**实验一 深度优先搜索**

一.实验题目

编写实现深度优先搜索的算法。

二.实验分析

创建如下的图

1.从根节点开始

2.放入一个节点（起始时放入的为根节点）

3.如果这个节点是第一次出现，则放入堆栈中

4.判断该节点的子节点是否搜索完成，

        a.如果是则将该节点出栈,判断该栈是否为空

             a.1 若为空则结束

            a.2 若不为空则取栈顶元素，并回到第2步

        b.如果没有完成搜索，取未被搜索的根节点，并回到第2步

三.实验内容

源代码：c++实现

#include<iostream>

#include<vector>

#include<stack>

#include<memory.h>

using namespace std;

vector<vector<int> > tree; //声明一个二维向量

int flag[10]; //用于搜索到了节点i的第几个节点

stack <int>stk; //声明一个堆栈

int ar\_tree[8] = { 1,1,1,3,5,3,5,7 };

void DFS(int node)

{

cout << node <<" ";

if (flag[node] == 0)

{

stk.push(node);

}

int temp;

//判断node的子节点是否搜索完成

if (flag[node] < tree[node].size())

{

temp = tree[node][flag[node]];

flag[node]++;

DFS(temp);

}

else

{ //若已经完成

stk.pop(); //弹出子节点

if (!stk.empty())

{ //若堆栈不为空

temp = stk.top(); //取此时的栈顶元素，即为node的上一级节点

DFS(temp);

}

}

}

int main()

{

ios::sync\_with\_stdio(false);

memset(flag, 0, sizeof(flag));

register int i,temp;

tree.resize(10);//图中的数共有九个节点

//生成树

for (i = 2; i <=9; i++)

{

temp = ar\_tree[i - 2];

tree[temp].push\_back(i); //表示第i个节点为第temp个节点的子节点

}

//DFS

cout << "DFS过程：" << endl;

DFS(1);

cout << endl;

return 0;

}

实验结果：



第一次：1 2

第二次：1 3 5 6

第三次：5 8

第四次：5 3 7 9

第五次：7 3

第六次：1 4 1