[关闭]
@rayiooo 2018-07-06 19:18 字数 4091 阅读 70

算法导论-图论 复习

计算机考试复习

实时更新。

资料更新度：8更。

更新完成。

> 发布地址：作业部落
>
> 发布总地址：Blog

Author	爱吃大板
Email	rayiooo@foxmail.com
Time	2018.7

优质的复习资料

- 算法导论——图论总结

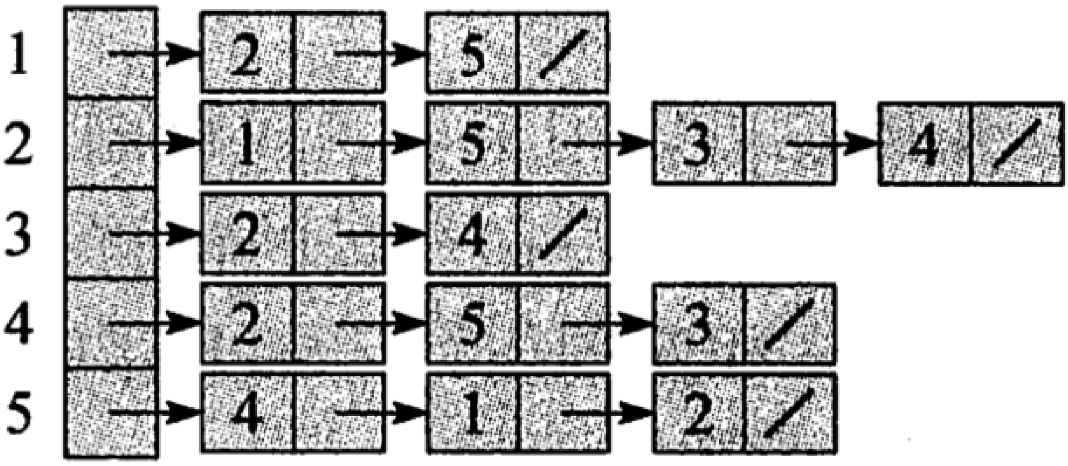
1 基本的图算法

1.1 图的表示

图G = (V, E)。

邻接链表

适用于稀疏图。



邻接矩阵

适用于稠密图。

$$
\begin{array}{c|ccccc|}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & 0 & 1 & 0 & 0 & 1 \\
2 & 1 & 0 & 1 & 1 & 1 \\
3 & 0 & 1 & 0 & 1 & 0 \\
4 & 0 & 1 & 1 & 0 & 1 \\
5 & 1 & 1 & 0 & 1 & 0 \\
\hline
\end{array}
$$

1.2 BFS：广度优先搜索

Queue实现，O(V + E)。

1.3 DFS：深度优先搜索

递归或Stack实现，O(V + E)。

概念

- **u.d**： 发现u的时间，即发现时间/入栈时间。
- **u.f**： 遍历完子代后回到u的时间，即完成时间/出栈时间。

白色路径定理

> 在有向或无向图G的DFS森林中，v是u的后代，当且仅当发现u时（即u.d时），存在由u到v的全部由白色点构成的路径。

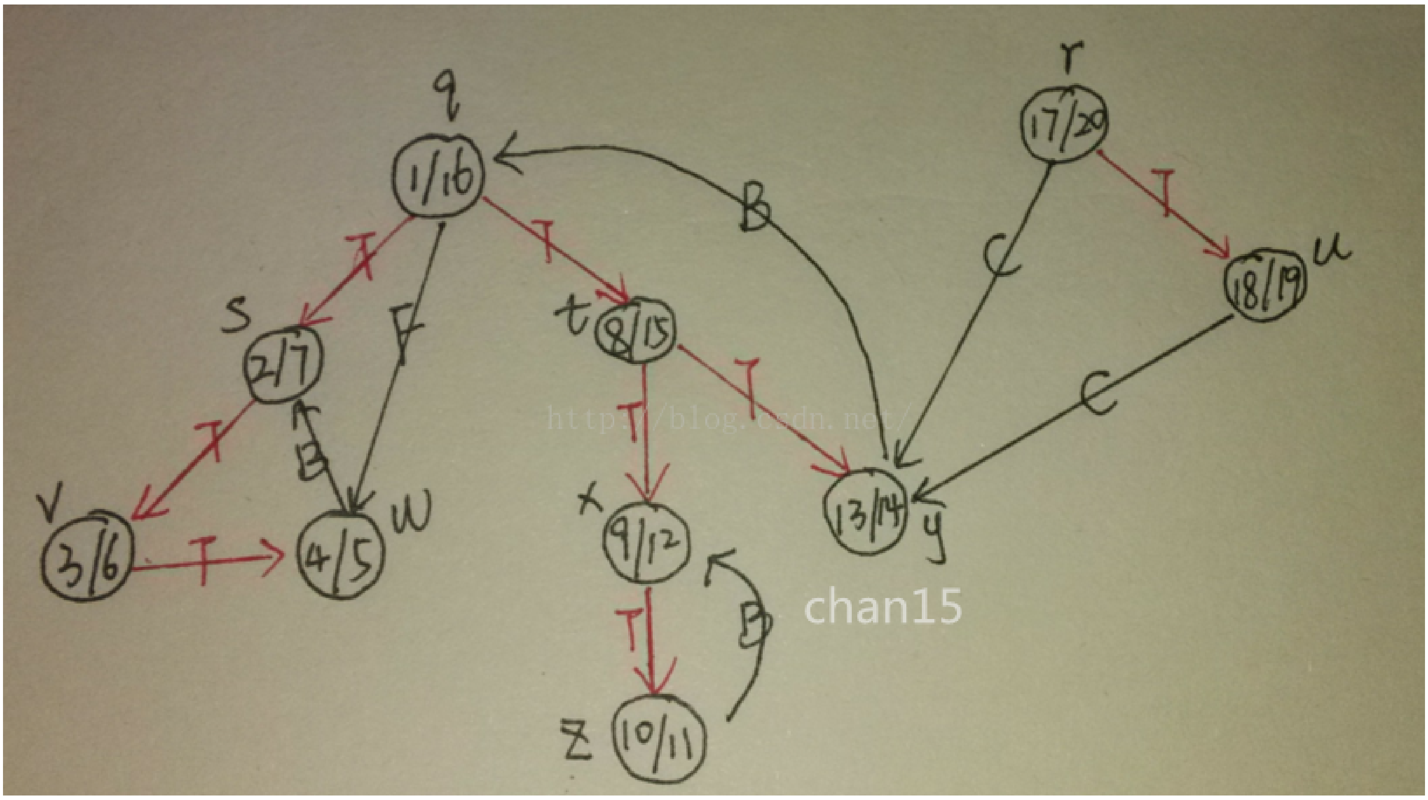
边的分类

1. **树边**： （π[u]，u）。
2. **反向边/返回边**： 后代指向祖先的边。
3. **前向边**： 祖先指向后代的非树边。
4. **交叉边**： 没有祖先后代关系。

无向图中只有树边和返回边。

题目22.3-2：

给出深度优先搜索算法在图22-6上的运行过程。假定深度优先搜索算法的第5~7行的 for 循环是以字母表顺序依次处理每个结点，假定每条邻接链表皆以字母表顺序对立面的结点进行了排序。请给出每个结点的发现时间和完成时间，并给出每条边的分类。



[更多题目：算法导论22.3深度优先搜索 练习总结]

1.4 拓扑排序

可以将图的拓扑排序看作是将图的所有结点在一条水平线上排开，图的所有有向边都从左指向右。

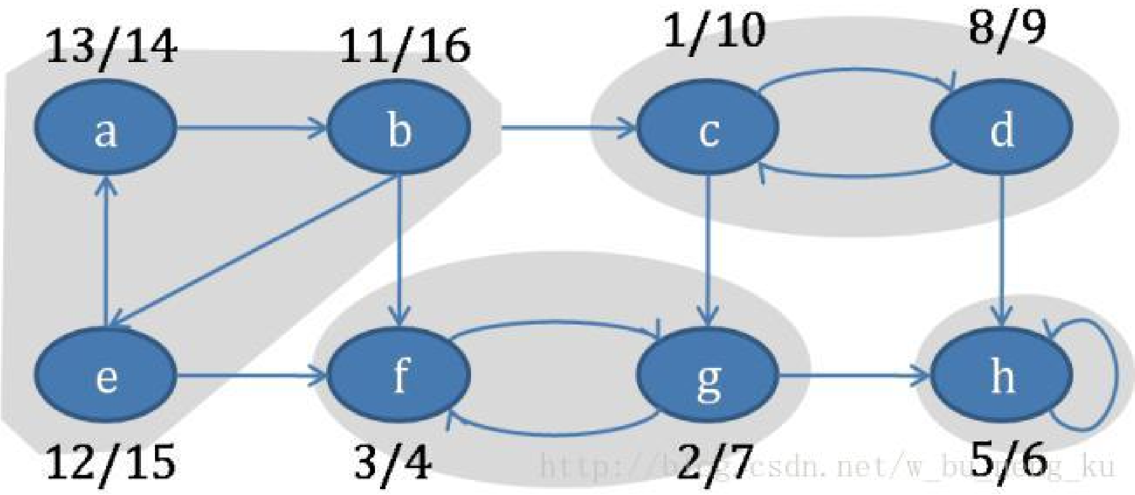
1.5 强连通分量

> [资料：图算法：强连通分量]
> [资料：Kosaraju算法解析: 求解图的强连通分量]

计算强连通分量的算法：两次DFS

1. 在原图G上进行DFS，找出拓扑排序；
2. 在反向图G'上，按照出栈顺序由大到小的顺序执行DFS，找出强连通分量。

题目（亲自练习）：



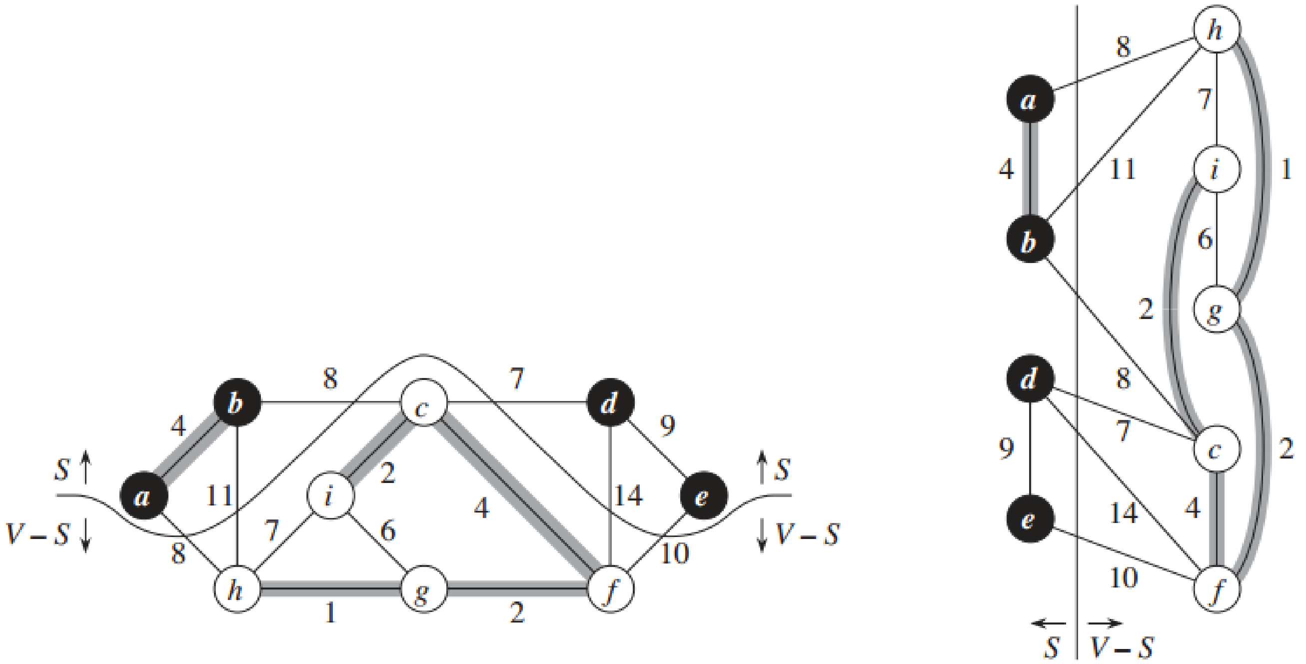
2 最小生成树

2.1 最小生成树的形成

无向图G = （V, S）的一个**切割**（S, V-S）是集合V的一个划分，如下图所示，如果一条边（u， v）∈E的一个端点位于集合S，另一个端点位于结合V—S，则称该条边**横跨**切割（S, V—S）。
如果结合A中不存在横跨该切割的边，则称该切割**尊重**集合A。在横跨一个切割的所有边中，权重最小的边称为**轻量级边**（轻量级边可能不是唯一）。一般的，如果一条边是满足某个性质的所有边中权重最小的，则称该条边是满足给定性质的一条轻量级边。

用例子说话：

下图中，切割**尊重**{a，b}和{c，f，g，h，i}，因为没有把它们割开；这条切割中(c，d)是**轻边**，因为在切割的边中最短。



(a) (b)

Figure 23.2 Two ways of viewing a cut $(S, V - S)$ of the graph from Figure 23.1. **(a)** Black vertices are in the set S, and white vertices are in $V - S$. The edges crossing the cut are those connecting white vertices with black vertices. The edge (d, c) is the unique light edge crossing the cut. A subset A of the edges is shaded; note that the cut $(S, V - S)$ respects A, since no edge of A crosses the cut. **(b)** The same graph with the vertices in the set S on the left and the vertices in the set $V - S$ on the right. An edge crosses the cut if it connects a vertex on the left with a vertex on the right.

安全边

```
if
    A in MST T;
    Cut respect A;
    (u, v) is 轻边;
then
    (u, v) is 安全边;
```

2.2 Kruskal算法和Prim算法

[资料：看图区分两种算法]

Prim算法和Kruskal算法都是从连通图中找出最小生成树的经典算法。从策略上来说，Prim算法是直接查找，多次寻找邻边的权重最小值，而Kruskal是需要先对权重排序后查找的。

所以说，Kruskal在算法效率上是比Prim快的，因为Kruskal只需一次对权重的排序就能找到最小生成树，而Prim算法需要多次对邻边排序才能找到。

Prim算法的实现过程

首先以一个结点作为最小生成树的初始结点，然后以迭代的方式找出最小生成树中各结点权重最小的边，并加到最小生成树中。（加入之后如果产生回路了就要跳过这条边，选择下一个结点。）当所有的结点都加入到最小生成树中后，就找出了这个连通图的最小生成树。

Kruskal算法的实现过程

Kruskal算法在找最小生成树结点之前，需要对权重从小到大进行排序。将排序好的权重边依次加入到最小生成树中，（如果加入时产生回路就跳过这条边，加入下一条边）。当所有的结点都加入到最小生成树中后，就找到了这个连通图的最小生成树。

---

3 单源最短路径

3.1 Bellman-Ford算法

要求：边的权重可以为负值。
方法步骤：固定运行|G.V|-1次Relax。

```
void Bellman-Ford(G, w, s){
    初始化单元最短路径(G, s);
    for i=1 to |G.V|-1
        for each edge(u, v)∈G.E
            Relax(u, v, w);
    for each edge(u, v)∈G.E
        if v.d>u.d+w(u, v)   //存在负圈
            return false;
    return true;
}
```

3.2 有向无环图（DAG图）中单源最短路径问题

方法步骤：拓扑排序后，按照点的顺序依次Relax。

3.3 Dijkstra算法

要求：所有边的权重都为非负值。
方法步骤：依次Relax每个点。

```
//算法运行时间要低于Bellman-Ford算法
void Dijkstra(G, w, s){
    初始化单元最短路径(G, s);
    S = Ø;
    Q = G.v;
    while(Q≠Ø){
        u = Extract_min(Q);
        S = S U {u};
        for each vertex v∈G.Adj[u]
            Relax(u, v, w);
    }
}
```

3.4 差分约束和最短路径

差分约束：就是线性规划问题的单源最短路径图解法。

[资料：差分约束的详细讲解（看书更快）]

题目（亲手练习）：

求下列线性规划的可行解（书上P388）。

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & -1 \\ -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x1 \\ x2 \\ x3 \\ x4 \\ x5 \end{pmatrix} \le \begin{pmatrix} 0 \\ -1 \\ 1 \\ 5 \\ 4 \\ -1 \\ -3 \\ -3 \end{pmatrix}$$

列出差分约束条件：

$$\begin{cases} x1 - x2 \leqslant 0 \\ x1 - x5 \leqslant -1 \\ x2 - x5 \leqslant 1 \\ x3 - x1 \leqslant 5 \\ x4 - x1 \leqslant 4 \\ x4 - x3 \leqslant -1 \\ x5 - x3 \leqslant -3 \\ x5 - x4 \leqslant -3 \end{cases}$$

画出约束图，求出单源最短路径，解得

$$x = (-5, -3, 0, -1, -4)$$

根据引理24.8，可以得到另一个解：

$$y = (0, 2, 5, 4, 1)$$

3.5 最短路径的性质证明（三上无路收钱）

1. **三角不等式性质：**
   δ(s, v) ⩽ δ(s, u) + w(u, v)
2. **上界性质：**
   d(v) ⩾ δ(s, v)，且一旦达到下界δ，就不变了。
3. **无路径性质：**
   如果v是s不可达的，则d(v) = δ(s, v) = ∞。
4. **收敛性质：**
   松弛后相对松弛前，d值更加收敛于最短路径δ。
5. **路径松弛性质：**
   任意一条最短路径一路上每个点挨个松弛过后，终点的d值就变成δ了，且之后一直不变。
6. **前驱子图性质：**
   一旦对于所有点v，有d(v) = δ(s, v)，则前驱图就是最短路径树。

---

4 所有结点对的最短路径问题

> [资料：三种算法讲解]()

4.1 矩阵乘法

matrix multiplication

此处矩阵乘法是新定义乘法，乘法A*B定义为A的i行与B的j列各位相加的最小值，就是结果的i行j列。

> L(1)=L(0)*W=W
> L(2)=L(1)*W=W^2
> ...
> L(n-1)=L(n-2)*W=W^(n-1)

题目（亲手练习）：

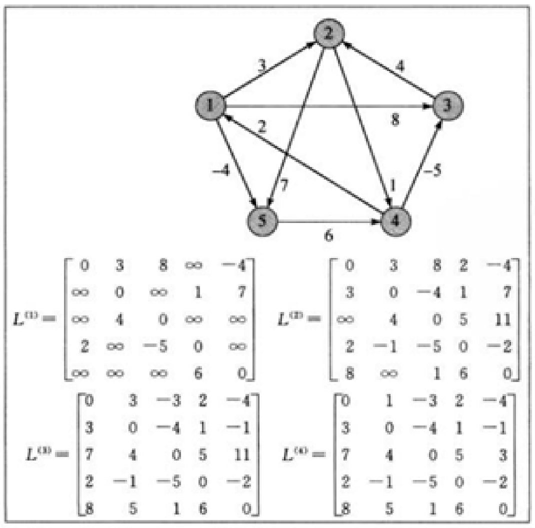


图 25-1　一个有向图和由 SLOW-ALL-PAIRS-SHORTEST-PATHS 所计算出的矩阵序列 $L^{(m)}$。读者可以自行验证 $L^{(5)}=L^{(4)}$，因此，对于所有的 $m\geq 4$，有 $L^{(m)}=L^{(4)}$

时间复杂度为$O(n^4)$。

improved matrix mult.

改进算法的运行时间：

> L(1) = W
> L(2) = W * W
> L(4) = W^2 * W^2
> L(8) = W^4 * W^4
> …

时间复杂度为$O(n^3 \log n)$。

4.2 Floyd-Warshall算法

> **要求：** 可以有负边，但不可以有负环。
> **算法思路：** 动态规划。

弗洛伊德刮痧公式：

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k = 0\,, \\ \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & \text{if } k \geq 1\,. \end{cases}$$

题目（亲手练习）：

$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(0)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & \text{NIL} & 4 & \text{NIL} & \text{NIL} \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(1)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(2)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(3)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \quad \Pi^{(4)} = \begin{pmatrix} \text{NIL} & 1 & 4 & 2 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \quad \Pi^{(5)} = \begin{pmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

传递闭包

4.3 用于稀疏图的Johnson算法

算法中运用Dijkstra、Bellman-Ford算法，使用的技术是重新赋予权重。

解法：

- 如果图G = (V, E)中的边全为非负值，则通过对所有结点运行一次Dijkstra算法找出所有结点对的最短路径。
- 如果有负值，但没有负环，则运行重新赋予权重方法，然后再用相同的方法即可。

重新赋予权重技术求解：

- 新增一个结点s，使w(s, v)=0
- 利用w'(u, v) = w(u, v) + h(u) - h(v)重新生成非负权重
- 去掉s点，在新的不含负边的图中运行Bellman-Ford或Dijkstra算法求出每个点的单源最短路径

题目：

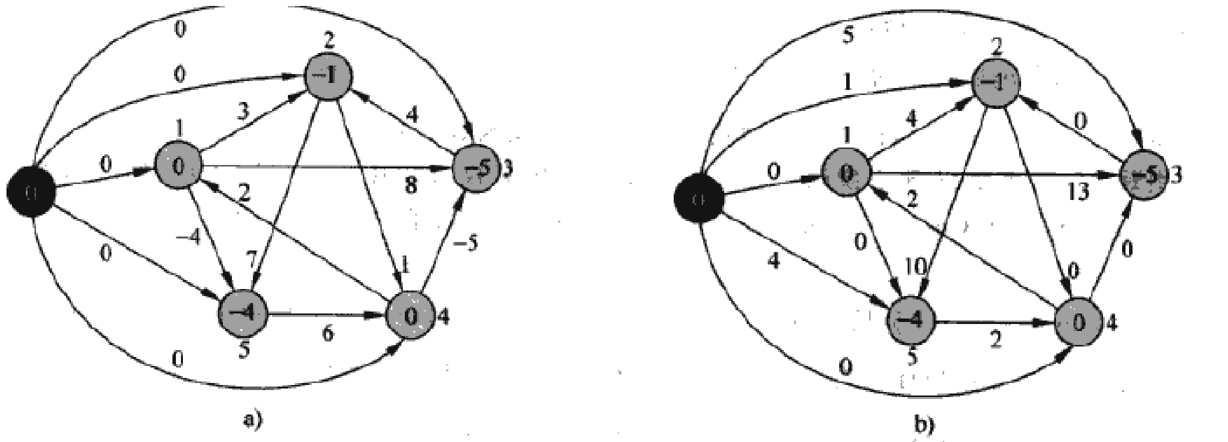


图 25-6 Johnson 每对顶点最短路径算法在图 25-1 所示的图上的运行过程。a)用原始加权函数 w 的图G'。新顶点 s 是黑色的。在每个顶点 v 内的是 $h(v)=\delta(s, v)$。b)每条边(u, v)用加权函数$\hat{w}(u, v)=w(u, v)+h(u)-h(v)$重赋权。c)～g)用加权函数$\hat{w}$在图 G 的每个顶点上执行 Dijkstra 算法的结果。在每个子图中，源顶点 u 是黑色的，而阴影边是在算法计算出的最短路径树内。每个顶点 v 内包含值$\hat{\delta}(u, v)$和$\delta(u, v)$，用一个斜线来分隔。$d_{uv}=\delta(u, v)$的值等于$\hat{\delta}(u, v)+h(v)-h(u)$

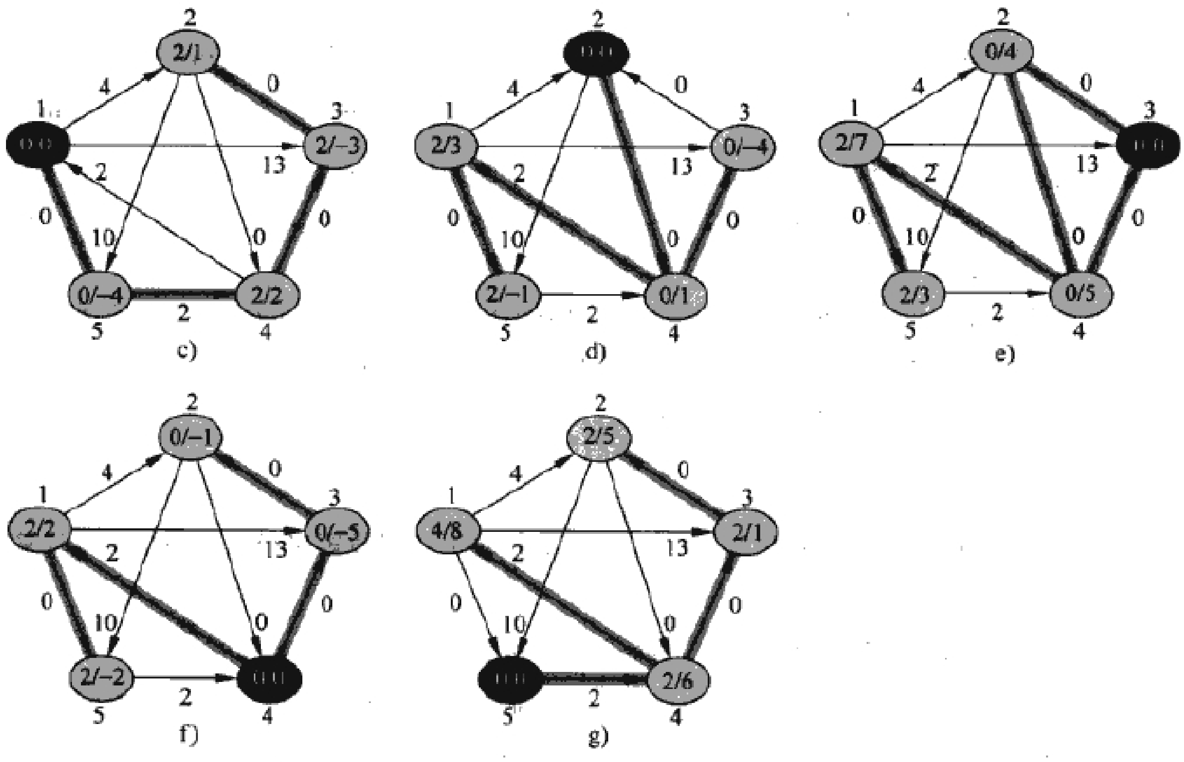


图 25-6 （续）

---

5 最大流

讲解资料：

[图的匹配问题与最大流问题(一)]

图的匹配问题与最大流问题(二)——最大流问题Ford-Fulkerson方法
图的匹配问题与最大流问题(五)——计算二分图的最大匹配

5.1 流网络

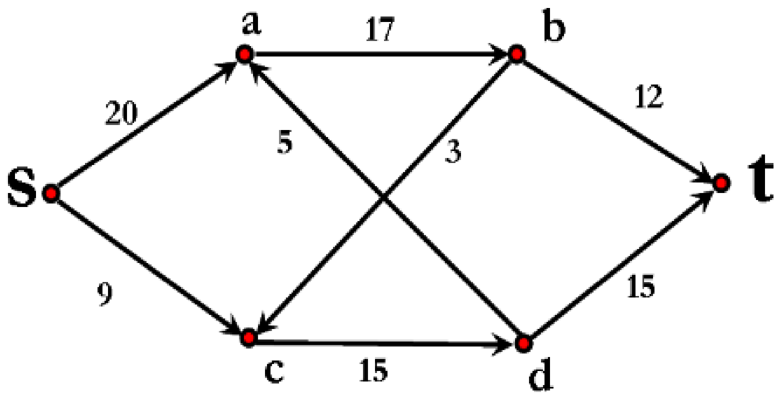
5.2 Ford-Fulkerson方法

方法步骤:

- 计算剩余网络
- 寻找增广路径
- 将增广路径的流量加到原图中
- 往复循环，直至没有增广路径
- 最终得到的就是最大流网络

题目:

题 5、求下图最大流。

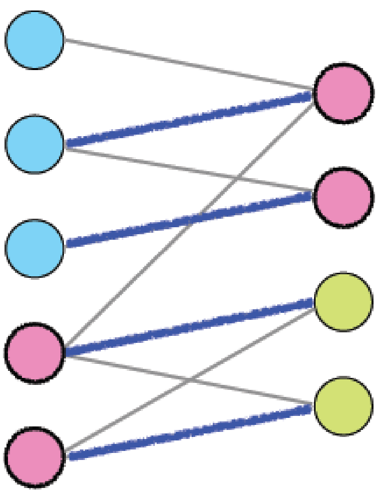


我求得的结果是15+9=24。

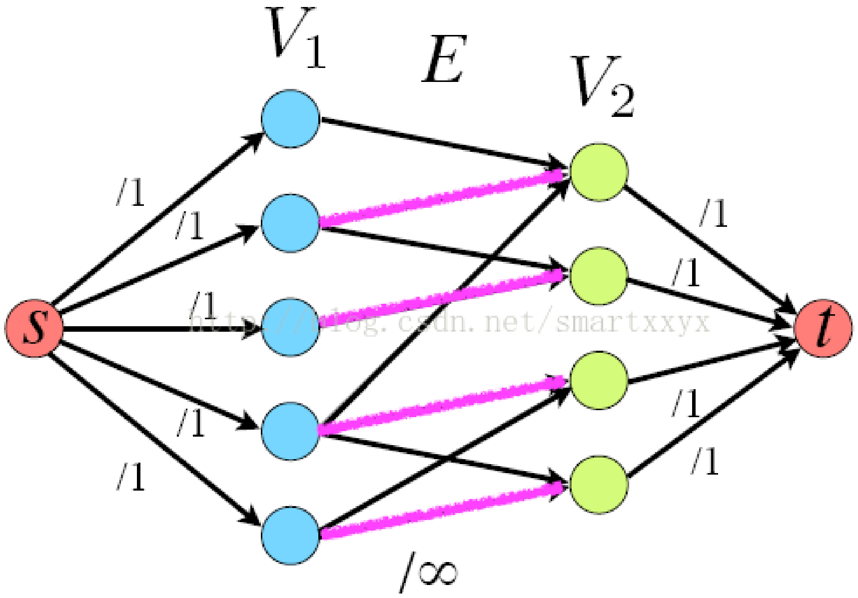
5.3 最大二分匹配

使用Ford-Fulkerson方法寻找最大二分匹配。

给定如下的二分图（忽略颜色）：



把已有的边设为单向边（方向 L -> R），且各边容量设为 ∞； 增加源结点 s 与汇点 t，将 s 与集合 L 中各个结点之间构造单向边，且各边容量设为 1；同样的，将集合 R 中各个结点与 t 之间构造单向边，且各边容量设为1。这时得到一个流网络 G'，如下:



这时，最大匹配数值就等于流网络 G' 中最大流的值。

（转自简书：Ford-Fulkerson 方法——最大流问题）

习题

- 图论练习题
- 算法导论图论课后习题答案

附录

Table of running times

algorithm	running time
Dijkstra's	$O(n^2 \log n + nm)$
Bellman-Ford	$O(n^2 m)$
matrix multiplication	$O(n^4)$
improved matrix mult.	$O(n^3 \log n)$
Floyd-Warshall	$O(n^3)$
Johnson's	$O(n^2 \log n + nm)$
Transitive closure	$O(n^3)$

- **内容目录**

- 私有
- 公开
- 删除

查看更早的 5 条回复

回复批注

×

通知

取消 确认