Contest Problem B - 3152. Special Array II

An array is considered special if every pair of its adjacent elements contains two numbers with different parity. You are given an array of integer `nums` and a 2D integer matrix `queries`, where for `queries[i] = [from_i, to_i]` your task is to check that subarray `nums[from_i..to_i]` is special or not. Return an array of booleans `answer` such that `answer[i]` is `true` if `nums[from_i..to_i]` is special.

Example 1:

Input: nums = [3,4,1,2,6], queries = [[0,4]]

Output: [false]

Explanation:

The subarray is `[3,4,1,2,6]`. 2 and 6 are both even.

Example 2:

Input: nums = [4,3,1,6], queries = [[0,2],[2,3]]

Output: [false,true]

Explanation:

1. The subarray is `[4,3,1]`. 3 and 1 are both odd. So the answer to this query is `false`.
2. The subarray is `[1,6]`. There is only one pair: `(1,6)` and it contains numbers with different parity. So the answer to this query is `true`.

Constraints:

- `1 <= nums.length <= 10`^{5}
- `1 <= num s[i] <= 10`^{5}
- `1 <= queries.length <= 10`^{5}
- `queries[i].length == 2`
- `0 <= queries[i][0] <= queries[i][1] <= nums.length - 1`

Intuition:

1. First observation was as i have to answer for every query if in that given range there is a parity similarity.
2. In naive way i could check for every query from queries[i][0] to queries[i][1] if there exists a parity similarity or not. But it would take O (n * m) which can cause a Time limit exceed. So for every query we have to know if there is a parity similarity.
3. To do so i took a prefix array with all of them initially zero. And traverse from 0 to n on every nums[i]. If the present nums[i] parity is same as the previous nums[i] parity then we increase the prefix[i] = prefix[i-1]+1; else prefix[i] = prefix[i-1];
4. So for every query for range L to R if the prefix[R] - prefix[L] is 0 then the subarray of range L To R is beautiful else no.

Solution:

Complexity O (N)

```
class Solution {
public:
    vector<bool> isArraySpecial(vector<int>& nums, vector<vector<int>>&
queries) {
        int n = nums.size();
        int prefix[n];
        for(int i=0; i<n; i++) prefix[i]=0;
        int prev = nums[0]%2;
        for(int i=1; i<n; i++){
            int x = nums[i]%2;
            if(prev==x){
                prefix[i] = prefix[i-1]+1;
            }
            else{
                prev = x;
                prefix[i] = prefix[i-1];
            }
        }

        vector<bool> ans;

        for(int i=0; i<queries.size(); i++){
            int x = queries[i][0]; int y = queries[i][1];
            if(prefix[y]-prefix[x]>0){
                ans.push_back(false);
            }else{
                ans.push_back(true);
                }
            }

            return ans;
        }
};
```