算法设计

——贪心算法

陈卫东

chenwd@scnu.edu.cn

华南师范大学计算机学院

2021-10

贪心算法

- 贪心法的基本思想
- 贪心法的应用
  - 4.1 区间调度
  - 4.2 缓存维护
  - 4.3 单源最短路
  - 4.4 最小生成树(Kruskal算法、Prim算法)
  - 4.5 压缩问题 (Huffman算法)
- 拓展应用举例

贪心法的基本思想

- ## 贪心法的适用问题

贪心法又叫优先策略，顾名思义就是“择优录取”，它是一种非常简单好用的求解最优化问题的方法，在许多方面的应用非常成功。

- ## 贪心法的基本思想

  - 贪心算法是根据一种**贪心准则(Greedy Criterion)**来逐步构造问题的解的方法。在每一个阶段，都作出了相对该准则最优的决策。决策一旦作出，就不可更改。
  - 由贪心法得到的问题的解可能是最优解，也可能只是近似解。能否产生问题的最优解需要加以证明。
  - 所选的贪心准则不同，则得到的贪心算法不同，贪心解的质量当然也不同。因此，贪心算法的好坏关键在于正确的选择贪心准则。

举例

【找零钱问题】

假定有足够多的各种面值的人民币，要给顾客找零钱y元，如何找替才能使得钱币张数最少?

- 假设找给顾客人民币y=15元，如何找?
  ——最少需要2张钱币（贪心法）
  注：可以证明对任何y，上述贪心法都有效。

- 假设某个国家货币面值是1,5,11三种，该如何找给顾客y=15元?
  ——最少需要3张钱币。
  注：上述贪心法不会总是有效。

举例

【找零钱问题】

假定有足够多各种面值的人民币，其面值分别为1,2,5,10,20,50,100，现在需要给顾客找零钱y元，如何找替才能使得钱币张数最少?

- 求解方法：贪心法

思考： 如何证明对任何y，贪心法都有效?

举例

【背包问题】

给定一个承重量为M的背包，n个重量分别为$w_1,w_2,\ldots,w_n$的物品。已知物品i放入背包能产生p_i的价值（放入单位重量的物品i，产生的价值为p_i/w_i），i=1,2,…,n。如何装包才能获得最大价值?

实际上，就是要求找到一组非负且不超过1的实数$x_1,x_2,\ldots,x_n$满足$\sum_{i=1}^{n} x_i w_i \le M$，且使得$\sum_{i=1}^{n} x_i p_i$达到最大值。

- **实例**

  n=3，M=20,

  (w_1, w_2, w_3)=(18, 15, 10),　(p_1, p_2, p_3)=(25, 24, 15).

- **求解算法**

贪心法1——根据物品的价值由大到小来放。
（上述实例的贪心解为 (1, 2/15, 0)，价值为 25+24×2/15=**28.2**）

贪心法2——根据物品的重量由小到大来放。
（上述实例的贪心解为 (0, 2/3 , 1)，价值为 24×2/3+15=**31** ）

贪心法3——根据价值/重量（即单位价值）由大到小来放。
（上述实例的贪心解为(0, 1, 0.5)，价值为 24+15×0.5=**31.5**）

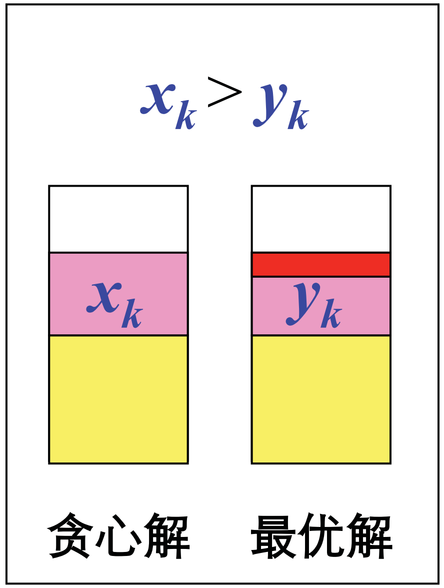
*注：可证明贪心法3对背包问题任何实例都能产生最优解。

■ 贪心法3的性能分析:

✓ 时间复杂度: $O(n\log n)$

✓ **正确性证明思路:**

不妨假设有 $p_1/w_1 \geq p_2/w_2 \geq \ldots \geq p_n/w_n$



贪心解: $X=(x_1, x_2, \ldots, x_{k-1}, \boldsymbol{x_k}, x_{k+1}, \ldots, x_n)$

$\vdots \qquad \vdots$

$\Uparrow \qquad \Uparrow$

最优解: $Z=(x_1, x_2, \ldots, x_{k-1}, \boldsymbol{x_k}, z_{k+1}, \ldots, z_n)$

最优解: $Y=(x_1, x_2, \ldots, x_{k-1}, \boldsymbol{y_k}, y_{k+1}, \ldots, y_n)$

保持领先法

交换论证法

举例

【0-1背包问题】

已知：一个承重量为M的背包，n个重量分别为w_1，w_2，…，w_n的物品。每个物品要么整个放入背包，要么不放。且已知物品i放入背包能产生$p_i(>0)$的价值，$i=1,2,\ldots,n$。问：如何装包才能获得最大价值?

- 求解思想：可用多种贪心法来求解该问题，其中的一种贪心准则是：按物品的单位价值由大到小的次序来考虑物品的放还是不放入包中。

【实例1】考虑n=4，w=[2,4,5,7]，p=[6,10,12,13]，M=11时的0/1背包问题。

- **解：** 用上述方法容易得到其解为：

  $x_1=1$，$x_2=1$，$x_3=1$，$x_4=0$，装包价值为：28。

注：可看出它是最优解。

【实例2】考虑n=4，w=[2,4,6,7]，p=[6,10,12,13]，M=11时的0/1背包问题。

- **解：** 用上述方法容易得到其解为：

  $x_1=1$，$x_2=1$，$x_3=0$，$x_4=0$，总价值为：16。

注：最优解为：$x_1=0$，$x_2=1$，$x_3=0$，$x_4=1$，装包价值为：23。

4.1 区间调度

【区间单机调度问题】

有n件任务和一台机器，任务可在机器上得到处理。每件任务j的开始时间为s_j，完成时间为f_j，$s_j<f_j$，即$[s_j,f_j]$为处理任务j的时间范围。规定机器在任何时刻最多只处理一件任务，且一件任务的处理不允许间断，要连续处理直到结束。

找出一种安排任务方案使得该机器能完成最多数目任务。

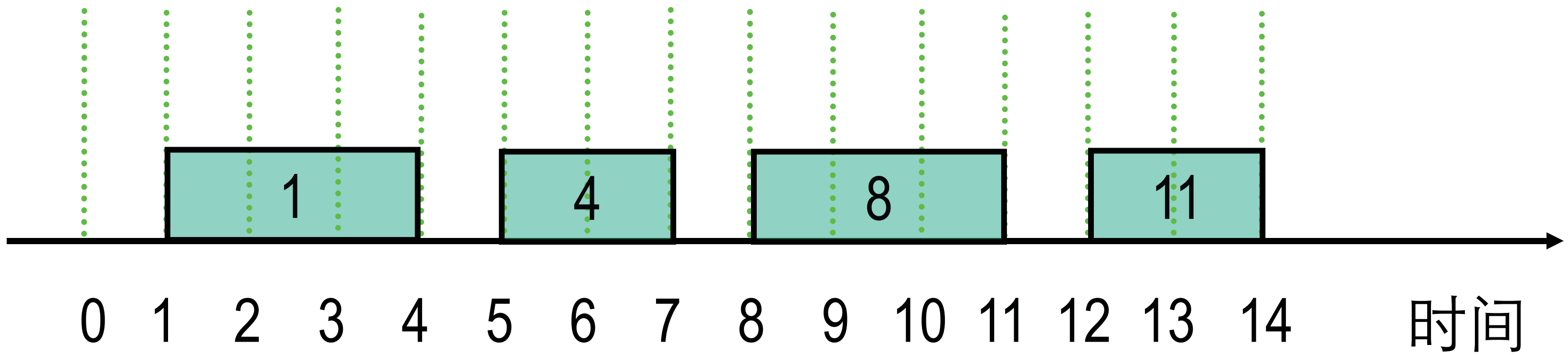
贪心算法

先将任务按完成时间从小到大排序，即$f_1 \leqslant f_2 \leqslant \cdots \leqslant f_n$。然后依此次序来考虑各个任务的安排：**如果当前任务与已经安排的任务没有时间冲突，就安排该任务，否则就不安排该任务。**

实例

设有11个任务待安排，它们的开始时间和结束时间如下：

任务 j:	1	2	3	4	5	6	7	8	9	10	11
开始时间s_j:	1	3	0	5	3	5	6	8	8	2	12
结束时间f_j:	4	5	6	7	8	9	10	11	12	13	14

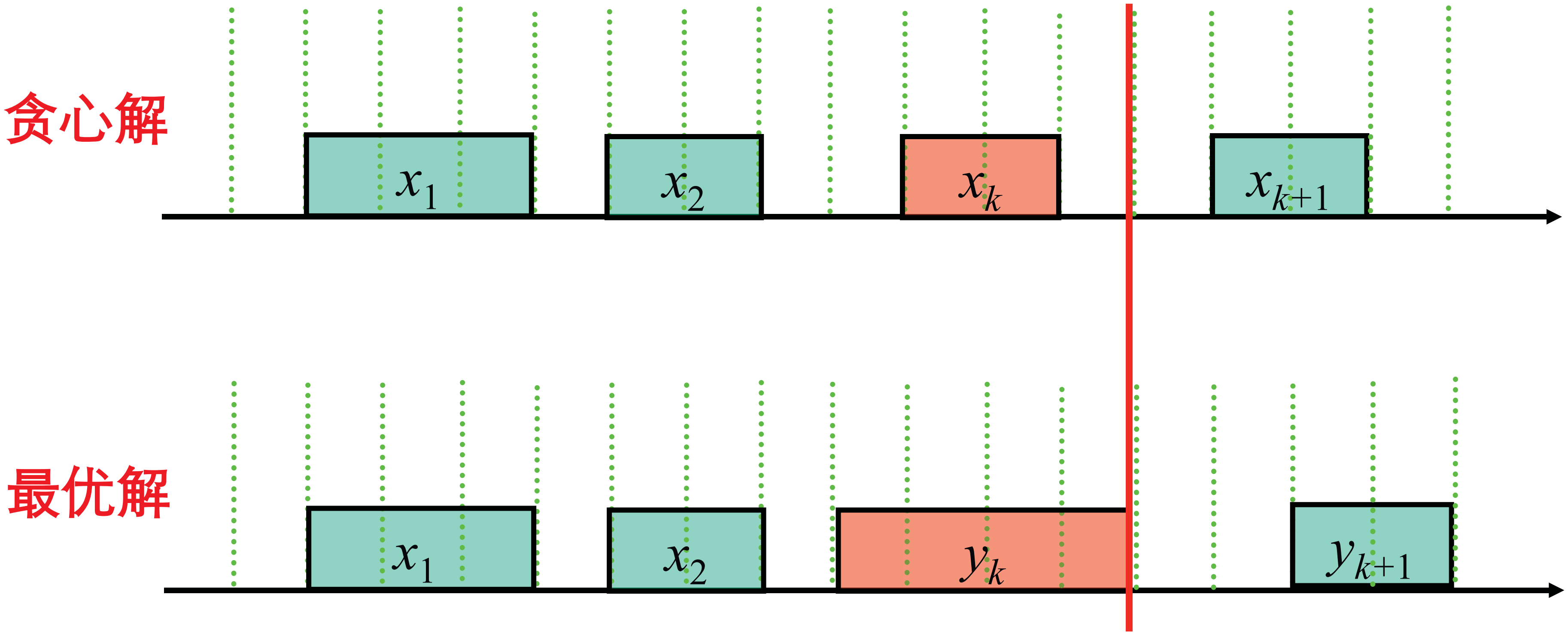


■ 贪心算法性能分析

✓ **时间复杂度：** O($n$log$n$)

✓ **正确性证明思路：**

假设任务按已经按完成时间从小到大有序，即有 $f_1 \le f_2 \le \dots \le f_n$。



举例

【区间多机调度问题】

有n件任务和足够多台机器，任务可在机器上得到处理。每件任务j的开始时间为s_j，完成时间为f_j，$s_j<f_j$，即$[s_j,f_j]$为处理任务j的时间范围。规定每台机器在任何时刻最多只处理一件任务，且一件任务的处理不允许间断，要连续处理直到结束。

找出一种任务安排方案使得用最少的机器完成所有的任务。

贪心算法

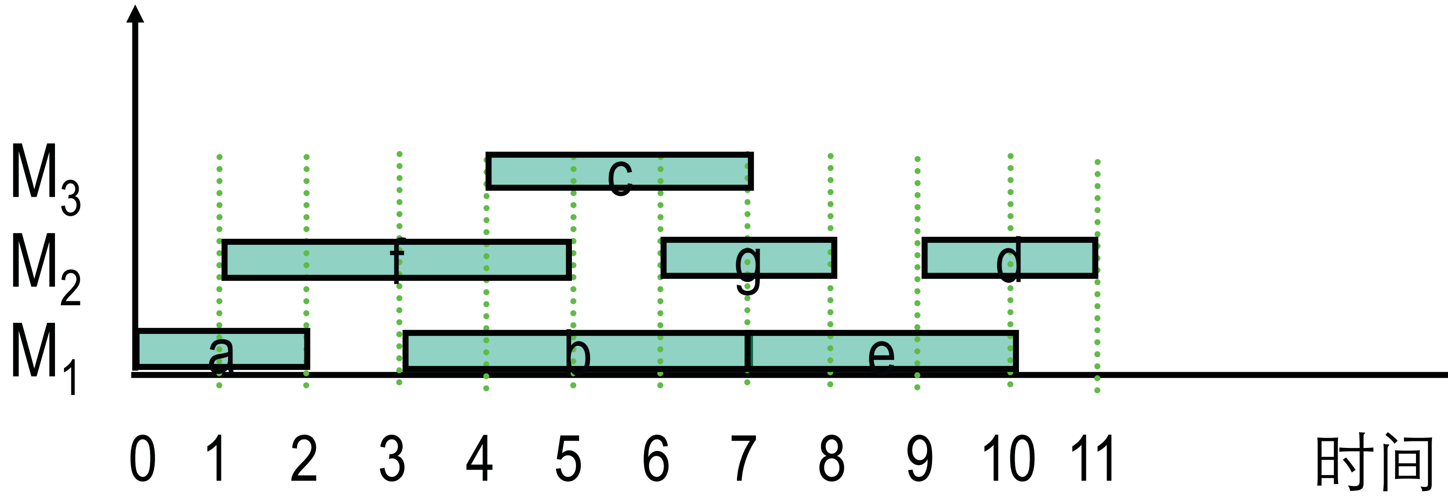
先将任务按开始时间从小到大排序。即$s_1 \le s_2 \le \ldots \le s_n$。然后依此次序来考虑任务的安排（即按贪心准则来安排），一旦某个任务考虑过了，即从剩余任务中去掉。考虑每个任务的安排规则如下：

若已至少有一件任务分配给了某台机器，则称该机器是旧的，否则就是新的。按任务的开始时间的非减次序依次来安排任务。若此时有了旧的可用，则将任务分配给旧的，否则，将任务分配给一台新机器。

实例

设有7个任务待安排，它们的开始时间和结束时间如下：

任务:	a	b	c	d	e	f	g
开始时间:	0	3	4	9	7	1	6
结束时间:	2	7	7	11	10	5	8

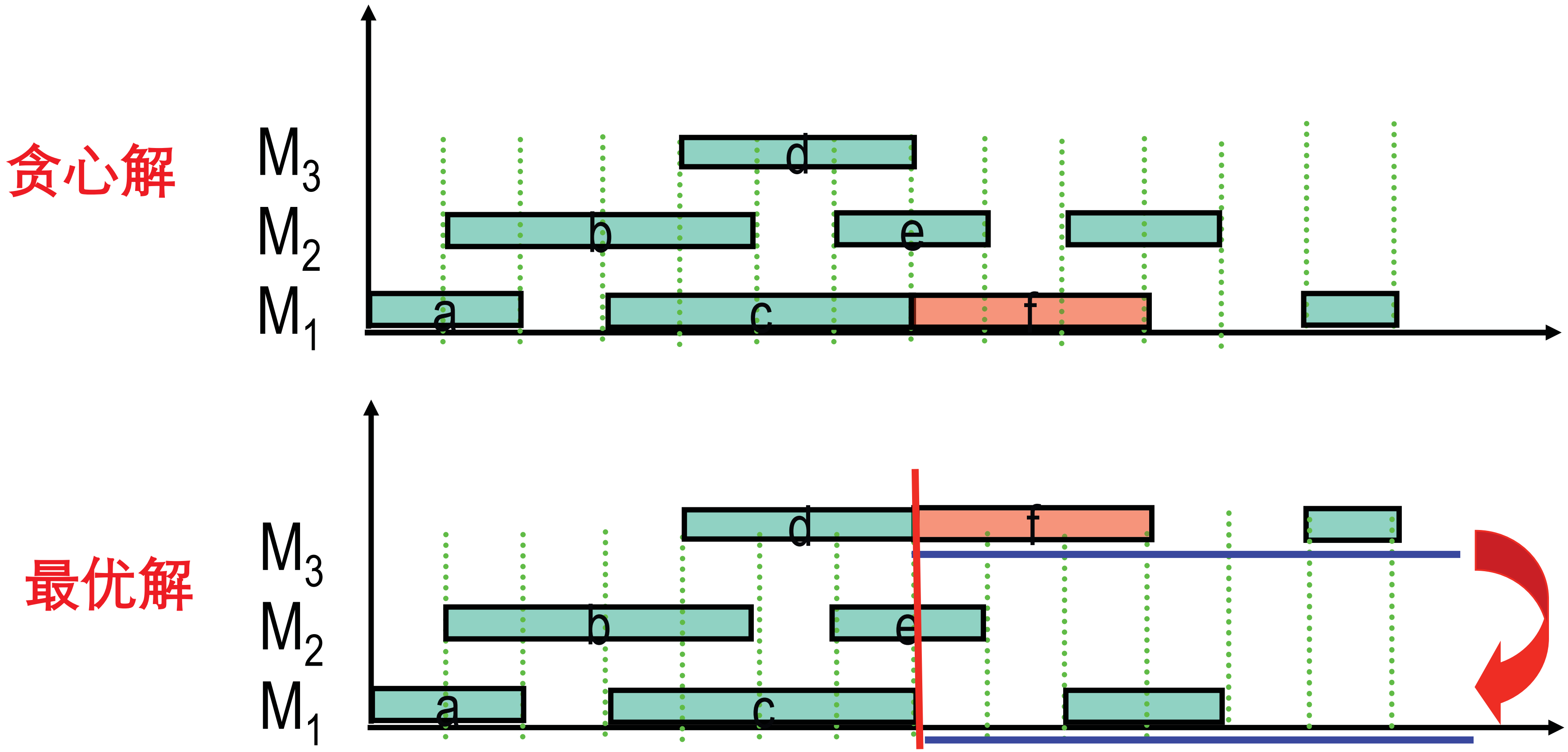


■ 贪心算法性能分析

✓ **时间复杂度：** $O(n\log n)$

✓ **正确性证明思路：**

假设任务按已经按开始时间从小到大有序，即有 $s_1 \le s_2 \le \ldots \le s_n$。



4.3 单源最短路

【单源最短路径问题】

设 G=<V,E> 是一个有向图，图中每一条边都有一个非负长度。单源最短路径问题就是要求出从图中一定点s（称为源点）到其它各点的长度最短的路径。

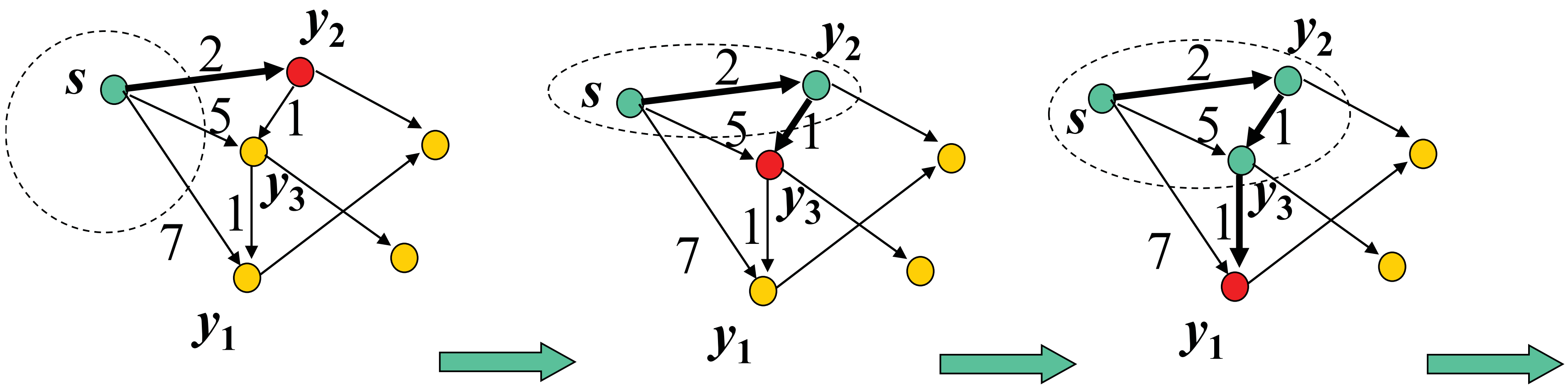
Dijkstra算法（贪心法）

贪心准则

按照从源点s到各点最短路径的长度由小到大依次构造s到各点的最短路径。

示意图

从图得知，这样得到了从s到每点v只经过圈中节点的那些路径中的最短路径，其长度记作$d'[v]$。可以证明这等于从s到v的最短路径的长度$d[v]$(距离)。



Dijkstra 算法(G, l)

设 S 是被探查的结点的集合

对每个 $u \in S$，我们存储一个距离 $d(u)$

初始 $S=\{s\}$ 且 $d(s)=0$

While $S \neq V$

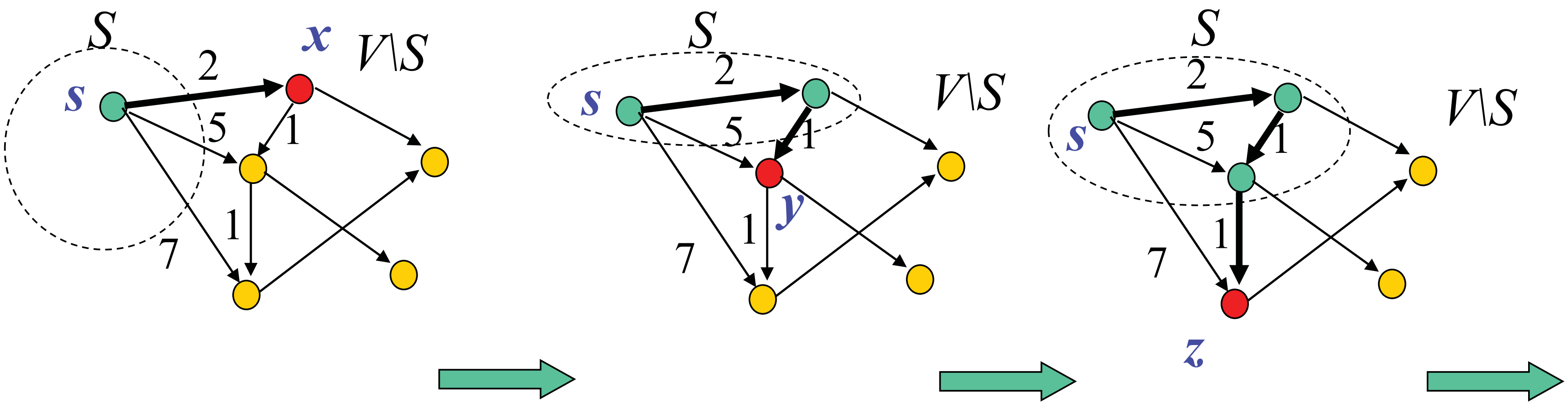
选择一个结点 $v \notin S$ 使得从 S 到 v 至少有一条边并且

$$d'(v)=\min_{e=(u,v):u\in S} d(u)+l_e \text{ 最小}$$

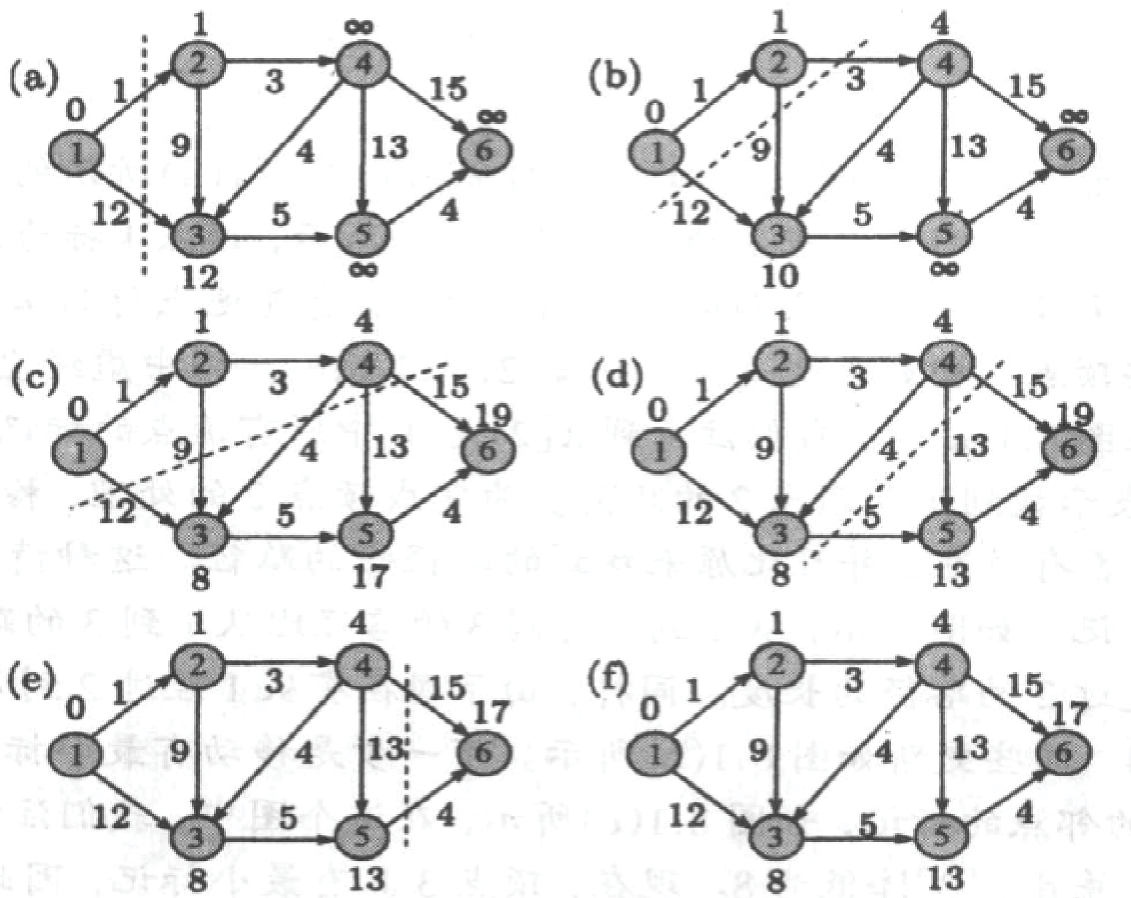
将 v 加入 S 并且定义 $d(v)=d'(v)$

Endwhile

$$d'[v] \leftarrow \mathbf{min}_{e[u,v]:\, u\in S} \{d[u]+l_e\}$$



➢ 实例

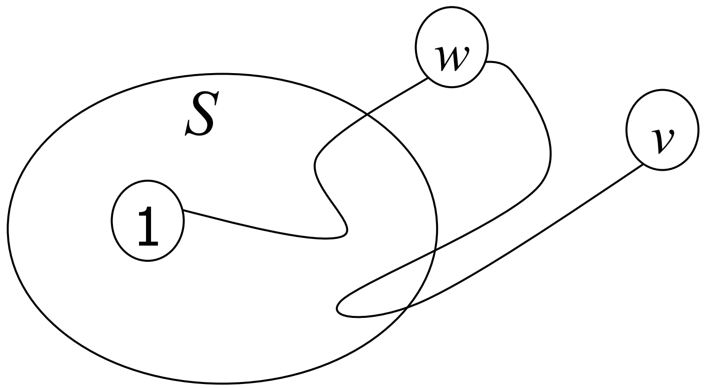


迭代次数	集合S中的元素	d'[1]	d'[2]	d'[3]	d'[4]	d'[5]	d'[6]	最短路径
初始化	1	-	**1**	12	∞	∞	∞	
1	1, 2	-	-	10	**4**	∞	∞	1→2
2	1, 2, 4	-	-	**8**	-	17	19	1→2→4
3	1, 2, 4, 3	-	-	-	-	**13**	19	1→2→4→3
4	1, 2, 4, 3, 5	-	-	-	-	-	**17**	1→2→4→3→5
5	1, 2, 4, 3, 5, 6	-	-	-	-	-	-	1→2→4→3→5→6

➢ 算法正确性

定理 Dijkstra算法每步选择一个结点v时,如果 $d'[v] < \infty$，则$d'[v] = d[v]$，其中$d[v]$表示从源点到v的距离。

证明思路： 对结点进入S的次序进行归纳证明。



- 初始时（第0步时），结论显然成立。
- 归纳假设：设第k步之前都有结论成立。
- 要证第k步选择节点v时结论成立。用反证法，假定结论不成立，即$d'[v] \neq d[v]$。此时必有$d'[v] > d[v]$，且由算法思路知必存在有一条长为$d[v]$的从1到v的路径其上有S外的其它节点w(不妨设w是这条路径上从1出发后第一个在S外的节点)。因此有$d'[w] \leq d[v] < d'[v]$。于是，算法此时应该选择w而不是v，这与算法的贪心选择相矛盾。

- 算法复杂度

——时间复杂度：$\Theta(n^2)$

——空间复杂度：$\Theta(n^2)$

定理　给出一个边具有非负权的有向图 G 和源顶点 s，算法 DIJKSTRA 在时间 $\Theta(n^2)$ 内找出从 s 到其他每一顶点距离的长度。

4.3 单源最短路

【扩展问题1】

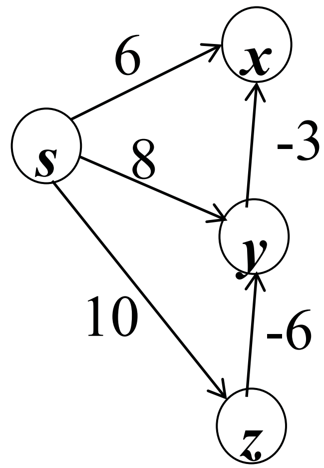
设$G=<V,E>$ 是一个有向图，图中每一条边都有一个非负长度。s,t是图G中任意两个不同点，如何求从点s到点t的最短路径?

——思路：采用Dijkstra算法，在求从s到其它所有点的最短路径的过程中只要求得从s到t的最短路径即可终止。

如果另外给定一个子集$D\subseteq V-\{s,t\}$，又如何求从点s到点t且中间须经过D中每个点一次的最短路径?

——思路：这里边权值为非负,所以最短路径中要求经过D中每个点意味着经过D中每个点恰好一次.

采用Floyd算法求出所有点对的最短距离,然后得到新图G^*，其点集为$D\cup\{s,t\}$,G^*每条边(u,v)的权值为G中相应点对(u,v)之间最短距离,最终求图G^*中从s到t的最短Hamilton路径即可。



【扩展问题2】

设$G=<V,E>$ 是一个有向图，图中每一条边都有一个长度(长度可能为负)，但图中不存在负长度的回路。s是图G中任意点，如何求从点s到其它所有点的最短路径?

——思路：动态规划法设计的算法（Bellman-Ford算法）

当图用邻接表表示时，时间复杂度为：$O(mn)$

当图用邻接矩阵表示时，时间复杂度为：$O(n^3)$

4.4 最小生成树

【最小生成树问题】

设$G=<V,E>$是一个每条边都有权的无向连通图。G的一棵生成树$<V,T>$是G的一个生成子图且该子图是一棵树，简记作T。G的权最小的生成树 T 被称作是最小生成树。

给定无向带权图$G=<V,E>$，求G的一棵最小生成树。

4.4 最小生成树

【实例】最小代价通讯网络

考虑n个城市的通讯网络建设问题。要求所有城市能互相通讯，且总的建设成本最小。

城市与城市间所有可能的通讯连接可被视作一个无向图$G=<V,E>$，图的每条边都赋予一个权值，权值表示建成由这条边所表示的通讯连接所要付出的代价。包含图中所有顶点（城市）的无回路连通子图是一个可行解，它是图的一棵生成树，其代价为该子图中边的权之和。设所有的权值都非负，要求找出代价最小的生成树。

4.4 最小生成树

- 求解思想

✓ **Prim算法**

该算法要求选择n-1条边。其贪心准则：从剩下的边中，选择一条权值最小的边，并且它的加入应使得和已经选定的边集仍然构成一棵树。

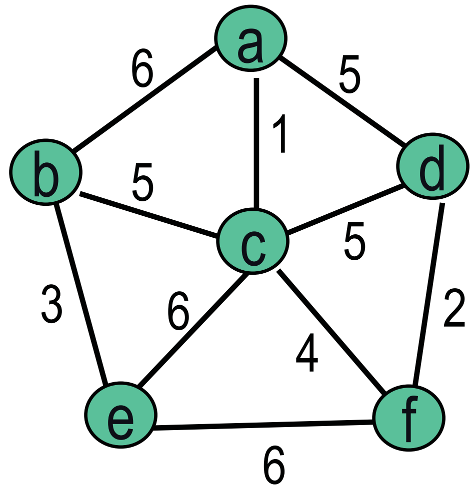
✓ **Kruskal算法（避圈法）**

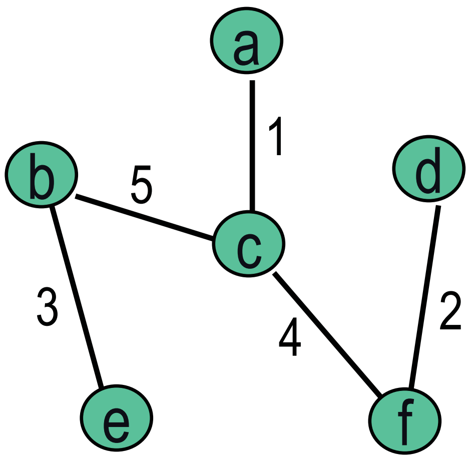
该算法要求选择n-1条边。其贪心准则：从剩下的边中选择一条和已经选定的边集不构成圈（即回路）的有最小权值的边。

✓ **破圈法**

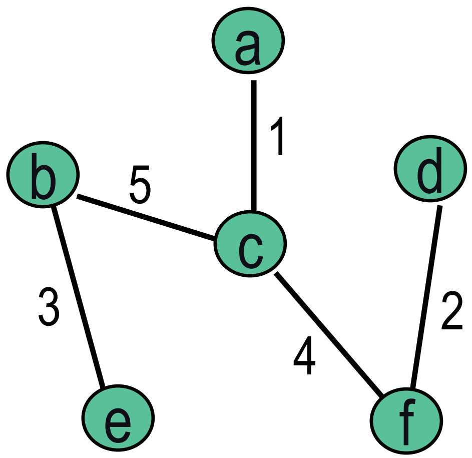
该算法是通过删除原图中的若干条边来得到最小生成树。其贪心准则：在剩下的图中任意选择一条回路，去掉其一条最大权值的边来打破该回路。

【例】 用三种方法求右图所表示的通讯网络的最小代价生成树。

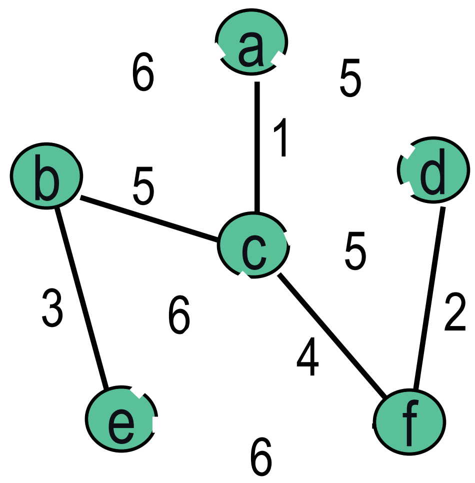




Prim算法



Kruskal算法



破圈法

4.4 最小生成树（Kruskal算法）

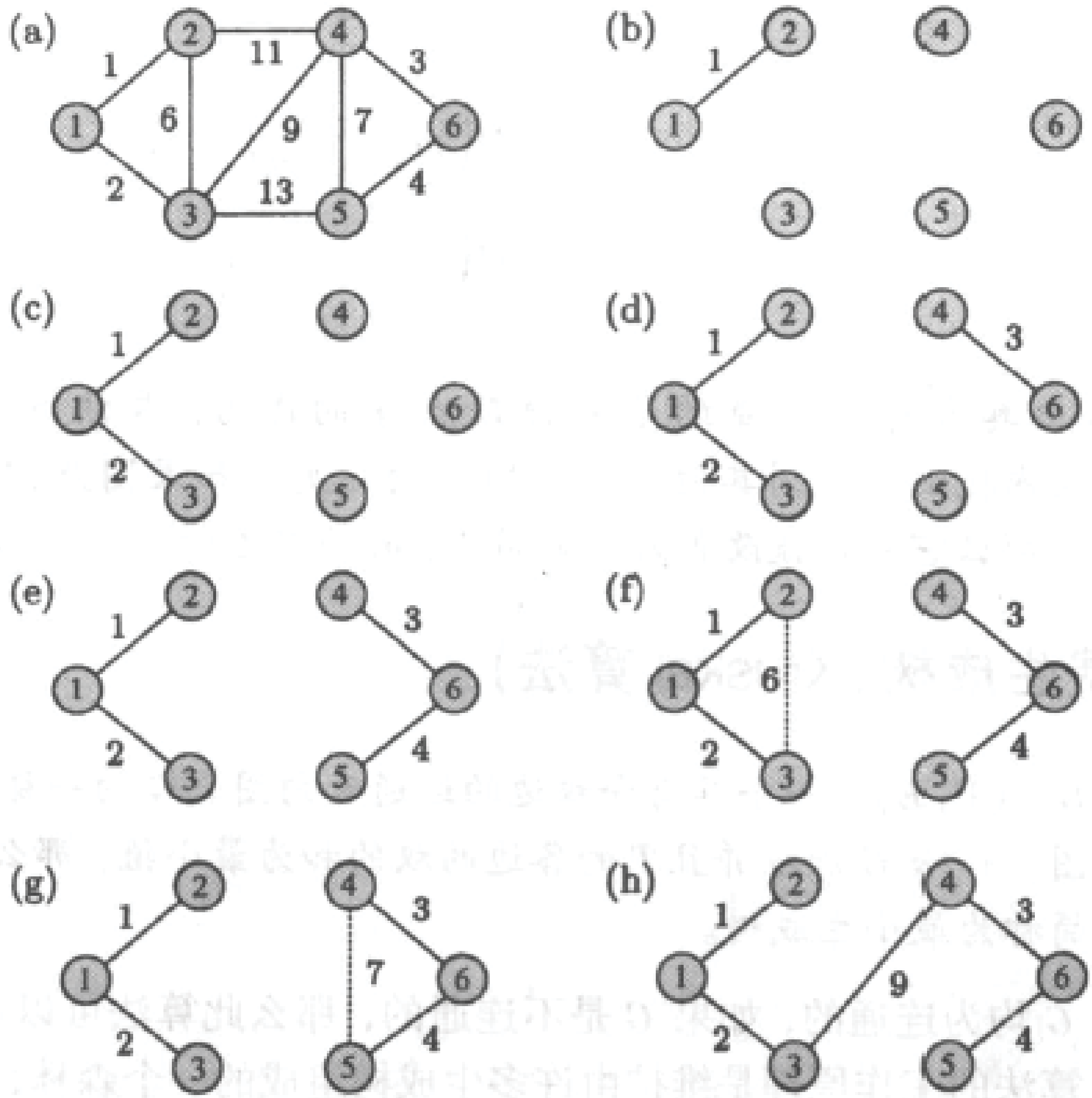
➢ 贪心准则

每次从剩下的边中选择一条和已经选定的边集不构成圈（即回路）的有最小权值的边。

➢ Kruskal算法的基本思想

（1）对G中的边按照其权值的非减次序排序；置$T=\{\}$。

（2）依次来考虑图中的每一条边是否加入生成树T中。若当前被考虑的边与T中已经有的边不构成回路，将该边加入T中；否则该边不加入T中。然后再考虑下一条边。直到所有的边都考虑完为止。



Kruskal 算法的一个例子

算法　KRUSKAL

输入：包含 n 个顶点的含权连通无向图 $G=(V,E)$。

输出：由 G 生成的最小耗费生成树 T 组成的边的集合。

```
按非降序权重将 E 中的边排序
for 每条边 v∈V
    MAKESET({v})
end for
T = {}
while |T| < n − 1
    令(x, y)为 E 中的下一条边
    if FIND(x) ≠ FIND(y) then
        将(x, y) 加入 T
        UNION(x, y)
    end if
end while
```

➢ 算法KRUSKAL的实现方法

- 用树表示不相交的集合
- 加权合并 Union(x,y)
- 压缩查找 Find(x)
- Union-Find算法

定理 设 $T(m)$表示用按秩合并和路径压缩处理 m 个合并和寻找运算的交替序列σ所需的运行时间，那么在最坏情况下 $T(m)=O(m\log^* n)$。

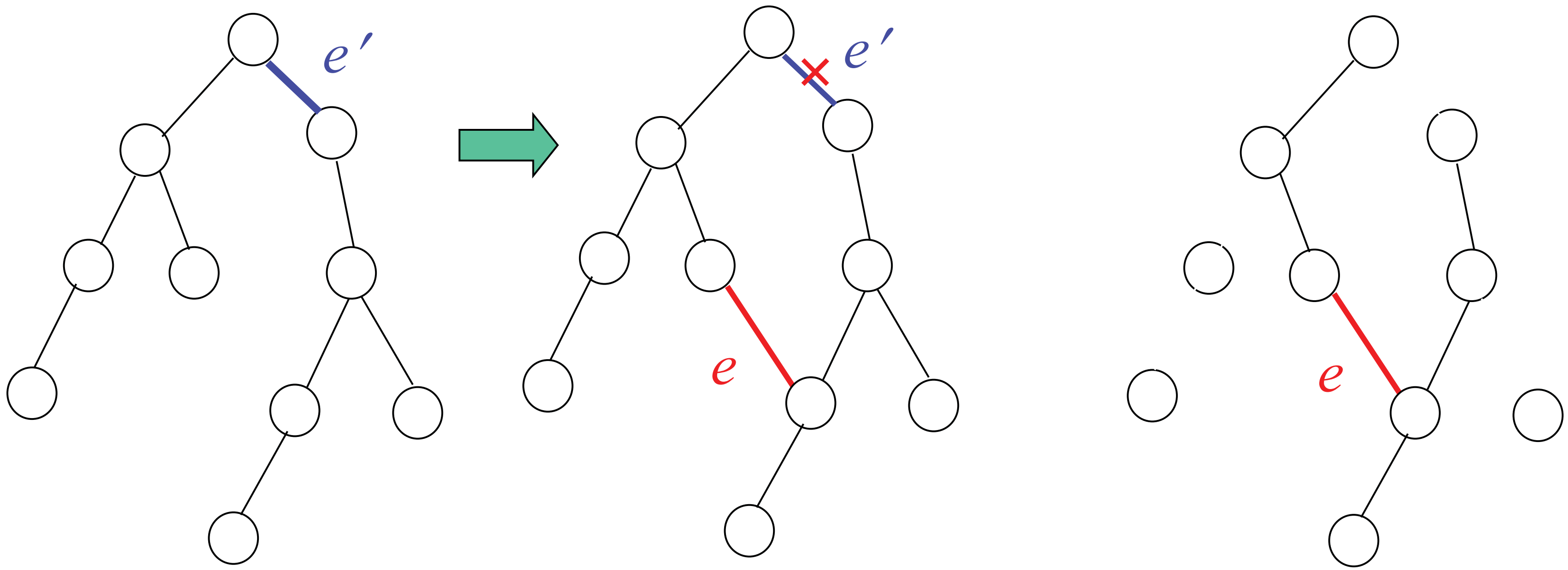
注意对于几乎所有的实际用途，$\log^* n\leqslant 5$，这说明事实上对于所有的实际应用，运行时间是 $O(m)$。

➢ 算法 KRUSKAL的正确性

定理 算法Kruskal在无向带权连通图中找到一棵最小生成树。

证明思路: 对 T的大小进行归纳证明，其中T是一棵最小生成树的边集的一个子集。

由算法Kruskal可知： $cost(e) \le cost(e')$



最小生成树（最优解）　　最小生成树（最优解）　　算法Kruskal产生的生成树（贪心解）

➢ 算法 KRUSKAL时间复杂度分析

——时间复杂度: $O(m\log m)$

- 边排序时间: $O(m\log m)$
- Find-Union时间: $O(m\log^* m)=O(m)$
- 其它操作的时间: $O(n)+O(m)=O(m)$

因此，算法时间复杂度为 $O(m\log m)$。

定理 在一个有 m 条边的含权无向图中，算法 KRUSKAL 可在 $O(m\log m)$时间内找出最小耗费生成树。

4.4 最小生成树（Prim算法）

- **贪心准则**

  每次从剩下的边中，选择一条权值最小的边，并且它的加入应使得和已经选定的边集仍然构成一棵树。

- **Prim算法的基本思想**

  (1) $T \leftarrow \{\}; X \leftarrow \{1\}; Y \leftarrow V - \{1\}$

  (2) while $Y \neq \{\}$

  (3) 设(x,y) 是一条满足$x \in X$且$y \in Y$的权值最小的边

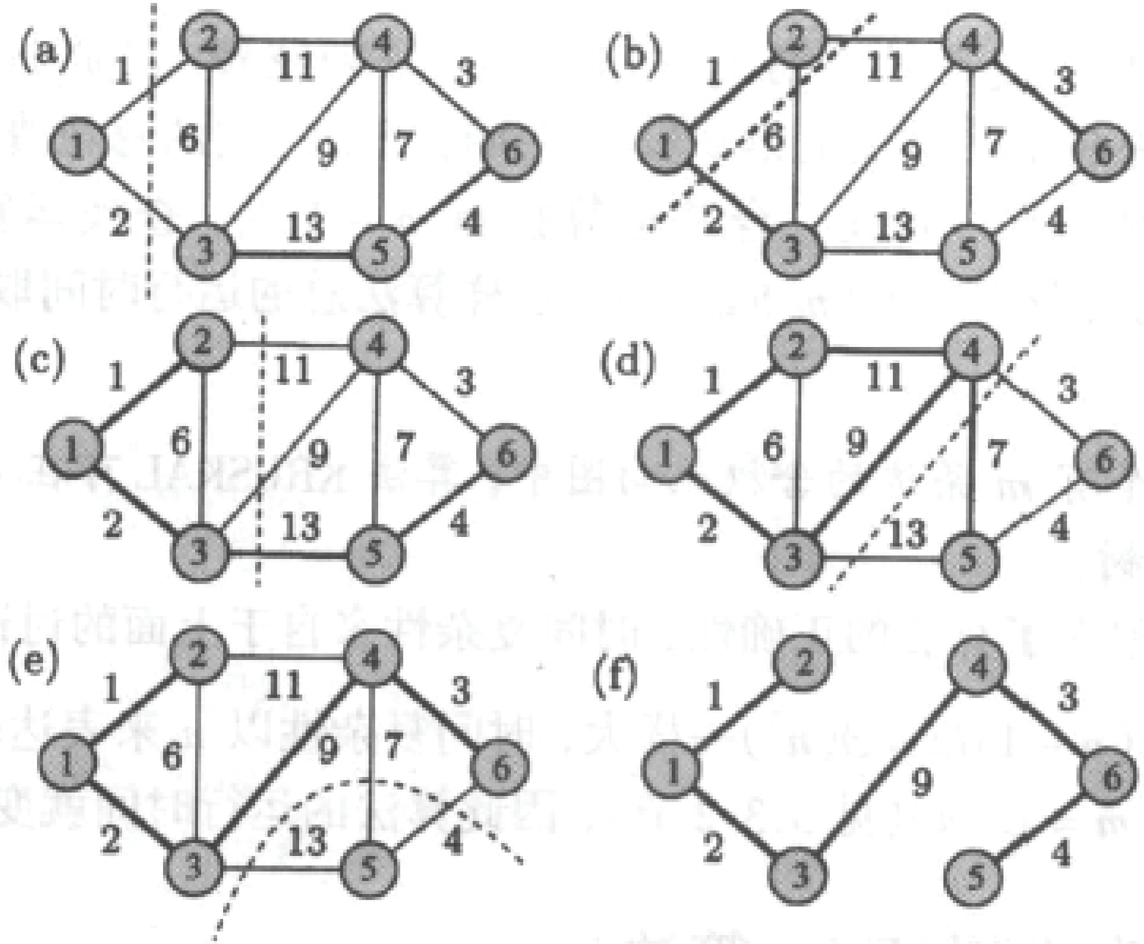
  (4) $X \leftarrow X \cup \{y\}$

  (5) $Y \leftarrow Y - \{y\}$

  (6) $T \leftarrow T \cup \{(x,y)\}$

  (7) end while

实例



Prim 算法的一个例子

算法 PRIM

输入： 含权连通无向图 $G=(V,E)$，$V=\{1,2,\cdots,n\}$。

输出： 由 G 生成的最小耗费生成树 T 组成的边的集合。

```
T ← {}; X ← {1}; Y ← V - {1}
for y ← 2 to n
    if y 邻接于 1 then
        N[y] ← 1
        C[y] ← c[1, y]
    else C[y] ← ∞
    end if
end for
for j ← 1 to n - 1   {寻找 n - 1 条边}
    令 y∈Y, 使得 C[y]最小
    T ← T ∪ {(y, N[y])}   {将边(y, N[y])加入 T}
    X ← X ∪ {y}   {将顶点 y 加入 X}
    Y ← Y - {y}   {从 Y 删除顶点 y}
    for 每个邻接于 y 的顶点 w∈Y
        if c[y, w] < C[w] then
            N[w] ← y
            C[w] ← c[y, w]
        end if
    end for
end for
```

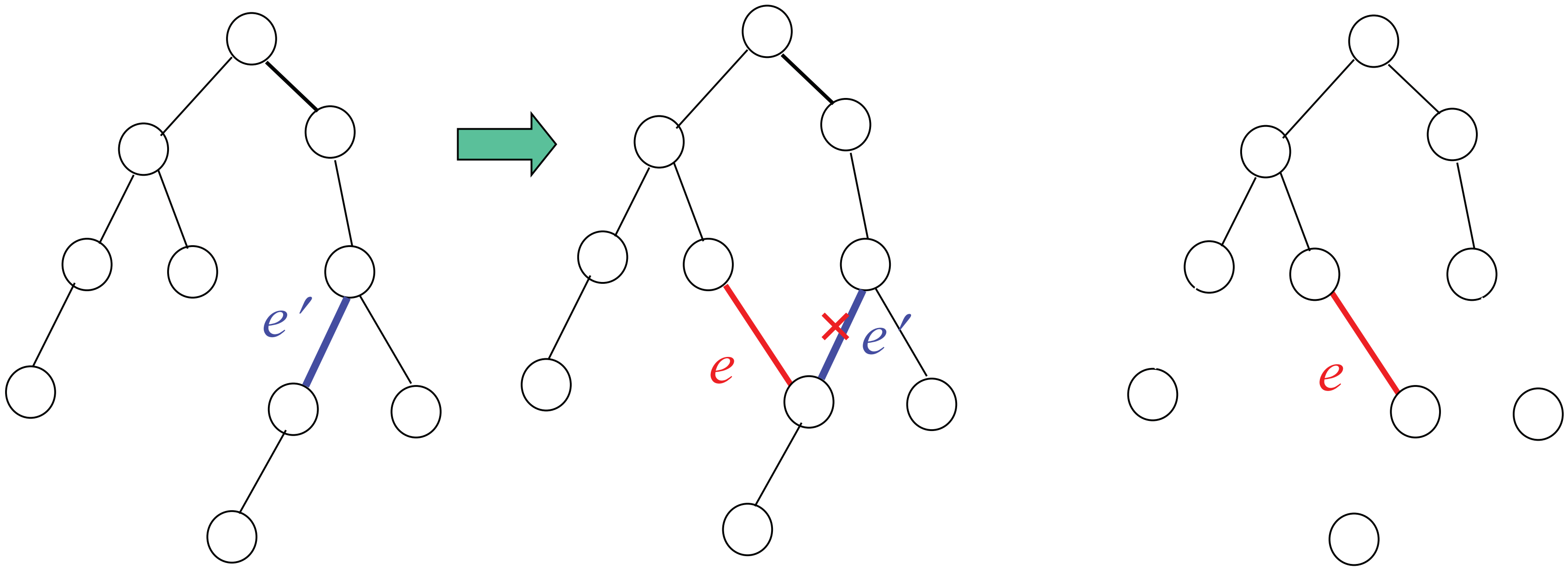
实现方法与Dijkstra算法类似

➢ 算法正确性

定理 算法PRIM在无向带权连通图中找到一棵最小生成树。

证明思路： 通过对T的大小进行归纳证明，其中T满足 (X,T)是一棵最小生成树的子树。

由算法PRIM可知：$cost(e) \le cost(e')$



最小生成树（最优解）　　最小生成树（最优解）　　算法PRIM产生的生成树（贪心解）

➢ 算法分析

——时间复杂度: $\Theta(n^2)$

定理 算法 PRIM 在一个有 n 个顶点的含权无向图中找出最小耗费生成树要用时间 $\Theta(n^2)$。

4.5 文件压缩（Huffman算法）

【文件压缩问题】

设 F 是一个字符流文件，其中的字符取自字母表 $C=\{c_1, c_2,\ldots, c_n\}$。设 $f(c_i)$ 表示字符 c_i 出现在文件中的频率，$1\le i\le n$。希望用某种方式将文件压缩得尽可能小，且易于根据压缩文件恢复得到原文件。

4.5 文件压缩（Huffman算法）

- **直接算法**

  使用定长码给每个字符编码。

- **哈夫曼编码（Huffman算法）**

  使用变长码给字符编码，使得出现频率高的字符的编码尽可能短。

➢ 实例

假设有一个数据文件包含100 000个字符，我们要用压缩的方式来存储它。该文件中各个字符出现的频率如下表所示。

	a	b	c	d	e	f
频率（千次）	45	13	12	16	9	5
定长码	000	001	010	011	100	101
变长码	0	101	100	111	1101	1100

➢ 求解思路

要压缩表示这个文件中的信息有多种方法。我们考虑用0，1码串来表示字符的方法，即每个字符用唯一的一个0，1串来表示。问题要求使用二进制数字0，1码串来编码，保证编码是前缀码的前提下使得压缩文件最短。（为了使得译码容易，编码必须具有性质：任一字符的代码都不是其它字符代码的前缀。我们称这样的编码为**前缀码**。）

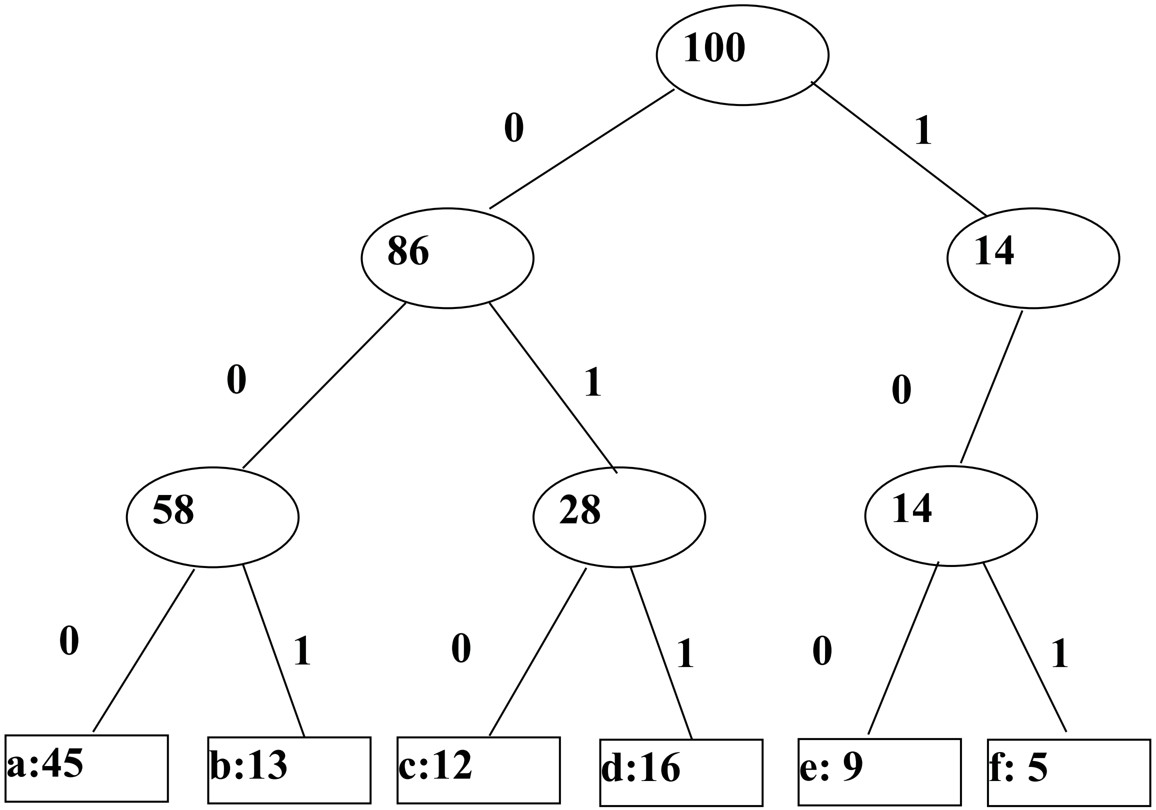
	a	b	c	d	e	f
频率（千次）	45	13	12	16	9	5
定长码	000	001	010	011	100	101
变长码	0	101	100	111	1101	1100

- ✓ 使用定长码，则表示每个不同的字符最少需要3位。用这种方法对整个文件编码需要300 000位。
- ✓ 给这个文件编码最少需要多少位呢?

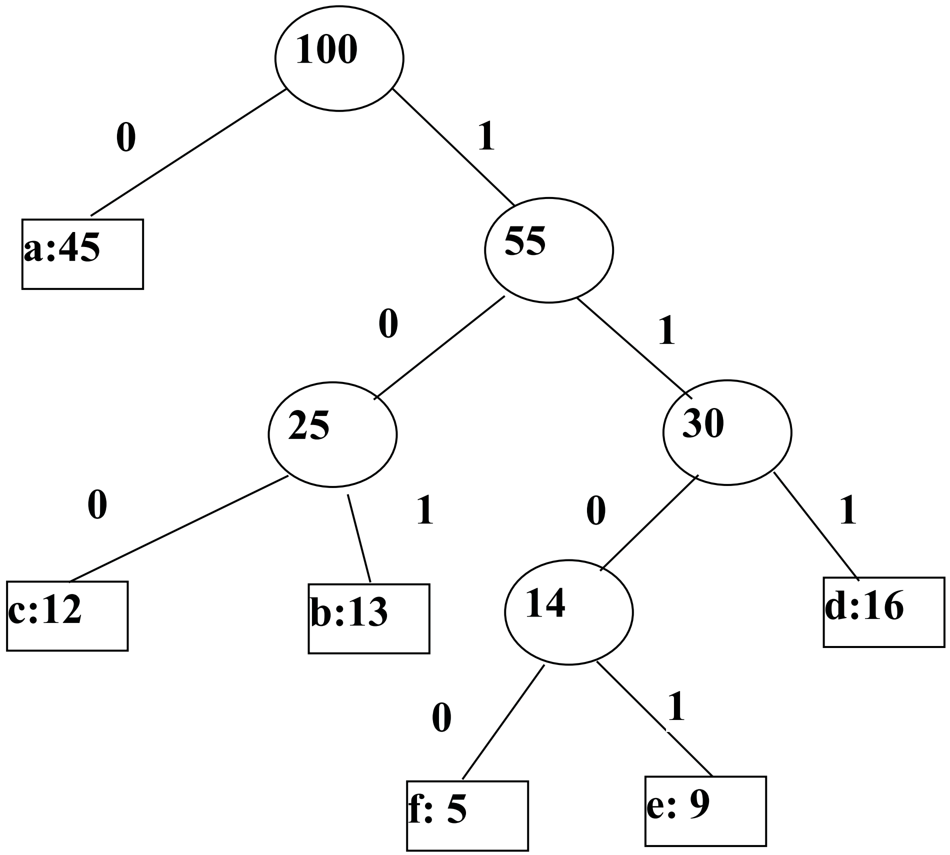
  ——使用上表中的变长码给文件编码需要224 000位，它比定长码减少了约25%。事实上，这是该文件的一个最优编码方案。
- ✓ 如何得到最优编码?

  ——使用哈夫曼算法。

任何一个前缀码都对应一棵二叉树。反之，任意一棵二叉树的树叶可对应一个前缀码。例如，表中的两个前缀码对应的二叉树为下图。显然，最优编码应该满足：出现频率高的字符应该使用较短的编码。根据该例可描述构造最优前缀码的方法——**哈夫曼算法。**



定长编码的构造过程



哈夫曼编码的构造过程

➢ 文件压缩问题的数学模型

求一棵二元树使得外部加权路径之和最小（画示意图）

$$\min_{Tree} \sum_{i=1}^{n} l_i w_i$$

算法 HUFFMAN

输入: n 个字符的集合 $C=\{c_1,c_2,\cdots,c_n\}$ 和它们的频度 $\{f(c_1),f(c_2),\cdots,f(c_n)\}$。

输出: C 的 Huffman 树 (V,T)。

```
根据频度将所有字符插入最小堆 H
V ← C; T = {}
for j ← 1 to n - 1
    c ← DELETEMIN(H)
    c′ ← DELETEMIN(H)
    f(v) ← f(c) + f(c′)    {v 是一个新节点}
    INSERT(H, v)
    V = V ∪ {v}    {添加 v 到 V}
    T = T ∪ {(v, c), (v, c′)}    {使 c 和 c′成为 T 中 v 的孩子}
end while
```

- **算法正确性**

  （略）

- **算法分析**

  ——时间复杂度： **O($n$log n)**

小结

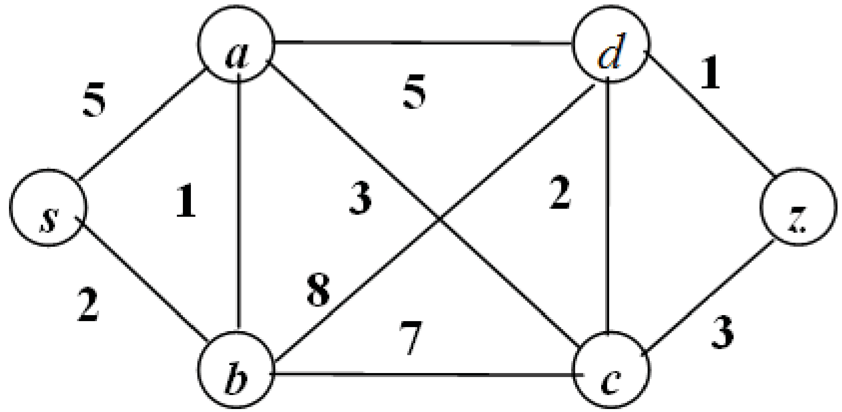
- 贪心算法的特点

  ——设计简单、直观，正确性需要证明

- 贪心算法正确性证明方法

  ——证明最优解可用贪心法逐步构造得到（数学归纳法）

拓展应用举例



【例1】单源最大带宽路径问题

设加权无向图$G=\langle V, E\rangle$表示一个电话网络，其顶点表示交换站，边表示两个交换站之间的通信线路。每条边(u,v)上权$length[u,v]$表示其带宽。一条路径的带宽定义为其上带宽最小的边上的带宽。例如，图中从s到d的一条路径s,a,c,z,d的带宽为1，另一条路径s,a,c,d的带宽为2。

给定一个电话网络图$G=\langle V, E\rangle$，要计算从网络中的一个交换站s（称为源点）到其它所有交换站之间的最大带宽的路径。

$d[v]$ 表示源点s=1只经过S中的结点到达点v的最大带宽路径的带宽。算法中每当选择$d'[v]$最大的v加入到S时，即求得源点s到v的最大带宽路径的带宽为$d[v]$。

Dijkstra 算法(G, l)

设 S 是被探查的结点的集合

对每个 $u \in S$，我们存储一个距离 $d(u)$

初始 $S = \{s\}$ 且 $d(s) = 0$

$d[s] \leftarrow \infty$

While $S \neq V$

选择一个结点 $v \notin S$ 使得从 S 到 v 至少有一条边并且

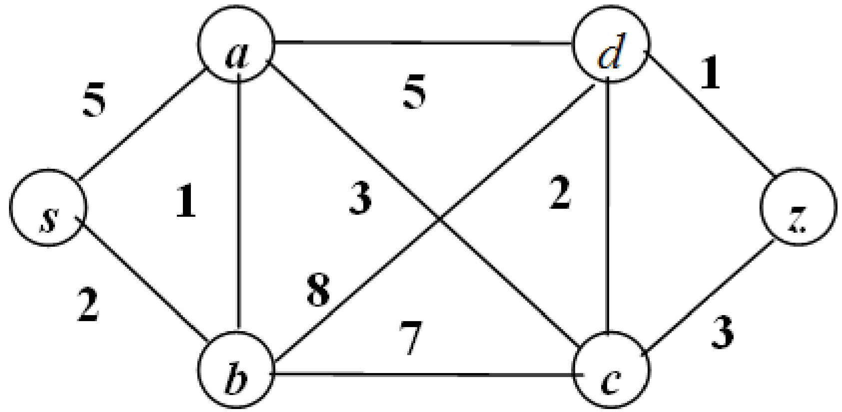
$d'(v) = \min_{e=(u,v):u \in S} d(u) + l_e$ 最小

将 v 加入 S 并且定义 $d(v) = d'(v)$

Endwhile

$d'[v] \leftarrow \mathbf{max}_{e[u,v]:\, u \in S}\ \mathbf{min}\{d[u], l_e\}$

拓展应用举例

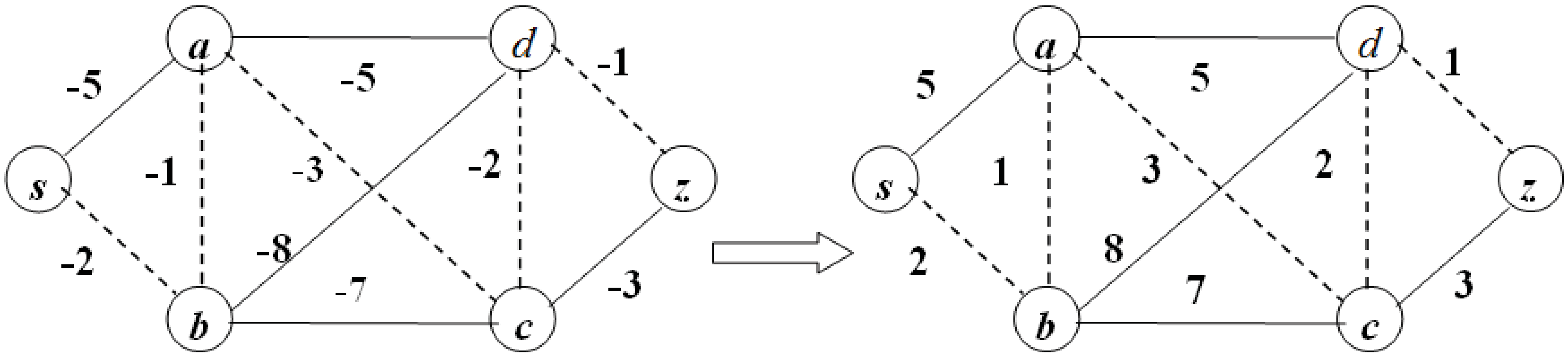


【例2】任意点对间的最大带宽路径问题

设加权无向图G=⟨V, E⟩表示一个电话网络中任意两点间的最大带宽路径可通过求图G的最大生成树得到：

图G的最大生成树中任意两点x, y间的唯一路径即为图G中两点x, y间的一条最大带宽路径。

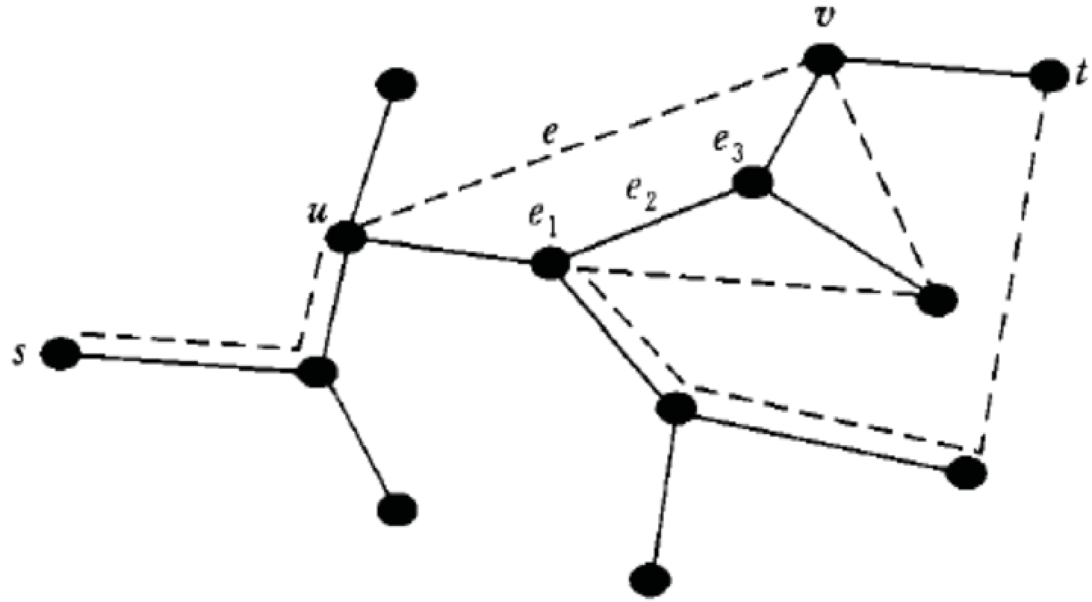
修改PRIM算法可得上图所示带权图的一棵最大权生成树如下：



【例2】任意点对间的最大带宽路径问题

设加权无向图G=⟨V, E⟩表示一个电话网络中任意两点间的最大带宽路径可通过求图G 的最大生成树得到:

图G的最大生成树中任意两点x, y间的唯一路径即为图G中两点x, y间的一条最大带宽路径。



最大生成树和最大带宽路径(其中实线表示最大生成树,虚线表示最大带宽路径)

[1] 陈建二等.关于实际构造最大带宽路径算法的研究. 计算机学报, 2002,25(10): 1116-1120.

拓展应用举例

【例3】最大间隔聚类问题

最大间隔聚类　最小生成树在我们这里描述的一种最基本的形式化中起了作用.假设给定集合 U,其 n 个个体标记为 $p_1,p_2,\cdots,p_n$. 对于每对个体 p_i 与 p_j,有一个数值距离 $d(p_i,p_j)$. 我们只要求 $d(p_i,p_i)=0$;对于不同的 p_i 与 p_j,$d(p_i,p_j)>0$;并且这个距离是对称的:$d(p_i,p_j)=d(p_j,p_i)$.

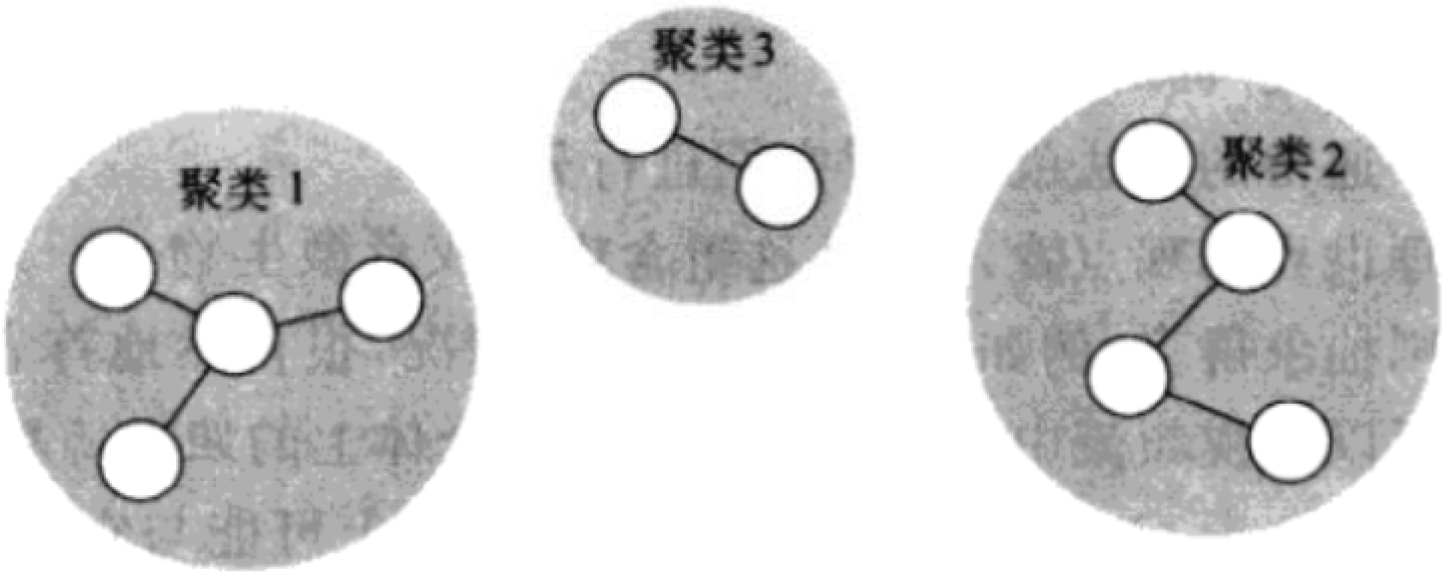
对于给定的参数 k,假设我们正在寻求将 U 中的个体划分成 k 组.我们说一个 U 的 k 聚类是一个把 U 分成 k 个非空集合 $C_1,C_2,\cdots,C_k$ 的划分.我们定义一个 k 聚类的间隔是处在不同聚类中的任何一对点之间的最小距离.我们希望在不同聚类中的点尽可能地相互远离,给定这个前提,一个自然的目标是寻求具有最大可能间隔的 k 聚类.

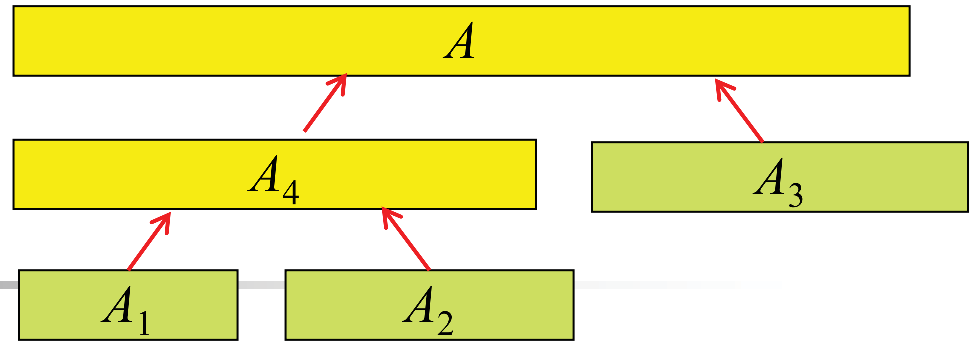
现在就有了下面的问题.一个集合有指数多个不同的 k 聚类;我们怎么能有效地找出有最大间隔的聚类?

【例3】最大间隔聚类问题

与最小生成树的联系是什么？这是非常简单的：尽管我们的图的长出过程是由聚类合并的思想引起的，但我们的过程正好就是 **Kruskal 最小生成树算法**. 如果给定一个 U 上的图，其中每对结点 (p_i, p_j) 之间存在一条费用为 $d(p_i, p_j)$ 的边，**Kruskal 算法**所做的恰好就是我们现在正在做的. 唯一的区别就是我们寻找一个 k 聚类，因此一旦得到了 k 个连通分支我们就停止这个过程.

换句话说，我们正在运行 **Kruskal 算法**，但是就在它加最后的 $k-1$ 条边之前停止. 这等价于取整棵最小生成树 T(好像 **Kruskal 算法**已经把它产生了)，删除 $k-1$ 条最贵的边(我们从来没有真正把它们加上)，并且定义 k 聚类是所得到的连通分支 $C_1, C_2, \cdots, C_k$. 于是，反复地合并聚类等价于计算一棵最小生成树并且删除这些最贵的边.





拓展应用举例

【例4】最优二路归并问题

设 $A_1, A_2, \cdots, A_m$ 是 m 个已按非降序排列的整数数组，每个数组 A_j 的大小是 n_j。假设我们想用与 1.4 节中描述的算法 MERGE 类似的算法把所有的数组合并成一个数组。用贪心策略给出一个合并数组的序，使所有合并中元素移动次数最少。例如，如果 $m=3$，我们可以合并 A_1 和 A_2 得到 A_4，然后合并 A_3 和 A_4 得到 A。另一种方法是合并 A_2 和 A_3 得到 A_4，然后合并 A_1 和 A_4 得到 A。还有另一种方法是合并 A_1 和 A_3 得到 A_4，然后合并 A_2 和 A_4 得到 A（提示：给出一个类似算法 HUFFMAN 的算法）。

由于二路归并问题的数学模型与文件压缩问题的数学模型相同，因此可用哈夫曼算法求解最优二路归并问题。