Muldimensional Fun

Thursday, 26 August 2021 7:36 PM

2D/Arrays Context

int mat[3][4]

→ 3 rows / Horizontal

→ 4 columns / vertical

	0	1	2	3
0	(0,0)			
1			(1,2)	
2				(2,3)

1(a) Given a matrix [N][M], print data row wise

Ex: N = 3, M = 4

	0	1	2	3
0	3	6	2	9
1	8	4	12	15
2	11	16	17	19

output:

3 6 2 9 8 4 12 15 11 16 17 19

Pseudo Code

Mat[N][M] — N rows, M columns

{0, M-1}

```
for( i = 0; i < N; i++){
    for( j=0; j<M; j++){
        print(mat[i][j])
    }
}
```

2(a) Given a matrix [N][M], Get max column wise sum

Ex: N = 3, M = 4

	0	1	2	3
0	3	6	2	9
1	8	4	12	15
2	11	16	17	19
	22	26	41	33

{Step}

-2	-3	-4	-6
-3	-4	-6	-9
-2	-3	-1	0
-7	-10	-11	-15

Input: Say we want to [j] col sum?

ms = INT_MIN | Integer.MIN_VAL

```
j = 0; j < M; j++
    s = 0; getting j^th col sum
    i = 0; i < N; i++
        s = s + mat[i][j]
    // s contain j^th col sum
    if(s > ms){ ms = s }
return ms
```

Constant — Max Integer Value which we can store in Integer

→ Interview Q: Time Complexity Section

TC → Count Iterations

→ for

→ Nested loops → Iterations

3(a) Given N*M where every row & column are sorted in increasing order, Search if k is present in given matrix?

Ex:

	0	1	2	3	4	5
0	1	2	4	5	9	11
1	1	4	7	8	10	14
2	3	7	9	10	1	18
3	6	10	12	14	16	20
4	9	13	16	19	22	24
5	11	15	19	21	24	27
6	14	20	25	29	31	37

k = 15, k = 28

11×28 Solution:

i) Search in Matrix

28×29 TC: O(N*M) SC: O(1)

2) If a array is sorted, we can apply BS

i) Apply BS on every row

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

2) Apply BS on every column

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

Note: BS on N Elements TC: log N

Idea:

If mat[i][j] > k

// skip column

// goto left

else if mat[i][j] < k

// skip row

// go down

Breaks

1) If you found

2) If we go out of matrix we break

3)

{Idea: At every row we either skip a row or skip a column}

Top-right(P3) → Todo(P3)

row & last column: 0, M-1

(i, j) = (0, M-1)

```
while( i<N && j>=0 )
    if( mat[i][j] == k ){
        return True
    }
    else if( mat[i][j] > k ){
        // goto left
        j = j-1
    }
    else{
        // goto down
        i++;
    }
return false
```

TC:

j = M-1 → M-2 → M-3 → 0 → -1

i = N, j = 0 → Stop

(i,j) > k → j > k

TC: There are N rows & M Column

At every Iteration we either skip a row or we skip a column

{N+M} TC: O(N+M) SC: O(1)

4(a) Given N*M, where every element is either {0/1} & every row is sorted in {increasing order} {find max no. of 1's in a row}

Ex:

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

j = 0 | j = j+1

[a, b] : b - a + 1

Ex2:

	0	1	2	3	4	5
0	0	0	0	0	1	1
1	0	0	0	0	0	1
2	0	0	0	1	1	1
3	0	0	0	1	1	1
4	0	0	0	1	1	1
5	0	0	0	1	1	1
6	0	0	0	1	1	1

j = 1 || j = j+1

Solution:

1) For every row get sum

find max row sum

2)

Input: If we can get 1st col which contain 1, are we done?

ans = M - {1^st col number which contain 1's}

[n^th col, M-1] = M-1 - n + 1 = M - n

Pseudocode, to get 1st col number which contains 1

i = 0, j = M-1, Put c = 0

```
while( i<N && j>=0 ){
    if( mat[i][j] == 1){
        // at j^th we have 1, skip j^th col
        {j--}, c++
    }
    else{
        {i++}
    }
}
{j = j+1} // j 1^st col with 1
Final ans: M - j, return c
```

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

i = N, j = 2 → out of bounds

i = N or j = -1 → out of bounds

{10:50 PM}

With c variable

j = -1 | j = j+1 | j = 0

M - j | 4 - 0 | = 4

Ag

→ Courses: low

→ BF / Not opt

Saturday: 30 min {Sunday}

{2 hrs: 30 Doubts}

{Friday | Saturday End | Monday}

Sahil: 30 min

2 min

k = 29

TC: N*M

5(a) Given a N*N Matrix print data spirally {Google / Amazon}

Ex1:

len = 3

(0,0) 3H

(1,1)

3V

1H

(0,0) → (0,1) → (0,2) → (0,3) → (0,4)

Ex2:

N = 5

(0,0) len = 4

(0,0) – (0,3)	H
(0,4) – (3,4)	V
(4,4) – (4,1)	H
(4,0) – (1,0)	V

(1,1) len = 2

(1,1) – (1,2)	H
(1,3) – (3,3)	V
(3,3) – (3,2)	H
(3,1) – (2,1)	V

print(2,2) len = 0

{print center element}

Ex3:

center (0,0), len = N-1 = 5

(0,0) – (0,4)	H
(0,5) – (4,5)	V
(5,5) – (5,1)	H
(5,0) – (1,0)	V

len = len - 2

(1,1) len = 3

(1,1) – (1,3)	
(1,4) – (3,4)	
(4,4) – (4,2)	
(4,1) – (2,1)	

l = l - 2

(2,2) len = 1

(2,2)
(2,3)
(3,3)
(3,2)

l = l - 2

len = -1

Intuition

Pattern: → loops

{i = 0, j = 0}

Break

Mat[N][N] // print it in spiral

(c = 0) {l = N-1}

```
while( l>0 ){
    (i = c, j = c) } Initializing Centre
    Step1: → l: length we need to iterate
    k = 0; k < l; k++ {
        print(mat[i][j]
        j++;
    }
    Step2: l:
    k = 0; k < l; k++){
        print(mat[i][j]
        i++;
    }
    Step3: l:
    k = 0; k < l; k++){
        print(mat[i][j]
        j--;
    }
    Step4: l
    k = 0; k < l; k++){
        print(mat[i][j]
        i--;
    }
    {l = l - 2  c++}
    update l  update center
}
if( l == 0 ){ print(mat[c][c]) }
```

{TC: O(N*N) || SC: O(1) ||}