基础算法——二分法

二分查找的各种题型总结

https://blog.csdn.net/a1097304791/category_8008146.html

https://blog.csdn.net/queque_heiya/category_9692068.html

https://www.cnblogs.com/wuyuegb2312/archive/2013/05/26/3090369.html

https://www.cnblogs.com/grandyang/p/6854825.html 很好的总结，下面的题目也要做。

一些细节：如中间位置按照形式等价可以写成`int middle = (left + right) / 2`，这样可能存在溢出的问题，比如`right`是`int`类型的最大值，目标值恰好比最大值少1，第一次循环不会溢出，但是第二次循环就会溢出。利用位运算是加速考虑。

思考：容易出错的地方：初始边界怎么写，循环退出条件怎么写不容易出错？如果数组是降序排列的又该怎么办？

有序数组查找，按维数分为两大类，一维查找和二维查找，查找类型：

第一类： 需查找和目标值完全相等的数

```
int binarySearch(vector<int>& nums, int target)
{
    int n = nums.size();
    if (n == 0 || target < nums[0] || target > nums.back())
return -1;
    int left = 0, right = n - 1;
    while (left <= right) {
        int middle = left + ((right - left) >> 1);
        if (nums[middle] == target) return middle;
        else if (nums[middle] < target) left = middle + 1;
        else right = middle - 1;
```

```
    }

    return -1;
}
```

可能的变形是数组的某部分顺序颠倒，更复杂就是存在重复元素。

典型应用：

- [x] 349.Intersection of Two Arrays
- [x] 33.search in rotated array
- [x] 81.search in rotated array 2
- [x] 704.Binary Search
- [x] 367.Valid Perfect Square
- [x] leetcode 69 Sqrt(x)
- [x] 数组中数值和下标相等的元素

假设一个单调递增的数组里的每个元素都是整数并且是唯一的。请编程实现一个函数找出数组中任意一个数值等于其下标的元素。例如，在数组{-3, -1, 1, 3, 5}中，数字3和它的下标相等

输入：第一行是case的个数n，接下来n行，每行第一个数是数组里元素的个数m，后面跟m个数

```
4
5 -3 -1 1 3 5
1 0
2 0 2
2 -1 1
```

```
#include <vector>
#include <iostream>
#include <algorithm>
```

```cpp

using namespace std;

int findEquall(vector<int> & nums) {
    int left = 0, right = nums.size() - 1;
    while (left <= right) {
        int middle = left + ((right - left) >> 1);
        if (nums[middle] == middle) return middle;
        else if (middle < nums[middle]) right = middle - 1;
        else left = middle + 1;
    }

    return -1;
}


int main()
{
    int tmp, caseNum, n;

    cin >> caseNum;
    while (caseNum--) {
        vector<int> nums;
        cin >> n;
        while (n--) {
            cin >> tmp;
            nums.push_back(tmp);
        }
        cout << findEquall(nums) << endl;
    }

    return 0;
}
```

当然更完善的做法是考虑没找到的情况就返回-1。

标准库里有算法`bool binary_search(v.begin(), v.end(), target)`

第二类： 查找第一个不小于目标值的数，可变形为查找最后一个小于目标值的数

类似于标准库的`lower_bound()`和`upper_bound()`

典型应用：

- 35.search insert position
- 34.find first and last position of element in sorted array（等价于找元素在数组内出现的次数或范围）

```cpp
//leetcode 35
class Solution {
public:
    int searchInsert(vector<int>& nums, int target) {
        int left = 0, right = nums.size();
        while (left < right) {
            int mid = left + ((right - left) >> 1);
            if (nums[mid] < target) left = mid + 1;
            else right = mid;
        }

        return left;
    }
};
```

查找最后一个小于目标值的数，只需要将找到第一个不小于目标值的位置向前移动一个位置即可。

第三类： 查找第一个大于目标值的数，可变形为查找最后一个不大于目标值的数

典型题目：

- leetcode 668 Kth Smallest Number in Multiplication Table

```
class Solution {
public:
    int searchInsert(vector<int>& nums, int target) {
        int left = 0, right = nums.size();
        while (left < right) {
            int mid = left + ((right - left) >> 1);
            if (nums[mid] <= target) left = mid + 1;
            else right = mid;
        }

        return left;
    }
};
```