搜索求解2

主讲： 王亚星、刘夏雷、郭春乐
南开大学计算机学院

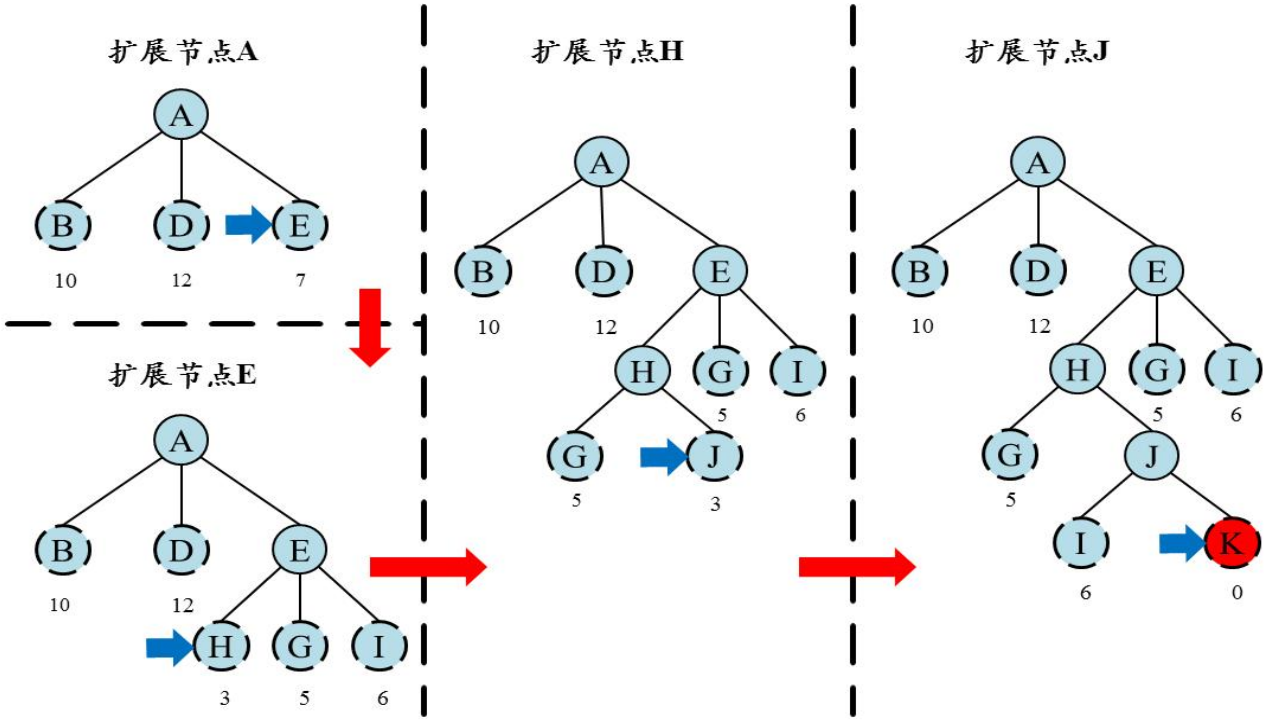
致谢：本课件主要内容来自浙江大学吴飞教授、
南开大学程明明教授

提纲

- 搜索算法基础
- 启发式搜索
- 对抗搜索
- 蒙特卡洛树搜索

内容回顾：贪婪最佳优先搜索

- **贪婪最佳优先搜索：评价函数$f(n)$=启发函数$h(n)$**
  - 例：启发函数为任意一个城市与终点城市K之间的直线距离



贪婪最佳优先搜索的过程

算法找到了一条从起始结点到终点结点的路径A → E → H → J → K，但这条路径并不是最短路径，实际上最短路径为A → E → G → K。

内容回顾：A*算法

- **评价函数：** $f(n) = g(n) + h(n)$
  - $g(n)$表示从起始结点到结点n的开销代价值
  - $h(n)$表示从结点n到目标结点路径中所估算的最小开销代价值
  - $f(n)$可视为经过结点n、具有最小开销代价值的路径。

$$\underbrace{f(n)}_{\text{评价函数}} = \underbrace{g(n)}_{\substack{\text{起始结点到结点}n\text{代价} \\ \text{(当前最小代价)}}} + \underbrace{h(n)}_{\substack{\text{结点}n\text{到目标结点代价} \\ \text{(后续估计最小代价)}}}$$

对抗搜索

- 对抗搜索(Adversarial Search)也称为博弈搜索(Game Search)
- 在一个竞争的环境中，智能体(agents)之间通过竞争实现相反的利益，一方**最大化**这个利益，另外一方**最小化**这个利益。

对抗搜索：主要内容

- **最小最大搜索(Minimax Search)**
  - 最小最大搜索是在对抗搜索中最为基本的一种让玩家来计算最优策略的方法
- **Alpha-Beta剪枝搜索(Pruning Search)**
  - 一种对最小最大搜索进行改进的算法，即在搜索过程中可剪除无需搜索的分支节点，且不影响搜索结果。
- **蒙特卡洛树搜索(Monte-Carlo Tree Search)**
  - 通过采样而非穷举方法来实现搜索。

对抗搜索

- **本课程目前主要讨论在确定的、全局可观察的、竞争对手轮流行动、零和游戏(zero-sum)下的对抗搜索**
  - 所谓零和博弈是博弈论的一个概念，属非合作博弈。指参与博弈的各方，在严格竞争下，一方的收益必然意味着另一方的损失，博弈各方的收益和损失相加总和永远为“零”，双方不存在合作的可能。与“零和”对应，“双赢博弈”的基本理论就是“利己”不“损人”，通过谈判、合作达到皆大欢喜的结果。

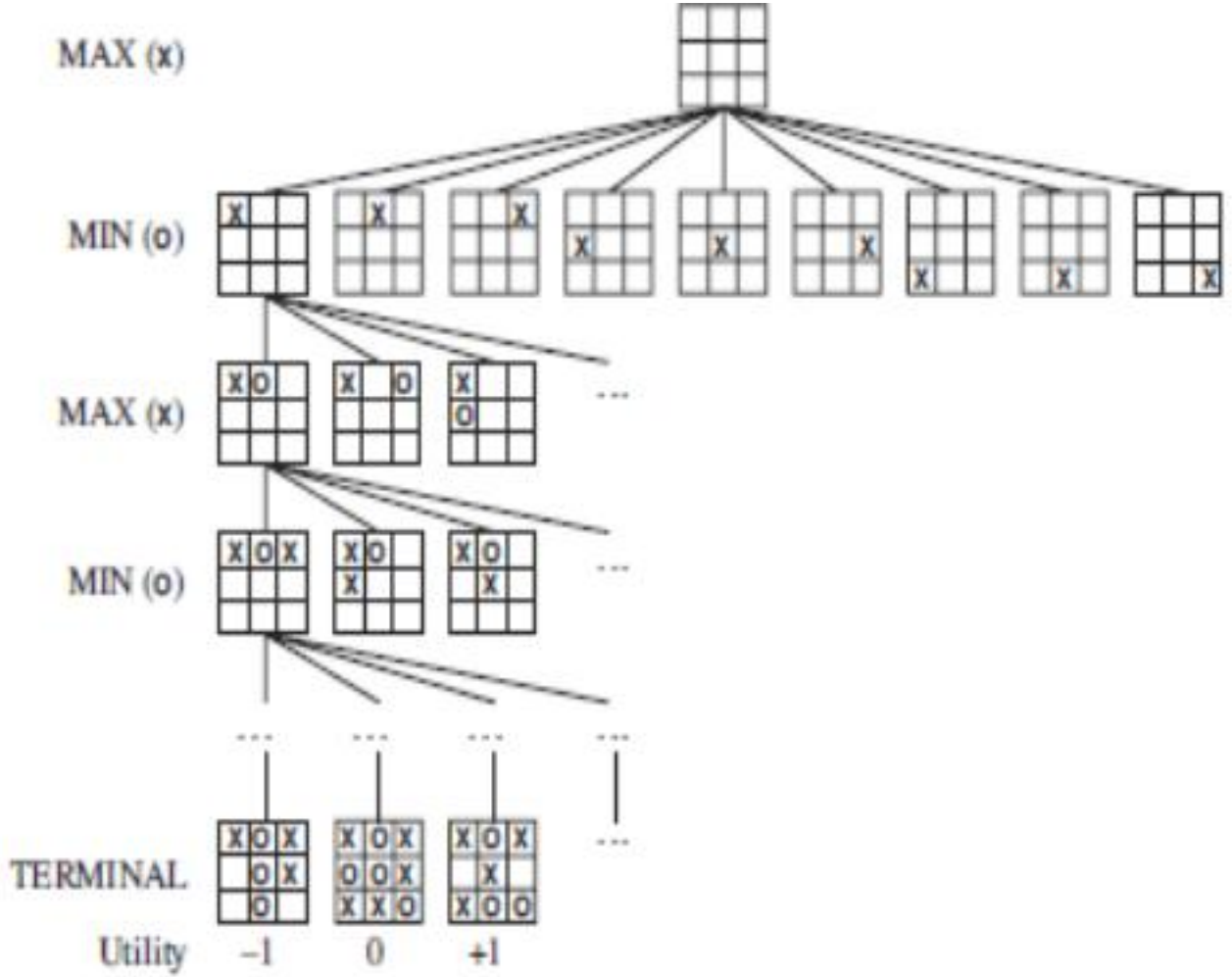
对抗搜索

- 本课程目前主要讨论在确定的、全局可观察的、竞争对手轮流行动、零和游戏(zero-sum)下的对抗搜索
- 两人对决游戏 (MAX and MIN, MAX先走) 可如下形式化描述，从而将其转换为对抗搜索问题

初始状态S_0	游戏所处于的初始状态
玩家$PLAYER(s)$	在当前状态s下，该由哪个玩家采取行动
行动$ACTIONS\ (s)$	在当前状态s下所采取的可能移动
状态转移模型$RESULT\ (s,a)$	在当前状态s下采取行动a后得到的结果
终局状态检测$TERMINAL-TEST\ (s)$	检测游戏在状态s是否结束
终局得分$UTILITY\ (s,p)$	在终局状态s时，玩家p的得分。

对抗搜索

- **Tic-Tac-Toe游戏的对抗搜索**
  - MAX先行，可在初始状态的9个空格中任意放一个X
  - MAX希望游戏终局得分高、MIN希望游戏终局得分低
  - 所形成游戏树的叶子结点有9! = 362,880,国际象棋的叶子节点数为10^{40}



Tic-Tac-Toe中部分搜索树

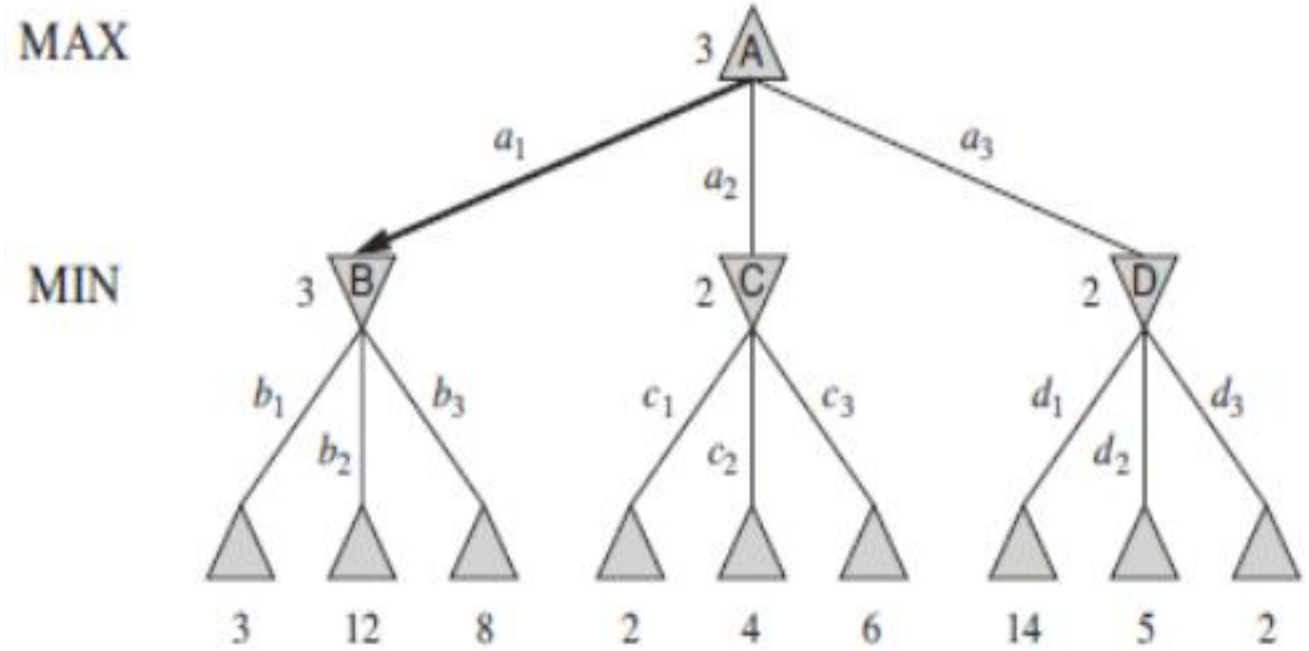
对抗搜索：minimax算法

- **给定一个游戏搜索树，minimax算法通过每个节点的minimax值来决定最优策略。**
  - MAX希望最大化minimax值，而MIN则相反

$$\textsc{Minimax}(s) = \begin{cases} \textsc{Utility}(s) & \text{if } \textsc{Terminal-Test}(s) \\ \max_{a \in Actions(s)} \textsc{Minimax}(\textsc{Result}(s,a)) & \text{if } \textsc{Player}(s) = \textsc{max} \\ \min_{a \in Actions(s)} \textsc{Minimax}(\textsc{Result}(s,a)) & \text{if } \textsc{Player}(s) = \textsc{min} \end{cases}$$

对抗搜索：minimax算法

- **给定一个游戏搜索树，minimax算法通过每个节点的minimax值来决定最优策略。**
  - 通过minimax算法，我们知道，对于MAX而言采取a_1行动是最佳选择，因为这能够得到最大minimax值(收益最大)。



对抗搜索：minimax算法

- **给定一个游戏搜索树，minimax算法通过每个节点的minimax值来决定最优策略。**
  - 通过minimax算法，我们知道，对于MAX而言采取a_1行动是最佳选择，因为这能够得到最大minimax值(收益最大)。

对抗搜索：minimax算法

- **Complete ?**
  - Yes (if tree is finite)
- **Optimal?**
  - Yes (against an optimal opponent)
- **Time complexity ?**
  - $O(b^m)$
  - m 是游戏树的最大深度
  - 在每个节点存在b个有效走法
- **Space complexity ?**
  - $O(b \times m)$ (depth-first exploration)

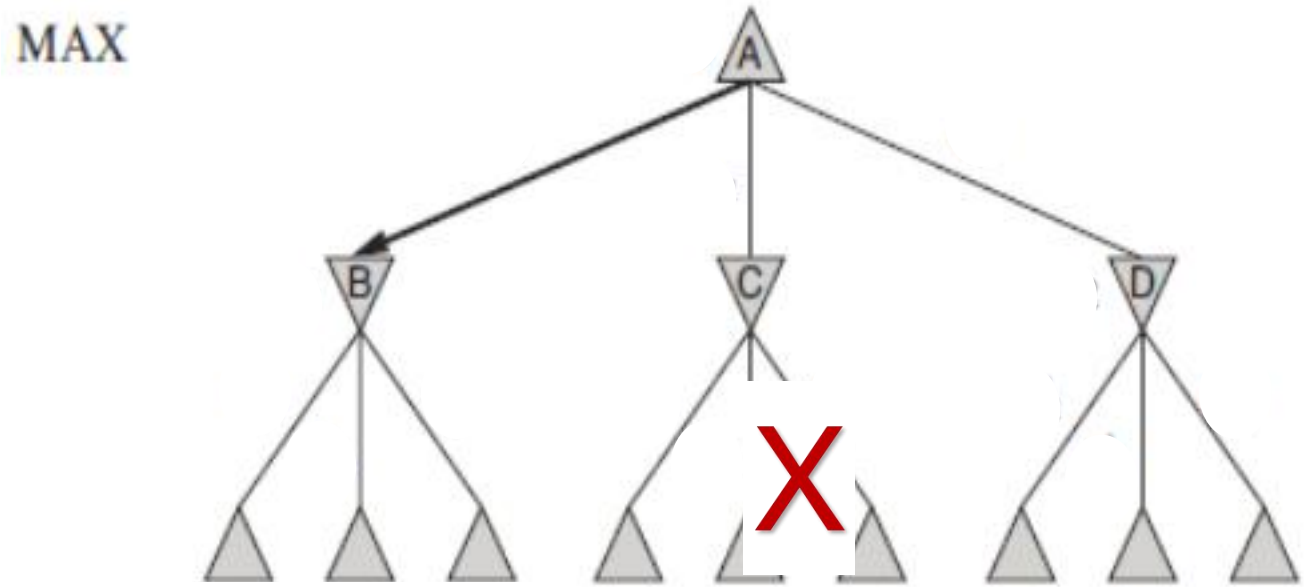
For chess, b ≈ 35, m ≈100
for "reasonable" games
→ exact solution
completely infeasible

对抗搜索：minimax算法

- **优点:**
  - 算法是一种简单有效的对抗搜索手段
  - 在对手也“尽力而为”前提下，算法可返回最优结果
- **缺点:**
  - 如果搜索树极大，则无法在有效时间内返回结果
- **改善:**
  - 使用alpha-beta pruning算法来减少搜索节点
  - 对节点进行采样、而非逐一搜索 (i.e., MCTS)

对抗搜索： Alpha-Beta 剪枝搜索

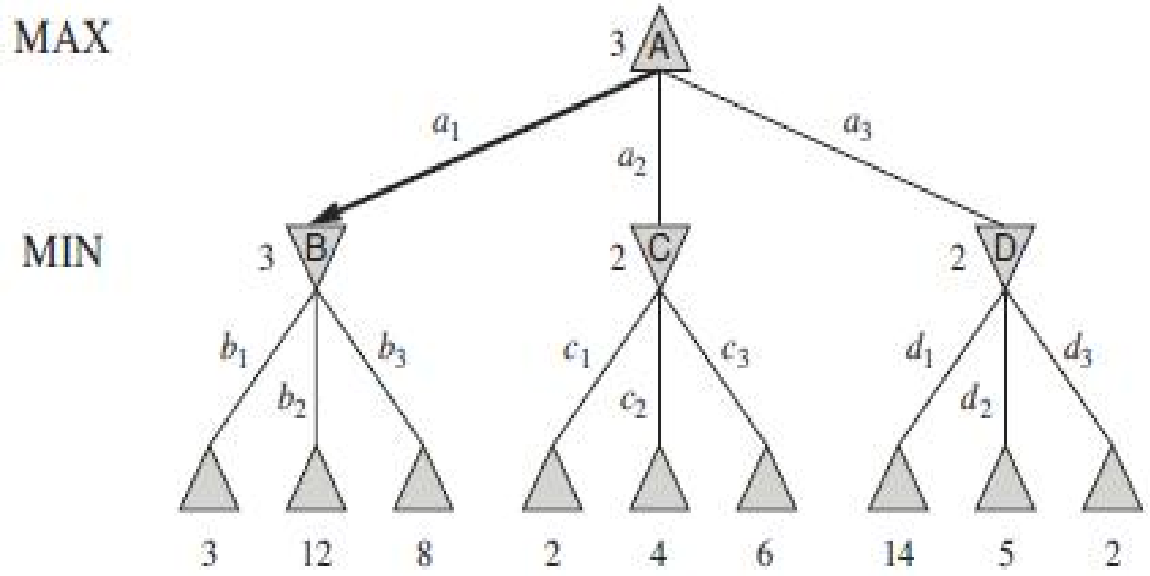
动机：



对抗搜索：Alpha-Beta 剪枝搜索

- 在极小化极大算法(minimax算法)中减少所搜索的搜索树节点数。该算法和极小化极大算法所得结论相同，但剪去了不影响最终结果的搜索分枝。

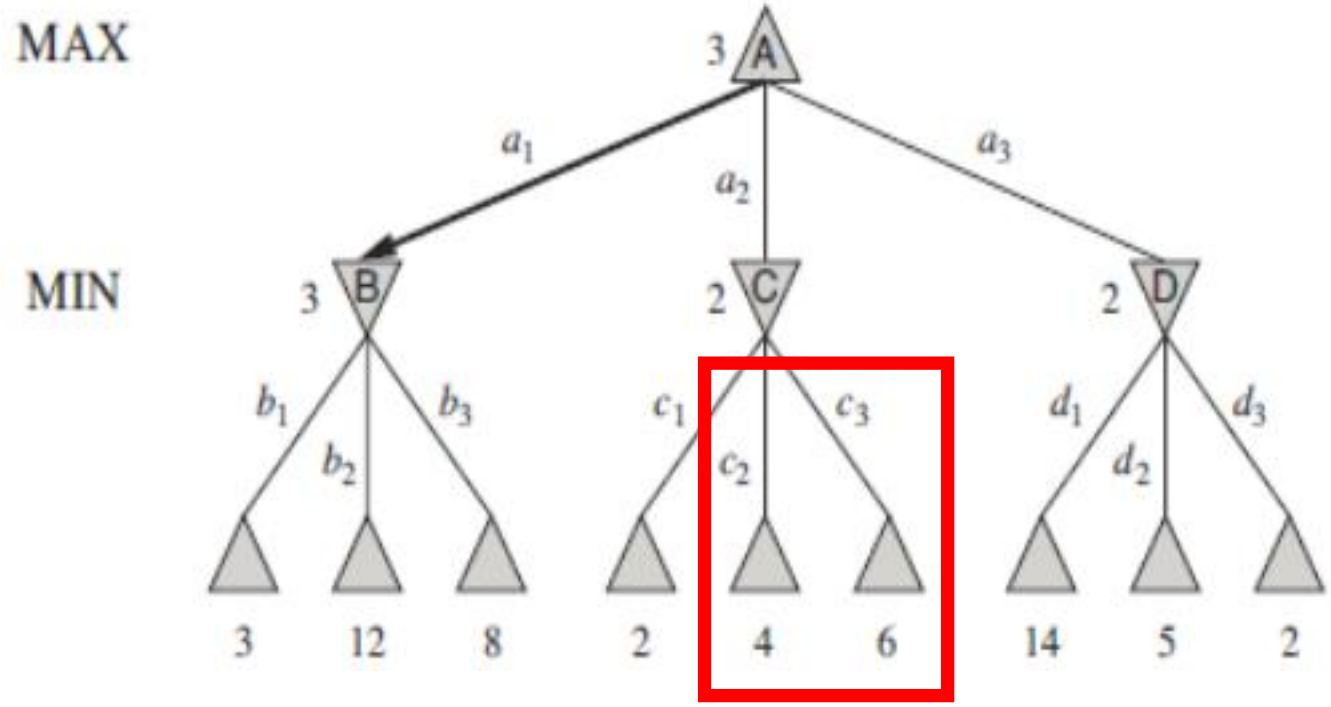
MINIMAX(*root*)



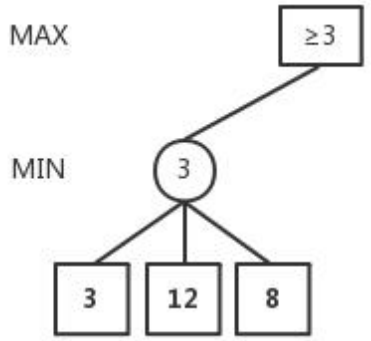
对抗搜索：Alpha-Beta 剪枝搜索

- 剪去不影响最终结果的搜索分枝。

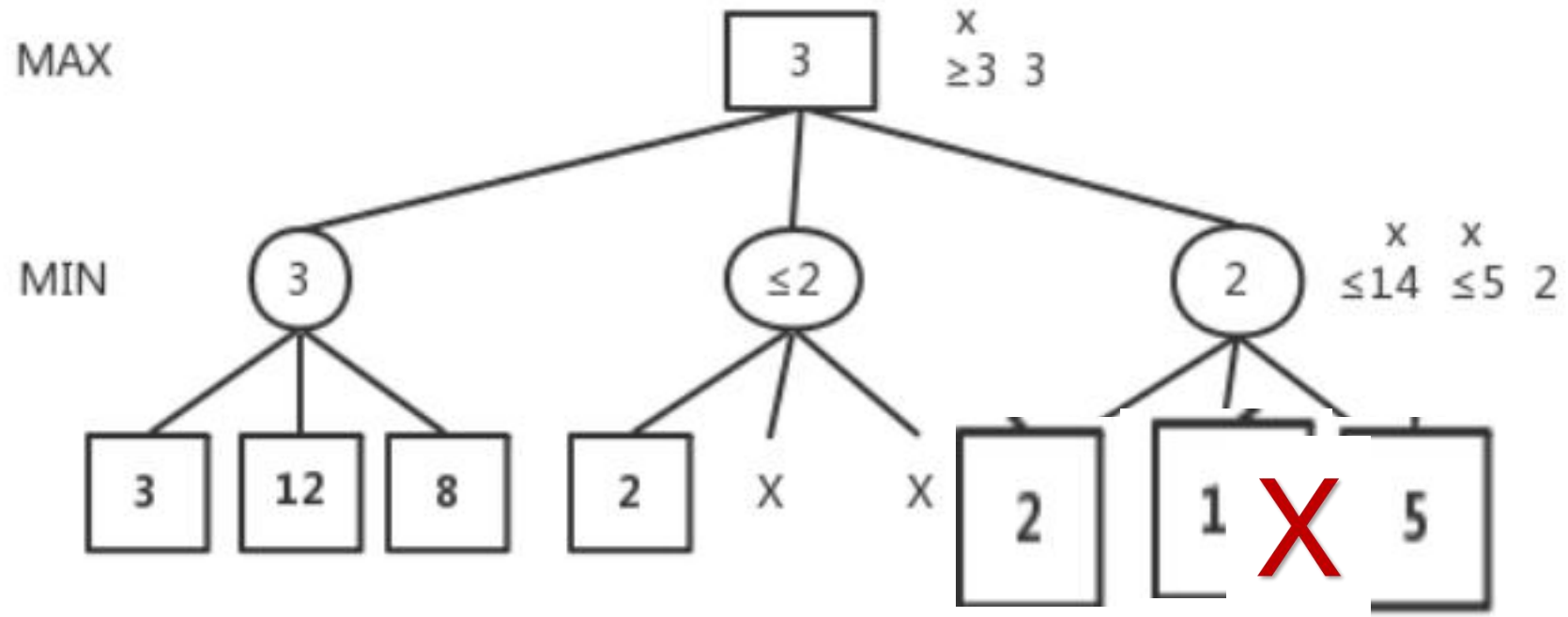
图中MIN选手所在的节点C下属分支4和6与根节点最终优化决策的取值无关，可不被访问。



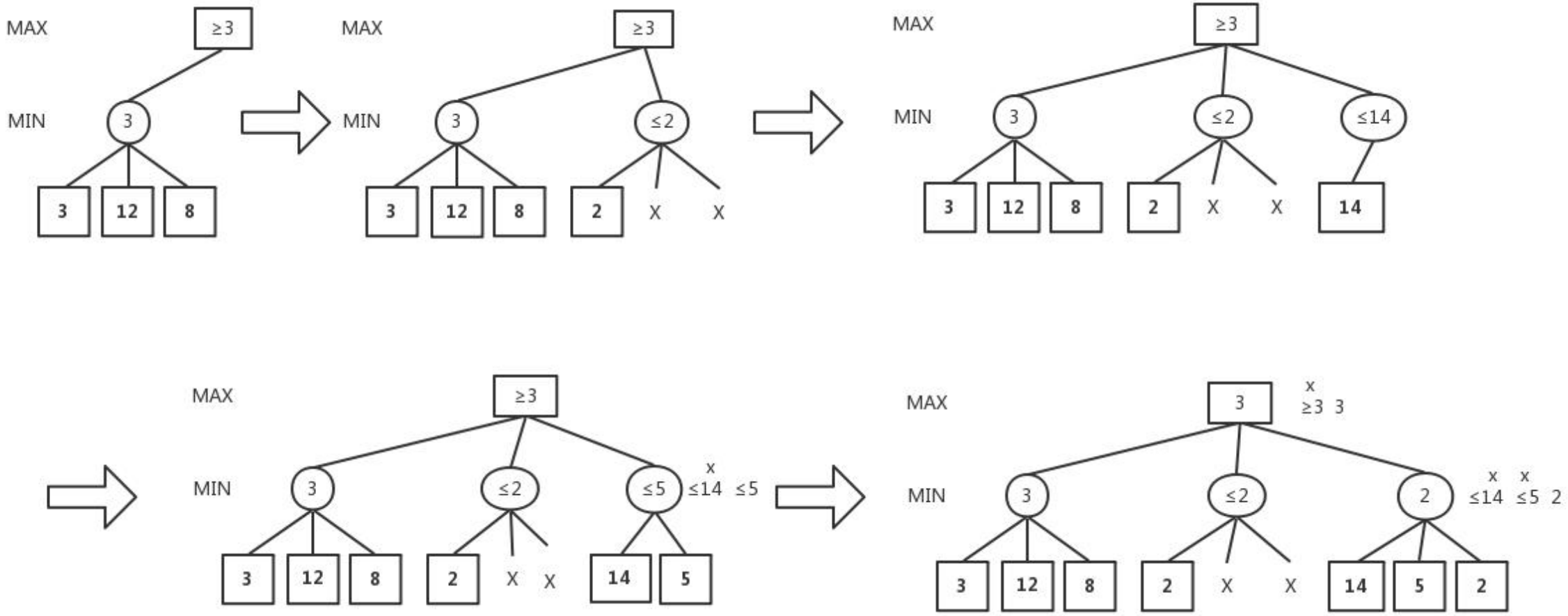
对抗搜索：Alpha-Beta 剪枝搜索



对抗搜索：Alpha-Beta 剪枝搜索



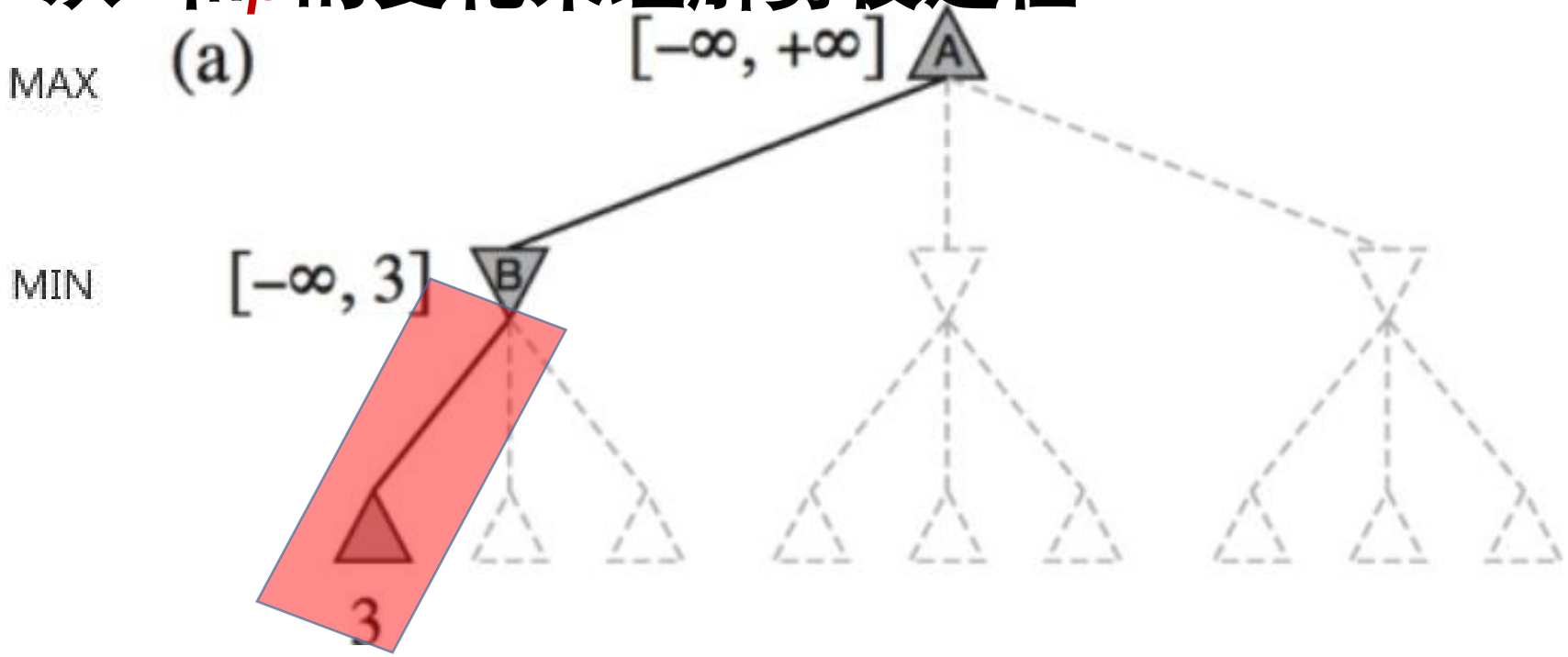
对抗搜索：Alpha-Beta 剪枝搜索



Alpha值(α)	MAX节点目前得到的最高收益
Beta值(β)	MIN节点目前可给对手的最小收益
α和β的值初始化分别设置为$-\infty$和∞	

对抗搜索：Alpha-Beta 剪枝搜索

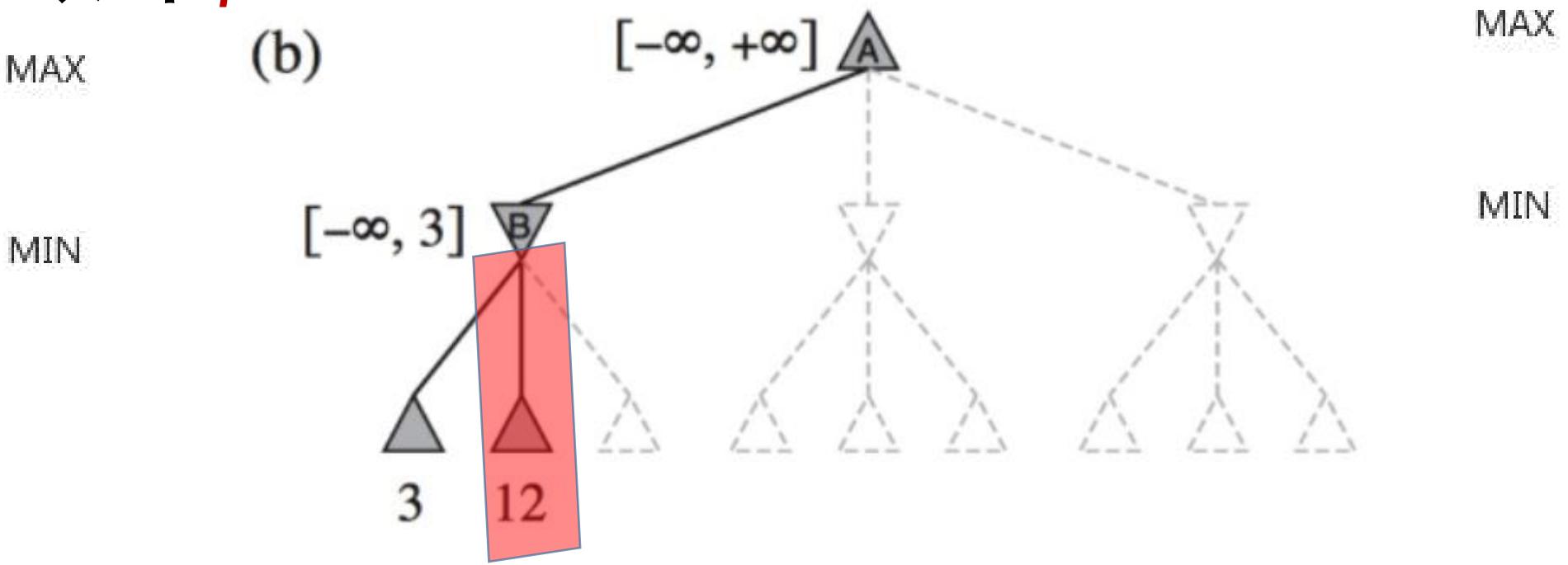
- 从α和β的变化来理解剪枝过程



- B第一个节点，其值为3，B(MIN)节点目前可给对手的最小收益，β 为3.

对抗搜索：Alpha-Beta 剪枝搜索

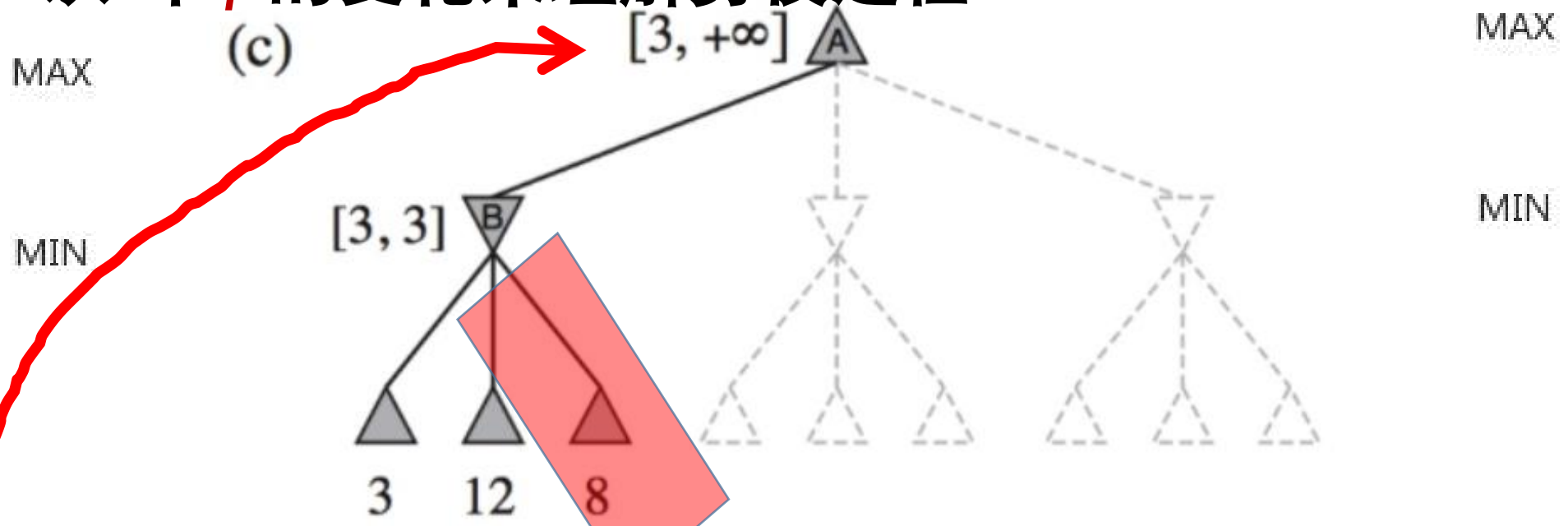
- 从α和β的变化来理解剪枝过程



- B第二个节点，其值为12(>3)，B(MIN)节点目前可给对手的最小收益，β 为3.

对抗搜索：Alpha-Beta 剪枝搜索

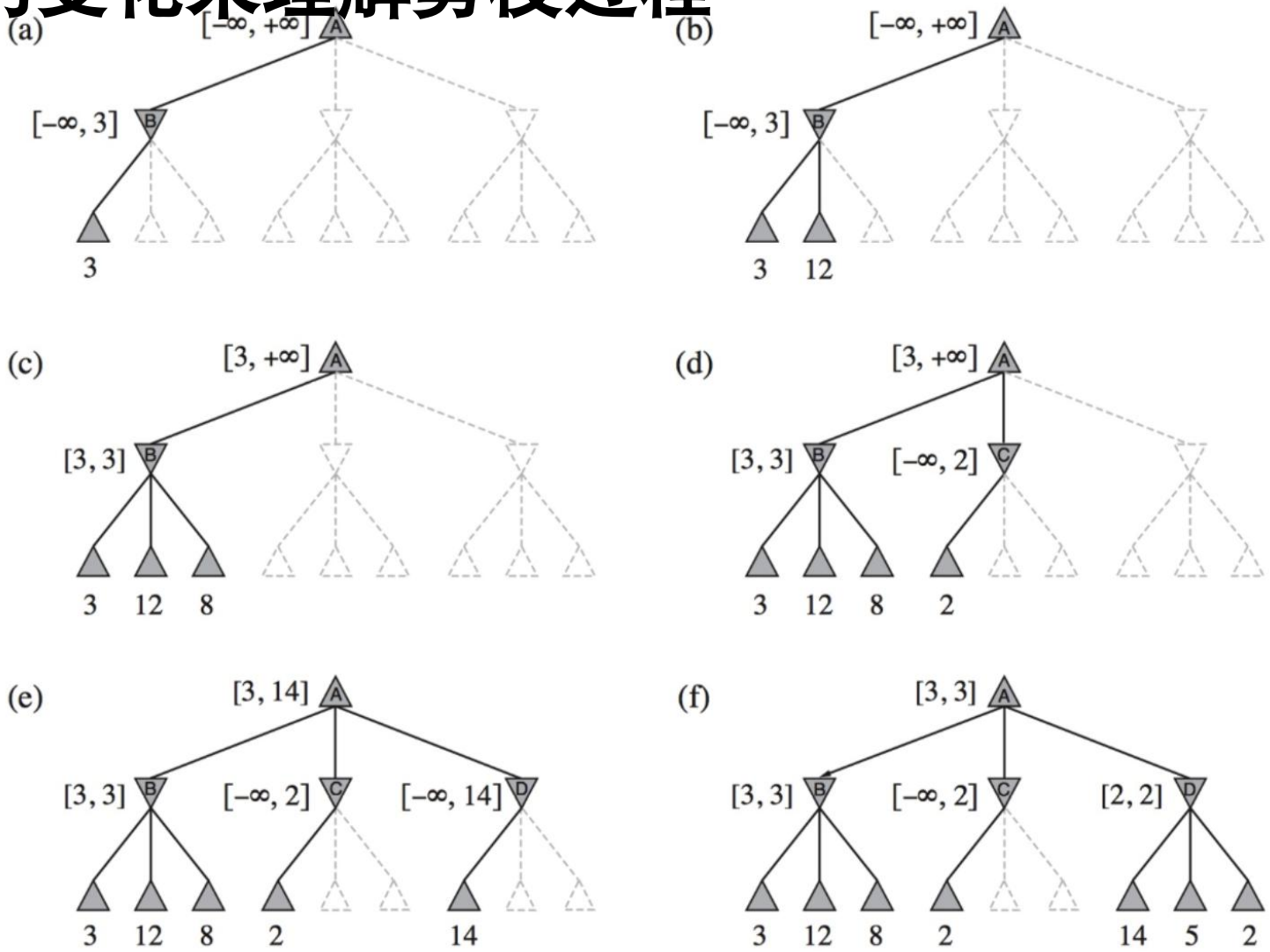
- 从α和β的变化来理解剪枝过程



- B第三个节点，其值为8(>3)，B(MIN)节点目前可给对手的最小收益，β 为3.
- A的第一个节点B，A节点目前得到的最高收益是3，α为3

对抗搜索： Alpha-Beta 剪枝搜索

- 从α和β的变化来理解剪枝过程



对抗搜索：如何利用Alpha-Beta 剪枝

Alpha值(α)	玩家MAX(根节点)目前得到的最高收益
	假设n是MIN节点，如果n的一个后续节点可提供的收益小于α，则n及其后续节点可被剪枝

对抗搜索：如何利用Alpha-Beta 剪枝

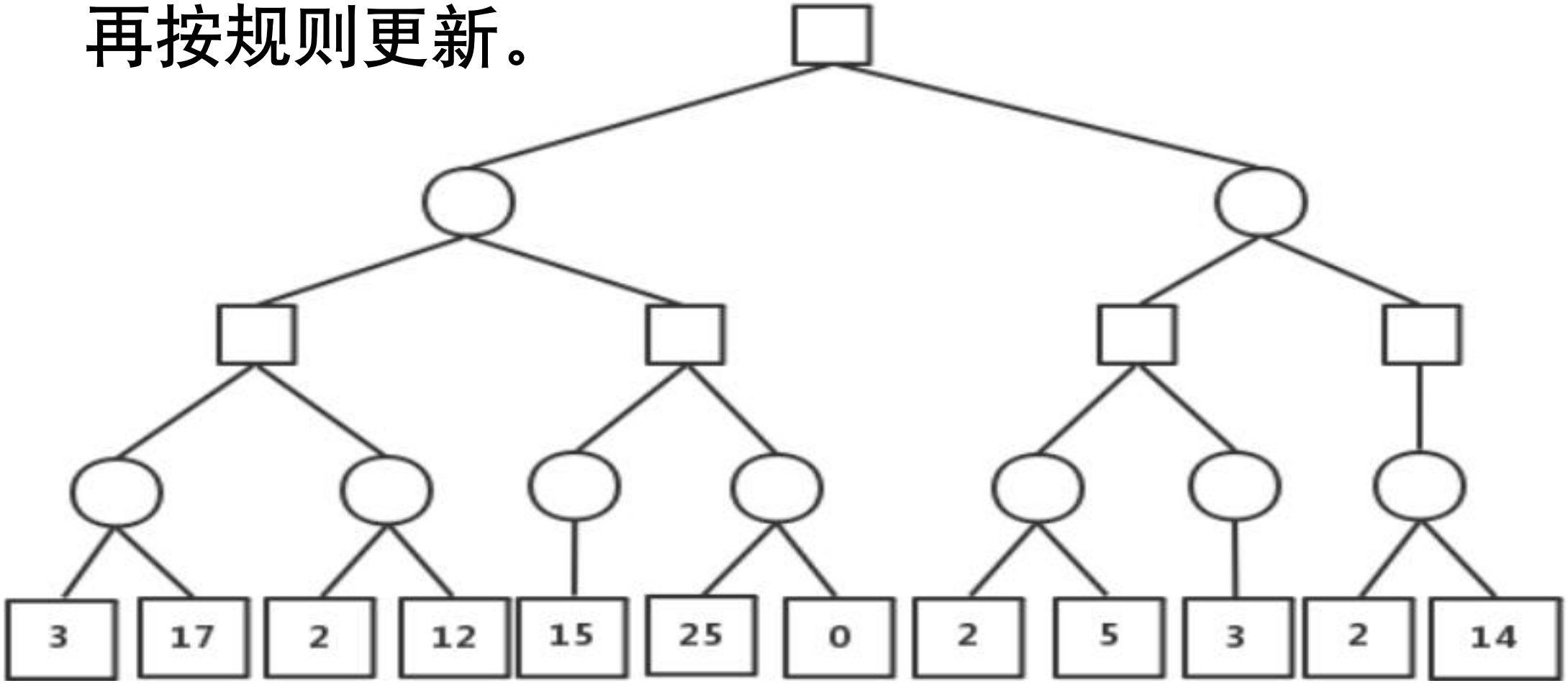
- α为可能解法的最小下界
- β为可能解法的最大上界

对抗搜索：如何利用Alpha-Beta 剪枝

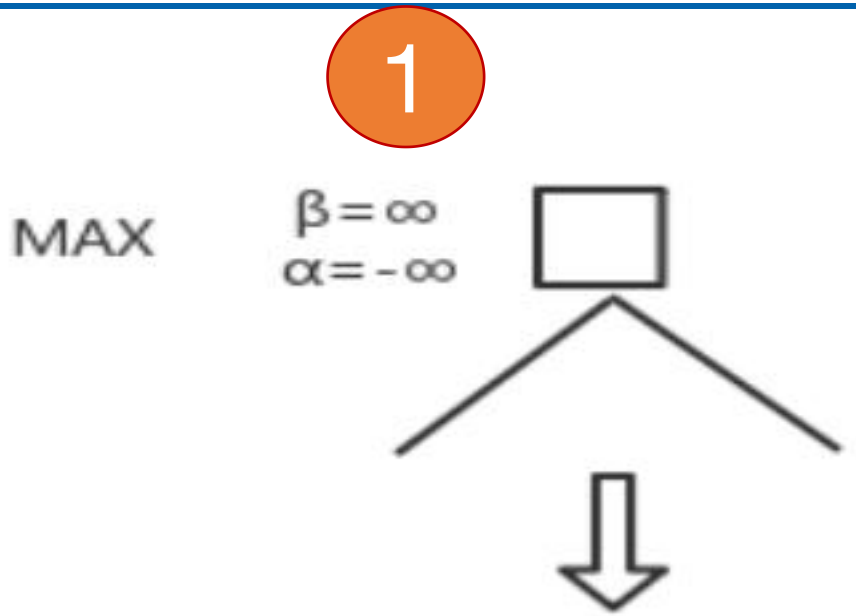
- α为可能解法的最小下界
- β为可能解法的最大上界

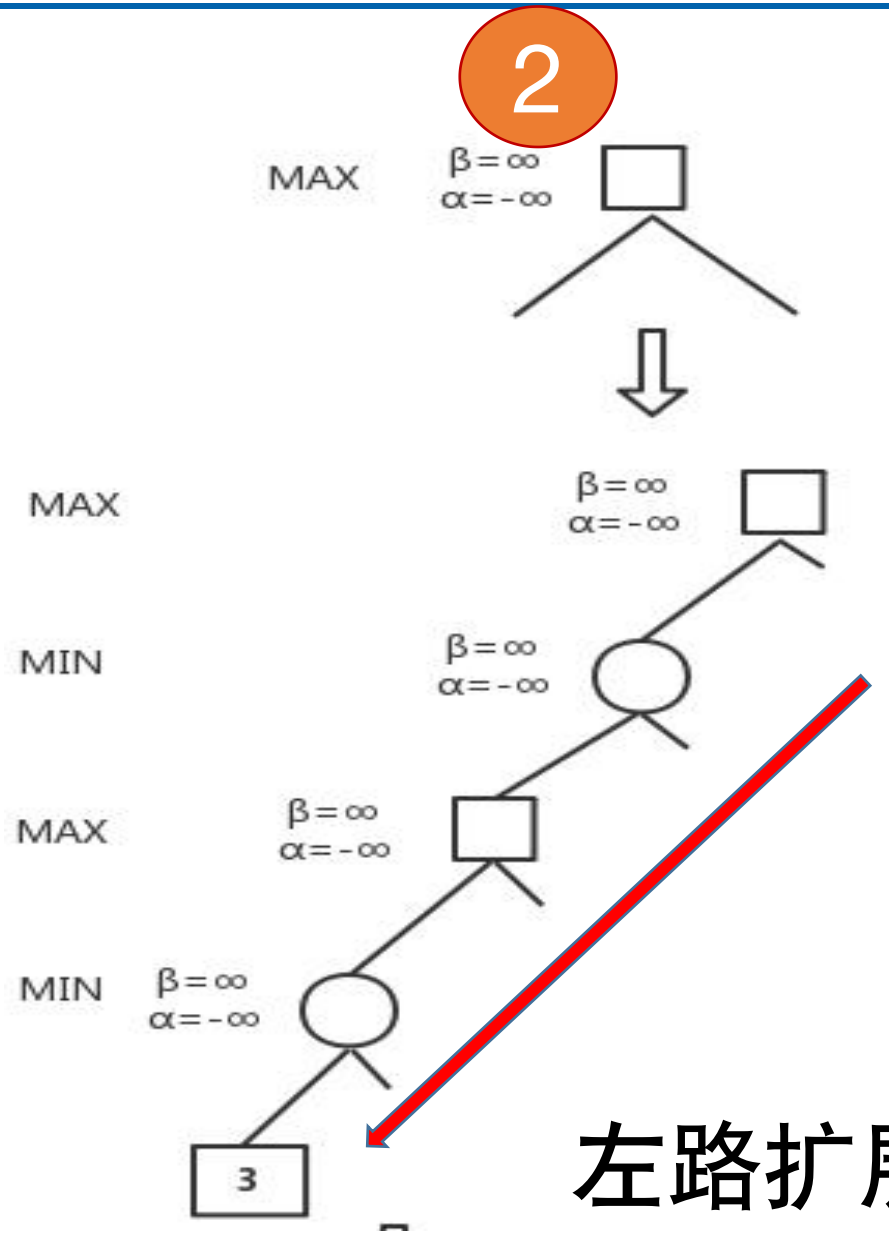
对抗搜索：Alpha-Beta 剪枝搜索示意

没有必要枚举，可以**先继承父节点**的α, β，
再按规则更新。



对抗搜索： Alpha-Beta 剪枝搜索示意

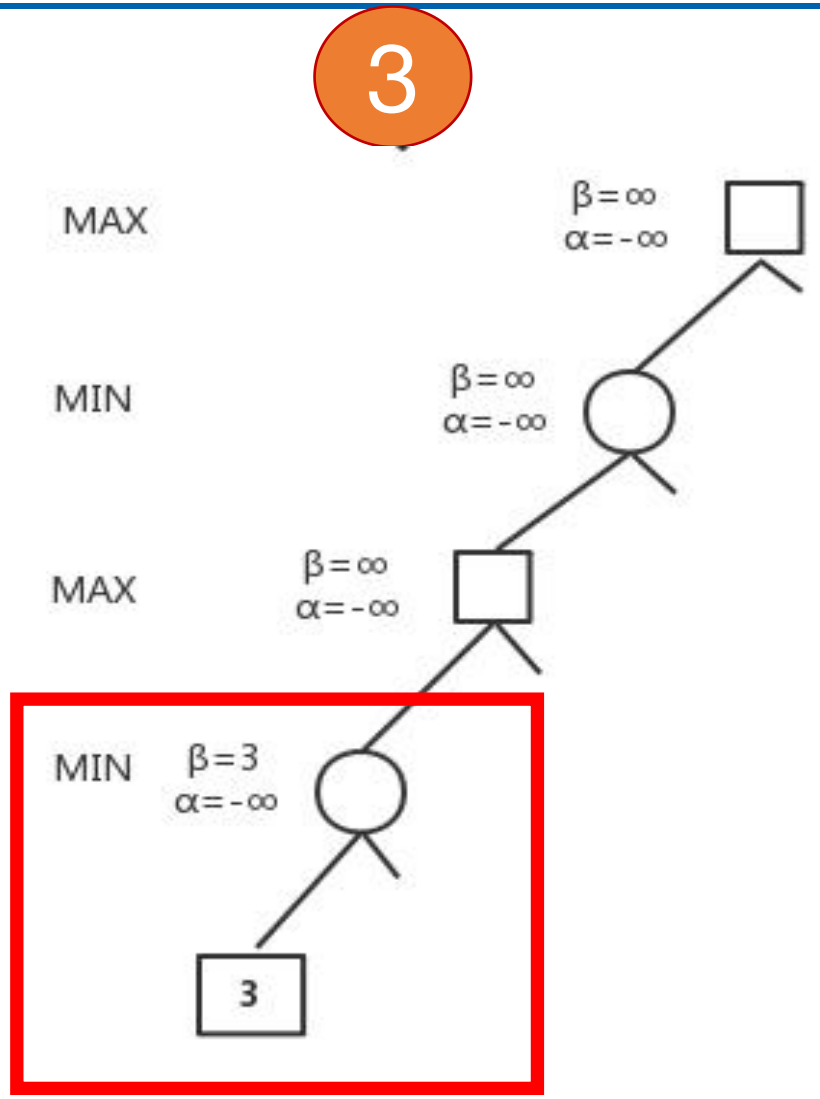


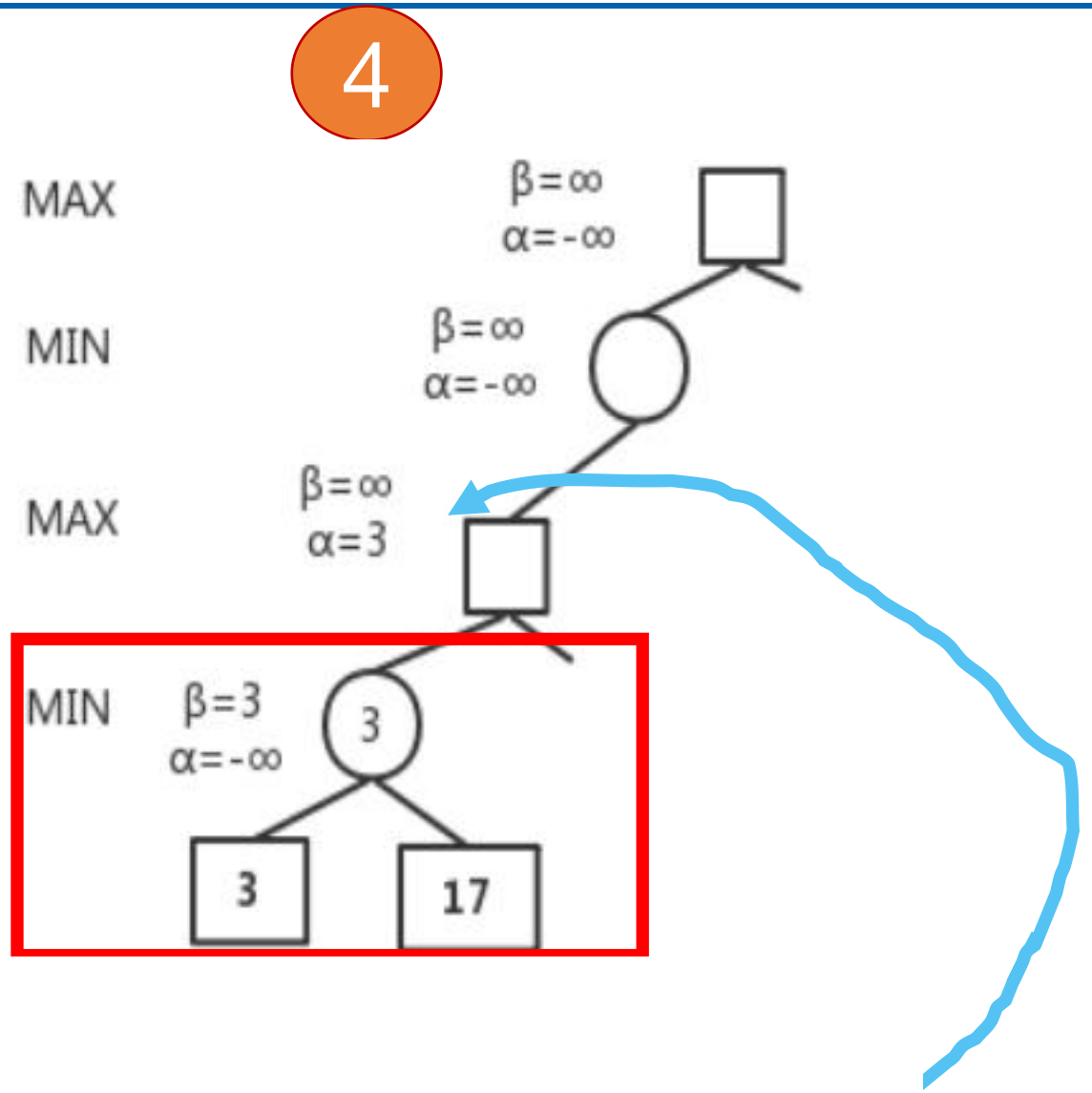


α, β赋值

左路扩展到底

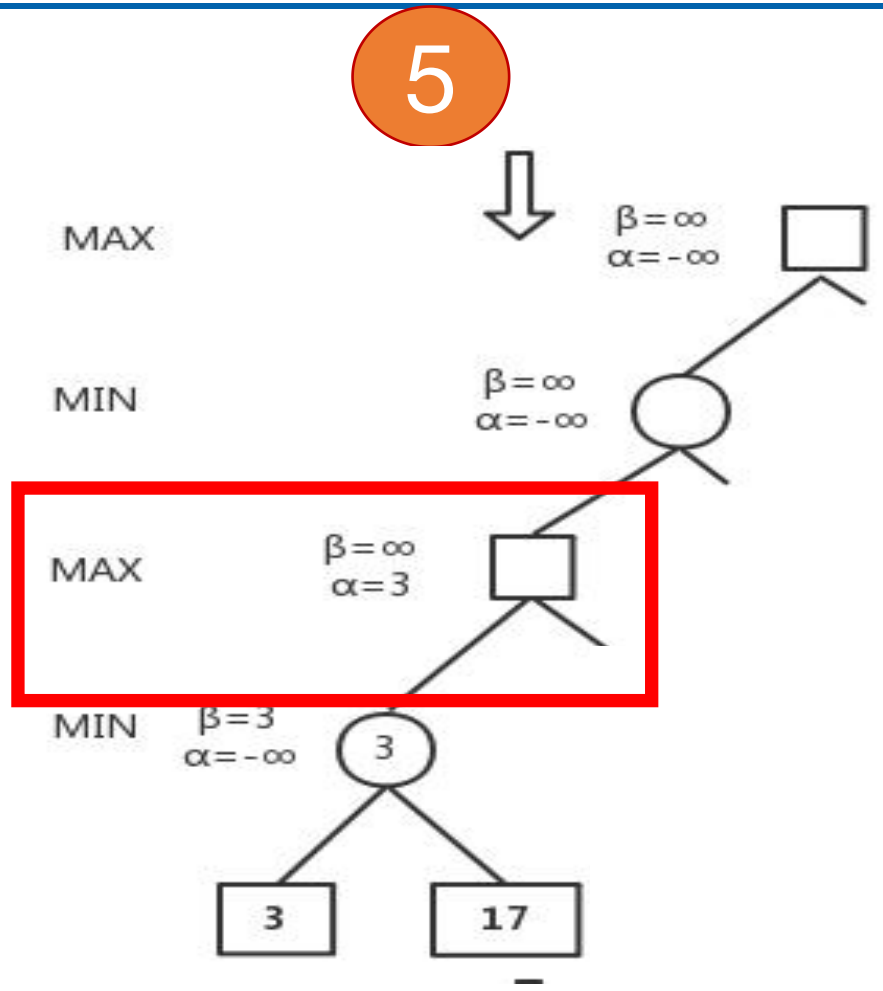
对抗搜索：Alpha-Beta 剪枝搜索示意

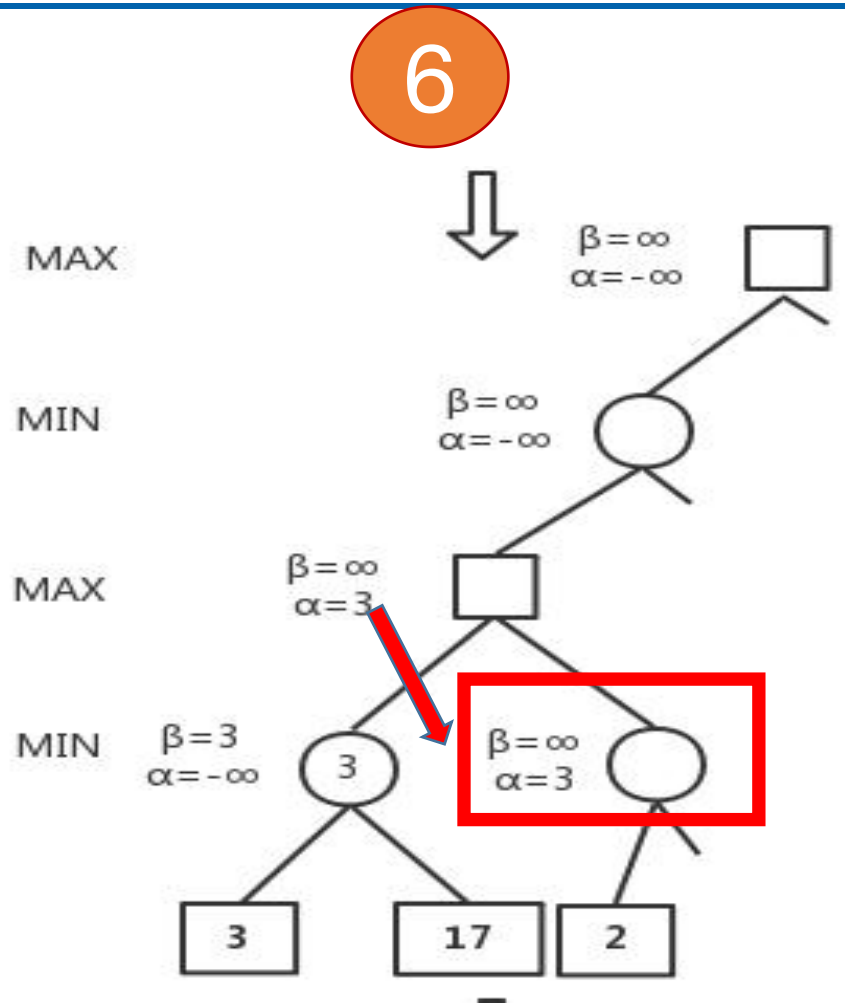




*MIN*节点，$\beta = 3$

对抗搜索：Alpha-Beta 剪枝搜索示意

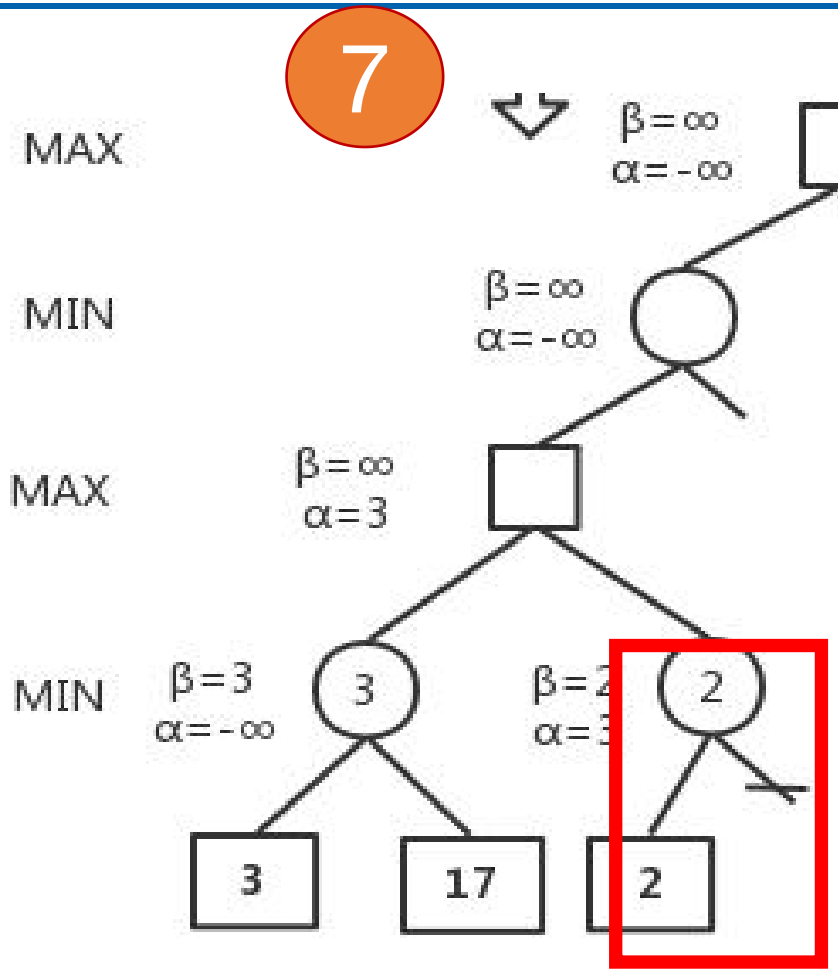


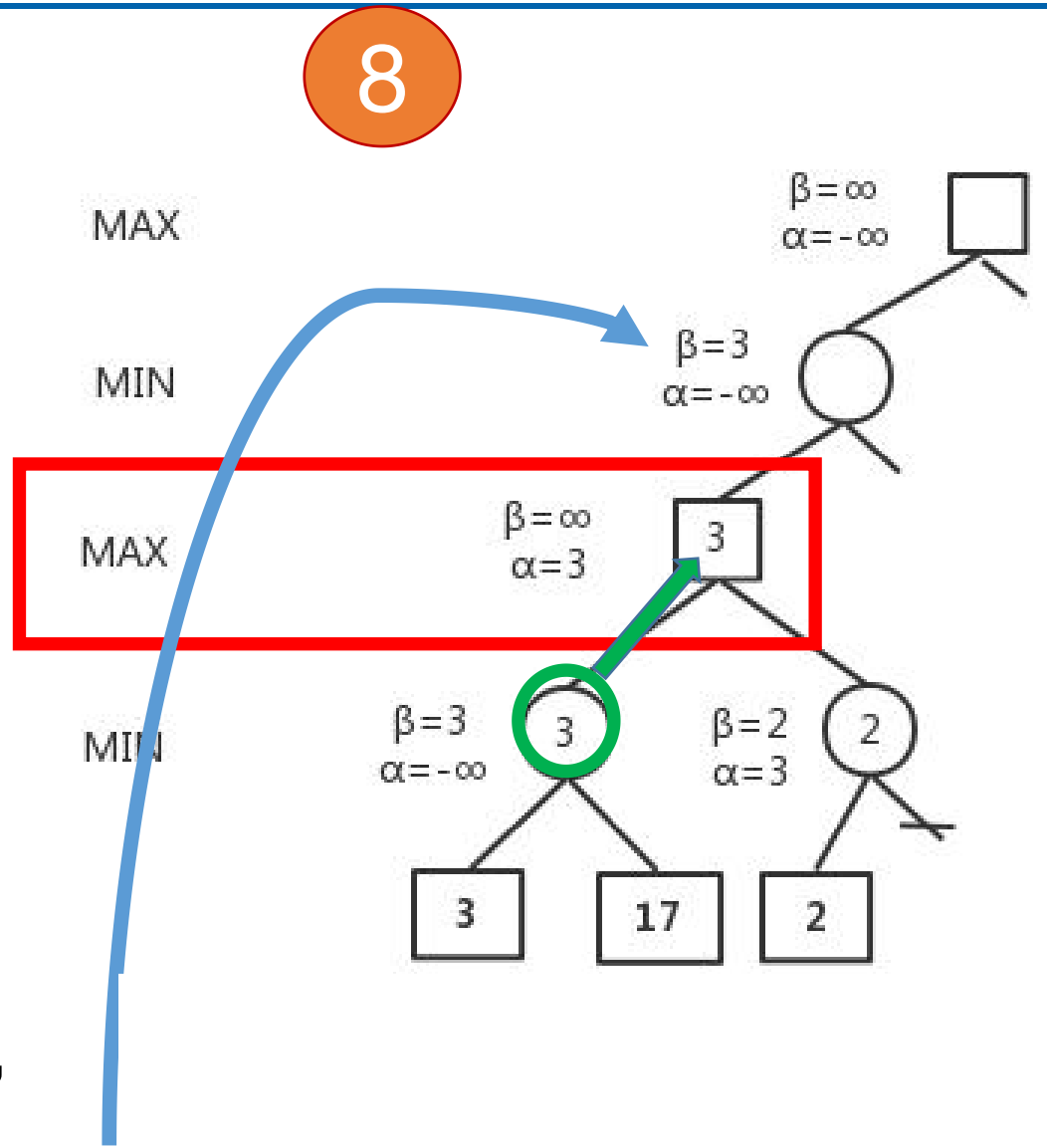


*MAX*节点，$\alpha = 3$

*MIN*节点，先继承，α为3

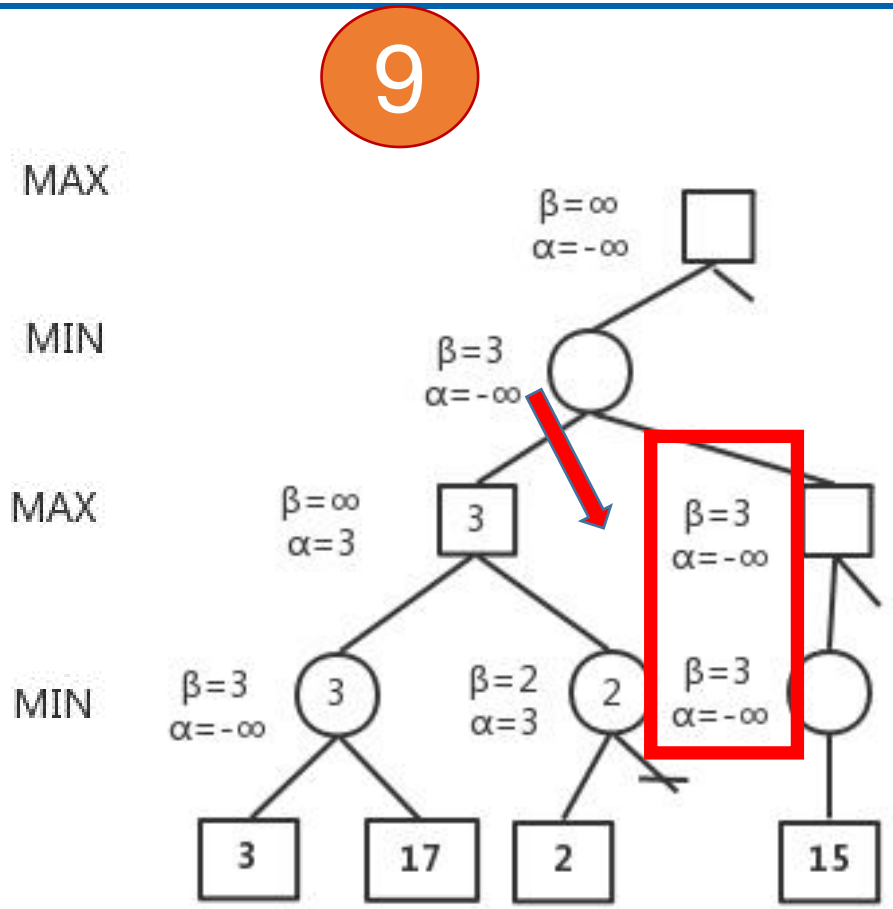
对抗搜索：Alpha-Beta 剪枝搜索示意

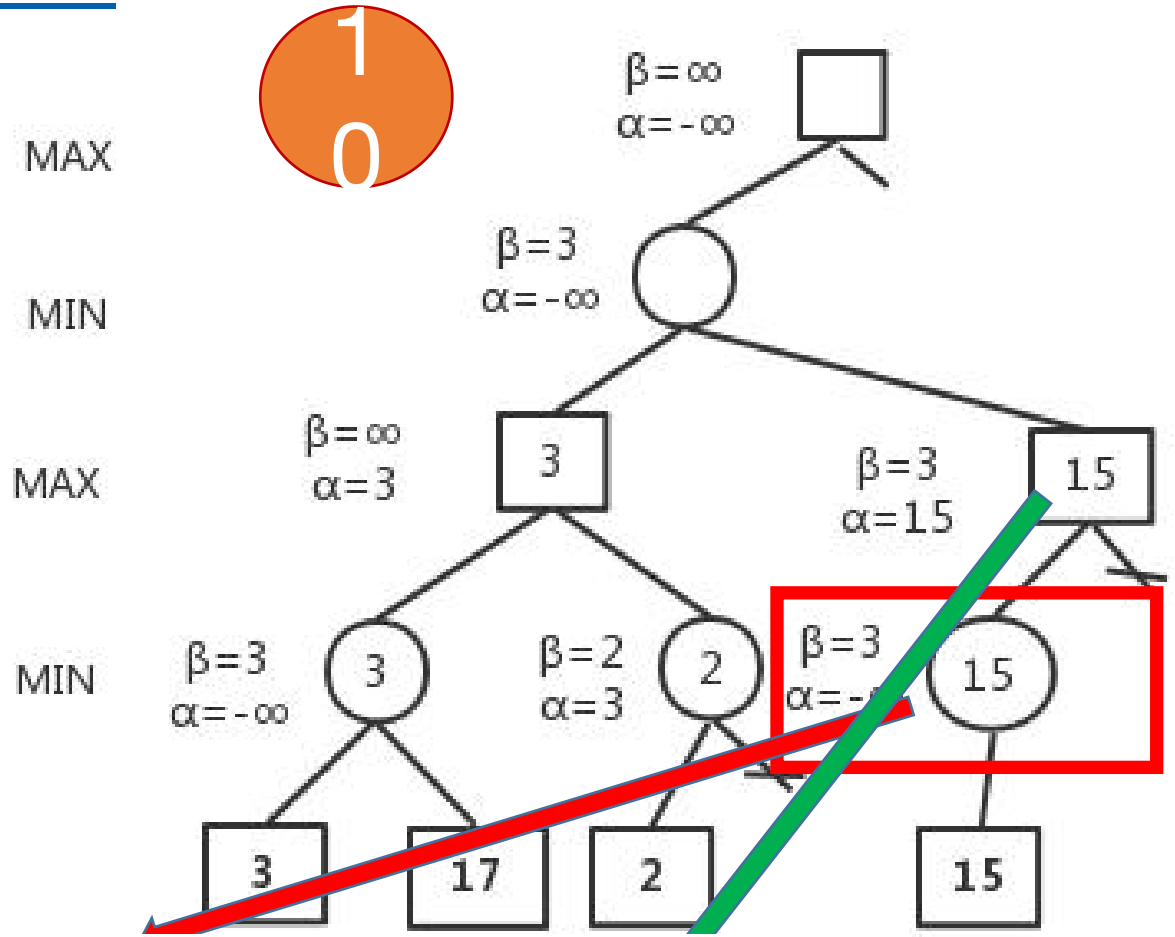




*MIN*节点，$\beta = 2$(因β仅减少)，$\alpha > \beta$,剪枝。

对抗搜索：Alpha-Beta 剪枝搜索示意

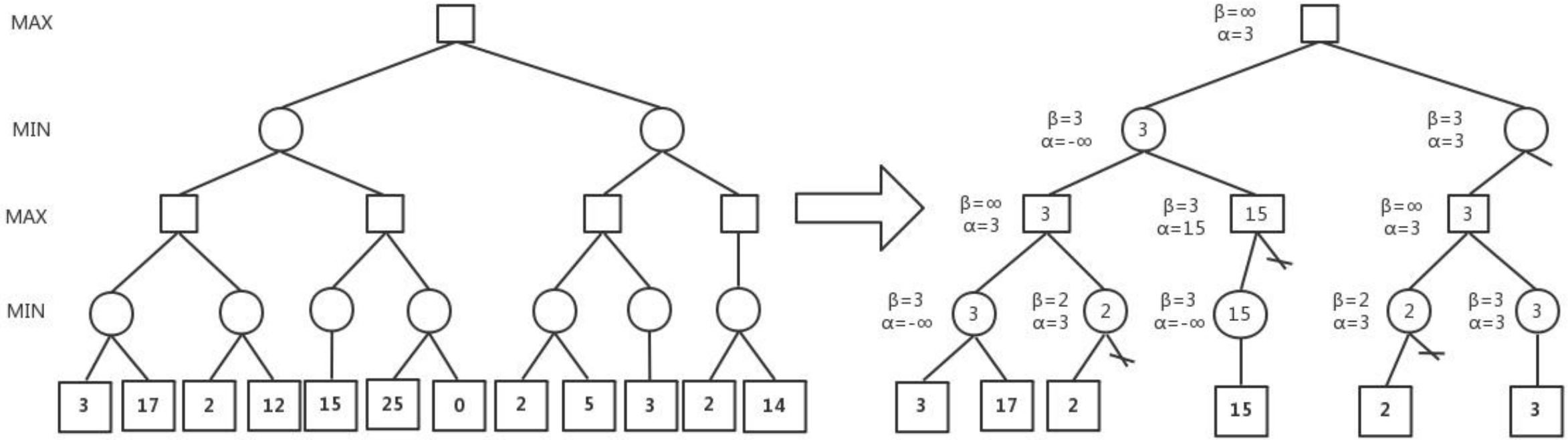




继承父节点，

对抗搜索：Alpha-Beta 剪枝搜索示意

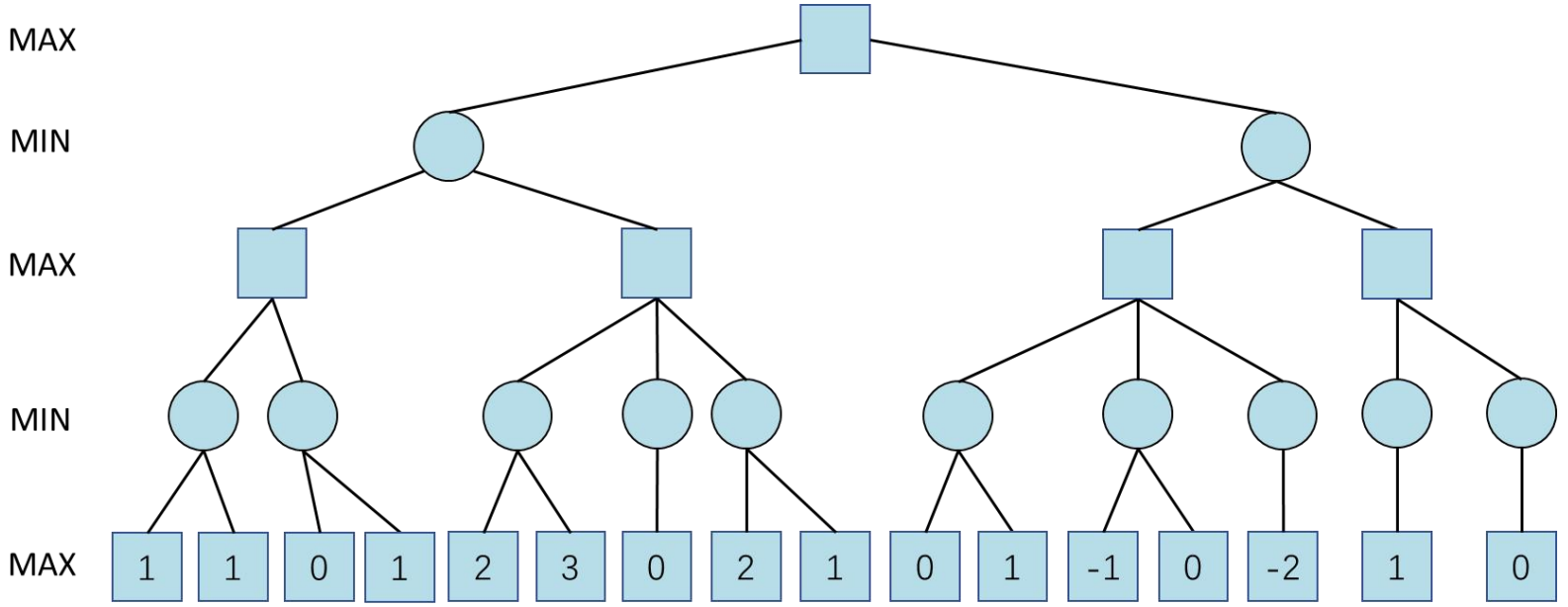
最终剪枝：



课上习题

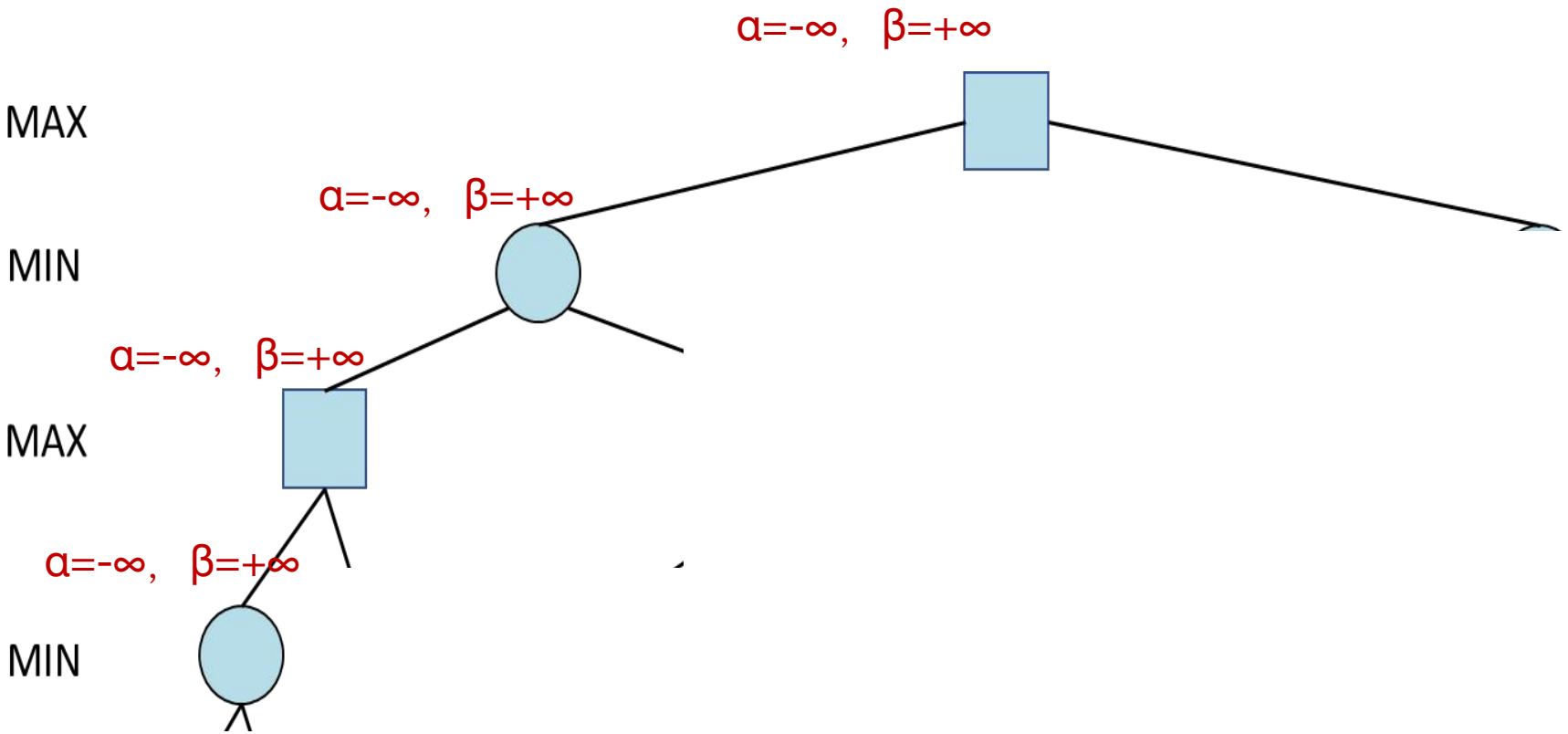
图2展示了一棵Minimax搜索树，可采用alpha-beta剪枝算法进行对抗搜索。假设对于每个节点的后继节点，算法按照从左向右的方向扩展。同时假设当alpha值等于beta值时，算法不进行剪枝。请问：

- (1)对图2(a)中所示搜索树进行搜索，请画出在算法结束时搜索树的状态，用“×”符号标出被剪枝的子树，并计算该算法扩展的节点数量。

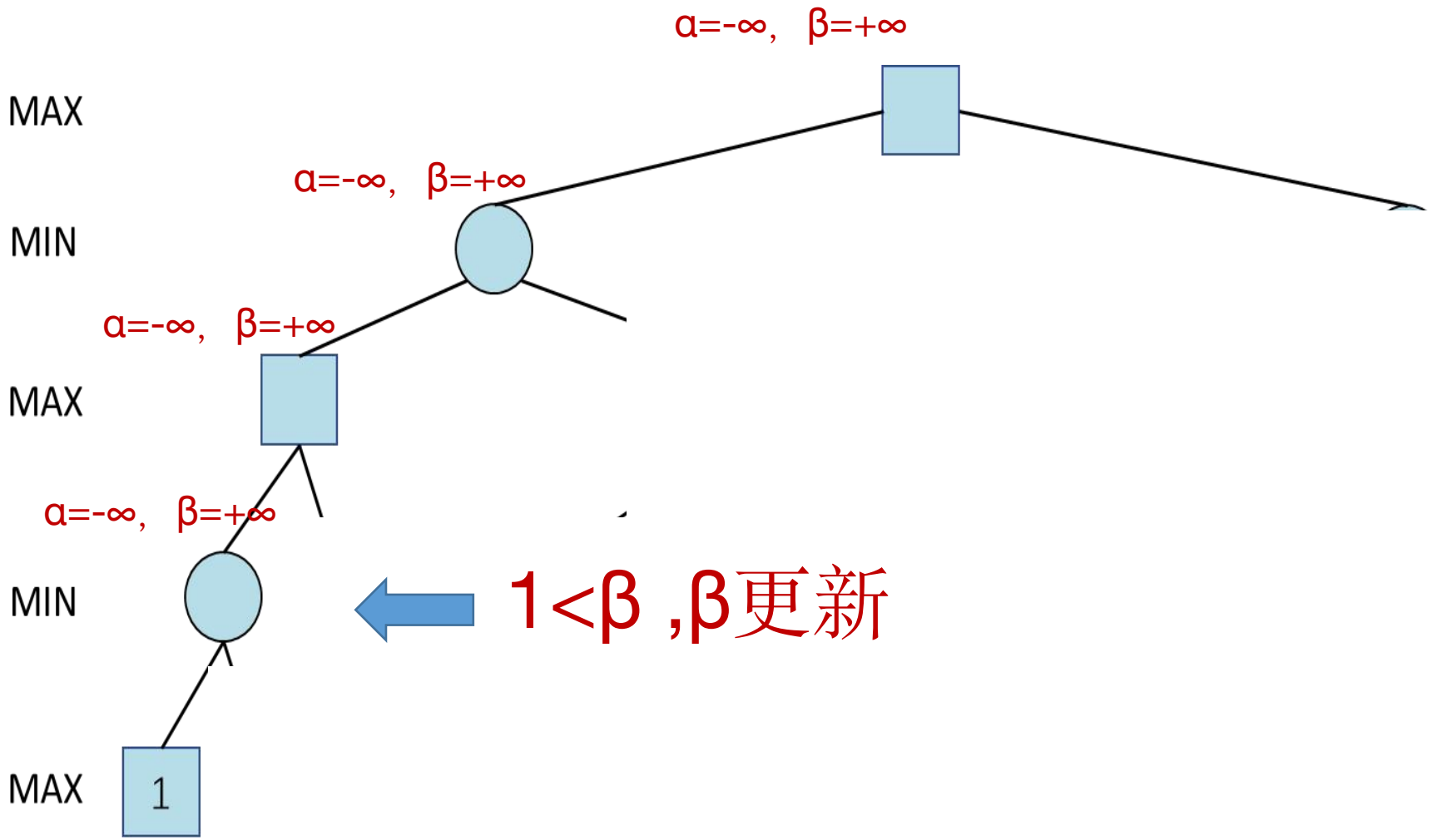


(a)

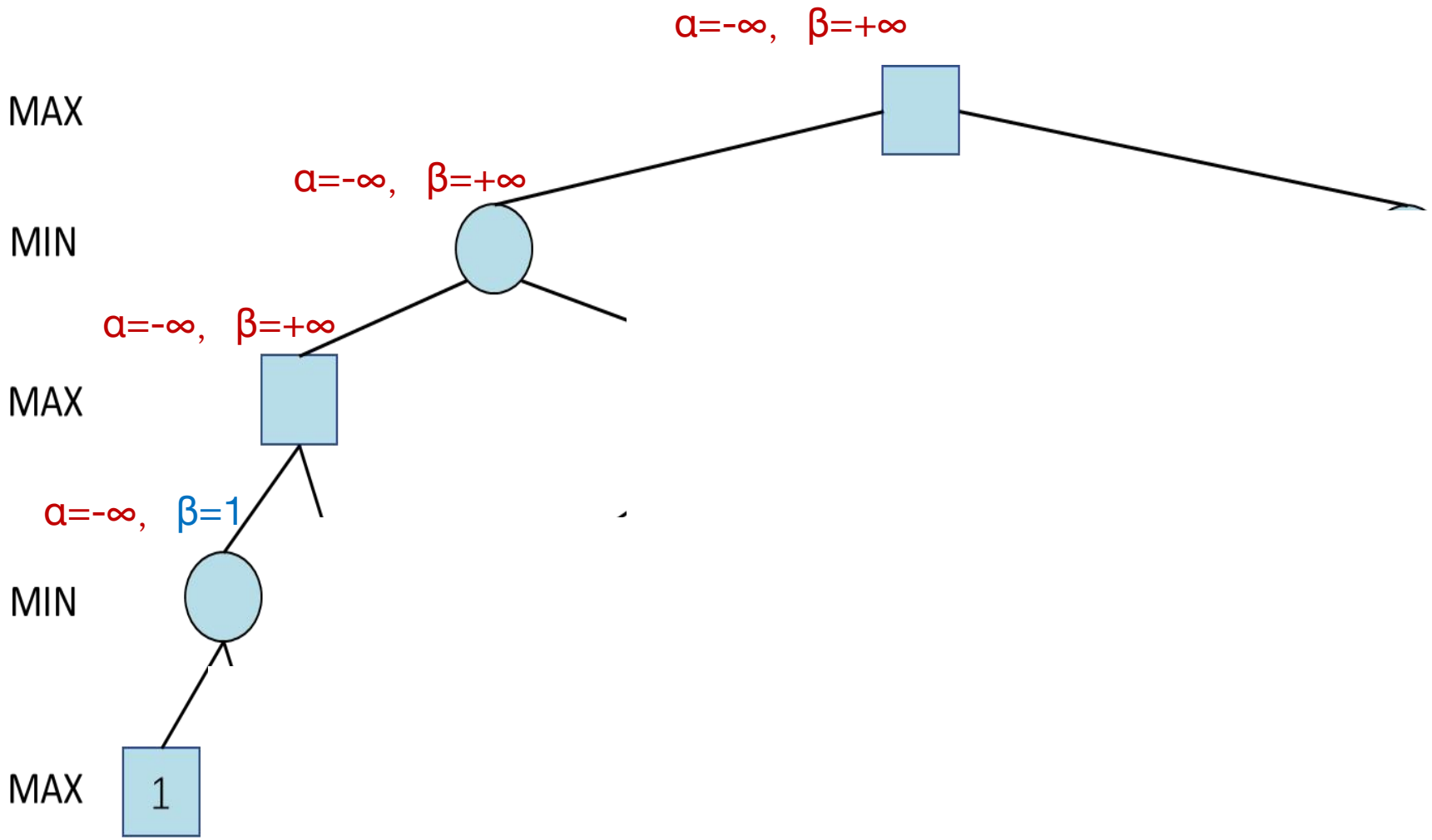
课上习题



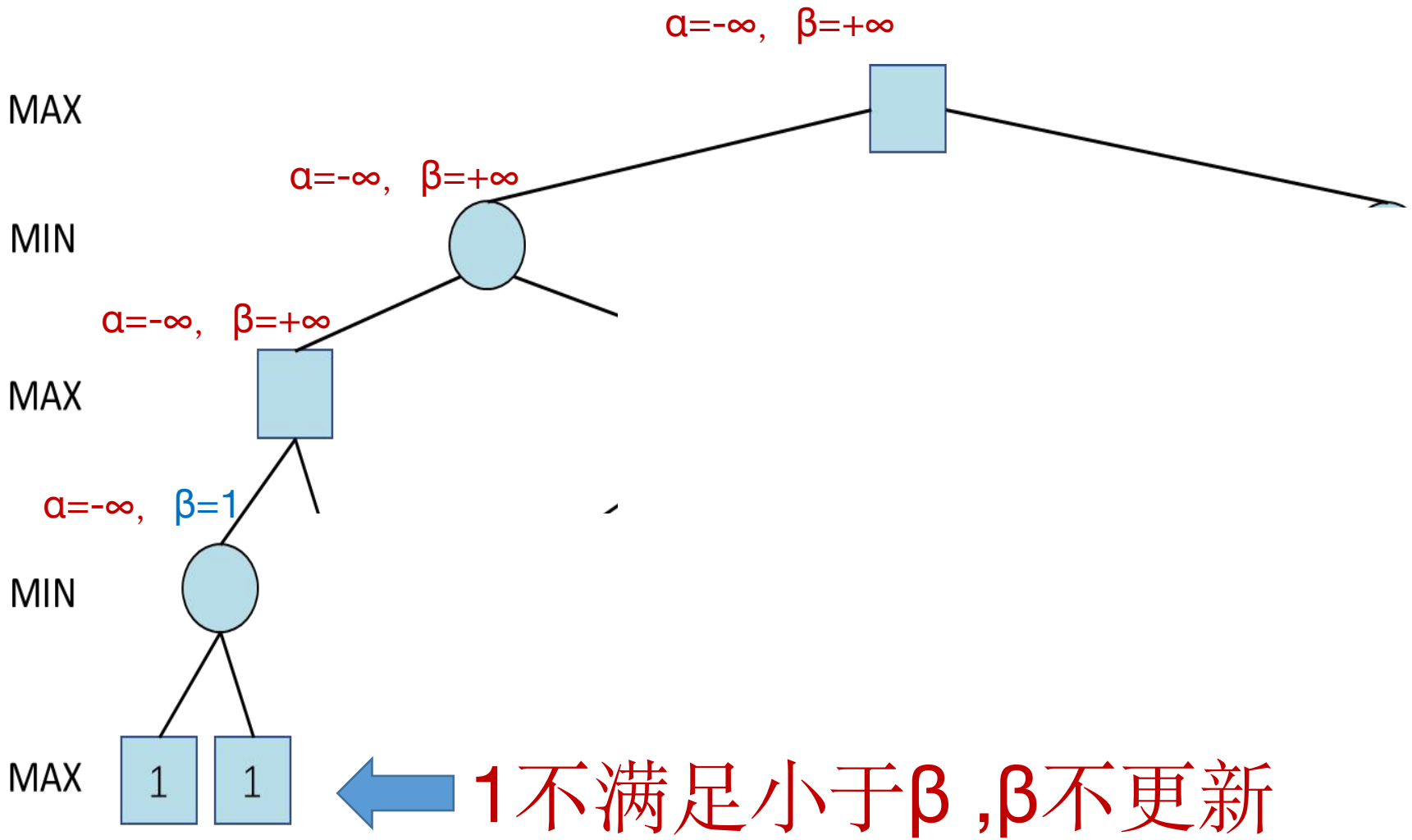
课上习题



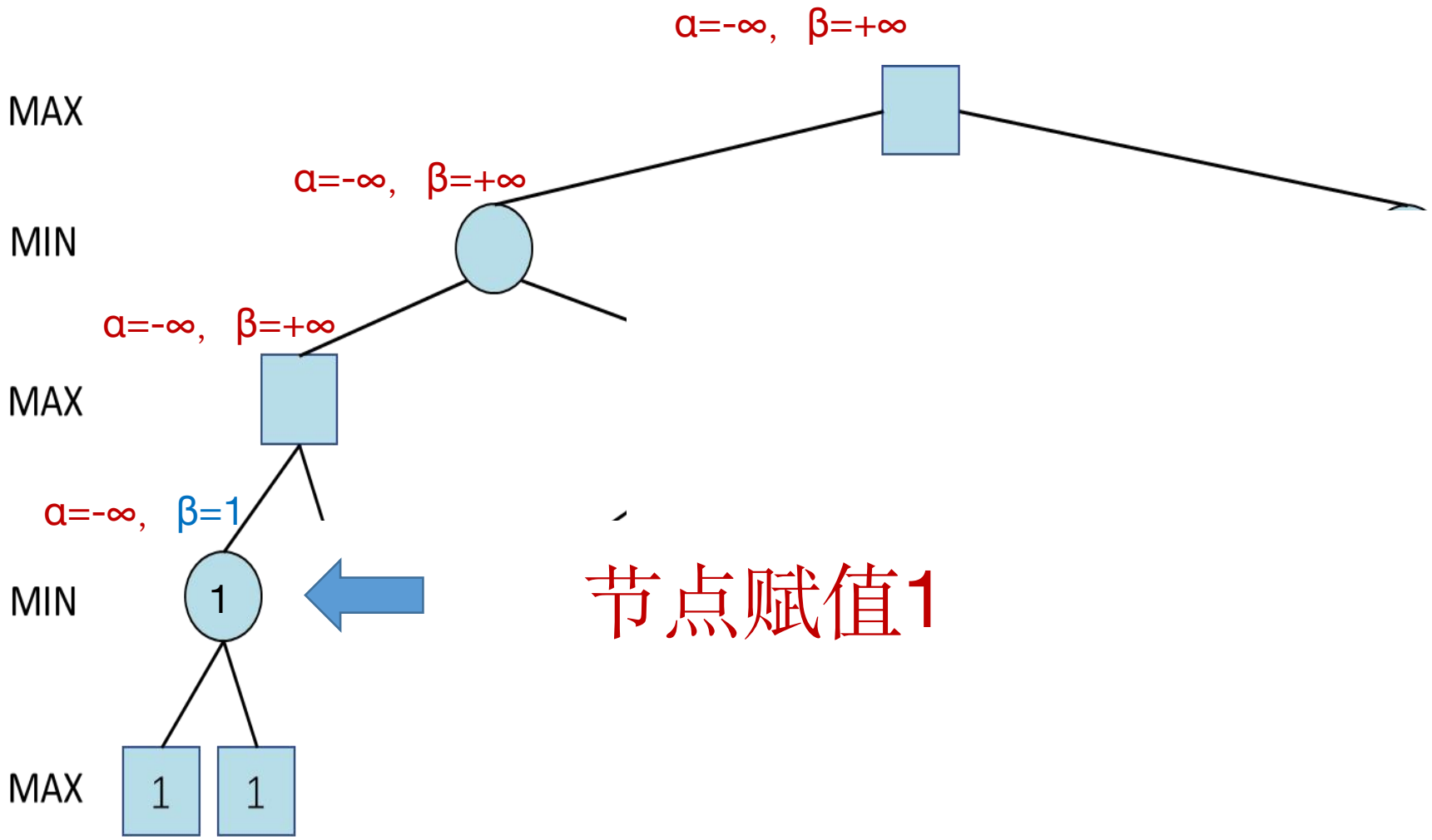
课上习题



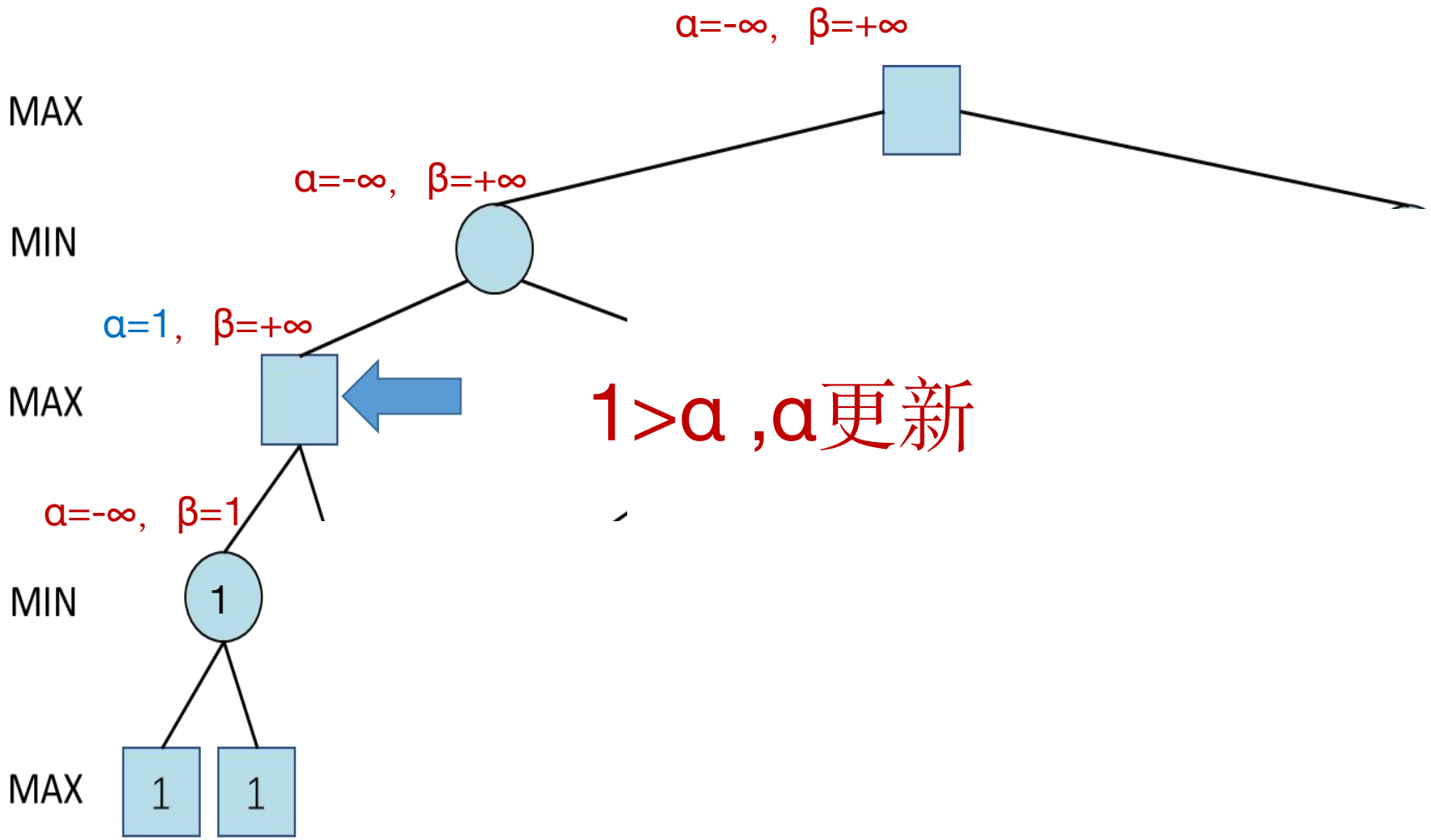
课上习题



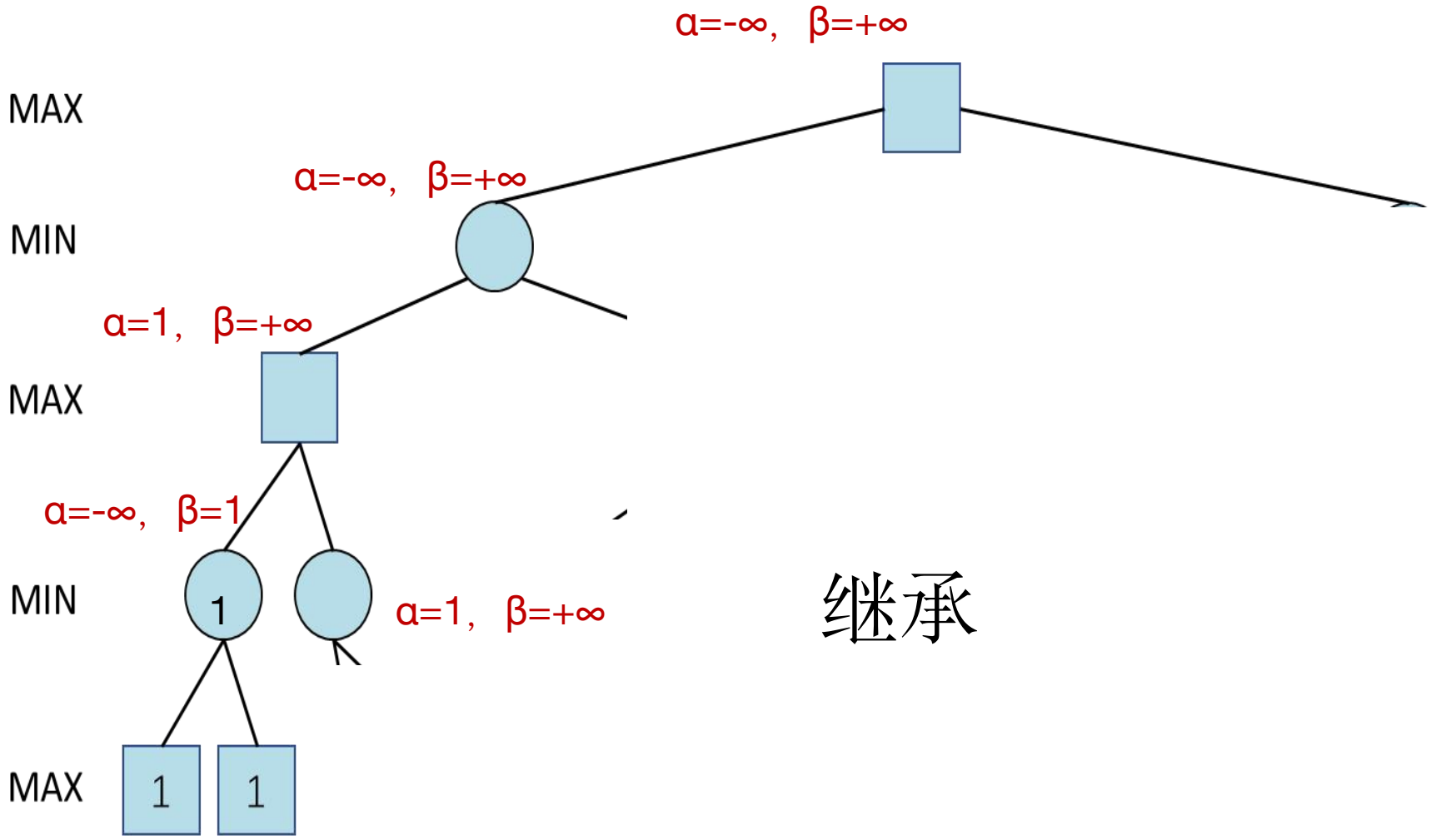
课上习题



课上习题

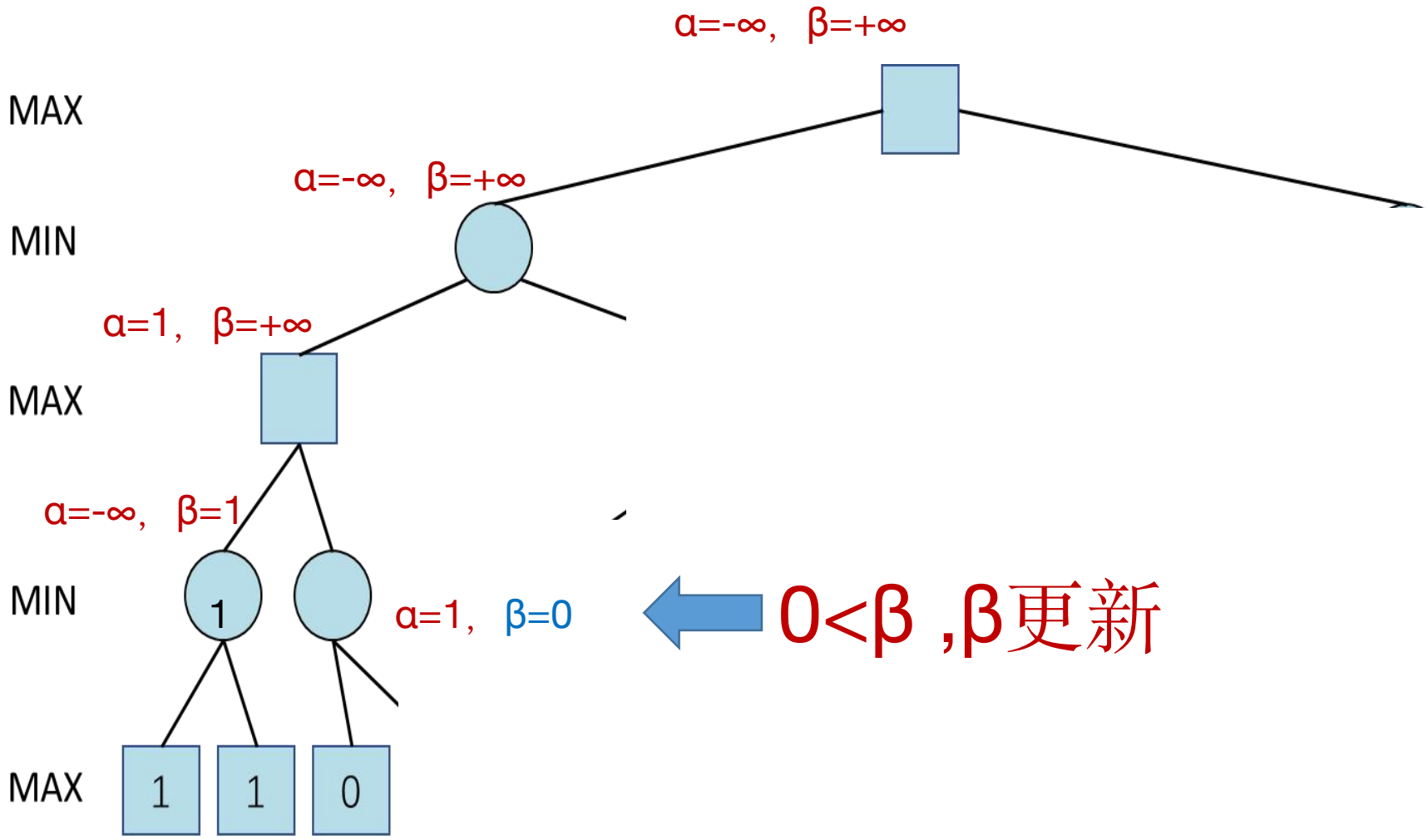


课上习题

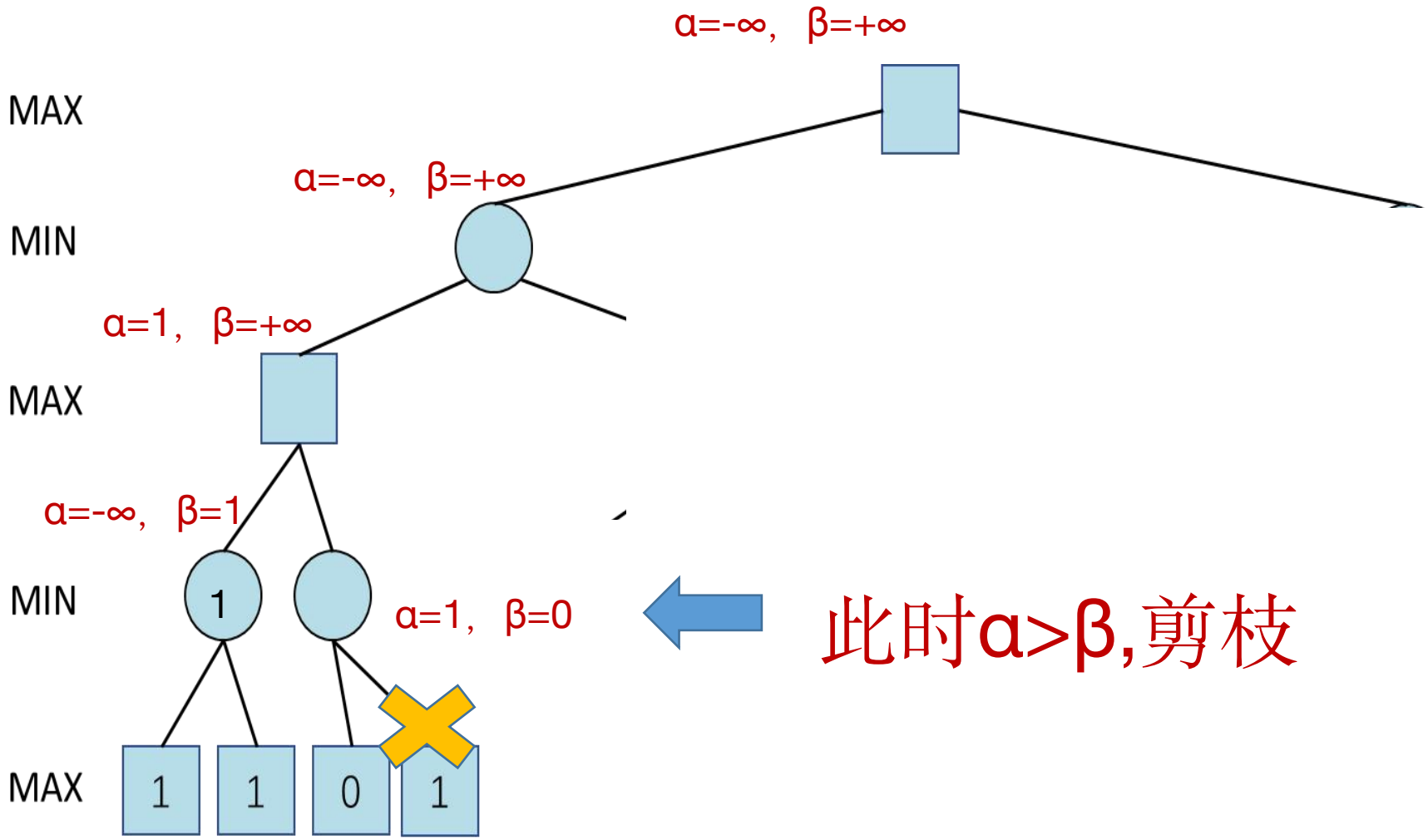


继承

课上习题

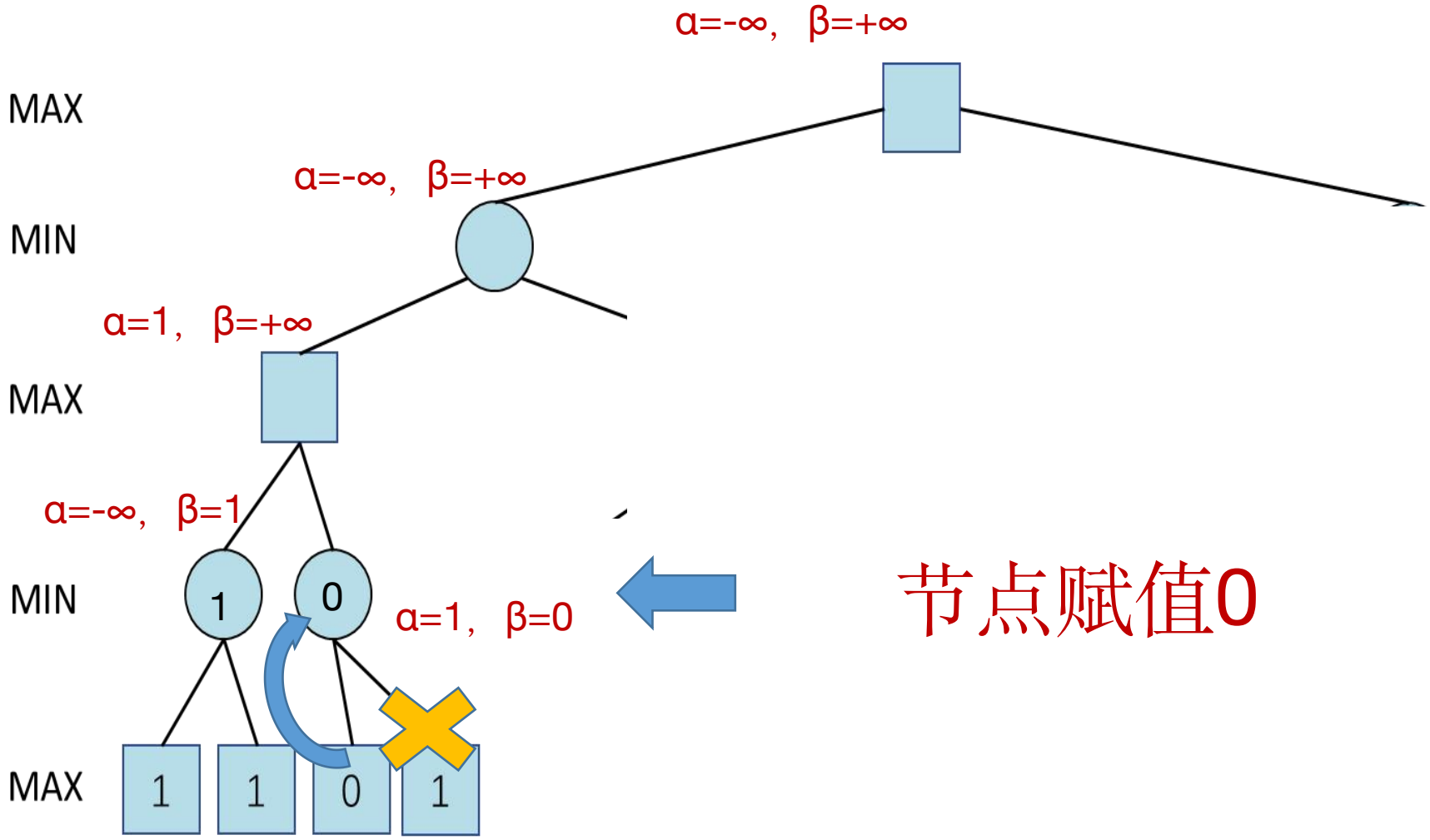


课上习题



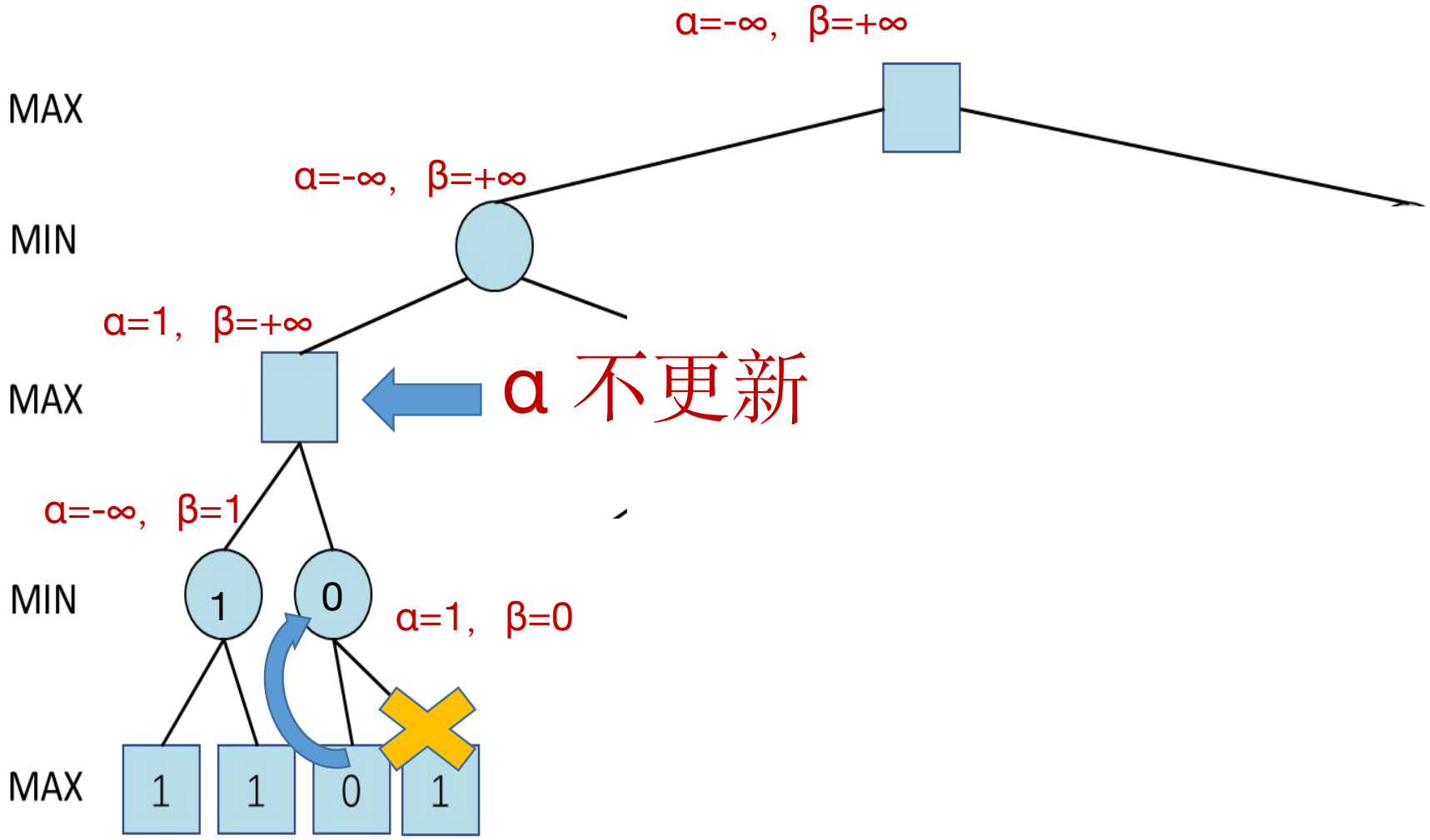
此时α>β,剪枝

课上习题

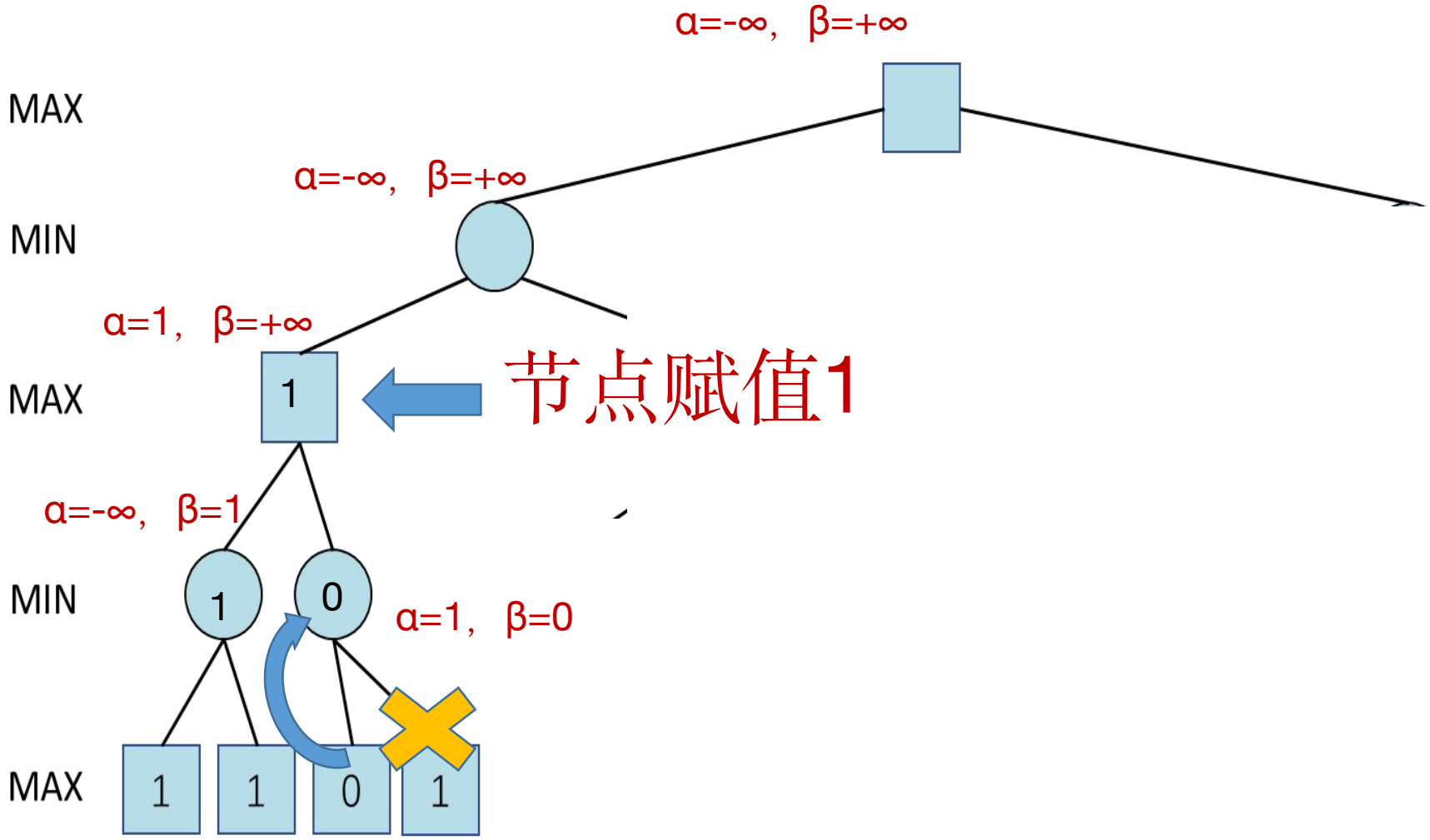


节点赋值0

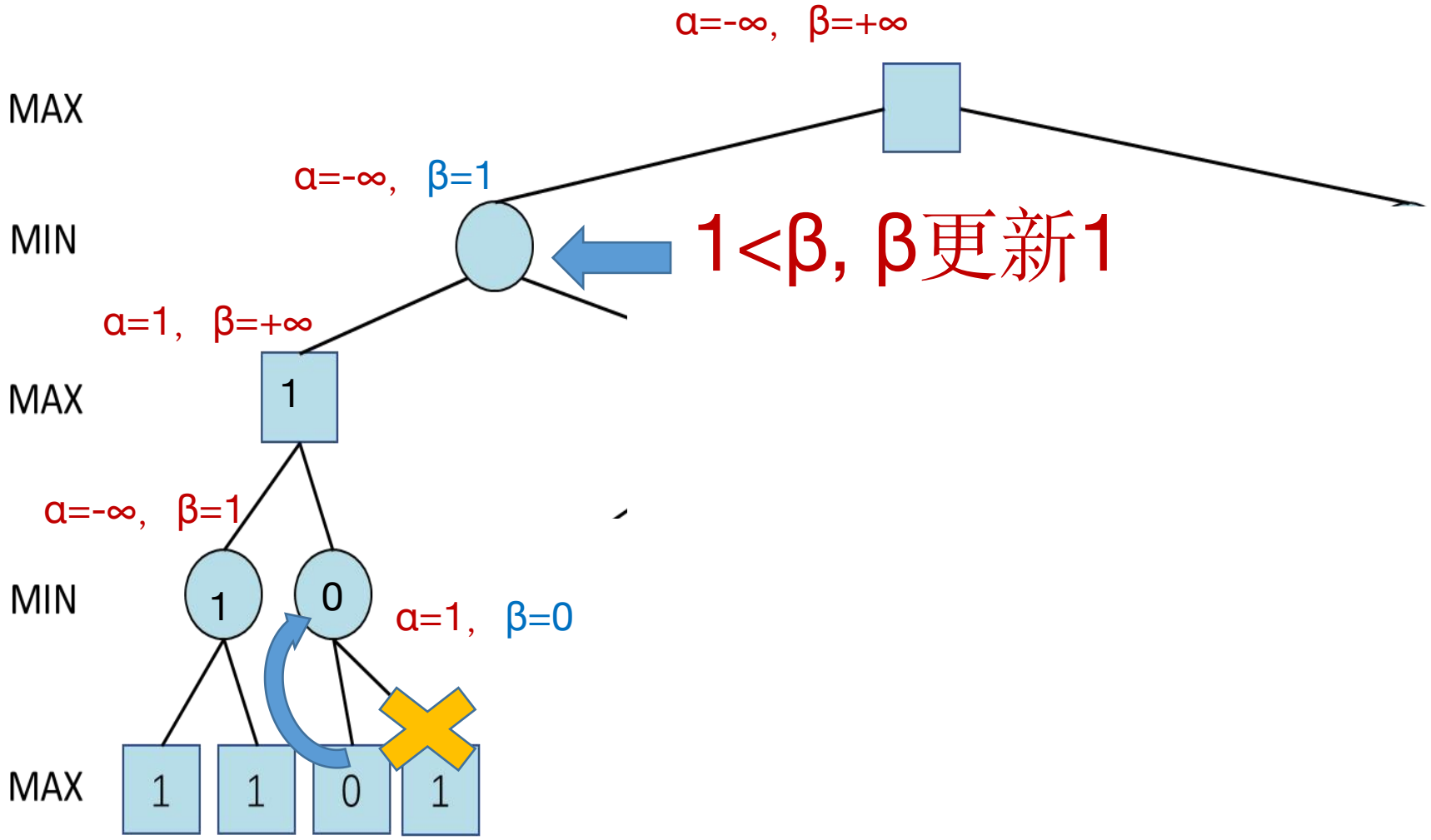
课上习题



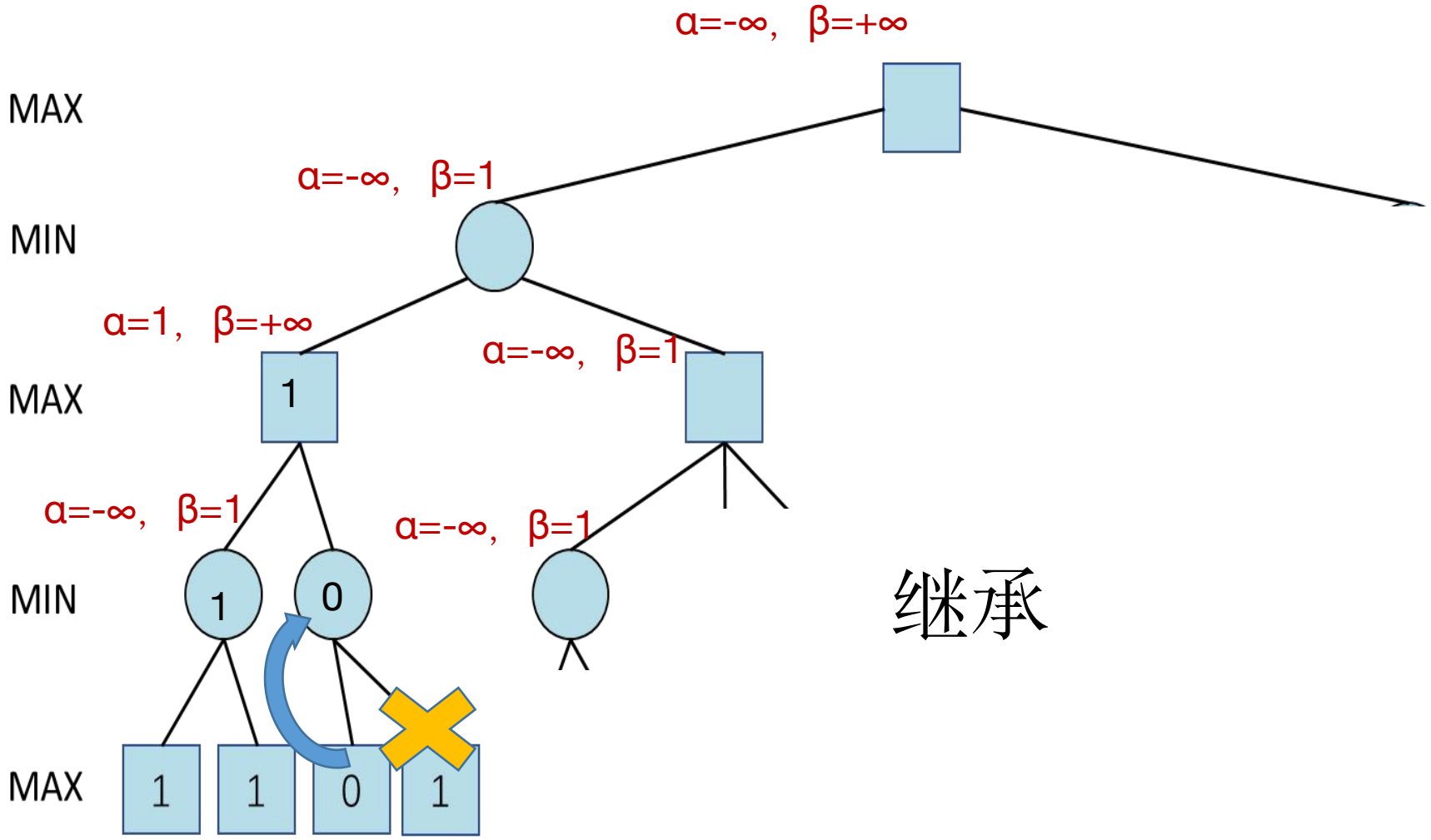
课上习题



课上习题

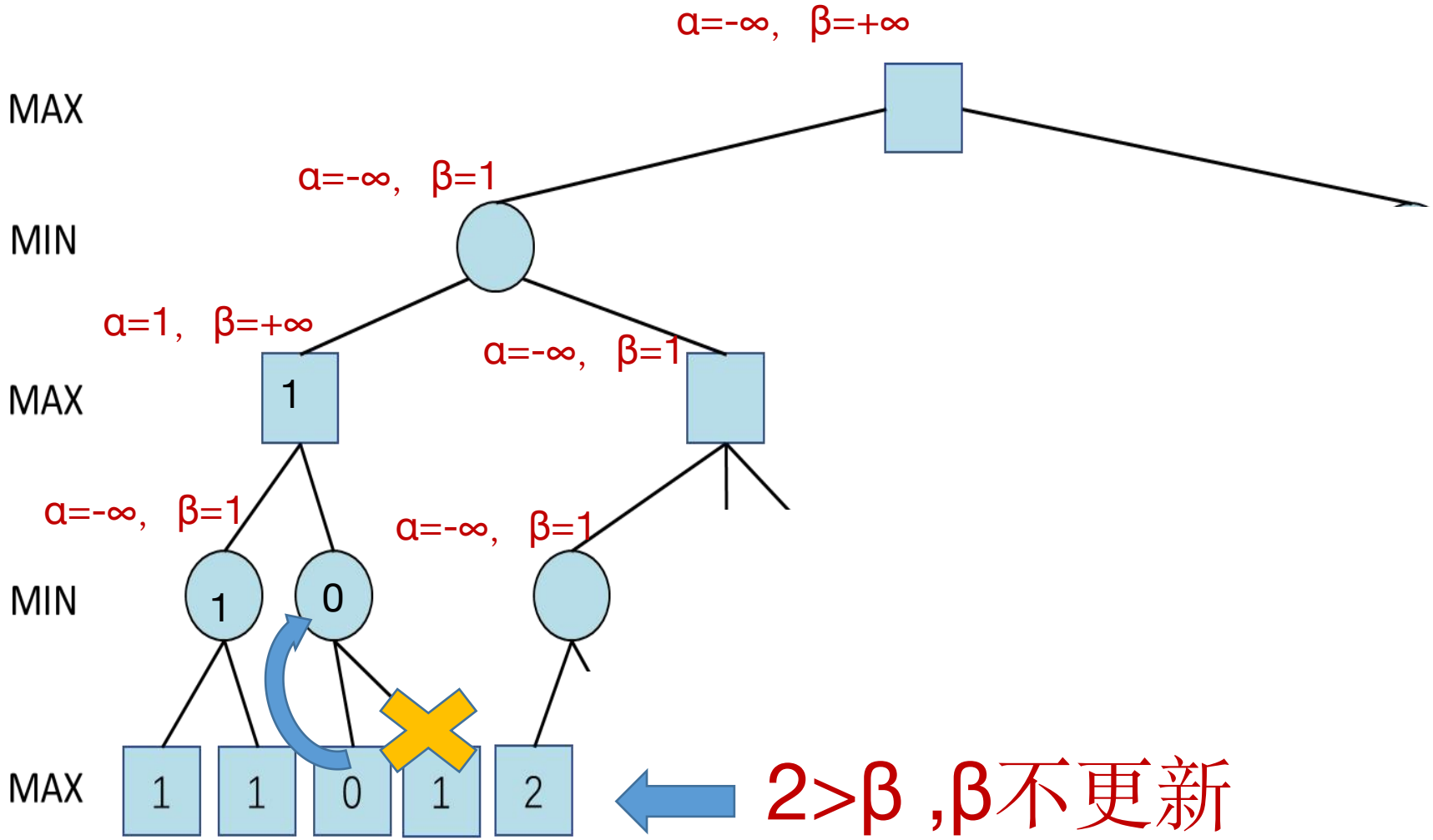


课上习题

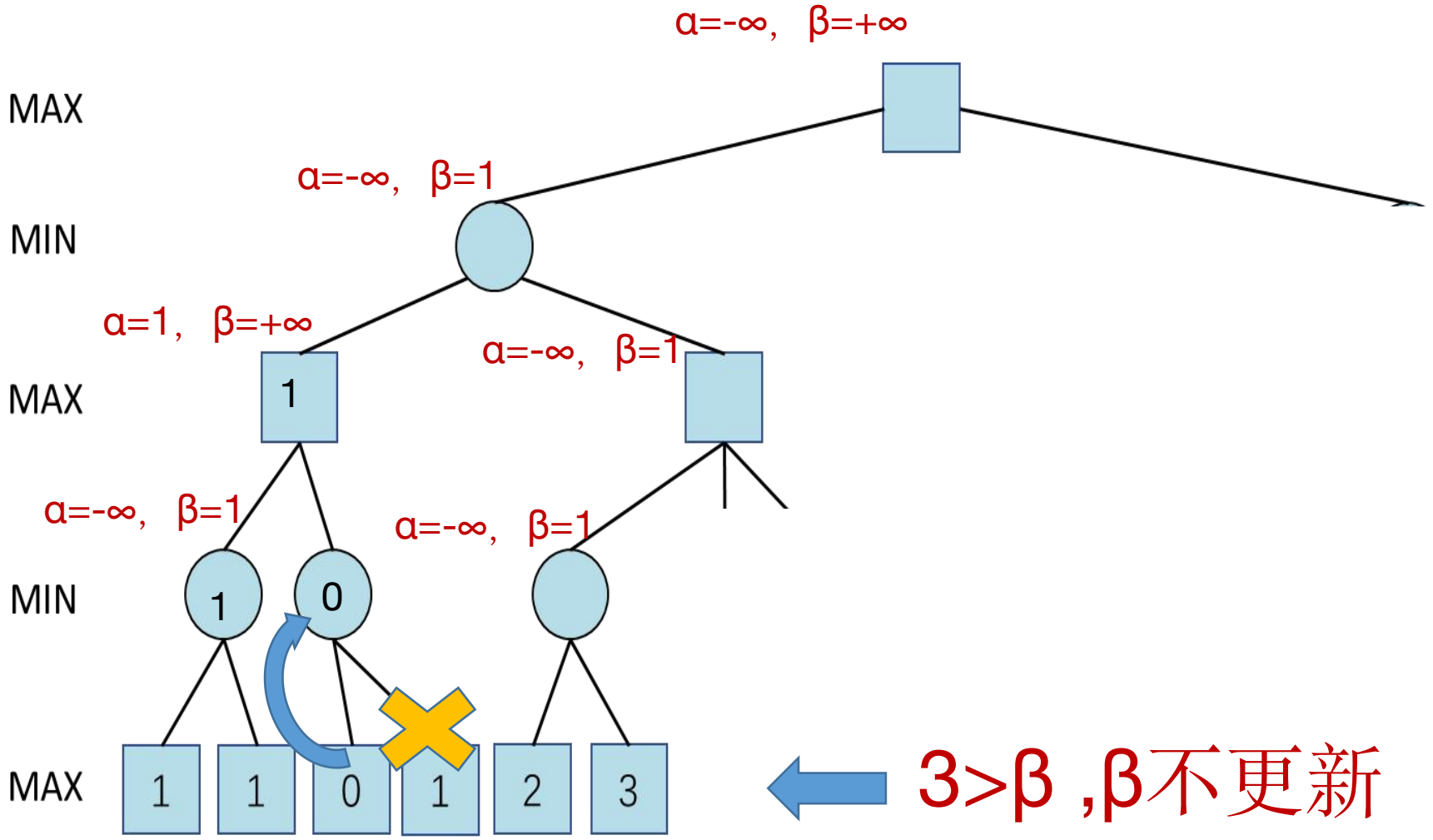


继承

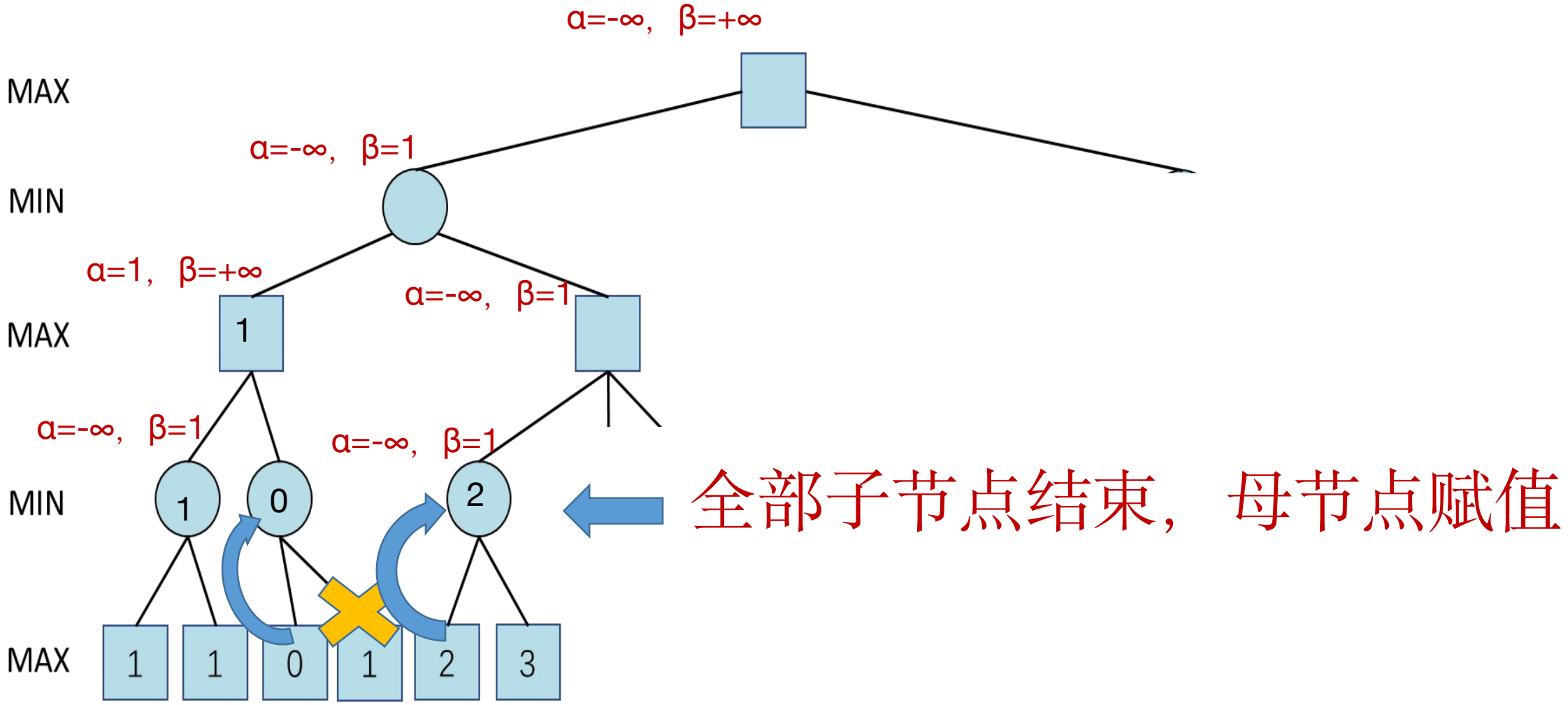
课上习题



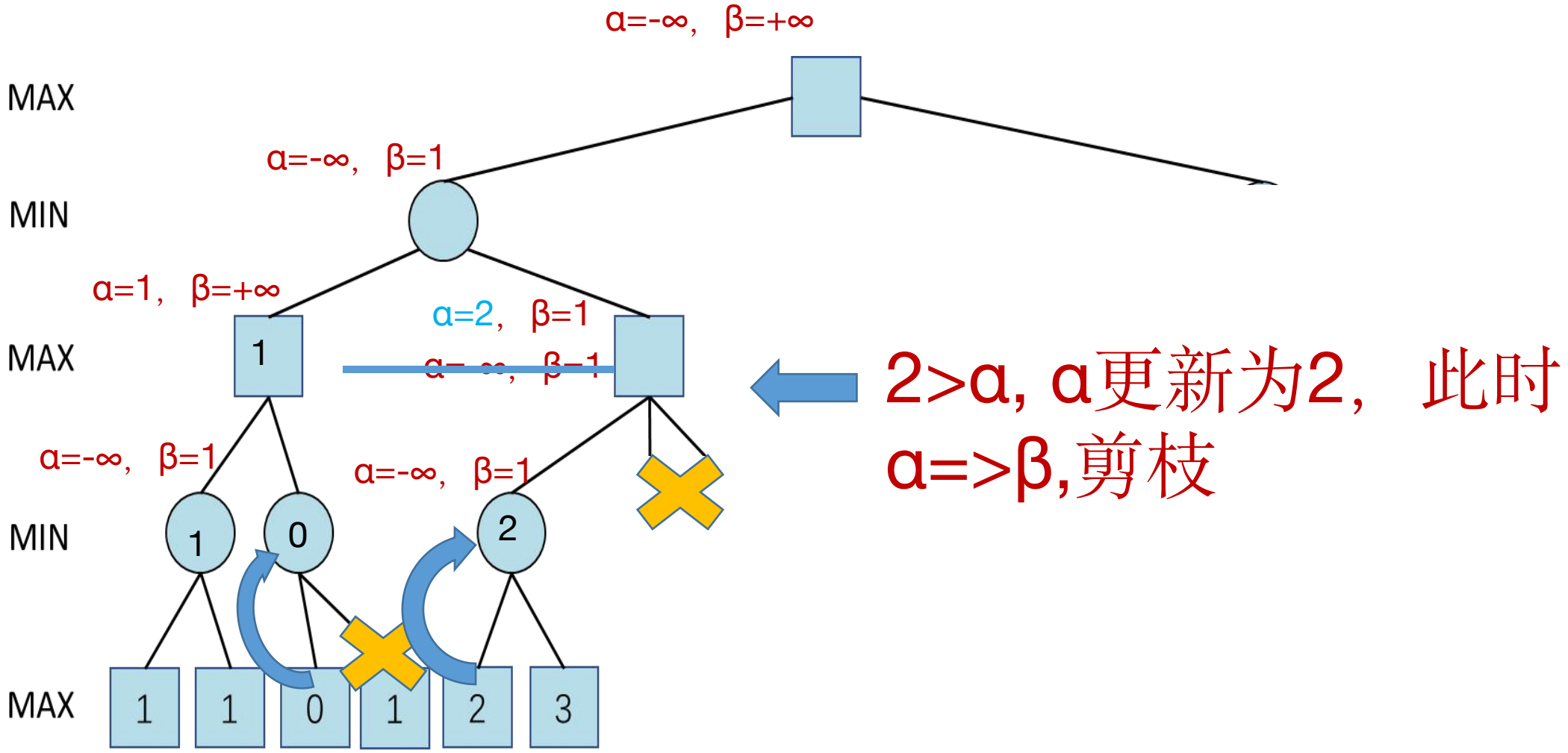
课上习题



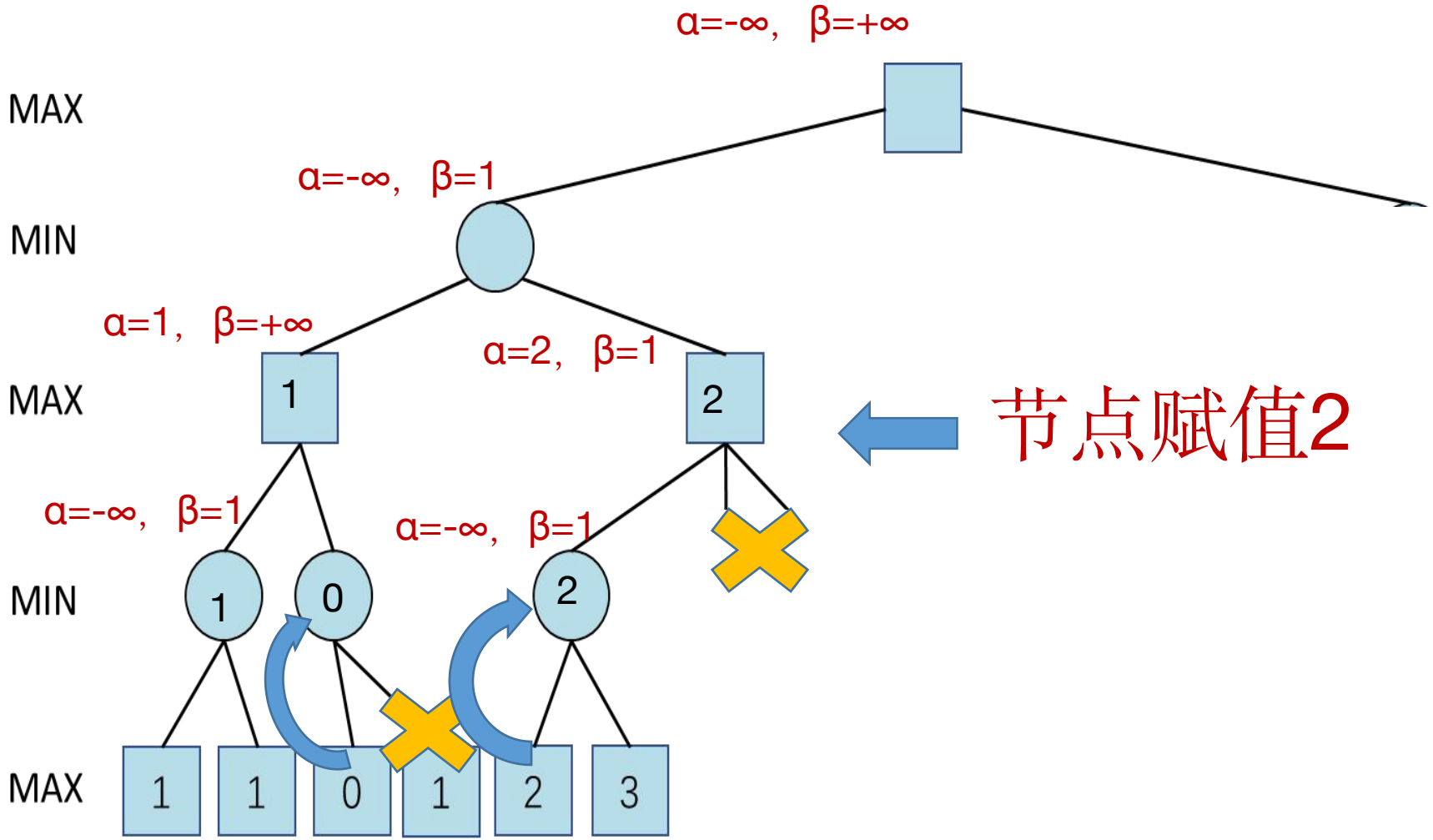
课上习题


α=-∞, β=+∞
MAX
α=-∞, β=1
MIN
α=1, β=+∞
α=-∞, β=1
MAX
1
α=-∞, β=1
α=-∞, β=1
MIN
1
0
2
MAX
1
1
0
1
2
3
全部子节点结束，母节点赋值

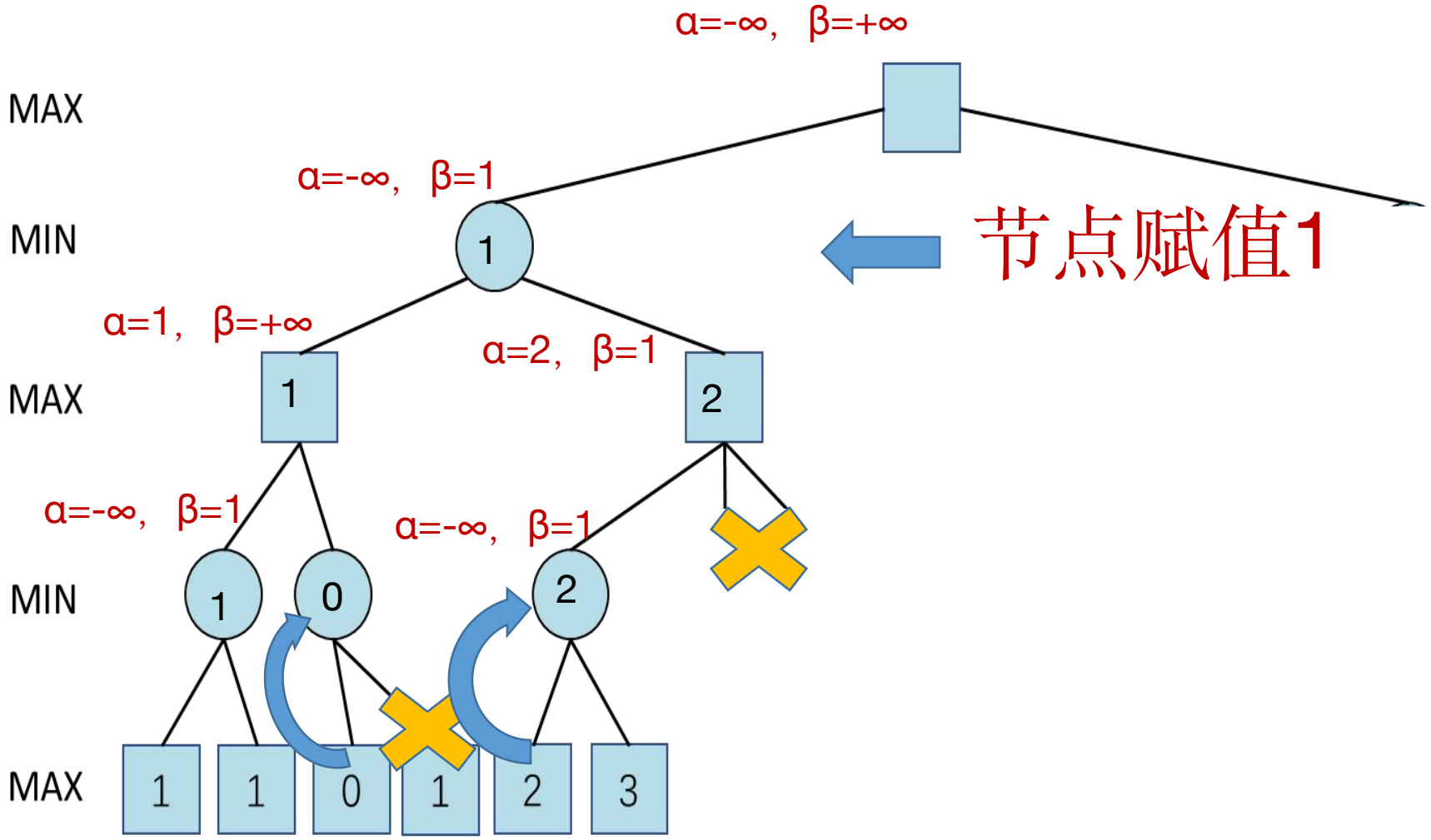
课上习题


α=-∞, β=+∞
MAX
α=-∞, β=1
MIN
α=1, β=+∞
α=2, β=1
MAX
1
α=-∞, β=1
α=-∞, β=1
MIN
1
0
2
MAX
1
1
0
1
2
3
2>α, α更新为2， 此时
α=>β,剪枝

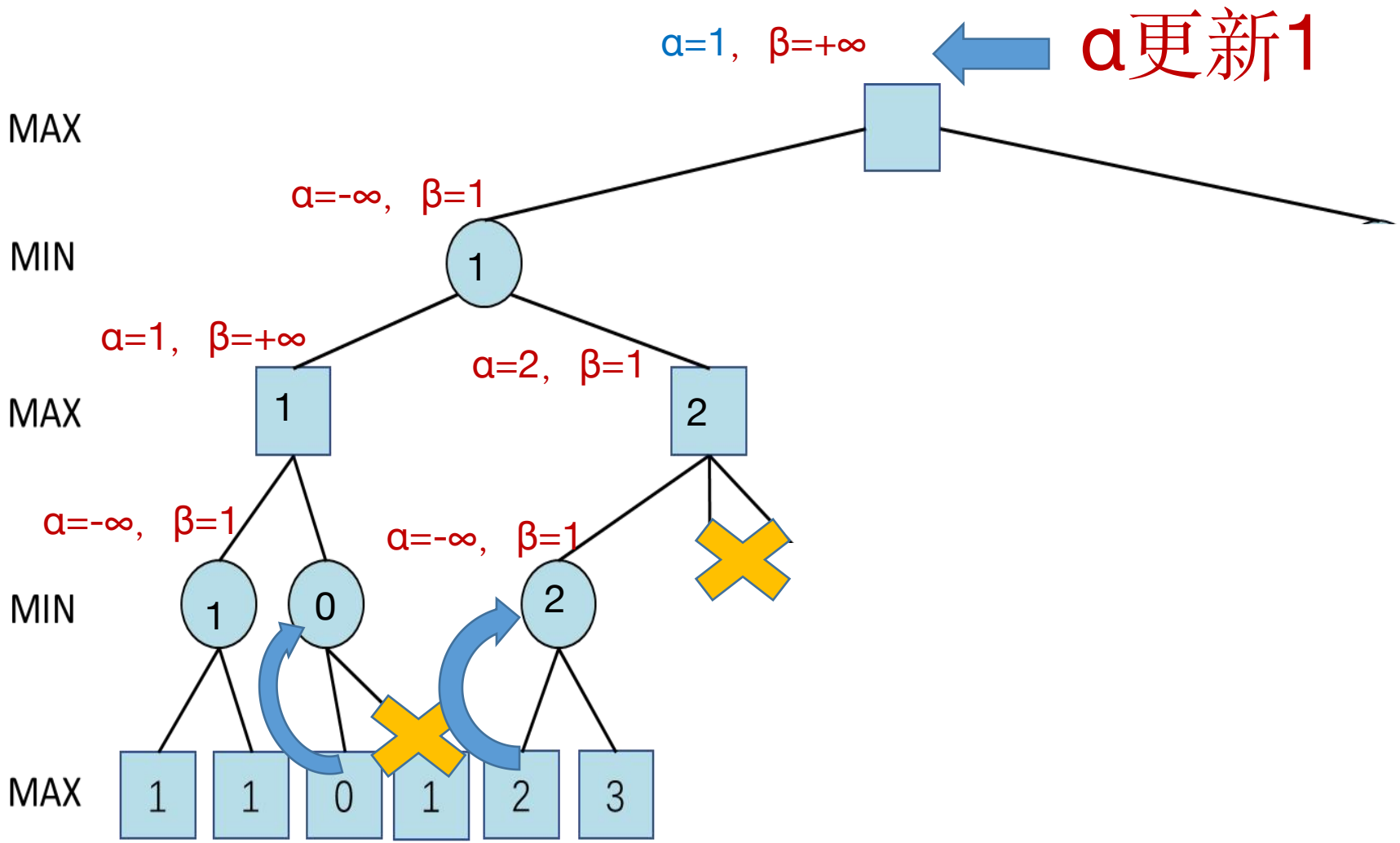
课上习题



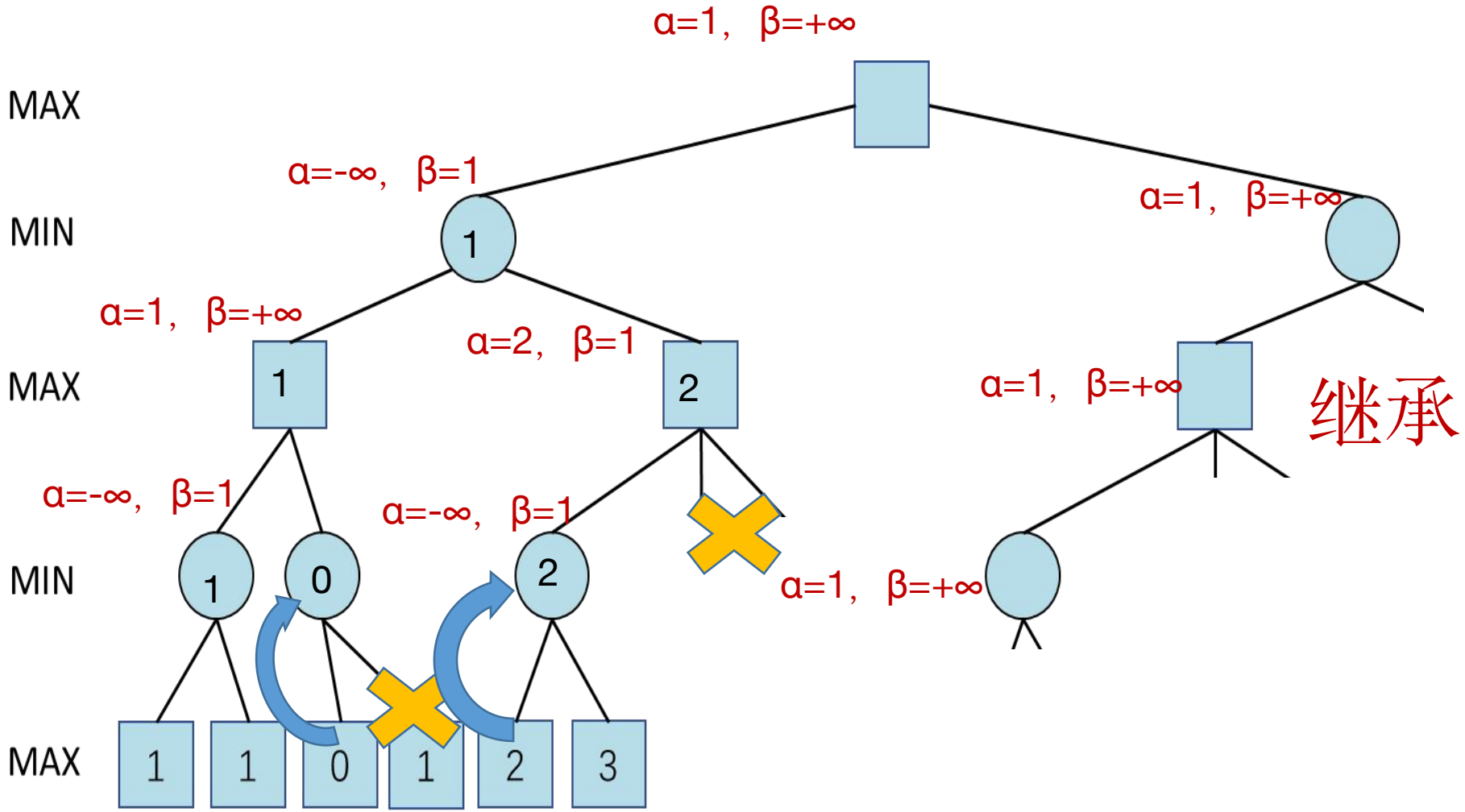
课上习题


α=-∞, β=+∞
MAX
α=-∞, β=1
MIN
1
节点赋值1
α=1, β=+∞
α=2, β=1
MAX
1
2
α=-∞, β=1
α=-∞, β=1
MIN
1
0
2
MAX
1
1
0
1
2
3

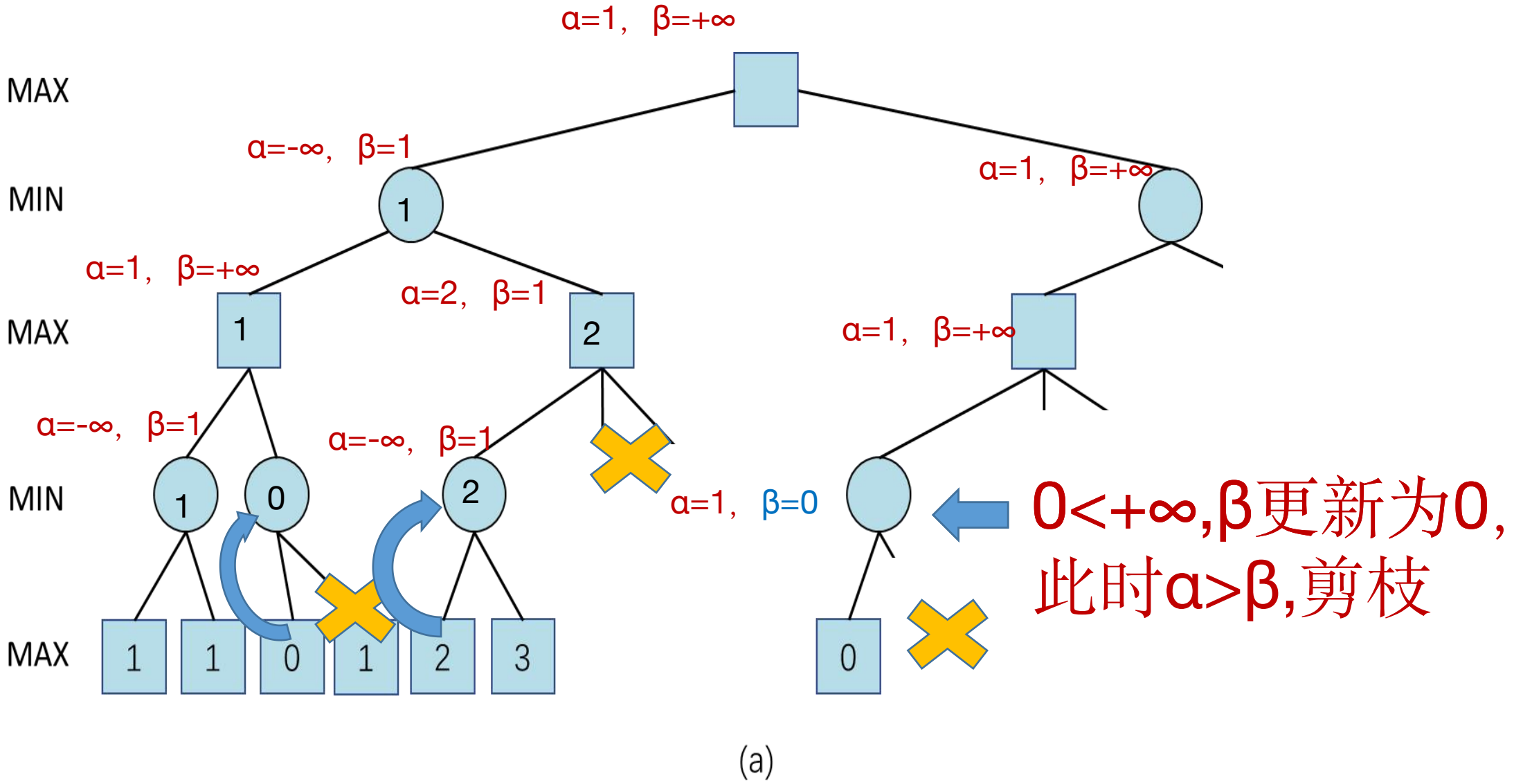
课上习题


α=1， β=+∞
α更新1
MAX
α=-∞， β=1
MIN
1
α=1， β=+∞
α=2， β=1
MAX
1
2
α=-∞， β=1
α=-∞， β=1
MIN
1
0
2
MAX
1
1
0
1
2
3

课上习题

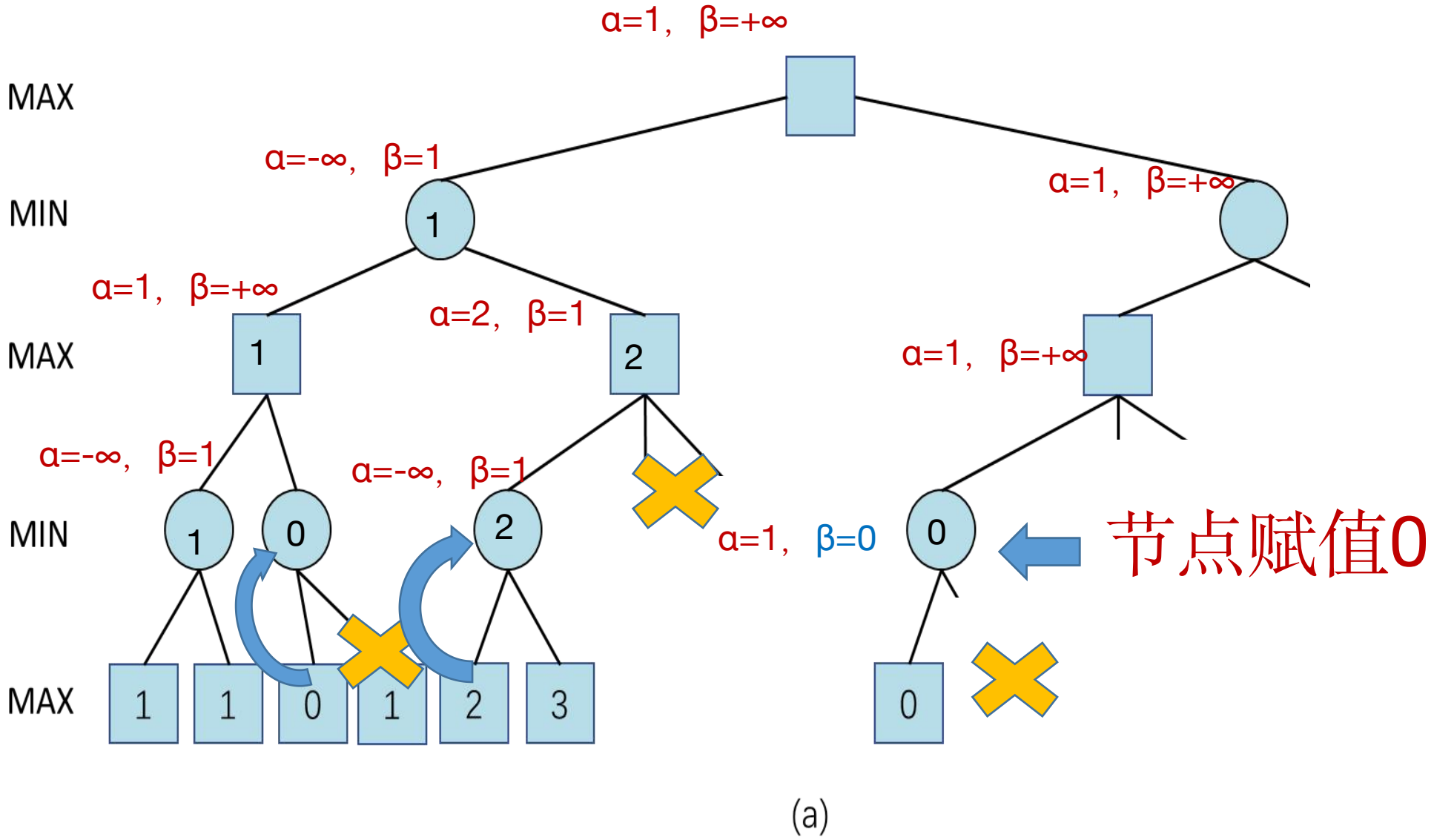

α=1， β=+∞
MAX
α=-∞， β=1
MIN
1
α=1， β=+∞
α=1， β=+∞
α=2， β=1
MAX
1
2
α=1， β=+∞
继承
α=-∞， β=1
α=-∞， β=1
MIN
1
0
2
α=1， β=+∞
MAX
1
1
0
1
2
3

课上习题


α=1， β=+∞
MAX
α=-∞， β=1
MIN
1
α=1， β=+∞
α=1， β=+∞
α=1， β=+∞
α=2， β=1
MAX
1
2
α=-∞， β=1
α=-∞， β=1
α=1， β=0
MIN
1
0
2
MAX
1
1
0
1
2
3
0
0<+∞,β更新为0,
此时α>β,剪枝

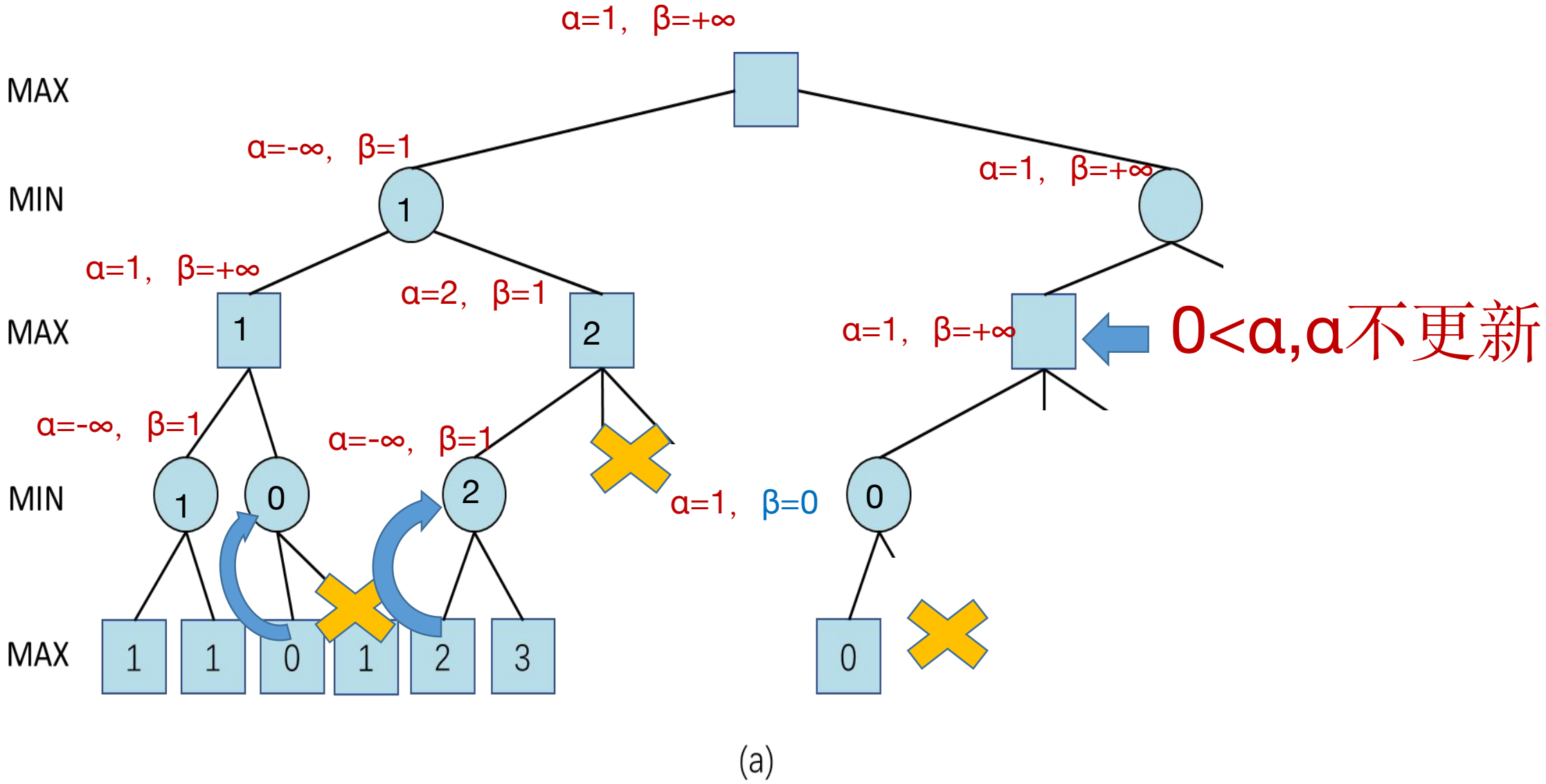
(a)

课上习题

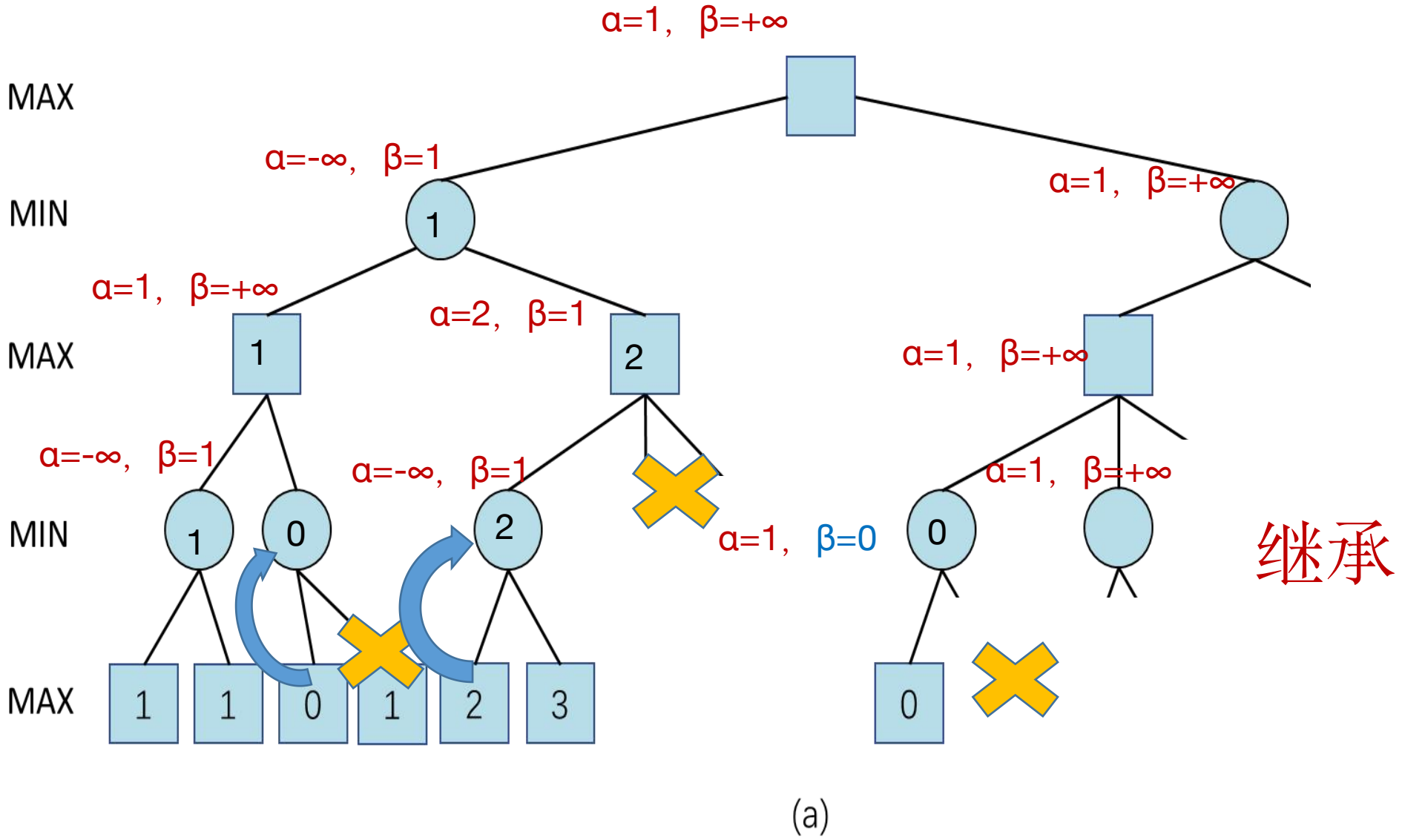

α=1， β=+∞
MAX
α=-∞， β=1
MIN
1
α=1， β=+∞
α=1， β=+∞
α=2， β=1
MAX
1
2
α=1， β=+∞
α=-∞， β=1
α=-∞， β=1
MIN
1
0
2
α=1， β=0
0
节点赋值0
MAX
1
1
0
1
2
3
0

(a)

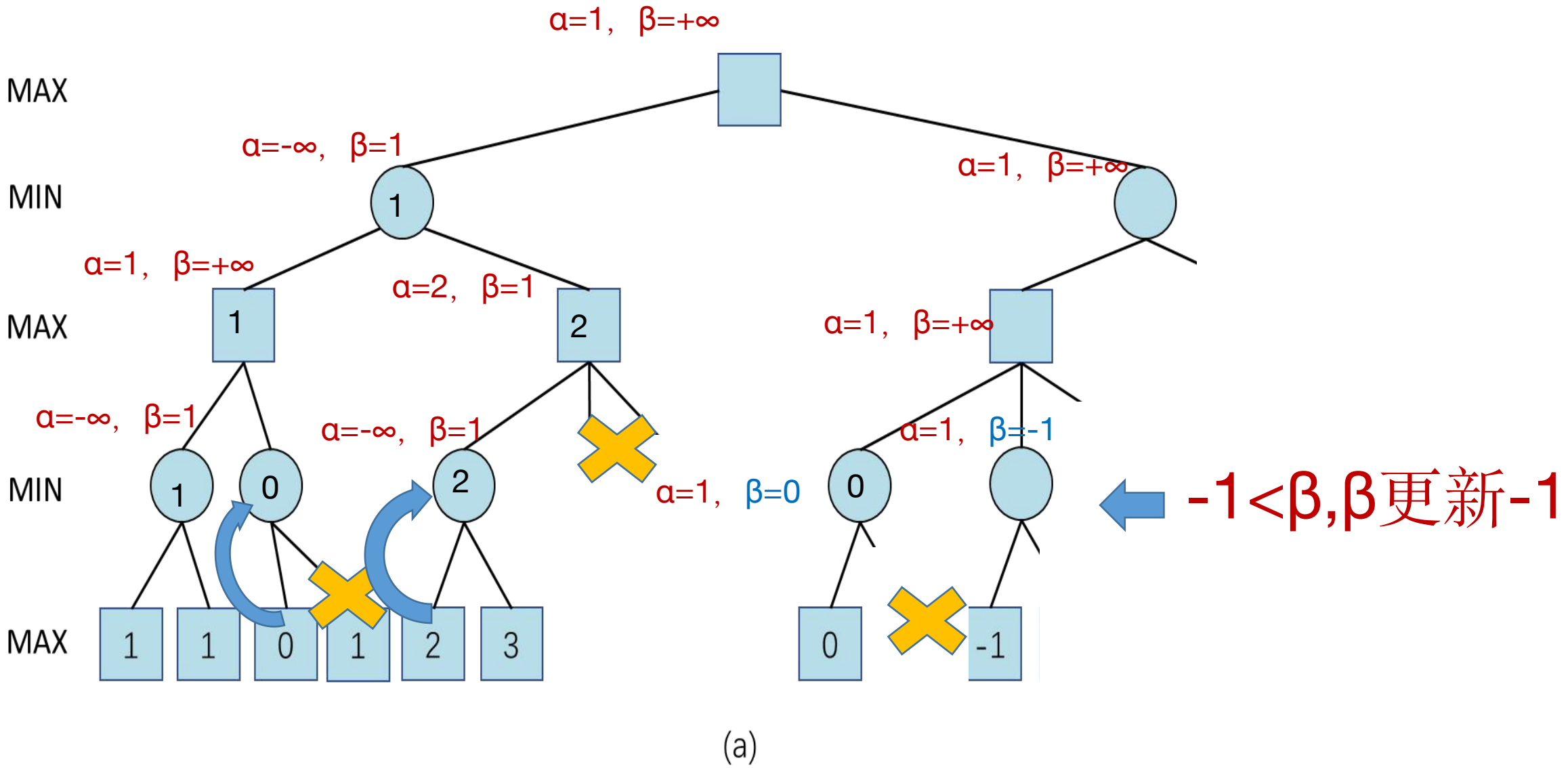
课上习题


α=1， β=+∞
MAX
α=-∞， β=1
MIN
1
α=1， β=+∞
α=1， β=+∞
α=2， β=1
MAX
1
2
α=1， β=+∞
0<α,α不更新
α=-∞， β=1
α=-∞， β=1
MIN
1
0
2
α=1， β=0
0
MAX
1
1
0
1
2
3
0
(a)

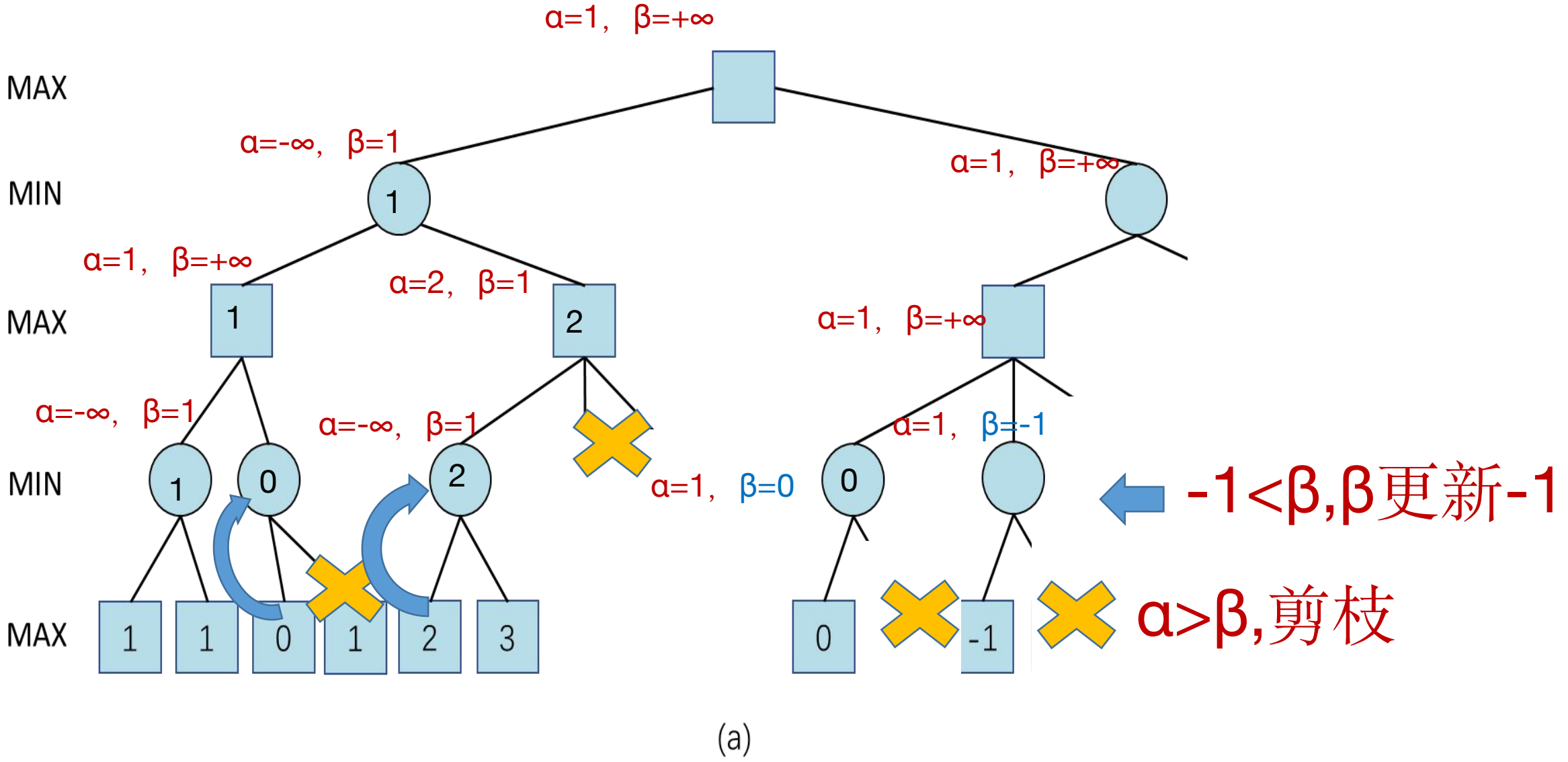
课上习题


α=1, β=+∞
MAX
α=-∞, β=1
MIN
1
α=1, β=+∞
α=1, β=+∞
α=2, β=1
MAX
1
2
α=1, β=+∞
α=-∞, β=1
α=-∞, β=1
α=1, β=+∞
MIN
1
0
2
α=1, β=0
0
继承
MAX
1
1
0
1
2
3
0
(a)

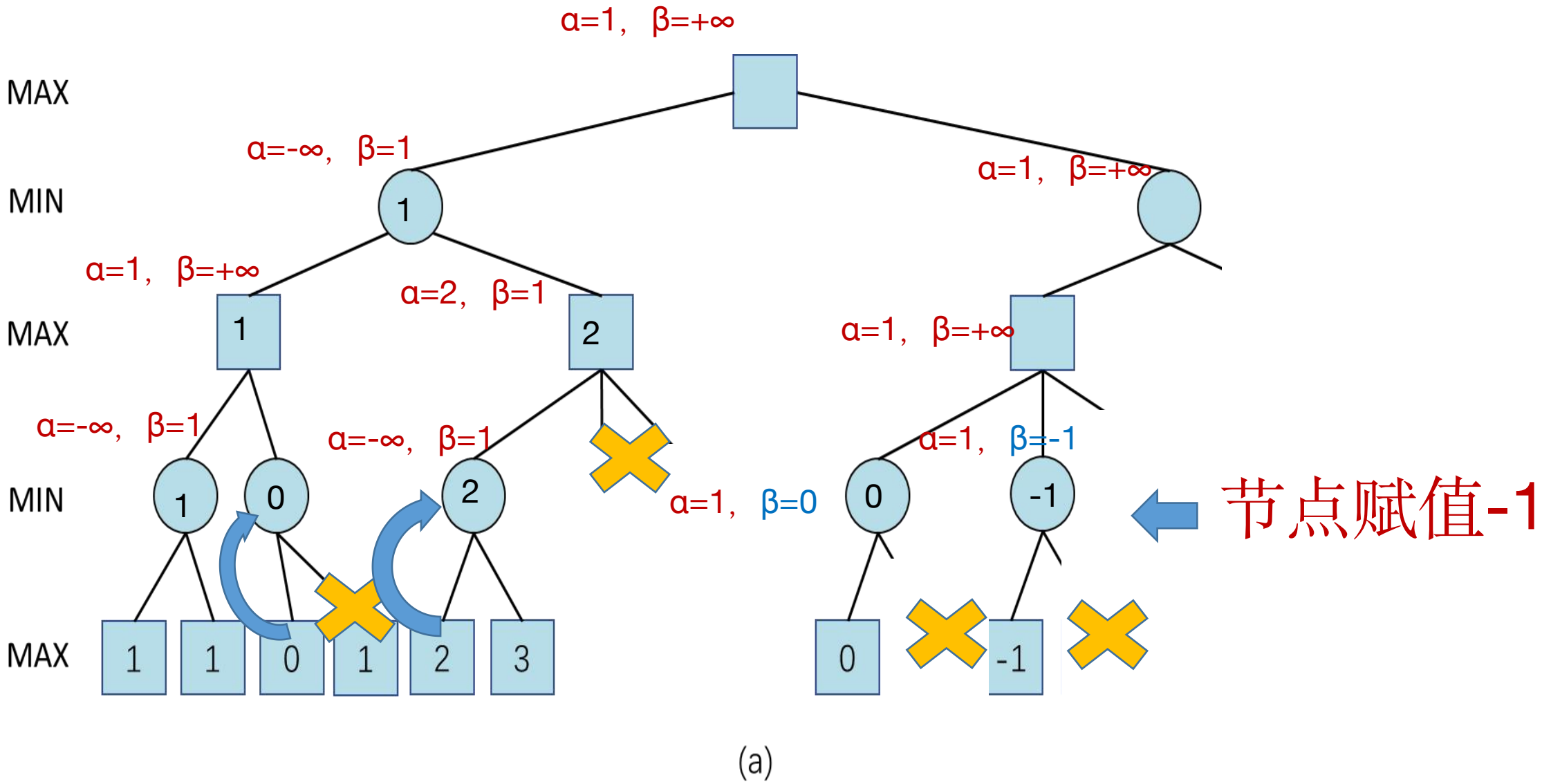
课上习题


α=1， β=+∞
MAX
α=-∞， β=1
MIN
1
α=1， β=+∞
α=2， β=1
MAX
1
2
α=-∞， β=1
α=-∞， β=1
α=1， β=0
MIN
1
0
2
MAX
1
1
0
1
2
3
α=1， β=+∞
α=1， β=+∞
α=1， β=-1
0
0
-1
-1<β,β更新-1
(a)

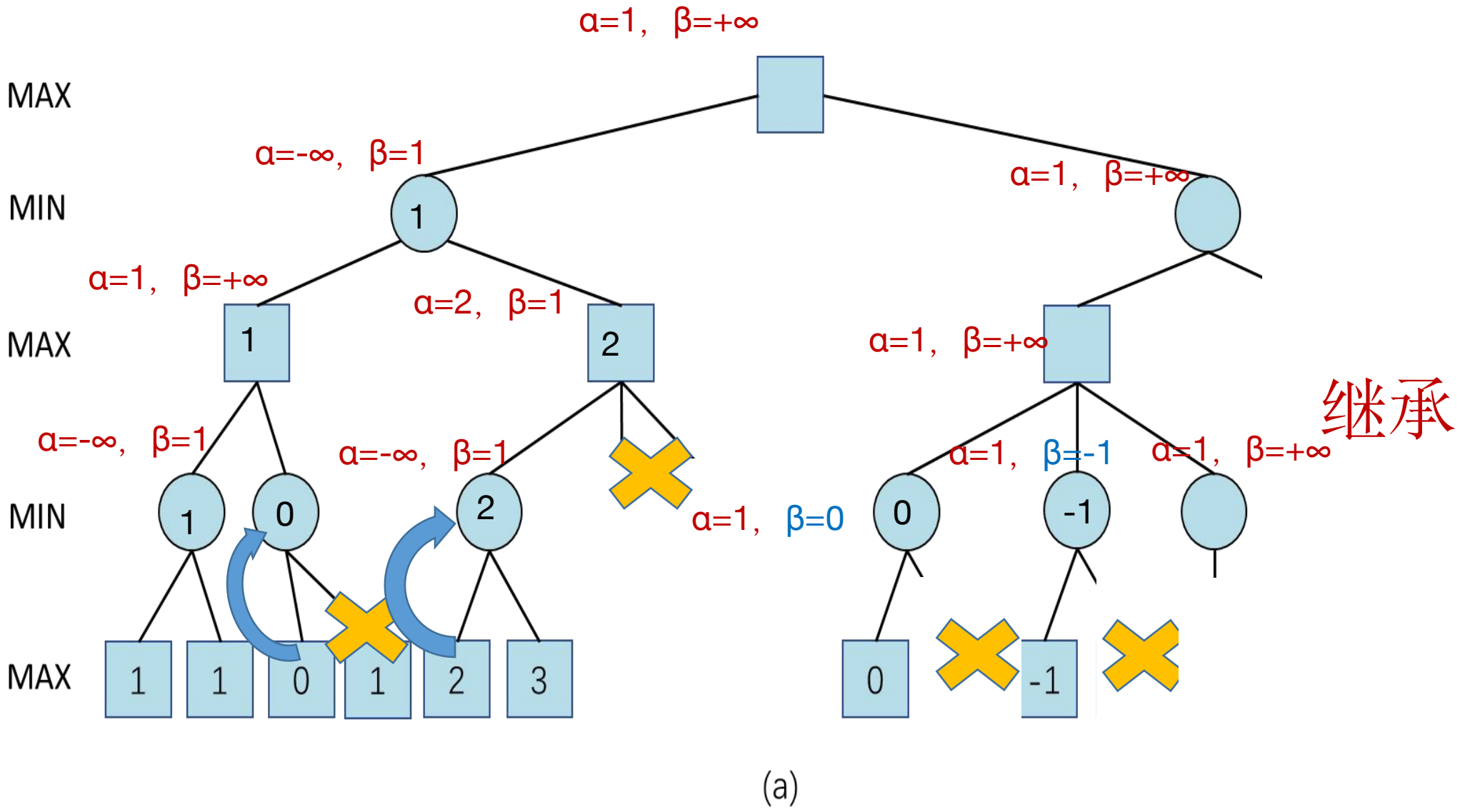
课上习题


α=1， β=+∞
MAX
α=-∞， β=1
MIN
1
α=1， β=+∞
α=2， β=1
MAX
1
2
α=-∞， β=1
α=-∞， β=1
α=1， β=0
MIN
1
0
2
MAX
1
1
0
1
2
3
α=1， β=+∞
α=1， β=+∞
α=1， β=-1
0
0
-1
-1<β,β更新-1
α>β,剪枝
(a)

课上习题

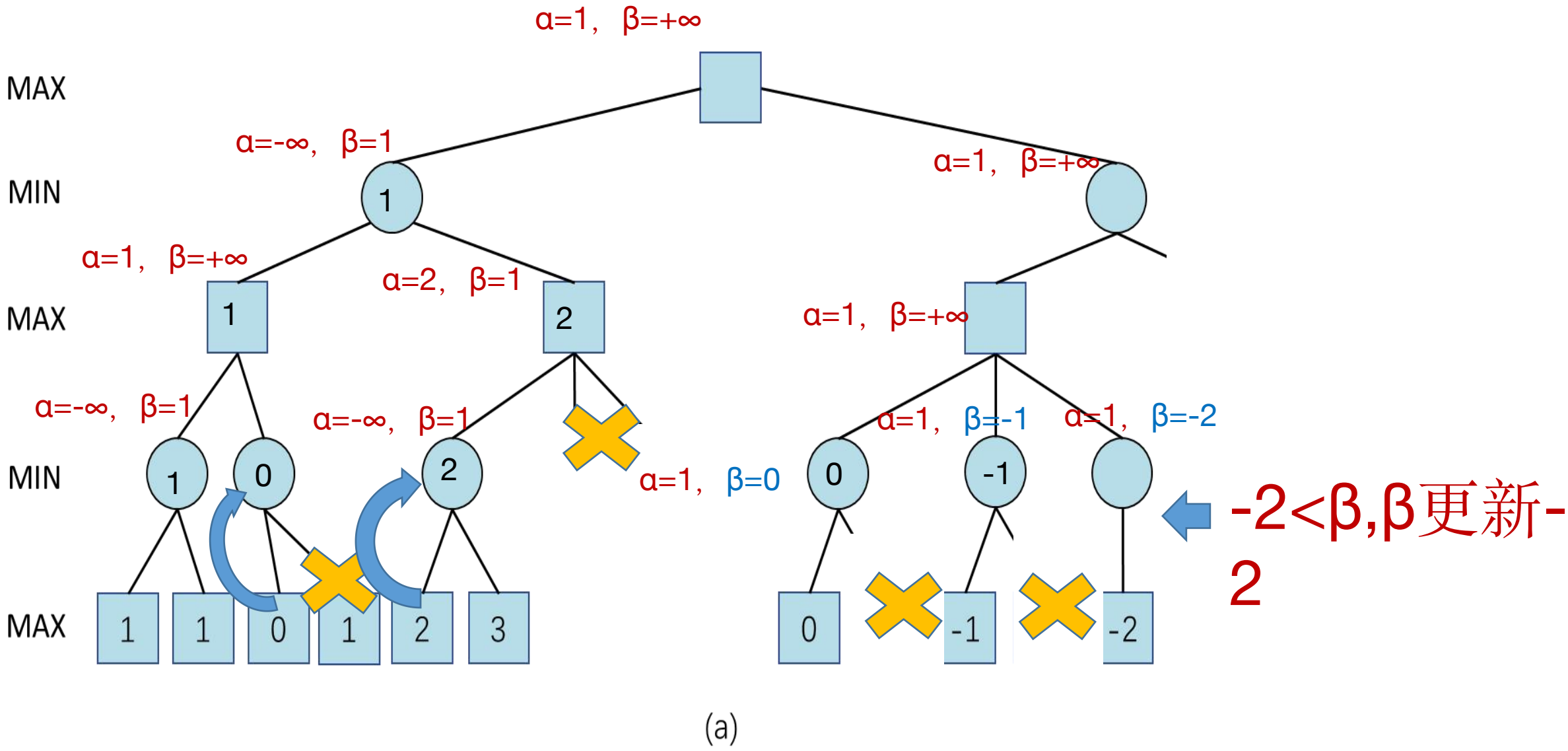

α=1， β=+∞
MAX
α=-∞， β=1
MIN
1
α=1， β=+∞
α=2， β=1
MAX
1
2
α=-∞， β=1
α=-∞， β=1
α=1， β=0
MIN
1
0
2
MAX
1
1
0
1
2
3
α=1， β=+∞
α=1， β=+∞
α=1， β=-1
0
-1
0
-1
节点赋值-1
(a)

课上习题

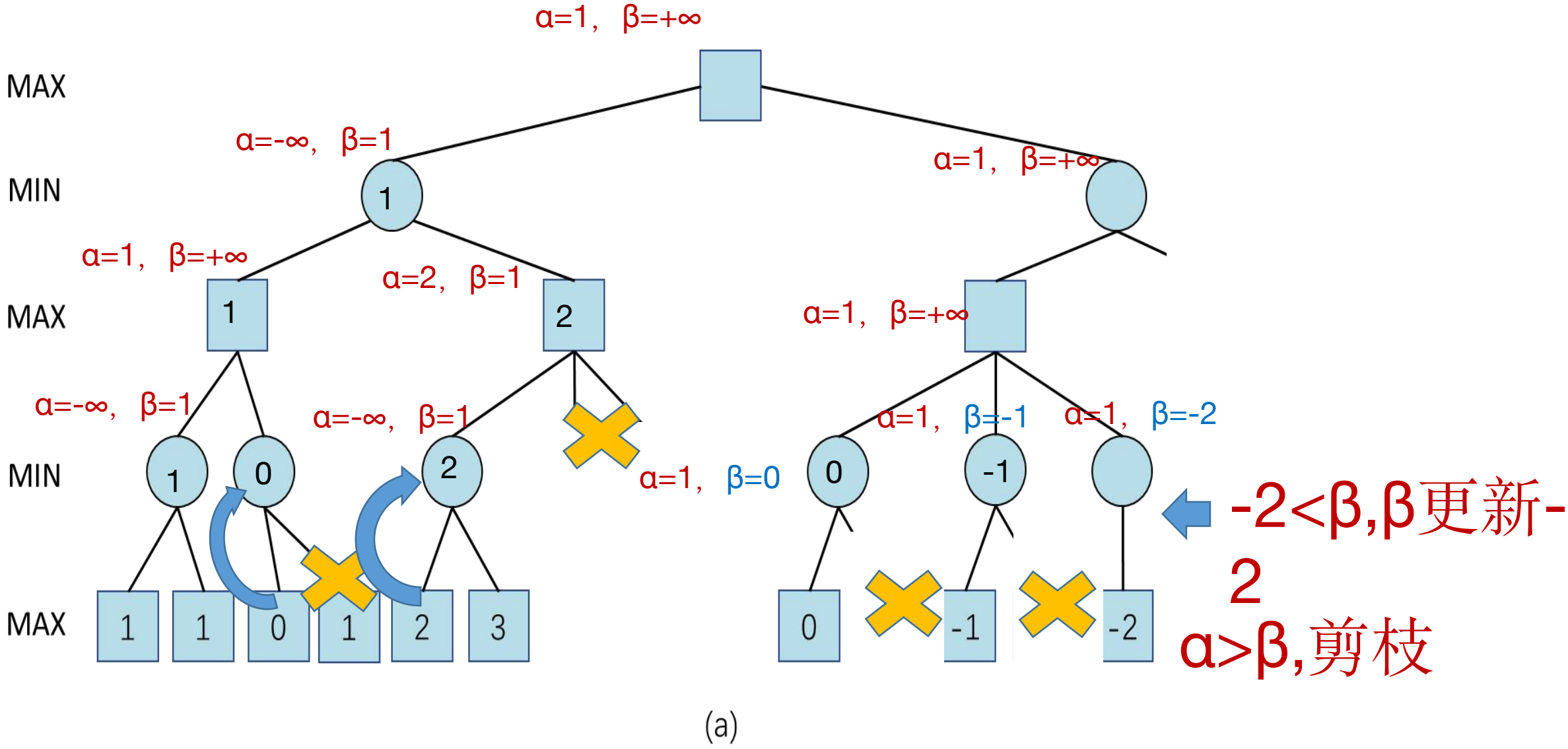

α=1, β=+∞
MAX
α=-∞, β=1
MIN
1
α=1, β=+∞
α=1, β=+∞
α=2, β=1
MAX
1
2
α=1, β=+∞
α=-∞, β=1
α=-∞, β=1
α=1, β=-1
α=1, β=+∞
MIN
1
0
2
α=1, β=0
0
-1
继承
MAX
1
1
0
1
2
3
0
-1

(a)

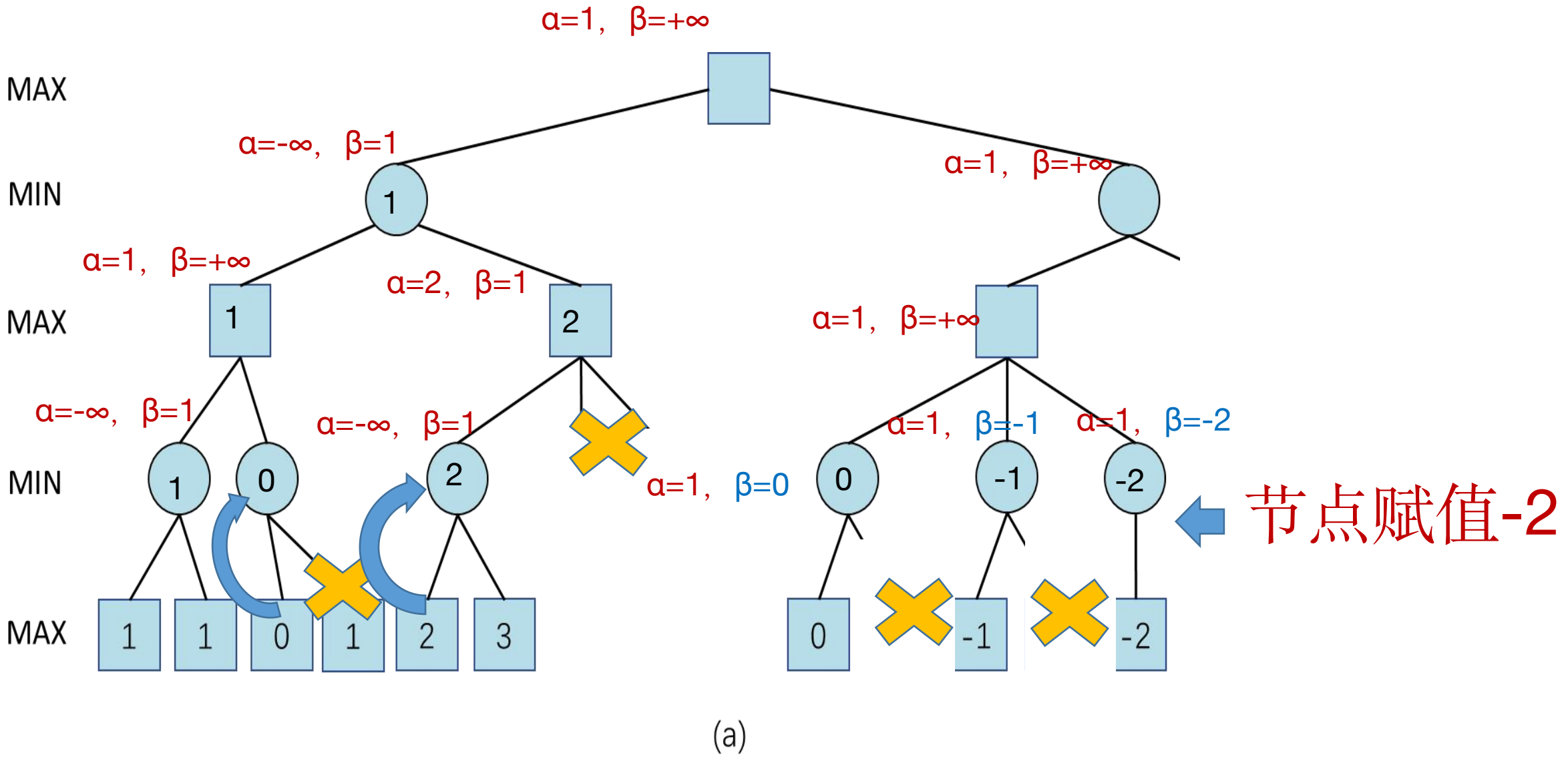
课上习题


α=1, β=+∞
MAX
α=-∞, β=1
MIN
α=1, β=+∞
α=1, β=+∞
α=2, β=1
MAX
α=1, β=+∞
α=-∞, β=1
α=-∞, β=1
α=1, β=-1
α=1, β=-2
MIN
α=1, β=0
1
0
2
0
-1
1
1
0
1
2
3
0
-1
-2
-2<β,β更新-2
(a)

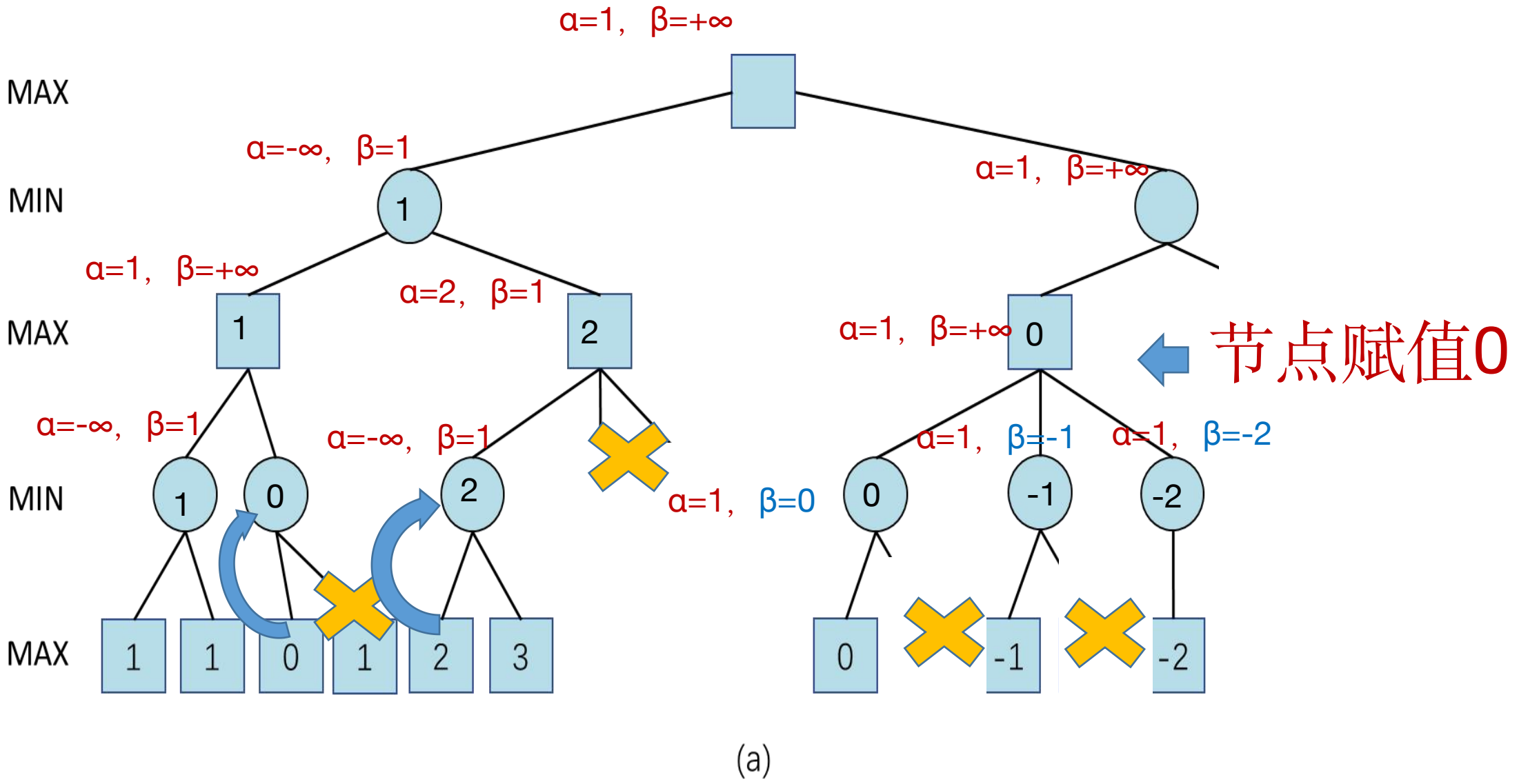
课上习题


α=1, β=+∞
MAX
α=-∞, β=1
MIN
1
α=1, β=+∞
α=1, β=+∞
α=2, β=1
MAX
1
2
α=1, β=+∞
α=-∞, β=1
α=-∞, β=1
α=1, β=-1
α=1, β=-2
MIN
1
0
2
α=1, β=0
0
-1
-2
MAX
1
1
0
1
2
3
0
-1
-2
-2<β,β更新-2
α>β,剪枝
(a)

课上习题

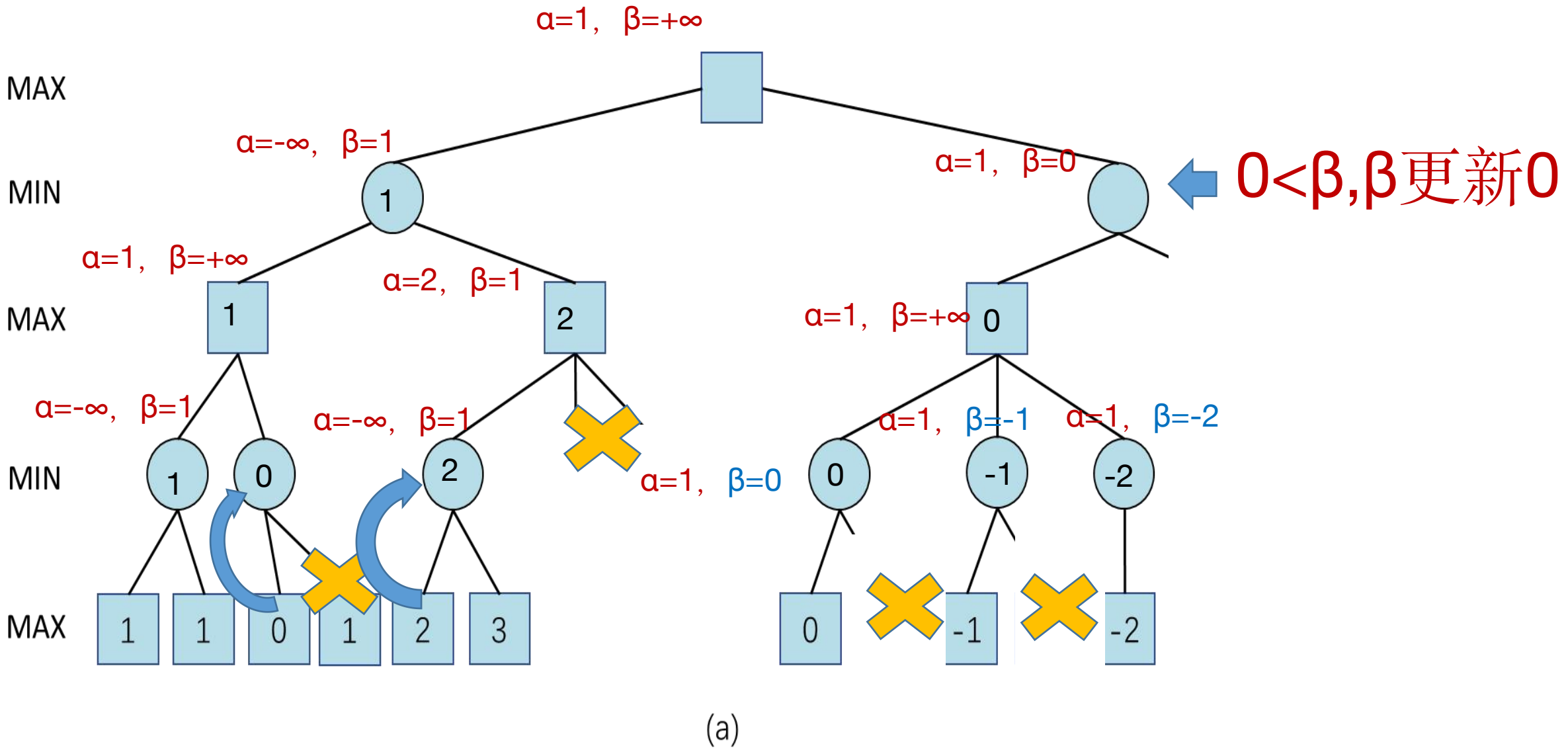

α=1, β=+∞
MAX
α=-∞, β=1
MIN
α=1, β=+∞
α=1, β=+∞
α=2, β=1
MAX
α=1, β=+∞
α=-∞, β=1
α=-∞, β=1
α=1, β=-1
α=1, β=-2
MIN
α=1, β=0
1
0
2
0
-1
-2
MAX
1
1
0
1
2
3
0
-1
-2
节点赋值-2
(a)

课上习题

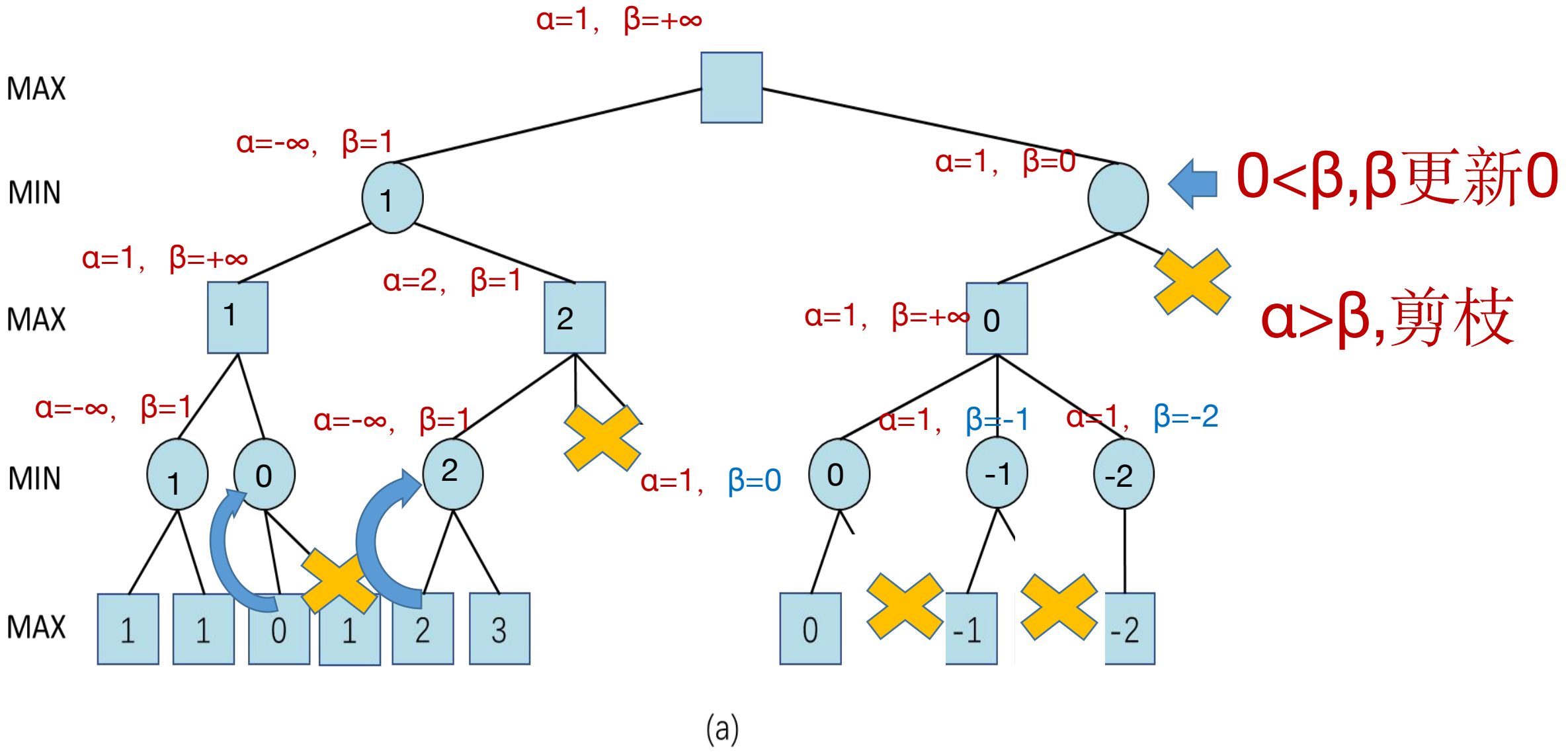

α=1, β=+∞
MAX
α=-∞, β=1
MIN
1
α=1, β=+∞
α=1, β=+∞
α=2, β=1
MAX
1
2
α=1, β=+∞
0
节点赋值0
α=-∞, β=1
α=-∞, β=1
α=1, β=-1
α=1, β=-2
MIN
1
0
2
α=1, β=0
0
-1
-2
MAX
1
1
0
1
2
3
0
-1
-2

(a)

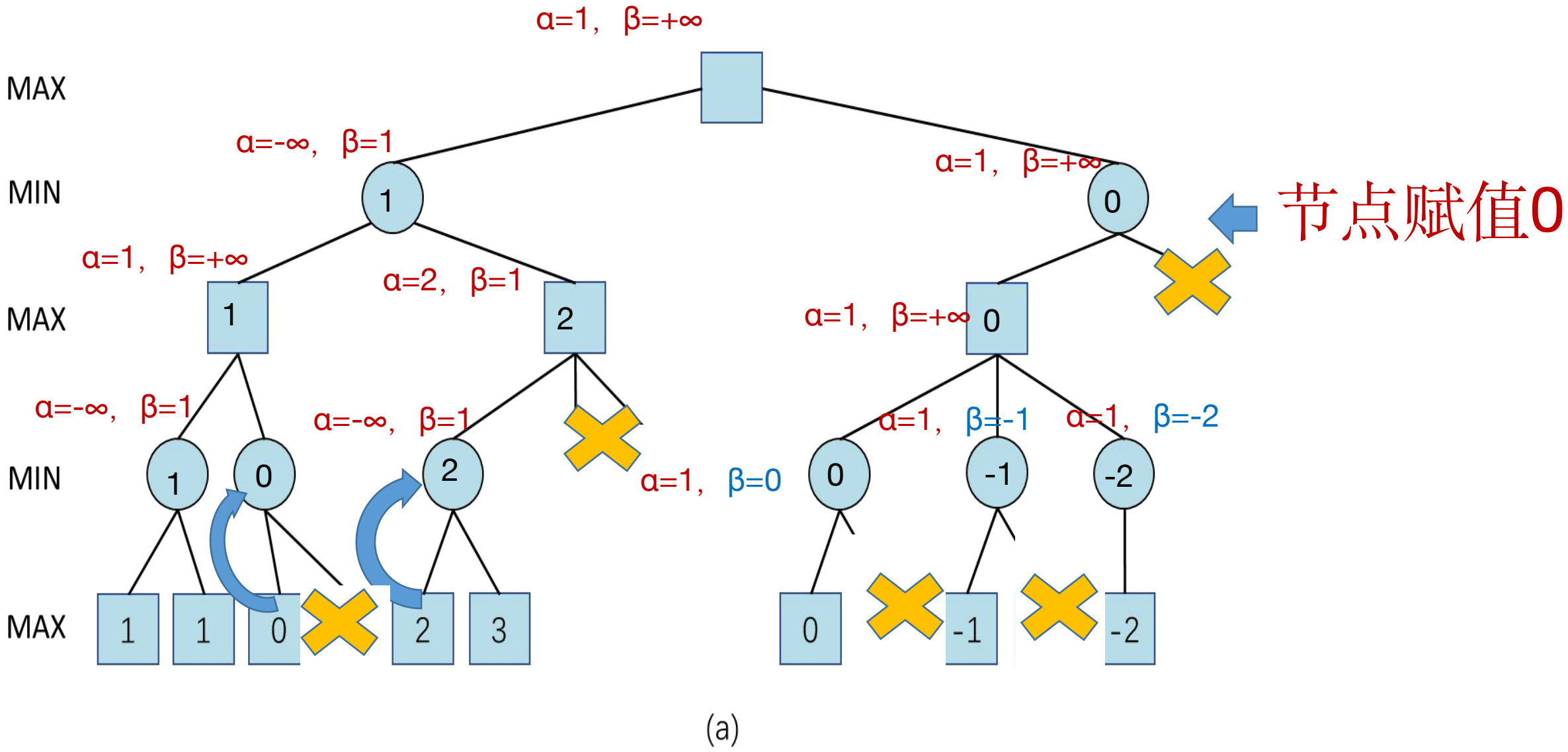
课上习题


α=1， β=+∞
MAX
α=-∞， β=1
MIN
1
α=1， β=0
0<β,β更新0
α=1， β=+∞
α=2， β=1
MAX
1
2
α=1， β=+∞
0
α=-∞， β=1
α=-∞， β=1
α=1， β=-1
α=1， β=-2
MIN
1
0
2
α=1， β=0
0
-1
-2
MAX
1
1
0
1
2
3
0
-1
-2
(a)

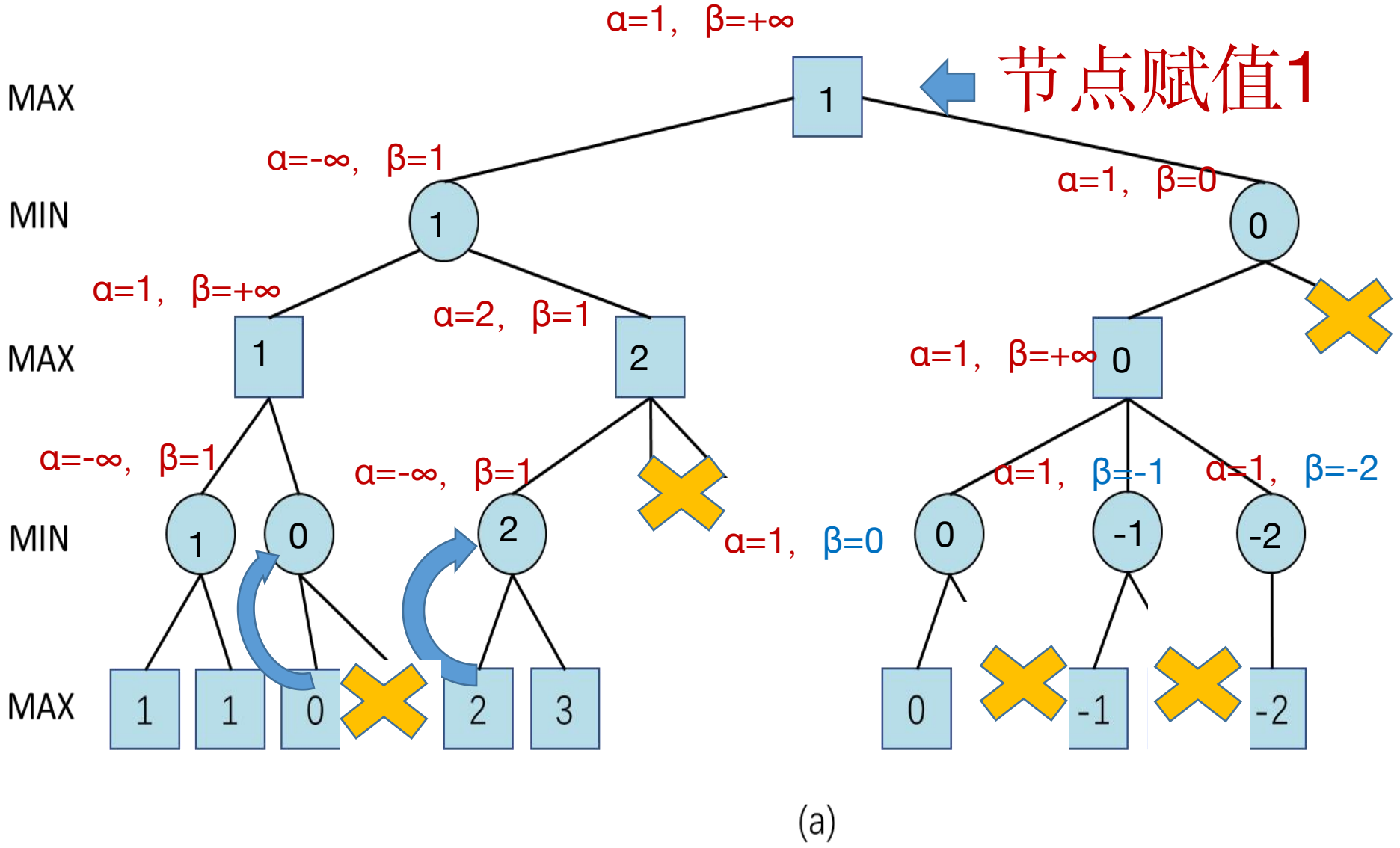
课上习题


α=1, β=+∞
MAX
α=-∞, β=1
MIN
1
α=1, β=+∞
α=2, β=1
MAX
1
2
α=-∞, β=1
α=-∞, β=1
α=1, β=0
MIN
1
0
2
MAX
1
1
0
1
2
3
α=1, β=0
0<β,β更新0
α=1, β=+∞
0
α=1, β=-1
α=1, β=-2
0
-1
-2
0
-1
-2
α>β,剪枝
(a)

课上习题

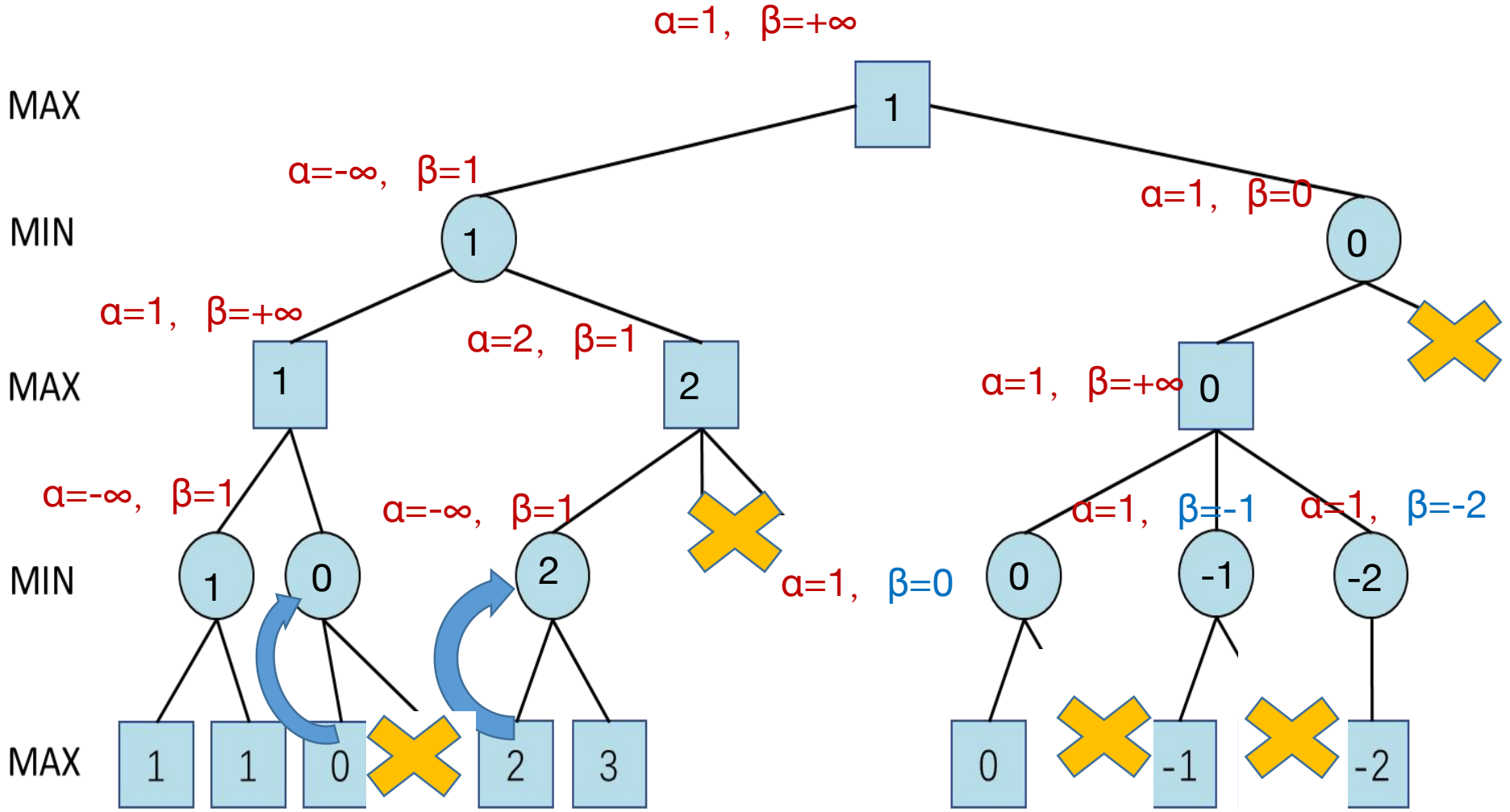

α=1, β=+∞
MAX
α=-∞, β=1
MIN
α=1, β=+∞
α=2, β=1
MAX
α=-∞, β=1
α=-∞, β=1
α=1, β=0
MIN
MAX
α=1, β=+∞
α=1, β=-1
α=1, β=-2
1
0
1
2
1
0
2
1
1
0
2
3
0
0
-1
-2
0
-1
-2
节点赋值0
(a)

课上习题


α=1, β=+∞
MAX
1
节点赋值1
α=-∞, β=1
MIN
1
α=1, β=0
0
α=1, β=+∞
α=2, β=1
MAX
1
2
α=1, β=+∞
0
α=-∞, β=1
α=-∞, β=1
α=1, β=-1
α=1, β=-2
MIN
1
0
2
α=1, β=0
0
-1
-2
MAX
1
1
0
2
3
0
-1
-2

(a)

课上习题


α=1, β=+∞
MAX
1
α=-∞, β=1
MIN
1
α=1, β=0
0
α=1, β=+∞
MAX
1
α=2, β=1
2
α=1, β=+∞
0
α=-∞, β=1
α=-∞, β=1
α=1, β=-1
α=1, β=-2
MIN
1
0
2
α=1, β=0
0
-1
-2
MAX
1
1
0
2
3
0
-1
-2

(a)

课上习题

图2展示了一棵Minimax搜索树，可采用alpha-beta剪枝算法进行对抗搜索。假设对于每个节点的后继节点，算法按照从左向右的方向扩展。同时假设当alpha值等于beta值时，算法不进行剪枝。请问：

- (1)对图2(a)中所示搜索树进行搜索，请画出在算法结束时搜索树的状态，用“×”符号标出被剪枝的子树，并计算该算法扩展的节点数量。

  (1) 扩展节点数量为20。