3347. Maximum Frequency of an Element After Performing Operations II

Solved

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**.

Example 1:

> **Input:** `nums = [1,4,5], k = 1, numOperations = 2`
>
> **Output:** 2
>
> **Explanation:**
>
> We can achieve a maximum frequency of two by:
>
> - Adding 0 to `nums[1]`, after which `nums` becomes `[1, 4, 5]`.
> - Adding -1 to `nums[2]`, after which `nums` becomes `[1, 4, 4]`.

Example 2:

> **Input:** `nums = [5,11,20,20], k = 5, numOperations = 1`
>
> **Output:** 2
>
> **Explanation:**
>
> We can achieve a maximum frequency of two by:
>
> - Adding 0 to `nums[1]`.

Constraints:

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

Python:

class Solution:

```
    def maxFrequency(self, nums: List[int], k: int, numOperations: int) -> int:
        points_cover = collections.Counter()
        cnt_points = collections.Counter()
        points = set()
        for num in nums:
            cnt_points[num] += 1
            points_cover[num - k] += 1
            points_cover[num + k + 1] -= 1
            points.update({num, num - k, num + k + 1})
        res = points_cover_this_point = 0
        for point in sorted(points):
            points_cover_this_point += points_cover[point]
            res = max(res, cnt_points[point] +
                      min(points_cover_this_point - cnt_points[point], numOperations))
        return res
```

JavaScript:

```
var maxFrequency = function(nums, k, numOperations) {
    nums.sort((a, b) => a - b);
    const n = nums.length;

    let left = 0, right = 0;
    let sumCount = 0;
    let result = 0;
    let left2 = 0;
    let sumCount2 = 0;
    let count = 0;
    let prev = null;

    for (const num of nums) {
        if (num === prev) {
            count++;
        } else {
            prev = num;
            count = 1;
        }

        while (nums[left] < num - k) {
            sumCount--;
            left++;
        }

        while (right < n && nums[right] <= num + k) {
```

```
            sumCount++;
            right++;
        }

        result = Math.max(result, count + Math.min(sumCount - count, numOperations));

        sumCount2++;
        while (nums[left2] < num - 2 * k) {
            sumCount2--;
            left2++;
        }
        result = Math.max(result, Math.min(sumCount2, numOperations));
    }

    return result;
};
```

Java:

```
class Solution {
    public int maxFrequency(int[] nums, int k, int numOperations) {
        if (nums.length == 1) return 1;
        Arrays.sort(nums);
        int right = Math.min(numOperations, prepareMaxNums(nums, k));
        int index = 0, left = 0, freq = 0;

        for (int i = 0; i < nums.length; i++) {
            int n = nums[i];

            freq = (i > 0 && nums[i] == nums[i - 1]) ? freq + 1 : 1;

            int min = n - k, max = n + k;

            while (true) {
                if (index < nums.length && nums[index] < min) {
                    index++;
                } else if (left < nums.length && nums[left] <= max) {
                    left++;
                } else {
                    break;
                }
            }

            right = Math.max(right, Math.min(freq + numOperations, left - index));
        }
```

```
        return right;
    }

    public int prepareMaxNums(int[] nums, int k) {
        int left = 0;
        int right = 0;

        for (int i = 0; i < nums.length; i++) {
            int target = nums[i] + 2 * k;

            while (left < nums.length && nums[left] <= target) {
                left++;
            }

            right = Math.max(right, left - i);
        }

        return right;
    }
}
```