1524. Number of Sub-arrays With Odd Sum

Given an array of integers `arr`, return *the number of subarrays with an **odd** sum*.

Since the answer can be very large, return it modulo 10^9+7 .

Example 1:

```
Input: arr = [1,3,5]
Output: 4
Explanation: All subarrays are [[1],[1,3],[1,3,5],[3],[3,5],[5]]
All sub-arrays sum are [1,4,9,3,8,5].
Odd sums are [1,9,3,5] so the answer is 4.
```

Example 2:

```
Input: arr = [2,4,6]
Output: 0
Explanation: All subarrays are [[2],[2,4],[2,4,6],[4],[4,6],[6]]
All sub-arrays sum are [2,6,12,4,10,6].
All sub-arrays have even sum and the answer is 0.
```

Example 3:

```
Input: arr = [1,2,3,4,5,6,7]
Output: 16
```

Constraints:

- `1 <= arr.length <=` 10^5
- `1 <= arr[i] <= 100`

1524. Number of Sub-arrays With Odd Sum

Overview

We are given an array of integers, and our task is to count the number of subarrays whose sums are odd. Since the number of possible subarrays can be large, we return the count modulo 10^9+7 .

A subarray is a contiguous portion of the array, meaning we must consider all possible starting and ending indices. The sum of a subarray is simply the sum of its elements.

For example, given `arr = [1,3,5]`, the possible subarrays are:

- `[1]` → `sum = 1 (odd)`
- `[1,3]` → `sum = 4 (even)`
- `[1,3,5]` → `sum = 9 (odd)`
- `[3]` → `sum = 3 (odd)`
- `[3,5]` → `sum = 8 (even)`
- `[5]` → `sum = 5 (odd)`

Here, the subarrays with an odd sum are `[1]`, `[1,3,5]`, `[3]`, and `[5]`, giving us the answer `4`.

A sliding window approach is generally useful when dealing with subarrays, but in this case, we cannot use it effectively. The problem requires counting all valid subarrays, not just maintaining a fixed window of size `k` or optimizing a contiguous segment. Since the valid subarrays are scattered across different positions and lengths, sliding window techniques do not provide any direct optimization here.

1524. Number of Sub-arrays With Odd Sum

Approach 1: Brute Force (TLE)

Intuition

The most direct way to solve this problem is to explicitly generate all possible subarrays and check which ones have an odd sum.

To do this, we iterate over every possible starting index in the array. For each start, we extend the subarray one element at a time, maintaining a running sum as we go. Each time we add a new element to the sum, we check if it is odd. If it is, we increment our count.

Since we check all possible start and end pairs, the number of subarrays we examine is proportional to the square of the array size, leading to a time complexity of $O(n2)$. This means that for large arrays, the approach becomes too slow to be practical, leading to a Time Limit Exceeded (TLE) error.

```
class Solution {
public:
    int numOfSubarrays(vector<int>& arr) {
        const int MOD = 1e9 + 7;
        int n = arr.size();
        int count = 0;

        // Generate all possible subarrays
        for (int startIndex = 0; startIndex < n; startIndex++) {
            int currentSum = 0;
            // Iterate through each subarray starting at startIndex
            for (int endIndex = startIndex; endIndex < n; endIndex++) {
                currentSum += arr[endIndex];
                // Check if the sum is odd
                if (currentSum % 2 != 0) {
                    count++;
                }
            }
        }

        return count % MOD;
    }
};
```

1524. Number of Sub-arrays With Odd Sum

```
/*
   Prefix sum+counting
   Time compelxity: O(n)
   Space compelxity: O(1)
*/
class Solution {
public:
   int numOfSubarrays(vector<int>& arr) {
      const int MOD=1e9+7;
      int n=arr.size();
      int prefix_sum=0;
      int even_freq=1; // 0 of the prefix sum is even
      int odd_freq=0;
      int ans=0;
      for(int i=0;i<n;++i){
         prefix_sum+=arr[i]; // Build prefix sum from 0 to i
         even_freq+=prefix_sum%2==0; // Count frequency of even sums from 0 to i
         odd_freq+=prefix_sum%2==1; // Count frequency of odd sums from 0 to i

         // If sum from 0 to i is odd, add to the answer the frequency of even sums
         // Otherwise, add to the answer the frequency of odd sums
         ans=(prefix_sum%2?(ans+even_freq%MOD):(ans+odd_freq%MOD))%MOD;
      }

      return ans;
   }
};
```