C++方向编程题答案

第七周

day46

771发邮件

链接：https://www.nowcoder.com/questionTerminal/95e35e7f6ad34821bc2958e37c08918b

【题目解析】

经典的装错信封问题，该题目的核心在于递归的思想，具体参见解题思路。

【解题思路】

用A、B、C……表示写着n位友人名字的信封，a、b、c……表示n份相应的写好的信纸。

把错装的总数为记作Der(n)。假设把**a错装进B里了**，包含着这个错误的一切错装法分两类：

1. **b装入A里**，这时每种错装的**其余部分都与A、B、a、b无关，应有Der(n - 2)种错装法。**
2. **b装入A、B之外的一个信封**，这时的**装信工作实际是把（除a之外的）n - 1份信纸b、 c ……装入（除B以外的）n - 1个信封A、C……，显然这时装错的方法有Der(n - 1)种。**

总之在**a装入B的错误之下，共有错装法Der(n - 2) + Der(n - 1)种。**

a装入C，装入Der……的n - 2种错误之下，同样都有Der(n - 1) + Der(n - 2)种错装法，因此Der(n) = (n - 1)[Der(n - 1) + Der(n - 2)]

```
#include<stdio.h>
int main (void)
{
    long long der[ 21 ] = { 0, 0, 1 };
    int i;
    for ( i = 3; i < 21; i++ ){
        der[ i ] = ( i - 1 ) * ( der[ i - 2] + der[ i - 1 ] );
    }

    int n;
    while ( scanf( "%d", &n ) != EOF ){
        printf("%lld\n", der[ n ] );
    }
    return 0;
}
```

805 最长上升子序列

https://www.nowcoder.com/questionTerminal/d83721575bd4418eae76c916483493de

【题目解析】

在一个序列中找最长递增子序列，动态规划的典型应用，详细见解题思路。

【解题思路】

动态规划的难点在于定义数组和创建“状态转移方程”。

1. 定义height来存储数据，**f[i]为以height[i]结尾的元素的最长上升子序列元素个数**，初始时将f所有内容全部初始化成1，因为子序列中至少包含一个元素。
2. 定义"状态转移方程"

   一开始先将f中的数据全部置为1，因为最小的子序列长度为1

   然后**对于每个height[i]，通过遍历height[0]~~height[i-1]之间的数据，如果在该区间中找到一个height[j]比height[j]小的元素，开始比较f[j]+1和f[i]的大小，如果f[j]+1>f[j]则更新f[i]**,因此：

   - 当height[i] > height[j]: **f[i] = max(f[i], f[j]+1)**
   - 当height[i] <= height[j]：继续取下一个数据

```
#include <iostream>
#include <vector>
using namespace std;
int main(){
    int n;
    while(cin >> n){
        // 接受用户输入的数据
        vector<int> height(n, 0);
        for(int i = 0; i < n; i ++){
            cin >> height[i];
        }

        // f用来保存状态转移方程的结果，f[i]表示以height[i]结尾的最长上升子序列所包含元素的个数
        vector<int> f(n, 1);

        int result = 1;
        // 子序列中的数据一个一个增加
        for(int i = 1; i < n; i ++){
            // 从0开始统计到i位置，最长上升子序列长度
            for(int j = 0; j < i; j ++){
                if(height[i] > height[j]){
                    f[i] = max(f[i], f[j] + 1);
                }
            }
            // 获取从0到i位置的最长子序列长度
            result = max(result, f[i]);
        }

        cout << result << endl;
    }
}
```