



贪心算法

东南大学计算机学院 方效林

本章内容

- 贪心算法要素
- 活动选择问题
- 哈夫曼编码问题
- 最小生成树问题
- 单源最短路径问题

贪心算法要素

■ 贪心算法的基本思想

- 求解最优化问题的算法包含一系列步骤
- 每一步都有一组选择
- 作出在当前看来最好的选择
- 希望通过作出局部最优选择达到全局最优选择
- 贪心算法不一定总产生最优解
- 贪心算法是否产生优化解，需严格证明

■ 贪心算法产生最优解的条件

- 最优子结构
- 贪心选择性

贪心算法要素

■ 最优子结构

- 当一个问题最优解包含子问题的最优解时，称这个问题具有最优子结构

■ 贪心选择性

- 当一个问题全局最优解可以通过局部最优解得到，称这个问题具有贪心选择性
- 证明思路：
 - 假定首选元素不是贪心选择所要的元素，证明将首选元素替换成贪心选择所需元素，依然得到最优解
 - 数学归纳法证明每一步均可通过贪心选择得到最优解

贪心算法要素

- 动态规划方法可用的条件
 - 最优子结构
 - 子问题重叠性
- 贪心算法产生最优解的条件
 - 最优子结构
 - 贪心选择性
- 适用贪心算法时，动态规划可能不适用
- 适用动态规划时，贪心算法可能不适用

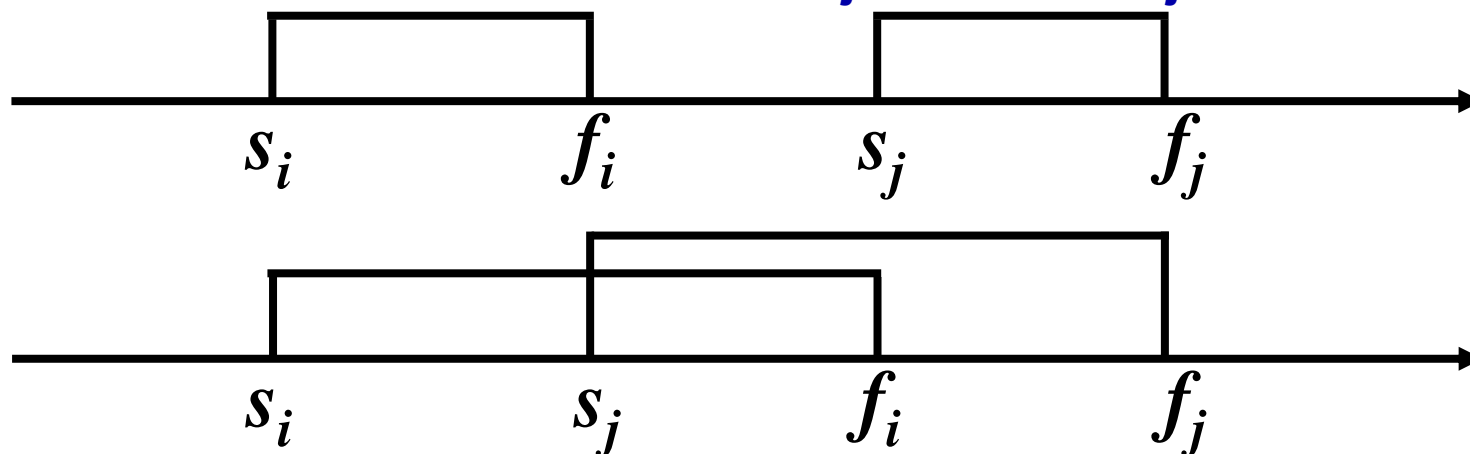
活动选择问题

■ 活动

- $S=\{1,2,\dots,n\}$ 是 n 个活动的集合，活动共用同一资源，同一时间只有一个活动使用。活动 i 有起始时间 s_i ，终止时间 f_i ， $s_i \leq f_i$ ，表示为 $x_i=[s_i, f_i]$

■ 相容活动

- 活动 i 和 j 是相容的，若 $s_j \geq f_i$ 或 $s_i \geq f_j$



活动选择问题

■ 问题定义

- 输入: $S=\{1, 2, \dots, n\}$, $x_i=[s_i, f_i]$, $1 \leq i \leq n$
- 输出: S 的最大相容集合

■ 贪心思想

- 为了选择更多活动, 每次选择 f_i 最小的活动

活动选择问题

S按结束时间排序, $f_1 \leq f_2 \leq \dots \leq f_n$

Greedy-Activity-Selector(S, F)

n = length(S);

A = {1};

j = 1;

for i=2 to n do

if $s_i \geq f_j$ then

A = A \cup {i};

j=i;

return A

$$\begin{aligned} T(n) &= \theta(n) + \theta(n \log n) \\ &= \theta(n \log n) \end{aligned}$$

活动选择问题

■ 最优子结构性质

- 设活动 $S=\{1, 2, \dots, n\}$ 已按结束时间递增排序, 即 $f_1 \leq f_2 \leq \dots \leq f_n$, 设 A 是包括活动 1 的最优解, 则 $A'=A-\{1\}$ 是 $S'=\{i \in S | s_i \geq f_1\}$ 的最优解。
- 证明:
 - 显然 A' 中的活动是相容的, 只需证 A' 是最大的。
 - 若不然, 假设 B' 是最大的, 且 $|B'| > |A'|$ 。
 - 那么 $B=\{1\} \cup B'$ 是最优解, 但 $|B| = 1 + |B'| > 1 + |A'| = |A|$
 - 这与 A 是最大的 (最优解) 矛盾。

问题的最优解包含子问题的最优解

活动选择问题

■ 贪心选择性

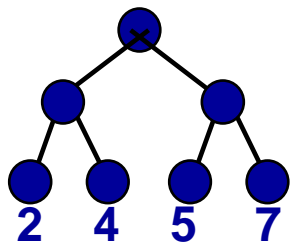
- 设活动 $S=\{1, 2, \dots, n\}$ 已按结束时间递增排序, 即 $f_1 \leq f_2 \leq \dots \leq f_n$ 。每次选结束时间最小的相容活动, 可得最优解 A 。
- 证明:
 - 设贪心最优解 A 也按结束时间递增排序, 设其第一个活动为 k , 第二个活动为 j
 - 若 $k=1$, 则成立
 - 若 $k \neq 1$, 由于 A 中活动相容, 有 $f_k \leq s_j$, 由于 $f_1 \leq f_k$, 因此, 可以用活动 1 代替活动 k

哈夫曼编码

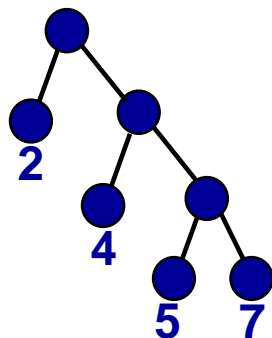
■ 带权路径长度

- 假设二叉树中每个叶结点有一个权值 w_i ，到根的路径长度为 l_i ，其他结点权值为0，则有n个叶子结点的树的带权路径长度为

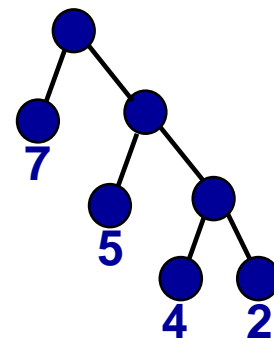
$$WPL = \sum_{i=0}^{n-1} w_i * l_i$$



$$WPL = 2 * (2 + 4 + 5 + 7) = 36$$



$$WPL = 2 + 2 * 4 + 3 * (5 + 7) = 46$$



$$WPL = 7 + 2 * 5 + 3 * (4 + 2) = 35$$

带权路径长度达到最小的二叉树即为Huffman树。
在Huffman树中，权值越大的结点离根越近。

哈夫曼编码

- 构造权值为 $\{w_1, w_2, \dots, w_n\}$ 的Huffman树
 - 构造 n 棵二叉树的森林 $F=\{T_1, T_2, \dots, T_n\}$, 每棵二叉树 T_i 只有一个带权值为 w_i 的根结点
 - 重复以下步骤, 直到只剩一棵树为止:
 - 1. 在 F 中选两棵根结点权值最小的二叉树, 作为左、右子树构造一棵新的二叉树, 新树的根结点权值等于其左、右子树根结点权值之和
 - 2. 在 F 中删除这两棵二叉树
 - 3. 把新构造的二叉树加入 F

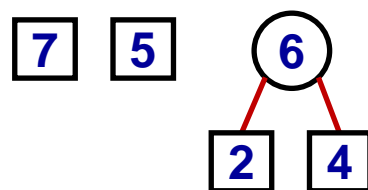
哈夫曼编码

F: {7} {5} {2} {4}



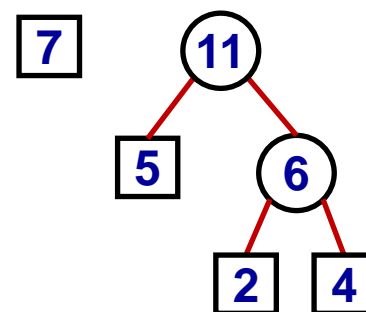
(a) 初始

F: {7} {5} {6}



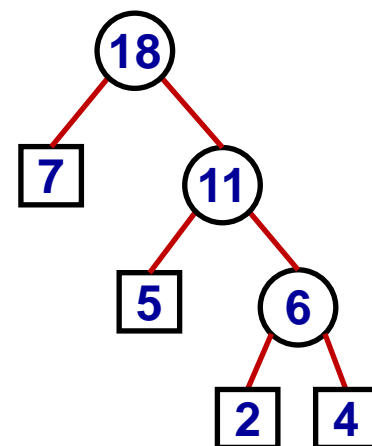
(b) 合并{2}{4}

F: {7} {11}



(b) 合并{2}{4}

F: {18}



(b) 合并{7}{11}

哈夫曼编码

- Huffman编码，进行数据压缩
 - 计算机领域数据用二进制表示
 - 若已统计出某文本中各字符出现的概率，可以用Huffman编码进行数据压缩

$$\text{概率} = \frac{\text{某字符出现次数}}{\text{总字符数}}$$

哈夫曼编码

■ Huffman编码，进行数据压缩

- 假设某文本共有1000个字符，且只由 a, b, c, d, e 5种字符组成
 - 固定长度编码可将每个字符用3比特表示，整个文本要 $3 \times 1000 = 3000$ 比特表示，平均编码长度3比特

符号	定长编码
a	000
b	001
c	100
d	101
e	110

哈夫曼编码

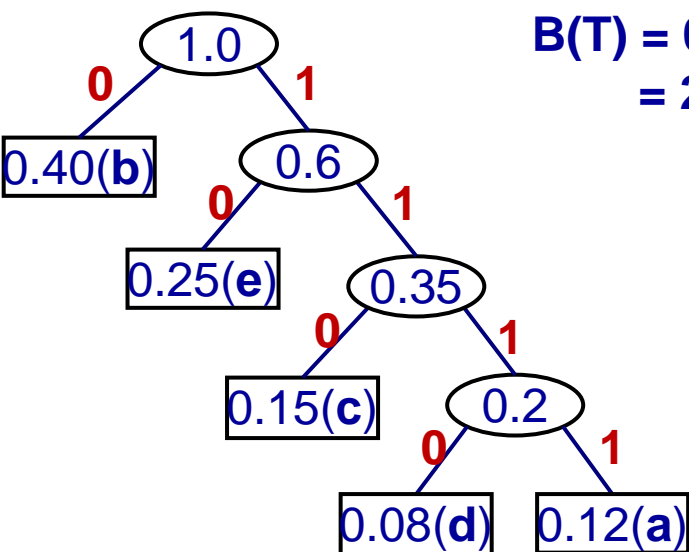
■ Huffman编码，进行数据压缩

□ 假设某文本共有1000个字符，且只由 a, b, c, d, e 5种字符组成

➤ 若已统计出各字符出现的概率分别为0.12, 0.40, 0.15, 0.08, 0.25，则整个文本可用2150比特表示

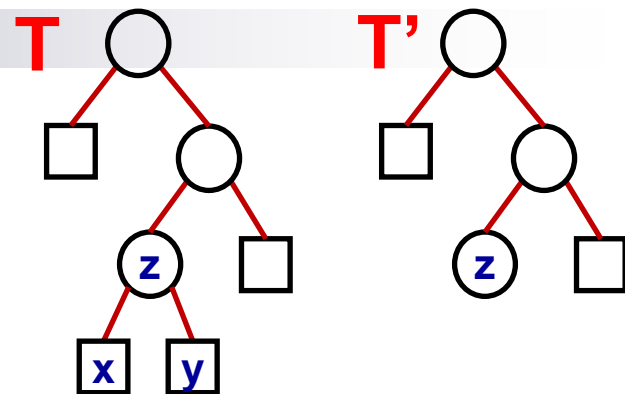
平均编码长度(带权路径长度)

$$B(T) = 0.12 \times 4 + 0.08 \times 4 + 0.15 \times 3 + 0.25 \times 2 + 0.40 \times 1 \\ = 2.15$$



符号	概率	Huffman编码
a	0.12	1111
b	0.40	0
c	0.15	110
d	0.08	1110
e	0.25	10

哈夫曼编码

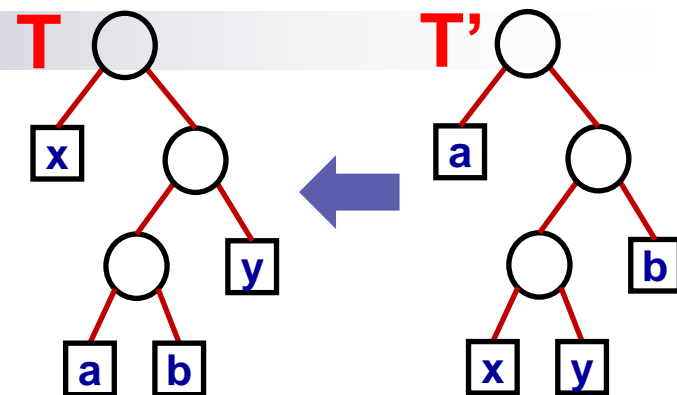


■ 最优子结构

- T是字符表C一棵哈夫曼树。设字符x和y是T中任意两个相邻叶结点，z是其父结点， $f(x)$ 和 $f(y)$ 是其频率。将z看作频率是 $f(z)=f(x)+f(y)$ 的字符， $T'=T-\{x,y\}$ 是字母表 $C'=C-\{x,y\}\cup\{z\}$ 的哈夫曼树
- 证明：
 - 若T'不是C'的哈夫曼树(最优解)，则必存在T''，使 $B(T'')<B(T')$ 。因为z是C'中字符，它必为T''中的叶子。把结点x与y加入T''，作为z的子结点，得到C的哈夫曼树T'''， $B(T''')=B(T'')+f(x)+f(y)<B(T')+f(x)+f(y)<B(T)$ 。这与T是C的哈夫曼树矛盾。

问题的最优解包含子问题的最优解

哈夫曼编码



■ 贪心选择性

- 设C是字符表，字符x和y是C中频率最小的两个字符，则存在一棵哈夫曼树，使得x与y编码长度最长(即到根路径长度最长)
- 证明：
 - 假设一哈夫曼树编码长度最长的不是x和y，是a和b，且 $f(a) \geq f(x)$, $f(b) \geq f(y)$ ，则 $L(a) \geq L(x)$, $L(b) \geq L(y)$ ，L指编码长度，如左图所示。分别将a和x，b和y交换，得到
 - $B(T') = B(T) - f(x)L(x) - f(y)L(y) - f(a)L(a) - f(b)L(b)$
 - $+ f(x)L(a) + f(y)L(b) + f(a)L(x) + f(b)L(y)$
 - $= B(T) + (f_x - f_a)(L_a - L_x) + (f_y - f_b)(L_b - L_y) \leq B(T)$

优先选择频率最小的x和y结果并不会变坏

并查集

- 互不相交的集合 (等价类的划分)
 - 需要经常合并集合 (等价类的合并)
 - 需要经常查找元素属于哪个集合

并查集

- 并查集支持以下操作
 - 初始化 $\text{UFSets}(s)$: 将 s 中每个元素自成一个集合
 - 合并 $\text{Union}(\text{Root1}, \text{Root2})$: 集合 Root2 并入 Root1
 - 查找 $\text{Find}(x)$: 查找元素 x 属于哪个集合

并查集

■ 用树表示集合

- 初始化时，每个元素自成为一棵树



全集合S初始化时形成的森林

下标	0	1	2	3	4	5	6	7	8	9
父指针	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

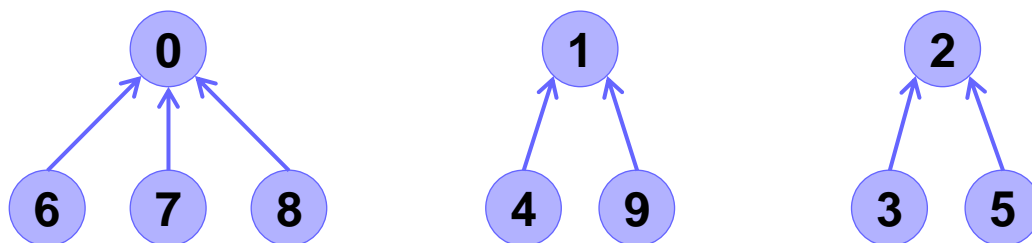
初始化时形成的父指针表示

负数表示没有父结点
-1表示树中有1个结点
-2表示有2个结点，
以此类推

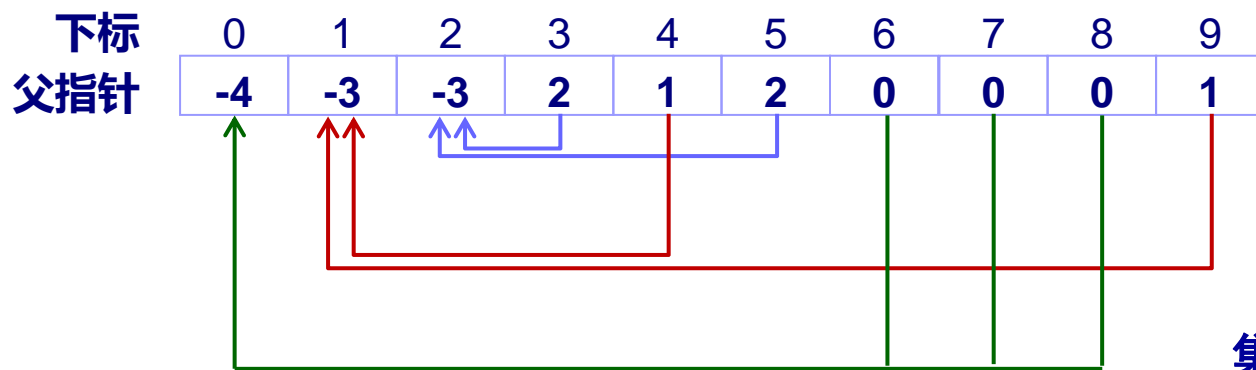
并查集

■ 用树表示集合

- 假设经过若干合并后有3个集合{0,6,7,8}, {1,4,9}, {2,3,5}



集合的树形表示



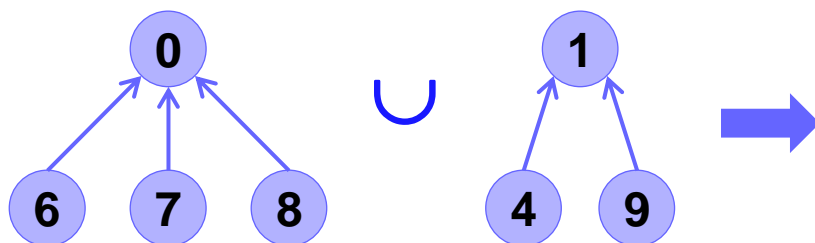
集合的父指针表示

并查集

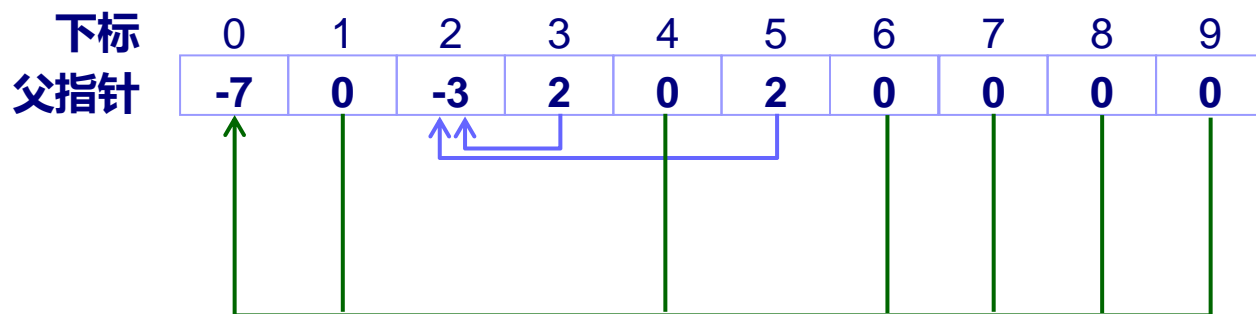
■ 用树表示集合

□ 合并 $s_1 = \{0, 6, 7, 8\} \cup s_2 = \{1, 4, 9\}$

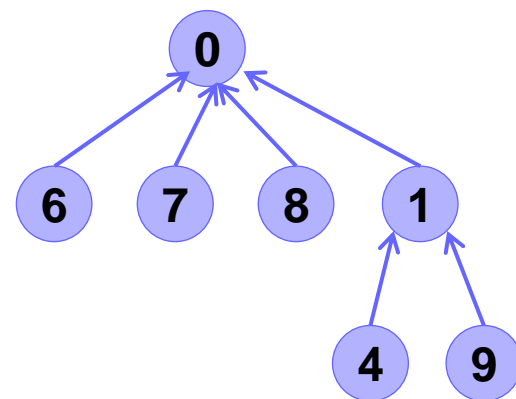
➤ s_2 作为 s_1 根的子树，或相反



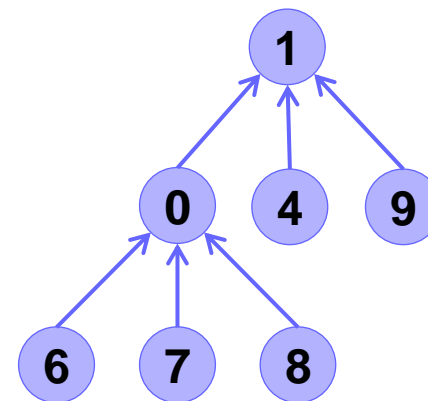
合并集合



第一种合并的父指针表示



第一种合并方式



第二种合并方式

并查集

■ 用树表示集合

□ 合并 $s_1=\{0,6,7,8\} \cup s_2=\{1,4,9\}$

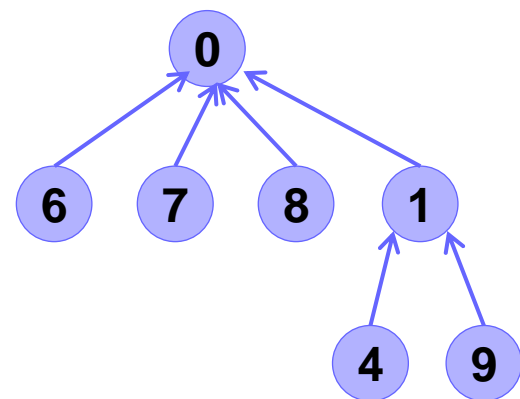
➤ s_2 作为 s_1 根的子树，或相反

➤ 哪一种合并方式好？

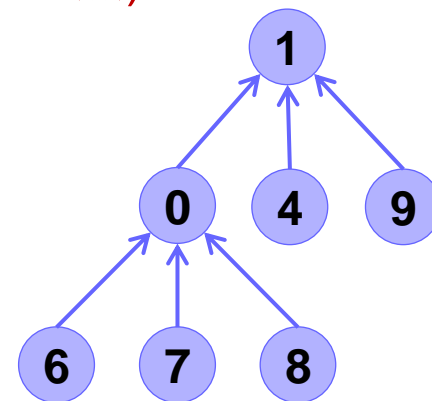
➤ 先介绍查找Find操作

□ Find(4)表示4所属集合，返回值0号结点(即根结点)

□ Find操作时间相当于结点到根路径长度



第一种合并方式



第二种合并方式

第一种方式好：查找时间更优

合并方式：

将结点数更少的集合作为结点数更多的集合的根的子树

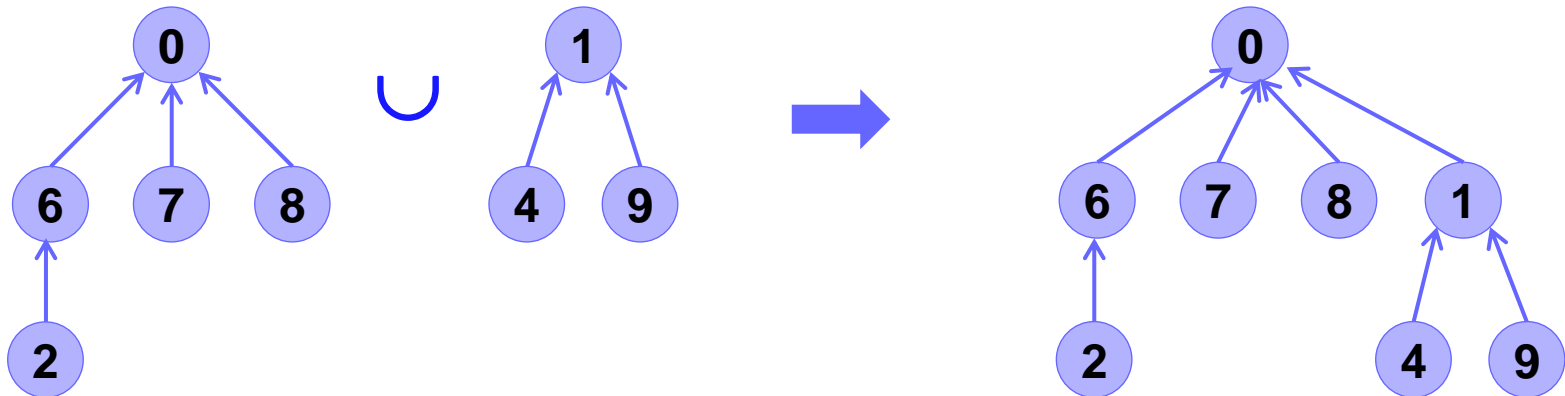
并查集

■ 用树表示集合

□ 合并 $s_1 = \{0, 2, 6, 7, 8\} \cup s_2 = \{1, 4, 9\}$

➤ 令秩表示树的深度

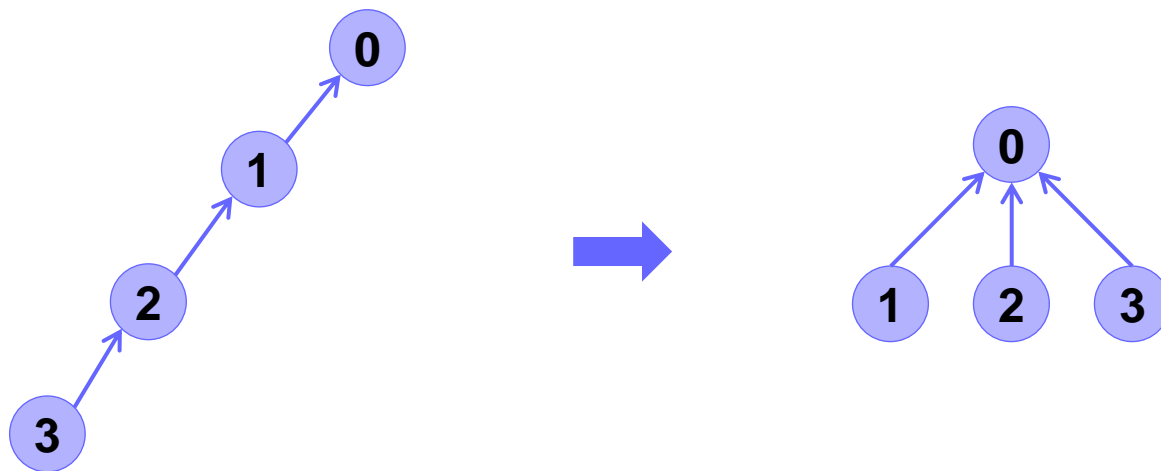
➤ 将秩更小的树作为秩更大的树的根的子树



并查集

■ 用树表示集合

- 路径压缩，在find过程中顺便压缩



按秩合并与路径压缩方式，
操作的平均代价为 $O(\alpha(n))$

$\alpha(n)$ 是 $n = f(x) = A(x, x)$ 的反函数
其中 A 是急速增加的阿克曼函数
 $\alpha(n)$ 在 n 十分巨大时还是小于 5.

最小生成树

■ 生成树

- 设 $G=(V, E)$ 是一个边加权无向连通图。 G 的生成树是无向树 $S=(V, T)$, $T \subseteq E$ 。 W 是 G 的权函数, T 权值定义为 $W(T) = \sum_{(u,v) \in T} W(u,v)$

■ 最小生成树

- G 的最小生成树是 $W(T)$ 最小的 G 的生成树

■ 问题的定义

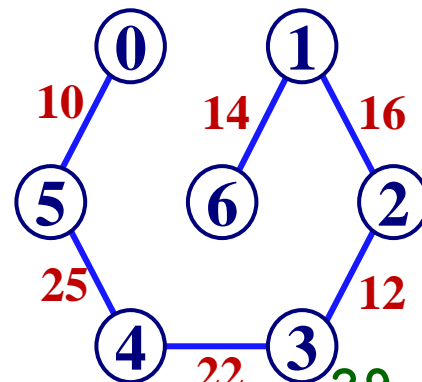
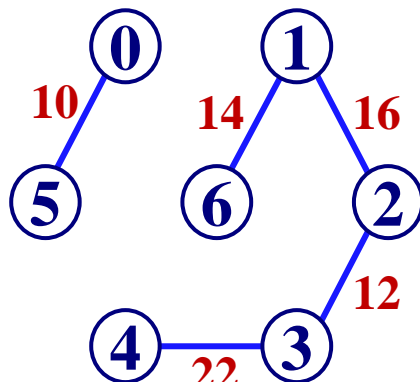
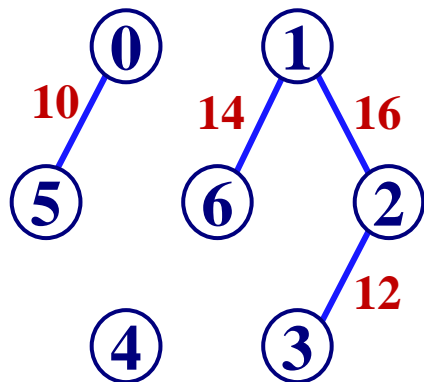
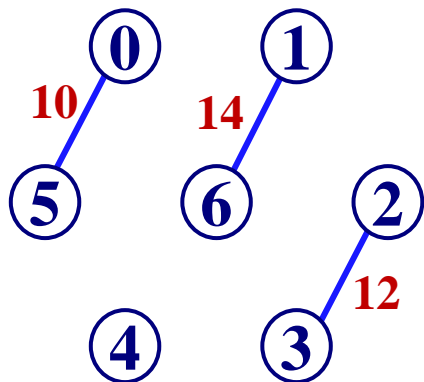
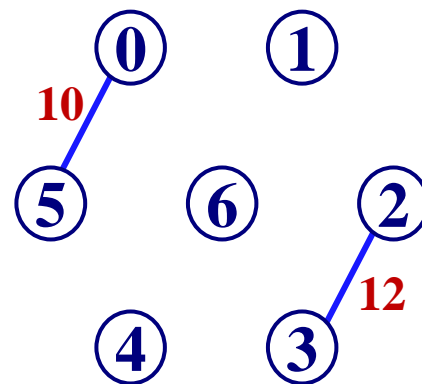
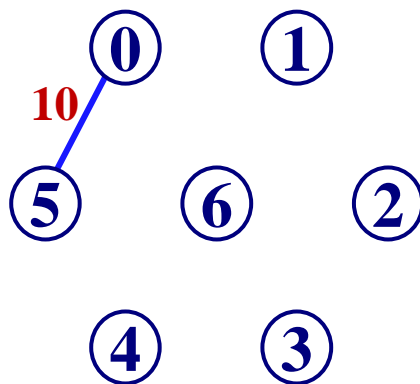
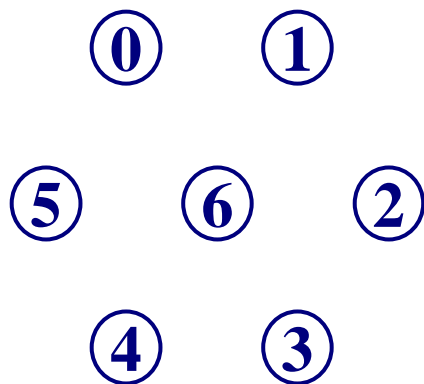
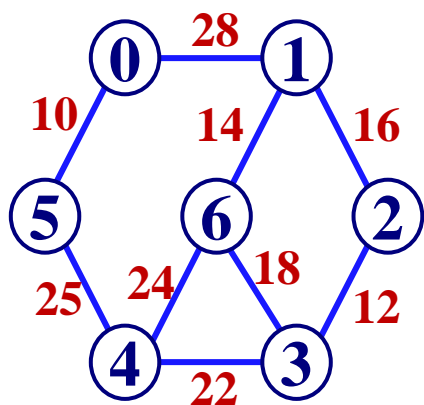
- 输入: 无向连通图 $G=(V, E)$, 权函数 W
- 输出: G 的最小生成树

最小生成树

- 克鲁斯卡尔 (Kruskal) 算法
 - 初始时所有顶点自成一集合
 - 在图上选权值最小的边 e_{\min} ，判断 e_{\min} 两端点是否属于不同集合 c_i, c_j
 - 若是，将 c_i, c_j 用 e_{\min} 连接成同一个集合
 - 否则，舍弃 e_{\min}
 - 重复上一过程，直到所有顶点在同一集合

最小生成树

■ 克鲁斯卡尔 (Kruskal) 算法

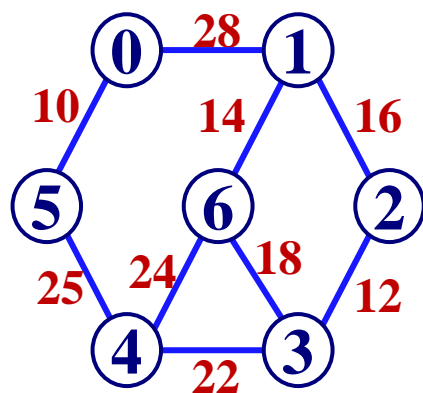


最小生成树

■ 克鲁斯卡尔 (Kruskal) 算法

- 经常需要判断权值最小的边的两端是否属于不同连通分量

➤ 可使用并查集技术加快判断速度



10 (0,5)
12 (2,3)
14 (1,6)
16 (1,2)
18 (3,6)
22 (3,4)
24 (4,6)
25 (4,5)

① ② ③ ④ ⑤ ⑥

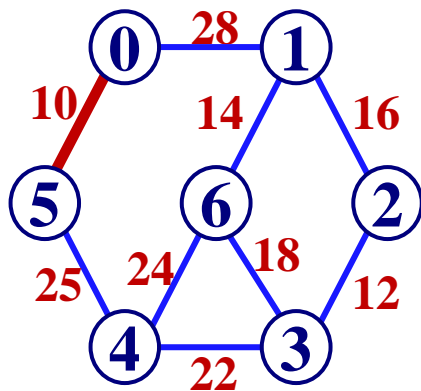
初始时

最小生成树

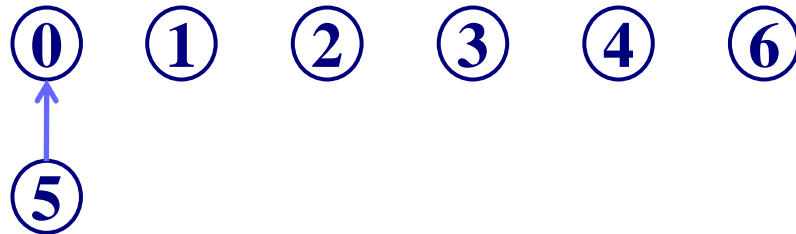
■ 克鲁斯卡尔 (Kruskal) 算法

- 经常需要判断权值最小的边的两端是否属于不同连通分量

➤ 可使用并查集技术加快判断速度



10 (0,5)
12 (2,3)
14 (1,6)
16 (1,2)
18 (3,6)
22 (3,4)
24 (4,6)
25 (4,5)



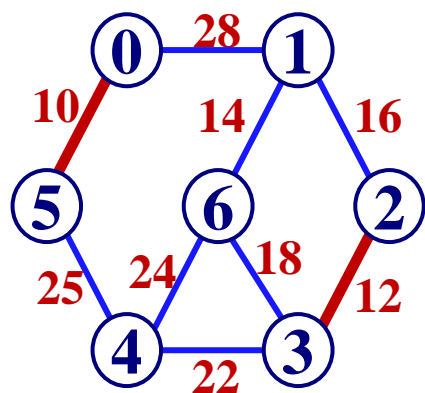
取边(0,5)

最小生成树

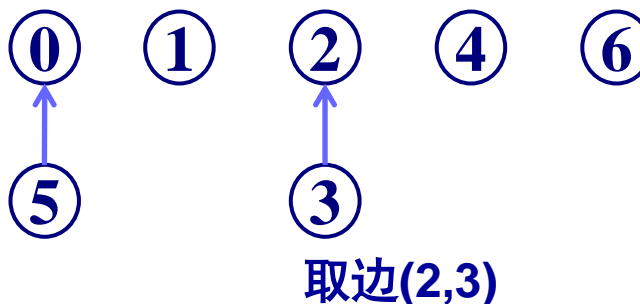
■ 克鲁斯卡尔 (Kruskal) 算法

- 经常需要判断权值最小的边的两端是否属于不同连通分量

➤ 可使用并查集技术加快判断速度



10 (0,5)
12 (2,3)
14 (1,6)
16 (1,2)
18 (3,6)
22 (3,4)
24 (4,6)
25 (4,5)

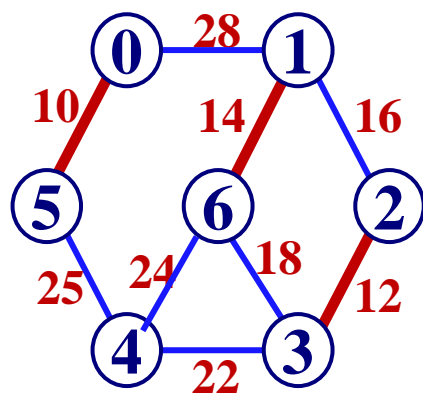


最小生成树

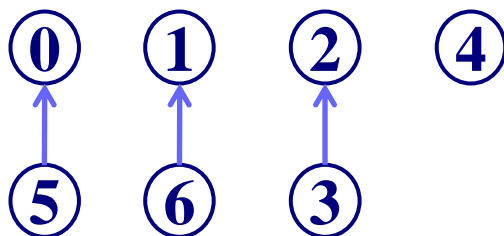
■ 克鲁斯卡尔 (Kruskal) 算法

- 经常需要判断权值最小的边的两端是否属于不同连通分量

➤ 可使用并查集技术加快判断速度



10 (0,5)
12 (2,3)
14 (1,6)
16 (1,2)
18 (3,6)
22 (3,4)
24 (4,6)
25 (4,5)

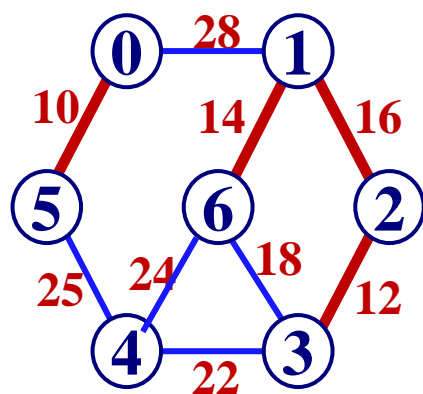


取边(1,6)

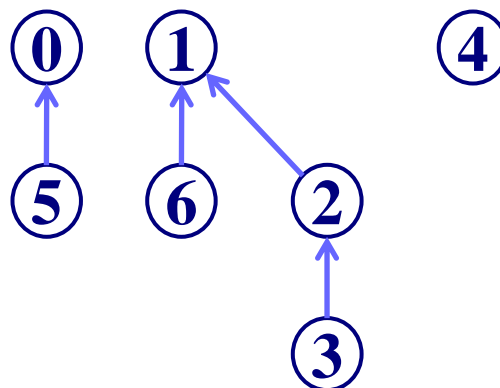
最小生成树

■ 克鲁斯卡尔 (Kruskal) 算法

- 经常需要判断权值最小的边的两端是否属于不同连通分量
 - 可使用并查集技术加快判断速度



10 (0,5)
12 (2,3)
14 (1,6)
16 (1,2)
18 (3,6)
22 (3,4)
24 (4,6)
25 (4,5)



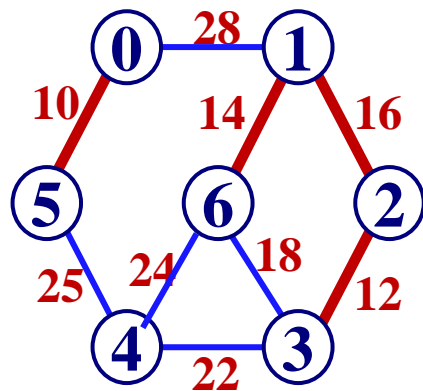
取边(1,2)

最小生成树

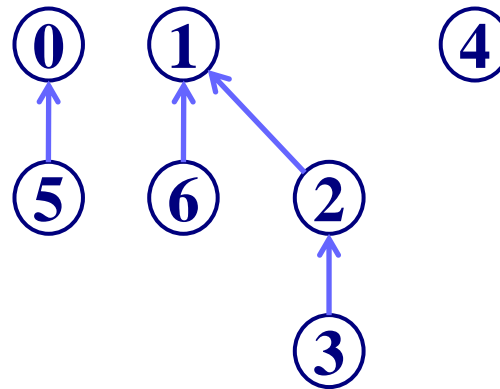
■ 克鲁斯卡尔 (Kruskal) 算法

- 经常需要判断权值最小的边的两端是否属于不同连通分量

➤ 可使用并查集技术加快判断速度



10 (0,5)
12 (2,3)
14 (1,6)
16 (1,2)
18 (3,6)
22 (3,4)
24 (4,6)
25 (4,5)



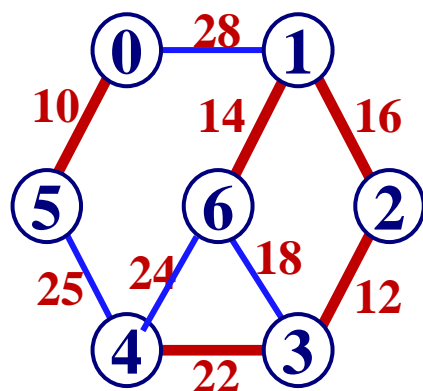
取边(3,6)

最小生成树

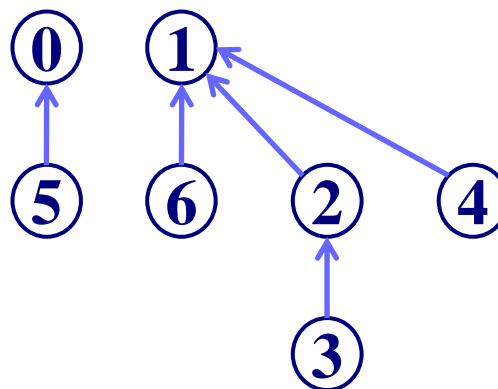
■ 克鲁斯卡尔 (Kruskal) 算法

- 经常需要判断权值最小的边的两端是否属于不同连通分量

➤ 可使用并查集技术加快判断速度



10 (0,5)
12 (2,3)
14 (1,6)
16 (1,2)
18 (3,6)
22 (3,4)
24 (4,6)
25 (4,5)



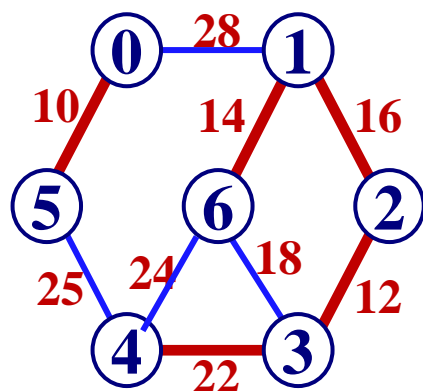
取边(3,4)

最小生成树

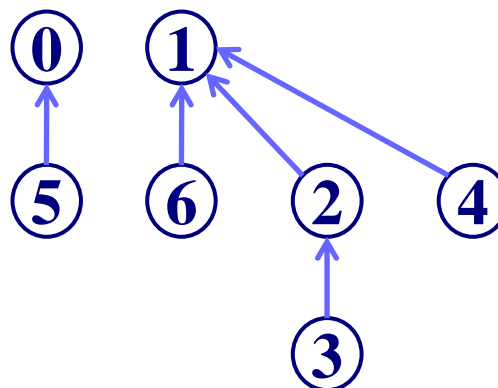
■ 克鲁斯卡尔 (Kruskal) 算法

- 经常需要判断权值最小的边的两端是否属于不同连通分量

➤ 可使用并查集技术加快判断速度



10 (0,5)
12 (2,3)
14 (1,6)
16 (1,2)
18 (3,6)
22 (3,4)
24 (4,6)
25 (4,5)



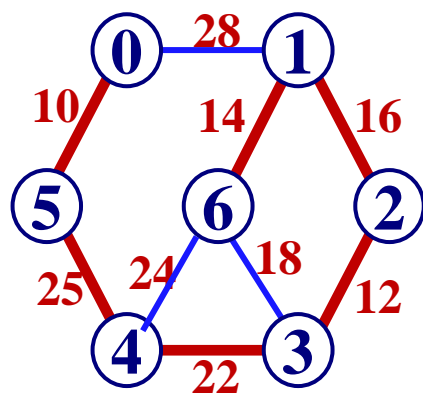
取边(4,6)

最小生成树

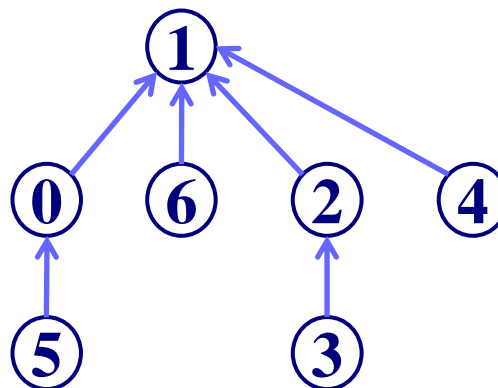
■ 克鲁斯卡尔 (Kruskal) 算法

- 经常需要判断权值最小的边的两端是否属于不同连通分量

➤ 可使用并查集技术加快判断速度



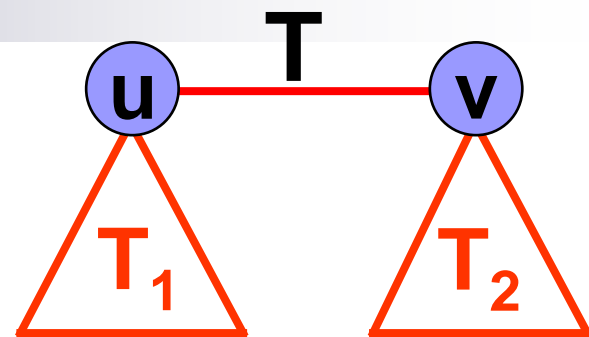
10 (0,5)
12 (2,3)
14 (1,6)
16 (1,2)
18 (3,6)
22 (3,4)
24 (4,6)
25 (4,5)



取边(4,5)

此时所有顶点属于同一连通分量，结束

最小生成树



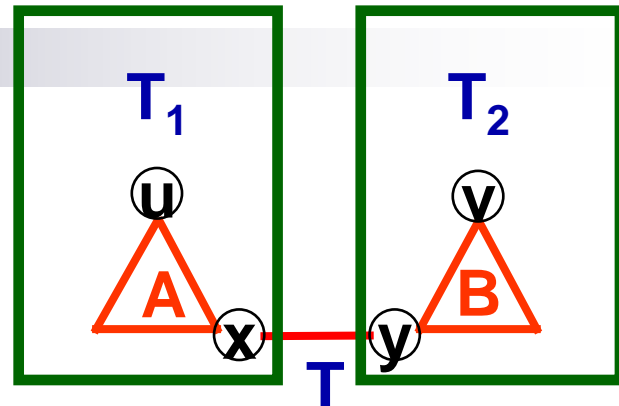
■ 最优子结构

- 设 T 是 G 的最小生成树。不妨设 T 包含子树 T_1 和 T_2 ， T_1 是 G 的子连通图 G_1 的生成树， T_2 是 G 的子连通图 G_2 的生成树，则 T_1 是 G_1 的最小生成树， T_2 是 G_2 的最小生成树

□ 证明：

- 若 T_1 和 T_2 不是 G_1 和 G_2 的最小生成树，而最小生成树分别是 T'_1 和 T'_2 ，即 $W(T'_1) \leq W(T_1)$ ， $W(T'_2) \leq W(T_2)$ ，则存在最小生成树 $W(T') = W(T'_1) + W(T'_2) + W(u, v) \leq W(T_1) + W(T_2) + W(u, v) = W(T)$ ，与 T 是最小生成树矛盾。

最小生成树



■ 贪心选择性

- 设边 (u,v) 是当前权值最小且两端点分别属于不同两个集合 A 和 B 的边，则必然存在一棵最小生成树包含边 (u,v) 。

□ 证明：

- 如图所示，假设不存在一棵最小生成树包含边 (u,v) 。为了得到生成树， A 和 B 之间必然有一条路径连接。假设这条路径连接 A 和 B 包含边 (x,y) 得到最小生成树 T 。 $W(u,v) \leq W(x,y)$ ，边 (x,y) 可将 T 划分两棵子树 T_1 和 T_2 。不妨设 $A \subseteq T_1$ ， $B \subseteq T_2$ ， $W(T') = W(T_1) + W(T_2) + W(u,v) \leq W(T_1) + W(T_2) + W(x,y) = W(T)$ ，与 T 最小矛盾。得证。

选择当前权值最小且两端点属两集合结果并不会变坏

最短路径

- 单源最短路径的Dijkstra算法
 - 给定带权图 (每条边权值 ≥ 0)
 - 给定源点 v ，求 v 到其他顶点的最短路径

最短路径

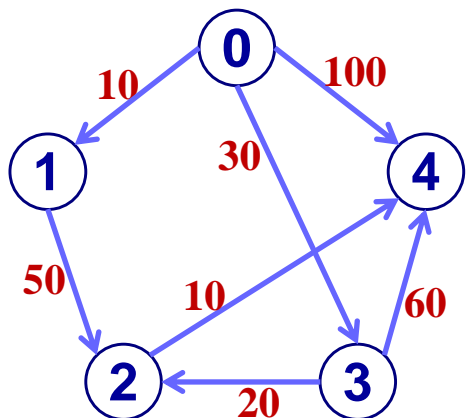
■ Dijkstra算法

- 初始时令 $S=\{v_0\}$, $\text{dist}[v_0]=0$, $\text{dist}[i]=\text{Edge}[v_0][i]$
- 找 $u \in S, v \notin S$, 且 $\text{dist}[u] + \text{Edge}[u][v]$ 最小, 则将 v 加入 S 中, $\text{dist}[v] = \text{dist}[u] + \text{Edge}[u][v]$
- 重复上一步骤, 直到所有顶点都加入 S 中

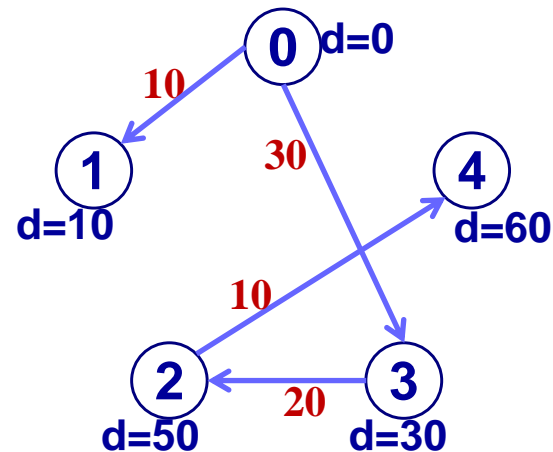
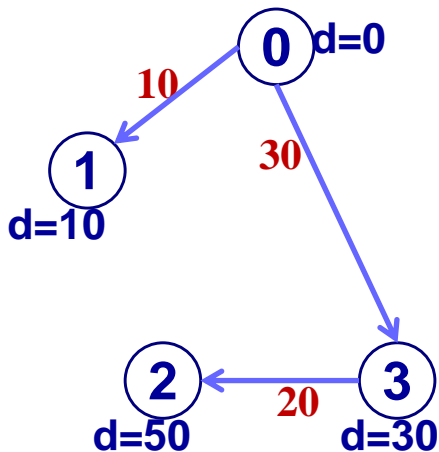
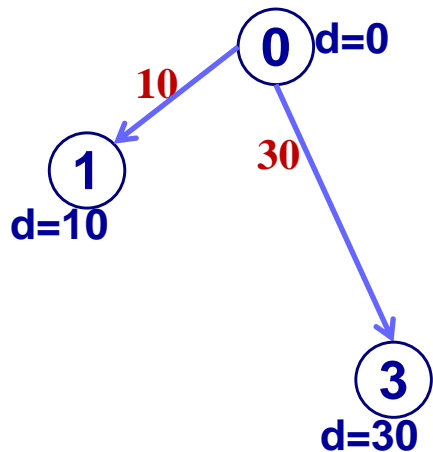
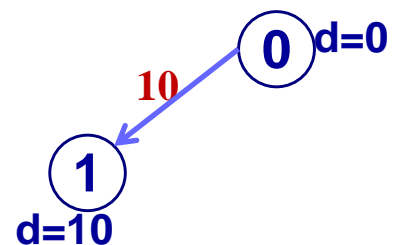
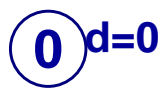
最短路径

$$\text{Edge} = \begin{bmatrix} 0 & 10 & \infty & 30 & 100 \\ \infty & 0 & 50 & \infty & \infty \\ \infty & \infty & 0 & \infty & 10 \\ \infty & \infty & 20 & 0 & 60 \\ \infty & \infty & \infty & \infty & 0 \end{bmatrix}$$

■ Dijkstra算法(不记录路径)



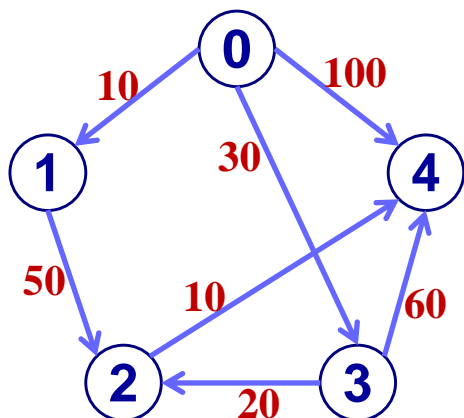
原图



最短路径

■ Dijkstra算法(记录路径)

$$\text{Edge} = \begin{bmatrix} 0 & 10 & \infty & 30 & 100 \\ \infty & 0 & 50 & \infty & \infty \\ \infty & \infty & 0 & \infty & 10 \\ \infty & \infty & 20 & 0 & 60 \\ \infty & \infty & \infty & \infty & 0 \end{bmatrix}$$



原图

$$\begin{matrix} \textcircled{0} & d[0]=0 \\ & p[0]=-1 \end{matrix}$$

$$\begin{matrix} \textcircled{0} & d[0]=0 \\ & p[0]=-1 \\ \swarrow 10 & \\ \textcircled{1} & d[1]=10 \\ & p[1]=0 \end{matrix}$$

$$\begin{matrix} \textcircled{0} & d[0]=0 \\ & p[0]=-1 \\ \swarrow 10 & \\ \textcircled{1} & d[1]=10 \\ & p[1]=0 \\ \searrow 30 & \\ \textcircled{3} & d[3]=30 \\ & p[3]=0 \end{matrix}$$

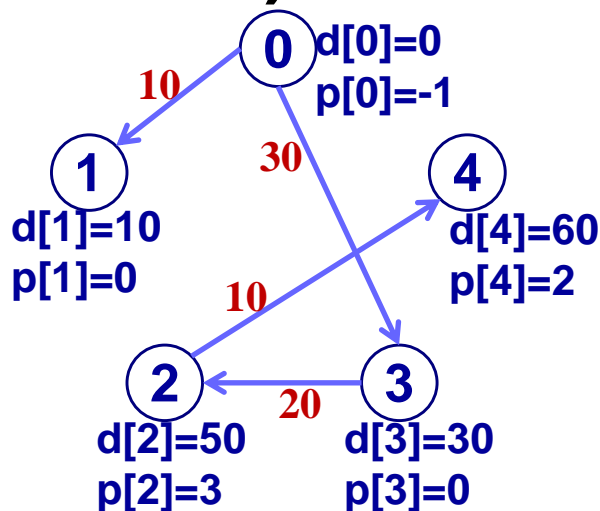
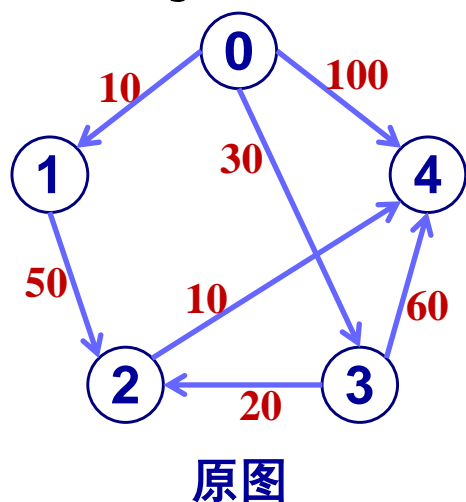
$$\begin{matrix} \textcircled{0} & d[0]=0 \\ & p[0]=-1 \\ \swarrow 10 & \\ \textcircled{1} & d[1]=10 \\ & p[1]=0 \\ \searrow 30 & \\ \textcircled{3} & d[3]=30 \\ & p[3]=0 \\ \swarrow 20 & \\ \textcircled{2} & d[2]=50 \\ & p[2]=3 \end{matrix}$$

$$\begin{matrix} \textcircled{0} & d[0]=0 \\ & p[0]=-1 \\ \swarrow 10 & \\ \textcircled{1} & d[1]=10 \\ & p[1]=0 \\ \searrow 30 & \\ \textcircled{3} & d[3]=30 \\ & p[3]=0 \\ \swarrow 20 & \\ \textcircled{2} & d[2]=50 \\ & p[2]=3 \\ \nearrow 10 & \\ \textcircled{4} & d[4]=60 \\ & p[4]=2 \end{matrix}$$

最短路径

$$\text{Edge} = \begin{bmatrix} 0 & 10 & \infty & 30 & 100 \\ \infty & 0 & 50 & \infty & \infty \\ \infty & \infty & 0 & \infty & 10 \\ \infty & \infty & 20 & 0 & 60 \\ \infty & \infty & \infty & \infty & 0 \end{bmatrix}$$

■ Dijkstra算法(记录路径)



获取最短路径方法，以顶点4为例：

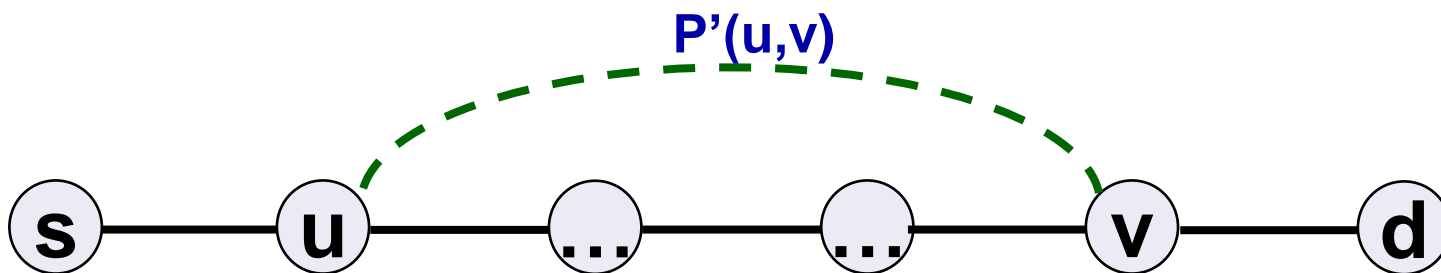
$p[4]=3 \Rightarrow p[3]=2 \Rightarrow p[2]=0$

反向读得0到4的最短路径为 (0,3,2,4)

最短路径

■ 最优子结构

- 设 $P(s,d)$ 是 s 到 d 的最短路径，那么这条路径上的子路径 $P(u,v)$ 是 u 到 v 的最短路径。
- 证明：
 - 如若不然，假设 u 到 v 之间存在更短路径 $P'(u,v)$ ，则用其替代 $P(u,v)$ ，得到一条 s 到 d 的更短路径 $P'(s,d)$ ，与 $P(s,d)$ 是 s 到 d 是最短路径矛盾。



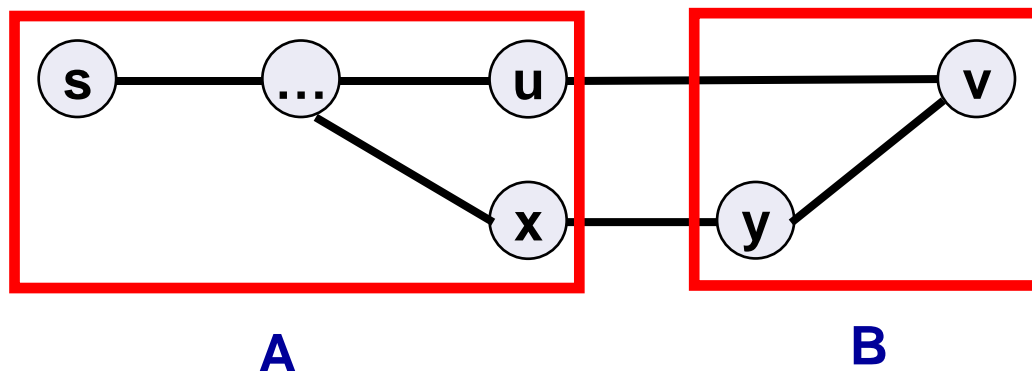
最短路径

■ 贪心选择性

- 设A是已计算好最短路径的顶点集合，B是未计算好最短路径的顶点集合。P是s到u的最短路径。假设 $W(P) + W(u, v)$ 对任意 $u \in A, v \in B$ 最小，则 $P + \{(u, v)\}$ 是s到v的最短路径

- 证明：

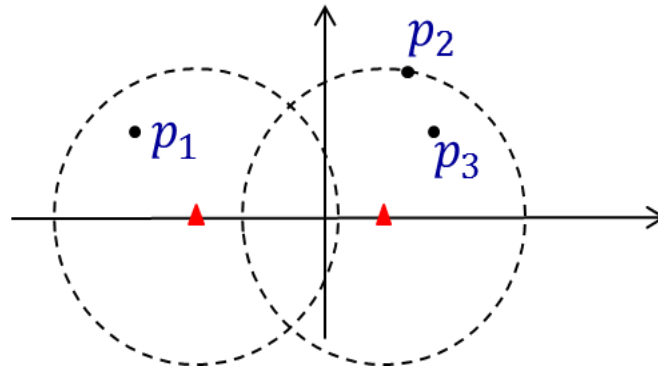
➤ 略



选择当前累积权值最小且两端点属两集合结果不会变坏

POJ 2393 1328

- (1) 给定 n 个物品，物品价值分别为 P_1, P_2, \dots, P_n ，物品重量分别为 W_1, W_2, \dots, W_n ，背包容量为 M 。每种物品可部分装入到背包中。输出 X_1, X_2, \dots, X_n ， $0 \leq X_i \leq 1$ ，使得 $\sum_{1 \leq i \leq n} P_i X_i$ 最大，且 $\sum_{1 \leq i \leq n} W_i X_i \leq M$ 。试设计一个算法求解该问题，分析算法的正确性。
- (2) 海面上有一些船需要与陆地进行通信，需要在海岸线上布置一些基站。现将问题抽象为，在 x 轴上方，给出 N 条船的坐标 p_1, p_2, \dots, p_N ， $p_i = (x_i, y_i)$ ， $x_i \geq 0, y_i \leq d, 1 \leq i \leq N$ ，在 x 轴上安放的基站可以覆盖半径为 d 的区域内的所有点，问在 x 轴上至少要安放几个点才可以将 x 轴上方的点都覆盖起来。试设计一个算法求解该问题，并分析算法的正确性。



- 某公司有个工厂和仓库。由于原材料等价格波动，工厂每个月的生产成本也会波动，令第 i 个月产品的单位生产成本为 c_i （该月生产一个产品的成本为 c_i ）。仓库储存产品的也有成本，假设每个月产品的单位储存成本为固定值1（存储一个产品一个月的成本为1）。令第 i 个月需要供应给客户的产品数量为 y_i ，仓库里的和生产的产品均可供应给客户。假设仓库的容量无限大，供应给客户剩余的产品可储存在仓库中。若已知 n 个月中各月的单位生产成本 c_i 、以及产品供应量 y_i ，设计一算法决策每个月的产品生产数量 x_i ，使得 n 个月的总成本最低。例如： $n = 3$ ， $c_i: 2, 5, 3$ ， $y_i: 2, 4, 5$ ，则 $x_i: 6, 0, 5$ ，即第1个月生产6个供应2个（代价 $2 \times 2 = 4$ ），储存4个供应给第2个月（代价 $(2+1) \times 4 = 12$ ），第3个月生产5个供应5个（代价 $3 \times 5 = 15$ ），使总成本 $4 + 12 + 15 = 31$ 最小。
- 给定直线上 $2n$ 个点的序列 $P[1, 2, \dots, 2n]$ ，每个点 $P[i]$ 要么是白点要么是黑点，其中共有 n 个白点和 n 个黑点，相邻两个点之间距离均为1，请设计一个算法将每个白点与一黑点相连，使得连线的总长度最小。例如，图中有4个白点和4个黑点，以图中方式相连，连线总长度为 $1+1+1+5=8$ 。



- 有 n 个作业需要在一台机器上执行，一个时刻机器上只能执行一个作业，每个作业可在单位时间内完成，作业 i 有截止时间 d_i ，当作业 i 在截止时间被执行完，则可获得 p_i 的收益，请设计算法获得最大收益，并分析算法的正确性。
- 假设有数目不限的面值为25美分，10美分，5美分，1美分的硬币，请使用最少个数的硬币凑出3.33美元。

