2024.10.20

2道递归模板 + 1道模拟栈

tips:UB(Undifined Behavior) 未定义行为 指程序的行为在语言标准中没有明确定义，因此可以表现为任何结果。

这意味着当程序出现未定义行为时，它可能会产生不可预测的结果，包括程序崩溃、数据损坏、安全漏洞，甚至可能看起来正常运行。

未定义行为是C语言中一个重要的概念，因为它涉及到程序的正确性和安全性。

1.luoguP1605.

题目描述

给定一个 $N \times M$ 方格的迷宫，迷宫里有 T 处障碍，障碍处不可通过。

在迷宫中移动有上下左右四种方式，每次只能移动一个方格。数据保证起点上没有障碍。

给定起点坐标和终点坐标，每个方格最多经过一次，问有多少种从起点坐标到终点坐标的方案。

输入格式

第一行为三个正整数 N, M, T，分别表示迷宫的长宽和障碍总数。

第二行为四个正整数 SX, SY, FX, FY，SX, SY 代表起点坐标，FX, FY 代表终点坐标。

接下来 T 行，每行两个正整数，表示障碍点的坐标。

输出格式

输出从起点坐标到终点坐标的方案总数。

输入输出样例

输入 #1 复制

```
2 2 1
1 1 2 2
1 2
```

输出 #1

```
1
```

说明/提示

对于 100% 的数据，$1 \le N, M \le 5$，$1 \le T \le 10$，$1 \le SX, FX \le n$，$1 \le SY, FY \le m$。

思路:dfs模板.

```
#include<bits/stdc++.h>
using namespace std;
int q[101][101];
int sum=0;
int i,j,n,m,t,sx,sy,x,y,ex,ey;
void dfs(int a,int b)
{
    if (a==ex&&b==ey)
    {
        sum++;
        return;
    }
    else
    {
        q[a][b]=0;
        if(q[a-1][b]) {dfs(a-1,b);q[a-1][b]=1;}
        if(q[a][b-1]) {dfs(a,b-1);q[a][b-1]=1;}
        if(q[a][b+1]) {dfs(a,b+1);q[a][b+1]=1;}
        if(q[a+1][b]) {dfs(a+1,b);q[a+1][b]=1;}
    }
}
int main()
{
    memset(q,0,sizeof(q));
    cin>>n>>m>>t;
    cin>>sx>>sy>>ex>>ey;
    for(i=1;i<=n;i++)
        for(j=1;j<=m;j++)
            q[i][j]=1;
    for(i=1;i<=t;i++)
    {
        cin>>x>>y;
        q[x][y]=0;
    }
    dfs(sx,sy);
    cout<<sum<<endl;
    return 0;
}
```

2.luoguP1255.

题目描述

楼梯有 N 阶，上楼可以一步上一阶，也可以一步上二阶。

编一个程序，计算共有多少种不同的走法。

输入格式

一个数字，楼梯数。

输出格式

输出走的方式总数。

输入输出样例

输入 #1 复制	输出 #1
4	5

说明/提示

- 对于 60% 的数据，$N \leq 50$;
- 对于 100% 的数据，$1 \leq N \leq 5000$。

思路:高精度+斐波那契.开3个数组,两个状态一个结果.

```
#include<bits/stdc++.h>
using namespace std;
int a[5000],b[5000],c[5000];
int main()
{
    int n;
    int x=1;
    cin>>n;
    if(n<3)
    {
```

```
            cout<<n;
            return 0;
        }
        a[1]=1;b[1]=2;
        for(int i=3;i<=n;i++)
        {
            for(int j=1;j<=x;j++)
                c[j]=a[j]+b[j];
            for(int j=1;j<=x;j++)
            {
                if(c[j]>9)
                {
                    c[j+1]=c[j+1]+c[j]/10;
                    c[j]%=10;
                    if(j+1>x)
                        x++;
                }
            }
            for(int j=1;j<=x;j++)
                a[j]=b[j];
            for(int j=1;j<=x;j++)
                b[j]=c[j];
        }
        for(int i=x;i>0;i--)
            cout<<b[i];
        return 0;
    }
```

3.模拟栈

7-1 出栈序列 分数 10

退出全屏

作者 DS课程组 单位 临沂大学

堆栈是一种经典的后进先出的线性结构，相关的操作主要有“入栈”（在堆栈顶插入一个元素）和“出栈”（将栈顶元素返回并从堆栈中删除）。我们用P表示入栈操作，用O表示出栈操作。给定一个入栈序列和一个结果序列，假如入栈过程中允许出栈，请判断结果序列是否是一个合法的出栈序列。例如入栈序列为1,2,3,4,5，则3,4,5,2,1是一个合法的出栈序列，对应的操作序列为PPPOPOPOOO，而3,4,5,1,2不是一个合法的出栈序列，因为子序列3,4,5可以经过PPPOPOPO操作序列得到，但是此时栈顶元素为2，无法与结果序列剩余子序列匹配，则判断结果不合法，此时堆栈里面剩余序列从栈底到栈顶依次为2,1。

输入格式:

输入为两行，第一行为入栈序列，第二行为结果序列，两个序列都是长度不超过10仅有0-9组成的字符串。

输出格式:

输出为两行，如果结果序列是合法的出栈序列，则第一行输出“right”，第二行输出入栈出栈操作的顺序序列，否则，第一行输出“wrong”，第二行按照从底到顶的顺序输出无法继续进行入栈出栈操作时堆栈里面的剩余元素。

思路:对于所有方法可以分成2类:

- 第一种: 元素入栈之后立刻出栈，那么出栈序列的顺序还是1，2，...，n的顺序;
- 第二种: 一直入栈，然后开始出栈，那么出栈序列的顺序就是k，k-1，.....，1;

```
    #include <iostream>
    #include <stack>
    #include <string>

    using namespace std;
```

```
int main() {
string in_seq, out_seq;
getline(cin, in_seq);
getline(cin, out_seq);

stack<char> s;
string op_seq = "";
int in_idx = 0, out_idx = 0;

while (out_idx < out_seq.length()) {
    if (!s.empty() && s.top() == out_seq[out_idx]) {
    s.pop();
    op_seq += "O";
    out_idx++;
    } else {
    while (in_idx < in_seq.length() && in_seq[in_idx] != out_seq[out_idx]) {
        s.push(in_seq[in_idx]);
        op_seq += "P";
        in_idx++;
    }
    if (in_idx == in_seq.length()) {
        cout << "wrong" << endl;
        string remaining = "";
        while (!s.empty()) {
        remaining = s.top() + remaining;
        s.pop();
        }
        cout << remaining << endl;
        return 0;
    } else {
        s.push(in_seq[in_idx]);
        op_seq += "P";
        in_idx++;
        s.pop();
        op_seq += "O";
        out_idx++;
    }
    }
}
cout << "right" << endl;
cout << op_seq << endl;

return 0;
}
```