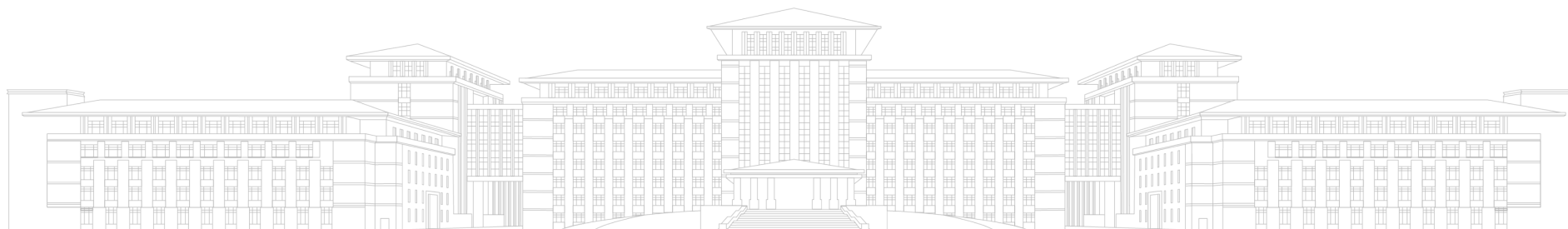




第八章 分支限界法



目录

- 8.1 一般方法
- 8.2 LC-检索
- 8.3 15-谜问题
- 8.4 求最小成本的分支限界法
- 8.5 带有期限的作业调度问题
- 8.6 0/1背包问题
- 8.7 小结

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
- 算法8.1 分支限界法的抽象化描述
- 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，解向量的表达、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。

找一个可行解
- 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，输出到根的路径，操作结束；
 - 如果 X 满足约束函数 B ，则将 X 添加到活结点表中；否则舍弃 X 。
 - 从活结点表中选出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。

约束函数剪枝作用：避免生成那些不包含可行解的子树。

算法8.1 分支限界法的抽象化描述

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 表中不再有活结点 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

找一个可行解

思考：如果想获得所有可行解，算法怎样设计？

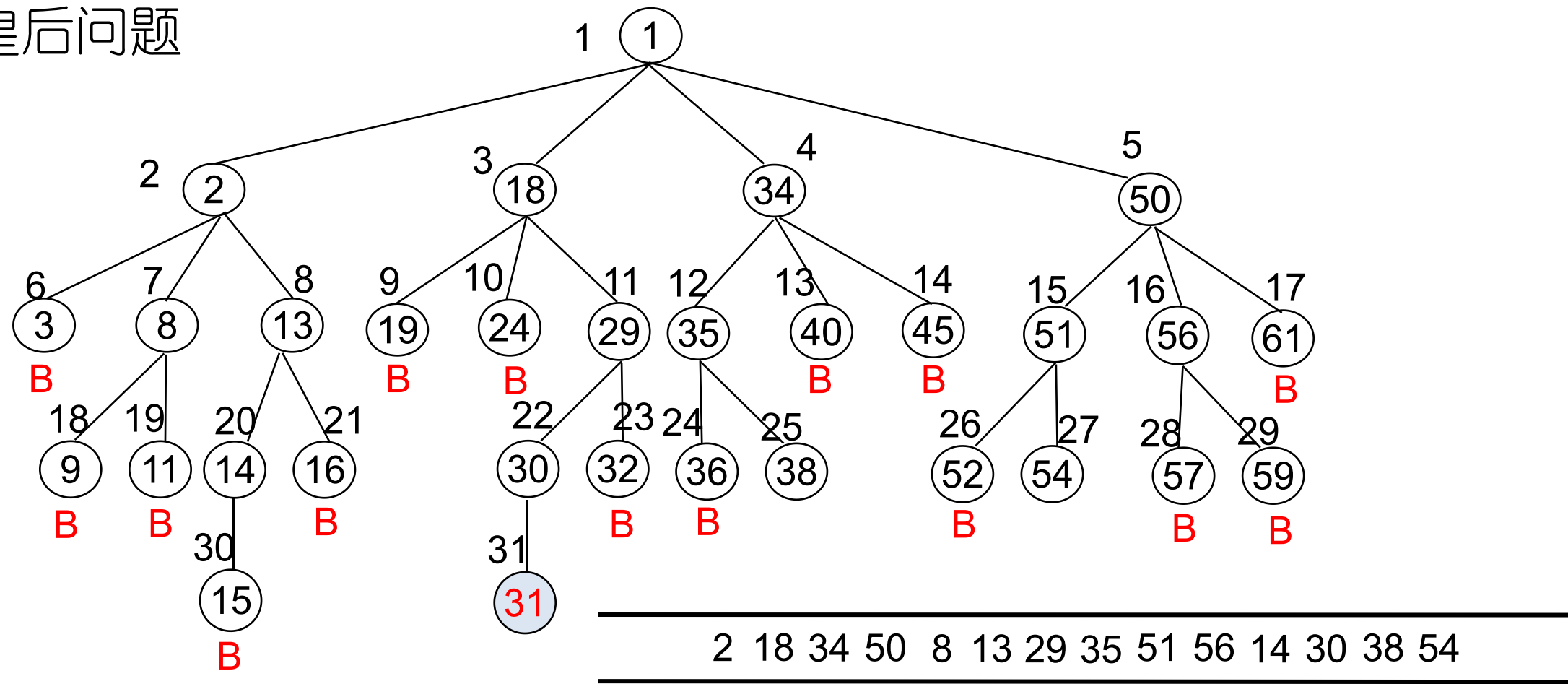
分支限界法的不同检索方式

- 根据活结点检索次序，分支限界策略可以分为：
 - 顺序队列式
 - 活结点依赖进入活结点表的顺序从活结点表中被选出
 - 常见先进先出方式(FIFO)，活结点表采用队列实现。
 - 优先队列式
 - 活结点依赖成本估计函数 \hat{c} ， \hat{c} 最小/最大的活结点优先从活结点表中被选出。
 - 活结点表采用极小堆/极大堆实现。
- 若LEAST遵循：
 - FIFO-顺序队列式：算法即为FIFO-检索
 - 极小堆优先队列式：算法即为LC-检索

类似于BSF-检索

理解FIFO-检索

●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 \hat{c} 定义
- LC-检索总结

LC-检索的优点

- 在**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。
- $c(X)$ 的定义关注最小成本
 - 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定义
 - 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定义
 - 寻找路径长度最短的答案结点
 - 方法3：基于问题描述中的目标函数定义
 - 寻找使目标函数取极值的答案节点

若问题中不存在目标函数，通常采用方法2定义 c 。

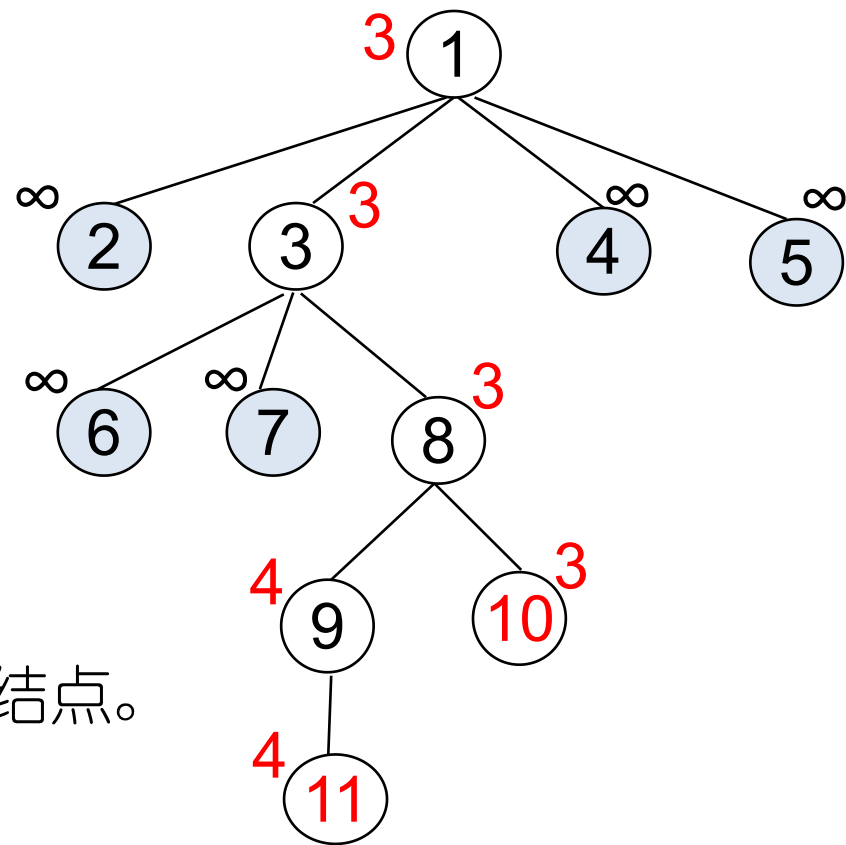
成本函数c定义

- $\text{cost}(X)$
 - 表示根到达答案结点X的**真实成本/代价**，如路径长度、目标函数值等
- 从**上帝视角**对状态空间树里面的任何结点X定义成本函数c
- $$c(X) = \begin{cases} \text{cost}(X) & \text{叶节点X是答案结点} \\ \min\{\text{cost}(P) \mid \text{答案结点 } P \in \text{子树X}\} & \text{子树X中包含答案结点} \\ \infty & \text{子树X不包含答案结点} \end{cases}$$

思考：在什么样的状态空间树中会出现答案结点X不是叶结点的情况？

k-元组表达

成本函数 c 的例子

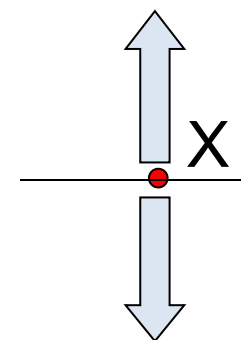


- 设 $c(X)$ =到达答案结点的最短路径长度
- 图中，蓝色结点表示不能到达一个答案结点。结点10和11是答案结点。

叶节点 X 是答案结点时， $c(X)$ =根结点到 X 的成本
子树 X 中包含答案结点时， $c(X)$ =子树 X 中最小成本答案结点的成本

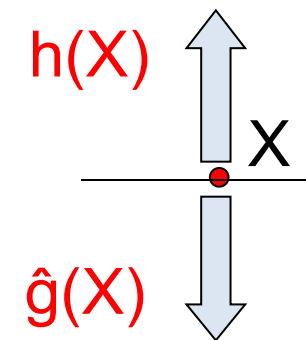
成本函数 c 的问题

- 要得到函数 c 很困难
 - $c(X)$ 基于答案结点的真实成本定义，为求出该值，需要生成状态空间树。
 - c 的计算工作量与原问题具有相同的复杂度。
- 基于 c ，定义一个易于计算的估计成本函数 \hat{c} ，来替代 c 对活结点表进行检索。定义 $\hat{c}(X)$ 时：
 - 明确根到 X 已经产生的成本很容易
 - 估计 X 到一个答案结点可能会产生的成本，这是一个富于创造性的工作



成本估计函数 \hat{c} 定义

- 成本估计函数 $\hat{c}(X) = f(h(X)) + \hat{g}(X)$
 - $h(X)$: 根结点到结点 X 的成本
 - $\hat{g}(X)$: 子树 X 中, X 到最小成本答案结点的估计成本
 - f : 为调整 h 和 \hat{g} 在成本估计函数 \hat{c} 中的影响比例而定义的非负函数
- 对成本估计函数 \hat{c} 的要求
 - 易于计算, 能够替代 c 对活结点表进行检索
 - $\hat{c}(X) \leq c(X)$
 - 当叶节点 X 是答案节点时, $c(X) = \hat{c}(X)$



LC-检索

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

- **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。
- **BFS-检索**: $f(h(X))$ =根到结点X的路径长度, $\hat{g}(X)=0$

BFS是LC-检索的特殊情况

LC-检索的优点:

算法利用 \hat{c} 对活结点表进行检索, 优先选择更靠近答案结点, 同时又离根结点较近的结点。

LC-检索总结

- T 是状态空间树， c 是 T 中结点的成本函数， $c(X)$ 表示 X 为根的子树中答案结点的最小成本。找到这样一个易于计算的 c 是很困难的，因此基于 c 设计一个成本估计函数 \hat{c} 。
- 成本估计函数 \hat{c} 要易于计算，且 $\hat{c}(X) \leq c(X)$ ；当叶节点 X 是答案节点时， $c(X) = \hat{c}(X)$ 。
- 在分支限界法中，使用 \hat{c} 替代 c 对活结点表进行检索。
- LC-检索适用于求最优解问题。

8.3 15-谜问题

- 问题描述
- 状态空间树
- **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	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

(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， $1 \leq i \leq 16$ ，**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	2	3	4
5	6		8
9	10	7	11
13	14	15	12

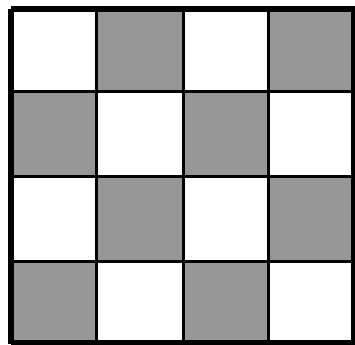
(a)

位置16

牌面1,2,3,4,5,6,7和11, 12的LESS值为0

牌面 <i>i</i>	8	9	10	13	14	15	16
LESS(<i>i</i>)	1	1	1	1	1	1	9

判定定理



(c)

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 $X=0$ 。

状态(a)有 $\sum LESS(i)+X=15+1$ 是偶数，可达状态(b)

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

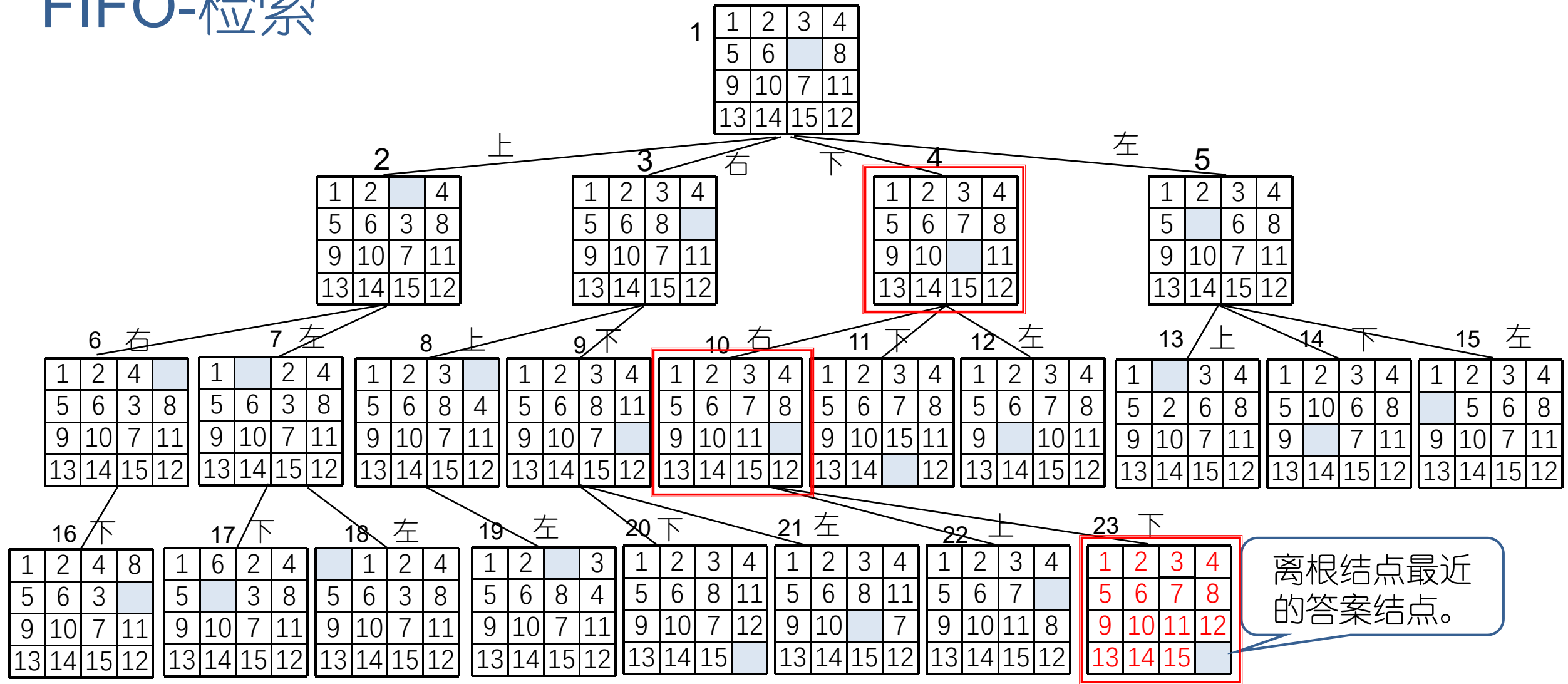
(a)

初始状态判定定理：

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

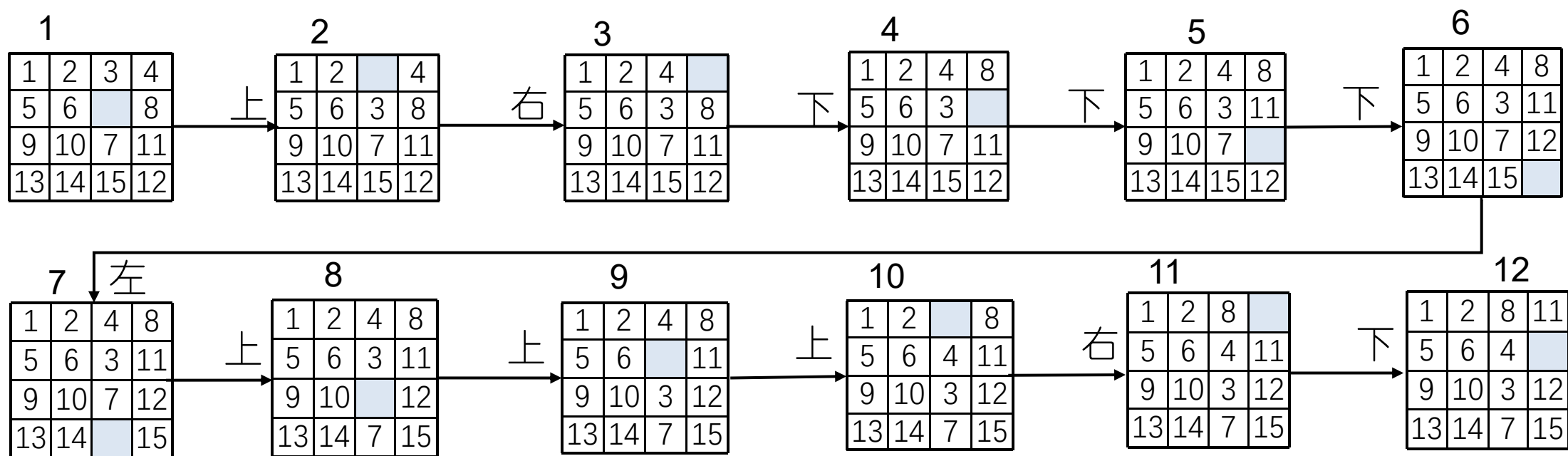
由于移动牌与移动空格等效，因此状态空间树中，边表示为空格的一次合法移动，按上、右、下、左的顺序进行。

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

- 15谜问题的 $c(X)$ 和 $\hat{c}(X)$ 定义
 - 定义 $c(X)$: 从初始排列到达目标排列时, 棋牌最少移动次数
 - 基于 c 定义 $\hat{c}(X)$: $\hat{c}(X)=f(X)+\hat{g}(X)$
 - $f(X)$: 从初始排列到 X 时, 棋牌已经移动的次数
 - $\hat{g}(X)$: 不在其目标位置的非空白棋牌数目

例如: 右图(a)中, 不在目标位置的非空白棋牌有{7,11,12}, 所以 $\hat{g}(X)=3$

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

(a)

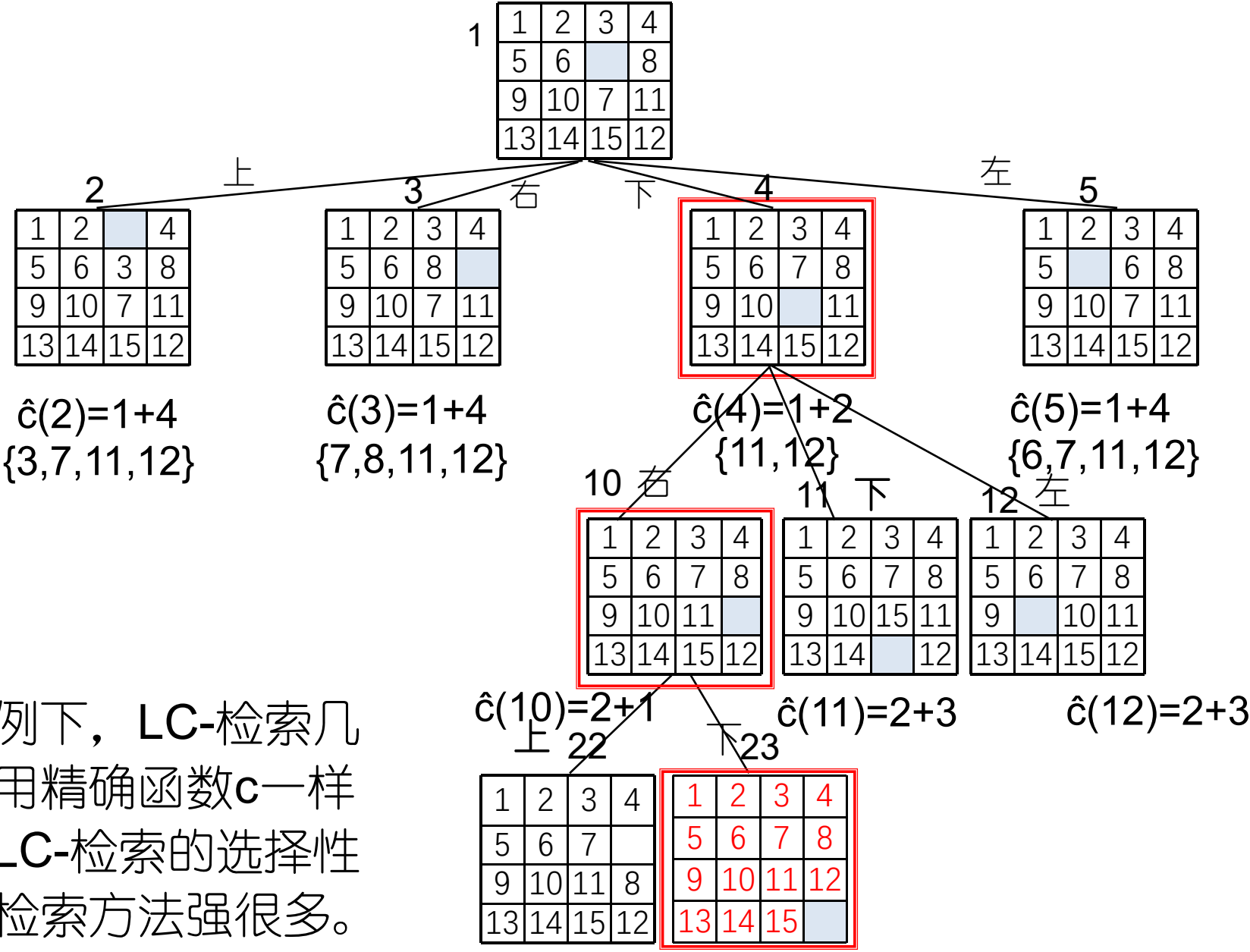
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

(b)

LC-检索

- 在状态空间树中分析15谜问题的 $c(X)$ 和 $\hat{c}(X)$
- $c(X)$ 寻找离根结点最近的答案结点。
 - 根结点到最近答案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 ∞ 。
- 基于 c 定义 $\hat{c}(X)$: $\hat{c}(X)=f(X)+\hat{g}(X)$
 - $f(X)$ 对应根到结点 X 的路径长度。
 - $\hat{g}(X)$ 对应以 X 为根的子树中, X 到答案结点的最短路径长度的最小估计值。
- 综上, $\hat{c}(X)$ 的设计能满足LC-检索要求。

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

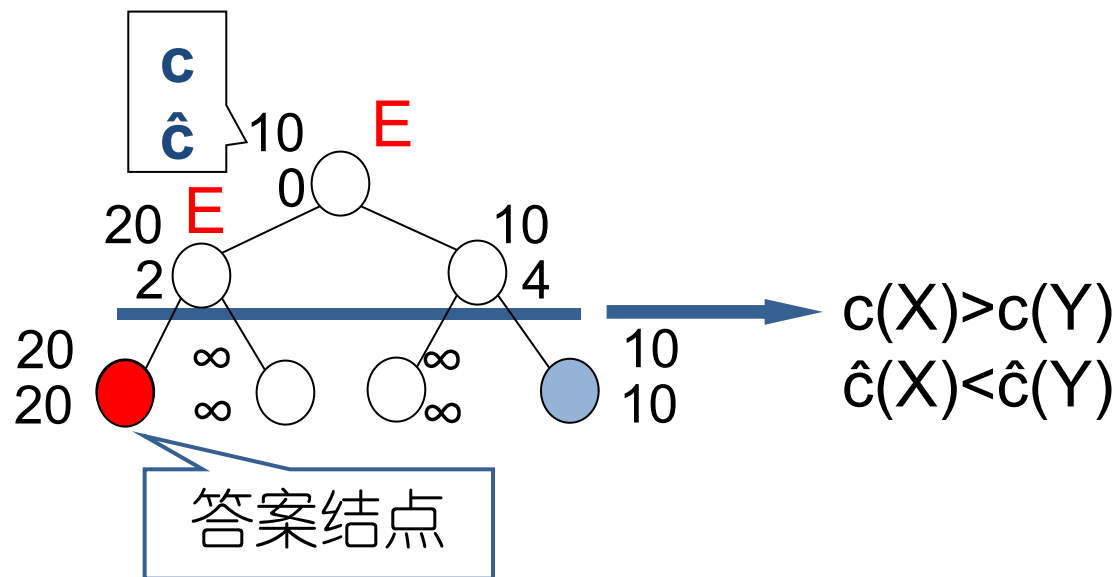
- 寻找最小成本
- 算法8.2 基于 \hat{c} 求最小成本的LC-检索算法
- 求最小成本的分支限界法基本思想
- 最小成本上界U
- 算法8.3 界函数UB
- 算法8.4 求最小成本的FIFO-分支限界算法
- 算法8.5 求最小成本的LC-分支限界算法
- 求极大化问题
- 总结求最优解问题的分支限界法

可行解：满足约束条件的答案结点。
最优解：使目标函数取极值的答案结点。

寻找最小成本

BB算法中：算法一旦判断出儿子结点X是答案结点，则打印路径，操作结束。

- 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？



- 对 \hat{c} 追加要求：对于每一对结点X、Y，当 $c(X) < c(Y)$ 时有 $\hat{c}(X) < \hat{c}(Y)$

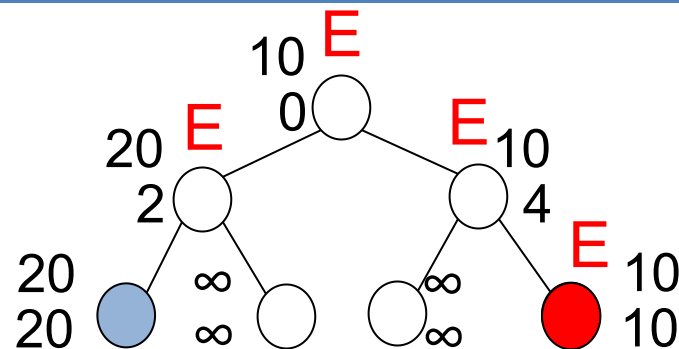
寻找最小成本

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

难实现

改进算法：

算法从活结点表中选出E结点时，再判断E是否是答案结点，若是则打印路径，操作结束。



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 基于 \hat{c} 求最小成本的LC-检索算法

procedure LC(T, \hat{c}) //为找出最小成本答案结点

$E \leftarrow T$, 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) \leftarrow E; endif

repeat

if 表中不再有活结点 then print("no answer node"); return; endif

call LEAST(E)

repeat

end LC

选择 \hat{c}
最小的

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

若答案结点E不满足 $\hat{c}(E) = c(E)$, 则算法LC也不能给出成本最小的答案结点

加速寻找最小成本

- 基本思想：
 - 满足约束条件的可行解并不一定成本最小，因此在寻找最优解过程中，若发现结点 X 不能导致最小成本答案节点，则不必再搜索子树 X ，子树 X 被剪枝。
- 加速：
 - 设置一个最小成本上界 U ，问题的最小成本不会大于这个上界 U 。如果 $\hat{c}(X) > \text{或} \geq \text{上界} U$ ，则算法无需检索以 X 为根的子树。
 - 算法借助 U 和 $\hat{c}(X)$ ，进一步提高剪枝能力。

基于 \hat{c} 和 U 求最小成本的分支限界法基本思想

分支限界法求极小化问题时，问题转化为在解空间树中寻找最小成本答案节点。

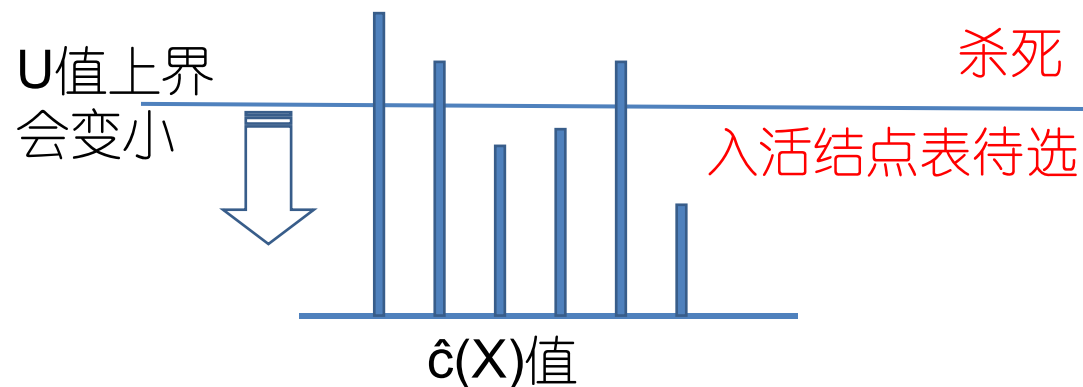
- 1) 对于极小化问题，把目标函数作为成本函数 c
- 2) 约束条件作为约束函数 B
- 3) 问题转化为寻找解空间树中最小成本答案结点
- 4) 设计成本估计函数 $\hat{c}(X)$ ， $\hat{c}(X) \leq c(X)$
- 5) 还可以设计最小成本的上界 U ， $c(X) \leq U$
- 6) 基于 $\hat{c}(X)$ 和 U 进行分支限界搜索

U 是界

思考：极大化问题如何转化？

最小成本上界U

- U的含义
 - U是当前算法生成的所有状态结点对最小成本上界估计的最小值
- U的取值
 - 初值 ∞ ，或通过启发式方法得到；初值 \geq 最小成本答案结点的成本
 - 随着结点的访问不断改小
- U的作用
 - 界定当前结点的死活
 - 通过剪枝加快找到最优解



成本上界函数 $u(X)$

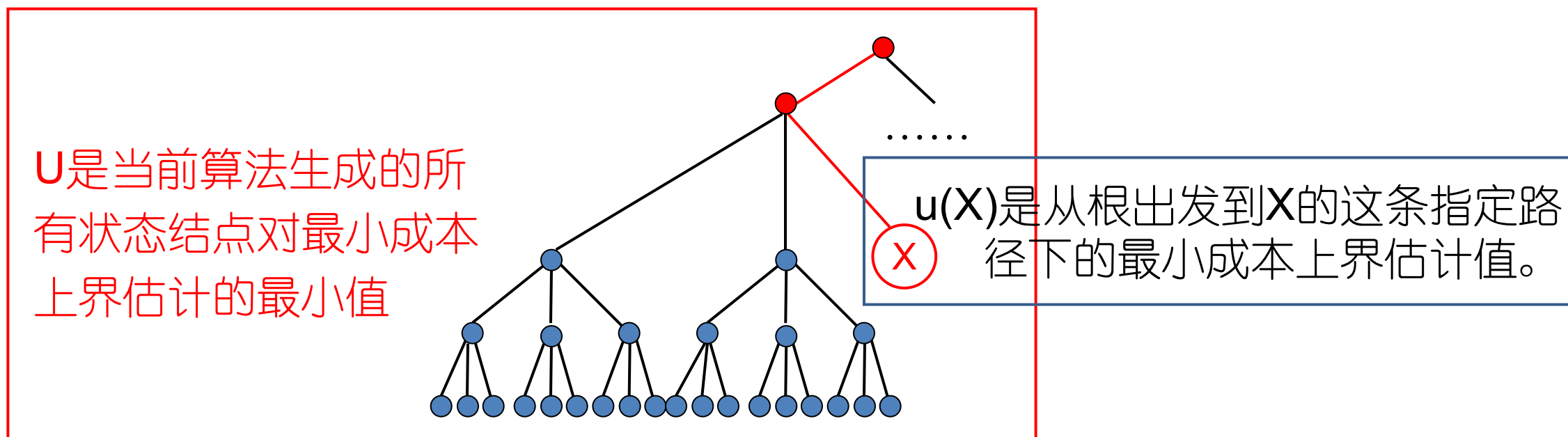
为提高 U 的剪枝能力，希望能不断减小 U 值。 U 可以通过 $u(X)$ 加速减小。

- $u(X)$ 是对成本 $c(X)$ 的上界估计，以下简称成本上界函数
- $u(X)$ 可仿照 $\hat{c}(X)$ 定义，或其他方式：
 - 根到 X 的成本
 - 子树 X 中， X 到最小成本答案节点的成本上界估计
- 对成本上界函数 u 的要求：
 - 易于计算
 - $c(X) \leq u(X)$
 - 当叶结点 X 是答案结点时， $c(X) = u(X)$

对于结点 X ，有 $\hat{c}(X) \leq c(X) \leq u(X)$

区分界 U 和 $u(X)$

算法当前生成的状态空间树如图所示， X 是当前正在生成的结点



U 会检查 x 的 \hat{c} 值，界定 X 结点的死活；也会利用 $u(X)$ 值改小自己

细谈界U的剪枝

算法在生成状态结点时，当X是答案结点且 $\text{cost}(X) < U$ 时，也可以改小U值。

• U改值

- 发现一个答案结点X，有更小的真实成本 $\text{cost}(X)$ ，修改U
- 发现一个状态结点X，有更小的成本上界 $u(X)$ ，修改U

界定当前结点的死活

• U剪枝

- 若U值来自一个真实成本， $\hat{c}(X) \geq U$ 的活结点X都可以被杀死
 - 此时U记录当前最小成本的答案结点，想找更小成本的答案结点
- 若U值来自一个成本上界， $\hat{c}(X) > U$ 的所有活结点X都可以被杀死
 - 此时能确定存在，但还没有找到一个成本小于等于U的答案结点

选值小的

思考：答案结点X非叶节点，如果 $\text{cost}(X)$ 和 $u(X)$ 都优于当前U值，该怎样选择？

改进界U的剪枝

目前的问题：U值来源不同，剪枝策略不同。

为方便剪枝，当U将从一个 $u(X)$ 获取数值时，U值调高一点。

- 一个改进技巧
 - 定义一个足够小的正常数 ϵ ，对任意结点 X, Y ，当 $u(X) < u(Y)$ 时，有 $u(X) < u(X) + \epsilon < u(Y)$
 - 当U从一个 $u(X)$ 获得数值时，再追加一个 ϵ 。即 $U \leftarrow u(X) + \epsilon$
 - 当U从一个 $\text{cost}(X)$ 获得数值时，无需调整。即 $U \leftarrow \text{cost}(X)$
- 改进后的U剪枝：
 - 当前结点 X 若满足 $\hat{c}(X) \geq U$ ， X 被杀死

算法8.3 界函数UB

- 一个足够小的正常数 ε ：对任意结点 X, Y ，如果 $u(X) < u(Y)$ ， $u(X) < u(X) + \varepsilon < u(Y)$

procedure UB(X, ε, U, ans)

// X 是当前活结点， U 是当前上界估计值， ans 是当前最小成本的答案结点。

//结点 X 满足 $\hat{c}(X) \leq c(X) \leq u(X)$ ；当 X 是答案节点时， $cost(X)$ 表示根到 X 的真实成本。

if $\hat{c}(X) \geq U$ then return false

if X 是解结点 and $cost(X) < U$

then $U \leftarrow \min(cost(X), u(X) + \varepsilon)$; $ans \leftarrow X$

else if $u(X) + \varepsilon < U$ then $U \leftarrow u(X) + \varepsilon$ endif

endif

return true

end UB

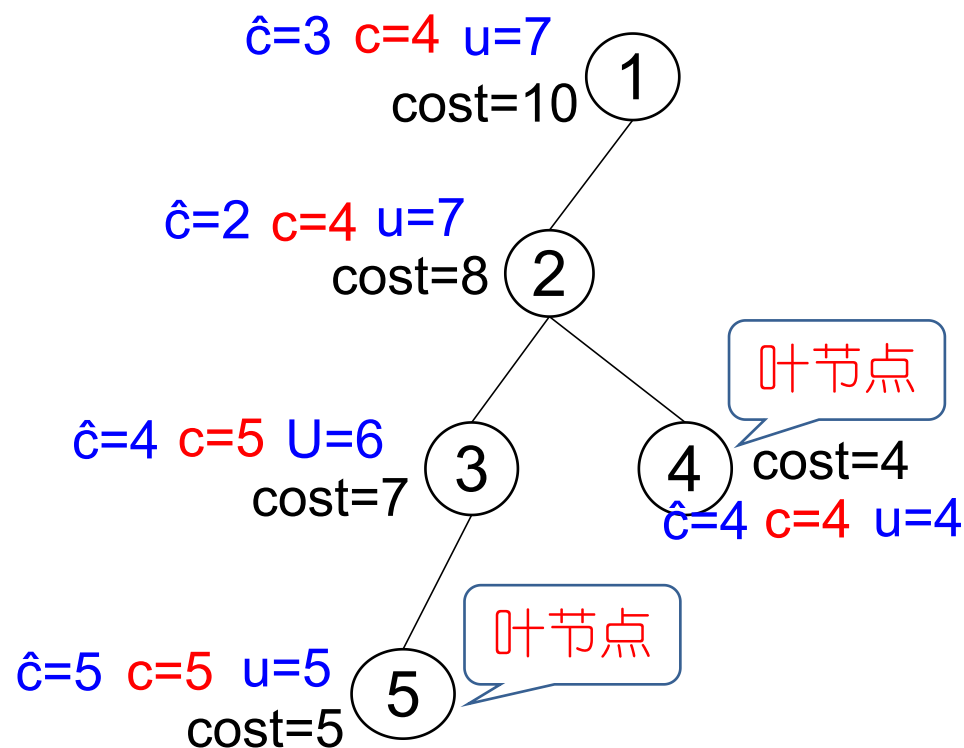
界定杀死，否则探查
是否能够更改 U 值

$U \leftarrow \min(cost(X), u(X) + \varepsilon)$ 较 $U \leftarrow cost(X)$ 更能提高剪枝

区别函数 cost , \hat{c} , c 和 u

- $\text{cost}(X)$: 根到答案结点 X 的真实成本
- $c(X)$: 子树 X 中所有答案结点的最小成本值, 即最小 cost 值
- $\hat{c}(X)$ 和 $u(X)$: 对子树 X 的 c 值估计, 满足 $\hat{c}(X) \leq c(X) \leq u(X)$

设当前每个结点都满足约束条件



图中, 当 $\text{cost}(X) > u(X)$ 时, 意味着以 X 为根的子树中一定有更优解。

算法8.4 求最小成本的FIFO-分支限界算法

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 $\hat{c}(X)=\infty$

procedure FIFOBB(T, \hat{c} ,u, ϵ ,cost)

E \leftarrow T; PARENT(E) \leftarrow 0; U $\leftarrow\infty$, ans \leftarrow 0

UB(E, ϵ ,U,ans)

将队列初始化为空

loop

for E的每个儿子X do

if B(X) and UB(X, ϵ ,U,ans) then call ADDQ(X); PARENT(X) \leftarrow E endif

repeat

loop

if 队列为空 then print ('least cost = ', U); 输出从ans到T的路径; return endif

call DELETEQ(E); if $\hat{c}(E)<U$ then exit endif

repeat

repeat

end FIFOBB

思考：请验证算法的正确性

算法8.5 求最小成本的LC-分支限界法

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 $\hat{c}(X)=\infty$

procedure LCBB(T, \hat{c} ,u, ϵ ,cost)

//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

E \leftarrow T; PARENT(E) \leftarrow 0; U \leftarrow ∞ , ans \leftarrow 0

UB(E, ϵ ,U,ans)

将活结点表初始化为空

loop

for E的每个儿子X do

if B(X) and UB(X, ϵ ,U,ans) then call ADD(X); PARENT(X) \leftarrow E endif

repeat

if 表中不再有活结点 or 堆顶结点 \hat{c} 值 \geq U

then print ('least cost = ',U); 输出从ans到T的那条路径; return endif

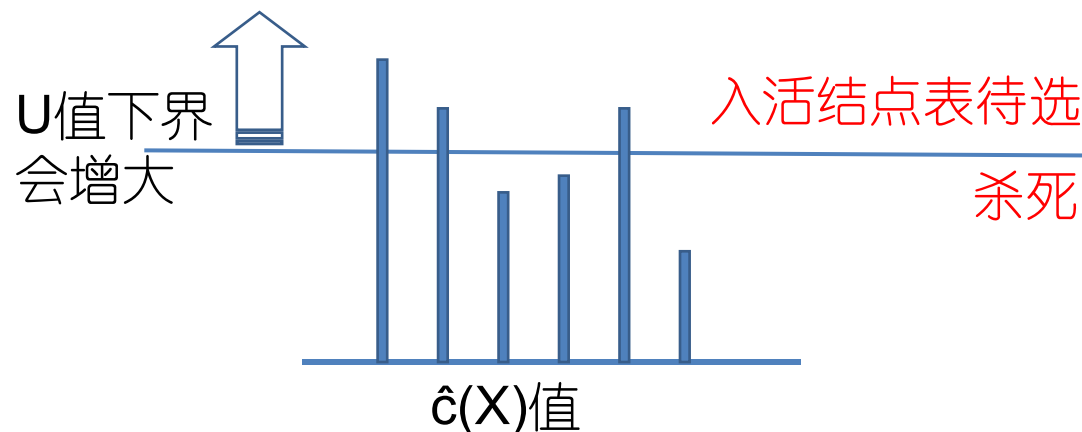
call LEAST(E)

repeat

end LCBB

求极大化问题

- 很多极值问题求目标函数极大值，可以采用以下两种方法解决：
- 方法1：修改目标函数，将问题转化为极小化问题
 - 取目标函数的相反数作为成本函数 c
- 方法2：对照极小化问题做镜像修改
 - 把目标函数作为成本函数 c ，问题转化为寻找解空间树中最大成本答案结点，此时 $u(X) \leq c(X) \leq \hat{c}(X)$ 。



总结求最优解问题的分支限界法

- 算法剪枝的依据
 - 约束函数 B ：限定是否存在可行解
 - 成本估计函数 $\hat{c}(X)$ 和界 U ：界定是否存在最优解
- 算法剪枝的发生点
 - X 入活结点表时，接受 B 检验和 U 检验
 - X 出活结点表时，接受 U 检验
- 算法终止的条件
 - 活结点表为空
 - 活结点表中再没有通过 U 检验的活结点

能

思考：用回溯法解决最优解问题时， $\hat{c}(X)$ 和界 U 能否用于提高算法效率？

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
- 成本估计函数 \hat{c}
- 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。

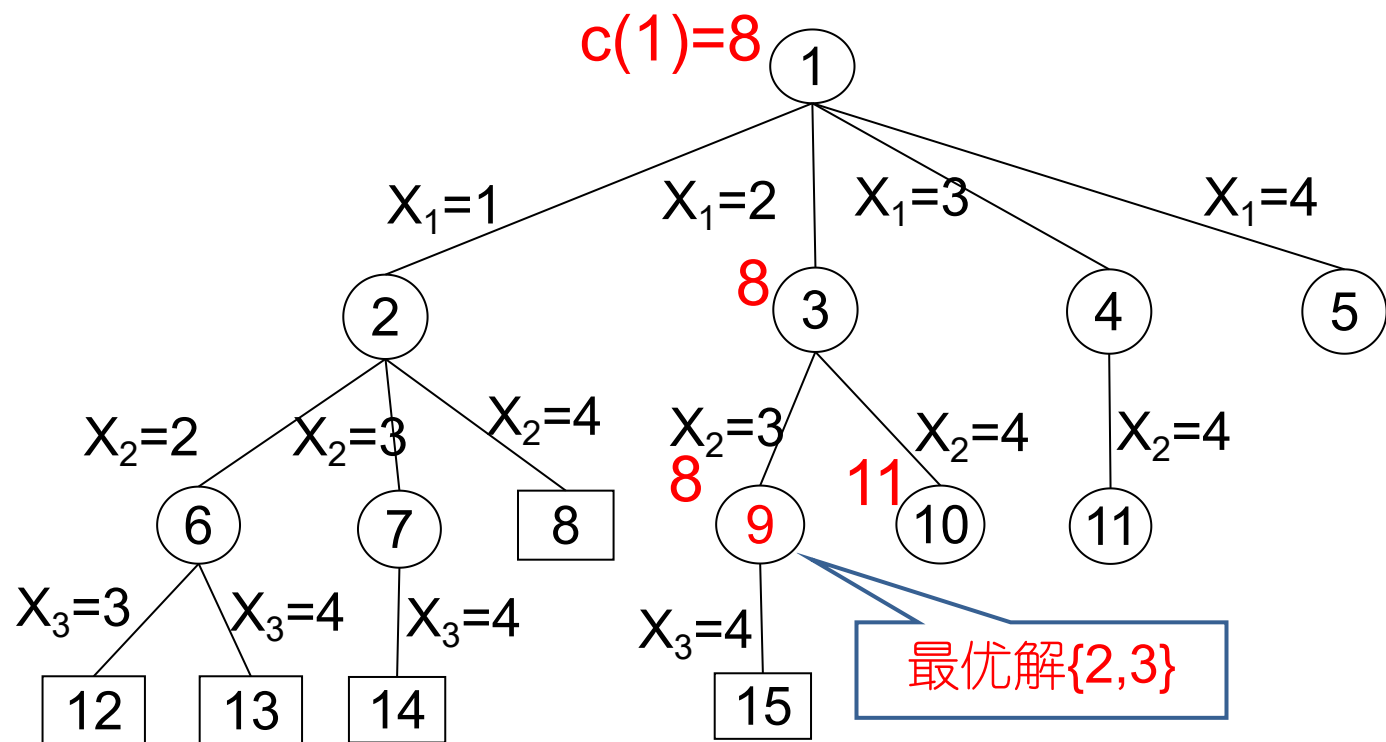
极小化问题

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, \dots, X_k) , $k \leq n$
 - 显式约束条件: X_j 表示选中的作业下标, $x_i \leq x_{i+1}$, $1 \leq i < k$
 - 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
 - 圆形结点表示满足约束条件的结点
 - 方形结点表示不可行结点

约束函数B

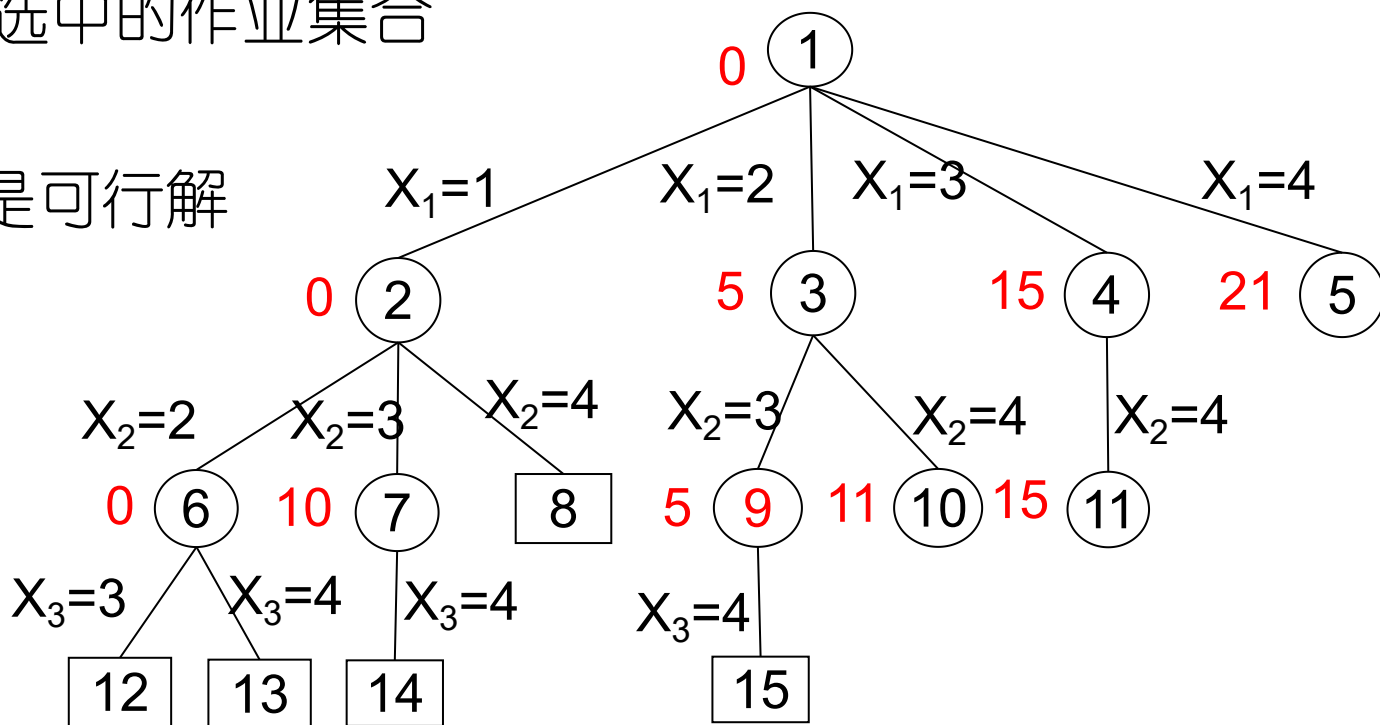
- $n=4$
- k -元组表示状态空间树
- 按层次遍历为结点编号
- 作业实例
 - $(p_1, d_1, t_1) = (5, 1, 1)$
 - $(p_2, d_2, t_2) = (10, 3, 2)$
 - $(p_3, d_3, t_3) = (6, 2, 1)$
 - $(p_4, d_4, t_4) = (3, 1, 1)$



10个活结点通过B检验

成本估计函数 \hat{c}

- 定义成本估计函数 $\hat{c}(X)$ ，使得 $\hat{c}(X) \leq c(X)$ ：
 - 设 S_x 是根结点到达结点 X 时选中的作业集合
 - 令 $m = \max\{|i| i \in S_x\}$
 - $\hat{c}(X) = \sum p_i, i < m, i \in S_x$ ，若 X 是可行解
 - $\hat{c}(X) = \infty$ ，若 X 是不可行解
- 作业实例
 - $(p_1, d_1, t_1) = (5, 1, 1)$
 - $(p_2, d_2, t_2) = (10, 3, 2)$
 - $(p_3, d_3, t_3) = (6, 2, 1)$
 - $(p_4, d_4, t_4) = (3, 1, 1)$



只当解结点 x 是叶结点时， $\text{cost}(X) = c(X) = \hat{c}(X)$ ，因此LC算法并不能求出最优解，只能考虑FIFOBB算法，或LCBB算法。

成本上界U

- 上界估计函数 $u(X)=\sum p_i, i \in S_x$
- 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 $U=\min\{u(X)\}$

$U \leftarrow 24$

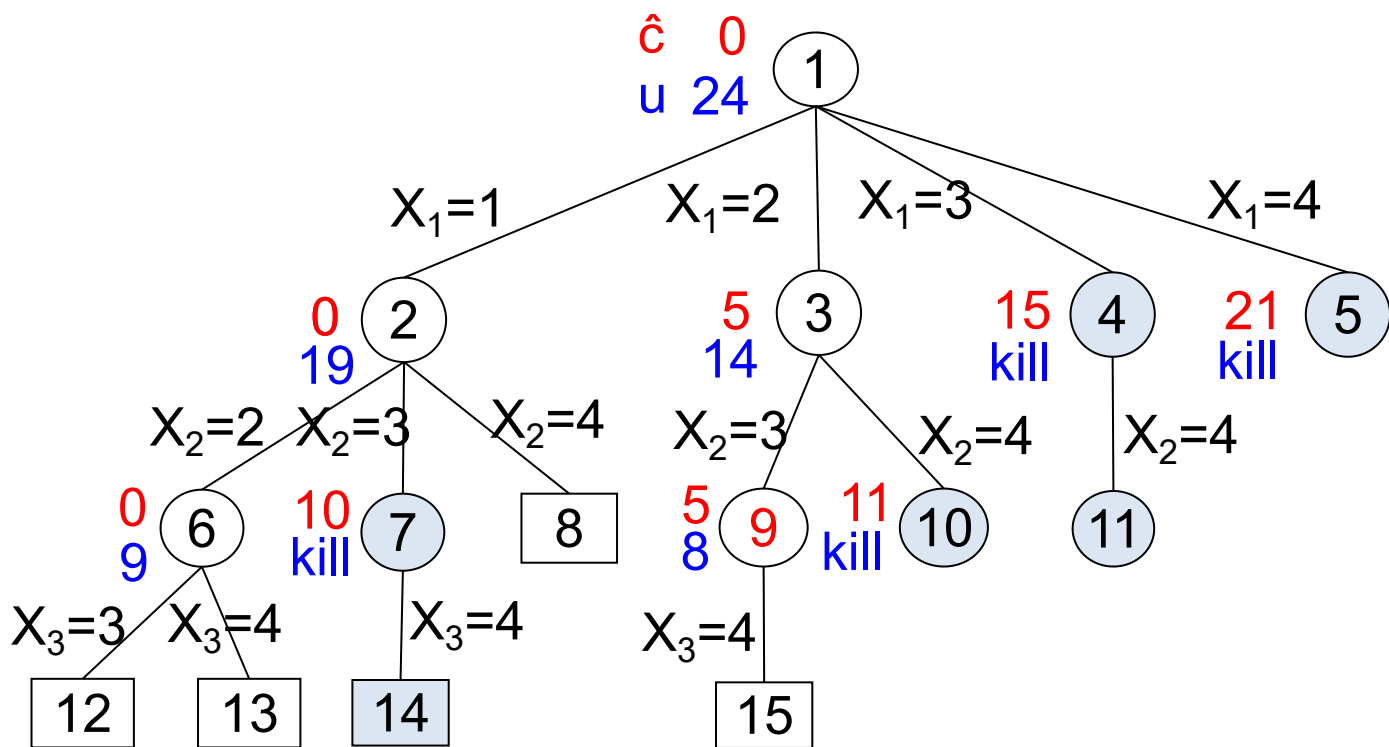
$U \leftarrow 19$

$U \leftarrow 14$

$U \leftarrow 9$

$U \leftarrow 8$

FIFO-检索下考虑U的剪枝作用



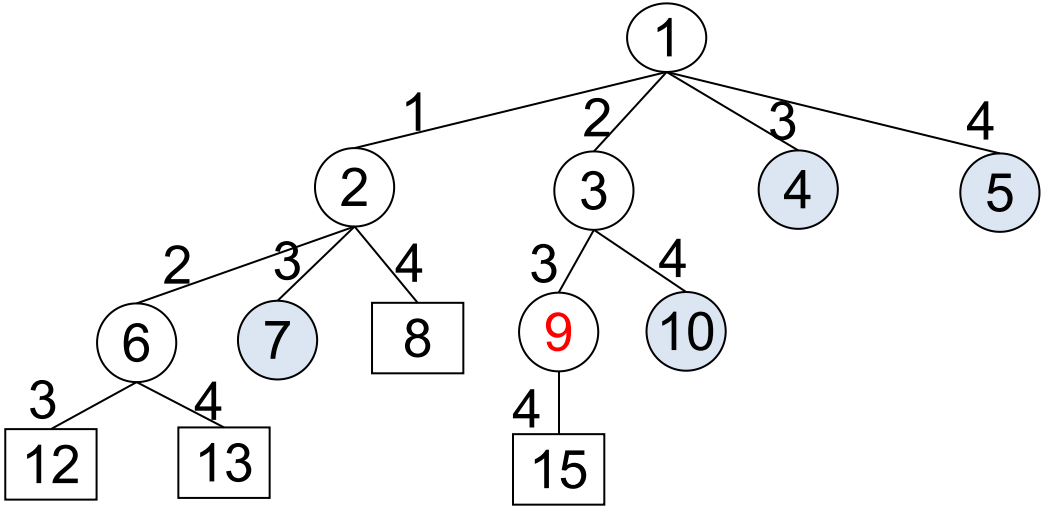
5个活结点通过U检验

FIFO-分支限界法实例运行

E	儿子X	U	ans	队列
1		24	1	
	2	19	2	
	3	14	3	2 3
2		14	3	
	6	9	6	
	7弃			
	8弃			3 6
3		9	6	
	9	8	9	
	10弃			6 9
6		8	9	
	12弃			
	13弃			9
9		8	9	
	15弃			

$\epsilon=0.1$, U不断减小; $\hat{c}(X) \geq U$ 时, X被杀死

编号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
\hat{c}	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	∞	∞	∞	∞	∞



队列为空, 算法结束。此时U等于8, ans等于9。

8.6 0/1背包问题

- 问题描述
- 一个问题实例
- 成本函数 c
- 分析贪心解
- 成本估计函数 \hat{c} 与成本下界函数 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。

极大化问题

0/1背包问题的形式描述

$$\begin{array}{ll} \text{极大化} & \sum_{1 \leq i \leq n} p_i x_i \\ \text{约束条件} & \sum_{1 \leq i \leq n} w_i x_i \leq M \end{array} \quad \begin{array}{l} x_i = 0 \text{ 或 } 1, \quad p_i > 0 \\ w_i > 0, \quad 1 \leq i \leq n \end{array}$$

对于极大化问题有 $u(X) \leq c(X) \leq \hat{c}(X)$ 。算法实现上, 活结点表采用极大堆; ϵ 改为一个足够小的负常数; U 不断增大, 当前结点 X 若满足 $\hat{c}(X) \leq U$, X 被杀死。

一个问题实例

- $n=4$, $M=10$, $(p_1, p_2, p_3, p_4)=(10, 10, 14, 18)$, $(w_1, w_2, w_3, w_4)=(2, 4, 7, 10)$
- 解空间的表示方法
 - n -元组 (x_1, \dots, x_n) , $n=4$
 - 显示约束条件: $x_i = 0/1$, $1 \leq i \leq n$
 - 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

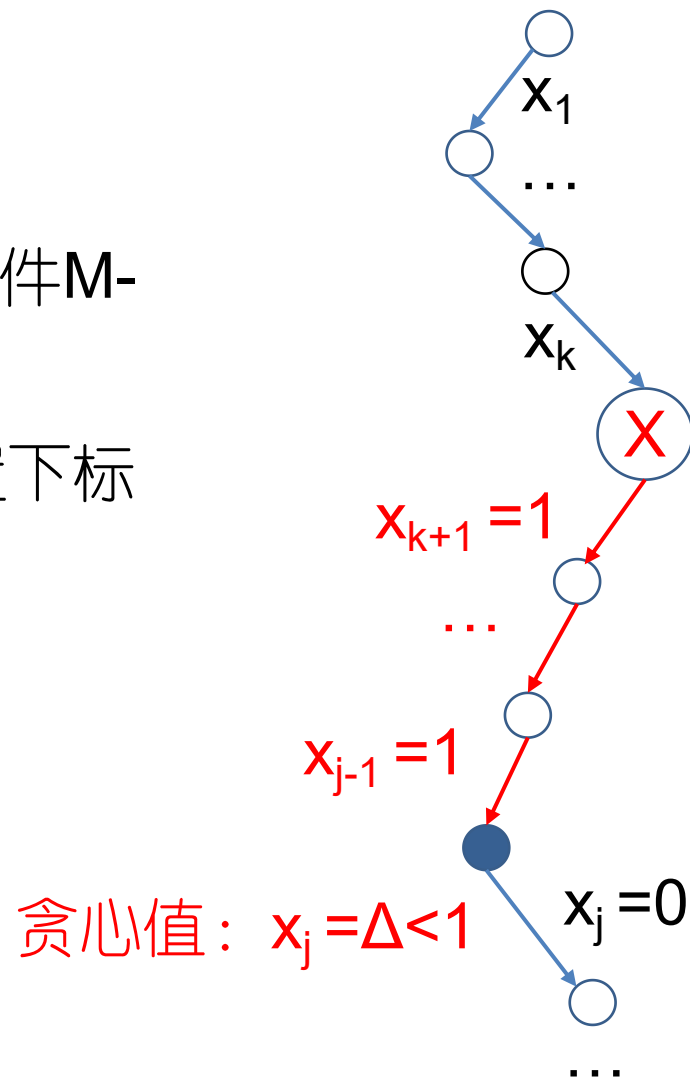
思考: 怎样构造成本估计函数 \hat{c} 和成本下界函数 u , 满足 $u(X) \leq c(X) \leq \hat{c}(X)$?

分析贪心解

- X 表示路径 x_1, \dots, x_k 到达的结点, $p = \sum_{1 \leq i \leq k} p_i x_i$
- 求子问题 X 的贪心解:
 - 已知物品按照 p/w 非增次序排列, 基于约束条件 $M - \sum_{1 \leq i \leq k} w_i x_i$, 求物品 $k+1, \dots, n$ 的贪心解
- 设 j 是贪心解向量 $x_{k+1} \dots x_n$ 中第一个取值非1的位置下标
 - 设 $x_j = \Delta$, $0 \leq \Delta < 1$
 - $j \leq n$ 时, 令 $pp = \sum_{k < i < j} p_i$, 贪心解 $= pp + \Delta p_j$
 - $j = n+1$ 时, 物品 $k+1, \dots, n$ 全部选中, $\Delta = 0$

易知: $pp \leq 0/1$ 背包问题 X 的最优解 \leq 贪心解

思考: $j = n+1$ 时, 不等式各项数值如何变化?



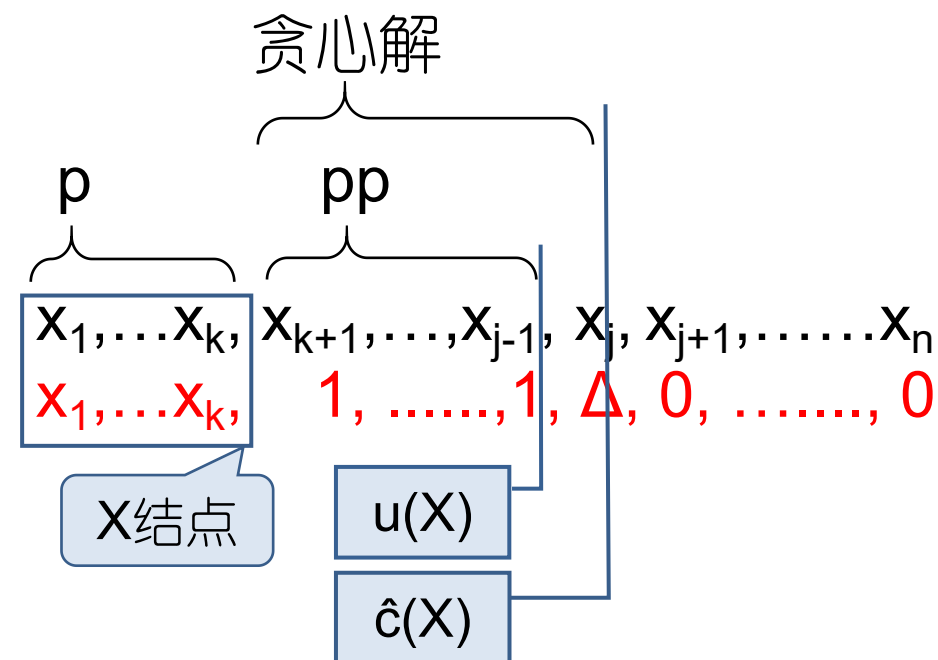
\hat{c} 函数与u函数

$$p = \sum_{1 \leq i \leq k} p_i x_i \quad pp = \sum_{k < i < j} p_i \quad \text{贪心解} = pp + \Delta p_j$$

$$\boxed{p + pp} \leq \boxed{p + \text{0/1背包问题X的最优解}} \leq \boxed{p + \text{贪心解}}$$

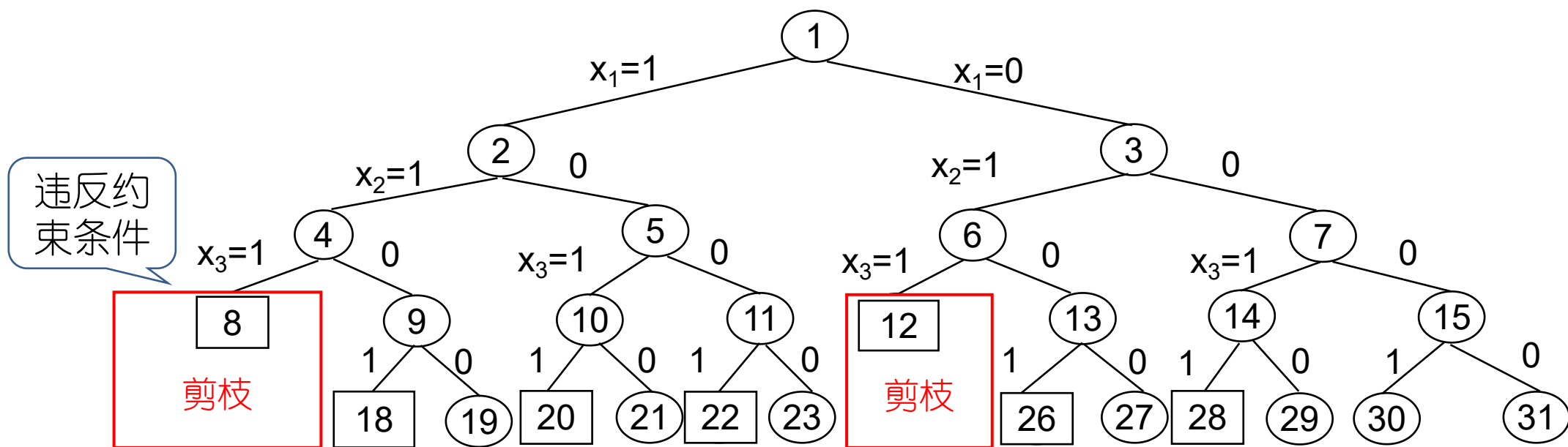
$$u(X) \leq c(X) \leq \hat{c}(X)$$

- $\hat{c}(X) = p + pp + \Delta p_j$
- $u(X) = p + pp$
- 满足 $u(X) \leq c(X) \leq \hat{c}(X)$, X是叶结点时, 三个函数值相等



约束函数B

- $n=4$, $M=10$, $(p_1, p_2, p_3, p_4)=(10, 10, 14, 18)$, $(w_1, w_2, w_3, w_4)=(2, 4, 7, 10)$
- 状态结点按层次遍历顺序编号, 15个非叶结点+16个叶结点
- 方形结点表示被约束函数B剪枝: 选中的物品重量和大于M

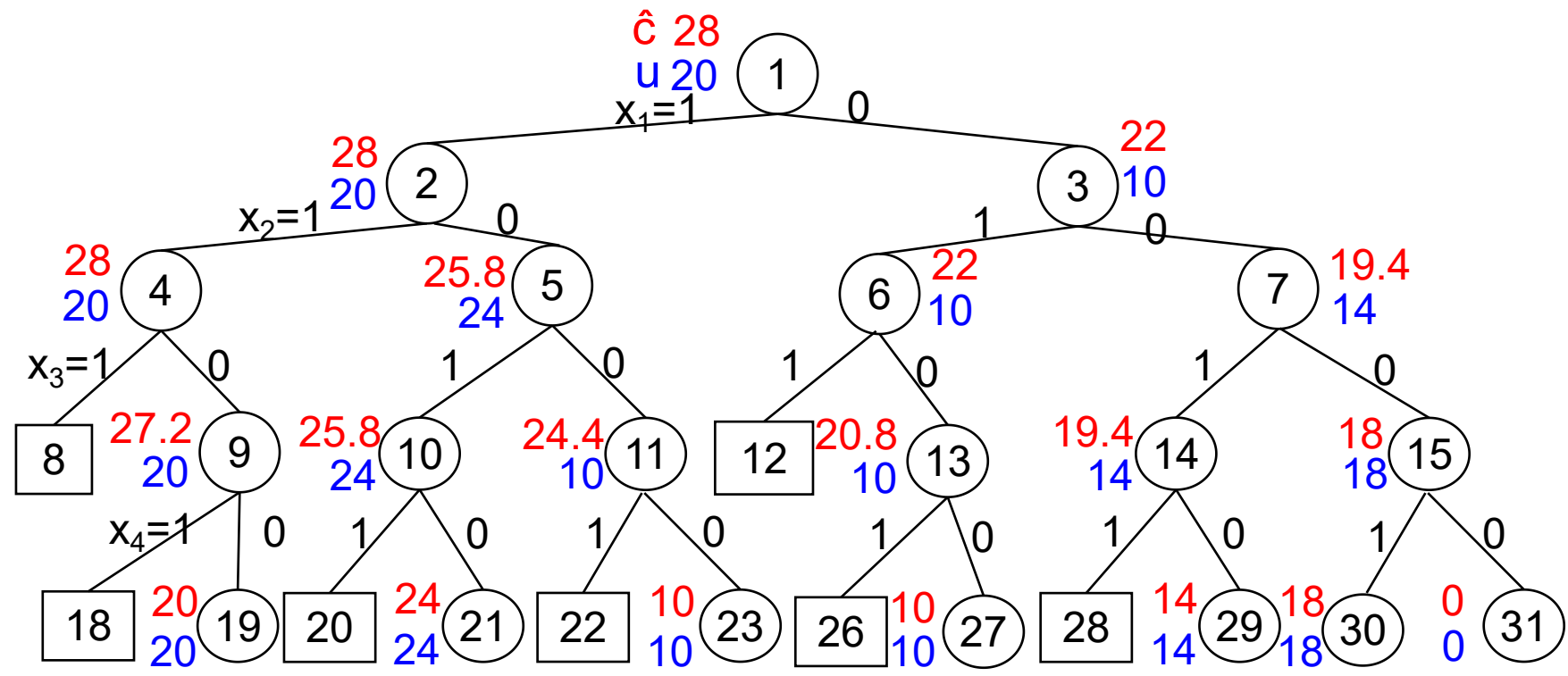


成本估计函数 $\hat{c}(X)$ 和 $u(X)$

- $n=4$, $M=10$, $(p_1, p_2, p_3, p_4)=(10, 10, 14, 18)$, $(w_1, w_2, w_3, w_4)=(2, 4, 7, 10)$

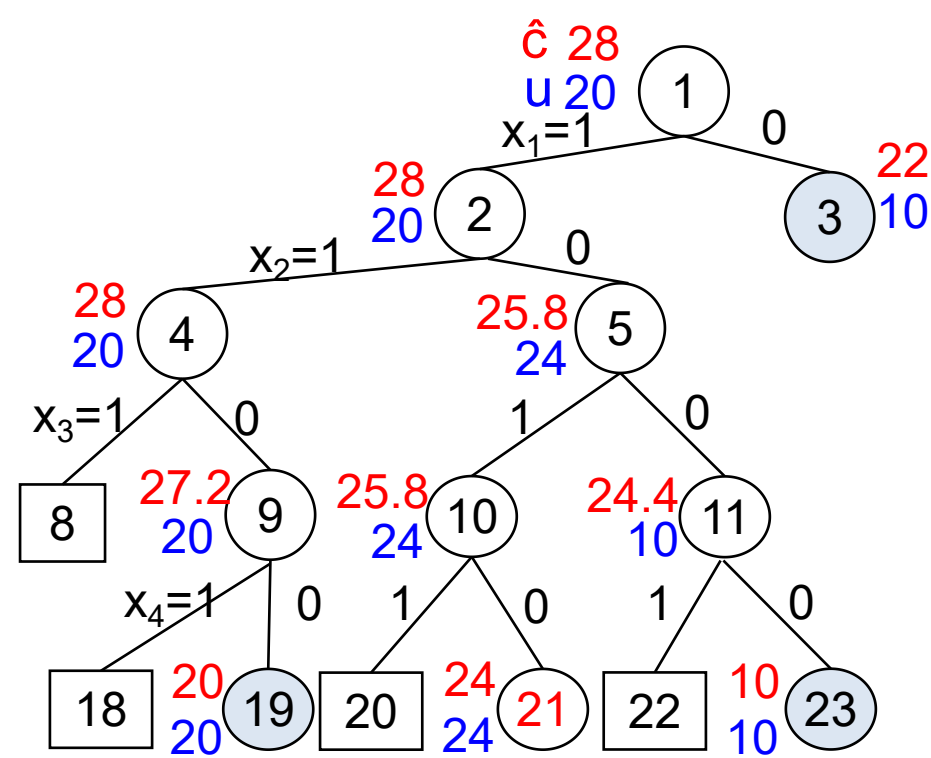
贪心解向量

结点 编号	x_1	x_2	x_3	x_4
1	1	1	4/7	0
2	1	1	4/7	0
3	0	1	6/7	0
4	1	1	4/7	0
5	1	0	1	1/10
6	0	1	6/7	0
7	0	0	1	3/10



LC-分支限界法实例运行

$\epsilon=-0.1$, U 不断增大; $\hat{c}(X)\leq U$ 时, X 被杀死



堆顶结点 $\hat{c}(21)\leq U$, 算法结束。
此时 U 等于24, ans 等于21。

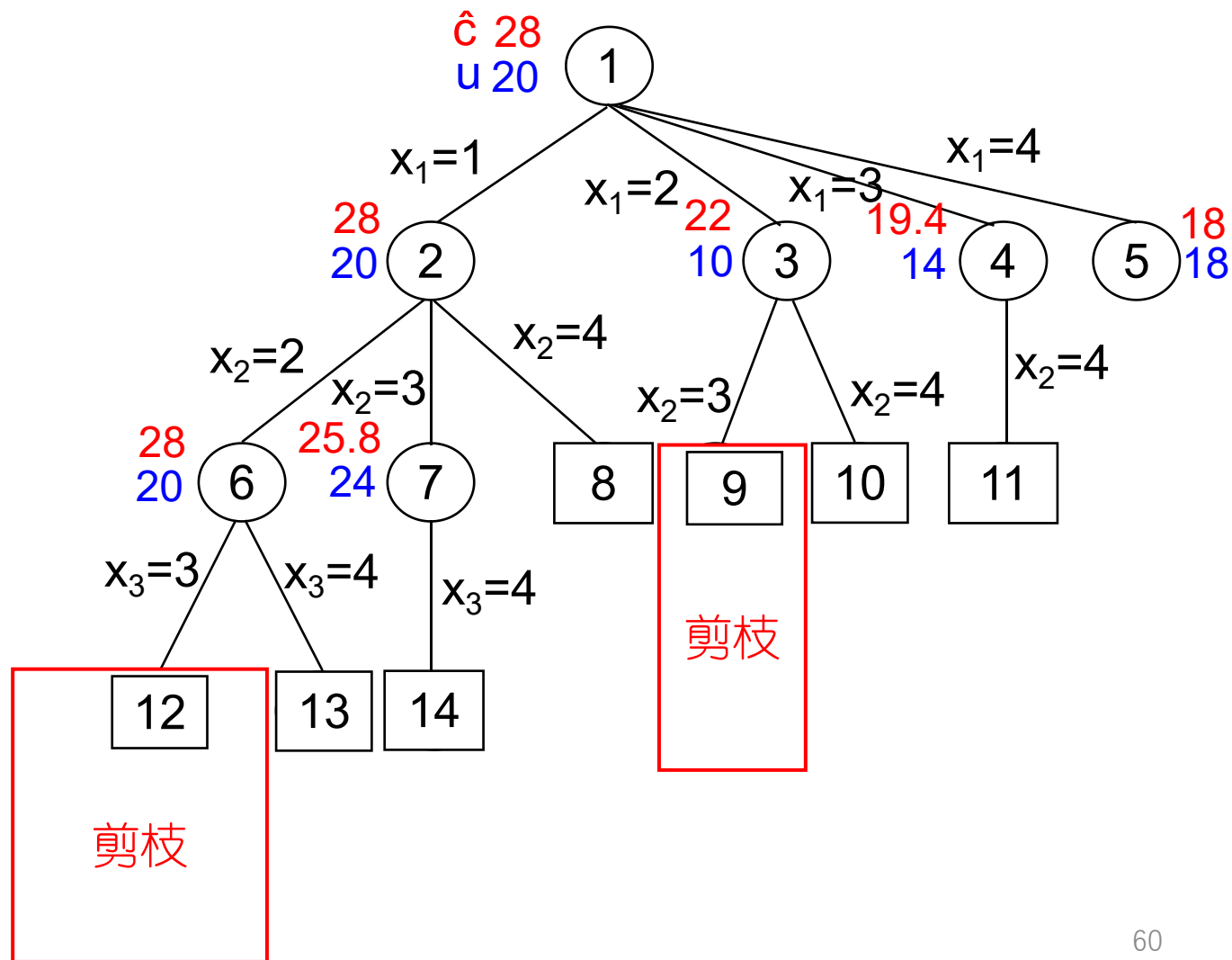
E	儿子X	U	ans	活结点表
1		19.9		
	2 3			<u>2</u> 3
2		19.9		
	4 5	23.9		3 <u>4</u> 5
4		23.9		
	8弃 9			3 5 <u>9</u>
9		23.9		
	18弃 19弃			3 <u>5</u>
5		23.9		
	10 11			3 <u>10</u> 11
10		23.9		
	20弃 21	24	21	3 <u>11</u> 21
11		24	21	
	22弃 23弃			3 <u>21</u>

K-元组表示下的0/1背包问题实例

- $n=4$, $M=10$, $(p_1, p_2, p_3, p_4)=(10, 10, 14, 18)$, $(w_1, w_2, w_3, w_4)=(2, 4, 7, 10)$

贪心解向量

结点编号	x_1	x_2	x_3	x_4
1	1	1	4/7	0
2	1	1	4/7	0
3	0	1	6/7	0
4	0	0	1	3/10
5	0	0	0	1
6	1	1	4/7	0
7	1	0	1	1/10



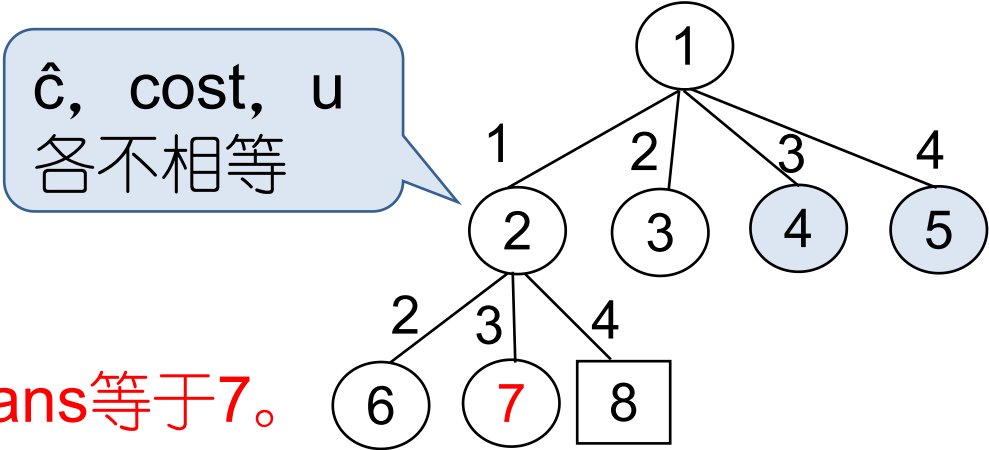
LC-分支限界法实例运行

- $n=4, M=10, (p_1, p_2, p_3, p_4)=(10, 10, 14, 18), (w_1, w_2, w_3, w_4)=(2, 4, 7, 10)$

$\epsilon=-0.1, U$ 不断增大; $\hat{c}(X)\leq U$ 时, X 被杀死

E	儿子X	U	ans	活结点表
1		19.9	1	
	2			
	3			
	4			弃
	5			弃
<hr/>				
2		19.9		
	6	20	6	
	7	24	7	3 6 <u>7</u>
	8			舍

编号	1	2	3	4	5	6	7
\hat{c}	28	28	22	19.4	18	28	25.8
cost	0	10	10	14	18	20	24
u	20	20	10	14	18	20	24



堆顶结点 $\hat{c}(7)\leq U$, 算法结束。此时 U 等于24, ans 等于7。

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
 - 都会借助约束函数**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**c**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

- 分支限界法求极小化问题，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
 - 约束条件作为约束函数 B
 - 设计成本估计函数 $\hat{c}(X)$ ， $\hat{c}(X) \leq c(X)$
 - 设计最小成本的上界 U ， $c(X) \leq U$
 - 基于 $\hat{c}(X)$ 和 U 进行分支限界搜索
- 分支限界法求极大化问题
 - 目标函数取相反数，转化为极小化问题
 - 或对照极小化问题做镜像修改
- 分支限界法中的剪枝
 - 约束函数 B ：限定是否存在可行解
 - 成本估计函数 $\hat{c}(X)$ 和界 U ：界定是否存在最优解

- 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 \hat{c} 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 \hat{c} 设计
- 8.4 求最小成本的分支限界法
 - 掌握基于 $\hat{c}(X)$ 和U优化算法的一般思想，掌握分支限界法求最优解问题的一般算法FIFOBB和LCBB

- 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、约束函数 B 的设计思想，掌握 $\hat{c}(X)$ 和 U 的设计方法，掌握求最优解问题的分支限界算法

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

