图的遍历算法设计示例

【示例-1】假设图G采用邻接表存储，设计一个算法，判断无向图G是否连通。若连通则返回1；否则返回0。

解：

采用某种遍历方法判断无向图G是否连通。这里用深度优先遍历DFS方法。

- ✓ 先给visited[ ]数组置初值0。
- ✓ 然后从0顶点开始遍历该图。
- ✓ 在一次遍历之后，若所有顶点*i*的visited[*i*]均为1，则该图是连通的；否则不连通。

```
int Connect(ALGraph *G)    //判断无向图G的连通性
{
   int i,flag=1;
   DFS(G,0);    //调用DFS算法,从顶点0开始深度优先遍历
   for(i=0;i<G->n;i++)
     if (visited[i]==0)
     {  flag=0;
        break;
     }
   return flag;
}
```

【示例-2】 假设图G采用邻接表存储，设计一个算法判断顶点u到顶点v之间是否有简单路径。

解：

采用深度优先遍历思路设计求解算法HasaPath(G，u，v)。

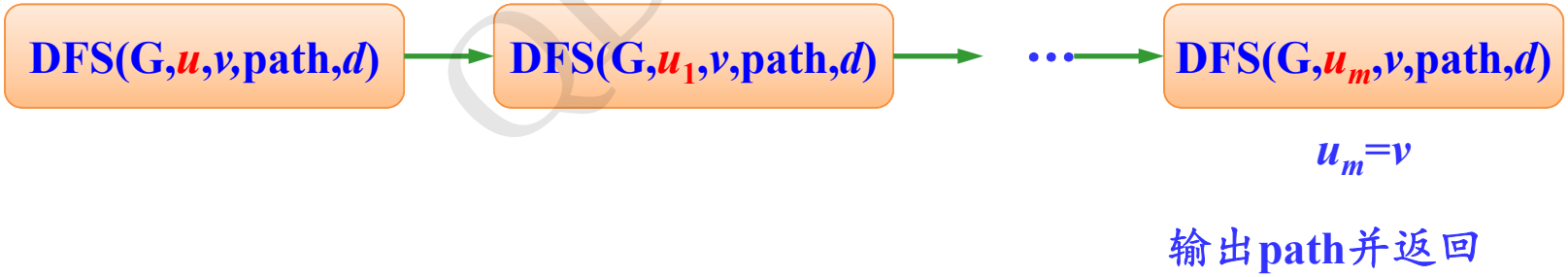
- ✓ 先置全局visited数组的所有元素值为0。
- ✓ 从顶点u出发进行深度优先遍历，置visited[u]=1；找到顶点u的一个未访问过的邻接点u_1，置visited[u_1]=1；找到顶点u_1的一个未访问过的邻接点u_2，置visited[u_2]=1；以此类推。
- ✓ 当找到的某个未访问过的邻接点u_n=v时，说明顶点u到v有一条简单路径，返回1。当整个遍历中都没有找到顶点v，说明u到v没有路径，返回0。

```
int visited[MAXVEX];                    //全局数组
int HasaPath(ALGraph *G,int u,int v)
{
   ArcNode *p;
   int w;
   visited[u]=1;
   p=G->adjlist[u].firstarc;   //p指向u的第一个邻接点
   while (p!=NULL)
   {  w=p->adjvex;                      //邻接点的编号为w
      if (w==v)                         //找到顶点v后返回1
         return 1;
      if (visited[w]==0)                //若顶点w没有访问过
      {  if (HasaPath(G,w,v))           //从w出发进行深度优先遍历
           return 1;                    //若从w出发找到顶点v返回1
      }
      p=p->nextarc;                     //p指向下一个邻接点
   }
   return 0;                            //没有找到顶点v，返回0
}
```

【示例-3】假设图G采用邻接表存储，设计一个算法输出图G中从顶点u⇨ v的一条简单路径（假设图G中从顶点u⇨ v至少有一条简单路径）。

求解思路

- 采用深度优先遍历的方法。
- 增加path和d形参，其中path存放顶点u到v的路径，d表示path中的路径长度，其初值为-1。
- 当从顶点u遍历到顶点v后，输出path并返回。



```
void FindaPath(ALGraph *G, int u, int v, int path[], int d)
{  //d表示path中的路径长度，初始为-1
    int w, i;  ArcNode *p;
    visited[u]=1;
    d++; path[d]=u;                    //路径长度d增1，顶点u加入到路径中
    if (u==v)                          //找到一条路径后输出并返回
    {     printf("一条简单路径为:");
          for (i=0;i<=d;i++)
               printf("%d ", path[i]);
          return;                                  //找到一条路径后返回
    }
    p=G->adjlist[u].firstarc;          //p指向顶点u的第一个相邻点
    while (p!=NULL)
    {    w=p->adjvex;                  //相邻点的编号为w
         if (visited[w]==0)
             FindaPath(G, w, v, path, d);
         p=p->nextarc;                 //p指向顶点u的下一个相邻点
    }
}
```

深度优先遍历

END