第4章 减治法

*Steven*

1. 图的深度优先查找和广度优先查找，可看作是减一技术在图上的应用。分别设计使用深度优先查找和广度优先查找方法求得一个图的连通分量的算法。

连通分量是整个图结构的极大连通子图，在无向连通图中只有1个，即原图本身，在非连通图中则有多个。所以需要先判断原图是否为连通图，再进行求解连通分量。

(1)深度优先查找

Graph[] DFS\_Search(Graph G) { //深度优先查找

Graph connected[]; //存放连通分量

for (v = 0; len(v) < G.vexnum; v++) visited[v] = false; //初始化遍历情况

for (v = 0; len(v) < G.vexnum; v++) {

if (!visited[v]) connected[0].append(DFS(G, v));

//若当前点未被访问过，则从该点开始深度优先遍历获取到一个连通分量

}

return connected;

}

Graph DFS(Graph G, int v) {

Graph sub\_graph; //存放当前遍历的子图，即当前的连通分量

visited[v] = true;

sub\_graph.append(v); // 将该点及其邻边放入当前连通分量中

for (w = FirstAdjVes(G, v); w >= 0; w = NextAdjVes(G, v, w)) {

if (!visited[w]) G.extends(DFS(G, w)); //递归地深度优先遍历，并逐级存储

}

int a = length(visited);

return sub\_graph;

}

(2)广度优先查找

Graph[] BFS\_Search(Graph G) {

Graph connected[]; //存放连通分量

for (v = 0; len(v) < G.vexnum; v++) visited[v] = false; //初始化遍历情况

for (v = 0; len(v) < G.vexnum; v++) {

if (!visited[v]) connected.append(BFS(G, v)); //添加连通分量

}

return connected;

}

Graph BFS(Graph G, int v) {

Graph sub\_graph; //存放当前遍历的子图

visited[v] = true;

Queue Q; //创建一个队列

EnQueue(Q, v); //将v入队

while (!Empty(Q)) {//当队列不为空时

sub\_graph.append(DeQueue(Q, v)); //队列中的顶点存入子图

for (w = FirstAdjVes(G, v); w != NULL; w = NextAdjVes(G, v, w)) {

if (!visited[w]) {

visited[w] = true;

EnQueue(Q, w);

}

}

}

return sub\_graph;

}

2. 一个包含*n*个顶点的有向图，最多会有多少个不同的拓扑排序结果？

设存在个点时，拓扑排序结果数为，则在此基础上再添加一个点时，该点在种结果中均可以插入到个位置。故可得递推公式。此外，容易知，故递推公式为：

故可得

3. 对于以下两个有向图，都分别应用删源算法和基于DFS的算法来解拓扑排序问题（给出求解过程）

（1）



|  |  |
| --- | --- |
| 删源算法 | DFS算法 |
| ①取一源顶点(D)，删除其输出边    ②取一源顶点(DA)，删除其输出边    ③取一源顶点(DAB)，删除其输出边    ④取一源顶点(DABC)，删除其输出边    ⑤取一源顶点(DABCG)，删除其输出边    ⑥取一源顶点(DABCGE)，删除其输出边    ⑦取一源顶点(DABCGEF)，至此，得到拓扑排序结果：DABCGEF | ①任选一起点(D)    ②访问其邻接点(DA)    ③访问其邻接点(DAC)    ④访问其邻接点(DACF)    ⑤回溯  ⑥访问其下一邻接点(DACFB)    ⑧访问其下一邻接点(DACFBE)    ⑨访问其下一邻接点(DACFBEG)    ⑩至此，得到拓扑排序结果：DACFBGE |

（2）



因为原图中存在自环A→B→C→D→G→E，故无法进行拓扑排序。