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[l] + arr[l + 1] … + arr[r]
```

for `Q` pairs `(l, r)` satisfying `1 <= l <= r <= N`.

CP Initiative
joincpi.org

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

Calculating the Sum

Let `sum(a, b)` represent the sum of elements in `arr` between `a` and `b`, or 0 if `a > b`. Then,

```
sum(l, r) = sum(1, r) - sum(1, l - 1)
```

By definition of the prefix sum array,

```
sum(1, i) = pre[i]
```

Which means,

```
sum(l, r) = pre[r] - pre[l - 1]
```

CP Initiative
joincpi.org

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).

```cpp
#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
}
```

CP Initiative
joincpi.org

Example Problem

USACO - Subsequences Summing to Sevens

Solution Sketch

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

CP Initiative
joincpi.org

Solution Code

[USACO Guide - Subsequences Summing to Sevens]

CP Initiative
joincpi.org

Challenge Problem

USACO - Painting the Barn

CP Initiative
joincpi.org