SCALER

2D Matrices

TABLE OF CONTENTS

Which algorithm would you choose as most efficient to find the maximum subarray sum efficiently?

[ ] Brute Force algorithm
[ ] Kadane's Algorithm
[ ] Dijkstra's Algorithm
[ ] Moore's Voting algorithm

In the context of finding the maximum subarray sum, what does the 'carry forward' technique imply?

[ ] Using a stack to store elements

[ ] Using recursion for sum calculation

[ ] Continuously adding elements while maintaining the maximum sum

[ ] Using two pointers for sum calculation

How can the time complexity be optimized for performing multiple increment operation queries from index i to last index in an array?

[ ] by using stack
[ ] by using prefix sums
[ ] by using recursion
[ ] by using a queue

What is the optimized time complexity for handling multiple range queries using the prefix sum technique?

[ ] O(Q + N)
[ ] O(Q * N)
[ ] O(Q^2^)
[ ] O(N^2^)

In the problem of rainwater trapping, what does the water stored over a building depend on?

[ ] the width of the building
[ ] the height of the building
[ ] the minimum of the maximum heights on either side of the building
[ ] the total number of buildings

water = Min (l_{max} , r_{max}) - h[i]

Search element “k” in row-wise & column-wise sorted 2D array

$$\begin{bmatrix} -5 & -2 & 1 & 13 \\ -4 & 0 & 3 & 14 \\ -3 & 2 & 5 & 18 \\ 2 & 6 & 10 & 20 \end{bmatrix}_{N * M}$$

Search (6) → True

Search (15) → False

Search (0) → True

Quiz : 1

BF Idea

Iterate on all elements & check if current element = k.

T.C : O(N*M)

S.C : O(1)

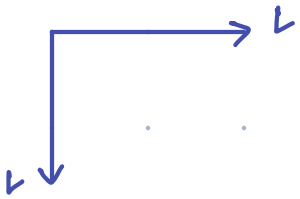
Optimisation Approach

Observation Matrix is Row-wise & Col-wise sorted.

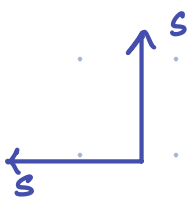
	0	1	2	3
0	-5	-2	1	13
1	-4	0	3	14
2	-3	2	5	18
3	2	6	10	20

Search (6)

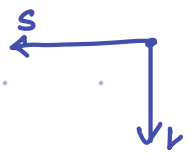
Case I : Start from TOP LEFT ✗



Case II : Start from BOTTOM RIGHT. ✗



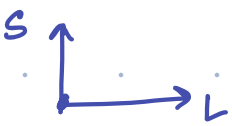
Case III : Start from TOP RIGHT



mat[i][j]

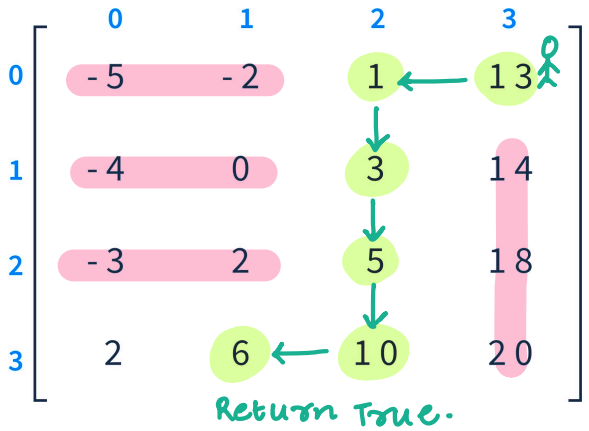
- < k : larger elements are in column. Discard ROW & search in column.
- == k : Return TRUE
- > k : Smaller elements are in Row. Discard col & search in Row.

Case IV : Start from BOTTOM LEFT

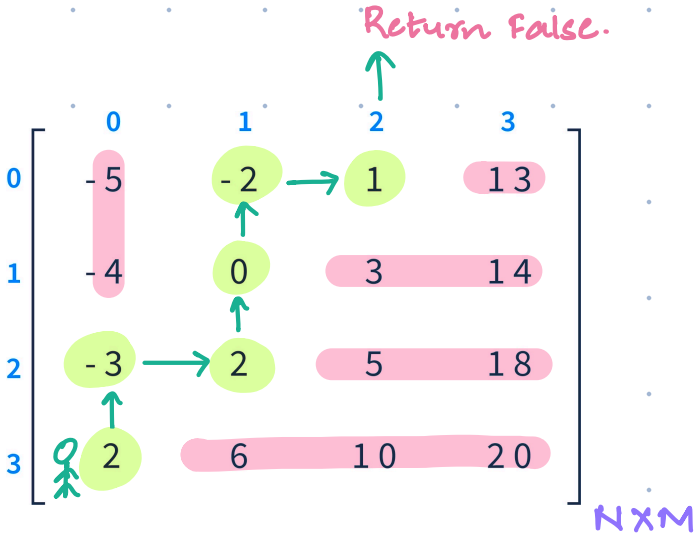


mat[i][j]

- < k : Larger elements are in Row. Discard col & search in Row.
- == k : Return TRUE
- > k : Smaller elements are in column. Discard ROW & search in column.



Search (6)



Search (-1)

< / > Code

```
// Starting from TOP RIGHT
i = 0 , j = M-1 ;
while ( i < n && j >= 0) {
    if (mat[i][j] == k)  return True ;

    else if (mat[i][j] < k)   // Move in downward direction
        i++ ;

    else                      // Move in leftward direction
        j-- ;
}
return False ;
```

T.C : O(N+M)
S.C : O(1)

Print Boundary Elements

Given a matrix of N * N i.e. Mat [N][N], print boundary elements in clockwise direction.

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

N = 5

4 ele →

4 ele ↓

4 ele ←

4 ele ↑

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

Quiz : 3

1	2	3
4	5	6
7	8	9

N=3

O/p : 1 , 2 , 3 , 6 , 9 , 8 , 7 , 4

2 ele →

2 ele ↓

2 ele ←

2 ele ↑

Approach

N * N

(N-1) ele →

(N-1) ele ↓

(N-1) ele ←

(N-1) ele ↑

N * M

(M-1) ele →

(N-1) ele ↓

(M-1) ele ←

(N-1) ele ↑

T.C : O(N+M)

S.C : O(1)

< / > Code

```
Void print Boundary( arr[N][N]) {
    i = 0 , j = 0 ;
    || (N-1) ele ⟶
    for( K = 1 ; K < N ; K++) {
        print (arr[i][j]) ;
        j++ ;
    }
    || i = 0 , j = (N-1)   print (N-1) ele ↓
    for( K = 1 ; K < N ; K++) {
        print (arr[i][j]) ;
        i++ ;
    }
    || i = (N-1) , j = (N-1)   print (N-1) ele ⟵
    for( K = 1 ; K < N ; K++) {
        print (arr[i][j]) ;
        j-- ;
    }
    || i = (N-1) , j = 0   print (N-1) ele ↑
    for( K = 1 ; K < N ; K++) {
        print (arr[i][j]) ;
        i-- ;
    }
}
```

T.C: O(4 * (N-1))
= O(N)

S.C: O(1)

Quiz : 3

1	2	3
4	5	6
7	8	9

N = 3

i =	j =	k =
0	0	1
1	1	2
2	2	3
1	1	1
0	0	2
		3
		1
		2
		3
		1
		2
		3

Spiral Matrix (clockwise)

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

i	j	N	(N-1)
0 (+1)	0 (+1)	6 (N-2)	5
1	1	4 (N-2)	3
2	2	2	1
3	3	0 ↓ STOP	

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]

	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

i	j	N	(N-1)
0	0	5	4
1	1	3	2
2	2	1 ↓ If (N==1) Print arr[i][j]	0

o/p: 1 2 3 4 5 10 15 20 25 24 23 22 21 16 11 6 7 8 9

14 19 18 17 12 13.

Quiz : 4

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

o/p :

13 14 12 8 7 0 11 6 5 10 0 9

1 2 3 4

Idea Print boundary elements for matrices of decreasing size. (size dec by N-2)

< / > Code

```
i = 0 , j = 0 ;
while( N > 1) {
    // (N-1) ele ⟶
    for(K = 1 ; K < N ; K++){
        print(arr[i][j]);
        j++;
    }
    // i = 0, j = (N-1)  print (N-1) ele
    for(K = 1 ; K < N ; K++){
        print(arr[i][j]);
        i++;
    }
    // i = (N-1) , j = (N-1)  print (N-1) ele ⟵
    for(K = 1 ; K < N ; K++){
        print(arr[i][j]);
        j--;
    }
    // i = (N-1) , j = 0  print (N-1) ele ↑
    for(K = 1 ; K < N ; K++){
        print(arr[i][j]);
        i--;
    }
    // i = 0 , j = 0
    i++, j++, N = N-2;
}
if(N == 1)
    print(arr[i][j]);
```

T.C : $O(N^2)$
S.C : $O(1)$

Sub - Matrix

Contiguous part of a matrix.

- Single cell is a sub matrix.
- Entire matrix is also a s sub matrix.

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

How can we uniquely identify a rectangle?

- TL & BR
- TR & BL

Sum of all Sub-matrices Sum

Given mat[N] [M]. Find sum of all sub-matrix sums.

	0	1	2
0	4	9	6
1	5	-1	2

4 [4] 13 [4 9] 19 [4 9 6] 9 [4; 5] 17 [4 9; 5 -1] 25 [4 9 6; 5 -1 2]

9 [9] 15 [9 6] 8 [9; -1] 16 [9 6; -1 2] 6 [6] 8 [6; 2]

5 [5] 4 [5 -1] 6 [5 -1 2] -1 [-1] 1 [-1 2] 2 [2]

Sum = 166

BF Idea

Generate all submatrices, find their sum & calculate total sum.

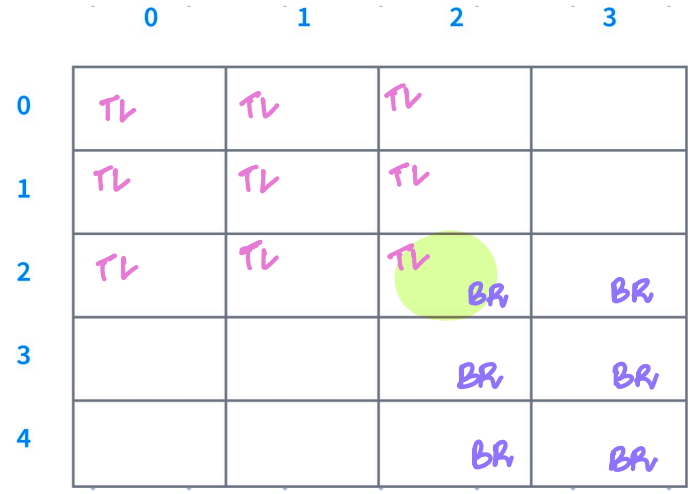
T.C : $O(N^3M^3)$ S.C : $O(1)$

HW

Total NO of Submatrices for matric of size N*M. = $\frac{NM(N+1)(M+1)}{4}$

Idea Contribution Technique

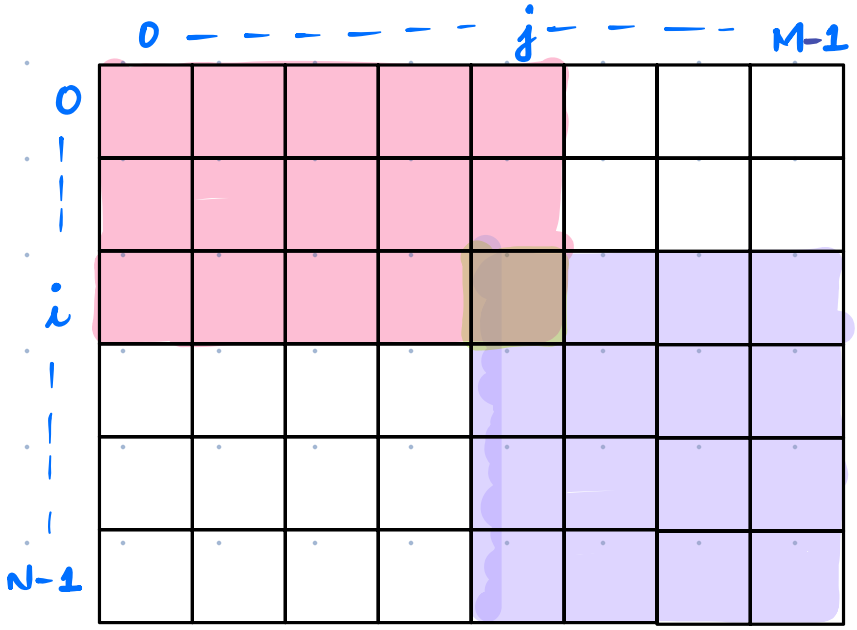
Quiz : 5



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

Options for TL * Options for BR = 9 * 6 = 54

- In how many sub - matrices (i , j) will be = (i+1) * (j+1) * (N-i) * (M-j)



Options for TL :
[0....i] [0....j]
(i+1) (j+1)

Options for BR :
[i....N-1] [j...M-1]
(N-1-i+1) (M-1-j+1)
= (N-i) * (M-j)

< / > Code

```
ans = 0;
for (i=0; i<N; i++) {
    for (j=0; j<M; j++) {
        top_left = (i+1) * (j+1);
        bottom_right = (N-i) * (M-j);
        contribution = arr[i][j] * top_left * bottom_right;
        ans += contribution;
    }
}
return ans;
```

T.C : O(N*M)
S.C : O(1)