

## 二分查找

```
1  # 1. 有序数组
2  def base_search(nums, target):
3      '''
4      实现一个基础二分查找
5      输入：一个顺序list
6      输出：待查找的元素的位置
7      '''
8      left, right = 0, len(nums) - 1
9      while left <= right:
10         mid = left + (right - left) // 2 # (left + right) >> 1 除2 向下取整
11         if nums[mid] == target:
12             return mid
13         elif target < nums[mid]:
14             right = mid - 1
15         elif target > nums[mid]:
16             left = mid + 1
17     return -1
18
19
20 def left_search(nums, target):
21     '''
22     存在多个相同数字
23     左边界
24     '''
25     left, right = 0, len(nums) - 1
26     while left <= right:
27         mid = (left + right) // 2
28         if nums[mid] == target:
29             right = mid - 1
30         elif nums[mid] < target:
31             left = mid + 1
32         elif nums[mid] > target:
33             right = mid - 1
34     if left == len(nums) or nums[left] != target:
35         return -1
36     return left
37
38 def right_search(nums, target):
39     '''
40     右边界
41     '''
42     left, right = 0, len(nums) - 1
43     while left <= right:
44         mid = (left + right) // 2
45         if nums[mid] == target:
46             left = mid + 1
47         elif nums[mid] < target:
48             left = mid + 1
49         elif nums[mid] > target:
50             right = mid - 1
51     if right < 0 or nums[right] != target:
52         return -1
53     return right
```

```

54
55
56 # 因为我们初始化 right = len(nums) - 1
57 # 所以决定了我们的「搜索区间」是 [left, right]
58 # 所以决定了 while (left <= right)
59 # 同时也决定了 left = mid + 1 和 right = mid - 1
60
61
62 # 2. 旋转有序数组
63 def search(nums, target):
64     '''
65         在数组中搜索 target 值
66         [2,5,6,0,0,1,2]
67     '''
68     left, right = 0, len(nums) - 1
69     while left <= right:
70         mid = (left + right) // 2
71         if nums[mid] == target: return True
72         if nums[mid] < nums[right]: # mid 在右半部分
73             if target > nums[mid] and target <= nums[right]: # target 在右半
部分
74                 left = mid + 1
75             else: # target 在左半部分
76                 right = mid - 1
77
78         elif nums[mid] > nums[right]:
79             if target < nums[mid] and target >= nums[left]:
80                 right = mid - 1
81             else:
82                 left = mid + 1
83
84         elif nums[mid] == nums[right]: # 无法判断 存在相同元素
85             right -= 1
86
87     return False

```