540. Single Element in a Sorted Array

Solution 1 (brute force)

INTUTION:

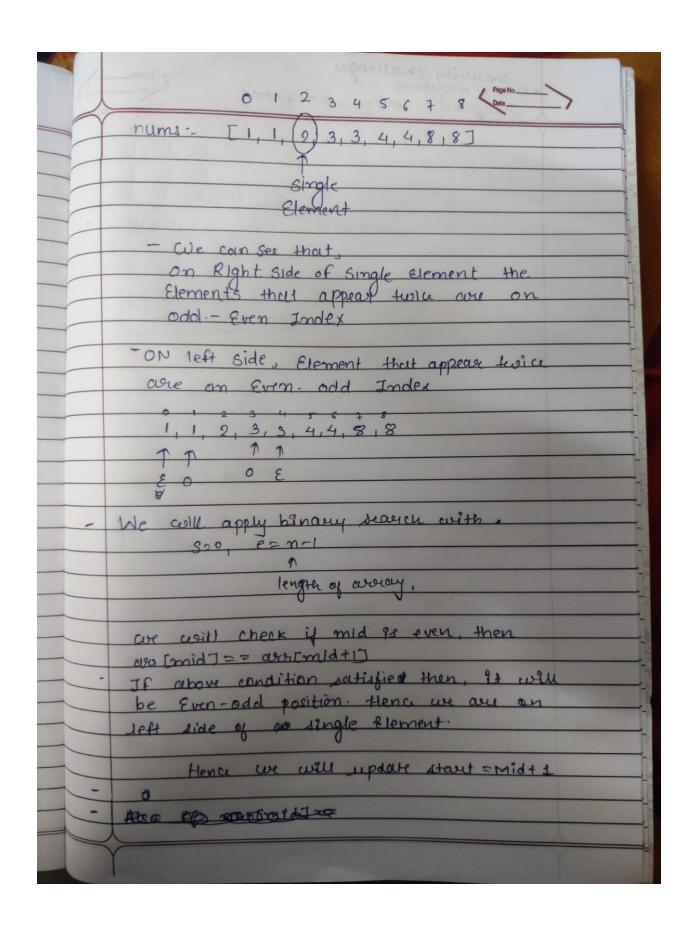
Here we have given that the array contains every elements twice and only element appear only once. So we get solution by XOR each element. As we know XORing same numbers will give 0 and XORing 0 with any number give the number itself. So we will get the single element by XORing all the elements.

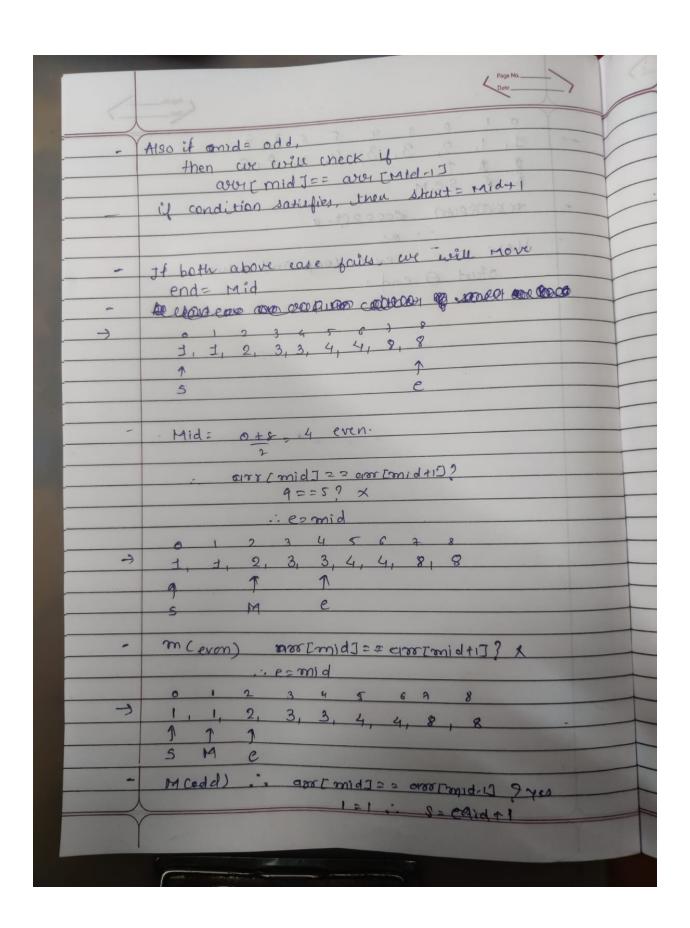
$ex.1^{1} = 0$ and $0^{5} = 5$

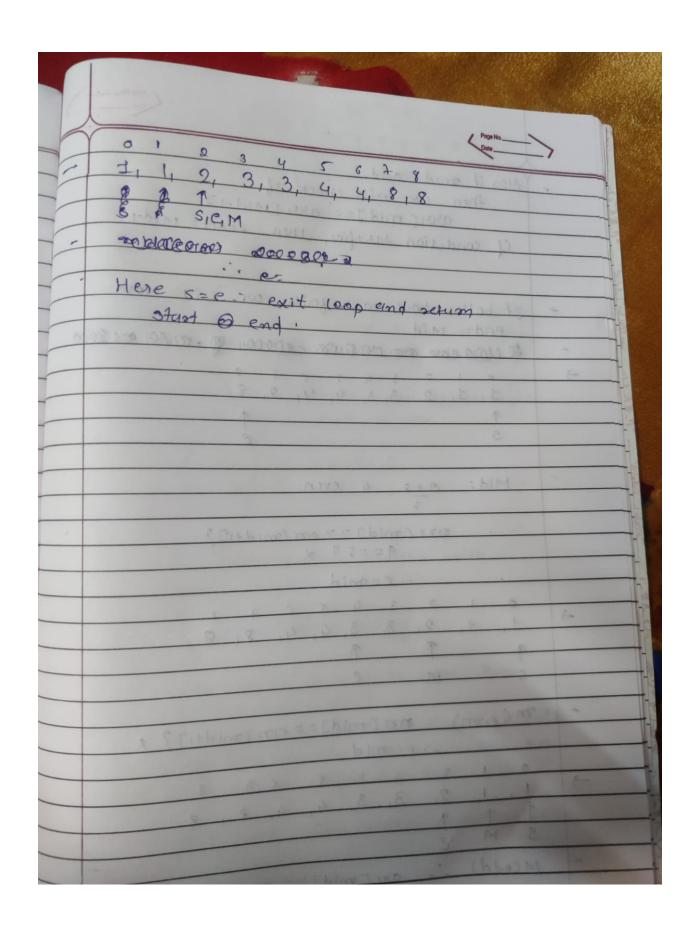
```
class Solution{
public int singleNonDuplicate(int[] nums) {
  int x=0;
  for(int i =0; i<nums.length; i++){
    x= x^nums[i];
  }
  return x;
}
</pre>
```

Time complexity : O(n)
Space complexity : O(1)

Solution 2 Binary Search







```
class Solution {
                 public int singleNonDuplicate(int[] nums) {
                                 int n = nums.length;
                int start =0;
                 int end = nums.length-1;
                 int mid =0;
                 if(nums.length == 1) return nums[0];
                 while(start<end)</pre>
                 /* why not start<=end? because it will lead to infinite loop. At last step we can
see that start, \min and end are on same index and if we make condition start<=end, then w
e will never exit the loop*/
                         {
                                  mid = start +(end-start)/2;
                                   if((mid\%2==0 \& nums[mid]==nums[mid+1] )|| (mid\%2==1 \& nums[mid]==nums[mid-1])) \\ \{ (mid\%2==0 \& nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==nums[mid]==n
                                                    start = mid+1;
                                  else{
                                                   end = mid;
                                  }
                 return nums[start]; //we can also return nums[end]
                }
}
```

Time complexity : O(logn) Space complexity : O(1)