCol = n - 1

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

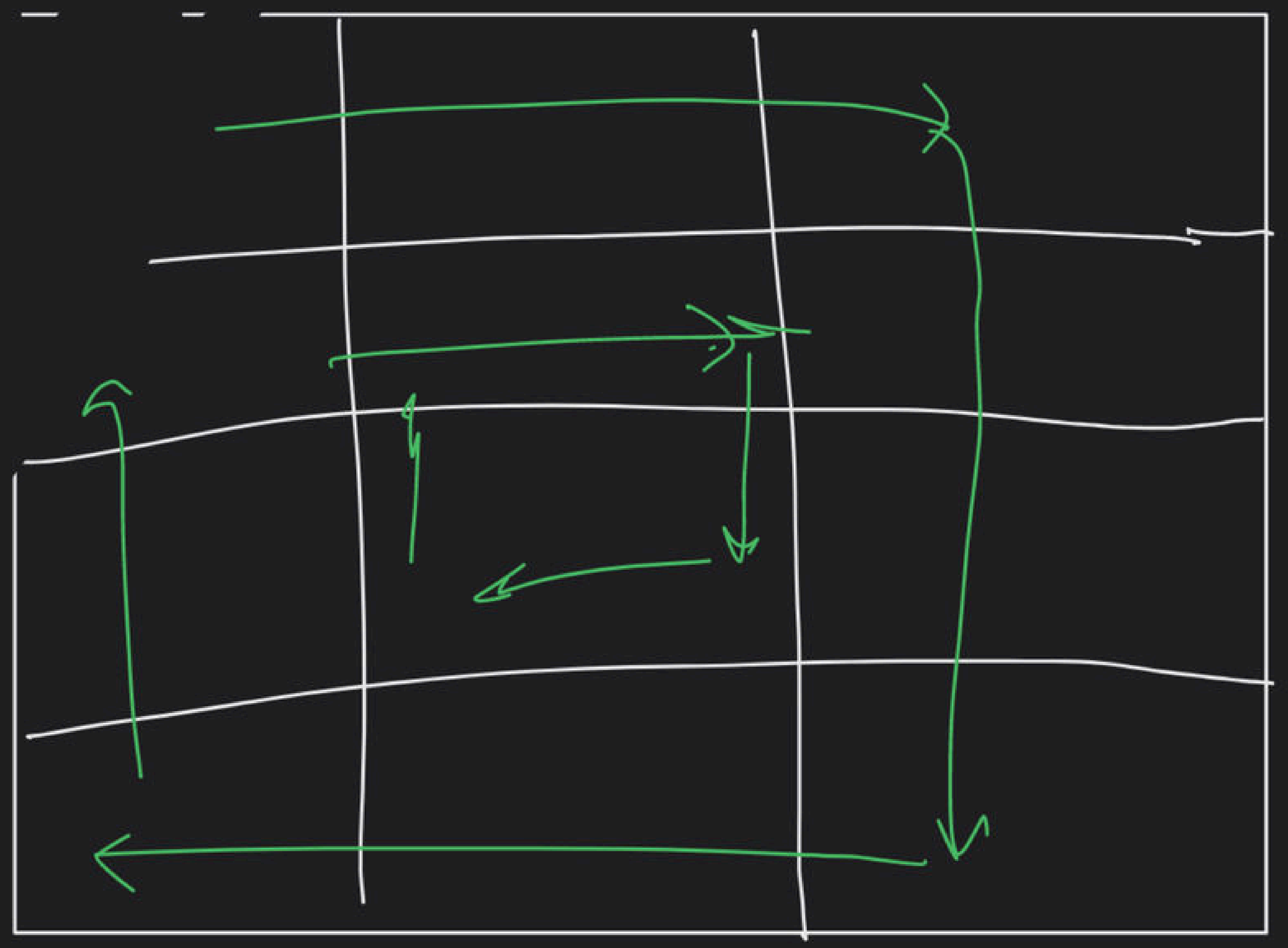
→ Wave print



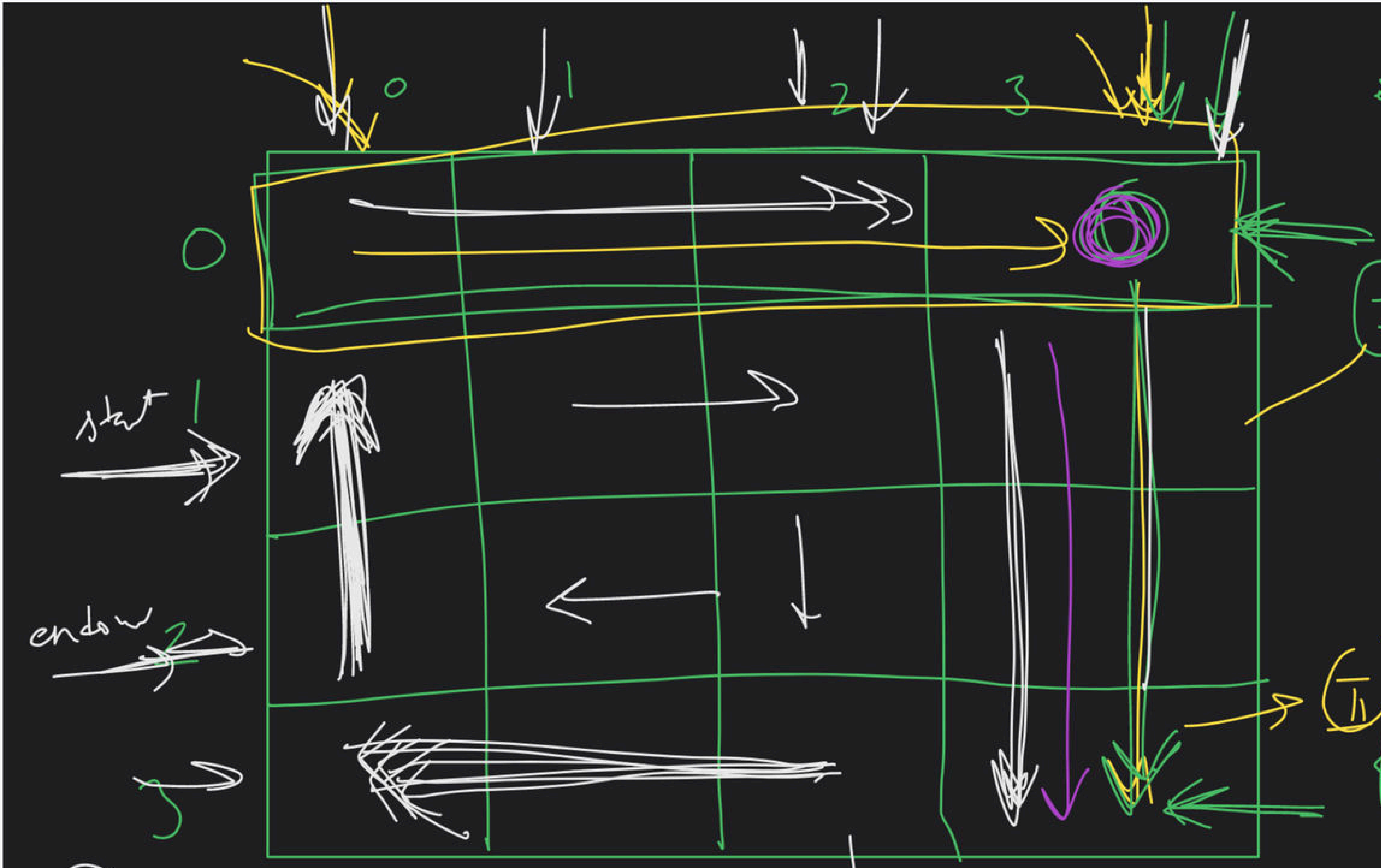
	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12
3	16	15	14	13

o/p

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



startRow = 0 , startCol = 0 , endRow = n-1 , endCol = n-1



I // starting row

```
for (i -> startCol -> endCol
{ count +1
  cout << arr[startRow][i]
}
startRow ++ ;
```

II

```
for (i -> startRow -> endRow)
{ count + 1
  cout << arr[i][endCol])
}
endCol -- ;
```

III

```
for (i = endCol -> startCol
{ count + 1
  cout << arr[endRow](i)
}
endRow --
```

4...

(IV)

```
for( i -> endRow -> startRow
          count++
  {
      cout << arr[i][startCol];
  }
  startCol++
```

count = 0

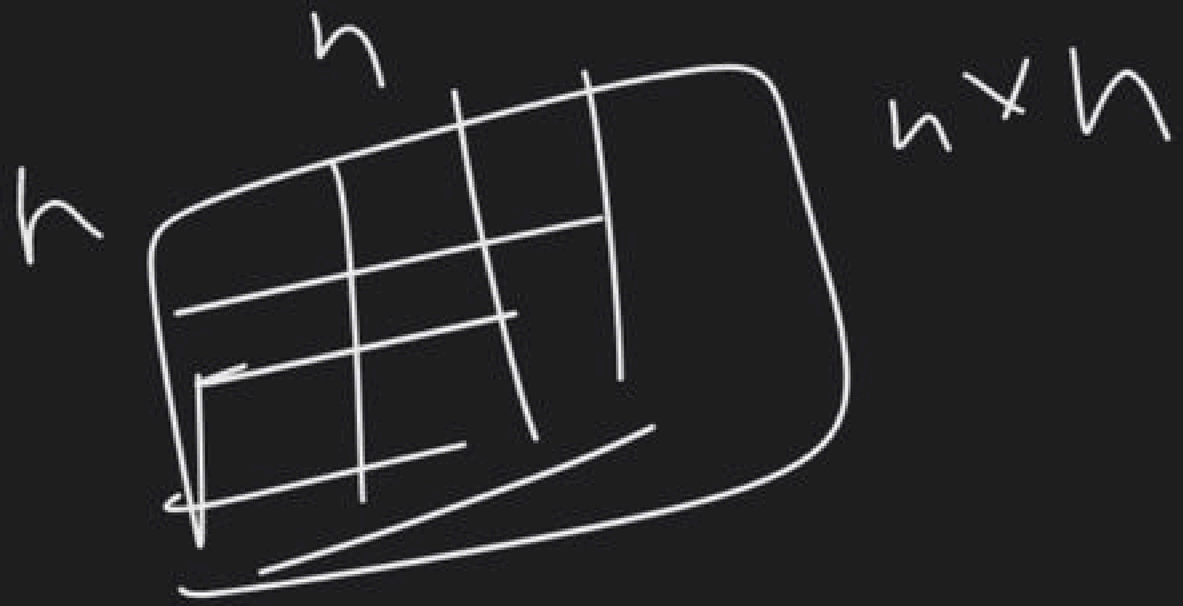
count++

int total = n*n

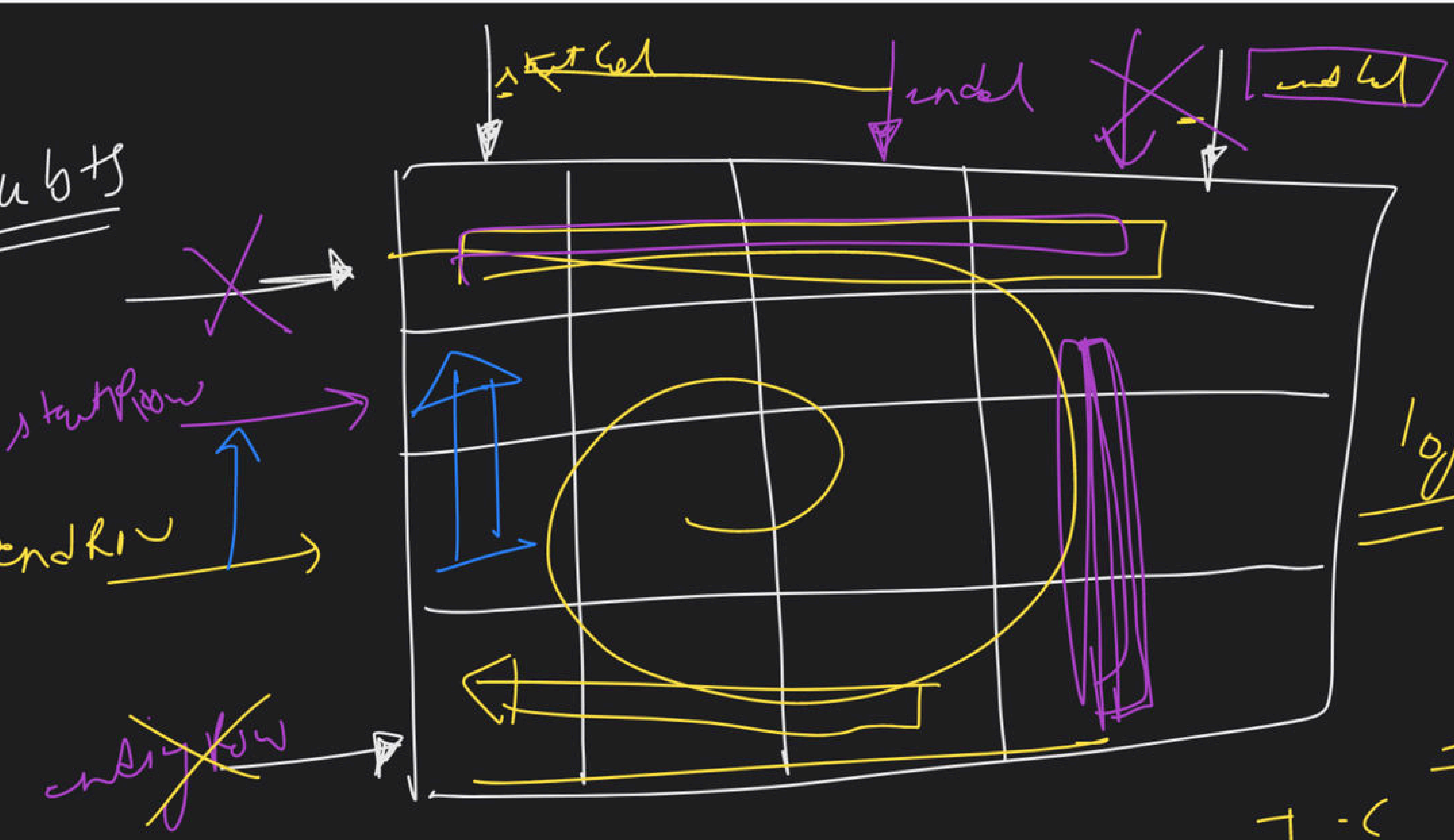
total

```
while ( count != total
        total -- )
{
    I

    II

    III

    IV
}
```

code



→ Doubts



Algo

Scan

sorting

DRY RUN

3-4 Ex

total = n × n = n^2

S.C →

BS

log n

O(n)

T.C → O(n^2)

S.C → O(1)

```
int count = 0                    → total = n×n

while ( count < total )
{
    I
    II
    III
    IV
}
```

total--

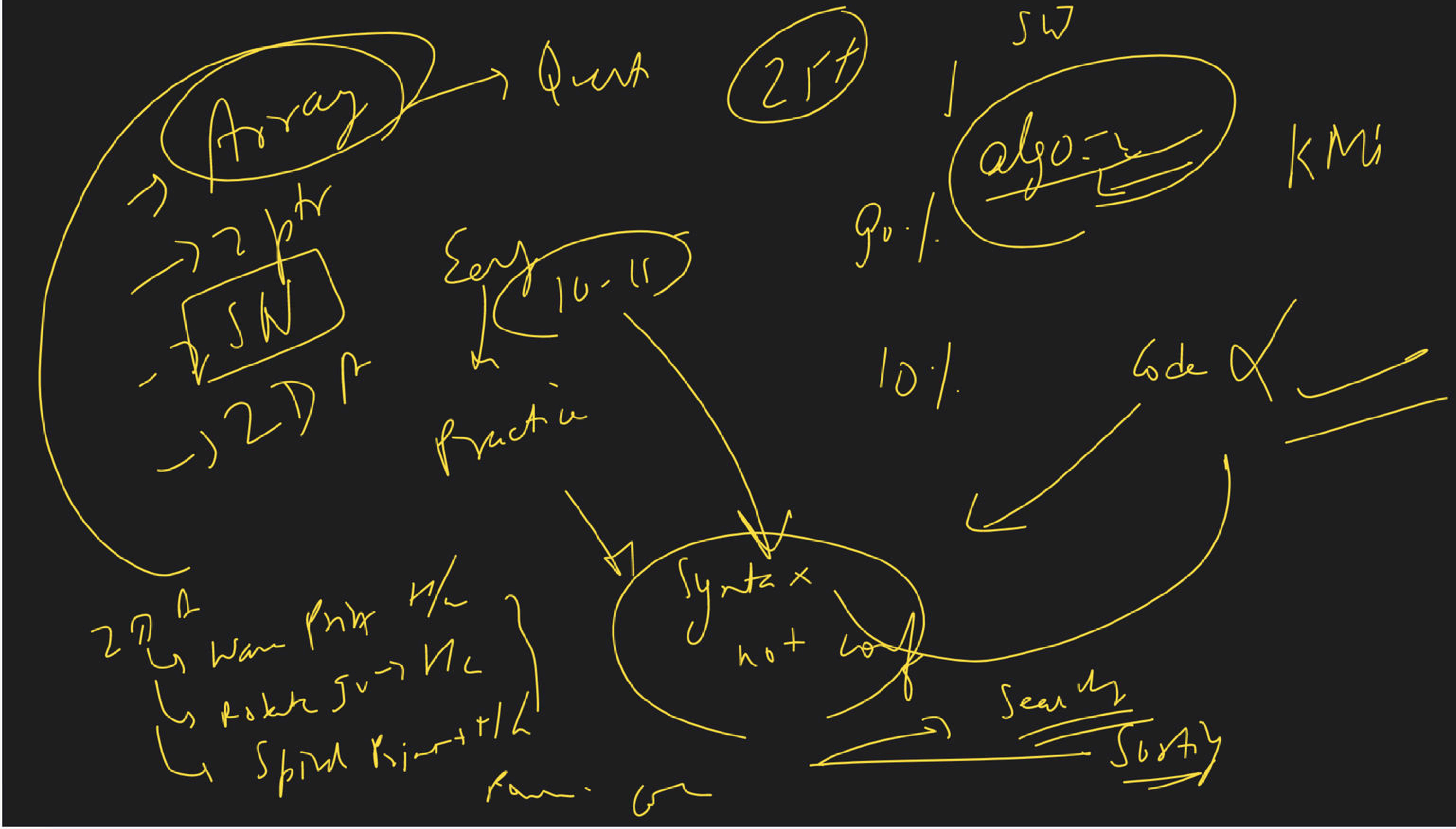
Code

Code

n = 3

algo:-

n = 4


Array → Quest

21st

SW

algo:-

90%

10%

KMi

Code X

→ 2 ptr

→ SW

→ 2D Ar

Easy 10-15

Practice

Syntax not conf

2D Ar
↳ Wave Print H/W
↳ Rotate 90 → H/W
↳ Spiral Print H/W

Famil Con

→ Searchy

Sorting

→ Saturd → 2D

→ explore

→ play

GitHub

→ Sunday → Doubts → Toofani

Web Dev → FREE

Com

13 May

DBMS

15th May

FREE

placement