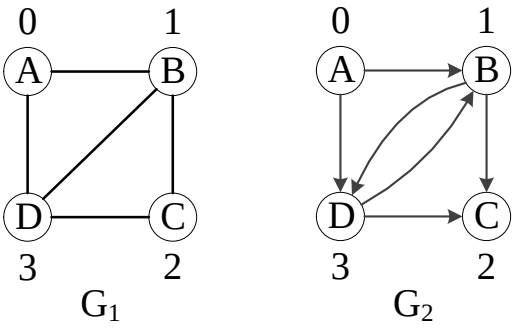
第 6 章 图

6.1 图类型

1. 概念

图（分无向图和有向图）、顶点、边（分无向边和有向边）、有向边的起点（弧尾）和终点（弧头）、完全图（分无向完全图和有向完全图）、稀疏图和稠密图、边的权、带权图（网）、子图、邻接点（注意有向图中邻接点的含义）、顶点的度、有向图的顶点的入度和出度、路径和简单路径、路径长度、回路（环）和简单回路（简单环）、连通图和连通分量、强连通图和强连通分量、生成树和生成森林



2. 图状结构

在无向图中，一个顶点与任何一个其他顶点之间可以没有无向边，也可以有一条无向边。在有向图中，一个顶点与任何一个其他顶点之间可以没有有向边，也可以有一条有向边，还可以有两条反方向的有向边。

6.2 图的实现

1. 邻接矩阵

A	B	C	D
0	1	2	3

G_1的

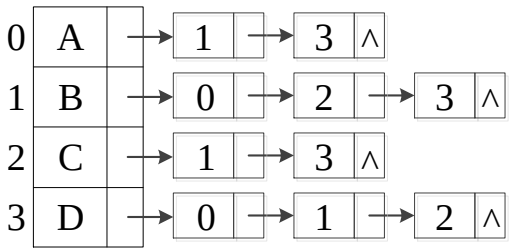
	0	1	2	3
0	0	1	0	1
1	1	0	1	1
2	0	1	0	1
3	1	1	1	0

A	B	C	D
0	1	2	3

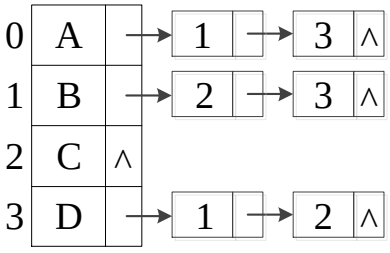
G_2的

	0	1	2	3
0	0	1	0	1
1	0	0	1	1
2	0	0	0	0
3	0	1	1	0

2. 邻接表

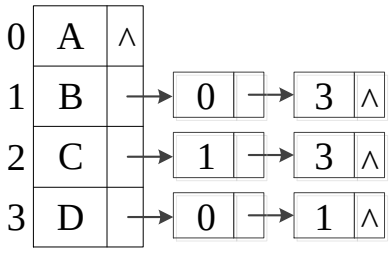


G_1的



G_2的

3. 逆邻接表（有向图独有）



G_2的

6.3 图的遍历

1. 说明

因为图的任一种遍历的序列不仅与该遍历方案有关，还与第一个访问的顶点（人为指定）以及顶点和边的排列顺序（人为指定）有关，所以不是唯一的。

为防止遍历时在回路中转个没完，人们想出了一个解决方案，就是在遍历过程中，给每个顶点赋予一种状态：等待或就绪或已访问。

2. 深度优先搜索

图的深度优先搜索是树的先序遍历的推广，递归形式，要用辅助栈。

算法 1. 深度优先搜索

所有顶点的状态都设置为等待;

for（每个顶点 v_i）

{

if（v_i的状态为等待）

{

v_i进栈;

while（栈非空）

{

栈顶顶点出栈，访问，其状态改为已访问;

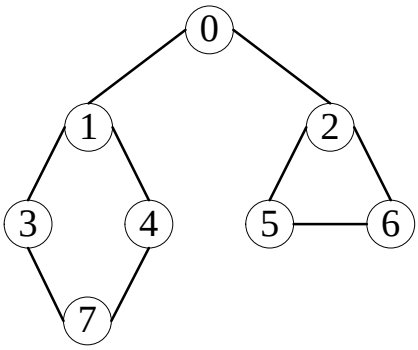
该顶点的所有状态为等待的邻接点(按序号从大到小)进栈(若某顶点栈中已有，则只保留刚进栈的);

}

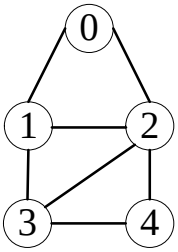
}

}

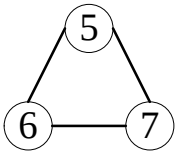
图用邻接矩阵实现时，T(n,e)=O(n^2)。图用邻接表实现时，T(n,e)=O(n+e)。

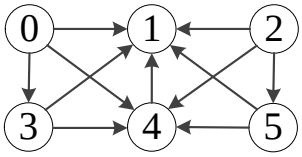


G_3



G_4





G_5

3. 广度优先搜索

图的广度优先搜索是树的层序遍历的推广，非递归形式，要用辅助队列。

算法 2. 广度优先搜索

所有顶点的状态都设置为等待;

for (每个顶点 v_i)

{

    if (v_i的状态为等待)

    {

        v_i进队，其状态改为就绪;

        while (队列非空)

        {

            队头顶点出队，访问，其状态改为已访问;

            该顶点的所有状态为等待的邻接点(按序号从小到大)进队，其状态改为就绪;

        }

    }

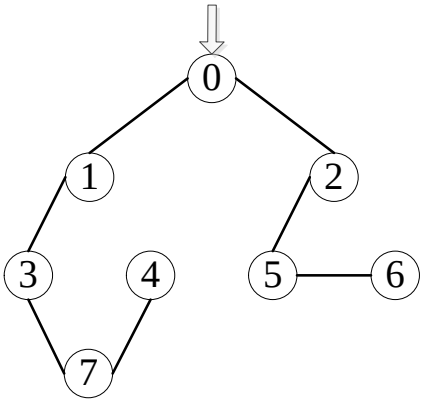
}

图用邻接矩阵实现时，T(n,e)=O(n^2)。图用邻接表实现时，T(n,e)=O(n+e)。

4. 遍历所得到的生成树和生成森林

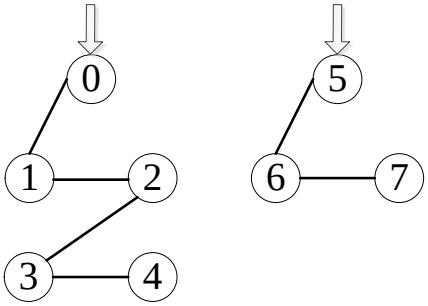
由图的所有顶点和遍历时所经过的边构成的生成树或生成森林，一般不是唯一的。若无向图是连通图，则构成生成树，否则构成生成森林。若有向图是强连通图，则构成生成树，否则一般构成生成森林。

通过深度优先搜索得到的生成树或生成森林称为深度优先生成树或生成森林。通过广度优先搜索得到的生成树或生成森林称为广度优先生成树或生成森林。



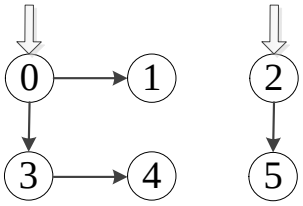
G_3的深度优先生成树

G_3的深度优先序列：0，1，3，7，4，2，5，6



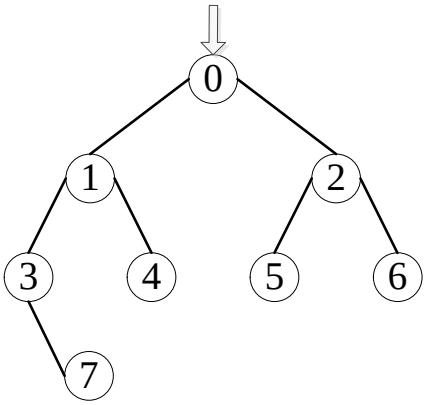
G_4的深度优先生成森林

G_4的深度优先序列：0，1，2，3，4，5，6，7



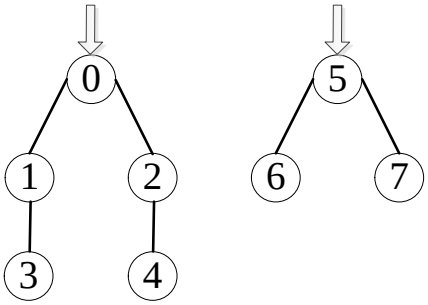
G_5的深度优先生成森林

G_5的深度优先序列：0，1，3，4，2，5



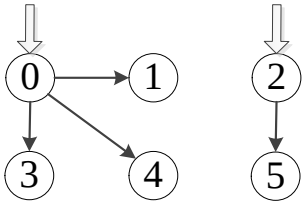
G_3的广度优先生成树

G_3的广度优先序列：0，1，2，3，4，5，6，7



G_4的广度优先生成森林

G_4的广度优先序列：0，1，2，3，4，5，6，7



G_5的广度优先生成森林

G_5的广度优先序列：0，1，3，4，2，5

6.4 图的连通性问题

1. 无向图的连通分量

   可以通过深度或广度优先搜索求无向图的连通分量。

2. 有向图的强连通分量

3. 带权连通图的最小（代价）生成树

（1）概念

最小生成树

（2）普里姆（Prim）算法

p.217

$T(n,e)=O(n^2)$，适合于稠密带权连通图。

（3）克鲁斯卡尔（Kruskal）算法

p.216

$T(n,e)=O(e\log_2 e)$，适合于稀疏带权连通图。

（4）说明

一个带权连通图的最小生成树可能不止一棵。

6.5 最短路径

1. 单源最短路径

   迪杰斯特拉（Dijkstra）算法

设 G=（V，E）是带权图，SPT=(V’，E’)是 G 的最短路径树，起初 V’={源点}，E’={}。反复执行下面的操作，直至 V’=V 为止：对 V-V’中的是 V’中的顶点的邻接点的所有顶点，考察源点到其中哪个顶点的哪条路径最短，将该顶点并入 V’，该路径上的最后一条边并入 E’。

该算法不适用于边的权出现负数的情形。

2. 各对顶点间的最短路径

弗洛伊德（Floyd）算法

设 G=（V，E）是带权图，V={1，2，3，…，n}，C 是 G 的带权邻接矩阵（若无第 i 个顶点到第 j 个顶点的边，则 C[i][j]=+∞），A 是 n×n 矩阵，起初 A_0[i][j]等于 C[i][j]（i≠j）或 0（i=j）。对 A 按下式进行 n 次迭代：A_k[i][j]=min（A_{k-1}[i][j]，A_{k-1}[i][k]+A_{k-1}[k][j]）（k=1，2，3，…，n；A_k表示第 k 次迭代后的 A）。A_n[i][j]等于第 i 个顶点到第 j 个顶点的最短路径长度。

该算法适用于边的权出现负数的情形，但不能有路径长度是负数的回路。

6.6 拓扑排序

1. 概念

有向无环图（DAG）、工程、活动（子工程）、顶点表示活动的网（AOV 网）

2. 拓扑排序

拓扑排序用于判断工程能否实施，即判断 AOV 网是否无环。

拓扑排序算法 p.228

图用邻接表实现时，T(n,e)=O(n+e)。

若在一个有向无环图中，顶点 v_i 是 v_j 的前驱，则在其拓扑有序序列中，v_i 一定在 v_j 的前面。

6.7 关键路径

1. 概念

边表示活动的网（AOE 网）

2. 关键路径

求关键路径是为了获得工程的最短工期。

源点 v_0 表示工程的开始，汇点 v_{n-1} 表示工程的结束。

事件 v_i 的最早发生时间 ve(i)=max(直接前驱的最早发生时间+之间边的权)，规定 ve(0)=0。

事件 v_i 的最迟发生时间 vl(i)=min(直接后继的最迟发生时间-之间边的权)，规定 vl(n-1)=ve(n-1)。

活动 a_i 的最早开始时间 e(i)=起点的最早发生时间。

活动 a_i 的最迟开始时间 l(i)=终点的最迟发生时间-边的权。

求关键路径算法 p.231

图用邻接表实现时，T(n,e)=O(n+e)。
一个 AOE 网的关键路径可能不止一条。
加快某个关键活动不一定能缩短工程的工期。