Todays Content

1. Row wise & Column wise sum
2. Identity matrix
3. Diagonal Printing

Matrix: Declaration:

1. int mat[4][5]
   - [4] → Rows/Horizontal
   - [5] → Columns/Vertical
2. vector< vector<int> > v(4, vector<int>(5));
   - 4 → 4 rows

Matrix Index:

	0	1	2	3	4
0					
1			10		
2					20
3					

mat[1][2] = 10

mat[2][4] = 20

Q1: mat[N][M]
- N → N rows
- M → M Columns

mat[0][0]

mat[0][M-1]

	0	1	.....	j	....		M-1
0				0,j			
1				1,j			
⋮							
i	i,0	i,1	i,2	i,j	....		i,M-1
⋮							
N-1				N-1 j			

mat[N-1][0]

mat[N-1][M-1]

obs:

1. Iterate in i^{th} Row: Col change [0..M-1]
2. Iterate in j^{th} col: Row change [0.. N-1]

Q Sum of elements in each row

Ex: mat[4][5];

Idea: For every row, iterate & calculate sum & print

	0	1	2	3	4	Output
0	10	20	30	40	50	150
1	1	2	3	4	5	15
2	6	7	8	9	10	40
3	10	20	30	40	50	150

```
void sumRow( int mat[][], int N, int M){
    for(int i=0; i<N; i++){
        # i^th Row: iterate & calculate sum.
        long sum=0;
        for(int j=0; j<M; j++){
            sum=sum+mat[i][j];
        }
        print(sum);
    }
}
```

TC: O(N*M) SC: O(1)

Q Sum of elements in each col

Ex: mat[4][5];

	0	1	2	3	4
0	10	20	30	40	50
1	1	2	3	4	5
2	6	7	8	9	10
3	10	20	30	40	50

Output:

27 49 71 93 115

```
void colRow( int[][] mat, int N, int M)
    for(int j=0; j<M; j++){
        # j: Column, iterate & calculate sum;
        long sum=0;
        for(int i=0; i<N; i++){
            sum= sum + mat[i][j];
        }
        print(sum);
    }
    TC: O(N*M) SC: O(1)
}
```

Identity Matrix: N = M

Given a square matrix, check if its identity matrix or Not.

An Identity matrix, all main diagonal has only 1 and all other cells 0.
↳ mat[r][c], r==c.

Ex1:

$$I = \begin{array}{c|cccc|} & 0 & 1 & 2 & 3 \\ \hline 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 \\ 3 & 0 & 0 & 0 & 1 \end{array}$$

Ex2:

$$I = \begin{array}{c|ccc|} & 0 & 1 & 2 \\ \hline 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \end{array}$$

Ex3:

$$I = \begin{array}{c|ccc|} & 0 & 1 & 2 \\ \hline 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{array}$$

Idea:

Note:

Ex3:

$$I = \begin{array}{c|ccc|} & 0 & 1 & 2 \\ \hline 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{array}$$

Ex4:

$$I = \begin{array}{c|ccc|} & 0 & 1 & 2 \\ \hline 0 & 2 & 3 & 4 \\ 1 & 1 & 4 & 3 \\ 2 & 1 & 6 & 6 \end{array}$$

```
public int solve(int[][] A){

}
```

Dry Run:

$$I = \begin{array}{c|ccc|} & 0 & 1 & 2 \\ \hline 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{array}$$

28 Given a mat[N][N] print both main diagonals in new line.

Ex1: mat[4][4]; square

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

Output

d1 (L→R) : 6 3 6 9

d2 (R→L) : 4 2 7 10

Idea:

	0	1	2	3
0	0,0	7	3	4
1	9	1,1	2	8
2	4	7	2,2	9
3	10	3	2	3,3

d1 : (L→R)

i=0;

print(mat[0][0]) i++

print(mat[1][1]) i++

print(mat[2][2]) i++

print(mat[3][3]) i++

i=4; stop

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

4,-1

d2:

i=0, j=3

print(Mat[0][3]) i++, j--;

print(Mat[1][2]) i++, j--;

print(Mat[2][1]) i++, j--;

print(Mat[3][0]) i++, j--;

i=4, j=-1 ; Stop.

Notes:

1. If single variable based loops: Prefer for loop
2. If >1 variable based loops: Prefer while loop.

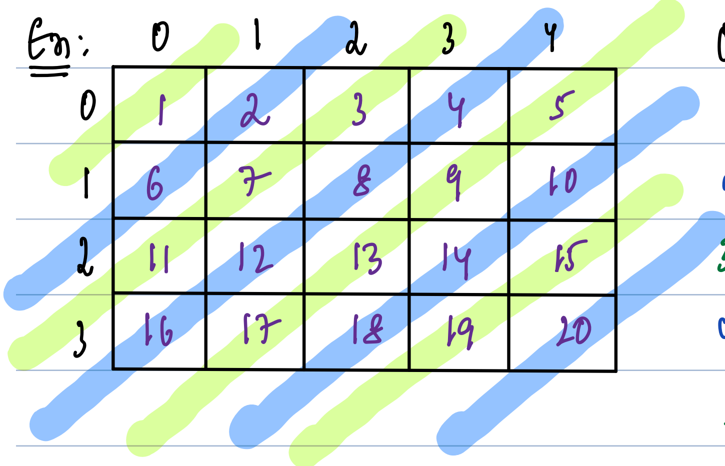
```
void printdiagonal(int mat[][], int N){
    #Printing d1: L→R
        for(int i=0; i<N; i++){ TC:O(N)
            print(mat[i][i]);
        }

    #Printing d2: R→L
        int i=0, j = N-1;
        while(i<N && j>=0){
            print(mat[i][j]);
            i++; j--;
        }
}
```

4Q

Given a mat[N][M]

Print all diagonals going from Right to Left & Top to down.

Ex:

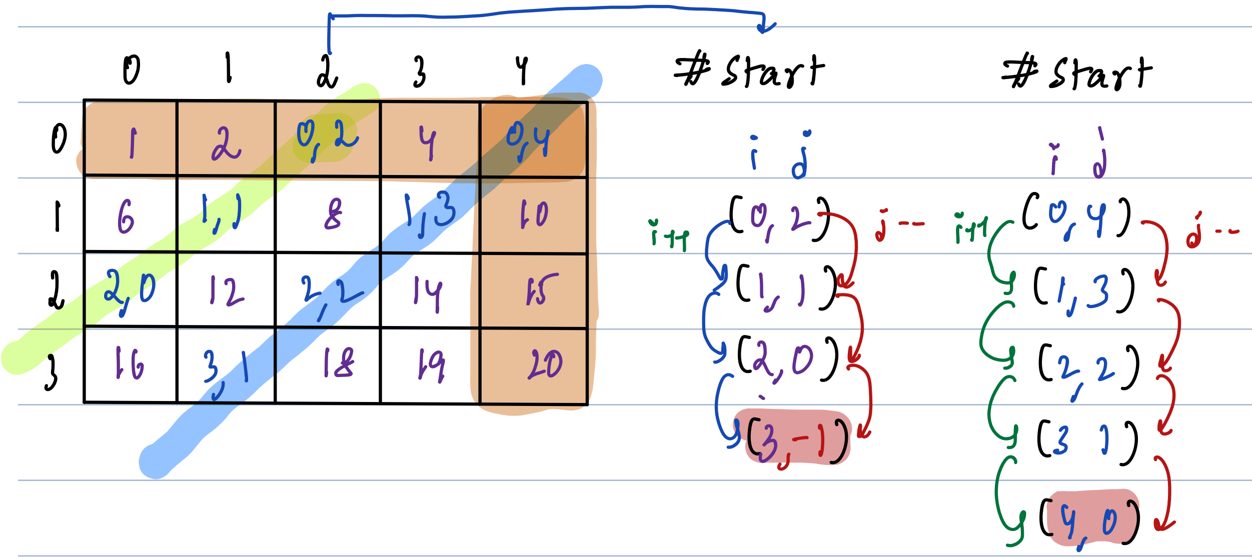


	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20

Output:

1
2 6
3 7 11
4 8 12 16
5 9 13 17
10 14 18
15 19
20

Hint1:



	0	1	2	3	4
0	1	2	0,2	4	0,4
1	6	1,1	8	1,3	10
2	2,0	12	2,2	14	15
3	16	3,1	18	19	20

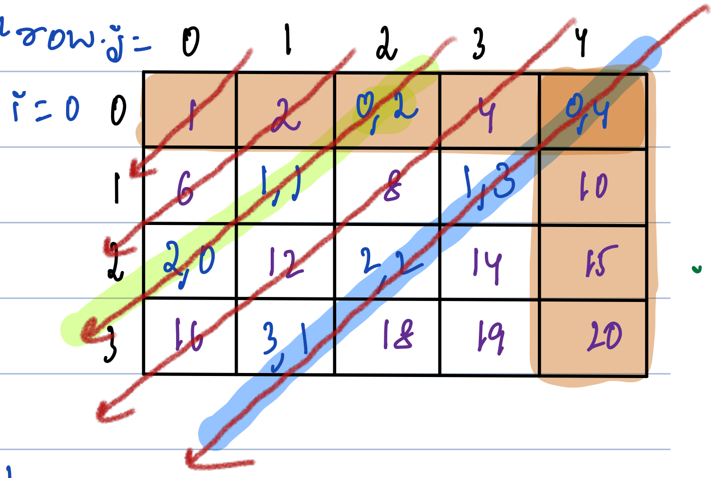
```
void printRL(int mat[][], int N, int M, int i, int j){
    # (i, j) is start of R→L diagonal
    while(i<N && j>=0){
        print(mat[i][j]);
        i++; j--;
    }
}
```

#Idea2: R→L diagonals.

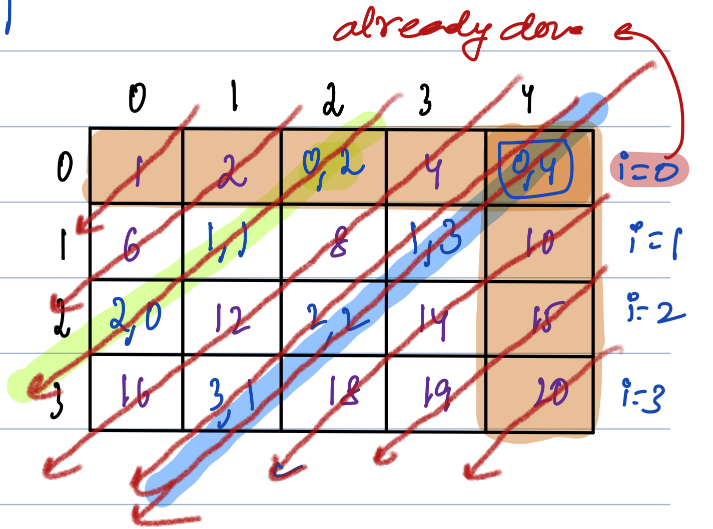
1. They can start at 0th row : Iterate & take every cell in 0th row as start point
2. They can start at last col : Iterate & take every cell in last col as start point

```
void printRL(int mat[][], int N, int M, int i, int j){
    # (i,j) is start of R→L diagonal
    while(i<N && j>=0){
        print(mat[i][j]);
        i++; j--;
    }
}
```

```
void printdiagonal(int mat[][], int N, int M){ TC: O(N*M) SC: O(1)

    #Step1: Print diagonals starting at 0th row
    for(int j=0; j<M; j++){
        # Start point: (0,j)
        printRL(mat, N, M, 0, j);
    }

    #Step2: Print diagonals start at last col
    for(int i=1; i<N; i++){
        #Start point: (i, M-1)
        printRL(mat, N, M, i, M-1);
    }
}
```



j= 0 1 2 3 4
i=0
0 1 2 3
1 | 2 | 0,2 | 4 | 0,4
6 | 1,1 | 8 | 1,3 | 10
2,0 | 12 | 2,2 | 14 | 15
16 | 3,1 | 18 | 19 | 20



already done
0 1 2 3 4
0 1 2 3
1 | 2 | 0,2 | 4 | 0,4
6 | 1,1 | 8 | 1,3 | 10
2,0 | 12 | 2,2 | 14 | 15
16 | 3,1 | 18 | 19 | 20
i=0
i=1
i=2
i=3