More on Prefix Sums

https://usaco.guide/silver/prefix-sums-2

CP Initiative
joincpi.org

Introduction

Similarly to 1D prefix sums, let's say we had a 2D array, `arr`, of length `N` and width `M`. We want to answer `Q` queries asking to compute the sum of values in a subrectangle.

3	1	5	4
2	6	2	7
1	1	5	2
9	8	3	6

Example

Using the previously given array, consider the example:

3	1	5	4
2	6	2	7
1	1	5	2
9	8	3	6

The sub-rectangle sum would be 6 + 2 + 1 + 5 = 14.

Naive Solution

We can use a nested for loop to iterate through the cells in the subrectangle.

```
int sum = 0;

for (int i = l1; i <= r1; i++) {

    for (int j = l2; j <= r2; j++) {

        sum += arr[i][j];

    }

}
```

If we have `Q` queries, and each query takes up to `N * M` operations to calculate the sum, the time complexity would be `O(N * M * Q)`.

Faster Solution (Prefix Sums)

Let's create a zero-indexed 2D array, `pre`, that stores the sum of values in the sub-rectangle in the upper left corner.

3	1	5	4
2	6	2	7
1	1	5	2
9	8	3	6

For example, the value at `(2,2)` would equal the sum of highlighted values.

Faster Solution (Prefix Sums)

For the given array, the prefix sum array would look like this:

3	4	9	13
5	12	19	30
6	14	26	39
15	31	46	65

Creating the Prefix Sums

The value of `pre[i][j]` equals:

```
pre[i - 1][j] + pre[i][j - 1] - pre[i - 1][j - 1] + arr[i][j]
```

To understand this, consider computing the prefix sum of `(2, 2)`.

3	1	5	4
2	6	2	7
1	1	5	2
9	8	3	6

Creating the Prefix Sums

We start by adding `pre[i - 1][j]`, which is `pre[1][2]`.

3	1	5	4
2	6	2	7
1	1	5	2
9	8	3	6

Creating the Prefix Sums

Then we add `pre[i][j - 1]`, which is `pre[2][1]`.

3	1	5	4
2	6	2	7
1	1	5	2
9	8	3	6

Creating the Prefix Sums

Then we subtract `pre[i - 1][j - 1]`, which is `pre[1][1]`.

3	1	5	4
2	6	2	7
1	1	5	2
9	8	3	6

Creating the Prefix Sums

Finally, we add `arr[i][j]`.

3	1	5	4
2	6	2	7
1	1	5	2
9	8	3	6

Calculating the Sum

Now that we know how to compute the prefix sum array, we can use it to answer queries.

Let's say the query wants to compute the sum of values `arr[i][j]` in subrectangle with:

$$l1 \leq i \leq r1, l2 \leq j \leq r2$$

Calculating the Sum

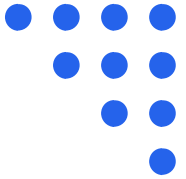
To calculate the sum, we compute:

```
pre[l2][r2] - pre[l1-1][r2] - pre[l2][r1-1] + pre[l1-1][r1-1]
```

Let's compute the sum of the sub-rectangle with upper left corner `(1, 1)` and bottom right corner `(2,2)`.

3	1	5	4
2	6	2	7
1	1	5	2
9	8	3	6

Calculating the Sum



First, we add `pre[l2][r2]`, which is `pre[2][2]`.

3	1	5	4
2	6	2	7
1	1	5	2
9	8	3	6

Calculating the Sum

Then, we subtract `pre[l1 - 1][r2]`, which is `pre[0][2]`.

3	1	5	4
2	6	2	7
1	1	5	2
9	8	3	6

Calculating the Sum

Then, we subtract `pre[l2][r1 - 1]`, which is `pre[2][0]`.

3	1	5	4
2	6	2	7
1	1	5	2
9	8	3	6

Calculating the Sum

Then, we add `pre[l1 - 1][r1 - 1]`, which is `pre[0][0]`.

3	1	5	4
2	6	2	7
1	1	5	2
9	8	3	6

Implementation

We need to be cautious of off-by-one errors, as intervals can be inclusive, exclusive, 1-indexed, etc.

Here is a code snippet from the problem Forest Queries which computes 2D prefix sums. Notice that it uses 1-indexing to avoid out of bounds errors.

```
for (int i = 1; i <= N; i++) {
    for (int j = 1; j <= N; j++) {
        pfx[i][j] = arr[i][j] + pfx[i-1][j] + pfx[i][j-1] - pfx[i-1][j-1];
    }
}

for (int i = 0; i < Q; i++) {
    int x1, x2, y1, y2;
    cin >> x1 >> y1 >> x2 >> y2;
    cout << pfx[x2][y2] - pfx[x1-1][y2] - pfx[x2][y1-1] + pfx[x1-1][y1-1] << "\n";
}
```

Example Problem

[USACO Silver - Painting the Barn]

Challenge Problem

[USACO Gold - Painting the Barn]