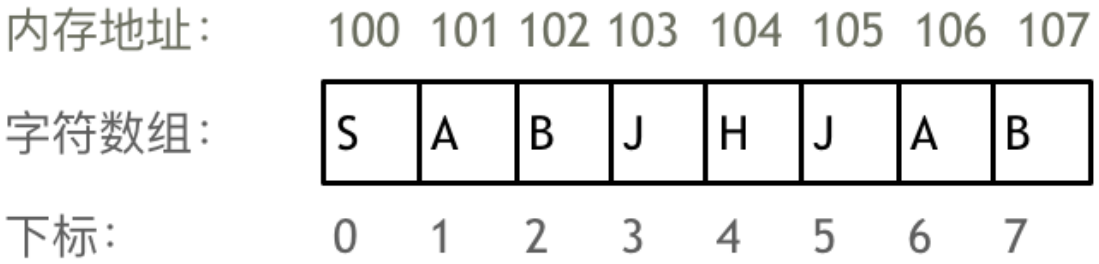
数组理论
1.数组是存放在连续内存空间上的相同类型数据的集合。
2.数组可以方便的通过下标索引的方式获取到下标对应的数据。Index从0开始。



3. 删除或者增添元素的时候，要移动其他元素的地址。

704二分查找

1. 使用前提:有序数组
2. 二分的最大优势是在于其时间复杂度是O(logn)

有序数组 nums，目标值 target，写一个函数搜索 nums 中的 target，如果目标值存在返回下标，否则返回 -1。

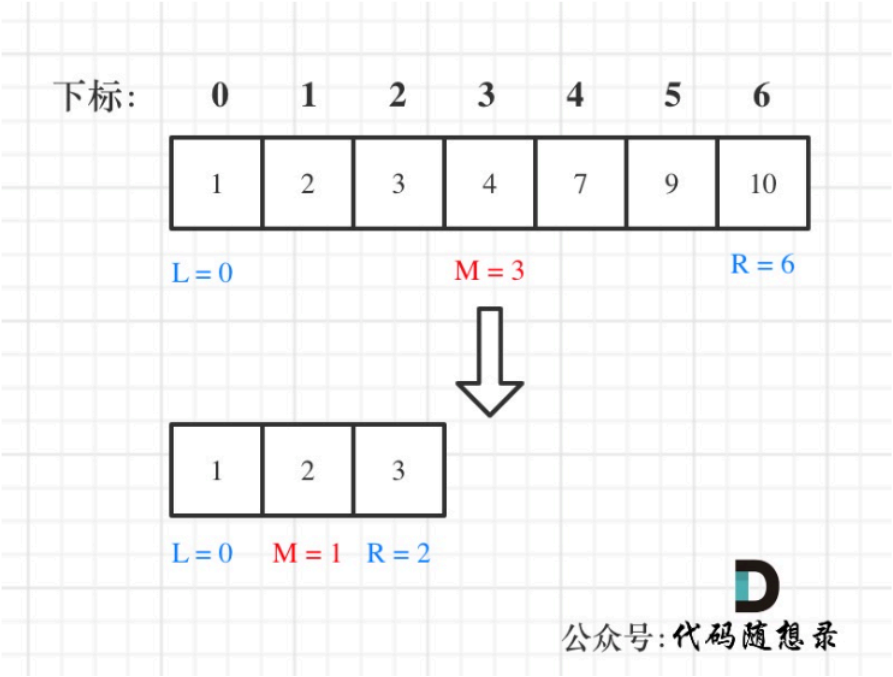
输入: nums = [-1,0,3,5,9,12], target = 9
输出: 4
解释: 9 出现在 nums 中并且下标为 4

答案:
1. 左闭右闭 [l,r]

```
def search(self, nums: List[int], target: int) -> int:
    l = 0
    r = len(nums) -1
    m = (l+r) // 2
    while l<=r:
        if target < nums[m]:
            r = m-1
        elif target > nums[m]:
            l = m+1
        else:
            return m
        m = (l+r) // 2
    return -1
```

重点:

（1）m = (l+r) // 2 用整除
（2）区间不变量：区间取左闭右闭的话 那么每次区间二分 范围都是新区间的左闭右闭 后面做判断时 要一直基于这个左闭右闭的区间。当l = 0, r = n - 1的时候因为r这个值我们在数组中可以取到,while(l <= r) 是正确写法。因为右边在范围内可以取到，所以使用 r = m-1

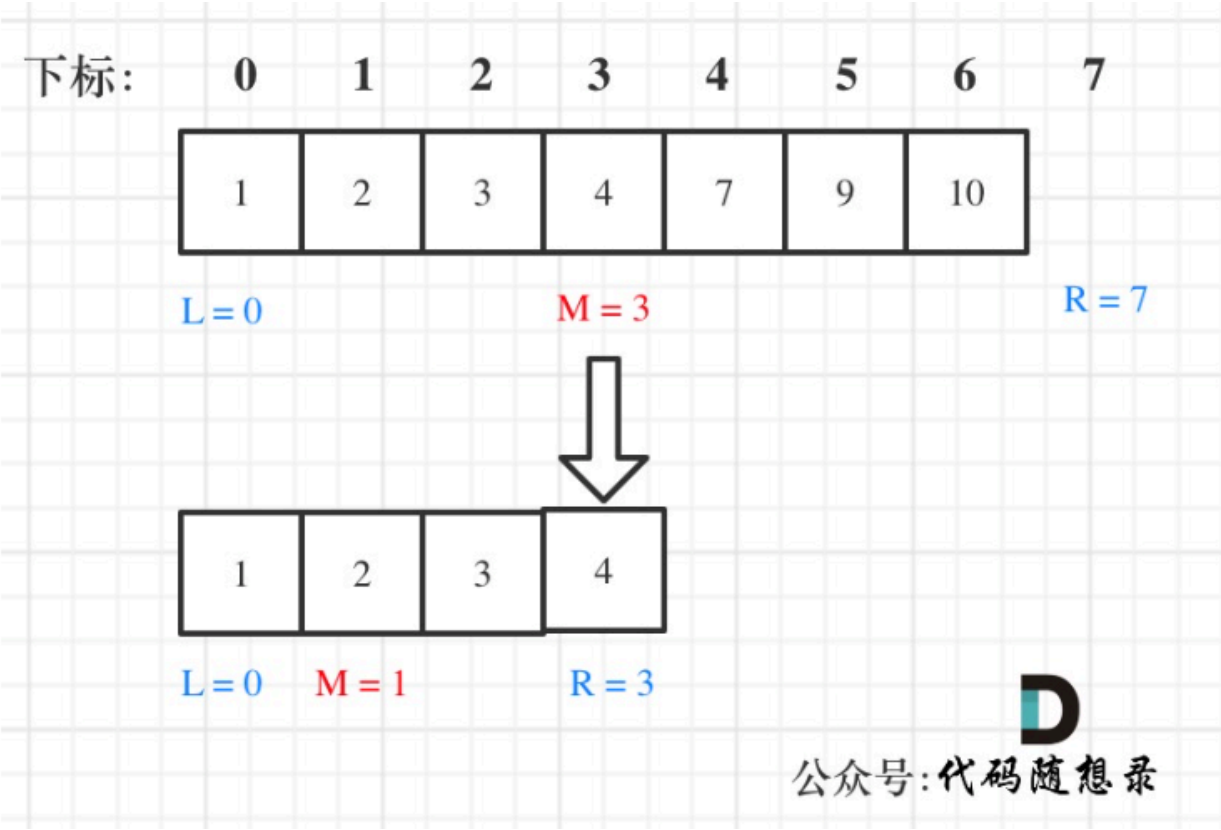


2. 左闭右开[l,r)

```
def search(self, nums: List[int], target: int) -> int:
    l = 0
    r = len(nums)
    while l < r:
        m = (l+r) // 2
        if target < nums[m]:
            r = m
        elif target > nums[m]:
            l = m+1
        else:
            return m
        m = (l+r) //2
    return -1
```

重点：

（1）区间不变量：当l = 0, r = n的时候因为r这个值我们在数组中无法取到,while(l < r) 是正确写法。因为右边取不到，所以r = m。其实区间定义成开或者闭都没有什么关系 只是要明确每次收缩范围后 范围内的元素是哪些 注意会不会漏掉边界就好。



27.移除元素

一个数组 nums 和一个值 val，[原地](#) 移除所有数值等于 val 的元素，然后返回 nums 中与 val 不同的元素的数量。

输入：nums = [3,2,2,3], val = 3
输出：2, nums = [2,2,_,_]
解释：你的函数函数应该返回 k = 2, 并且 nums 中的前两个元素均为 2。
你在返回的 k 个元素之外留下了什么并不重要（因此它们并不计入评测）。

1.双指针法

- 时间复杂度：O(n)
- 空间复杂度：O(1)

```
def removeElement(self, nums: List[int], val: int) -> int:
    index = 0
    cur = 0
    count = 0
    while cur <= len(nums) - 1:
```

```
            if nums[cur] != val:
                count += 1
                nums[index] = nums[cur]
                cur += 1
                index += 1
            else:
                cur += 1
        return count
```

977. 有序数组的平方

非递减顺序 排序的整数数组 nums，返回 每个数字的平方 组成的新数组，要求也按 非递减顺序 排序。

输入：nums = [-4,-1,0,3,10]

输出：[0,1,9,16,100]

解释：平方后，数组变为 [16,1,0,9,100]

排序后，数组变为 [0,1,9,16,100]

1.双指针法

```
    def sortedSquares(self, nums: List[int]) -> List[int]:
        l = 0
        r = len(nums) -1
        res = []
        while l <= r:
            if abs(nums[l]) >= abs(nums[r]):
                res = [nums[l] * nums[l]] + res
                l += 1
            elif abs(nums[l]) < abs(nums[r]):
                res = [nums[r] * nums[r]] + res
                r -= 1
        return res
```

重点：

当内部顺序很难整理的时候，创建新的地址来记录