思路：

1. 确定目标元素和有序数组。
2. 初始化搜索范围为整个数组。
3. 比较目标元素与中间元素。
    - 如果相等，返回中间元素的索引，查找成功。
    - 如果目标元素小于中间元素，缩小搜索范围为左半部分。
    - 如果目标元素大于中间元素，缩小搜索范围为右半部分。
4. 重复步骤2和步骤3，直到找到目标元素或搜索范围为空。查找成功返回目标元素的索引，否则返回-1。

这个算法的时间复杂度为 O(log n)，适用于有序数组的快速查找操作。

不使用递归实现(while)：

```
    public int search(int[] nums, int target) {
        int left = 0;
        int right = nums.length - 1;
        while(left <= right) {
            int mid = left + (right - left) / 2;
            if(nums[mid] > target) {
                right = mid - 1;
            } else if(target > nums[mid]) {
                left = mid + 1;
            } else if(target == nums[mid]) {
                return mid;
            }
        }
        return -1;
    }
}
```

递归实现

```
    public static int binarySearch(int[] arr, int target, int left, int
right) {
        if( arr[left] > target || arr[right] < target || left > right ) {
            return -1;
        }
        int mid = left + ( right - left ) / 2;
        if (arr[mid] == target) {
            return mid;
        } else if (arr[mid] > target) {
            return binarySearch(arr, target, left, right - 1);
        } else if (target > arr[mid]) {
            return binarySearch(arr, target, left + 1, right);
        }
        return -1;
    }
}
```