算法期末复习题按章节整理

按今年复习范围整理：复习贴纸内容已放大重排，截图题目按章节归档。

第 1-2 章 基础与证明

一、组合数递归算法正确性（数学归纳法证明）

递归定义：

C(n,m) = C(n-1,m) + C(n-1,m-1)，边界 C(n,0)=C(n,n)=1。递归函数 CRecursive(n,m)：若 m=0 或 m=n 返回 1；否则返回 CRecursive(n-1,m)+CRecursive(n-1,m-1)。

证明（对 n 用第二数学归纳法）：

命题 P(n)：对任意满足 0≤m≤n 的 m，CRecursive(n,m) 能正确返回 C(n,m)。

① 基础步（n=0）：此时只有 m=0，满足 m=0，函数返回 1 = C(0,0)。P(0) 成立。

② 归纳假设：设对所有 k < n（k≥0），P(k) 成立，即 CRecursive(k,m) 对一切 0≤m≤k 正确返回 C(k,m)。

③ 归纳步：考察 CRecursive(n,m)，0≤m≤n。

- 若 m=0 或 m=n：函数直接返回 1。而 C(n,0)=C(n,n)=1，正确。
- 若 0<m<n：函数返回 CRecursive(n-1,m)+CRecursive(n-1,m-1)。由归纳假设（n-1<n），两次调用分别正确返回 C(n-1,m) 和 C(n-1,m-1)。由帕斯卡公式

C(n,m)=C(n-1,m)+C(n-1,m-1)，故返回值 = C(n,m)，正确。

④ 结论：由第二数学归纳法，对一切 n≥0 及 0≤m≤n，CRecursive(n,m) 正确计算 C(n,m)。•

第 5 章 分治法

二、二分（对半）搜索正确性（数学归纳法证明）

算法：

在有序表 l[left..right] 中查找 x。取 m=(left+right)/2：若 x=l[m] 返回 m；若 x<l[m] 在 l[left..m-1] 中递归找；若 x>l[m] 在 l[m+1..right] 中递归找；若 left>right 返回失败(-1)。

证明（对子表长度 n = right-left+1 用第二数学归纳法）：

命题 P(n)：对任意长度为 n 的有序子表，BSearch 能正确判断 x 是否存在：存在则返回其下标，不存在则返回 -1。

① 基础步（n=0）：此时 left>right，子表为空，x 必不存在，函数返回 -1，正确。P(0) 成立。（n=1 时：子表只有 l[m]，比较 x 与 l[m]，相等返回 m，否则进入空子表返回 -1，亦正确。）

② 归纳假设：设对所有长度 k < n 的有序子表，P(k) 成立。

③ 归纳步：考察长度为 n（n≥1）的有序子表 l[left..right]，取 m=(left+right)/2。

- 若 x=l[m]：返回 m，正确。
- 若 x<l[m]：因表有序，x 若存在必在左半 l[left..m-1] 中（右半元素均≥l[m]>x）。左半长度 = m-left < n，由归纳假设递归调用正确。
- 若 x>l[m]：同理 x 若存在必在右半 l[m+1..right] 中，右半长度 = right-m < n，由归纳假设递归调用正确。

三种情形都返回正确结果，故 P(n) 成立。

④ 结论：由第二数学归纳法，对一切 n≥0，BSearch 在长度为 n 的有序表上正确判断 x 的存在性并返回正确结果。•

关键点：① 表必须有序，是“另一半可被排除”的依据；② 子问题长度严格变小，保证递归终止且可施归纳。

5-8 三分搜索算法

5-8 三分搜索算法的做法是：它先将待查元素 x 与 $n/3$ 处的元素比较，然后将 x 与 $2n/3$ 处的元素比较。比较结果或者是找到 x，或者将搜索范围缩小到原来的 $n/3$。

（1）编写 C++程序实现算法；

（2）分析算法时间复杂度。

```
template<class T>
intSortableList<T>::Search(T x)const   //返回搜索的元素 x 的下标
{ return Search(0, n-1, x);
}

template<class T>
intSortableList<T>::Search(int left, int right, T x)const   //平均时间复杂度为θ(log3n)
{   int m1,m2;

    if (left==right)     { if (l[left]==x) return left; }
    if (left<right)
    {   m1=left+(right-left)/3;//左分界点
        m2=(int)ceil((double)(right-(right-left)/3));//右分界点
        if (x==l[m1])     return m1;
        else if (x<l[m1])   return Search(left,m1-1,x);
        else if (x<l[m2])   return Search(m1+1,m2-1,x);
        else if(x==l[m2])   return m2;
```

```
intSortableList<T>::Search(int left, int right, T x)const    //平均时间复杂度为θ(log3n)
{   int m1,m2;

    if (left==right)      { if (l[left]==x) return left; }
    if (left<right)
    {   m1=left+(right-left)/3;//左分界点
        m2=(int)ceil((double)(right-(right-left)/3));//右分界点
        if (x==l[m1])      return m1;
        else if (x<l[m1])    return Search(left,m1-1,x);
        else if (x<l[m2])    return Search(m1+1,m2-1,x);
        else if(x==l[m2])    return m2;
        else    return Search(m2+1,right,x);
    }
    else return -1;
}
```

5-11.对两组数据：(1,1,1,1,1)和(5,5,8,3,4,3,2)执行程序 5-12 的快速排序，按照表 5-11 的格式分别列表表示执行

步数	0	1	2	3	4	5
初始时						
1						
2						
3						
排序结果						

解答：

步数	0	1	2	3	4	5
初始时	$1_{(0)}$	$1_{(1)}$	$1_{(2)}$	$1_{(3)}$	$1_{(4)}$	∞
1	$[1_{(3)}$	$1_{(4)}]$	$1_{(0)}$	$[1_{(2)}$	$1_{(1)}]$	∞
2	$[1_{(4)}]$	$1_{(3)}$	$1_{(0)}$	$[1_{(2)}$	$1_{(1)}]$	∞
3	$1_{(4)}$	$1_{(3)}$	$1_{(0)}$	$[1_{(2)}$	$1_{(1)}]$	∞
4	$1_{(4)}$	$1_{(3)}$	$1_{(0)}$	$[1_{(0)}]$	$1_{(2)}$	∞
排序结果	$1_{(4)}$	$1_{(3)}$	$1_{(0)}$	$1_{(1)}$	$1_{(2)}$	∞

步数	0	1	2	3	4	5	6	7
初始时	5	5	8	3	4	3	2	∞
1	[4	2	3	3]	5	[8	5]	∞
2	[3	2	3]	4	5	[8	5]	∞
3	[3	2]	3	4	5	[8	5]	∞

3

2	[3	2	3]	4	5	[8	5]	∞
3	[3	2]	3	4	5	[8	5]	∞

3

4	[2]	3	3	4	5	[8	5]	∞
5	2	3	3	4	5	[5]	8	∞
排序结果	2	3	3	4	5	5	8	∞

第 6 章 贪心法

6-1 背包问题；6-8 最佳合并树

第六章

6-1 设有背包问题实例 n=7, M=15, $(x_0,x_1,\ldots,x_6)$=(2,3,5,7,1,4,1)，物品装入背包的收益 $(p_0,p_1,\ldots,p_6)$=(10,5,15,7,6,18,3)。求这一实例的最优解和最大收益。

01 2 3 4 5 601 2 3 4 5 6

解：由题可得：$\left(\frac{p_0}{w_0},\frac{p_1}{w_1},\frac{p_2}{w_2},\frac{p_3}{w_3},\frac{p_4}{w_4},\frac{p_5}{w_5},\frac{p_6}{w_6}\right)=\left(\frac{10}{2},\frac{5}{3},\frac{15}{5},\frac{7}{7},\frac{6}{1},\frac{18}{4},\frac{3}{1}\right)$=(5, 1.67, 3, 1, 6, 4.5, 3)，

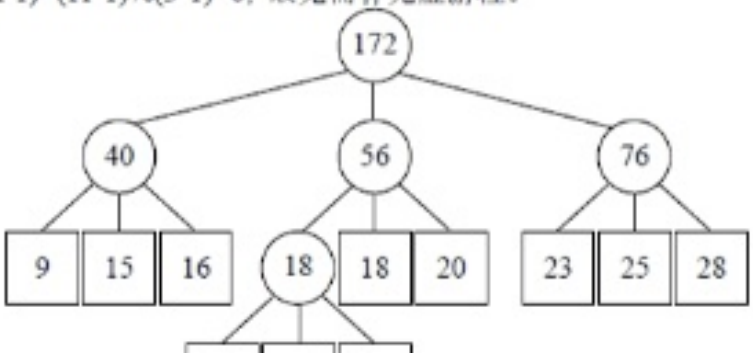
0 1 2 3 4 5 6

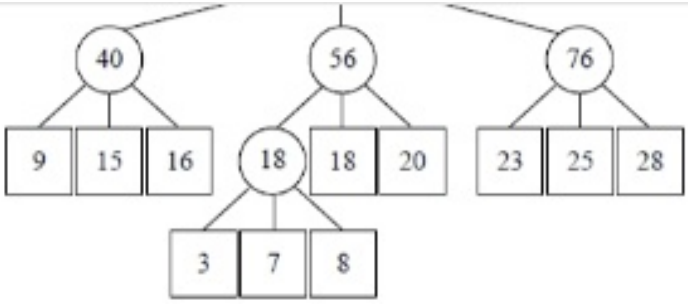
所以，最优解为$(x_0,x_1,x_2,x_3,x_4,x_5,x_6)=\left(1,\frac{2}{3},1,0,1,1,1\right)$，或$(x_0,x_1,x_2,x_3,x_4,x_5,x_6)$=(1, 1, 1, 1, 1, 2/3, 0)。

最大收益为$10+5\times\frac{2}{3}+15+6+18+3=55\frac{1}{3}$=166/3=55.33

6-8 设 W={3,7,8,9,15,16,18,20,23,25,28}，构造一棵最优 3 路合并树。

题解：因 (n-1)%(K-1)=(11-1)%(3-1)=0，故无需补充虚游程。





6-9 图 G=(V,E)是一个无向连通图，$n=|V|$，$m=|E|$，且 $m=O(n^{1.99})$，试问选择何种算法求最小代价生成树，普里姆还是克鲁斯卡尔算法？

题解：用普里姆算法。

因为图 G 是一个无向连通图，所以 $n-1 \leqslant m \leqslant n(n-1)/2$; $O(n) \leqslant O(m) \leqslant O(n^2)$;

克鲁斯卡尔对边数较少的带权图有较高的效率，而 $m=O\left(n^{1.99}\right) \approx O\left(n^2\right)$，此图边数较多，接近完全

故选用普里姆算法。

第 7 章 动态规划法

三、关键路径问题（AOE 网）程序实现

两个递推式：

```
earliest■■■■■■■■■■■■■■■ max■■earliest(0)=0■earliest(j)=max{earliest(i)+w(i,j) | i∈P(j)
■■■}■
latest■■■■■■■■■■■■■■■■ min■■latest(n-1)=earliest(n-1)■latest(i)=min{latest(j)-w(i,j) |
j∈S(i) ■■■}■
```

关键活动判定：边 <i,j> 满足 latest(j)-earliest(i)=w(i,j) 即为关键活动。

算法（结点已按拓扑次序编号，邻接矩阵 a，无边为 INFTY）：

```
void CriticalPath() {
T earliest[n], latest[n]; int i, j;
// Step1: compute earliest in topological order (take MAX)
earliest[0] = 0;
for (j = 1; j < n; j++) {
earliest[j] = 0;
for (i = 0; i < j; i++) // predecessors i of j (i<j)
if (a[i][j] < INFTY)
if (earliest[i] + a[i][j] > earliest[j])
earliest[j] = earliest[i] + a[i][j];
}
// Step2: compute latest in reverse topo order (take MIN)
latest[n-1] = earliest[n-1];
for (i = n-2; i >= 0; i--) {
latest[i] = earliest[n-1];
for (j = i+1; j < n; j++) // successors j of i (j>i)
if (a[i][j] < INFTY)
if (latest[j] - a[i][j] < latest[i])
latest[i] = latest[j] - a[i][j];
}
// Step3: output critical activities
for (i = 0; i < n; i++)
for (j = i+1; j < n; j++)
if (a[i][j] < INFTY)
if (latest[j] - earliest[i] == a[i][j])
cout << "<" << i << "," << j << "> ";
// shortest project time = earliest[n-1]
}
```

代码步骤中文对照：Step1 按拓扑序算 earliest（前驱取 max）；Step2 按逆拓扑序算 latest（后继取 min）；Step3 逐边判定关键活动；工程最短完成时间 = earliest[n-1]。

要点：

① earliest 取 max（等所有前驱完成），latest 取 min（被所有后继约束）；② 结点按拓扑序编号后，前驱满足 i<j、后继满足 j>i，无需显式拓扑排序；③ 时间复杂度 $O(n^2)$（邻接表可优化到 O(n+e)）；④ 关键路径由关键活动 <i,j> 确定，不能只看 latest(i)=earliest(i) 的事件结点；⑤ 工程最短完成时间 = earliest(n-1) = 关键路径长度（最长路径）。

7-1 多段图后向递推

7-1. 写出对图 7-19 所示的多段图采用向后递推动态规划算法求解时的计算过程。

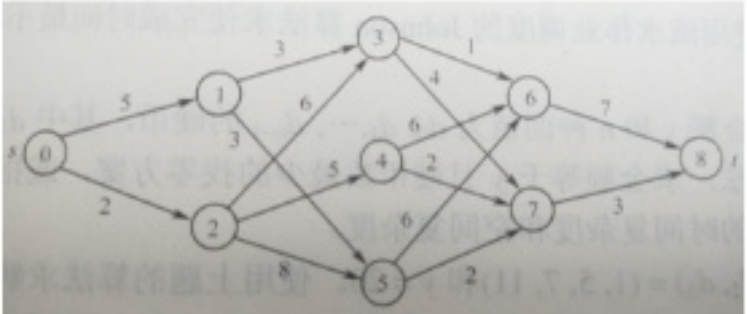


图 7-19　习题 7-1 的图

向后递推过程如下：

Bcost(1,0)=0　　d(1,0)=-1;

Bcost(2,1)=c(1,1)+Bcost(1,0)=5　　d(2,1)=0;

Bcost(2,2)=c(1,2)+Bcost(1,0)=2　　d(2,2)=0;

Bcost(3,3)=min{c(2,3)+Bcost(2,2),c(1,3)+Bcost(2,1)}=min{6+2,3+5}=8　　d(3,3)=2;

Bcost(3,4)=c(2,4)+Bcost(2,2)=5+2=7　　d(3,4)=2;

Bcost(3,5)=min{c(1,5)+Bcost(2,1),c(2,5)+Bcost(2,2)}=min{3+5,8+2}=8　　d(3,5)=1;

Bcost(4,6)=min{c(3,6)+Bcost(3,3),c(4,6)+Bcost(3,4),c(5,6)+Bcost(3,5)}=min{1+8,6+7,6+8}=9　d(4,6)=3

Bcost(4,7)=min{c(3,7)+Bcost(3,3),c(4,7)+Bcost(3,4),c(5,7)+Bcost(3,5)}=min{4+8,2+7,6+8}=9　d(4,7)=4

Bcost(5,8)=min{c(6,8)+Bcost(4,6),c(7,8)+Bcost(4,7)}=min{7+9,3+9}=12d(5,8)=7;

最短路径长度为 Bcost(5,8)=12。

最短路径为（0，2，4，7，8）。

7-2 多段图最短路径

7-2.使用向前递推和向后递推两种方法，求图 7-19 所示的多段图中从 s 到 t 的最短路径及路径长度。

向后递推的计算过程如上题所示，

最短路径长度为 Bcost(5,8)=12。

注意，d[i,j]存储最短路径上第 i 阶段 j 顶点前的一个顶点号。

从汇点 s(即 8)开始，向前查找。

取 d(5,8)(即 7)，即 8 的前一个顶点事 7，取 d(4,7)(即 4)；取 d(3,4)(即 2)，取 d(2,2)(即 0)，到达源点

故从 s 到 t 的最短路径为 (d(2,2)=0, d(3,4)=2, d(4,7)=4, d(5,8)=7, 8)，即（0，2，4，7，8）。

7-9. 给定字符串 A="xzyzzyx"，B="zxyyzxz"，使用 LCS 算法求最长公共子串，并给出一个最长公共

char A[8]={'0','x','z','y','z','z','y','x'}

B[8]={'0','z','x','y','y','z','x','z'}

$$\begin{bmatrix} 0&0&0&0&0&0&0&0\\ 0&0&1&1&1&1&1&1\\ 0&1&1&1&1&2&2&2\\ 0&1&1&2&2&2&2&2\\ 0&1&1&2&2&3&3&3\\ 0&1&1&2&2&3&3&4\\ 0&1&1&2&3&3&3&4\\ 0&1&2&2&3&3&4&4 \end{bmatrix} \quad \begin{bmatrix} 0&0&0&0&0&0&0&0\\ 0&2&1&3&3&3&1&3\\ 0&1&2&2&2&1&3&1\\ 0&2&2&1&1&2&2&2\\ 0&1&2&2&2&1&3&1\\ 0&1&2&2&2&1&2&1\\ 0&2&2&1&1&2&2&2\\ 0&2&1&2&2&2&1&2 \end{bmatrix}$$

(a) c[i][j]　　(b) s[i][j]

7-15 0/1 背包阶跃点

7-15. 设有 0/1 背包实例$(w_0,w_1,w_2w_3)=(10,15,6,9)$，$(p_0,p_1,p_2,p_3)=(2,5,8,1)$和 M=32。试计算 S^i,0 给出最优解值和最优解(要有计算过程)。另使用启发式方法再计算一次。(题意要求用教材 P165 的计算步骤 由于没有讲，故用 P162 的步骤计算)

题解：$S^{-1}=\{(0,0)\}$，在 S^{-1} 上加上$(w_0,p_0)=(10,2)$，得 $S_1^0=\{(10,2)\}$。

合并：$S^{-1}\cup S_1^0$，得 $S^0=\{(0,0),(10,2)\}$。

在 S^0 上加上$(w_1,p_1)=(15,5)$，得 $S_1^1=\{(15,5),(25,7)\}$。

合并：$S^0\cup S_1^1$，得 $S^1=\{(0,0),(10,2),(15,5),(25,7)\}$。

在 S^1 上加上$(w_2,p_2)=(6,8)$，得 $S_1^2=\{(6,8),(16,10),(21,13),(31,15)\}$，

合并：$S^1\cup S_1^2$，得 $S^2=\{(0,0),(6,8),(16,10),(21,13),(31,15)\}$，其中(10,2)、(15,5)、(25,7)被(6,8)支被舍弃。

在 S^2 上加上$(w_3,p_3)=(9,1)$，得 $S_1^3=\{(9,1),(15,9),(25,11),(30,14),(40,16)\}$，

合并：$S^2\cup S_1^3$，得 $S^3=\{(0,0),(6,8),(15,9),(16,10),(21,13),(30,14),(31,15)\}$，其中有三个阶跃点已(9,1)被(6,8)支配，(25,11)被(21,13)支配，而(40,16)，由于 40>M(=32)，故此三个点被舍弃。

最优解值(最大收益)为 15(注意取得最大收益 15 时，背包装了 10+15+6=31，即背包没有装满，还有剩余

确定最优解：从(31,15)开始，看它属于S_1^3还是 S^2。由于$(31,15)\in S^2$，故$x_3=0$。

(31,15)(不用减去(9,1))，看(31,15)属于S_1^2还是 S^1。由于$(31,15)\in S_1^2$，故$x_2=1$。

(31,15)减去(6,8)，得(25,7)，看它属于S_1^1还是 S^0。由于$(25,7)\in S_1^1$，故$x_1=1$。

(25,7)减去(15,5)，得(10,2)，看它属于S_1^0还是 S^{-1}。由于$(10,2)\in S_1^0$，故$x_0=1$。

因此最优解(x_0,x_1,x_2,x_3)=(1, 1, 1, 0)。

第 8 章 回溯法

8-2 n 皇后：输出全部可行解

8-2 程序 8-4 求 *n*-皇后问题的全部可行解，请修改这一程序，使之在求得第一个可行解后算法终止。

下面的递归函数 NQueens 可以输出 N(可以指定 N=1)个解：

```
int count=0; int N=1;
voidNqueens(int k,int n,int *x)
{   for(inti=0;i<n&&count<N;i++)
      if(place(k,i,x))
      {   x[k]=i;
          if(k==n-1)
          { for(i=0;i<n;i++)cout<<x[i];
              cout<<endl;
              count++;
          }
          else Nqueens(k+1,n,x)
      }
}
voidNqueens(int n,int *x)
{   Nqueens(0,n,x);
}
```

下面的递归函数 NQueens 只输出一个解：

```
voidNQueens(int k, int n, int *x, bool &flag)
{    for(int i=0; i<n&&flag;i++)
     {   if(Place(k,i,x))
         {   x[k]=i;
             if(k==n-1)
             {   for(i=0;i<n;i++)   cout<<x[i]<<" ";
                 cout<<endl;        flag=false;
```

```
{   for(int i=0; i<n&&flag;i++)
    {   if(Place(k,i,x))
        {   x[k]=i;
            if(k==n-1)
            {   for(i=0;i<n;i++)   cout<<x[i]<<" ";
                cout<<endl;          flag=false;
            }
            else NQueens(k+1,n,x,flag);
        }//end if(Place)
    }//end for
}
voidNqueens(int n,int *x)
{   bool flag=true;
    NQueens(0,n,x,flag);
}
```

8-6 设有子集和数问题的实例 $W=(w_0,w_1,\ldots,w_6)=(5,7,10,12,15,18,20)$和 $M=35$。求 W 中元素之和等于 M 的所有
画出对于这一实例由 **SumOfSub** 算法实际生成的那部分状态空间树。

状态空间树如下：

第 9 章 分支限界法

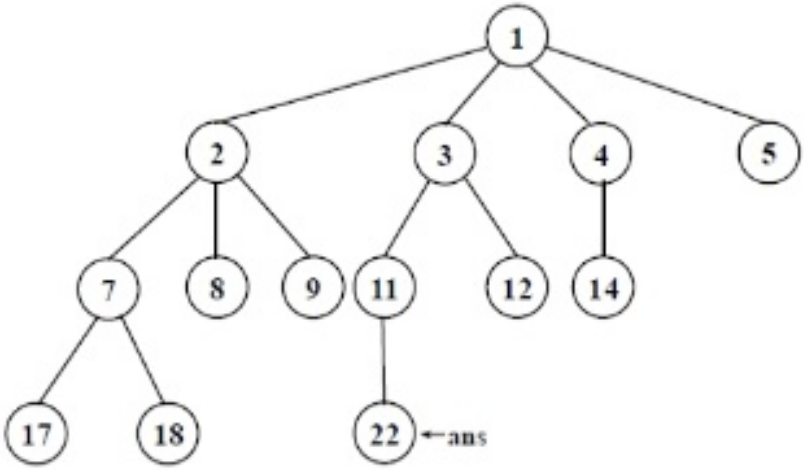
9.2 设有带时限的作业排序问题实例$(p_1,p_2,\dots,p_5)=(6,3,4,8,5)$，$(t_1,t_2,\dots,t_5)=(2,1,2,1,1)$和$(d_1,d$
$=(3,1,4,2,4)$.求问题的最优解及对应于最优解的收益损失.画出 JSFIFOBB 算法实际生成的那部分状态

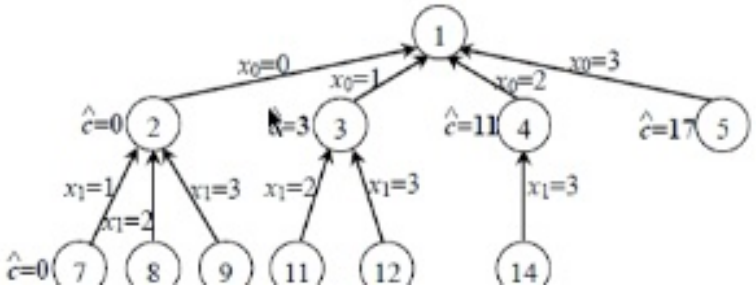
0 1 2 3 4

p={3,8,6,4,5}　　total=26　　U=7

d={1,2,3,4,4}最优解值=19

t={1,1,2,2,1}最优解 X={1,2,4}　　(p,d,t)= (8,2,1) , (6,3,2) , (5,4,1)





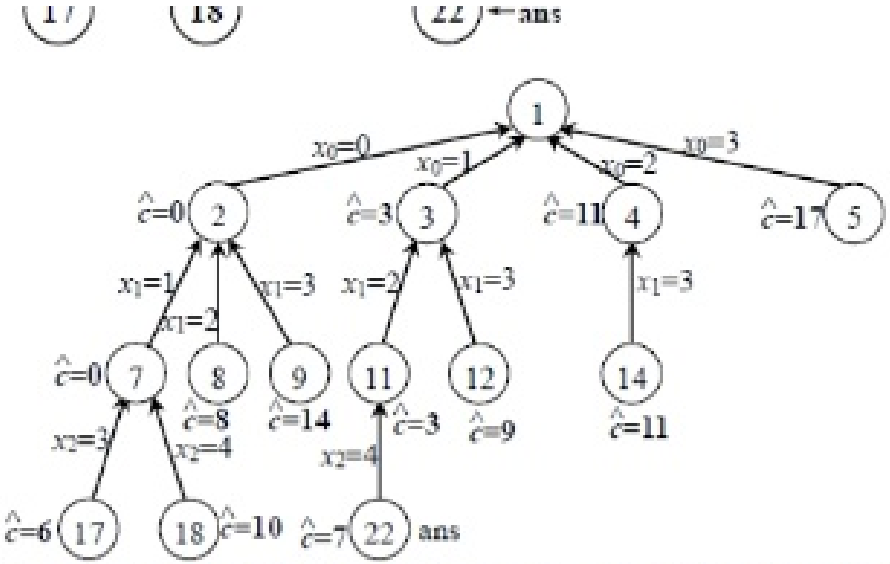


图 JSLIFOBB 算法的状态空间树(固定长度元组，按生成结点的顺序编号，不

9-8 设有 0/1 背包问题实例 n=5，对于以下两种情况：

（1）$\{p_1, p_2, \ldots, p_5\}=\{10,15,6,8,4\}$，$\{w_1, w_2, \ldots, w_5\}=\{4,6,3,4,2\}$和 $m=12$；

（2）$\{p_1, p_2, \ldots, p_5\}=\{w_1, w_2, \ldots, w_5\}=\{4,4,5,8,9\}$和 $m=15$。

分别求问题的最优解和最优解值，并画出采用 LC 分枝限界算法实际生成的那部分状态空间树。

（2）$\{p_1, p_2, \ldots, p_5\}=\{w_1, w_2, \ldots, w_5\}=\{4,4,5,8,9\}$和 $m=15$。

分别求问题的最优解和最优解值，并画出采用 LC 分枝限界算法实际生成的那部分状态空间

解：

（1）