Difference Array

Concepts & Qns

Video-
3

(M)otivation :-

Whatever people say about you,
don't let it effect you.
Stay silent, grind alone.
Let your success answer those
people...

MIK...

3356. Zero Array Transformation II

Medium | Topics | Companies | Hint

You are given an integer array `nums` of length `n` and a 2D array `queries` where `queries[i] = [l_i, r_i, val_i]`.

Each `queries[i]` represents the following action on `nums`:

- Decrement the value at each index in the range `[l_i, r_i]` in `nums` by **at most** `val_i`
- The amount by which each value is decremented can be chosen **independently** for each index.

A **Zero Array** is an array with all its elements equal to 0.

Return the **minimum** possible **non-negative** value of `k`, such that after processing the first `k` queries in **sequence**, `nums` becomes a **Zero Array**. If no such `k` exists, return -1.

Example :- nums = [2, 0, 2]

queries = [(0, 2, 1), (0, 2, 1), (1, 1, 3)]

output :- 2 [2, 0, 2] → [1, 0, 1] → [0, 0, 0] → Zero arr.

Thought Process

nums = [3, 7, 5, 6, 4, 2, 5, 3, 1]

queries = [(0,4,3), (1,4,4), (3,8,5), (4,7,4)], Q

i

Ans = 3

Brute Force:-

i = 0 (0,4,3) – [0, 4, 2, 3, 1, 2, 5, 3, 1]

i = 1 (1,4,4) – [0, 0, 0, 0, 0, 2, 5, 3, 1]

i = 2 (3,8,5) – [0, 0, 0, 0, 0, 0, 0, 0, 0]

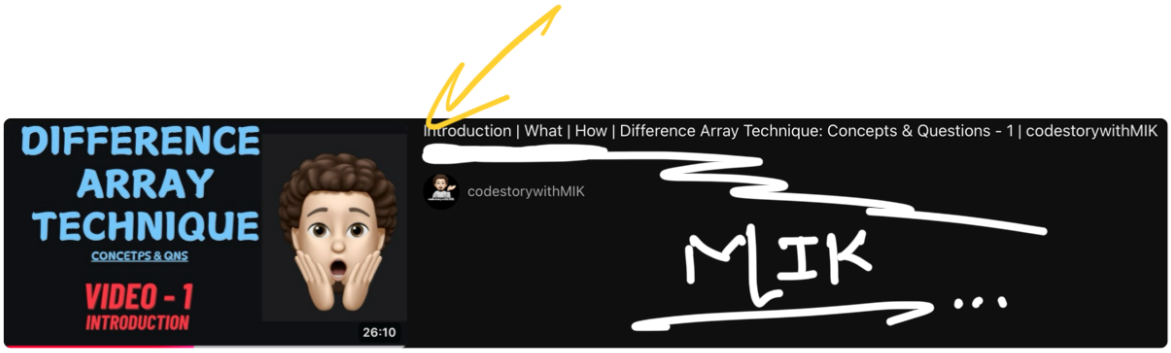
3 Queries

T.C = O(Q * n) Brute Force.

Range → Decrement

Range (+/-) → Segment Tree

→ Difference Array Tech..

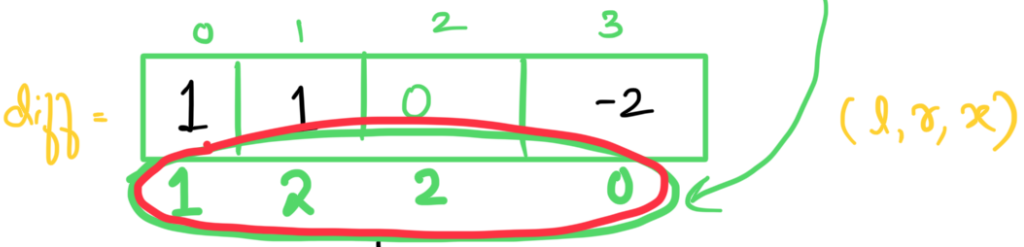


Difference array :-

length = n

queries = [(0, 2, 1), (1, 2, 2), (1, 2, -1)]

(1, 2, 2, 0)



- diff [l] += x
- diff [r+1] -= x → if (r+1 < n)

last step
- Cumsum diff Array.

Query → O(1)

Q → 1

CumSum → O(N)

Q + N

(Q + N)

nums = [2, 4, 5, 5], n = 4

Queries = [(0, 2, 1), (1, 2, 2), (1, 2, -1)]

diff = [1, 1, 0, -2], n = 4

CumSum =

Add / Dec. (Subtract).

nums = [2, 3, 4, 5], n = 4

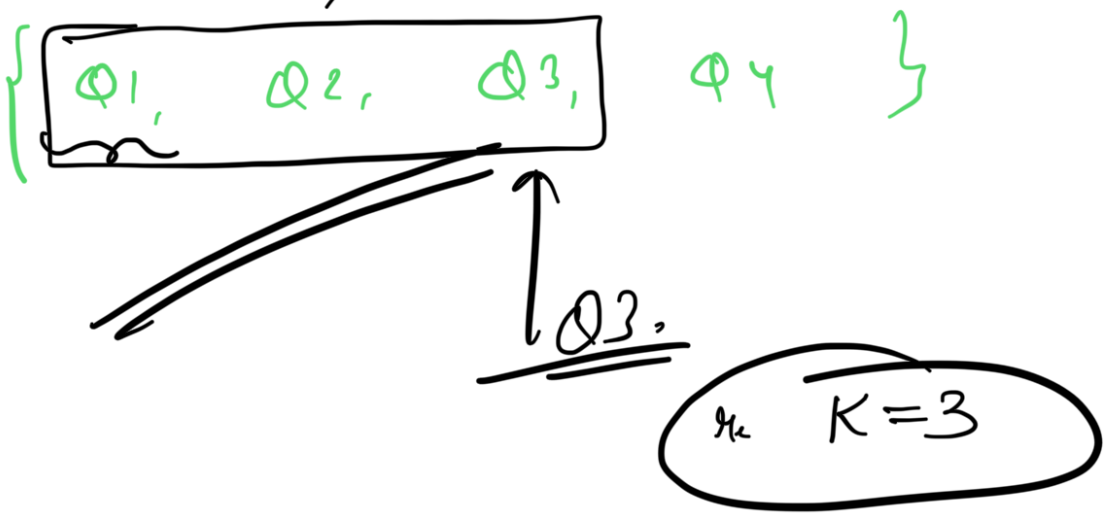
Queries = [~~~~~]

diff = [2, 1, 3, 5]

↳ Cum. Sum

- decreasing { nums[1] = 3
diff [1] = 1

if (nums[i] > diff[i]) {
ret false.
}



Ex K=3

{ (•) Queries = { 0 Q1, 1 Q2, 2 Q3, 3 Q4, ... }

(•) Diff Array Technique.

T.C = Q * (Q+n).

Little Improvement

nums = [2, 0, 2]

Queries = [(0, 2, 1), (0, 2, 1), (1, 1, 3), Q, Q, Q]

(mid + 1)

```
while (l <= r) {
for (int i = 0; i < Q; i++) {        log(Q)
    mid = l + (r - l)/2;
    if ((CheckWithDiffArrEq(nums, Queries, mid) == True){
        result = mid+1; //count;
        r = mid-1;
    } else {
        l = mid+1;
    }
}
return result;
```

T.C = log(Q) * (Q + n)