Day1

704 Binary search(E)

Given an array of integers nums which is sorted in ascending order, and an integer target, write a function to search target in nums. If target exists, then return its index. Otherwise, return -1.

You must write an algorithm with **O(log n)** runtime complexity.

Example 1:

```
Input: nums = [-1,0,3,5,9,12], target = 9
Output: 4
Explanation: 9 exists in nums and its index is 4
```

Example 2:

```
Input: nums = [-1,0,3,5,9,12], target = 2
Output: -1
Explanation: 2 does not exist in nums so return -1
```

```
class Solution{
  pubilc int search(int[] nums, int target){
    //左右指针
    int left = 0;
    int right = nums.length - 1;

    //边界
    int mid = left + (right - left) / 2;
```

```
    //二分
    while(left <= right){
      mid = left + (right - left) / 2;
      if(nums[mid] == target){
        return mid;
      }else if(nums[mid] > target){
        right = mid - 1;
      }else{
        left = mid + 1;
      }
    }
    return -1;
  }
}
```

Binary Search

```
class binarySearch{
  public int solution(int[] nums, int target){
    int left = 0;
    int right = nums.length - 1;

    int mid = (l + r + 1) / 2;

    while(left <= right){
      mid = (l + r + 1) / 2;
      if(check(mid)){
        l = mid;
      } else {
        r = mid - 1;
      }
    }

    public boolean check(int num){
      if(){
```

```
            return true;
          }
          return false;
        }
      }
    }
```

Integer Binary Search(**No intersection**)

1. mid = (l + r + 1) / 2

   Check(mid):

   - True [mid, r], **l = mid**
   - False [l, mid - 1] r = mid - 1
2. mid = (l + r) / 2

   Check(mid):

   - True [l, mid], r = mid
   - False [mid + 1, r] l = mid + 1

27. Remove Element(E)

Given an integer array nums and an integer val, remove all occurrences of val in nums in-place. The relative order of the elements may be changed.

Since it is impossible to change the length of the array in some languages, you must instead have the result be placed in the first part of the array nums. More formally, if there are k elements after removing the duplicates, then the first k elements of nums should hold the final result. It does not matter what you leave beyond the first k elements.

Return k after placing the final result in the first k slots of nums.

Do not allocate extra space for another array. You must do this by modifying the input array in-place with O(1) extra memory.

Custom Judge:

The judge will test your solution with the following code:

```
int[] nums = [...]; // Input array
int val = ...; // Value to remove
int[] expectedNums = [...]; // The expected answer with
correct length.
                            // It is sorted with no
values equaling val.

int k = removeElement(nums, val); // Calls your
implementation

assert k == expectedNums.length;
sort(nums, 0, k); // Sort the first k elements of nums
for (int i = 0; i < actualLength; i++) {
    assert nums[i] == expectedNums[i];
}
```

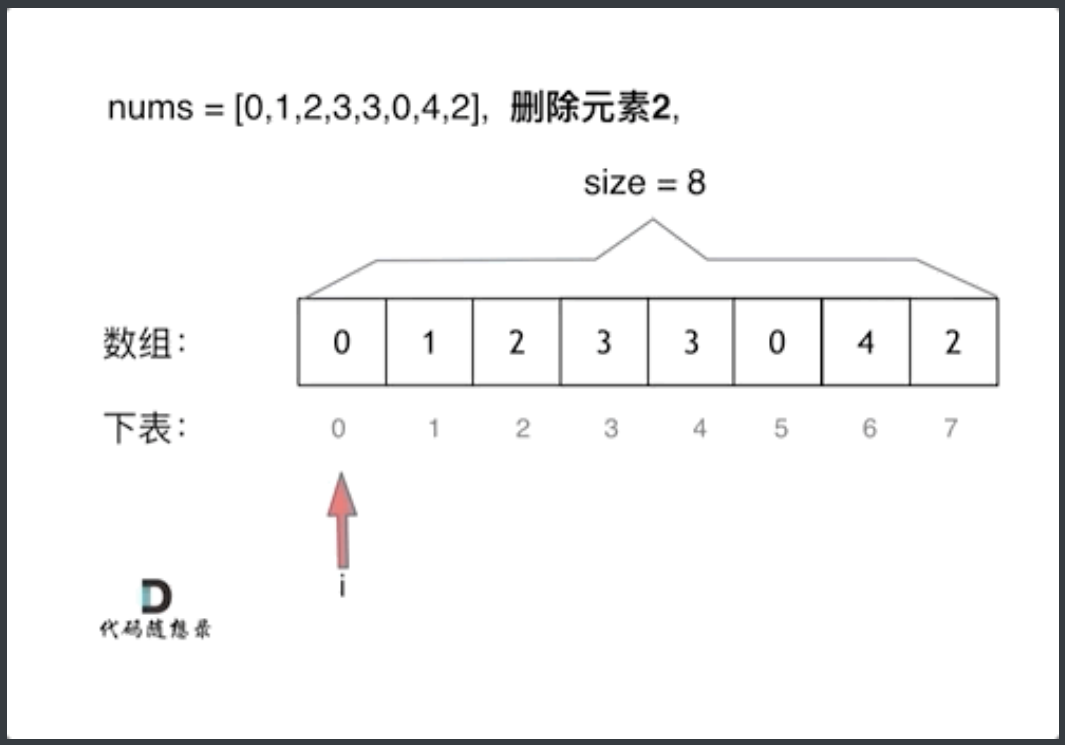
Example 1:

```
Input: nums = [3,2,2,3], val = 3
Output: 2, nums = [2,2,_,_]
Explanation: Your function should return k = 2, with the
first two elements of nums being 2.
It does not matter what you leave beyond the returned k
(hence they are underscores).
```

Example 2:

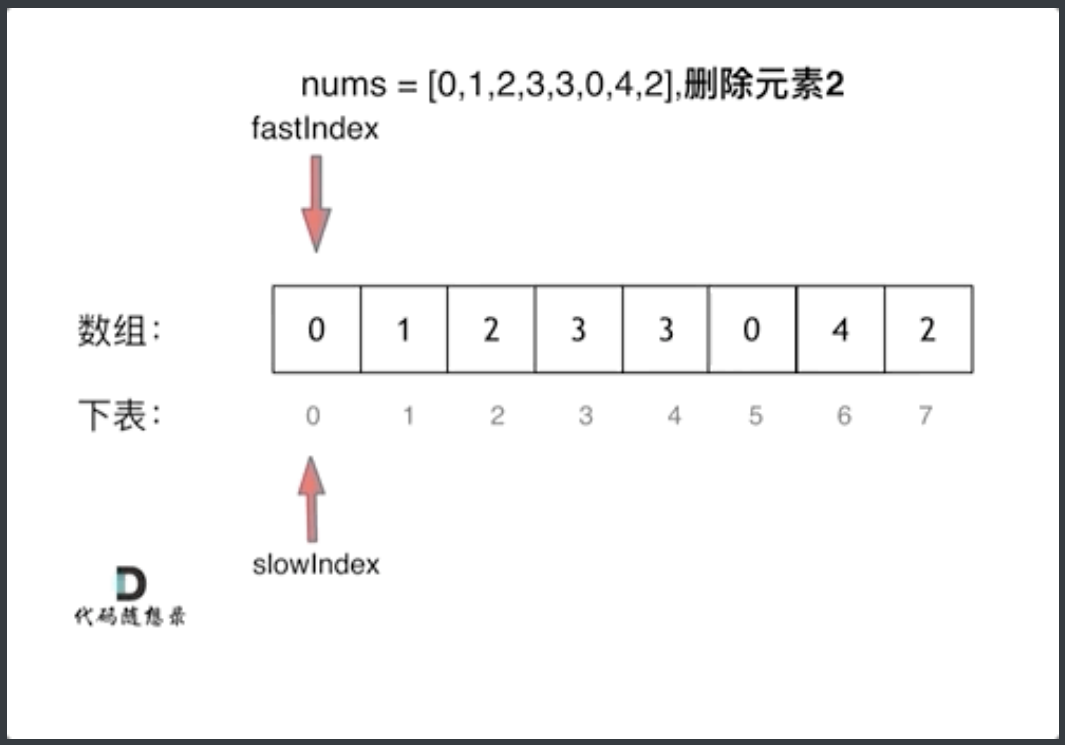
```
Input: nums = [0,1,2,2,3,0,4,2], val = 2
Output: 5, nums = [0,1,4,0,3,_,_,_]
Explanation: Your function should return k = 5, with the
first five elements of nums containing 0, 0, 1, 3, and 4.
Note that the five elements can be returned in any order.
It does not matter what you leave beyond the returned k
(hence they are underscores).
```

- **Method 1**

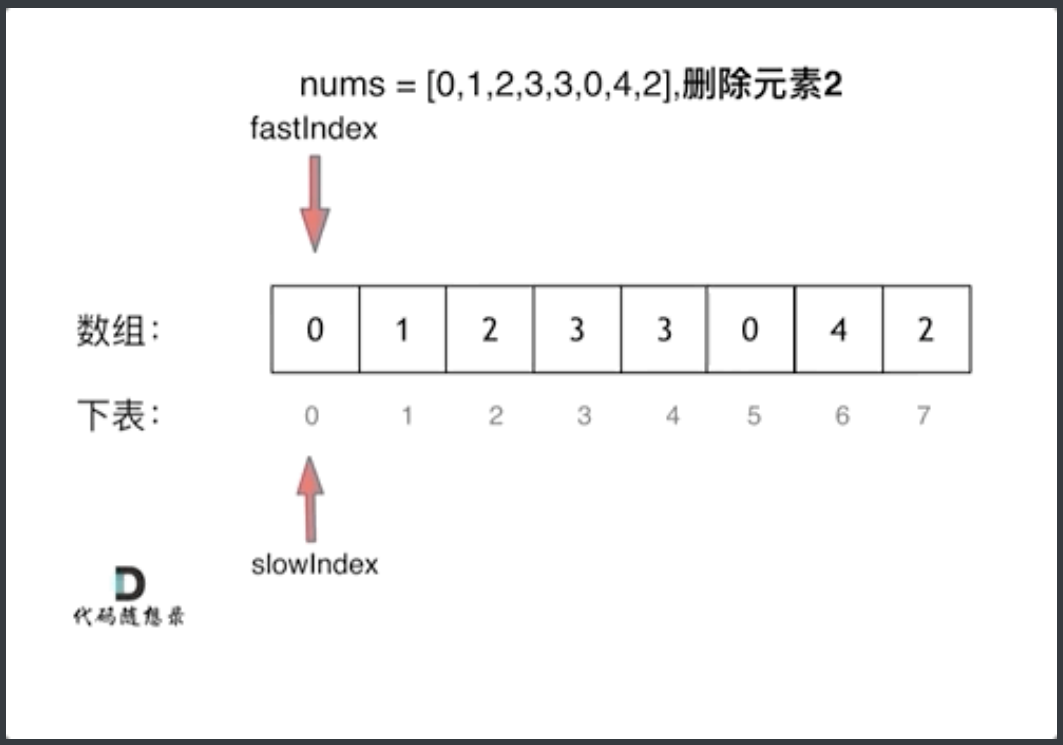



```java
//双重for循环，数组的内存地址的连续性
class solution{
  public int removeElement(int[] nums, int val){
    int size = nums.length;
    for (int i = 0; i < size; i++) {
            if(nums[i] == val){
                for (int j = i + 1; j < size; j++) {
                    nums[j - 1] = nums[j];
                }
                i--;
                size--;
            }
        }
        return size;
  }
}
```

- **Method 2**




```
class solution{
  public int removeElement(int[] nums, int val){
       int size = nums.length;
        int slowIndex = 0;
        for (int fastIndex = 0; fastIndex < size; fastIndex++) {
            if (nums[fastIndex] != val){
                nums[slowIndex] = nums[fastIndex];
                slowIndex++;
            }
        }
        return slowIndex;
  }
}
```


nums = [0,1,2,3,3,0,4,2],删除元素2
fastIndex
数组:
0 1 2 3 3 0 4 2
下表:
0 1 2 3 4 5 6 7
slowIndex
代码随想录