3152. Special Array II

Medium | Topics | Companies | Hint

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.

Brute force Time Limit Exceeded

```python
class Solution:
    def isArraySpecial(self, nums: List[int], queries: List[List[int]]) -> List[bool]:
        dic = []
        for i in range(len(nums)-1):
            if nums[i] % 2 != nums[i+1] % 2:
                dic.append(True)
            else:
                dic.append(False)

        res = []
        for q in queries:
            f, t = q
            if False in dic[f:t]:
                res.append(False)
            else:
                res.append(True)

        return res
```

$O(n^2)$

⇒

```python
class Solution:
    def isArraySpecial(self, nums: List[int], queries: List[List[int]]) -> List[bool]:
        n = len(nums)

        special = [nums[i] % 2 != nums[i + 1] % 2 for i in range(n - 1)]

        prefix_sum = [0] * len(special)
        prefix_sum[0] = special[0]
        for i in range(1, len(special)):
            prefix_sum[i] = prefix_sum[i - 1] + special[i]

        result = []
        for f, t in queries:
            # Calculate the number of `True` in the range
            if t - 1 < f:
                result.append(True)
            else:
                total = prefix_sum[t - 1] - (prefix_sum[f - 1] if f > 0 else 0)
                result.append(total == t - f)

        return result
```

$O(n)$

- Use special[ ] to replace dic[ ], this has no effect on complexity
- Use prefix_sum[ ] to avoid repeated comparison, this reduce complexity from O(n^2) to O(n)