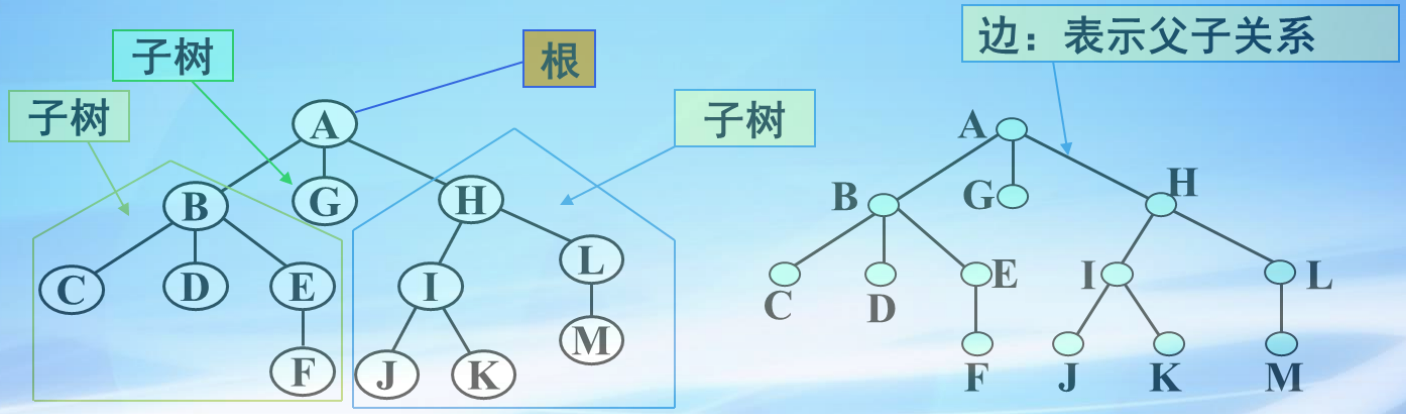
《数据结构》

图的遍历

主讲人：陈卫卫



学习目标和要求

1.准确描述图的先深搜索基本思想

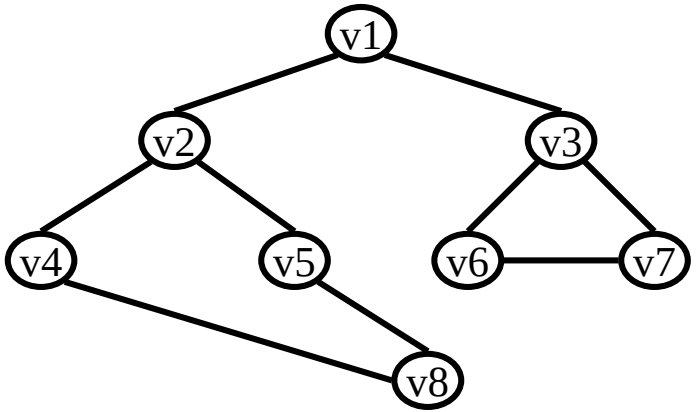
2.准确描述图的先广搜索基本思想

3.编程实现图的先深搜索算法和先广搜索算法

图的遍历

什么是图的遍历？

从图中的某一顶点出发，沿着图中的边按照特定规律访遍图中的每个顶点，且仅访问一次的过程，叫做图的遍历



图的遍历

图的遍历也叫图的搜索，包括两种类型：

（1）先深搜索(Depth-First Searching, DFS)

类似于二叉树的先序遍历

（2）先广搜索(Breadth-First Searching, BFS)

类似于二叉树的按层遍历

图的遍历是图的基本运算，能够用于求解：找出图的生成树、图的连通分量、迷宫问题等

先深搜索

◆ 遍历规则

步骤1）将图G中所有顶点作“未访问过”标记。
步骤2）任选某个未访问过的顶点v作搜索起点。
步骤3）访问v。
步骤4）选择v的每一个未访问过的邻接点w作起点，递归的搜索图G。
步骤5）若所有顶点均已访问过，则搜索结束；否则，转步骤2。

先深搜索——结构化描述形式

主控函数

步骤1）{ 将图中每个顶点置未访问标记；

步骤2） 检查图中每个顶点v，如果v未访问过，则调用dfs(v)；

}

递归的搜索函数dfs(v)

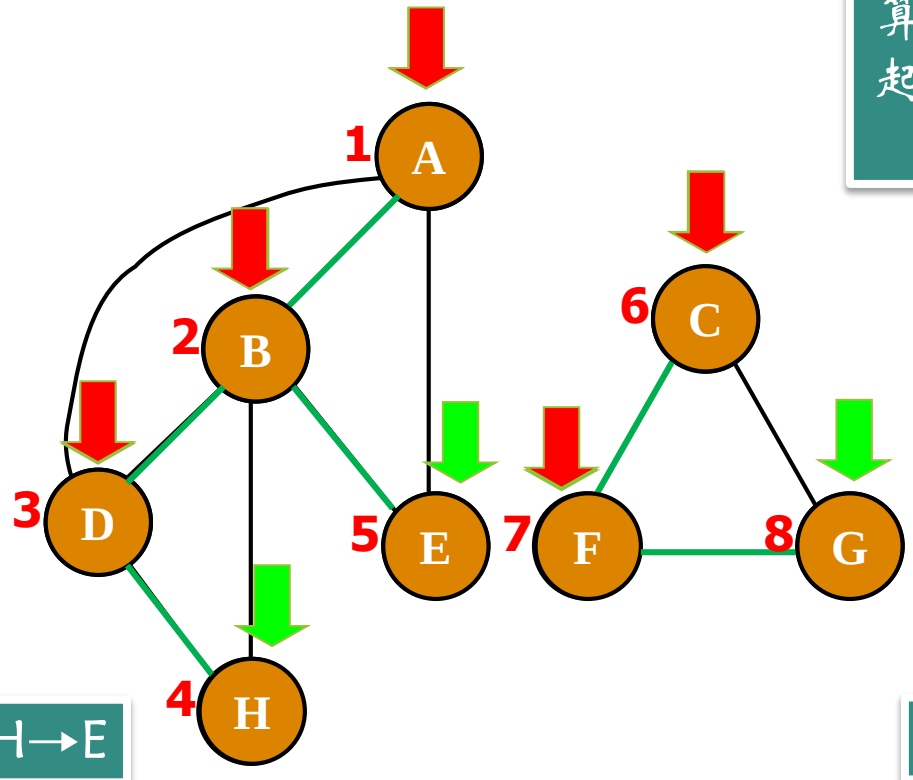
步骤3）{ 访问v，并对v作“已访问”标记；

步骤4） 检查v的每个邻接点w，如果w未访问过，则调用dfs(w)；

步骤5） 返回上一次调用点；

}

◆ 示例



算法每选择一次搜索起点，就产生一个连通分量

A→B→D→H→E

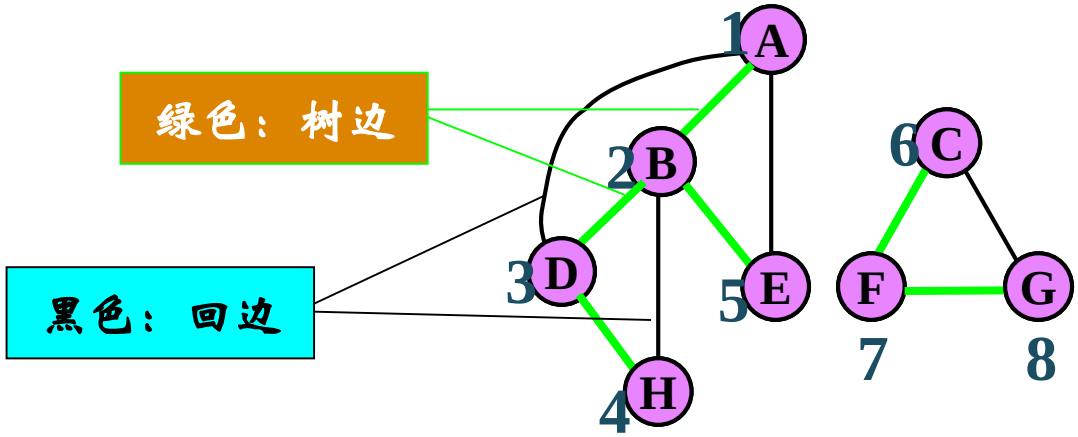
C→F→G

先深搜索

对无向图进行先深搜索的特点

- 连通图必然能够得到一棵生成树（先深生成树）
- 图的边被分成树边集T和回边集B

引起递归的边：树边（tree edge），其余的边是回边，或余边（back edge）



先深搜索

对无向图进行先深搜索的特点

主控函数

步骤1）{ 将图中每个顶点置未访问标记；

步骤2） 检查图中每个顶点v，如果v未访问过，则调用dfs(v)；

}

递归的搜索函数dfs(v)

步骤3）{ 访问v，并对v作“已访问”标记；

步骤4） 检查v的每个邻接点w，如果w未访问过，则调用dfs(w)；

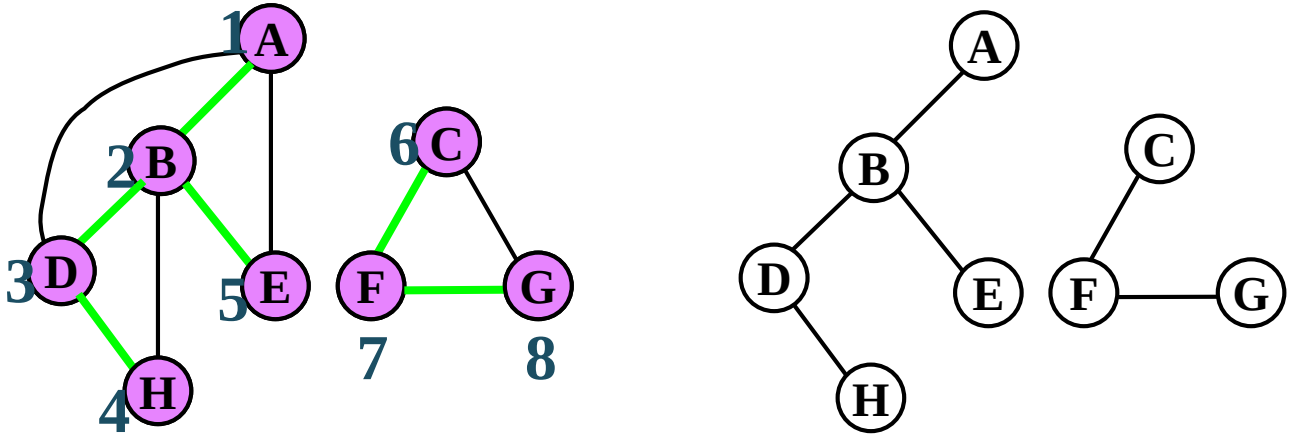
步骤5） 返回上一次调用点；

}

- 若搜索步骤2和4中，如有多个邻接顶点可选，可任选其一。搜索路线不唯一，生成树不唯一

对无向图进行先深搜索的特点

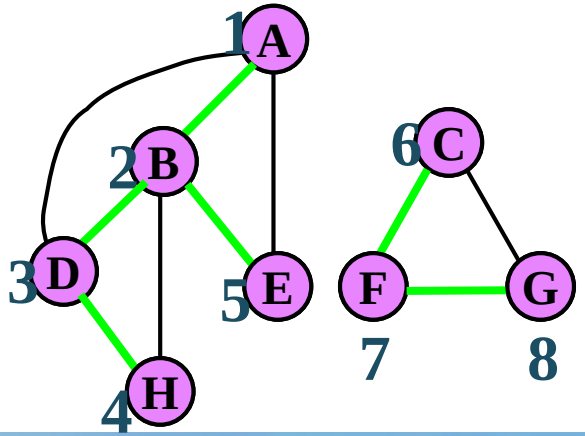
- 不是连通图，每个连通分量产生一棵生成树（先深生成林）

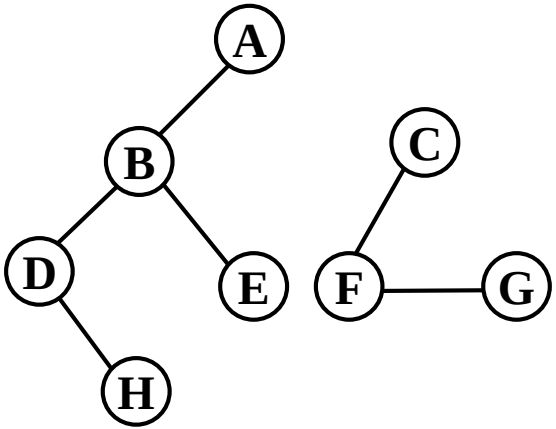


对无向图进行先深搜索的特点

- 不同的子生成树之间不可能有回边相连

Why?



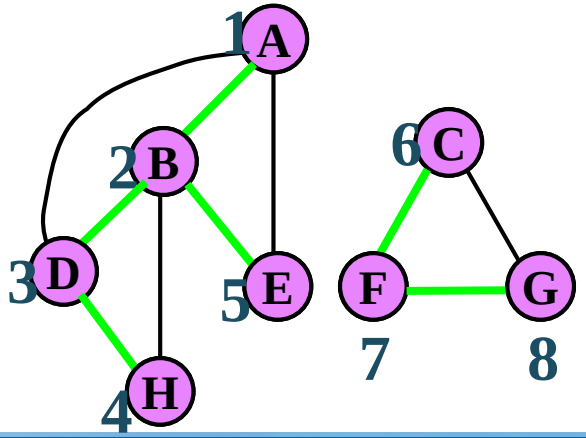


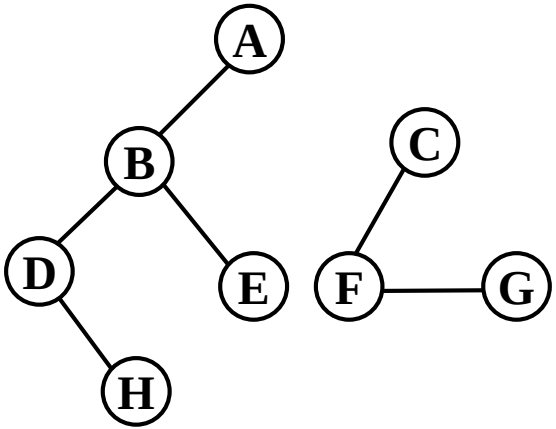
先深搜索

对无向图进行先深搜索的特点

- 不同的子生成树之间不可能有回边相连
- 祖先的先深号必小于子孙的先深号

对子孙的搜索先终止，而对祖先的搜索后终止

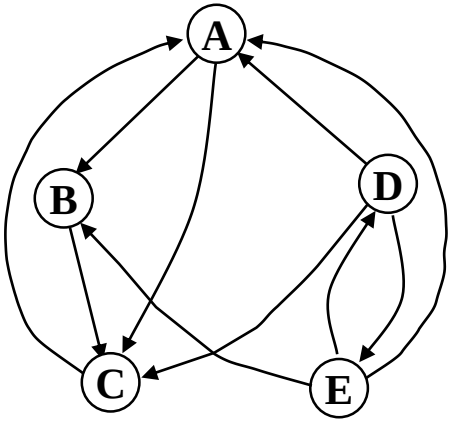




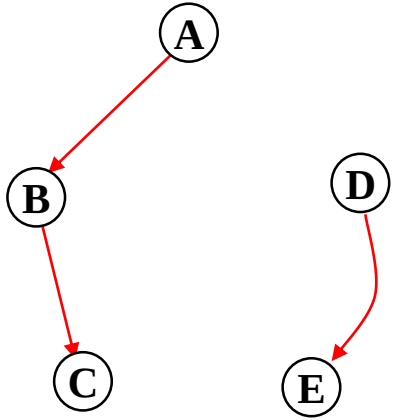
对有向图进行先深搜索的特点

（1）若强连通，可得到先深生成树

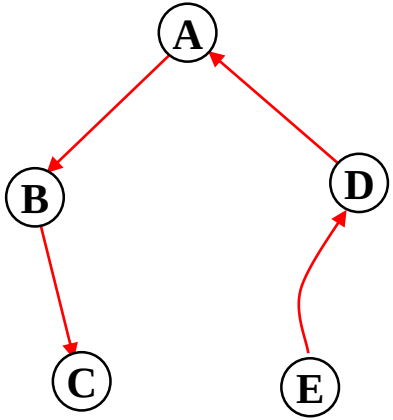
（2）非强连通，也不一定不能到先深生成树



非强连通图



A为起点的生成林



E为起点的生成树

对有向图进行先深搜索的特点

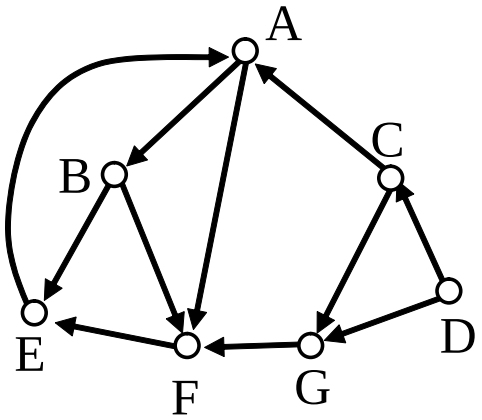
（3）可将边集E划分成： 树边T、回边B、向前边F和交叉边C

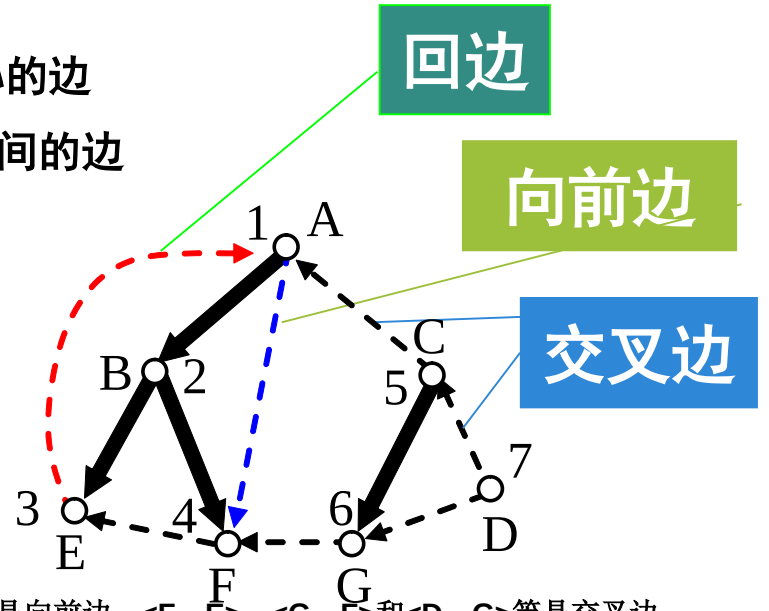
树边：引起递归调用的边

回边：由子孙射向祖先的边

向前边F（forward edges）：由祖先射向子孙的边

交叉边C（cross edges）：无祖孙关系顶点之间的边



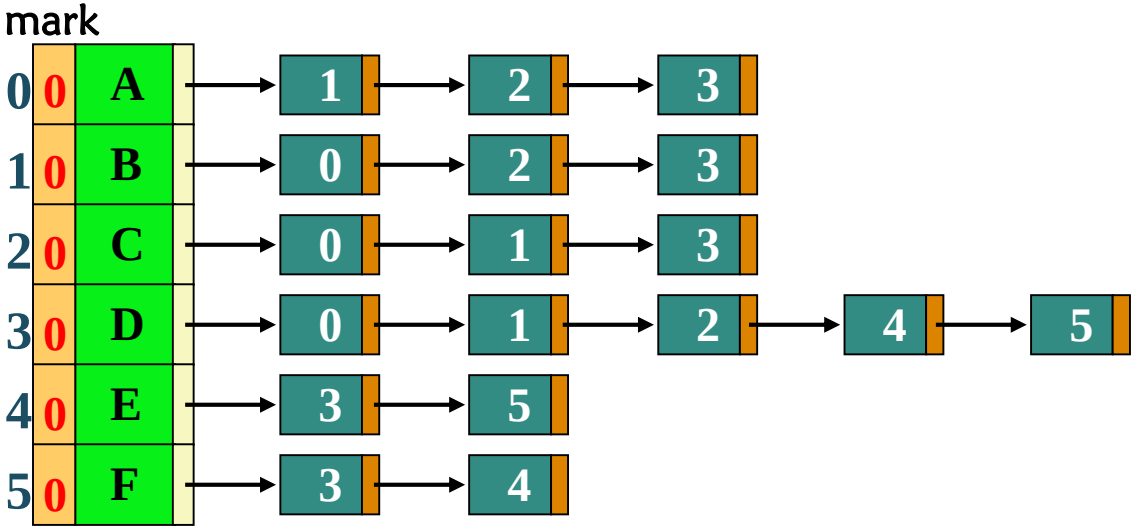


<E，A>是回边，<A，F>是向前边，<F，E>、<G，F>和<D，G>等是交叉边

先深搜索算法的实现

先深搜索的实现

- 图的存储形式：邻接表
  - 顶点结点含有访问否标志域mark
  - 未访问点，mark值为0；已访问点，mark值为1



先深搜索算法的实现

先深搜索的实现：主控函数

v是顶点编号

```
void main_1( )
    { int v;
1.    for (v=0;v<n;v++) L[v].mark=0;  //置未访问标记
2.    for (v=0;v<n;v++)   //检查各顶点是否访问过
        if(L[v].mark==0) //如果v未访问过
          dfs(v);  //选v作搜索起点，调用搜索函数
      …… //  其他处理操作
    }
```

先深搜索算法的实现

先深搜索的实现：搜索函数

```
void  dfs (int v)   //递归的搜索函数，v是顶点编号
  { Eptr  p;  int w;
3.   visit(v);  // 访问v
4.   L[v].mark=1; // 作访问标记
5.   p=L[v].firstedge;  //p指向v的邻接表首结点
6.   while (p!=NULL)  //检查v的所有邻接点
7.    { w=p->adjacent;  //w是v的邻接点
8.      if (L[w].mark==0)  dfs(w);  //若w未访问过，递归调用dfs
9.      p=p->next;  //递归返回后，再查看v的下一个邻接点
       }
   }
```

搜索函数dfs(v)花费O(n+m)时间

先深搜索算法的时间复杂性

设共有n个顶点，m条边

主控函数花费O(n)时间，因为只有两句并列循环

for (v=0;v<n;v++) ……

搜索函数dfs(v)花费O(n+m)时间

两部分总共花费O(n+m)时间

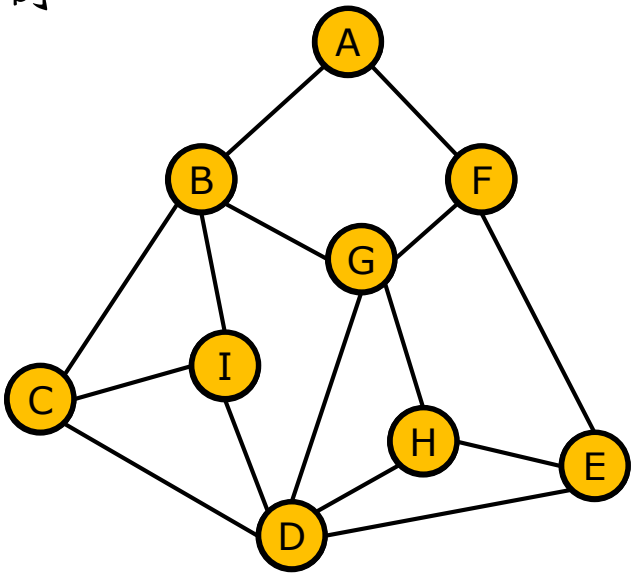
先广搜索

遍历规则（假设出发顶点为v_0）

图的先广搜索是树按层遍历的推广，算法描述如下：
步骤1）将图中所有顶点作“未访问过”标记。
步骤2）任选图中一个尚未访问过的顶点v作搜索起点。
步骤3）访问v。
步骤4）相继地访问与v相邻而尚未访问的所有顶点w_1，w_2，…，并依次访问与这些顶点相邻而尚未访问过的所有顶点。反复如此，直到找不到这样的顶点。
步骤5）若图中尚有未访问过的顶点，则转步骤2；否则，搜索结束。

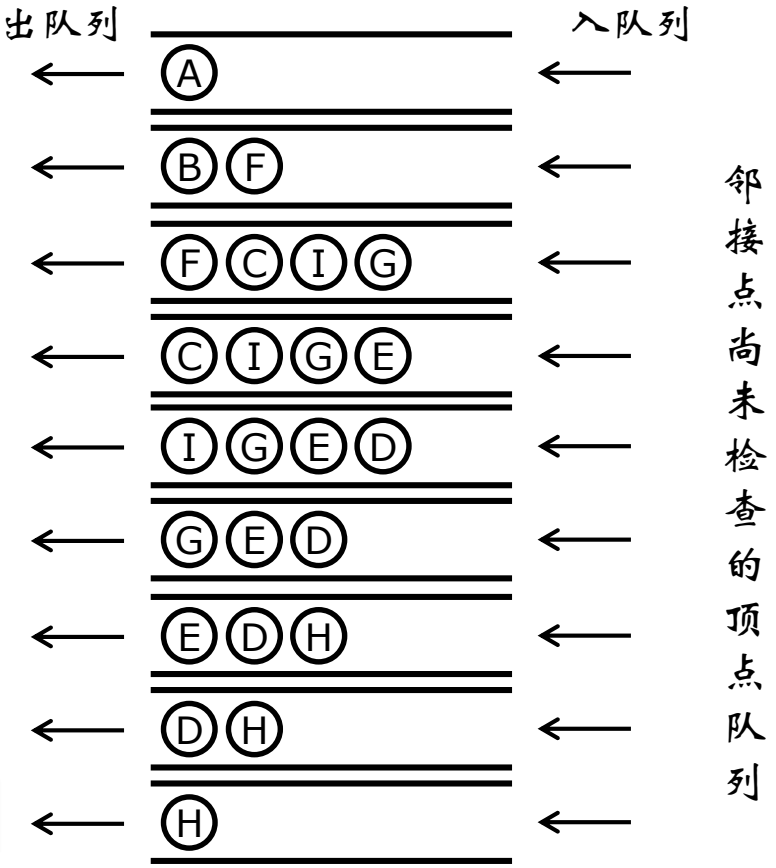
先广搜索算法运行示例

示例



A B F C I G E D H

先广搜索同样可以得到先广生成树（林）



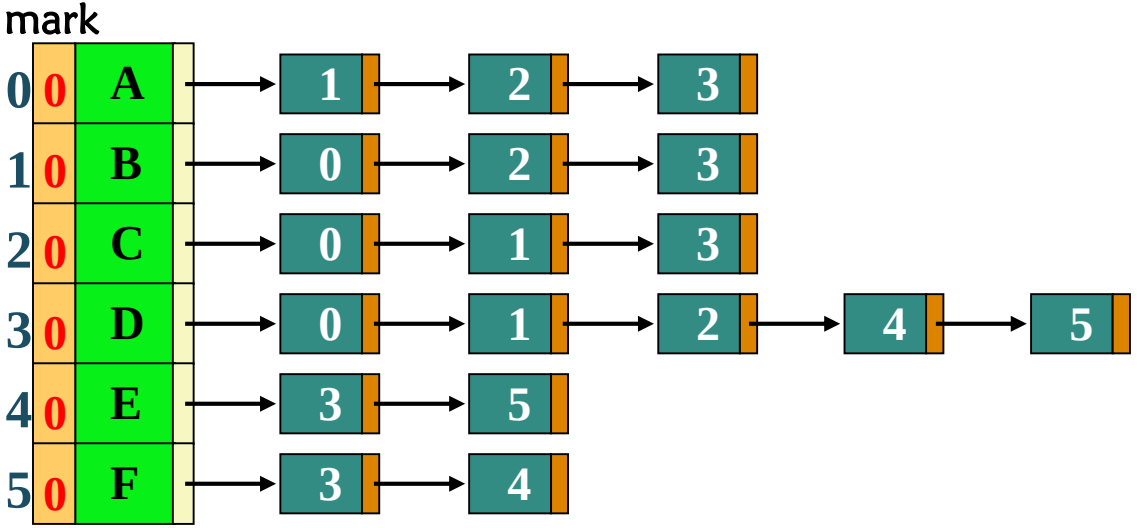
先广搜索算法的实现

图存储形式：邻接表

用一个队存放等待访问的顶点

顶点结点含有进队否标志域mark

未进队，mark值为0；已进过队，mark值为1



搜索函数

```
void  bfs( )
  { Eptr  p;     int  u,v,w,first,last,q[n];
1.  first=last=0;  //队列初始化
2.  for(u=0;u<n;u++) L[u].mark=0;
                    //初始化
3.  for(u=0;u<n;u++) //找未进过队的顶点
4.    if(L[u].mark==0)  //若u未进队
5.     {  q[last++]=u;  //u进队
6.        L[u].mark=1;  //置u进队标记
7.        while(first!=last) //当队列不空时循环
8.          { v=q[first++];  //v出队
9.    visit(v);  //访问v
10.   p=L[v].firstedge; //检查v的邻接点
11.    while(p!=NULL)
12.   { w=p->adjacent;  // w是v的邻接点
13.     if (L[w].mark==0)  //若w未进过队
14.       { q[last++]=w; L[w].mark=1; } w进队
15.         p=p->next; //找v的下一个邻接点
        } // 与句12对应
     } // 与句8对应，到队空
   } // 与句5对应，以u为起点的搜索结束
}
```

先广搜索算法的时间复杂性

算法时间复杂性计算

假设图共有n个顶点，m条边

对mark域初始化和找搜索起点各花费O(n)时间

每个顶点访问一次，访问花费O(n)时间

沿各顶点邻接表检查，花费O(m)时间

几部分总共花费O(n+m)时间

先深搜索与先广搜索

- 图的先深搜索是沿着图的一个邻接点纵向递归地访问图的顶点
- 图的先广搜索是依次横向扫描图的所有邻接点来访问图的顶点

The End, Thank You!