DAY 30

3 October, 2021

DSA

2-D ARRAY

Agenda for Today :-

1. → 2d Array Introduction → CW
2. → Matrix multiplication → HW
3. → Wave Display → CW
4. → Spiral Display → HW
5. → Exit point of Matrix → HW
6. → Rotate by 90. → HW

2-D array के बारे में 2 तरीको से बात करेंगे

1. Usage of 2-D array.
2. Memory used by 2-D array.

1. Usage of 2-D array.

Matrix की information अगर हमें चाहिए तो हमे 2-Darray बनाना पड़ता है।

```
int [][] arr;
```

→ ये है 2-d array का नाम

Declare हुआ है 2-Darray.

```
int [][] varr = new int [3][4];
```

Define भी होगया है अब

फिर no. of columns लिखे जाते हैं।

पहले no. of rows लिखी जाती हैं

Page No. | Date

```
int[][] arr = new int [3][4];
```

	0	1	2	3
0	[0,0]	[0,1]	[0,2]	[0,3]
1	[1,0]	[1,1]	[1,2]	[1,3]
2	[2,0]	[2,1]	[2,2]	[2,3]

इस line से एक ऐसा array बनगया है।

ये spot है [2][0] मतलब 2nd row का 0th column.

अब इस 2-D array में values डालते हैं फिर loop लगाके print करेंगे।

values डालने पर ये बनगया

```
arr[0][0] = 11;
arr[0][1] = 12
arr[0][2] = 13
arr[0][3] = 14

arr[1][0] = 21
arr[1][1] = 22
arr[1][2] = 23
arr[1][3] = 24

arr[2][0] = 31
arr[2][1] = 32
arr[2][2] = 33
arr[2][3] = 34
```

11	12	13	14
21	22	23	24
31	32	33	34

i	j	input at arr[i][j]
0	0	arr[0][0]
	1	arr[0][1]
	2	arr[0][2]
	3	arr[0][3]
1	0	arr[1][0]
	1	arr[1][1]
	2	arr[1][2]
	3	arr[1][3]
2	0	arr[2][0]
	1	arr[2][1]
	2	arr[2][2]
	3	arr[2][3]

अब इस array को print कैसे करायेंगे ?

No. of rows मिलती है → arr.length से
No. of column मिलते है → arr[0].length से

for no of rows → `for(int i=0; i< arr.length ; i++)`
`for(int j=0; j< arr[0].length ; j++)`
`{ System.out.println( arr[i][j] +"\t");`
`}`

for no. of columns

arr[i][j] : i → row no., j → column no.

```
→ public class Main
→{ public static void main (String args[])
→ { int[][] arr = new int[3][4];
    arr[0][0] = 11;
    arr[0][1] = 12;
    arr[0][2] = 13;
    arr[0][3] = 14;

    arr[1][0] = 21;
    arr[1][1] = 22;
    arr[1][2] = 23;
    arr[1][3] = 24;

    arr[2][0] = 31;
    arr[2][1] = 32;
    arr[2][2] = 33;
    arr[2][3] = 34;
    for(int i=0; i < arr.length; i++)
    {   for(int j=0; j< arr[0].length ; j++)
        { System.out.print(arr[i][j] + "\t");
        }
        System.out.println();
    }
}
}
```

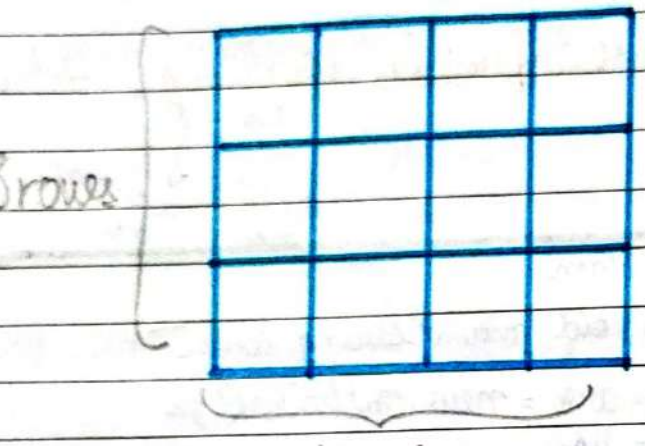
→ \t का tab का gap बनाता है print करने के बाद

Page No.
Date

तो अभी तक हमने देखा Usage of 2-D array. अब समझते हैं Memory Map.

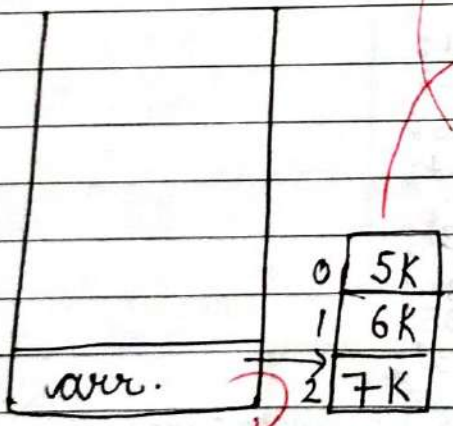
2. Memory Map

int [][] arr = new int [3][4];



हम सोचते हैं की memory में ऐसे Saved है. Lekin ये एक illusion है।

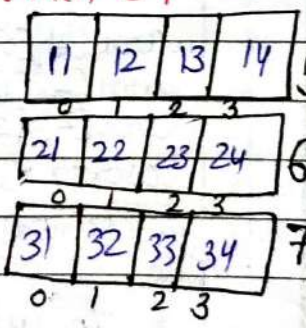
Reality में कैसे store हुआ?



4K - address पर एक arr नाम का variable बना जिसका Size है 3 जितनी rows है, उतना ही Size होगा 2-d array का। तो जब हम arr .length करते है तो हमे no. of rows मिलती है।

तो जब हमे column निकालने हो तब हम array[0].length या array[1].length या array[2].length निकले, So equal होंगे और columns बतायेंगे।

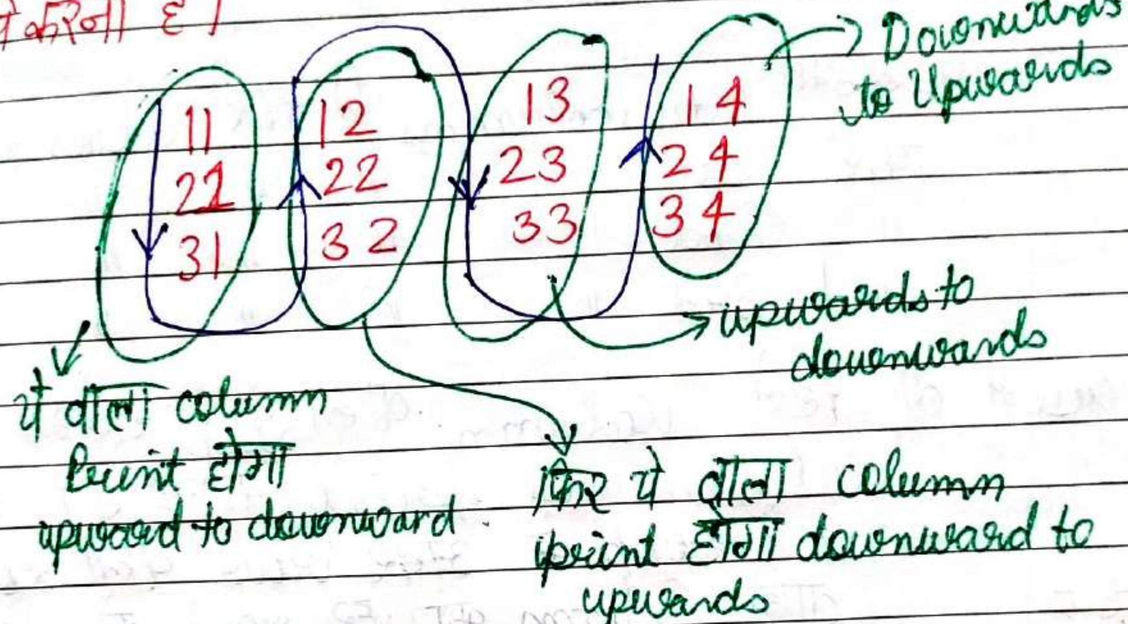
ये वाला array असलियत में सिर्फ Pointers रखता है actual arrays का address रखता है।

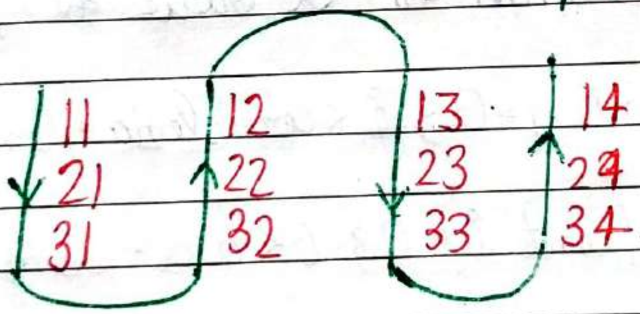


ये arrays होंगे column के size जितने और यहीं पर actual values store होंगी।

Ques. 1) Wave Display → The State of Wakanda - 1

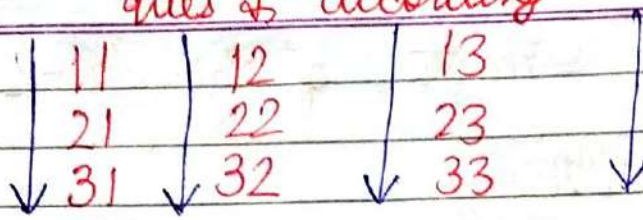
1) Take as Input 2d Array.
2) फिर उस Input array को print कुछ अलग तरीके से करना है।





Output → 11
21
31
32
22
12
13
23
33
34
24
14

पहले सारी rows को upwards to downward print कराते हैं। फिर देखेंगे ques के according



तो हम पहले 1st col print करना है
फिर 2nd " " " "

पहले हम 0th columns की सारी rows print करेंगे
फिर 1st " " " " " "
" 2nd " " " " " "
" 3rd " " " " " "

इस case में तो पहले column वाला loop चलेगा। क्योंकि हम column wise print कर रहे हैं। और column वाले loop के अंदर row वाला loop लगेगा ताकि column की हर row को print करा पाए

Output
11
21
31
12
22
32
13
23
33
14
24
34

```
for(int j=0; j<arr[0].length; j++)
{
  for(int i=0; i<arr.length; i++)
  { System.out.println(arr[i][j]);
  }
}
```

→ DRY RUN करके जरूर देखना है खुद से तब ही code करेंगे।

अब हम अपने Code को देखते हैं.

11	12	13	14
22	22	23	24
31	32	33	34

Even columns हैं, तो जाएंगे नीचे की तरफ
Odd columns हैं तो जाएंगे ऊपर की तरफ

last row से first row तक

arrayLength − 1 → 0th row

3 − 1 → 2nd row से 0th row तक जाएगा तो loop में decrement होता जाएगा.

```
for(int j=0; j<arr[0].length; j++)
{
  if (j%2==0)
  { for(int i=0; i<arr.length; i++)
    { System.out.println(arr[i][j]);
    }
  }
  else{
    for(int i=arr.length-1; i>=0; i--)
    {
    } System.out.println(arr[i][j];
}
```

```
→ import java.util.*;
→ public class Main {
    public static void main (String[]args)
  { Scanner scn= new Scanner( System.in);
    int r= scn.nextInt();
    int c= scn.nextInt();
int[][] arr= new int[r][c];
for(int i=0; i<arr.length; i++)
 { for int j=0; j< arr.length; j++)
    { arr[i][j] =scn.nextInt();
    }
 }
for (int j=0; j< arr[0].length; j++)
{   if (j%2==0)
    { for(int i=0; i<arr.length; i++)
      { System.out.println(arr[i][j]);
      }
    }else.
     { for(int i=arr.length; i>=0; i--)
       {; System.out.println(arr[i][j]);
     }
}}
```

Input ↘

11	12	13	14
21	22	23	24
31	32	33	34

Output ↘

11
21
31
32
22
12
13
23
33
34
24
14

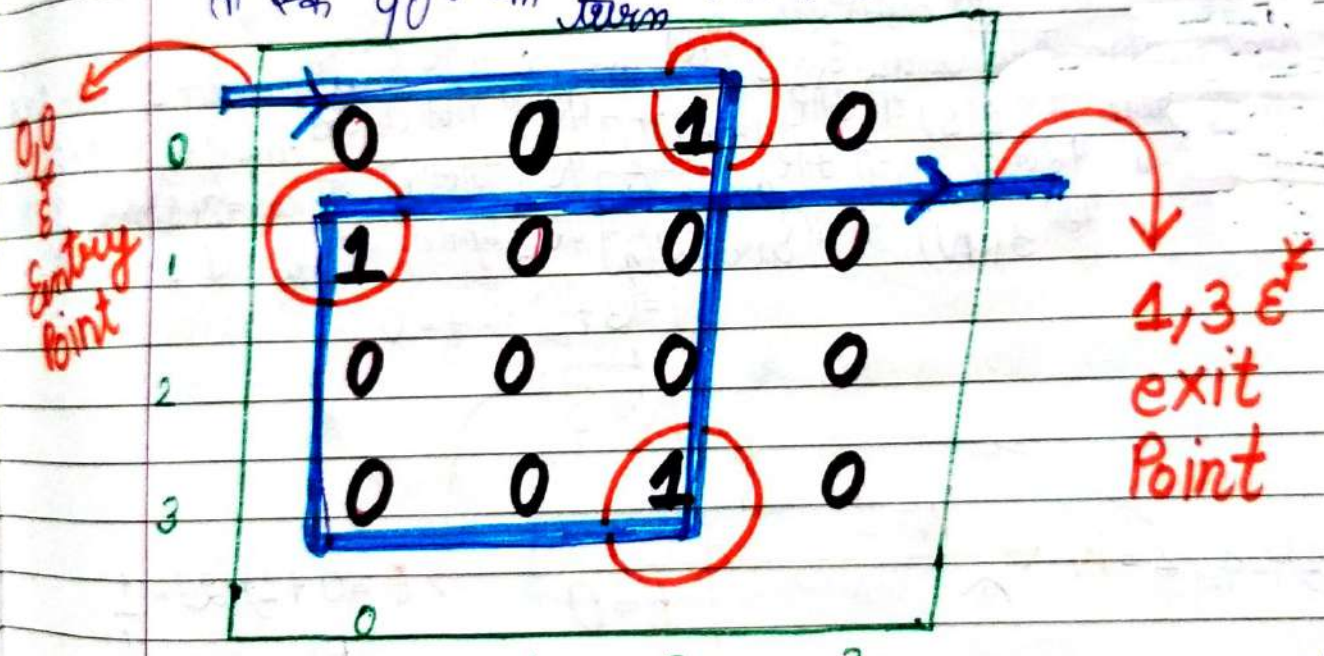
* Java में by default dynamic array ही बनता है।

Ques 2. Exit Point of a Matrix

1. Input लेना है एक 2-D array जिसमे बस 0 और 1 value होगी

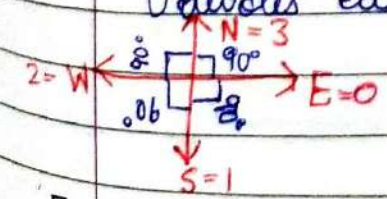
0	0	1	0
1	0	0	0
0	0	0	0
1	0	1	0

एक बंदा इस array को traverse करता है। सीधा-सीधा चलते रहना है। जैसे ही 1 आएगा तो एक 90° का turn लेलेना है



तो direction नाम का एक variable बनाते है जिसमे east direction है मतलब 0 है value तो हम शुरू करेंगे east direction से।

0 denotes east dir.



हमे ऐसे 1 आते ही 90° rotate करने के लिए बस इस line को ध्यान से समझना है।

```
dir = (dir + arr[i][j]) %4
```

E = 0
S = 1
W = 2
N = 3

0 → E → East → (0+1) % 4 = 1
1 → S → South → (1+1) % 4 = 2
2 → W → West → (2+1) % 4 = 3
3 → N → North → (3+1) % 4 = 0

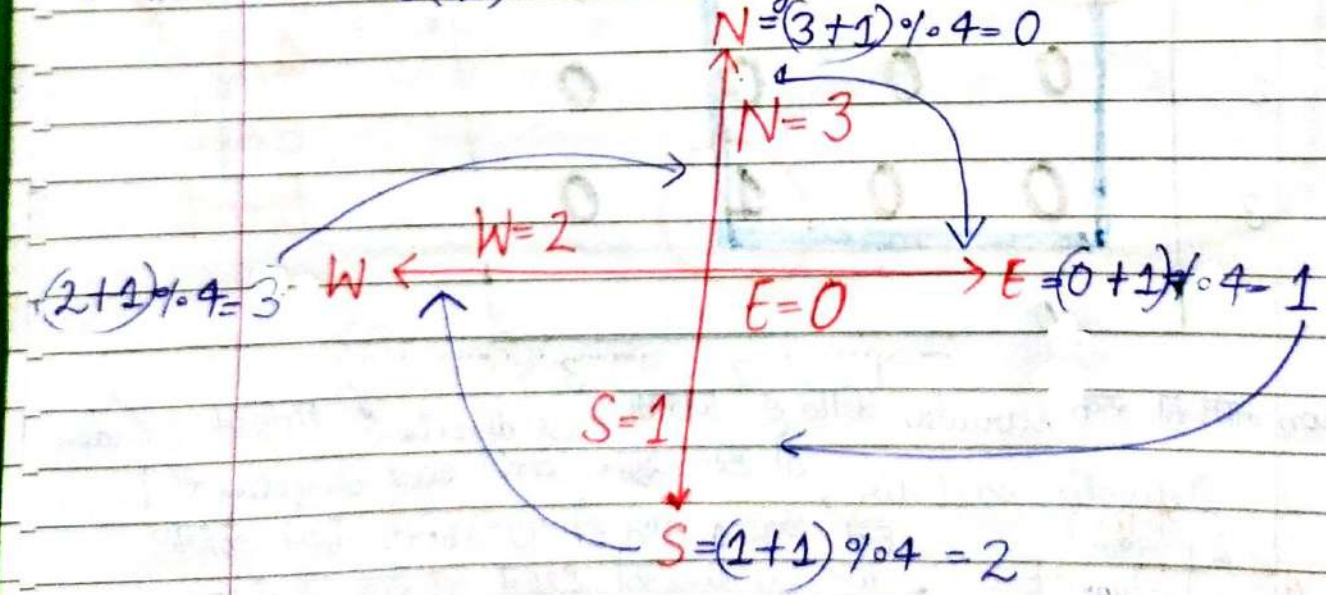
0 = East ← & continues

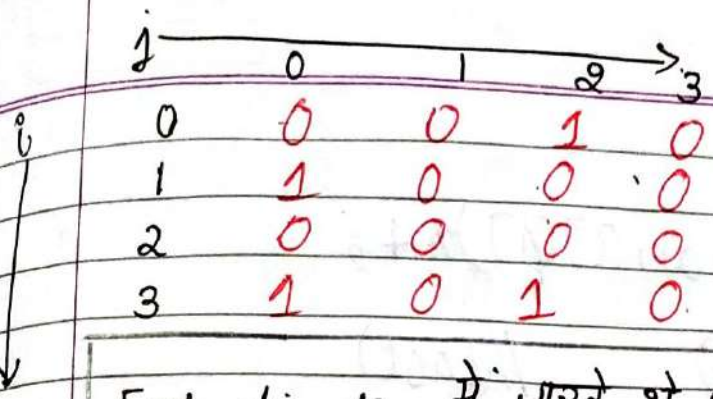
dir = (dir + arr[i][j]) % 4

हमें जब जब arr[i][j] पर value 0 मिलेगी तब तब तो सीधा चलेगा और dir में कुछ add भी नहीं होगा तो अगर पहले dir की value 0 थी तो अब भी 0 ही रहेगी।

जैसी ही arr[i][j] पर हमें value 1 मिलेगी तो दो dir में add होना शुरू होगी

अगर dir की value पहले 0 थी और 1 मिला arr[i][j] पर तो अब dir = (0+1) % 4 = 1 = S
" " " value पहले 1(S) थी और arr[i][j] पर 1 मिला तो अब dir = (1+1) % 4 = 2 = W
" " " " पहले 2(W) थी और arr[i][j] पर 2 मिला तो अब dir = (2+1) % 4 = 3 = N
" " " " पहले 3(N) थी और arr[i][j] पर 3 मिला तो अब dir = (3+1) % 4 = 0 = E





East direction में जारहे थे अब 90° turn लेके South में जाना है तो i बढ़ जाएगा.

South → i++

South में जारहे थे अब 90° turn लेके West में जाना है तो j-- होरहा है मतलब j की value कम होरही है।

W. ←

West → j--

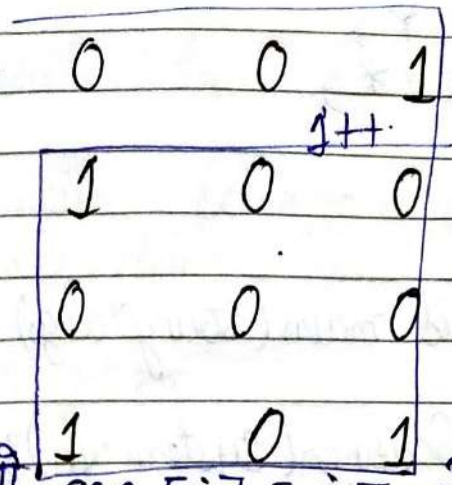
West में जारहे थे अब 90° turn लेके North में जाना है तो i-- होरहा है मतलब i की value कम होती जारही है।

North → i--

North में जारहे थे 90° turn लेके East में जाना है तो j++ होरहा है मतलब j की value increase होरही है.

East → j++

```
while (true)
{
  dir = (dir + arr[i][j]) % 4;
  if (dir == 0)          //(East)
  {
    j++;
  }
  else if (dir == 1)     //(South)
  {
    i++;
  }
  else if (dir == 2)     //west.
  {
    j--;
  }
  else                   // North
  {
    i--;
  }
}
```



वो arr[i][j] की value जिसमे last row या column 1 subtract या add करते ही या तो i की value < 0 हो जाए या i की value > arr.length. हो जाए या फिर j की value < 0 हो जाए या j की value > arr[0].length हो जाए वो होगा exit point

j की value होगी 4 > arr[0].length. तो ये होगा exit point और वही output देना है। या वो

```
int i=0;
int j=0;
int dir=0;
while(true)
{dir=(dir+arr[i][j])%4;
 if(dir==0)
 {j++;
  if(j==arr[0].length)
    {j--;
     break;
    }
 else if(dir==1)
 {i++;
  if(i==arr.length)
  {i--;
  break;
  }
 }else if(dir==2)
 {j--;
   if(j==-1)
   {j++;
    break;
   }
 }
 else{i--;
   if(i==-1){i++;
   break;
```

Exit Point Of A Matrix . java

```
import java.io.*;
import java.util.*;

public class Main
{
public class void main(String[] args)
{
Scanner scan= new Scanner(System.in);
int r = scan.nextInt();
int c= scan.nextInt();

int[][] arr = new int[r][c];
for(int i=0; i<arr.length; i++)
{ for(int j=0; j<arr[0].length; j++)
{
arr[i][j] = scn.nextInt();
}
}
int i=0;
int j=0;
int dir = 0;

// 0 is East
// 1 is South
// 2 is West
// 3 is north.
```

→ while (true)
{
dir = (dir + arr[i][j]) % 4;
if (dir == 0) // East
{
j++;
if (j == arr[0].length)
{
j--;
break;
}
}
else if (dir == 1) // South
{
i++;
if (i == arr.length)
{
i--;
break;
}
}
else if (dir == 2) // West
{
j--;
if (j == -1)
{
j++;
break;
}
}
}

वापस आए 1 बार और break करदो क्योंकि हमे तो exit point print करना है।

```
else                                    // dir==3 north
  {  i--;
     if(i==-1)
     {  i++;
        break;
     }
  }
}
System.out.println(i);
System.out.println(j);
  }
}
```

2d array > is better than 2-d list

Head First Java.
Head First C.
Head First OOAD
Head First Design Pattern

} ये 4 Books हर बच्चे को पढ़नी चाहिए

HW

Page No.
Date

2D ARRAY

1. Article — 2D Arrays Demo.
2. Code — 2D Arrays Demo. — Done in CW
3. Article — Matrix Multiplication
4. Code — Matrix Multiplication
5. Article — The state of Wakanda 1
6. Code — The state of Wakanda 1 — Done in CW
7. Article — Spiral Display
8. Code — Spiral Display
9. Article — Exit point of Matrix.
10. Code — Exit point of Matrix. — Done.
11. Article — Rotate by 90 Degrees.
12. Code — Rotate by 90 Degrees

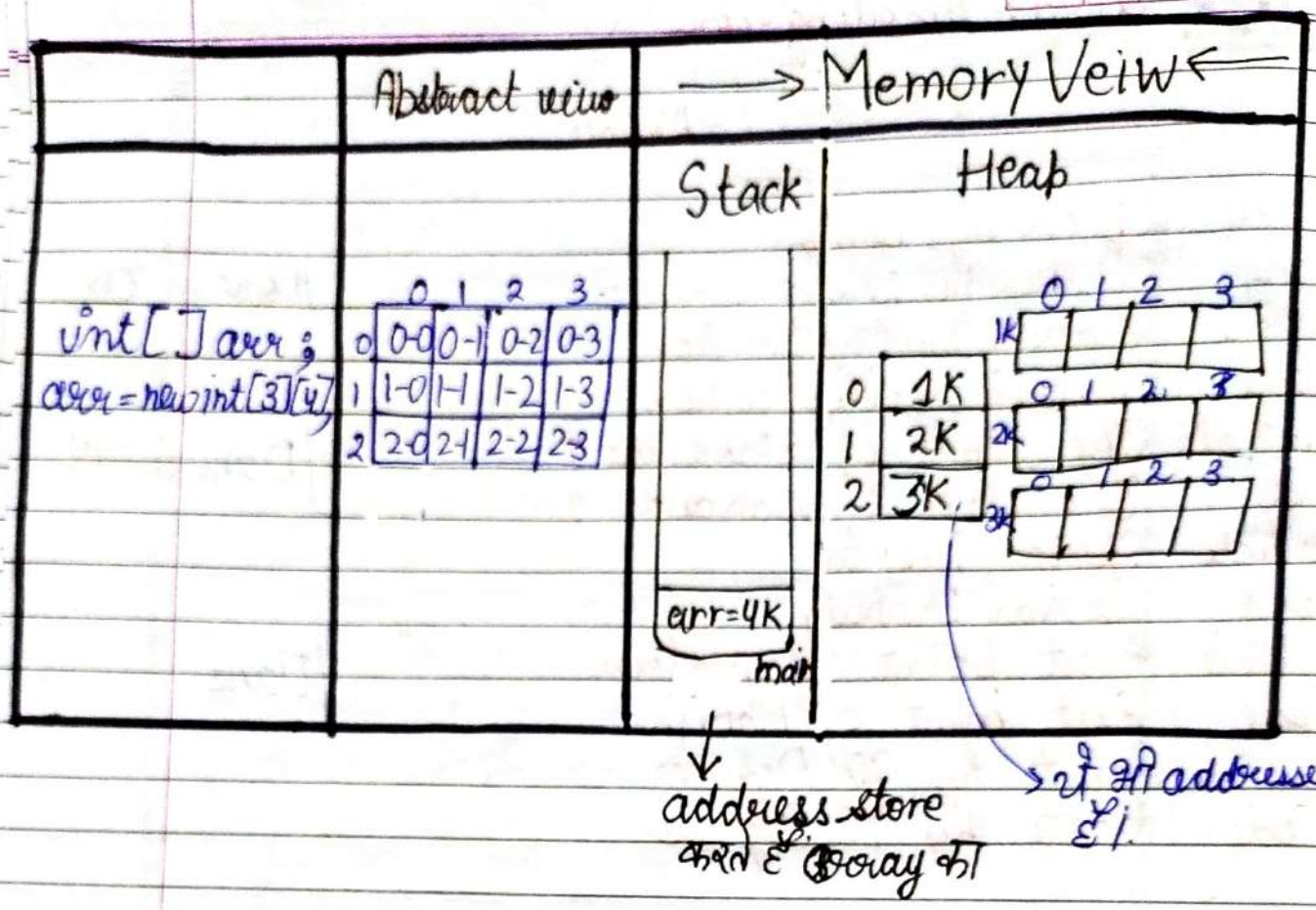
1) 2-D Array Article

1-D Array → 1-D array are the list that represent many elements and are one-dimensional.

Similarly, 2D arrays are 2 dimensional arrays which are used to represent matrices.

* We are going to meditate on 2 aspects of 2-D array :- Abstract View & Memory View
  - Abstract View → Usage
  - Memory View → Memory used by 2d memory

for most of our discussion, the abstract view is sufficient However, Memory view allows to understand 2D array with deep understanding.



1. * If we declare arr[][]
int arr[][];
then a stack is created in the memory which has a variable arr which stores a null value (i.e it doesn't point to anything).

2. * When we define this array as arr = new int[3][4] then the first dimension = 3 denotes the rows & the second dimension = 4 denotes the column.

Abstract View

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

Here we see a grid/matrix with rows and a column, each starting from index 0.

↳ Inside each cell, the position of that cell is written in the format "row - column". For eg → 2-1 denotes the cell at 2nd row & 1st column.

Memory View

If we actually consider var = new int[3][4], then we realize that, it denotes that we have 3 arrays (1-D) of size 4 each. As we can see in the Memory View, that the variable var is stored at address 4K.
This 4K address is storing an array of size 3 which consists of 3 addresses and at each of these 3 addresses, an array of size 4 is present.

Do we even realize when we define the given array, actually 4 arrays are formed.

How? → 3 are the arrays with 4 Sizes each which will contain intigers and the 4th array is of Size 3 which contains the addresses (1K, 2K, 3K) as shown in the Memory view

Hence, arr is an array of arrays.

3. * The Programming Implementation in 2-D array is done using 2 loops to iterate over the arrays and lly storing elements in the cell. Similar to the input process, the output process is also carried out using 2 loops.

4. * Time Complexity :- $O(n^2)$
This time complexity is quadratic due to the use of nested for loop.

* Space Complexity :-
As a 2D array is used to store input values, therefore space complexity is Quadratic

MATRIX MULTIPLICATION

Page No.
Date

Ques-3) This question takes 2 matrices as the Input & return the product of those matrices on the multiplication.

Solution

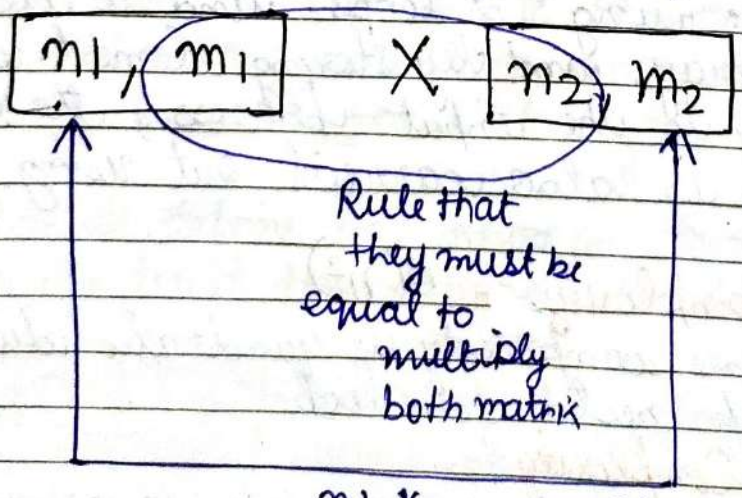
* Here let n1 = no. of rows of Matrix 1.
  let m1 = no of column of Matrix 1.

  let n2 = no. of rows of Matrix 2.
  let m2 = no. of column of Matrix 2.

We must understand that, not all the matrices can be multiplied with each other.
They can be multiplied iff
no. of column of Matrix 1 = no. of rows of Matrix 2

i.e **m1 == n2**

Only if (m1 == n2) then a product matrix can be formed of size (n1 * m2).



n1 * m2 = Size of the Product Matrix

Eg → Lets assume, we are given a matrix A[2,3] and another matrix B[3,4]

Page No. 6
Date

Can Matrix A be multiplied to Matrix B?

Yes, Both of these Matrix can be multiplied to each other & this is because no. of column of Matrix A is equal to no. of rows of Matrix B.

A new matrix will be formed of size $(m_1 * m_2)$ i.e P[2][4].

↓

P is the name of product matrix

	0	1	2
0	a_{00}	a_{01}	a_{02}
1	a_{10}	a_{11}	a_{12}

A

×

	0	1	2	3
0	b_{00}	b_{01}	b_{02}	b_{03}
1	b_{10}	b_{11}	b_{12}	b_{13}
2	b_{20}	b_{21}	b_{22}	b_{23}

B.

	0	1	2	3
0	P_{00}	P_{01}	P_{02}	P_{03}
1	P_{10}	P_{11}	P_{12}	P_{13}

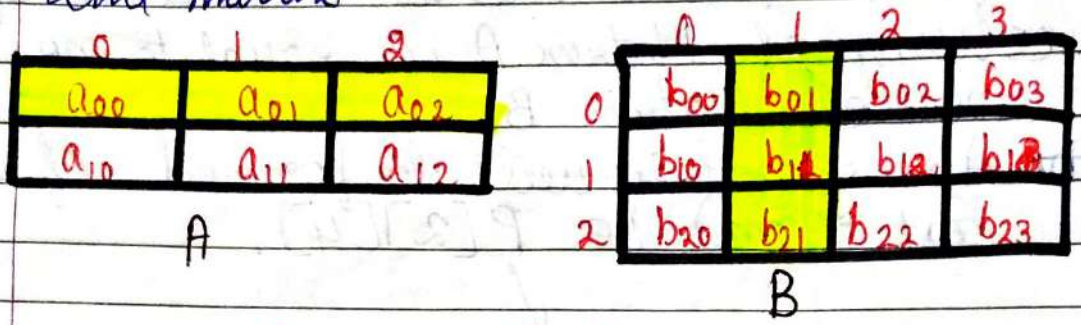
P.

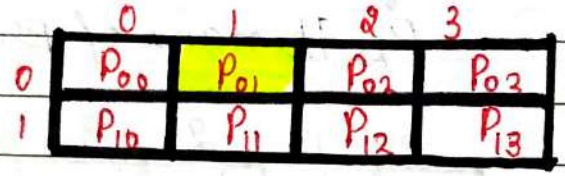
Now, we need to understand how to calculate the elements in the product matrix P.

$$P_{ij} = a_{i0} b_{0j} + a_{i1} b_{1j} + \cdots + a_{i(n-1)} b_{(n-1)j}$$

$$= \sum_{k=0}^{n-1} a_{ik} b_{kj}$$

The value at P01 is the sum of the product of elements of the 0th row of the 1st matrix with the elements at the 1st column of the 2nd matrix.





$$P_{01} = a_{00} * b_{01} + a_{01} * b_{11} + a_{02} * b_{21}$$

P → 2 QTR column K times inc in arr1

a_{00} → row K time arr2.

* **Time Complexity** $\Rightarrow O(n^3)$

The time complexity will be cubic i.e n^3 because 3 nested for loops are used.

* **Space Complexity** $\Rightarrow O(n^2)$

As 2D arrays are used to store numbers, therefore space complexity is quadratic.

Now, lets code it:-

```
→ import java.util.*;
→ public class Main{
→ public static void main(String[]args).
{ Scanner scan = new Scanner (System.in);
int row1 = scan.nextInt();
int column1 = scan.nextInt();
int arr1[][] = new int[row1][column1];
for(int i=0; i<row1; i++)
{for(int j=0; j<column1; j++)
{ arr1[i][j] = scan.nextInt();
}
}
int row2 = scan.nextInt();
int column2 = scan.nextInt();
int arr2[][] = new int[row2][column2];
for(int i=0; i<row2; i++)
{for(int j=0; j<column2; j++)
{ arr[i][j] = scan.nextInt();
}
}
if(column1==row2)
{int product[][] = new int[row1][column2]
for(int i=0; i<row1; i++)
{for(int j=0; j<column2; j++)
{ int sum=0;
for(int k=0; k<column1; k++)
{ sum = sum + (arr1[i][k] * arr2[k][j]);
}
product[i][j] = sum;
}}
for(int i=0; i<row1; i++)
{for(int j=0; j<column2; j++)
{ System.out.print(product[i][j] + " ");
}
System.out.println();
}}
else { System.out.println("Invalid Input"); }
}
```

Page No.
Date

Arr1 Input

Array2 Input.

Check if Arr1 & Arr2 can be multiplied

Multiplication

K is inc with each loop.

column no of arr1 is inc with each loop so K is written here

row no. of arr2 is increasing with each loop. so k is here

Print output

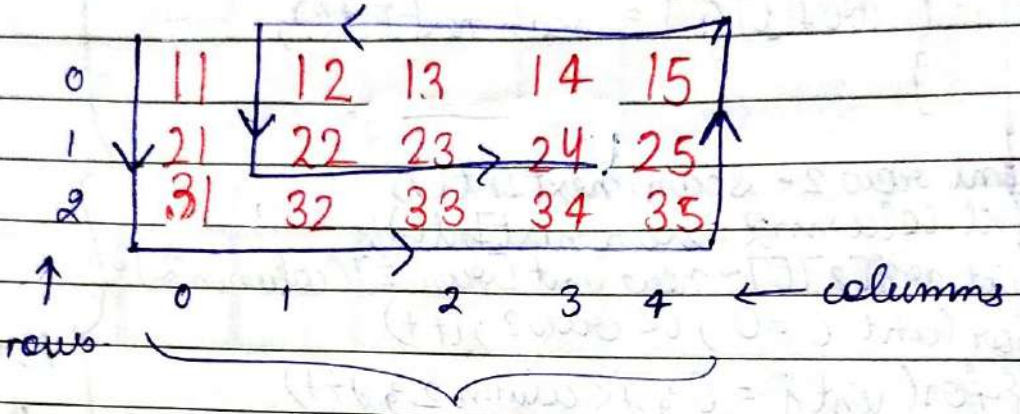
Invalid Input

(Ques 4) SPIRAL DISPLAY

Page No. | Date

* Remember that algorithm for success in coding is
while (! success)
tryagain ();

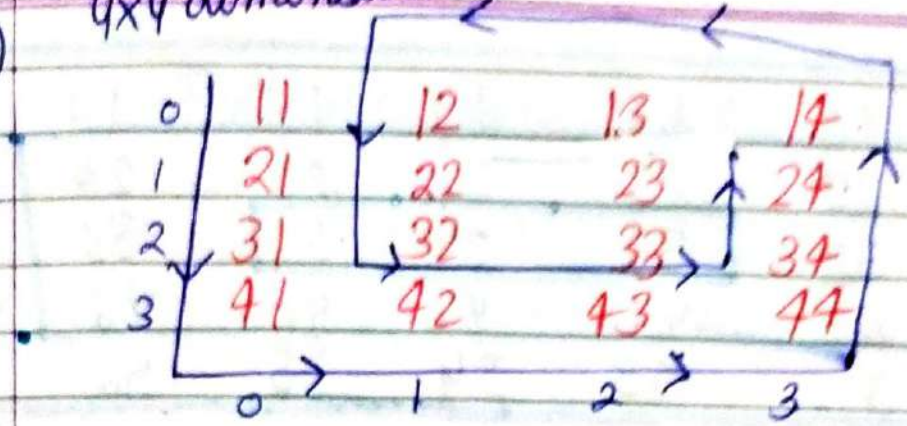
eg-1)



Output

11
21
31
32
33
34
35
25
15
14
13
12
22
23
24

eg-2) 4x4 dimension 2-darray.

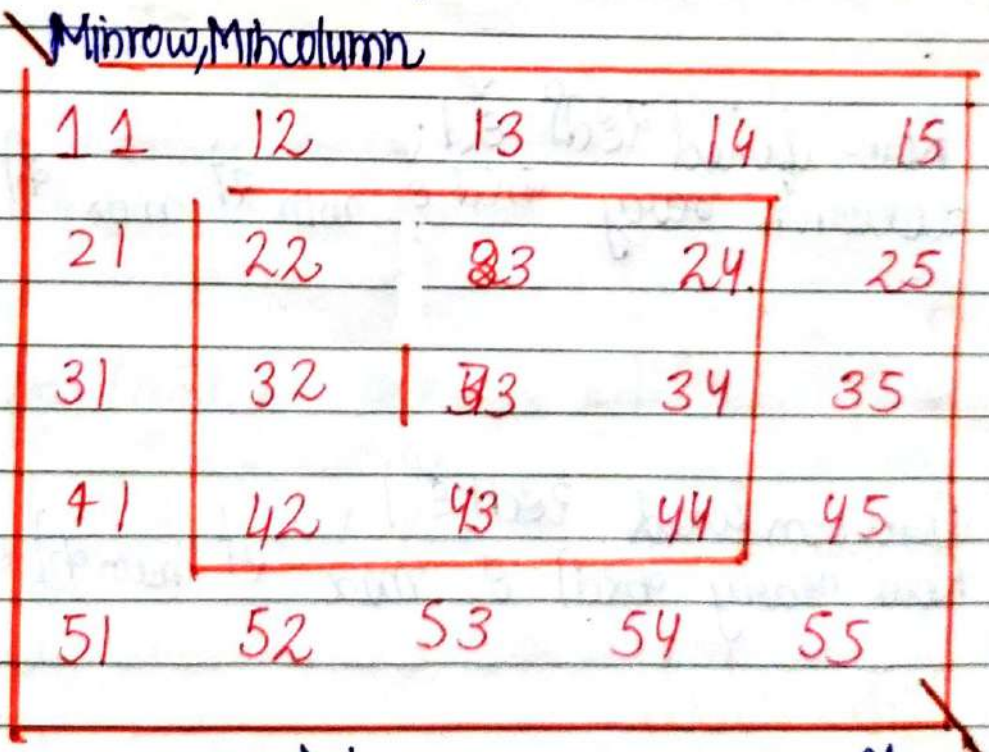


Output

11 21 31 41 42 43 44 34 24 14 13
12 22 32 33 23

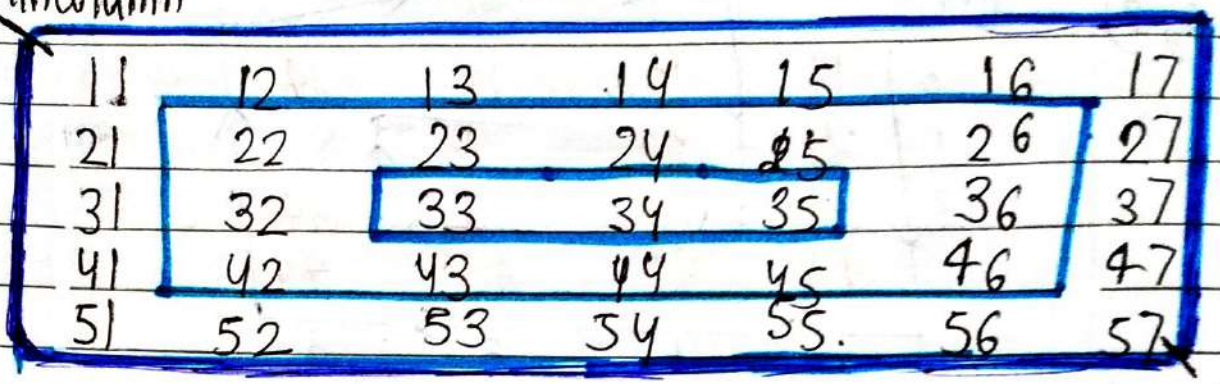
Spiral way travel करने से अच्छा होगा कि हम pattern को box by box wise traverse करें।

To identify 1 box, we need min row, & min col.
& max row, max col.



1) Box के अंदर Box Print करेंगे
2) एक Box Print करने के 4 wall Print करनी होगी
3) 4 wall Print करने के लिए 4 loops लगाने पड़ेंगे चारों walls के
4) सबसे पहले left wall print करेंगे फिर bottom वाली फिर Right और फिर Top wall.

Minrow, MinColumn



Max Row, Max Column

1.* Left Wall

1) Column fixed रहता है।
2) Row vary करती है Min से Max की तरफ

2. Bottom Wall

1) Row fixed रहती है। i.e.
2) Column vary करते है min से max की तरफ

3) Right Wall

1) Column fixed रहता है।
2) Row vary करती है max से min की तरफ

4) Top Wall

1) Row fixed रहती है।
2) Column vary करते है Max से Min की तरफ

पहले 1 Side को print करने की try करते हैं।
और अभी हम Corners को 2 बार print होने देंगे।
बस Box की values print करा के देखना है।

```
int minr = 0;
int minc = 0;
int maxr = arr.length - 1;
int mn   = arr[0].length - 1;
while(true){
// left wall.
    for (int i = minr, j = minc; i<=maxr; i++)
    { System.out.println(arr[i][j];
    }
// bottom wall
    for (int i = maxr, j = minc; j<=maxc; j++)
    { System.out.println(arr[i][j];
    }
// right wall.
    for (int i=maxr, j = maxc; i>=minr; i--)
    { System.out.println(arr[i][j];
    }
// top wall.
    for(int i=minc, j=maxc; j>=minc; j--)
    { System.out.println(arr[i][j];
    }
// Box तो Print हो रहा है लेकिन Corners repeat होते हैं।
```

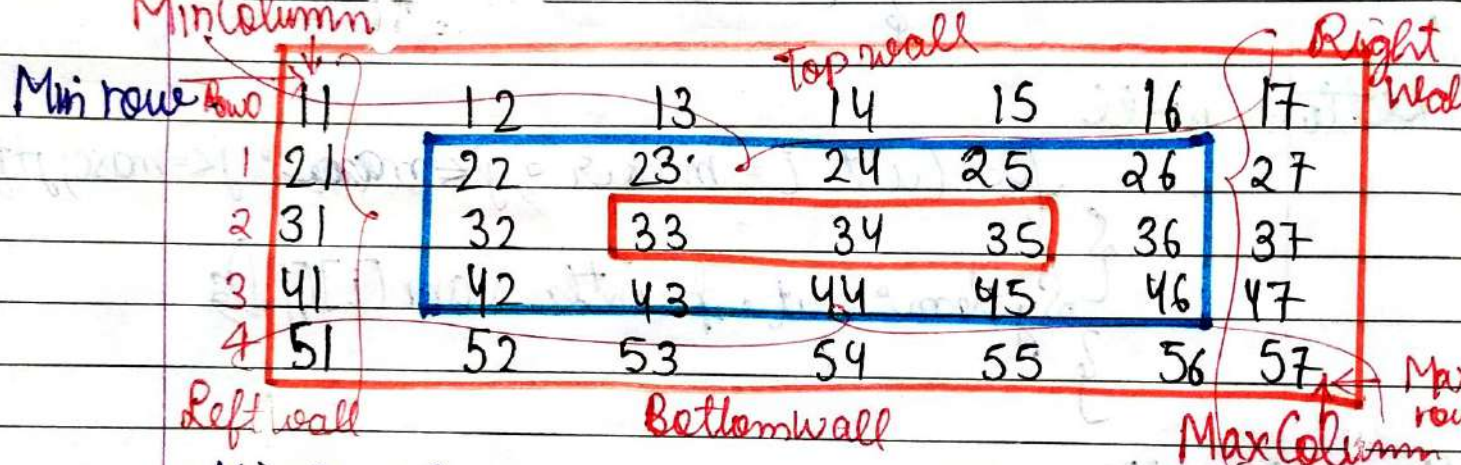
अब हमें repeated corners के लिए कुछ करना पड़ेगा।

left wall → को indicate करता है → min column

bottom wall → को indicate करता है → max row

right wall → को indicate करता है → max column

top wall → को indicate करता है → min row



जैसे ही हमने left wall print करदी, तभी हमें min column को बढ़ा देना है।

1. जैसे ही Left Wall print की min column ++ करदो Left Wall के बाद Bottom wall print होता है। उसे Bottom wall में columns change होते हैं: min column से max column तक। अब अगर हम Bottom wall print करेंगे तो min column start होगा 2nd column से और 2 बार 51 Print नहीं होगा।

```
while (true)
{
// left wall.
for (int i = minr, j = minc ; i <= maxr ; i++)
{
System.out.println (arr[i][j]);
}
minc++;
```

2. जैसे ही Bottom wall print हो जाए, उस max row को -- (decrease) कर दीजिए। अब जब max row decrease हो गया है तो वो Row 4 को नहीं 3 को point करेगा तो 57 से अब Bottom wall के बाद तो Right wall print करो है और right wall को indicate करता है max column और max column (Right wall print होगी max row से min row तक

```
for (int i = maxr, j = minc ; j <= maxc ; j++)
{
System.out.println (arr[i][j]);
}
maxr--;
```

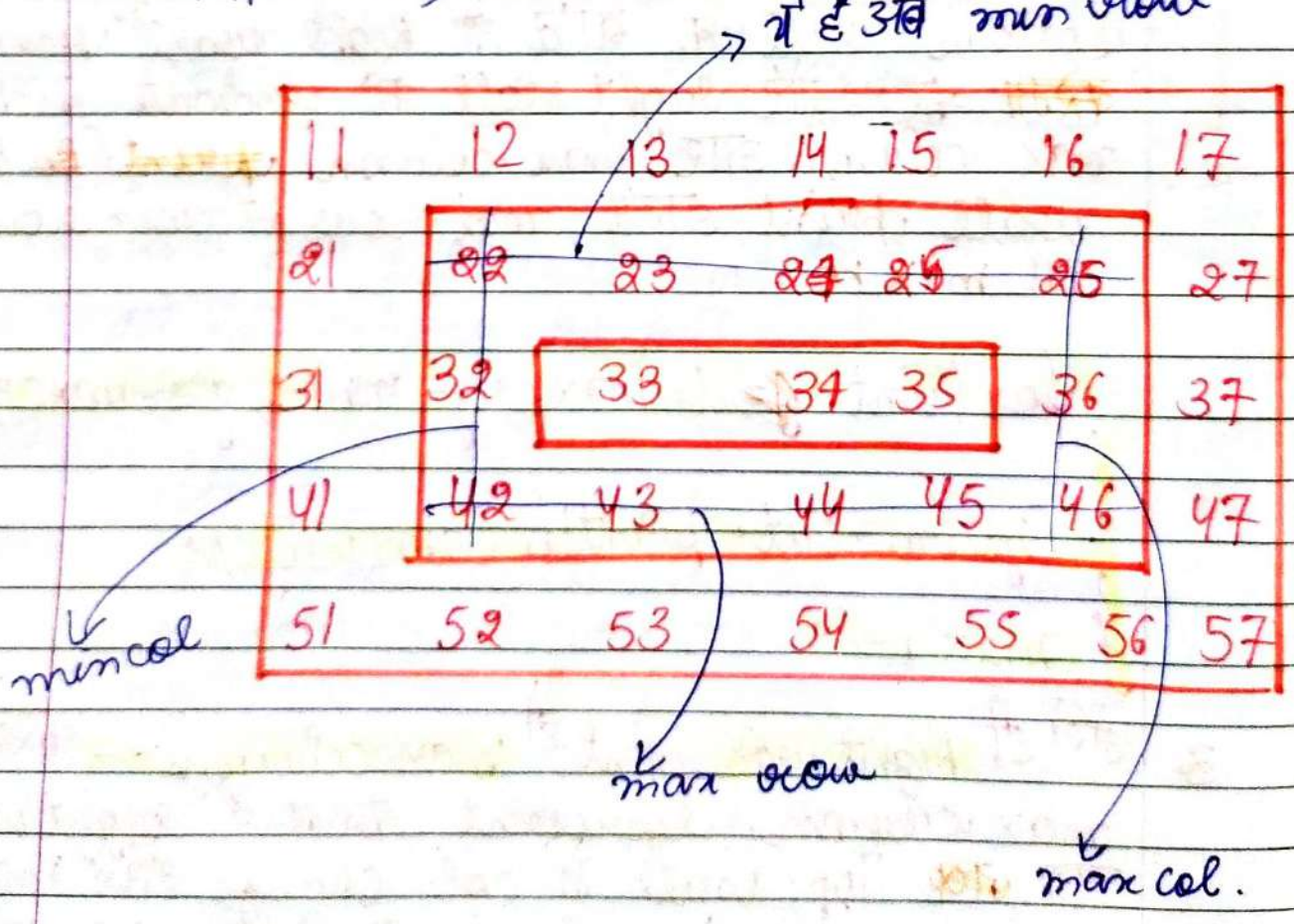
3. जैसे ही Right wall print हो max columns -- करदो bcz max column represent करता है right wall को। अब Top wall में col. change होगा max column से min column तक। अब अगर हम Top wall print करेंगे तो max column के लिए 1 या 2 बार Print नहीं होगा।

// right wall

```
for(int i = maxr , j = maxc ; i > minr ; i--)
{
  System.out.println(arr[i][j]);
}
  maxc--;
```

↳ जैसे ही Top wall print होगया, min row ++ करदो ताकि जो अगला Box है वो 1st (अगले) row से शुरू हो।

```
minr++;
```



कबतक Loop चलता रहेगा? जब तक no. of digit count in array, no. of digit printed नहीं हो जाती।

Page No.
Date

last box में जरूरी नहीं है कि चारो walls हो हमारे पास तो एक count variable का check करते रहेंगे (no. of digit count)

```
import java.util.*;
import java.io.*;
public class Main{
public static void main(String[]args)
{ Scanner scan=new Scanner(System.in);
int n= scn.nextInt();
int m= scn.nextInt();
int arr[][]=new Int[n][m];
for(int i=0; i< arr.length; i++)
{for(int j=0; j<arr[0].length; j++)
  { arr[i][j] = scn.nextInt();
  }
}
int minrow=0;
int mincol=0;
int maxrow= arr.length-1;
int maxcol= arr[0].length-1;
int tne=n*m  // total number of elements
int count=0;
while(count< tne)
{ // left wall.
  if(count< tne)
  {for(int i=minrow; i<=maxrow; i++)
    {System.out.println(arr[i][mincol]);
    count++;
    }
  }mincol++;
```

```
// bottom wall
 if (count < itne)
{ for(int i = mincol ; i<=maxcol ; i++)
{
{System.out.println(arr[maxrow][i]);
count++;
}
}.
maxrow--;
//right wall.
 if (count< itne)
{ for(int i = maxrow ; i>=minrow ; i--)
  { System.out.println(arr[i][maxcol]);
    count++;
  }
}
  maxcol--;
// top wall;
  if (count < itne)
  {
  { for(int i = maxcol; i>= mincol; i--)
      { System.out.println( arr[minrow][i]);
        count++;
      }
    }

  minrow++;
  }
}
```

Page No.
Date

Time Complexity :-

$O(n^2)$

As there is nested for loop & the outer for loop runs n times. There are 2 inner for loops. Either one will run in each iteration.

Making the time complexity : $O(n) * O(m)$

$= O(n * m)$.

Space Complexity

$O(1)$.

Since we are not using any auxiliary space & hence the space complexity is $O(1)$.

Ques. 5).

Rotate by 90° Degrees

Page No.
Date

→ We are given a square matrix and we have to rotate by 90° degrees.

	0	1	2	3
0	11	12	13	14
1	21	22	23	24
2	31	32	33	34
3	41	42	43	44

Clockwise 90° Rotation

After 90° rotation

	0	1	2	3
0	41	31	21	11
1	42	32	22	12
2	43	33	23	13
3	44	34	24	14

: Rotate Matrix 90° by 90° degree

So this is a 90° turn This turn can be clockwise also, & this turn can be counter clockwise also, it only Depend on what question wants. Our question want 90° turn to be clockwise

Extra Space नहीं लेना और n^2 complexity में ही करना है।

Here, the first row become the last column. 0th row will be 3rd column.

* For rotating a matrix to 90° degrees clockwise. We need to transform each row of a Matrix to a column in rotated matrix

Page No.
Date

* For rotating matrix to 90°, we need to transform the rows into columns & the columns to rows in a result matrix.
So, the number of rows in rotated matrix will be equal to no. of columns of Original matrix and number of columns in rotated matrix will be equal to no. of rows. no. of row (m) * no. of column (n) (original) → no. of col (n) * (m)

```
int [][] rotatedMatrix = new int
        [cols of Original Matrix][rows of Original Matrix]
```

→ Size of Rotated Matrix will be n * m

In 90° Clockwise Rotation

So, to summarise:- If Size of original was m * n.

1. First row of Original Matrix will be last column of the RotatedMatrix.
2. Second row of original Matrix will be second last column of the Rotated Matrix
3. Third row of Original Matrix will be the last third column of the Rotated matrix & so on.

& So on

* We have a way of converting the rows of a matrix in the column of a matrix and that way is called Transpose of a Matrix.

So, What is Transpose of a Matrix?

The transpose of a Matrix is nothing but a flipped version of the matrix. When we exchange the rows & columns of a matrix such that each element A(i,j) gets swapped with A(j,i), then the matrix obtained is called the Transpose of Matrix.

इसे Counter Clock Wise rotation बोलते हैं।

Page No.
Date

Anticlockwise 90° rotation → TRANSPOSE

11	12	13	14
21	22	23	24
31	32	33	34
41	42	43	44

⟹

11	21	31	41
12	22	32	42
13	23	33	43
14	24	34	44

Transpose में Anti-clockwise / counter clockwise 90° rotation होती है।

Transpose

↓

90° Anti Clockwise Rotation

→ 1) पहली row पहला column बन जाता है।

→ 2) दूसरा row दूसरा column बन जाता है।

→ 3) Third row → Third column बन जाता है।

& so on

While Doing Transpose

element at arr[i][j] बन जाएगा element at arr[j][i]

90° Clockwise Rotation

1) पहली row last column बन जाती है। 2) दूसरी row last 2nd column बन जाती है।

3) Third row, last 3rd col बन जाती है

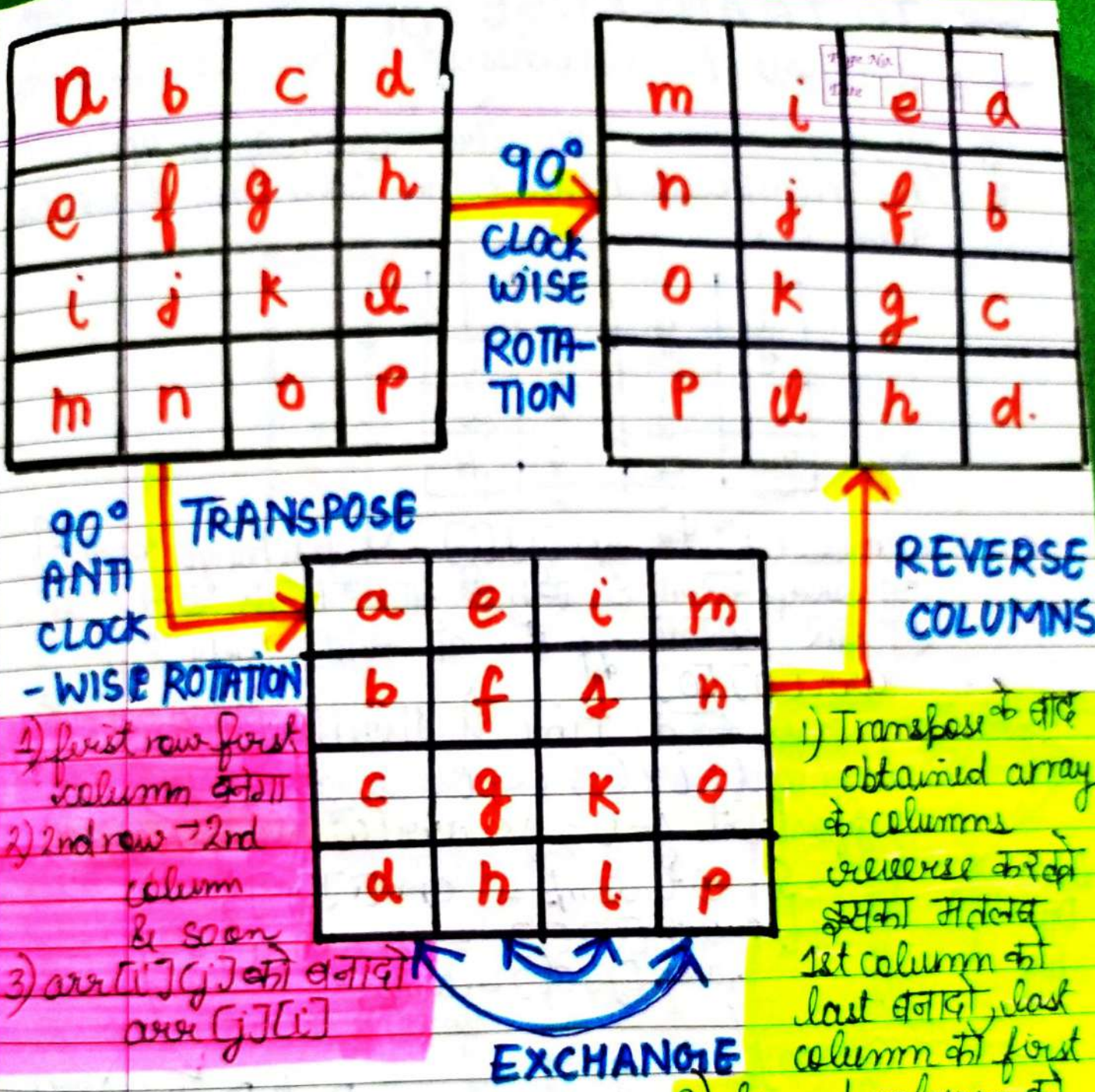
--

Clockwise 90° Rotation

11	12	13	14
21	22	23	24
31	32	33	34
41	42	43	44

⟹

41	31	21	11
42	32	22	12
43	33	23	13
44	34	24	14



1) first row first column करदो
2) 2nd row → 2nd column & so on
3) arr[i][j] को बनादो arr[j][i]

1) Transpose के बाद obtained array के columns reverse करदो इसका मतलब 1st column को last बनादो, last column को first
2) Second column को last second column बनादो
3) Third Column को last Third column बनादो & so on

→ TO TRANSPOSE or
→ To rotate 90° Clockwise

Page No.
Date

* तो पहले हमारा काम होगा Transpose करना
* फिर Transposed Matrix के Columns को reverse करना है।

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

arr[i][j] को arr[j][i] से exchange करना है या swap करना है एक ही बात है और बस एक ही बार करना है ये बात main है।

हमें Transpose करते Time ये गलती नहीं करनी है।

```
for(int i=0; i<arr.length; i++)
{for(int j=0; j<arr[0].length; j++)
  { int temp = arr[i][j];
    arr[i][j] = arr[j][i];
    arr[j][i] = temp;
  }
}
```

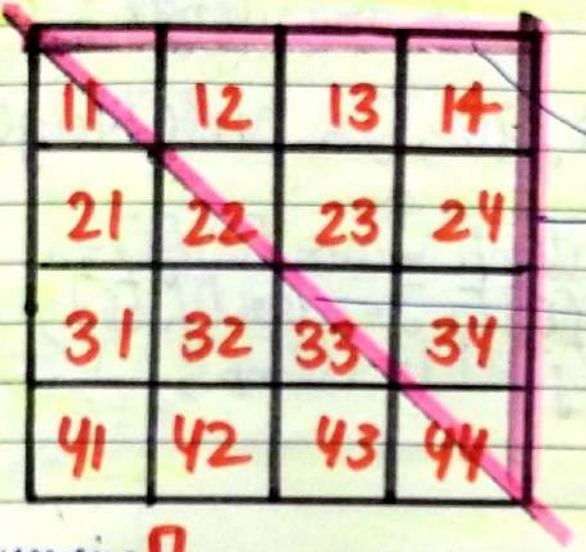
Dont Do like this — row के हर column के लिए ये loop चलेगा।

Don't Do this Mistake

Since we know that Transpose is a matrix in which arr[i][j] gets swapped or interchanged with arr[j][i], and this is exactly what we are going to do. If we traverse the entire matrix and swap arr[i][j] with arr[j][i], it is not going to work, and we will get the input matrix only as the result.

ऐसा इसलिए होरहा है क्योंकि हर पूरा matrix ही Traverse करते हैं तो सारी row की सारी columns को Traverse किया जा रहा है तो जो values हम पहले ही Swap कर चुके होते हैं, उन्हें हम वापस से Swap कर देते हैं।

ऐसा करने पर हमें अपना Input Matrix ही मिल लेगा as Output और Transpose नहीं होगा। तो अगर हमें Transpose करना है Matrix को, तो या तो हम Upper Traingle को ही Swap करेंगे या फिर सिर्फ Lower Traingle को।



After Traversing Only the Upper Part or only the Lower Part of the 2-d array & then doing the transpose, we get this 2-d array

We will traverse only this part for a Transpose.

11	21	31	41
12	22	32	42
13	23	33	43
14	24	34	44.

TRANSPOSED MATRIX

अगर हमें Transpose of Lower Triangle करना है, Transpose of Lower Triangle के लिए loop

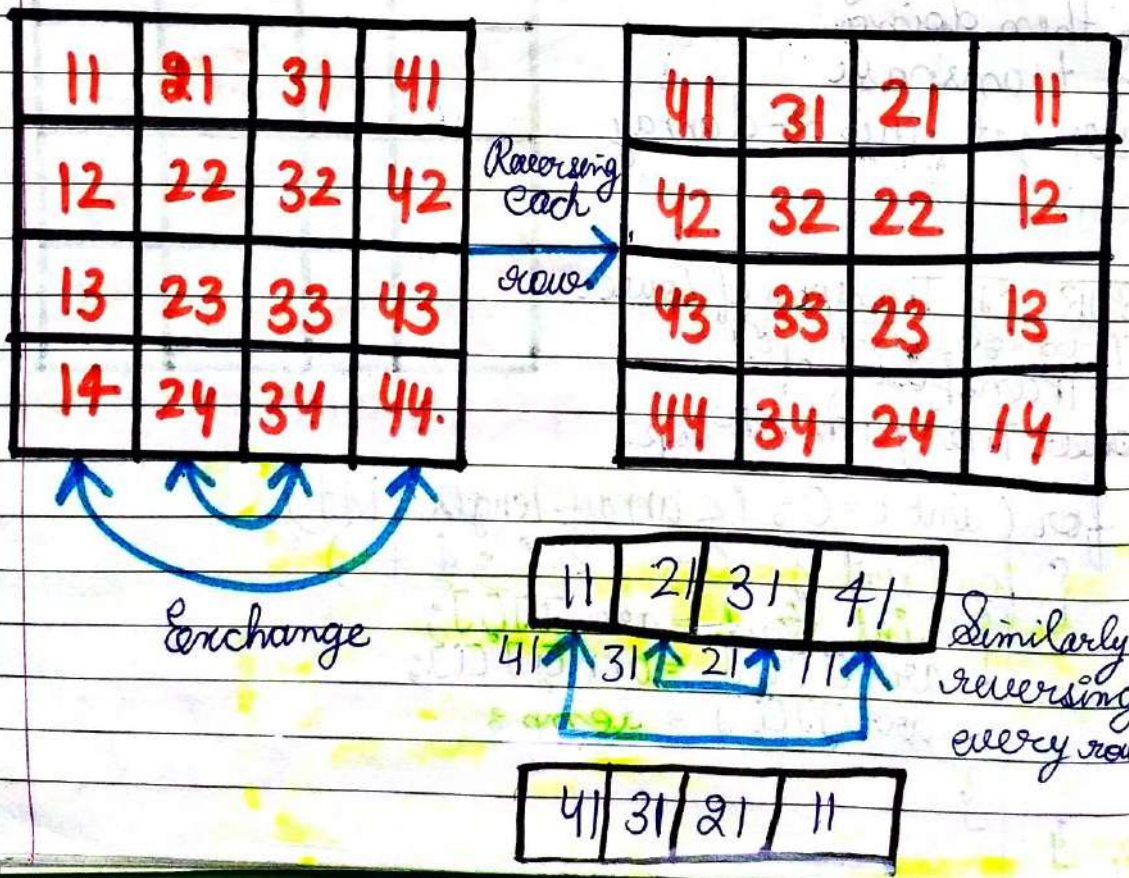
```
{ for (int i = 0; i < array.length; i++)
    { for (int j = 0; j <= i; j++)
        { int temp = arr[i][j];
          arr[i][j] = arr[j][i];
          arr[j][i] = temp;
        }
    }
}
```

अगर हमें Transpose of Upper Triangle करना है।

तो Transpose of Upper Triangle के लिए Loop

```
for (int i=0; i< arr.length; i++)
{
  for (int j=i; j< arr[0].length; j++)
  {
    int temp = arr[i][j];
    arr[i][j] = arr[j][i];
    arr[j][i] = temp;
  }
}
```

→ To reverse the Columns of Transposed Matrix



Row by Row को line से column reverse करो

Reverse column row by row

```
for(int i=0; i<arr.length; i++)
{
  int li = 0;                    ---> min col
  int ri = arr[0].length - 1;    ---> max col
  while (li < ri)                ---> right index from col
  {                                   left index of col
    int temp = arr[i][li];
    arr[i][li] = arr[i][ri];
    arr[i][ri] = temp;
    li++;
    ri--;
  }
}
```

-> यह तक left index of column right index से कम हो तभी तक ही करना है.

```
public class Main
public static void main (String[]
args) {
    Scanner scan = new Scanner (System.in);
    int n = scn.nextInt();

    int[][] arr = new int[n][n];
    for (int i=0; i<arr.length; i++)
    {
      for (int j=0; j<arr[0].length; j++)
        { arr[i][j] = scn.nextInt();
    }}

    //transpose
    for (int i=0; i<arr.length; i++)
     { for (int j=0; j<=i; j++)
        {
          int temp = arr[i][j];
          arr[i][j] = arr[j][i];
          arr[j][i] = temp;
        }
    }
    // reverse columns
for (int i=0; i<arr.length; i++)
{   int li=0;
    int ri = arr[i].length-1;
    while(li<ri)
      {int temp = arr[i][li];
       arr[i][li] = arr[i][ri];
       arr[i][ri] = temp;
                li++;
              ri--;
} }
```

```
for(int i=0; i<arr.length; i++)
{
  for (int j=0; j<arr[0].length; j++)
  {
    System.out.print(arr[i][j]+" ");
  }
  System.out.println();
}
}
```

DAY 31

4 Oct, 2021

Page No.
Date

DSA

2-D ARRAY

TO BE CONT....

1. Matrix Multiplication CW1
2. Rotate by 90° CW2
3. Spiral Display CW3

} Done as Day 30 HW.

DAY 32

5 Oct, 2021

DSA

2-D ARRAY - TO BE CONTINUED...

(CW 1) Spiral Display ✓ Done via Day 30 HW.

(CW 2). Diagonal Display (The State of Wakanda-2)

Ques 2) **The state of Wakanda - 2**

In this question, all we have to do is print the upper diagonal of the matrix, but the main point is, we need to print the elements ___ wise

Input ↓

a00	a01	a02	a03
11	12	13	14
	a11	a12	a13
21	22	23	24
		a22	a23
31	32	33	34
			a33
41	42	43	44

Output ↓

11 22 33 44
12 23 34
13 24
14

Diagonal 3 gap b/w row no & col no is 3.

Diagonal 2 gap b/w row no & col no is 2

Diagonal 1 gap between row no & col no is 1

Diagonal 0 gap between row no & col no is 0

Diagonal 0 से Diagonal 3 = 4 Diagonal in total हो

Size of 2-D Array में row column की 4 थी हो 4 ही Diagonal होंगी in total

Date

```
import java.util.*
public class Main
{
public static void main(String args[]){
Scanner scan = new Scanner(System.in)
{
int n= scn.nexInt();
int[][]arr =new int[n][n];
for(int i=0; i<arr.length; i++)
{
for(int j=0; j<arr[0].length; j++)
{
arr[i][j]= scn.nextInt();
}
```

क्योंकि Square Matrix है तो row column तो equal होंगे, तो एक ही input लेलेते हैं row col के लिए

अब जब हम Print करेंगे तो diagonals को gap wise print करेंगे पहले Diagonal 0 print होगा मतलब diagonal with gap 0, for diagonal with gap 1, for diagonal with gap 2, & so on...

Normally हम row wise print कराते हैं मतलब पहले हम एक row के सारे column print करते हैं फिर row 2 के सारे column Print करते हैं

लेकिन हम इस बार Diagonal wise Traverse करेंगे तो पहले सारे 1st Diagonal (Diagonal with 0 gap) के सारे columns Print करेंगे फिर Diagonal 1 के सारे column फिर Diagonal 2 & so on

```
for(int g=0; g<arr.length; g++)   ] gap
{ for(int i=0; j=g; j<arr.length; i++, j++)
System.out.println(arr[i][j]);
}
}
}
```

Normally we access columns like this ↑

Diagonal Gap Method, we access col like this

gap को identify कैसे करें? →

जितने cols उतने तक gap बढ़ सकता है।

i होता है gap के बराबर शुरू में

0 से < row size जितना हो सकता है।

→ कोई भी diagonal Start हो रहा है तो i तो 0 से ही शुरू होता है

इस ques में GAP Method use हो रहा है।
DP के almost 15-20 ques में ऐसे Gap method use करना होगा। So this very very imp Method

Approach →

1. Run an outer loop from 0 to n-1.
2. Run an inner loop for i and j and printing the element.

Pseudocode

1) gap from 0 to n-1. // first loop
2) i = 0 and j = gap till j < n.
   (initialization, condition)
3) print element i, j
4) increment i
5) increment j

1) * So, we can first write an outer loop that will iterate through the n diagonals i.e from 0 to n-1 gap. Its inner loop will print the elements in the diagonal such that j-i=gap.

2) But, this printing should go how long?

⟹ Until (j<n), the last elements of the diagonals have 4as its column index i.e j=3.

Rukne का Mark J तय करता है।

DP में रोजाना ही almost हमें Gap Method से वाकिफ होना पड़ेगा।

j की value जबतक <n है तबतक ही Print करते रहो।

Page No.
Date

* Time Complexity of Diagonal Display

When gap = 0, the inner loop runs n times.
When gap = 1, the inner loop runs (n-1) times.
When gap = 2, the inner loop runs (n-2) times.
.
.
.

So considering all the gaps the inner loop will be running.

→ $n+(n+1)+(n-2)+(n-3)$ ——— $\frac{n*(n+1)}{2}$.

which is of order $O(n^2)$

* Space Complexity

→ We are not using any auxiliary space & hence the space complexity is $O(1)$.

Day 34

7 Oct, 2021

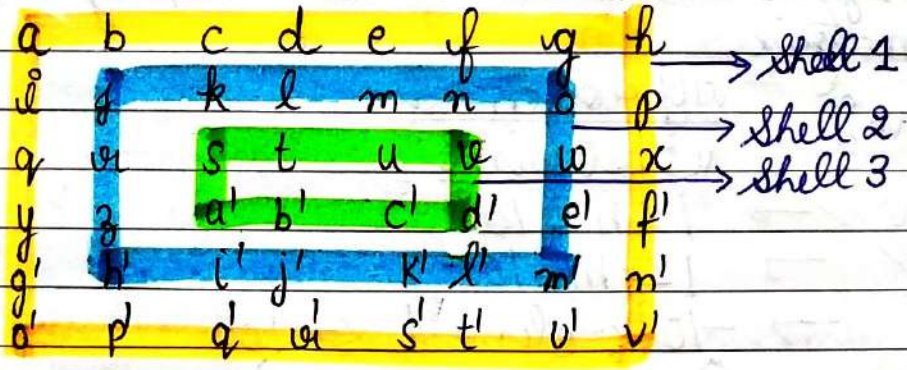
DSA

2-D ARRAY TO BE CONTINUED...

Done ✓ CW1 → Diagonal Display → State of Wakanda 2
In CW of Day 34.

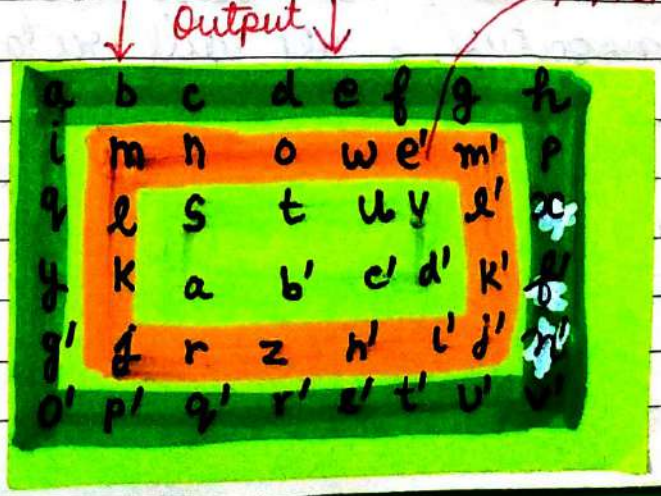
CW2 → Ring Rotate

Ques 1. Ring Rotate



अगर हमें ये बोलदिया जाए कि ring 2 के हर element को 3 से rotate करदो तो हर element ring 2 का, अपनी Position से 3 position anticlockwise rotate होजाएगा और 3 position नीचे या पीछे की तरफ shift होजाएगा।

S = 2
r = 3



Page No.
Date

1. *

Let's know better what the question says

We are given a n*m matrix where n is the no. of rows & m is the no. of columns.

हमें एक s नाम का variable भी input लेना पड़ेगा और एक r नाम का variable भी।

→ S will denote the shell no.
→ r will denote the rotations in an anticlockwise manner of the specified ring/shell.

2. *

We are required to rotate the sth ring/shell by r rotation and display the rotated matrix

3. *

Important Point

Approach & Constraints

→ $1 \leq n \leq 10^2$
→ $1 <= m \leq 10^2$
→ $-10^9 <= elements <= 10^9$
→ $0 < s <= \frac{min(n,m)}{2}$
→ $-10^9 < r < 10^9$ } This means r can be negative too.

⇒ If r is positive, we will rotate it in anticlockwise / counterclockwise direction
⇒ If r is negative, we will rotate it in clockwise direction.

→ Approach

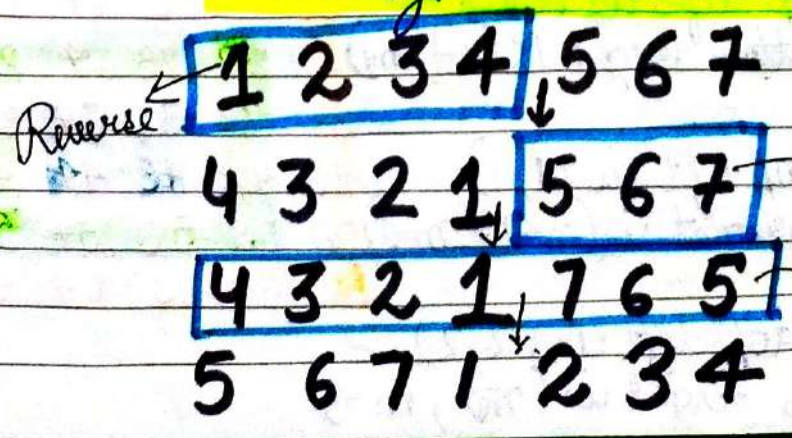
1. Make a one-d array (of size of no. of element in sth ring) & fill it with the ring (Use Spiral Display to fill 1D array with element of sth ring.
   Size of 1-D array ~~निकालने के लिए~~ का formula ~~दिया गया~~.
2. rotate the 1-D array with r rotations.
3. Now, again fill back with the correct rotated elements from the updated & rotated 1-D array to the sth ring.

4) * Pseudocode for Ring Rotate

```
a) oned = fillOnedFromShell (arr, s)
b) rotate (oned, r)
c) fillShellFromOned (arr, s, oned)
```

5) * How to Rotate a One-d array with K rotations,

We will just call the reverse() function thrice to shift a one dimensional array with K rotations



Shift = 3

[1, 2, 3, 4], 5, 6, 7

Reverse

4, 3, 2, 1, [5, 6, 7] → Reverse

4, 3, 2, 1, 7, 6, 5 → Reverse

5, 6, 7, 1, 2, 3, 4

* Pseudocode for Rotate

1) Reverse(oned, 0, oned.length - r - 1);
   (0 → starting point; oned.length - r - 1 → ending point)

2) Reverse(oned, oned.length - r, oned.length - 1)

3) Reverse(oned, 0, oned.length - 1);

6) * How to get an array from shell & vice-versa

Lets know that,

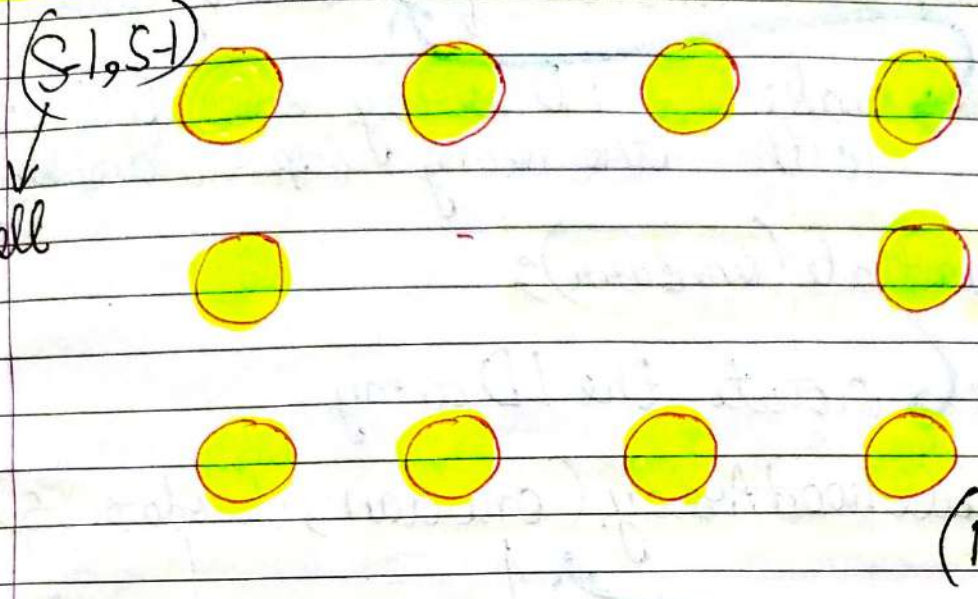
for shell 1, the top left is (0,0) & the bottom right is (n-1, m-1)

for shell 2, the top left is (1,1) & the bottom right is (n-2)(m-2)

for shell 3, the top left is (2,2) & the bottom right is (n-3, m-3)

11	12	13	14	15	16
21	22	23	24	25	26
31	32	33	34	35	36
41	42	43	44	45	46
51	52	53	54	55	56
61	62	63	64	65	66

→ For shell S, the top left is (S-1, S-1)
& the bottom right is (n-S, m-S)



7) * What is number of elements in a particular shell?

अगर हम ये करे

(2 × no. of element in row) + (2 × no. of element in col)

ऐसा अगर हम करते है तो शायद count ज्यादा होवे

Eg → Do for the above given shell

→ (2×4 + 2×3) = 14

So total element in above shell would be = 14 - 4 = 10

14 is 4 more than the actual no. of elements. It is because we are counting the vertex 2 times (In no. of cols & in no. of rows also)

So, the No. of element in Shell =
= (2 × no. of row in shell) + (2 × no. of cols in shell) - 4

तो इस ques में हमें 3 function बनाने पड़ेंगे।

1) `int [] onedarr = fillOnedarray (twodarr);`

make a 1D array and fill it with the ring (spiral display)

2) `rotate (onedarr);`

This func is void return type

rotate the 1D array

3) `fillTwodArray (onedarr, twodarr, 5);`

Void return Type

rotated Onedarray

Input Shell.

fill the ring back from rotated 1-d array

Time Complexity taken by Ring Rotate
→ O(n+m)

Space Complexity taken by Ring Rotate
→ O(n+m).

पहले rotate करने वाला function बनाते हैं.

Page No. | Date

2nd function → Rotate.

```
public static void rotate(int[] onedarr, int r)
```

return type void है मतलब कुछ return नहीं करता ये func. | कितनी rotations लगानी हैं। | Parameter onedarr देना है जिसको rotate करना है.

```
{
    r = r % onedarr.length;
```

अगर onedarr की length 4 है और rotations 7 given हैं तो overall हमें 3 ही rotation करनी हैं bcz 7%4 = 3.

```
    if (r < 0)
    {
        r += onedarr.length);
    }
```

अगर हमें rotation -2 given हैं तो हमें overall -2+4 = 2 rotation ही लगानी bcz afterall दोनों का output तो same ही होगा।

.. हमें अब 3 reverse लगाने पड़ेंगे। या 3 बार reverse function को call करना पड़ेगा. क्योंकि) पहले हम first (onedarr.length - r) elements को reverse करना है।
2) फिर हमें last (r) elements को reverse करना है।
3) फिर हमें पूरे ही 1-d array को reverse करना है।

```
    reverse(a   , onedarr.length - r - 1);
    reverse(onedarr, onedarr.length - r, onedarr.length - 1);
    reverse(onedarr, 0, onedarr.length - 1);
}
```

Reverse an Array -func

```
public static void (int[] onedarr,
                    int left,
                    int right)
{
  while(left < right)
  int temp = onedarr[left];
  onedarr[left] = onedarr[right];
  onedarr[right] = temp;
```

→ reverse किसी reverse करता है।

→ given array को कहाँ से

→ कहाँ तक reverse करना है।

→ left index पर जो element पड़ा है उसे temp में store करलो।

→ अब right index पर temp वाली value store करदो।

→ left वाले index पर right index वाली value डालदो।

```
  left ++;
  right --;
  }
}
```

→ अब जो left index है उसको 1 से बढ़ाते चलो और बस तब तक ही बढ़ाना है जबतक वो right index से कम रहता है।

→ right index की value 1 से decrease करते चलो।

FILL 2-D ARRAY FROM 1-D ARRAY

Page No.
Date

```
public static void fillTwoDArray
    (int[] onedarr, int[][] twodarr, int s)
{
}
```

minrow → mr (0,0)

11	12	13	14	15	16	17	18
21	22	23	24	25	26	27	28
31	32	33	34	35	36	37	38
41	42	43	44	45	46	47	48
51	52	53	54	55	56	57	58
61	62	63	64	65	66	67	68

max row, maxcol.

	rmin	cmin	rmax	cmax
S1	0	0	5	7
S2	1	1	4	6.
S3	2	2	3	5

row=6,
col= 8.

row min और column min shell-1 के equal होते हैं।

eg. for S1
S-1. { row min=0, col min= 0.

row max होता है row-shellno.
col max होता है col-shellno. ← row-S { row max, col-S. { col max

```
int rows = twodarr.length;
int cols = twodarr[0].length;
int rmin = s-1;
int cmin = s-1;
int rmax = rows-s;
int cmax = cols-s;
int idx = 0;
```

] Row, col var लिये

idx चाहिए → traversal index
जो 1-Darr में traversal करने के लिए

अब हमें 2-d array में shell के सारे elements 1-d array के elements के according change करते हैं।

तो क्योंकि हमें Anticlockwise rotation करनी है तो हम पहले left wall करेंगे, फिर bottom wall करेंगे, फिर right wall करेंगे, और फिर Top wall करेंगे।

Just like Spiral Display

Page No.
Date

// left wall (col at cmin, row varies from rmin to rmax).

Left wall

```
for (int i=rmin, j=cmin; i<=rmax; i++)
{ twodarr[i][j] = onedarr[idx];
  idx++;
}
```

idx को बढ़ाते चलना for 1d array traversal

cmin left wall को represent करता है तो जब left wall पूरी हो जाए तो हम cmin बढ़ा देंगे।

```
cmin++;
```

// Bottom wall (row at rmax, col varies from cmin to cmax)

Bottom line/wall

```
for(int i=rmax, j=cmin; j<=cmax; j++)
{ twodarr[i][j] = onedarr[idx];
  idx++;
}
rmax--;
```

Bottom wall line is represented by rmax, जब bottom line पूरी हो जाए तो rmax को decrease कर दो

// Right wall [col at cmax, row varies from rmax to rmin]

Right wall

```
for(int i=rmax, j=cmax; i>=rmin; i--)
{
  twodarr[i][j] = onedarr[idx];
  idx++;
}
cmax--;
```

Right wall is represented by cmax, जब Right wall पूरी हो जाए तो cmax को decrease करदो।

// top wall [row at min, col varies from cmax to cmin]

Top wall

```
for(int i=rmin, j=cmax; j>=cmin; j--)
{
  twodarr[i][j] = onedarr[idx];
  idx++;
}
rmin++;
```

// Top wall is represented by rmin, जब top wall पूरी हो जाए तो rmin ++ करदो।

} → The function fillTwoDArray ends here

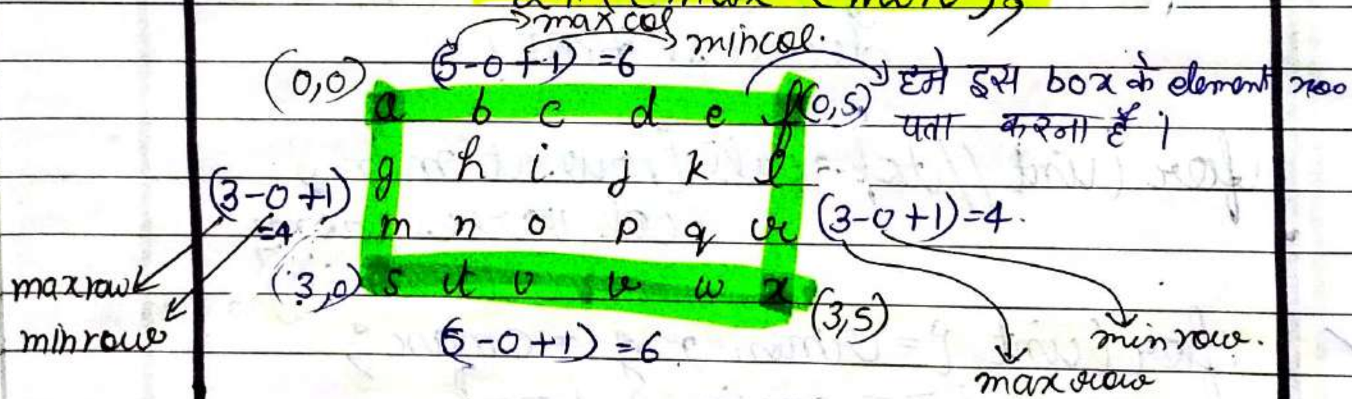
FILL 1-D ARRAY FROM 2-D ARRAY OR FROM INPUT ARRAY

Page No.
Date

→ public static void fillOneDArray
(int[][] arr, int s)
{ ↳ give the input as 2-d array

int rmin = s-1;
int cmin = s-1; — s is the shell no.
int rmax = arr.length - s; → row
int cmax = arr[0].length - s; → cols.

int size = 2*(rmax - rmin) +
2*(cmax - cmin);



= 2x(maxrow - minrow + 1) +
2x(maxcol - mincol + 1)
= 2 + 2(max row - min row) +
2 + 2(max col - min col)
= 4 + 2(maxrow - minrow) +
2(maxcol - min col)

ये 4 nos vertex हैं included बार है तो इन्हें ये नहीं चाहिए. ∴ formula बनेगा

Size = 2(rmax - rmin) + 2(cmax - cmin)

```
int [] oneDArray = new int [size];
int index = 0;
for(int i = rmin ; j = cmin ; i<=rmax;
                (i++)
{
  oneDArray [index] = arr[i][j];
  index++;
} cmin++;
for(int i=rmax ; j=cmin ; ; j<=cmax
  ; j++)
{ OneDArray [index] = arr[i][j];
  index++;
}
  rmax--;
for(int i=rmax ; j = cmax; i>=rmin;
  i--)
{
  OneDArray [index] = arr[i][j];
  index++;
} cmax--;
for(int i=rmin ; j =cmax-1; j>=minc; j--)
{ OneDArray [index]= arr[i][j];
  index++;
}
return OneDArray;
}
```

Left Wall की elements 1D में store

Bottom wall की elements 1-D array में store

Right Wall की elements 1-D array में store

Top wall की element 1-D array में store

Ring Rotate <Code>

```
import java.util.*;
import java.io.*;
public class Main
{
    public static void main (String[] args)
    {
        Scanner scn = new Scanner (System.in);
        int n = scn.nextInt();
        int m = scn.nextInt();

        int[][] arr = new int[n][m];
        for(int i=0; i<n; i++)
        { for (int j=0; j<m; j++)
            {
                arr[i][j]=scn.nextInt();
            }
        }

        int s= scn.nextInt();   → shell no
        int r= scn.nextInt();   → rotations

        ringrotate ( arr,s, r);   → ring rotate func
        display (arr);   → To print the Output 2d array

    }

    public static void ringrotate (int[][]arr,
        int s, int r)
    { int [] oneDArr = fillOneDArr (arr, s);
        rotate (oneDArr, r)
        fillTwoDArray (arr, s, oneDArr);
    }
```

We are taking the input 2d array

```
public static void rotate (int [] OneDArr,
    int r)
{ r = r % OneDArr.length;
  if (r < 0)
  {
    r = r + OneDArr.length;
  }
reverse( OneDArr, 0, OneDArr.length - r - 1);
reverse( OneDArr, OneDArr.length - r,
                  OneDArr.length - 1)
reverse( OneDArr, 0, OneDArr.length - 1);
}
public static void reverse (int [] arr,
  int left, int right)
{ while (left < right)
  { int temp = oneDArr[left];
    OneDArr[left] = OneDArr[Right];
    OneDArr[Right] = temp;
  left ++;
  right --;
  }
}
```

```
public static void fillTwoDArray
  (int [] OneDArr, int s, int [][] TwoDArr )
{
  int rows = TwoDArr.length ;
  int cols = TwoDArr[0].length;
  int rmin = s-1;
  int rmax = rows - s;
  int cmin = s-1;
  int cmax = cols - s;

  int idx = 0;

  for(int i = rmin, j = cmin ; i<=rmax;
      i++)
  {
    TwoDArr[i][j] = OneDArr[idx];
    idx++;
  }

  cmin++;

  for(int i=rmax, j = cmin ; j<= cmax ;
      j++)
  {
    TwoDArr[i][j] = OneDArr[idx];
    idx++;
  }

  rmax--;
```

```
for(int i=rmax, j=cmax; i>=rmin
    ;i--)
{ TwoDArr[i][j]= OneDArr[idx];
   idx++;
}
cmax--;
for(int i=rmin; j= cmax; j>=cmin
    ;j--)
{ TwoDArr[i][j]= OneDArr[idx];
   idx++;
}
  rmin++;
}
}
```

```
public static void fillOneDArr
( int[][] arr, int s.                    )
{ int row = arr.length;
  int column = arr[0].length;
int rmin = s-1;
int cmin = s-1;
int rmax =  row -s;
int cmax =  column -s;

int size = 2(rmax - rmin) + 2(cmax - cmin);
int[] oneDArray = new int[size];

int index =0;

for(int i= rmin, int j= cmin; i<= rmax;
    i++).
{ OneDArray[index] = arr[i][j];
  index++;
}
cmin++;

for( int i = rmax, int j= cmin,
     j<= cmax; j++)
  { OneDArray[index]= arr[0][j];

    index++;
  }
  rmax--;.
```

```
for( int i=rmax , int j= cmax ;
    i >=rmin ; i --)
{ OneDArray [index ]= arr[i][j];
  index ++;
}
cmax --;

for( int i = rmin , int j = cmax ;
  j >= cmin ; j--)
  {
  OneDArray[index ]= arr[i][j];
    index ++;
  }

rmin ++;
} return OneDArray ;
}
  public static void Display (int[][]arr)
{
{ for(int i=0 ; i<arr.length ; i++)
{ for(int j=0 ; j<arr[0].length ; j++)
  { System.out.print (arr[i][j]+" ");
  }
  System.out.println ();
}
}
```

DAY 35

DSA

8 Oct, 2021

2D-ARRAY TO BE CONT....

CW1) Ring Rotate } Done in Class 34 CW

CW2) Search in a 2D Sorted Array

CW3) Saddle Price

Ques.2 Search in a 2D Sorted Array

This question wants us to search a given element in a array with 2-dimensional and the array is sorted in rows as well as in columns.

→ हमे अगर element मिले तो उसकी row column print करानी है।

किसी भी element को array मे search करने के हमने 2 तरीके सीखे थे।

1) Linear Search
2) Binary Search

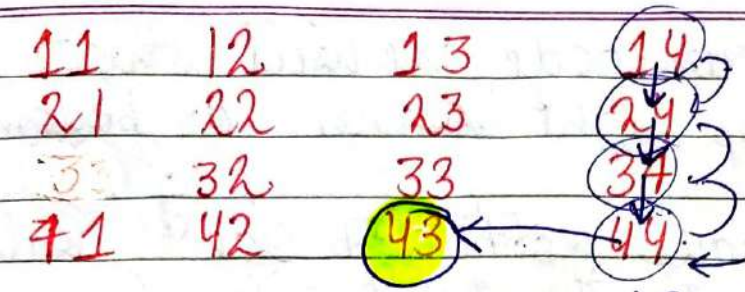
हम 2-D array मे भी Liniar Search apply कर सकते है but vo most of the cases मे ज्यादा Time consuming होगा।

हमें यहाँ इस ques मे Lenear Search apply कर सकते थे लेकिन हम यहाँ linearly search नही करना चाहते

Trying to do Binary Search in 2D Array

Applying Binary Searching Method in 2-D

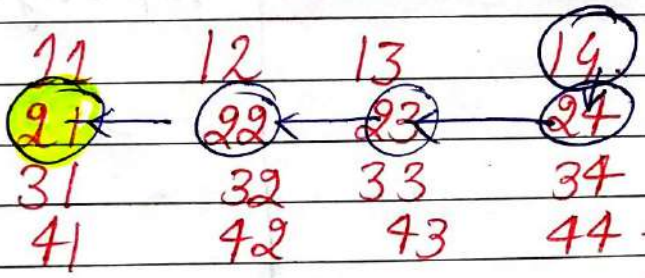
Page No.	
Date	



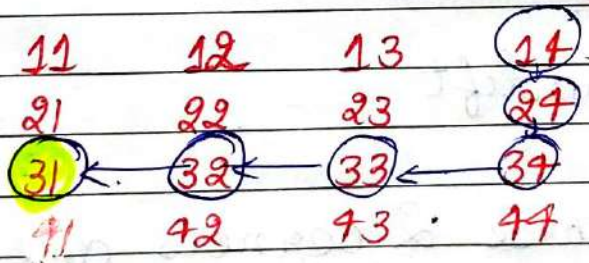
Eg → हमे अगर 43 search करना है।

→ हम किसी भी Corner से Searching start कर सकते हैं
→ अगर 43 arr[i][j] से छोटा / बड़ा है तो accordingly direction change करनी है next element पर search करने के लिए।

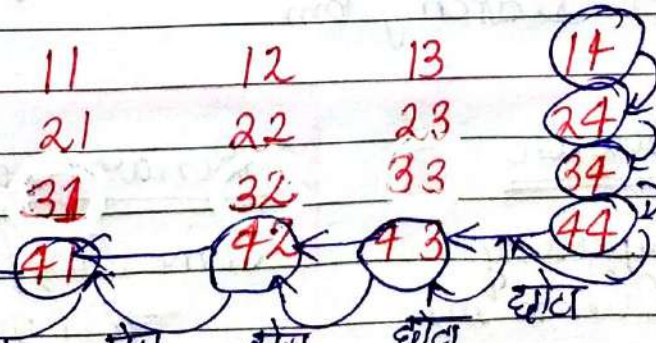
Search 21



Search 31.



Search 40

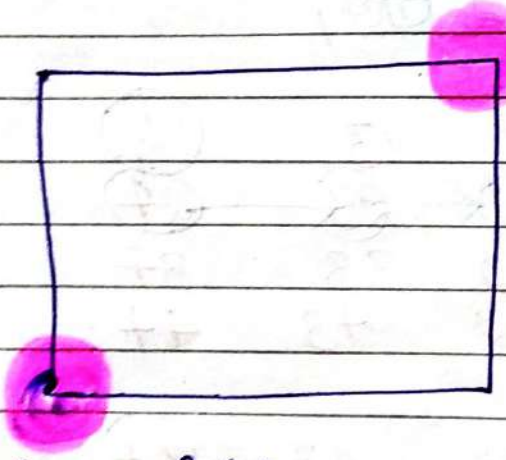


अब तो बाहर आगए Matrix से 40

col < 0 ∴ Not found

for our code, we will start from Top right corner. or bottom left.

Because उनकी एक side से value कम हो रही है और दूसरी side value बढ़ रही है तो यहाँ Binary Search वाला scene बनता है कि 50 part में ही search करना होगा अब तो, बाकी 50% की तरफ तो देखना ही नहीं है।



So, these 2 corners are good to start search from.

Linear Search

Time Complexity
→ $O(n * m)$

Space Complexity
→ $O(1)$

Binary Search

Time Complexity
→ $O(m+n)$

Space Complexity
→ $O(1)$

Search in 2D Array using Binary Search

<code>

```
import java.util.*;
import java.io.*;
public class Main
{
    public static void main (String[]args)
    {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();

int[][] arr = new int[n][n];
for(int i=0; i<arr.length; i++)
    {
        for(int j=0; j<arr[0].length; j++)
    {   arr[i][j]= scn.nextInt();
    }
    }
}

int x = scn.nextInt();        ] जो element find करना है
int i = 0;                    } Top Right
int j = arr[0].length - 1;    } Corner.
while (i<arr.length && j>=0)
    {
    if (x > arr[i][j])
        {
        i++;
        }
    else if (x<arr[i][j])
    {j--;
    }
```

जब तक i की value arr की length से कम है और j >=0 है तब तक ही Searching होगी

Next row में आजाओ क्योंकि element बड़ा है तो

LHS चलो अगर element छोटा है तो

```
else
{
    System.out.println(i);
    System.out.println(j);
    return;
}
}
System.out.println("Not found");
}
```

row print करो फिर column

(Recess 3)

Page No.
Date

Saddle Price

The Saddle Price is defined as the least value in the row and the maximum value in the column.

और अगर ऐसा कोई Point न मिले, इसका मतलब उस Matrix में Sadle Point/Price है ही नहीं।

2 Saddle Points are not possible, why?

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

→ Prove f & k can't be?

f & K can't be the Saddle Points at the same time.

$i < f < g$
$g < k < j$
} ऐसा नहीं है Possible

$i < f < g < k < j$

Not Possible.

2) Prove p & a can't be saddle Points?

$m < a < d$

$d < p < m$

$m < a < d < d < m$, Not Possible.

Date

Approach

1. एक loop लगेगा row 0 से n-1 तक। उसमे सबसे पहली वाली row मे देखो कि कौनसा element सबसे छोटा है फिर उस element को हमे columnwise भी check करना पड़ेगा तो उस column को save करलो

2) अब col 0 से n-1 तक देखेंगे कि क्या वो अपने col का सबसे बड़ा है।

* Time Complexity

In the worst case scenario, we might need to traverse all the elements in, so the time complexity comes out to be n square $(n * n)$.

* Space Complexity.

Since, we are using only a handful of variables, space complexity is constant, $O(1)$.

```
→ import java.util.*;
→ import java.io.*;
→ public class Main{
    {
      public static void main(String[]args)
{
{ Scanner scn = new Scanner(System.in);
  int n = scn.nextInt();

int[][] arr = new int[n][n];
for(int i=0; i<arr.length; i++)
{
 { for (int j=0 ; j<arr[0].length ; j++)
   { arr[i][j]=scn.nextInt();
   }
 }
}
for(int i=0; i<arr.length; i++)
{
{ int min = arr[i][0];
  int psj = 0;                    → potential saddle j
  for(int j=1; j<arr[0].length ; j++)
   {
     if (arr[i][j]<min)
     {min = arr[i][j];
      psj =j;    } save करते हैं इस column no को
   }
}
```

```
boolean isSaddle = false;
break;
}
}
}

if (isSaddle == true)
{
System.out.println(min);
return;
}
```

2D ARRAY DONE