备忘-必备算法

- 一些必备算法，主要是 C++ 版本 Index

二分查找

离散版

my_binary_search(vector<int>, int)

- 没有重复元素时，目标值若存在，则返回索引；若不存在，返回 -1
- 存在重复元素时，目标值若存在，则返回最小索引；若不存在，返回 -1

```
int my_binary_search(vector<int>& nums, int v) {
    if (nums.size() < 1) return - 1;

    int lo = -1, hi = nums.size();  // hi = nums.size() - 1

    while (hi - lo > 1) {
        int mid = lo + (hi - lo) / 2;
        if (nums[mid] < v)
            lo = mid;
        else
            hi = mid;
    }

    return nums[lo + 1] == v ? lo + 1 : -1;
}
```

my_lower_bound(vector<int>, int)

- 返回大于、等于目标值的最小索引（第一个大于或等于目标值的索引）

```
int my_lower_bound(vector<int>& nums, int v) {
    if (nums.size() < 1) return -1;

    int lo = -1, hi = nums.size();  // hi = nums.size() - 1
```

```
    while (hi - lo > 1) {                       // 退出循环时有：lo + 1 == hi
        int mid = lo + (hi - lo) / 2;
        if (nums[mid] < v)
            lo = mid;                           // 因为始终将 lo 端当做开区间，所以没有必要
        else
            hi = mid;                           // 而在 else 中，mid 可能就是最后的结果，所
    }

    return lo + 1; // 相比 binary_search，只有返回值不同
}
```

- **为什么返回 `lo + 1`？**
  - 模板开始时将 (lo, hi) 看做是一个开区间，通过不断二分，最终这个区间中只会含有一个值，即 (lo, hi]
  - 返回 lo+1 的含义是，结果就在 lo 的下一个；
  - 在迭代的过程中，hi 会从开区间变为闭区间，而 lo 始终是开区间，返回 lo+1 显得更加统一。
  - 当然，这跟迭代的写法是相关的，你也可以使最终的结果区间是 [lo, hi)，这取决于个人习惯。

`my_upper_bound(vector<int>, int)`

- 返回大于目标值的最小索引（第一个大于目标值的索引）

```
int my_upper_bound(vector<int>& nums, int v) {
    if (nums.size() < 1) return -1;

    int lo = -1, hi = nums.size();  // hi = nums.size() - 1

    while (hi - lo > 1) {
        int mid = lo + (hi - lo) / 2;

        if (nums[mid] <= v)                     // 相比 lower_bound，唯一不同点：`<` -> `
            lo = mid;
        else
            hi = mid;
    }

    return lo + 1;
}
```

排序

堆排序