# 5.3图的遍历

从图中某个顶点出发，沿路径使图中**每个顶点被访问且仅被访问一次**的过程，称为遍历图。

两种常用遍历图的方法
- **深度优先搜索**
- **广度优先搜索**

# 5.3.1深度优先搜索DFS（depth-first-search)

## 1. 深度优先搜索法遍历图的过程为：

1). 访问指定的某顶点V，将V作为当前顶点

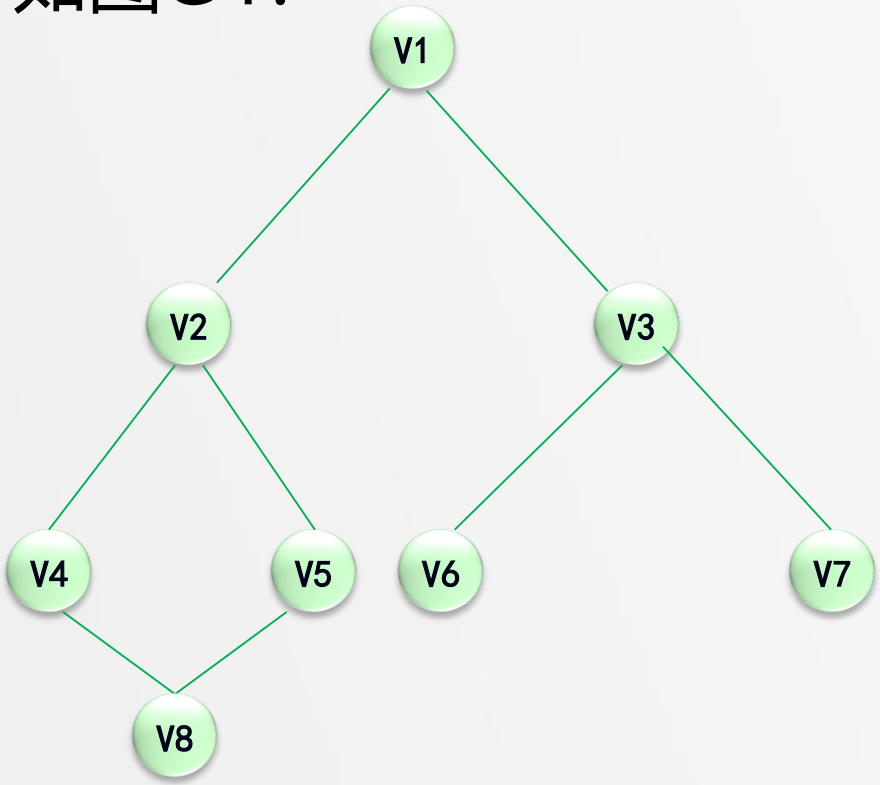
2). 访问当前顶点的下一未被访问过的邻接点，并将该顶点作为当前顶点

3). 重复2，直到当前顶点的所有邻接点都被访问过

4). 沿搜索路径回退，退到尚有邻接点未被访问过的某结点，将该结点作为当前结点，重复2，直到所有顶点被访问过的为止

# 图的遍历

如图G4：




深到底 —访问→ 回退

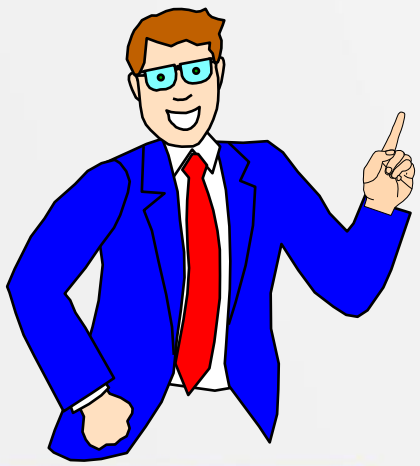
深到底 → **V1V2V4V8V5** → 回退

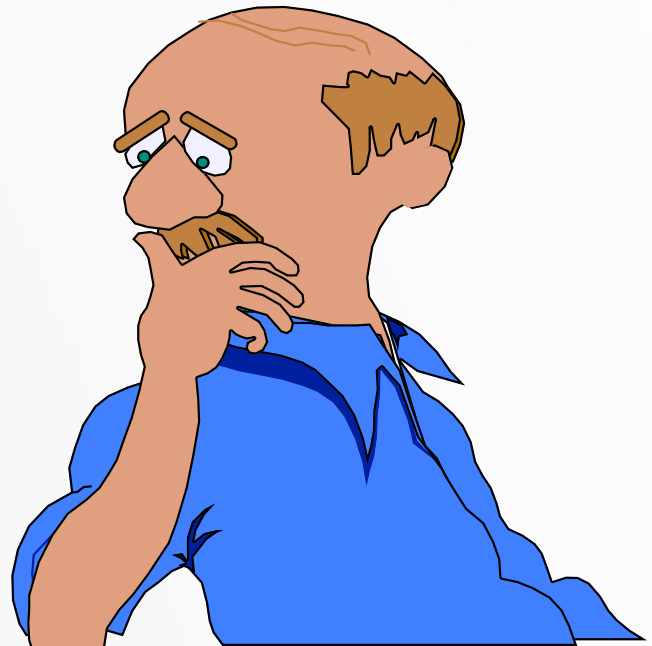
（V2V8均已访问）

深到底 → **V3V6V7**

# 图的遍历

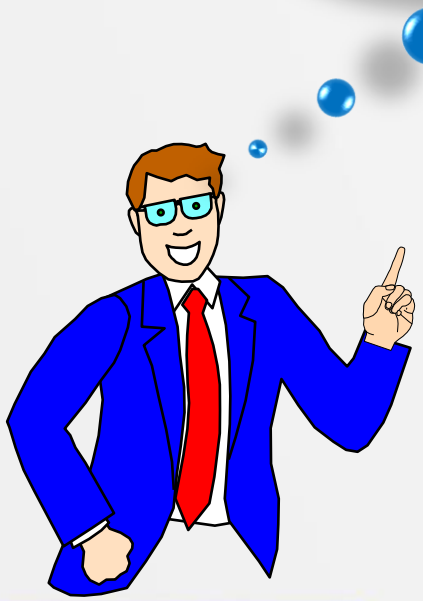
怎么写程序实现深度优先遍历?

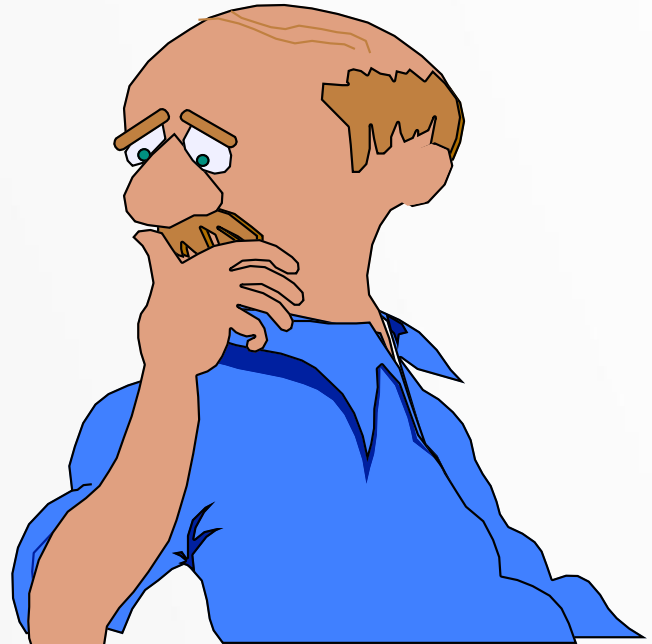




# 图的遍历

可以采用递归和非递归2种方法
实现相同的深度优先遍历算法!





# 图的遍历

## 深度优先遍历的递归算法:

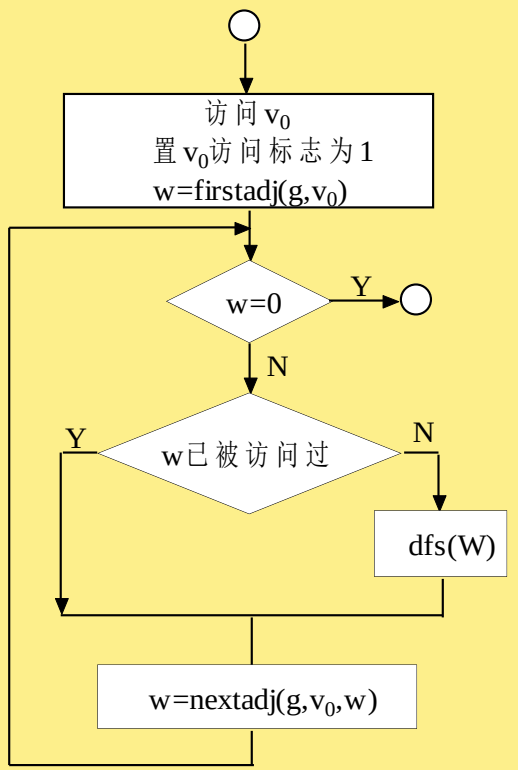


图 4.16 深度遍历算法流程图

算法4.1 从顶点v0出发深度优先遍历g中能访问的各个顶点

```
void dfs(int v0)
{  visited[v0]=1; /*访问标志置为 1，表示已被访问*/
   w=firstadj(g,v0); /* w是vo的第一个邻接点 */
   while (w!=0)
   { if(visited[w]==0) dfs(w); /*顶点未被访问，则递归的进行深度遍历 */
   w=nextadj(g,v0,w) /*顶点已访问，则取顶点v0在w后面的下一个邻接点 */
}
}
```

# 图的遍历

几点说明：

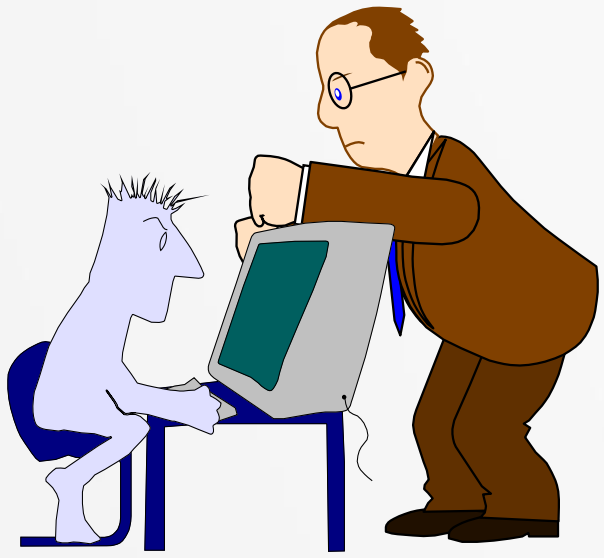
1.visited[1..n]**是一个辅助数组，记载顶点是否被访问过**

$$\textbf{visited[Vi]}=\begin{cases}\textbf{1, 已访问过}\\ \textbf{0, 未访问过（初值）}\end{cases}$$

2. firstadj (g,V0)**和**nextadj(g,V0,w)两个**函数的实现与图的具体存储结构有关**

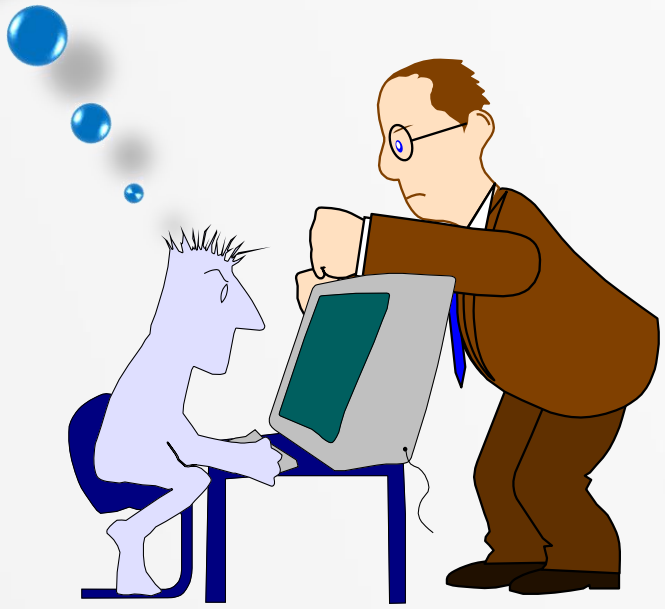
# 图的遍历

如何实现非递归算法?



# 图的遍历

当然是栈了!



# 讨 论

求下图的以1为开始点的深度优先遍历结果