7.3图的遍历

DS/SS

遍历图

图的遍历(Traversing Graph)：从图的某一个顶点出发，按照某条路径访问每个结点,使得每个结点均被访问一次，而且仅被访问一次。

- 在图中查找具有某种特征的结点
- 需要对结点进行访问（处理、输出等）操作
- 图的遍历算法是求解图的连通性问题、拓扑排序和求关键路径等算法的基础
- 遍历本质：结点之间非线性关系线性化的过程
- 遍历思路：

  *深度优先：类似于树的先根遍历；

  *广度优先：类似于树的层次遍历。

7.3图的遍历（续）

深度优先搜索

遍历思路：

从图的某个顶点v出发，访问此顶点，然后依次从v的未被访问的邻接点出发深度优先遍历图，直至图中所有和v有路径相通的顶点都被访问到；若此时图中尚有顶点未被访问，则另选图中一个未曾被访问的顶点作为始点，重复上述过程，直至图中所有顶点都被访问到为止。

这是一个递归算法，遍历过程中，当某个顶点的所有邻接点都被访问到后，需要“回溯”，即沿着调用的逆方向回退到上一层顶点，继续考察该顶点的下一个邻接点。

7.3图的遍历（续）

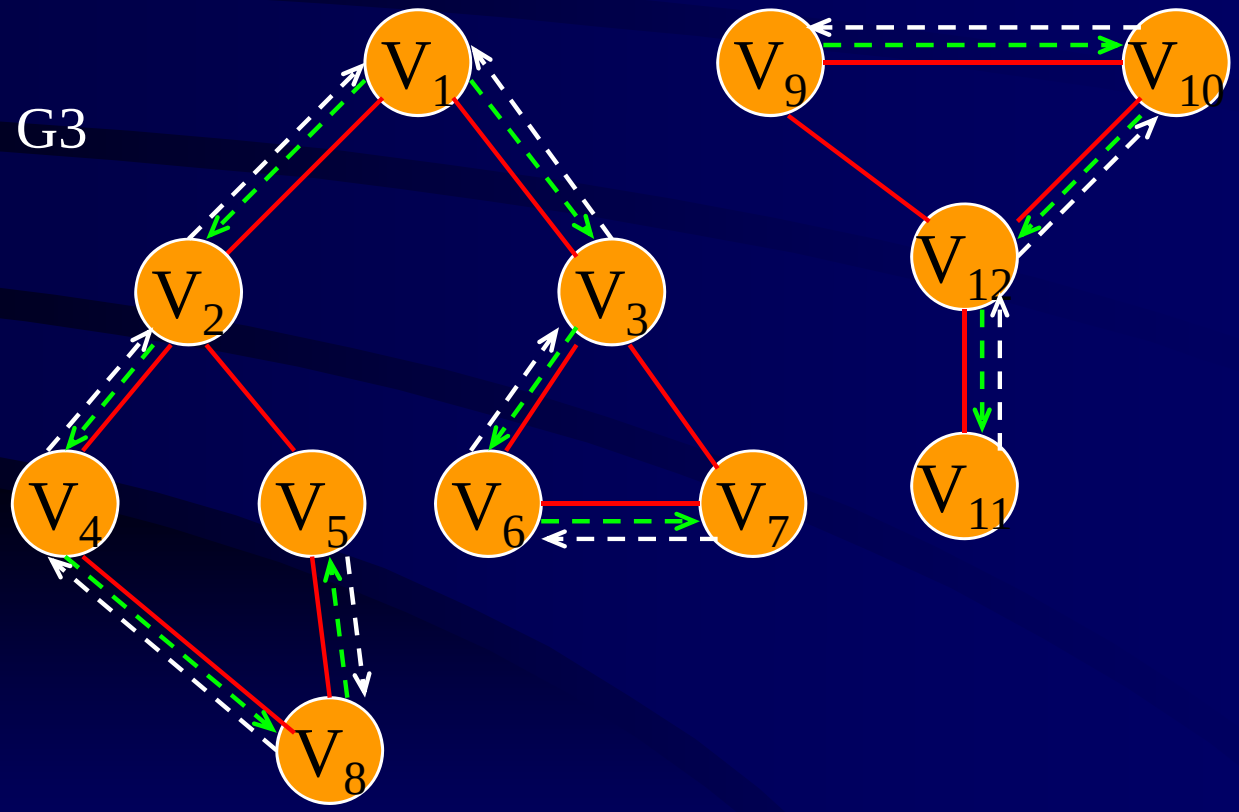
```
Boolean visited[MAX];     //访问标志数组
Status (*VisitFunc) (int v);//函数变量
Void DFSTraverse(Graph G,Status (*Visit)(int v)){
    VisitFunc = Visit;    //使用全局变量VisitFunc，使DFS不必设函数指针参数
    for (v =0; v<G.vexnum; ++v) visited[v] = FALSE; //标志数组初始化
    for (v =0; v<G.vexnum; ++v)   if (!visited[v])  DFS(G,v);   }
Status DFS (Graph G, int v){
  //从第v个顶点出发递归地深度优先遍历图G，对每个元素调用函数Visit
    visited[v] = TRUE;      VisitFunc(v);  //访问第v个顶点
    for ( w =FirstAdjVex(G,v); w>=0; w =NextAdjVex(G,v,w))
        if (!visited[w])   DFS(G,w);   } // 算法时间复杂度与存储结构相关
```

邻接矩阵存储:$T(n)=O(n^2)$;

邻接表存储:$T(n)=O(n+e)$

7.3图的遍历（续）

深度优先搜索



一种DFS遍历序列：$V_1, V_2, V_4, V_8, V_5, V_3, V_6, V_7, V_9, V_{10}, V_{12}, V_{11}$

当存储结构和算法确定后，遍历序列唯一。

7.3图的遍历（续）

广度优先搜索

遍历思路：

从图的某个顶点v出发，访问此顶点，然后依次访问v的未被访问的邻接点，然后从这些邻接点出发依次访问它们的邻接点，并使“先被访问的顶点的邻接点”先于“后被访问的顶点的邻接点”被访问，直至图中所有已被访问的顶点的邻接点都被访问到；若此时图中尚有顶点未被访问，则另选图中一个未曾被访问的顶点作为始点，重复上述过程，直至图中所有顶点都被访问到为止。

一层层遍历。如何保证上层次序先于下层次序？保证“先被访问的顶点的邻接点”先于“后被访问的顶点的邻接点”被访问？可以通过队列来控制实现。

7.3图的遍历（续）

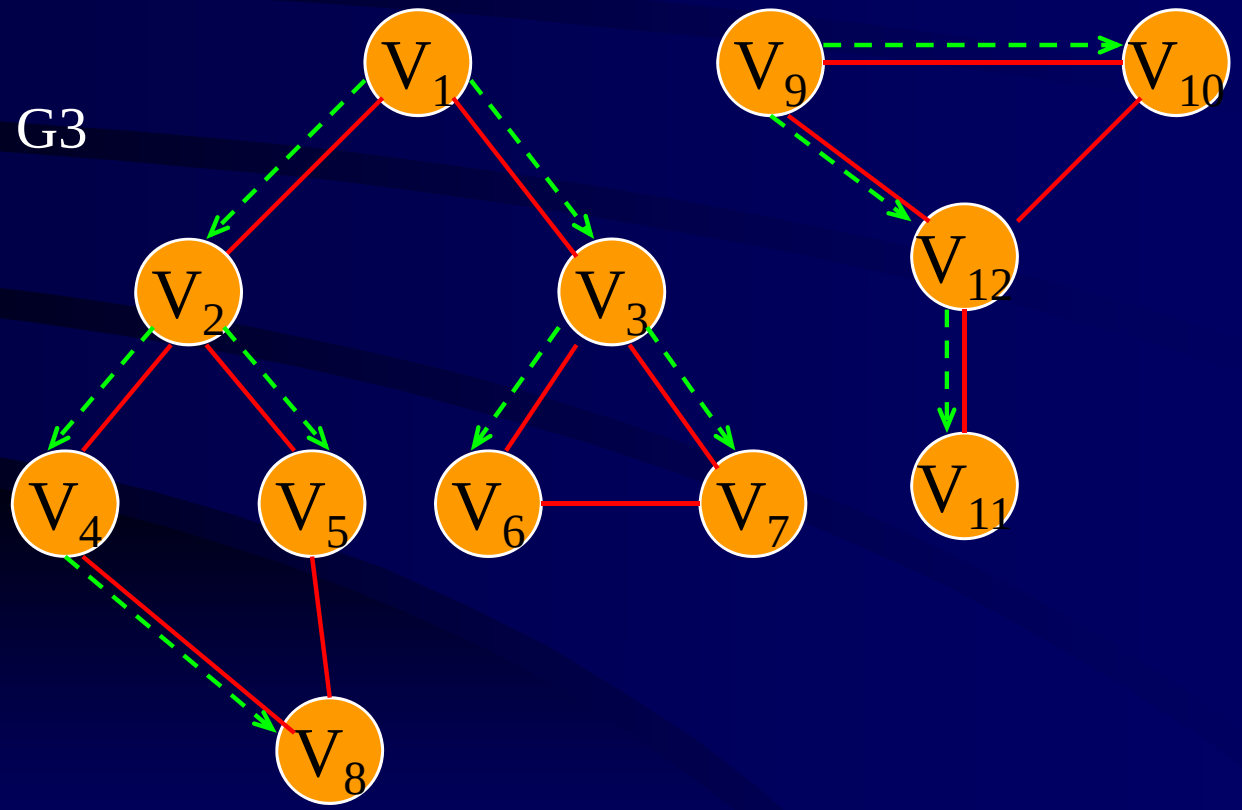
```
Boolean visited[MAX];     //访问标志数组
Void BFSTraverse(Graph G,Status (*Visit)(int v)){
    for (v =0; v<G.vexnum; ++v) visited[v] = FALSE; //标志数组初始化
    InitQueue(Q);              //置空的辅助队列
    for (v =0; v<G.vexnum; ++v)
        if (!visited[v])  {
          visited[v] = TRUE;        Visit (v);          EnQueue(Q,v);
          While(!QueueEmpty(Q))  {
              DeQueue(Q,u);
                  for ( w =FirstAdjVex(G,u); w>=0; w =NextAdjVex(G,u,w))
                     if (!visited[w]) {Visited[w] =TRUE; Visit(w); Enqueue(Q,w); }
        } //while    }if    } // 算法时间复杂度与存储结构相关
```

邻接矩阵存储:$T(n)=O(n^2)$;

邻接表存储:$T(n)=O(n+e)$

7.3图的遍历（续）

广度优先搜索



一种BFS遍历序列：$V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8, V_9, V_{10}, V_{12}, V_{11}$

当存储结构和算法确定后，遍历序列唯一。

7.4图的应用

DS/SS

典型应用

- 无向图的连通分量
- 无向图的生成树
- 连通网的最小生成树
- 有向无环图的拓扑排序
- 有向无环图的关键路径
- 有向网的最短路径

7.4图的应用（续）

DS/SS

无向图的连通分量

- 思路：利用遍历图的算法求解无向图的连通性。

- 方法：利用图的遍历（DFS与BFS均可）算法，对连通图，从任一定点出发遍历，可访问到所有顶点；对非连通图，需从多个顶点出发进行遍历，每一次从一个新的起始点出发遍历得到的顶点序列恰为其各个连通分量中的顶点集。最终求得一个无向图的所有连通分量。

7.4图的应用（续）

DS/SS

无向图的生成树

•思路：

利用遍历图的算法求解无向连通图的生成树和无向非连通图的生成森林。

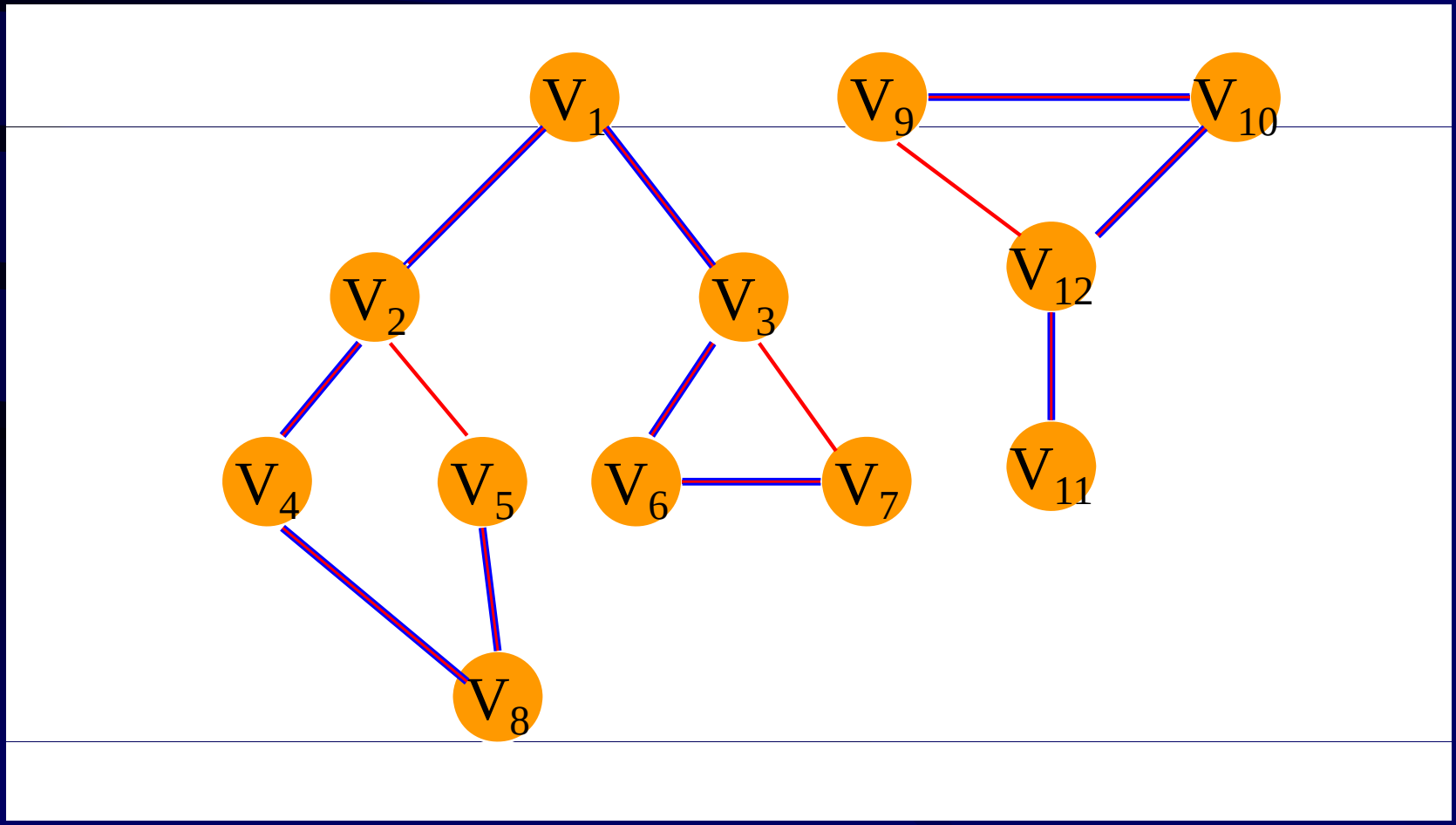
•方法：

对连通图G，设E(G)为连通图G中所有边的集合，则从图中任一顶点出发遍历图时，必将E(G)分为两个集合T(G)和B(G)，其中T(G)是遍历图过程中历经边的集合；B(G)是剩余边的集合。T(G)和图G中所有顶点一起构成连通图G的极小连通子图，它是连通图的一棵生成树。根据遍历方法，生成树分为DFS生成树和BFS生成树。

对无向非连通图，每个连通分量对应一个生成树，形成生成森林。

7.4图的应用（续）

无向图的DFS生成树



7.4图的应用（续）

无向图的BFS生成树