SORT ARRAY BY PARITY

LEETCODE:SORT ARRAY BY PARITY
QUESTION:Given an integer array nums, move all the even integers at the beginning of the array followed by all the odd integers.
Return ***any array*** *that satisfies this condition*.

Example 1:

```
Input: nums = [3,1,2,4]
Output: [2,4,3,1]
Explanation: The outputs [4,2,3,1], [2,4,1,3], and [4,2,1,3] would also be accepted.
```

Example 2:

```
Input: nums = [0]
Output: [0]
```

Constraints:

- 1 <= nums.length <= 5000
- 0 <= nums[i] <= 5000

APPROACH ONE:
I HAVE TAKEN ONE ADDITIONAL ARRAY AND USED TWO FOR LOOPS AND FIRST PUSHED ALL THE ELEMENTS DIVISIBLE BY 2 INTO IT..
ANOTHER FOR LOOP AND PUSHED ALL THE ELEMENTS NOT DIVISIBLE BY 2...
SO HERE I TOOK:- O(N) SPACE AND O(N) TIME

```
class Solution {
public:
    vector<int> sortArrayByParity(vector<int>& nums) {
        vector<int> ans;
        int n=nums.size();
```

```
        for(int i=0;i<n;i++){
            if(nums[i]%2==0){
                ans.push_back(nums[i]);
            }
        }
        for(int i=0;i<n;i++){
            if(nums[i]%2!=0){
                ans.push_back(nums[i]);
            }
        }
        return ans;
    }
};
```

APPROACH TWO:

I TOOK TWO POINTERS ONE AT BEGINING AND ONE AT END ELMENT OF ARRAY and executed faster than previous one and time complexity here is: o(n) space complexity is o(1);

```
class Solution {
public:
    vector<int> sortArrayByParity(vector<int>& nums) {

        int n=nums.size();
        int i=0,j=n-1;
        while(i<j){
            int temp=0;
            if(nums[i]%2==0){
                i++;
            }
            else{
                temp=nums[i];
                nums[i]=nums[j];
                nums[j]=temp;
                j--;
            }
        }
        return nums;
          }
};
```