Prefix Sums
Difference Arrays
Circular Arrays

- Srivaths P

Goals:

- Learn about 1D prefix sums.
- Learn about 2D prefix sums.
- Learn about difference arrays.
- Learn about various use cases of difference arrays.
- Learn about circular arrays and solve related problems.

Prefix Sums

A prefix sum stores the sum of the prefix of an array at each index. Takes O(N) time complexity to compute.

```
prefix[k] = sum of array from 0 to k
```

Prefix sums can be used to answer queries such as “Sum of elements of array from [L, R]” in O(1) time complexity

Implementation

- $O(N^2)$:

```
for (int i = 0; i < n; i++) {
    prefix_sum[i] = 0;
    for (int j = 0; j <= i; j++)
        prefix_sum[i] += a[i];
}
```

- $O(N)$:

```
prefix_sum[0] = a[0];
for (int i = 1; i < n; i++)
    prefix_sum[i] = prefix_sum[i-1] + a[i];
```

Sum of range in O(1)

We can write sum from [L, R] as
sum from [0, R] – sum from [0, L-1]

Which can be written as

```
prefix_sum[r] - prefix_sum[l-1]
```

Note: Pre-computation takes O(N)

2D Prefix Sums – Computation

2D prefix sums are similar to 1D prefix sums, but extended to two dimensions arrays/grids.

The index of (i, j) in the prefix sum will store the sum of the sub-grid [0…i][0…j] in the original grid.

Prefix sum of index (i, j) can be computed as

$$\left(P_{i-1,\ j} + P_{i,\ j-1} - P_{i-1,\ j-1}\right) + A_{i,\ j}$$

2D Prefix Sums – Range Sums

- To calculate the sum of the subarray [a…c][b…d]:
$$P_{c,d} - P_{c,b-1} - P_{a-1,d} + P_{a-1,b-1}$$

With $O(NM)$ precomputation, we can compute the sum of any submatrix in $O(1)$.

This can be used to answer range queries in $O(1)$ in two dimensional grids/arrays.

Difference Arrays

- You are given Q queries of the form $[L,\ R],\ K$ meaning you add the value K to the range $[L,\ R]$. The array is initially filled with zeros. What is the final array?

The brute-force solution would be the update the array for each query, and output the array at the end.

The time complexity of this approach is $O(QN)$.

Difference Arrays

- Difference arrays can update the array in $O(1)$, but the changes won't be visible until we perform a specific function on it.

We initially have an array D of size $N + 1$ filled with zeros.

To add K to range $[L, R]$, we just take $D_L \mathrel{+}= K$ and $D_{R+1} \mathrel{-}= K$.

At the end of the queries, we can take the prefix sum of D for the final array.

If our array had some values originally, it has to be added later.

Problems to Solve:

- https://cses.fi/problemset/task/1652
- https://www.codechef.com/ZCOPRAC/problems/ZCO22001
- Range product updates in $O(1)$ where $K \leq 1e9$ (will not overflow)
- https://www.hackerrank.com/challenges/crush/problem
- https://www.codechef.com/BYTR20B/problems/AGCY

Circular Arrays

- Circular arrays are arrays where the index after $N - 1$ is the index 0.

Say we traverse through this circular array X times. Then if we take $\mathrm{X} \equiv R \ (mod \ N)$, then R is the index we will end up at.

Traversing through a circular array would mean that we loop infinitely.

In a way, rotating a circular array would mean that the array does not change at all, only the indices change.

Circular Arrays

A trick to solve problems related to circular arrays is to double the array's length.

Example problem:

https://leetcode.com/problems/next-greater-element-ii/

https://codeforces.com/contest/1681/problem/B

Resources

- https://usaco.guide/silver/more-prefix-sums?lang=cpp
- https://codeforces.com/blog/entry/78762 (basic)
- https://codeforces.com/blog/entry/86420 (advanced)
- https://en.wikipedia.org/wiki/Modulo_operation (properties of mod)