2425. Bitwise XOR of All Pairings

You are given two **0-indexed** arrays, `nums1` and `nums2`, consisting of non-negative integers. There exists another array, `nums3`, which contains the bitwise XOR of **all pairings** of integers between `nums1` and `nums2` (every integer in `nums1` is paired with every integer in `nums2` **exactly once**).

Return *the* ***bitwise XOR*** *of all integers in* `nums3`.

Example 1:

```
Input: nums1 = [2,1,3], nums2 = [10,2,5,0]
Output: 13
Explanation:
A possible nums3 array is [8,0,7,2,11,3,4,1,9,1,6,3].
The bitwise XOR of all these numbers is 13, so we return 13.
```

Example 2:

```
Input: nums1 = [1,2], nums2 = [3,4]
Output: 0
Explanation:
All possible pairs of bitwise XORs are nums1[0] ^ nums2[0], nums1[0] ^ nums2[1],
nums1[1] ^ nums2[0],
and nums1[1] ^ nums2[1].
Thus, one possible nums3 array is [2,5,1,6].
2 ^ 5 ^ 1 ^ 6 = 0, so we return 0.
```

Constraints:

- `1 <= nums1.length, nums2.length <=` 10^5
- `0 <= nums1[i], nums2[j] <= 109`

2425. Bitwise XOR of All Pairings

```
/*
   XOR properties
   Time complexity: O(n+m)=O(n)
   Space complexity: O()
*/
class Solution {
   public:
      int xorAllNums(std::vector<int>& nums1, std::vector<int>& nums2){
         auto solve=[&](std::vector<int>&A)->int{
            int ans=0;
            for(auto& e: A) ans^=e;
            return ans;
         };

         int n=nums1.size();
         int m=nums2.size();

         if(n%2==0 && m%2==0) return 0;
         if(n%2==1 && m%2==1) return solve(nums1)^solve(nums2);
         if(n%2==0) return solve(nums1);
         return solve(nums2);
      }
   };
```