3318. Find X-Sum of All K-Long Subarrays I

Solved

Easy | Topics | Companies | Hint

You are given an array `nums` of `n` integers and two integers `k` and `x`.

The **x-sum** of an array is calculated by the following procedure:

- Count the occurrences of all elements in the array.
- Keep only the occurrences of the top `x` most frequent elements. If two elements have the same number of occurrences, the element with the **bigger** value is considered more frequent.
- Calculate the sum of the resulting array.

Note that if an array has less than `x` distinct elements, its **x-sum** is the sum of the array.

Return an integer array `answer` of length `n - k + 1` where `answer[i]` is the **x-sum** of the subarray `nums[i..i + k - 1]`.

Example 1:

> **Input:** `nums = [1,1,2,2,3,4,2,3], k = 6, x = 2`
>
> **Output:** `[6,10,12]`
>
> **Explanation:**
>
> - For subarray `[1, 1, 2, 2, 3, 4]`, only elements 1 and 2 will be kept in the resulting array. Hence, `answer[0] = 1 + 1 + 2 + 2`.
> - For subarray `[1, 2, 2, 3, 4, 2]`, only elements 2 and 4 will be kept in the resulting array. Hence, `answer[1] = 2 + 2 + 2 + 4`. Note that 4 is kept in the array since it is bigger than 3 and 1 which occur the same number of times.
> - For subarray `[2, 2, 3, 4, 2, 3]`, only elements 2 and 3 are kept in the resulting array. Hence, `answer[2] = 2 + 2 + 2 + 3 + 3`.

Example 2:

> **Input:** nums = [3,8,7,8,7,5], k = 2, x = 2
>
> **Output:** [11,15,15,15,12]
>
> **Explanation:**
>
> Since `k == x`, `answer[i]` is equal to the sum of the subarray `nums[i..i + k - 1]`.

Constraints:

- `1 <= n == nums.length <= 50`
- `1 <= nums[i] <= 50`
- `1 <= x <= k <= nums.length`

Python:

```
from collections import defaultdict
from typing import List

class Solution:
    def findXSum(self, nums: List[int], k: int, x: int) -> List[int]:
        n = len(nums)
        freq = defaultdict(int)

        for i in range(k):
            freq[nums[i]] += 1

        def compute_x_sum(freq, x):
            items = [(v, f) for v, f in freq.items()]
            items.sort(key=lambda t: (-t[1], -t[0]))
            total = 0
            for i in range(min(x, len(items))):
                v, f = items[i]
                total += v * f
            return total

        ans = [compute_x_sum(freq, x)]
```

```
        for i in range(k, n):
            add = nums[i]
            rem = nums[i - k]
            freq[add] += 1
            freq[rem] -= 1
            if freq[rem] == 0:
                del freq[rem]
            ans.append(compute_x_sum(freq, x))

        return ans
```

JavaScript:

```
/**
 * @param {number[]} nums
 * @param {number} k
 * @param {number} x
 * @return {number[]}
 */
var findXSum = function(nums, k, x) {
  const n = nums.length;
  const ans = [];
  const freq = new Map();

  for (let i = 0; i < k; i++) {
    freq.set(nums[i], (freq.get(nums[i]) || 0) + 1);
  }

  ans.push(computeXSum(freq, x));

  for (let i = k; i < n; i++) {
    const add = nums[i];
    const rem = nums[i - k];

    freq.set(add, (freq.get(add) || 0) + 1);
    const fr = (freq.get(rem) || 0) - 1;
    if (fr === 0) freq.delete(rem);
    else freq.set(rem, fr);

    ans.push(computeXSum(freq, x));
  }

  return ans;

  function computeXSum(map, x) {
    const items = [];
```

```
    for (const [v, f] of map.entries()) items.push([v, f]);
    items.sort((a, b) => {
      if (a[1] !== b[1]) return b[1] - a[1];
      return b[0] - a[0];
    });
    let sum = 0;
    const take = Math.min(x, items.length);
    for (let i = 0; i < take; i++) {
      sum += items[i][0] * items[i][1];
    }
    return sum;
  }
};
```

Java:

```
class Solution {
    public int[] findXSum(int[] nums, int k, int x) {
        int[] ans=new int[nums.length-k+1];

        int left=0;
        int right=k-1;
        int[] freq= new int[51]; //for storing the count
        for(int i=left;i<=right;i++)freq[nums[i]]++; //counted
        ans[0]=find(freq,x); //firstSubArray'sAnswer

        int ind=1; //ans array pointer
        while(right<nums.length){
            right++; //next one in the array
            if(right==nums.length)break; //if reached out break
            freq[nums[right]]++; //count the incoming number
            freq[nums[left]]--; //decrement left most number's count
            left++; //update left

            ans[ind++]=find(freq,x); //find the top x for this freq
        }
        return(ans);
    }
    private int find(int[] f, int x){
        //create a deep copy of f as f needs to be preserved for next subarrays
        int[] freq= new int[51];
        for(int i=0;i<51;i++) freq[i]=f[i]; //done
        int[] nums=new int[51];
        for(int i=0;i<51;i++)nums[i]=i; //we need a freq:value pointer, hence this array

        //perform any sort (used insertion below) on freq to be descending
```

```
        //while modifying freq modify corresponding num as well in the nums array
        //so the freq:value pair order is preserved
        for(int i=1;i<51;i++){
            int key=freq[i];
            int numKey=nums[i];
            int j=i-1;
            while(j>=0 && freq[j]<=key){
                freq[j+1]=freq[j];
                nums[j+1]=nums[j];
                j--;
            }
            freq[j+1]=key;
            nums[j+1]=numKey;
        }
        //in the end we get the descending order of the freq
        //along with which number's freq each is
        int sum=0;
        for(int i=0;i<x;i++){ //simply calculate the sum of numbers upto x elements
            sum+=(nums[i]*freq[i]); //each i in nums have repeated freq[i] times, so add that
        }
        return(sum);
    }
}
```