All Divisions With the Highest Score of a Binary Array(LeetCode Weekly 278)

Question:

https://leetcode.com/contest/weekly-contest-278/problems/all-divisions-with-the-highest-score-of-a-binary-array/

You are given a 0-indexed binary array `nums` of length `n`. `nums` can be divided at index `i` (where `0 <= i <= n)` into two arrays (possibly empty) `numsleft` and `numsright`:

- *`numsleft` has all the elements of `nums` between index `0` and `i - 1` (inclusive), while `numsright` has all the elements of nums between index `i` and `n - 1` (inclusive).*
- *If `i == 0`, `numsleft` is empty, while `numsright` has all the elements of `nums`.*
- *If `i == n`, `numsleft` has all the elements of nums, while `numsright` is empty.*

The division score of an index `i` is the sum of the number of `0`'s in `numsleft` and the number of `1`'s in `numsright`.

Return all distinct indices that have the highest possible division score. You may return the answer in any order.

Example 1:

```
Input: nums = [0,0,1,0]
Output: [2,4]
Explanation: Division at index
- 0: numsleft is []. numsright is [0,0,1,0]. The score is 0 + 1 = 1.
- 1: numsleft is [0]. numsright is [0,1,0]. The score is 1 + 1 = 2.
- 2: numsleft is [0,0]. numsright is [1,0]. The score is 2 + 1 = 3.
- 3: numsleft is [0,0,1]. numsright is [0]. The score is 2 + 0 = 2.
- 4: numsleft is [0,0,1,0]. numsright is []. The score is 3 + 0 = 3.
Indices 2 and 4 both have the highest possible division score 3.
Note the answer [4,2] would also be accepted.
```

Approach 1:

Counted the zeros from left(NumsLeft) and ones from the right(NumsRight).

Find the total sum for each index **i** and if it is the maximum sum, appended the index in a list.

Solution 1:

https://gist.github.com/vermaayush680/62267ea35546f8826b674e7fba85909d

```
class Solution:
    def maxScoreIndices(self, nums: List[int]) -> List[int]:
            n=len(nums)
            zero=[0]*(n+1)
            one=[0]*(n+1)
            for i in range(n):
                zero[i+1]=zero[i]+(nums[i]==0)
            for i in range(n-1,-1,-1):
                one[i]=one[i+1]+(nums[i]==1)
            total = [0]*(n+1)
            m=0
            res=[]
            for i in range(n+1):
                total[i]=zero[i]+one[i]
                if total[i]>m:
                    res=[]
                    m=total[i]
                if total[i]==m:
                    res+=[i]
            return res
```

Time Complexity: O(n)

Space Complexity: O(n)

Approach 2:

Another approach was to use the PREFIX sum.

We use a prefix sum which calculates the total prefix sum at each index i.

This combines the first two loops from the previous solution into one.

Solution 2:

https://gist.github.com/vermaayush680/b15add70e1934dcd99bc47c25ed9600a

```
class Solution:
    def maxScoreIndices(self, nums: List[int]) -> List[int]:
        n=len(nums)
        res=[]
        pref=[0]*(n+1)
        for i in range(n):
            pref[i+1]=pref[i]+nums[i]
        zero,total,one=0,0,0
        m=-1
        for i in range(n+1):
            one=pref[n]-pref[i]
            zero=i-pref[i]
            total=zero+one
            if total>m:
                m=total
                res=[]
            if total==m:
                res+=[i]
        return res
```

Time Complexity: O(n)

Space Complexity: O(n)

Approach 3:

Instead of using a prefix array, we can just count the number of ones and use it to calculate prefix sum at each index i.

This reduces the Space Complexity to O(1).

Solution 3:

https://gist.github.com/vermaayush680/98de5e20ec99d106a393f70275f1d06a

```
class Solution:
    def maxScoreIndices(self, nums: List[int]) -> List[int]:
        n=len(nums)
        res=[0]
        onecount=0
        for i in range(n):
            onecount+=(nums[i]==1)
        m=onecount
        for i in range(n):
            onecount+=(nums[i]==0)-(nums[i]==1)
            if onecount>=m:
                if onecount!=m:
                    m=onecount
                    res=[]
                res+=[i+1]
        return res
```

Time Complexity: O(n)

Space Complexity: O(1)