# 033. Search in Rotated Sorted Array

## 033 Search in Rotated Sorted Array

• Binary Search+array

### **Description**

Suppose an array sorted in ascending order is rotated at some pivot unknown to you beforehand.

```
(i.e., 0 1 2 4 5 6 7 might become 4 5 6 7 0 1 2).
```

You are given a target value to search. If found in the array return its index, otherwise return -1.

You may assume no duplicate exists in the array.

### 1. Thought line

- 1. Find pivot
- 2. Do binary search on left half;
- 3. Do binary search on right half;
- 4. Binary search processing

#### 2. Binary Search+array

```
1 class Solution {
 3 private:
      void binarySearch(vector<int>& nums, int target, int st, int ed, int& res){
        // finish process condition
         if (st>ed) return;
 6
          if (target<nums[st]||target>nums[ed]) return;
 8
 9
         // no target
10
          if (st==ed && nums[st]!=target) return;
11
12
          // find target
          if (st==ed && nums[st]==target) res = st;
13
14
15
          // keep finding process
16
          else{
17
              // middle spot in array
              int mid = (st+ed)/2;
18
19
              if (target<=nums[mid])</pre>
20
                  binarySearch(nums, target, st, mid, res);
21
22
              binarySearch(nums, target, mid+1, ed, res);
23
24
25
26 public:
27
      int search(vector<int>& nums, int target) {
28
       int res = -1;
29
          int pivot = 0:
30
          if (nums.empty()) return -1;
          // find pivot
31
          for (int i = 1; !nums.empty() && i<=nums.size()-1; ++i){
32
33
              if (nums[i-1]>nums[i]){
34
                  pivot = i;
35
                   break;
```

```
37 }
38 // process binary search on left half
39 binarySearch(nums, target, 0, pivot-1, res);
40 // process binary search on right half
41 binarySearch(nums, target, pivot, nums.size()-1, res);
42 return res;
43 }
44 };
```