通信网理论基础

第三章　通信网的结构

第二节　最短径问题

北京邮电大学　　信息与通信工程学院
授课教师：武穆清
电子信箱：wumuqing@bupt.edu.cn

课程内容介绍

第一章　引论

通信系统和通信网的种类和基本要求

第二章　通信网的组成要素

通信系统和网络的构成部件、功能、特性

第三章　通信网的结构

图论基础，最短径、最大流、最佳流算法

第四章　网内业务分析

排队论基础，业务模型与分析，网络效率

第五章　通信网的可靠性

可靠性理论，系统可靠性，网络可靠性

第三章 通信网的结构

3.1 图论基础

3.2 最短径问题

3.3 站址问题

3.4 流量分配

第二节 最短径问题

3.2 最短径问题

- 从本节开始讨论有权图
  - 这是解决实际问题中常用的
- 在实际问题中，边的权值可以表示各种物理量
  - 如费用，几何距离，时延，衰减，噪声，容量等
  - 有时也用**距离**泛指各种权值
  - **最短**表示这些权值之和最小
- 有些应用中需要研究转接的次数
  - 即：定义径长为径内的边数，或转接的次数、跳数
  - 对于有权图来说，这种径长的定义隐含着每边都赋予权值为1
  - 即：如果**把各边的权值设为1**，则此时的径长就是跳数

- 最短径问题的求解可用于:
  - △ 通信路由选择
  - △ 通信网络结构优化
- ≡ 局站和信道代价已知的情况下
  - △ 如何以最小费用规划联结的网
- ≡ 通信网络结构已知的情况下
  - △ 如何选择局站来建立最佳路由和备用路由
    - 以控制转接次数等指标
  - △ 如何选择站址和埋设管道的路由
- ≡ 以上大多问题是优化算法问题
  - △ 有些算法可得到最优解
  - △ 有些则只能得到准最优解
  - △ 大部分算法的计算量都是按节点数**n**的多项式增长的
  - △ 需借助计算机来实现
  - △ 对网络结构优化有重要应用

3.2.1 最短主树

- 联结图G若其本身不是一棵树，则主树不止一个
- 但满足一定条件的最短主树至少存在一个
- 寻找最短主树是一个常见的优化问题

– 定义：
  - 联结图G有n个端，端间距离为d_{ij}
    - （i，j＝1，2，…，n）
    - 若v_i与v_j之间无边，则令$d_{ij}=\infty$
  - 最短主树：n－1条边（树枝）的长度之和（即权之和）最小的联结子图

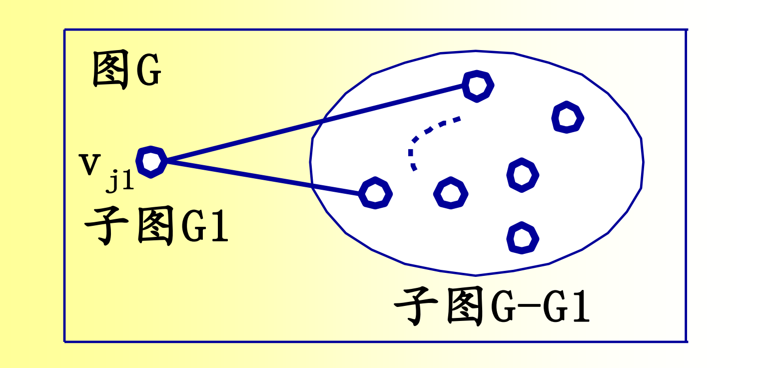
– 无限制条件的情况

= Prim 算法（普瑞姆算法，简称P算法）

≡ 特点：顺序取端，边取最小

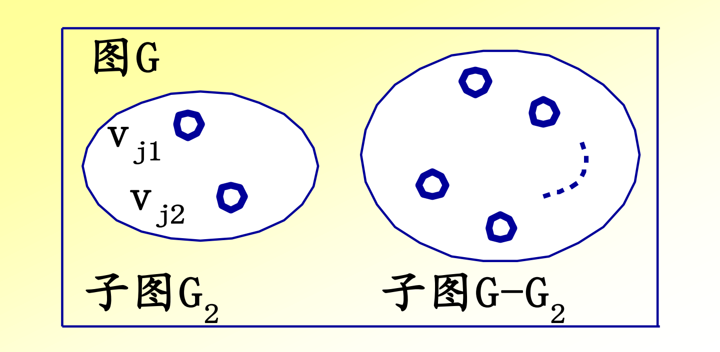
≡ 步骤：

Δ 第0步：置邻接阵为全0阵



Δ 第一步：

- 任取一端v_{j1}，作子图G_1 = {v_{j1}}
- 从G_1到$G - G_1$，取最短边，及所关联的端v_{j2}
- 数学表示为： $$\min_{j \in G-G_1} d_{j_1 j} = d_{j_1 j_2}$$
- 作子图G_2 = {v_{j1}, v_{j2}}
- 置邻接阵元素$c_{j1\,j2} = c_{j2\,j1} = 1$

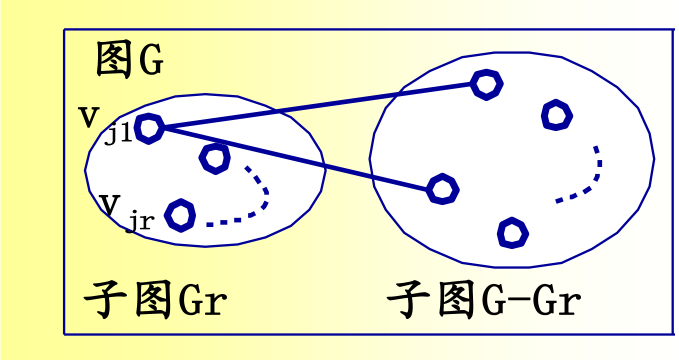


Δ 第二步：

- 设Gr是已得到的含有r个端的子图
- 从子图Gr到G－Gr，取最短边，及所关联的端
- 数学表示为：$\min\limits_{\substack{i\in G_r \\ j\in G-G_r}} d_{ij}=d_{sj_{r+1}}$ 其$v_s\in G_r$，$v_{j_{r+1}}\in G-G_r$
- 作子图 $G_{r+1}=G_r\cup\left\{v_{j_{r+1}}\right\}=\left\{v_{j_1},v_{j_2},\ldots\ldots v_{j_r},v_{j_{r+1}}\right\}$
- 置邻接阵元素 $c_{sj_{r+1}}=c_{j_{r+1}s}=1$

Δ 第三步：

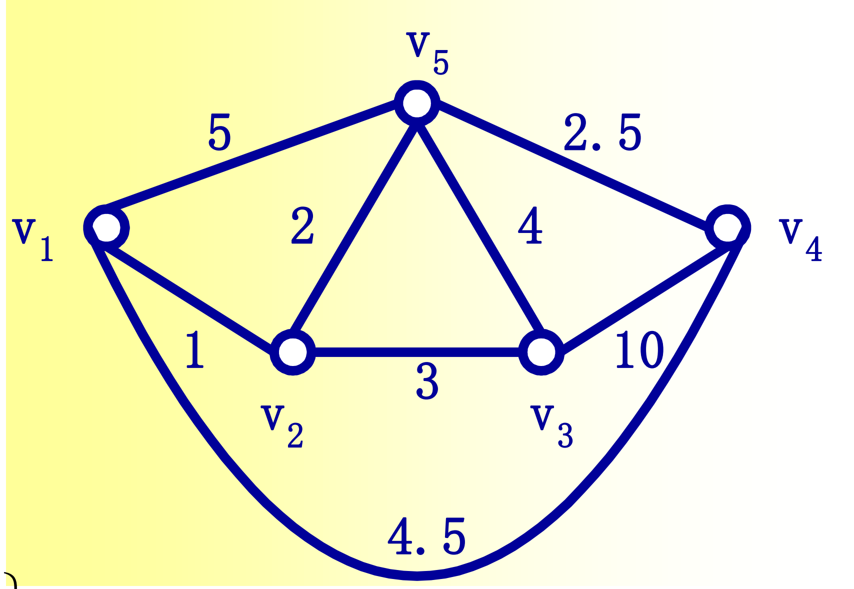
- 若r＋1＜n，则重复第二步
- 若r＋1＝n，则终止
- 此时已得到了最短主树Gn，及其邻接阵C



Prim 算法举例

- 如图：
- 取G_1 = $\{v_1\}$
  - $G - G_1$ = $\{v_2, v_3, v_4, v_5\}$
  - $\min\limits_{j \in G - G_1} \{d_{ij}\} = \min\{d_{12} = 1, d_{13} = \infty, d_{14} = 4.5, d_{15} = 5\} = d_{12}$
  - 关联端为v_2
  - 邻接阵元素$c_{12} = c_{21} = 1$

$$C = \begin{array}{c c} & \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} & \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$



- 作子图G_2 = $\{v_1, v_2\}$
  - $G - G_2$ = $\{v_3, v_4, v_5\}$

$$\min_{\substack{i \in G_2 \\ j \in G - G_2}} \{d_{ij}\} = \min \left\{ \begin{matrix} d_{13} = \infty, & d_{14} = 4.5, & d_{15} = 5 \\ d_{23} = 3, & d_{24} = \infty, & d_{25} = 2 \end{matrix} \right\} = d_{25}$$

- 关联端为v_5
- 邻接阵元素$c_{25} = c_{52} = 1$

$$C = \begin{array}{c c} & \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} & \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

$$C = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

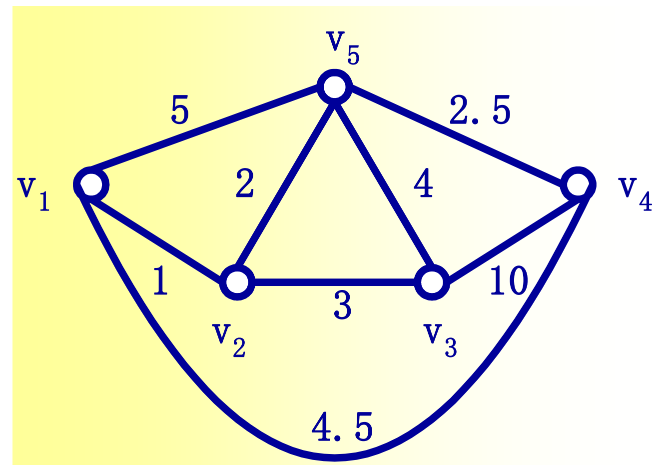
≡ 作子图 G_3 = $\{v_1, v_2, v_5\}$

Δ $G - G_3$ = $\{v_3, v_4\}$

Δ $$\min_{\substack{i \in G_3 \\ j \in G - G_3}} \{d_{ij}\} = \min \begin{Bmatrix} d_{13} = \infty & , & d_{14} = 4.5 \\ d_{23} = 3 & , & d_{24} = \infty \\ d_{53} = 4 & , & d_{54} = 2.5 \end{Bmatrix} = d_{54}$$

Δ 关联端为 v_4

Δ 邻接阵元素 $c_{54} = c_{45} = 1$



≡ 作子图 G_4 = $\{v_1, v_2, v_5, v_4\}$

Δ $G - G_4$ = $\{v_3\}$

Δ $$\min_{\substack{i \in G_4 \\ j \in G - G_4}} \{d_{ij}\} = \min \{d_{13} = \infty \;,\; d_{23} = 3 \;,\; d_{43} = 10 \;,\; d_{53} = 4\} = d_{23}$$

$$C = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Δ 关联端为 v_3

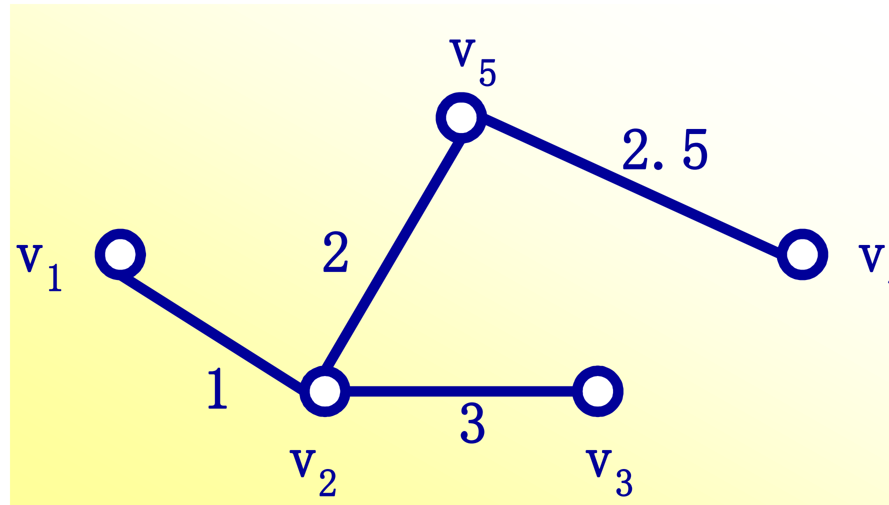
Δ 邻接阵元素 $c_{23} = c_{32} = 1$

≡作子图G_5 = {v_1, v_2, v_5, v_4, v_3}

Δ 至此，已找出了最短主树，如图：

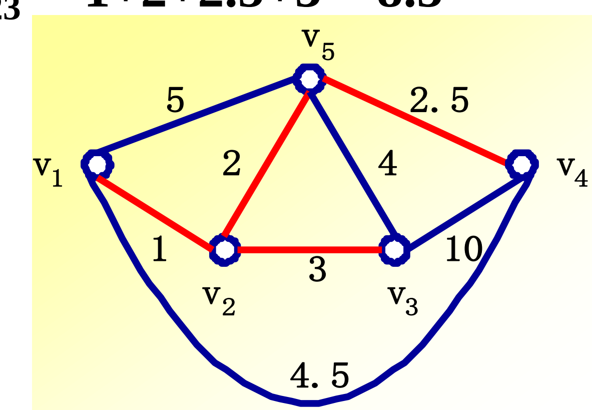
Δ 其邻接阵为：

$$C = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$



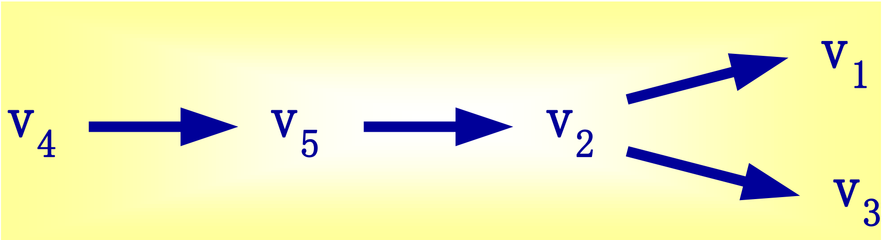
Δ 树枝总长为：：$l = d_{12} + d_{25} + d_{54} + d_{23}$ = 1+2+2.5+3 = 8.5

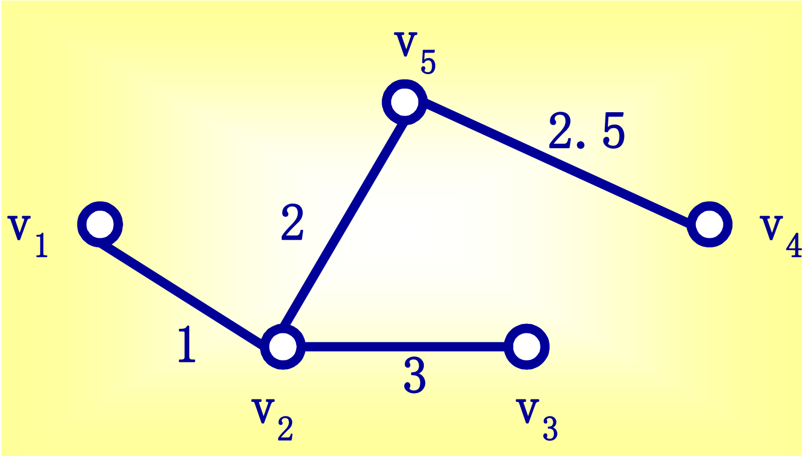
Δ 这是一个最短主树，设它为P树



≡ 我们再来找一棵Q树，仍用此法，但从V_4开始，反过来找

Δ 结果为：





Δ 可见，它与P树完全一致

Δ 所以，P树为最短主树

≡ 当所有的边都不等长时（即权值均不相等时），最短主树P和Q是重合的

≡ 当某些边等长（等权）时，Q树与P树可能不同，但树枝的总长度是一样的

Prim 算法的复杂性

- 从算法开始到算法结束，共进行n－1步
- 每一步都要把子图G_r中的r个端与$G－G_r$中的（n－r）个端间的距离作比较运算，从中找出最小距离
- 所以，第r步中要作 r(n－r)－1次比较运算
- P算法的计算量为:

$$\sum_{r=1}^{n-1}\left[r(n-r)-1\right] = n\sum_{r=1}^{n-1} r-\sum_{r=1}^{n-1} r^2-(n-1)$$

$$= n\cdot\frac{n(n-1)}{2}-\frac{1}{6}\cdot n(n-1)(2n-1)-(n-1)$$

$$= \frac{1}{6}\cdot(n-1)\left[3n^2-2n^2+n-6\right] = \frac{1}{6}\cdot(n-1)(n-2)(n+3)$$

- 可见，P算法的计算量是n^3量级

- 其实，P算法中的比较是有重复的
  - 如果每次比较的结果都记录下来，则比较次数可降至n^2量级，但算法会复杂一些，要占用更多的存储空间

- 总之，P算法的复杂性属于多项式型，n^2，n^3
  - 是非NP问题，是可实现的
  - 当n较大时，可用计算机来计算
  - 若计算量是n^n量级，则属于非多项式型的复杂问题
    - 即NP问题，无法实现
    - （NP：Non－Polynomial）

=P算法所得的树必为最短主树

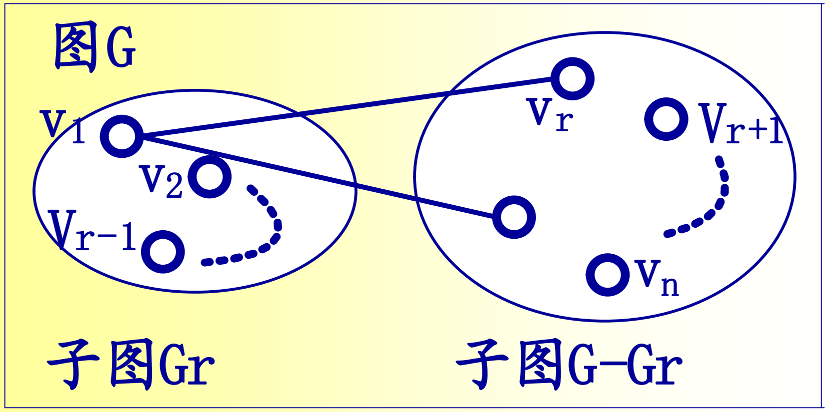
【证明】：

Δ设用P算法所得主树P中，

- 每次所增加的端顺序为：$v_1, v_2, \ldots, v_n$
- 这并不失一般性，因为端的编号本来就是任意的
- 各次取边的长度分别为：$d_{i_r r}$ （$r = 2, 3, \ldots, n$, $i_2=1$）

Δ若有另一棵主树Q是最短主树

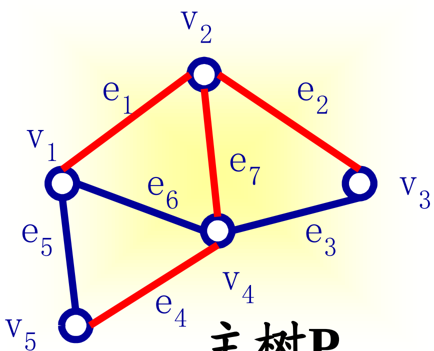
- 在所有端间距离d_{ij}都不相等的情况下，可证明Q与P是重合的



Δ 若有另一棵主树Q是最短主树

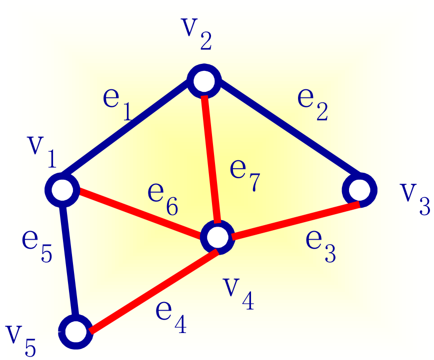
Δ 在所有端间距离d_{ij}都不相等的情况下，可证明Q与P是重合的

Δ 令$V_{r-1}=\{ v_1, v_2, \ldots, v_{r-1} \}$，则$V- V_{r-1}=\{ v_r, v_{r+1}, \ldots, v_n \}$



主树P

- 下面来证明P中边长为$d_{i_r r}$的边一定也是Q中的某一边
- 因为在Q中，v_{i_r} 和 v_r 之间必有一条径，
- 若这条径不是边长为$d_{i_r r}$的边，
- 则这条径将与此长度为$d_{i_r r}$的边形成环
- 环内至少另有一条边穿过V_{r-1}与$V- V_{r-1}$间的界面
- 而这条边的长度必大于 $d_{i_r r}$（这是P算法所规定的）
- 可见，若用长为$d_{i_r r}$的边替代上述边，则仍是一主树，而且比Q树更短
- 但是，由于已经假定Q树为最短，
- 所以，长为 $d_{i_r r}$ 的边一定在Q树内
- 这个结论对$r=2, 3, \ldots, n$均成立，所以，Q与P是重合的



主树Q

Δ 若各端间距离d_{ij}有相等的，则虽然P树与Q树可以不同，但树枝的总长度将是一样的

Δ （证毕）

= **Kruskal算法（克鲁斯格尔算法，简称：K算法）**

≡ 特点：顺序取边

≡ 步骤：

Δ 第O步：置邻接阵为全0阵

- 把所有边按长度排序

Δ 第一步：

- 取最短边$d_{i1\ j1}$，及相关联的端v_{i1}，v_{j1}
- 作子图$G1 = \{v_{i1}, v_{j1}\}$
- 置邻接阵元素$c_{i1\ j1} = c_{j1\ i1} = 1$
- 在边的长度排序队列中，删去该边

Δ 第二步:

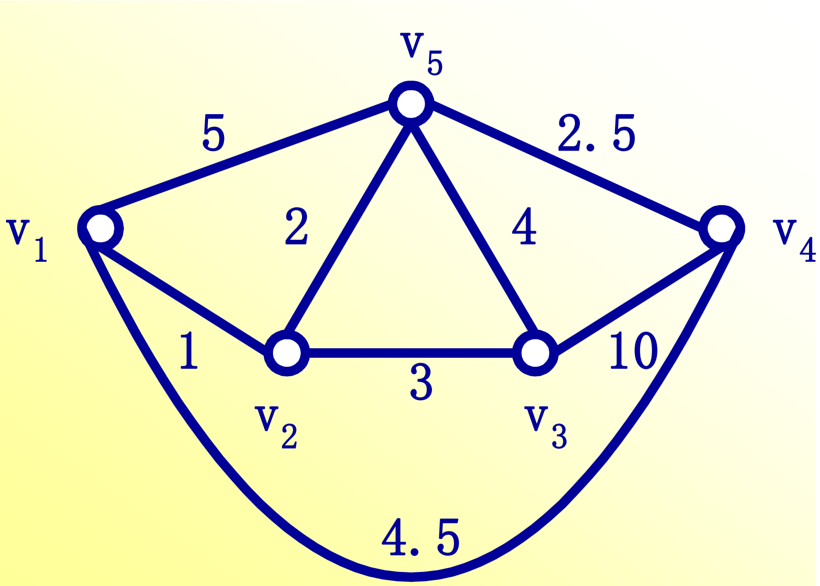
- 设Gr是已经得到的有r条边和至少r+1（至多2r）个端的子图
- 在所剩的边的队列中，取最短边
- 检查Gr在加上这条边后，是否有环
- 若有环，则从边的队列中删去这条边，并继续取最短边，直至找出不形成环的最短边
- 若无环，设所取的边长为$d_{i(r+1)\, j(r+1)}$，相关联的端为$v_{i\,(r+1)}$，$v_{j\,(r+1)}$
- 作子图$G_{r+1} = G_r \cup \{ v_{i\,(r+1)}, v_{j\,(r+1)} \}$
- 置邻接阵元素$c_{i\,(r+1)\ \ j\,(r+1)} = c_{j\,(r+1)\ \ i\,(r+1)} = 1$
- 在边的长度排序队列中，删去该边

Δ 第三步:

- 若$r < n - 1$，则重复第二步
- 若$r = n - 1$，则终止
- 此时已得到最短主树，及其邻接阵C

≡ K算法举例

≡ 如前例:



≡ 置邻接阵为全0 阵

Δ 把所有边按长度排序:

v_1v_2	v_2v_5	v_4v_5	v_2v_3	v_3v_5	v_1v_4	v_1v_5	v_3v_4
1	2	2.5	3	4	4.5	5	10

≡ 取最短边: $d_{12} = 1$，所关联的端为v_1, v_2

Δ 作子图G_1 = $\{v_1, v_2\}$

Δ 置邻接阵元素$c_{12} = c_{21} = 1$

Δ 把该边从边队列中删除

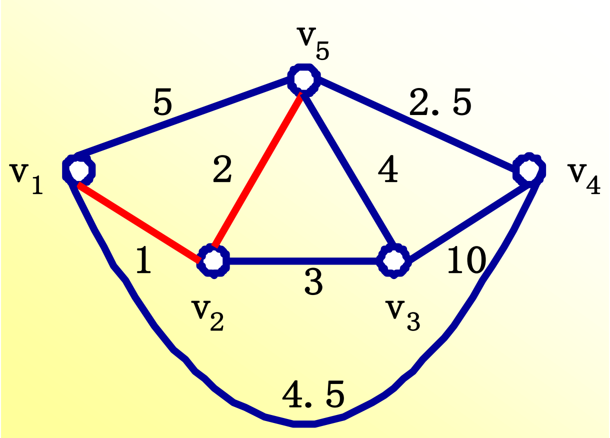
Δ 剩余边的队列为: 2，2.5，3， 4，4.5，5， 10

$$C = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

≡ 从剩余边的队列中，取最短边，$d_{25} = 2$

$$C = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

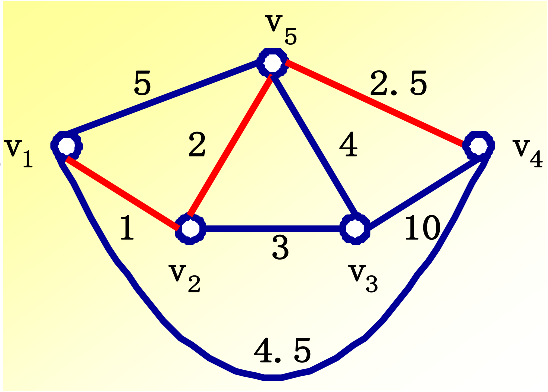
- Δ 所关联的端为v_2，v_5
- Δ 子图G_1加上边$v_2 - v_5$后，不形成环
- Δ 作子图$G_2 = G_1 \cup \{v_2, v_5\} = \{v_1, v_2, v_5\}$
- Δ 置邻接阵元素$c_{25} = c_{52} = 1$
- Δ 把该边从边队列中删除
- Δ 剩余边的队列为：2.5, 3, 4, 4.5, 5, 10



≡ 从剩余边的队列中，取最短边，$d_{45} = 2.5$

$$C = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

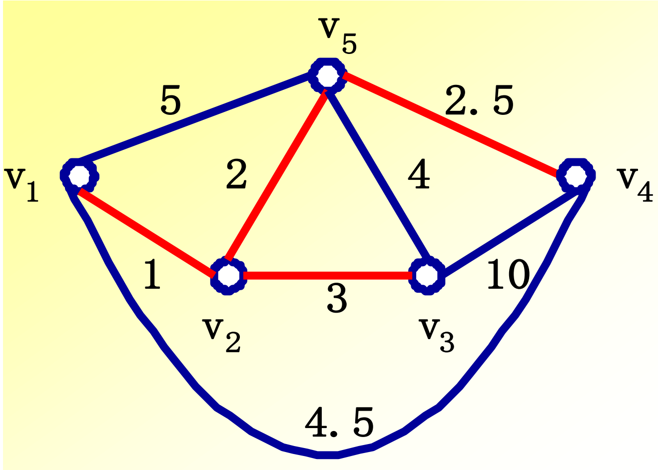
- Δ 所关联的端为v_4, v_5
- Δ 子图G_2加上边$v_4 - v_5$后，不形成环
- Δ 作子图$G_3 = G_2 \cup \{v_4, v_5\} = \{v_1, v_2, v_5, v_4\}$
- Δ 置邻接阵元素$c_{45} = c_{54} = 1$
- Δ 把该边从边队列中删除
- Δ 剩余边的队列为：3，4，4.5，5，10



≡从剩余边的队列中，取最短边，$d_{23}=3$

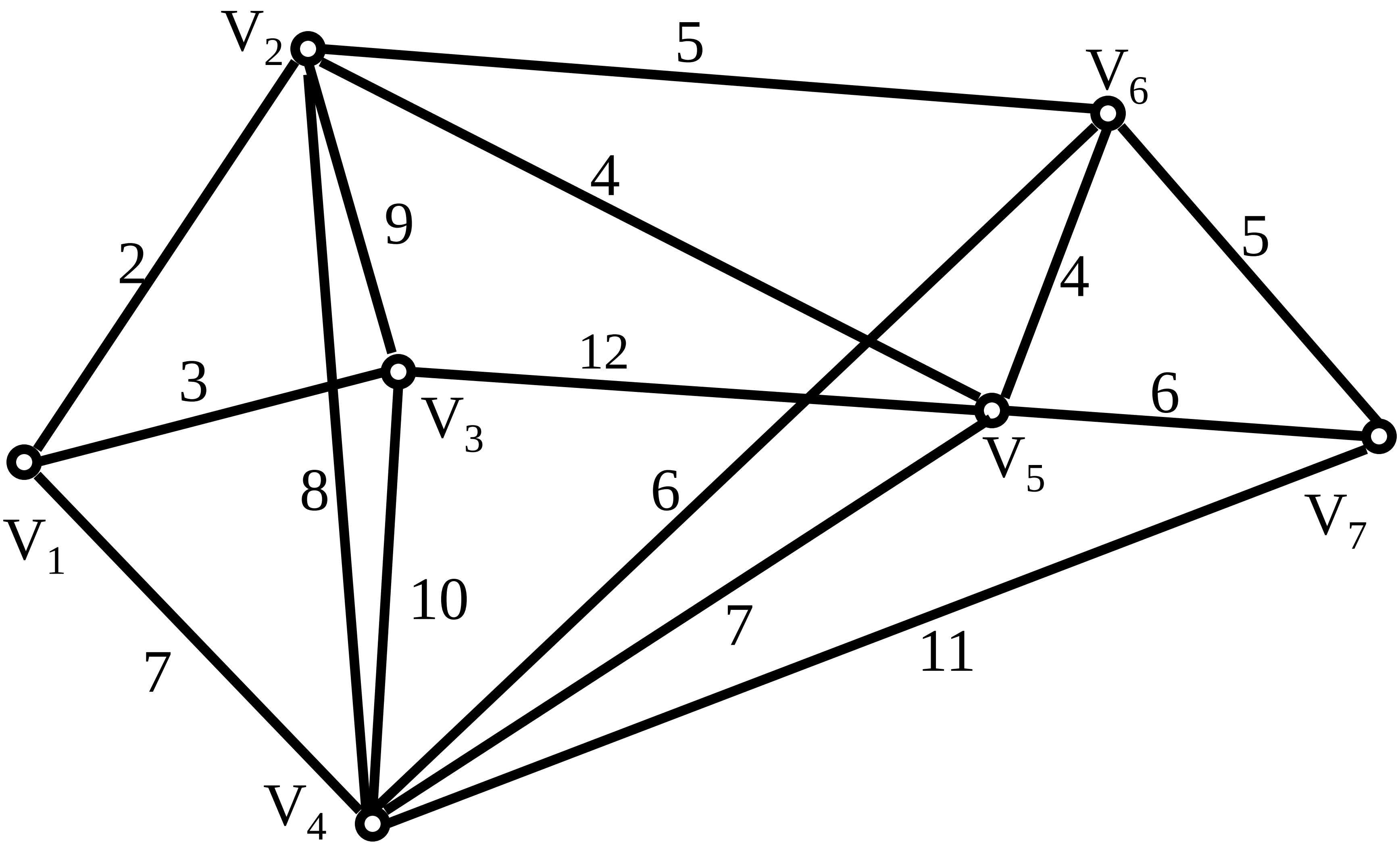
- Δ 所关联的端为v_2，v_3
- Δ 子图G_3加上边v_2-v_3后，不形成环
- Δ 作子图$G_4=G_3\cup\{v_2, v_3\}=\{v_1, v_2, v_5, v_4, v_3\}$
- Δ 置邻接阵元素$c_{23}=c_{32}=1$
- Δ 至此，子图G_4已覆盖了全部端点，$r=4$，$n-1=4$，终止
- Δ 其邻接阵为：（完全同前例）

$$
C=\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array}
\begin{array}{c}
\begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\
\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}
\end{array}
$$

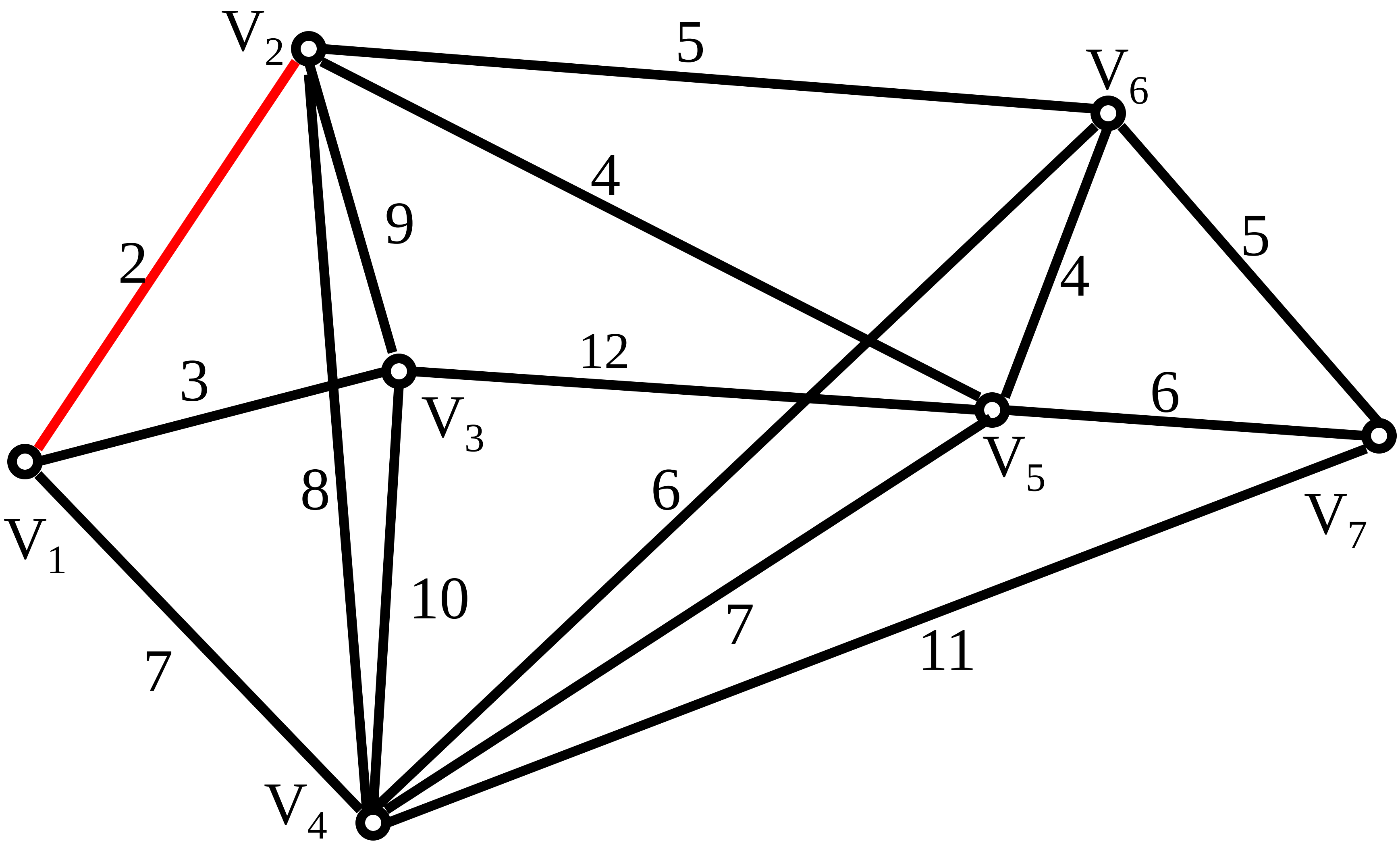


- **K算法所得的树必为最短主树**
  - Δ 即不存在另一棵主树，其边长之和小于K算法得到的边长之和
    - 最多是相等
- 【证明】：
  - Δ 设用K算法所得主树为K
  - Δ 任取不在K内的边，其长度为d_{ij}
  - Δ 把这边加到K上，必形成环
  - Δ 在这环上任去掉一条边，设其长度为d_{rs}, 则将形成另一棵主树
  - Δ 下面分两种情况来讨论:
    - 若$d_{rs} \le d_{ij}$, 这样的置换并不减小树枝的总长度;
    - 若$d_{rs} > d_{ij}$, 则说明在边的序列中，d_{ij}排在d_{rs}之前，则在选用d_{rs}时必已取d_{ij}而不会取d_{rs}。这与算法的前提相矛盾。
  - Δ 因此，这两种情况下的置换都不能减小总长度
  - Δ 所以，K算法得到的主树是最短的
  - Δ （证毕）

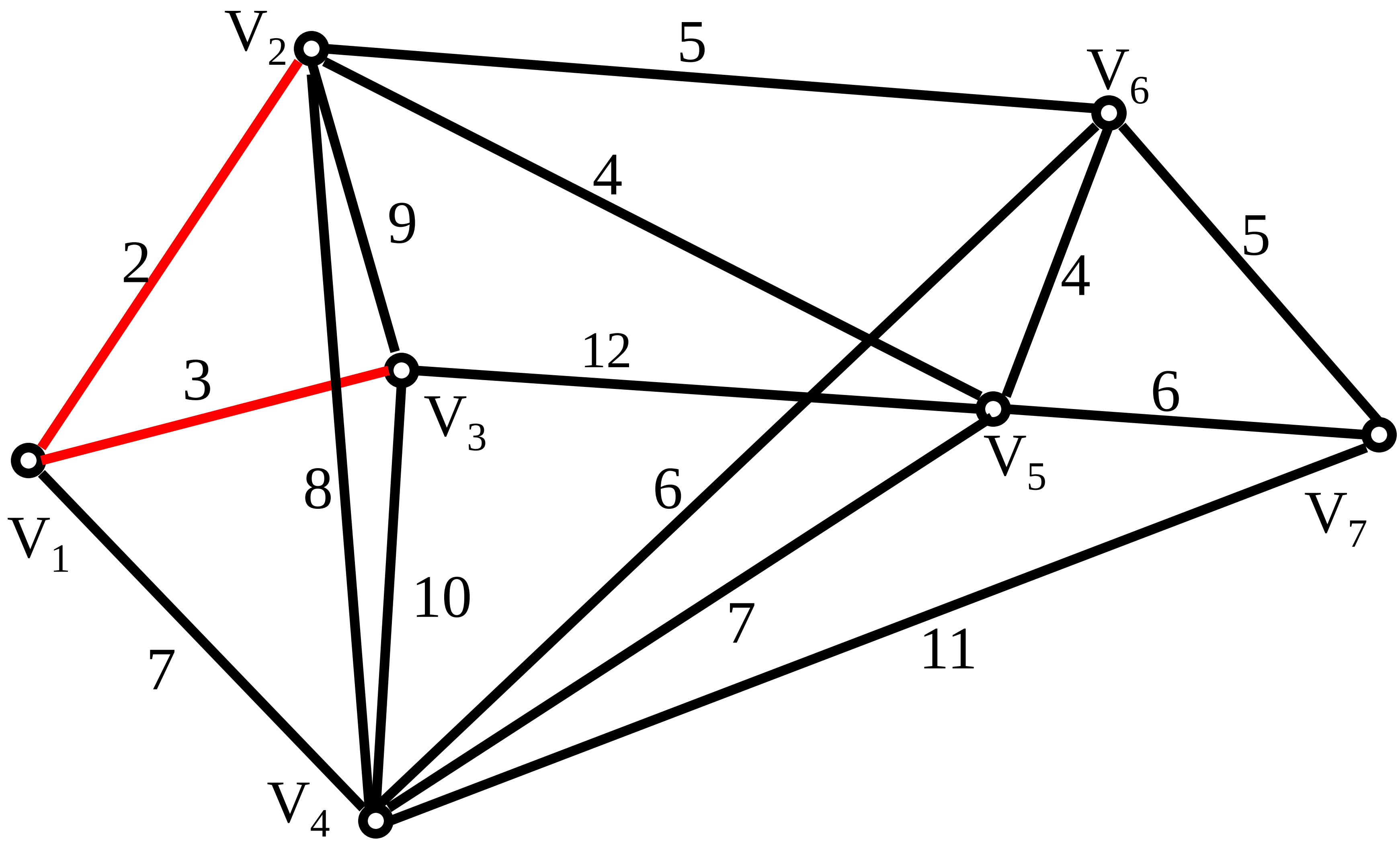
＝K算法举例（2） (第1页)



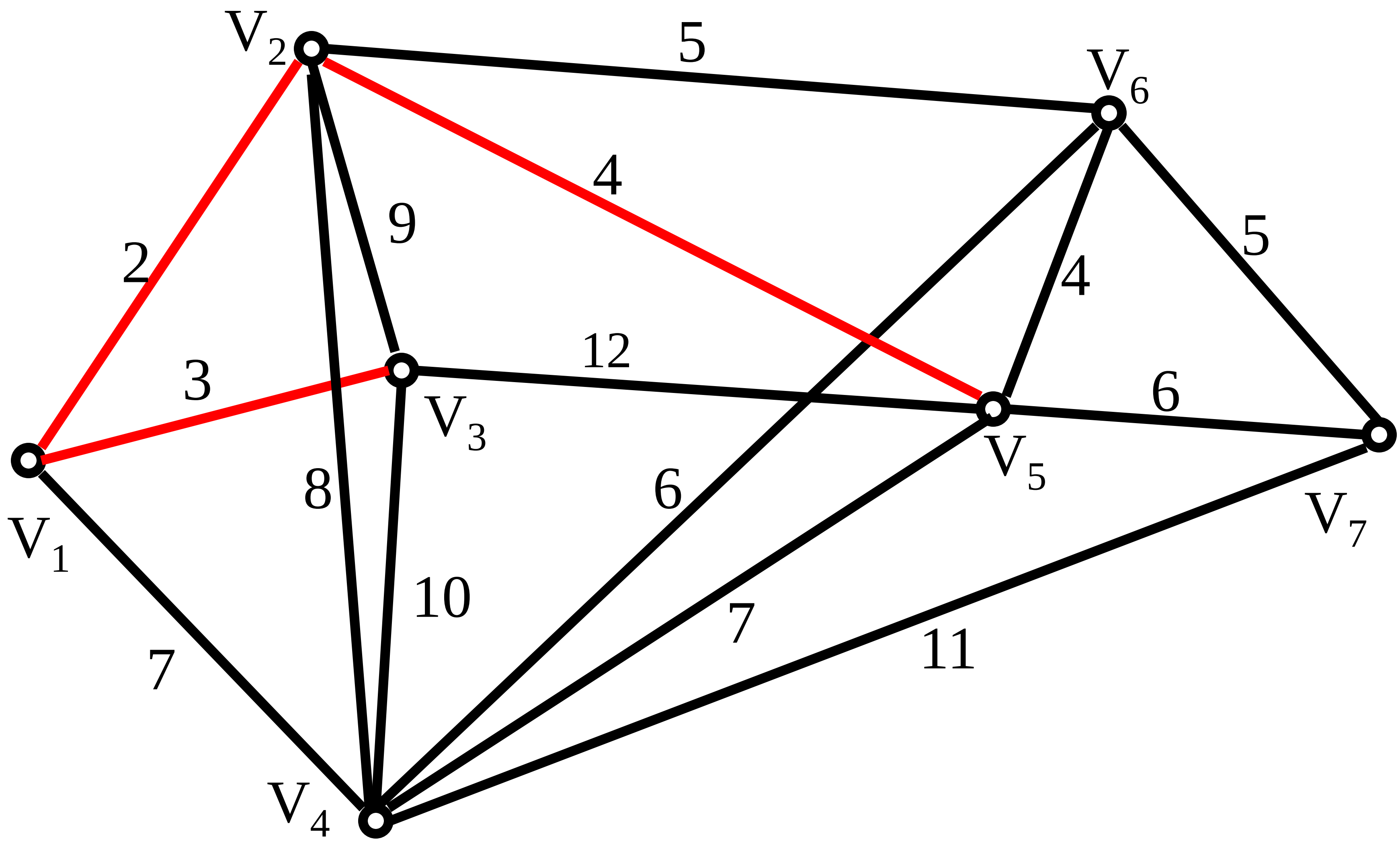
＝K算法举例（2）　　　　(第2页)



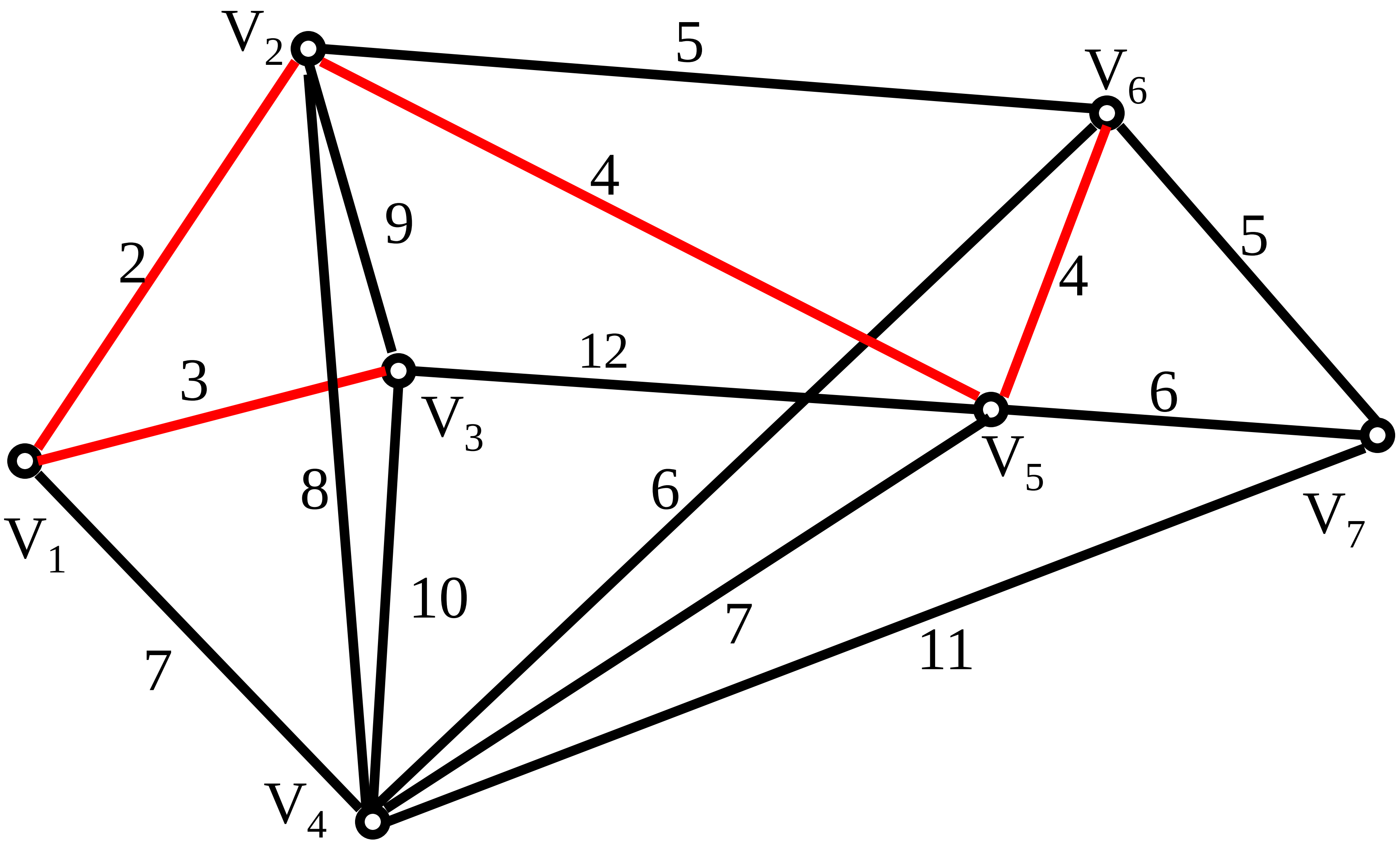
＝K算法举例（2）　　(第3页)



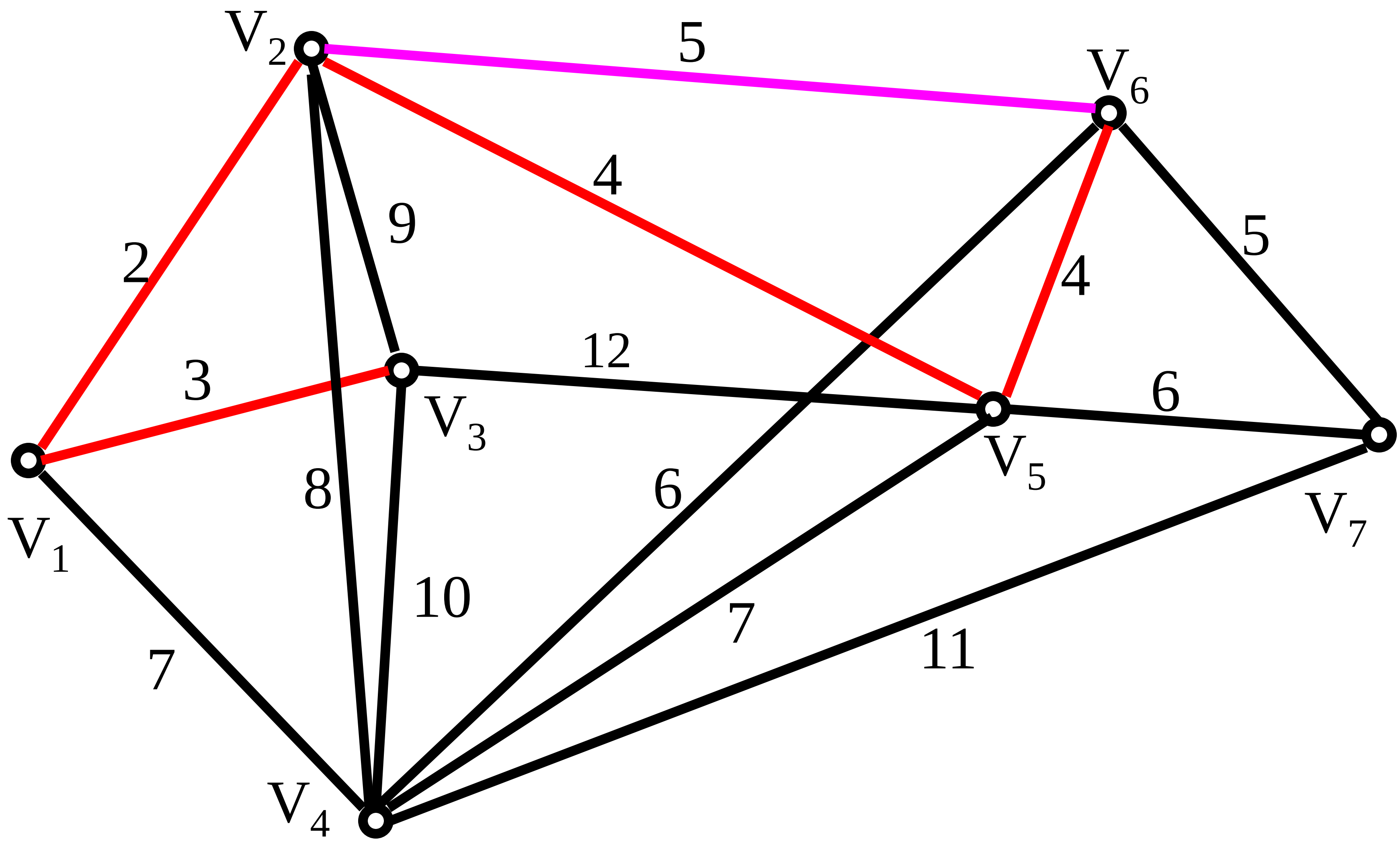
＝K算法举例（2）　　(第4页)



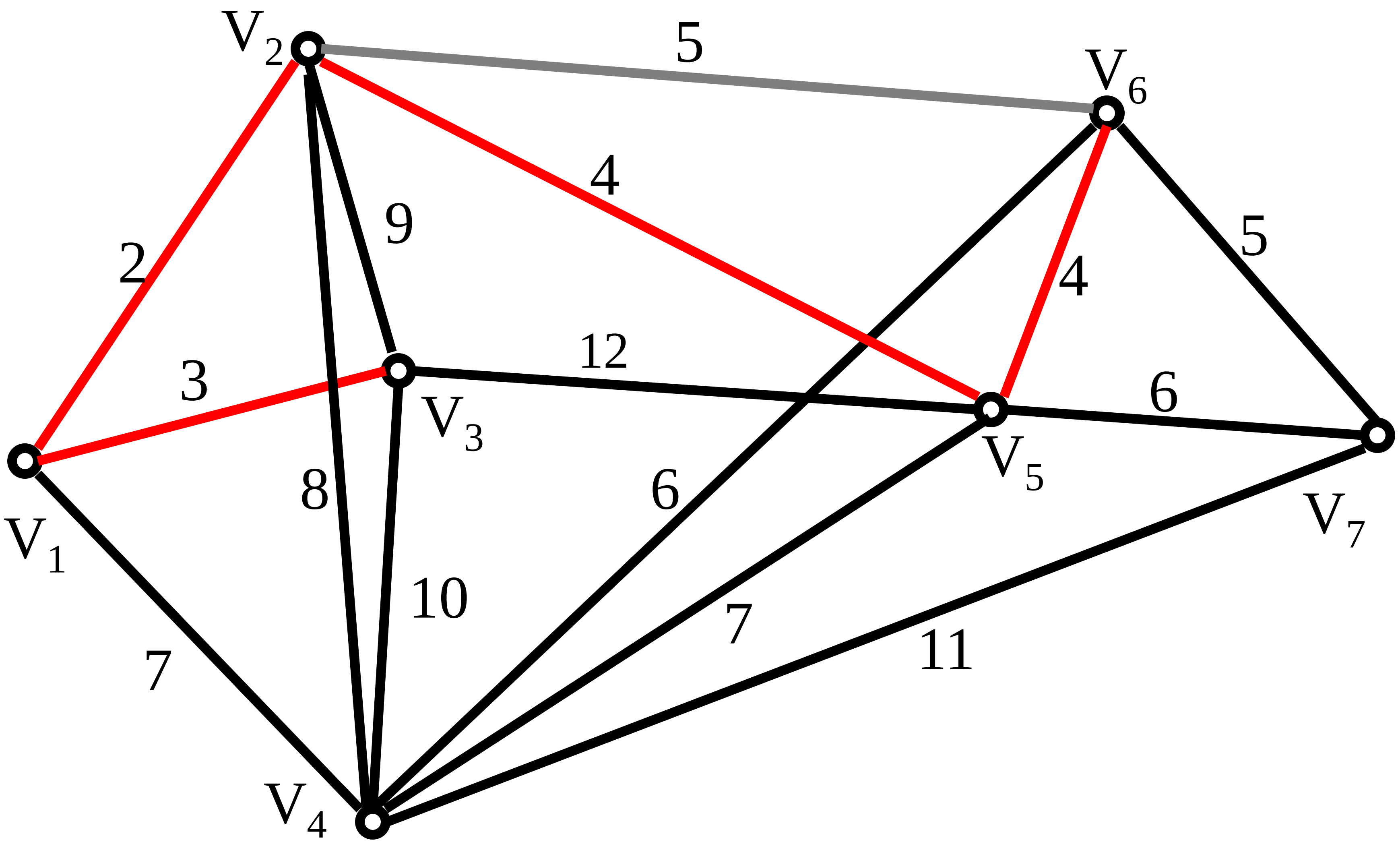
＝K算法举例（2） (第5页)



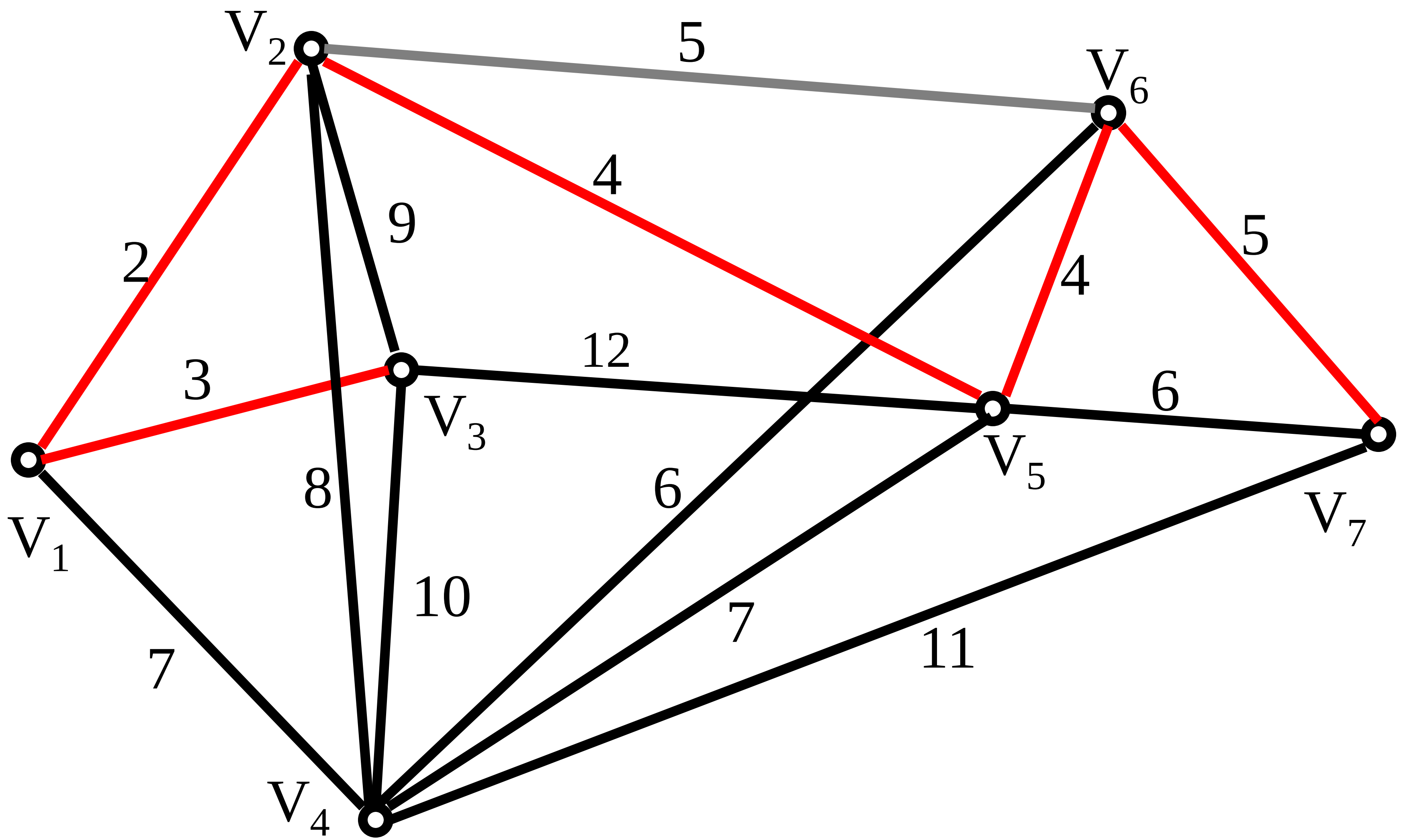
＝K算法举例（2）　　　　(第6页)



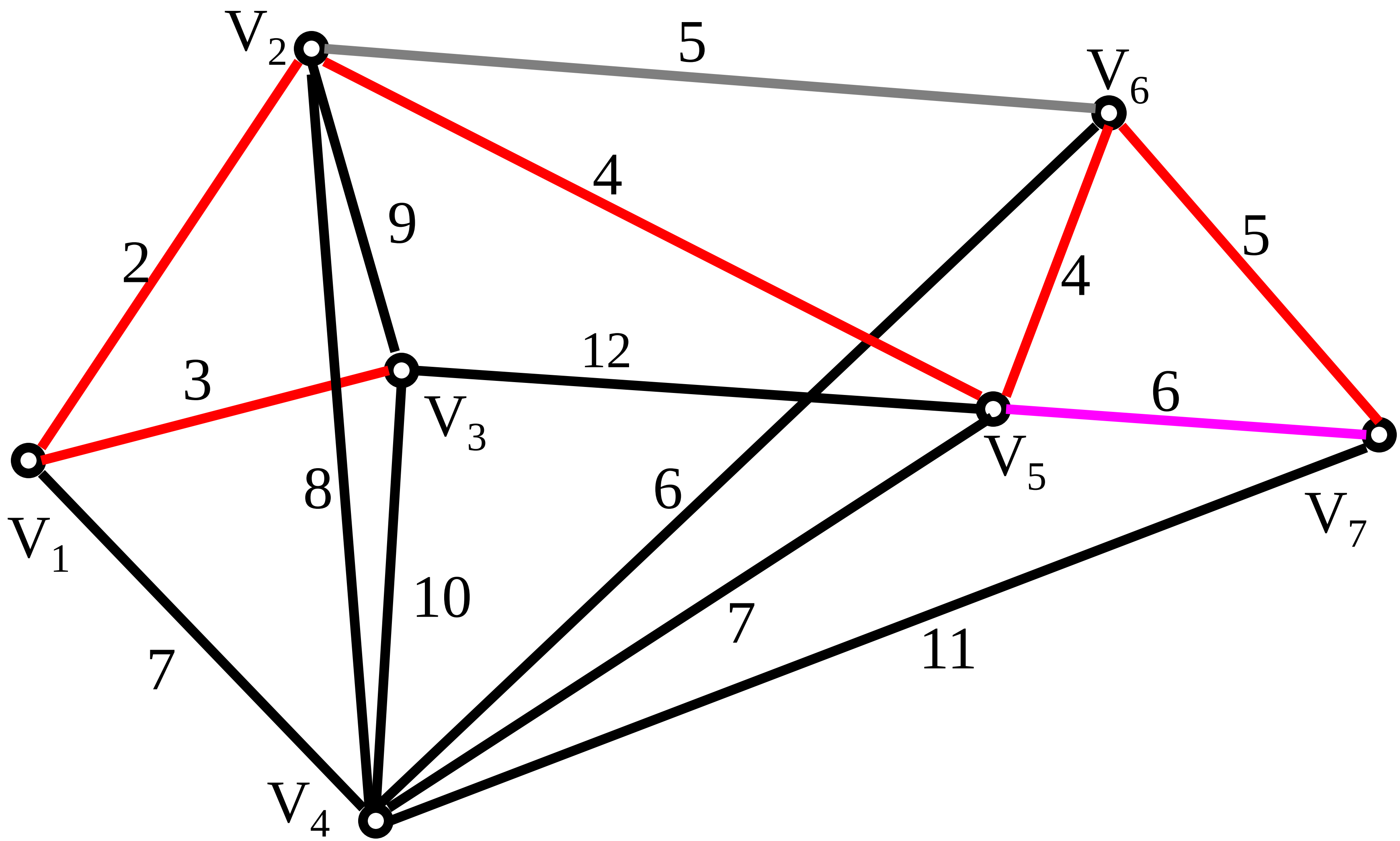
＝K算法举例（2）　　(第7页)



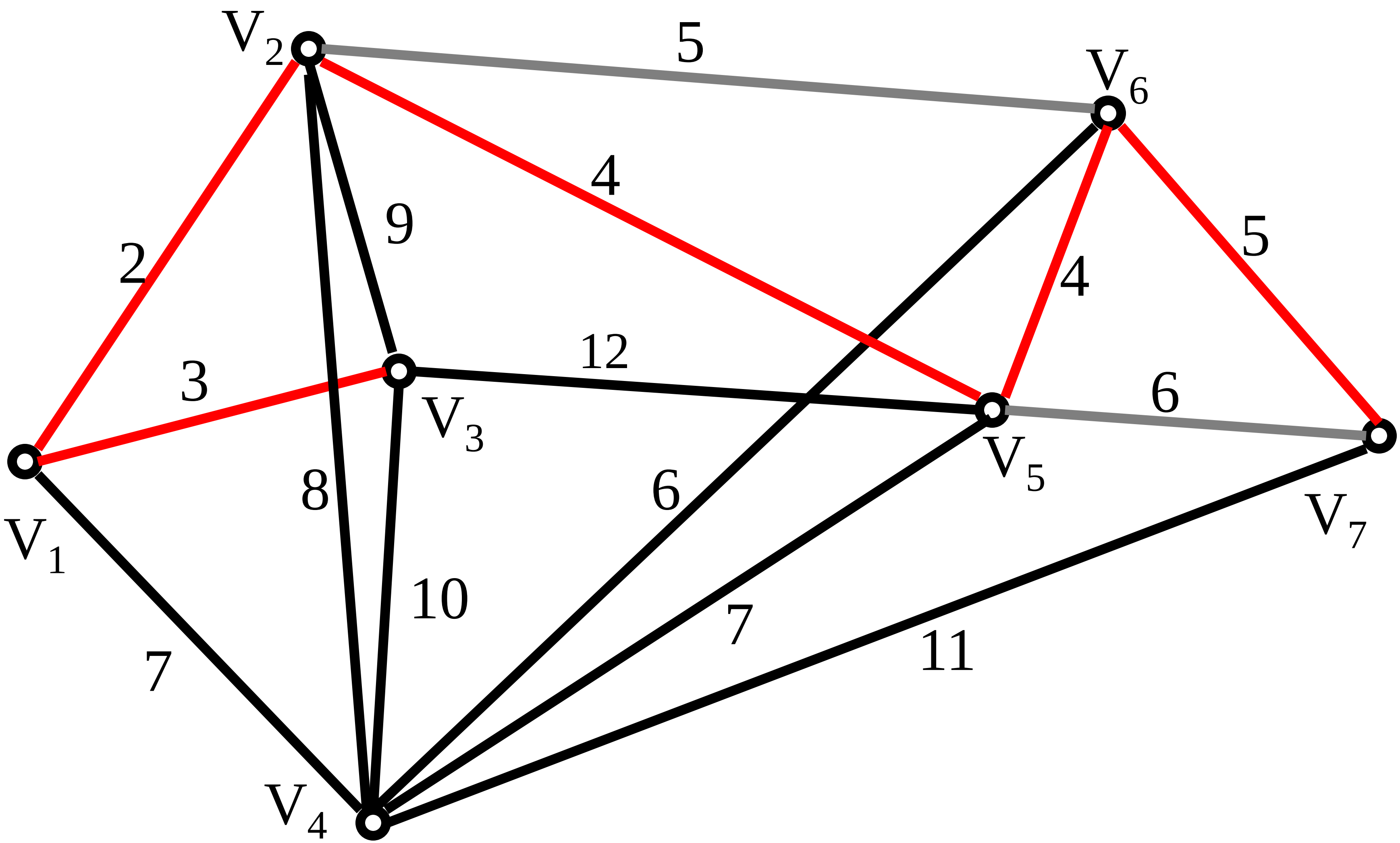
＝K算法举例（2） (第8页)



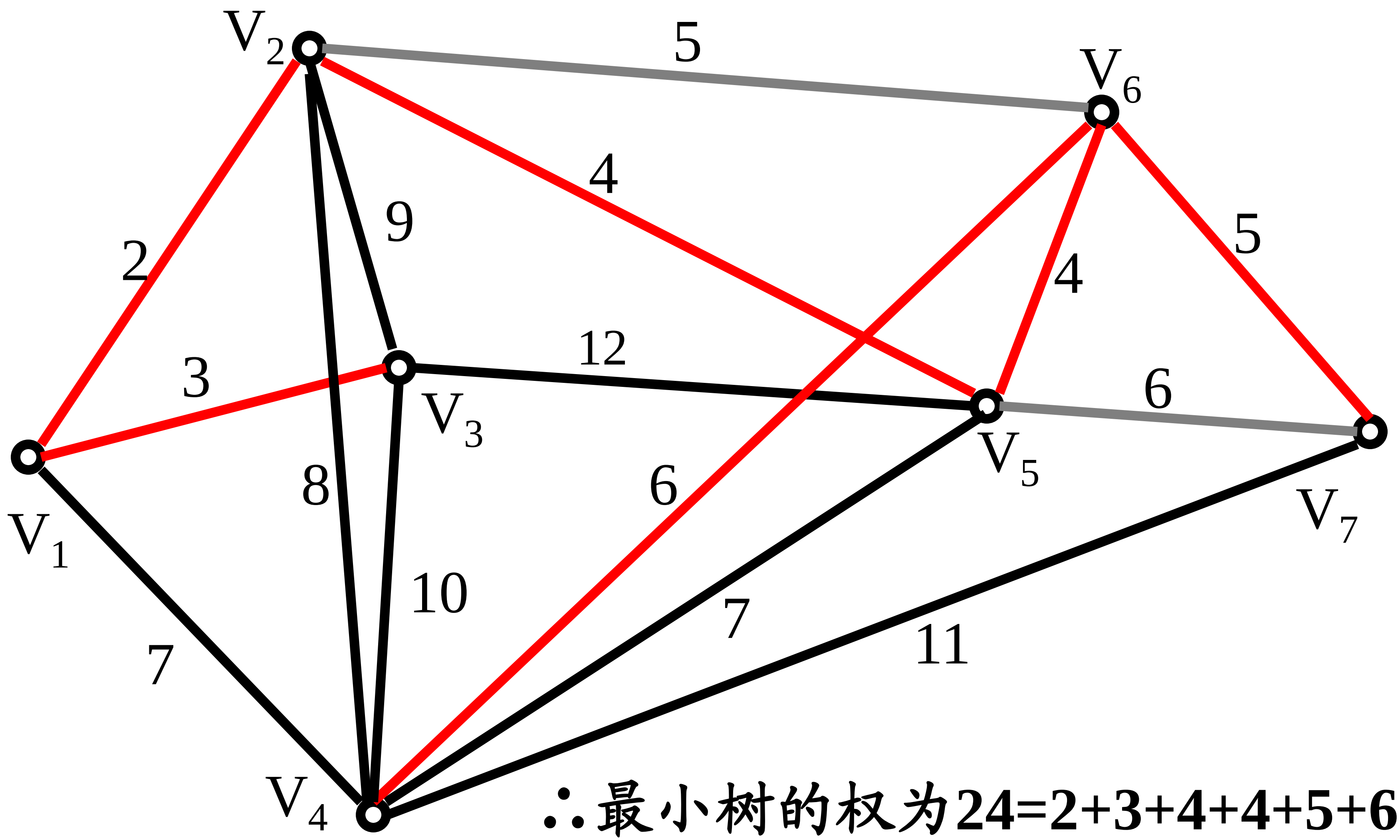
＝K算法举例（2）　　　(第9页)



＝K算法举例（2）　　　　(第10页)



＝K算法举例（2） (第11页)



∴最小树的权为24=2+3+4+4+5+6

最小树为T={e_{12},e_{13},e_{25},e_{56},e_{67},e_{64}}

K算法的复杂性

- K算法的复杂性主要决定于对边的排序
- 设原图有m条边，则共有m!种排法
  - 相当于$\log_2(m!)$ 次比较
- 设原图有n个端，则最多可以有$C_n^2 = n(n-1)/2$条边
  - 复杂性为$n^2 \log_2 n$量级，所以，也是多项式型的复杂问题
- 这里忽略了检查是否形成环的计算量
  - 可以证明，检查是否形成环的计算量肯定低于$n^2 \log_2 n$
  - 当n不大时，可以不计

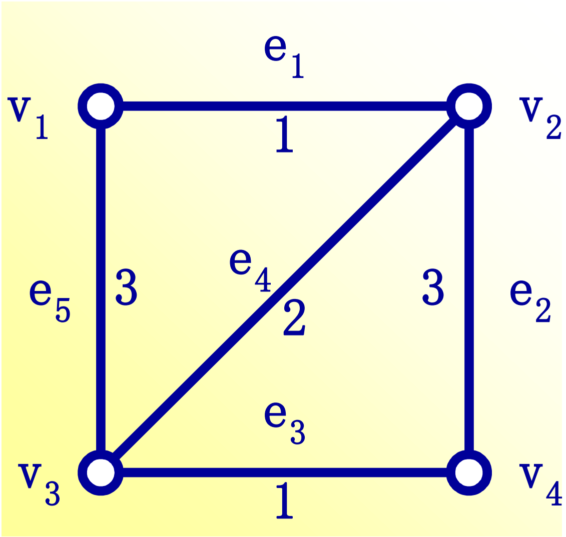
以上两种算法（P算法和K算法）的特点：

  - 都是预先设定了n个端
  - 再用边把它们联起来，所形成的最短树
- 若容许再增加一些端，而后联结起来
  - 则可以得到更短的树，称之为斯顿树

- 有限制条件的情况
  - ≡ 在许多情况下，网内n个站除了联通的要求外，还会有些其它要求
    - Δ 如：站间通信时，转接次数不宜过多
    - Δ 某一条线路上的业务量不能太大
    - Δ 某个连接的衰减，时延，噪声，··等不能超过规定值
  - ≡ 这类问题可归结为：在限制条件下求最短树
  - ≡ 有两种方法可以解决这类问题
    - Δ 穷举法和调整法
    - Δ 各有优缺点

＝穷举法

- △即把图中所有主树穷举出来
- △再按条件筛选，选出符合条件的最短主树
- △这是一种最直观，又最繁杂的方法
- △可以得到最佳解，但计算量往往很大
- △如n个端的全联结网，其主树数为n^{n-2}个
- △已属于非多项式的难题（**NP－Hard**）
- △因此，穷举法只能用于端数较少的网络



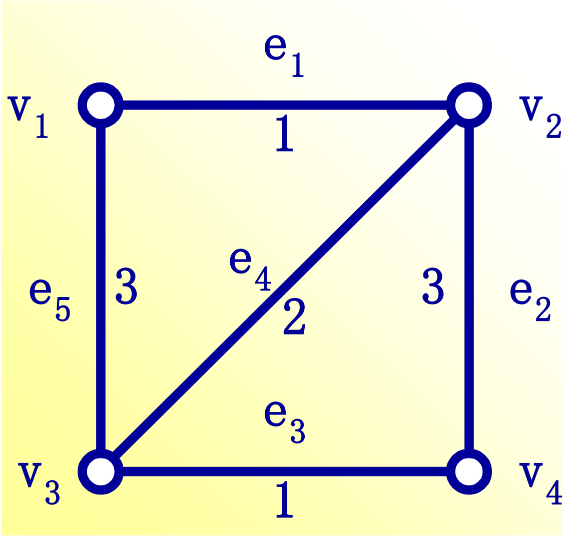
≡系统地求解全部主树的方法--置换法

- △以右图为例
- △第0步：首先确定主树的数量
  - 对角线上的元为该端的度数
  - 其它元非0即－1
  - 两端间无边，则为0
  - 两端间有边，则为－1

$$A_0A_0^T = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ 0 & -1 & -1 & 2 \end{bmatrix} \end{array}$$

△ 第0步：首先确定主树的数量（2）

- 从$A_0A_0^T$中去掉v_3所对应的行与列
- 可得到：



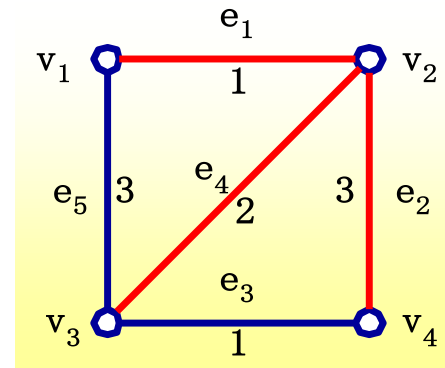
$$\left|AA^T\right| = \begin{vmatrix} 2 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{vmatrix} = 12+0+0-0-2-2=8$$

$$A_0A_0^T = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ 0 & -1 & -1 & 2 \end{bmatrix} \end{array}$$

- 实际上，去掉任一端所对应的行与列均可
- 通常去掉最后一行一列
- 此处为保留0，以简化计算，所以保留了v_4的行与列

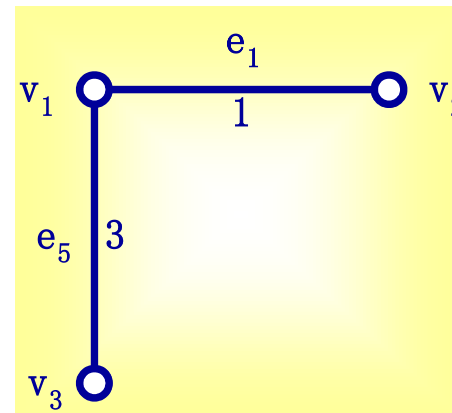
△ 第1步：随意找一棵主树作为参考树t_0

- 如t_0 = $\{e_1, e_2, e_4\}$

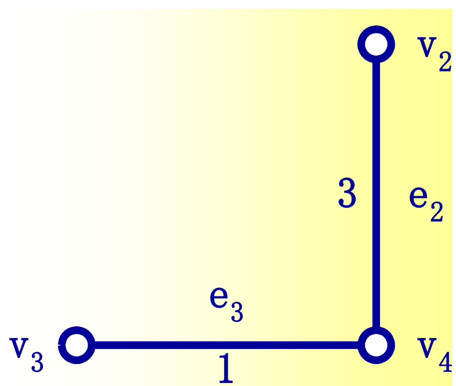


△ 第2步：找出t_0的基本割集

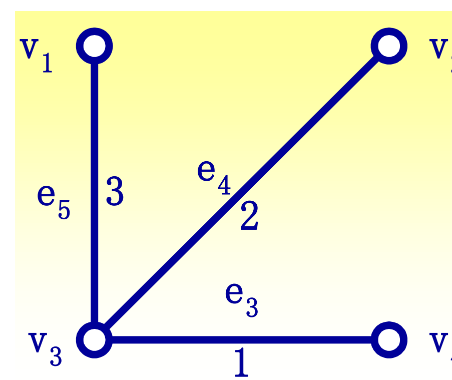
- $S_{e1}(t_0)$ = $\{e_1, e_5\}$



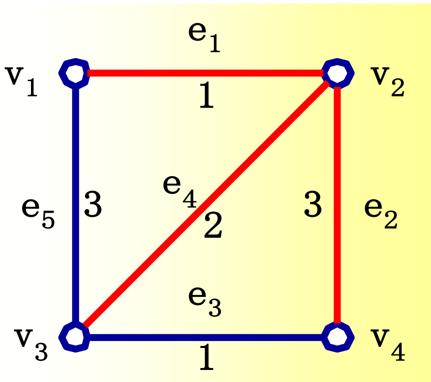
- $S_{e2}(t_0)$ = $\{e_2, e_3\}$



- $S_{e4}(t_0)$ = $\{e_5, e_3, e_4\}$



Δ 第3步：用基本割集中的连枝置换树枝

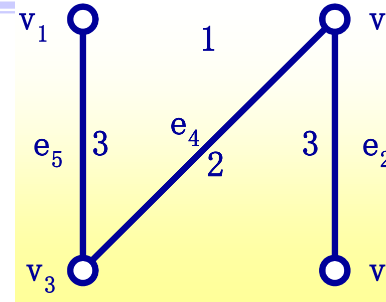


- 一定可得到另一主树
- 对全部基本割集都作置换，可得到**与参考树t_0距离为1的主树族**
- 亦即：与t_0只有一条边不同
- 可记为： $|t - t_0| = 1$

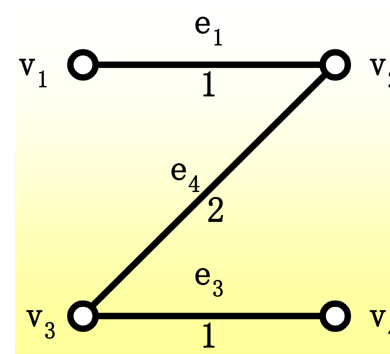
- 与t_0距离为1 的主树族为：
- $t_1 = \{e_5, e_2, e_4\}$ e_5换t_0的e_1
- $t_2 = \{e_1, e_3, e_4\}$ e_3换t_0的e_2
- $t_3 = \{e_1, e_2, e_3\}$ e_3换t_0的e_4
- $t_4 = \{e_5, e_1, e_2\}$ e_5换t_0的e_4

- **构成主树族$T_1 = \{t_1, t_2, t_3, t_4\}$**

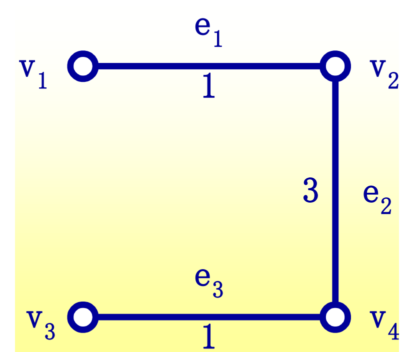
t_1



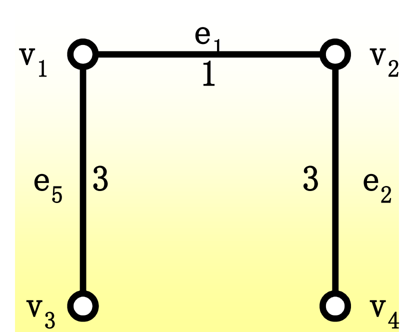
t_2



t_3

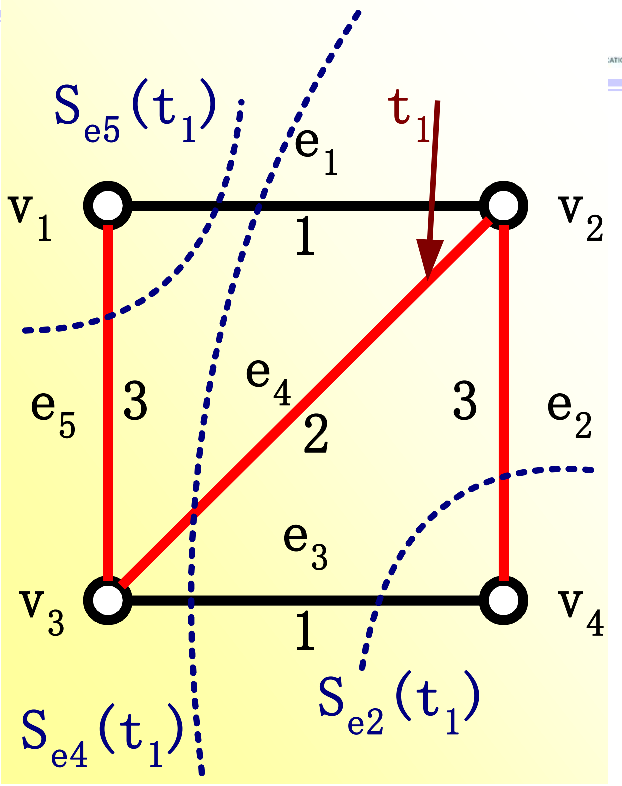


t_4

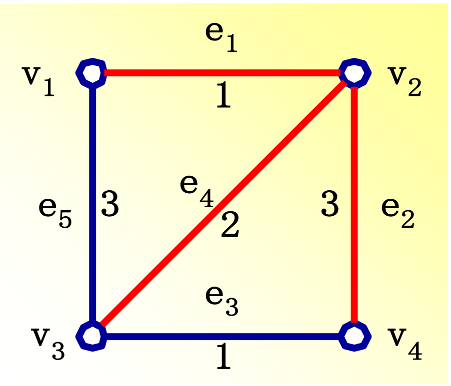


Δ 第4步：求距离t_0为2的主树族T_2

- 方法：对T_1中的树同样做
  连枝置换树枝
- 注意：不要回到t_0去

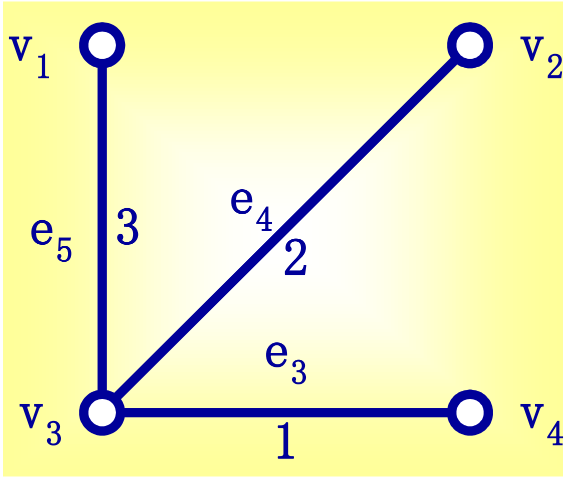


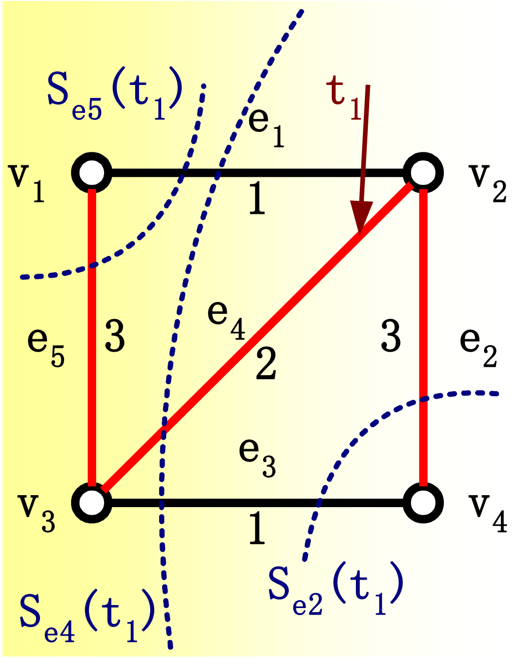
- 求t_1的基本割集
- $S_{e2}(t_1)=\{e_2, e_3\}$
- $S_{e4}(t_1)=\{e_4, e_1, e_3\}$
- $S_{e5}(t_1)=\{e_1, e_5\}$



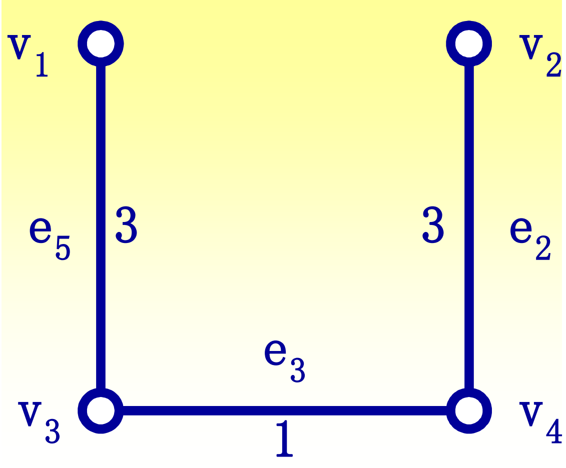
- 用连枝置换树枝
- 为了不重取t_0中的树枝
- 可先求上述割集与t_0中相应割集的交
- $S_{e2}(t_1)\cap S_{e2}(t_0)=\{e_2, e_3\}$
- $S_{e4}(t_1)\cap S_{e4}(t_0)=\{e_4, e_3\}$
- 其中e_3为连枝，e_2，e_4为树枝
- 只有当这些交集不空时，才能置换，否则将重取t_0或T_1中的某个树。曾作过t_0树枝的连枝，就不必再作置换了，因为这样也会重取t_0或T_1中的某个树

- 连枝e_3换t_1的树枝e_2：$t_5 = \{e_5, e_3, e_4\}$

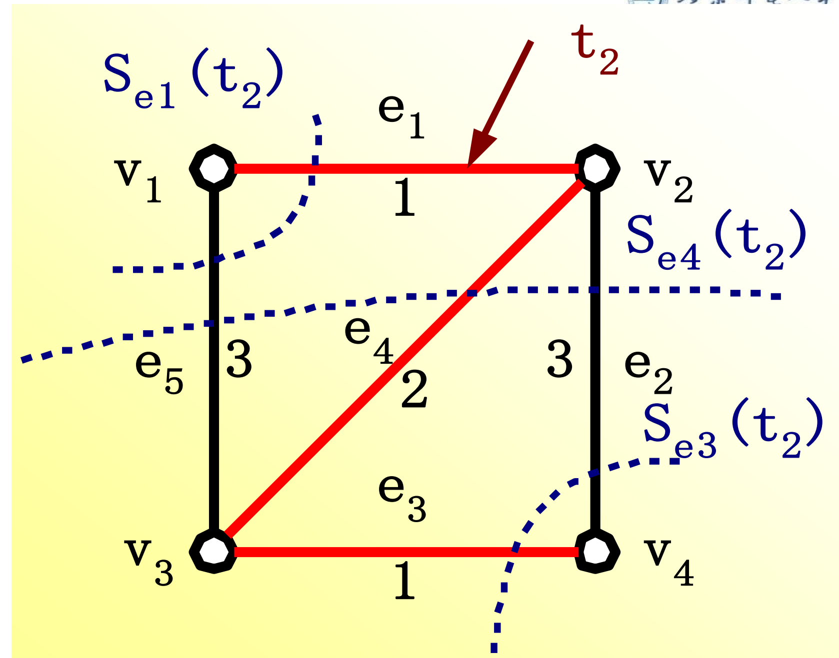




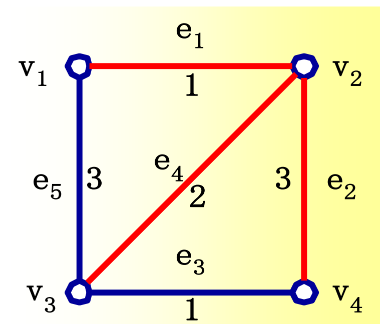
- 连枝e_3换t_1的树枝e_4：$t_6 = \{e_5, e_2, e_3\}$



- **求t_2的基本割集:**
- $S_{e1}(t_2) = \{e_1, e_5\}$
- $S_{e3}(t_2) = \{e_3, e_2\}$
- $S_{e4}(t_2) = \{e_4, e_2, e_5\}$

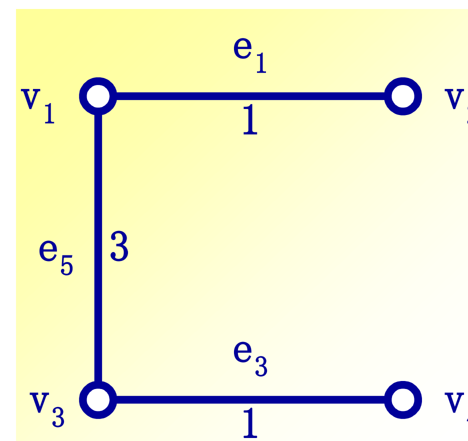


- 用连枝置换树枝
- 先求上述割集与t_0中相应割集的交



- $S_{e1}(t_2) \cap S_{e1}(t_0) = \{e_1, e_5\}$
- $S_{e4}(t_2) \cap S_{e4}(t_0) = \{e_4, e_5\}$
- 其中e_5为连枝，e_1，e_4为树枝

- 连枝e_5替换t_2的树枝e_4: $t_7 = \{e_1, e_3, e_5\}$
- 连枝e_5替换t_2的树枝e_1: $t_8 = \{e_3, e_4, e_5\} = t_5$
- 可见t_5与t_8重复了，只取其一即可



- **对t_3和t_4同样处理**
- 但会发现，所产生的树都是重复的
- 于是构成距离t_0为2的主树族$T_2 = \{t_5, t_6, t_7\}$

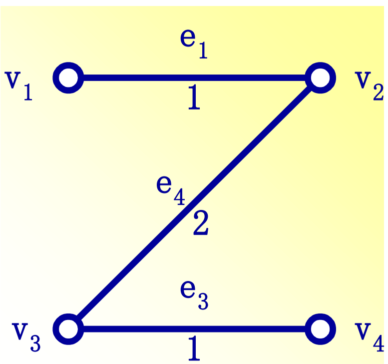
Δ 第5步：求距离t_0为3的主树族T_3

- 本例中，距离t_0为3的树已不存在

Δ 第6步：至此，我们已找出了全部主树：t_0，t_1，··t_7

- 每棵树的总长度为：

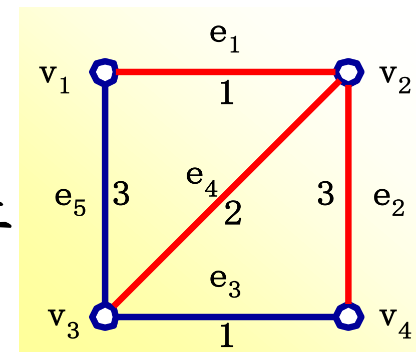
$$d_{t0}=6,\quad d_{t1}=8,\quad d_{t2}=4,\quad d_{t3}=5,\quad d_{t4}=7,\quad d_{t5}=6,\quad d_{t6}=7,\quad d_{t7}=5$$

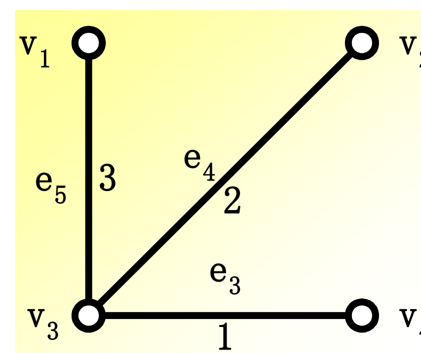


t_2

- 如果没有限制条件，则t_2是最短的
- （用P算法和K算法更易得到）

- 如果有限制条件，如：不许转接两次以上
- 则只有 t_0和 t_5满足条件
- 这两棵的总长度刚好是一样的
- 所以，任取一棵就是满足条件的最短树





3.2.2 端间的最短径问题

- 若通信网的拓扑结构已确定，寻找站间最短路由是有意义的
  - 比如互联网（IP）的选路原则就是最短路由原则
    - OSPF: Opening Shortest Path First
- 一般有如下几种情况需要研究
  - 求指定端到其它端的最短路径
  - 求任意两端间的最短路径
  - 求次短路径或供备用的可用径
  - 求图的中心和中点

- 指定端到其它端的最短径算法
  - 迪克斯恰算法（**Dijkstra**算法，简称**D**算法）
    - 给定图**G**
    - 已知所有边的权值d_{ij}
    - 求指定端v_s至其它端的最短径
    - **D**算法是有效算法之一
  - 算法思路
    - 把端集分为两组:
      - 置定端集 (标定端集) G_p
      - 未置定端集$G-G_p$
    - 每端都逐步给予一个<u>标值</u>
      - 对于置定端：这个标值就是v_s到该端的最短径的长度和最后一个端的编号
      - 对于未置定端：这个标值是一个暂定值，是当前的最短径长
        - 其值还会随着算法的进展而调整

≡在图G=(V,E)中，每一条边e_{ij}上都有一个权值 $w(e_{ij}) = d_{ij} \geq 0$

≡设μ是图G中的一条链，定义链μ的权值为：

$$w(\mu) = \sum_{e_{ij} \in \mu} w(e_{ij})$$

≡每个端v_i的<u>标值</u>为（w_i, v_l）

Δw_i：是v_s端到v_i端当前的最短径的长度

Δv_l：是端点编号，是当前v_s端到v_i端的最短径上的最后一个端

- 这两个值还会随着算法的进展而调整
- 这种路由的方法叫做**回溯路由** 或 **反向路由**

- ≡ 开始时，只有指定端v_s为置定端，标值为零
  - Δ 即$G_p = \{v_s\}$
  - Δ 其它端都是未置定端，暂标值为$w_j = \infty \quad (v_j \in G - G_p)$
- ≡ 若 $d_{s1} = \min\limits_{v_j \in G - G_p} d_{sj}$
  - Δ 则将v_1置定，$G_p = \{v_s, v_1\}$
  - Δ 其标值为：$w_1 = d_{s1}$
  - Δ 这就是v_1与v_s之间的最短径
  - Δ 因为再经其它端转接，径长必大于w_1
- ≡ 此后，其它端的最短径可以经v_1转接
  - Δ 所以需重新计算未置定端的标值：$w_j^* = \min\limits_{v_j \in G - G_p} (w_j, w_1 + d_{1j})$
- ≡ 设取出的最小值为w_2
  - Δ 则v_2端被置定$G_p = \{v_s, v_1, v_2\}$
  - Δ 再更新各未置定端的标值
  - Δ 如此一直计算下去，直到置定了所有端
  - Δ 所有置定的标值就是各最短径长

= **D算法步骤:**

≡ **D_0：起始**

- △ 置定v_s， G_p = {v_s}
- △ 置w_s = 0， w_s：是v_s − v_s的距离
- △ 暂置w_j = ∞ （$v_j \in G - G_p$）， w_j：是v_s − v_j的距离

≡ **D_1：计算暂置标值**

△

$$w_j^* = \min_{\substack{v_j \in G - G_p \\ v_i \in G_p}} (w_j, w_i + d_{ij})$$

- △ 其中：w_i是前一次置定的端v_i的标值
- △ w_j是v_i被置定前的v_j端的暂置标值
- △ w_j^*是v_i被置定后的v_j端的暂置标值
- △ 计算完成后，w_j^*即改记为w_j

≡ D_2：取最小值

$$\Delta \quad w_{j*} = \min_{v_j \in G - G_p} w_j$$

Δ 将v_{j*}并入G_p

Δ 若 $|G_p| = n$，则终止

Δ 若 $|G_p| < n$，则返回D_1

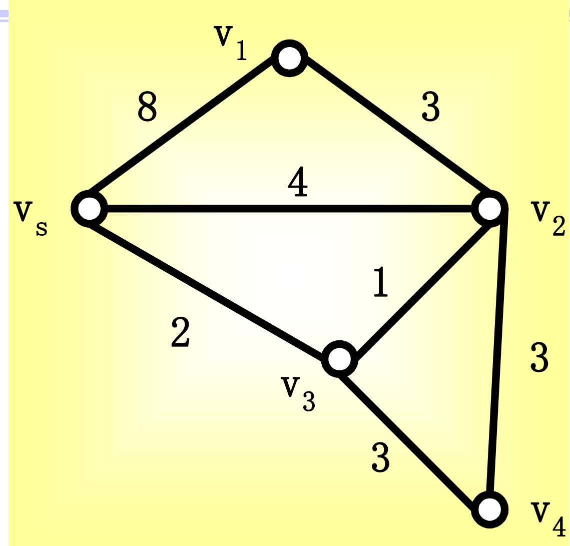
≡ 用以上步骤，可得到所有端到v_s的最短径长

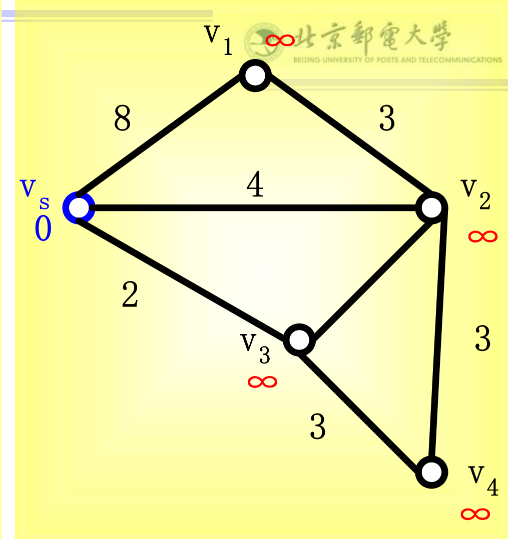
Δ 最短径可由计算过程看出

Δ 当暂置标值变更时，说明经过一次转接

= D算法举例

≡ 如图：





≡ D_0：

△ 置定v_s，$G_p = \{v_s\}$

△ 置$w_s = 0$，$w_j = \infty$

• $j = 1, 2, 3, 4$

≡ D_1：计算暂置标值

△ $w_1^* = \min(w_1, \ w_s + d_{s1}) = \min(\infty, \ 0+8) = 8$

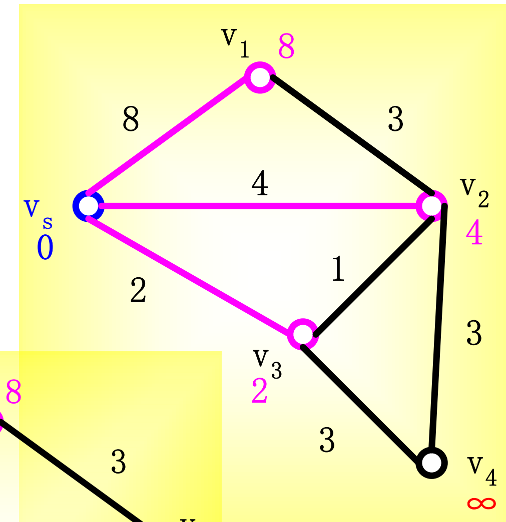
△ $w_2^* = \min(w_2, \ w_s + d_{s2}) = \min(\infty, \ 0+4) = 4$

△ $w_3^* = \min(w_3, \ w_s + d_{s3}) = \min(\infty, \ 0+2) = 2$

△ $w_4^* = \min(w_4, \ w_s + d_{s4}) = \min(\infty, \ 0+\infty) = \infty$

• w_i^*与w_i相比发生变化的，

• 则用w_i^*更新w_i，如右图

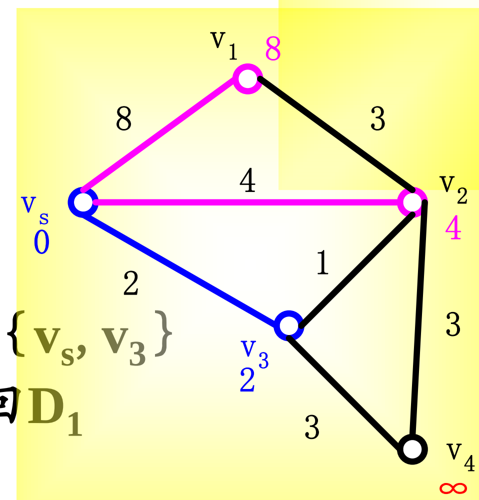


≡ D_2：取最小标值

△ w_3是其中最小标值

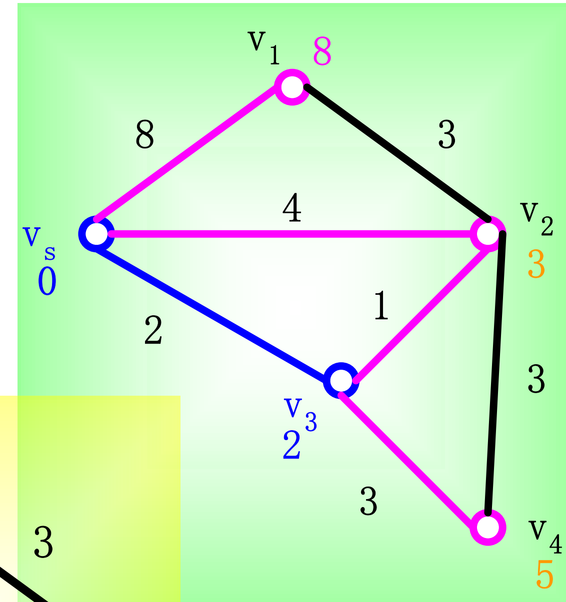
△ 置定v_3，将v_3并入置定端集$G_p = \{v_s, v_3\}$

△ $|G_p| = 2 < n = 5$，所以，返回D_1



≡ D_1：更新暂标值

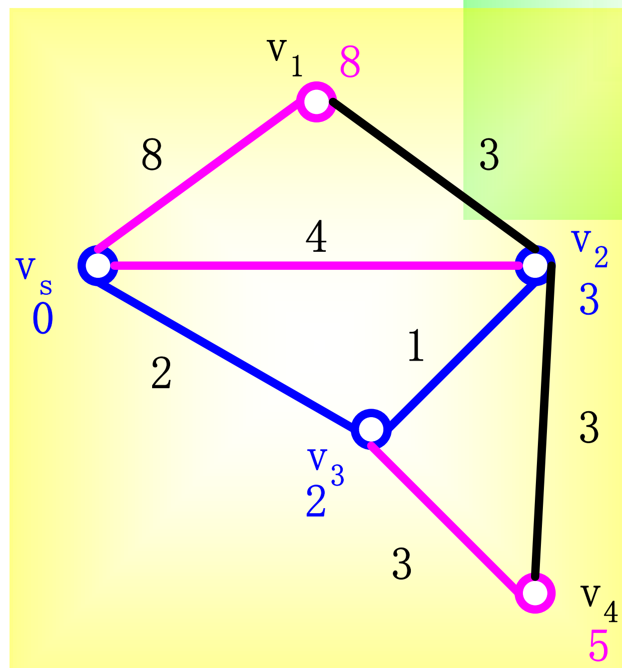
- △ $w_1^* = \min(w_1,\ w_3 + d_{31}) = \min(8,\ 2+\infty) = 8$
- △ $w_2^* = \min(w_2,\ w_3 + d_{32}) = \min(4,\ 2+1) = 3$ （发生变化）
- △ $w_4^* = \min(w_4,\ w_3 + d_{34}) = \min(\infty,\ 2+3) = 5$ （发生变化）
  - w_i^* 与 w_i 相比发生变化的，
  - 则用 w_i^* 更新 w_i，如右图



≡ D_2：取最小标值

- △ w_2 是其中最小标值
- △ 置定 v_2，将 v_2 并入 G_p
  - $G_p = \{v_s, v_3, v_2\}$
- △ v_s 至 v_2 的最短径为：

  $v_s \rightarrow v_3 \rightarrow v_2$
- △ $|G_p| = 3 < n = 5$
  - 所以，返回 D_1

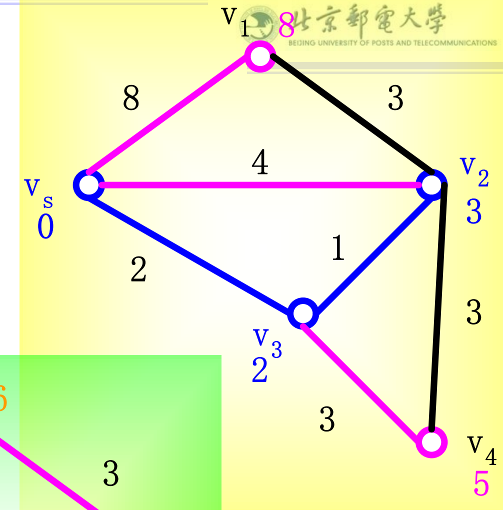


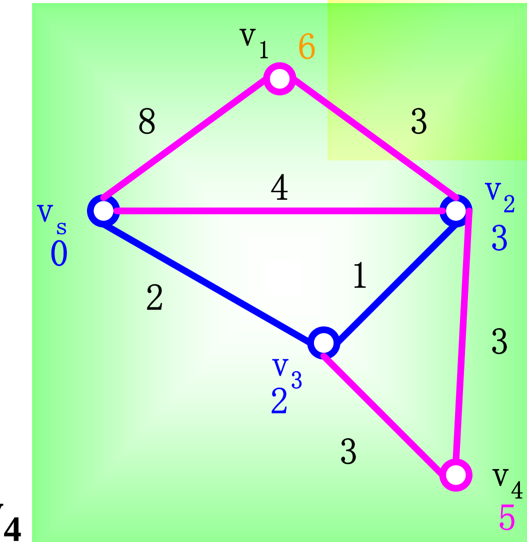
≡D_1：更新暂标值

△ $w_1^* = \min(w_1,\ w_2 + d_{21}) = \min(8,\ 3+3) = 6$

△ $w_4^* = \min(w_4,\ w_2 + d_{24}) = \min(5,\ 3+3) = 5$

- w_i^*与w_i相比发生变化了，
- 则用w_i^*更新w_i，如图





≡D_2：取最小标值

△ w_4为最小值

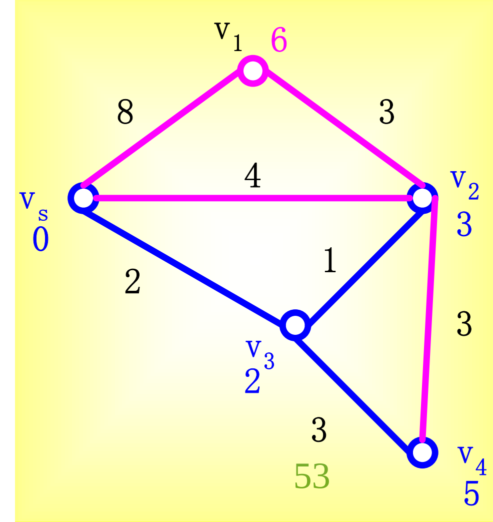
△ 置定v_4，将v_4并入G_p

- G_p = ｛v_s, v_3, v_2, v_4｝

△ v_s至v_4的最短径为：$v_s \to v_3 \to v_4$

- 此次w_4未发生变化，说明未经v_2转接
- w_4采用的是上次的值
- 而上一次w_4发生了变化，说明经过v_3转接了

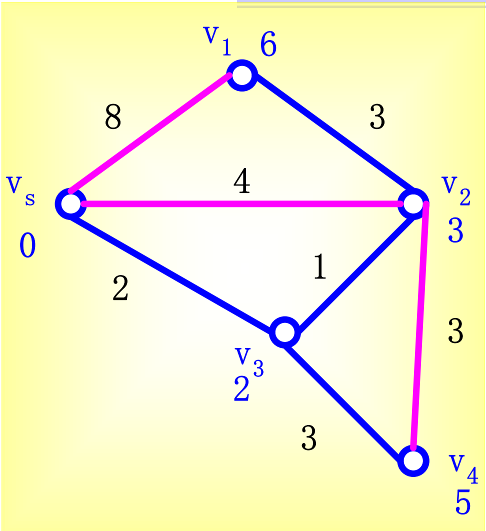
△ ｜G_p｜＝4 ＜ n ＝ 5，所以，返回D_1



≡ D_1：更新暂标值

Δ $w_1^* = \min(w_1,\ w_4 + d_{41}) = \min(6,\ 5+\infty) = 6$

≡ D_2：取最小标值



Δ 置定v_1，将v_1并入G_p，$G_p = \{v_s, v_3, v_2, v_4, v_1\}$

Δ v_s至v_1的最短路径为：$v_s \to v_3 \to v_2 \to v_1$

- 此次w_1未发生变化, 所以说明未经v_4转接
- w_1采用的是上次的值
- w_1的上一次发生了变化，说明经过v_2转接了
- 而v_2的最短径为：$v_s \to v_3 \to v_2$

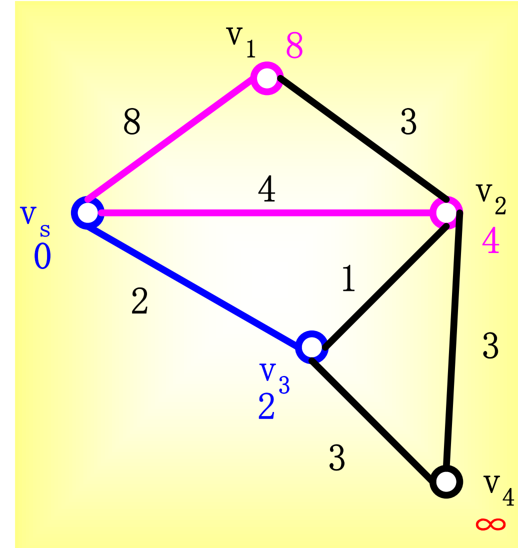
Δ ｜G_p｜＝5＝n，终止

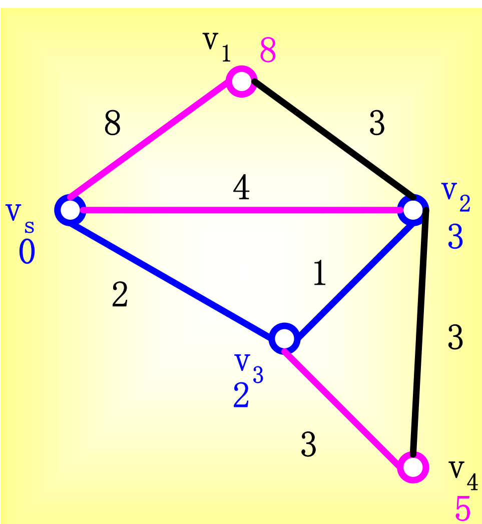
≡ 计算结果可列表如下：

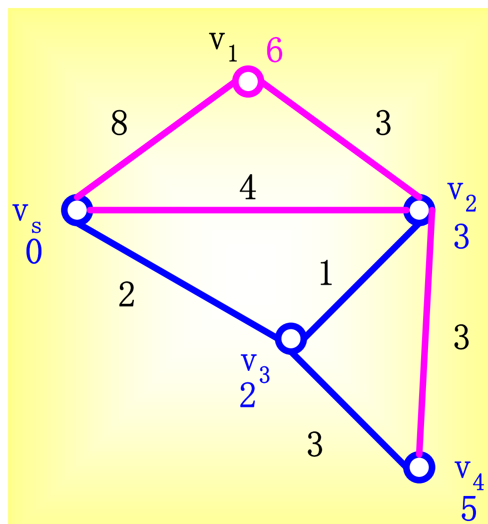
径长 w_j					置定	最短径长	最短路径
v_s	v_1	v_2	v_3	v_4			
0	∞	∞	∞	∞	v_s	$w_s = 0$	
	8	4	2	∞	v_3	$w_3 = 2$	$v_s \to v_3$
	8	3		5	v_2	$w_2 = 3$	$v_s \to v_3 \to v_2$
	6			5	v_4	$w_4 = 5$	$v_s \to v_3 \to v_4$
	6				v_1	$w_1 = 6$	$v_s \to v_3 \to v_2 \to v_1$

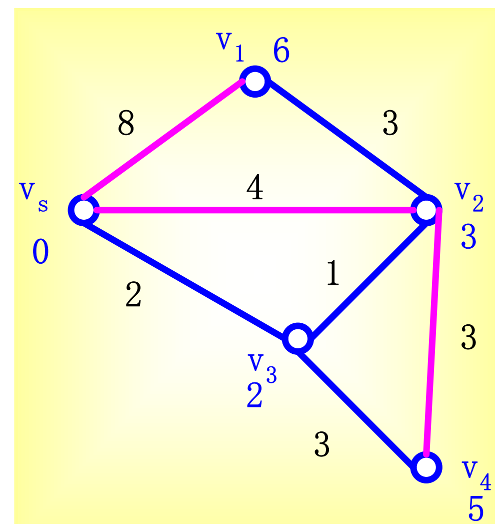
D算法的实质

- D算法求出的其实是一棵最短主树
- 只是比最短主树多了一个限制条件：
  - 从v_s到各端去的路径最短
- 所以，它与Prim算法很像
  - 都是取两个端集G_p与$G-G_p$之间的最短边
  - 但要限定从v_s到各端去的路径最短





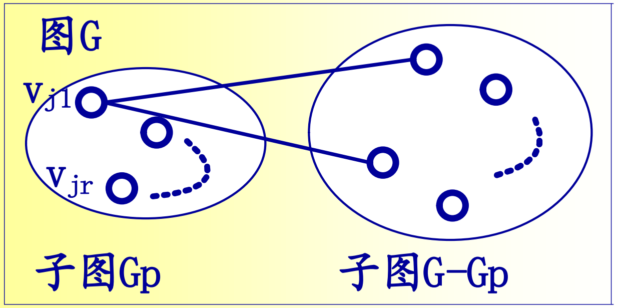




D算法运算量的估计

$$w_j^* = \min_{\substack{v_j \in G-G_p \\ v_i \in G_p}} (w_j, w_i + d_{ij})$$

- 在第k步时
  - 更新各端的暂置值：要做（n－k）次加法
  - 再做（n－k）次比较
  - 求最小值：要做（n－k－1）次比较。
  - 共需：3(n－k)－1次运算



- 总运算量约为：
  - $$\sum_{k=1}^{n} 3\cdot(n-k) = \frac{3}{2} n\cdot(n-1)$$

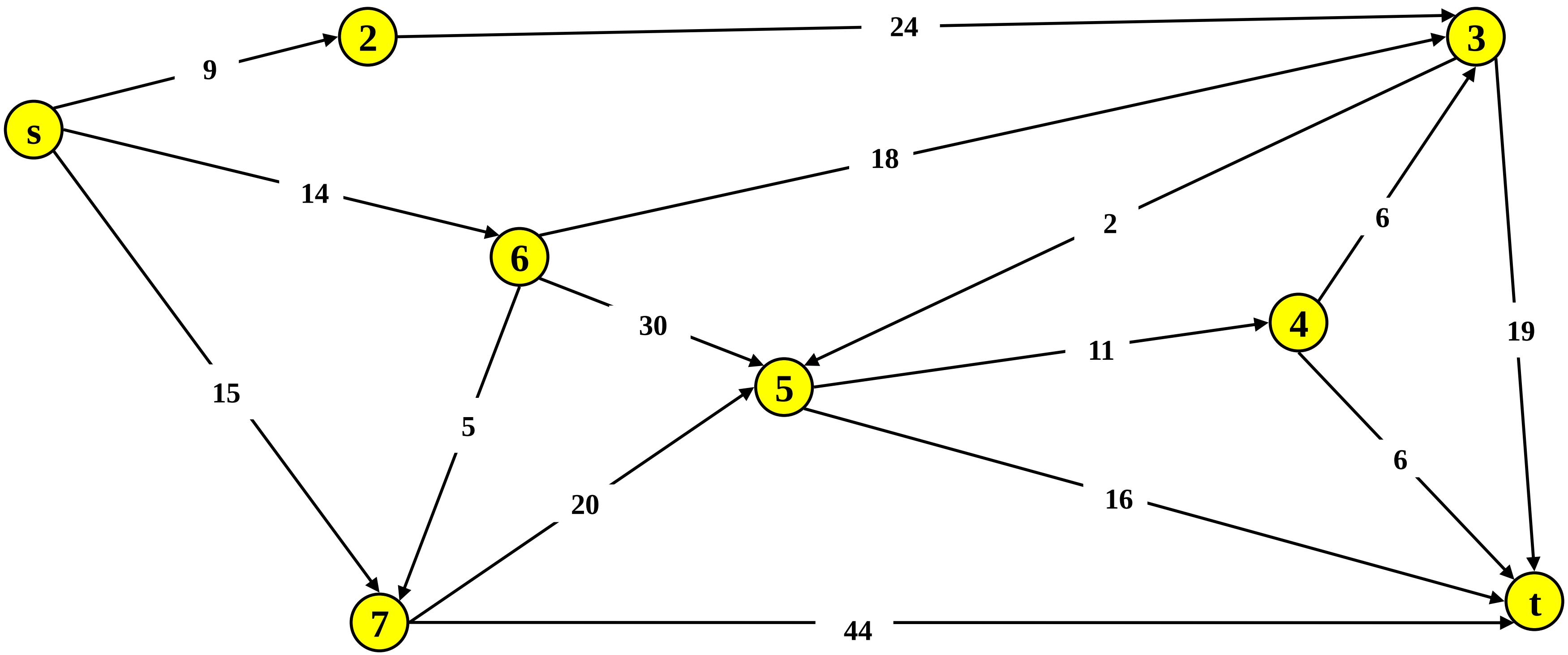
- 计算复杂性为n^2量级
  - 可以在计算机上实现

- 当图中的端点也有权值时
  - 可以对图做一些变换，使满足使用D算法的条件
  - 设：端点v_i的权值为w_i，
  - 可将w_i的1/2 加到v_i端的所有邻边上
  - 此端作为转接端时，一入一出径长就多出了w_i
  - 这样新图的端就没权值了，可以使用D算法求解了
  - 最后，将终点的权值从相应的总距离中扣除掉
  - 即，对距离做一点修正即可
- 当图中边的权有正有负时
  - 则不能应用D算法
  - 因为这种情况下无法保证算法的第2步中选出的边
  - 能使v_i到v_s的距离最小
- D算法对有向图也是适用的

D算法举例（之二）

– 找到s 到 t的最短路径

= 如下图



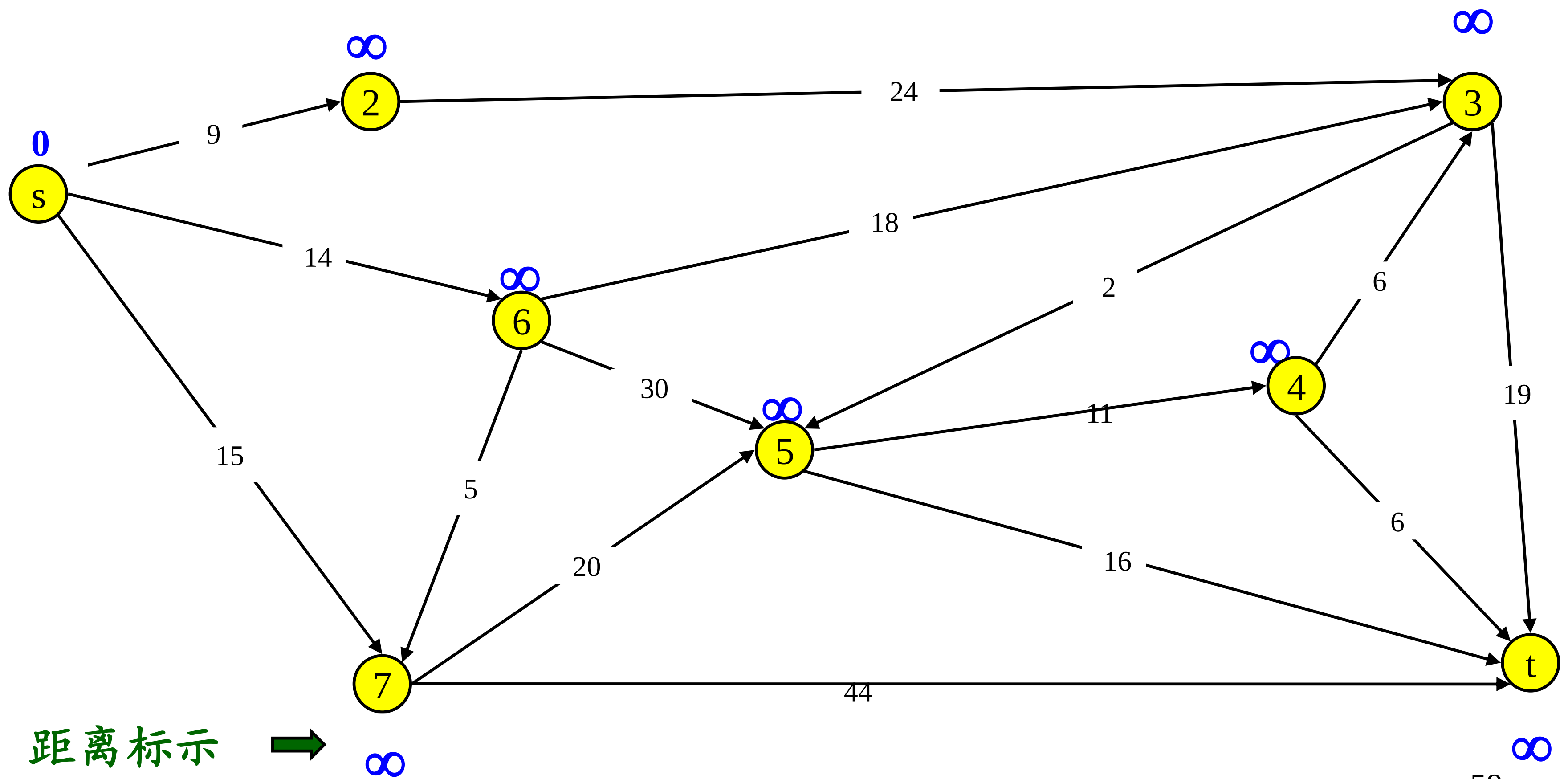
D算法举例（之二）

– 初始化:

Gp = { }

G−Gp = { s, 2, 3, 4, 5, 6, 7, t }

= 距离标示为从s端至本端的距离

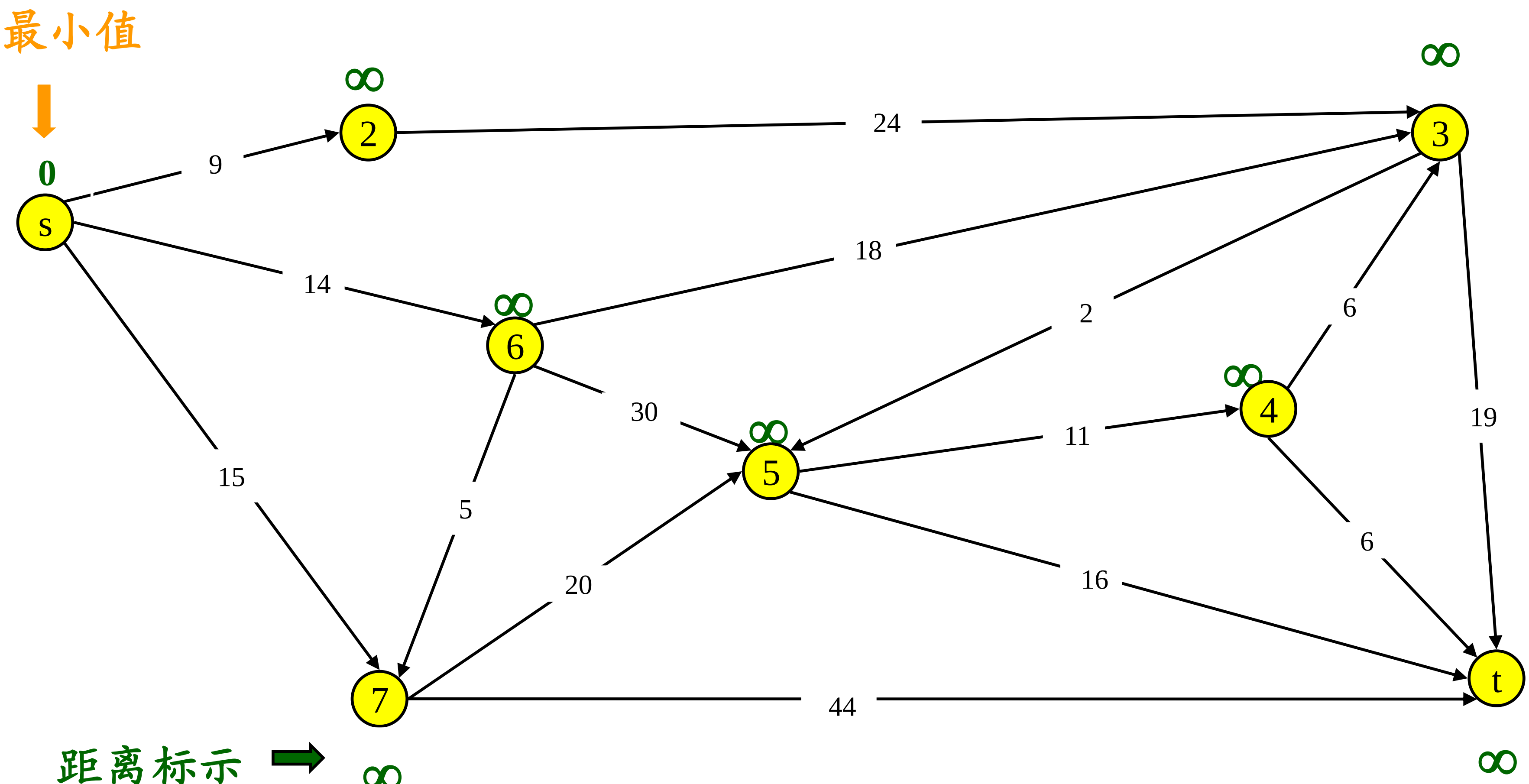


距离标示 ➡

D算法举例（之二）

Gp = { }

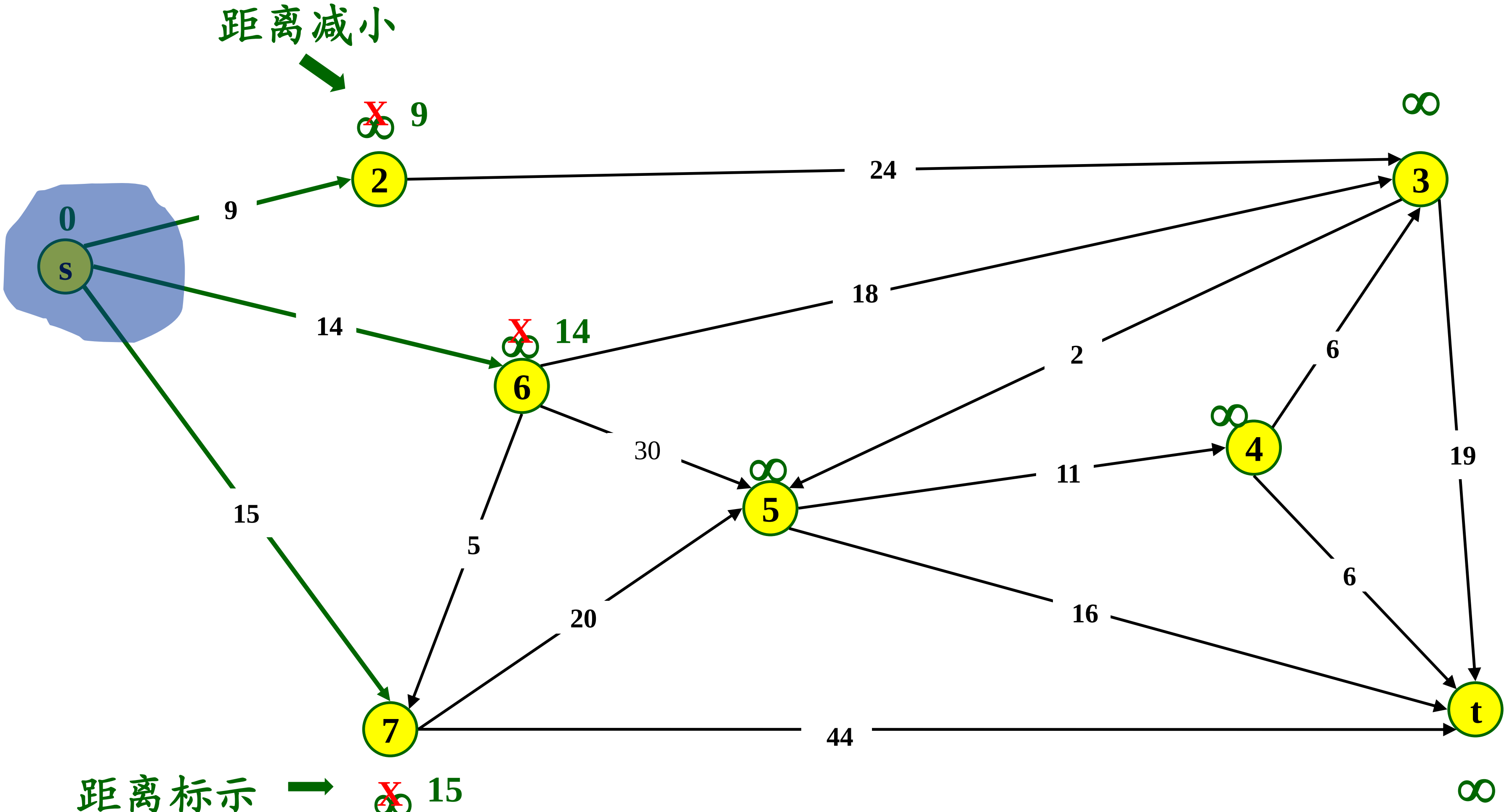
G−Gp = { s, 2, 3, 4, 5, 6, 7, t }



D算法举例（之二）

Gp = { s }

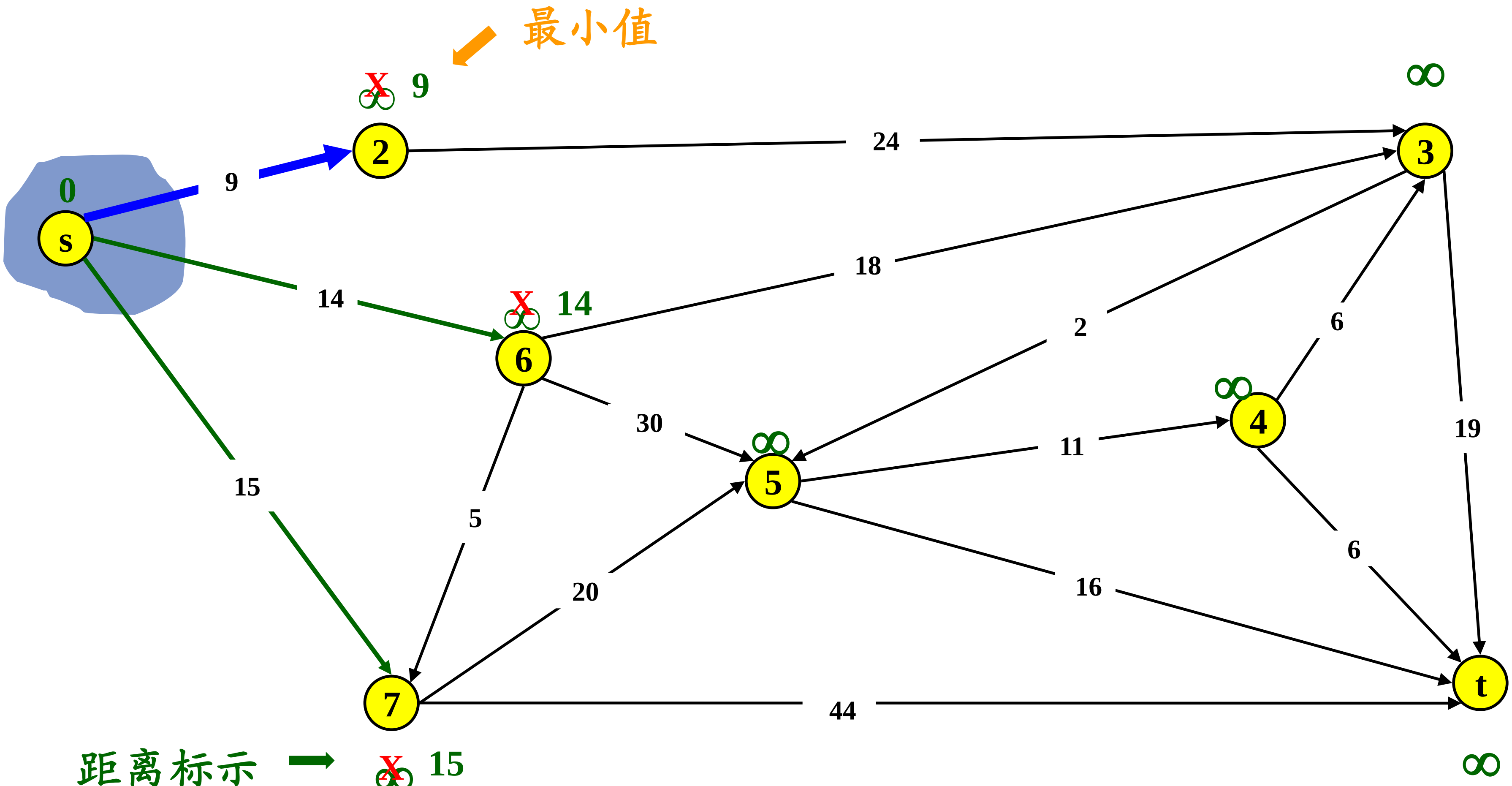
G−Gp = { 2, 3, 4, 5, 6, 7, t }



D算法举例（之二）

Gp = { s }

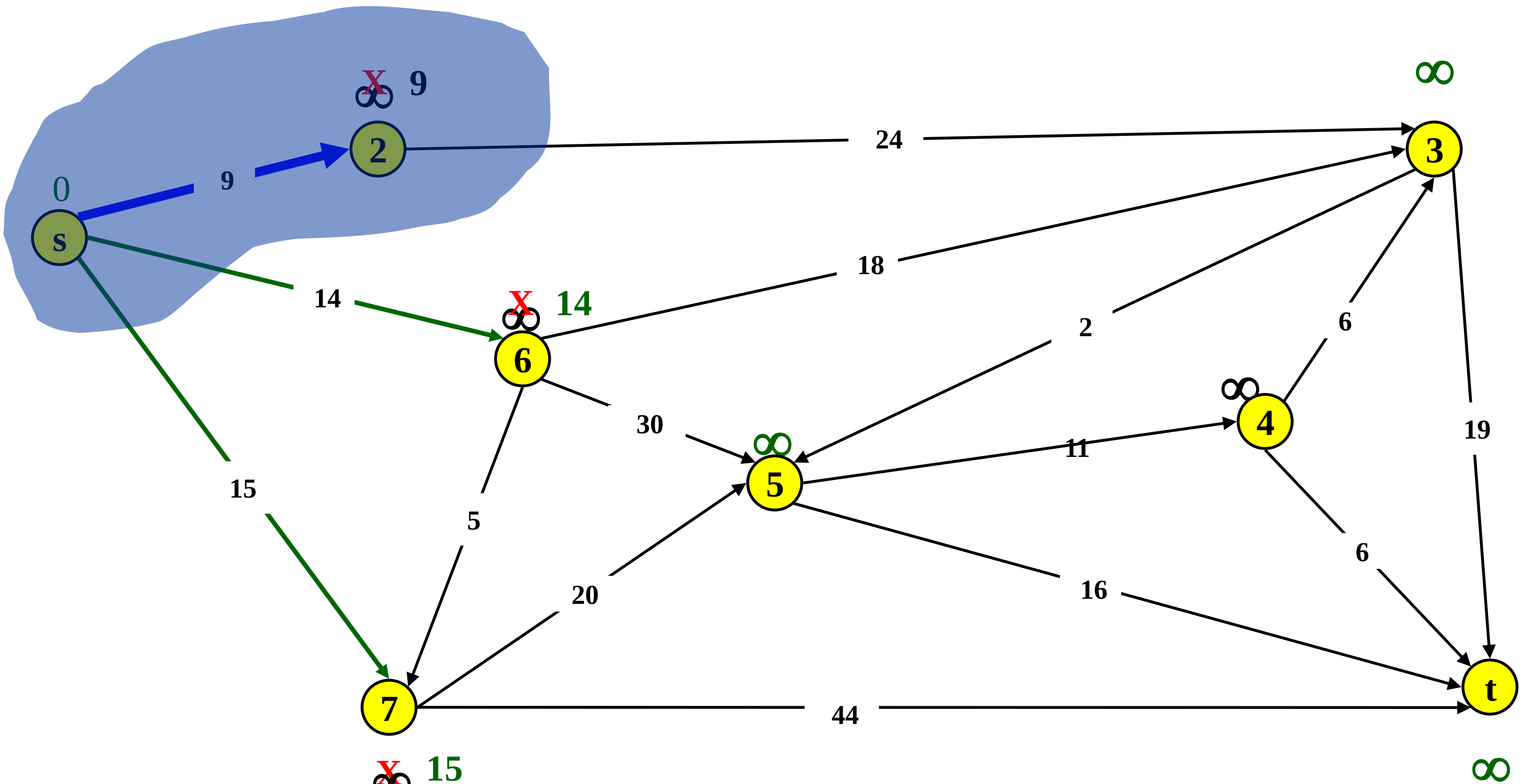
G−Gp = { 2, 3, 4, 5, 6, 7, t }



D算法举例（之二）

Gp = { s, 2 }

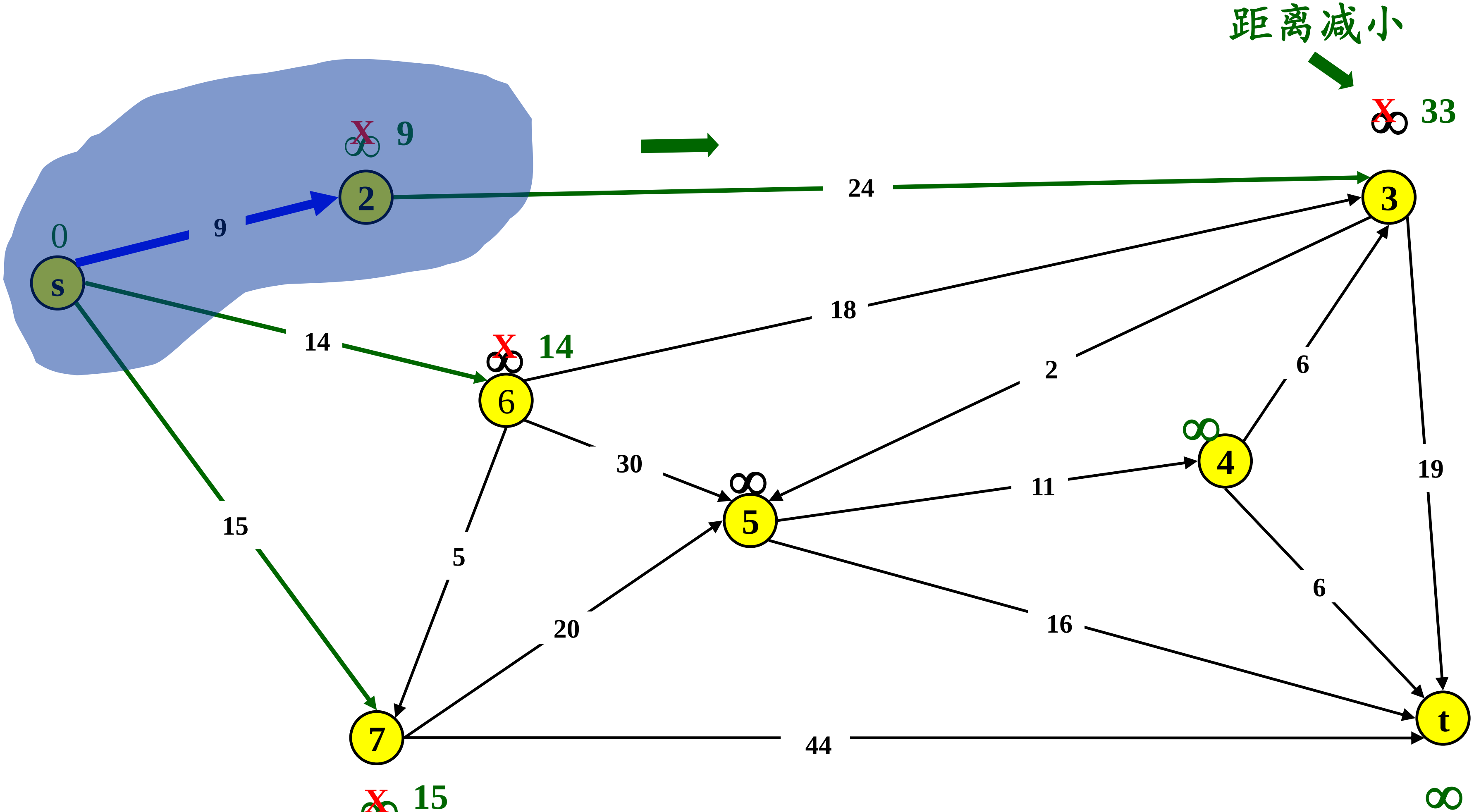
G−Gp = { 3, 4, 5, 6, 7, t }



D算法举例（之二）

Gp = { s, 2 }

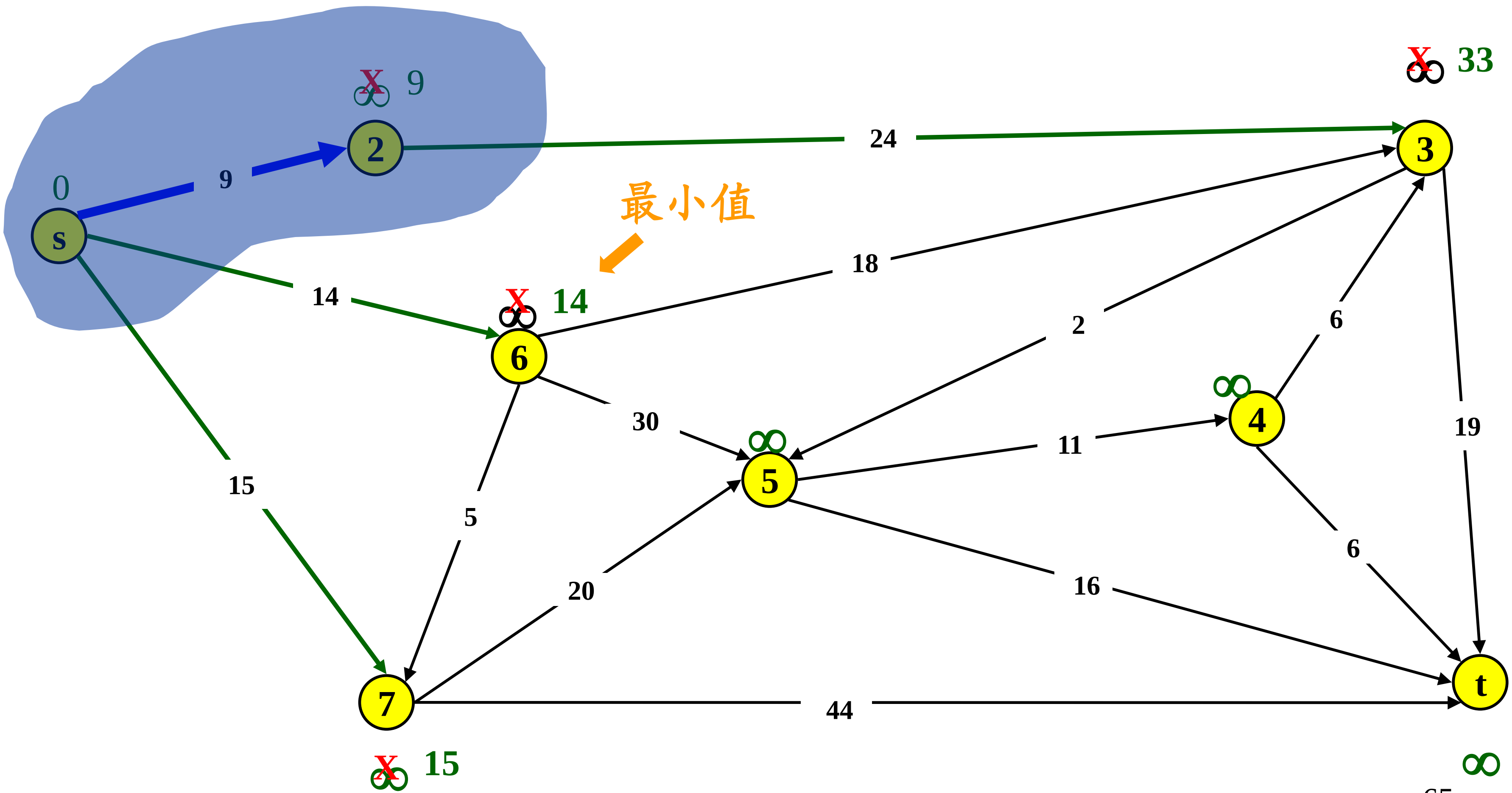
G−Gp = { 3, 4, 5, 6, 7, t }



D算法举例（之二）

Gp = { s, 2 }

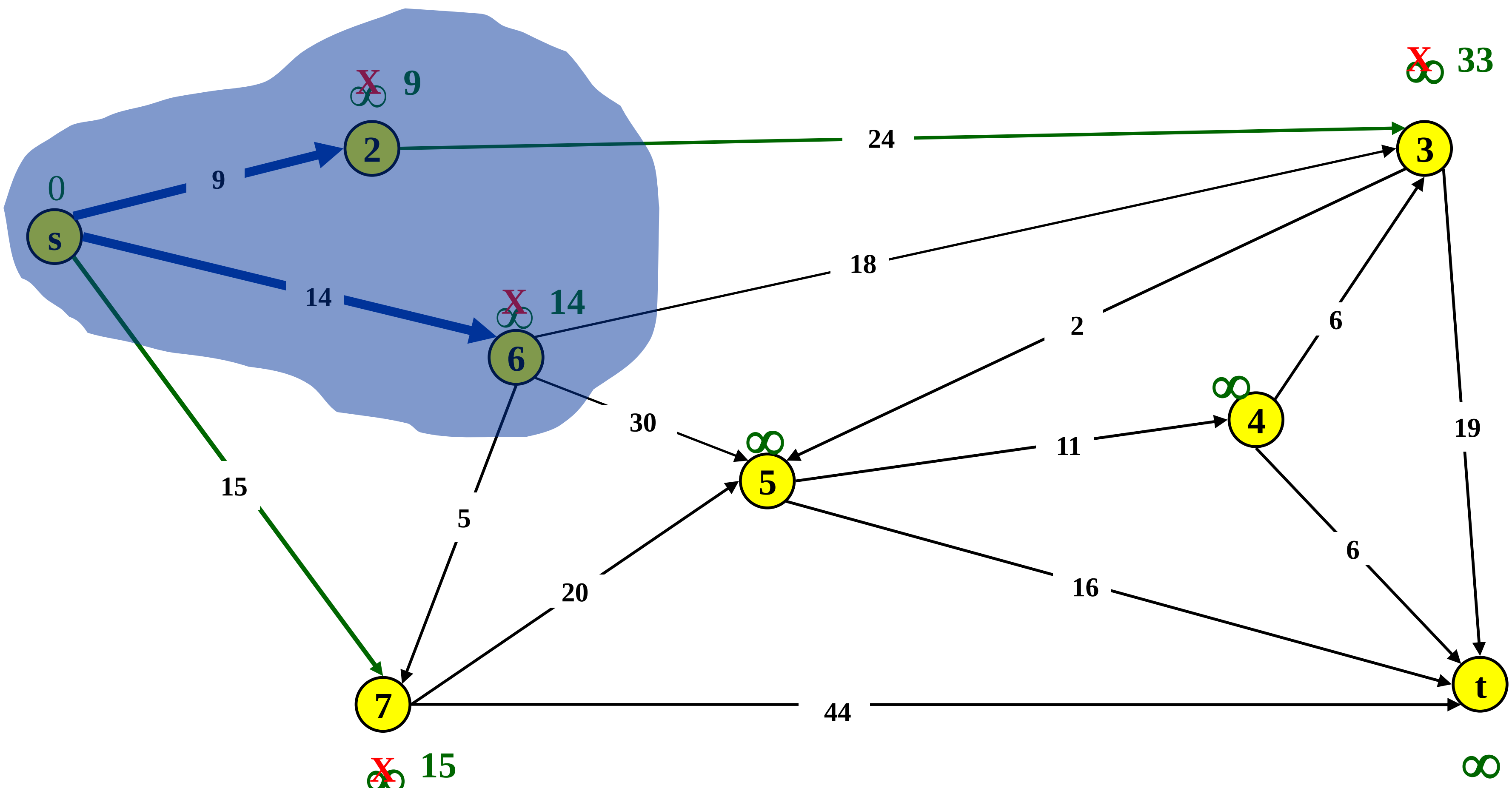
G−Gp = { 3, 4, 5, 6, 7, t }



D算法举例（之二）

Gp = { s, 2, 6 }

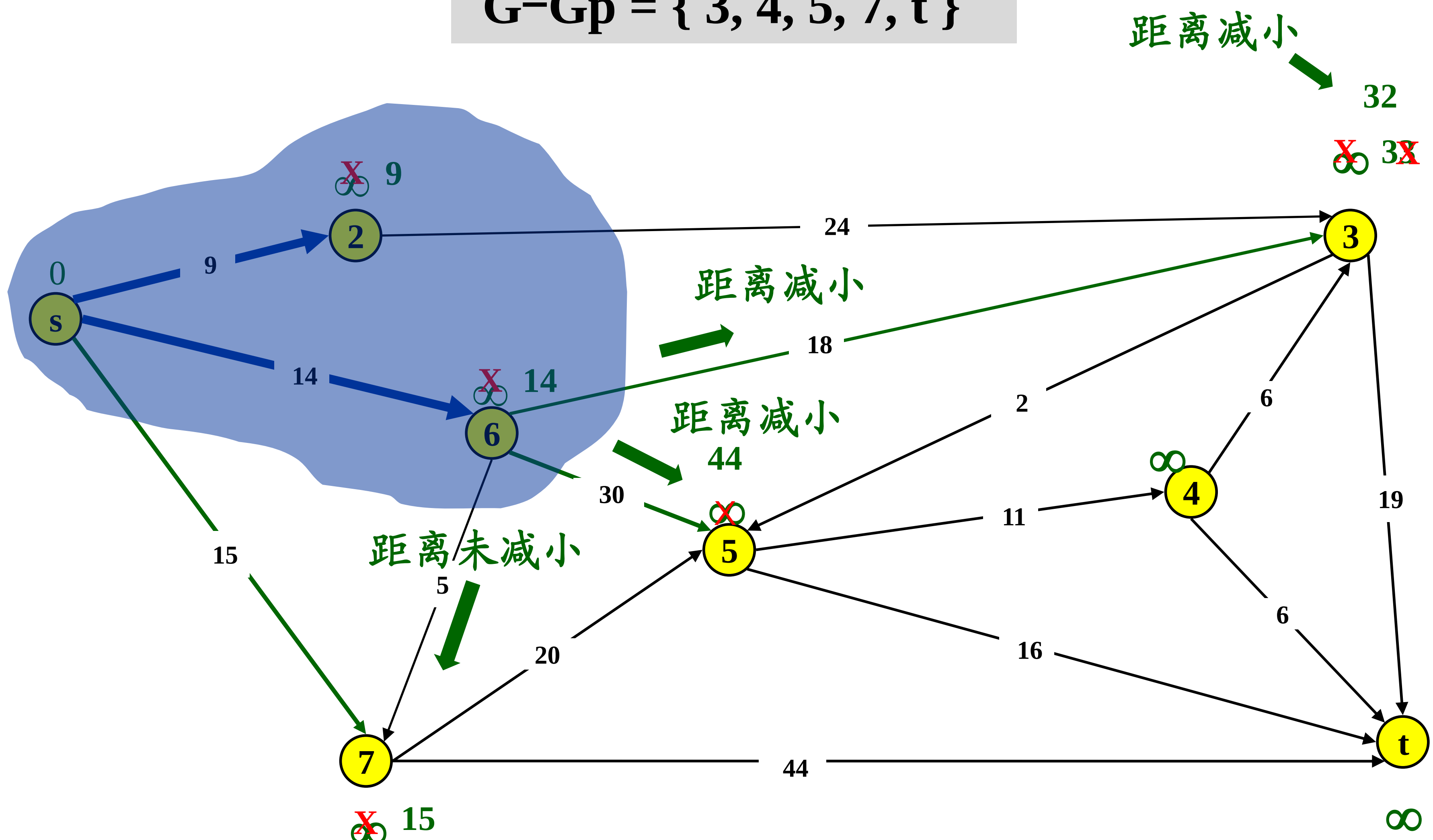
G−Gp = { 3, 4, 5, 7, t }



D算法举例（之二）

Gp = { s, 2, 6 }

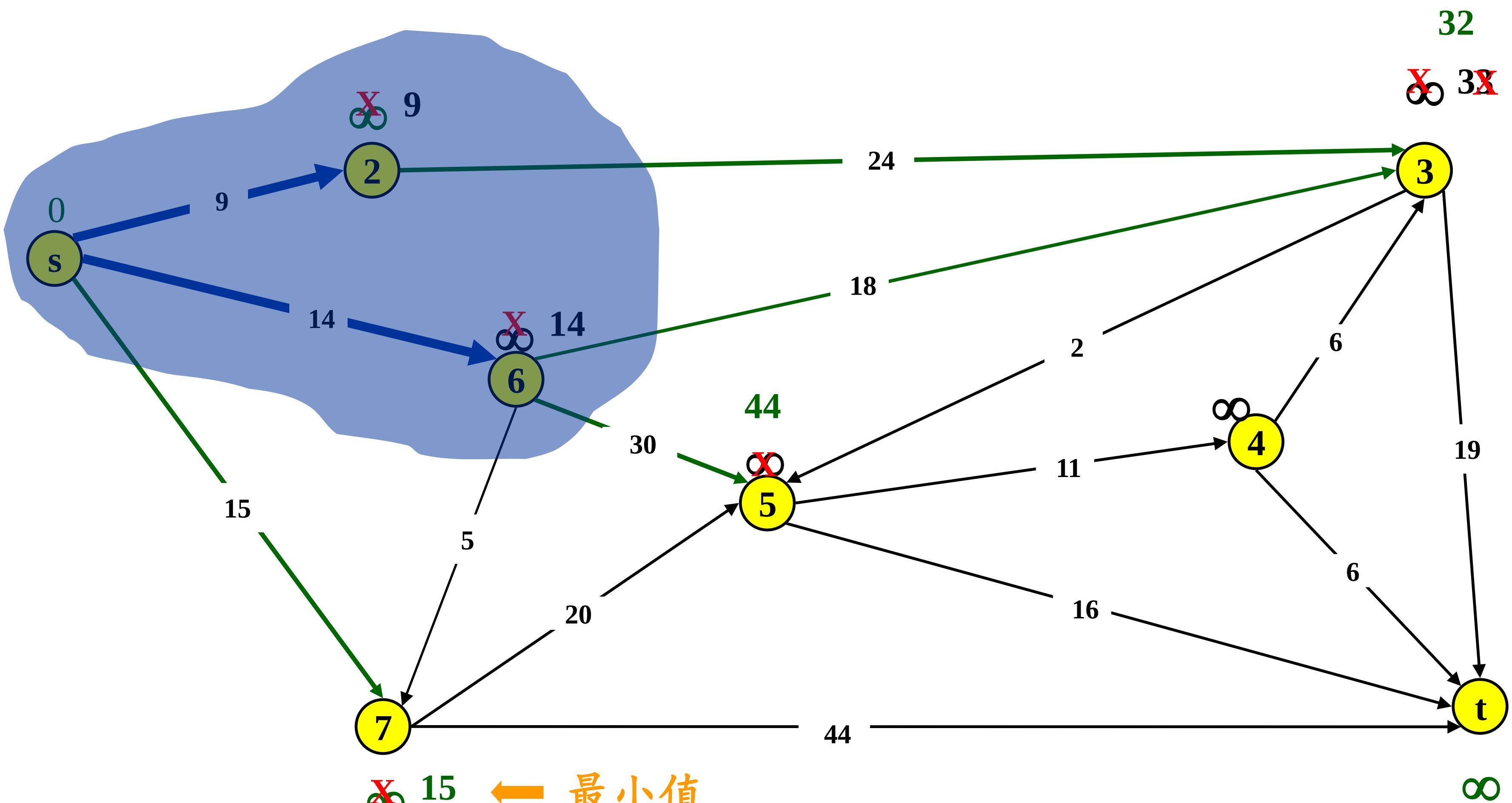
G−Gp = { 3, 4, 5, 7, t }



D算法举例（之二）

Gp = { s, 2, 6 }

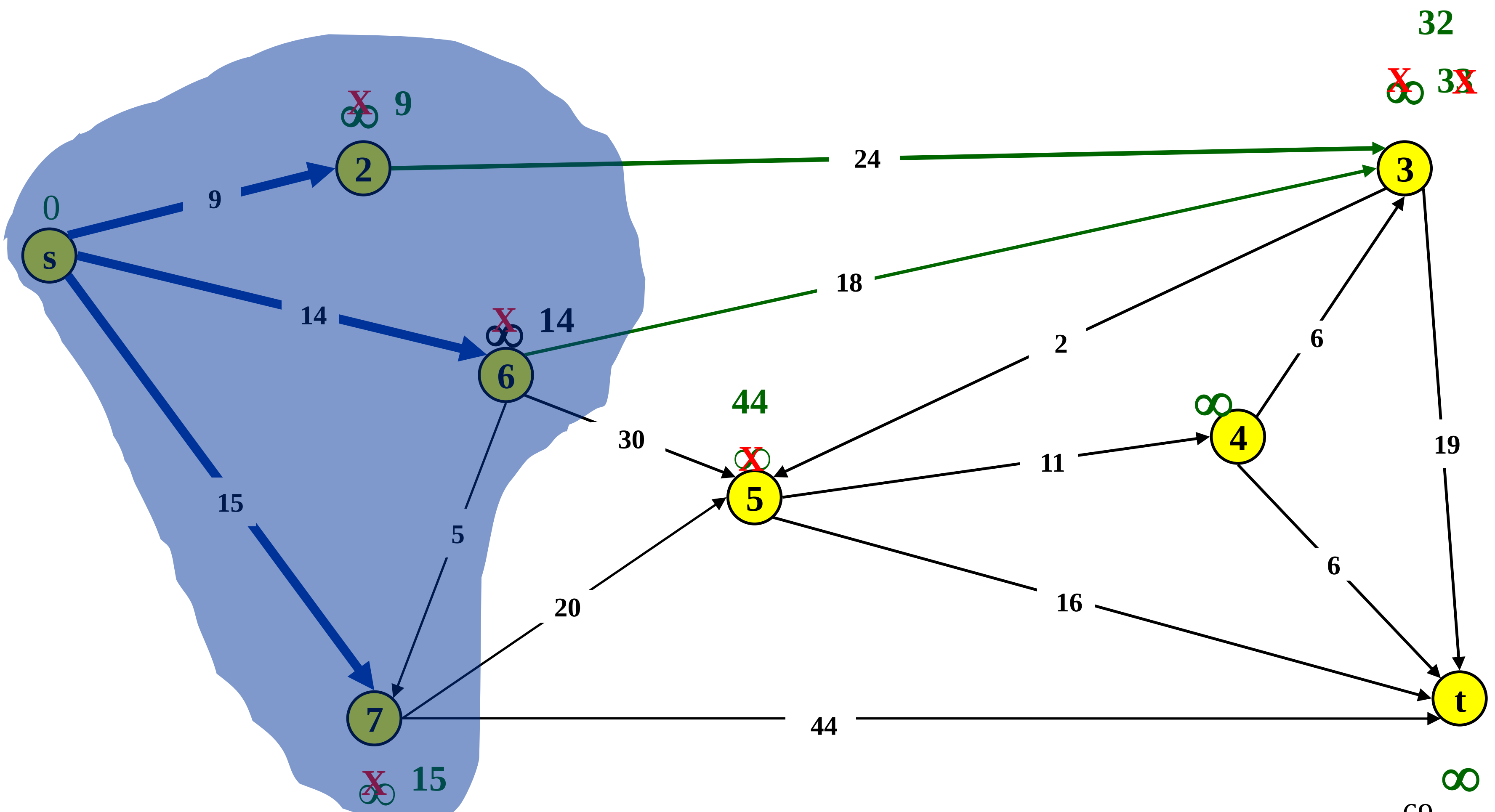
G−Gp = { 3, 4, 5, 7, t }



D算法举例（之二）

Gp = { s, 2, 6, 7 }

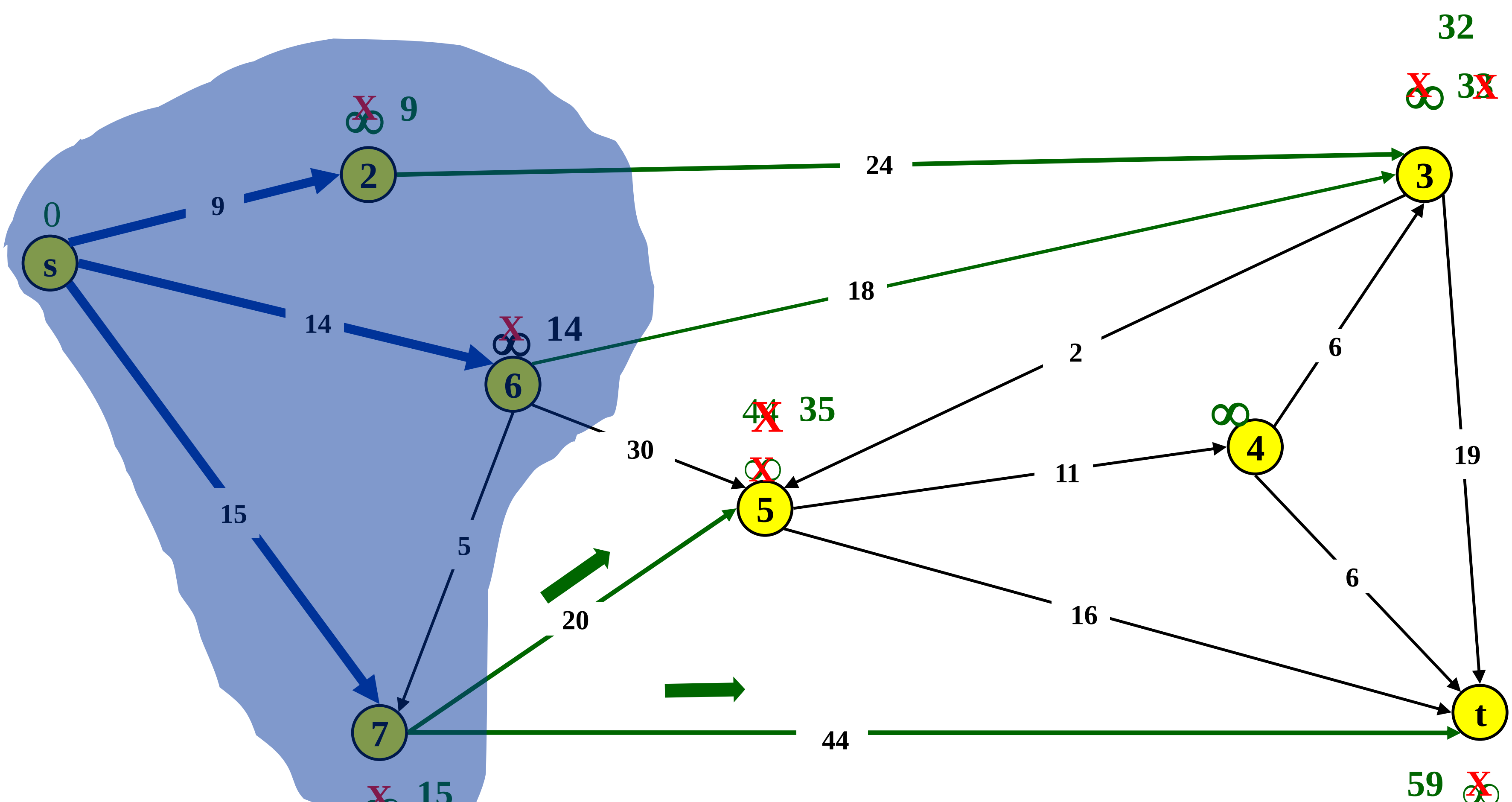
G−Gp = { 3, 4, 5, t }



D算法举例（之二）

Gp = { s, 2, 6, 7 }

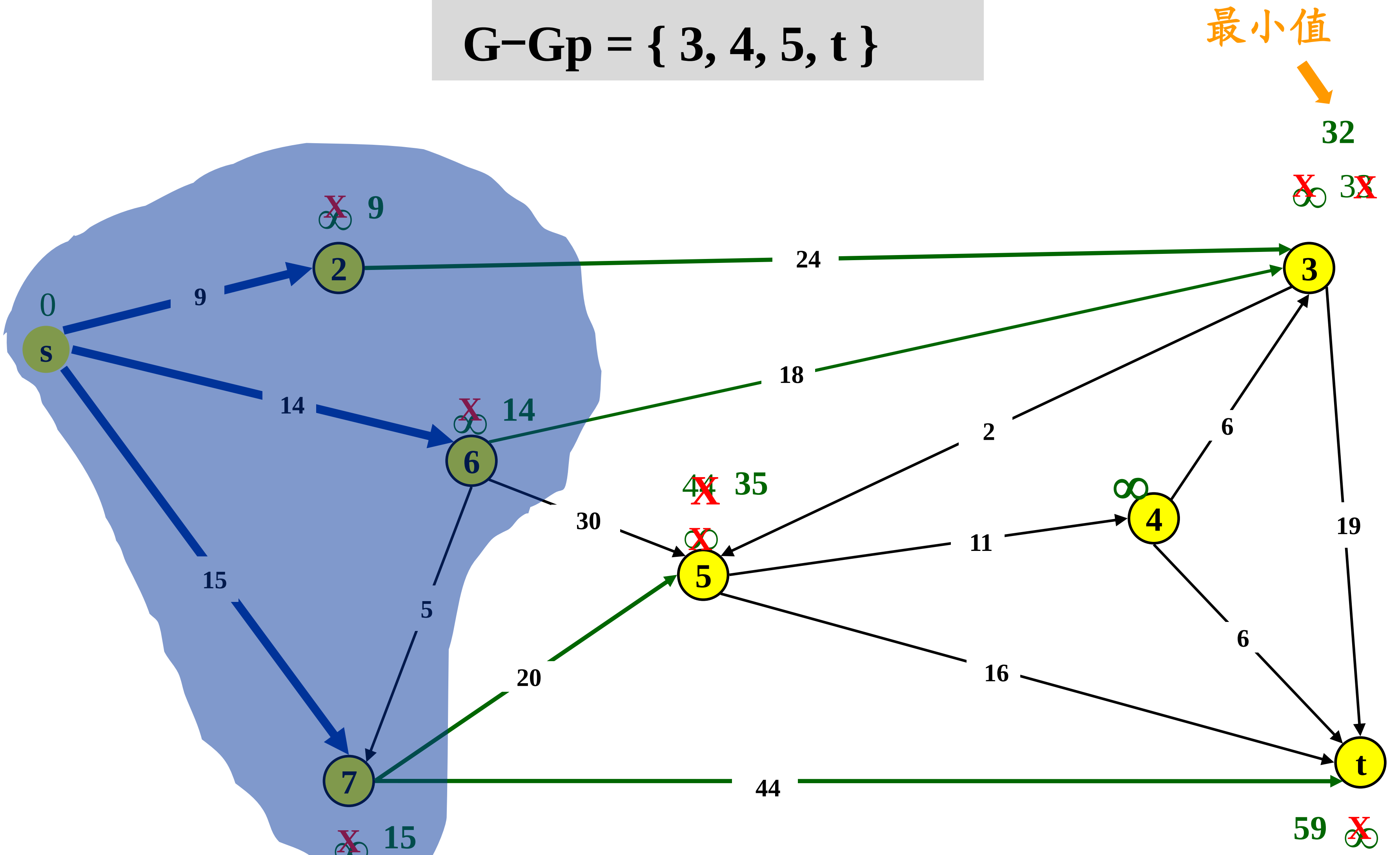
G−Gp = { 3, 4, 5, t }



D算法举例（之二）

Gp = { s, 2, 6, 7 }

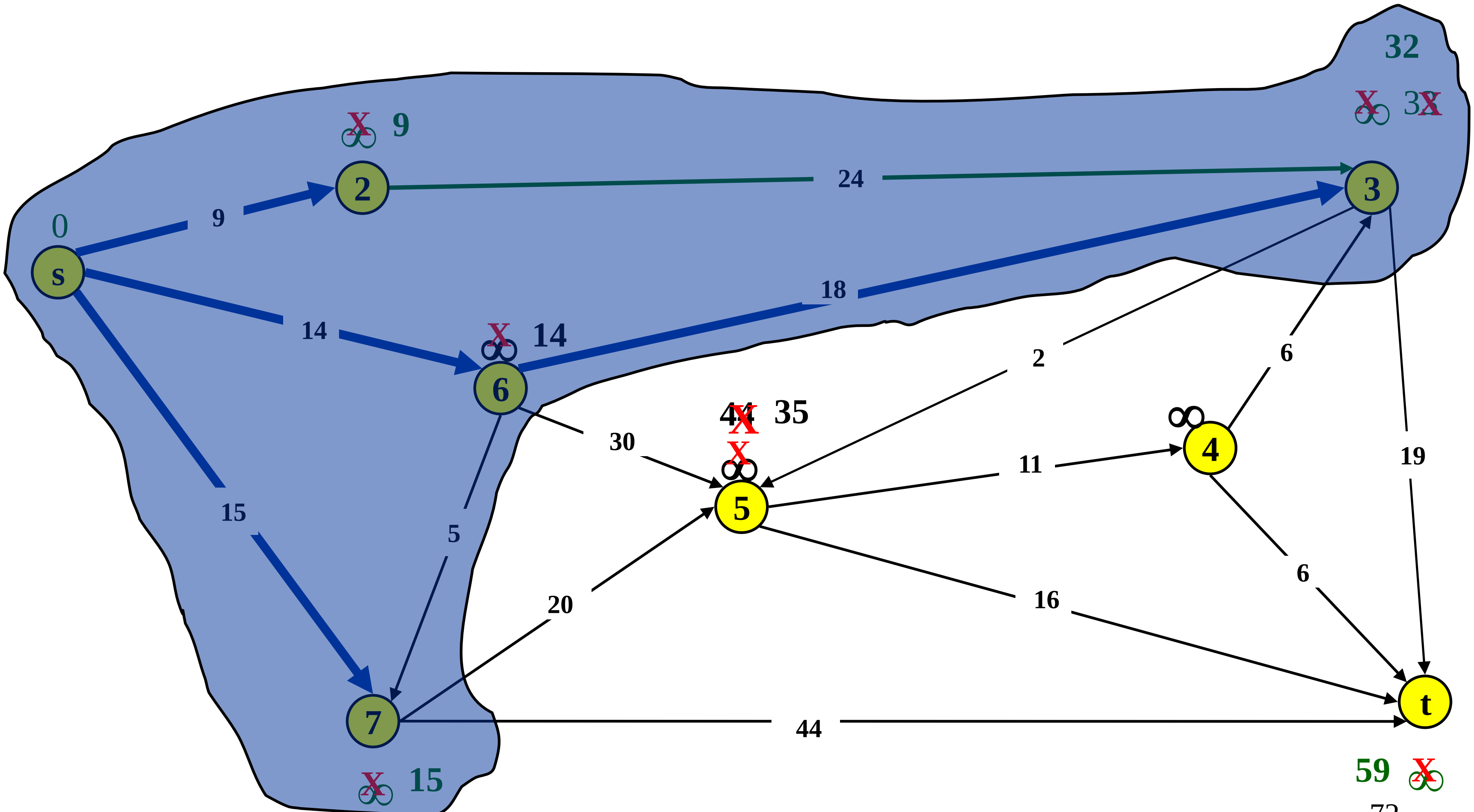
G−Gp = { 3, 4, 5, t }



D算法举例（之二）

Gp = { s, 2, 3, 6, 7 }

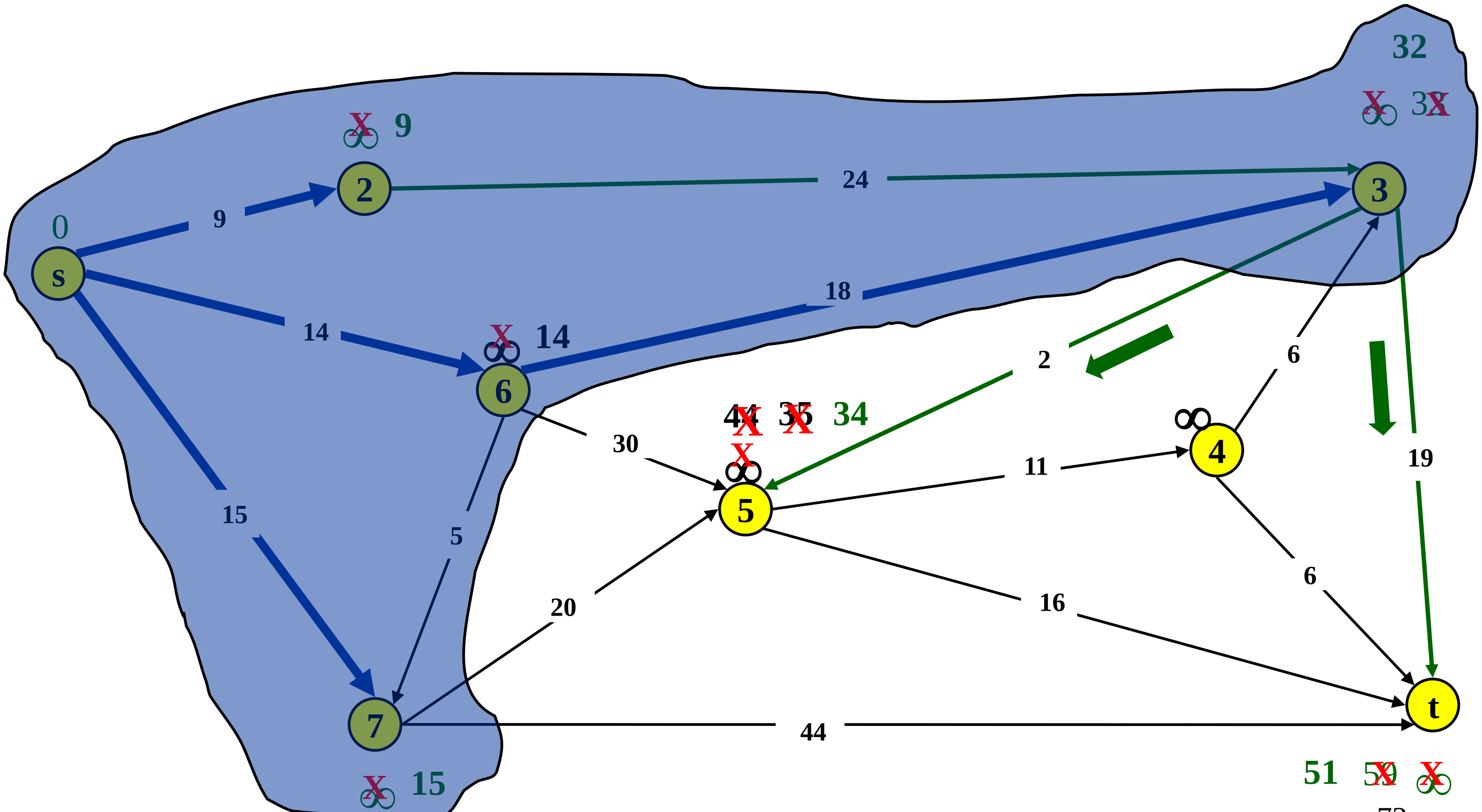
G−Gp = { 4, 5, t }



D算法举例（之二）

Gp = { s, 2, 3, 6, 7 }

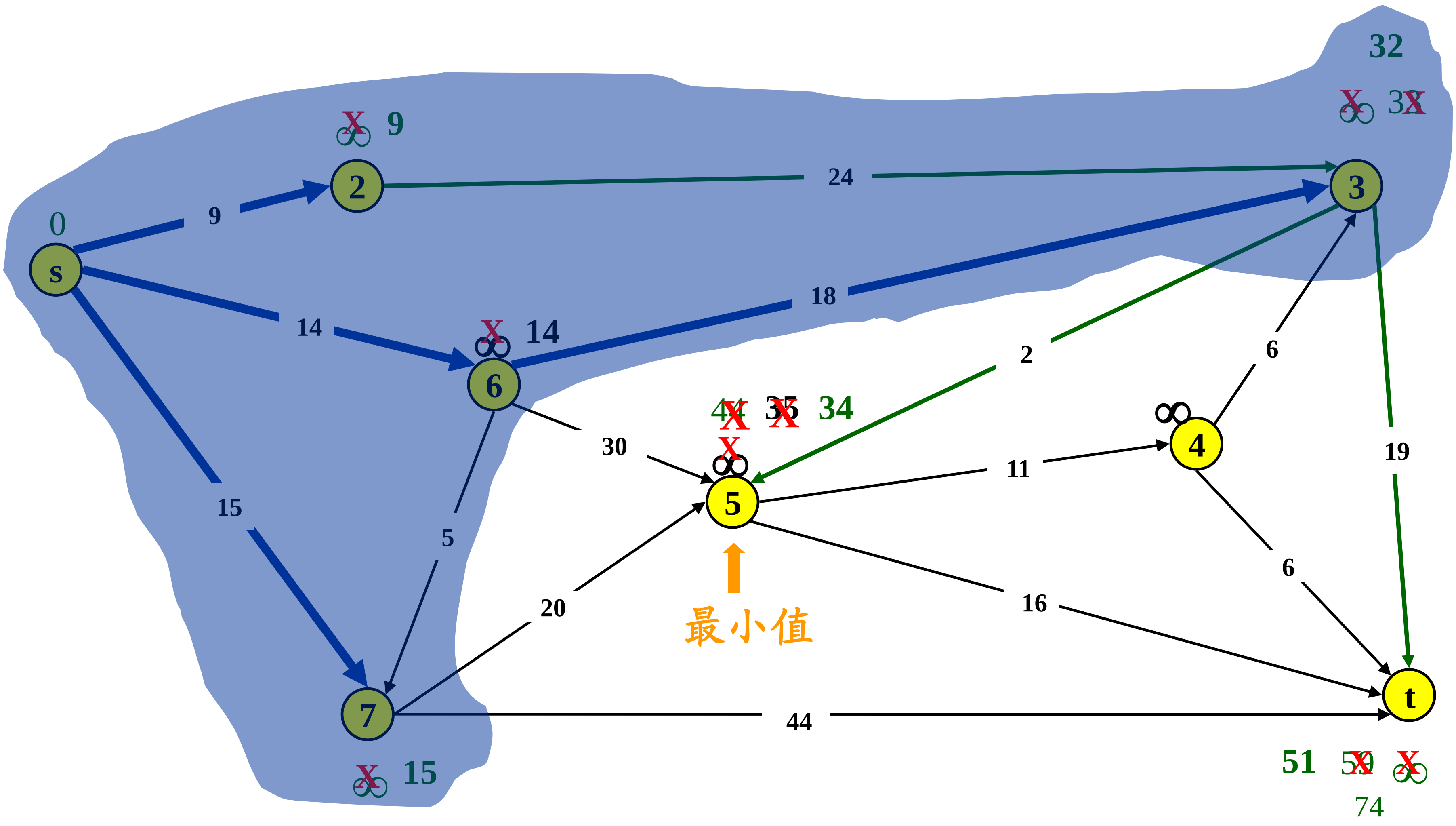
G−Gp = { 4, 5, t }



D算法举例（之二）

Gp = { s, 2, 3, 6, 7 }

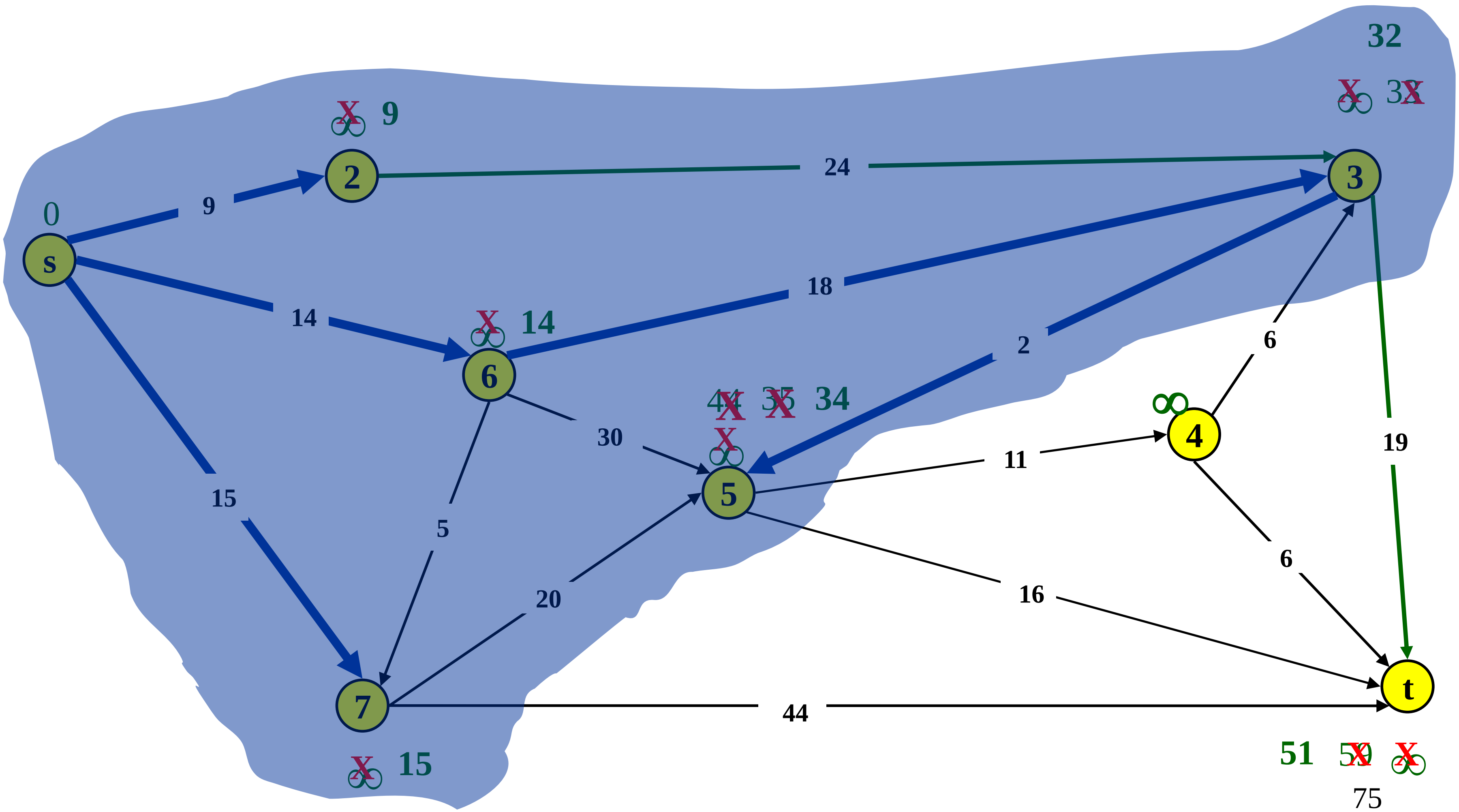
G−Gp = { 4, 5, t }



D算法举例（之二）

Gp = { s, 2, 3, 5, 6, 7 }

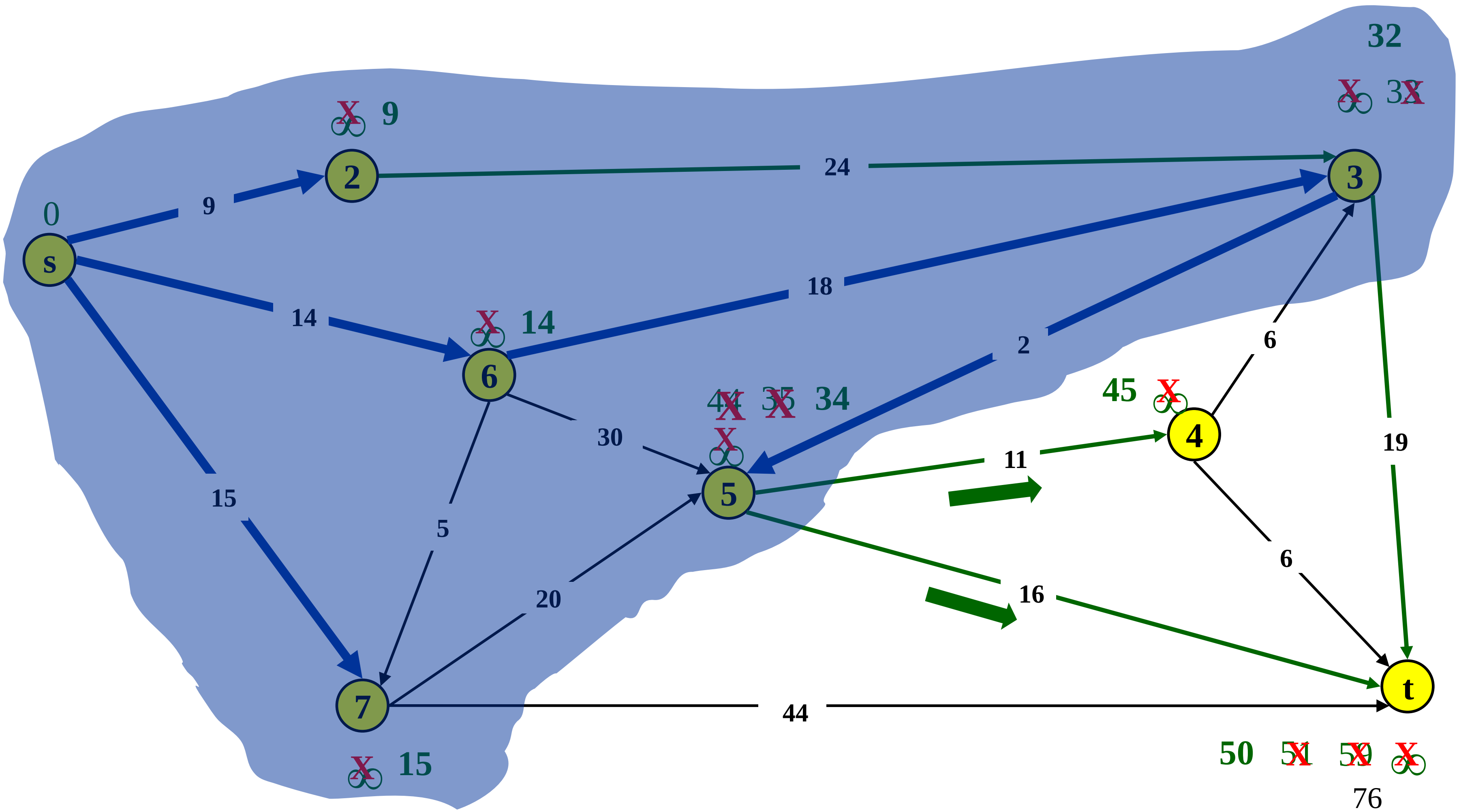
G−Gp = { 4, t }



D算法举例（之二）

Gp = { s, 2, 3, 5, 6, 7 }

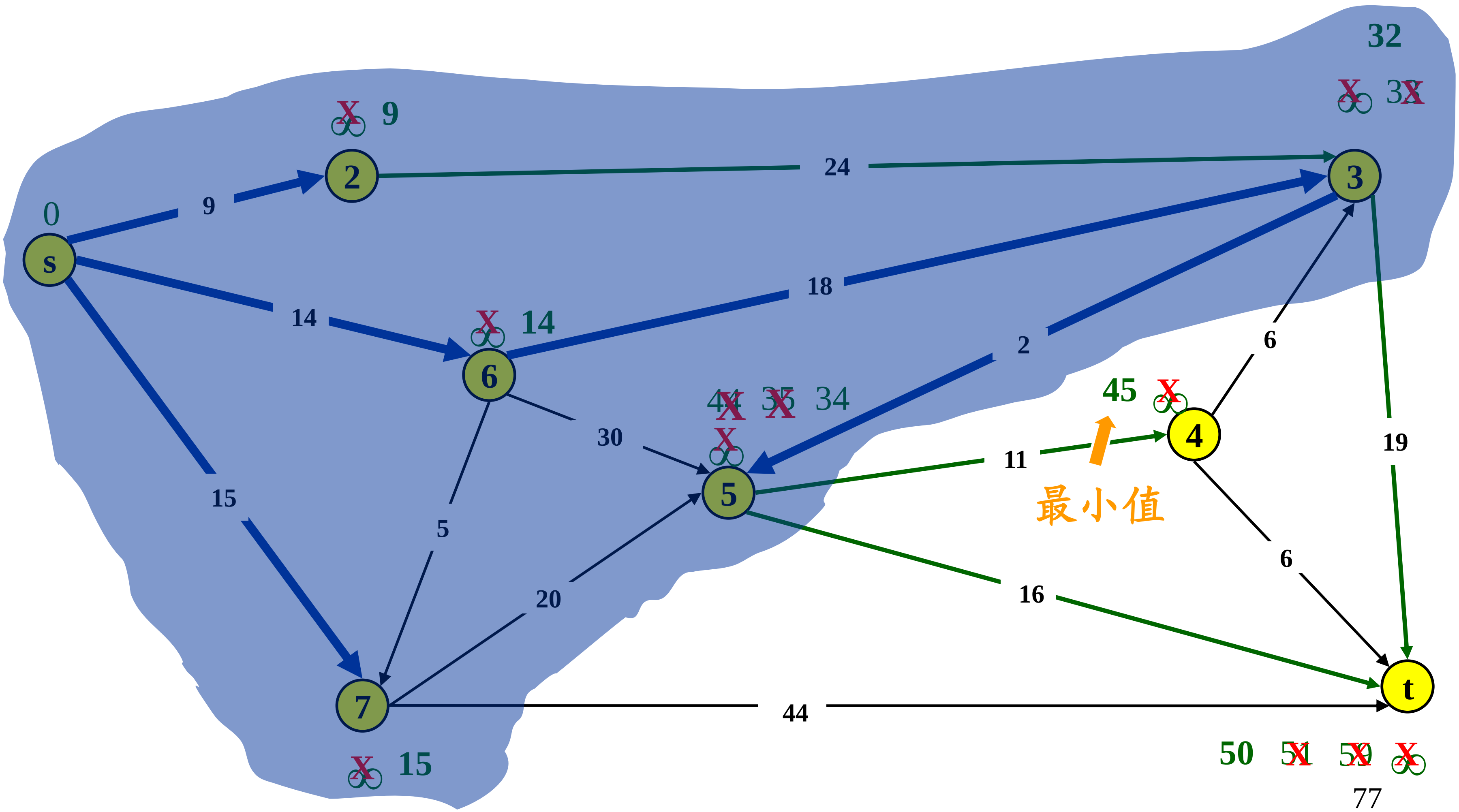
G−Gp = { 4, t }



D算法举例（之二）

Gp = { s, 2, 3, 5, 6, 7 }

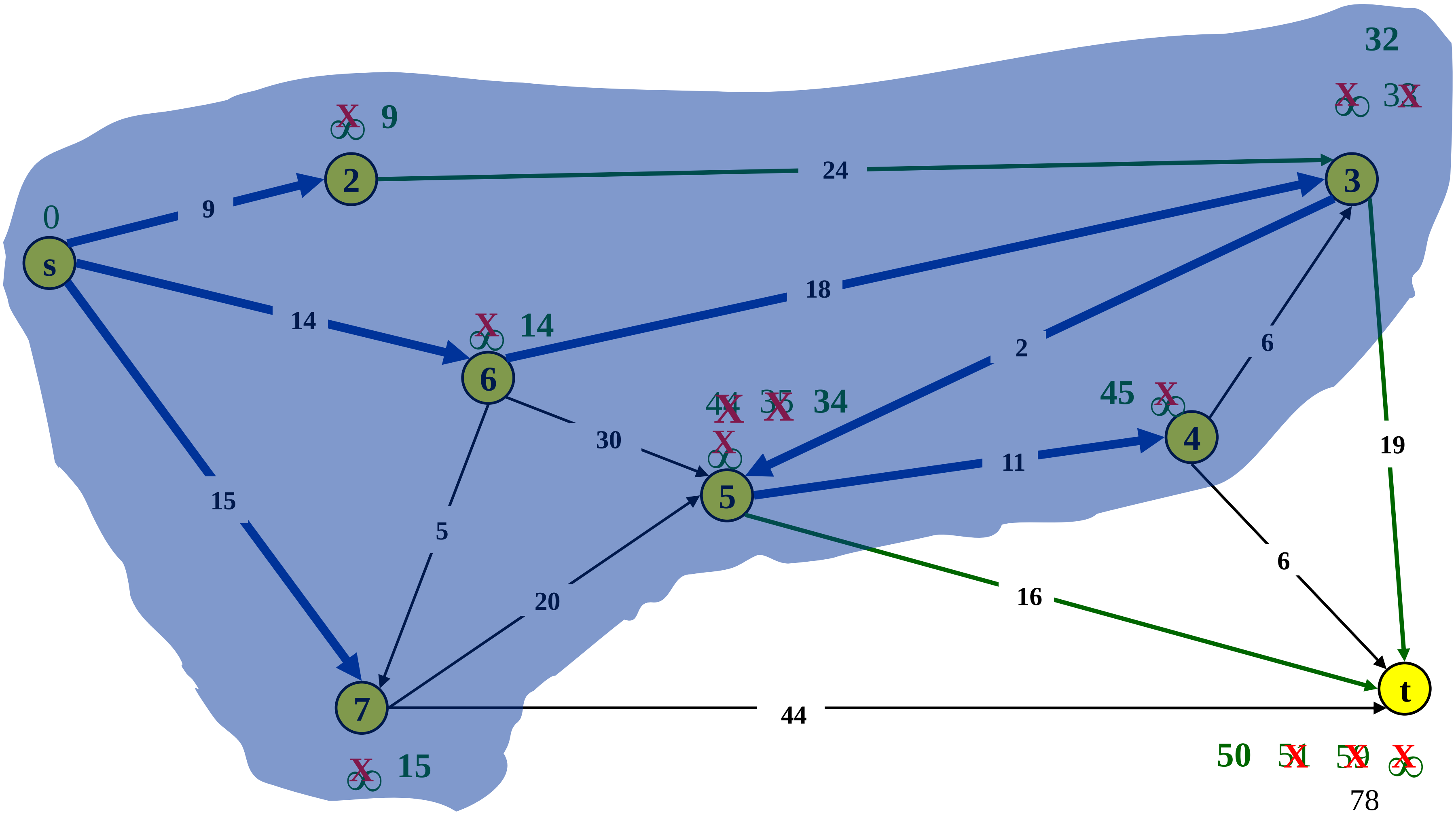
G−Gp = { 4, t }



D算法举例（之二）

Gp = { s, 2, 3, 4, 5, 6, 7 }

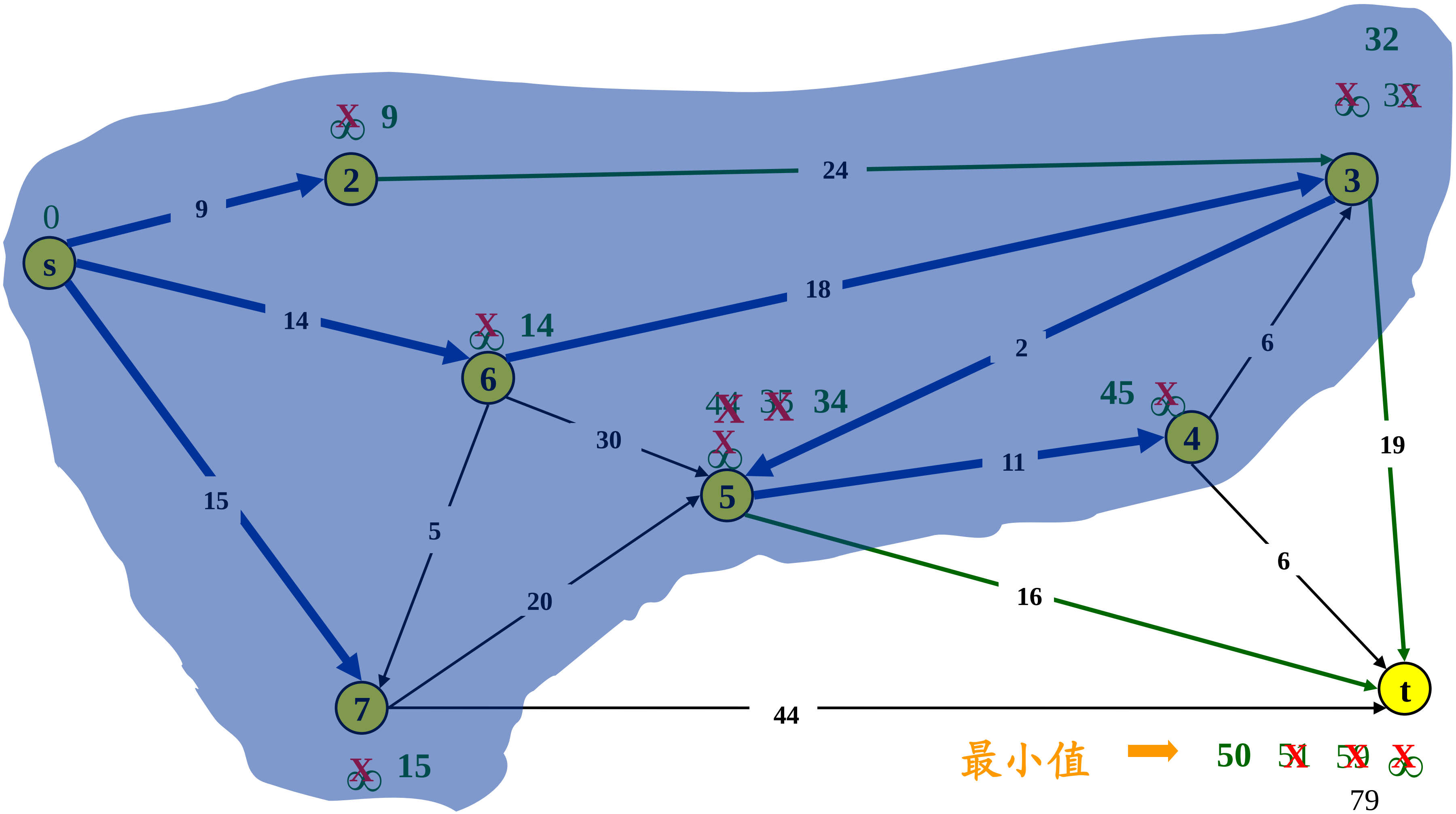
G−Gp = { t }



D算法举例（之二）

Gp = { s, 2, 3, 4, 5, 6, 7 }

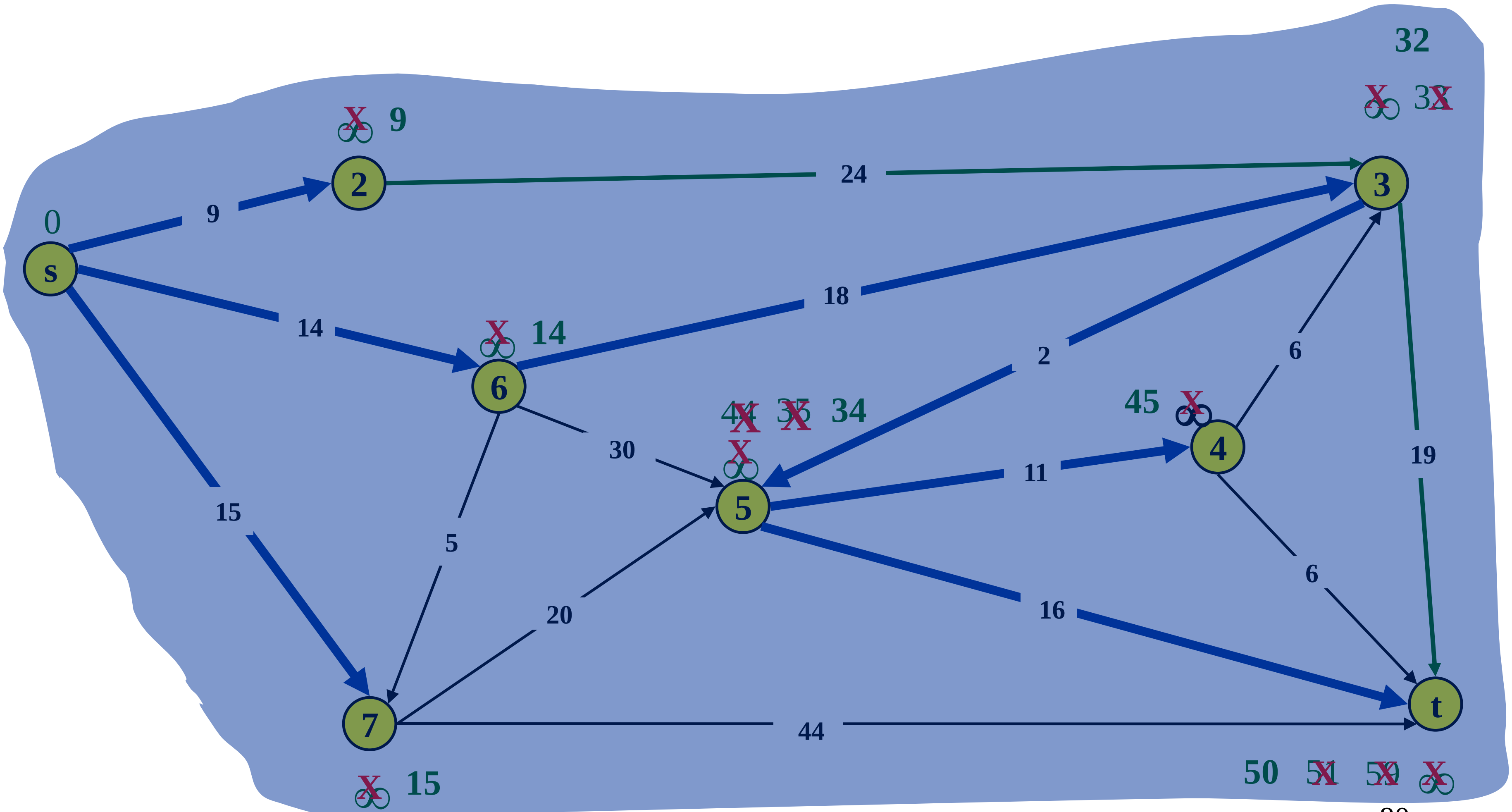
G−Gp = { t }



D算法举例（之二）

Gp = { s, 2, 3, 4, 5, 6, 7, t }

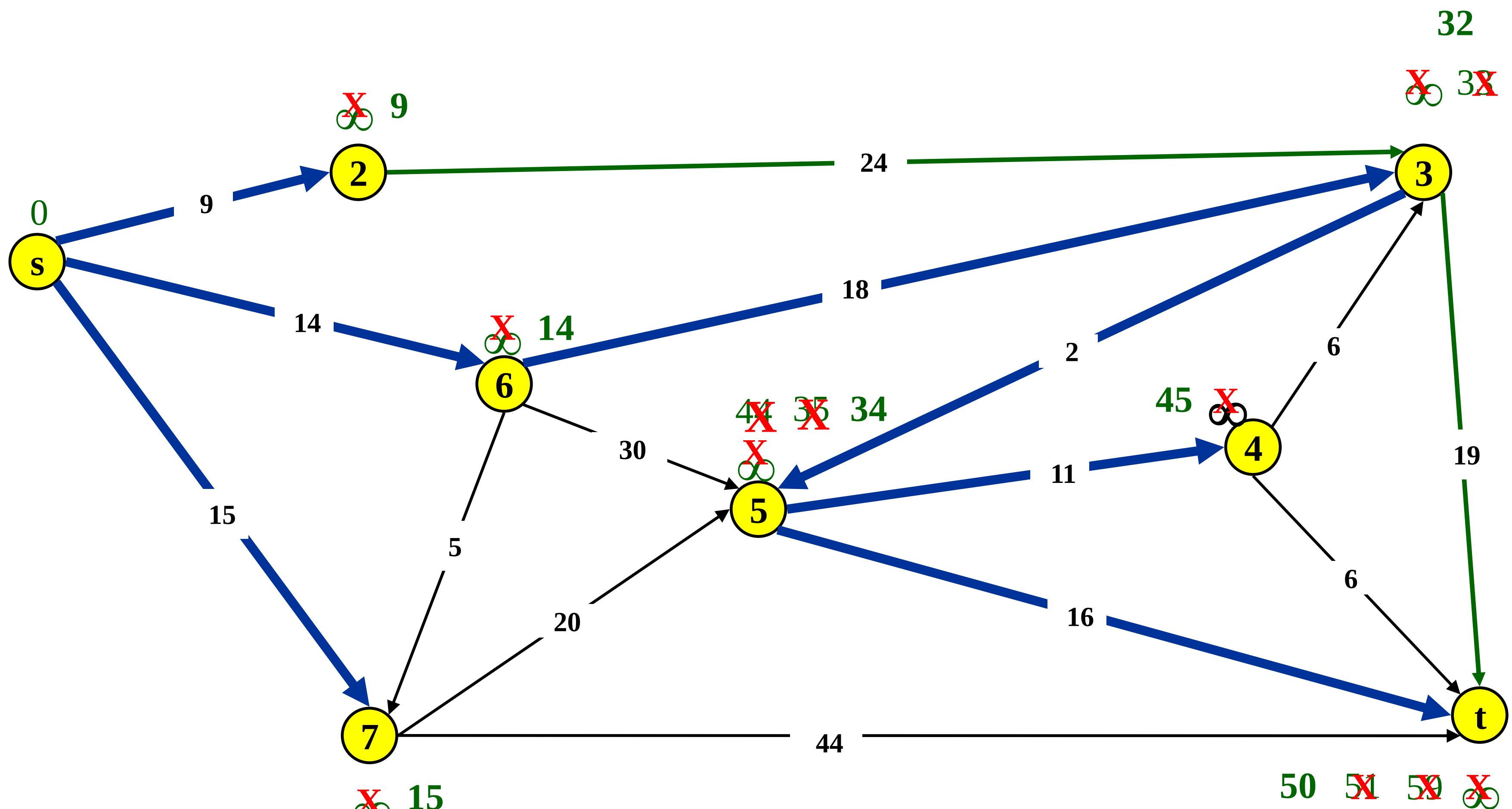
G−Gp = { }



D算法举例（之二）

Gp = { s, 2, 3, 4, 5, 6, 7, t }

G−Gp = { }



- 所有端间最短径的算法
  - 当然可以用前面介绍过的D算法作n次
    - 每次取一个不同的端作为指定端
  - 我们还有更为有效的算法
    - 即弗洛埃德算法（Floyd算法，简称F算法）
  - F算法的原理
    - 其原理与D算法一样
    - 只是用矩阵形式表达
    - 并进行系统化的计算
    - 对于n个端的图G，所有边长d_{ij}给定
      - n×n的W阵：径长矩阵，即端间距离矩阵
      - n×n的R阵：转接路由矩阵

= F算法的步骤

≡ F_0：初始化

Δ 置 $W^{(0)} = [w_{ij}^{(0)}]_{n\times n}$ 即直通阵

- $$w_{ij}^{(0)} = \begin{cases} d_{ij} & \text{若} v_i \text{与} v_j \text{之间有边} \\ \infty & \text{若} v_i \text{与} v_j \text{之间无边} \\ 0 & \text{若} i = j \end{cases}$$

Δ 置 $R^{(0)} = [r_{ij}^{(0)}]_{n\times n}$

- $$r_{ij}^{(0)} = \begin{cases} j & \text{若} w_{ij}^{(0)} < \infty \quad (\text{即} v_i \text{与} v_j \text{之间有边}) \\ 0 & \text{若} w_{ij}^{(0)} = \infty \text{或} i = j \quad (\text{即} v_i \text{与} v_j \text{之间无边}) \end{cases}$$

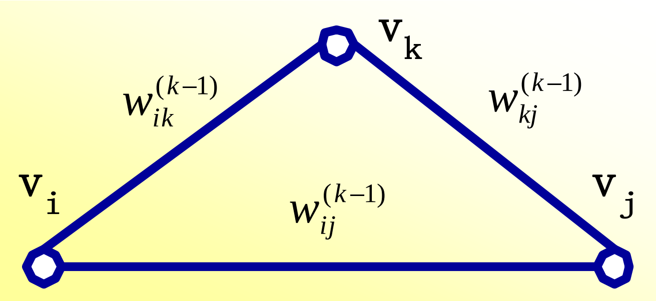
≡ 依次以 $v_1, v_2, v_3, \cdots, v_n$ 作转接，相应地

Δ 从 $W^{(0)} \to W^{(1)} \to W^{(2)} \to \cdots \to W^{(n)}$

Δ 从 $R^{(0)} \to R^{(1)} \to R^{(2)} \to \cdots \to R^{(n)}$

≡ F_1： 假设已得到$W^{(k-1)}$和$R^{(k-1)}$阵

- △ 欲求$W^{(k)}$和$R^{(k)}$
- △ $w_{ij}^{(k)} = \min\{w_{ij}^{(k-1)}, w_{ik}^{(k-1)} + w_{kj}^{(k-1)}\}$



- △ $$r_{ij}^{(k)} = \begin{cases} r_{ij}^{(k-1)} & \text{若} w_{ij}^{(k)} = w_{ij}^{(k-1)} \\ k & \text{若} w_{ij}^{(k)} < w_{ij}^{(k-1)} \end{cases}$$

≡ F_2:

- △ 若$k < n$，则返回F_1
- △ 若$k = n$，则终止

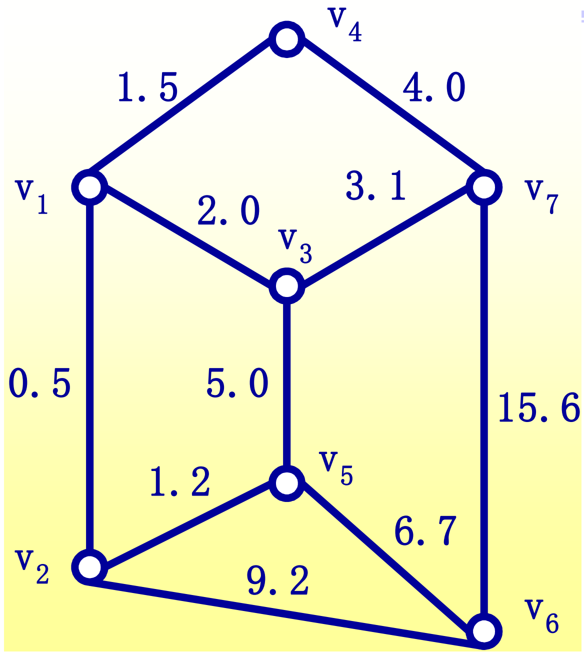
≡ 从以上步骤可以看出

- △ $W^{(k-1)} \rightarrow W^{(k)}$是计算经$v_k$转接时是否能缩短径长
  - 如能缩短，则更改w_{ij}，并在R阵中记录转接端
- △ 最后算得的$W^{(n)}$和$R^{(n)}$中，就是最短径长和转接路由

F算法举例

如图:

F_0: 初始化



$$w_{ij}^{(0)}=\begin{cases} d_{ij} & 若v_i与v_j之间有边 \\ \infty & 若v_i与v_j之间无边 \\ 0 & 若i=j \end{cases}$$

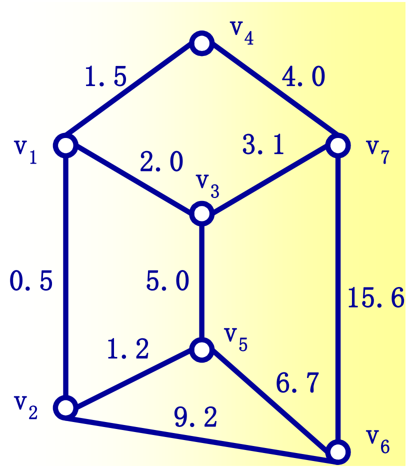
$$r_{ij}^{(0)}=\begin{cases} j & 若w_{ij}^{(0)}<\infty\ (即v_i与v_j之间有边) \\ 0 & 若w_{ij}^{(0)}=\infty或i=j\ (即v_i与v_j之间无边) \end{cases}$$

$$W^{(0)}=\begin{array}{c} \\ v_1\\v_2\\v_3\\v_4\\v_5\\v_6\\v_7 \end{array}\begin{array}{c} \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \begin{bmatrix} 0 & 0.5 & 2 & 1.5 & \infty & \infty & \infty \\ 0.5 & 0 & \infty & \infty & 1.2 & 9.2 & \infty \\ 2 & \infty & 0 & \infty & 5 & \infty & 3.1 \\ 1.5 & \infty & \infty & 0 & \infty & \infty & 4 \\ \infty & 1.2 & 5 & \infty & 0 & 6.7 & \infty \\ \infty & 9.2 & \infty & \infty & 6.7 & 0 & 15.6 \\ \infty & \infty & 3.1 & 4 & \infty & 15.6 & 0 \end{bmatrix} \end{array}$$

$$R^{(0)}=\begin{array}{c} \\ v_1\\v_2\\v_3\\v_4\\v_5\\v_6\\v_7 \end{array}\begin{array}{c} \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \begin{bmatrix} 0 & 2 & 3 & 4 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 5 & 6 & 0 \\ 1 & 0 & 0 & 0 & 5 & 0 & 7 \\ 1 & 0 & 0 & 0 & 0 & 0 & 7 \\ 0 & 2 & 3 & 0 & 0 & 6 & 0 \\ 0 & 2 & 0 & 0 & 5 & 0 & 7 \\ 0 & 0 & 3 & 4 & 0 & 6 & 0 \end{bmatrix} \end{array}$$

≡ F_1: k=1

$\Delta\ w_{ij}^{(1)} = \min\{w_{ij}^{(0)},\ w_{i1}^{(0)} + w_{1j}^{(0)}\}$



- 具体作法是:
- 任取一个元素，如$w_{23}^{(0)}$（除第k=1行和k=1列上的元素外）
- 第k=1行上找到所对应当元素2，即$w_{13}^{(0)}=2$
- 第k=1列上找到所对应当元素0.5，即$w_{21}^{(0)}=0.5$
- 将二者之和与$w_{23}^{(0)}$比较，取小者作为$w_{23}^{(1)}$,
- 所以，$w_{23}^{(1)}=2.5$

$$W^{(0)} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \begin{bmatrix} 0 & 0.5 & 2 & 1.5 & \infty & \infty & \infty \\ 0.5 & 0 & \boxed{\infty} & \boxed{\infty} & 1.2 & 9.2 & \infty \\ 2 & \boxed{\infty} & 0 & \boxed{\infty} & 5 & \infty & 3.1 \\ 1.5 & \boxed{\infty} & \boxed{\infty} & 0 & \infty & \infty & 4 \\ \infty & 1.2 & 5 & \infty & 0 & 6.7 & \infty \\ \infty & 9.2 & \infty & \infty & 6.7 & 0 & 15.6 \\ \infty & \infty & 3.1 & 4 & \infty & 15.6 & 0 \end{bmatrix} \end{array}$$

$$R^{(0)} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \begin{bmatrix} 0 & 2 & 3 & 4 & 0 & 0 & 0 \\ 1 & 0 & 0\color{red}{1} & 0\color{red}{1} & 5 & 6 & 0 \\ 1 & 0\color{red}{1} & 0 & 0\color{red}{1} & 5 & 0 & 7 \\ 1 & 0\color{red}{1} & 0\color{red}{1} & 0 & 0 & 0 & 7 \\ 0 & 2 & 3 & 0 & 0 & 6 & 0 \\ 0 & 2 & 0 & 0 & 5 & 0 & 7 \\ 0 & 0 & 3 & 4 & 0 & 6 & 0 \end{bmatrix} \end{array}$$

- 遍历所有元素后（除第k行和第k列上的元素外），得到:

$$W^{(1)}=\begin{array}{c|ccccccc} & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \\ \hline v_1 & 0 & 0.5 & 2 & 1.5 & \infty & \infty & \infty \\ v_2 & 0.5 & 0 & \boxed{2.5} & \boxed{2} & 1.2 & 9.2 & \infty \\ v_3 & 2 & \boxed{2.5} & 0 & \boxed{3.5} & 5 & \infty & 3.1 \\ v_4 & 1.5 & \boxed{2} & \boxed{3.5} & 0 & \infty & \infty & 4 \\ v_5 & \infty & 1.2 & 5 & \infty & 0 & 6.7 & \infty \\ v_6 & \infty & 9.2 & \infty & \infty & 6.7 & 0 & 15.6 \\ v_7 & \infty & \infty & 3.1 & 4 & \infty & 15.6 & 0 \end{array}$$

$$R^{(1)}=\begin{array}{c|ccccccc} & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \\ \hline v_1 & 0 & 2 & 3 & 4 & 0 & 0 & 0 \\ v_2 & 1 & 0 & \boxed{1} & \boxed{1} & 5 & 6 & 0 \\ v_3 & 1 & \boxed{1} & 0 & \boxed{1} & 5 & 0 & 7 \\ v_4 & 1 & \boxed{1} & \boxed{1} & 0 & 0 & 0 & 7 \\ v_5 & 0 & 2 & 3 & 0 & 0 & 6 & 0 \\ v_6 & 0 & 2 & 0 & 0 & 5 & 0 & 7 \\ v_7 & 0 & 0 & 3 & 4 & 0 & 6 & 0 \end{array}$$

- **规律:**
- 第k行中，∞项所对应的列都不会发生变化
- 第k列中，∞项所对应的行都不会发生变化

$$\Delta \qquad r_{ij}^{(1)}=\begin{cases} k=1 & \text{若}w_{ij}^{(1)}\text{与}w_{ij}^{(0)}\text{相比，发生了变化} \\ r_{ij}^{(0)} & \text{若}w_{ij}^{(1)}\text{与}w_{ij}^{(0)}\text{相比，未发生变化} \end{cases}$$

- 于是可得$R^{(1)}$

≡ F_2： k<n 未结束，返回F_1

≡ F_1: k=2

△ 在$W^{(1)}$中任取元素$w_{ij}^{(1)}$（除第k=2行和第k=2列上的元素以外）

- 找到它在第k = 2行上所对应的元素$w_{2j}^{(1)}$
- 找到它在第k = 2列上所对应的元素$w_{i2}^{(1)}$
- 将二者之和与$w_{ij}^{(1)}$相比较，取小者作为$w_{ij}^{(2)}$

△ 遍历所有元素之后（除第k=2行和第k=2列上的元素以外）可得:

$$W^{(1)}=\begin{array}{cc} & \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{matrix} \\ \begin{matrix} v_1\\ v_2\\ v_3\\ v_4\\ v_5\\ v_6\\ v_7 \end{matrix} & \begin{bmatrix} 0 & 0.5 & 2 & 1.5 & \boxed{\infty} & \boxed{\infty} & \infty \\ 0.5 & 0 & 2.5 & 2 & 1.2 & 9.2 & \infty \\ 2 & 2.5 & 0 & 3.5 & \boxed{5} & \boxed{\infty} & 3.1 \\ 1.5 & 2 & 3.5 & 0 & \boxed{\infty} & \boxed{\infty} & 4 \\ \boxed{\infty} & 1.2 & \boxed{5} & \boxed{\infty} & 0 & 6.7 & \infty \\ \boxed{\infty} & 9.2 & \boxed{\infty} & \boxed{\infty} & 6.7 & 0 & 15.6 \\ \infty & \infty & 3.1 & 4 & \infty & 15.6 & 0 \end{bmatrix} \end{array}$$

$$R^{(1)}=\begin{array}{cc} & \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{matrix} \\ \begin{matrix} v_1\\ v_2\\ v_3\\ v_4\\ v_5\\ v_6\\ v_7 \end{matrix} & \begin{bmatrix} 0 & 2 & 3 & 4 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 5 & 6 & 0 \\ 1 & 1 & 0 & 1 & 5 & 0 & 7 \\ 1 & 1 & 1 & 0 & 0 & 0 & 7 \\ 0 & 2 & 3 & 0 & 0 & 6 & 0 \\ 0 & 2 & 0 & 0 & 5 & 0 & 7 \\ 0 & 0 & 3 & 4 & 0 & 6 & 0 \end{bmatrix} \end{array}$$

$$W^{(2)}=\begin{array}{cc} & \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{matrix} \\ \begin{matrix} v_1\\ v_2\\ v_3\\ v_4\\ v_5\\ v_6\\ v_7 \end{matrix} & \begin{bmatrix} 0 & 0.5 & 2 & 1.5 & \boxed{1.7} & \boxed{9.7} & \infty \\ 0.5 & 0 & 2.5 & 2 & 1.2 & 9.2 & \infty \\ 2 & 2.5 & 0 & 3.5 & \boxed{3.7} & \boxed{11.7} & 3.1 \\ 1.5 & 2 & 3.5 & 0 & \boxed{3.2} & \boxed{11.2} & 4 \\ \boxed{1.7} & 1.2 & \boxed{3.7} & \boxed{3.2} & 0 & 6.7 & \infty \\ \boxed{9.7} & 9.2 & \boxed{11.7} & \boxed{11.2} & 6.7 & 0 & 15.6 \\ \infty & \infty & 3.1 & 4 & \infty & 15.6 & 0 \end{bmatrix} \end{array}$$

$$R^{(2)}=\begin{array}{cc} & \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{matrix} \\ \begin{matrix} v_1\\ v_2\\ v_3\\ v_4\\ v_5\\ v_6\\ v_7 \end{matrix} & \begin{bmatrix} 0 & 2 & 3 & 4 & \boxed{2} & \boxed{2} & 0 \\ 1 & 0 & 1 & 1 & 5 & 6 & 0 \\ 1 & 1 & 0 & 1 & \boxed{2} & \boxed{2} & 7 \\ 1 & 1 & 1 & 0 & \boxed{2} & \boxed{2} & 7 \\ \boxed{2} & 2 & \boxed{2} & \boxed{2} & 0 & 6 & 0 \\ \boxed{2} & 2 & \boxed{2} & \boxed{2} & 5 & 0 & 7 \\ 0 & 0 & 3 & 4 & 0 & 6 & 0 \end{bmatrix} \end{array}$$

≡ F_2: k<n未结束，返回F_1

≡ F_1： k=3

$$W^{(2)} = \begin{array}{cc} & \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array} & \begin{bmatrix} 0 & 0.5 & 2 & 1.5 & 1.7 & 9.7 & \boxed{\infty} \\ 0.5 & 0 & 2.5 & 2 & 1.2 & 9.2 & \boxed{\infty} \\ 2 & 2.5 & 0 & 3.5 & 3.7 & 11.7 & 3.1 \\ 1.5 & 2 & 3.5 & 0 & 3.2 & 11.2 & 4 \\ 1.7 & 1.2 & 3.7 & 3.2 & 0 & 6.7 & \boxed{\infty} \\ 9.7 & 9.2 & 11.7 & 11.2 & 6.7 & 0 & \boxed{15.6} \\ \boxed{\infty} & \boxed{\infty} & 3.1 & 4 & \boxed{\infty} & \boxed{15.6} & 0 \end{bmatrix} \end{array}$$

$$R^{(2)} = \begin{array}{cc} & \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array} & \begin{bmatrix} 0 & 2 & 3 & 4 & 2 & 2 & 0 \\ 1 & 0 & 1 & 1 & 5 & 6 & 0 \\ 1 & 1 & 0 & 1 & 2 & 2 & 7 \\ 1 & 1 & 1 & 0 & 2 & 2 & 7 \\ 2 & 2 & 2 & 2 & 0 & 6 & 0 \\ 2 & 2 & 2 & 2 & 5 & 0 & 7 \\ 0 & 0 & 3 & 4 & 0 & 6 & 0 \end{bmatrix} \end{array}$$

$$W^{(3)} = \begin{array}{cc} & \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array} & \begin{bmatrix} 0 & 0.5 & 2 & 1.5 & 1.7 & 9.7 & \boxed{5.1} \\ 0.5 & 0 & 2.5 & 2 & 1.2 & 9.2 & \boxed{5.6} \\ 2 & 2.5 & 0 & 3.5 & 3.7 & 11.7 & 3.1 \\ 1.5 & 2 & 3.5 & 0 & 3.2 & 11.2 & 4 \\ 1.7 & 1.2 & 3.7 & 3.2 & 0 & 6.7 & \boxed{6.8} \\ 9.7 & 9.2 & 11.7 & 11.2 & 6.7 & 0 & \boxed{14.8} \\ \boxed{5.1} & \boxed{5.6} & 3.1 & 4 & \boxed{6.8} & \boxed{14.8} & 0 \end{bmatrix} \end{array}$$

$$R^{(3)} = \begin{array}{cc} & \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array} & \begin{bmatrix} 0 & 2 & 3 & 4 & 2 & 2 & \boxed{3} \\ 1 & 0 & 1 & 1 & 5 & 6 & \boxed{3} \\ 1 & 1 & 0 & 1 & 2 & 2 & 7 \\ 1 & 1 & 1 & 0 & 2 & 2 & 7 \\ 2 & 2 & 2 & 2 & 0 & 6 & \boxed{3} \\ 2 & 2 & 2 & 2 & 5 & 0 & \boxed{3} \\ \boxed{3} & \boxed{3} & 3 & 4 & \boxed{3} & \boxed{3} & 0 \end{bmatrix} \end{array}$$

≡ F_2： k<n未结束，返回F_1

≡ **F_1: k=4**

$$
W^{(3)} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array}
\begin{array}{c}
\begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\
\begin{bmatrix}
0 & 0.5 & 2 & 1.5 & 1.7 & 9.7 & 5.1 \\
0.5 & 0 & 2.5 & 2 & 1.2 & 9.2 & 5.6 \\
2 & 2.5 & 0 & 3.5 & 3.7 & 11.7 & 3.1 \\
1.5 & 2 & 3.5 & 0 & 3.2 & 11.2 & 4 \\
1.7 & 1.2 & 3.7 & 3.2 & 0 & 6.7 & 6.8 \\
9.7 & 9.2 & 11.7 & 11.2 & 6.7 & 0 & 14.8 \\
5.1 & 5.6 & 3.1 & 4 & 6.8 & 14.8 & 0
\end{bmatrix}
\end{array}
\qquad
R^{(3)} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array}
\begin{array}{c}
\begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\
\begin{bmatrix}
0 & 2 & 3 & 4 & 2 & 2 & 3 \\
1 & 0 & 1 & 1 & 5 & 6 & 3 \\
1 & 1 & 0 & 1 & 2 & 2 & 7 \\
1 & 1 & 1 & 0 & 2 & 2 & 7 \\
2 & 2 & 2 & 2 & 0 & 6 & 3 \\
2 & 2 & 2 & 2 & 5 & 0 & 3 \\
3 & 3 & 3 & 4 & 3 & 3 & 0
\end{bmatrix}
\end{array}
$$

Δ **$W^{(4)} = W^{(3)}$， $R^{(4)} = R^{(3)}$**

≡ **F_2: k<n未结束，返回F_1**

≡ F_1: k=5

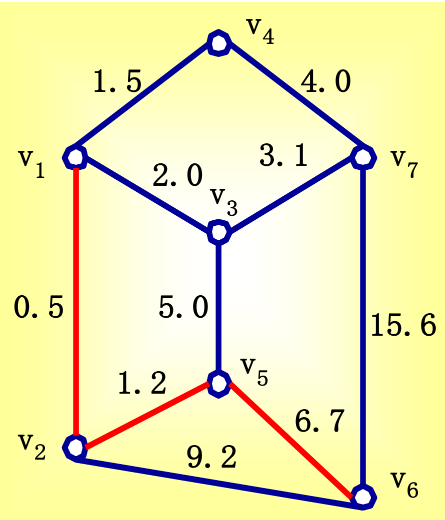
$$
W^{(4)} = \begin{array}{cc}
 & \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{matrix} \\
\begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{matrix} &
\begin{bmatrix}
0 & 0.5 & 2 & 1.5 & 1.7 & \boxed{9.7} & 5.1 \\
0.5 & 0 & 2.5 & 2 & 1.2 & \boxed{9.2} & 5.6 \\
2 & 2.5 & 0 & 3.5 & 3.7 & \boxed{11.7} & 3.1 \\
1.5 & 2 & 3.5 & 0 & 3.2 & \boxed{11.2} & 4 \\
1.7 & 1.2 & 3.7 & 3.2 & 0 & 6.7 & 6.8 \\
\boxed{9.7} & \boxed{9.2} & \boxed{11.7} & \boxed{11.2} & 6.7 & 0 & \boxed{14.8} \\
5.1 & 5.6 & 3.1 & 4 & 6.8 & \boxed{14.8} & 0
\end{bmatrix}
\end{array}
$$

$$
R^{(4)} = \begin{array}{cc}
 & \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{matrix} \\
\begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{matrix} &
\begin{bmatrix}
0 & 2 & 3 & 4 & 2 & 2 & 3 \\
1 & 0 & 1 & 1 & 5 & 6 & 3 \\
1 & 1 & 0 & 1 & 2 & 2 & 7 \\
1 & 1 & 1 & 0 & 2 & 2 & 7 \\
2 & 2 & 2 & 2 & 0 & 6 & 3 \\
2 & 2 & 2 & 2 & 5 & 0 & 3 \\
3 & 3 & 3 & 4 & 3 & 3 & 0
\end{bmatrix}
\end{array}
$$

$$
W^{(5)} = \begin{array}{cc}
 & \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{matrix} \\
\begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{matrix} &
\begin{bmatrix}
0 & 0.5 & 2 & 1.5 & 1.7 & \boxed{8.4} & 5.1 \\
0.5 & 0 & 2.5 & 2 & 1.2 & \boxed{7.9} & 5.6 \\
2 & 2.5 & 0 & 3.5 & 3.7 & \boxed{10.4} & 3.1 \\
1.5 & 2 & 3.5 & 0 & 3.2 & \boxed{9.9} & 4 \\
1.7 & 1.2 & 3.7 & 3.2 & 0 & 6.7 & 6.8 \\
\boxed{8.4} & \boxed{7.9} & \boxed{10.4} & \boxed{9.9} & 6.7 & 0 & \boxed{13.5} \\
5.1 & 5.6 & 3.1 & 4 & 6.8 & \boxed{13.5} & 0
\end{bmatrix}
\end{array}
$$

$$
R^{(5)} = \begin{array}{cc}
 & \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{matrix} \\
\begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{matrix} &
\begin{bmatrix}
0 & 2 & 3 & 4 & 2 & \boxed{5} & 3 \\
1 & 0 & 1 & 1 & 5 & \boxed{5} & 3 \\
1 & 1 & 0 & 1 & 2 & \boxed{5} & 7 \\
1 & 1 & 1 & 0 & 2 & \boxed{5} & 7 \\
2 & 2 & 2 & 2 & 0 & 6 & 3 \\
\boxed{5} & \boxed{5} & \boxed{5} & \boxed{5} & 5 & 0 & \boxed{5} \\
3 & 3 & 3 & 4 & 3 & \boxed{5} & 0
\end{bmatrix}
\end{array}
$$

- 后面会发现：$W^{(7)} = W^{(6)} = W^{(5)}$，$R^{(7)} = R^{(6)} = R^{(5)}$
- 我们从$R^{(7)}$中可以容易地得到任意两端间的最短径
- 而从$W^{(7)}$中可以容易地得到任意两端间的最短径长



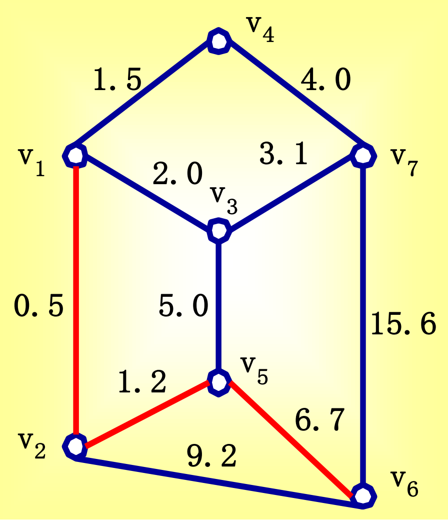
如v_1-- v_6间的最短径：

- 从v_1至v_6需经$r_{16}^{(7)}=5$端转接
- 而从v_1至v_5需经$r_{15}^{(7)}=2$端转接
- 从v_1至v_2需经$r_{12}^{(7)}=2$端转接，即走二者之间的边
- 所以，最短径为：$v_1 \to v_2 \to v_5 \to v_6$
- 最短径长为：$w_{16}^{(7)}=8.4$

$$W^{(7)} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \begin{bmatrix} 0 & 0.5 & 2 & 1.5 & 1.7 & \boxed{8.4} & 5.1 \\ 0.5 & 0 & 2.5 & 2 & 1.2 & 7.9 & 5.6 \\ 2 & 2.5 & 0 & 3.5 & 3.7 & 10.4 & 3.1 \\ 1.5 & 2 & 3.5 & 0 & 3.2 & 9.9 & 4 \\ 1.7 & 1.2 & 3.7 & 3.2 & 0 & 6.7 & 6.8 \\ 8.4 & 7.9 & 10.4 & 9.9 & 6.7 & 0 & 13.5 \\ 5.1 & 5.6 & 3.1 & 4 & 6.8 & 13.5 & 0 \end{bmatrix} \end{array}$$

$$R^{(7)} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \begin{bmatrix} 0 & 2 & 3 & 4 & 2 & 5 & 3 \\ 1 & 0 & 1 & 1 & 5 & 5 & 3 \\ 1 & 1 & 0 & 1 & 2 & 5 & 7 \\ 1 & 1 & 1 & 0 & 2 & 5 & 7 \\ 2 & 2 & 2 & 2 & 0 & 6 & 3 \\ 5 & 5 & 5 & 5 & 5 & 0 & 5 \\ 3 & 3 & 3 & 4 & 3 & 5 & 0 \end{bmatrix} \end{array}$$

△ 又如，反过来，或v_6-- v_1间的最短径



- 其结果肯定是一样的，只是找的顺序略有不同
- 从v_6至v_1需经$r_{61}^{(7)}=5$端转接
- 而从v_5至v_1需经$r_{51}^{(7)}=2$端转接
- 从v_2至v_1需经$r_{21}^{(7)}=1$端转接，即v_2直达v_1
- 所以，最短径为：$v_6 \to v_5 \to v_2 \to v_1$
- 最短径长为：$w_{61}^{(7)}=8.4$

$$W^{(7)}=\begin{array}{c} \\ v_1\\ v_2\\ v_3\\ v_4\\ v_5\\ v_6\\ v_7 \end{array}\begin{array}{c} \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array}\\ \begin{bmatrix} 0 & 0.5 & 2 & 1.5 & 1.7 & 8.4 & 5.1\\ 0.5 & 0 & 2.5 & 2 & 1.2 & 7.9 & 5.6\\ 2 & 2.5 & 0 & 3.5 & 3.7 & 10.4 & 3.1\\ 1.5 & 2 & 3.5 & 0 & 3.2 & 9.9 & 4\\ 1.7 & 1.2 & 3.7 & 3.2 & 0 & 6.7 & 6.8\\ \boxed{8.4} & 7.9 & 10.4 & 9.9 & 6.7 & 0 & 13.5\\ 5.1 & 5.6 & 3.1 & 4 & 6.8 & 13.5 & 0 \end{bmatrix}\end{array}$$

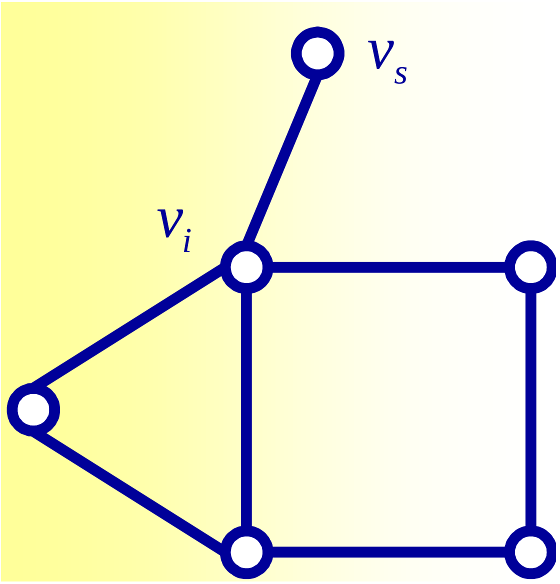
$$R^{(7)}=\begin{array}{c} \\ v_1\\ v_2\\ v_3\\ v_4\\ v_5\\ v_6\\ v_7 \end{array}\begin{array}{c} \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array}\\ \begin{bmatrix} 0 & 2 & 3 & 4 & 2 & 5 & 3\\ 1 & 0 & 1 & 1 & 5 & 5 & 3\\ 1 & 1 & 0 & 1 & 2 & 5 & 7\\ 1 & 1 & 1 & 0 & 2 & 5 & 7\\ 2 & 2 & 2 & 2 & 0 & 6 & 3\\ 5 & 5 & 5 & 5 & 5 & 0 & 5\\ 3 & 3 & 3 & 4 & 3 & 5 & 0 \end{bmatrix}\end{array}$$

= F算法的复杂性

- ≡F算法的计算量约为：$2\ n\ n^2 = 2n^3$量级
  - Δ因为，去掉第k行和第k列后，要做:
    - $(n-1)*(n-1)$次加法
    - $(n-1)*(n-1)$次比较
  - Δ 从第1行第1列到第n行第n列共需计算n次
- ≡所以，复杂度属非NP问题

三 F算法的简化

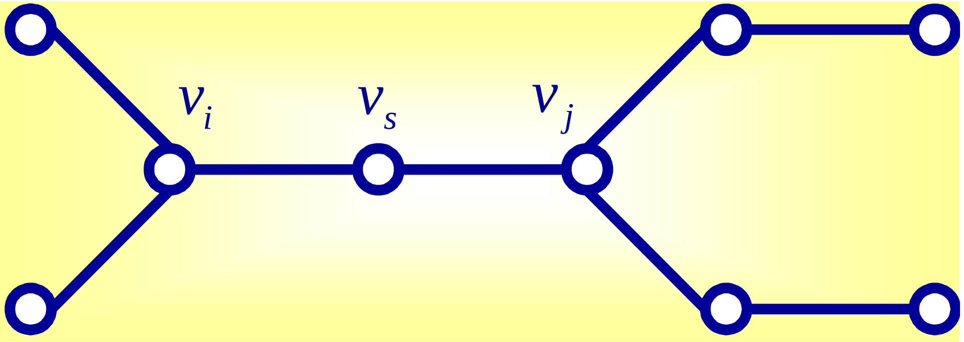
- F算法在某些情况下可以简化，即减少运算量
- 所有度数为1的端，可以先去掉，然后再计算
  - 即悬挂边可不参加运算
  - （其实孤立端也可不参加运算，因为它没有路由）
  - 如图：v_s是度数为1的端
  - v_s只与v_i相连，所以，其最短径必经过v_i
  - 若w_{ij}是去掉v_s端后，用F算法求得的最短径长
    - 则v_s与v_j间最短径长必为：$w_{sj} = d_{si} + w_{ij}$



F算法的简化 (2)

所有度数为2的端，也可以按下述方法去掉

- 如图：v_s是度数为2的端，仅与v_i和v_j相连



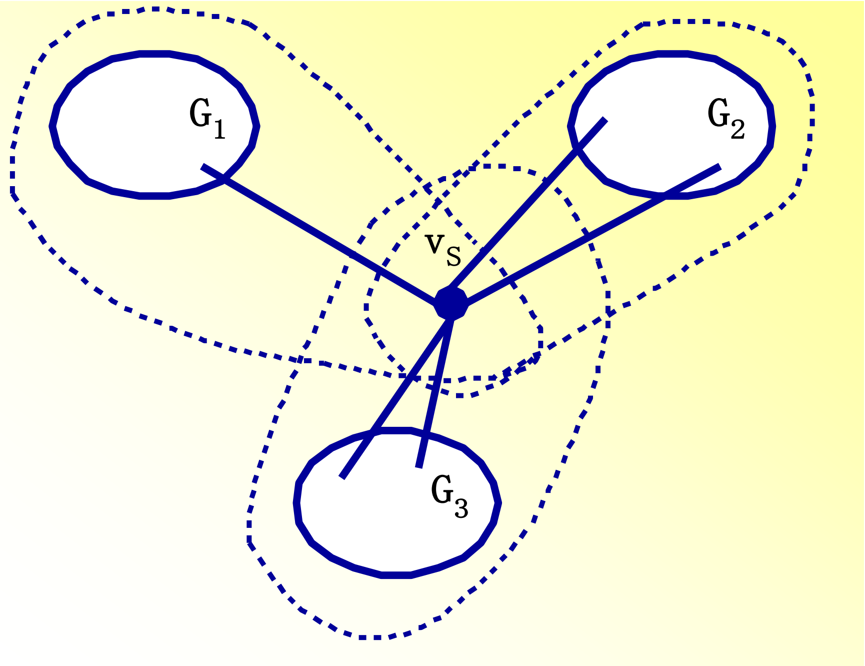
- 若$d_{si} + d_{sj} \geq d_{ij}$（若$v_i$与$v_j$间无边，则$d_{ij} = \infty$）
  - 则简单地去掉v_s即可
- 若$d_{si} + d_{sj} < d_{ij}$
  - 则把d_{ij}改为$d_{si} + d_{sj}$后，再去掉v_s即可
  - 当最短径的路由经过v_i与v_j间的边时，应理解为是经v_s转接的
- 不论哪种情况，v_s至其它端的最短径长为:
  - $w_{sk} = \min\{ d_{si} + w_{ik} , d_{sj} + w_{jk} \}$

F算法的简化(3)

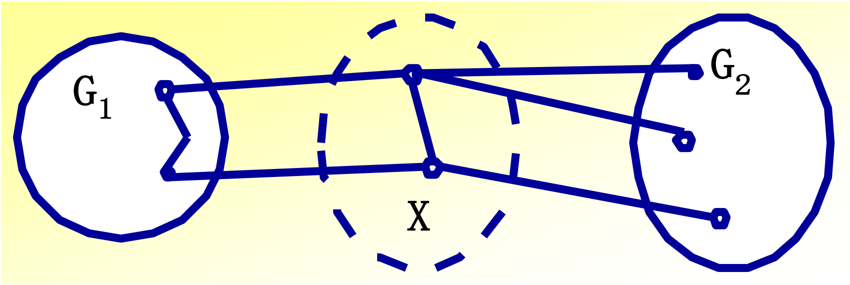
稀疏网的情况

△ 稀疏网：

- 全联结网的边数为： $m = \frac{1}{2} \cdot n \cdot (n-1)$
- 若一个网的边数远小于此值，则称之为稀疏网
- 稀疏网通常存在割端，或端数较少的割端集
- 如图所示：



图a



图b

图a中的v_s是割端

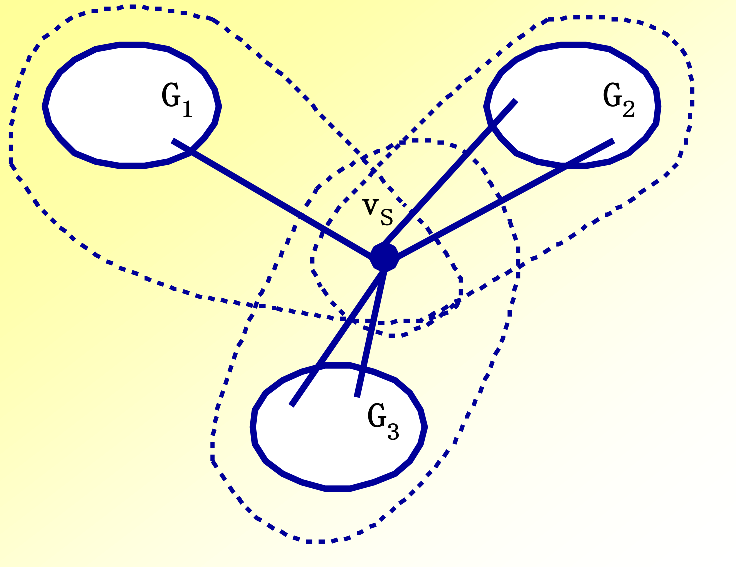
图b中的 X 是割端集

Δ 图a的简化算法

- 去掉割端v_s
- 图G变成了三个部分：$G - \{v_s\} = G_1 + G_2 + G_3$

- 用F算法分别求子图$G_1+\{v_s\}$， $G_2+\{v_s\}$， $G_3+\{v_s\}$的
  最短径长阵：W′，W″，W‴ 和
  路由阵 ：R' ，R'' ，R'''

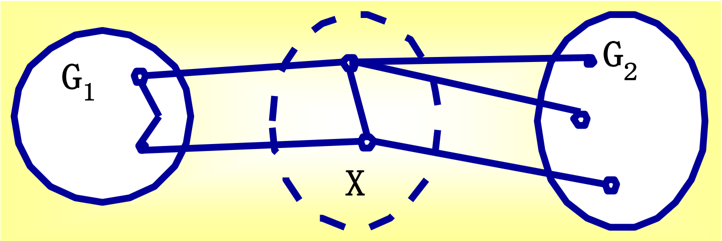
- 则两子集间的最短路径即为两子集内最短路径之和
- 即： 若 $v_i \in G_1$， $v_j \in G_2$，$w'_{is} \in W'$，$w''_{js} \in W''$
- 则 $w_{ij} = w'_{is} + w''_{js}$

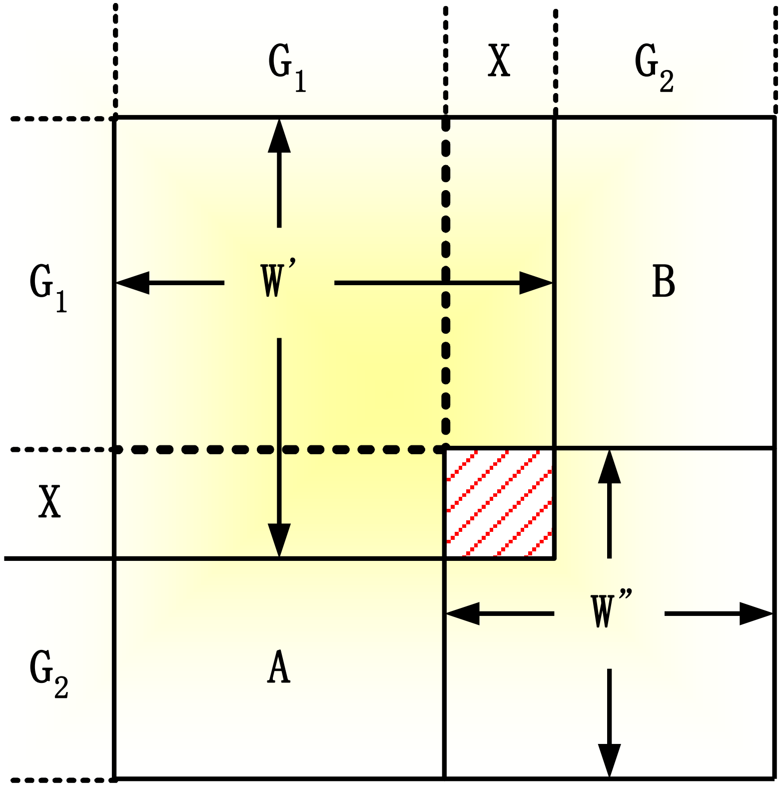


- v_i与v_j间路由也可从 R′ 和R''中求得

Δ 图b的简化算法

- 图G由三个部分构成：$G = G_1 + X + G_2$
- 应当使割端集X中的端尽量少
- 用F算法分别计算子图 $G_1 + X$ 和 $G_2 + X$ 的
  最短径长阵：W′和W″
  路由阵　　：R′和R″
- 将二者合起来，即得图G的相应阵





– 子图X所对应的部分是W′和W″的**公共部分**（**交叉部分**，或**阴影部分**）
  = 若二者的值不同，则取小者
  = 假设W′ 中的值比W″中的值小，则还应调整W″中，经X集转接的值
  = 这意味着X集里，端间的最短径是由G_1中的端转接的
  = G_2中的最短径，凡通过X集里的端转接的，也将由G_1转接
– 对于**A和B两部分**：
  = 设$v_i \in G_1$，$v_j \in G_2$，　则最短径长为：

$$w_{ij} = \min\{ w'_{ik} + w''_{kj} : v_k \in X \}$$

≡ 简化后F算法的复杂性

∆ F算法的计算量与端数n的三次方成正比，O(n^3)

∆ 若把图G分成k个子图，则计算量为:

- $$\left(\frac{n}{k}\right)^3 \cdot k = \frac{n^3}{k^2}$$

- 可见，k越大，减少的程度将越显著

= 用D算法和F算法得到的最短径，可以证明，是最优的

≡ 即不会存在更短的径

≡ 至多存在另一条等长的径

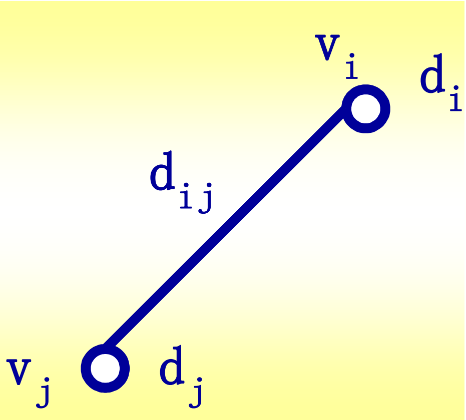
＝边和端均有权时，求两端间最短径和路由的方法

≡思路：将端的权值转化为边的权值

≡方法：

Δ 如图：

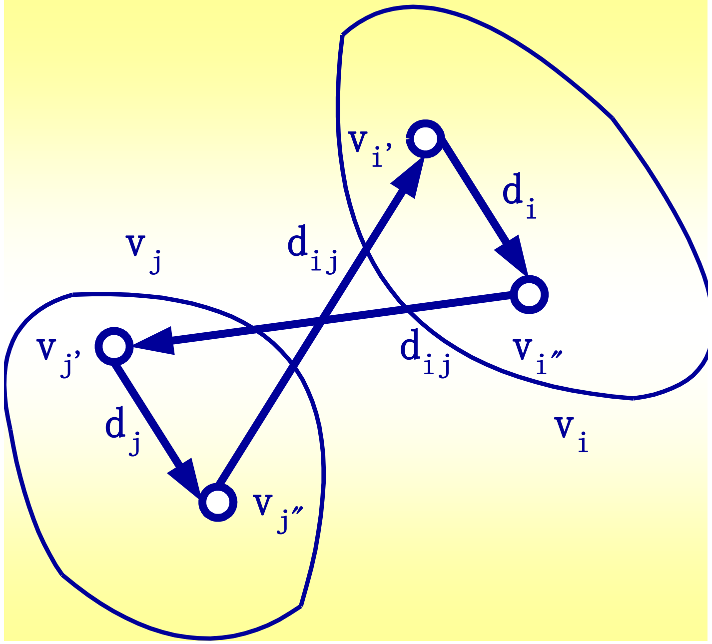
- d_{ij}：是v_i至v_j边的权值
- d_i：是v_i端的权值
- d_j：是v_j端的权值



Δ 做变换：

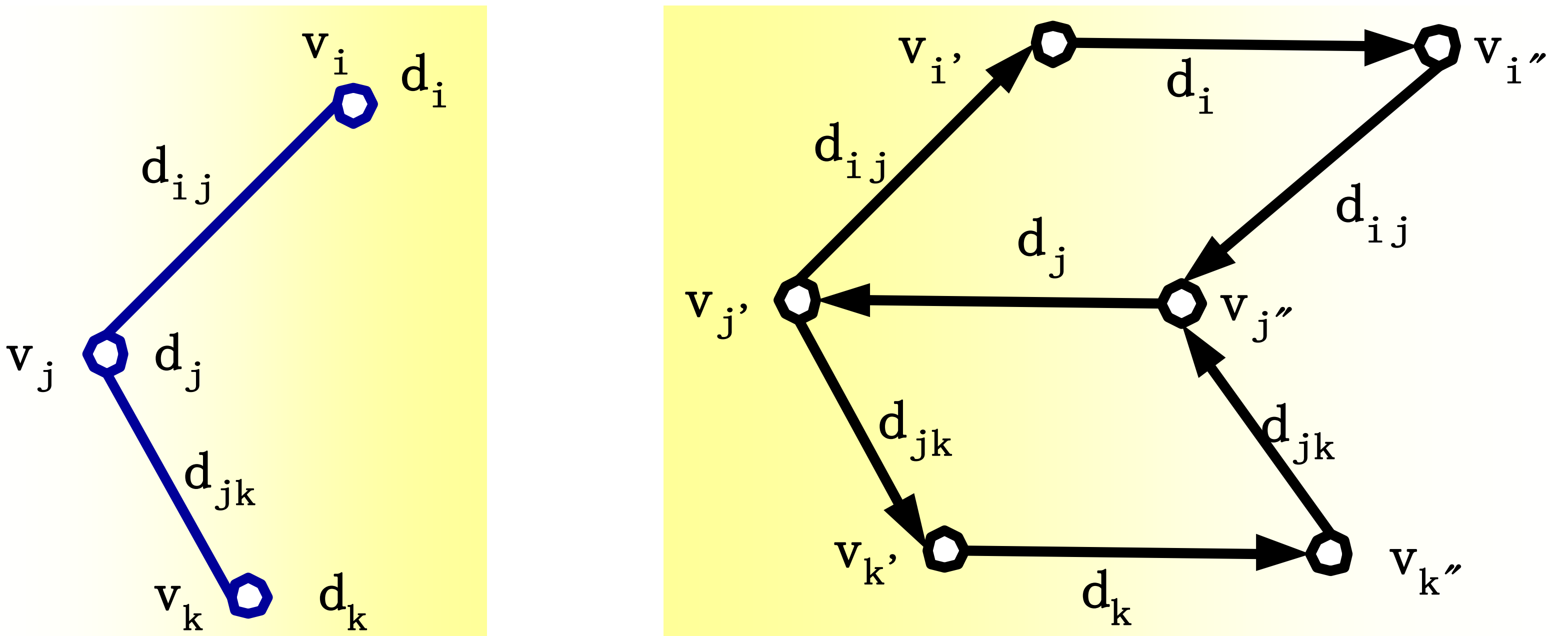
- 将v_i拆成两个端$v_{i'}$和$v_{i''}$
- 中间用有向边连起来
- 此边的权值即为端的权值d_j
- 同样处理所有的端

Δ 然后使用D算法或F算法



≡ 例:

△ 如图:



- v_i至v_k的路径为: $i'' \to j'' \to j' \to k'$
- v_i至v_k的径长为: $d_{ij} + d_j + d_{jk}$

- 当图中边的权有正有负时
  - 若图中有负价环（负圈）时，即某$w^{(k)}(i, i) \le 0$时
    - 则F算法与D算法一样，也无效
    - 只是不像D算法那样会出错
    - F算法还是可以运行的
    - 只是当W阵的对角线上的元素出现负值时
      - 意味着图中存在负价环（负圈）
      - 此时根据路由阵R可以找出负价环的路由构成
    - 当这种情况出现时， F算法终止即可
    - 亦即：使用F算法时并不需考虑图中是否有负权的边
  - 若图中没有负价环（负圈）
    - 则最后的结果就是各端对间的最短径和路由
    - 即F算法是有效的

- 当限定不能经过某些端点转接时
  - 仍然可以使用F算法，初始化的值也是一样的
  - 只是对那些不能转接的端点，运算时跳过去即可
  - 如： v_i不能转接，则只要跳过第$W^{(i)}$步即可
  - 总的迭代次数减少，别的不用变
  - 很容易实现限定转接节点的情况
- 欲求转接次数最少的路由时
  - 仍然可以使用F算法
  - 只是初始化时，将边的权值赋为1
  - 得到的最短距离即为转接次数加1

- 这两个小应用也说明了F算法能应用于多种情形

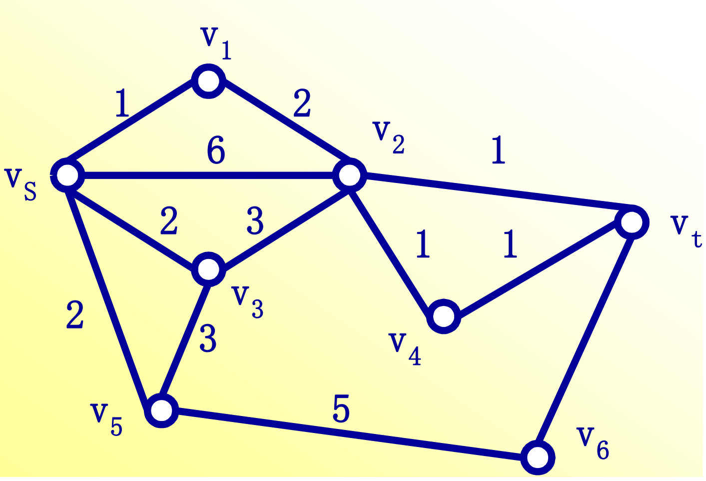
– 次短径和可用径

≡在实际应用中，除了最短径外

△通常还需要知道次短径和可用径

△作为备用路由和迂回路由

≡例如，当主路由业务量溢出时，或发生故障时，就会用到这些路由



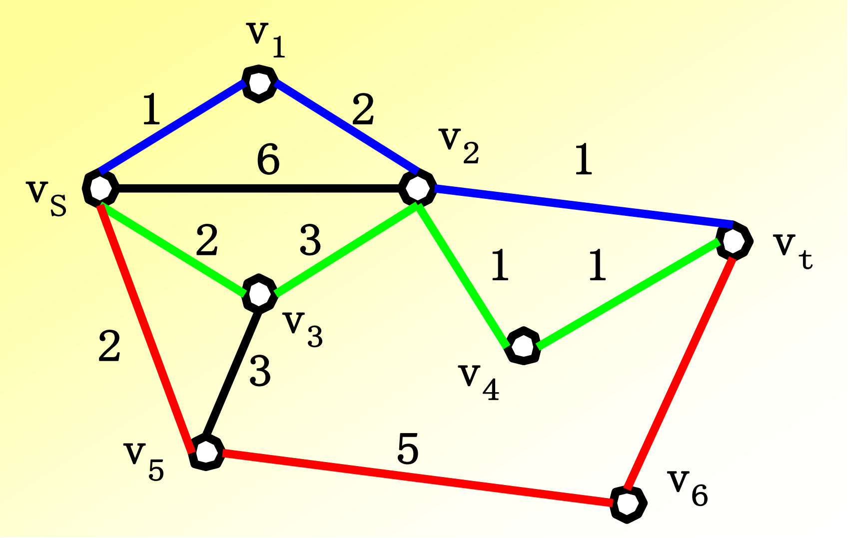
= 不共边径（边分离径）

≡如上图的网络中

△ P_1: $v_s \to v_1 \to v_2 \to v_t$

△ P_2: $v_s \to v_3 \to v_2 \to v_4 \to v_t$

△ P_3: $v_s \to v_5 \to v_6 \to v_t$

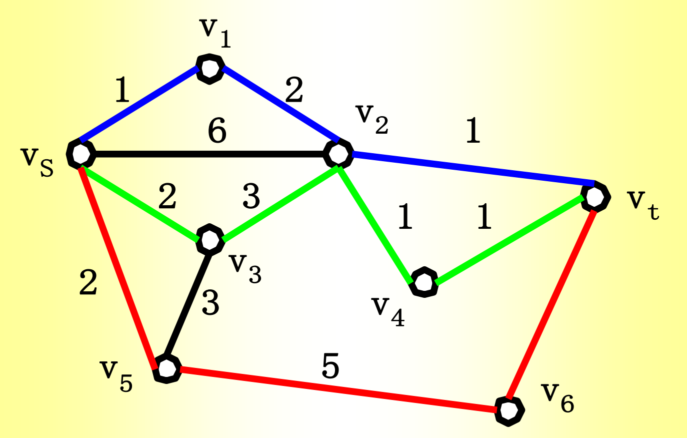


≡若P_1是最短径

△ P_2与P_1没有公共边，但有公共端

△则称P_1与P_2为不共边径，或边分离径

- 不共端径（端分离径）
  - P_1与P_3除了起点和终点外，没有公共端
    - 则称P_1与P_3为不共端径，或端分离径
  - 关系：
    - 端分离径必为边分离径
    - 反之则不一定



- 求最短径的边分离次短径
  - 首先用F算法或D算法求出最短径
  - 从图中去掉这条径的所有边
  - 在剩下的图中用D算法求v_s至v_t的最短径
    - 得到的便是次短径
  - 此法可以继续算下去
    - 还可以得到次短径的边分离次短径
  - 直至v_s与v_t间无径

- 求最短径的端分离次短径
  - 首先用F算法或D算法求出最短径
  - 从图中去掉这条径的所有中间端
    - 去掉一个端，意味着同时也去掉与之关联的边
    - 去掉一条边，则应保留这条边的两个端
  - 在剩下的图中求v_s至v_t的最短径
    - 便可得到最短径的端分离次短径
  - 此法也可以继续算下去，直至不存在径
  - 在前例中，
    - p_2是边分离次短径，
    - p_3是端分离次短径，也是边分离的更次短径

- 可用径
  - 在某些应用中，需要找一批满足某种限制条件的径
    - 但并不要求端分离或径分离
    - 称这些径为该限制条件下的可用径
  - 随限制条件的不同，解法也不同
  - 也可能会同时要求满足几个限制条件
    - 则可先用一个条件求可用径
    - 再从这些径中筛选满足其它条件的径

– 网的中心、中点和直径

= 网的中心

≡ 一个端v_i在网内的位置可用最长的最短径长t_i来表示

≡ 即：$t_i = \max\{ w_{ij} : 对所有j \}$

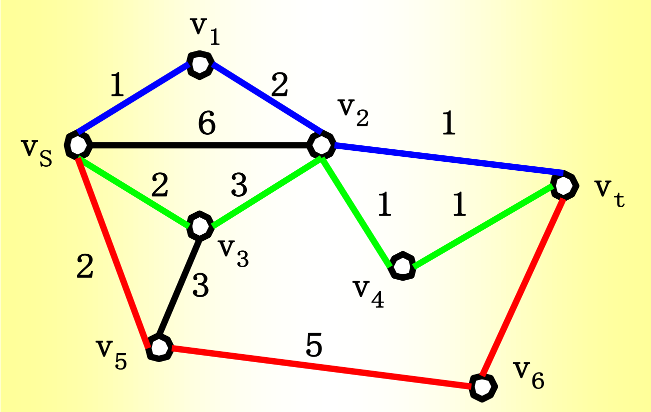
≡ 亦即：

Δ v_i到v_j端有一条最短径

Δ v_i到v_k端有一条最短径

Δ v_i到网内所有端都有一条最短径

Δ 在这些最短径中，找出最长的一条，其径长为t_i



≡ 最小的t_i值所对应的端称为网的中心

Δ 即：让v_i遍历网内所有的端，可以得到n个t_i（i = 1，2，··n）

Δ 亦即：网内任何一端都有其最长的最短径 t_i（i = 1，2，··n）

Δ 在这n个 t_i中，找出最小的一个 t_{i*}

- 即：
$$t_{i*} = \min_i t_i = \min_i \left\{ \max_j \left\{ w_{ij} \right\} \right\}$$

- 亦即：v_{i*}就是网的中心

- 网中心的含义:
  - 从距离的意义上说
  - 若按最短径行走
  - 从网中心到最远端去
  - 所走的路程
  - 比其它端到它们的最远端所走的路程都要小
- 网中心作为维修中心和服务中心是最有利的
  - 这一结论不仅对通信网有用
  - 对其它网络也如此

＝网的中点

≡平均最短径长最小的端称为网的中点

Δ即：$S_i = \sum_j w_{ij}$

Δ $$S_{i*} = \min_i S_i = \min_i \left[\sum_j w_{ij} \right]$$

Δ则v_{i*}为网的中点

Δ亦即：网内任一端v_i到其它$n-1$个端都有最短径w_{ij}

- 将所有最短径求和，为S_i

Δ将v_i遍历网内所有端，可以得到n个S_i

Δ在n个S_i中取最小者S_{i*}

Δ则它所对应的端v_{i*}就是网的中点

- 网中点的含义
  - 如果边的权值均取1
  - 则网的中点就是到其它各端去平均转接次数最少的端
  - 或者，从网的中点出发
  - 跑遍网内所有节点
  - 所走过的路程最短
- 网的中点可用作全网的收费中心、控制中心或交换中心

= 网的直径

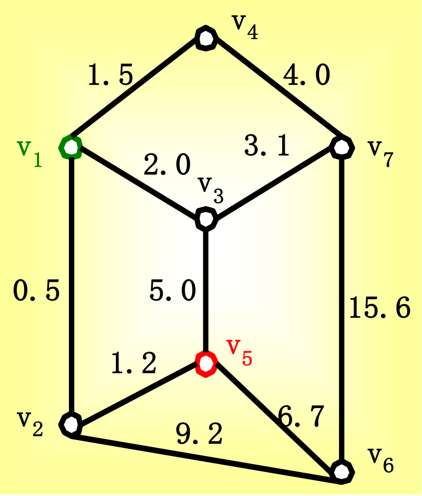
≡ 网内两端间最短径长的最大值称为网的直径

∆ 即：$D = \max\{ w_{ij} : \text{对所有} i, j\}$

= 举例:

≡ 同F算法例

$$W^{(7)}=\begin{array}{c|c} & \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array} & \begin{bmatrix} 0 & 0.5 & 2 & 1.5 & 1.7 & 8.4 & 5.1 \\ 0.5 & 0 & 2.5 & 2 & 1.2 & 7.9 & 5.6 \\ 2 & 2.5 & 0 & 3.5 & 3.7 & 10.4 & 3.1 \\ 1.5 & 2 & 3.5 & 0 & 3.2 & 9.9 & 4 \\ 1.7 & 1.2 & 3.7 & 3.2 & 0 & 6.7 & 6.8 \\ 8.4 & 7.9 & 10.4 & 9.9 & 6.7 & 0 & 13.5 \\ 5.1 & 5.6 & 3.1 & 4 & 6.8 & 13.5 & 0 \end{bmatrix} \end{array} \qquad \begin{array}{cc} \max_j [w_{ij}] & \sum_j w_{ij} \\ 8.4 & 19.2 \\ 7.9 & 19.7 \\ 10.4 & 25.2 \\ 9.9 & 24.1 \\ 6.8 & 23.3 \\ 13.5 & 56.8 \\ 13.5 & 38.1 \end{array}$$



$$\min_i \left\{ \max_j \left[w_{ij} \right] \right\} = 6.8$$ 即v_5为网的中心

≡ $$\min_i \left[\sum_j w_{ij} \right] = 19.2$$ 即v_1为网的中点

≡ 网的直径: $$D = \max_{i,j} w_{ij} = \max_i \left\{ \max_j \left[w_{ij} \right] \right\} = 13.5$$

– 作业

= （周先生著）P155. 3.7, 3.8, 3.9

= （张琳著）P142. 4.9

= 编一个程序实现D算法

= 编一个程序实现F算法

(本节结束)