**广度优先遍历**：O(V+E)

1队列的性质

队列中包含的是灰色节点的集合

2 正确性：

每个节点在进入队列之前被涂上灰色，在出队时被涂上黑色。

反证法

3 22.2-7 求树的直径

4 时间复杂度：O(V+E)

5 广度优先树

**深度优先遍历 theta（V+E）**

1性质：

括号定理：完全包含为后代 相互分离不是后代

后代区间的嵌套

白色路径定理：在有向或无向图G=（V,E）的深度优先森林中，结点v是结点u的后代当且仅当在发现结点u的时间u.d存在一条从结点u到结点v的全部由白色节点所构成的路径。

边的分类：

树边：第一次探索（u，v）时，v为白色说明是一条树边

后向边：u连接到一个祖先v的边，v为灰色说明是一条后向边

前向边：u连接到一个后代的边 v为黑色

横边：其他所有的边 同树非后代或者非同树 v为黑色

2 22.3-13 单连通图

3 深度优先树

**拓扑排序 theta（V+E）**

1 过程

调用深度优先遍历计算每个v的结束时间v.f 每个结点结束时将该点插入链表

2 有向图五环当且仅当进行深度优先搜索不产生后向边。

3 正确性 只需证明对于任意一对不同的结点u,v属于V如果F包含（u,v）那么v.f<u.f

4 22.4-2 输出从结点s到结点t之间的简单路径的数量

Add a field to the vertex representation to hold an integer count. Initially, set vertex t’s count to 1 and other vertices’ count to 0. Start running DFS with s as the start vertex. When t is discovered, it should be immediately marked as finished (BLACK), without further processing starting from it. Subsequently, each time DFS finishes a vertex v, set v’s count to the sum of the counts of all vertices adjacent to v. When DFS finishes vertex s, stop and return the count computed for s.

5 22.4-5重复寻找入度为0结点，输出该结点，将该结点及从其发出的边从图中删除。

首先,运行DFS或者是BFS可以在O(V+E)的时间内统计出每个点的入度和出度,以后在删除边的时候要维护这些信息. 每次输出入度为0的那个点并删除其出边,并维护信息,因此有E条边和V个点,所以要执行O(V)次输出和O(E)次的删除.所以总的运行时间是O(V+E)。

如果图有回路,那么有时候可能没有入度为0的点.

**强连通分量**

**1 过程**

**2 引理22.13**

**3 引理22.14**

**4 推论22.15**

**5 定理22.16 正确性**

**归纳假设算法第三行所生成的前面k棵树都是强连通分量，现在需要考虑第（k+1）棵树。**

**对于没访问过的强连通分量C’来说u.f=f(C)>f(C’)，C中所有节点都是白色的，所以都是u的后代。二在GT中所有从C发出的边只能指向已经访问过的强连通分量。**

**6 22.5-7**

强连通组件算法, 按第 2 次 DFS(GT) 强连通组件生成顺序编号, 假设是 SCC1, SCC2, ..., SCCn

如果存在边 (SCC1, SCC2), (SCC2, SCC3), ..., (SCCn-1, SCCn) , 即组件间边形成线性的链,

则图 G 是 半连通的. 稍后详细解释.

1. 强连通组件图算法, 时间 O(V+E)

2. 按强连通组件 SCC 顺序编号, 假设是SCC1, SCC2, ..., SCCn, 并求出组件 SCC 的邻接表,

这步骤可以在 (1) 中完成, 所以时间 O(V+E)

3. 遍历 SCC 邻接表, 如果对于范围 i = 1, 2, ..., n-1, 强连通组件SCCi.adj 中存在 SCCi+1 ,

既含有边 (SCCi, SCCi+1), 强连通组件图是线性链, 则 G = (V, E) 是 semiconnected, 时间 O(V+E).

详细解释, 按算法理论的支持, SCCi+1 不存在到SCCi 的边, 如果没有 (SCCi, SCCi+1),

则两者均无法到达对方, 即不符合 G 是 semiconnected 的概念. 算法时间复杂度 O(V+E)

**7 22-3 欧拉回路**

**最小生成树**

**1 树和它的性质**

**2 通用算法：每变循环之前A是某棵最小生成树的子集。每一步添加一条安全边。**

**3 最小生成树的三个性质**

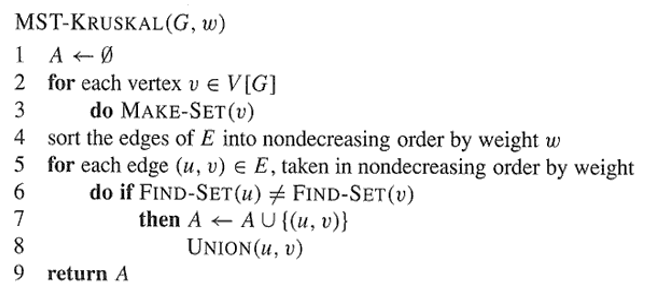
**图中权重最小的边一定在最小生成树中**

**如果每个权重都不同则最小生成树是唯一的**

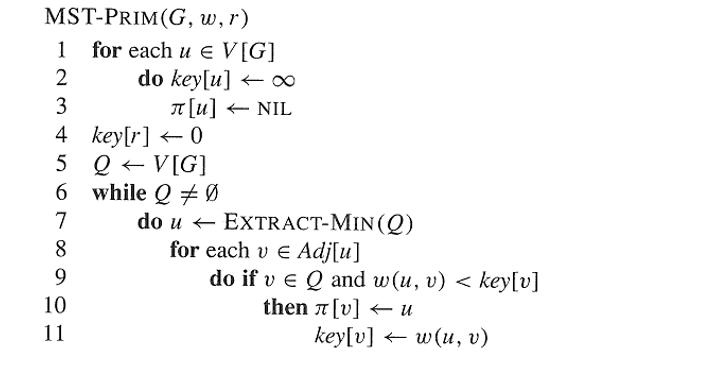
**将普通的生成树的边从小到大排列,每个边都大于等于对应最小生成树的边。**

**4 定理23.1 轻量级边是安全的**

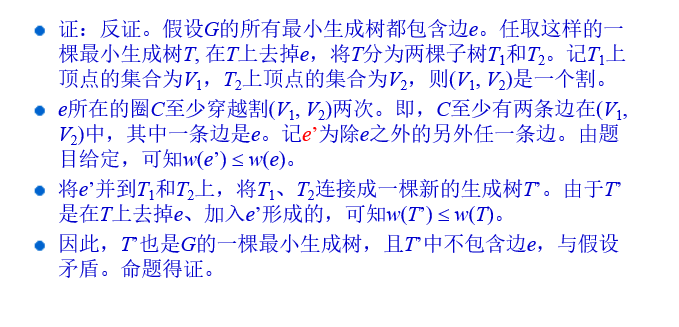
**5 Kruscal算法：O（ElgV）E小于V^2**



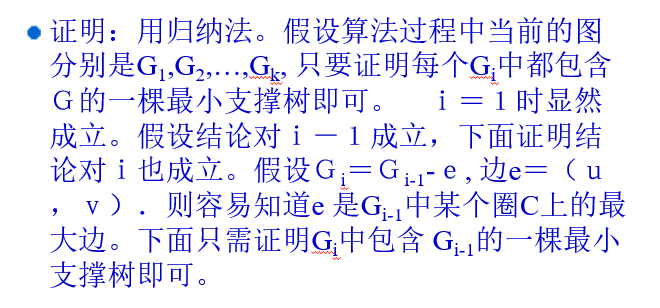
**Prim O（ElgV） 如果使用斐波那契堆来实现最小优先队列Q O（E+VlgV）**

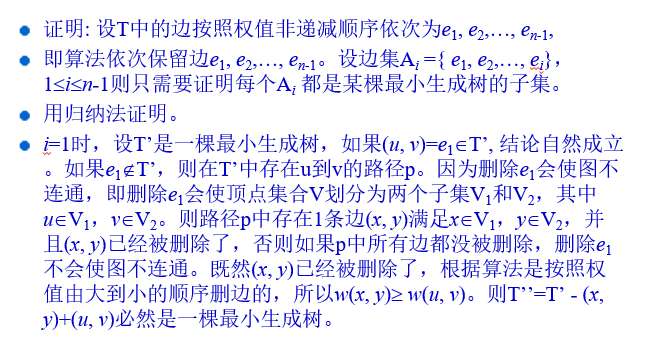


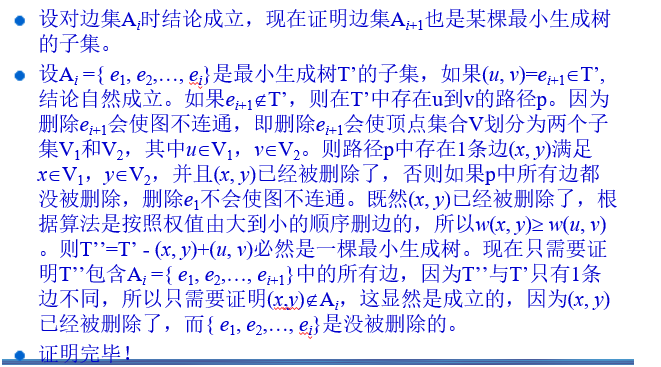
**6 23.1-5 G中存在一科不包含某条环路上权重最大的边的最小生成树**



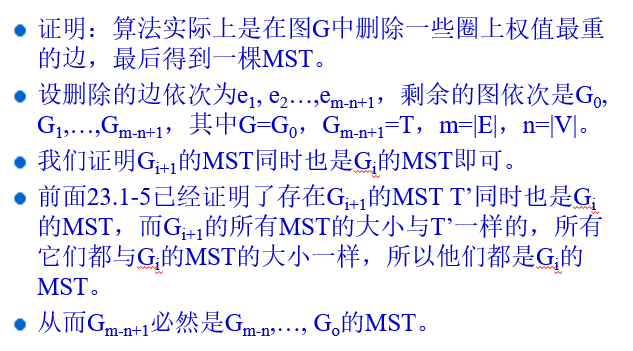
**7 23-4 a**







**8 23-4 c**



**单源最短路径**

**1 最短路径的几个变体**

**单目的地 单节点对 所有节点对最短路径问题**

**2 最短路径和松弛定理的性质**

**三角不等式 对于任何边 δ(*s*, *v*) ≤ δ(*s*, *u*) + *w*(*u*, *v*)**

**上界性质 *d*[*v*] ≥ δ(*s*, *v*)**

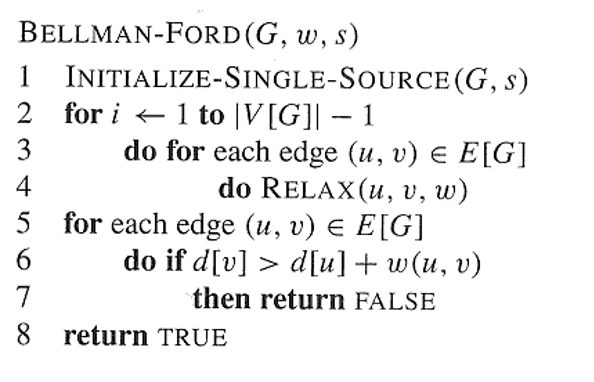
**非路径性质 s到v之间不存在路径 *d*[*v*] = δ(*s*, *v*) = ∞.**

**收敛性质**

**路径松弛定理**

**前驱子图性质**

**3 贝尔曼富特算法O（VE）**



**正确性**

**可以通过路径松弛定理来证明,v0到vk的每一条边都被松弛一遍，k<=|V|-1。**

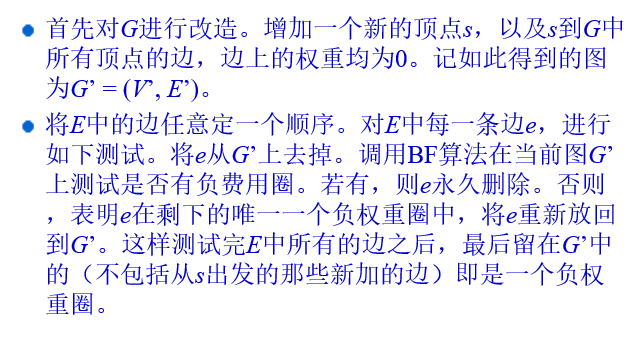
**如果没有负圈，根据三角不等式**



**所以不可能返回false**

**如果有福泉，**

**24.1-6**

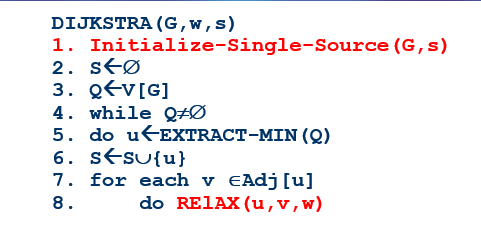


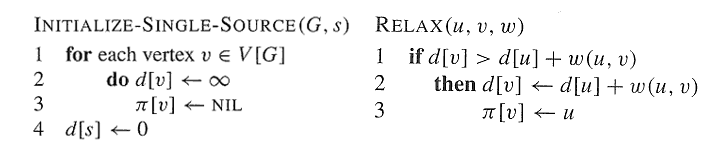
**4 DAG theta(V+E)**

**先进行拓扑排序然后对每个结点，对从该节点发出的所有的边进行松弛操作**

**5 Dijkstra**

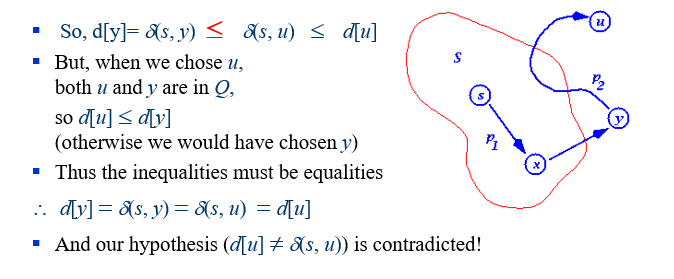
**要求所有边的权重为非负。**





**正确性证明：**

U每次被加入集合S时有u.d=最短路径。设u是第一个在加入到集合S时使得不变式不成立的结点。在将u加入到S中时，有S不为空，一定存在某条从结点s到结点u的路径，否则u.d=无穷大。也就只是存在一条从s到u的最短路径p。



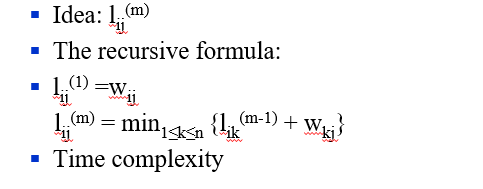
* + - **O( V2 )**
    - **O(VlgV+E) or less**

**6 24.3-2**

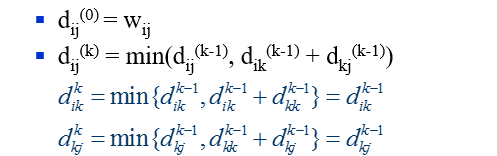
**7 24.3-3**

**所有节点对的最短路径**

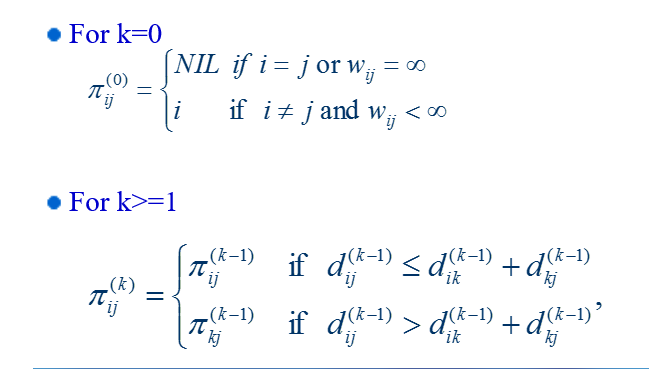
**1 矩阵乘法 theta(n^4) 改进后theta(n^3lgn)**



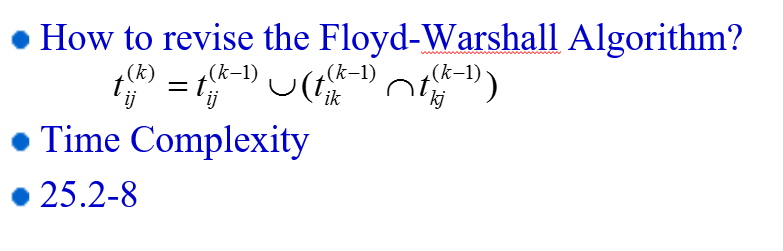
**2 Floyd算法 theta（n^3）**



**构造路径**



**3传递闭包**

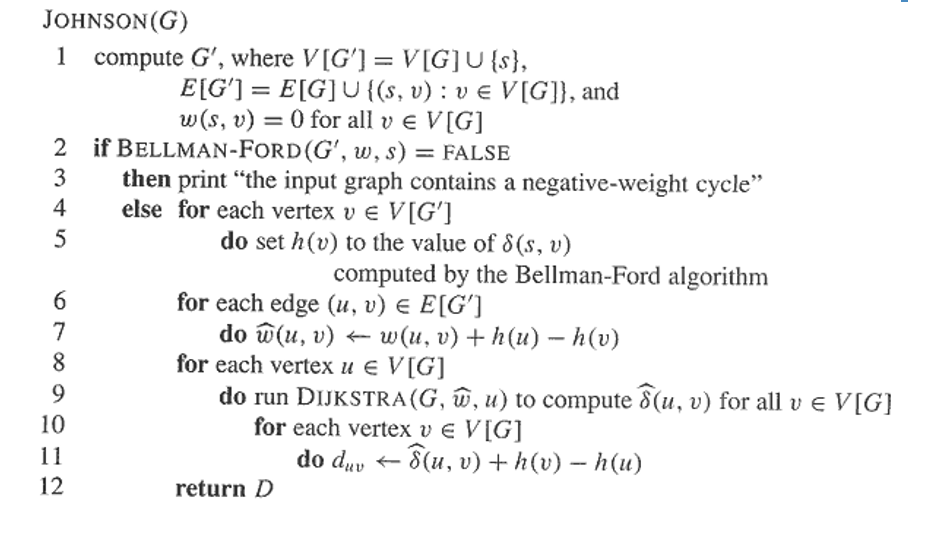


**时间复杂度 Theta(n^3)**

**25.2-8 给出一个O(VE)的算法求有向无环图的传递闭包**

**Johnson 算法**

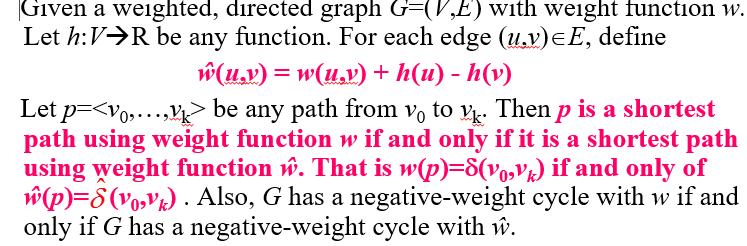
**1 过程**

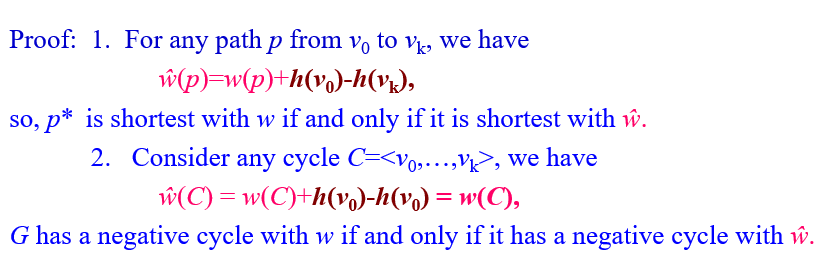


**2 时间复杂度**

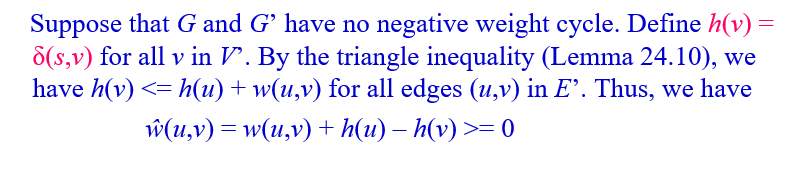
**O(V^2lgV+VE)**

**3 重新赋予权重不改变最短路径**



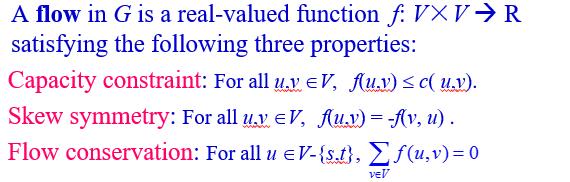


**4 重新赋值: h(v) = δ(s,v),**



**最大流**

**1 定义和性质**



**2 残存网络**

3 Edmods-Karp算法 O(VE^2)

每条边的权重为单位距离,选择的是最短路径。