吉林大学

第八章 分支限界法

目录

8.1 一般方法

- 方法适用的问题特点
- 分支限界法的基本思想
- 抽象化描述
- 分支限界法的不同检索方式
- 理解FIFO-分支限界法

方法适用的问题特点

- 与回溯法类似，分支限界法同样适用于求解组合数较大的问题，特别是组合优化问题(求最优解)。
- 分支限界法中，解的形式化表达、显示约束条件、隐式约束条件、解空间和解空间树等概念均与回溯法相同
- 两者主要区别在于E-结点(即扩展结点)处理方式不同

清华大学出版社出版的屈婉玲等编著的《算法设计与分析》中认为：“分支限界是回溯算法的变种”

分支限界法的基本思想

- 针对问题定义解空间树结构：元组、显式约束条件、隐式约束条件。
- 检验问题满足多米诺性质。
- 假设当前寻找一个答案结点，按下列方式搜索解空间树：
  - 如果根结点T是答案结点，输出T，操作结束；否则令T是当前扩展结点E。
  - 生成E的所有儿子结点，判断每个儿子结点X：
    - 如果X是答案结点，输出从X到T的路径，操作结束；
    - 如果X满足限界函数B，则将X添加到活结点表中；否则舍弃X。
  - 从活结点表中选择下一个结点成为新的E-结点，重复上述操作。如果活结点表为空，则算法以失败结束。

限界函数剪枝作用：避免生成那些不包含答案结点的子树。

算法8.1 分支限界法的抽象化描述

找一个答案结点

```
procedure BB(T)
If T是答案结点 then 输出T; return endif
E←T
将活结点表初始化为空
loop
      for E的每个儿子X do
            if X是答案结点 then 输出从X到T的那条路径; return; endif
            if B(X) then call ADD(X);PARENT(X)←E endif
      repeat
      if 不再有活结点 then print("no answer node"); return; endif
      call LEAST(E)
repeat
end BANDB
```

- ADD(X):将X添加到活结点表中
- LEAST(E):从活结点表中选中一个结点赋值给E，并从表中删除该结点

思考：如果想获得所有可行解，算法怎样设计？

分支限界法的不同检索方式

- 根据活结点检索次序，分支限界策略可以分为：
  - FIFO-检索：活结点采用先进先出的方式(队列)出活结点表。
  - LIFO-检索：活结点采用后进先出的方式(栈)出活结点表。
  - LC-检索：定义成本估计函数ĉ，令ĉ最小的活结点出活结点表。
- 若BB算法中ADD和LEAST遵循：
  - FIFO-检索，算法即为FIFO-分支限界；
  - LIFO-检索，算法即为LIFO-分支限界；
  - LC-检索，算法即为LC-分支限界。

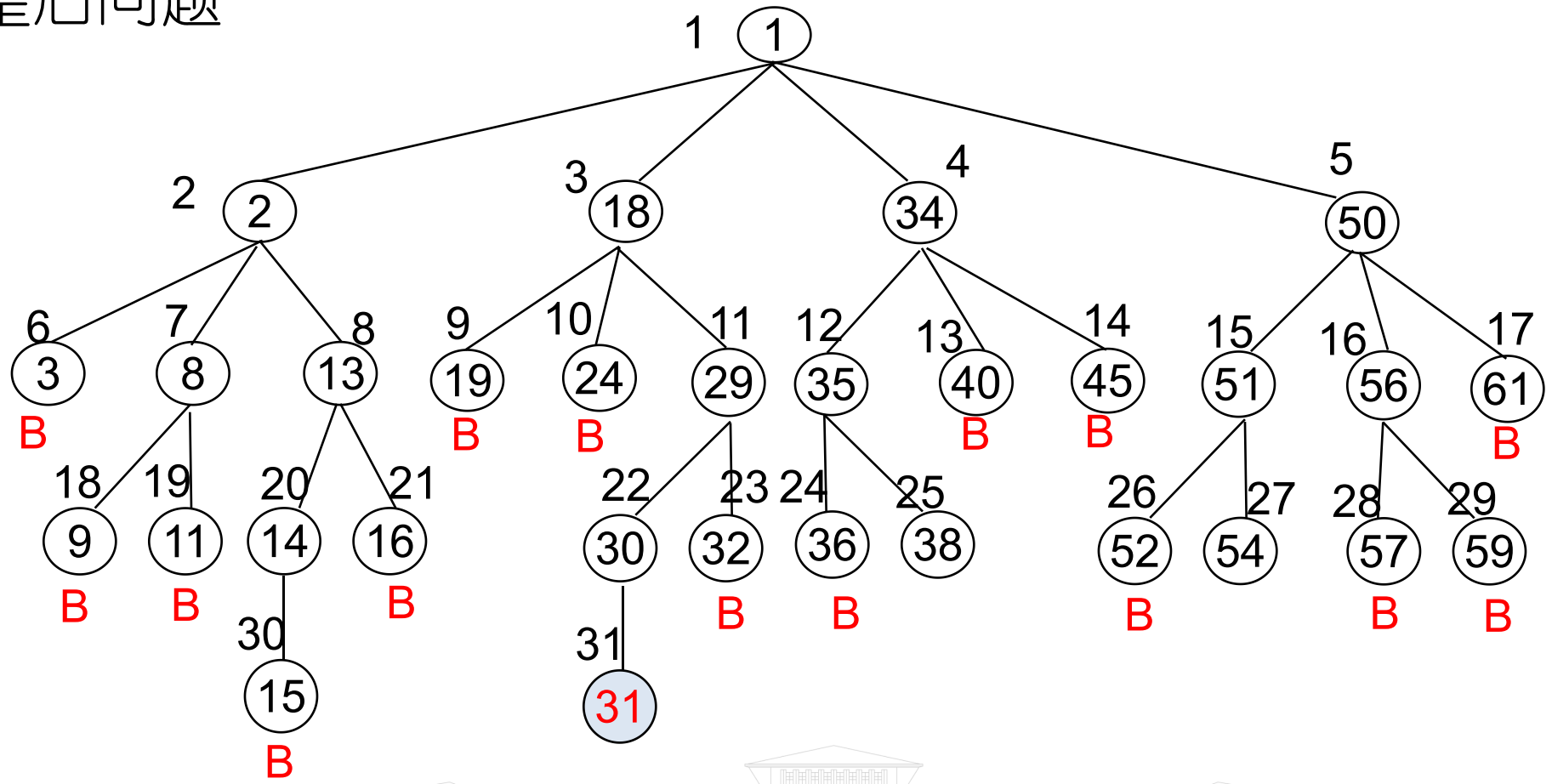
类似于BSF-检索

类似于D-检索

限界指限界函数

理解FIFO-分支限界法

- 4-皇后问题



FIFO-分支限界法一共处理了31个结点，回溯法一共处理了16个结点。

8.2 LC-检索

- LC-检索的优点
- 成本函数c的量化方法
- 区分状态空间树中结点X
- 成本函数c定义
- 成本估计函数ĉ定义
- LC-检索总结

LC-检索的优点

- 在LIFO和FIFO-分支限界法中，对下一个E-结点的选择是死板的、盲目的，对于可能快速检索到答案结点的结点没有给出任何优先权。
- 理想状态下，对活结点表使用一个“有智力的”成本函数c来选取下一个E-结点，从而加快到达答案结点的检索速度。

成本函数c的量化方法

- 要给那些可能导致答案结点的活结点赋以优先次序。
- c(X)的定义：
  - 方法1：基于X在生成一个答案结点之前需要生成的结点数
    - 寻找生成最小数目的答案结点
  - 方法2：基于距离X最近的那个答案结点的路径长度
    - 寻找路径长度最短的答案结点

选择后一种度量来考虑问题

区分状态空间树中结点X

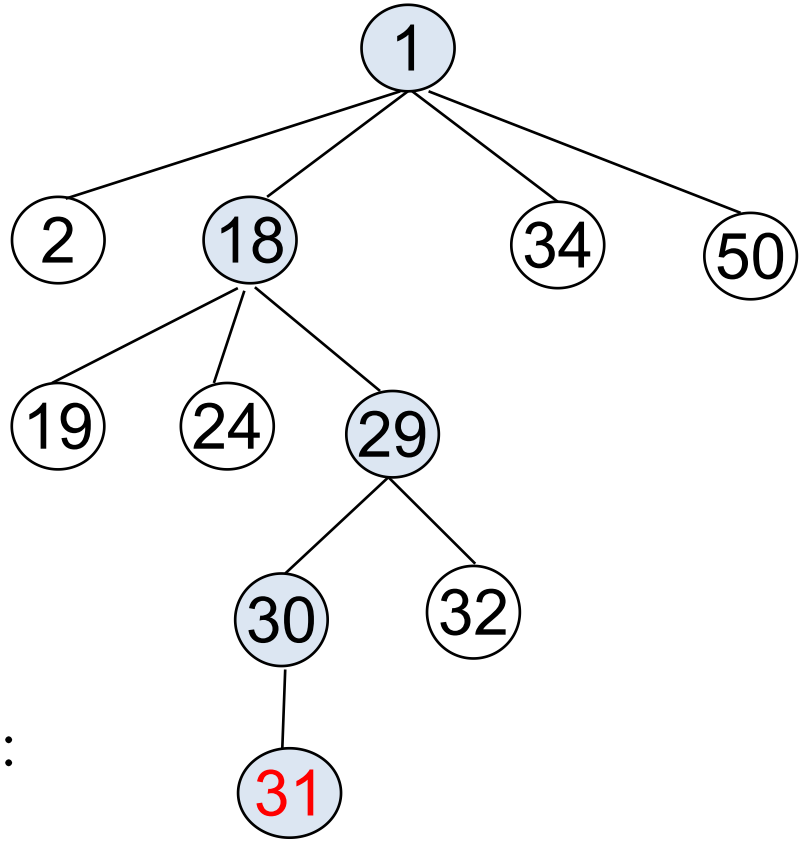
- 对于一个状态空间树里面的任何一个结点X，有如下几种可能：
  - X是答案结点；
  - X不是答案结点，但子树X中包含答案结点；
  - X不是答案结点，且子树X不包含任何答案结点；
- 从上帝视角定义一个结点成本函数c(X)

成本函数c定义

- 如果X是答案结点
  - c(X)=状态空间树的根结点到X的成本(即所用的代价，可以是路径长度、计算复杂度等)
- 如果X不是答案结点且子树X不包含任何答案结点
  - 则c(X)=∞
- 如果X不是答案结点，但子树X中包含答案结点
  - c(X)=子树X中具有最小成本的答案结点的成本

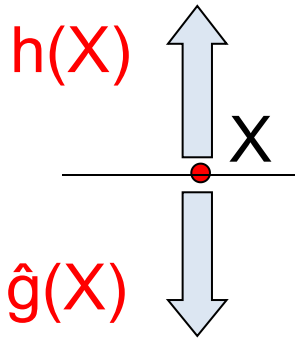
成本函数c的例子

- 用路径长度来表达结点成本
- X是答案结点：
  - c(31)=4（根结点到31的成本）
- X不是答案结点，但子树X中包含答案结点：
  - c(1)=c(18)=c(29)=c(30)
    =c(31)
    =4
- X不是答案结点且子树X不包含任何答案结点：
  - c(2) =∞, c(19) =∞, ……



- 成本函数c的问题：要得到函数c很困难
  - c(X)是结点X的真实成本函数。为求出该值，需要生成检索树。
  - c的计算工作量与原问题具有相同的复杂度。

成本估计函数ĉ定义



- 基于c，定义一个易于计算的成本估计函数ĉ
  - 考虑结点X到一个答案结点的估计成本ĝ(X);
  - 考虑根结点到结点X的成本h(X);
  - 为调整h和ĝ在成本估计函数ĉ中的影响比例，定义一个非负函数f
  - 成本估计函数：

$$\hat{c}(X)=f(h(X))+\hat{g}(X)$$

算法利用ĉ对活结点表进行检索时，优先选择更靠近答案结点但又离根结点较近的结点。

LC-检索总结

找到这样一个易于计算的c是很难的

- T是一棵状态空间树
- c是T中结点的成本函数
- c(X)是X为根的子树中答案结点的最小成本。
- c(T)是T中最小成本答案结点的成本。

- 用估计函数ĉ来代替c，规定：
  - 易于计算
  - ĉ(X)≤c(X)
  - 如果X是一个答案结点或者是一个叶结点，则c(X)=ĉ(X)

思考：ĉ(X) ≤ c(X)会为求最优解带来什么帮助？

LC-检索总结

$\hat{c}(X)=f(h(X))+\hat{g}(X)$

- LC-检索(Least Cost search)：选取成本估计函数$\hat{c}$的值最小的活结点作为下一个E-结点。
- LC-分支限界检索：伴有限界函数的LC-检索
- BFS-检索：$f(h(X))$=根到结点X的路径长度，$\hat{g}(X)=0$
- D-检索：$f(h(X))=0$，且每当Y是X的一个儿子时总有$\hat{g}(X)\geq\hat{g}(Y)$

BFS和D检索是LC-检索的特殊情况

8.3 15-谜问题

- 问题描述
- 状态空间树
- 宽度优先FIFO-检索
- 深度优先检索
- LC-检索

问题描述

- 在一个分成16格的方形棋盘上放有15块编了号码的牌，如(a)所示，要求通过一系列合法的移动转换成(b)所示那样的目标排列。

1	3	4	15
2		5	12
7	6	11	14
8	9	10	13

(a)

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

(b)

若当前牌邻接有空位置，则可将牌移动到空位置。

状态空间树

- 问题状态
  - 棋牌布局状态
  - 初始排列称为初始状态
  - 目标排列称为目标状态
- 状态空间树
  - 棋牌每移动一次，就会产生一个新的布局状态
  - 由所有可从初始状态经过一系列合法移动到达的状态构成
  - 儿子结点是当前结点通过一次合法的移动可以到达的布局状态。

初始状态满足某些条件时，才能达到目标状态(b)

函数定义

- 对棋盘的方格位置编上1~16的号码，编号顺序如图(b)所示，空格位是位置16
- 设POSITION(i)是棋牌i在初始状态时的位置号，1≤i<16，POSITION(16)表示空格的位置
- 设LESS(i)是牌面上j<i且POSITION(j)>POSITION(i)的j的数目，即反序的数目

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

(b)

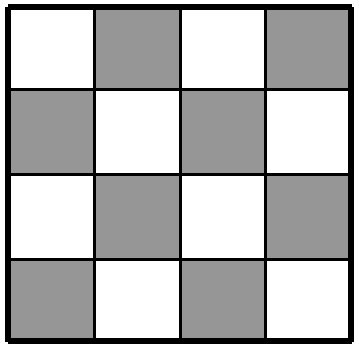
1	3	4	15
2		5	12
7	6	11	14
8	9	10	13

(a)

LESS(1)=0; {}
LESS(4)=1; {2}
LESS(12)=6; {7,6,11,8,9,10}

判定定理

在初始状态下，如果空格在(c)的阴影位置中，则令X=1；否则令X=0。

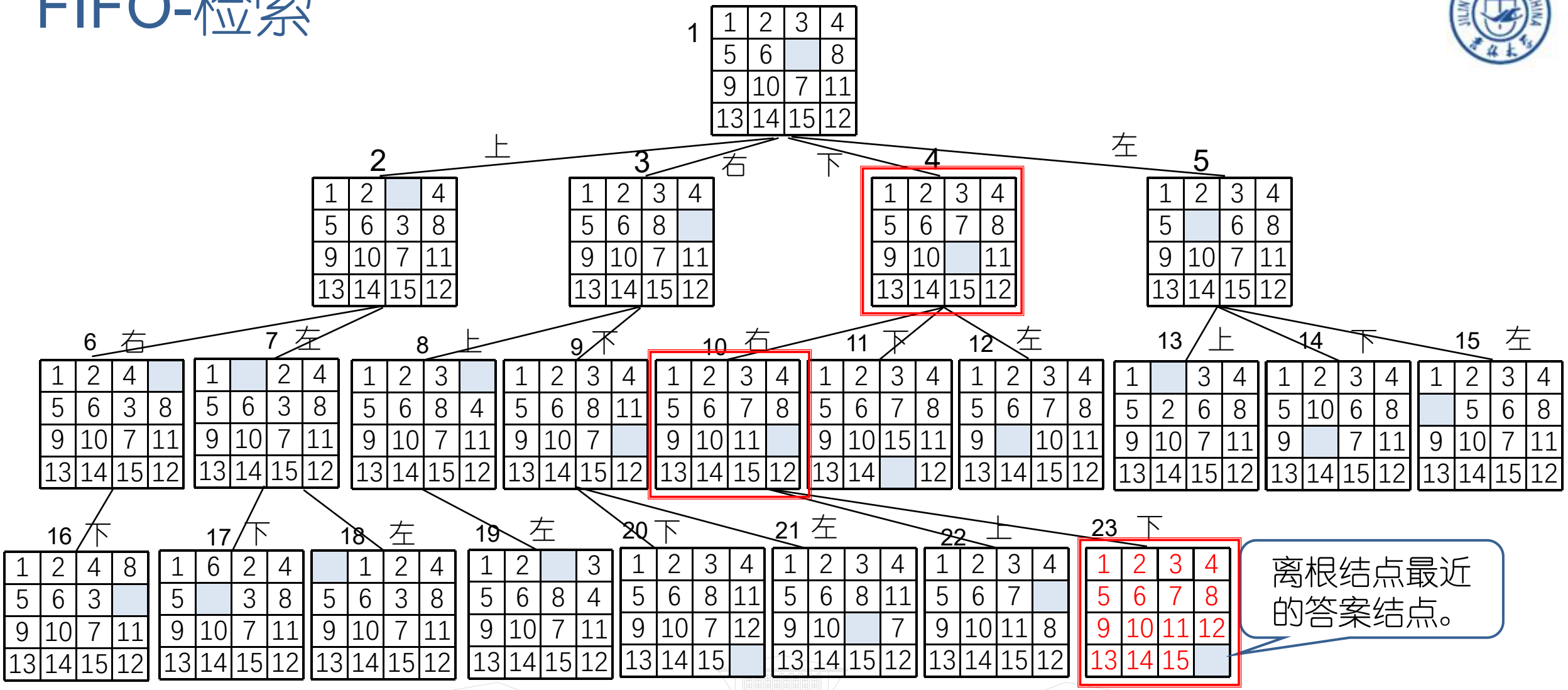


(c)

初始状态判定定理：
当且仅当初始状态的$\sum_{1\le i\le 16}$LESS(i)+X是偶数时，图(b)所示的目标状态可由此状态到达

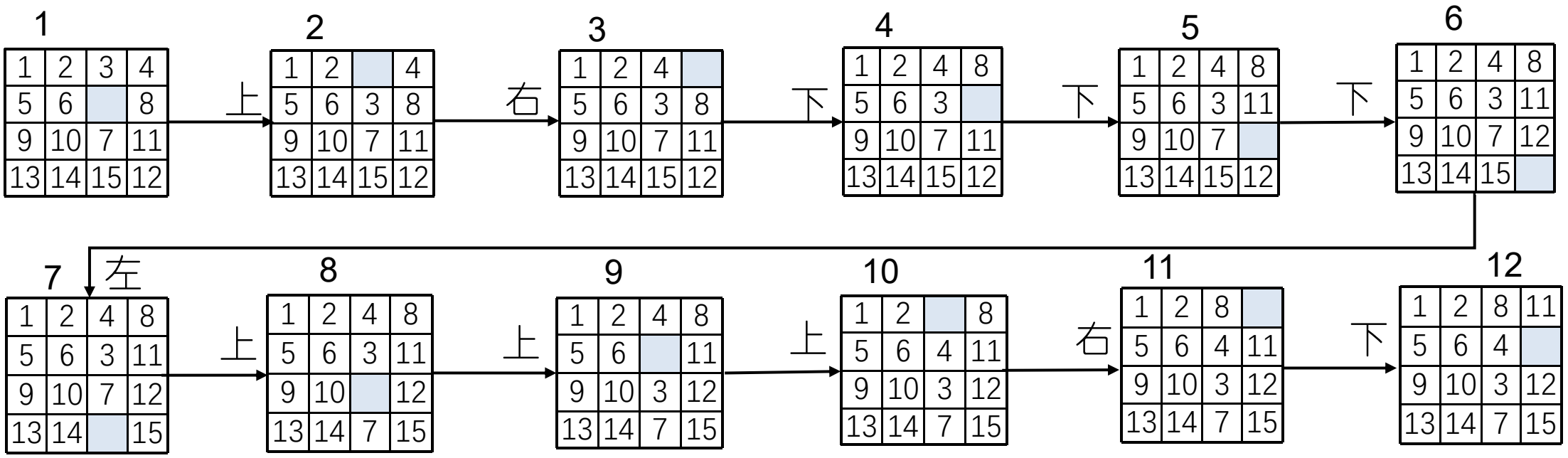
由于移动牌与移动空格等效，因此状态空间树中，边表示为空格的一次合法移动，按上、右、下、左的顺序进行。

FIFO-检索



能找到离根最近的答案结点，不管开局如何（不关心问题的具体实例），总是按千篇一律的顺序移动。

深度优先检索



不管开局如何（不关心问题的具体实例），总是采取由根开始的最左路径，呆板而盲目。搜索过程中有可能远离目标。

LC-检索

- 定义c(X)：
  - 将根结点到最近目标结点路径上的每个结点的成本设置为这条路径的长度。
  - 其余结点成本赋值为∞。

找到离根结点最近的答案结点。

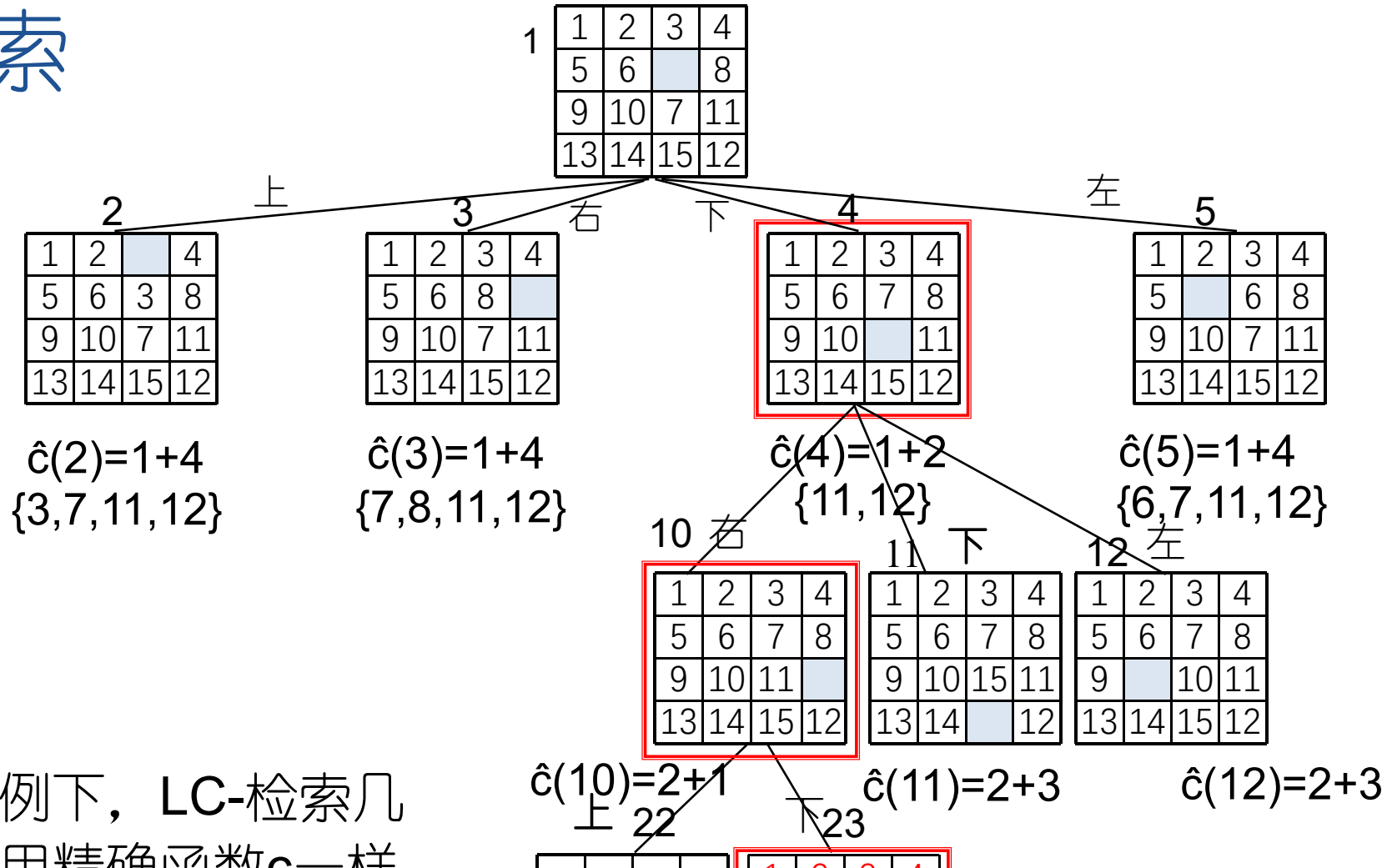
- LC-分支限界：
  - 首先将根结点作为E-结点，将成本为∞的子结点统统杀掉，只保留与根结点成本相同的子结点，并使其成为下一个E-结点。
- 定义ĉ(X)：ĉ(X)=f(X)+ĝ(X)
  - f(X)是由根到结点X路径的长度；
  - ĝ(X)是以X为根的子树中由X到目标状态的一条最短路径长度的估计值。

ĝ(X)=不在其目标位置的非空白棋牌数目

最小移动数

ĉ(X)是c(X)的下界

LC-检索



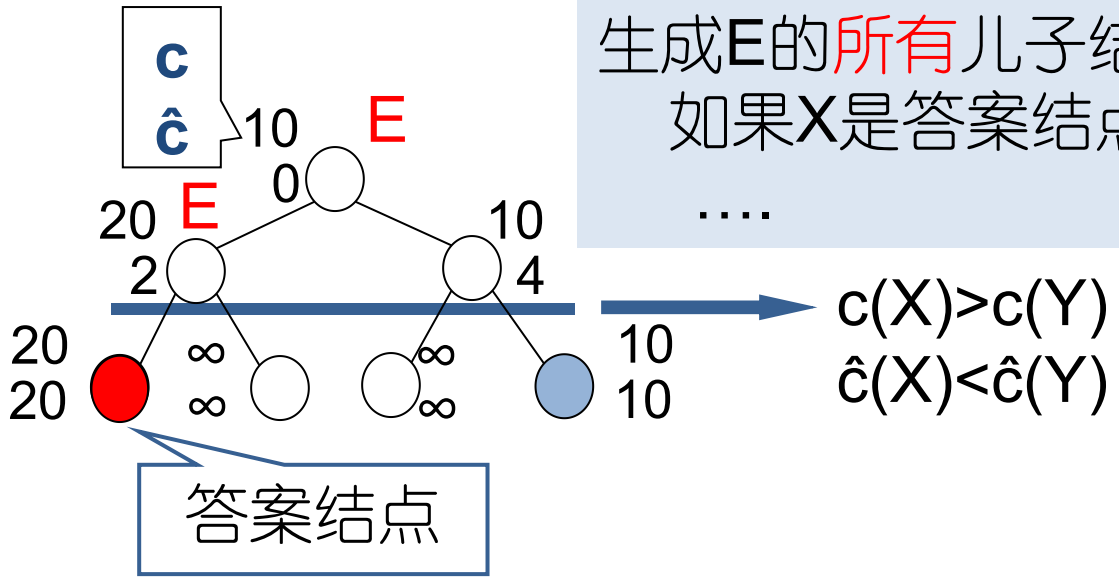
当前实例下，LC-检索几乎和使用精确函数c一样有效，LC-检索的选择性比很多检索方法强很多。

8.4 求最小成本的分支限界法

- ĉ的特性
- 求最小成本的LC-分支限界算法
- 加速寻找最小成本
- 最小成本上界U
- 基于ĉ和U求最小成本的分支限界法

ĉ的特性

- 基于LC-检索的算法BB是否一定能找到具有最小成本的答案结点G呢？



…..

生成E的所有儿子结点，判断每个子结点X：

如果X是答案结点，输出从X到T的路径，操作结束

….

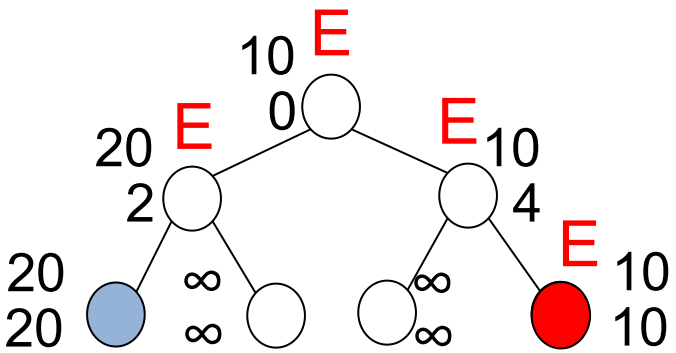
c(X)>c(Y)

ĉ(X)<ĉ(Y)

- 对于每一对结点X、Y，当且仅当c(X)<c(Y)时有ĉ(X)< ĉ(Y)。那么在使ĉ作为c的估计值时，算法BB到达一个最小的成本答案结点且终止。

ĉ的特性

- 算法BB基于LC-检索寻找具有最小成本的答案结点，则ĉ要满足：
  - 易于计算
  - 对于每一个结点X， ĉ(X) ≤ c(X)
  - 对于答案结点X，有ĉ(X) = c(X)
  - c(X)<c(Y)时，有ĉ(X)< ĉ(Y)　难以实现
- 一般只能找到满足前三项要求的ĉ



改进算法：

…..
如果E是答案结点，输出从E到T的路径，操作结束；
生成E的所有儿子结点：
….

思考：如何证明改进后找到的答案结点一定是最小成本

算法8.2 求最小成本的LC-分支限界算法

```
procedure LC(T, ĉ) //为找出最小成本答案结点
E←T, 将活结点表初始化为空
loop
    if E是答案结点 then 输出从E到T的那条路径; return; endif
    for E的每个儿子X do
        if B(X) then call ADD(X); PARENT(X)←E; endif
    repeat
    if 不再有活结点 then print("no answer node"); return; endif
    call LEAST(E)
repeat
end LC
```

选择ĉ最小的

- 对于活结点表中的每一个结点L，一定有ĉ(E)≤ĉ(L)。由ĉ定义知，E是答案结点时c(E)=ĉ(E)，则c(E)=ĉ(E)≤ĉ(L) ≤c(L)
- 因此，算法LC选中成本最小的答案结点

加速寻找最小成本

- 分支限界法找最小成本的答案结点
  - 基本思想：生成当前E-结点的所有儿子结点之后，再从活结点表中选择一个新的结点成为E-结点。
  - 智能优化：设置一个成本估计函数$\hat{c}(X)$，给出经由结点X到达答案结点的成本下界，$\hat{c}(X) \le c(X)$。
  - 进一步加速：设置一个最小成本上界U，U也可能恰好就是一个成本值。

注意：在寻找最小成本时，满足约束条件的解结点并不一定是答案结点，同时还要使目标函数取极小值。

基于ĉ和U求最小成本的分支限界法

- 1) 改变极大化问题的目标函数，将问题转化为极小化问题
- 2) 把目标函数作为成本函数c
- 3) 约束条件作为限界函数B
- 4) 问题转化为寻找状态空间树中最小成本的答案结点
- 5) 设计成本估计函数ĉ(X)，ĉ(X)≤c(X)
- 6) 还可以设计最小成本的上界U，c(X)≤U
- 7) 基于ĉ(X)和U进行分支限界搜索

最小成本上界U

- U的取值：
  - 初始值≥最小成本答案结点的成本，通过启发性方法得到，或∞
  - 随着结点的访问不断改小
- U的使用:
  - 若U是一个成本值, ĉ(X)>=U的所有活结点X可以被杀死
  - 若U是一个上界值, ĉ(X)>U的所有活结点X可以被杀死;
    - 情况1：还没有找到任何一个解结点
    - 情况2：找到一个解结点，但是它的成本值大于上界值，说明它的子孙中还有成本更小的解结点Y，U取Y的上界值。

当前解结点非叶结点，
常见于k-元组解空间树

合并U的使用情景

- U是当前遍历过的部分解空间树的上界估计值
- u(X)是结点X的上界估计函数
- 定义一个足够小的正常数ε
  - 满足对任意结点X,Y，如果u(X)<u(Y)，u(X)<u(X)+ε<u(Y)
- 判断结点X满足以下条件时修改U值
  - 如果X是解结点且成本<U，U←min(X的成本,u(X)+ε)
  - 如果X不是解结点且u(X)+ε<U，U←u(X)+ε
- 检验结点X满足以下条件时被杀死
  - ĉ(X)>=U时，X被杀死

FIFO算法思想

假定状态空间树T至少包含一个解结点，不可行结点的估计值ĉ(X)=∞；可行结点的估计值 ĉ(X)≤c(X)≤u(X) 。

- 步骤1：初始化
  - 令根结点T是当前E-结点；
  - 对U和答案解ans赋初值：如果T是解结点， U←min(T的成本,u(T)+ε)，ans←T；否则U←u(T)+ε，ans←0；
  - 活结点队列置为空；
- 步骤2：对E的每个子结点X进行检验，添加到队列中，并修改U值：
  - 如果ĉ(X)<U，将X加入到队列中
  - 对X进一步判断：如果X是解结点且成本<U，U←min(X的成本,u(X)+ ε)，ans←X；否则，如果u(X)+ε<U，U←u(X)+ε；
- 若队列为空，打印当前ans；否则从队列中获取下一个活结点E：若ĉ(E)<U,转到步骤2，否则转到步骤3。

算法8.3 找最小成本的FIFO-分支限界算法

```
procedure FIFOBB(T,ĉ,u,ε,cost)
   E←T; PARENT(E)←0;
   if T是解结点 then  U←min(cost(T),u(T)+ε),ans←T ; else U←u(T)+ε, ans←0 ;endif
   将队列初始化为空
   loop
     for E的每个儿子X do
         if ĉ(X)<U then call ADDQ(X), PARENT(X)←E;
            case X是解结点 and cost(X) <U: U←min(cost(X),u(X)+ε), ans←X ;
            case u(X)+ε<U: U←u(X)+ε    endcase endif  repeat
     loop
         if 队列为空 then print ('least cost = ', U); 输出从ans到T的那条路径; return; endif
         call DELETEQ(E);
         if ĉ(E)<U then exit; repeat
   repeat
end FIFOBB
```

思考：请验证算法的正确性

算法8.4 找最小成本的LC-分支限界法

```
procedure LCBB(T,ĉ,u,ε,cost)
    E←T; PARENT(E)←0;
    if T是解结点 then  U←min(cost(T),u(T)+ε),ans←T;else U←u(T)+ε, ans←0 ;endif
    将活结点表初始化为空
    loop
        for E的每个儿子X do
            if ĉ(X)<U then call ADD(X); PARENT(X)←E;
                case X是解结点 and cost(X)<U: U←min(cost(X),u(X)+ε), ans←X;
                case u(X)+ε<U: U←u(X)+ε  endcase endif repeat
        if 不再有活结点 or 下一个E-结点有ĉ≥U
            then print ('least cost = ' ,U); 输出从ans到T的那条路径; return; endif
        call LEAST(E);
    repeat
end FIFOBB
```

函数ADD：加一个结点到min-堆中；
函数LEAST：从min-堆中删去堆顶结点

8.5 带有期限的作业调度问题

- 问题描述
- 一个问题实例
- 限界函数B
- 成本下界函数ĉ
- 成本上界U
- FIFO-分支限界法实例运行

问题描述

- 假设有n个作业和一台处理机，每个作业i由一个三元组(p_i,d_i,t_i)表示，表示作业需要t_i个时间处理完毕，如果在期限d_i之前没有完成则要交付p_i的罚款。
- 问题目标：从这n个作业中选取一个子集合J，使J中作业都能在相应的期限内完成，而不在J中的作业罚款总数最小。

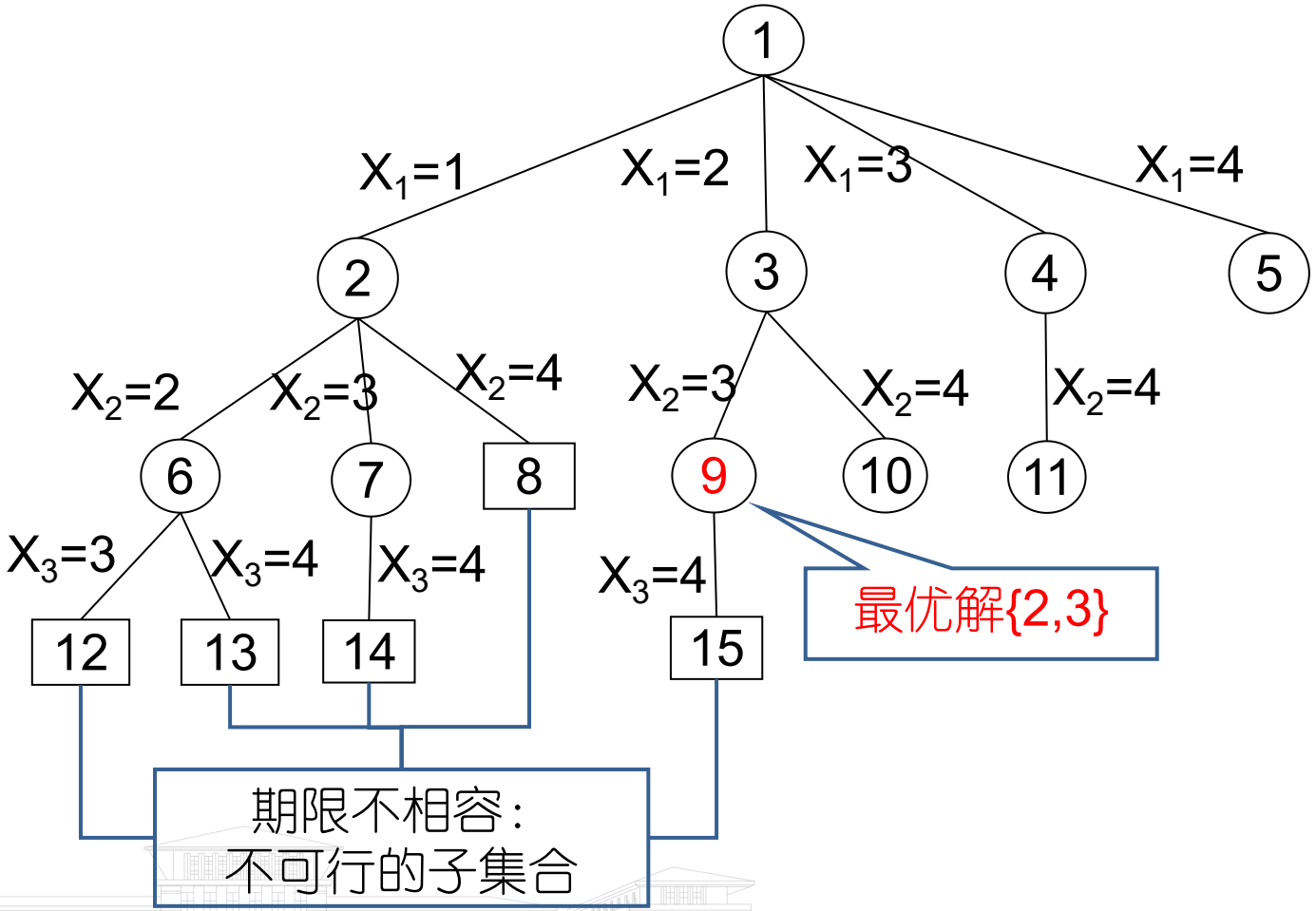
一个问题实例

- n=4, $(p_1,d_1,t_1)=(5,1,1)$; $(p_2,d_2,t_2)=(10,3,2)$; $(p_3,d_3,t_3)=(6,2,1)$;$(p_4,d_4,t_4)=(3,1,1)$;
- 解空间的表示方法
  - 不定长的k-元组$(X_1,..X_k)$, $k\le n$
  - 显示约束条件：元组X_j表示选中的作业下标, $x_i\le x_{i+1}$, $1\le i<k$
  - 隐式约束条件：作业能在期限前完成
  - 目标函数：未选中的作业罚款总数最小
- 状态空间树
  - 共计$2^n=16$个结点
  - 圆形结点表示答案结点(可行解)
  - 方形结点表示不可行结点

限界函数B

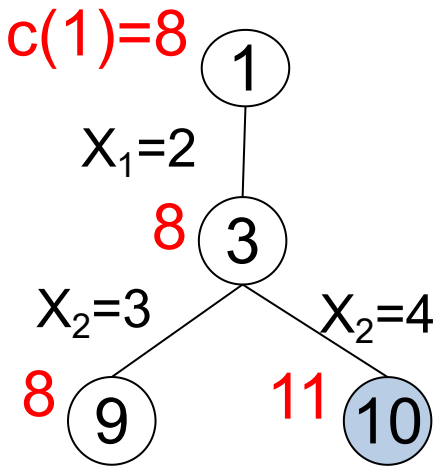
- n=4
- k-元组表示状态空间树
- FIFO-分支限界法
- 作业实例
  - $(p_1,d_1,t_1)=(5,1,1)$
  - $(p_2,d_2,t_2)=(10,3,2)$
  - $(p_3,d_3,t_3)=(6,2,1)$
  - $(p_4,d_4,t_4)=(3,1,1)$

10个活结点通过检验



成本下界函数ĉ

- 定义下界函数ĉ(X)，使得ĉ(X)≤c(X)：
  - 设Sx是根结点到达结点X时选中的作业集合
  - 令m=max{i|i∈Sx}
  - $\hat{c}(X)=\sum_{i<m,\ i\notin Sx} p_i$，若X是可行解
  - ĉ(X)=∞，若X是不可行解

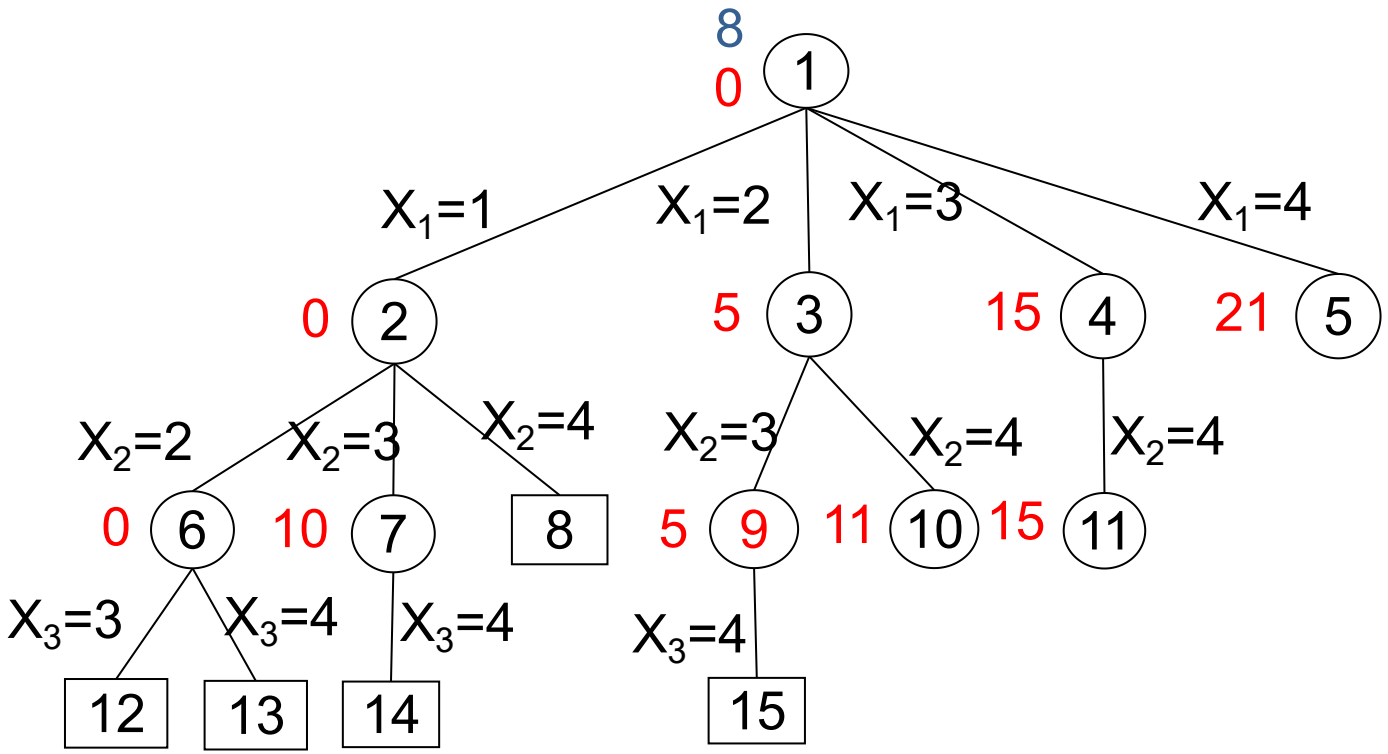


ĉ(1)=0
ĉ(3)=5 {1}
ĉ(9)=5 {1}
ĉ(10)=11 {1,3}

对于答案结点x，当是叶结点时，ĉ(x)=c(x)；否则，ĉ(x)≠c(x)。因此，算法LC并不能求出最优解。

每个答案结点的ĉ值

- 作业实例
  - $(p_1,d_1,t_1)=(5,1,1)$
  - $(p_2,d_2,t_2)=(10,3,2)$
  - $(p_3,d_3,t_3)=(6,2,1)$
  - $(p_4,d_4,t_4)=(3,1,1)$



成本上界U

- 成本估计函数ĉ(X)
  - $\hat{c}(X)=\sum_{i<m,\ i\notin Sx} p_i$，若X是可行解；
  - ĉ(X)=∞，若X不可行解。
- 最小成本上界$u(X)=\sum_{i\notin Sx} p_i$

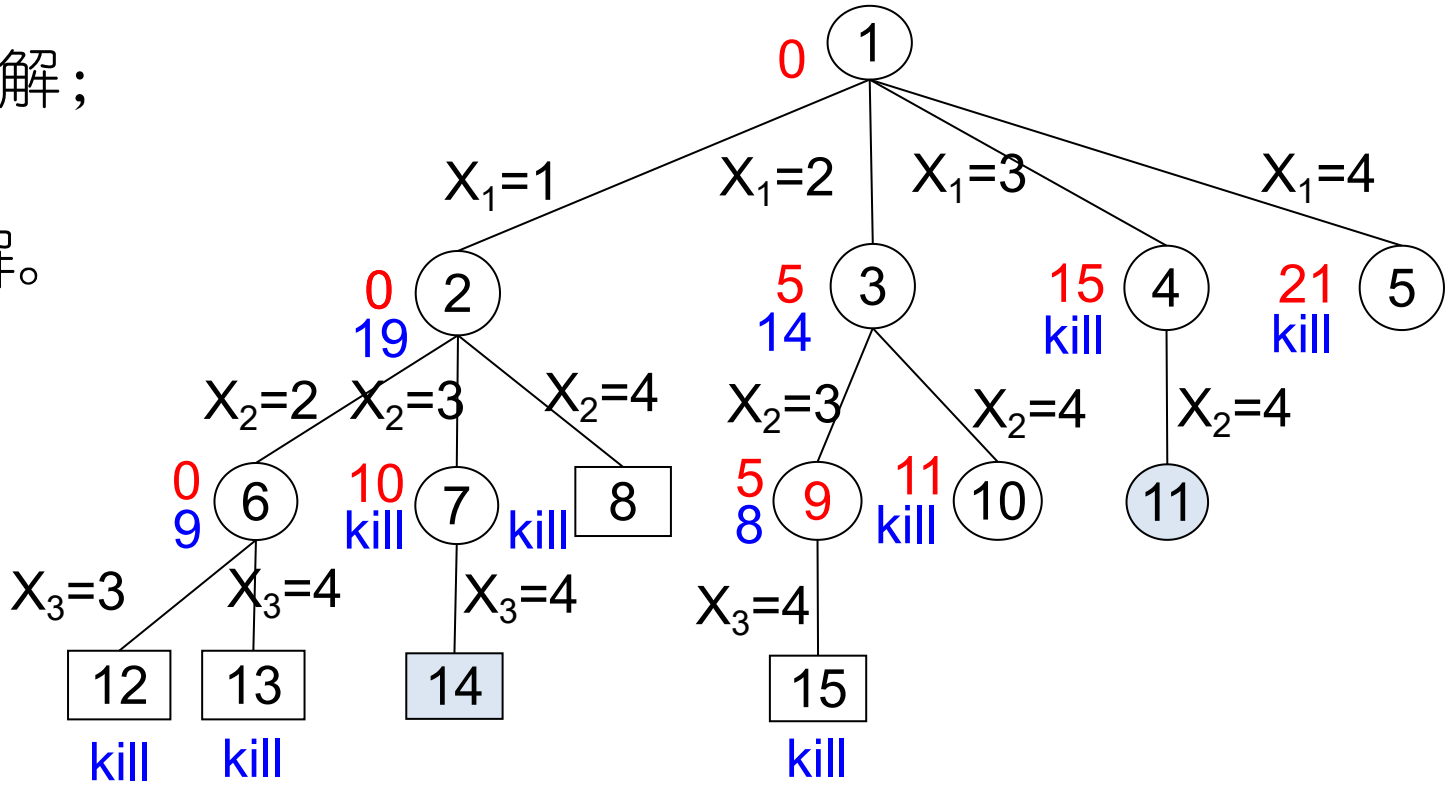
U=min{u(X)}

19

14

9

8



5个活结点通过检验

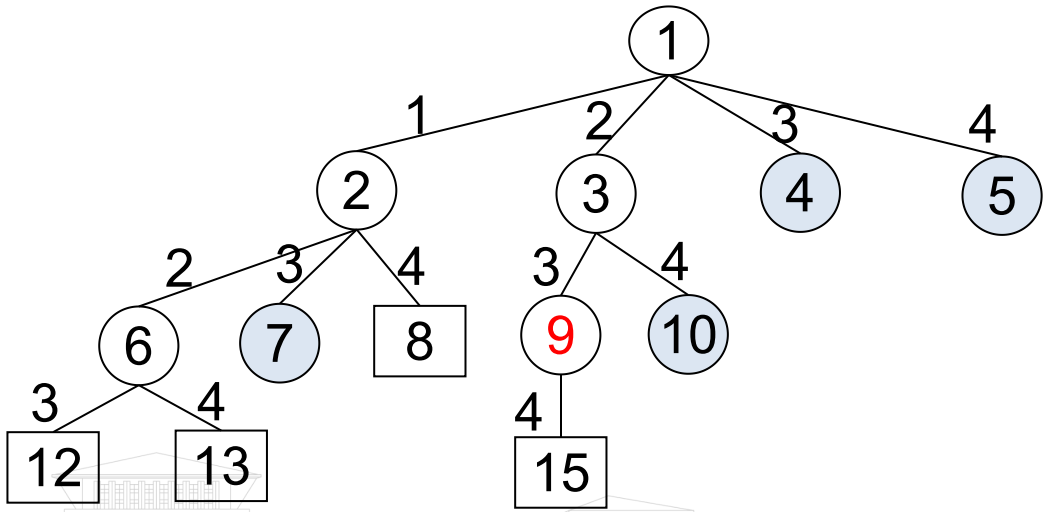
FIFO-分支限界法实例运行

- n=4; $(p_1,d_1,t_1)=(5,1,1)$; $(p_2,d_2,t_2)=(10,3,2)$;$(p_3,d_3,t_3)=(6,2,1)$; $(p_4,d_4,t_4)=(3,1,1)$;

结点编号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ĉ	0	0	5	15	21	0	10	∞	5	11	∞	∞	∞	∞	∞
cost	24	19	14	18	21	9	13	∞	8	11	∞	∞	∞	∞	∞
u	24	19	14	18	21	9	13	∞	8	11	∞	∞	∞	∞	∞

ε=0.1

U	ans	E
24	1	1
19	2	
14	3	2
9	6	3
8	9	6
		9



队列 2 3 6 9

8.6 0/1背包问题

- 问题描述
- 一个问题实例
- 成本函数c
- 成本下界函数ĉ与成本上界函数u
- LC-分支限界法实例运行

问题描述

- 0/1背包问题要求物品或者整件装入背包中，或者根本不装入(即不能装入物品的一部分)，所以x_i限定只能取0或1值。

0/1背包问题的形式描述

极大化 $\sum_{1\le i\le n} p_i x_i$ x_i=0或1， p_i>0

约束条件 $\sum_{1\le i\le n} w_i x_i \le M$ w_i>0, 1≤i≤n

一个问题实例

- n=4, M=15，$(p_1,p_2,p_3,p_4)=(10,10,12,18)$，$(w_1,w_2,w_3,w_4)=(2,4,6,9)$
- 解空间的表示方法
  - n-元组$(x_1,..x_n)$，n=4
  - 显示约束条件：x_i =0/1, 1≤i≤n
  - 隐式约束条件：选中的物品不违反M的限制
  - 目标函数：选中的物品效益和极大化
- 状态空间树
  - 共计31个结点，15个非叶结点和16个叶结点，答案结点在叶结点中
- 将物品按照p/w非增次序排列，利用贪心法探查

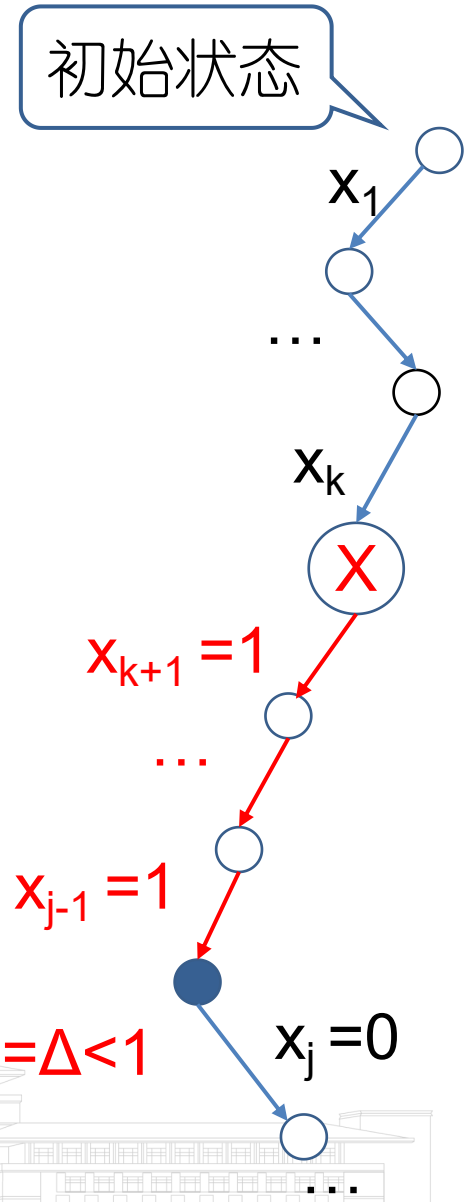
成本函数c

- 原问题寻找最大成本结点，转换目标函数，修改最优解定义为极小值。
  - $\min\{-\sum_{1\le i\le n} p_i x_i\}$
- 为了使最小成本答案结点与最优解相对应，结点X定义成本函数c：
  - $c(X)= -\sum_{1\le i\le n} p_i x_i$，X是答案结点
  - $c(X)=\infty$，X是不可行的叶结点
  - c(X)=min{c(X的左儿子)),c(X的右儿子)}，X是非叶结点

思考：怎样构造成本估计函数$\hat{c}$和成本上界函数u，满足$\hat{c}(X)\le c(X)\le u(X)\le 0$？

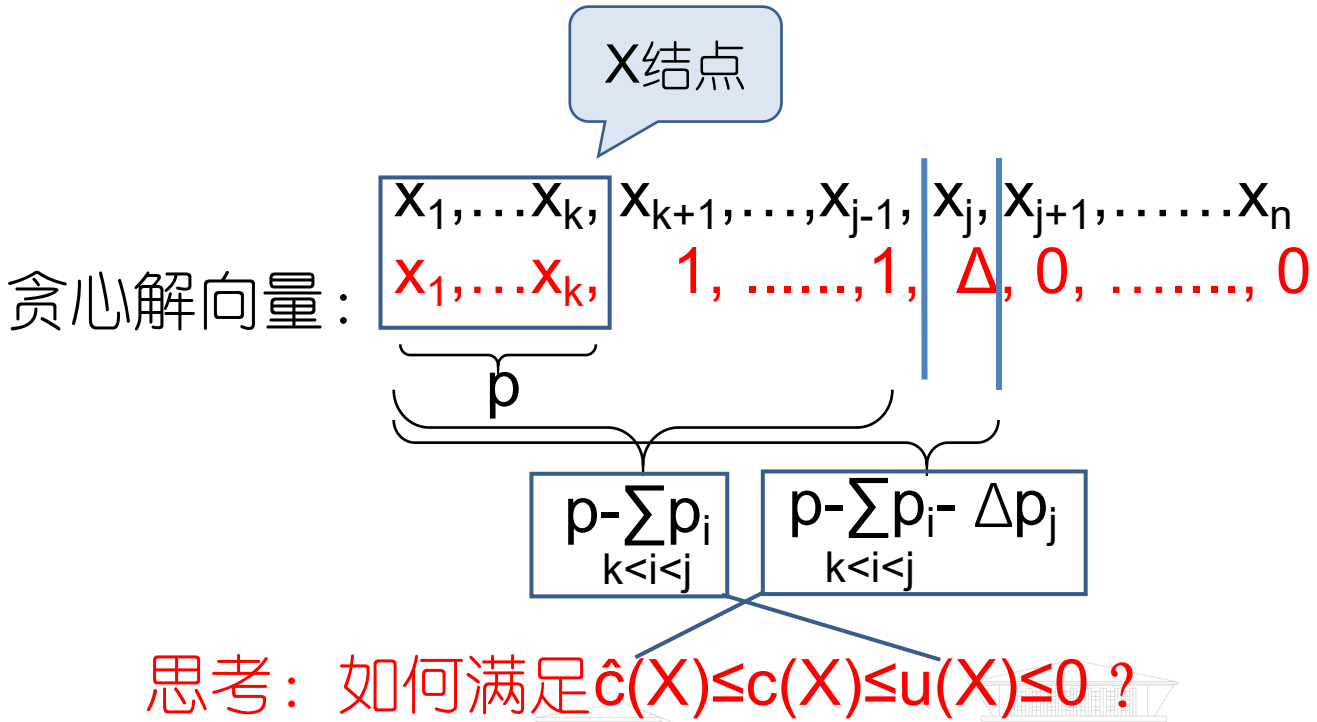
分析贪心解

- X表示路径$x_1,\ldots x_k$到达的结点，$p= -\sum_{1\le i\le k} p_i x_i$
- 已知物品按照p/w非增次序排列，从结点X开始基于贪心策略估计$x_{k+1}\ldots x_n$的取值，此时约束条件为$M-\sum_{1\le i\le k} w_i x_i$
- 设j是贪心解中第一个取值非1的位置下标
  - j>n时，物品全部选中
  - j≤n时，贪心解$= \sum_{k<i<j} p_i + \Delta p_j$
- 以X为根考虑，最优解一定小于等于贪心解。



ĉ函数与u函数

- 分析以X为根的贪心解向量，即考虑$x_{k+1}\ldots x_n$的取值：



思考：如何满足$\hat{c}(X)\le c(X)\le u(X)\le 0$？

ĉ函数与u函数

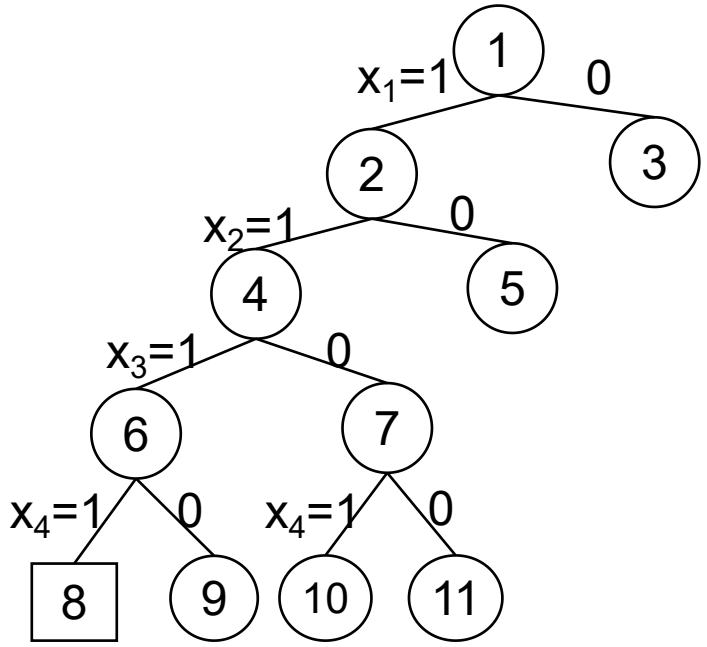
- 令X表示路径$x_1,\ldots x_k$到达的结点，$p= -\sum_{1\le i\le k} p_i x_i$
- 设计成本估计函数ĉ和成本上界函数u，满足$\hat{c}(X)\le c(X)\le u(X)\le 0$。
- ĉ(X):从根结点到结点X的贪心解的相反数
  - $\hat{c}(X) = p-\sum_{k<i<j} p_i - \Delta p_j$
- u(X):从根结点到结点X的贪心解整取部分的效益值的相反数
  - $u(X) = p-\sum_{k<i<j} p_i$

ĉ值与U值

- n=4, M=15，$(p_1,p_2,p_3,p_4)=(10,10,12,18)$，$(w_1,w_2,w_3,w_4)=(2,4,6,9)$

结点编号	贪心策略	U	ĉ
1	1,1,1,1/3	-32	-38
2	1,1,1,1/3	-32	-38
3	0,1,1,5/9	-22	-32
4	1,1,1,1/3	-32	-38
5	1,0,1,7/9	-22	-36
6	1,1,1,1/3	-32	-38
7	1,1,0,1	-38	-38
8(舍弃)			
9	1,1,1,0	-32	-32
10	1,1,0,1	-38	-38
11	1,1,0,0	-20	-20

=C



LC-分支限界法实例运行

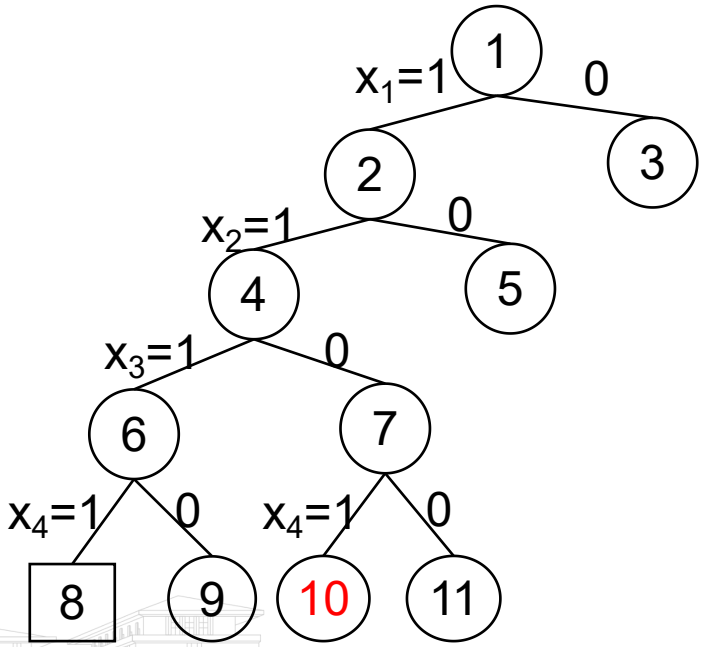
- n=4, M=15，$(p_1,p_2,p_3,p_4)=(10,10,12,18)$，$(w_1,w_2,w_3,w_4)=(2,4,6,9)$

结点编号	1	2	3	4	5	6	7	8	9	10	11
ĉ	-38	-38	-32	-38	-36	-38	-38	∞	-32	-38	-20
u	-32	-32	-22	-32	-22	-32	-38	∞	-32	-38	-20

ε=0.1

活结点表	U	ans	E	儿子结点
空	-31.9		1	2,3入表
2,3	-31.9		2	4,5入表
3,4,5	-31.9		4	6,7入表
3,5,6,7	-37.9		6	8和9的ĉ大于U
3,5,7	-37.9		7	10入表，11的ĉ大于U
3,5,10	-38	10	10	

ĉ≥U ,算法结束



8.7 小结

- 分支限界法与回溯法的相同点：
  - 同样适用于求解组合数较大的问题(/多阶段决策问题)
  - 都是在解空间树上搜索答案结点
  - 都会借助限界函数B进行剪枝
- 回溯法与分支限界法的不同点：
  - 最本质的区别在于E-结点(即扩展结点)处理方式不同，见第七章；分支限界法还可以基于ĉ选择，因此求最优解问题时效率更高
  - 存储空间上，分支限界法需要额外维护活结点表，回溯法不需要

- 分支限界法求解最优解，即寻找状态空间树中最小成本的答案结点
  - 目标函数作为成本函数c
  - 约束条件作为限界函数B
  - 设计成本估计函数ĉ(X)，ĉ(X)≤c(X)
  - 设计最小成本的上界U，c(X)≤U
  - 基于ĉ(X)和U进行分支限界搜索

- 8.1 一般方法
  - 掌握分支限界法适用的问题特点，掌握分支限界法求解问题的设计思想和一般方法，掌握分支限界法的不同检索方式的差异性
- 8.2 LC-检索
- 8.3 15-谜问题
  - 掌握结点成本函数c定义、成本估计函数ĉ定义和LC-检索定义，以15-谜为例理解LC-检索的优势，掌握15-谜问题判定定理和ĉ设计
- 8.4 求最小成本的分支限界法
  - 掌握分支限界法求最优解问题的一般算法设计
  - 掌握基于ĉ(X)和U优化算法的一般思想

- 8.5 带有期限的作业调度问题
- 8.6 0/1背包问题
  - 掌握经典问题的解空间构造方法，掌握限界函数B的设计和优化思想，掌握ĉ(X)和U的经典设计方法，掌握基于ĉ(X)和U的分支限界优化算法

能够识别出适合分支限界法的可计算性问题、独立设计算法和分析算法复杂度。

本 章 结 束