- 二分查找核心代码技巧
  分析二分查找的一个技巧是：不要出现 else，而是把所有情况用 else if 写清楚，这样可以清楚地展现所有细节。

左闭右闭区间代码

```
int binarySearch(int[] nums, int target) {
    int left = 0;
    int right = nums.length - 1; // 注意

    while(left <= right) {
        int mid = left + (right - left) / 2;
        if(nums[mid] == target)
            return mid;
        else if (nums[mid] < target)
            left = mid + 1; // 注意
        else if (nums[mid] > target)
            right = mid - 1; // 注意
    }
    return -1;
}
```

寻找左侧边界的二分搜索(左闭右开)

```
int left_bound(int[] nums, int target) {
    if (nums.length == 0) return -1;
    int left = 0;
    int right = nums.length; // 注意

    while (left < right) { // 注意
        int mid = left + (right - left) / 2;
        if (nums[mid] == target) {
            right = mid;
        } else if (nums[mid] < target) {
            left = mid + 1;
        } else if (nums[mid] > target) { //缩小区间（上限下沉）
            right = mid; // 注意
        }
    }
    return left;
    /*
     * if (left >= nums.size() || nums[left] != target) return -1;
     * return left;
     */
}
```

寻找右边界的二分搜索

```
int right_bound(int[] nums, int target) {
    if (nums.length == 0) return -1;
    int left = 0, right = nums.length;

    while (left < right) {
        int mid = left + (right - left) / 2;
        if (nums[mid] == target) { //缩小区间（下限上升）
            left = mid + 1; // 注意
            //mid = left - 1 返回相当于撤销不合格的一步
        } else if (nums[mid] < target) {
            left = mid + 1;
        } else if (nums[mid] > target) {
            right = mid;
        }
    }
    return left - 1; // 注意
    /*
     * if (left == 0) return -1;
     * return nums[left - 1] == target ? (left - 1) : -1;
     */
}
```

二分搜索题目泛化

- 从题目中抽象出一个自变量 `x`，一个关于 `x` 的函数 `f(x)`，以及一个目标值 `target`。

1. `f(x)` 是 `x` 上的单调函数.
2. 计算满足 `f(x) == target` 的值.

- 如何选取 **左右边界值** 也是一个需要思考的问题.

大致框架

```java
// 函数 f 是关于自变量 x 的单调函数
int f(int x) {
    // ...
}

// 主函数，在 f(x) == target 的约束下求 x 的最值
int solution(int[] nums, int target) {
    if (nums.length == 0) return -1;
    // 问自己：自变量 x 的最小值是多少？
    int left = ...;
    // 问自己：自变量 x 的最大值是多少？
    int right = ... + 1;

    while (left < right) {
        int mid = left + (right - left) / 2;
        if (f(mid) == target) {
            // 问自己：题目是求左边界还是右边界？
            // ...
        } else if (f(mid) < target) {
            // 问自己：怎么让 f(x) 大一点？
            // ...
        } else if (f(mid) > target) {
            // 问自己：怎么让 f(x) 小一点？
            // ...
        }
    }
    return left;
}
```

习题

- **704 二分查找**

```cpp
class Solution {
public:
    //左开右闭解法
    int search(vector<int>& nums, int target) {
        int left = 0;
        int right = nums.size();
        while (left < right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] == target)
                return mid;
            else if (nums[mid] < target)
                left = mid + 1;
            else if (nums[mid] > target)
                right = mid;
        }
        return -1;
    }
};
```

- **34 在排序数组中查找元素的第一个和最后一个位置**

```cpp
class Solution {
public:
    vector<int> searchRange(vector<int>& nums, int target) {
        int left = 0;
        int right = nums.size();
        vector<int> result(2, -1);
        //左边界搜索
        while (left < right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] == target)
                right = mid;
            else if (nums[mid] < target)
                left = mid + 1;
            else if (nums[mid] > target)
                right = mid;
        }
        if (left == nums.size() || nums[left] != target)
            result[0] = -1;
        else
            result[0] = left;

        left = 0;
        right = nums.size();
        //右边界搜索
        while (left < right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] == target)
                left = mid + 1;
            else if (nums[mid] < target)
                left = mid + 1;
            else if (nums[mid] > target)
                right = mid;
        }
        if (left == 0 || nums[left - 1] != target)
            result[1] = -1;
        else
            result[1] = left - 1;
        return result;
    }
};
```

- **山峰数组的顶部**

```
class Solution {
public:
    /* 二分优化解法
     * 如果目前是递增则山峰在后面(case1)
     * 如果目前是递减则山峰在前面(case2)
     */
    int peakIndexInMountainArray(vector<int>& arr) {
        int left = 1, right = arr.size() - 2;
        int result = 0;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (arr[mid] > arr[mid + 1]) { //case2
                result = mid;
                right = mid - 1;
            }
            else {                          //case1
                left = mid + 1;
            }
        }
        return result;
    }
};
```

- 解压7z

```
brew search 7z
brew install p7zip
7z e filename.7z
```

- 解压rar

```
brew install unrar
unrar x rar文件 # x参数 解压后所有内容都放在一个同名的文件夹里
```