HIT
MDC

第五章
贪心算法

张开旗
海量数据计算研究中心
计算学部

提要

5.1 贪心算法基本原理

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

贪心算法的基本概念

- 贪心算法总是做出在当前看来最优的选择
- 也就是说贪心算法并不从整体最优考虑，它所做出的选择只是在某种意义上的局部最优选择
- 希望贪心算法得到的最终结果也是整体最优的

贪心算法的基本概念

- 应用实例
  - 兑换硬币问题

已知有5种不同面值的硬币：1元、2角5分、1角、5分、1分

欲兑换钱数：6角7分

目标：用于兑换的硬币个数最少

贪心策略
每次尽可能选择面额最大的硬币
即：当前看来最优的选择

如何兑换？

1. 穷举所有可能性：代价高！
2. 贪心策略：按照面值从大到小选择硬币兑换

2角5分	：	2枚
1角	：	1枚
5分	：	1枚
1分	：	2枚

得到的兑换结果是最优解么？是

是否总能得到最优解呢？

贪心算法的基本概念

- 应用实例
  - 兑换硬币问题

若不同面值的硬币为:1角1分、5分、1分
欲兑换钱数：1角5分
目标：用于兑换的硬币个数最少

贪心策略：每次尽可能选择面额最大的硬币
即：当前看来最优的选择

1角1分 ：1枚
1分 ：4枚

5分 ：3枚
最优策略

得到的兑换结果是最优解么？ No!

贪心算法的基本概念

- 贪心算法
  - 求解优化问题
- 贪心算法的基本思想
  - 求解优化问题的算法包含一系列步骤
  - 每一步都有一组选择
  - 作出在当前看来最优的选择
  - 希望通过作出局部优化选择达到全局优化选择

最终不一定得到全局最优解！

HIT
MDC

贪心算法的基本概念

- 贪心算法不能对所有问题都得到整体最优解，但对许多问题它能产生整体最优解，如：单源最短路经问题、最小生成树问题等
- 在一些情况下，即使贪心算法不能得到整体最优解，其最终结果却是最优解的很好近似

什么情况下可以产生最优解呢？？

- 贪心算法产生优化解的条件
  - 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
  - 贪心选择性
    - 一个优化问题的全局优化解可以通过局部优化选择得到

- 贪心选择性的证明
  - 归纳法
    - 对算法步数归纳或问题规模归纳
      - 证明在每一步做得都比其它算法好，从而最终产生了一个最优解
  - 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
      - 证明贪心算法一定能够找到一个至少与其它最优解一样优化的解

与动态规划方法的比较

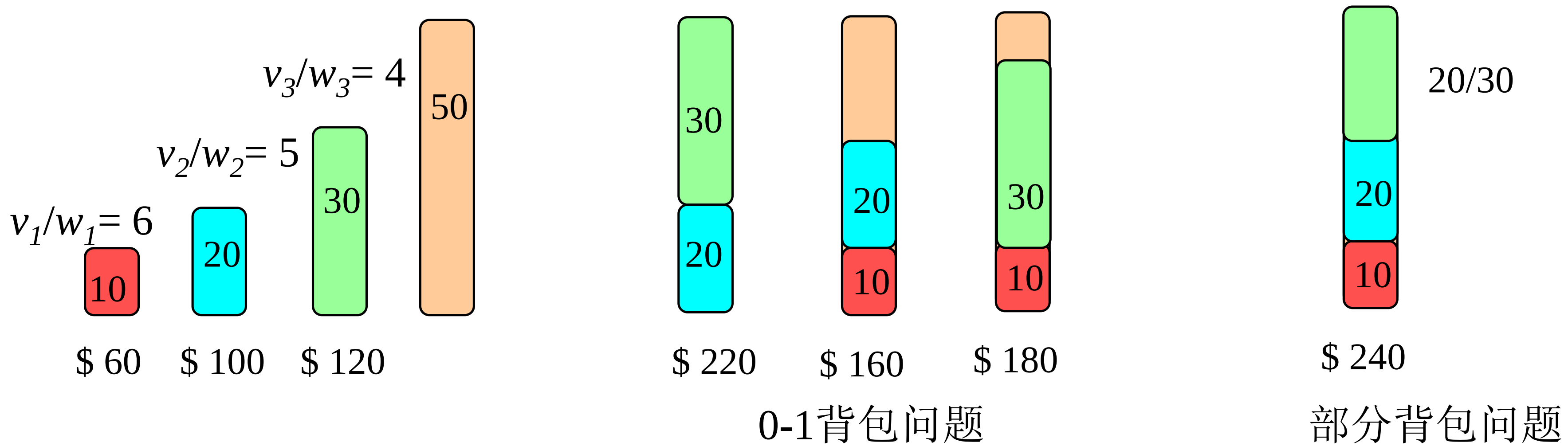
- 动态规划方法
  - 以自底向上方式，先解小子问题，再求解大子问题
  - 在每一步所做的选择通常依赖于子问题的解
- 贪心方法
  - 以自顶向下方式，逐步进行贪心选择，不断减小子问题规模
  - 在每一步先做出当前看起来最优的选择
  - 然后再求解本次选择后产生的剩余子问题
  - 每次选择既不依赖于子问题的解，也不依赖于未来的选择

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
- 贪心方法可用的条件
  - 优化子结构
  - 贪心选择性
- 可用贪心方法时，动态规划方法可能不适用
- 可用动态规划方法时，贪心方法可能不适用

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！
    - 计算每个物品每千克价值v_i/w_i，并按照每千克价值由大到小顺序取物品



准确贪心算法的设计步骤

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

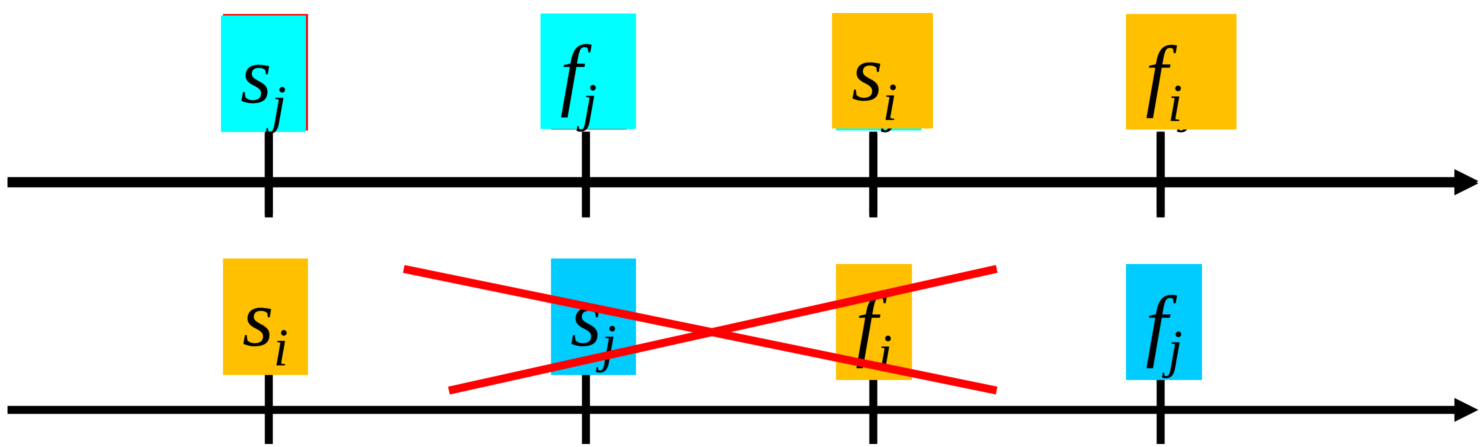
2. 证明：对于1中贪心选择算法来说，所求解的问题具有优化子结构
3. 证明：对于1中贪心选择算法来说，所求解的问题具有贪心选择性
4. 按照1中设计的贪心选择方法设计算法

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
  - 贪心选择性证明
  - 算法设计
  - 算法复杂性分析

问题的定义

- 活动
  - 设$S=\{1,2,\ldots,n\}$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，$s_i \le f_i$
- 相容活动
  - 活动i和j是相容的，若$s_j \ge f_i$或$s_i \ge f_j$，即



问题的定义

- 活动选择问题
  - 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

每次选择具有最小结束时间的活动f_i

剩余子问题：

$$S' = \{ j \in S \mid s_j \geq f_i \}$$

- 算法

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

Greedy-Activity-Selector(S, F)

$n \leftarrow$length(S);

$A \leftarrow \{1\}$

$j \leftarrow 1$

For $i \leftarrow 2$ To n Do

If $s_i \geq f_j$

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

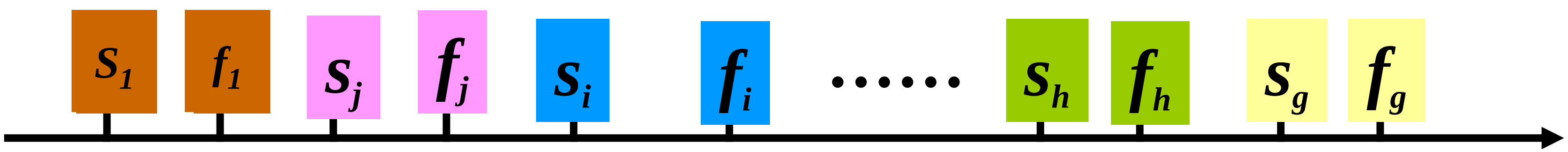
Return A

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$f_1 \le f_2 \le \ldots \le f_n$。

则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$k \ne 1$，令$B=A-\{k\} \cup \{1\}$,

由于A中活动相容，$f_1 \le f_k \le s_j$，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.

引理2. 设$S=\{1, 2, ..., n\}$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$S$的选择问题的一个优化解且包括活动1，则$A'=A-\{1\}$是$S'=\{i \in S | s_i \geq f_1\}$的选择问题的一个优化解.

证. 显然，A'中的活动是相容的.

我们仅需要证明A'是最大的.

设不然，存在一个S'的活动选择问题的优化解B'，$|B'|>|A'|$.

令$B=\{1\} \cup B'$. 对于$\forall i \in S'$, $s_i \geq f_1$, B中活动相容.

因此，B是S的一个解.

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

引理2说明活动选择问题具有优化子结构

引理3. 设 $S=\{1, 2, ..., n\}$ 是 n 个活动集合，$f_1 \le f_2 \le \ldots \le f_n$， $f_{l_0}=0$，
l_i 是 $S_i=\{ j \in S \mid s_j \ge f_{l_{i-1}}\}$ 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 $A=\bigcup_{i=1}^{k}\{l_i\}$

$S_1=\{1,2,3,4,5,6,7,8,9,10,11\}$, $l_0=0$	$l_1=1$
$S_2=\{4, 6,7,8,9,11\}$, $l_1=1$	$l_2=4$
$S_3=\{8,9,11\}$, $l_2=4$	$l_3=8$
$S_4=\{11\}$, $l_3=8$	$l_4=11$

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

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

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

……

$S_{k-1}=\{ j \in S \mid s_j \ge f_{l_{k-2}}\}$

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

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

引理3. 设 $S=\{1, 2, \ldots., n\}$ 是 n 个活动集合, $f_1 \le f_2 \le \ldots \le f_n$, $f_{l_0}=0$,

l_i 是 $S_i=\{ j \in S \mid s_j \ge f_{l_{i-1}}\}$ 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 $|A|\le k\text{-}1$ 时, 命题成立.

当 $|A|=k$ 时, 由引理2, $A=\{l_1=1\}\cup A_1$.

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

$|A_1|=k\text{-}1$, 由归纳假设, $A_1=\bigcup_{i=2}^{k}\{l_i\}$.

于是, $A=\bigcup_{i=1}^{k}\{l_i\}$.

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

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

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

……

$S_{k-1}=\{ j \in S \mid s_j \ge f_{l_{k-2}}\}$

$S_k=\{ j \in S \mid s_j \ge f_{l_k}\}$

- 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：
    - $S' = \{ j \in S \mid s_j \geq f_i \}$

- 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 $s_i \ge f_j$

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

Return A

- 如果结束时间已排序

$T(n)=O(n)$

- 如果 结束时间未排序

$T(n)=O(n)+O(nlogn)$
$=O(nlogn)$

定理1. Greedy-Activity-Selector算法能够产生最优解.

证.

(1) 由引理2可知活动选择问题具有优化子结构

(2) 由引理3知贪心选择方法具有贪心选择性

(3) Greedy-Activity-Selector算法确实按照引理3的贪心选择性进行局部优化选择.

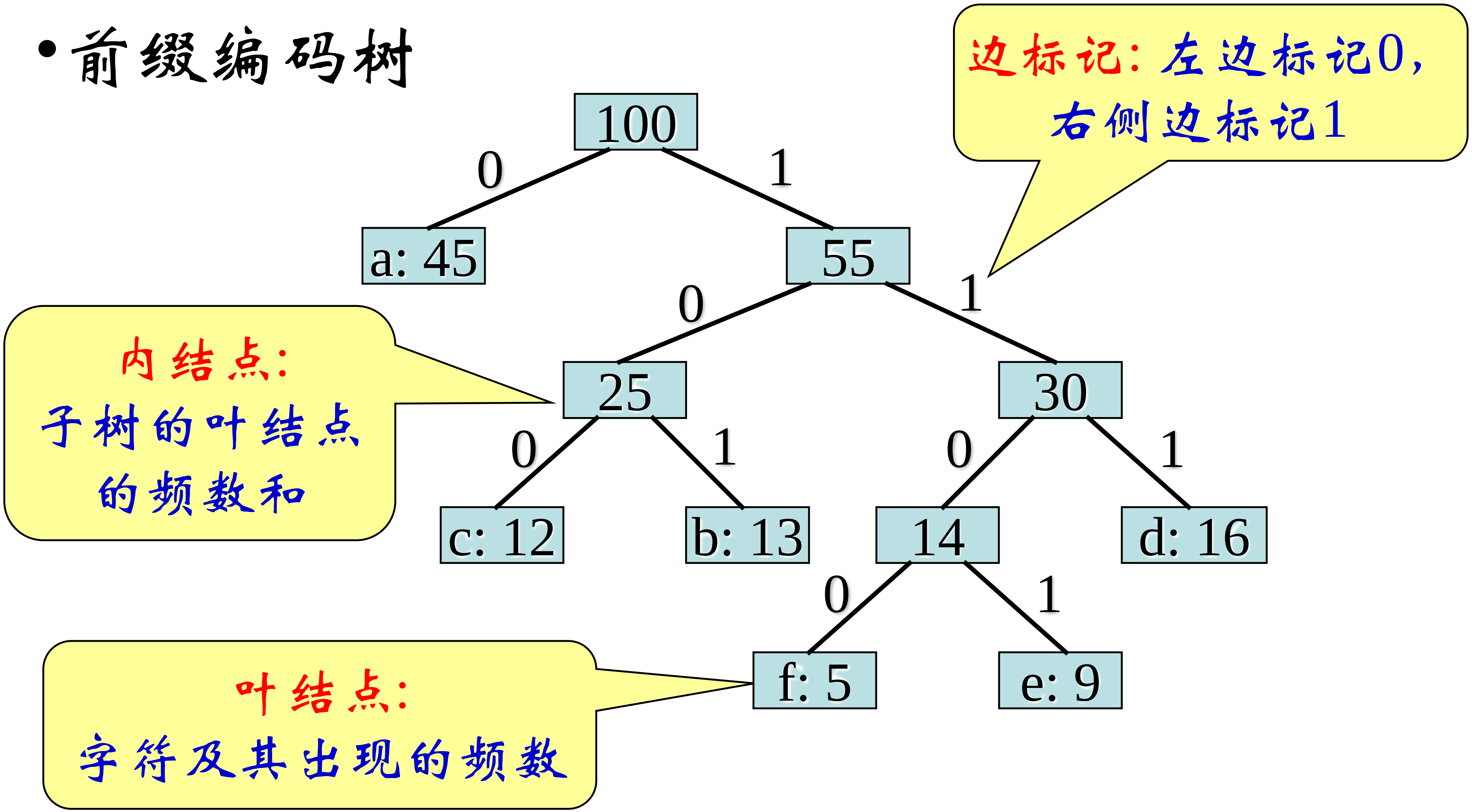
5.3 哈夫曼(Huffman)编码

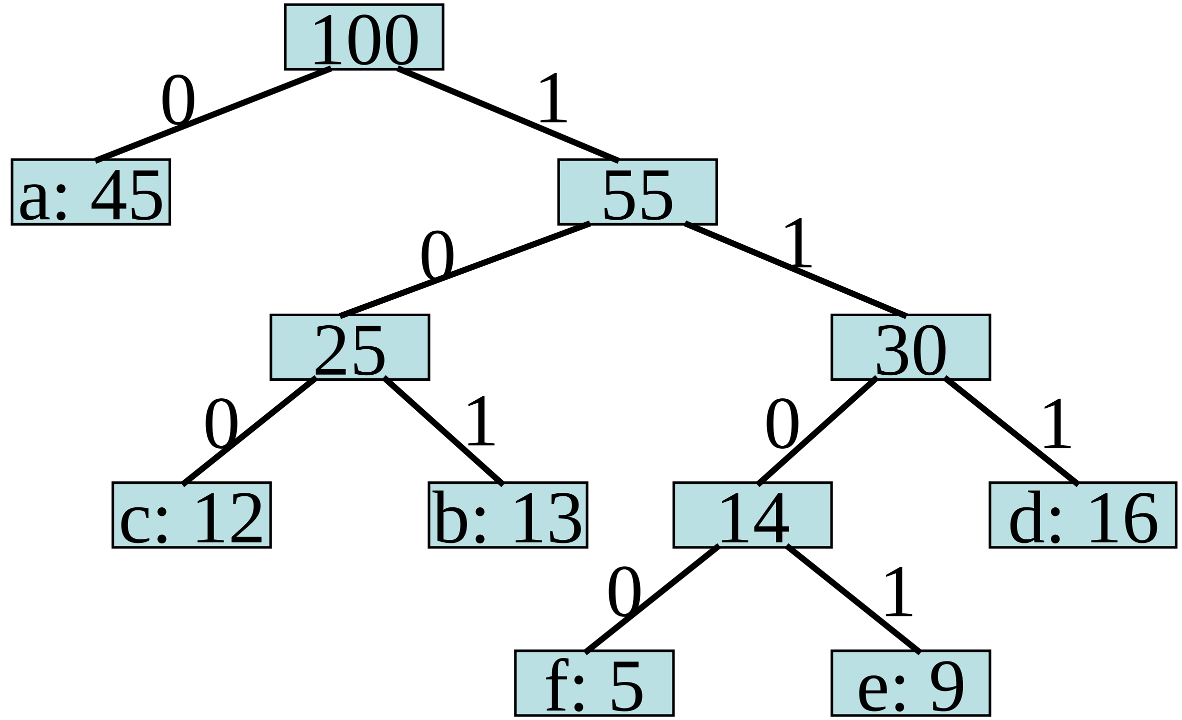
- 问题定义
- 问题求解
  - 设计贪心选择方法
  - 优化解的结构分析
  - 贪心选择性证明
  - 算法设计
  - 算法复杂性分析

问题的定义

- 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

- 前缀编码树


边标记：左边标记0，
右侧边标记1
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
内结点：
子树的叶结点
的频数和
叶结点：
字符及其出现的频数



- 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长度(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的前缀编码树

贪心思想：

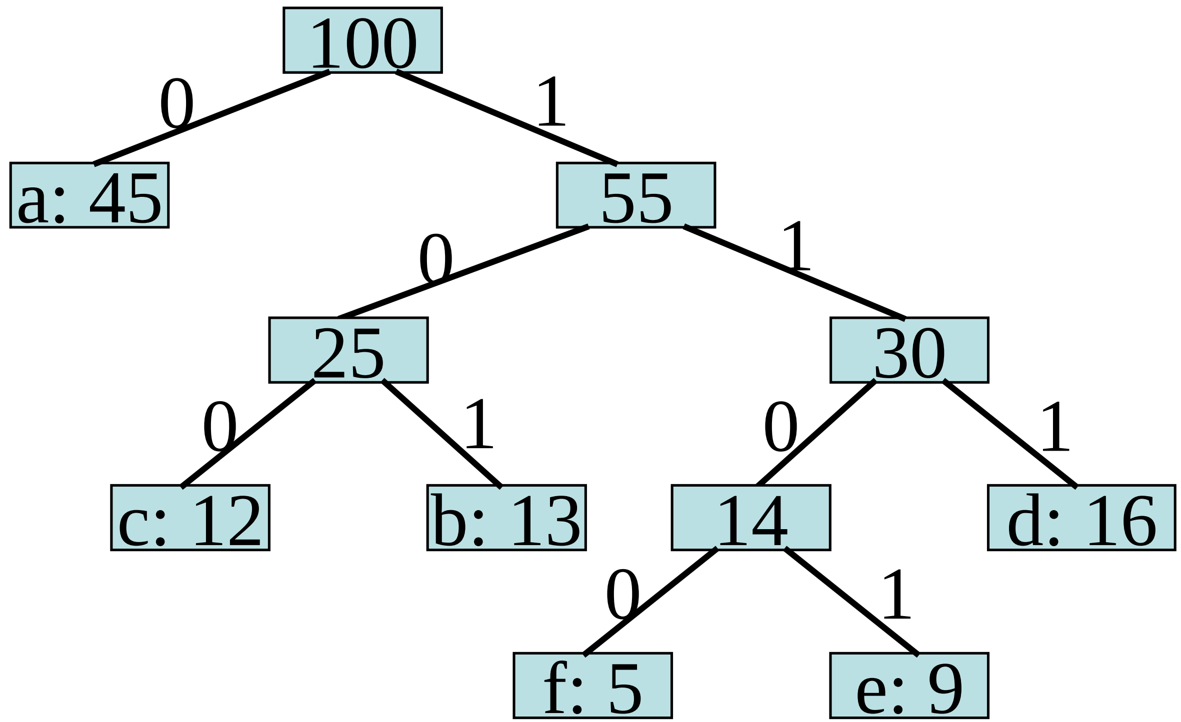
循环地选择具有最低频数的两个结点，生成一棵子树，直至形成树

剩余子问题：？？？

贪心思想：

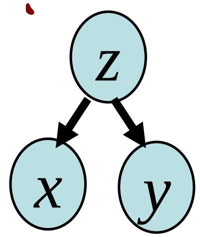
循环地选择具有最低频数的两个结点，
生成一棵子树，直至形成树

f: 5　e: 9　c: 12　b: 13　d: 16　a: 45



设计贪心选择方法

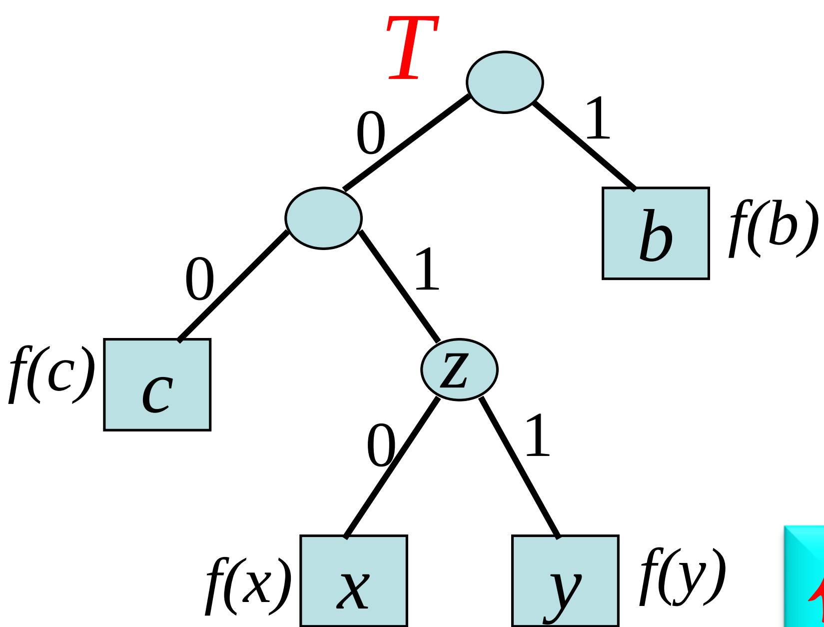
- 贪心选择方法
  - 选择方法：
    - 每次选择具有最低频数的两个节点x和y，构造一个子树：

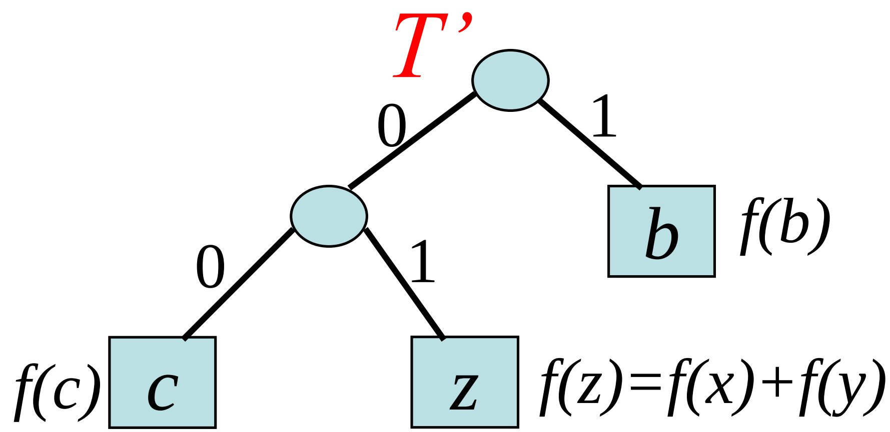


  - 剩余子问题的结构：
    - $C' = C - \{ x, y \} \cup \{ z \}$
    - $F' = F - \{f(x), f(y)\} \cup \{f(z)\}$, $f(z) = f(x) + f(y)$

优化解的结构分析

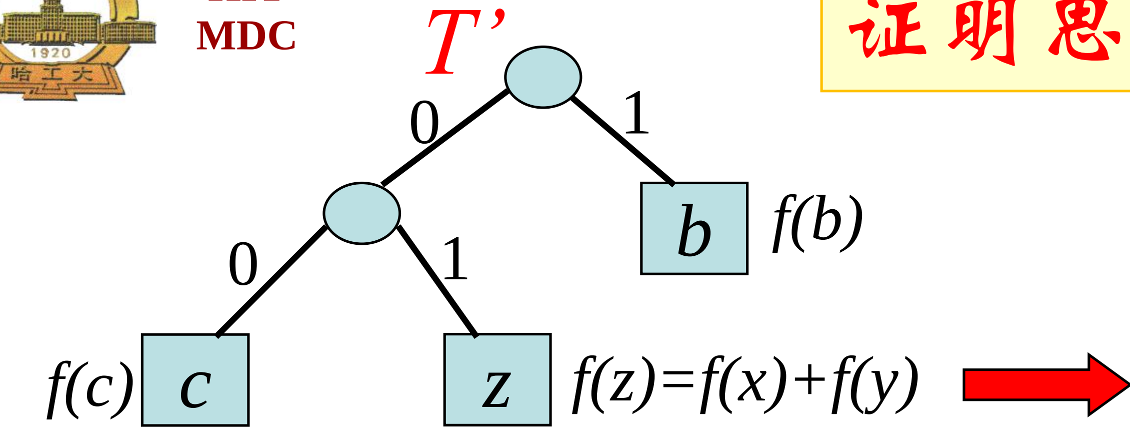
引理1. 设T是字母表C的优化前缀编码树，$\forall c \in C$，$f(c)$是c在文件中出现的频数.设x、y是T中任意两个相邻叶结点，z是它们的父结点，则z作为频数是$f(z)=f(x)+f(y)$的字符，$T'=T-\{x,y\}$是字母表$C'=C-\{x,y\}\cup\{z\}$的优化前缀编码树.

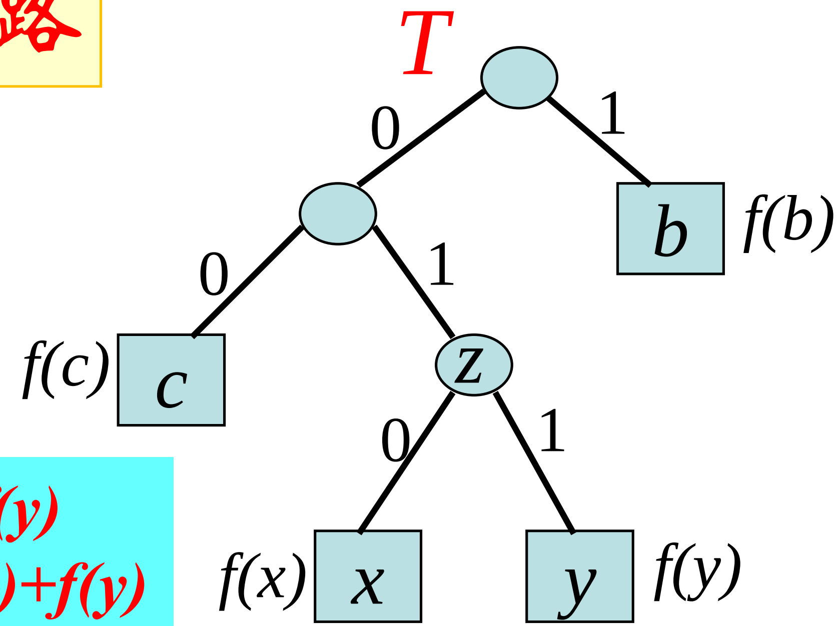




优化前缀树问题具有优化子结构

证明思路



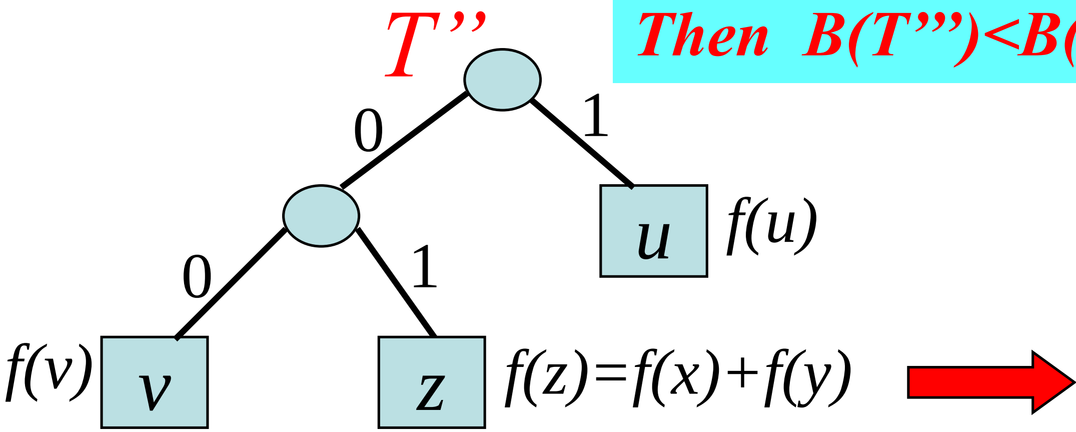


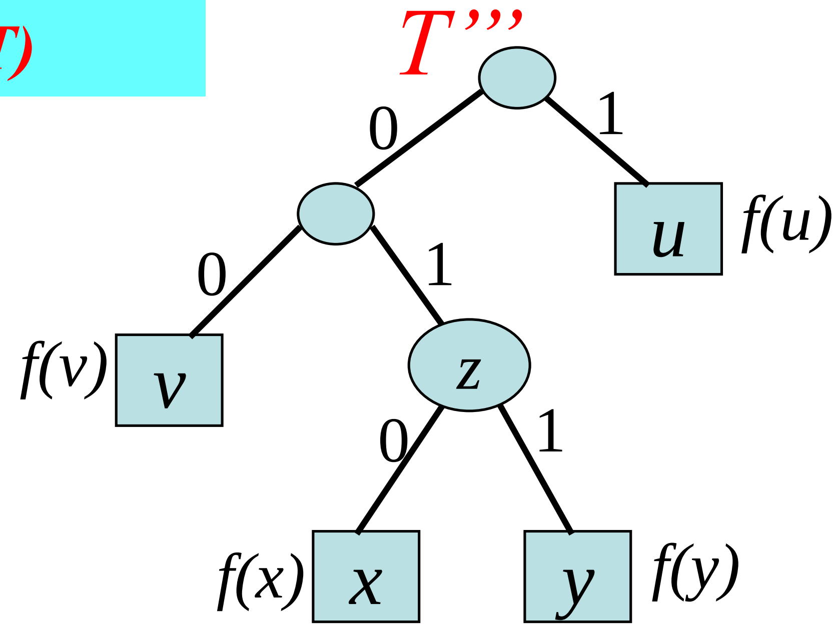
$B(T)=B(T')+f(x)+f(y)$

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

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

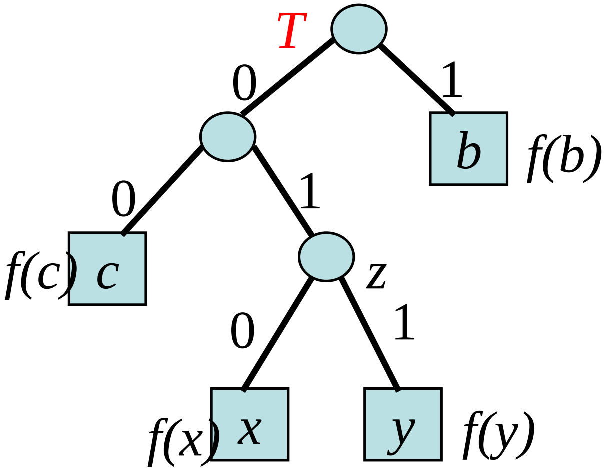
Then $B(T''')<B(T)$





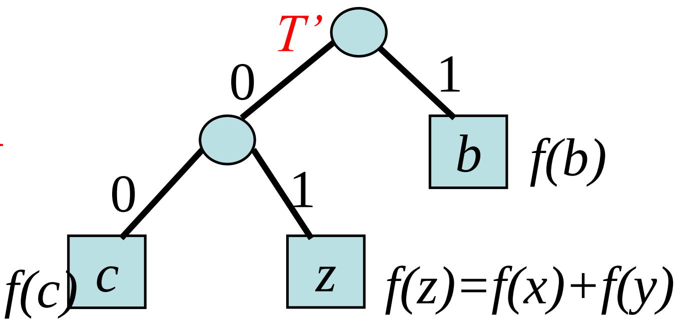
HIT
MDC

证. 往证 $B(T)=B(T')+f(x)+f(y)$.



$$B(T)=f(x)d_T(x)+f(y)d_T(y)+\underline{\Sigma_{k\in C-\{x,y\}}f(k)d_T(k)}$$

$$B(T')= f(z)d_{T'}(z)\ \underline{+\Sigma_{k\in C'-\{z\}}f(k)d_{T'}(k)}$$

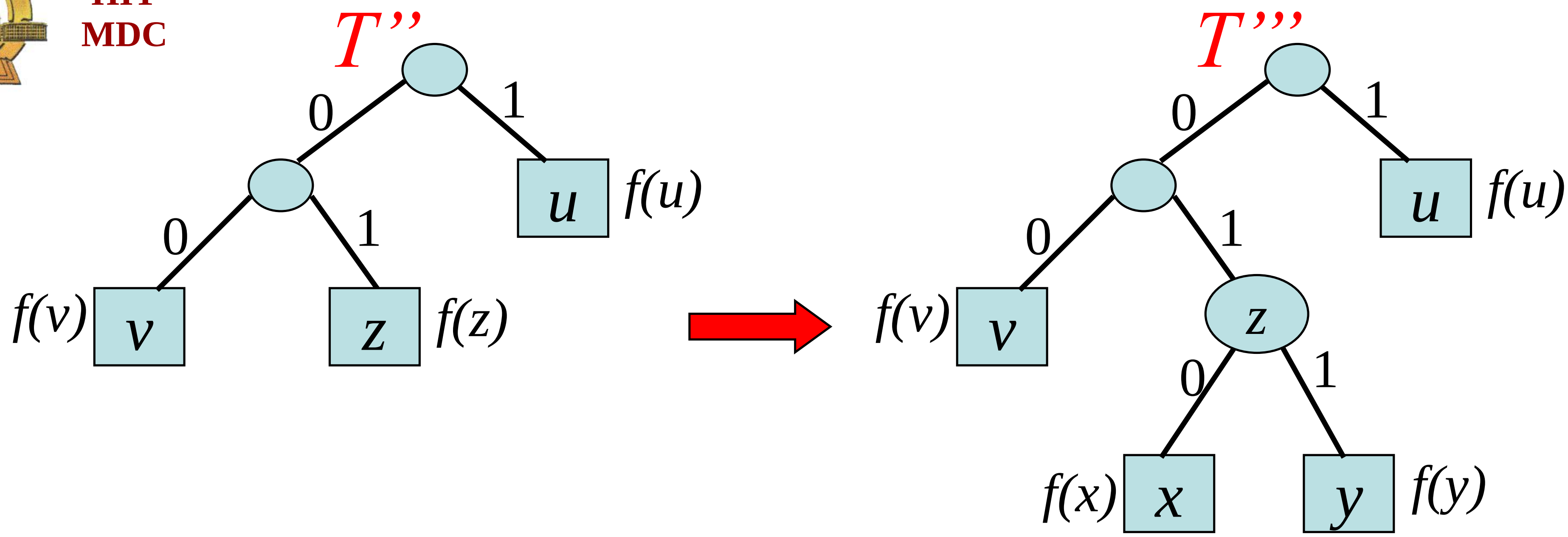


$$B(T)-B(T') = (f(x)d_T(x)+f(y)d_T(y)) - f(z)d_{T'}(z)$$

由于 $f(z) =f(x)+f(y)$, $d_T(x)=d_T(y)=d_{T'}(z)+1$

$$B(T)-B(T') = (f(x)+f(y))(d_{T'}(z)+1) - (f(x)+f(y))d_{T'}(z) =f(x)+f(y).$$

HIT
MDC



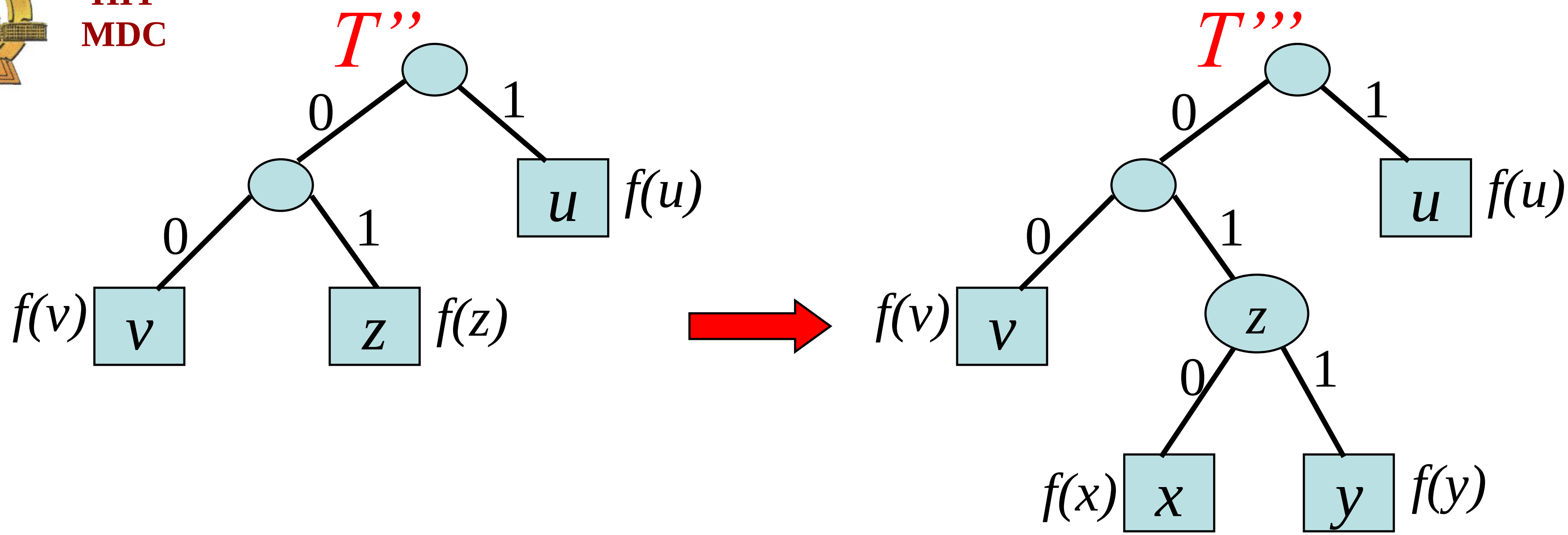
若T'不是C'的优化前缀编码树，

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$B(T'')<B(T')$，

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

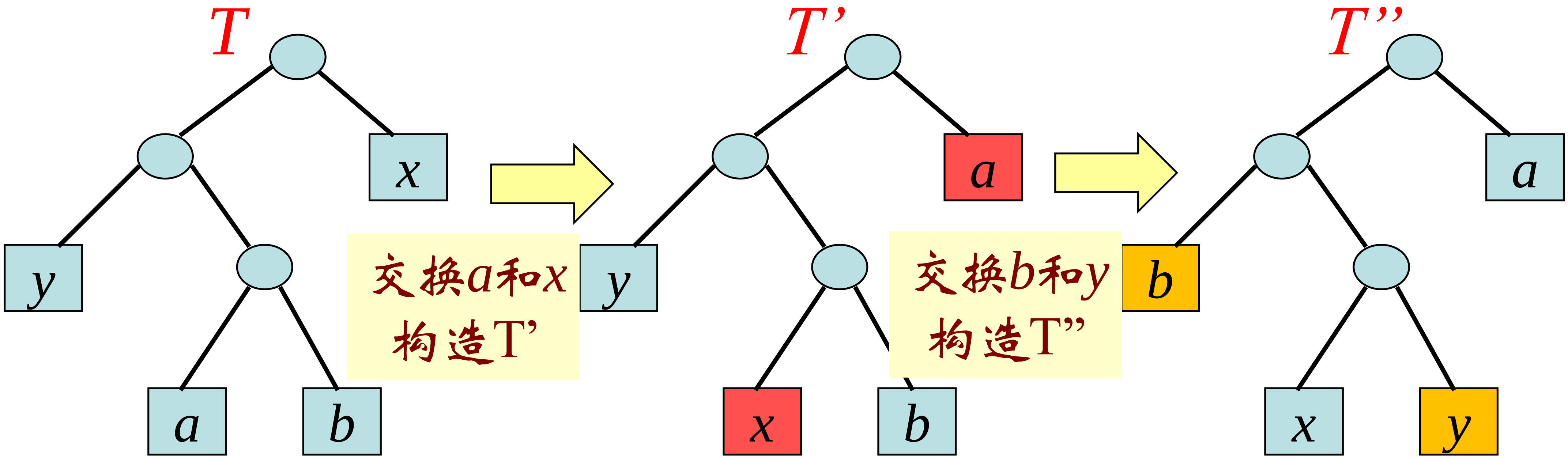
与T是优化的矛盾，故T'是C'的优化编码树.

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贪心选择性.

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a和b是具有最大深度的两个兄弟字符：

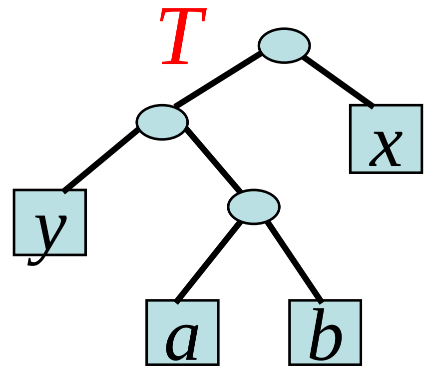


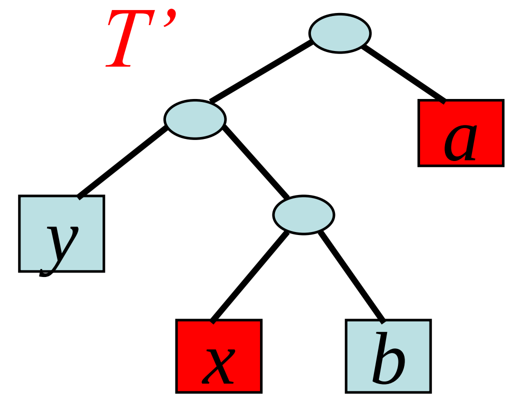
不失一般性，设$f(a) \leq f(b)$, $f(x) \leq f(y)$.

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

交换T的a和x, 从T构造T'； 交换T'的b和y, 从T'构造T''

往证 T'' 是最优化前缀树.





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

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

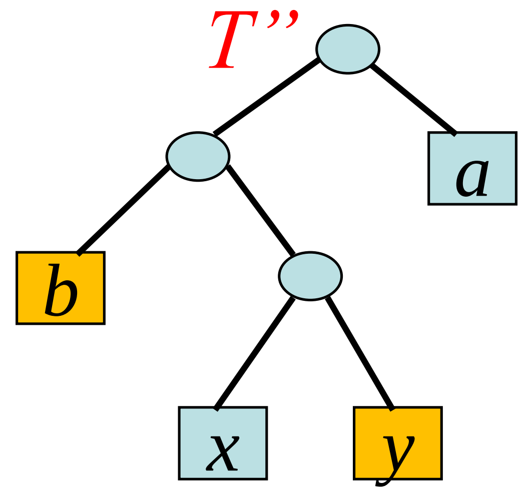
$$= \underline{f(x)d_T(x) + f(a)d_T(a)} - \underline{f(x)d_{T'}(x) - f(a)d_{T'}(a)}$$

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

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

$\because f(a)\geq f(x),\ d_T(a)\geq d_T(x)$ (因为 a 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

- 贪心算法 (Q是min-heap)

Huffman(C，F)

1. $n \leftarrow |C|$; $Q \leftarrow C$; //用堆排序的BUILD-HEAP建堆 $O(n)$
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$x$*/; $O(\log n)$
5. $right[z] \leftarrow y \leftarrow$Extract-min($Q$) /*从$Q$删除$y$ */; $O(\log n)$
6. $f(z) \leftarrow f(x)+f(y)$;
7. Insert(Q, z, $f(z)$); $O(\log n)$
8. Return Extract-min(Q) /* 返回树的根 */

循环n-1次，故：

$T(n)=O(n\log n)$

正确性证明

定理3. 哈夫曼算法产生一个优化前缀编码树

证. 由于引理1、引理2成立，

且哈夫曼算法按照引理2的贪心选择性确定的规则进行局部优化选择，所以哈夫曼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
- 克鲁斯卡尔(Kruskal)算法
  - 设计贪心选择法
  - 优化解结构分析
  - 贪心选择性证明
  - 算法复杂性
- 普里姆(Prim)算法

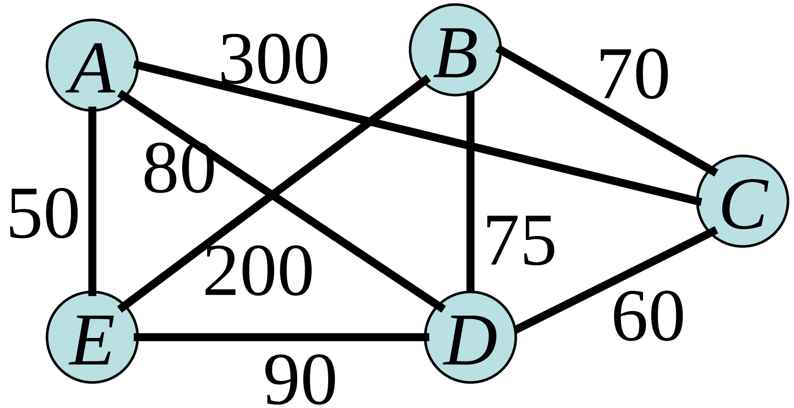
问题的定义

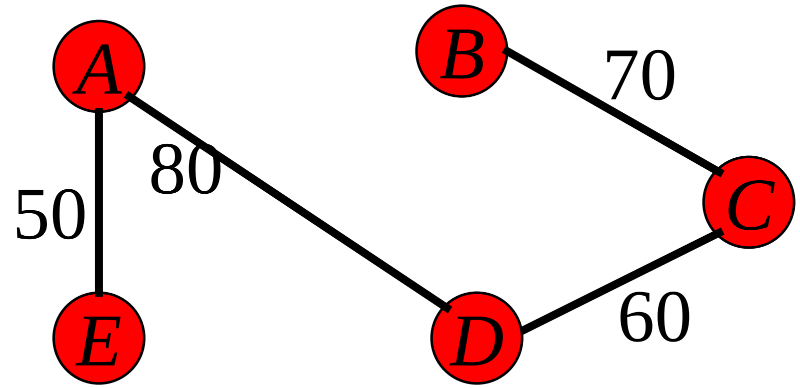
- 生成树
  - 设$G=(V, E)$是一个边加权无向连通图. G的生成树是无向树$T=(V, E_T)$, $E_T \subseteq E$.
  - 如果 $W: E \rightarrow \{实数\}$ 是G的权函数, T的权值定义为$W(T)=\sum_{(u,v)\in T} W(u,v)$.
- 最小生成树
  - G的最小生成树是$W(T)$最小的G之生成树.
- 问题的定义

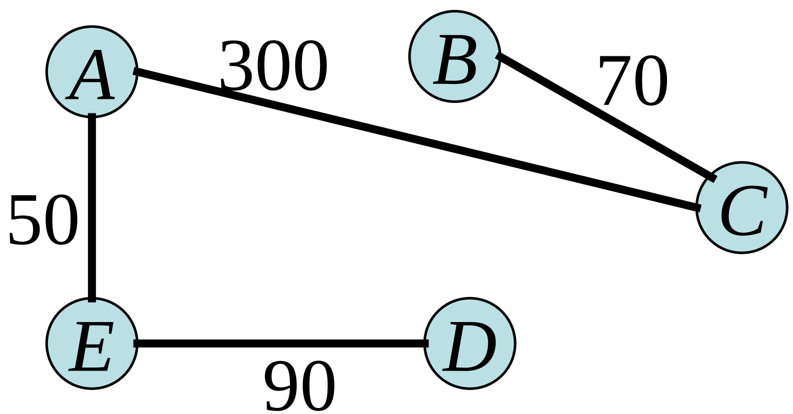
  输入: 无向连通图$G=(V, E)$, 权函数W

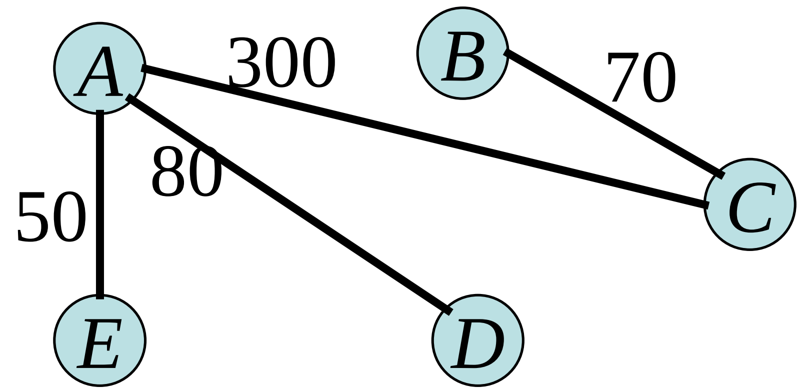
  输出: G的最小生成树

• 实例


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


A
B
C
D
E
70
80
50
60

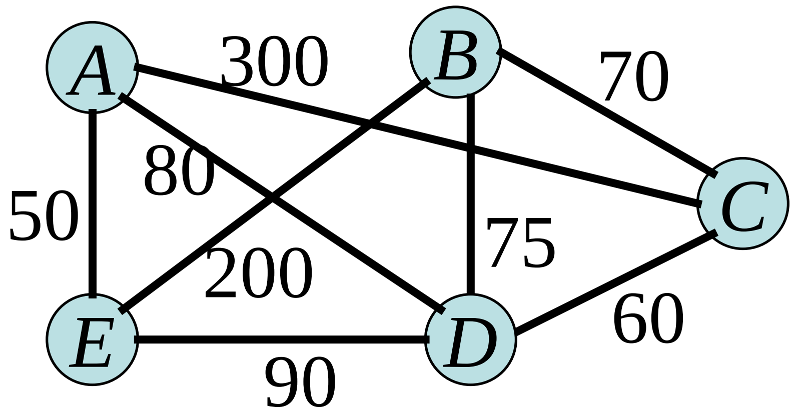

A
B
C
D
E
300
70
50
90


A
B
C
D
E
300
70
80
50

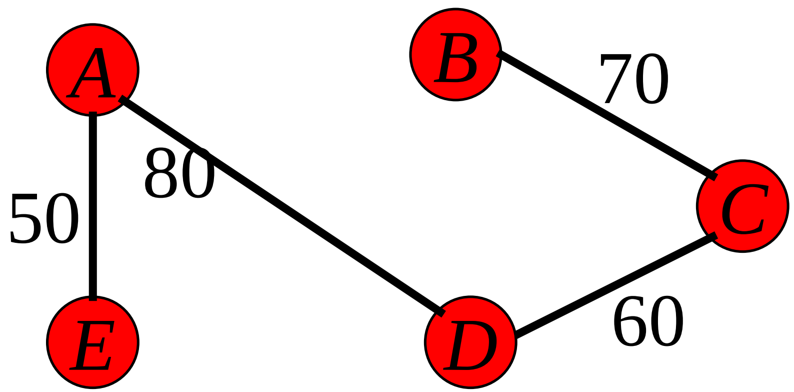
- 克鲁斯卡尔(Kruskal)算法
  - 设计贪心选择方法
  - 优化解结构分析
  - 贪心选择性证明
  - 算法复杂性

设计贪心选择方法

- 基本思想



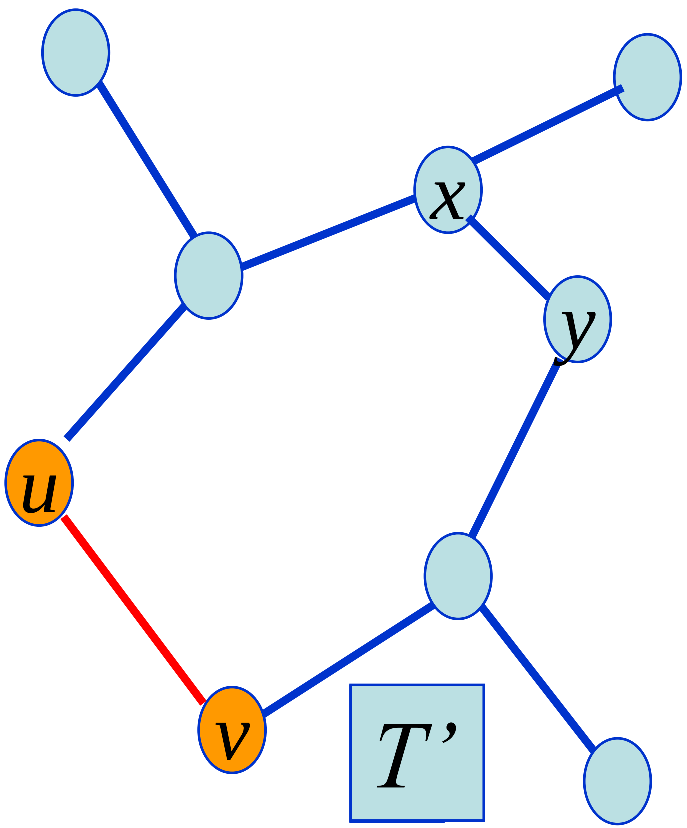
- 初始：S=空；构造森林$G_S=(V, S)$;



剩余子问题?

- 贪心策略：选择连接G_S中两棵树的具有最小权值的边加入S.

定理1. 设(u,v)是G中权值最小的边，则必有一棵最小生成树包含边(u,v).



证明：设T是G的一棵MST

若$(u,v) \in T$, 结论成立；

否则，如左图所示

在T中添加(u,v)边，必产生环

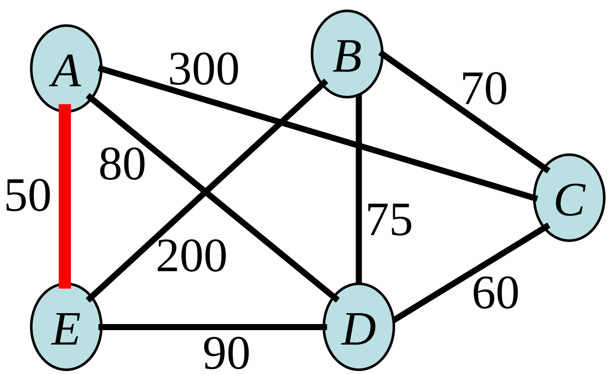
删除环中不同于(u,v)的权值最小的边，设为(x,y)，得到T'.

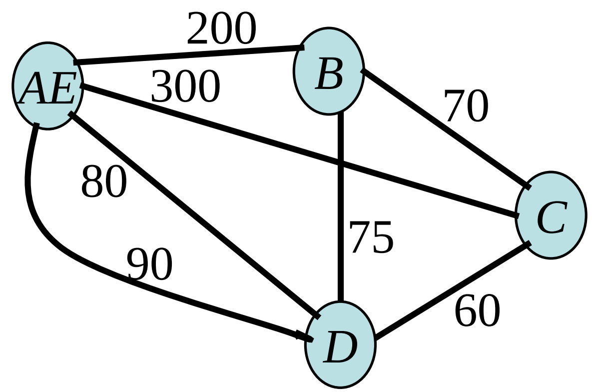
$w(T')=w(T)-w(x,y)+w(u,v) \leq w(T)$

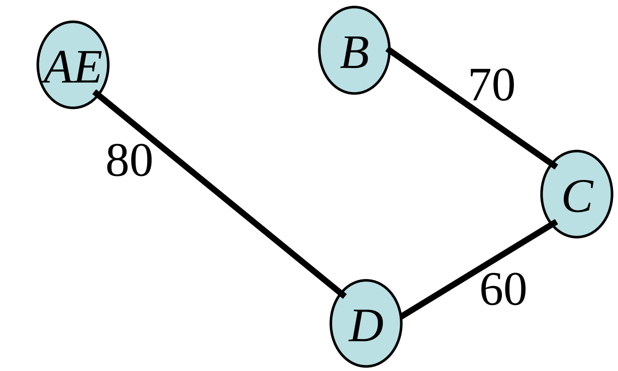
又T是最小生成树, $w(T) \leq w(T')$

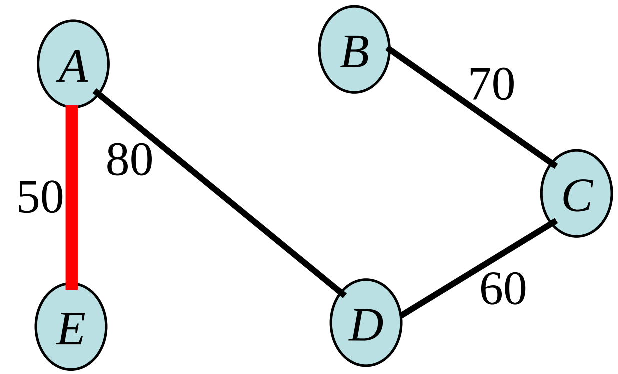
则T'也是一棵MST,且包含边(u,v).

优化解的结构分析









- 图G的边(x,y)的收缩：$G/(x,y)$
  - 用新顶点z代替边(x,y)
  - $\forall v \in V$，用边(z, v)代替边(x, v)或(y, v)
  - 删除z到其自身的边
  - G的其余部分保持不变
- 上述操作的逆操作称为**扩张**，**表示为**$G|_z^{(x,y)}$

定理2. 给定加权无向连通图$G=(V,E)$，权值函数为 $W:E\rightarrow R$，$(u,v)\in E$是G中权值最小的边。设T是G的包含(u,v)的一棵最小生成树，则$T/(u,v)$是$G/(u,v)$的一棵最小生成树.

证明.

由于$T/(u,v)$是不含回路的连通图且包含了$G/(u,v)$的所有顶点，因此，$T/(u,v)$是$G/(u,v)$的一棵生成树。

下面证明$T/(u,v)$是$G/(u,v)$的代价最小的生成树。

若不然，存在$G/(u,v)$的生成树T'使得$W(T')<W(T/(u,v))$。

显然，T'中包含顶点$z=(u,v)$且是连通的，因此$T''=T'|_z^{(u,v)}$包含G的所有顶点且不含回路，故T''是G的一棵生成树。

但，$W(T'')= W(T')+W(uv) <W(T/(u,v))+W(uv)=W(T)$，这与$T$是$G$的最小生成树矛盾。

Kruskal算法

MST-Kruskal(G(V,E), W)

1. A=Φ;
2. For $\forall v \in V$ 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中的边;
5. For $\forall (u, v) \in E$ (按W值的递增顺序) Do
6. If Find-Set(u)≠Find-Set(v) (判断是否出现回路)
7. Then $A=A\cup\{(u, v)\}$; Union(u, v); (合并u,v集合)
8. Return A

算法复杂性

MST-Kruskal($G(V,E)$, W)

1. A=Φ;
2. For $\forall v \in V$ 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;
5. For $\forall (u, v) \in E$ (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2-3步执行$O(n)$个Make-Set操作
- 第4步需要时间: $O(m\log m)$
- 第5-7步执行$O(m)$个Find-Set和Union操作

第2-3步和5-7步需要的时间为:

$O((n+m)\alpha(n))$

$\alpha(n)$是一个缓慢增长的函数

- $m \geq n-1$(因为G连通), 由$\alpha(n)<\log n<\log m$
- 总时间复杂性: $O(m\log m)$

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.

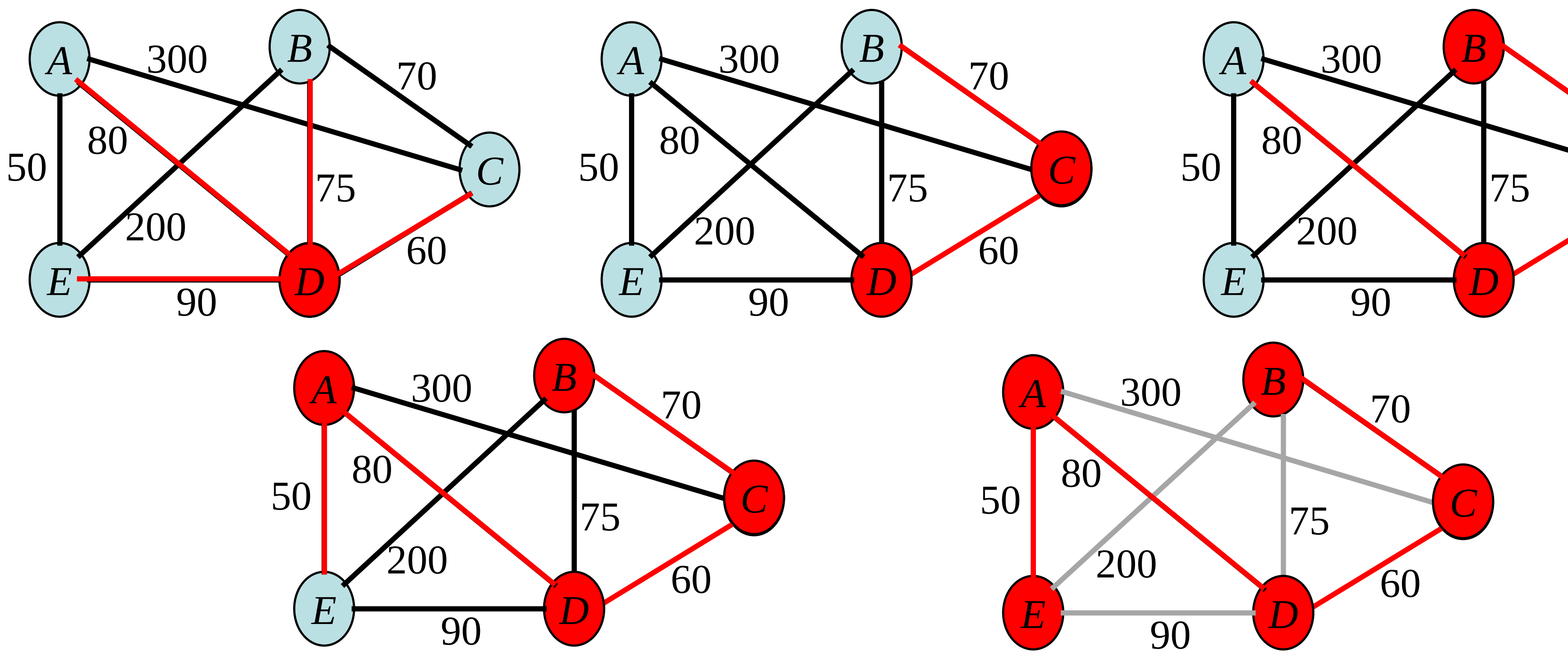
证.

因为算法按照贪心选择性进行局部优化选择，并且每次选择的都是权值最小的边.

- 普里姆(Prim)算法
  - 设计贪心选择方法
  - 优化解结构分析
  - 贪心选择性证明
  - 算法复杂性

设计贪心选择方法

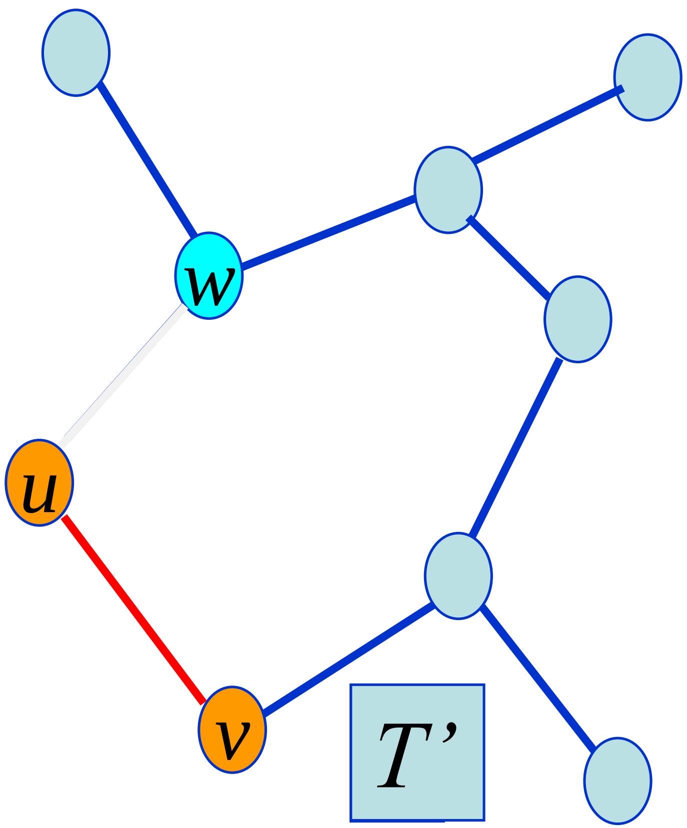
- 贪心策略
  - 以任意顶点v_r作为树根，初始$C=\{v_r\}$
  - 选择C和V-C之间权值最小的边(x,y), $x\in C$, $y\in V$-C
  - $C= C\cup\{y\}$



HIT
MDC

贪心选择性

定理1. 设(u,v)是G中与节点u相关联的权值最小的边，则必有一棵最小生成树包含边(u,v).



证明：设T是G的一棵MST

若$(u,v) \in T$, 结论成立；

否则，如左图所示

在T中添加(u,v)边，必产生环

删除环中与u相关联的边(u,w),

得到T'.

$w(T')=w(T)-w(u,w)+w(u,v) \leq w(T)$

又T是最小生成树, $w(T) \leq w(T')$

则T'也是一棵MST,且包含边(u,v).

定理2. 给定加权无向连通图$G=(V,E)$, 权值函数为 $W:E\rightarrow R$, $(u,v)\in E$是G中与节点u相关联的权值最小的边。设T是G的包含(u,v)的一棵最小生成树，则$T/(u,v)$是$G/(u,v)$的一棵最小生成树.

证明. 略

与Kruskal算法类似

Prim算法

MST-Prim(G,W,r)

Input 连通图G，权值函数W，树根r

Output G的一棵以r为根的生成树

1. $C \leftarrow \{r\}$，$T \leftarrow \varnothing$;
2. 建堆Q维护C与V-C之间的边
3. While $C \neq V$ do
4. $(u,v) \leftarrow$Extract_Min(Q) //$u \in C, v \in V$-C
5. $C \leftarrow C \cup \{v\}$; $T \leftarrow T \cup \{(u,v)\}$;
6. for $\forall x \in$ Adj[v] do **2 |*E*|遍**
7. if $x \in C$ then 将(v,x)从Q中删除 **log |*E*|**
8. Else 将(v,x)插入Q
9. Return T

$T(n)=O(|E| \times \log |E|)$

HIT MDC

Prim算法2

MST-Prim(G, W, r)

Input　连通图G，权值函数W，树根r

Output　G的一棵以r为根的生成树

1. For $\forall\ v \in V[G]$ Do
2. 　　key[v]$\leftarrow +\infty$　　//所有与v邻接的边的最小权值　　**$|V|$**
3. 　　$\pi[v] \leftarrow$null　　//与v邻接的具有最小权值的边
4. key[r] $\leftarrow 0$
5. $Q \leftarrow V[G]$　　**$|V|$**
6. While $Q \neq \varnothing$ do
7. 　　$u \leftarrow$Extract_Min(Q)　　**log $|V|$**
8. 　　for $\forall v \in$ Adj[u] do　　**2$|E|$遍**
9. 　　　　if $v \in Q$ 且 $w(u,v)<$key[v] then　　**常数时间**
10. 　　　　　　$\pi[v] \leftarrow u$
11. 　　　　　　key[v] $\leftarrow w(u,v)$　　//更新信息　　**log $|V|$**
12. Return $A=\{(v, \pi[v]) \mid v \in V[G] - r\}$

$T(n)=O(|E|\times\log|V|)$