689. Maximum Sum of 3 Non-Overlapping Subarrays

- Difficulty:Hard
- Total Accepted:4.8K
- Total Submissions:11.6K
- Contributor: 1337c0d3r

In a given array `nums` of positive integers, find three non-overlapping subarrays with maximum sum.

Each subarray will be of size `k`, and we want to maximize the sum of all `3*k` entries.

Return the result as a list of indices representing the starting position of each interval (0-indexed). If there are multiple answers, return the lexicographically smallest one.

Example:

```
Input: [1,2,1,2,6,7,5,1], 2
Output: [0, 3, 5]
Explanation: Subarrays [1, 2], [2, 6], [7, 5] correspond to the starting indices
[0, 3, 5].
We could have also taken [2, 1], but an answer of [1, 3, 5] would be
lexicographically larger.
```

Note:

- `nums.length` will be between 1 and 20000.
- `nums[i]` will be between 1 and 65535.
- `k` will be between 1 and floor(nums.length / 3).

```
class Solution {
public:
    vector<int> maxSumOfThreeSubarrays(vector<int>& nums, int k) {
        int n = nums.size();
        vector<int> W(n-k+1,0);
        int sum = 0;
        for(int i=0;i<n;i++)
        {
            sum+=nums[i];
            if(i>=k) sum-=nums[i-k];
            if(i>=k-1) W[i-k+1]=sum;
        }
        vector<int> left(W.size(),0);
        int best = 0;
        for(int i=0;i<W.size();i++)
        {
            if(W[i]>W[best]) best=i;
            left[i]=best;
        }
        best = W.size()-1;vector<int> right(W.size(),0);
        for(int i=W.size()-1;i>=0;i--)
        {
            if(W[i]>=W[best]) best=i;
            right[i]=best;
        }
        vector<int> ans(3,-1);
        for(int j=k;j<W.size()-k;j++)
        {
            int i=left[j-k];int kk=right[j+k];
            if(ans[0]==-1 ||W[i]+W[j]+W[kk]>W[ans[0]]+W[ans[1]]+W[ans[2]])
            {
                ans[0] = i;
                ans[1] = j;
                ans[2] = kk;
            }
        }
        return ans;
    }
};
```