《算法设计与分析》

第六章 回溯法

马丙鹏

2023年11月06日

第六章 回溯法

- 6.1 一般方法
- 6.2 8-皇后问题
- 6.3 子集和数问题
- 6.4 图的着色
- 6.5 0/1背包问题

6.3 子集和数问题

■ 问题描述

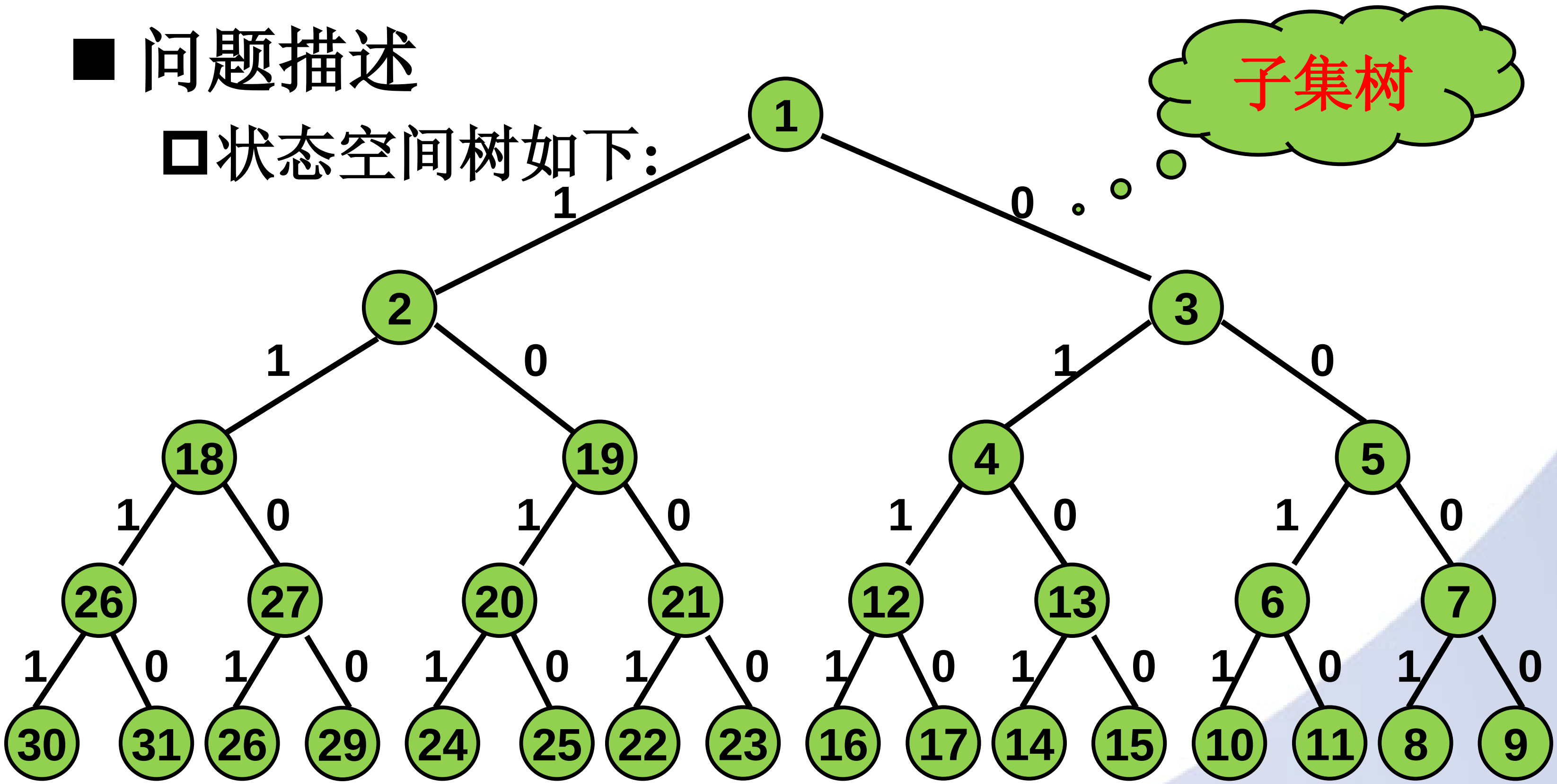
- 已知n个不同的正数$(w_1, w_2, \ldots, w_n)$。要求找出w_i的和数等于某个正数M的所有子集。
  - 例: 设有n = 4个正数的集合 $W = \{w_1, w_2, w_3, w_4\} = (11, 13, 24, 7)$和正整数$M = 31$，
  - 求W所有满足条件的子集，使得子集中的正数之和等于M。
- 采用固定长度n元组$(x_1, \ldots, x_n)$表示解, $x_i \in \{0, 1\}, 1 \le i \le n$。
  - $x_i=0$，表示w_i未入选子集；$x_i=1$，表示w_i入选子集。
  - 则显式约束为：$x_i \in \{0, 1\}\ (0 \le i \le n)$
  - 隐式约束为：$\sum_{i=1}^{k} W(i)X(i) = M$

中国科学院大学
University of Chinese Academy of Sciences

6.3 子集和数问题

■ 问题描述

□状态空间树如下:



子集和数问题（固定长度元组解）状态空间树

每个叶结点是一个解状态，代表一个候选解元组。
非叶结点代表部分向量。

6.3 子集和数问题

- 问题描述
  - 限界函数的选择
  - 约定：**W(i)按非降次序排列**

条件一：
$$\sum_{i=1}^{k} W(i)X(i)+\sum_{i=k+1}^{n} W(i)\geq M$$

条件二：
$$\sum_{i=1}^{k} W(i)X(i)+W(k+1)\leq M$$

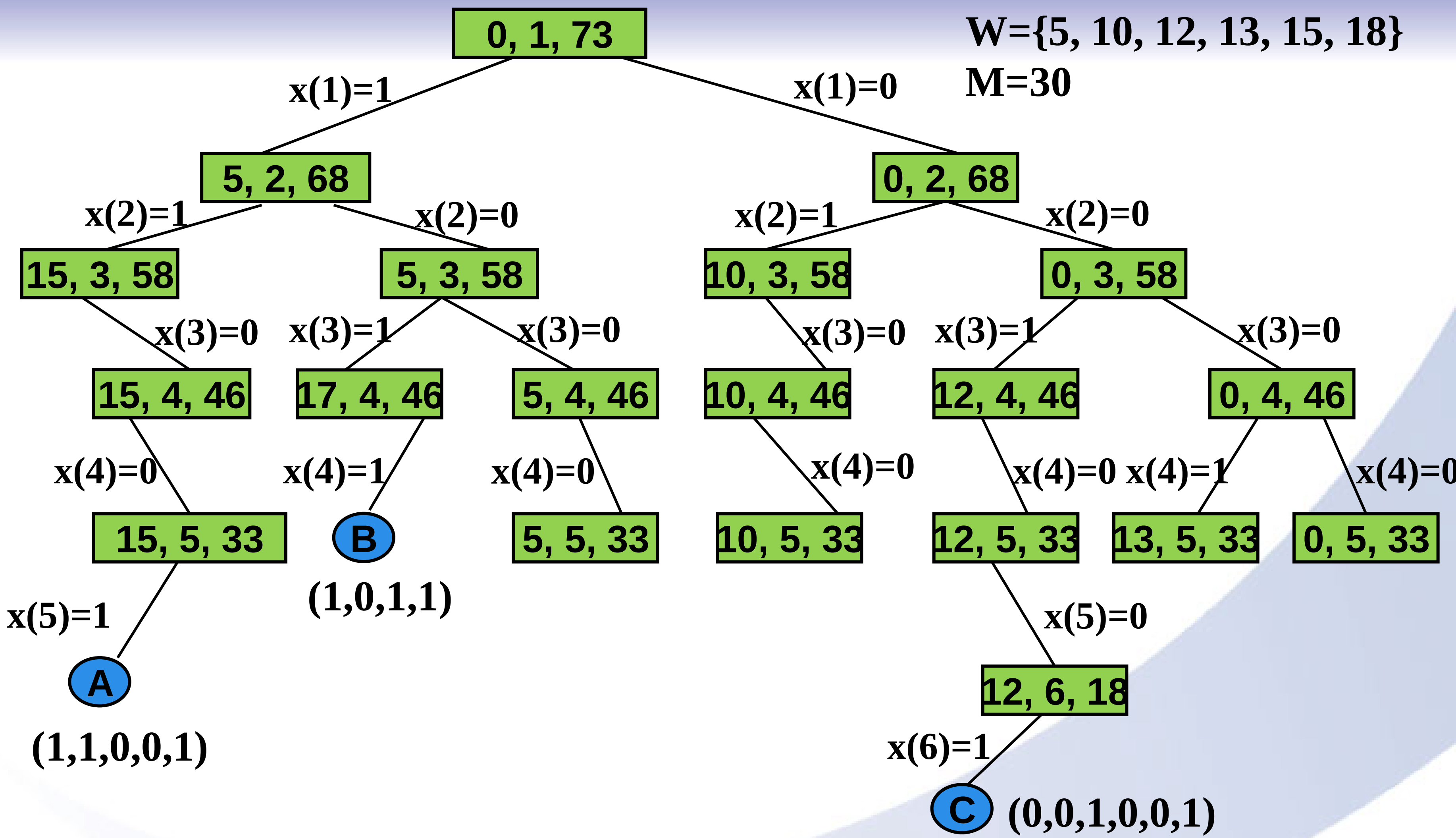
  - 仅当满足上述两个条件时，限界函数**B(X(1), …, X(k)) = true**
  - 注：如果不满足上述条件，则**X(1), …, X(k)**根本不可能导致一个答案结点。

```
procedure SUMOFSUB(s, k, r)
    global integer M, n; global real W(1:n); global boolean X(1:n)
    real r, s; integer k, j;
    //找W(1：n)中和数为M的所有子集。进入此过程时X(1), …, X(k-1)的值已确定。W(j)按非降次序排列。//
    //生成左儿子//
    X(k)←1
    if s+W(k)=M then
        print(X(j), j←1 to k)
    else
        if s+W(k)+W(k+1) ≤ M then
            call SUMOFSUB(s+W(k), k+1, r-W(k))
        endif
    endif
    //生成右儿子和计算Bk的值//
    if  s+r-W(k) ≥ M and s+W(k+1) ≤ M  then
        X(k)←0
        call SUMOFSUB(s, k+1, r-W(k))
    endif
end SUMOFSUB
```

6.3 子集和数问题

- 实例
  - 设有n=6个正数的集合W={5, 10, 12, 13, 15, 18}和整数M=30，求W的所有元素之和为M的子集。
  - 求解见下图
    - 方形结点：s, k, r
    - 圆形结点：输出答案的结点
    - 共生成20个结点

W={5, 10, 12, 13, 15, 18}
M=30



$$\sum_{i=1}^{k} W(i)X(i)+\sum_{i=k+1}^{n} W(i)\ge M \qquad \sum_{i=1}^{k} W(i)X(i)+W(k+1)\le M$$

第六章 回溯法

- 6.1 一般方法
- 6.2 8-皇后问题
- 6.3 子集和数问题
- 6.4 图的着色
- 6.5 0/1背包问题

6.4 图的着色

- 问题描述
  - m-着色判定问题：
    - 已知一个图G=(V, E)和m>0种不同的颜色，只允许使用这m种颜色对图G的结点着色，每个结点着一种颜色，
    - 问是否存在一种着色方案，使得图中任意相邻的两个结点都具有不同的颜色。
  - m-着色最优化问题：
    - 求可对图G着色的最小整数m。
    - 这个整数被称为图G的色数。

6.4 图的着色

- 问题描述
  - 地图的着色问题:
    - 1976年爱普尔(K.L. Apple)，黑肯(W. Haken)和考西(J. Koch)利用电子计算机证明了4种颜色足以对任何地图着色。
  - 四色定理
    - 每幅地图都可以用不多于4种颜色来着色，使得有共同边界的国家着不同的颜色。
  - 平面图的4-着色判定问题
    - 一幅地图很容易用一个平面图G表示。

6.4 图的着色

■ 问题描述

- 将地图转换为图
  - ➢将地图的每个区域用图G的一个结点表示，若两个区域相邻，则相应的两个结点用一条边连接起来。

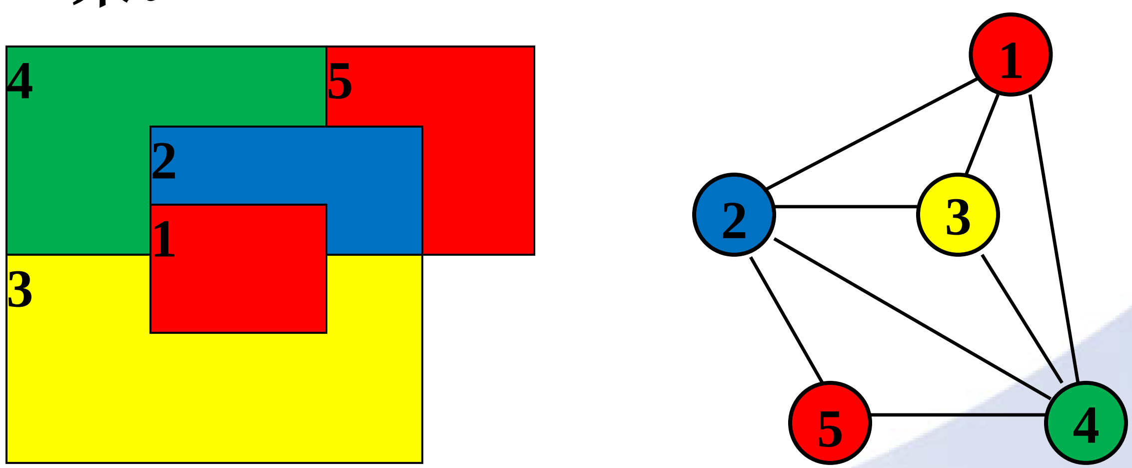


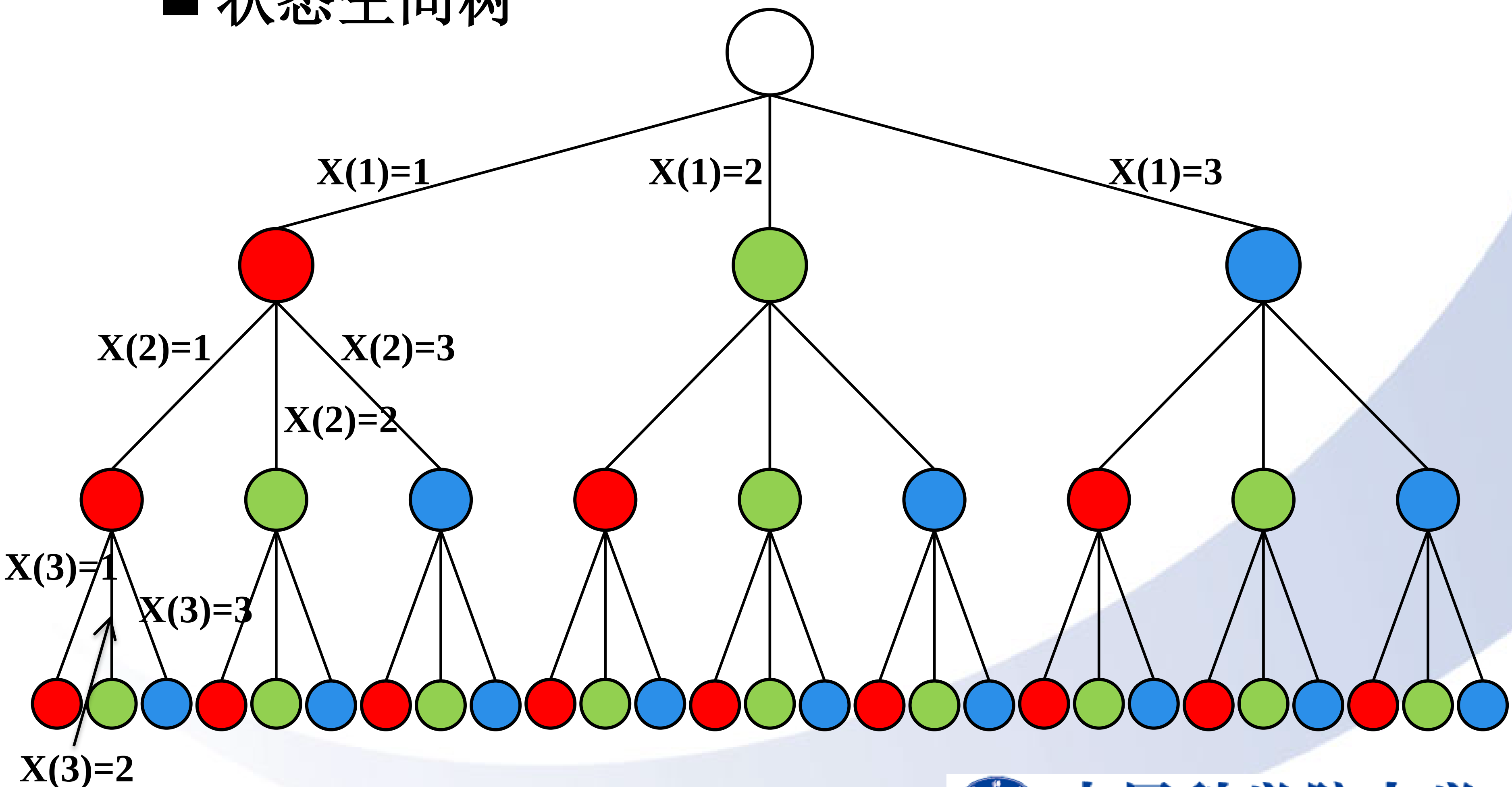
图6.10 一幅地图和他的平面图表示

6.4 图的着色

- 解空间构建
  - 颜色数用1, 2, …, m来表示。
  - 采用n-元组(x(1), …, x(n))表示图G的m-着色判定问题的解，x(i)是结点i的颜色。
  - 采用邻接矩阵表示无向图G=(V, E)。
- 限界函数
  - 任何相邻的两个结点都具有不同的颜色：如果边(i, j)∈E, i≠j, 则x(i) ≠ x(j)。
  - 将合法的颜色分配给x(i)。

6.4 图的着色

- 状态空间树



6.4 图的着色

- 算法实现
  - 以深度优先方式生成状态空间树中的结点，寻找所有答案结点，即m-着色方案。
  - 搜索中使用约束函数剪去不可能包含答案结点的分枝。
  - 对给定的无向图G和m，列出图中结点所有可能的m-着色方案。

中国科学院大学
University of Chinese Academy of Sciences

6.4 图的着色

- 算法实现
  - 初始化（起始状态）
  - 从第一个顶点开始
    - 由当前顶点安排下一个顶点可以设置的颜色
    - 如果找到可以设置的颜色
      - 置顶点颜色
      - 如果已经是最后一个顶点
        - ✓ 得到一个解
        - ✓ 撤掉该子，继续寻找下一个解
      - 否则(未到最后)
        - ✓ 准备处理下一个顶点
    - 否则（没有找到可以设置的颜色）
      - 回溯到上一个顶点，并去除该顶点的颜色

算法6.7 找一个图的所有m-着色方案

```
procedure MCOLORING(k)
  //这是图着色的一个递归回溯算法。图G用它的布尔邻接矩阵GRAPH(1:n,1:n)表示它计算并打印出符合以下要求的全部解，把整数1, 2, …, m分配给图中各个结点且使用相邻近的结点的有不同的整数。k是下一个要着色结点的下标//
    global integer m, n, X(1, …, n); boolean GRAPH(1:n, 1:n)
    integer k
    loop  //产生对X(k)所有合法赋值//
        call NEXTVALUE(k)  //将一种合法的颜色分配给X(k)//
        if X(k)=0 then exit endif //没有可用的颜色//
        if k=n
        then print(X)  //至多用了m种颜色分配给n个结点//
        else call MCOLORING(k+1)//所有m着色方案均在此反复递归调用产生//
        endif
    repeat
end MCOLORING
```

中国科学院大学
University of Chinese Academy of Sciences

6.4 图的着色

- 算法说明
  - 在最初调用call MCOLORING(1)之前，应对图的邻接矩阵置初值并对数组X置0值。
  - 在确定了X(1)到X(k-1)的颜色之后，过程NEXTVALUE从这m种颜色中挑选一种符合要求的颜色，并把它分配给X(k)，
  - 若无可用的颜色，则返回X(k)=0。

算法6.8 生成下一种颜色

```
Procedure NEXTVALUE(k)
//进入此过程前X(1), …, X(k-1)已分得了区域[1, m]中的整数且相邻近的结点有不同的整数。本过程在区域[0, m]中给X(k)确定一个值；如果还剩下一些颜色，他们与结点k邻接的结点分配的颜色不同，就将其中最高标准的颜色分配给结点k；如果没有剩下可用的颜色，则置X(k)为0//
  global integer m, n, X(1:n); boolean GRAPH(1:n, 1:n);  integer j, k
  loop
        X(k) ← (X(k)+1) mod (m+1)  //试验下一个最高标准值的颜色//
        if X(k)=0 then return endif  //全部颜色用完//
        for j ← 1 to n do  //检查此颜色是否与邻近结点的那些颜色不同//
          if GRAPH(k, j) and //如果(k, j)是一条边//
             X(k)=X(j)   //并且邻近的结点有相同颜色//
          then exit endif
        repeat
        if j=n+1 then return endif //找到一种颜色//
  repeat  //否则试着找另一种颜色//
end NEXTVALUE
```

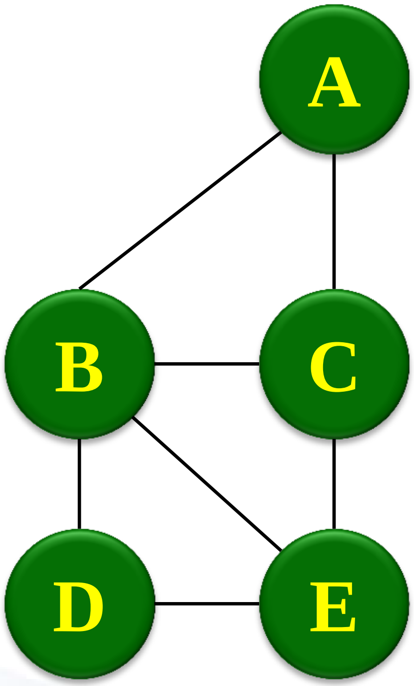
6.4 图的着色

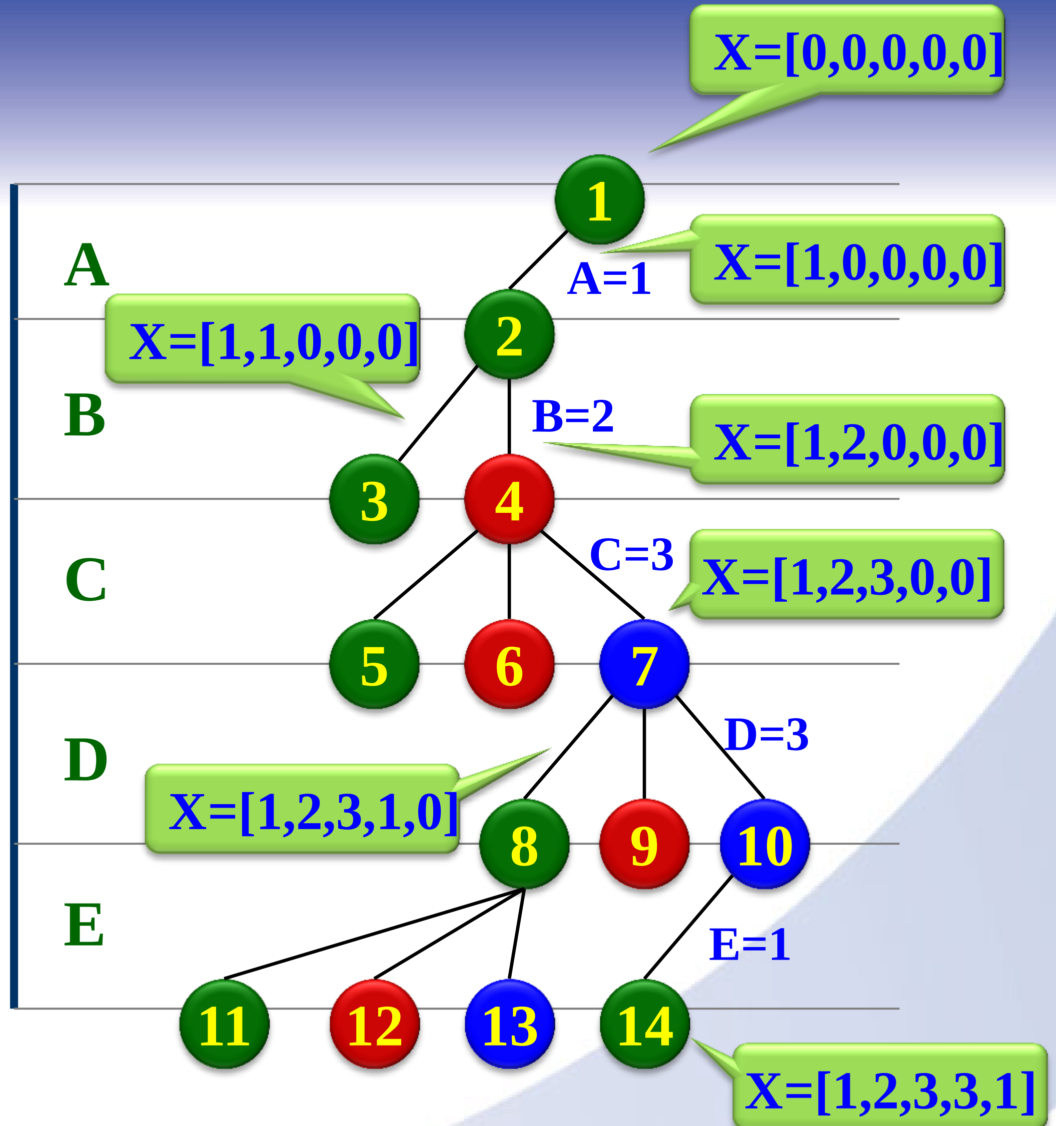
- **算法说明**
  - 算法6.7的计算时间上界可以由状态空间树的内部结点 $\sum_{i=0}^{n-1} m^i$ 得到。
  - 在每个内部结点处，为了确定它的儿子们所对应的合法着色，由NEXTVALUE所花费的时间是O(mn)。因此，总的时间由下式所限界。

$$\sum_{i=1}^{n} m^i n = n(m^{n+1} - m)/(m-1) = O(nm^n)$$

6.4 图的着色

- ■ 实例分析
  - □例 三着色图





总点数为1+3+9+27+81+243=364

实际搜索14个结点

第六章 回溯法

- 6.1 一般方法
- 6.2 8-皇后问题
- 6.3 子集和数问题
- 6.4 图的着色
- 6.5 0/1背包问题

6.5 0/1背包问题

- 问题描述
  - 假定n个物品的重量w_i，效益值p_i和背包容量M均为已知的正数，
  - 目标函数

$$\sum_{1\le i\le j} p_i x_i$$

  - 约束条件

$$\sum_{1\le i\le j} w_i x_i \le M$$

$$x_i = 0\text{或}1, p_i > 0, w_i > 0, 1 \le i \le j$$

6.5 0/1背包问题

- 问题描述
  - n-皇后问题、子集和数问题、m-图着色问题的求解目标都是求满足约束条件的全部可行解。
  - 0/1背包问题是最优化问题。
  - 0/1背包问题是一个困难问题(难以设计最坏情况下可多项式时间求解的算法)。
  - 回溯法本质上是一种深度优先搜索状态空间树的算法。
  - 如果不引入剪枝函数(约束函数+限界函数)，则是穷举算法。
  - 引入适当的限界函数，剪去已能确信不含最优答案结点的子树，使其成为一种启发式算法。

6.5 0/1背包问题

- 解的表示
  - 本节讨论采用固定长度元组解结构的0/1背包问题的回溯算法。
  - 0/1背包问题的解用n-元组表示：$X=(x_1, x_2, \ldots, x_n)$, $x_i=0$或$1(1\le i\le n)$。
  - 解空间大小为2^n。解空间树为高度为n+1的满二叉树。
- 显示约束：
  - $x_i=1$表示将第i件物品装入背包,
  - $x_i=0$表示第i件物品不装入背包。

6.5 0/1背包问题

■ 隐式约束：

$$\sum_{i=1}^{n} w_i x_i \le M \quad w_i > 0, x_i = 0\text{或}1$$

约束函数：$B_k(x_1, x_2, \ldots, x_k) = true$，当且仅当 $\sum_{i=1}^{k-1} w_i x_i + w_k \le M$

剪去不含可行解的分枝

对左孩子(x_k=1)起约束函数作用，可剪去不含可行解的分枝；

对右孩子(x_k=0)不需用约束函数判断，一定可行。

6.5 0/1背包问题

- 目标函数 $\sum_{1\le i\le j} p_i x_i$
- 下界函数：
  - 变量L初值为0，遇到一个答案结点便计算该答案结点的收益值fp，且令L=max{L, fp}。
  - 则L中始终保存迄今为止已经搜索到的答案结点中收益的最大值，0/1背包的最优解值必定大于等于L，因此最优解值的下界估计值为L。
- 限界函数：
  - 状态空间树中任一结点X，若其上界函数值bp<最优解值的下界估计值变量L，则可断定X子树上不含最优答案结点，可以剪去以X为根的子树。
  - 剪去不含最优解的分枝。

中国科学院大学
University of Chinese Academy of Sciences

6.5 0/1背包问题

- 上界函数：
  - 当前位于状态空间树的结点X处，
  - cw为背包当前重量，
  - cp为当前已装入背包物品的总收益，
  - 用贪心法求解剩余载重和剩余物品构成的一般背包问题(物品编号k+1, k+2, ..., n-1, 载重M-cw)，最大收益为rp。
  - 则以X为根的子树上所有可能答案结点的目标函数值 $cp + \sum_{k+1 \le i \le n} p_i x_i$ 不可能超过bp=cp+rp，因此结点X的上界函数值为bp。

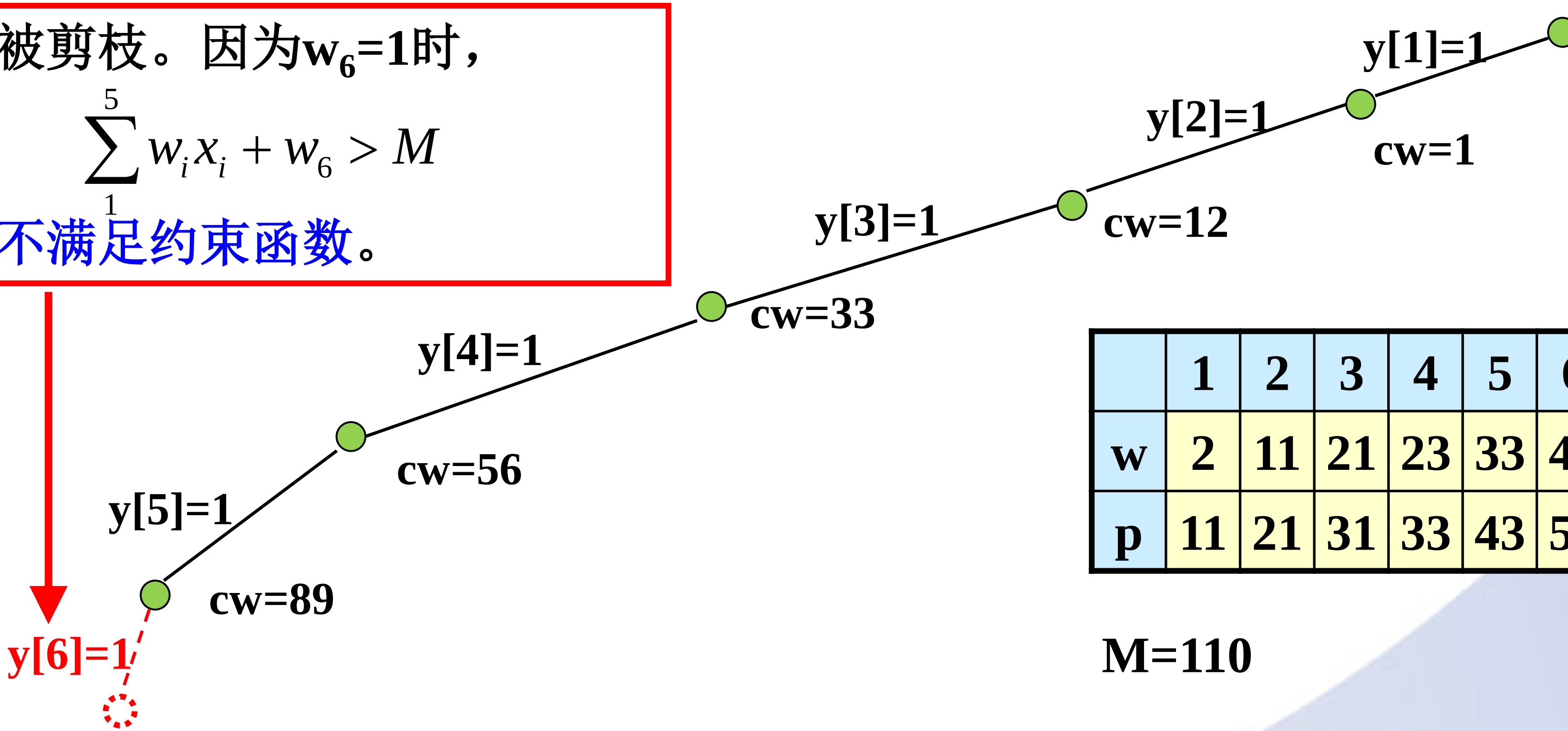
6.5 0/1背包问题

- 实例
  - 设有0/1背包n=8, M=110,
    $(w_1, w_2, ..., w_8) = (1, 11, 21, 23, 33, 43, 45, 55)$,
    $(p_1, p_2, ..., p_8) = (11, 21, 31, 33, 43, 53, 55, 65)$。
  - ——按p_i/w_i非增排列，即$p_i/w_i \geq p_{i+1}/w_{i+1}$

被剪枝。因为w_6=1时，

$$\sum_{1}^{5} w_i x_i + w_6 > M$$

不满足约束函数。

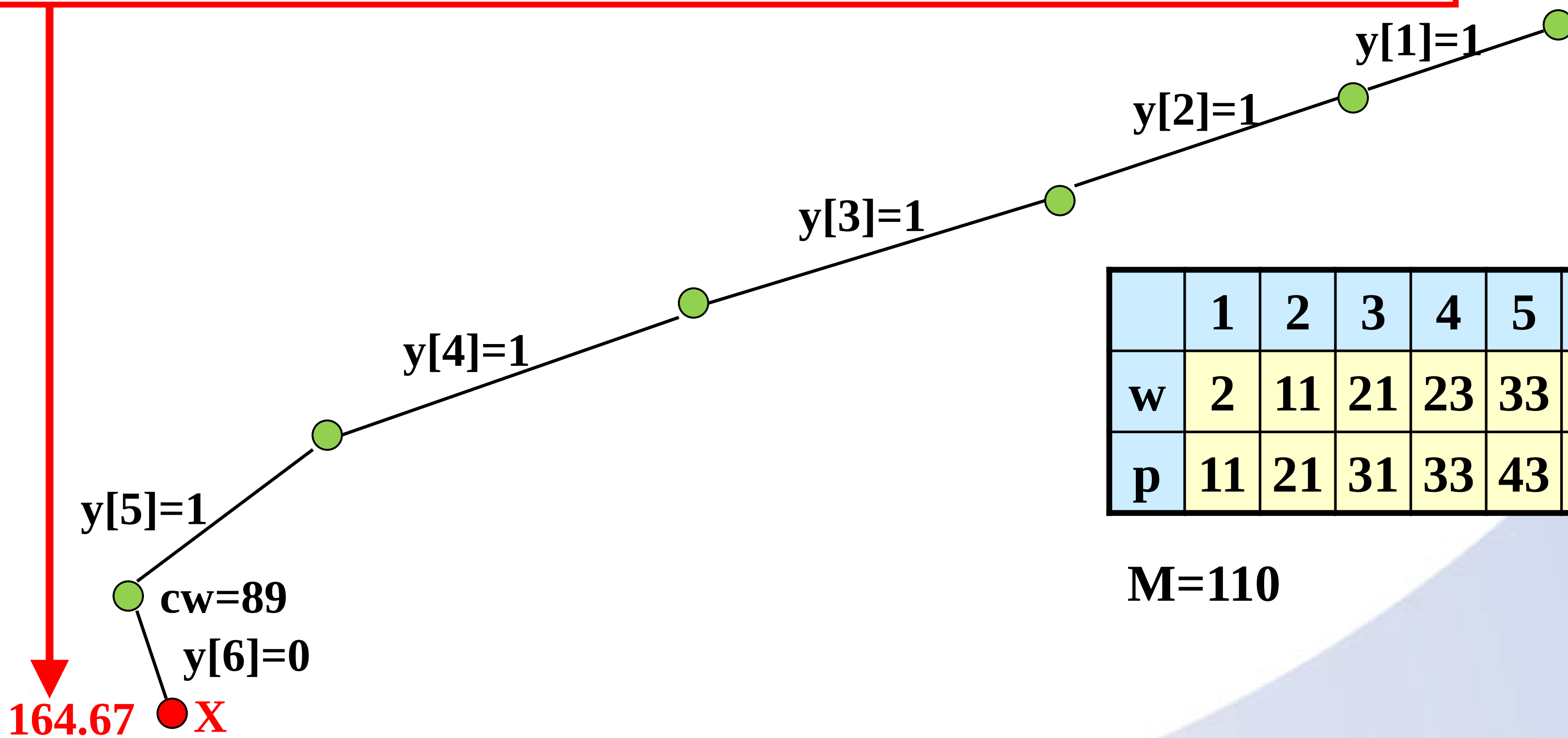


	1	2	3	4	5	6	7	8
w	2	11	21	23	33	43	45	55
p	11	21	31	33	43	53	55	65

M=110

$bp = cp+rp = p_1*1+p_2*1+p_3*1+p_4*1+p_5*1+p_6*0+ p_7/w_7*(110-89) = (11+21+31+33+43)+55/45*21 = 139+25.67 =164.67$

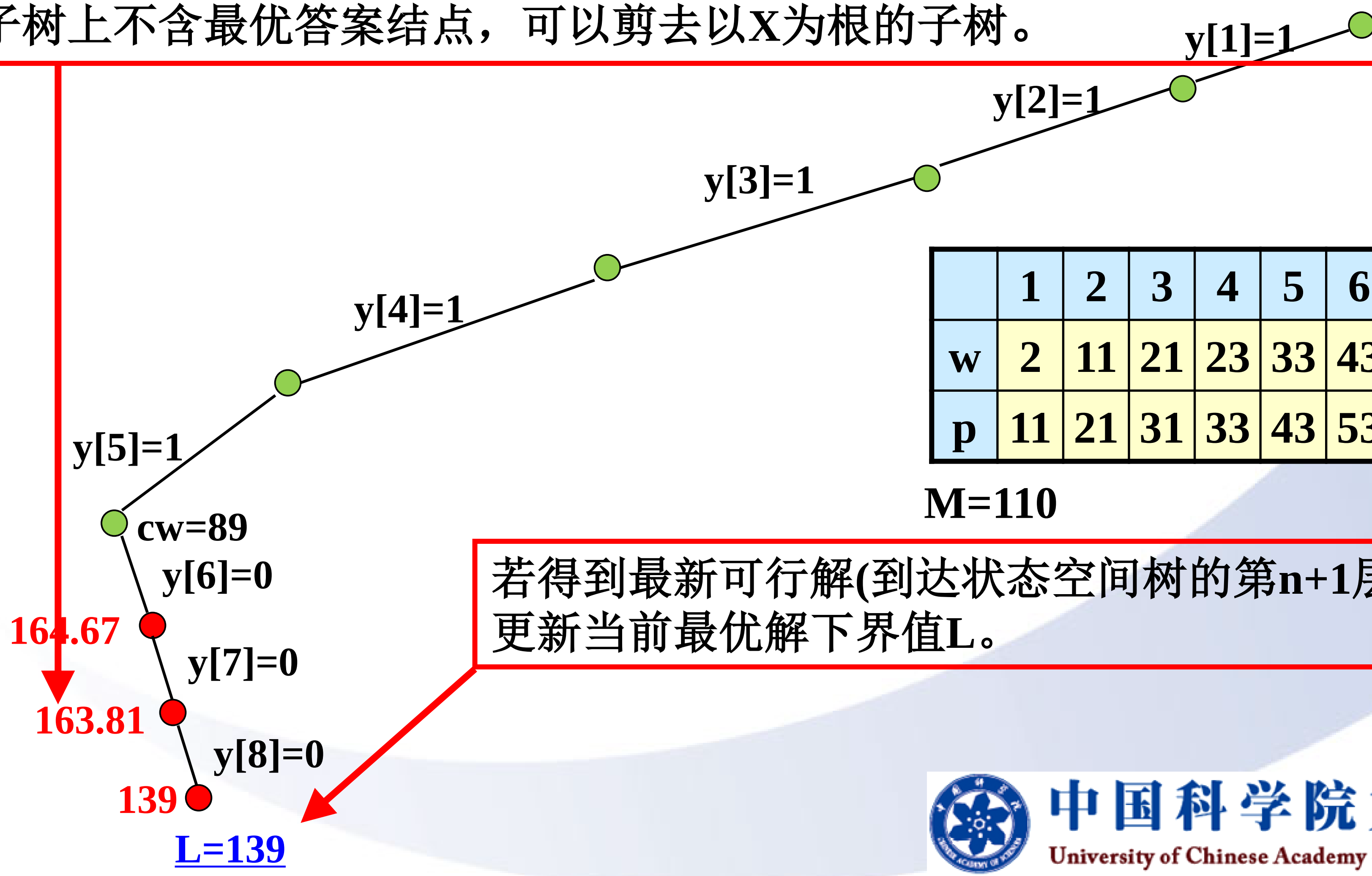
为当前结点X的上界函数值。若bp<当前最优解值下界估计值L，则可断定X子树上不含最优答案结点，可以剪去以X为根的子树。



	1	2	3	4	5	6	7	8
w	2	11	21	23	33	43	45	55
p	11	21	31	33	43	53	55	65

M=110

$bp = cp + rp = p_1*1+p_2*1+p_3*1+p_4*1+p_5*1+p_6*0+p_7*0+p_8/w_8*(110-89)$
$= (11+21+31+33+43)+65/55*21 = 139+24.81 = 163.81$
为当前结点X的上界函数值。若bp<当前最优解值下界估计值L，则可断定X子树上不含最优答案结点，可以剪去以X为根的子树。



	1	2	3	4	5	6	7	8
w	2	11	21	23	33	43	45	55
p	11	21	31	33	43	53	55	65

M=110

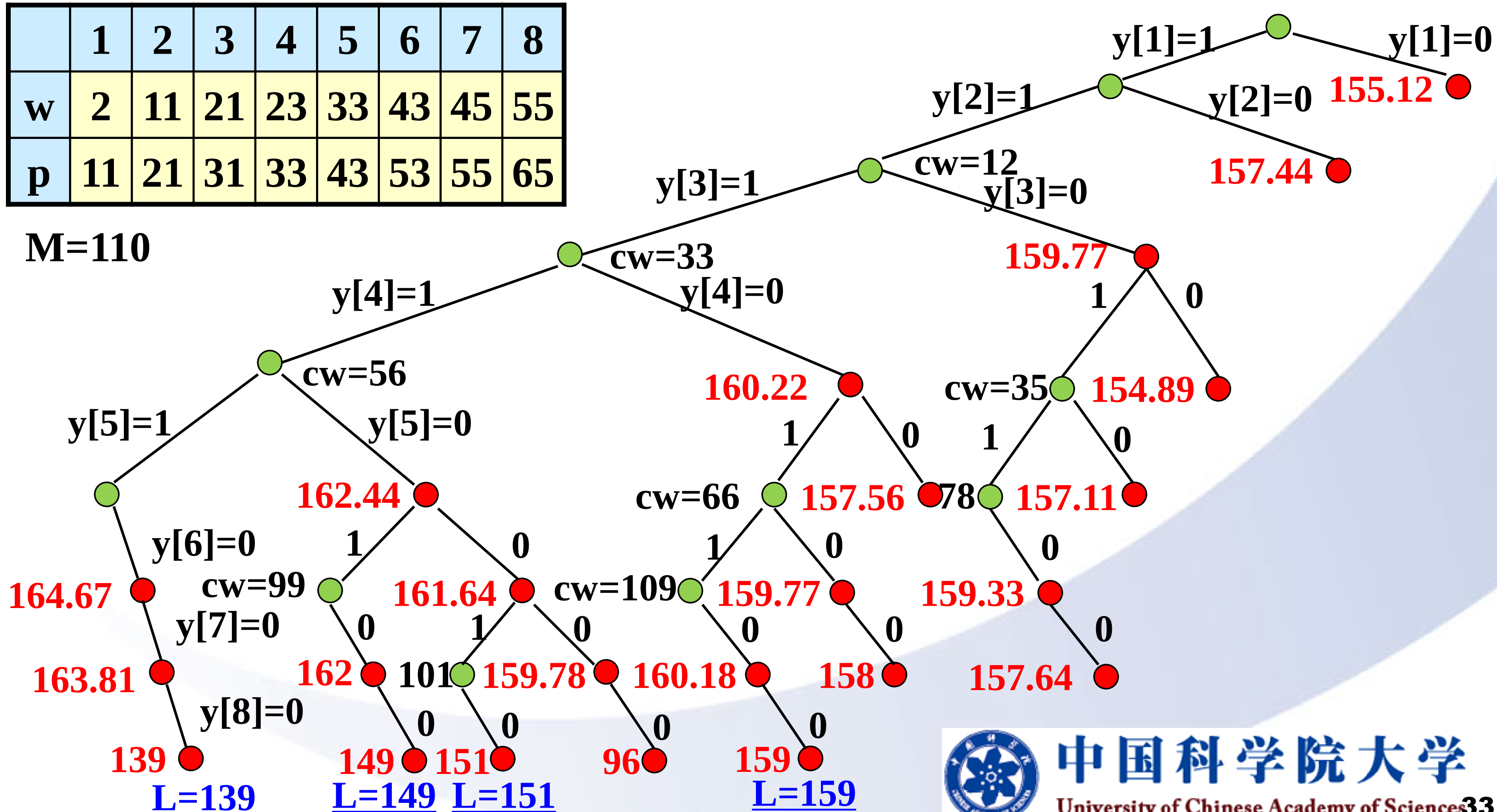
若得到最新可行解(到达状态空间树的第n+1层)，则更新当前最优解下界值L。

向左走：更新$\sum_{i=1}^{k-1} w_i x_i + w_k$值，bp值不变。用约束函数$\sum_{i=1}^{k-1} w_i x_i + w_k \le M$剪去不含可行解的分枝；

向右走：更新bp的值，$\sum_{i=1}^{k-1} w_i x_i$值不变。用限界函数bp≤L剪去不含最优解的分枝。

	1	2	3	4	5	6	7	8
w	2	11	21	23	33	43	45	55
p	11	21	31	33	43	53	55	65

M=110



算法6.11 限界函数

```
procedure BOUND(p, w, k, M)
 //p为当前效益总量；w为当前背包重量；k为上次去掉的物品；
   M为背包容量；返回一个新效益值//
    global n, P(1:n), W(1:n)
    integer k, i; real b, c, p, w, M
    b ← p; c ← w
    for i ← k+1 to n do
         c ← c+W(i)
         if c<M
         then b ← b+P(i)
         else return (b+(1-(c-M)/W(i))*P(i))
         endif
     repeat
     return (b)
end BOUND
```

作业-课后练习22

- 问题描述
  - 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。
- 要求
  - 作业提交到课程网站上

作业-课后练习23

- 问题描述
  - 求下面的0-1背包问题
  - ① $N = 5, M=12, (p_1, p_2, ..., p_5) = (10, 15, 6, 8, 4), (w_1, w_2, ..., w_5) = (4, 6, 3, 4, 2)$。
  - ② $N = 5, M=15, (w_1, w_2, ..., w_5) = (p_1, p_2, ..., p_5) = (4, 4, 5, 8, 9)$。
- 要求
  - 作业提交到课程网站上

END