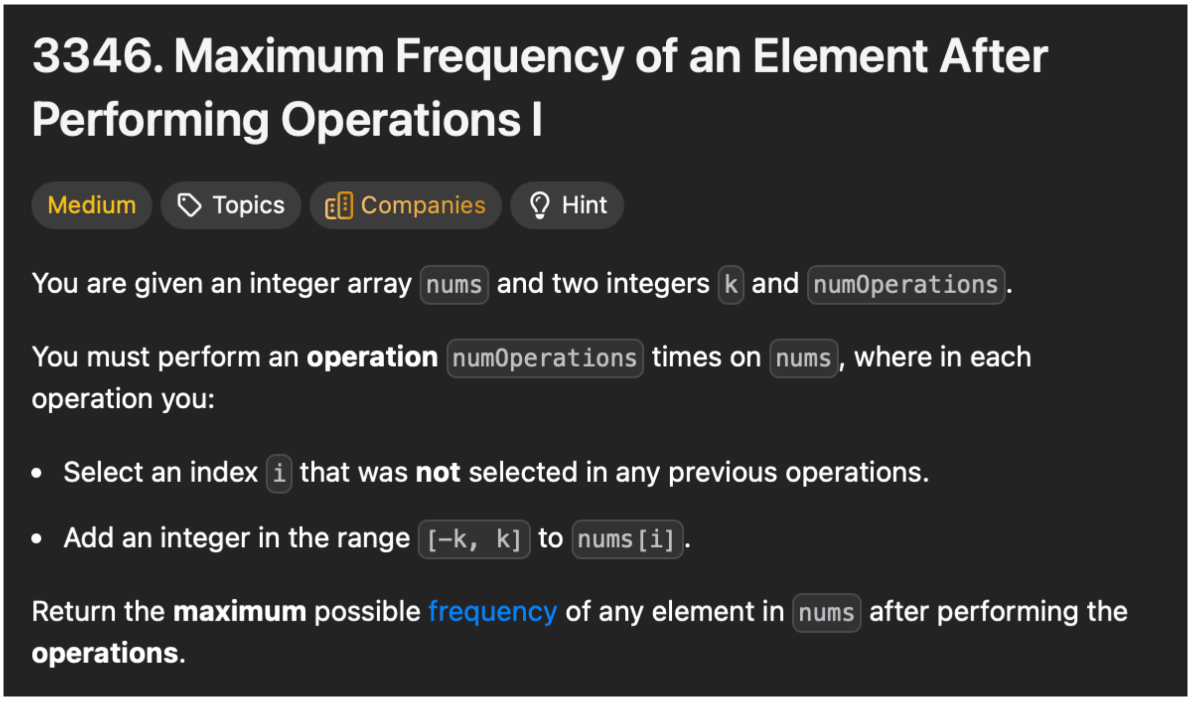


3346. Maximum Frequency of an Element After Performing Operations I

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

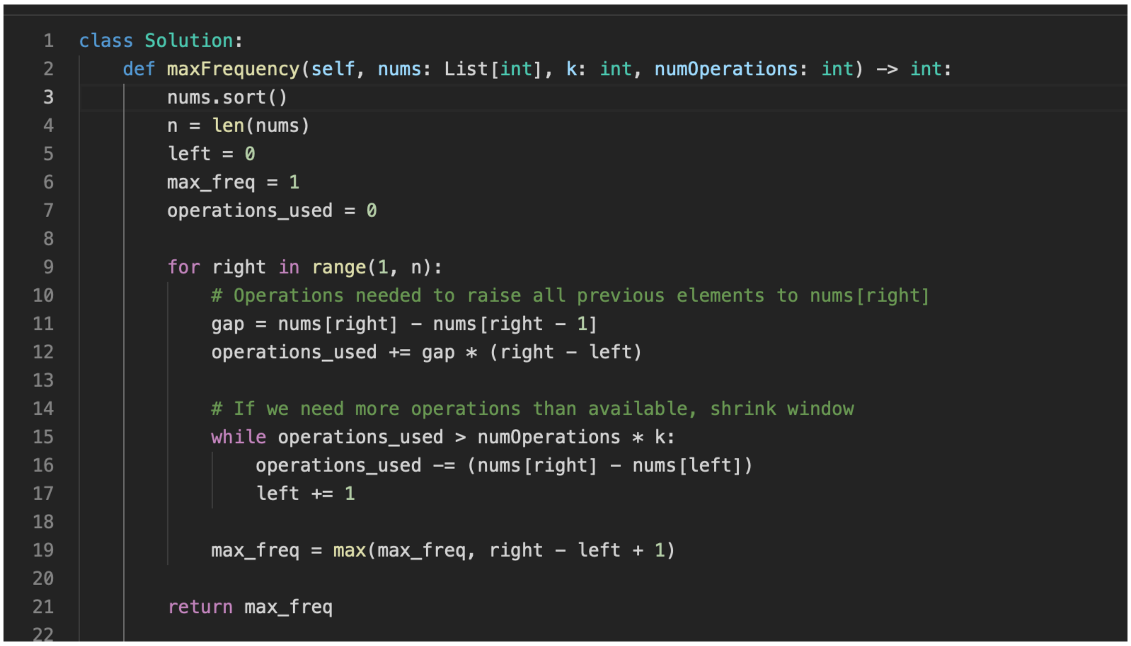
You must perform an **operation** `numOperations` times on `nums`, where in each operation you:

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

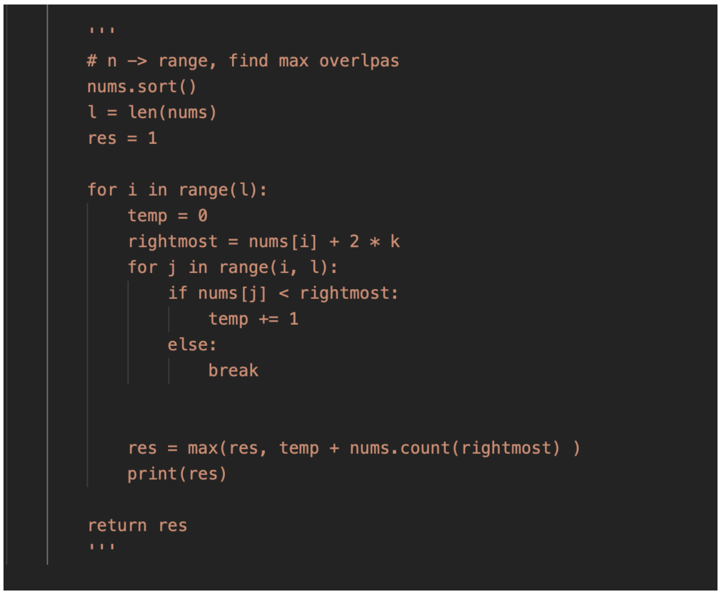
Return the **maximum** possible frequency of any element in `nums` after performing the **operations**.

solution: sliding window

1. sort
2. sliding window
   A. check the [curr - k, curr + k]
   B. check if exceed the numOperations allowed



```python
class Solution:
    def maxFrequency(self, nums: List[int], k: int, numOperations: int) -> int:
        nums.sort()
        n = len(nums)
        left = 0
        max_freq = 1
        operations_used = 0

        for right in range(1, n):
            # Operations needed to raise all previous elements to nums[right]
            gap = nums[right] - nums[right - 1]
            operations_used += gap * (right - left)

            # If we need more operations than available, shrink window
            while operations_used > numOperations * k:
                operations_used -= (nums[right] - nums[left])
                left += 1

            max_freq = max(max_freq, right - left + 1)

        return max_freq

```



```
'''
# n -> range, find max overlpas
nums.sort()
l = len(nums)
res = 1

for i in range(l):
    temp = 0
    rightmost = nums[i] + 2 * k
    for j in range(i, l):
        if nums[j] < rightmost:
            temp += 1
        else:
            break


    res = max(res, temp + nums.count(rightmost) )
    print(res)

return res
'''
```

old version: wrong

the range of nums needed to be checked is not in range and the additional added number of nums if not correct