6.3 图的遍历

从已给的连通图中某一顶点出发，沿着一些边访问图中所有的顶点，且使每个顶点仅被访问一次，就叫做**图的遍历**（Graph Traversal）。

图中可能存在回路，在访问完某个顶点之后可能会沿着某些边又回到了曾经访问过的顶点。

如何确保图中每个顶点仅被访问一次？

48

雨课堂 Rain Classroom

6.3 图的遍历

为了避免重复访问，可设置一个标志顶点是否被访问过的辅助数组 visited [ ] ;

辅助数组 visited [ ] 的初始状态为 0 ;

在图的遍历过程中，一旦某一个顶点 i 被访问，就立即让 visited [i] 为 1，防止它被多次访问。

49

雨课堂 Rain Classroom

6.3 图的遍历

- 图的两种遍历方法
  - 深度优先遍历
  - 广度优先遍历

50

雨课堂 Rain Classroom

深度优先搜索算法（DFS, Depth First Search ）

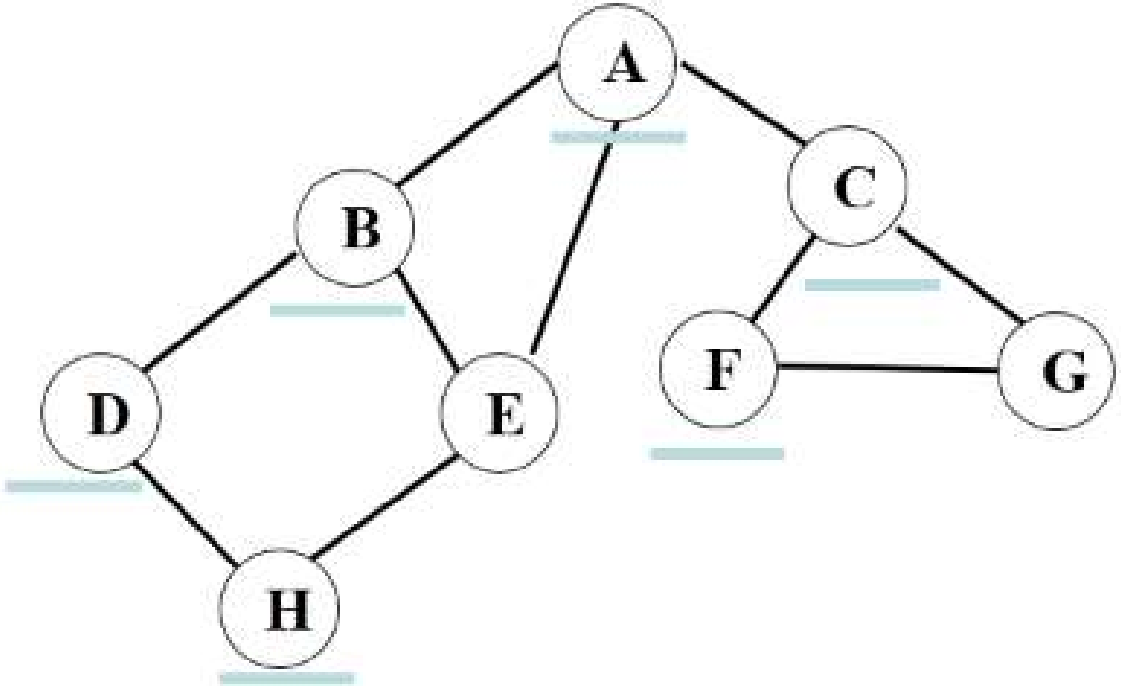
从图中某个顶点V_0出发，访问此顶点，然后依次从V_0的各个未被访问的邻接点出发深度优先搜索遍历图，直至V_0的所有邻接点都被访问到。

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起始点，重复上述过程，直至图中所有顶点都被访问到为止。

51

雨课堂 Rain Classroom

深度优先搜索算法



深度优先遍历的序列：A B D H E C F G

遍历过程中生成的树称为深度优先生成树。

图的深度优先遍历==树的先序遍历

32

深度优先搜索算法

基本思路：

1. **访问顶点v；**

2. **从v的未被访问的邻接点中选取一个顶点w，从w出发进行深度优先遍历；**

3. **重复上述两步，直至图中所有和v有路径相通的顶点都被访问到。**

53

雨课堂 Rain Classroom

深度优先搜索算法

算法描述：

1. 如何选择未被访问的顶点？

```
void dfsTraverse( ){
        for (v=1; v<=n; v++) visited[v]=false;
        for (v=1; v<=n; v++)
                if (!visited[v]) dfs(v);
}
```

2. 如何实现dfs(v)？

```
        void dfs(int v){
          visited[v]=true;  visitFunc(V);
          for (w=firstAdjVex(v); w; w=nextAdjVex(v,w))
                if (!visited[w])  dfs(w);
        }
```

54

雨课堂 Rain Classroom

DFS 算法效率分析：

设图中有 n 个顶点，e 条边，遍历时对图中每个顶点至多调用一次DSF函数，遍历图的过程就是对每个顶点查找其邻接点的过程，消耗的时间取决于所采用的存储结构

- 如果用邻接矩阵来表示图，查找每个顶点的邻接点所需的时间为$O(n^2)$。
- 如果用邻接表来表示图，查找每个顶点的邻接点所需的时间为$O(e)$，加上访问 n个头结点的时间，因此遍历图的时间复杂度为$O(n+e)$。

55

雨课堂 Rain Classroom

二、广度优先搜索*BFS*（Breadth First Search）

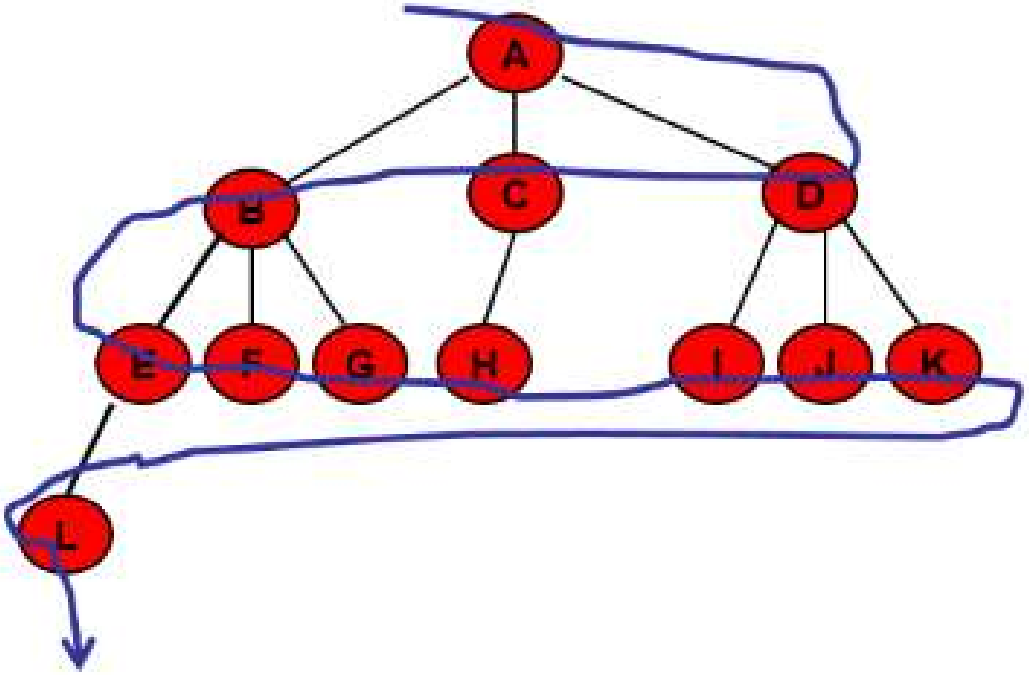
BFS算法思想：

从图中的某个顶点V_0出发，并在访问此顶点之后依次访问V_0的所有未被访问过的邻接点，之后按这些顶点被访问的先后次序依次访问它们的邻接点，直至图中所有和V_0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起始点，重复上述过程，直至图中所有顶点都被访问到为止。

雨课堂 Rain Classroom

二、广度优先搜索BFS



树的按层次进行访问的次序：

A、B、C、D、E、F、G、H、I、J、K、L

57

投票 最多可选1项

实现树的按层次进行访问，是否需要辅助结构？

A 需要

B 不需要

雨课堂 Rain Classroom

广度优先搜索BFS

广度优先搜索是一种分层的搜索过程，每向前走一步可能访问一批顶点，不像深度优先搜索那样有往回退的情况。因此，广度优先搜索不是一个递归的过程，其算法也不是递归的。

除了辅助数组 *visited*[ ]之外，为了实现逐层访问，算法中使用了一个队列，以记忆正在访问的这一层和上一层的顶点，以便于向下一层访问

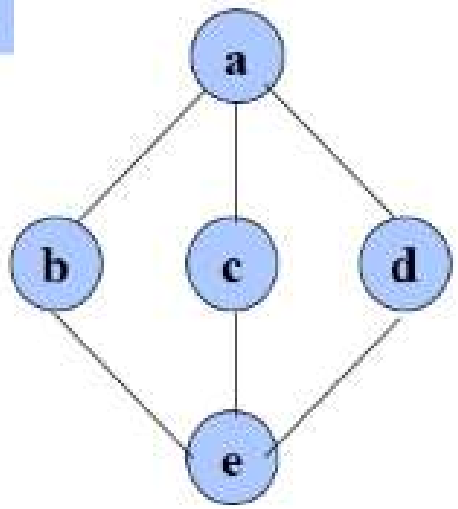
59

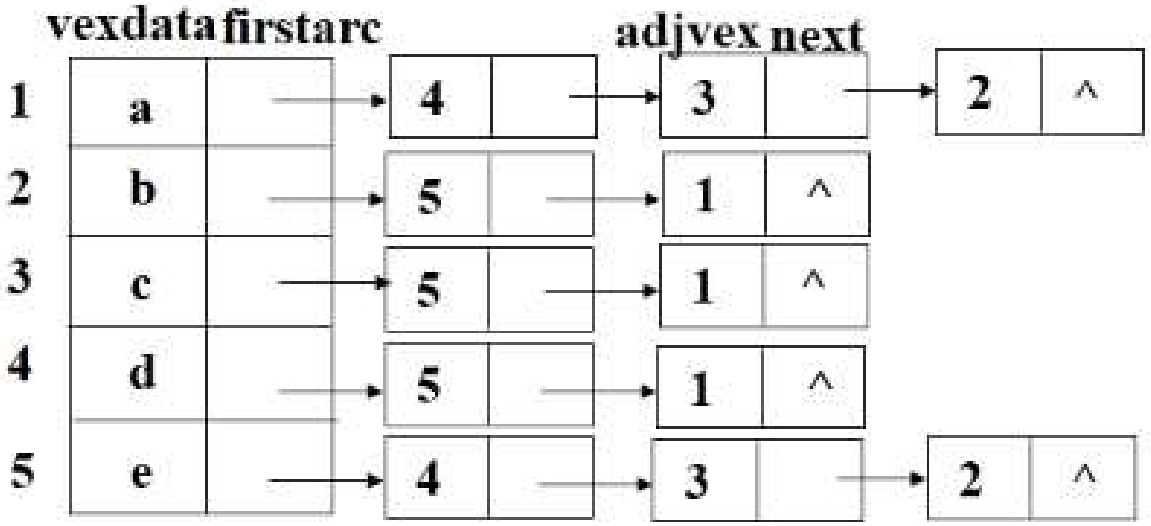
雨课堂 Rain Classroom

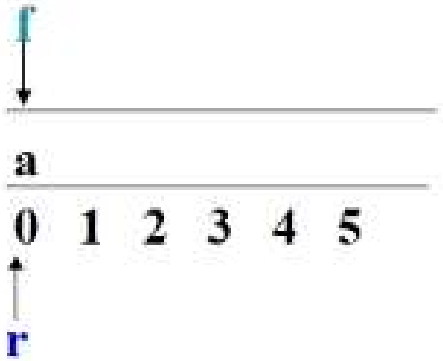
```
Void BFSTraverse( Gragh G, int v) {
1.for (v=0; v<G.vexnum; v++)  visited[v]=0; //清访问标记
 2.  InitQueue(Q);   //清队列Q;
 3. for (v=0; v<G.vexnum; v++) //对每一个顶点v
 4. if (!visited[v])   {
 5.      visited[v]=1; visit(v); //访问v; 标记v;
 6.       EnQueue(Q,v); //v未被访问
 7.       while (!QueueEmpty(Q)) {  // Q不空
 8.        DeQueue(Q,u);   //出队头元素到u
 9.       for (w=firstAdjVex(u); w;  w=nextAdjVex(u,w))
 10.           if (!visited[w]) {visited[w]=1; visit(w);
 11.          EnQueue(w); //将u的每个未被访问的邻接点w 入
队
          }//if
        }//while
     }//if
}
```

60

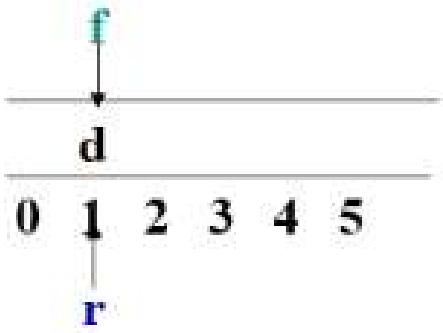
例



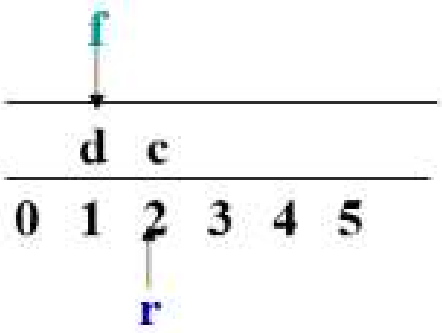




遍历序列：a



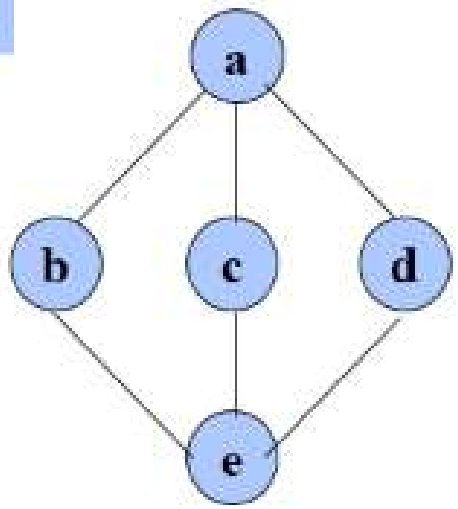
遍历序列：a d

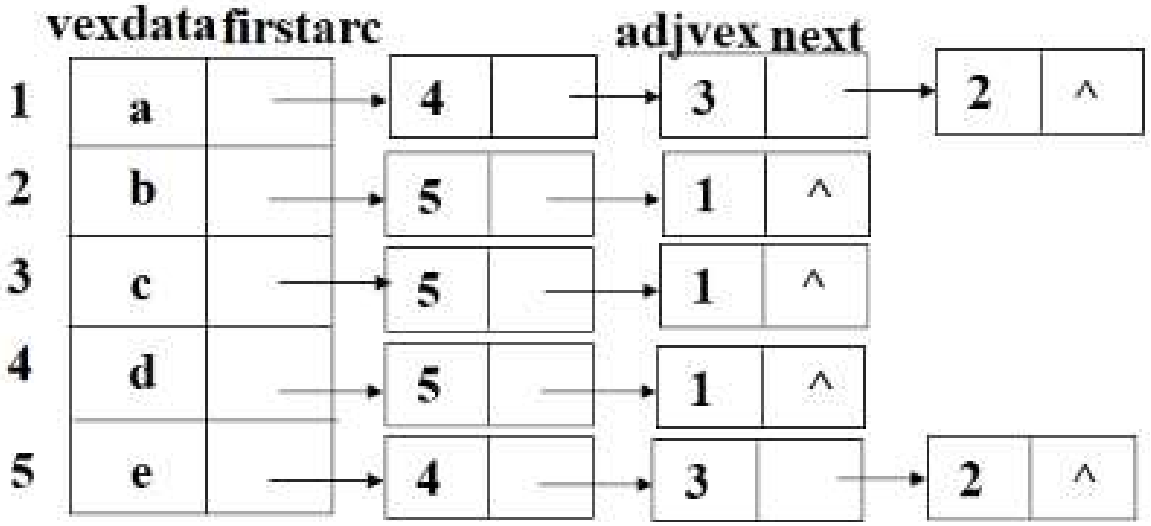


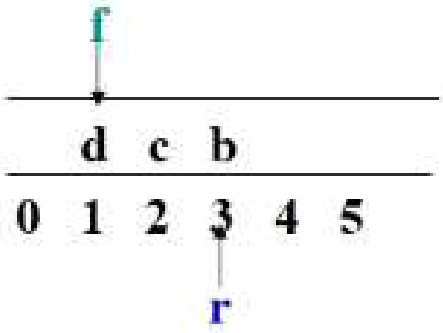
遍历序列：a d c

61

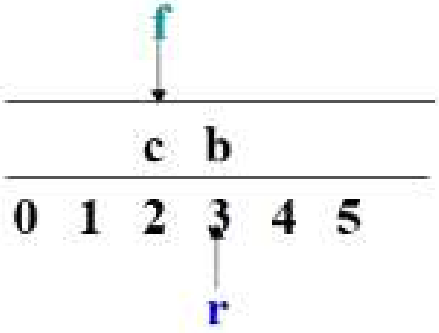
例



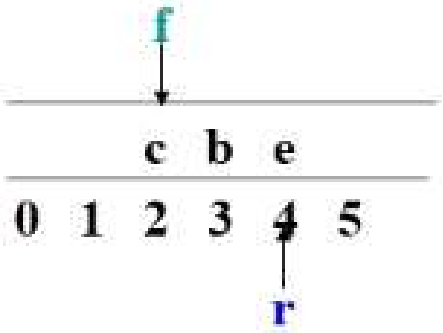




遍历序列：a d c b



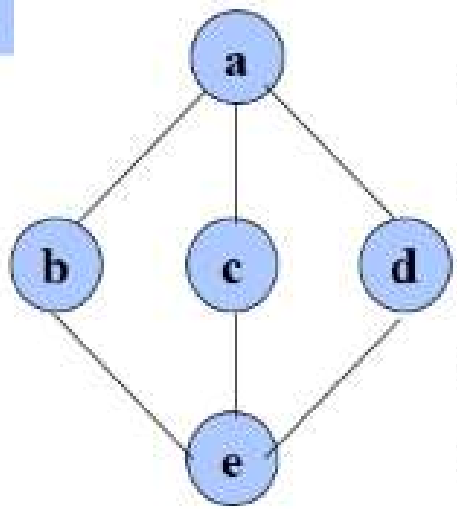
遍历序列：a d c b

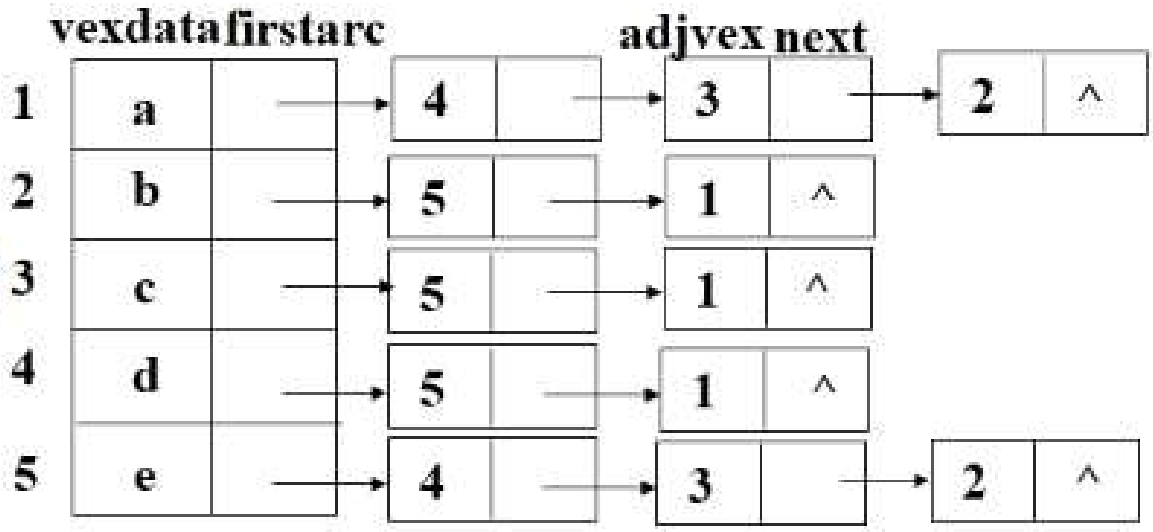


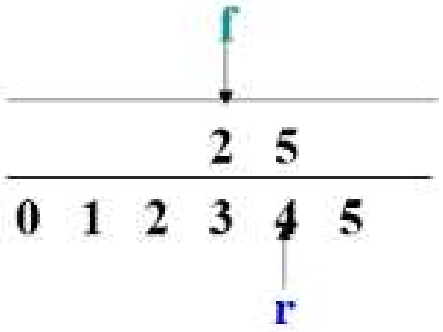
遍历序列：a d c b e

62

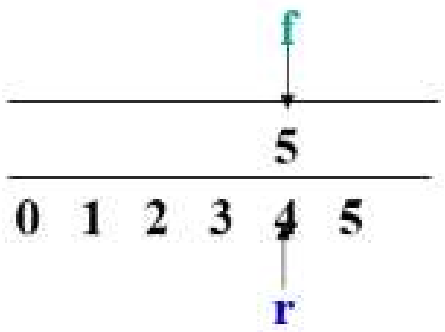
例



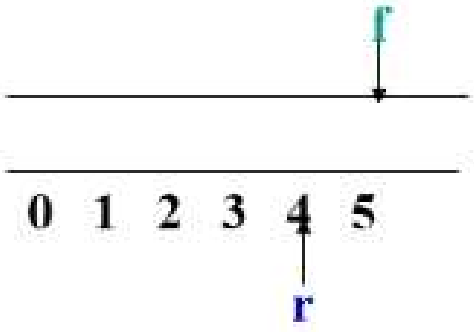




遍历序列：adcbe



遍历序列：adcbe



遍历序列：a d c b e

63

广度优先搜索BFS

算法分析:

- 如果使用邻接表表示图，则循环的总时间代价为 $d_0+d_1+\ldots+d_{n-1}=O(e)$，其中的 d_i 是顶点 i 的度
- 如果使用邻接矩阵，则对于每一个被访问过的顶点，循环要检测矩阵中的 n 个元素，总的时间代价为 $O(n^2)$。

DFS与BFS之比较：

- 空间复杂度相同，都是O(n)（借用了堆栈或队列）；
- 时间复杂度只与存储结构（邻接矩阵或邻接表）有关，而与搜索路径无关。

64

雨课堂 Rain Classroom

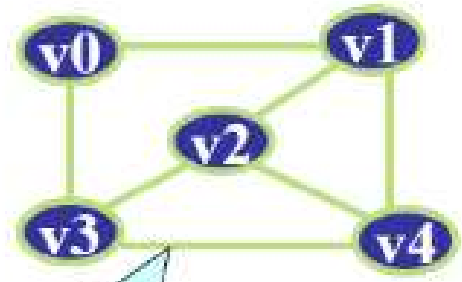
7.4 图的连通性问题

7.4.1 无向图的连通分量和生成树

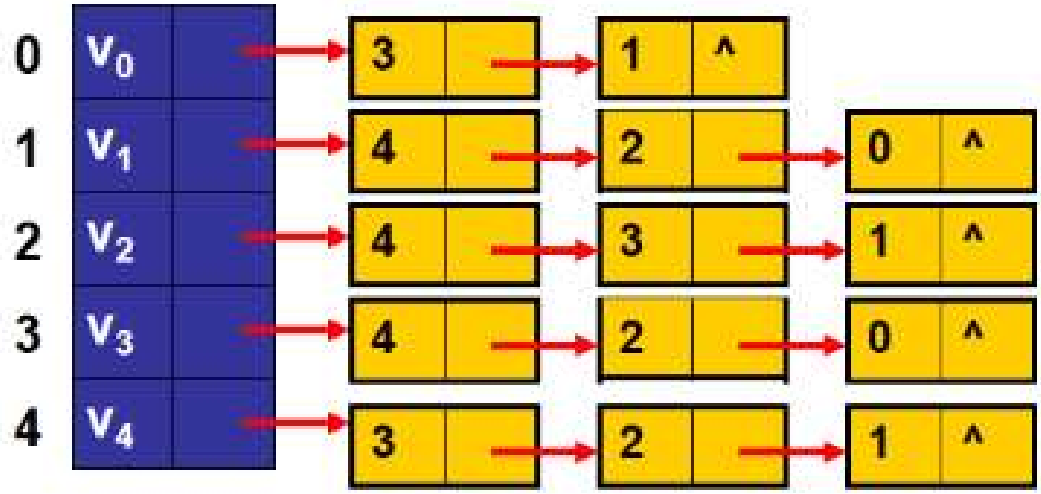
- 对无向连通图进行遍历得到的将是一个极小连通子图，即图的生成树！
  - ✓ 由深度优先搜索得到的生成树，称为深度优先搜索生成树。
  - ✓ 由广度优先搜索得到的生成树，称为广度优先搜索生成树。
- **对非连通图进行遍历，**得到的将是各连通分量的生成树，即图的生成森林！

雨课堂 Rain Classroom

例1：画出下图的生成树

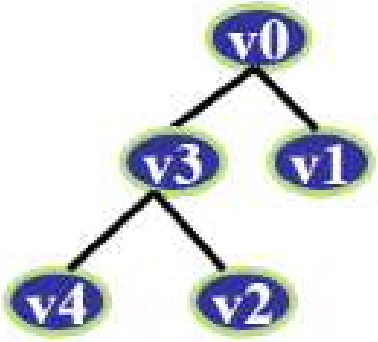


邻接表

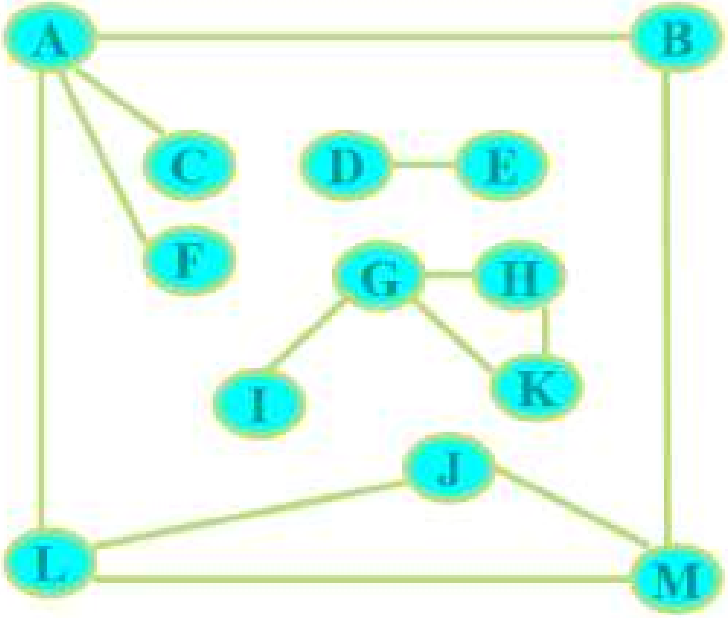


DFS 生成树

BFS 生成树



例2：画出下图的生成森林（或极小连通子图）



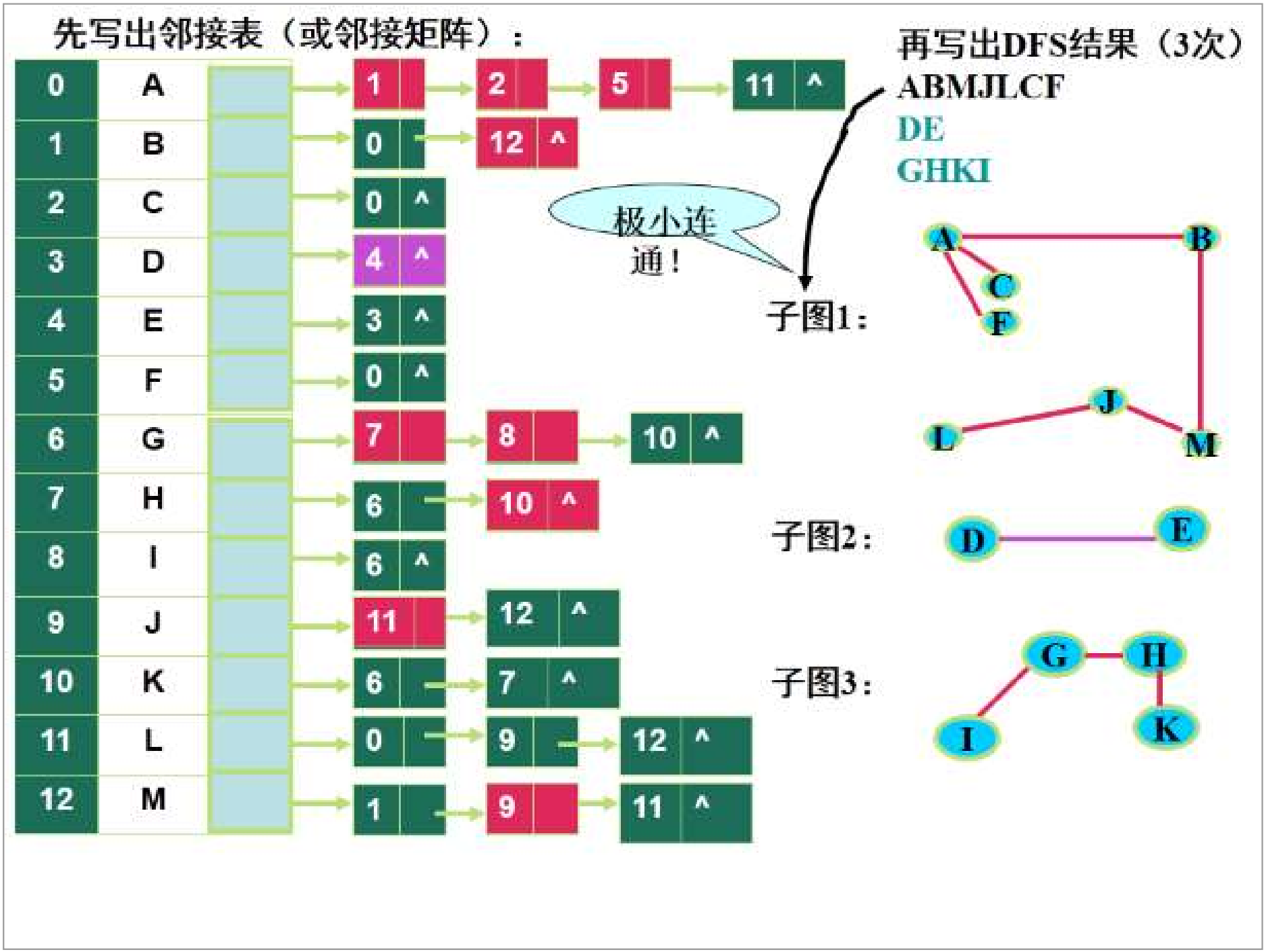
求解步骤：

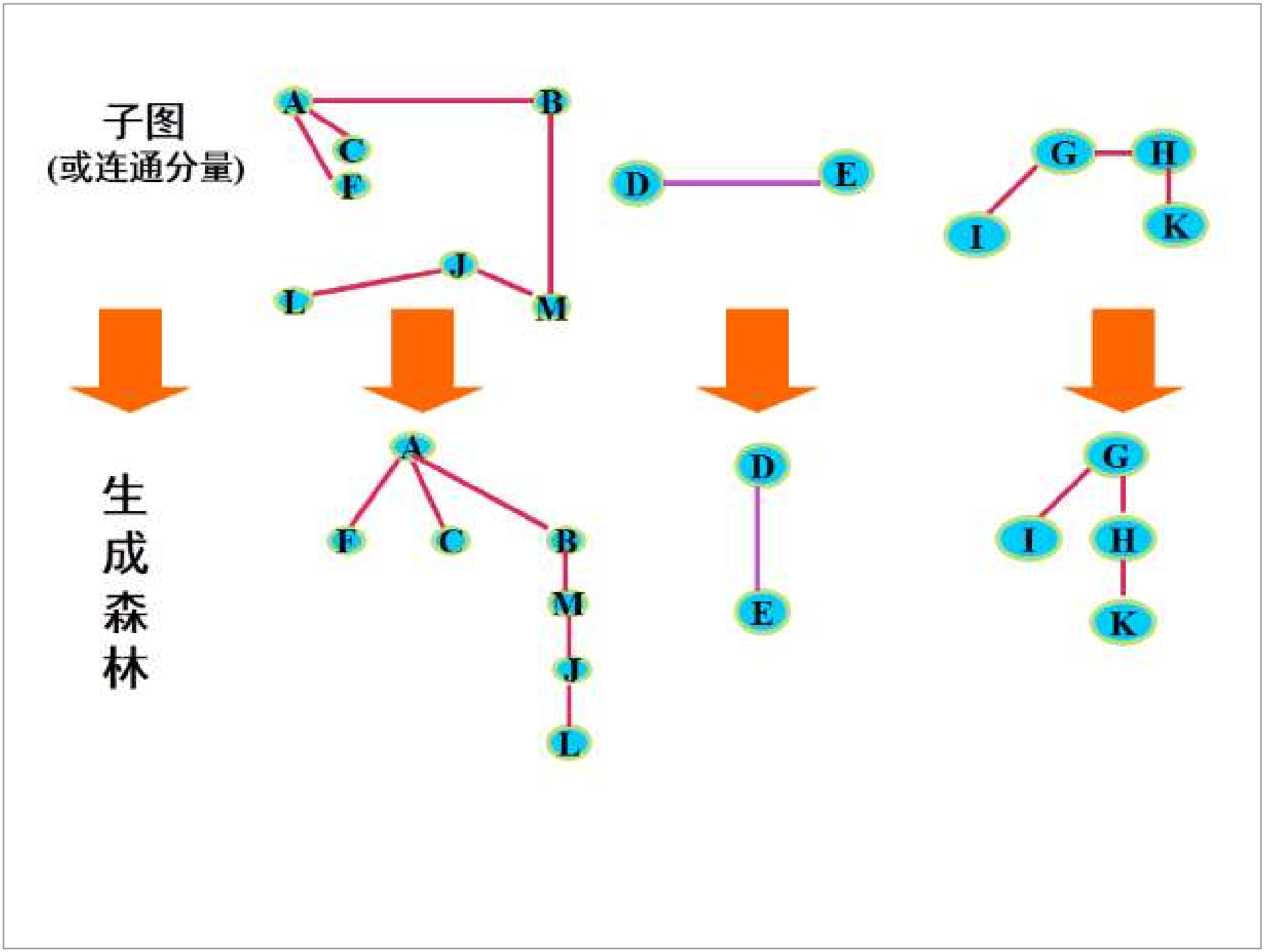
Step1：先求出邻接矩阵或邻接表；

Step2：写出DFS或BFS结果序列；

Step3：画出对应子图或生成森林。

我们选用邻接表方式来求深度优先搜索生成森林


先写出邻接表（或邻接矩阵）：
0 A
1 B
2 C
3 D
4 E
5 F
6 G
7 H
8 I
9 J
10 K
11 L
12 M
1 2 5 11 ^
0 12 ^
0 ^
4 ^
3 ^
0 ^
7 8 10 ^
6 10 ^
6 ^
11 12 ^
6 7 ^
0 9 12 ^
1 9 11 ^
极小连通！
再写出DFS结果（3次）
ABMJLCF
DE
GHKI
子图1：
A B C F J L M
子图2：
D E
子图3：
G H I K


子图
(或连通分量)
A
B
C
F
D
E
G
H
I
K
L
J
M
生成森林
A
F
C
B
M
J
L
D
E
G
I
H
K

7.4.3 最小生成树 MST(Minimum cost Spanning Tree)

- 使用不同的遍历图的方法，可以得到不同的生成树；
- 从不同的顶点出发，也可能得到不同的生成树。
- 按照生成树的定义，n 个顶点的连通网络的生成树有 n 个顶点、$n-1$ 条边。

MST:在网络的多个生成树中，寻找一个各边权值之和最小的生成树。

构造最小生成树的准则

- 必须只使用该网络中的边来构造最小生成树；
- 必须使用且仅使用$n-1$条边来联结网络中的n个顶点；
- 不能使用产生回路的边。

雨课堂 Rain Classroom