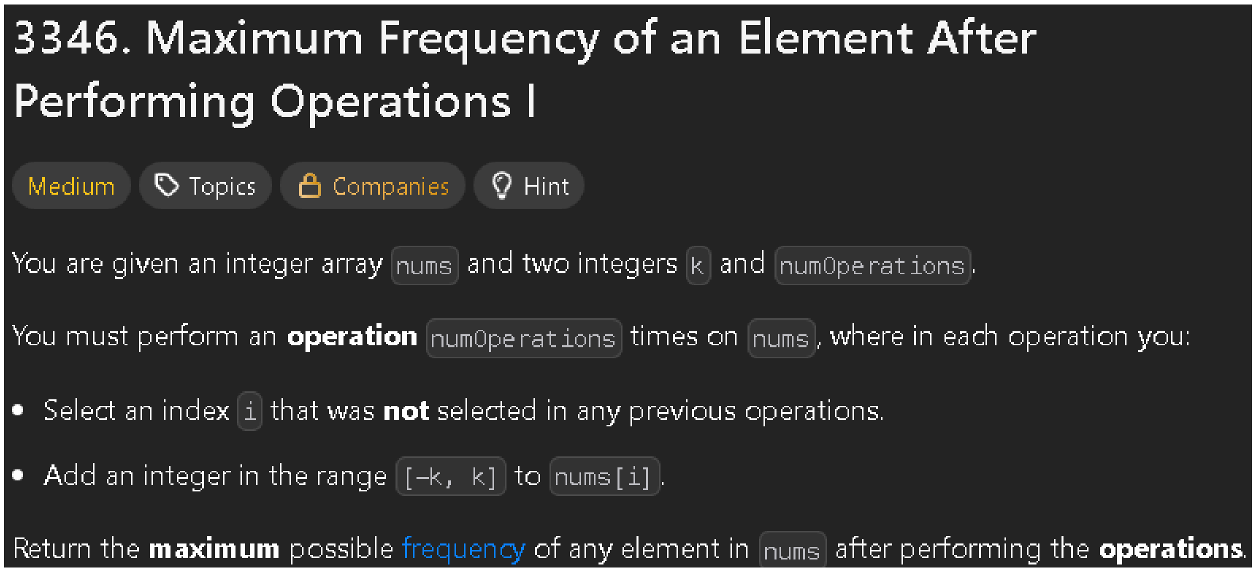


3346. Maximum Frequency of an Element After Performing Operations I

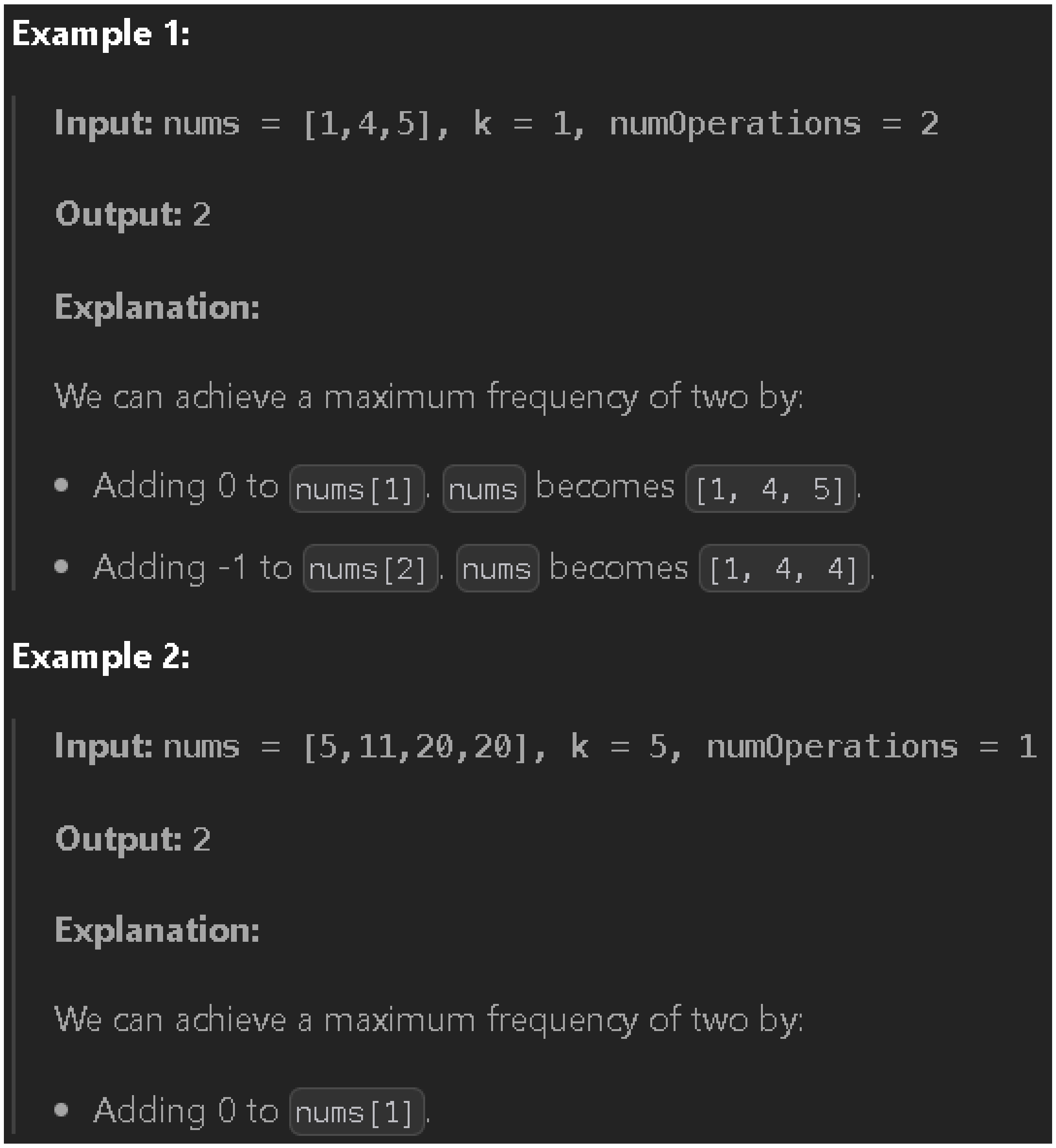
Medium · Topics · Companies · Hint

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]`.

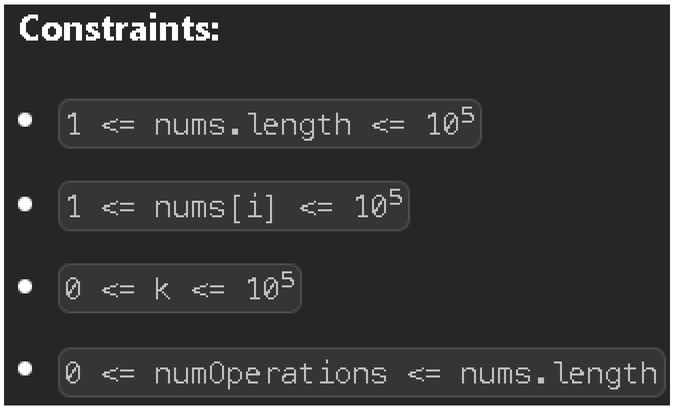
Return the **maximum** possible frequency of any element in `nums` after performing the **operations**.


Example 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]`. `nums` becomes `[1, 4, 5]`.
> - Adding -1 to `nums[2]`. `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:

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

Python:

```
class Solution:
    def maxFrequency(self, nums: List[int], k: int, numOperations: int) -> int:
        M=max(nums)+2
        freq, sweep=[0]*M, [0]*M
        mm=M
        for x in nums:
            freq[x]+=1
            s, t=max(1, x-k), min(M-1, x+k+1)
            sweep[s]+=1
            sweep[t]-=1
            mm=min(mm, s)
        ans, cnt=0, 0
        for x in range(mm, M):
            cnt+=sweep[x]
            ans=max(ans, freq[x]+min(numOperations, cnt-freq[x]))
        return ans
```

JavaScript:

```
const maxFrequency = (nums, k, numOps) => {
    const maxVal = Math.max(...nums) + k + 2;
    const count = new Array(maxVal).fill(0);

    for (const v of nums)
        count[v]++;

    for (let i = 1; i < maxVal; i++)
        count[i] += count[i - 1];

    let res = 0;
    for (let i = 0; i < maxVal; i++) {
        const left = Math.max(0, i - k);
        const right = Math.min(maxVal - 1, i + k);
        const total = count[right] - (left ? count[left - 1] : 0);
        const freq = count[i] - (i ? count[i - 1] : 0);
        res = Math.max(res, freq + Math.min(numOps, total - freq));
    }

    return res;
};
```

Java:

```
class Solution {
    public int maxFrequency(int[] nums, int k, int numOps) {
        int maxVal = Arrays.stream(nums).max().getAsInt() + k + 2;
        int[] count = new int[maxVal];
        for (int v : nums)
            count[v]++;

        for (int i = 1; i < maxVal; i++)
            count[i] += count[i - 1];

        int res = 0;
        for (int i = 0; i < maxVal; i++) {
            int left = Math.max(0, i - k);
            int right = Math.min(maxVal - 1, i + k);
            int total = count[right] - (left > 0 ? count[left - 1] : 0);
            int freq = count[i] - (i > 0 ? count[i - 1] : 0);
            res = Math.max(res, freq + Math.min(numOps, total - freq));
        }

        return res;
    }
}
```