HIT
CS&E

第五章

贪心算法的设计与分析原理

程思瑶
计算机科学与技术学院

提要

5.1 贪心算法的要素

5.2 活动选择(activity-selection)问题

5.3 Huffman编码

5.4 最小生成树问题

5.5 贪心算法的理论基础

5.6 任务调度问题

Introduction to Algorithms
Chapter 16

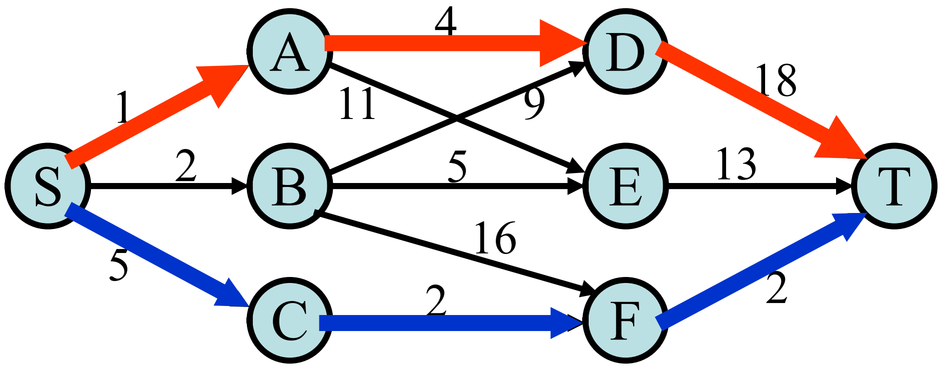
Pages 370-405

5.1 贪心算法的要素

- Greedy算法的基本概念
- Greedy算法与动态规划方法的比较
- Greedy算法的设计步骤

Greedy算法的基本概念

- Greedy算法的实例
  - 最短路径问题



  - Greedy求解过程

Greedy算法不一定产生正确解

- Greedy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
  - 希望通过作出局部优化选择达到全局优化选择
  - Greedy算法不一定总产生优化解
- Greedy算法产生优化解的条件
  - 优化子结构
  - Greedy选择性(Greedy-choice property)

- Greedy选择性

一个优化问题的全局优化解可以通过局部优化选择得到。

- Greedy选择性需证明
  - 归纳法
    - 对算法步数归纳或问题规模归纳
  - 交换论证法
    - 在保证最优性不变前提下，从一个最优解逐步替换，最终得到贪心算法的解

- 优化子结构

  若一个优化问题的优化解包含它的(剩余)子问题的优化解，则称其具有优化子结构

与动态规划方法的比较

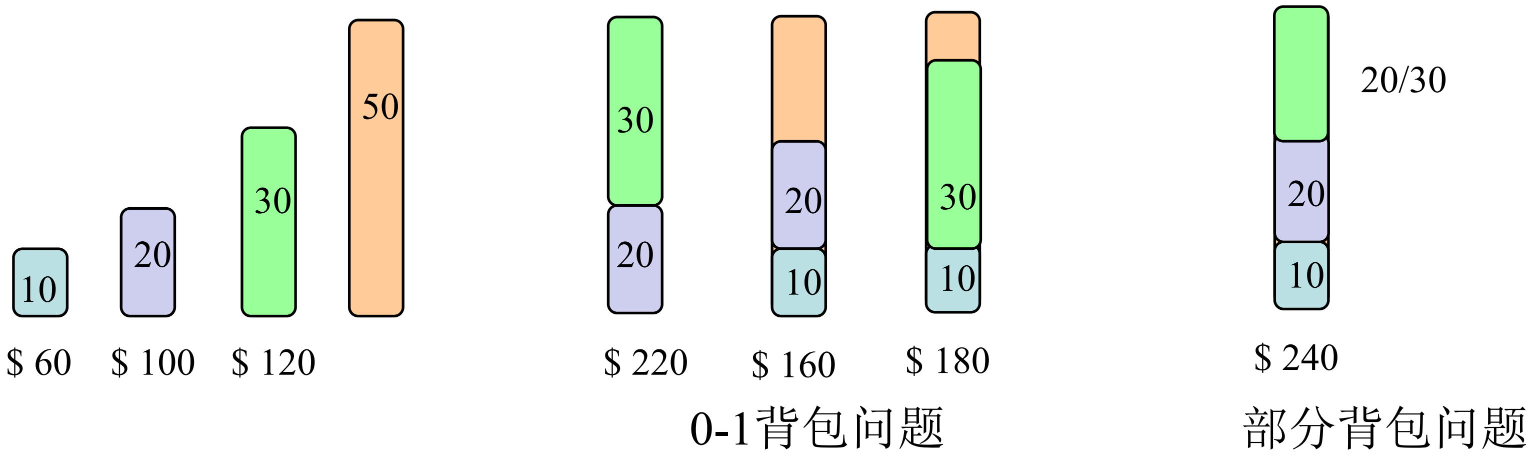
- 动态规划方法
  - 在每一步所做的选择通常依赖于子问题的解
  - 以自底向上方式，先解小子问题，再求解大子问题
- Greedy方法
  - 在每一步先做出当前看起来最好的选择
  - 然后再求解本次选择后产生的剩余子问题
  - 每次选择既不依赖于子问题的解，也不依赖于未来的选择
  - 以自顶向下方式，逐步进行贪心选择，不断减少子问题规模

与动态规划方法的比较

- 动态规划方法可用的条件
  - 优化子结构
  - 子问题重叠性
- Greedy方法可用的条件
  - 优化子结构
  - Greedy选择性
- 可用Greedy方法时，动态规划方法可能不适用
- 可用动态规划方法时，Greedy方法可能不适用

与动态规划方法的比较

- 例如：0-1背包问题与部分背包问题
  - 都具有优化子结构
  - 但是，部分背包问题可用贪心策略解决，而0-1背包问题却不行！
    - 计算每个物品每磅价值v_i/w_i，并按照每磅价值由大到小顺序取物品



准确Greedy算法的设计步骤

1. 设计贪心选择方法：
   - 贪心选择方法
   - 剩余子问题

很重要！
决定能否得到
全局最优解

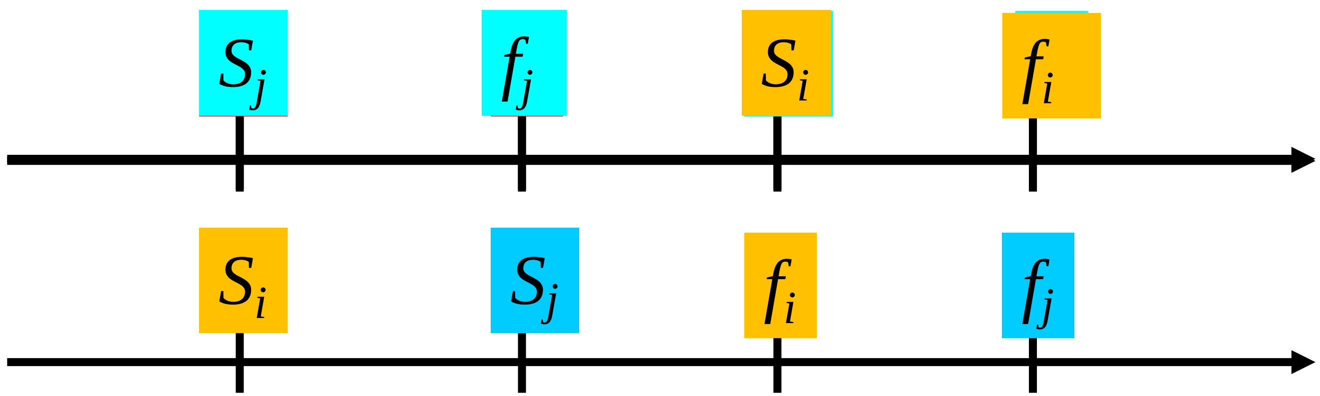
2. 证明：对于1中所求解的问题具有优化子结构
3. 证明：对于1中所求解的问题具有贪心选择性
4. 按照1中设计的贪心选择方法设计算法

5.2活动选择(activity-selection)问题

- 问题定义
- 问题求解
  - 设计贪心选择方法
  - 优化解的结构分析
  - Greedy选择性证明
  - 算法设计
  - 算法复杂性分析

问题的定义

- 活动
  - 设$S=\{1,2,...,n\}$是n个活动的集合，所有活动共享一个资源，该资源同时只能为一个活动使用
  - 每个活动i有起始时间s_i，终止时间f_i，$s_i \leq f_i$
- 相容活动
  - 活动i和j是相容的，若$s_j \geq f_i$或$s_i \geq f_j$，即



- 活动选择问题定义
  - 输入：$S=\{1, 2, \ldots, n\}$,
    $F=\{\ [s_i,\ f_i]\ \}$，$n \geq i \geq 1$
  - 输出：S的最大相容活动集合

贪心思想：

为了选择最多的相容活动，每次选f_i最小的活动，使我们能够选更多的活动

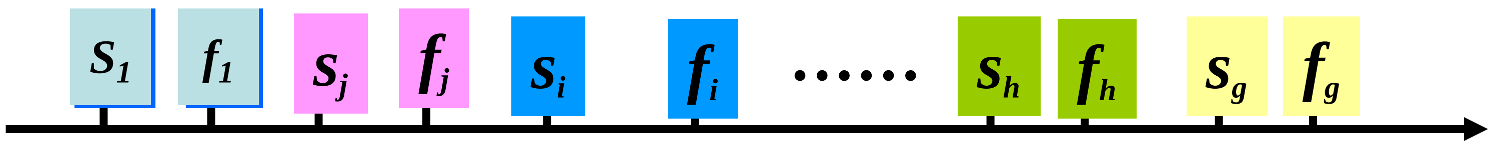
剩余子问题：$S_i = \{\ j \in S \mid s_j \geq f_i\ \}$

HIT
CS&E

优化解结构分析

引理1 设$S=\{1,2,\ldots,n\}$是n个活动集合，$[s_i, f_i]$是活动i的起始终止时间，且$f_1\leq f_2\leq\ldots\leq f_n$，$S$的活动选择问题的某个优化解包括活动1.

证 设A是一个优化解，按结束时间排序A中活动，设其第一个活动为k，第二个活动为j. ……



如果$k=1$，引理成立.

如果$k\neq 1$，令$B=A-\{k\}\cup\{1\}$,

由于A中活动相容，$f_1\leq f_k\leq s_j$, B中活动相容.

因为$|B|=|A|$, 所以B是一个优化解，且包括活动1.

定理1. 设$S=\{1, 2, \ldots, n\}$是n个活动集合，$[s_i, f_i]$是活动i的起始终止时间，且$f_1 \leq f_2 \leq \ldots \leq f_n$，设$A$是$S$的调度问题的一个优化解且包括活动1，则$A'=A-\{1\}$是$S'=\{i \in S | s_i \geq f_1\}$的调度问题的优化解.

证. 显然，A'中的活动是相容的.

我们仅需要证明A'是最大的.

定理1说明活动选择问题具有优化子结构

B是S的一个解.

由于$|A|=|A'|+1$, $|B|=|B'|+1>|A'|+1=|A|$, 与A最大矛盾.

定理2. 设 $S=\{1, 2, \ldots., n\}$ 是 n 个活动集合，$f_1 \leq f_2 \leq \ldots \leq f_n$，$f_{l_0}=0$，$l_i$ 是 $S_i=\{ j \in S \mid s_j \geq f_{l_{i-1}} \}$ 中具有最小结束时间 f_{l_i} 的活动。设 A 是 S 的包含活动1的优化解，则 $A=\bigcup_{i=1}^{k}\{l_i\}$

证. 对 $|A|$ 作归纳法.

当 $|A|=1$ 时，由引理1，命题成立.

设 $|A| \leq k-1$ 时，命题成立.

当 $|A|=k$ 时，由定理1，$A=\{l_1=1\} \cup A_1$.

A_1 是 $S_2=\{ j \in S \mid s_j \geq f_{l_1}=f_1 \}$ 的优化解.

由归纳假设，$A_1=\bigcup_{i=2}^{k}\{l_i\}$，于是，$A=\bigcup_{i=1}^{k}\{l_i\}$.

$S_1=\{1,2,3,4,5,6,7,8,9,10,11\}$, $l_0=0$

$S_2=\{4, 6,7,8,9,11\}$, $l_1=1$

$S_3=\{8,9,11\}$, $l_2=4$

$S_4=\{11\}$, $l_3=8$

$l_1=1$

$l_2=4$

$l_3=8$

$l_4=11$

$S_1=\{ j \in S \mid s_j \geq f_{l_0}=0\}$

$S_2=\{ j \in S \mid s_j \geq f_{l_1}=f_1\}$

$S_3=\{ j \in S \mid s_j \geq f_{l_2}\}$

……

$S_k=\{ j \in S \mid s_j \geq f_{l_{k-1}}\}$

$S_{k+1}=\{ j \in S \mid s_j \geq f_{l_k}\}=\theta$

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	3	5	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14

算法的设计

- 贪心选择方法
  - 选择：
    - 每次选择具有最小结束时间的活动f_i
  - 剩余子问题：
    - $S_i = \{ j \in S \mid s_j \geq f_i \}$

- 算法

（设$f_1 \leq f_2 \leq \ldots \leq f_n$已排序）

Greedy-Activity-Selector(S, F)

$n \leftarrow$lenyth(S);

$A \leftarrow \{1\}$

$j \leftarrow 1$

For $i \leftarrow 2$ To n Do

If $s_i \geq f_j$

Then $A \leftarrow A \cup \{i\}$； $j \leftarrow i$；

Return A

- 算法

（设$f_1 \le f_2 \le \ldots \le f_n$已排序）

Greedy-Activity-Selector(S, F)

$n \leftarrow$ lenyth(S);

$A \leftarrow \{1\}$

$j \leftarrow 1$

For $i \leftarrow 2$ To n Do

　　If $s_i \geq f_j$

　　Then $A \leftarrow A \cup \{i\}$； $j \leftarrow i$；

Return A

- 如果结束时间已排序

$T(n)=\theta(n)$

- 如果 结束时间未排序

$T(n)=\theta(n)+\theta(nlogn)=\theta(nlogn)$

定理3. Greedy-Activity-Selector算法能够产生最优解.

证.

(1) 由定理1可知活动选择问题具有优化子结构

(2) 由定理2知贪心选择方法具有Greedy选择性

(3) Greedy-Activity-Selector算法确实按照定理2的Greedy选择性进行局部优化选择.

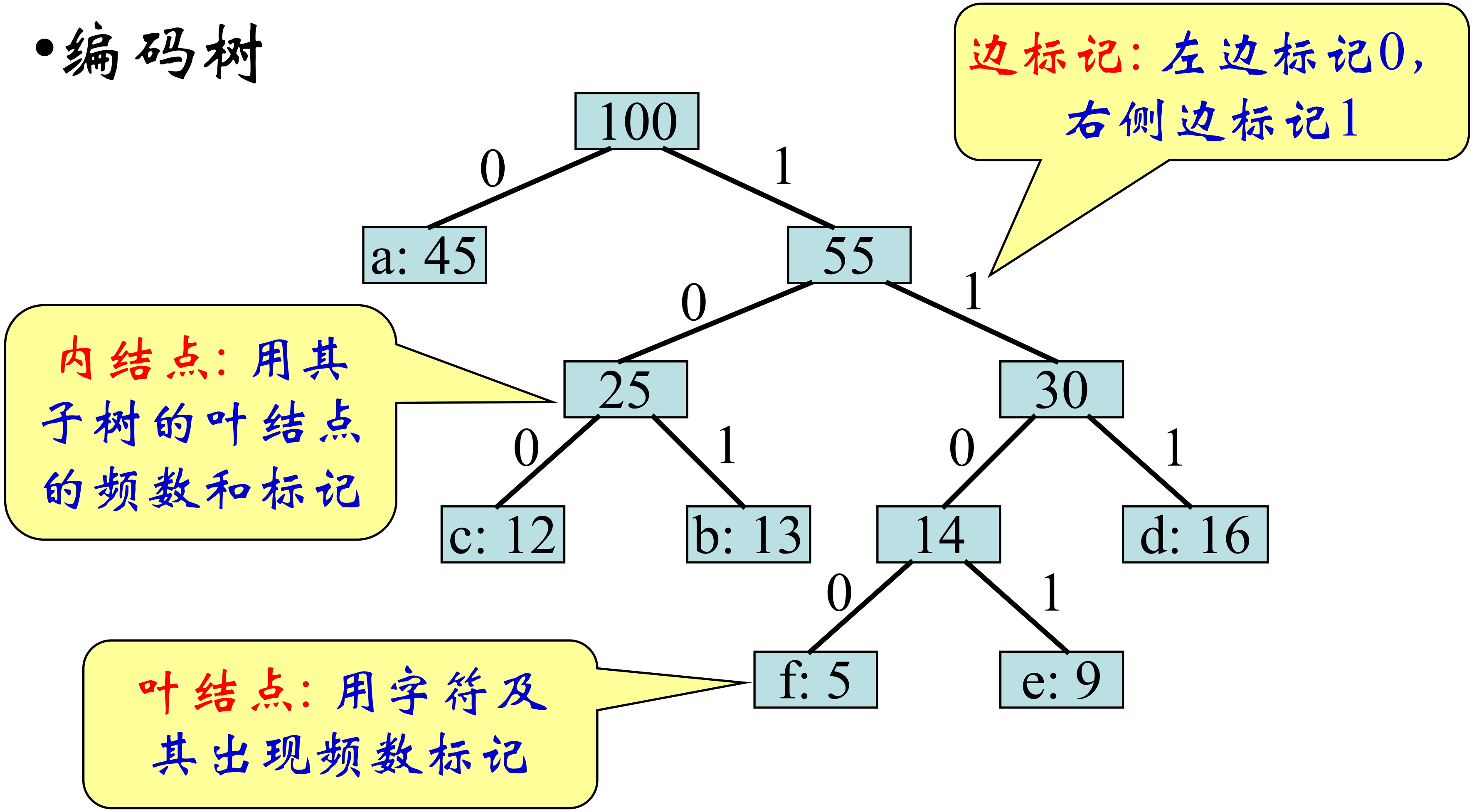
5.3 Huffman 编码

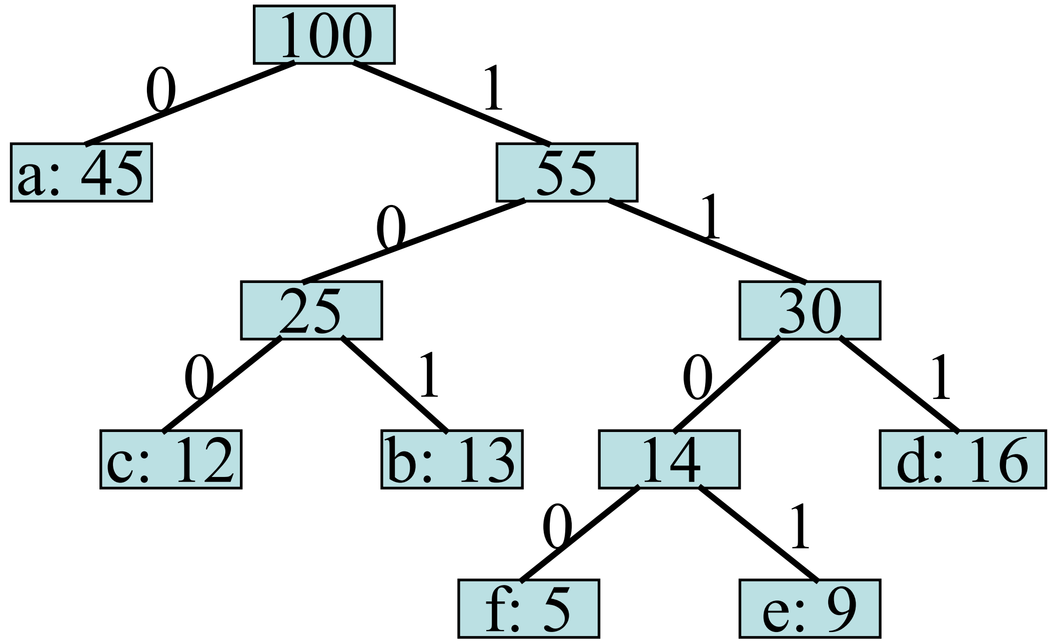
- 问题定义
- 问题求解
  - 设计贪心选择方法
  - 优化解的结构分析
  - Greedy选择性证明
  - 算法设计
  - 算法复杂性分析

问题定义

- 二进制字符编码
  - 每个字符用一个二进制0、1串来表示.
- 固定长编码
  - 每个字符都用相同长度的0、1串表示.
- 可变长编码
  - 经常出现的字符用短码，不经常出现的用长码
- 前缀编码
  - 无任何字符的编码是另一个字符编码的前缀

- 编码树


边标记：左边标记0，右侧边标记1
内结点：用其子树的叶结点的频数和标记
叶结点：用字符及其出现频数标记
100
0
1
a: 45
55
0
1
25
30
0
1
0
1
c: 12
b: 13
14
d: 16
0
1
f: 5
e: 9



- 编码树T的代价
  - 设C是字母表(给定文件中的字母集合),$\forall c \in C$
  - $f(c)$是c在文件中出现的频数
  - $d_T(c)$是叶子c在树T中的深度，即c的编码长度
  - T的代价是编码一个文件的所有字符的代码位数:

$$B(T) = \sum_{c \in C} f(c) d_T(c)$$

- 优化编码树问题

输入：字母表 $C = \{c_1, c_2, ..., c_n\}$，

频数表 $F = \{f(c_1), f(c_2), ..., f(c_n)\}$

输出：具有最小$B(T)$的C前缀编码树

贪心思想：

循环地选择具有最低频数的两个结点，生成一棵子树，直至形成树

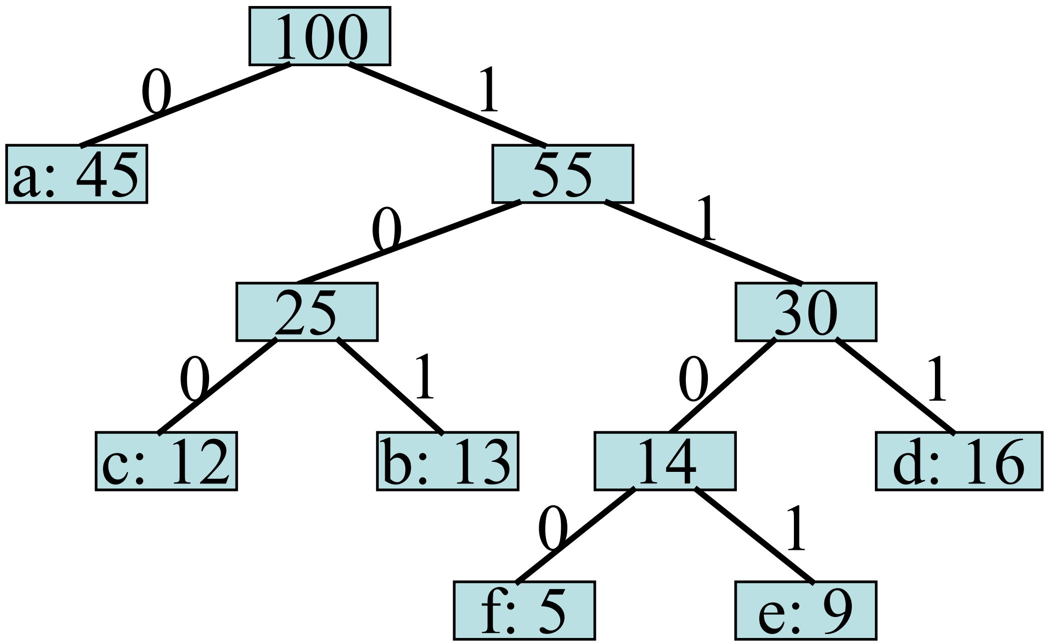
剩余子问题：？？？

HIT
CS&E

贪心思想：

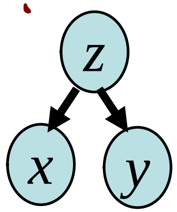
循环地选择具有最低频数的两个结点，生成一棵子树，直至形成树

f: 5 e: 9 c: 12 b: 13 d: 16 a: 45



设计贪心选择方法

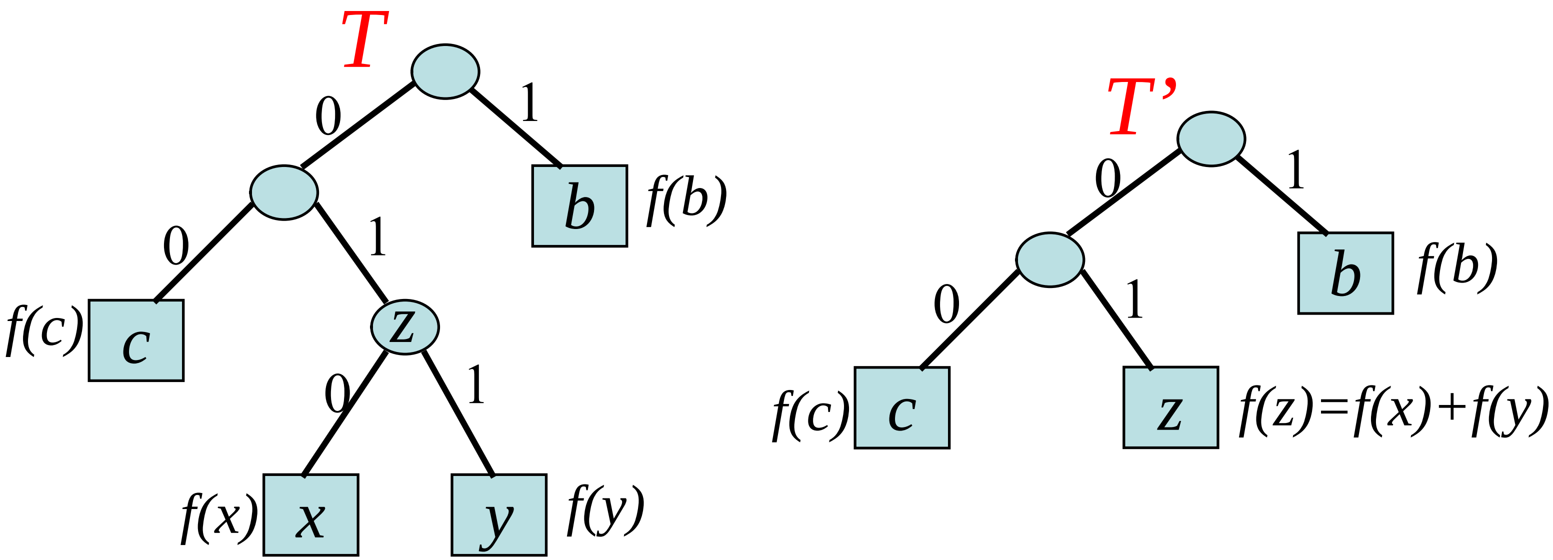
- 贪心选择方法
  - 选择方法：
    - 每次选择具有最低频数的两个节点x和y，构造一个子树：



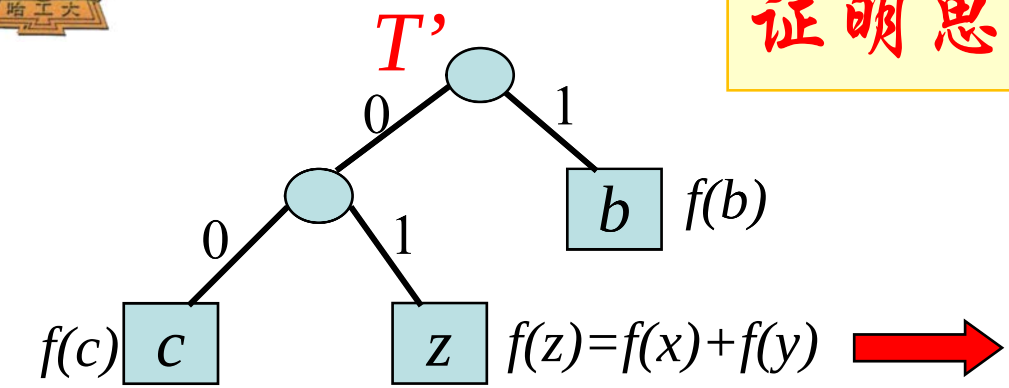
  - 剩余子问题的结构：
    - $C' = C - \{x, y\} \cup \{z\}$

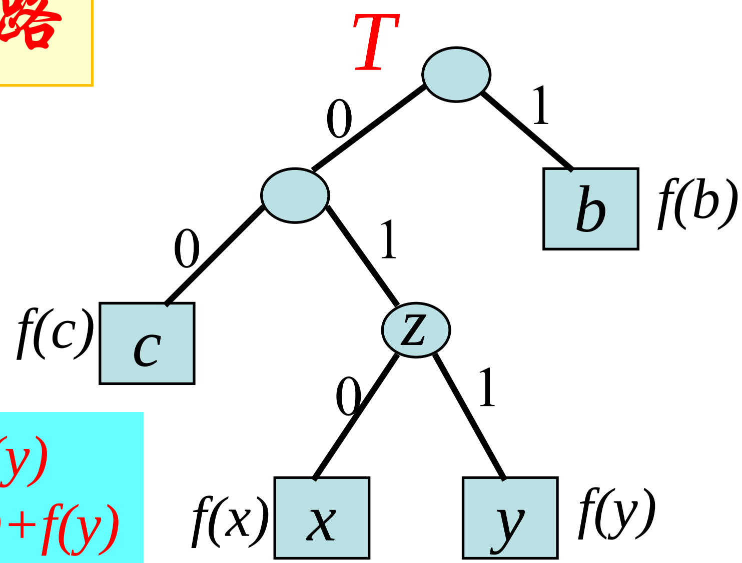
优化解的结构分析

引理1. 设T是字母表C的优化前缀树，$\forall c \in C$，$f(c)$是c在文件中出现的频数.设x、y是T中任意两个相邻叶结点，z是它们的父结点，则z作为频数是$f(z)=f(x)+f(y)$的字符，$T'=T-\{x,y\}$是字母表$C'=C-\{x,y\}\cup\{z\}$的优化前缀编码树.



证明思路



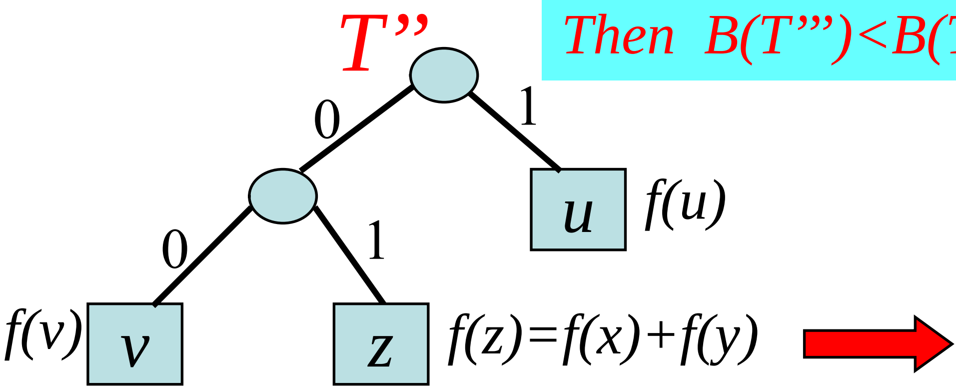


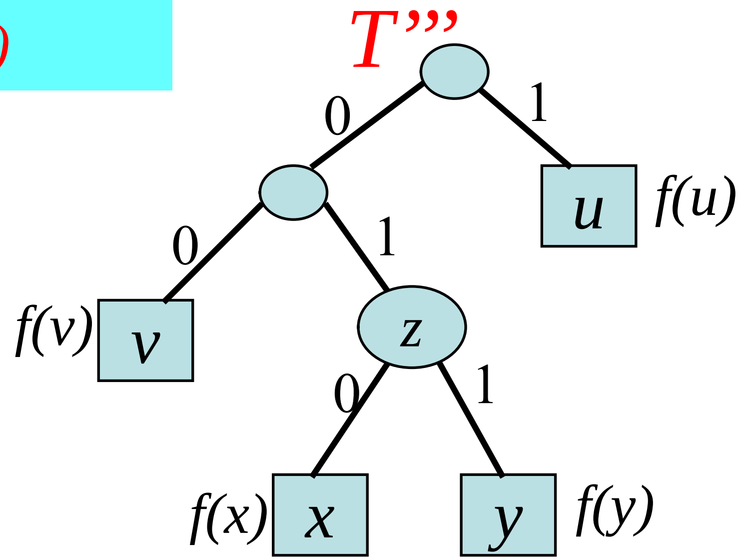
$B(T)=B(T')+f(x)+f(y)$

$B(T''')=B(T'')+ f(x)+f(y)$

If $B(T'')<B(T')$

Then $B(T''')<B(T)$





证．往证$B(T)=B(T')+f(x)+f(y)$.

$$B(T)=\Sigma_{c\in C}f(c)d_T(c)$$

$$=f(x)d_T(x)+f(y)d_T(y)+\Sigma_{c\in C'-\{z\}}f(c)d_T(c)$$

由于 $d_T(x)=d_T(y)=d_{T'}(z)+1$

$$=f(x)(d_{T'}(z)+1)+f(y)(d_{T'}(z)+1)+\Sigma_{c\in C'-\{z\}}f(c)d_{T'}(c)$$

$$=f(x)+f(y)+f(x)d_{T'}(z)+f(y)d_{T'}(z)+\Sigma_{c\in C'-\{z\}}f(c)d_{T'}(c)$$

由于$f(x)+f(y)=f(z)$

$$=f(x)+f(y)+f(z)d_{T'}(z)+\Sigma_{c\in C'-\{z\}}f(c)d_{T'}(c)$$

$$=B(T')+f(x)+f(y).$$

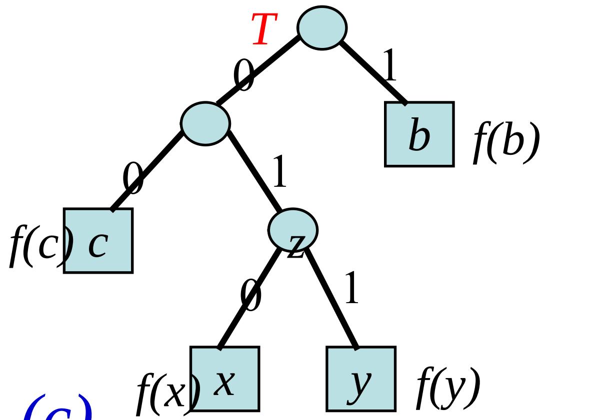
若T'不是C'的优化前缀编码树，

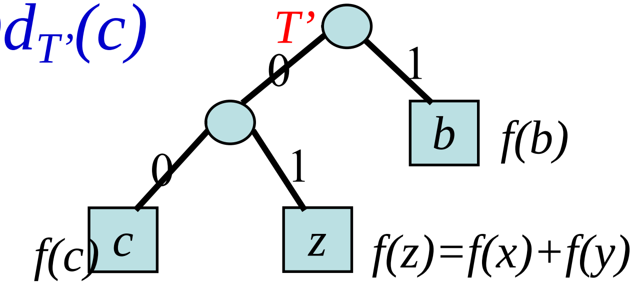
则必存在T''，使$B(T'')<B(T')$.

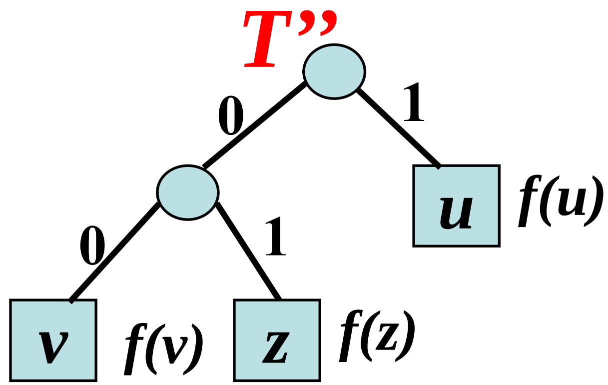
因为z是C'中字符,它必为T''中的叶子.

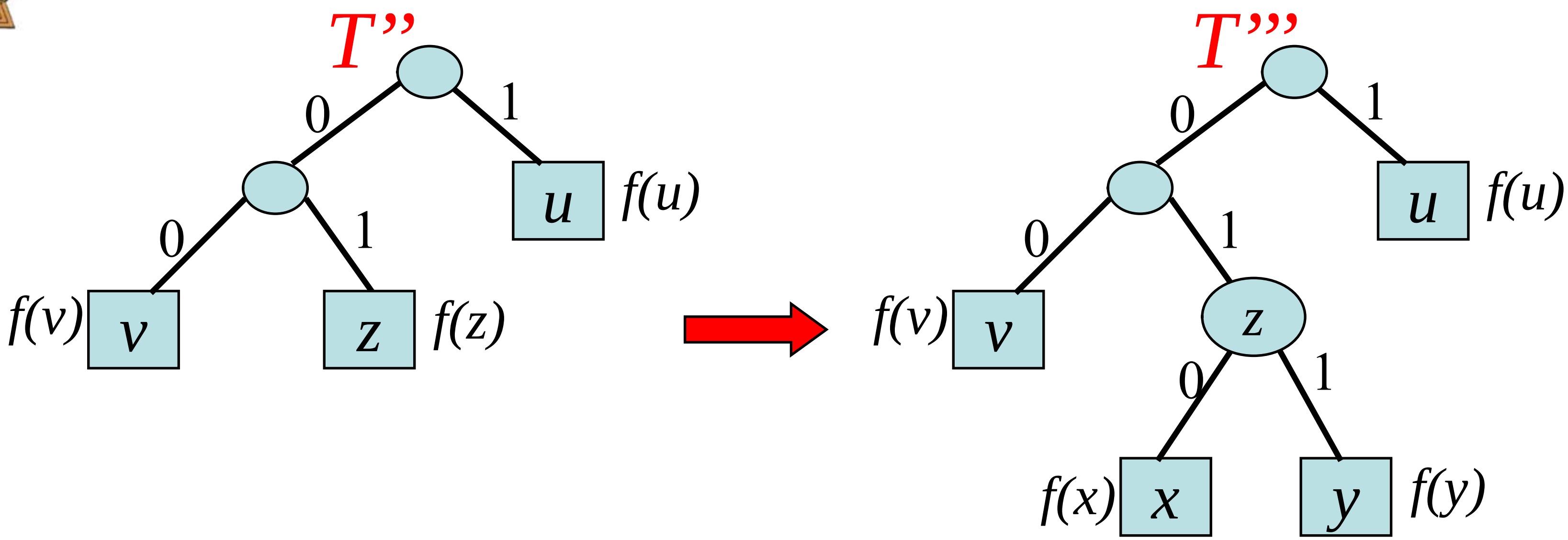
把结点x与y加入T''，作为z的子结点，

则得到C的一个如下前缀编码树T''':









如上可证：

$B(T''') = B(T'')+f(x)+f(y)$。

由于$B(T'')<B(T')$，

$B(T''') = B(T'')+f(x)+f(y) < B(T')+f(x)+f(y)=B(T)$

与T是优化的矛盾，故T'是C'的优化编码树.

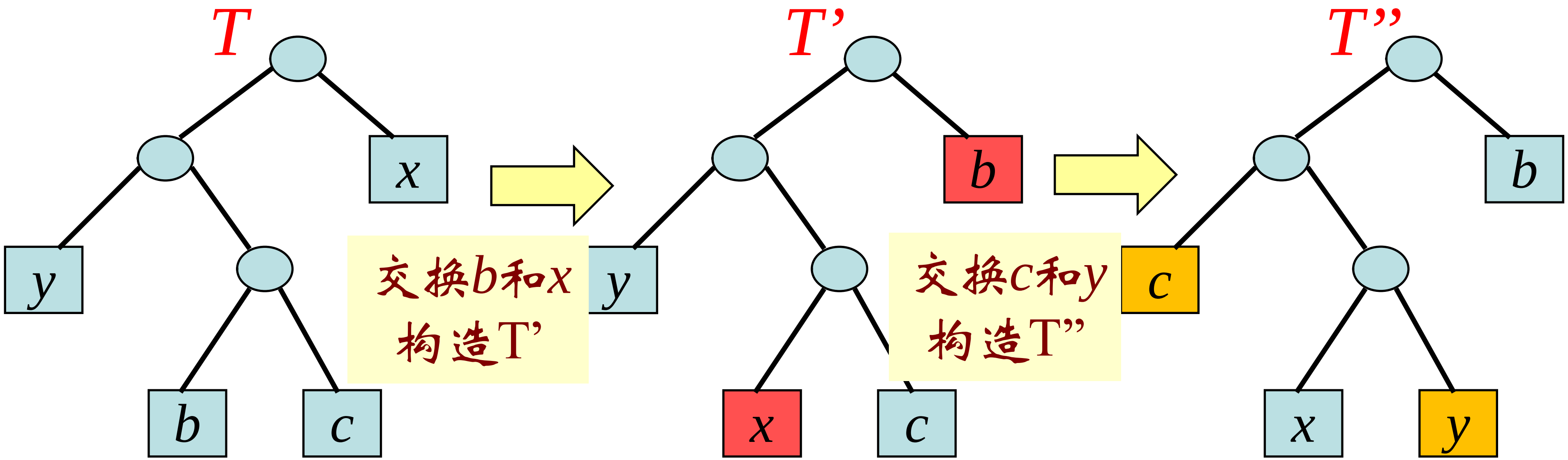
Greedy选择性

引理2. 设C是字母表，$\forall c \in C$，c具有频数$f(c)$, x、y是C中具有最小频数的两个字符，则存在一个C的优化前缀树，x与y的编码具有相同最大长度，且仅在最末一位不同.

优化前缀树问题具有Greedy选择性.

证：若T是C的优化前缀树，如果x和y是具有最大深度的两个兄弟字符，则命题得证。

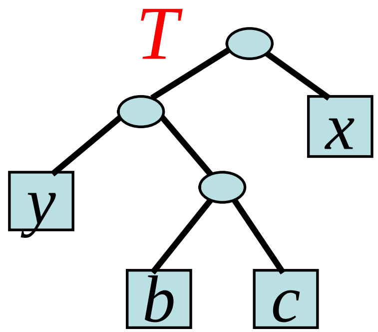
若不然，设b和c是具有最大深度的两个兄弟字符：

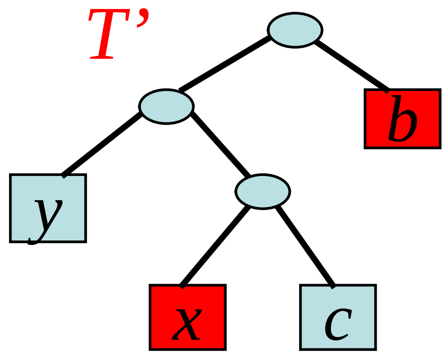


不失一般性，设$f(b) \le f(c)$, $f(x) \le f(y)$.

因x与y是具有最低频数的字符, $f(x) \le f(y) \le f(b) \le f(c)$.

交换T的b和····‘x, 从T构造T′；交换T′的c和y, 从T′构造T″





往证T''是最优化前缀树.

$B(T)-B(T')$

$$= \sum_{c \in C} f(c) d_T(c) - \sum_{c \in C} f(c) d_{T'}(c)$$

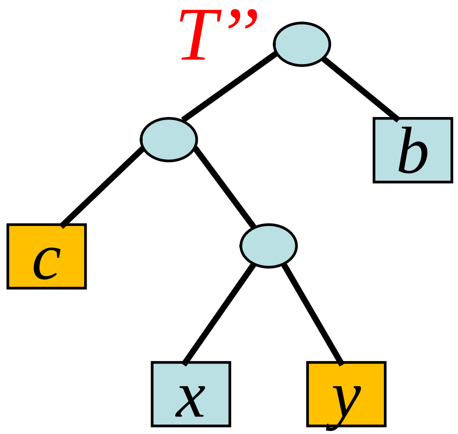
$$= f(x)d_T(x) + f(b)d_T(b) - f(x)d_{T'}(x) - f(b)d_{T'}(b)$$

$$= f(x)d_T(x) + f(b)d_T(b) - f(x)d_T(b) - f(b)d_T(x)$$

$$= (f(b)-f(x))(d_T(b)-d_T(x)).$$

$\because$ $f(b) \geq f(x)$, $d_T(b) \geq d_T(x)$ (因为b的深度最大)

$\therefore$ $B(T)-B(T') \geq 0$, $B(T) \geq B(T')$



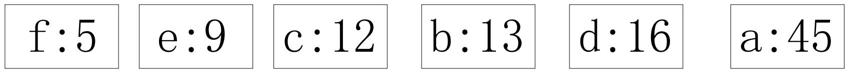
同理可证$B(T') \geq B(T'')$. 于是$B(T) \geq B(T'')$.

由于T是最优化的，所以$B(T) \leq B(T'')$.

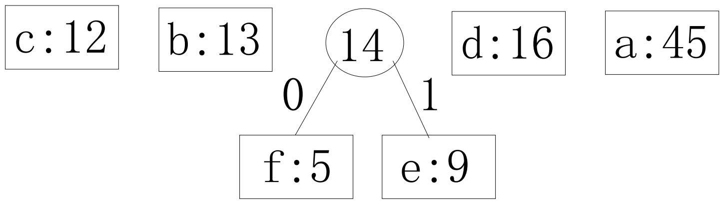
于是, $B(T)=B(T'')$，T''是C的最优化前缀编码树.

在T''中, x和y具有相同最大长度编码,且仅最后位不同.

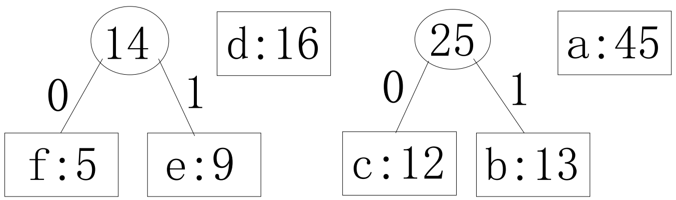
- 基本思想
  - 循环地选择具有最低频数的两个结点，生成一棵子树，直至形成树
  - 例子：f:5, e:9, c:12, b:13, d:16, a:45



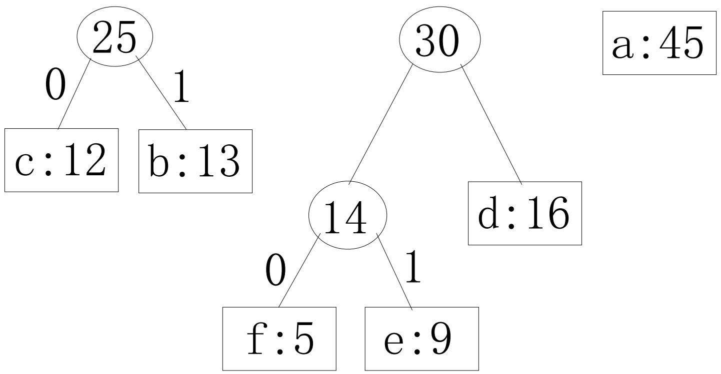
第一步:



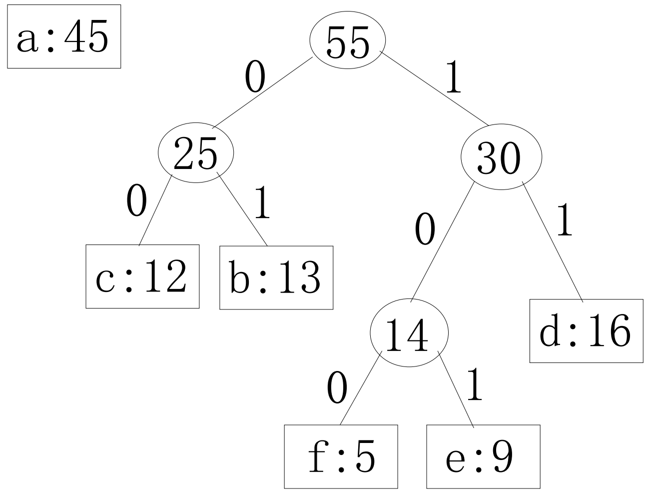
第二步:



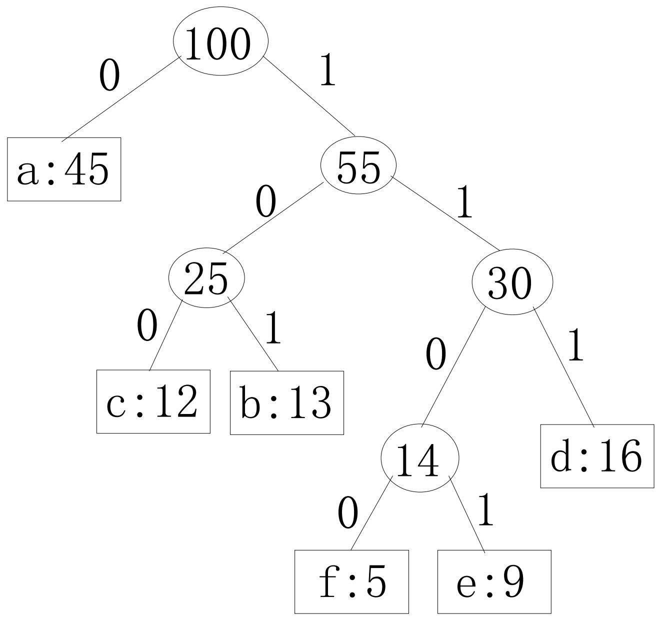
第三步:



第四步:



第五步:



- Greedy算法 (Q是min-heap)

Huffman(C，F)

1. $n \leftarrow |C|$; $Q \leftarrow$ 根据F排序C; T为一个空树;
2. FOR $i \leftarrow 1$ To n-1 Do
3. $z \leftarrow$Allocate-Node( );
4. $left[z] \leftarrow x \leftarrow$Extract-min($Q$) /* 从$Q$删除$x$*/;
5. $right[z] \leftarrow y \leftarrow$Extract-min($Q$) /* 从$Q$删除$y$ */;
6. $f(z) \leftarrow f(x)+f(y)$;
7. Insert(Q, z, $f(z)$);
8. Return Extract-min(Q) /* 返回树的根 */

复杂性分析

```
Huffman(C，F)
1.   n←|C|; Q←C(为C按照字符频数建立堆); T为一个空树；
2.   FOR i←1 To n-1 Do
3.        z←Allocate-Node( );
4.        left[z]← x ←Extract-min(Q) /*从Q删除x*/;
5.        right[z]← y ←Extract-min(Q) /*从Q删除y */;
6.        f(z)←f(x)+f(y);
7.        Insert(Q, z, f(z));
8.   Return Extract-min(Q)  /* 返回树的根 */
```

- 第1步：建堆$O(n)$
- 每次循环：$O(\log n)$，循环n-1次：$O(n\log n)$
- $T(n)=O(n\log n)$

定理3. Huffman算法产生一个优化前缀编码树

证. 由于引理1、引理2成立，

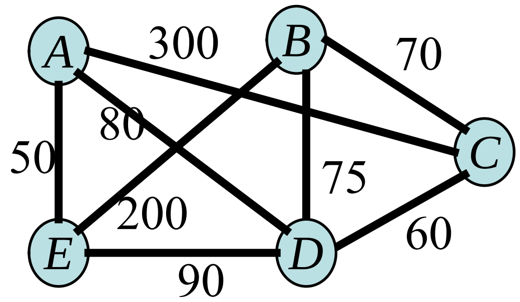
且Huffman算法按照引理2的Greedy选择性确定的规则进行局部优化选择，所以Huffman算法产生一个优化前缀编码树。

5.4 最小生成树问题

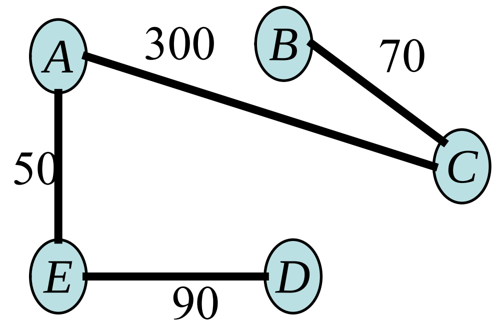
- 问题定义
- 问题求解
  - 设计贪心选择法
  - 优化解结构分析
  - Greedy选择性证明
  - Kruskal算法
  - 算法复杂性

问题的定义

- 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$T=(V, E_T)$, $E_T \subseteq E$.
- 生成树的权
  - 如果 $W: E \rightarrow \{实数\}$ 是G的权函数, T的权值定义为$W(T)=\sum_{(u,v)\in T} W(u,v)$.



- 最小生成树
  - G的最小生成树是$W(T)$最小的G之生成树.

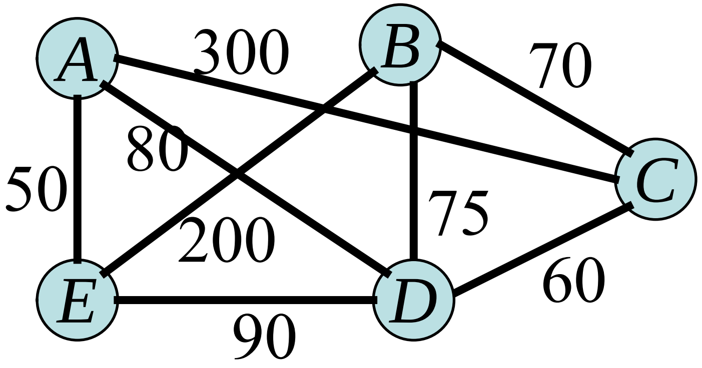


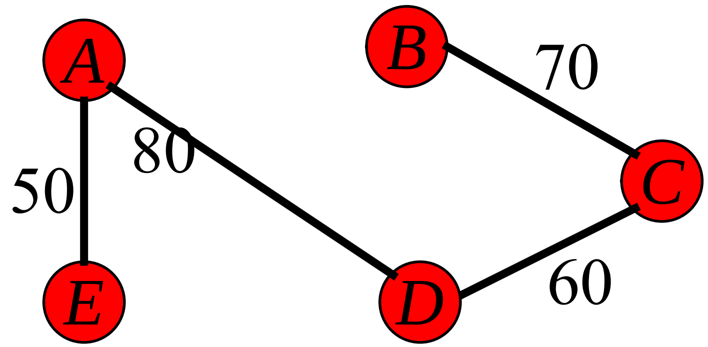
- 问题的定义

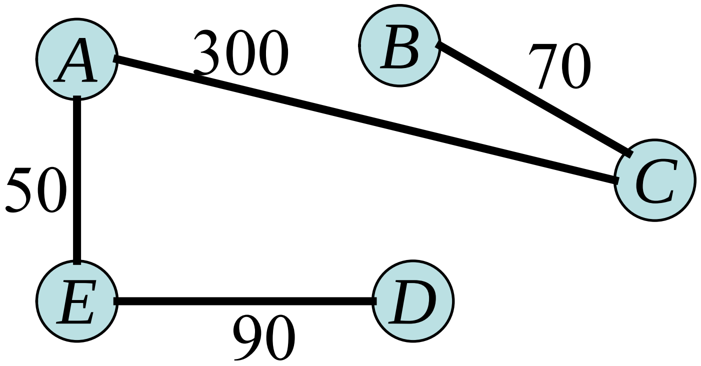
输入: 无向连通图$G=(V, E)$, 权函数W

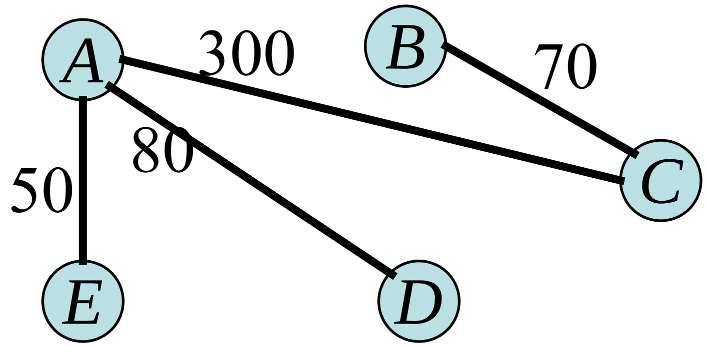
输出: G的最小生成树

• 实例


A
B
C
D
E
300
70
80
50
75
200
60
90

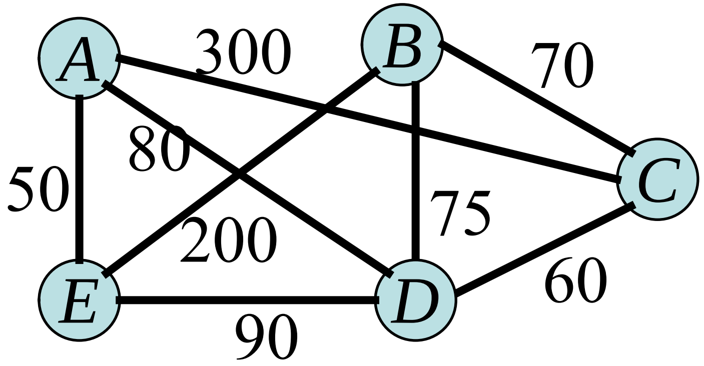

A
B
C
D
E
70
80
50
60


A
B
C
D
E
300
70
50
90

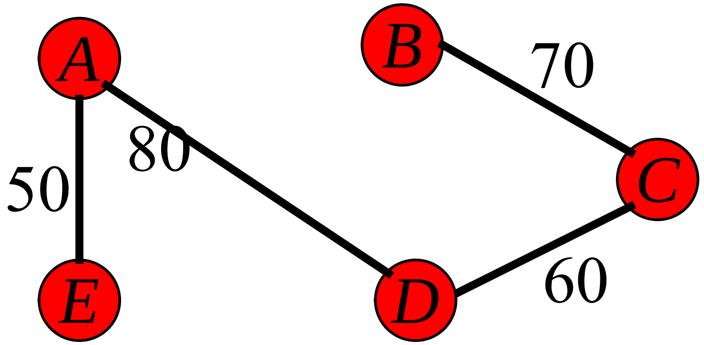

A
B
C
D
E
300
70
80
50

设计贪心选择方法

- 基本思想



- 初始：*A*=空；构造森林$G_A=(V, A)$;



设*A*是一个最小生成树的边子集合，如果$A\cup\{(u, v)\}$也是一个最小生成树的边子集合，则(u, v)称为对*A*是安全的.

- 循环：选择连接G_A中两棵树的最小安全边加入A，直至G_A是一棵生成树.

- 一般算法的定义

Generic-MST(*G*, *W*)

1. $A=\Phi$;

2. While *A* 不是生成树 Do

3. 寻找一个**最小安全边**(u, v);

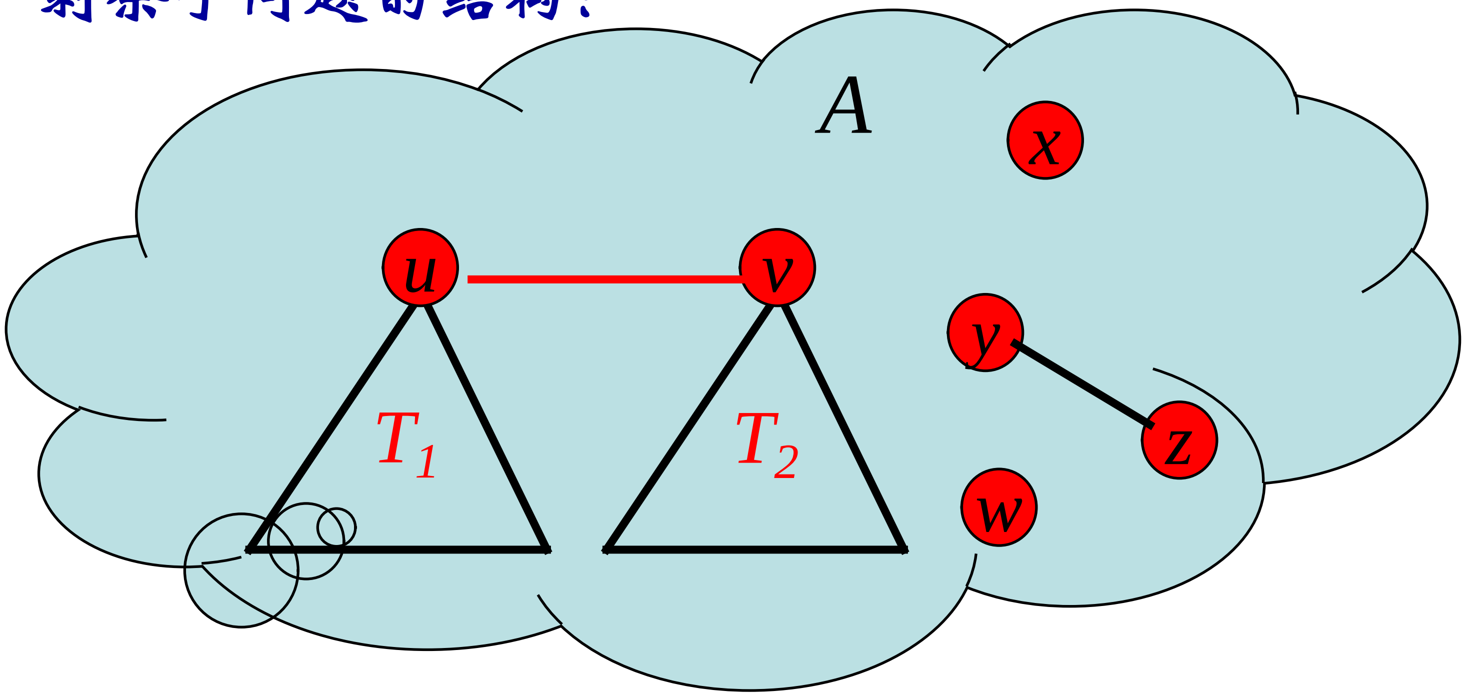
2. $A=A\cup\{(u, v)\}$;

3. Return *A*

算法的关键!

设A是一个最小生成树的边子集合，如果$A\cup\{(u, v)\}$仍是一个最小生成树的边子集合，则(u, v)称为对A是**安全的**.

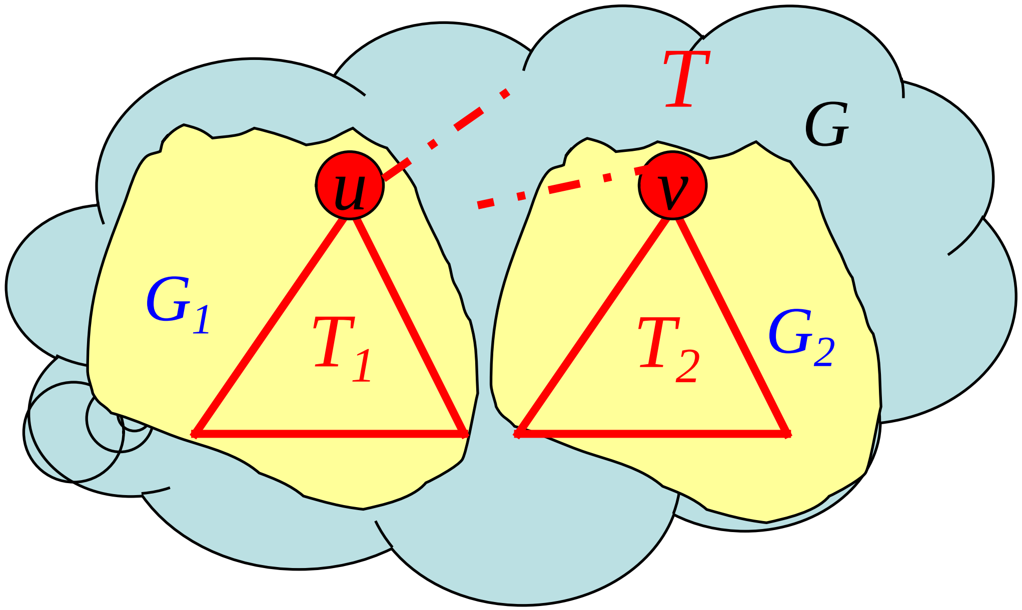
- 贪心选择方法
  - 贪心选择：
    - 从森林中选择一条最小安全边连接两棵子树为一棵子树，直至森林成为一棵树
  - 剩余子问题的结构：



优化解的结构分析

定理1. 设T是G的最小生成树. 如果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. (略)



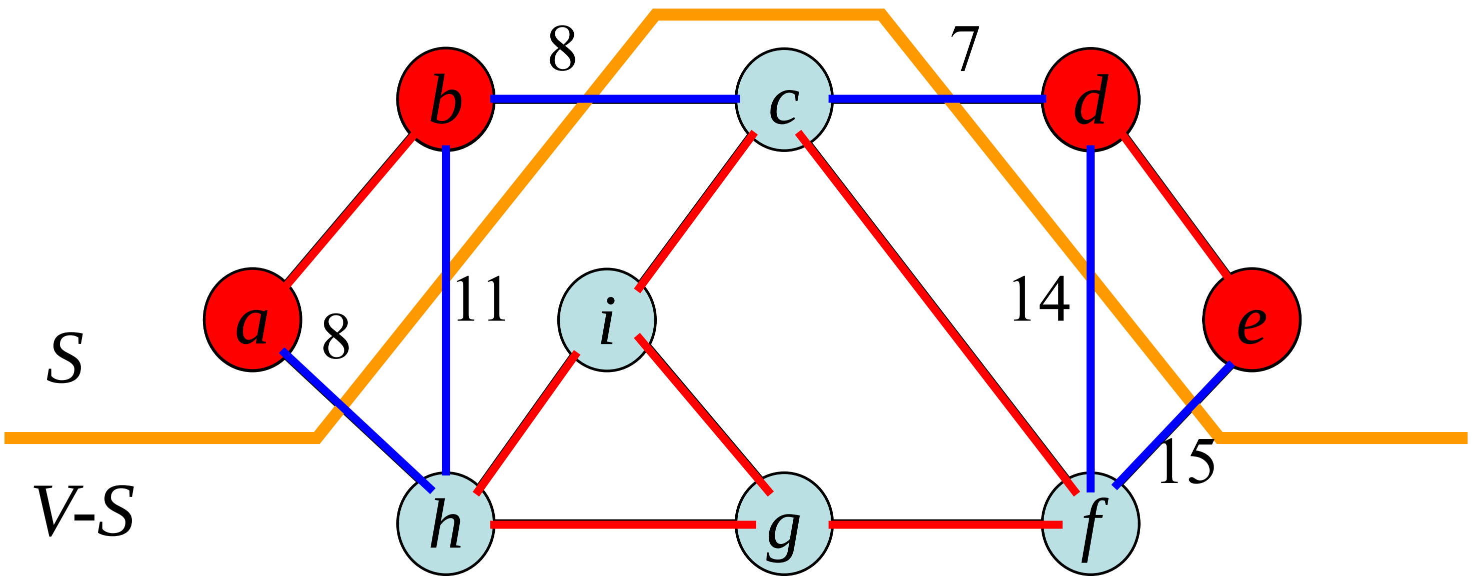
Greedy选择性

定义1. 无向图$G=(V, E)$的一个**划分**是V的划分$(S, V\text{-}S)$.

定义2. 若$u\in S$, $v\in V\text{-}S$, 则(u, v)称为划分$(S, V\text{-}S)$ 的**交叉边**.

定义3. 如果边集合A中没有边是划分$(S, V\text{-}S)$的交叉边, 则称划分**$(S, V\text{-}S)$尊重A**.

定义4. 划分$(S, V\text{-}S)$的交叉边(u, v)称为**轻边**, 如果在所有$(S, V\text{-}S)$的交叉边中, (u, v)的权值最小.



定理2. 设$G=(V, E)$是具有边加权函数W的无向连通图，$A\subseteq E$是包含在G的某最小生成树中的边集合，$(S, V-S)$是G的尊重A的任意划分，(u, v)是$(S, V-S)$的交叉轻边，则(u, v)对A是安全的.

证.

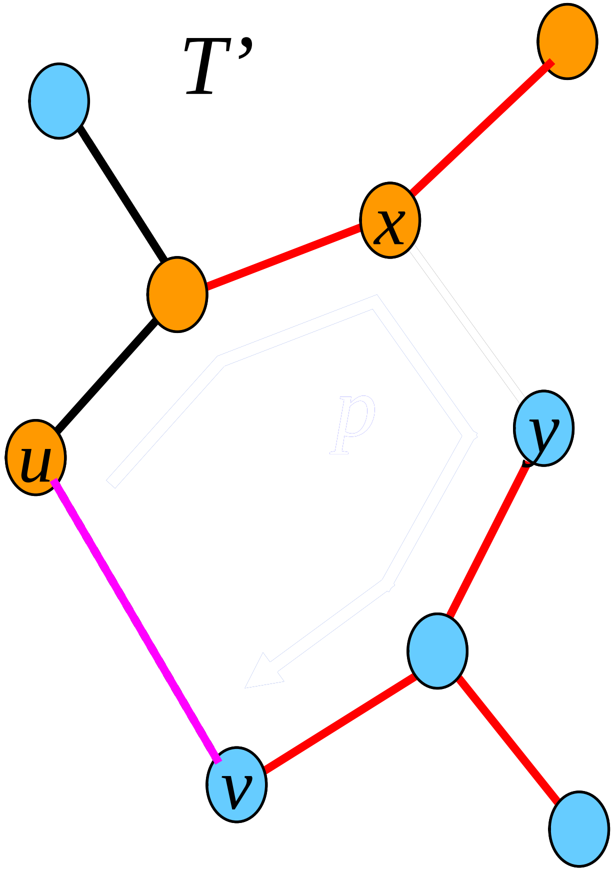
令T是包含A的最小生成树.

若(u, v)属于T，则(u, v)对A是安全的.

若(u, v)不属于T.

我们构造一个G的最小生成树T'，使其包含$A\cup\{(u, v)\}$，从而证明(u, v)安全.

设A是一个最小生成树的边子集合，如果$A\cup\{(u, v)\}$也是一个最小生成树的边子集合，则(u, v)称为对A是安全的.



- S=黄结点集合
- V-S=蓝结点集合
- A=红边集合

由于u和v在划分$(S, V\text{-}S)$ 的两边, T至少存在一条交叉边在从u到v的路径p中,

设该交叉边为(x, y).

由于划分尊重A, 故(x, y)不在A中.

删除p中的(x, y), 增加(u, v), 得到

$$T' = T - \{(x, y)\} \cup \{(u, v)\}.$$

往证T'是最小生成树.

因为(u, v)是交叉轻边, (x, y)是交叉边,

$W(u, v) \le W(x, y)$.

$W(T') = W(T) - W(x,y) + W(u,v) \le W(T)$

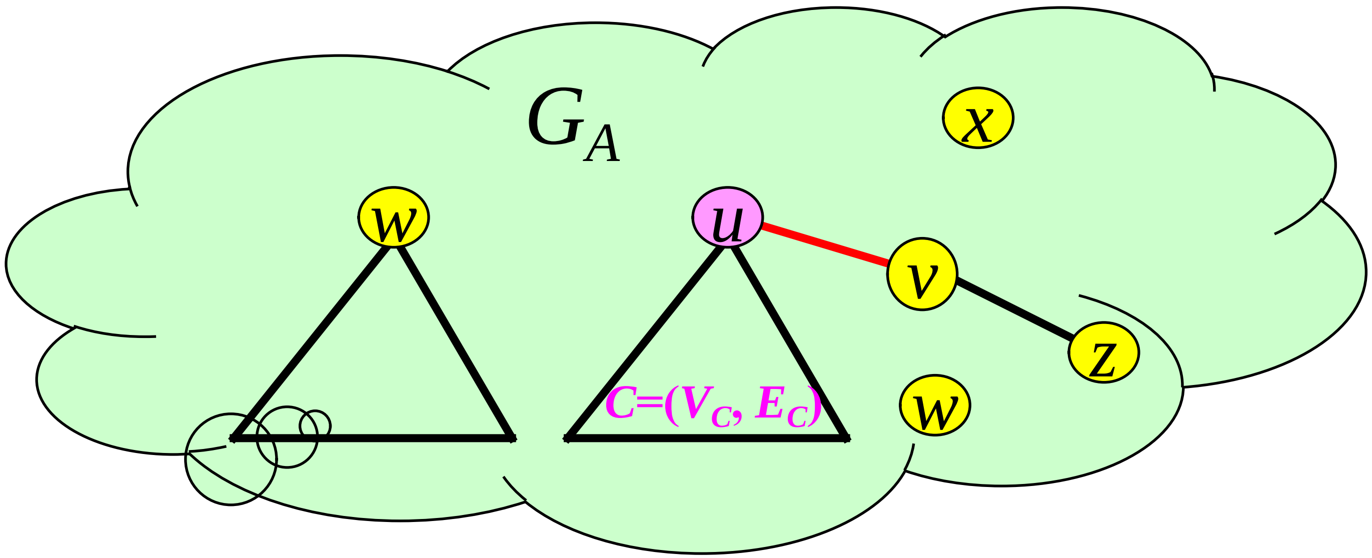
由于T是最小生成树, $W(T') = W(T)$.

T'是最小生成树, $A \cup \{ (u,v) \} \subseteq T'$,

(u, v)对于A是安全的.

推论1. 设$G=(V, E)$是具有边加权函数W的无向连通图，$A\subseteq E$是包含在G的某个最小生成树中的边集合，$C=(V_C, E_C)$是森林$G_A=(V, A)$中的树。如果(u, v)是连接C和G_A中另一个树的交叉轻边，则(u, v)对A是安全的。

证. 划分$(V_C, V-V_C)$尊重A，因为A的边要么在C中，要么在$G_A=(V, A)$的另一个树中。(u, v)是关于这个划分的交叉轻边。于是，(u, v)对A是安全的。



Kruskal算法

- 基本思想

  找到连接森林$G(V, A)$中两棵树的安全交叉轻边加入A，直至$G(V, A)$成为一棵树。

Kruskal算法

MST-Kruskal(G, W)

1. A=Φ;
2. For $\forall v \in V[G]$ Do
3. Make-Set(v); /* 建立只有v的集合 */
4. 按照W值的递增顺序排序$E[G]$;
5. For $\forall (u, v) \in E[G]$ (按W值的递增顺序) Do
6. If Find-Set(u)≠Find-Set(v)
7. Then $A = A \cup \{(u, v)\}$; Union(u, v);
8. Return A

算法复杂性

MST-Kruskal(G, W)
1. A=Φ;
2. For $\forall v \in V[G]$ Do
3. Make-Set(v); /* 建立只有v的集合 */
4. 按照W值的递增顺序排序$E[G]$;
5. For $\forall (u, v) \in E[G]$ (按W值的递增顺序) Do
6. If Find-Set(u)≠Find-Set(v)
7. Then $A = A \cup \{(u, v)\}$; Union(u, v);
8. Return A

- 令$n=|V|$, $m=|E|$
- 第4步需要时间: $O(m\log m)$
- 第2-3步执行$O(n)$个Make-Set操作
  第5-7步执行$O(m)$个Find-Set和Union操作
  需要时间: $O(m\alpha(n))$
- $m \geq n-1$(因为G连通), 由$\alpha(n) < \log n < \log m$
- 总时间复杂性: $O(m\log m)$

集合操作的复杂性见Intro. To Algo. 第21章（498-509）

算法正确性

定理2. MST-Kruskal(G, W)算法能够产生图G的最小生成树.

证. 因为算法按照Greedy选择性进行局部优化选择，并且每次选择的都是最小边.

5.5 贪心算法的基础理论

- Matroid (拟阵)
- Matroid 实例
- Matroid的性质
- 加权Matroid上的Greedy算法
- 算法的正确性

Matroid（拟阵）

- **Matroid定义**

  Matroid是一个序对$M=(S, I)$，满足：

  (1) S是一个有限非空集合.

  (2) $I\subseteq 2^S$, I非空，I中的子集称为S的独立子集.

  (3) 遗传性：如果$A\in I$，$B\subseteq A$，则$B\in I$

  (4) 交换性：如果$A\in I$，$B\in I$，$|A|<|B|$，则

  $\exists x\in B-A$使得$A\cup\{x\}\in I$.

遗传性和交换性是拟阵最根本的2条性质，
拟阵上的其他性质都是基于这2个性质发展出来的

- 实例(Graphic Matroid)

定义1. 设 $G=(V, E)$ 是一个无向连通图，$|E|\geq 1$.由 G 确定的 $M_G=(S_G, I_G)$定义如下: S_G是G的边集合E,

$I_G=\{A|A\subseteq E, G_A(V, A)$是森林$\}$.

定理1. 如果G是一个无向连通图, 则$M_G=(S_G, I_G)$是一个拟阵.

证. (1) S_G非空有限性: $S_G=E$是一个非空有限集合.

(2) I_G的非空性: $\forall e\in E$, $G_e(V, \{e\})$是一个森林，$\{e\}\in I_G$，

于是，I_G是S_G的非空集族.

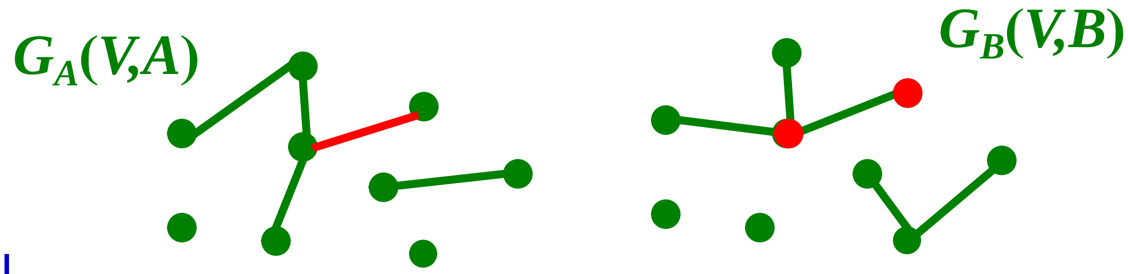
(3) M_G满足遗传性:

如果$A\in I_G$，$B\subseteq A$，则$G_B(V, B)$是一个森林.

于是，$B\in I_G$，M_G满足遗传性.

定理1. 如果G是一个无向连通图，则$M_G=(S_G, I_G)$是一个拟阵.

证(续).



(4) M_G满足交换性:

设$A\in I_G$, $B\in I_G$, $|A|<|B|$.

图论定理：具有k条边的森林具有$|V|-k$棵树.

$G_A=(V, A)$和$G_B=(V, B)$分别具有$|V|-|A|$和$|V|-|B|$棵树.

由于$|V|-|A|>|V|-|B|$, G_B中必存在树T, T的节点在G_A的不同树中.否则G_A中至少有一棵树的节点不在G_B中，与G_B的节点集合为V矛盾.

由于T是连通的, T必包含边(u, v): u, v在G_A的不同树中.

于是, $(V, A\cup\{(u,v)\})$是森林, $A\cup\{(u,v)\}\in I_G$, $(u,v)\in B-A$, M_G满足交换性.

- **Matroid的性质**

定义2. 设$M=(S, I)$是一个Matroid, $A \in I$.如果$A \cup \{x\} \in I$, $x \notin A$, x称为A的一个扩展(extension).

定义3. 设$M=(S, I)$是Matroid, $A \in I$. 若A没有extension, 则称A为最大独立子集合.

定理2. 一个Matroid的所有最大独立子集合都具有相同大小.

证. 设A是Matroid M的最大独立子集合，而且存在M的另一个独立子集合B, $|A|<|B|$.

根据M的交换性, $\exists x \in B-A$使$A \cup \{x\} \in I$, 与A最大矛盾.

加权Matroid上的Greedy算法

- Matroid的最优独立子集

实际背景：

很多可用Greedy算法获得最优解的问题可以归结为在加权Matroid中寻找最优独立子集问题，即给定$M=(S, I)$和权函数W，找到独立子集$A \in I$，使得$W(A)$最大。

- 实例1: 最大生成森林问题
  - 问题定义

    输入：无向图$G=(V, E)$，$|E|>0$，权函数$W: E\rightarrow$正整数集

    输出：边子集合$A\subseteq E$, $G_A(V, A)$是森林，$W(A)$最大
  - 转换为加权Matroid上寻找最优独立子集问题
    - 构造：

      $M_G=(S_G, I_G)$, $S_G = E$, $I_G=\{A|A\subseteq E, G_A(V,A)$是森林$\}$, M_G是拟阵
    - M_G的最优独立子集A满足：
      - (V, A)是森林
      - $W(A)$最大

- 加权Matroid最优独立子集问题

  输入：Matroid $M=(S, I)$, 函数$W: S \rightarrow$正数集

  输出：M的独立子集$A \in I$, 使得$W(A)$最大

设计贪心选择方法

- 贪心选择方法

  从空集合A开始，每次选择权值最大的$x\in S$，扩展A，使$A\cup\{x\}\in I$

引理1.设$M=(S, I)$是一个Matroid. 如果$\{x\}\notin I$, 则x不是任何独立子集的元素.

证. 设x是独立子集A的元素, 即$x\in A$, $A\in I$. 由M的遗传性, $\{x\}\in I$, 矛盾.

推论1.任何元素一旦不能被初始选中, 则永远不会被选中.

设计贪心选择方法

贪心选择方法

从空集合A开始，每次选择权值最大的$x\in S$，扩展A，使$A\cup\{x\}\in I$

第一次选中x之后，剩余子问题：

$S'=S-\{y|\ W(y)>W(x)\}-\{x\}$

$I'=\{B\subseteq S'\ |\ B\cup\{x\}\in I\}$

M'的权函数与M的权函数相同.

(如果元素y一旦没被选中,y不会属于任何最优独立子集)

Greedy选择性

定理1. 设$M=(S, I)$是一个加权Matroid, W是M的权函数, S按W值递减排序. 设x是S中第一个满足$\{x\}\in I$的元素, 则存在一个M的最优独立子集A, $x\in A$.

证. 若存在最优独立子集A包含x, 则定理得证.

否则, 设B是任意非空最优独立子集, $x\notin B$.

S中x之前的元素y必不在B中, 否则由遗传性, $\{y\}\in I$, 与"x是S中第一个满足$\{x\}\in I$的元素"矛盾.

显然, $\forall z\in B$, $W(x)\geq W(z)$. 如下构造含x的优化子集A:

初始: $A=\{x\}\in I$;

用交换性: $\forall z\in B-A$, 若$A\cup\{z\}\in I$, $A=A\cup\{z\}$, 直至$|A|=|B|$.

显然, $\exists w\in B$, $A=(B-\{w\})\cup\{x\}$.

于是, $W(A)=W(B)-W(w)+W(x)\geq W(B)$.

因为B是优化子集, 所以$W(A)\leq W(B)$, $W(A)=W(B)$.

A是优化子集, 且$x\in A$.

定理2. 设x是第一个被Greedy算法选中的元素，A是包含x的最优独立子集，$A'=A-\{x\}$是子问题$M'=(S', I')$的最优独立子集.

$M'=(S', I')$定义如下:

$S'=S-\{y \mid W(y)>W(x)\}-\{x\}$

$I'=\{B\subseteq S' \mid B\cup\{x\}\in I\}$

M'的权函数与M的权函数相同.

证.

因为$A=A'\cup\{x\}\in I$, 所以$A'=A-\{x\}\in I'$. (根据I'的定义)

若A'不是M'的最优独立子集, 则存在M'的一个最优独立子集B使得: $W(B)>W(A')$.

由于$B\cup\{x\}\in I$, $W(A)=W(A')+W(x)$,

$W(B\cup\{x\})=W(B)+W(x)>W(A')+W(x)=W(A)$,

与A最优矛盾.

Greedy(M, W)

1 $A=\Phi$;

2 按权W值递减排序S;

3 For $\forall x \in S$（按$W(x)$递减顺序）DO

4 If $A \cup \{x\} \in I$ /* 选择目前$W(x)$最大的x */

5 Then $A=A \cup \{x\}$;

6 Return A.

时间复杂性

step 2: $O(|S|\log|S|)$

step 4: $O(f(|S|))$

$T(|S|) = O(\ |S|\log|S|+|S|\ f(|S|)\ )$

Greedy正确性

引理2. Greedy算法返回一个独立子集合.

证. 设Greedy返回集合A，对$|A|$做数学归纳法.

当$|A|=0$时, A是空集, 由遗传性, A是独立子集合.

设$|A|\leq k$时, A是独立子集.

当$|A|=k+1$时, A由第4-5步得到, 即$A=A\cup\{x\}$.

第4步已判定$A\cup\{x\}\in I$，$A=A\cup\{x\}$是独立子集.

Greedy正确性

定理3. 设$M=(S, I)$是一个Matroid, W是M的权函数, Greedy(M,W)返回一个M的最优独立子集.

证. ①. 引理1说明, 任何没有被Greedy选中的S元素, 以后不会被选中，可以不再考虑.

②. 算法每次循环都按照定理1给出方法进行贪心选择最大元素x.

③. 算法每次循环都按照定理2的优化子结构求解子问题M'的最优独立子集的问题.

于是，*Greedy*(M, W)返回一个M的最优独立子集.

5.6 任务调度问题

展示如何把一个可用Greedy方法求解的问题转换为求解拟阵最优子集问题

问题定义

- 单位时间任务

  需要一个单位时间就能够完成的任务

- 单位时间任务的调度问题

  输入：

  单位时间任务集$S=\{1, 2, \ldots, n\}$

  正整数任务期限$D=\{d_1, d_2, \ldots, d_n\}$，任务$i$须在$d_i$前完成

  非负权集$W=\{w_1, w_2, \ldots, w_n\}$，任务$i$不在$d_i$前完成罚款$w_i$

  输出：

  S的一个调度(S的一个执行顺序)，具有最小总罚款数

- 转换为加权Matroid的最优子集问题

  定义1. 设S是一个任务调度.任务i在S中是迟任务如果它在d_i之后完成; 否则是早任务.

  定义2. 如果在一个调度中, 早任务总是先于迟任务, 则称该调度具有早任务优先形式.

  定义3. 如果一个调度具有早任务优先形式而且按期限单调递增顺序执行各个早任务, 则称该调度具有规范化形式.

例如：任务集合{1,2,3,4}，期限集合$\{d_1=1,d_2=1,d_3=4,d_4=3\}$，W={4,6,3,8}

可能的任务调度：1,2,4,3 或 2,1,4,3 或1,3,4,2 或1,4,3,2……

其中，1,3,4,2 及 1,4,3,2 是早任务优先调度。

调度1,4,3,2具有规范化形式。

HIT
CS&E

S不是独立集！

例如：任务集合$S=\{1,2,3,4\}$，期限集合$\{d_1=1,d_2=1,d_3=4,d_4=3\}$
$N_1(S)=2$，$N_2(S)=2$, $N_3(S)=3$, $N_4(S)=4$.

若$A=\{1,3,4\}$，则$N_1(A)=1$，$N_2(A)=1$, $N_3(A)=2$, $N_4(A)=3$；

定义4. 任务集合A称为独立集如果存在一个关于A的调度，使得A中的任务皆为早任务.

例. 一个优化调度的早任务集合是独立集.

以下：

用I表示所有独立任务集合的集族

用$N_t(A)$表示任务集合A中期限小于等于t的任务数

HIT
CS&E

任务集合{1,2,3,4}，期限集合{d_1=1,d_2=1,d_3=4,d_4=3}

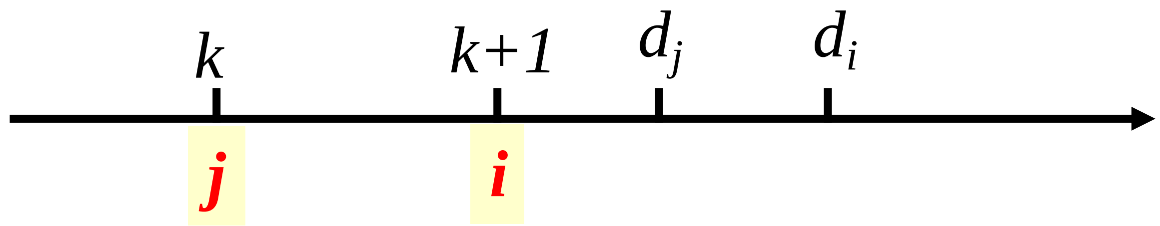
1,2,4,3 ⇒ 1,4,3,2　　1,3,4,2 ⇒ 1,4,3,2

- 任务调度的规范化

第一步：将调度安排成早任务优先形式，即如果早任务x跟在迟任务y之后，交换x和y的位置；

* 早任务优先形式不改变任何任务的早或迟状态

第二步：如果任务i和j是早任务，而且分别完成于时刻k和k+1，d_j<d_i，交换i和j的位置.



寻找最优调度问题成为寻找最优早任务集合A的问题。一旦A被确定后，就可以按期限单调递增序列出A中的所有元素，然后按任意顺序列出迟任务(即-A)

$N_t(A)$ =A中期限小于等于t的任务数

引理1. 对于任意任务集合A, 下面的命题等价:

1. A是独立集合,
2. 对于t=1, 2, …, n, $N_t(A) \le t$,
3. 如果按照期限递增顺序调度A中任务, 则A中无迟任务.

证. 1→2. 如果$N_t(A)>t$, 则有多于t个任务需要在t时间内完成, 不存在使得A中无迟任务的调度.

2→3. 若A中任务依其期限递增排列, 则2意味着排序后, 在时间t之前必须完成的A中任务数至多为t. 于是, 按期限递增顺序调度A中 任务, A无迟任务.

3→1. 显然.

定理1. 若S是一个带期限的单位时间任务的集合，且I为所有独立任务集构成的集族，则$M=(S, I)$是一个Matroid.

证明. 1. S的非空有限性： 显然.

2. I的非空性：

因为单个任务集属于I，所以I非空.

3. 遗传性:

若$A\in I$, $B\subseteq A$, 则$B\in I$.

$N_t(A)$ =A中期限小于等于t的任务数

A={1^1, 3^2, 4^3, 10^7, 8^8}, B={5^1, 2^3, 4^3, 6^5,10^7, 9^{10} , 7^{11}}

i^d表示任务i期限为d，选B中哪个任务加入A？

A'=A∪{9^{10}}={1^1, 3^2, 4^3, 10^7, 8^8 , 9^{10} }

4. 交换性:

设$A, B\in I$, $|B|>|A|$, 往证$\exists x\in B-A$, $A\cup\{x\}\in I$.

由于$|B|>|A|$, 不可能对于所有t, $N_t(B)\le N_t(A)$

令 $k=max\{t \mid N_t(B)\le N_t(A)\}$.

$N_t(B)\le N_t(A)$
Or $N_t(B)>N_t(A)$

$N_t(B)\le N_t(A)$

$N_t(B)>N_t(A)$

$t<k$ $t=k$ $t>k$

于是, B中包含了比A中更多的具有期限$k+1$的任务.

设$x\in B-A$, x具有期限$k+1$.

令$A'=A\cup\{x\}$. 往证A'是独立集.

对于$1\le t\le k$, $N_t(A')=N_t(A)\le t$, 因为A是独立集.

对于$k<t\le n$, $N_t(A'=A\cup\{x\})\le N_t(B)\le t$, 因为B是独立集.

于是, A'是独立集.

- 为$M=(S, I)$定义权值
  - 令$w=\max\{w_1, w_2, \ldots, w_n\}$
  - $\forall x_i \in S,\ W(x_i)=w_i$.
  - $\forall A \in I,\ W(A)=nw-\Sigma_{y\in S-A}W(y)$
- 任务调度问题转换为$M=(S, I)$上寻找最大独立子集（或优化子集）问题
  - 若A是优化子集，则$W(A)=nw-\Sigma_{y\in S-A}W(y)$最大
  - 即罚款$\Sigma_{y\in S-A}W(y)$最小