Problem Solving 4

→ 3 Matrix
→ Max Value Counting Algorithm
→ 3 pair
→ Min diff

→

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

Given a matrix N*N & Queries

In every query 2 cells are given (i, j) (k, l) } for k ≤ i ≤ l

{Top left of a submatrix | Bottom right of a submatrix}

For every query find sum of all element in submatrix

Sol:

	ans
(2,2) - (4,5) :	27
(3,3) - (5,5) :	27

BF

Iterate on entire submatrix & get sum

```
a = 1; a < k; a++{
    b = j; b < l; b++{
        sum = sum + mat[a][b]
    }
}
print(sum)
```

→ mat

→ cols

TC: O(N⁴n)

For Q Queries: O(Q * (N⁴n))

In arrays, Given a subarray query, we used to find sum of subarray

→ Arrays: By constructing prefix array.

→ pf[i]: Subarray sum starting at 0 ending at i

In Matrix, Given a submatrix query, we used to find sum of submatrix

→ Matrix: By constructing prefix Matrix

→ pf[i][j]: Submatrix sum starting at [0,0] ending at (i,j)

Eg

	0	1	2	3
0				
1				
2				
3				
4				

pf[2][3]

How to construct pf[][] →

	0	1	2	3
0	a_1	a_2	a_3	a_4
1	b_1	b_2	b_3	b_4
2	c_1	c_2	c_3	c_4

Row wise prefix sum

a_1	a_1+a_2	$a_1+a_2+a_3$	$a_1+a_2+a_3+a_4$
b_1	b_1+b_2	$b_1+b_2+b_3$	$b_1+b_2+b_3+b_4$
c_1	c_1+c_2	$c_1+c_2+c_3$	$c_1+c_2+c_3+c_4$

Column wise prefix sum (prefix Matrix)

	0	1	2	3
0	a_1	a_1+a_2	$a_1+a_2+a_3$	$a_1+a_2+a_3+a_4$
1	b_1	b_1+b_2	$b_1+b_2+b_3$	$b_1+b_2+b_3+b_4$
2	c_1	c_1+c_2	$c_1+c_2+c_3$	$c_1+c_2+c_3+c_4$

To construct prefix Matrix → O(N²n) In General: N*(Pn)

N*(Pn)

Part-2 → How to answer queries prefix matrix

pf(A) (0,0) (1,1) (2,2) (3,4)

pf[i,j] = pf[i,j-1] - pf[i-1,j] + pf[i-1,j-1]

pf[4,4] - pf[1,4] - pf[4,1] + pf[0,1]

(k, j-1)

(i, j-1)

(i, l) → pf[i,l] - pf[i,j-1] - pf[k-1,l] + pf[k-1,j-1]

j > 0 i > 0 k > 0 j > 0 && j > 0

If i = 0 || j = 0 // with in the bounds

Observe → [i, j] — [k, l]

sum = pf[k, l]

if(j != 0){
    sum = sum - pf[k, j-1]
}

if(i != 0){
    sum = sum - pf[i-1, l]
}

if(i != 0 && j != 0){
    sum = sum + pf[i-1][j-1]
}

→ Doubt

→ Can Trace the code

→ pf[]

→ Query

→ To answer a Q = O(1)

i.e. need pf[][] = O(N*n)

250 Given a Matrix, find the maximum sum of submatrix within the matrix.

{Find Max Submatrix Sum} → A single cell is a submatrix

Ex 5, 6

	0	1	2	3	4	5
0	3	-1	-8	-4	-1	2
1	-10	3	2	-1	5	1
2	-4	-1	6	4	7	10
3	-1	4	8	6	10	12
4	3	6	14	16	20	23

Ans = 23

{Bottom right cell}

Sol:

1) Compute pf[][] → O(N²n)

2) For every submatrix get sum

O(N²n²)

overall TC = O(N²n²n)

Solution

→ For every submatrix get the sum & calculate overall max

→ For the submatrix, [Top left & Bottom right]

i = 0; i < N; i++{
    j = 0; j < n; j++{
        (i, j) is top left of a submatrix
        k = i; k < N; k++{
            l = j; l < n; l++{
                (k, l) → is the a submatrix
                // using prefix Matrix get submatrix sum query
                // can find sum get max
            }
        }
    }
}

Any cell can be top left of a submatrix

Sol:

1) For every Matrix get sum

[N-1][n-1]

Top left

any where

TC: O(N²n) + O(N²n)

prefix Matrix Top left

ans = INT_MIN

i = 0; i < N; i++{
    j = 0; j < n; j++{
        [i, j] = (N-1)[n-1]
        // use prefix matrix to
        // can get sum
        ans = Max(ans, query(i, j, N-1, n-1))
    }
}

finding Top left

250

Given 3 sorted arrays A[], B[], C[], find i, j, k such that {max(A[i], B[j], C[k]) - min(A[i], B[j], C[k])}

minimum of

	0	1	2	3	4	5	6
N1 A[] :	3	14	16	20	29	40	55
N2 B[] :	-15	15	26	27	31	39	42
N3 C[] :	-15	23	29	30	35	50	

i = 0; j = 0; k = 0

3 -15 -15

Diff with difference

i	j	k	A[i]	B[j]	C[k]	Diff
0	0	0	3	-6	-15	18
0	0	1	3	-6	15	21
0	1	1	3	23	15	20
1	1	1	14	23	15	9
2	1	1	16	23	15	8
2	1	2	16	23	26	10
3	1	2	20	23	26	6
4	1	2	29	23	26	6
4	2	2	29	29	26	5
4	3	2	29	30	26	4
4	3	3	29	30	29	1

We move, as Potential with min value because we cannot get a better diff with that

i	j	k	A[i]	B[j]	C[k]	max	min	max - min
2	1	0	16	23	-15	23	-15	38
0	0	0	3	-6	-15	3	-15	18
6	5	6	55	50	42	55	42	13
5	4	5	40	35	29	40	35	5
4	2	2	29	29	26	29	29	5
4	3	3	29	30	29	30	29	1

Sol:

Check all triplets & get min value

ans = INT_MAX

i = 0; i < N1; i++{
    j = 0; j < N2; j++{
        k = 0; k < N3; k++{
            ans = min(max(A[i], B[j], C[k]) - min(A[i], B[j], C[k]), ans)
        }
    }
}

TC = O(N1*N2*N3)

Sol2: ans = INT_MAX

i = 0, j = 0, k = 0

while (i < N1 && j < N2 && k < N3){
    val = max3(A[i], B[j], C[k]) - min3(A[i], B[j], C[k])
    ans = min(ans, val)
    int x = min3(A[i], B[j], C[k])
    if (x == A[i]){ i++; }
    else if (x == B[j]){ j++; }
    else { k++; }
}

At every step either we move i, j, k

TC: {N1 + N2 + N3}

A - A[] : N1
B - B[] : N2
C - C[] : N3

→

N1 A[] : 3 14 16 20 29 40 55

N2 B[] : 16 17 18

C[] : 16 20 21

	3	16	16
MAX	3	—	—
	3	—	—
	3	—	—
	3	—	—

Min: 3

Max: >16

ans = 13

ans = 16

ans = 13

Sum Difference → Continuous of every element → Next Step

Next Step

→ Subsequence 3 →

	0	1	2	3	4
ar[5] :	3	-2	6	4	8

Construct Array : B < C

A < B < C

3 -2 8

Minimum - pass from element

50 - ∞ = ∞

Array sorted :

[B < C