4.二分

一些参考文章

- 极客时间
- https://blog.csdn.net/qq_41221520/article/details/108277801

两种模板、几点区分

- 两种情况

```
闭区间
<=
mid + 1, mid - 1;

int binarySearch(vector<int> nums, int target){
    //闭区间[]
    int left = 0;
    int right = nums.size() - 1;

    // <=
    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 - 1;
    }
    return -1
}
```

```
开区间
<
第一个mid, 第二个mid + 1/ mid - 1

int binarySearch(vector<int> nums, int target){
    int left = 0;
    int right = nums.size();
    while(left < right){
        int mid = mid + (right - left) / 2;
        if(nums[mid] == target) return mid;
        else if(nums[mid] < target) left = mid;
        else if(nums[mid] > target) right = mid - 1;

        else if(nums[mid] > target) right = mid;
        else if(nums[mid] < target) left = mid + 1;
    }
}
```

二分查找(找一个确定的值)

基本

```
int binarySearch(vector<int> nums, int target){
    int l = 0;
    int r = nums.size() - 1;
    while(l <= r){
        int mid = l + (r - l) / 2;
        if(nums[mid] == target){
            return ans;
        }else if(mid > target) {
            r = mid - 1;
        }else if{
            l = mid + 1;
        }
    }
    return -1;
}
```

左边界

```
int binarySearch(vector<int> nums, int target){
    int ans = -1;
    int l = 0;
    int r = nums.size() - 1;
    while(l <= r){
        int mid = l + (r - l) / 2;
        if(nums[mid] == target){
            r = mid - 1;
            ans = mid;
        }else if(nums[mid] > target) {
            r = mid - 1;
        }else if(nums[mid] < target){
            l = mid + 1;
        }
    }
    return ans;
}
```

右边界

```
int binarySearch(vector<int> nums, int target){
    int ans = -1;
    int l = 0;
    int r = nums.size() - 1;
    while(l <= r){
        int mid = l + (r - l) / 2;
        if(nums[mid] == target){
            l = mid + 1;
            ans = mid;
        }else if(nums[mid] > target) {
            r = mid - 1;
        }else if(nums[mid] < target){
            l = mid + 1;
```

```
        }
    }
    return ans;
}
```

STL中的模板库

```
//第一个位置
lower_bound(a.begin(), a.end(), target) - a.begin();
//找第一个大于等于target的位置
upper_bound(a.begin(), a.end(), target) - a.begin();
//找第一个大于target的位置


//最后一个位置
lower_bound(a.begin(), a.end(), target + 1) - a.begin();
//找第一个大于等于target+1的位置，然后-1

在从小到大的排序数组中，

lower_bound( begin,end,num)：从数组的begin位置到end-1位置二分查找第一个大于或等于num的

upper_bound( begin,end,num)：从数组的begin位置到end-1位置二分查找第一个大于num的数字，

在从大到小的排序数组中，重载lower_bound()和upper_bound()

lower_bound( begin,end,num,greater<type>() ):从数组的begin位置到end-1位置二分查找第-

upper_bound( begin,end,num,greater<type>() ):从数组的begin位置到end-1位置二分查找第-
```

二分答案(找一个可能满足的值)

```
//插入的位置、第一个>=atrget的位置
int searchInsert(vector<int>& nums, int target) {
    int n = nums.size();
    int left = 0, right = n - 1;
    while(left <= right){
        int mid = (right - left) / 2 + left;
        if(nums[mid] == target) return mid;
        else if(nums[mid] > target){
            right = mid - 1;
        }
        else{
            left = mid + 1;
        }
    }
    return left;
    //return right + 1（退出条件是left = right + 1）
    //如果while(left < right) return left/right都可以
    //>=插入位置时候返回left，像69题向下取整意思时候返回right
    }
```

```cpp
//某种要具体check里面算的逻辑
int binarySearch(vector<int> nums, int target){
    int l = 0;
    int r = nums.size() - 1;
    int ans = 0;
    while(l <= r){
        int mid = l + (r - l) / 2;
        if(!check()){
            l = mid + 1;
        }
        else if(check()){
            ans = mid;
            r = mid  - 1;
        }
    }
    return ans;
}
```

二分查找题目

- 剑指 Offer II 069. 山峰数组的顶部
- 240. 搜索二维矩阵 II

二分答案题目 275、69、410、441

- 475. 供暖器
- 剑指 Offer II 068. 查找插入位置
- 剑指 Offer II 071. 按权重生成随机数

二分题目

33. 搜索旋转排序数组

153. 寻找旋转排序数组中的最小值

162. 寻找峰值