深度优先搜索(DFS)

过程简要来说是对每一个可能的分支路径深入到不能再深入为止，而且每个节点只能访问一次.

深度优先搜索的特点：每次深度优先搜索的结果必然是图的一个连通分量.深度优先搜索可以从多点发起.如果将每个节点在深度优先搜索过程中的“结束时间”排序（具体做法是创建一个list然后在每个节点的相邻节点都已被访问的情况下，将该节点加入list结尾，然后逆转整个链表），则我们可以得到所谓的“拓扑排序”,即topological sort.

深度优先遍历图的方法是，从图中某顶点v出发：

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

递归实现

```
void dfs(int v)//以v开始做深度优先搜索
{
    list<int>::iterator it;
    visited[v] = true;
    cout << v << " ";
    for (it = graph[v].begin(); it != graph[v].end(); it++)
        if (!visited[*it])
            dfs(*it);
}
```

非递归实现（栈）

```
void dfs(int v)//以v开始做深度优先搜索
{
    list<int>::iterator it;
    visited[v] = true;
    cout << v << " ";
    stack<int>mystack;
    mystack.push(v);
    while (!mystack.empty())
    {
        v = mystack.top();
        mystack.pop();
```

```
        if (!visited[v])
        {
            cout << v << " ";
            visited[v] = true;
        }

        for (it = graph[v].begin(); it != graph[v].end(); it++)
        {
            if (!visited[*it])
            {
                mystack.push(*it);

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

From:
https://wiki.cvbbacm.com/ - **CVBB ACM Team**

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:namespace:kongyou:%E6%B7%B1%E5%BA%A6%E4%BC%98%E5%85%88%E6%90%9C%E7%B4%A2

Last update: **2020/05/16 11:29**