算法-搜索

ZeitHaum

2023 年 6 月 28 日

目录

1 二分搜索

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

1.1 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.
}
```

1.2 C++ 相关库函数

1.2.1 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;
```

1.2.2 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;
```

1.2.3 binary_search

返回元素是否在指定范围中。参数列表:

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

1.3 相关题目解析

1.3.1 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;
    }
};
```

1.3.2 力扣 300-最长上升子序列 (LIS) 问题

典型做法是 $\Theta(n^2)$ 的 DP，根据 Dilworth 定理可以提出 $\Theta(n\log(n))$ 的算法。Dilworth 定理的可推导出: 一个数组的最长严格上升子序列长度等于最短的非严格下降子序列的划分。于是可以通过简单模拟非严格下降子序列的划分做法 (贪心 + 二分) 求得。

代码:

```
class Solution {
public:
```

```
3       int lengthOfLIS(vector<int>& nums) {
4           int n = nums.size();
5           vector<int>dp(n,0);
6           dp[0] = 1;
7           int ans = 1;
8           for(int i = 1;i<n;i++){
9               int maxlen = 0;
10              for(int j = 0;j<i;j++){
11                  if(nums[j]<nums[i]) maxlen = max(maxlen,dp[j]);
12              }
13              dp[i] = maxlen+1;
14              ans = max(dp[i],ans);
15          }
16          return ans;
17      }
18  };
```

1.4 概念区别

严格递增是不包含相等的递增。非严格递增等价于非递减，包含相等的情况。

相反的概念类似。

2 反对角线搜索

2.1 一般概述

对于二元函数 $f(x,y)$, 假设其值域为 $[1,n]\times[1,m](x,y,n,m\in\mathbb{R})$。

设 (i,j) 为当前搜索值，

若存在一个关于 (i,j) 的判定函数 $Q(i,j)$ 使得: 当 $Q(i,j)=1$ 时, 对于任意 $x<i$, 均有 $f(i,j)$ 优于 $f(x,j)$; 当 $Q(i,j)=2$ 时, 对于任意 $y>j$, 均有 $f(i,j)$ 优于 $f(f(i,y))$;

则可以使用反对角线进行搜索。

定义" 优于": 对于 (i_1,j_1) 和 (i_2,j_2)，若二者都属于解空间，则解一定不会出现在 (i_2,j_2)，则称 (i_1,j_1) 优于 (i_2,j_2).

Problem: 是否存在对称的条件使得搜索方向为主对角线?

2.2 复杂度分析

对于搜索空间 $[a,n]\times[b,m]$: 若 $Q(a,m)=1$, 则搜索空间变为 $(a,m)\cup[a+1,n]\times[b,m]$.

若 $Q(a,m)=2$, 则搜索空间变为 $(a,m)\cup[a,n]\times[b,m-1]$.

迭代直到解空间为 1.

复杂度 $O(n+m)$.

2.3 例题

2.3.1 力扣 11

$Q(i,j)=h_i>h_j$, 可证满足条件。

```
class Solution {
public:
    int maxArea(vector<int>& height) {
        int l = 0;
        int r = height.size() - 1;
        int maxas = 0;
        while(l<=r){
            maxas = max(maxas, (r - l) * min(height[l],height[r
                ]));
            if(height[r] > height[l]){
                l++;
            }
            else{
                r --;
            }
        }
        return maxas;
    }
};
```

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

[4]. 单调函数