Now, when we want to query for the sum of the elements of `arr` between (1-indexed) indices a and b inclusive, we can use the following formula:

$$\sum_{i=a}^{b} \texttt{arr[i]} = \sum_{i=1}^{b} \texttt{arr[i]} - \sum_{i=1}^{a-1} \texttt{arr[i]}$$

Using our definition of the elements in the prefix sum array, we have

$$\sum_{i=a}^{b} \texttt{arr[i]} = \texttt{prefix[b]} - \texttt{prefix[a-1]}$$

Since we are only querying two elements in the prefix sum array, we can calculate subarray sums in $O(1)$ per query, which is much better than the $O(N)$ per query that we had before. Now, after an $O(N)$ preprocessing to calculate the prefix sum array, each of the Q queries takes $O(1)$ time. Thus, our total time complexity is $O(N+Q)$, which should now pass the time limit.

Let's do an example query and find the subarray sum between indices $a = 2$ and $b = 5$, inclusive, in the 1-indexed *arr*. From looking at the original array, we see that this is $\sum_{i=2}^{5} \texttt{arr[i]} = 6 + 4 + 2 + 5 = 17$.

Index i	0	1	2	3	4	5	6
`arr[i]`	0	1	6	4	2	5	3

Using prefix sums: Using prefix sums: $\texttt{prefix[5]} - \texttt{prefix[1]} = 18 - 1 = 17$.

Index i	0	1	2	3	4	5	6
`prefix[i]`	0	1	7	11	13	18	21

11.2 Two Dimensional Prefix Sums

Now, what if we wanted to process Q queries for the sum over a subrectangle of a N rows by M columns matrix in two dimensions? Let's assume both rows and columns are 1-indexed, and we use the following matrix as an example:

0	0	0	0	0	0
0	1	5	6	11	8
0	1	7	11	9	4
0	4	6	1	3	2
0	7	5	4	2	3

Naively, each sum query would then take $O(NM)$ time, for a total of $O(QNM)$. This is too slow.

Let's take the following example region, which we want to sum:

0	0	0	0	0	0
0	1	5	6	11	8
0	1	7	11	9	4
0	4	6	1	3	2
0	7	5	4	2	3

Manually summing all the cells, we have a submatrix sum of $7 + 11 + 9 + 6 + 1 + 3 = 37$.

The first logical optimization would be to do one-dimensional prefix sums of each row. Then, we'd have the following row-prefix sum matrix. The desired subarray sum of each row in our desired region is simply the green cell minus the red cell in that respective row. We do this for each row, to get $(28 - 1) + (14 - 4) = 37$.

0	0	0	0	0	0
0	1	6	12	23	31
0	1	8	19	28	32
0	4	10	11	14	16
0	7	12	16	18	21

Now, if we wanted to find a submatrix sum, we could break up the submatrix into a subarray for each row, and then add their sums, which would be calculated using the prefix sums method described earlier. Since the matrix has N rows, the time complexity of this is $O(QN)$. This is better, but still usually not fast enough.

To do better, we can do two-dimensional prefix sums. In our two dimensional prefix sum array, we have

$$\texttt{prefix[a][b]} = \sum_{i=1}^{a} \sum_{j=1}^{b} \texttt{arr[i][j]}$$

This can be calculated as follows for row index $1 \le i \le n$ and column index $1 \le j \le m$:

$$\texttt{prefix[i][j]} = \texttt{prefix[i-1][j]} + \texttt{prefix[i][j-1]} - \texttt{prefix[i-1][j-1]} + \texttt{arr[i][j]}$$

The submatrix sum between rows a and A and columns b and B, can thus be expressed as follows:

$$\sum_{i=a}^{A} \sum_{j=b}^{B} \texttt{arr[i][j]} = \texttt{prefix[A][B]} - \texttt{prefix[a-1][B]} - \texttt{prefix[A][b-1]} + \texttt{prefix[a-1][b-1]}$$

Summing the blue region from above using the 2d prefix sums method, we add the value of the green square, subtract the values of the red squares, and then add the value of the gray square. In this example, we have $65 - 23 - 6 + 1 = 37$, as expected.

0	0	0	0	0	0
0	1	6	12	23	31
0	2	14	31	51	63
0	6	24	42	65	79
0	13	36	58	83	100

Since no matter the size of the submatrix we are summing, we only need to access 4 values of the 2d prefix sum array, this runs in $O(1)$ per query after an $O(NM)$ preprocessing. This is fast enough.

11.3 Problems

1. USACO December 2015 Silver Problem 3: Breed Counting
   http://usaco.org/index.php?page=viewproblem2&cpid=572
2. USACO January 2016 Silver Problem 2: Subsequences Summing to Sevens
   http://usaco.org/index.php?page=viewproblem2&cpid=595
3. USACO December 2017 Silver Problem 1: My Cow Ate My Homework
   http://www.usaco.org/index.php?page=viewproblem2&cpid=762
4. USACO January 2017 Silver Problem 2: Hoof, Paper, Scissors
   http://www.usaco.org/index.php?page=viewproblem2&cpid=691
5. (2D Prefix Sums) USACO February 2019 Silver Problem 2: Painting the Barn
   http://www.usaco.org/index.php?page=viewproblem2&cpid=919