深度优先搜索算法DFS

20计科官昕

前言

- 本次课需要对之前的培训中的部分内容能够掌握才能比较顺利的理解，请同学们对该部分内容提前进行复习
  - STL用法中栈的部分
  - 递归与递推中递归的部分
  - 基本语法、二维数组、函数等基础内容
- 本次课难度较之前有所下降，与之前的贪心、dp等更像是一种思路的内容相比，本次课的内容是一种更加具体的算法，不过与之后课程中可以几乎不加改动的模板相比，本次课内容比较灵活，需要根据情况对模板进行改写，需要将算法的原理彻底搞清楚才能灵活运用

目录

栈

Stack

栈是什么?

- 栈是一种先进后出的数据结构，其特点为越先入栈的元素越后出栈，比如：
  - 入栈顺序为：1 2 3
  - 则出栈顺序为：3 2 1
- 在ACM竞赛中，使用栈时可以直接使用STL中的stack容器
- stack常用函数：
  - 入栈：s.push(x); 出栈：s.pop(); 栈顶元素：x=s.top();
  - 求栈中元素个数：x=s.size(); 判断是否为空：s.empty();

栈的例题

基本操作

- 给予n个对栈的操作，按要求进行操作，以下为操作种类：
  - 两个正整数：1 x 表示将x入栈
  - 一个正整数：2 表示将栈顶出栈
  - 一个正整数：3 表示输出栈顶元素
- 输入描述：
  - 第一行一个正整数T，表示测试组数
  - 对于每一组测试，第一行为一个正整数n，表示操作数
  - 接下来n行，每行一条操作指令

基本操作

- 输出描述：
  - 对于每一组测试，输出m行，m为第三种操作的数量，每行一个正整数，表示栈顶元素
  - 若进行第三种操作时栈为空，输出-1
  - 每组测试之间用空行隔开
- 提示：
  - 可以使用STL中的stack容器简化操作
  - 每组测试前记得将栈清空

```cpp
#include<iostream>
#include<stack>
using namespace std;
stack<int> s;
int t,n,x;
int main()
{
    cin>>t;
    while(t--)
    {
        if(!s.empty())
            s.pop();
        cin>>n;
        for(int i=0;i<n;i++)
        {
            cin>>x;
            switch(x)
            {
                case 1:cin>>x;s.push(x);break;
                case 2:s.pop();break;
                case 3:cout<<s.top()<<endl;break;
            }
        }
        cout<<endl;
    }
    return 0;
}
```

括号匹配

- 给予一个括号序列，判断其是否合法
  - 序列只包含六种字符 '(' ')' '[' ']' '{' '}'
  - 同种类的括号在同一层次能够两两配对为合法，否则为非法
  - 例：({}[()])合法，([)]和(()[]非法
- 输入描述：
  - 一行一个括号序列
- 输出描述：
  - YES或NO，表示是否合法

计算表达式

- 读入一个只包含 +, -, *, / 的非负整数计算表达式，计算该表达式的值
  - 例：4 + 2 * 5 - 7 / 11
- 输入描述：
  - 一个计算表达式
- 输出描述：
  - 一个数字表示计算结果，保留到小数点后两位
- 提示：
  - 注意乘除优先级高于加减
  - 输入时应小心留意，好好思考

深度优先搜索

Depth-First-Search

什么是DFS?

- 是一种在树和图上的搜索算法,其特点为按照特定的搜索方式搜索到最深处或目标后,再逐级回溯,进入其他搜索支
- 与之相对的是广度优先搜索BFS，也就是下节课的内容，其特点为依次对每个节点扩散出所有搜索支，直到找到目标
- 树的前序，中序，后序遍历均可使用DFS实现，而层次遍历则是用BFS实现
- 无论是DFS还是BFS，搜索的对象都可以是抽象的树和图，凡是能抽象成树或图的结构的，均能进行搜索

DFS模板

- 栈版：

```
while(!s.empty())
{
    type x=s.top();
    s.pop();
    xxx //具体搜索操作，一般会用到循环
    {//设搜索到的下一个节点为y
        s.push(y);
        //搜索状态标记，比如更新visit数组
    }
}
```

DFS模板

- 但实际上，由于在c++中，函数的调用关系本身即为栈的结构，我们可以不使用栈来进行辅助，而是使用更简洁明了的递归来实现DFS：

```
void dfs(int x,int dep)
{
    if(x == dep) return ; ///边界条件
    xxx //具体搜索操作，一般会用到循环
    {
        xxx//搜索状态标记，比如更新visit数组
        dfs(u + 1,dep); ///搜索的分支
        xxx//状态还原，有时不需要这一步
    }
}
```

DFS例题

走迷宫

- 有一个$n \times m$的迷宫，入口在(1,1)，出口在(n,m)，每次可以向上下左右四个方向上的空地移动一步，求从入口到出口的最少步数
- 输入描述：
  - 第一行为两个整数n，m，表示迷宫大小
  - 之后n行，每行m个数：0或1，0表示墙壁，1表示空地
- 输出描述：
  - 一个整数，表示最少的步数

代码

```
//主要部分
int dfs(int x,int y,int step)
{
        int num=2147483647;
        if(x==n&&y==m)
                return step;
        for(int i=0;i<4;i++)
                if(ma[x+dx[i]][y+dy[i]]&&!vis[x+dx[i]][y+dy[i]])
                {
                        vis[x+dx[i]][y+dy[i]]=1;
                        num=min(num,dfs(x+dx[i],y+dy[i],step+1));
                        vis[x+dx[i]][y+dy[i]]=0;
                }
        return num;
}
```

代码

```
//其他部分
bool ma[100][100],vis[100][100];
int dx[4]={0,1,0,-1},dy[4]={1,0,-1,0};
int n,m;
//DFS函数所在位置
int main()
{
        cin>>n>>m;
        for(int i=1;i<=n;i++)
                for(int j=1;j<=m;j++)
                        cin>>ma[i][j];
        vis[1][1]=1;
        cout<<dfs(1,1,0)<<endl;
        return 0;
}
```

全排列

- 给予一个正整数n，请输出1~n的全排列
- 输入描述：
  - 一个正整数n，意义如题目描述
- 输出描述：
  - A_n^n行，每行一种排列方式

八皇后

- 八皇后问题（英文：Eight queens），是由国际西洋棋棋手马克斯·贝瑟尔于1848年提出的问题，是回溯算法的典型案例。
- 问题表述为：在8×8格的国际象棋上摆放8个皇后，使其不能互相攻击，即任意两个皇后都不能处于同一行、同一列或同一斜线上，问有多少种摆法。

讲解结束，同学们可以开始快乐做题了