Difference Array

Concepts & Qns

Video-6

(M)otivation :-

Few months of focus can set many years of your future.

MIK...

Choice is yours.

3346. Maximum Frequency of an Element After Performing Operations I

Solved

Prefix

Medium | Topics | Companies | Hint

You are given an integer array `nums` and two integers `k` and `numOperations`.

You must perform an **operation** `numOperations` times on `nums`, where in each operation you:

- Select an index `i` that was **not** selected in any previous operations.
- Add an integer in the range `[-k, k]` to `nums[i]`.

Return the **maximum** possible frequency of any element in `nums` after performing the **operations**.

Example :- nums = [1, 4, 5], K = 1, numOperations = 2

K = 1
(-K, K] = -1, 0, 1
4 + 0 = 5
5 + 0 = 5

Output : 2

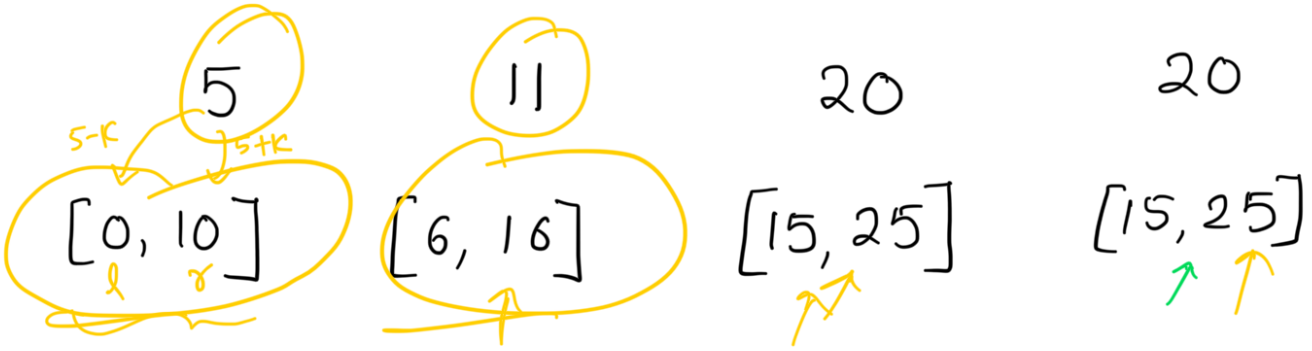
Thought Process

nums = [5, 11, 20, 20]

K = 5

numOperations = 1

5 + (-4) = 1



0, 1, 2, 3, 4, 5, 6, 7, ... 9, 10, 11, 12, ... 15, 16, 17, 18, ... 20, 21, 22, 23, 24, 25, 26

1 0 0 0 0 0 1 0 0 0 -1 0 2 0 -1 0 0 0 0 0 0 0, -2

[1, 1, 1, 1, 1, 1, 1, 2, 2, .. 2, 2, 1, 1, 1.... 3, 3, 2, 2 .... 2, 2, 2, 2, 2, 2, 0]

O(9 * n)

totalCount = 3

ElemOwnFreq = 0

needConversion = 3 - 0 = 3

maxPerOperant = min(3, numOperand)

= min(3, 1) = 1

totalOps = 0 + 1 = 1

DAT

① $[l, r] \rightarrow$ diff[l] ++;
diff[r+1]--;

② Cumulative Sum.

maxVal = (max-element) + K

```
diff <maxEl+2, 0> ;
unordered_map<int, int> freq;

for ( int i=0; i< nums.size(); i++) {
    freq[nums[i]]++;

    l = max( nums[i] - K, 0);
    r = min(nums[i] + K, maxVal);

    diff[l]++;
    diff[r+1]--;

}
```

Step-1 (DAT)

// CumSum .

```
for (int target = 0; target <= maxval; target++) {
    diff[target] = (target > 0 ? diff[target - 1] : 0);
    int targetFreq = freq[target];
    int needConversion = diff[target] - targetFreq;
    int maxPossFreq = min(needConversion,
                          numOperat);
    result = max(result, targetFreq +
                 maxPossFreq);
}

ret result;
```

O(maxVal)

S.C O(maxval)

Part - II

3347. Maximum Frequency of an Element After Performing Operations II

Hard | Topics | Companies | Hint

You are given an integer array `nums` and two integers `k` and `numOperations`.

You must perform an **operation** `numOperations` times on `nums`, where in each operation you:

- Select an index `i` that was **not** selected in any previous operations.
- Add an integer in the range `[-k, k]` to `nums[i]`.

Return the **maximum** possible frequency of any element in `nums` after performing the **operations**

Constraints:

- $1 \le nums.length \le 10^5$
- $1 \le nums[i] \le 10^9$
- $0 \le k \le 10^9$
- $0 \le numOperations \le nums.length$

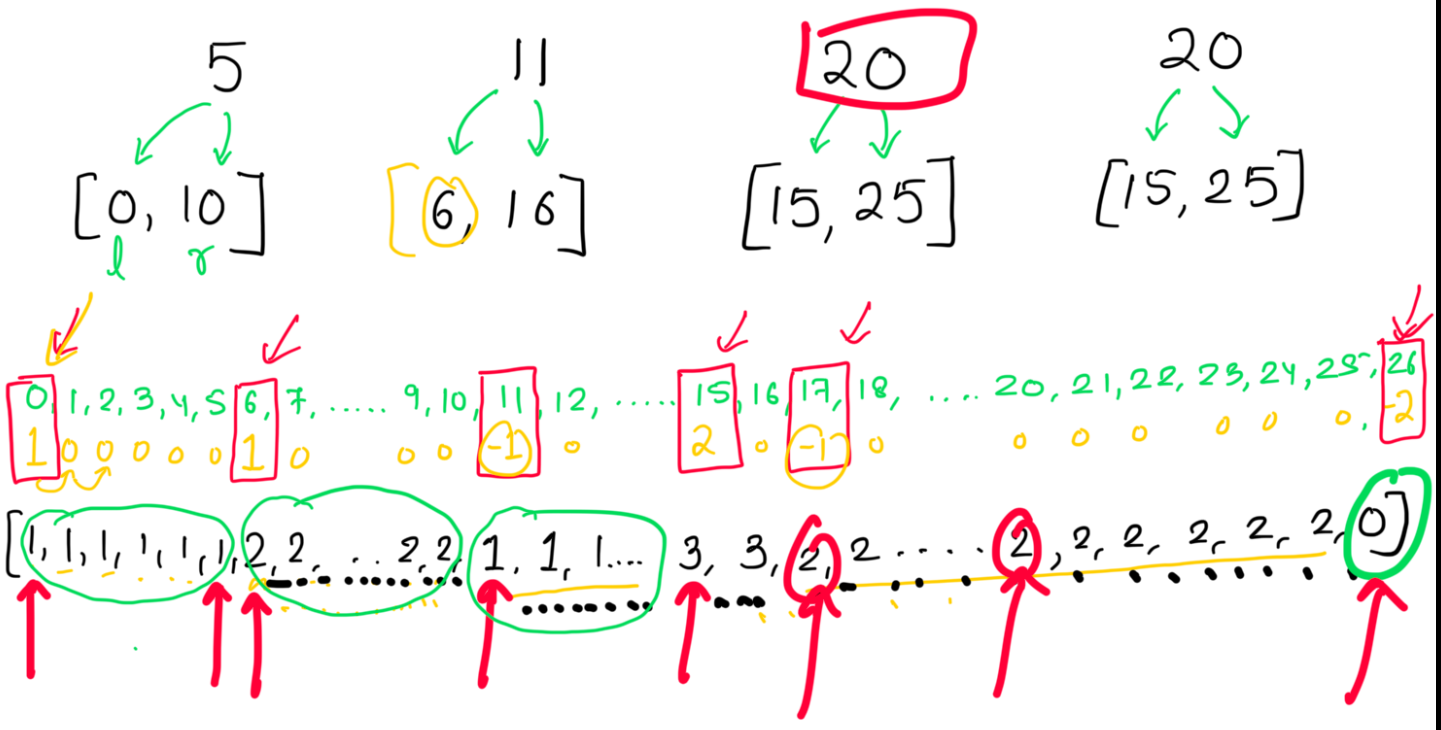
what's wrong with our solution of Part - I

```cpp
int maxFrequency(vector<int>& nums, int k, int numOperations) {
    int maxVal = *max_element(begin(nums), end(nums)) + k;

    vector<int> diff(maxVal+2, 0);
    unordered_map<int, int> freq;

    for(int i = 0; i < nums.size(); i++) {
        freq[nums[i]]++;

        int l = max(nums[i]-k, 0);
        int r = min(nums[i]+k, maxVal);

        diff[l]++;
        diff[r+1]--;
```

```
}

int result = 1;

for(int target = 0; target <= maxVal; target++) {
    diff[target] += (target > 0 ? diff[target-1] : 0);

    int targetFreq     = freq[target];
    int needConversion = diff[target] - targetFreq;

    int maxPossibleFreq = min(needConversion, numOperations);

    result = max(result, targetFreq + maxPossibleFreq);
}
```

// 20
2
// 2
2 - 2 = 0
min(0, 1) = 0
2 + 0 = 2

nums = {5, 11, 20, 20}, K = 5



0 : 1
6 : 1
11 : -1
15 : 2
17 : -1
26 : -2

{0 : 1, 5 : 0, 6 : 1, 11 : -1 + 0, 15 : 2, 17 : -1, 20 : 0, 26 : -2}

comm. map

mp:

{0:1, 5:1, 6:2, 11:1, 15:3, 17:2, 20:2, 26:0}

target target :.

DAT :-

① diff[l]++;
diff[r+1]--;

② cum :.

CumSum = 0;

for (it = mp.begin(); it != end ; it++) {

target = it → first;

it → second += CumSum;

{

int targetFreq = freq[target];

int needConversion = (it → second − targetFr);

int maxPossFreq = min(needConver, numOpts);

result = max(result, targetFr + maxPoss);

CumSum = it → second;

}

n₁ n₂

T.C :-

nums = {a, b, c, d} n

{ a-K , a+K, b-K, b+K } → 2*n.

T.C: O(n) ✓ n elements

S.C = O(n).

map: log

↳ (n * log(n)).