



Prefix Sums

<https://usaco.guide/silver/prefix-sums>



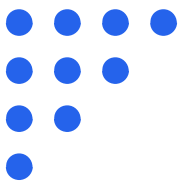
CP Initiative
joincpi.org

Introduction

Let's say we have a one-indexed integer array, arr , of length N and we want to compute the value of:

$$arr[1] + arr[1 + 1] \dots + arr[r]$$

for Q pairs (l, r) satisfying $1 \leq l \leq r \leq N$.

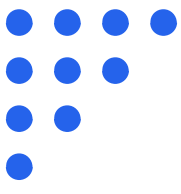


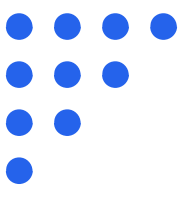
Example

For $N = 6$, consider the example:

index	1	2	3	4	5	6
$arr[i]$	1	6	4	2	5	3

Then, for the pair $(3, 5)$, the value would be $arr[3] + arr[4] + arr[5]$ which equals 11.





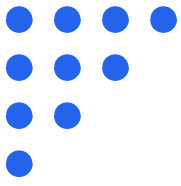
Naive Solution

Naively, we can iterate over the elements in the array from $l \dots r$ for each pair with a for loop.

```
int sum = 0;
for (int i = l; i <= r; i++) {
    sum += arr[i];
}
```

If we have Q queries, and each query takes up to N operations to calculate the

sum, the time complexity would be $O(N * Q)$.



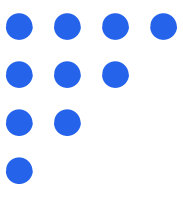
Faster Solution (Prefix Sums)

Let's create a zero-indexed array, `pre`, that stores the sum of some prefix of the array. Since `arr` is one-indexed, `pre[0] = 0`.

Then, $pre[i] = pre[i-1] + arr[i]$ for $i > 0$.

For our example, `pre` would look like this:

index	0	1	2	3	4	5	6
<code>pre[i]</code>	0	1	7	11	13	18	21
<code>arr[i]</code>		1	6	4	2	5	3



Calculating the Sum

Let $\text{sum}(a, b)$ represent the sum of elements in a between a and b , or 0 if $a > b$. Then,

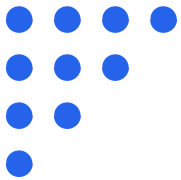
$$\text{sum}(l, r) = \text{sum}(1, r) - \text{sum}(1, l - 1)$$

By definition of the prefix sum array,

$$\text{sum}(1, i) = \text{pre}[i]$$

Which means,

$$\text{sum}(l, r) = \text{pre}[r] - \text{pre}[l - 1]$$



Example

Let's compute `sum(2, 5)` using prefix sums. This equals `arr[2] + arr[3] + arr[4] + arr[5]`.

index	1	2	3	4	5	6
arr[i]	1	6	4	2	5	3

Using prefix sums, this equals `pre[5] - pre[1]`.

index	0	1	2	3	4	5	6
pre[i]	0	1	7	11	13	18	21

Code

In C++, we can use `std::partial_sum`, although it doesn't shorten the code by much (you probably won't ever need use this).

```
#include <bits/stdc++.h>
using namespace std;

#define sz(x) (int)size(x)

using ll = long long;
using vl = vector<ll>;

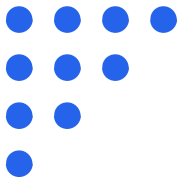
vl psum(const vl& a) {
    vl psum(sz(a)+1);
    for (int i = 0; i < sz(a); ++i)
        psum[i+1] = psum[i]+a[i];
    // or partial_sum(begin(a), end(a), begin(psum)+1);
    return psum;
}

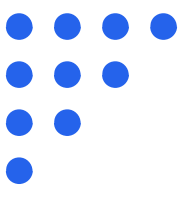
int main() {
    for (ll i: psum({1,2,3,4,5}))
        cout << i << " ";
    // 0 1 3 6 10 15
}
```



Example Problem

[USACO - Subsequences Summing to Sevens](#)





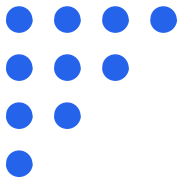
Solution Sketch

- Compute prefix sums
- We want $(pre[r] - pre[l - 1]) \bmod 7 = 0$
- This is equal to $(pre[r] \bmod 7) - (pre[l - 1] \bmod 7) = 0$
- Loop left to right, counting for each index i how many values of $pre[j]$ with $j < i$ have the same value $\bmod 7$ as the current prefix sum



Solution Code

[USACO Guide - Subsequences Summing to Sevens](#)



Challenge Problem

[USACO - Painting the Barn](#)

