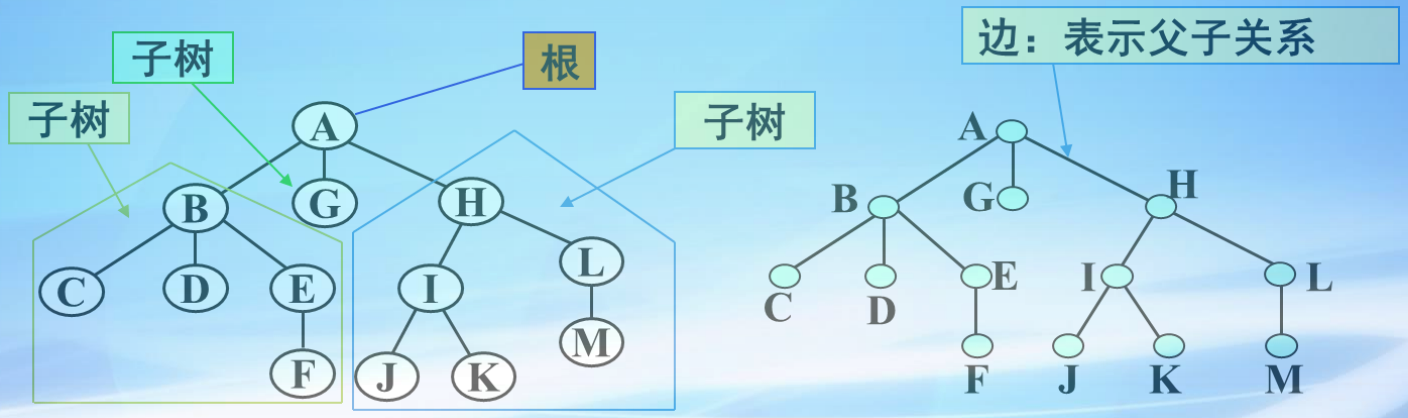
《数据结构》

遍历的应用

主讲人：陈卫卫



遍历的应用

搜索的目的不同，对顶点的访问操作也不同

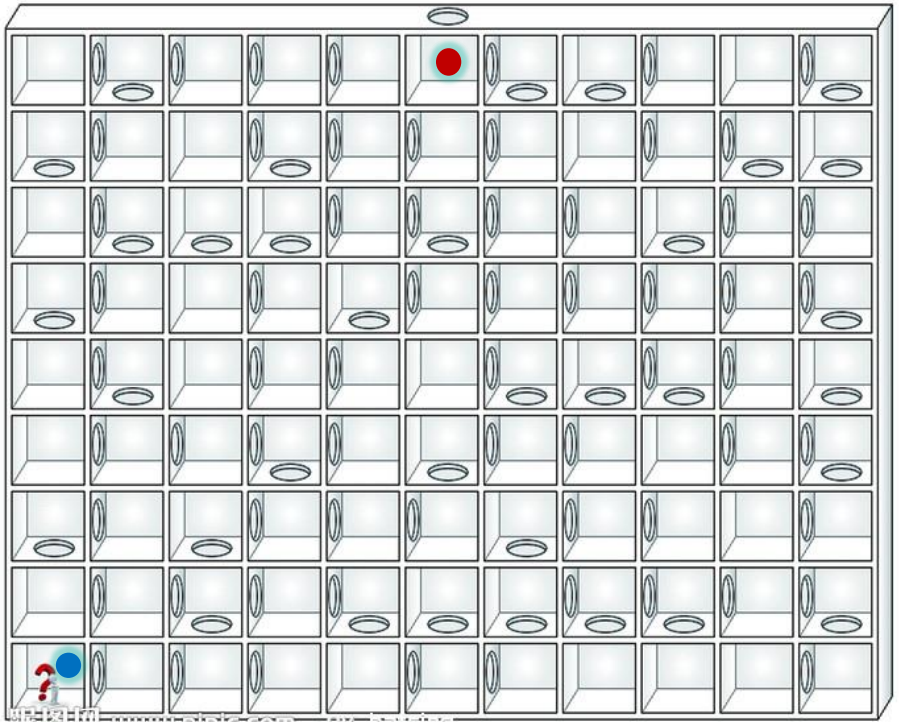
对无向图的先深搜索，可以

- 判断是否连通
- 找出连通分量、双连通分量
- 找出生成树或生成林

对有向图的先深搜索，可以

- 判断图中是否存在回路
- 找出强连通分量
- 对顶点进行柘朴排序

迷宫脱险



逃出

立体迷宫

大家有没有看出这个迷宫非常的特别呢？对！没错，这是三维立体迷宫，更真实、更立体。

TA是否能逃出这个“复式”的地下室呢？这就要看你的喽！

你的位置

“出口房间”的位置

实现方式

主控函数

```
void  dfsmain( )
{   int v;
1.  for(v=0;v<n;v++) L[v].mark=0;
          //对顶点作未访问标记
2.  for(v=0;v<n;v++)
      if (L[v].mark==0) dfs(v);
   ……
          //其他处理操作
}
```

实现方式

主控函数

```
void  dfsmain( )
{   int v;
1.  for(v=0;v<n;v++) L[v].mark=0;
          //对顶点作未访问标记
2.          v=0;
          dfs(v);
          ……
                        //其他处理操作
}
```

实现方式

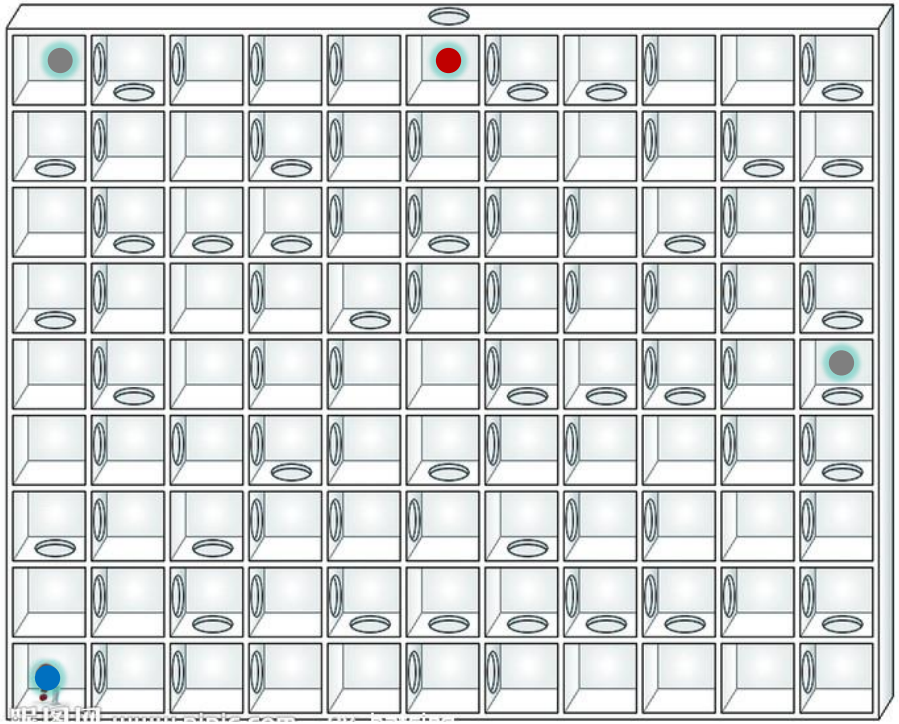
主控函数

```
void  dfsmain( )
{   int v;
1.  for(v=0;v<n;v++) L[v].mark=0;
          //对顶点作未访问标记
2.          v=0;  found=0;
          dfs(v);
          if (found==0)  无解;
            否则   输出路径s1;
}
```

实现方式

```
void  dfs(int v)          //v是顶点编号
{   Eptr  p;  int  w;
3.   push(s,v);           //将顶点v入栈
4.   L[v].mark=1;                  //置访问标记
5.   p=L[v].firstedge; //p指向v的邻接表
6.   while (p!=NULL && found==0) //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)
9.          if (w是终点) {found=1; strcpy(s1,s); break;}
10.                 else  {  dfs(w);
11.                          pop(s);} //将顶点w出栈
12.      p=p->next;     }
}
```

迷宫脱险2



11×9

逃出

立体迷宫

大家有没有看出这个迷宫非常的特别呢？对！没错，这是三维立体迷宫，更真实、更立体。

TA是否能逃出这个“复式”的地下室呢？这就要看你的喽！

你的位置

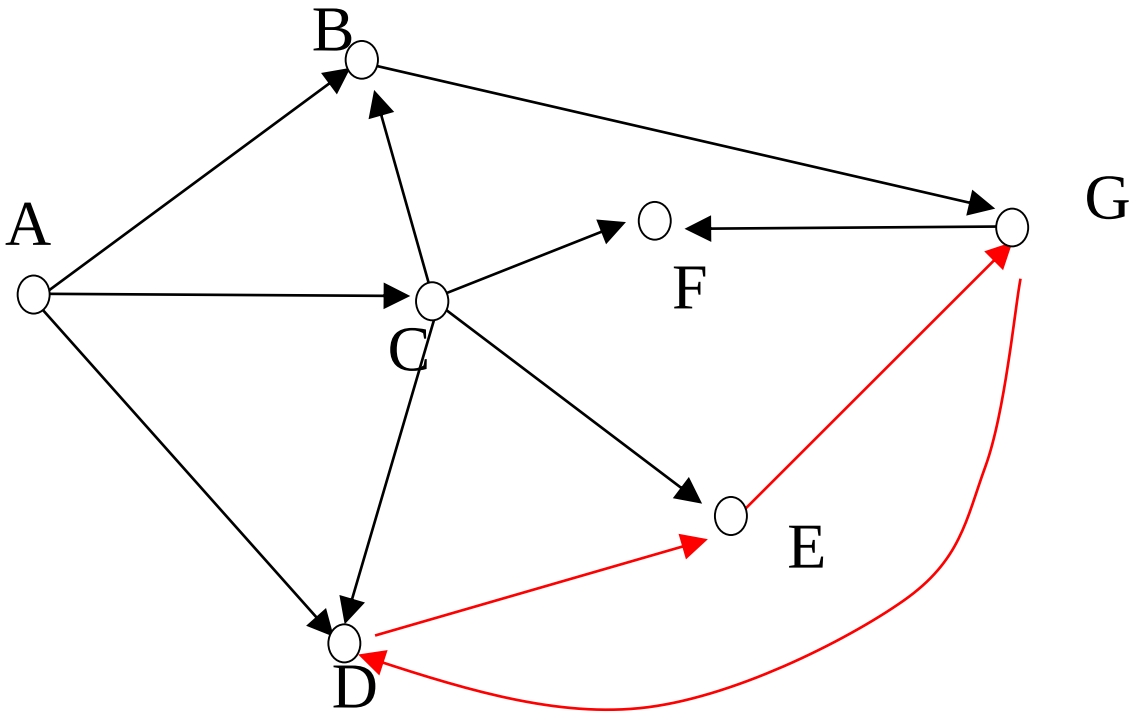
“出口房间”的位置

藏有“钥匙”的房间位置

走出迷宫的条件：1、到达出口房间；2、找到迷宫中的2个藏有“钥匙”的房间，拿到钥匙。

任务的死锁问题

工程项目可以实施吗？



判断有向图是否存在回路?

执行dfs(v) 期间，区分：T、B、F、C

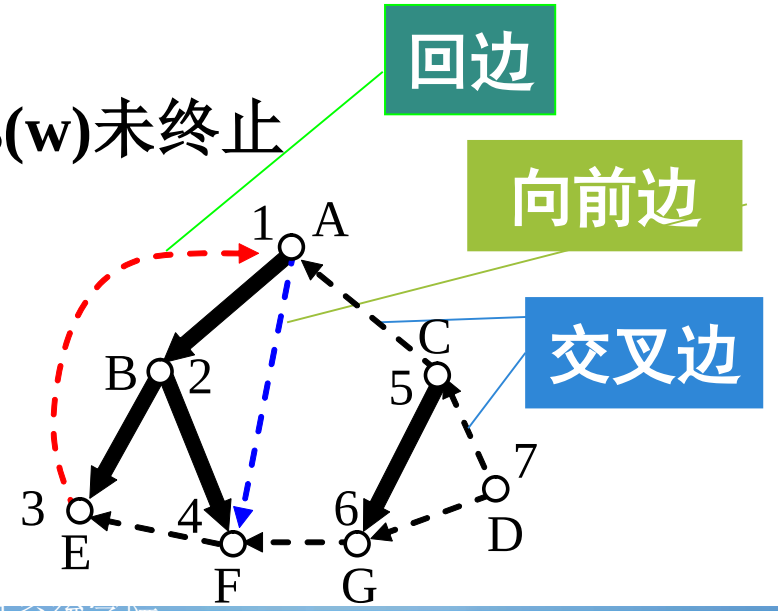
<v,w>是T，v是w父，w未访问过，dfs(v)未终止

<v,w>是F，v是w祖，dfs(w)已终止

<v,w>是C，v在w之右，dfs(w)已终止

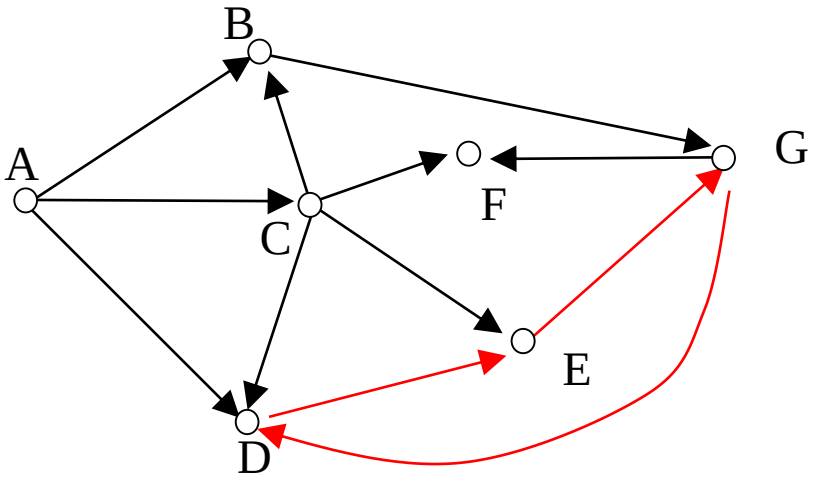
<v,w>是B，w是v祖，w已访问过，dfs(w)未终止

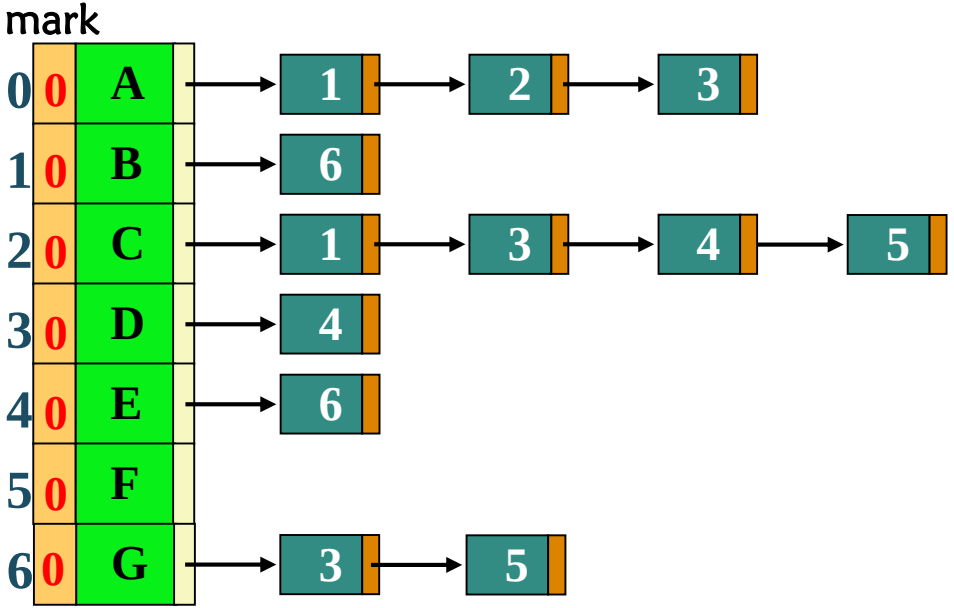
结论：如果w已访问过，但dfs(w)尚未终止，则<v，w>必是回边



判断有向图是否存在回路?

图的邻接表

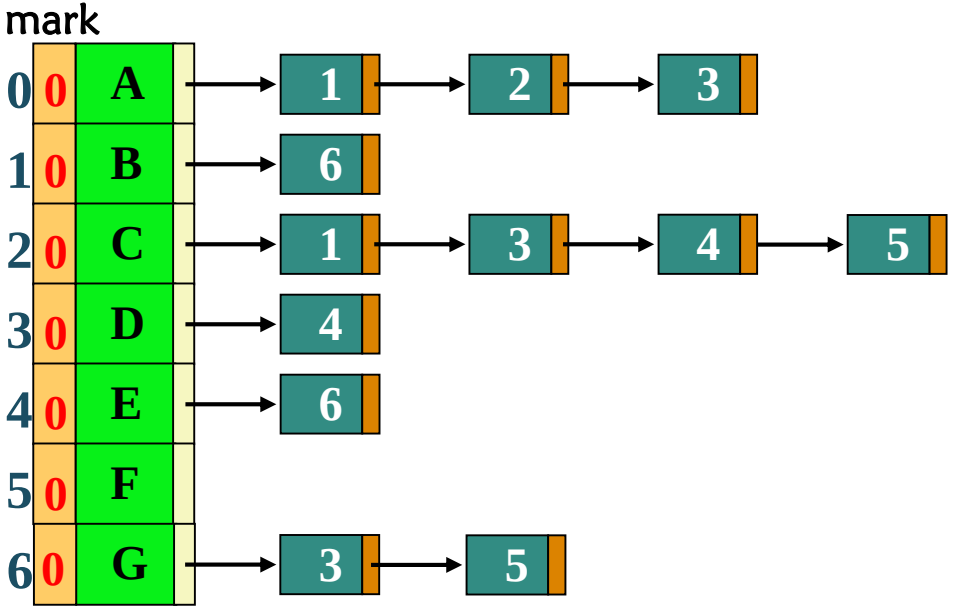




判断有向图是否存在回路?

改标记mark的作用：

- L[v].mark=0，表示v尚未访问过
- L[v].mark=1，表示v已经访问过，但dfs(v)尚未终止
- L[v].mark=2，表示v已经访问过，且dfs(v)已经终止



在调用dfs(v)之前，置L[v].mark=0

在进入dfs(v)之后，置L[v].mark=1

dfs(v)结束处，置L[v].mark=2

判回路算法描述

主控函数

```
void  dfsmain( )
 { int v;
    cycle=0;                    //cycle是整体量
    for(v=0;v<n;v++)  L[v].mark=0;
    for(v=0;v<n;v++)
      if(L[v].mark==0)dfs(v);
    if (cycle==1) printf("图中有回路\n");
      else  printf("图中没有回路\n");
 }
```

判回路算法描述

```
   void  dfs(int v)            //v是顶点编号
   {   Eptr  p;  int  w;
3.   visit(v);                  //访问v
4.   L[v].mark=1;                                    //置访问标记
5.   p=L[v].firstedge;                          //p指向v的邻接表
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);   //如果w未访问过，则递归调用dfs
            else   if(L[w].mark==1) cycle=1;  //dfs(w)尚未终止，置有回路标记
9.      p=p->next;          //（递归返回后）查看v的下一个邻接点
      }
10.  L[v].mark=2;             //置dfs(v)终止标记
    }
```

思考

- 微信的朋友圈关系如何判断？
  - 人与人之间的关系
  - 信息流动的关系
  - ……

The End, Thank You!