Leetcode May Challenge DAY: 12

The target number must appear in even index position, like 0, 2, 4, ... 6, 8, ...
Because all numbers before it must appear as pairs. For example, index 0, 1, 2, 3 are before even index 4.
So even index positions are supposed to be the first number of a pair, while odd index positions are supposed to be the second number of a pair.
For example, 0-1, 2-3 are two pairs. Even index 0 and 2 are the first number while odd index 1 and 3 are the second number.

Apply Binary search to calculate mid = left + (right-left)//2
If mid is at odd index, we compare it with mid-1. (odd index positions are supposed to be the second number of a pair)
nums[mid-1]==nums[mid] means the single number must be after mid. So left = mid + 1
nums[mid-1] < nums[mid] means the single number must before mid. So right = mid -1 (let right = mid also OK)
If mid is at even index, we compare it with mid + 1 (even index positions are supposed to be the first number of a pair)
nums[mid]==nums[mid+1] means the single number must be after mid. So left = mid + 2 (mid+1 gives the same number)
nums[mid] < nums[mid+1] means the single number must be the mid or before it, so right = mid

1. Python

```
class Solution:
 def singleNonDuplicate(self, nums: List[int]) -> int:
  left, right = 0, len(nums)-1
  while left < right:
   mid = left + (right - left)//2
   if mid % 2: # Mid is at odd index
    if nums[mid] > nums[mid-1]:
     right = mid - 1 # or right = mid
    else:
     left = mid + 1
   else: # Mid is at even index
    if nums[mid] < nums[mid+1]:
     right = mid
```

```
            else:
                left = mid + 2
        return nums[left]
```

2. C++

```
class Solution {
public:
    int singleNonDuplicate(vector<int>& nums) {
        int lo=0,hi=nums.size()-1;
        while(lo<hi){
            int mid=lo+(hi-lo)/2;
            if(nums[mid]==nums[mid-1]){
                if((mid-lo-1)%2==0)
                    lo=mid+1;
                else
                    hi=mid-2;
            }
            else if(nums[mid]==nums[mid+1]){
                if((hi-mid-1)%2==0)
                    hi=mid-1;
                else
                    lo=mid+2;
            }
            else
                return nums[mid];
        }
        return nums[lo];
    }
};
```

3. JAVA

```
public int singleNonDuplicate(int[] nums) {
    int left = 0;
    int right = nums.length - 1;
    while (left < right) {
        int mid = left + (right - left) / 2;
        if ((mid - left) % 2 == 1) {
            if (nums[mid] == nums[mid - 1]) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        } else {
            if (nums[mid] == nums[mid - 1]) {
                right = mid - 2;
            } else {
                left = mid;
            }
        }
    }
    return nums[left];
}
```