**实验一**

实验题目：DFS（深度优先搜索算法）

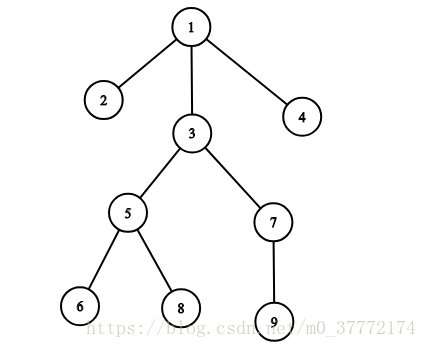
题目讲解：深度优先搜索算法是一种用于遍历或搜索树或图的算法。沿着树的深度遍历树的节点，尽可能深的搜索树的分支。当节点v的所在边都已被探寻过或在搜索时节点不满足条件，搜索将回到发现节点v的那条边的起始节点。整个过程反复进行直到所有节点都被访问为止。

深度搜索：时间复杂度大，空间复杂度小。

算法过程

从根节点开始，放入一个节点（起始时放入的为根节点）。如果这个节点是第一次出现，则放入堆栈中，判断该节点的子节点是否搜索完成，如果是则将该节点出栈,判断该栈是否为空，若为空则结束，若不为空则取栈顶元素，并回到第2步，如果没有完成搜索，取未被搜索的根节点，并回到第2步。

图：



实验代码：

#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);//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;

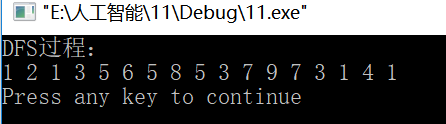
DFS(1);

cout << endl;

return 0;

}

运行结果：



实验总结：

DFS用栈来实现搜索

vector<vector<int> > tree;<int>后要有一个空格。