算法-二分

ZeitHaum

2023 年 3 月 1 日

目录

Stop leaning useless algorithm, go and solve some problems, learn how to use binary search.——Um_nik

1 通用模板

标准的二分模板至关重要。可以二分查找的数组需要满足:

1. 数组前面的元素都符合性质 1。
2. 数组后面的元素都符合性质 2。
3. 性质 1 和性质 2 是互斥对立的，一个元素要么属于性质 1，要么属于性质 2。

定义 l 为符合性质 1 的最右侧元素的索引，r 为符合性质 2 的最左侧元素的索引。显然有 $r-l=1$.

编程时需要将 l 赋值为左边第一个满足性质 1 的索引，r 为右边最后一个满足性质 2 的索引。如果不存在则允许越界。即默认向数组最左侧和最右侧分别添加一个满足性质 1 和性质 2 的元素。

此时定义中间值 $mid=l+(r-l)//2$，此时便不会出现 $mid=l$ 或 $mid=r$ 无法停止循环的问题。典型模板如下:

```cpp
#include <bits/stdc++.h>
using namespace std;

int data[5] {1,3,5,6,8};

auto binary_search(int n,int target){
    //二分查找数组第一个大于等于target的位置。
    //性质1: 小于target.
    //性质2: 大于等于target.
    int l = -1;
    int r = n;
    while(r - l>1){
        int mid = l+(r - l)/2;
        auto check = [&](){
            if(data[mid]<target) return 1; //满足性质1
            else return 2;
        };
```

```
        if(check()==1) l = mid;
        else r = mid;
    }
    return r;
}

int main(){
    auto p = binary_search(5,6);
    cout<<p;//data[3] = 6,输出3.
}
```

2 C++ 相关库函数

2.1 lower_bound

返回第一个大于等于 target 的元素索引 (迭代器)。参数列表:

```
    lower_bound(first,end,target)->iterator;
```

2.2 upper_bound

返回第一个大于 target 的元素索引 (迭代器)。参数列表:

```
    upper_bound(first,end,target)->iterator;
```

2.3 binary_search

返回元素是否在指定范围中。参数列表:

```
    binary_search(first,end,target)->bool;
```

3 相关题目解析

3.1 力扣 287-寻找重复数

链接:寻找重复数。

此题较为巧妙，因为二分的集合不是给定数组而是给定范围区间。对于处于 [1,n] 的整数 i, 记函数 $f(i)$ 为数组中小于等于 i 的元素个数，可以发现 $f(i)$ 满足二分性。复杂度 $\Theta(n\log(n))$.

代码:

```
class Solution {
public:
    int findDuplicate(vector<int>& nums) {
        int n = nums.size()-1;
        int l = 0;
        int r = n;
        while(r-l>1){
            int mid = l + (r - l)/2;
            auto check = [&](){
                int cnt = 0;
                for(int i = 0;i<n+1;i++){
                    if(nums[i]<=mid) cnt++;
                }
                if(cnt<=mid) return 1;
                else return 2;
            };
            if(check()==1) l = mid;
            else r = mid;
        }
        return r;
    }
};
```

参考资料

[1]. github-Competitive Programming

[2]. C++ 二分查找库函数 lower_bound, upper_bound, binary_search 的简单使用

[3]. 力扣