Largest Sub Rectangle Sum

(Max 2d Range Sum)

By

Ashap-ud-doula Bappy

Introduction

Sub rectangle – A rectangle inside another rectangle.

For example, consider the below **2*3** rectangle. How many sub rectangles are there? Let's count, Shall we?

1,1	1,2	1,3
2,1	2,2	2,3

1. **(1,1)**
2. **(1,2)**
3. **(1,3)**
4. **(2,1)**
5. **(2,2)**
6. **(2,3)**
7. **(1,1) – (1,2)**
8. **(1,1) – (1,3)**
9. **(1,1) – (2,1)**
10. **(1,1) – (2,2)**
11. **(1,1) – (2,3)**
12. **(1,2) – (1,3)**
13. **(1,2) – (2,2)**
14. **(1,2) – (2,3)**
15. **(1,3) – (2,3)**
16. **(2,1) – (2,2)**
17. **(2,1) – (2,3)**
18. **(2,2) – (2,3)**

So, there are **18** sub rectangles.

Let us derive a formula for number of sub rectangles for a **N*M** rectangle.

If the grid is **1×1**, there is **1** rectangle.
If the grid is **2×1,** there will be **2 + 1 = 3** rectangles.
If it grid is **3×1**, there will be **3 + 2 + 1 = 6** rectangles.
we can say that for **N*1** there will be **N + (N-1) + (N-2) … + 1 = (N)(N+1)/2** rectangles.

If we add one more column to **N×1**, firstly we will have as many rectangles in the 2nd column as the first,
and then we have that same number of **2×M** rectangles.
So, **N×2 = 3 (N)(N+1)/2.**

After deducing this we can say,
For **N*M** we'll have **(M)(M+1)/2 * (N)(N+1)/2** = **M(M+1) * (N)(N+1)/4.**

So, the formula for total rectangles will be **M(M+1) * (N)(N+1)/4.**

Now, the main problem is

Given a rectangle of size **N * M**, we have to find the largest sub rectangle sum, that means a sub rectangle which has the largest sum. The sum will be counted by adding the value of the cells of that corresponding rectangle.

For example,

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

The above **4 × 4** rectangle has a **3 × 2** (Yellow shaded area) sub rectangle on the lower-left with maximum sum of

9 + 2 - 4 + 1 - 1 + 8 = 15.

An $O(n^2)$ approach to find the cumulative sum from cell (1, 1) to (i, j)

The solution for the **Max 1D Range Sum** can be extended to **two (or more)** dimensions as long as the **inclusion-exclusion principle** is properly applied. The only difference is that while we dealt with overlapping sub-ranges in Max 1D Range Sum, we will deal with overlapping sub-matrices in Max 2D Range Sum. We can turn the **N*M** input matrix into an **N*M cumulative sum matrix** where **A[i] [j]** no longer contains its own value, but the sum of all items within

sub-matrix (1, 1) to (i, j). This can be done simultaneously while reading the input and still runs in $O(n^2)$. The formula shown below turns the input matrix (Table 01) into a cumulative sum matrix (Table 02).

Table 01: Input matrix

	1	**2**	**3**	**4**
1	0	-2	-7	0
2	9	2	-6	2
3	-4	1	-4	1
4	-1	8	0	-2

Table 02: Cumulative sum matrix

	1	**2**	**3**	**4**
1	0	-2	-9	-9
2	9	9	-4	2
3	5	6	-11	-8
4	4	13	-4	-3

2,3

3.3

3,2

Let’s say we want to update the value of the cell **(3, 3)** so that it will contain the sum of all items within sub matrix **(1, 1)** to **(3, 3)**.

So, the formula is,

A (3, 3) = A (3, 3) + A (2,3)

A (3, 3) = A (3, 3) + A (3,2)

A (3, 3) = A (3, 3) - A (2,2)

Look carefully, we are adding the value from the immediate **top cell** and immediate **left cell** and subtracting from the immediate **upper left diagonal cell** of the cell **(3, 3)**.

The reason behind that, the top cell which is **(2,3)** is containing the cumulative sum of sub matrix **(1, 1)** to **(2, 3)** and the left cell which is **(3, 2)** is containing the cumulative sum of sub matrix **(1, 1)** to **(3, 2)**. So, if we add those value with the cell **(3, 3)** then we will get the cumulative sum of sub matrix **(1, 1)** to **(3, 3).** That's it?

If that's all then why the hell subtraction part is needed?

Look carefully again in Table 02, we have added **(2, 2)** twice!!!!

One with the **top cell** and the other one with the **left cell**. Because, the cell (2, 2) lies both the sub matrices which are **(2, 3)** and **(3, 2)**. That is why we need to subtract the double count of the cell (2, 2).

So, the general formula is,

A (i, j) = A (i, j) + A (i-1, j) // add from top

A (i, j) = A (i, j) + A (i, j-1) // add from left

A (i, j) = A (i, j) - A (i-1, j-1) // avoid double count (inclusion-exclusion principle)

> **N.B.**
> Inclusion-exclusion principle,
> n (A U B) = n(A) + n(B) – n (A ∩ B)

An O(n^2) approach to find the cumulative sum from cell (i, j) to (k, l)

Now, how to find the sum of sub-matrix (i, j) to (k, l) in O(1) with the help of the previously calculated cumulative sum matrix?

```
subRect = A[k][l] // sum of all items from (0, 0) to (k, l)
if (i > 0) subRect -= A[i - 1][l]
if (j > 0) subRect -= A[k][j - 1]
if (i > 0 && j > 0) subRect += A[i - 1][j - 1]
```

At first, we will take the whole rectangle from (0, 0) to (k, l) then we will subtract all the unnecessary rectangles.

An illustration is given below,

The green shaded area is the unnecessary part.

	0	1	2	3	4
0	i-1 , j-1			i-1 , l	
1		i , j			
2					
3	k , j-1			k , l	
4					

Programming exercises

1. UVa – 00108
2. UVa – 10827
3. UVa – 11951