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.

分析：示例中可能的排序组合有以下几种情况：

0 1 2 3 4 5 6 7

7 0 1 2 3 4 5 6

6 7 0 1 2 3 4 5

5 6 7 0 1 2 3 4

4 5 6 7 0 1 2 3

3 4 5 6 7 0 1 2

2 3 4 5 6 7 0 1

1 0 2 3 4 5 6 7

分析：

用二分查找需要解决的问题是确认每一步是从middle位置的左侧找还是右侧找。

如上图所示，如果middle位置的数小于最右边的数，则右侧是有序的，这时判断（middle位置数<target && 最右位置数>=target）是否成立，如果成立则从右侧找，否则从左侧找。

如果middle位置的数大于最右边的数，则左侧是有序的，这时判断（middle位置数>target && 最左位置数<=target）是否成立，如果成立则从左侧找，否则从有侧找。

class Solution {

public:

int search(vector<int>& nums, int target) {

if (nums.size() == 0) {

return -1;

}

int low = 0;

int high = nums.size() - 1;

while (low <= high) {

int middle = low + (high - low) / 2;

if (nums[middle] == target) {

return middle;

} else if (nums[middle] < nums[high]) {

if (nums[middle] < target && nums[high] >= target)

low = middle + 1;

else

high = middle - 1;

} else {

if (nums[middle] > target && nums[low] <= target)

high = middle - 1;

else

low = middle + 1;

}

}

return -1;

}

};