计算机算法分析—习题课

2008年5月

第四章：**2 、3、 5、 6、10、11、23**

P99-2

在下列情况下求解 4.1 节的递归关系式

$$T(n)=\begin{cases} g(n) & n\text{足够小} \\ 2T(n/2)+f(n) & \text{否则} \end{cases}$$

当①g(n)=O(1)和 f(n)=O(n);

②g(n)=O(1)和 f(n)=O(1)。

P99-2 递推表达式

设$n=2^k$则:

$$T(n)=T(2^k)=2T(2^{k-1})+f(2^k)$$

$$=2(2\ T(2^{k-2})+f(2^{k-1}))\ +f(2^k)$$

$$=2^2T(2^{k-2})+2^1\ f(2^{k-1})+\ f(2^k)$$

$$=\quad \ldots\ldots\ldots\ldots$$

$$=2^kT(1)+2^{k-1}f(2)+2^{k-2}f(2^2)+\ldots+2^0f(2^k)$$

$$=2^kg(n)+\ 2^{k-1}f(2)+2^{k-2}f(2^2)+\ldots+2^0f(2^k)$$

P99-2 $g(n)=O(1)$和$f(n)=O(n)$

当$g(n)=O(1)$和$f(n)=O(n)$时

不妨设$g(n)=a$，$f(n)=bn$，则：

$$T(n)=T(2^k)$$

$$= 2^k a+ 2^{k-1}*2b+2^{k-2}*2^2 b+\ldots+2^0*2^k b$$

$$=2^k a+kb2^k$$

$$=an+bn\log_2 n=O(n\log_2 n)$$

P99-2 g(n)=O(1)和f(n)=O(1)

当g(n)=O(1)和f(n)=O(1)时，

不妨设g(n)=c，f(n)=d，则：

$T(n)=T(2^k)$

$=c2^k+2^{k-1}d+2^{k-2}d+\ldots+2^0d$

$=c2^k+d(2^k-1)$

$=(c+d)\ n-d=O(n)$

P99-3

- 根据2.2节开始所给出的二分检索策略，写一个二分检索的递归过程。

```
Procedure BINSRCH(A, low, high, x, j)
    integer  mid
    if low≤high then
        mid←⌊(low+high)/2⌋
        if x=A(mid) then
                j←mid;
        endif
        if x>A(mid) then
                BINSRCH(A, mid+1, high, x, j);
        endif
        if x<A(mid) then
                BINSRCH(A, low, mid-1, x, j);
        endif
    else
        j←0;
    endif
end BINSRCH
```

P99-5

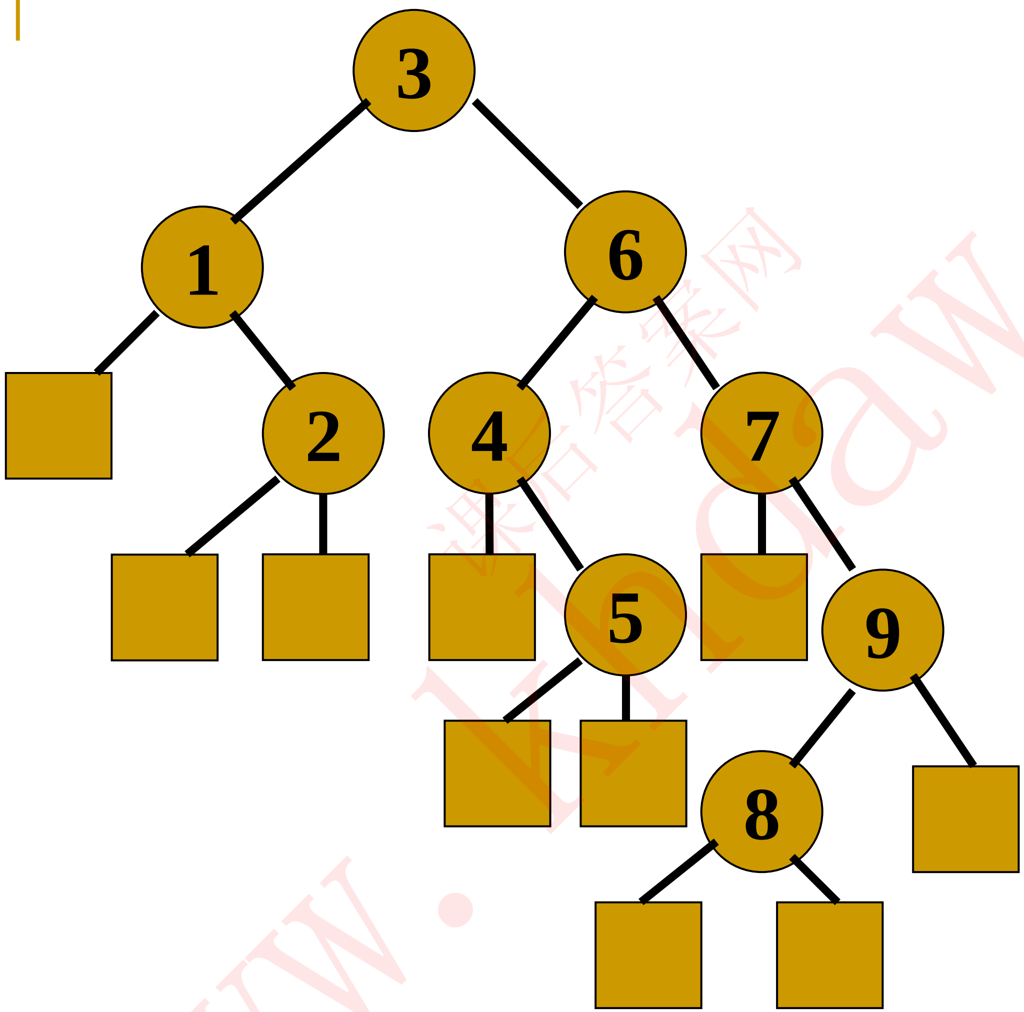
- 作一个“三分”检索算法，它首先检查n/3处的元素是否等于某个x的值，然后检查2n/3处的元素。这样，或者找到x，或者把集合缩小到原来的1/3。分析此算法在各种情况下的计算复杂度。

```
Procedure ThriSearch(A, x, n, j)
    integer low, high, p1, p2
    low←1; high←n
    while low≤high do
      p1←⌊(high+2low)/3⌋
      p2←⌊(2high+low)/3⌋
      case
         :x=A(p1):  j←p1;  return
         :x=A(p2):  j←p2;  return
         :x<A(p1):  high←p1-1
         :x>A(p2):  low ←p2+1
         :else:     low←p1+1;  high←p2-1
```

```
    end  case
  repeat
  j←0
end ThriSearch
```

实例运行

{ 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 }



12 **3** 456789

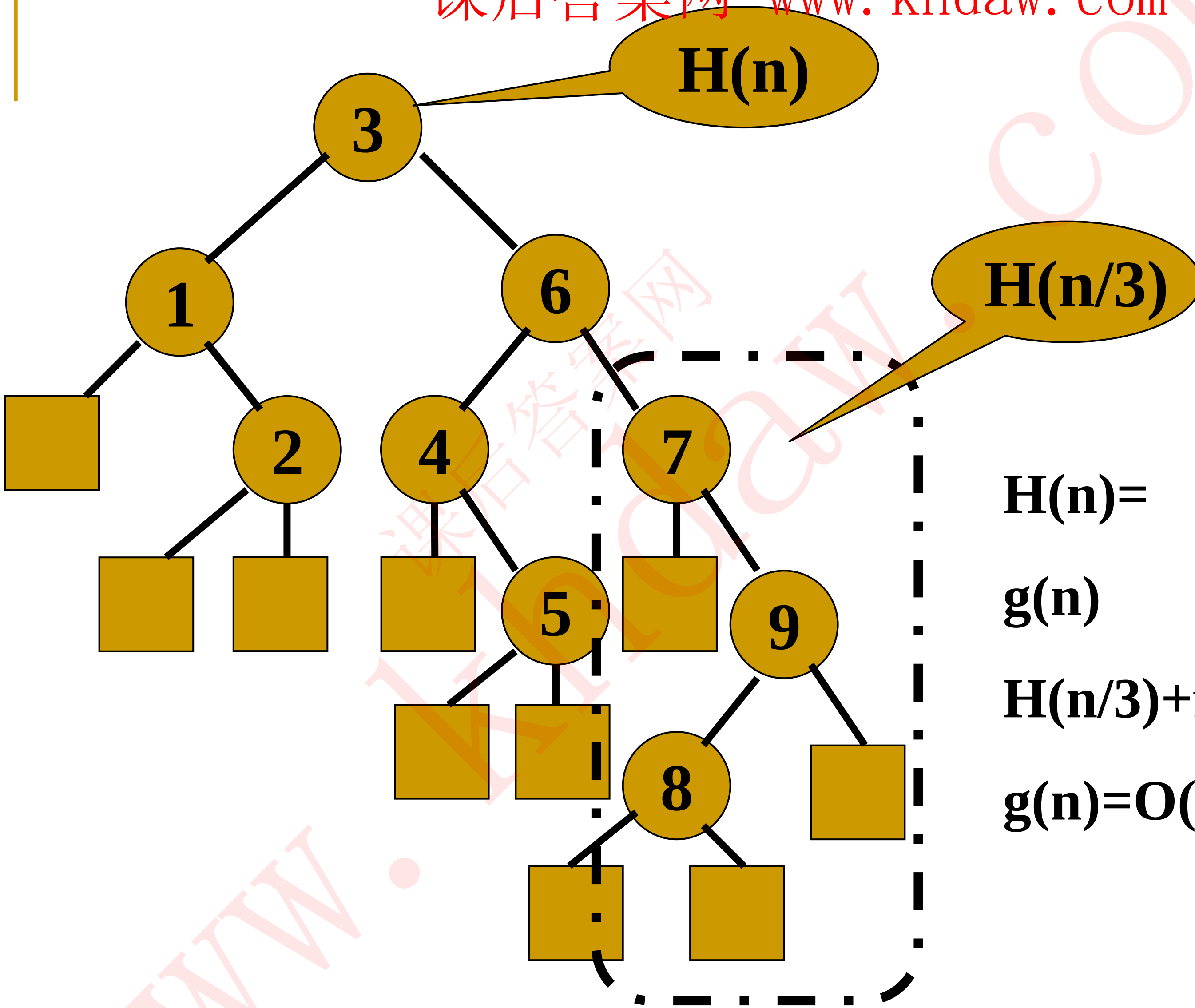
1 23456789

1 **2** 3456789

12345 **6** 789

123 **4** 56789

1234 **5** 6789



$H(n)=H(n/3)+2$

$H(3)=3$

$H(2)=2$

$H(1)=1$

$$H(n)=\begin{cases} g(n) & n\text{足够小} \\ H(n/3)+f(n) & \text{否则} \end{cases}$$

$g(n)=O(1)$ $f(n)=O(1)$

时间复杂度

- 成功:
  - O(1), $O(\log_3(n))$, $O(\log_3(n))$
  - 最好, 平均, 最坏
- 失败:
  - $O(\log_3(n))$, $O(\log_3(n))$, $O(\log_3(n))$
  - 最好, 平均, 最坏

P99-6

对于含有n个内部结点的二元树，证明

$$E=I+2n$$

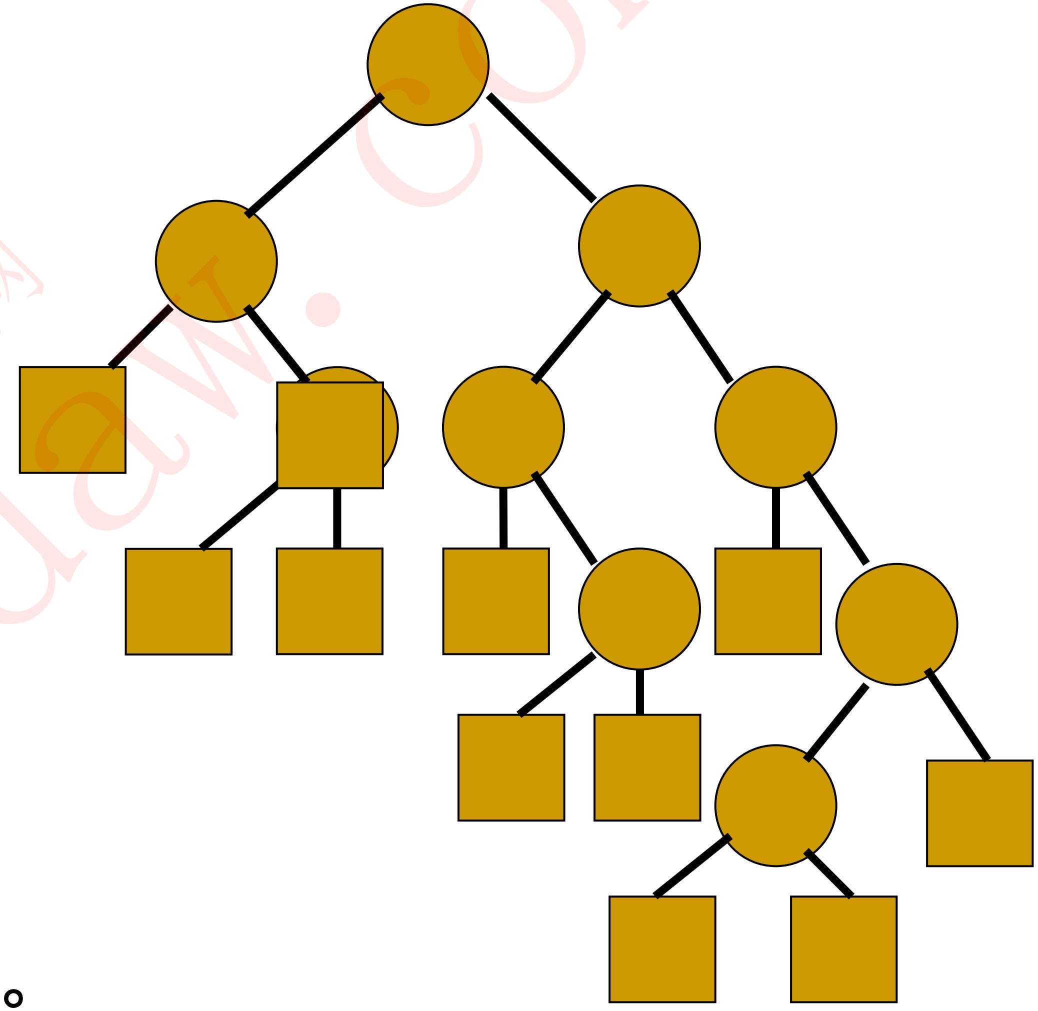
其中，E，I分别为外部和内部路径长度。

证明：数学归纳法

当n=1时，易知E=2，I=0，所以E=I+2n成立；

假设n≤k(k>0)时，E=I+2n成立；

则当**n=k+1**时，不妨认定某个内结点**x**，而且它为叶结点（根据二元扩展树的定义，一定存在这样的结点**x**，且设该结点的层数为**h**），将结点**x**及其左右子结点（外结点）从原树中摘除（**x**替换为外结点）。



此时新树内部结点为k个，则满足：

$$E_k=I_k+2k \qquad （1）$$

考察原树的外部路径长度和内部路径长度：

$$E_{k+1}= E_k-h+2(h+1) \qquad （2）$$

$$I_{k+1}=I_k+h \qquad （3）$$

综合（1）（2）（3）式：

$$\begin{aligned} E_{k+1} &= I_k+2k+h+2 \\ &= I_{k+1}- h+2k+h+2 \\ &= I_{k+1}+2(k+1) \end{aligned}$$

故命题成立。

P99-10

过程MERGESORT的最坏情况时间是O(nlogn)，它的最好情况时间是什么？能说归并分类的时间是Θ(nlogn)吗？

- 最好情况：
  - 对有序文件进行排序
- 分析
  - 归并的次数不会发生变化----log(n)次
  - 归并中比较的次数会发生变化（两个长n/2序列归并）
    - 最坏情况
      - 两个序列交错大小
      - 需要比较n-1次
    - 最好情况
      - 一个序列完全大于/小于另一个序列
      - 比较n/2次
  - 差异都是线性的，不改变复杂性的阶
- 最好情况也是nlog(n), 平均复杂度nlog(n)。

P99-11

- 写一个“由底向上”的归并分类算法，从而取消对栈空间的利用。
  - 见《数据结构》---第九章 **P238**

算法MPass（R，n，1ength．X）

MP1 [初始化]

i←1 .

MP2 [合并相邻的两个长度为length的子文件]

WHILE i ≤ n – 2*length + 1 DO

（Merge（R，i，i＋length–l，i＋2*length–1．X）.

i←i＋2*length ）.

MP3 [处理余留的长度小于2*length的子文件]

IF i+length–1 < n THEN

Merge（R，i，i+length–1，n. X）

ELSE

FOR j = i TO n DO X_j←R_j ▌

算法MSort(R，n) // 直接两路合并排序算法，X是辅助文件，其记录结构与R相同

MS1 [初始化]

length←1 .

MS2 [交替合并]

WHILE length < n DO

（MPass（R，n，length．X）.

length←2*length .

MPass（X，n，length．R）.

length←2*length）■

P99-23

- 通过手算证明（4.9）和(4.10)式确实能得到C_{11},C_{12},C_{21}和C_{22}的正确值。

$P=(A_{11}+A_{22})(B_{11}+B_{22})$

$T=(A_{11}+A_{12})B_{22}$

$Q=(A_{21}+A_{22})B_{11}$

$U=(A_{21}-A_{11})(B_{11}+B_{12})$

$R=A_{11}(B_{12}-B_{22})$

$V=(A_{12}-A_{22})(B_{21}+B_{22})$

$S=A_{22}(B_{21}-B_{11})$

$C_{11}=P+S-T+V$

$=(A_{11}+A_{22})(B_{11}+B_{22})+A_{22}(B_{21}-B_{11})-(A_{11}+A_{12})B_{22}+(A_{12}-A_{22})(B_{21}+B_{22})$

$=A_{11}B_{11}+A_{22}B_{11}+A_{11}B_{22}+A_{22}B_{22}+A_{22}B_{21}-A_{22}B_{11}-A_{11}B_{22}-A_{12}B_{22}+A_{12}B_{21}+A_{12}B_{22}-A_{22}B_{21}-A_{22}B_{22}$

$=A_{11}B_{11}+A_{12}B_{21}$

$P=(A_{11}+A_{22})(B_{11}+B_{22})$ $T=(A_{11}+A_{12})B_{22}$

$Q=(A_{21}+A_{22})B_{11}$ $U=(A_{21}-A_{11})(B_{11}+B_{12})$

$R=A_{11}(B_{12}-B_{22})$ $V=(A_{12}-A_{22})(B_{21}+B_{22})$

$S=A_{22}(B_{21}-B_{11})$

$C_{12}=R+T$

$= A_{11}B_{12}-A_{11}B_{22} +A_{11}B_{22}+A_{12}B_{22}$

$= A_{11}B_{12} +A_{12}B_{22}$

$C_{21}=Q+S$

$= A_{21}B_{11}+A_{22}B_{11} +A_{22}B_{21}-A_{22}B_{11}$

$= A_{21}B_{11} +A_{22}B_{21}$

$P=(A_{11}+A_{22})(B_{11}+B_{22})$ $\quad$ $T=(A_{11}+A_{12})B_{22}$

$Q=(A_{21}+A_{22})B_{11}$ $\quad$ $U=(A_{21}-A_{11})(B_{11}+B_{12})$

$R=A_{11}(B_{12}-B_{22})$ $\quad$ $V=(A_{12}-A_{22})(B_{21}+B_{22})$

$S=A_{22}(B_{21}-B_{11})$

$C_{22}=P+R-Q+U$

$=(A_{11}+A_{22})(B_{11}+B_{22})+A_{11}(B_{12}+B_{22})-(A_{21}+A_{22})B_{11}+(A_{21}-A_{11})(B_{11}+B_{12})$

$= A_{11}B_{11}+A_{22}B_{11}+A_{11}B_{22}+A_{22}B_{22}+A_{11}B_{12}-A_{11}B_{22}-A_{21}B_{11}-A_{22}B_{11}+A_{21}B_{11}+A_{21}B_{12}-A_{11}B_{11}-A_{11}B_{12}$

$=A_{22}B_{22}+A_{21}B_{12}$

计算机算法分析—习题课

2008年5月

第五章：2、3 、6、8 、9 、10、11 、12

P121-2.

- 求以下情况背包问题的最优解，n=7，m=15，（p1 ,…, p7）=(10,5,15,7,6,18,3)和(w1,…,w7)=(2,3,5,7,1,4,1)。
- 将以上数据情况的背包问题记为I。设FG(I)是物品按pi的非增次序输入时由GREEDY-KNAPSACK所生成的解，FO(I)是一个最优解。问FO(I)/ FG(I)是多少?
- 当物品按wi的非降次序输入时，重复②的讨论。

P121-2.

- ① 按照pi/mi的非增序可得(p5/w5，p1/w1, p6/w6, p3/w3, p7/w7, p2/w2, p4/w4)=(6,5,9/2,3,3,5/3,1)
  所以最优解为：（1,2/3,1,0,1,1,1），并且FO(I)=166/3
- ② 按照pi的非增次序输入时，得到(p6, p3, p1, p4, p5, p2, p7)= (18,15,10,7,6,5,3)，对应的(w6, w3, w1, w4, w5, w2, w7)= (4,5,2,7,1,3,1)，则FG(I)的解为（1,0,1,4/7,0,1,0），并且FG(I)=47，所以FO(I)/ FG(I)=166/141.

P121-2.

- ③ 按照wi的非降次序输入时，得到(w5, w7, w1, w2, w6, w3, w4)=(1,1,2,3,4,5,7)，相应的（p5, p7, p1, p2, p6, p3, p4）=(6,3,10,5,18,15,7)，则FW(I)的解为（1,1,4/5,0,1,1,1），并且FW(I)=54，所以FO(I)/ FW(I)=83/81.

P122-3

- 3. （0/1背包问题）如果将5.3节讨论的背包问题修改成
- 极大化 $\sum_{1}^{n} p_i x_i$
- 约束条件 $\sum_{1}^{n} w_i x_i \le M$
- x_i=0或1 1≤i≤n
- 这种背包问题称为0/1背包问题。它要求物品或者整件装入背包或者整件不装入。求解此问题的一种贪心策略是：按p_i/w_i的非增次序考虑这些物品，只要正被考虑的物品能装的进就将其装入背包。证明这种策略不一定能得到最优解。

P122-3

- 证明：
  - 当按照p_i/w_i的非增次序考虑物品存放背包时，如果所装入的物品恰能装满背包时，显然为最优解，否则未必是最优解.

P122-3

- 可举例如下：

  设n=3，M=6，

  （p_1,p_2,p_3）=(3,4,8)，（w_1,w_2,w_3）=(1,2,5)

按照p_i/w_i的非增序得到

（$p_1/w_1,p_2/w_2,p_3/w_3$）=(3,2,1.6)，

则其解为（1,1,0），而事实上最优解是(1,0,1)。

问题得证。

P122- 6.

- 假定要将长为$l_1,l_2,\ldots,l_n$的n个程序存入一盘磁带，程序i被检索的频率是f_i。如果程序按$i_1,i_2,\ldots,i_n$的次序存放，则期望检索时间（ERT）是:

$$\left[\sum_{j}\left(f_{i_j}\sum_{k=1}^{j} l_{i_k}\right)\right] / \sum f_i$$

- ① 证明按l_i的非降次序存放程序不一定得到最小的ERT。
- ② 证明按f_i的非增次序存放程序不一定得到最小的ERT。
- ③ 证明按f_i/l_i的非增次序来存放程序时ERT取最小值。

① l:(4,1,2)　f:(0.8,0.1,0.1)

按 **li**的 非降序存放程序

ERT=0.1*1+0.1*3+0.8*7=6

而最优解为**0.8*4+0.1*5+0.1*7=4.4**

②l:(16,1,2)　f:(0.8,0.1,0.1)

按 **fi**的 非增序存放程序

ERT=0.8*16+0.1*17+0.1*19=16.4

而最优解为**0.1*1+0.8*17+0.1*19=15.6**

证明结论③是正确的,证明如下:

假设$l_{i1},l_{i2},\ldots,l_{in}$按照$f_i/l_i$的非增次序存放，即$f_{i1}/l_{i1} \geq f_{i2}/l_{i2} \geq \ldots \geq f_{in}/l_{in}$，则得到$ERT=[f_{i1}l_{i1}+f_{i2}(l_{i1}+l_{i2})+\ldots+f_{ik}(l_{i1}+l_{i2}+\ldots+l_{ik})+\ldots+f_{in}(l_{i1}+l_{i2}+\ldots+l_{in})]/\sum f_i$

假设该问题的一个最优解是按照j1,j2,…,jn的顺序存放，并且其期望检索式是ERT'，我们只需证明ERT≤ERT'，即可证明按照fi/li的非增次序存放得到的是最优解。易知

- $ERT' = [f_{j1}l_{j1} + f_{j2}(l_{j1}+l_{j2}) + \ldots + f_{jk}(l_{j1}+l_{j2}+\ldots+l_{jk}) + \ldots + f_{jn}(l_{j1}+l_{j2}+\ldots+l_{jn})] / \sum f_i$，从前向后考察最优解中的程序，不妨设程序$j_k$是第一个与其相邻的程序$j_{k+1}$存在关系 $f_{j_k}/l_{j_k} < f_{j_{k+1}}/l_{j_{k+1}}$，则交换程序$j_k$和程序$j_{k+1}$，得到的期望检索时间记为ERT''，容易证明ERT'' ≤ERT'，显然ERT''也是最优解，将原来的最优解中所有这样类似于反序对的程序互换位置，得到的解不比原来的最优解差，所以最终变换后得到的解也是最优解，而最终的解恰是程序按fi/li的非增次序来存放得到的顺序。命题得证。

P123-8

- ① 当n=7，（p_1 ,…., p_7）=(3,5,20,18,1,6,30)和(d_1,…,d_7)=(1,3,4,3,2,1,2)时，算法5.5所生成的解是什么？
- ② 证明即使作业有不同的处理时间定理5.3亦真。这里，假定作业i的效益p_i>0，要用的处理时间t_i>0，限期$d_i \geq t_i$.

P123-8

- 解：①根据p_i的非增排序得到（p_7, p_3, p_4, p_6, p_2, p_1, p_5）=(30,20,18,6,5,3,1)，对应的期限为(2,4,3,1,3,1,2)，按照算法3.5生成的解为：

1. J(1)=7(2),
2. J(1)=7(2), J(2)=3(4);
3. J(1)=7(2), J(2)=4(3),J(3)=3(4);
4. J(1)=6(1), J(2)=7(2),J(3)=4(3),J(4)=3(4);

P123-8

- ② 证明即使作业有不同的处理时间定理5.3亦真。这里，假定作业i的效益p_i>0，要用的处理时间t_i>0，限期$d_i \geq t_i$.(P106)
- 定理5.3:设J是K个作业的集合, $\sigma = i_1 i_2 \ldots i_k$是J中作业的一种排序,它使得$d_{i1} \leqslant d_{i2} \leqslant \ldots \leqslant d_{ik}$ .J是一个可行解,当且仅当J中的作业可以按照σ的次序又不违反任何一个期限的情况下来处理.

P123-8

证明：显然即使 $t_i>0$（$d_i \geqslant t_i$），如果 J 中的作业可以按照σ的次序而又不违反任何一个期限来处理，即对σ次序中的任一个作业 k，应满足 $d_k \geqslant \sum_{j=1}^{k} t_j$，则 J 就是一个可行解。下面证明如果 J 是可行解，$\sigma = i_1 i_2 \cdots i_k$ 使得 J 中的作业可以按照 $d_{i1} \leqslant d_{i2} \leqslant \cdots \leqslant d_{in}$ 序列排列而又不违反任何一个期限。

J 是可行解，则必存在σ'$=r_1r_2\ldots r_n$，使得对任意的 r_k，都有 $d_k \geq \sum_{j=1}^{k} t_j$，我们设σ是按照 $d_{i1} \leq d_{i2} \leq \ldots \leq d_{in}$ 排列的作业序列。假设σ'≠σ，那么令 a 是使 $r_a \neq i_a$ 的最小下标，设 $r_b=i_a$，显然 b>a，在σ'中将 r_a 与 r_b 相交换，因为 $d_{rb} \leq d_{ra}$，显然 r_a 和 r_b 可以按期完成作业，我们还要证明 r_a 和 r_b 之间的作业也能按期完成。因为 $d_{rb} \leq d_{ra}$，而显然二者之间的所有作业 r_t，都有 $d_{rb} \leq d_{rt}$，又由于σ'是可行解，所以 $\sum_{k=1}^{b} t_k \leq d_{r_b} \leq d_{r_t}$，所以作业 r_a 和 r_b 交换后，仍满足 $\sum_{k=1}^{t} t_k \leq d_{r_t}$，即所有作业可依新产生的排列σ''$==s_1s_2\ldots s_n$ 的次序处理而不违反任何一个期限，连续使用这种方法，σ'就可转换成σ且不违反任何一个期限，定理得证。

P123-9

- ① 对于5.3节的作业排序问题证明：当且仅当子集合J中的作业可以按下述规则处理时它表示一个可行解；如果J中的作业I还没分配处理时间，则将它分配在时间片[a-1，a]处理，其中a是使得$1 \leqslant r \leqslant d_i$的最大整数r，且时间片[a-1，a]是空的。
- ② 仿照例5.4的格式，在习题5.8的①所提供的数据集上执行算法5.5。

P123-9

- 易证如果J中的作业能按上述规则处理，显然J是可行解；
- 如果J是可行解，根据定理5.3可知，J中的作业根据时间期限的非降次序排列，得到$i_1 i_2 \ldots i_k \ldots i_n$，并且按照这个顺序，可以处理J中所有作业，而对这一序列中的任意作业i_k，如果它的时间期限是d_k，且时间片$[d_k-1,d_k]$是空的，则分配之；若时间片$[d_k-1，d_k]$非空，则向前找最大的非空$[r-1,r]$时间片，$1 \leqslant r \leqslant d_k$因为J是可行解，所以一定可以找到如此时间片。故命题得证。

- n=7

  (p1,…, p7)=(3,5,20,18,1,6,30)

  (d1,…,d7)=(1,3,4,3,2,1,2)

- (p_7, p_3, p_4, p_6, p_2, p_1, p_5)

  =(30,20,18,6,5,3,1)，

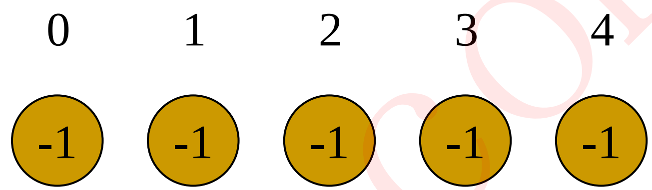
  对应的期限为(2,4,3,1,3,1,2)

  b=min{ n,max{d(i)} }

  =min{7,4 }

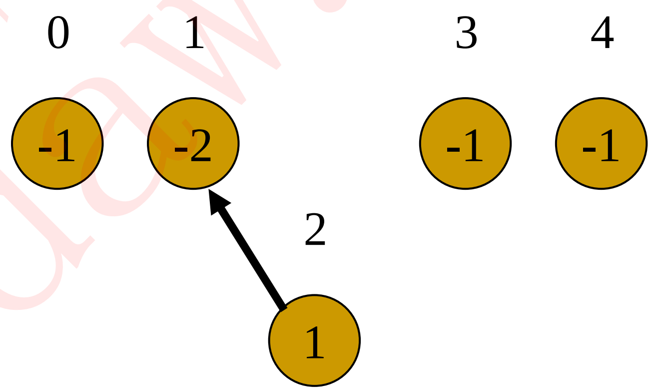
  =4

F(0)	F(1)	F(2)	F(3)	F(4)
0	1	2	3	4



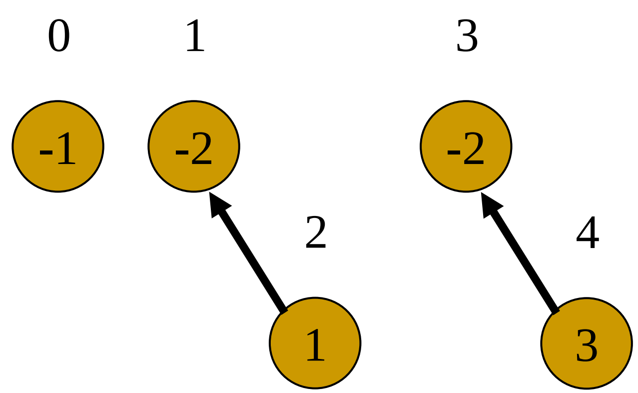
空

F(0)	F(1)	F(2)	F(3)	F(4)
0	1	1	3	4



{7}

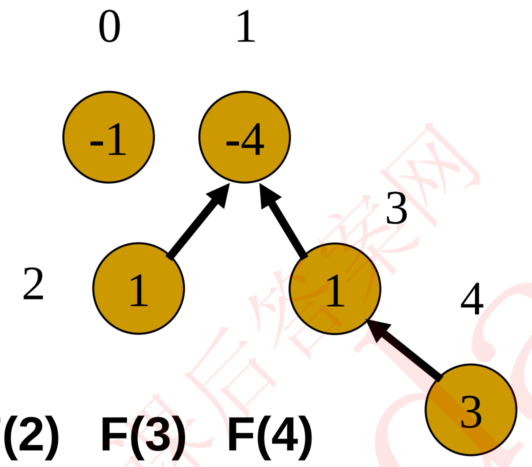
F(0)	F(1)	F(2)	F(3)	F(4)
0	1	1	3	3



{7,3}

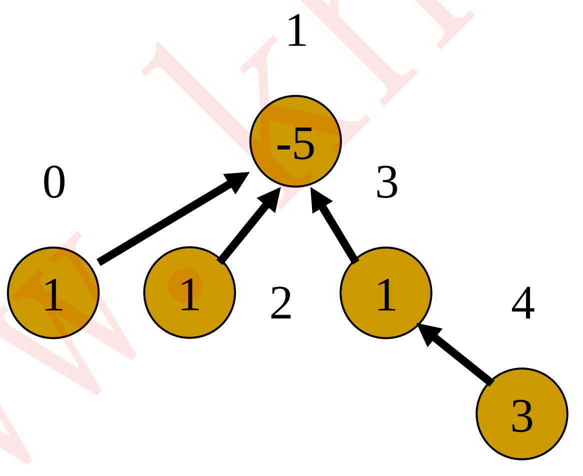
(p7, p3, p4, p6, p2, p1, p5)=(30,20,18,6,5,3,1)，对应的期限为(2,4,3,1,3,1,2)

F(0)	F(1)	F(2)	F(3)	F(4)
0	1	1	1	3



{7,3,4}

F(0)	F(1)	F(2)	F(3)	F(4)
0	0	1	1	3



{7,3,4,6}

(p7, p3, p4, p6, p2, p1, p5)=(30,20,18,6,5,3,1)，对应的期限为(2,4,3,1,3,1,2)

P123-11

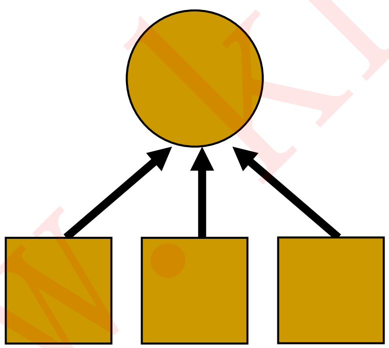
- ① 证明如果一棵树的所有内部节点的度都为k，则外部节点数n满足n mod (k-1)=1.
- ② 证明对于满足 n mod (k-1)=1的正整数n，存在一棵具有n个外部节点的k元树T(在一棵k元树中，每个节点的度至多为k)。进而证明T中所有内部节点的度为k.

P123-11

- 证明：① 设某棵树内部节点的个数是**i**，外部结点的个数是**n**，边的条数是**e**，则有
- **e=i+n-1**
- **ik=e**
- ⇒ **ik=i+n-1**
- ⇒ **(k-1)i=n-1**
- ⇒ **n mod (k-1)=1**

P123-11

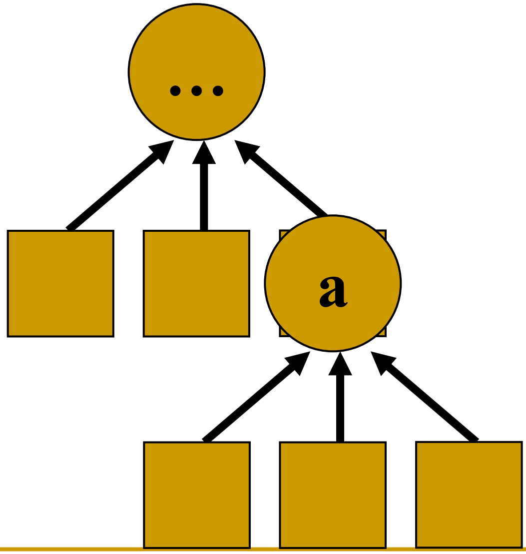
- ② 利用数学归纳法(m表示外部结点数目)。
- 当m =k时，存在外部结点数目为k的k元树T，并且T中内部结点的度为k；



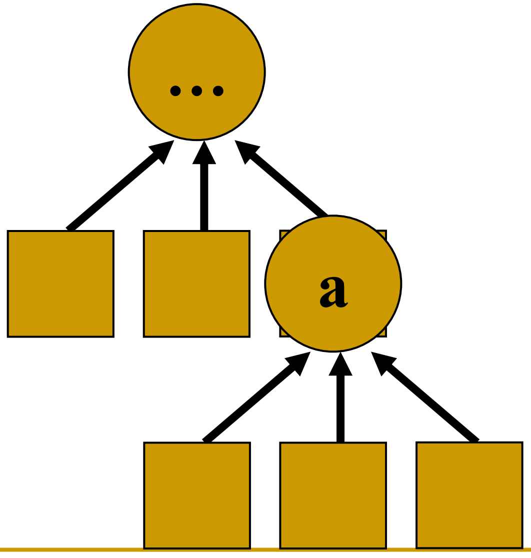
m=3

3mod(3-1)=1

- 假设当 **m <n**，且满足**m mod (k-1)=1**时，存在一棵具有**m**个外部结点的**k**元树**T**，且所有内部结点的度为**k**;
- 我们将外部结点数为**m**的符合上述性质的树**T**中某个外部结点用内部结点 **a**替代，且结点**a**生出**k**个外部结点.



- 易知新生成的树T’中外部结点的数目为n= m -1+k= m +(k-1)，因为 m mod (k-1)=1显然n为满足n mod (k-1)=1，且比m大的最小整数，而树T’每个内结点的度为k，所以n= m +(k-1)时，存在符合上述性质的树。故命题得证。



P123-12

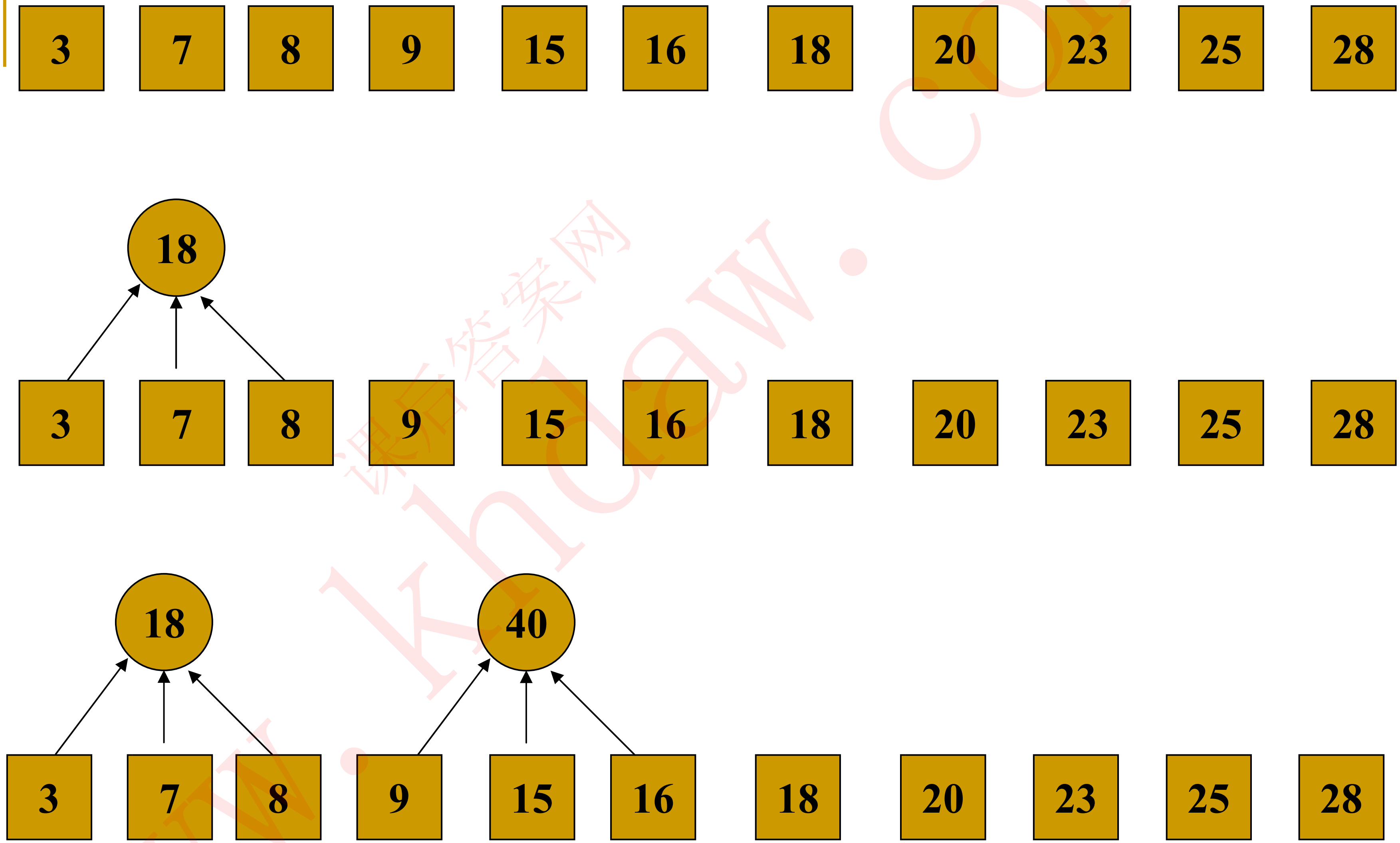
- ① 证明如果**n mod (k-1)=1**，则在定理**5.4**后面所描述的贪心规则对于所有的（$q_1,q_2,\ldots,q_n$）生成一棵最优的**k**元归并树。**(P111)**
- ② 当（$q_1,q_2,\ldots,q_{11}$）=（**3,7,8,9,15,16,18,20,23,25,28**）时，画出使用这一规则所得到的最优**3**元归并树。

P123-12

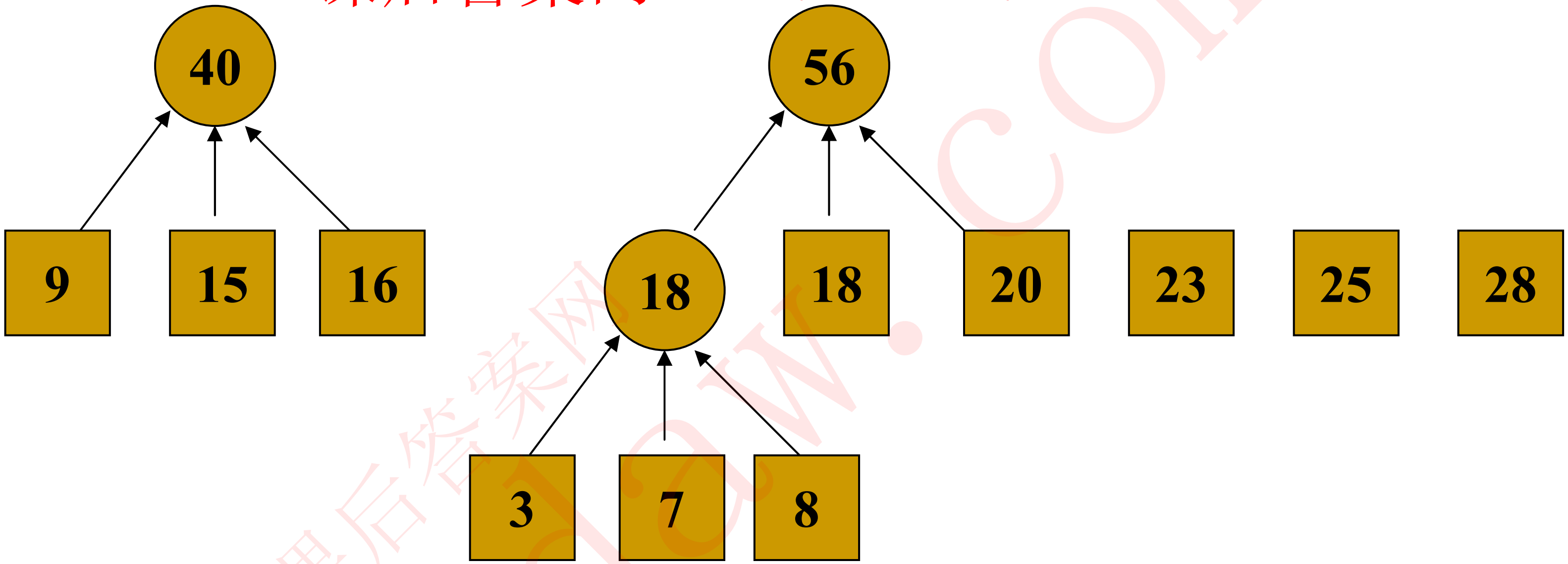
- 通过数学归纳法证明：
- 对于**n=1**，返回一棵没有内部结点的树且这棵树显然是最优的。
- 假定该算法对于（$q_1,q_2,\ldots,q_m$），其中$m=(k-1)s+1$ ($s\geqslant 0$)，都生成一棵最优树，
- 则只需证明对于($q_1,q_2,\ldots,q_n$)，其中$n=(k-1)(s+1)+1$，也能生成最优树即可。

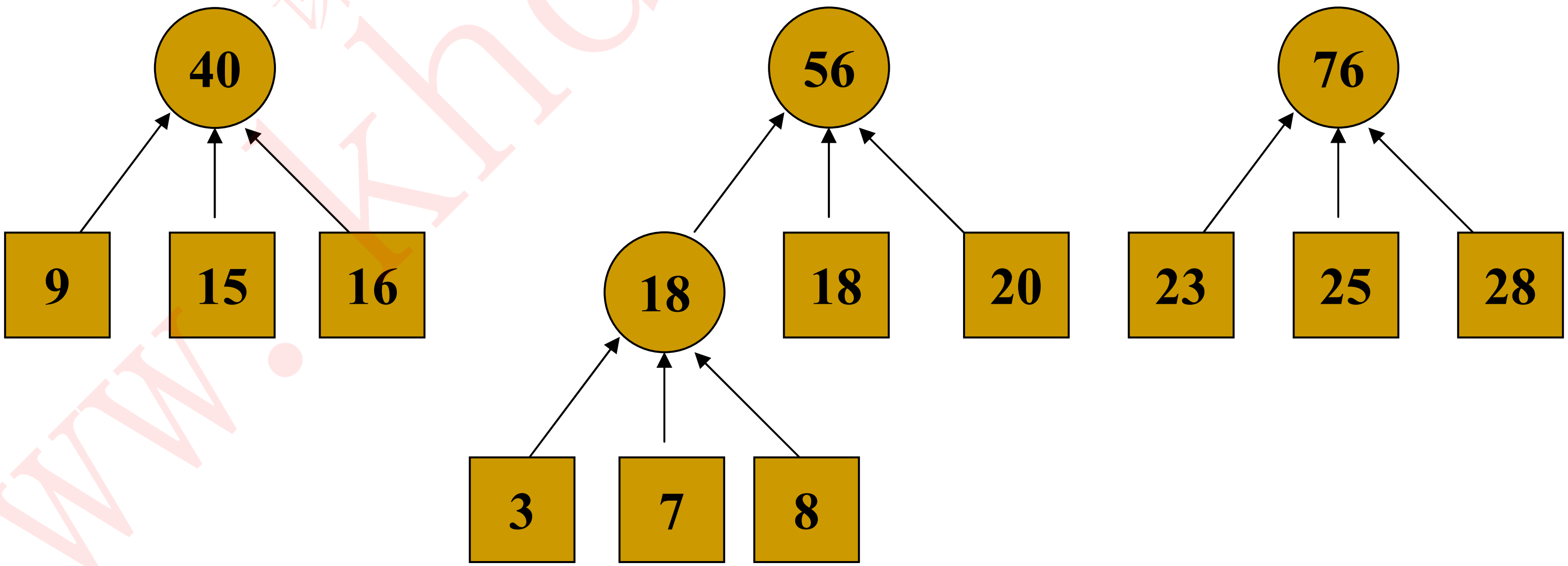
- 不失一般性，假定$q_1 \leqslant q_2 \leqslant \ldots \leqslant q_n$，且$q_1, q_2, \ldots, q_k$是算法所找到的k棵树的WEIGHT信息段的值。于是$q_1, q_2, \ldots, q_k$可生成子树T，设T'是一棵对于（$q_1, q_2, \ldots, q_n$）的最优k元归并树。设P是距离根最远的一个内部结点。如果P的k个儿子不是$q_1, q_2, \ldots, q_k$，则可以用$q_1, q_2, \ldots, q_k$和P现在的儿子进行交换，这样不增加T'的带权外部路径长度。

- 因此T也是一棵最优归并树中的子树。于是在T'中如果用其权为$q_1+q_2+\ldots+q_k$的一个外部结点来代换T，则所生成的树T''是关于$(T,q_{k+1},\cdots,q_n)$的一棵最优归并树。由归纳假设，在使用其权为$q_1+q_2+\ldots+q_k$的那个外部结点代换了T以后，过程TREE转化成去求取一棵关于$(T,q_{k+1},\cdots,q_n)$的最优归并树。因此TREE生成一棵关于$(q_1,q_2,\cdots,q_n)$的最优归并树。

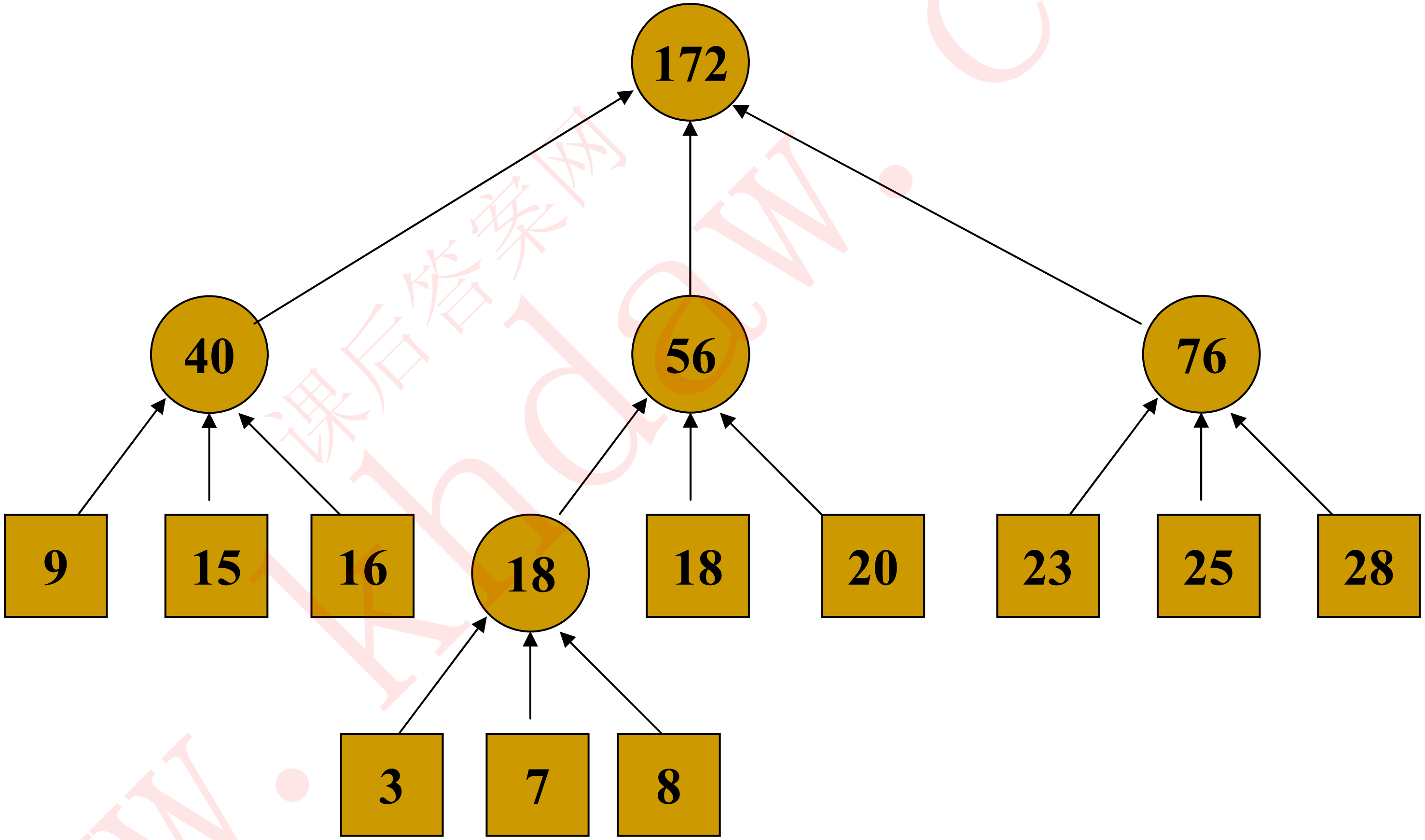


$(q_1, q_2, \ldots, q_{11}) = (3,7,8,9,15,16,18,20,23,25,28)$





$(q_1, q_2, \ldots, q_{11}) = (3,7,8,9,15,16,18,20,23,25,28)$



$(q_1, q_2, \ldots, q_{11}) = (3,7,8,9,15,16,18,20,23,25,28)$

计算机算法分析—习题课

2008年5月

第六章1 2 3 4 5 6 8 13 17

动态规划

1. 多阶段过程
2. 满足**最优性原理**
3. 建立递推关系式

P151-1

- ①递推关系式（**6.8**）对右图成立吗？为什么？
- ②递推关系式（**6.8**）为什么对于含有负长度环的图不能成立？
  - 解：
  - ①成立，不包含负长度环
  - ②可以使节点间的长度任意小。

P151-2

- 修改过程**ALL_PATHS**，使其输出每对结点（**i,j**）间的最短路径，这个新算法的时间和空间复杂度是多少？
- 回忆算法：
  P127 算法 6.1
  P131 算法 6.3

P127 算法6.1

- D(i,j)/D(j) :从节点j到汇点t的最优路径中下一个节点，即最优路径中j的后继节点。
- 算法6.1在计算COST(j)的同时也计算了D(j) 3－7行
- 计算出D(j) 之后，即可计算最短路径。9－11行

P131 算法6.3

- 对矩阵进行初始化，每个元素赋值为边的长度（如果没边则赋值成MAX）
  1—5行
- 迭代计算最短路径长度
  6—12行
- 仿照6.1，在每次计算最短路径的时候计算出D(j) 再通过D(j) 就可以表示出最短路径

```
Procedure ShortestPath(COST, n, A, Max)
    integer  i , j, k
    real COST(n, n), A(n, n), Path(n, n), Max
    for i←1 to n do //初始化最优路径矩阵
        for j←1 to n do
            A(i ,j)←COST(i ,j)
            if i≠j and A(i, j)≠Max  then
                Path(i, j )←j
            else
                Path(i, j)←0
            endif
        repeat
    repeat
```

Path(i,j)是从i到j的最短路径上，结点i的后续结点编号。

```
for k←1 to n do //迭代计算
    for i←1 to n do
        for j←1 to n do
            if A(i,j)>A(i,k)+A(k,j) then
                A(i,j)←A(i,k)+A(k,j)
                Path(i,j)←Path(i,k)
            endif
        repeat
    repeat
repeat
```

```
for i←1 to n do//输出最优路径
  for j←1 to n do
        print("the path of i to j is " i )
        k←path(i, j)
        while k≠0 do
              print(k)
              k←path(k, j)
        repeat
  repeat
repeat
end ShortestPath
```

分析

- 时间复杂度
  第一个循环：$O(n^2)$
  第一个循环： $O(n^3)$
  第一个循环： $O(n^2)$
- 空间复杂度
  Cost(n,n) A(n,n) Path(n,n)
  $O(n^2)$

P151-3

- 对于标识符集（a_1,a_2,a_3,a_4）=（end, goto, print, stop），已知成功检索概率为P(1)=1/20, P(2)=1/5, P(3)=1/10, P(4)=1/20，不成功检索概率为Q(0)=1/5, Q(1)=1/10, Q(2)=1/5, Q(3)=1/20, Q(4)=1/20，用算法OBST对其计算W(i ,j)，R(i, j)和C(i, j)（0≤i, j≤4）。

P136 算法6.5

- P（i）
  $P(1)=1/20, P(2)=1/5, P(3)=1/10, P(4)=1/20$
- Q（i）
  $Q(0)=1/5, Q(1)=1/10, Q(2)=1/5, Q(3)=1/20, Q(4)=1/20$
- P（i）
  $P(1)=1, P(2)=4, P(3)=2, P(4)=1$
- Q（i）
  $Q(0)=4, Q(1)=2, Q(2)=4, Q(3)=1, Q(4)=1$

W					R					C				
4	4+2+1 =7				0	1				0	7			
	2	2+4+4 =10				0	2				0	10		
		4	4+1+2 =7				0	3				0	7	
			1	1+1+1 =3				0	4				0	3
				1					0					0

W					R					C				
4	4+2+1 =7	7+4+4 =15			0	1	2			0	7	22		
	2	2+4+4 =10	10+2+1=13			0	2	2			0	10	20	
		4	4+1+2 =7	7+1+1 =9			0	3	3			0	7	12
			1	1+1+1 =3				0	4				0	3
				1					0					0

W					R					C				
4	4+2+1 =7	7+4+4 =15	15+2+ 1=**18**	18+1+ 1=**20**	0	1	2	2	2	0	7	22	32	39
	2	2+4+4 =10	10+2+ 1=13	13+1+ 1=**15**		0	2	2	2		0	10	20	27
		4	4+1+2 =7	7+1+1 =9			0	3	3			0	7	12
			1	1+1+1 =3				0	4				0	3
				1					0					0

P151-4

- ①证明算法OBST的计算时间是$O(n^2)$。
- ②在已知根R(i, j)，$0 \leqslant i < j \leqslant 4$的情况下写一个构造最优二分检索树T的算法。证明这样的树能在$O(n)$时间内构造出来。

① 将 C 中元素的加法看做基本运算，则算法 OBST 的时间复杂性为：

$$\sum_{m=2}^{n}\sum_{i=0}^{n-m}(R(i+1,j)-R(i,j-1)+1)$$

$$=\sum_{m=2}^{n}\sum_{i=0}^{n-m}(R(i+1,i+m)-R(i,i+m-1)+1)$$

$$=\sum_{m=2}^{n}(R(n-m+1,n)-R(0,m-1)+n-m+1)$$

$O(n^2)$

② Procedure BuildTree(m, n, R, Root)

integer R(n,n), k

TreeNode Root, LR, RR

k←R(m,n)

if k≠0 then data(Root)←k,

BuileTree(m, k-1, R, LR), BuileTree(k, n, R, RR)

left(Root)←LR, right(Root)←RR

else data(Root)←m, left(Root)←null, right(Root)←null,
endif

end BuildTree

时间复杂性分析：T(n)=c+T(k)+T(n-k-1)，此递推式保证算法的时间复杂性为O(n)，也可从递归的角度出发，递归的次数正是结点的个数，而每次递归时间复杂性为常数，所以算法的时间复杂度也为O(n)。

P151-5

- 由于我们通常只知道成功检索和不成功检索概率的近似值，因此，若能在较短的时间内找出几乎是最优的二分检索树，也是一件很有意义的工作。所谓几乎是最优的二分检索树，就是对于给定的P和Q，该树的成本（由（6.9）式计算）几乎最小。已经证明，由以下方法获得这种检索树的算法可以有O(nlogn)的时间复杂度，选取这样的k为根，它使|W(0, k-1)- W(k, n) |尽可能地小。重复以上步骤去找这根的左、右子树。

① 对于题6.3的数据，用上述方法找出一棵这样的二分检索树。它的成本是什么？

② 用SPAKS写一个实现上述方法的算法，你的算法的计算时间为O(nlogn)吗？

解：①矩阵W如下所示，相应的k从1到4得式如下：

|W(0,0)-W(1,4)|=11,|W(0,1)-W(2,4)|=2,|W(0,2)-W(3,4)|=12,|W(0,3)-W(4,4)|=17

所以该树的根是T(0,4)=2，依次计算得到T(0,1)=1，T(2,4)=3，T(3,4)=4

总体成本是4+2*（2+1）+2*（4+2+4）+3*（1+1+1）=39

```
②Procedure CreateTree(m, n, root, w)
  integer r, k, root(n, n), w(n, n)
  real ww, min←∞
  if m=n then root(m, n)←0
    else if m=n-1 then root(m,n)←n
    else
       for k←m+1 to n do
          ww←|w(m, k-1)-w(k, n)|
          if ww< min then min←ww, r←k  endif
       repeat
       root(m, n)←r
       Call CreateTree(m, r-1)
       Call CreateTree(r, n)
   endif
End BuildTree
```

时间复杂性最好和平均的情况应该是O(nlogn)，但最坏的情况是$O(n^2)$

P151-6

设$(w_1,w_2,w_3,w_4)=(10,15,6,9)$，$(p_1,p_2,p_3,p_4)=(2,5,8,1)$。生成每个$f_i$阶跃点的序偶集合$S_i$，$0\leqslant i\leqslant 4$。

$S^0=\{(0,0)\}$ $S_1^1=\{(2,10)\}$

$S^1=\{(0,0), (2,10)\}$ $S_1^2=\{(5,15), (7,25)\}$

$S^2=\{(0,0), (2,10), (5,15), (7,25)\}$ $S_1^3=\{(8,6), (10,16), (13,21), (15,31)\}$

$S^3=\{(0,0), (8,6), (10,16), (13,21), (15,31)\}$ $S_1^4=\{(1,9), (9,15), (11,25), (14,30), (16,40)\}$

$S^4=\{(0,0), (8,6), (9,15), (10,16), (13,21), (14,30), (15,31), (16,40)\}$

P151-8

- 给出一个使得DKNAP(算法6.7)出现最坏情况的例子，它使得$|S^i|=2^i$, $0\leqslant i<n$。还要求对n的任意取值都适用。

P151-8

取 (P1,P2,…,Pi,…)

=(W1,W2,…,Wi,…)

$=(2^0,2^1,\ldots,2^{i-1},\ldots)$

P和W取值相同，使支配原则成立，也就是说不会因为支配原则而删除元素；只要说明不会出现相同元素被删除一个的情形，即可知是最坏的情况。可用归纳法证明此结论。

P152-13

- 假定两个仓库W_1和W_2都存有同一种货物，其库存量分别为r_1和r_2。想要将其全部发往n个目的地$D_1,D_2,\ldots,D_n$. 设发往D_j的货物为d_j，因此$r_1+r_2=\sum d_j$. 如果由仓库W_i发送量为x_{ij}的货物到目的地D_j的花费为C_{ij}，那么仓库问题就是求各个仓库应给每个目的地发多少货才使总的花费最小。即要求出这些非负整数xij，$1\le i\le 2, 1\le j\le n$，它使得$x_{1j}+x_{2j}=d_j$, $1\le j\le n$，并且使$\sum_{ij}c_{ij}(x_{ij})$取最小值。假设当W_1有x的库存且按最优方式全部发往目的地$D_1,D_2,\ldots,D_i$时所需的花费为$g_i(x)$(W_2的库存为$\sum d_j-x$)。于是此仓库问题的最优总花费是$g_n(r_1)$.

P128-13

①求$g_i(x)$的递推关系式

② 写一个算法求解这个递推关系式并要能得到x_{ij}的决策值得最优序列，$1 \leqslant i \leqslant 2, 1 \leqslant j \leqslant n$.

$$g_i(x) = \min_{0 < x_{1i} \le \min\{x, d_i\}} \{c_{1i}(x_{1i}) + c_{2i}(d_i - x_{1i}) + g_{i-1}(x - d_i)\}$$

$$x > d_i$$

$$g_i(x) = c_{1i}(x_{1i}) + c_{2i}(d_i - x_{1i}) \quad x \le d_i$$

P152-17

- 最优性原理并不总是对可以将其解看成是一系列决策结果的所有问题成立。找两个最优性原理不成立的例子，并说明对这两个问题最优性原理为什么不成立。

- 多段图问题：路径和改为路径乘积并允许出现负数

计算机算法分析—习题课

2006年12月

第八章：1、3、8、9

P215-1

修改算法8.1和8.2，使它们只求出问题的一个解而不是问题的全部解

算法8.1

```
procedure BACKTRACK(n)
integer k,n; local X(1: n)
k ← 1
while k>0 do
  if还剩有没检验过的X(k)使得
X(k) ∈ T ( X(1), … ,X(k-1)) and Bk ( X(1), … ,X(k)) = true
    then if ( X(1), … ,X(k) ) 是一条抵达一答案节点的路径
        then print ( X(1), … ,X(k))
                return
endif
        k ←k ＋ 1
else k ←k － 1
endif
repeat
end BACKTRACK
```

算法8.2

```
procedure RBACKTRACK(k)
 global n, X(1:n)
 for 满足下式的每个X(k)
    X(k) ∈ T ( X(1), … ,X(k-1)) and Bk ( X(1), … ,X(k)) = true
  do
   if  ( X(1), … ,X(k))是一条抵达一答案结点的路径
   then print ( X(1), … ,X(k))
         return
  endif
  call RBACKTRACK(k+1)
 repeat
end RBACKTRACK
```

P215-3

重新定义过程**PLACE(k)**，使它的返回值或者是第**k**个皇后可以放置于其上的合法列号，或者是一个非法值，这样可以提高一些**NQUEENS**的效率，按以上策略重写这两个过程。

算法8.4

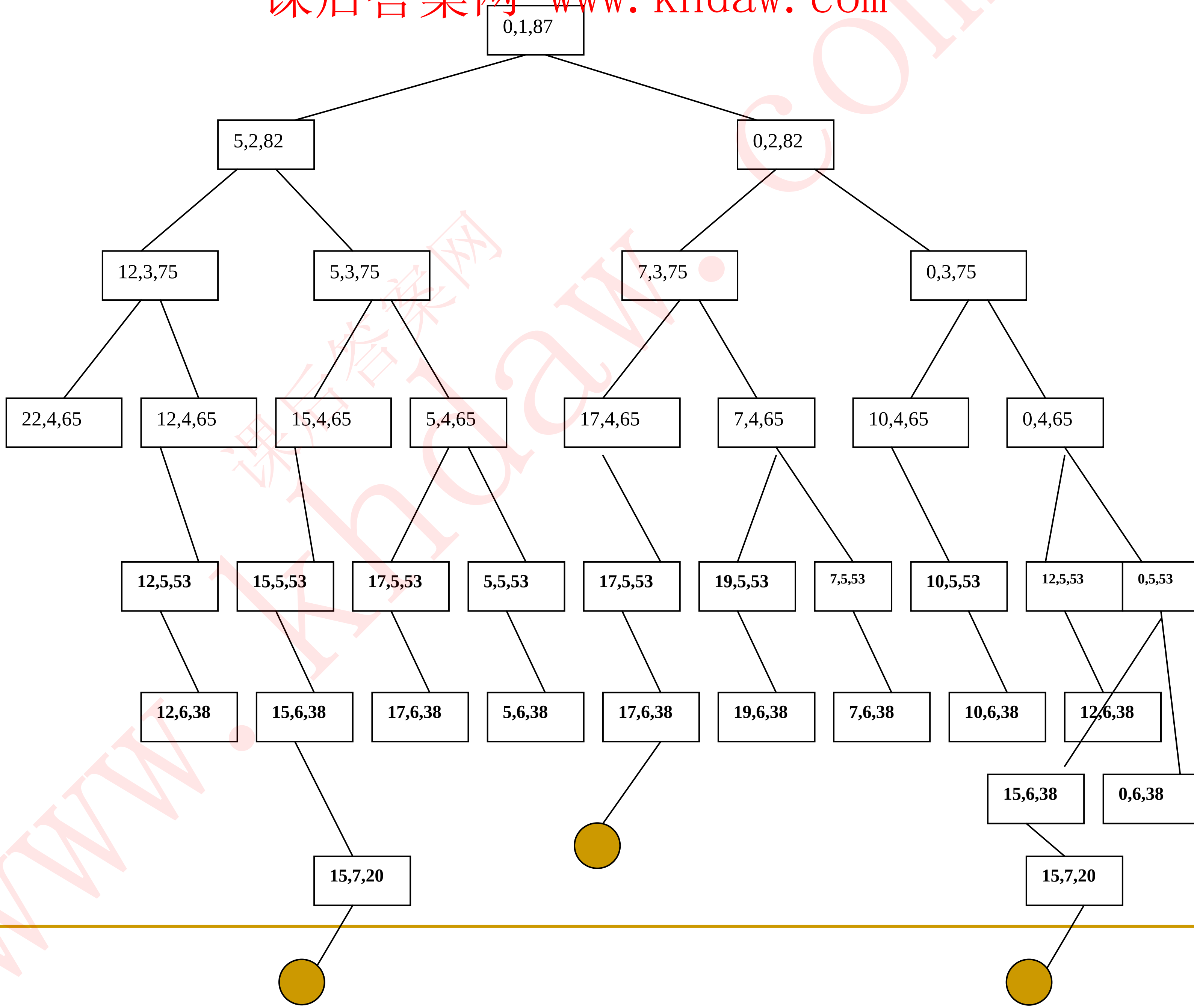
```
procedure PLACE(k)
 global  X(1:K) ; integer i, k
j ← -1
while  X(k)≤n  do
i ← 1
while i<k do
      if  X(i) = X(k) or ABS( X(i)－X(k)) = ABS(i－k)
      then  exit
      endif
     i ←i + 1
repeat
  if i=k then j←X(k)  exit endif
  X(k) ←X(k)+1
 repeat
return( j )
end PLACE
```

算法8.5

```
procedure NQUEENS(k)
 integer  k，  n，  X(1:n)
 X(1) ← 0; k←1
while k>0  do
   X(k) ← X(k)+1
   if  PLACE(k) ≠ -1     //找到一个位置
       then if k＝n
               then print（X）
               else k ← k＋1;  X(k) ← 0
               endif
       else  k ← k－1
  endif
repeat
end NQUEENS
```

P215-8

- 设W=(5,7,10,12,15,18,20) 和M=35， 使用过程SUMOFSUB找出W中使得和数等于M的全部子集并画出所生成的部分状态空间树。


0,1,87
5,2,82
0,2,82
12,3,75
5,3,75
7,3,75
0,3,75
22,4,65
12,4,65
15,4,65
5,4,65
17,4,65
7,4,65
10,4,65
0,4,65
12,5,53
15,5,53
17,5,53
5,5,53
17,5,53
19,5,53
7,5,53
10,5,53
12,5,53
0,5,53
12,6,38
15,6,38
17,6,38
5,6,38
17,6,38
19,6,38
7,6,38
10,6,38
12,6,38
15,6,38
0,6,38
15,7,20
15,7,20

P215-9

用以下数据运行过程SUMOFSUB，M=35和

①W=(5,7,10,12,15,18,20)

②W=(20,18,15,12,10,7,5) (0,1,0,0,1,1,0)

③W=(15,7,20,5,18,10,12)

这三种情况的计算时间有明显的差别吗？