第5讲 博弈问题求解

尹慧琳，yinhuilin@tongji.edu.cn
同济大学 电子与信息工程学院

第 5 讲 博弈问题求解

- 搜索：寻求问题最优解的一种迭代方法

	经典的搜索方法	超越经典的优化搜索
问题	可观察、确定、已知	问题的复杂度和不确定性更高——优化
方法	系统地探索问题空间	局部搜索、迭代趋优

- 对抗搜索（Adversarial Search）/ 博弈（Game）
  - 两个或多个智能主体，按照预定的规则和各自的策略，从中取得有利于己方的结果或收益的行为

多Agent环境；每个Agent的目标之间是有冲突的

第 5 讲 博弈问题求解

5.1 博弈问题

5.2 博弈问题的类型

5.3 博弈问题的求解

5.4 博弈中的优化决策

5.5 博弈中的实时决策

第5讲 博弈问题求解

5.1 博弈问题

- 博弈问题三要素：
  - 局中人（玩家）：局中人集合 $I=\{1,2,\ldots,n\}$
    - 有权决定自己行动方案的博弈对策参加者
    - 假定是理智的、聪明的
  - 策略集
    - 策略：可供局中人选择的一个实际可行的完整的行动方案
    - 策略集S_i：局中人 i 的所有行动方案的集合
    - 局势S：一局对策中，各局中人选定的策略形成的策略组$S=(S_1, S_2, \ldots, S_n)$
  - 赢得函数/收益：局人可得到的赢得值
- 挑战性：
  - 复杂且不确定环境下的最优决策
  - 在有限的时间内进行决策

第5讲 博弈问题求解

5.1 博弈问题

- 举例：人工智能研究的常见游戏类博弈问题

棋盘游戏（Board games）	井字棋（Tic-tac-toe）
	西洋跳棋（Checkers）
	国际象棋（Chess）
	围棋（Go）
纸牌游戏（Card games）	德州扑克（Texas Hold'em）
	桥牌（Bridge）
骰子游戏（Dice games）	僵尸骰子（Zombie Dice）
棋牌游戏（Tile games）	麻将（Mahjong）
猜拳游戏（Hand games）	石头剪子布（Rock-paper-scissors）

第5讲 博弈问题求解

5.1 博弈问题

- 举例：囚徒困境（Prisoner's dilemma）

两个犯罪分子被分别监禁，无法沟通。每个囚徒只有二选一的机会：揭发对方并证明其犯罪，或者保持沉默。可能的选项如下：

- 1）若囚徒A和B彼此揭发对方，则每个囚徒被监禁7年；
- 2）若A揭发B、而B保持沉默，则A被释放而B被监禁9年，反之亦然；
- 3）若A和B都保持沉默，则他们仅被监禁1年。

（A获罪，B获罪）		囚徒A	
		坦白	不坦白
囚徒B	坦白	(-7, -7)	(-9, 0)
	不坦白	(0, -9)	(-1, -1)

第5讲 博弈问题求解

5.1 博弈问题

- 最后通牒博弈（Ultimatum game, UG）
  - 两人分一笔钱，一人是提议者（proposer），另一人是响应者（responder）。提议者提出分钱方案，响应者可以选择同意与否
  - 若响应者同意，就按提议者的方案来分；若不同意，则两人都将一无所得。
- 海盗分金难题（Pirate Puzzle）

  5个海盗抢得100枚金币，约定按顺序每人提出一个金币分配方案，然后大家表决。规则如下：
  - 提案若超过半数同意，分配金币后散伙；
  - 若是票数相等，则提案人有决定权；
  - 若这两种情况都未成立，则提案人将被扔入大海
  - 依此类推，重复上述规则。

第 5 讲 博弈问题求解

5.1 博弈问题

5.2 博弈问题的类型

5.3 博弈问题的求解

5.4 博弈中的优化决策

5.5 博弈中的实时决策

第5讲 博弈问题求解

5.2 博弈问题的类型

- 合作博弈 vs 非合作博弈
  - 合作性博弈（Cooperative game）

    参与者之间达成协议或形成联盟，其结果对各方均有利。
  - 非合作性博弈（Non-cooperative game）

    参与者无法形成约束性协议或联盟，因而选择对抗性的行为。
- 零和博弈 vs 非零和博弈
  - 零和博弈（Zero-sum game）

    一方的收益是另一方的损失，或者双方的结果是平局。博弈中各方的收益和损失之和永远为零。
  - 非零和博弈（Non-zero-sum game）

    各方的收益总和小于或大于零，即非零。三种情况：win-win, double-win, lose-lose。

第5讲 博弈问题求解

5.2 博弈问题的类型

- 完美信息博弈 vs 不完美信息博弈
  - 完美信息（Perfect information）博弈

    每个参与者在博弈过程中完全了解所有的信息。大多数棋盘游戏，例如国际象棋、围棋等都属于完美信息博弈。
  - 不完美信息（Imperfect information）博弈

    反之。例如：扑克和桥牌之类的纸牌游戏、麻将等棋牌游戏等。

计算机超过人类	计算机与顶级人类玩家媲美
国际象棋Chess：IBM深蓝1997年	桥牌
围棋Go：谷歌AlphaGo2016年	西洋双陆棋

第5讲 博弈问题求解

5.2 博弈问题的类型

- 完美信息博弈 vs 不完美信息博弈
  - 完美信息（Perfect information）博弈

    每个参与者在博弈过程中完全了解所有的信息。大多数棋盘游戏，例如国际象棋、围棋等都属于完美信息博弈。
  - 不完美信息（Imperfect information）博弈

    反之。例如：扑克和桥牌之类的纸牌游戏、麻将等棋牌游戏等。
- 完全信息 vs 与不完全信息博弈
  - 完全信息（Complete information）指的是每个参与者都具有该博弈所需要的完整知识，例如规则、效用、收益、策略等。
  - 典型的具有不完美信息但具有完全信息的博弈，包括扑克、桥牌、以及麻将等

第5讲 博弈问题求解

5.2 博弈问题的类型

- 对称博弈 vs 非对称博弈
  - 对称博弈（Symmetric games）：参与者在博弈中采取与对手相同的策略时能得到同样的收益。例如：“囚徒困境”
  - 非对称博弈（Asymmetric games）：反之。例如：“最后通牒”
- 随机博弈 vs 非随机博弈
  - 随机博弈（Stochastic game）：具有概率变迁（Probabilistic transitions）性质的动态游戏。例如：单人的老虎机、多人的掷骰子游戏
- 同步博弈 vs 顺序博弈
  - 同步博弈（Simultaneous game）：所有的博弈者同时选择自己的动作。
  - 顺序博弈（Sequential game）：博弈者轮流、交替动作的博弈。后动作博弈者可对先动作博弈者的行为有所了解，但未必能得到的完美信息。

第 5 讲 博弈问题求解

5.1 博弈问题

5.2 博弈问题的类型

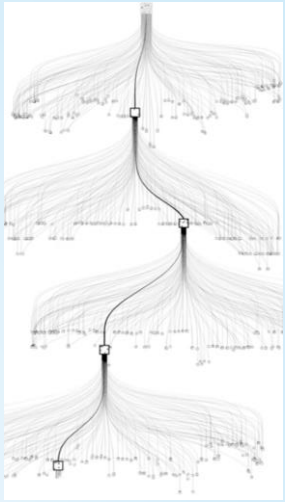
5.3 博弈问题的求解

5.4 博弈中的优化决策

5.5 博弈中的实时决策

第5讲 博弈问题求解

5.3 博弈问题的求解：博弈的复杂性



- 问题空间巨大
  - 约翰 麦卡锡曾说：国际象棋的角色就像是人工智能的果蝇
  - 围棋是棋盘类游戏中最为复杂的

棋类	棋盘格子数	树的分支数	树的深度	可能走子数	可能棋局数
国际象棋	64	≈ 35	≈ 80	35^{80}	10^{120}
围棋	361	≈ 250	≈ 150	250^{150}	10^{170}

- 决策时间有限
  - 对于现实世界的博弈，即使在无法计算出最优决策的情况下，也需要能够给出某种决策的能力
  - 现实问题中，对决策的时间也有所要求

第5讲 博弈问题求解

5.3 博弈问题的求解：对抗搜索

- 对抗搜索（Adversarial search）是博弈问题求解的一个重要方法

	经典搜索	对抗搜索
环 境	单智能主体	多智能主体
搜索方式	启发式搜索	对抗式搜索
优 化	用启发式方法可找到最优解	因时间受限而被迫执行近似解
评价函数	路径的代价估计	博弈策略和局势评估

- 问题形式化

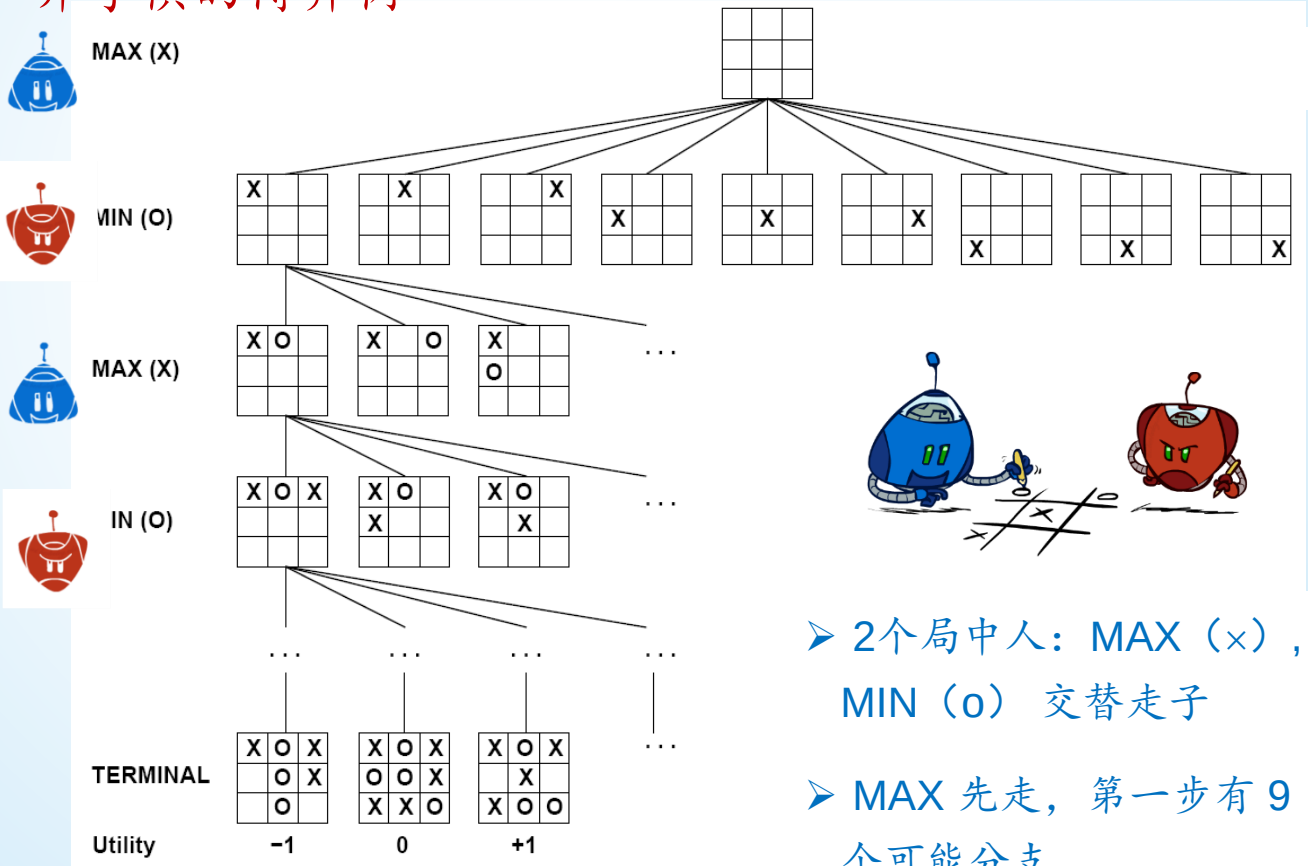
以2个玩家、确定性、完美信息博弈为例

S_0	初始状态，规定博弈开始时的设定
Player(s)	定义此时由哪个玩家采取行动
Actions(s)	返回此状态下的合法动作集合
Result(s, a)	转换模型，定义一步动作的结果
Terminal-Test(s)	终止测试，博弈结束时为 true，否则为 false
Utility(s, p)	效用/收益函数，定义玩家 p在状态 s下的值

第5讲 博弈问题求解

5.3 博弈问题的求解：对抗搜索

- 举例：井字棋的博弈树



- 2个局中人：MAX（×），MIN（o） 交替走子
- MAX 先走，第一步有 9 个可能分支

第 5 讲 博弈问题求解

5.1 博弈问题

5.2 博弈问题的类型

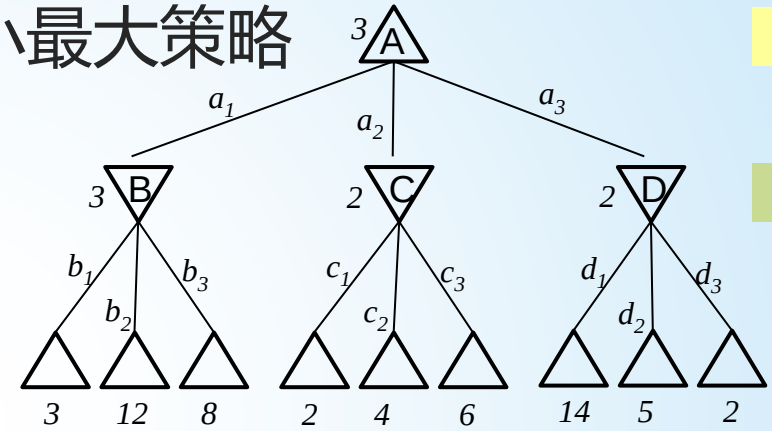
5.3 博弈问题的求解

5.4 博弈中的优化决策

5.5 博弈中的实时决策

第5讲 博弈问题求解

5.4 博弈中的优化决策：最小最大策略



- 举例
  - 局中人：MAX，MIN（零和博弈）
  - MAX的策略集：a_1, a_2, a_3
  - MIN的策略集：$\{b_1, b_2, b_3\}$, ...

$$MINIMAX(s) = \begin{cases} UTILITY(s) & s\text{为终止状态} \\ max_{a \in Actions(s)} MINIMAX\big(RESULT(s,a)\big) & s\text{为}MAX\text{节点} \\ min_{a \in Actions(s)} MINIMAX\big(RESULT(s,a)\big) & s\text{为}MIN\text{节点} \end{cases}$$

- 最小最大策略（Minimax strategy）

当存在两种相互冲突的策略时，己方应采取的策略是：

- 最小收益最大化
- 最大损失最小化

第5讲 博弈问题求解

5.4 博弈中的优化决策：最小最大策略

- 最小最大定理（Minimax theorem）

  对于两个玩家和有限多个策略的零和博弈，每个玩家存在一个值V和一个混合策略，使得：

  (a)给定玩家2的策略，则玩家1可能的最好收益是V，

  (b)给定玩家1的策略，则玩家2可能的最好收益是-V。

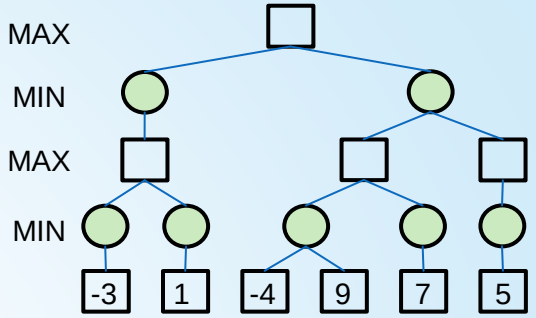
- 最小最大算法

  假定在对手执行最佳动作的前提下，最小化己方损失；或者在各种获得最小收益的策略中选择有最大收益的策略（又可称为最大最小算法）

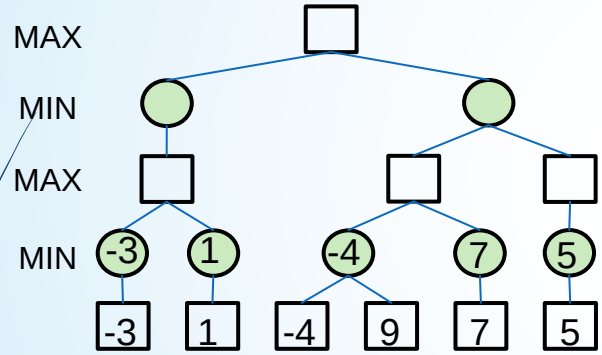
  - 基于最小最大定理实现的博弈算法
  - 以放弃最优策略为代价的保守方法
  - 适用于零和博弈问题

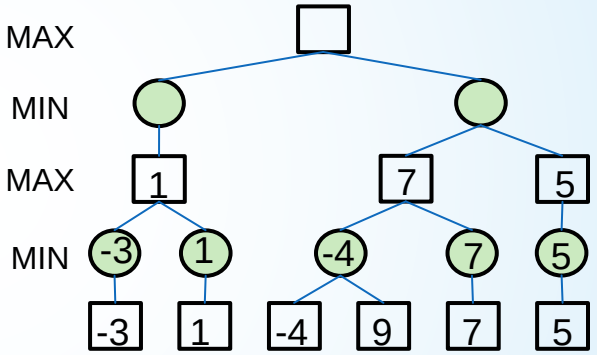
第5讲 博弈问题求解

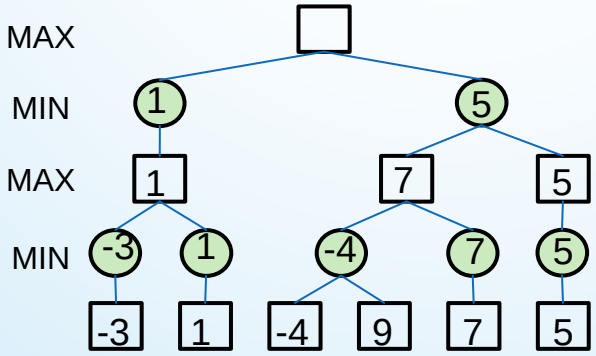
5.4 博弈中的优化决策：最小最大策略

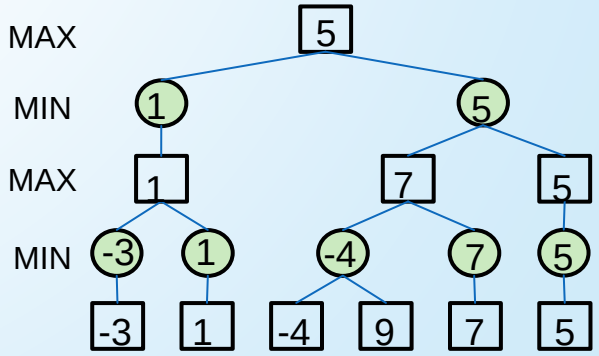


- 举例：2个玩家4层（二步）博弈树





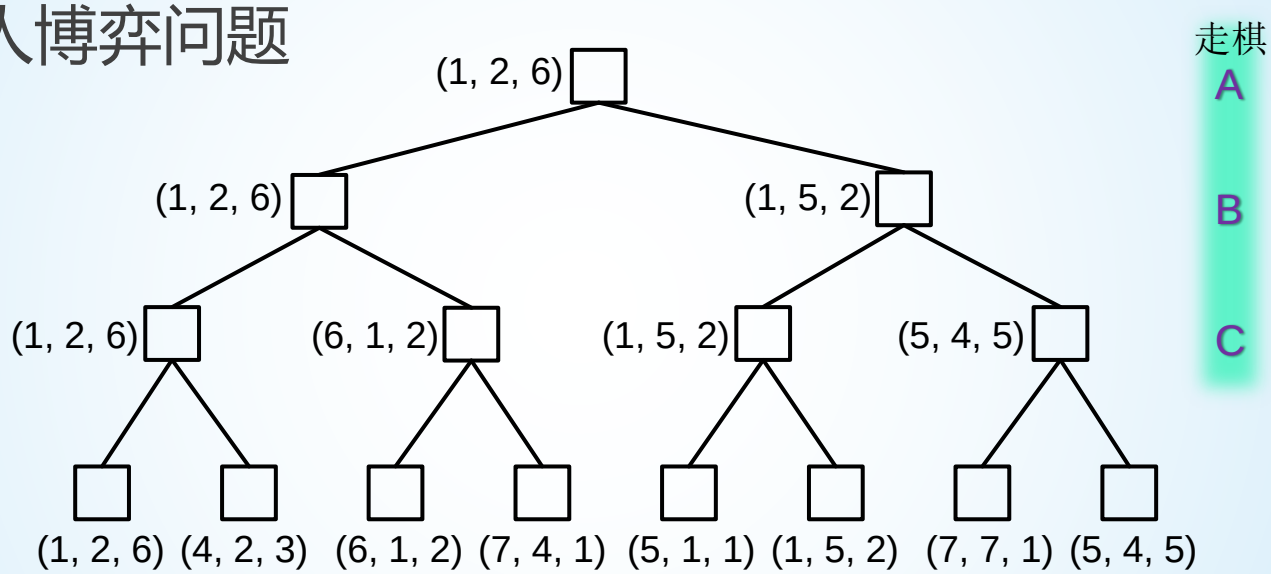




第5讲 博弈问题求解

5.4 博弈中的优化决策：多人博弈时的最优决策

- 多人博弈问题

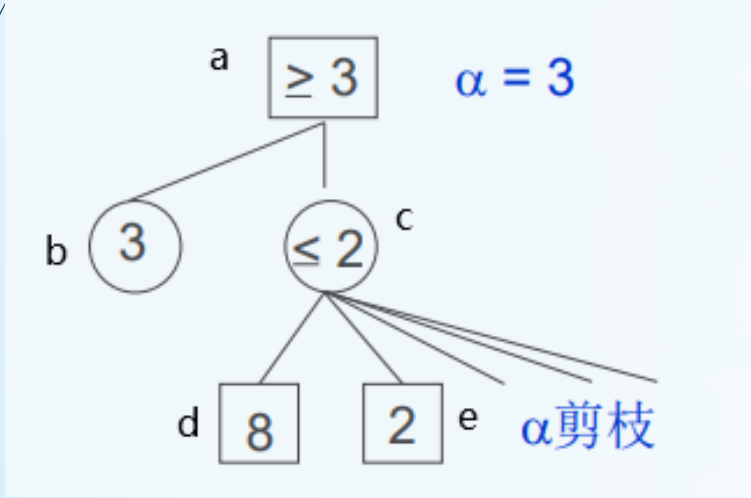


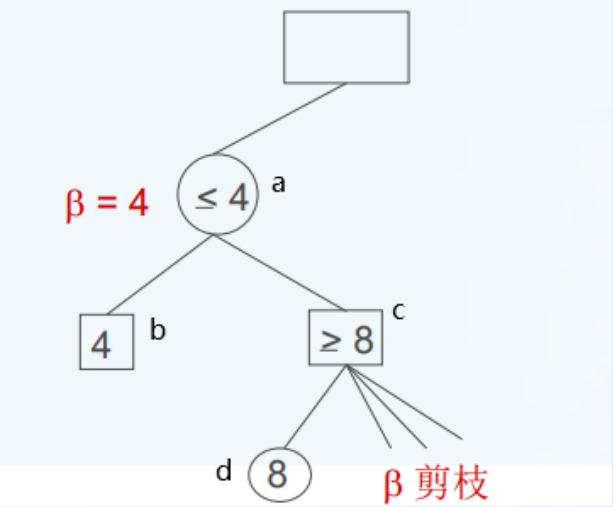
- 效用值向量（v_a, v_b, v_c），每位选手的决策依据相对应的效用值
- 选手之间有可能出现正式或非正式联盟。随着游戏的进行，联盟不断建立或者解散。
- 多人博弈可能同时涉及对抗和合作；决策亦可能同时涉及直接和间接利益的权衡

第5讲 博弈问题求解

5.4 博弈中的优化决策：α-β剪枝

- α为MAX方的最小得分，β为MIN方的最大得分
- 在对子节点进行搜索时，子节点是否需要进一步展开搜索受到其兄弟节点值的影响。

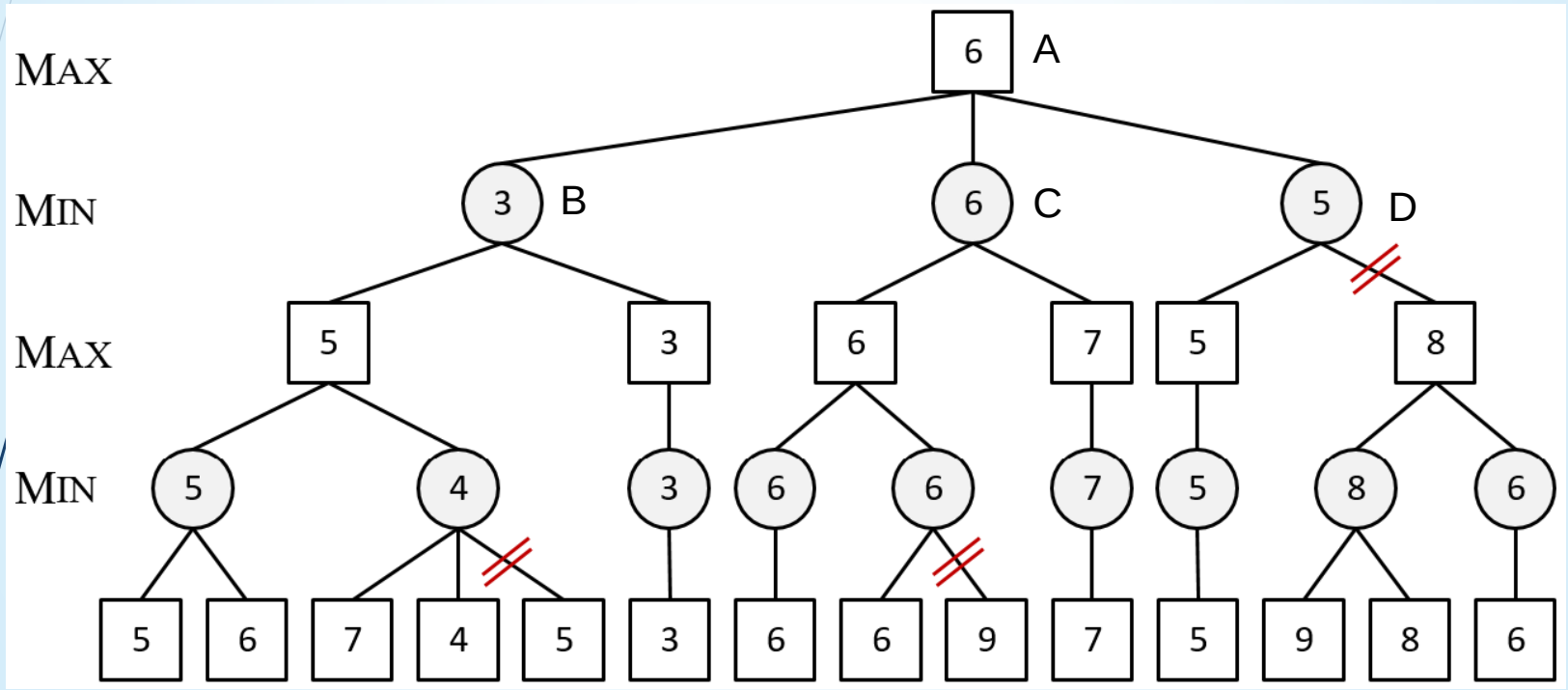




（矩形为MAX节点，圆形为MIN节点）

第5讲 博弈问题求解

5.4 博弈中的优化决策： α-β剪枝



第5讲 博弈问题求解

5.4 博弈中的优化决策：α-β剪枝

- 过程伪代码（下页）

function ALPHA-BETA-SEARCH(state) **returns** an action

$v \leftarrow$ MAX-VALUE($state$, $-\infty, +\infty$)

return the action in ACTIONS($state$) with value v

function MAX-VALUE(state, α, β) **returns** a *utility value*

if TERMINAL-TEST($state$) then return UTILITY($state$)

$v \leftarrow -\infty$

for each a **in** ACTIONS(state) **do**

$v \leftarrow \text{MAX}(v, \text{MIN} - \text{VALUE}(\text{Result}(s, a), \alpha, \beta))$

$\alpha \leftarrow MAX(\alpha, v)$

if $\alpha \geq \beta$ **then** β -pruning

return v

function MIN-VALUE(state, α, β) **returns** a *utility value*

if TERMINAL-TEST($state$) then return UTILITY($state$)

$v \leftarrow +\infty$

for each a **in** ACTIONS(state) **do**

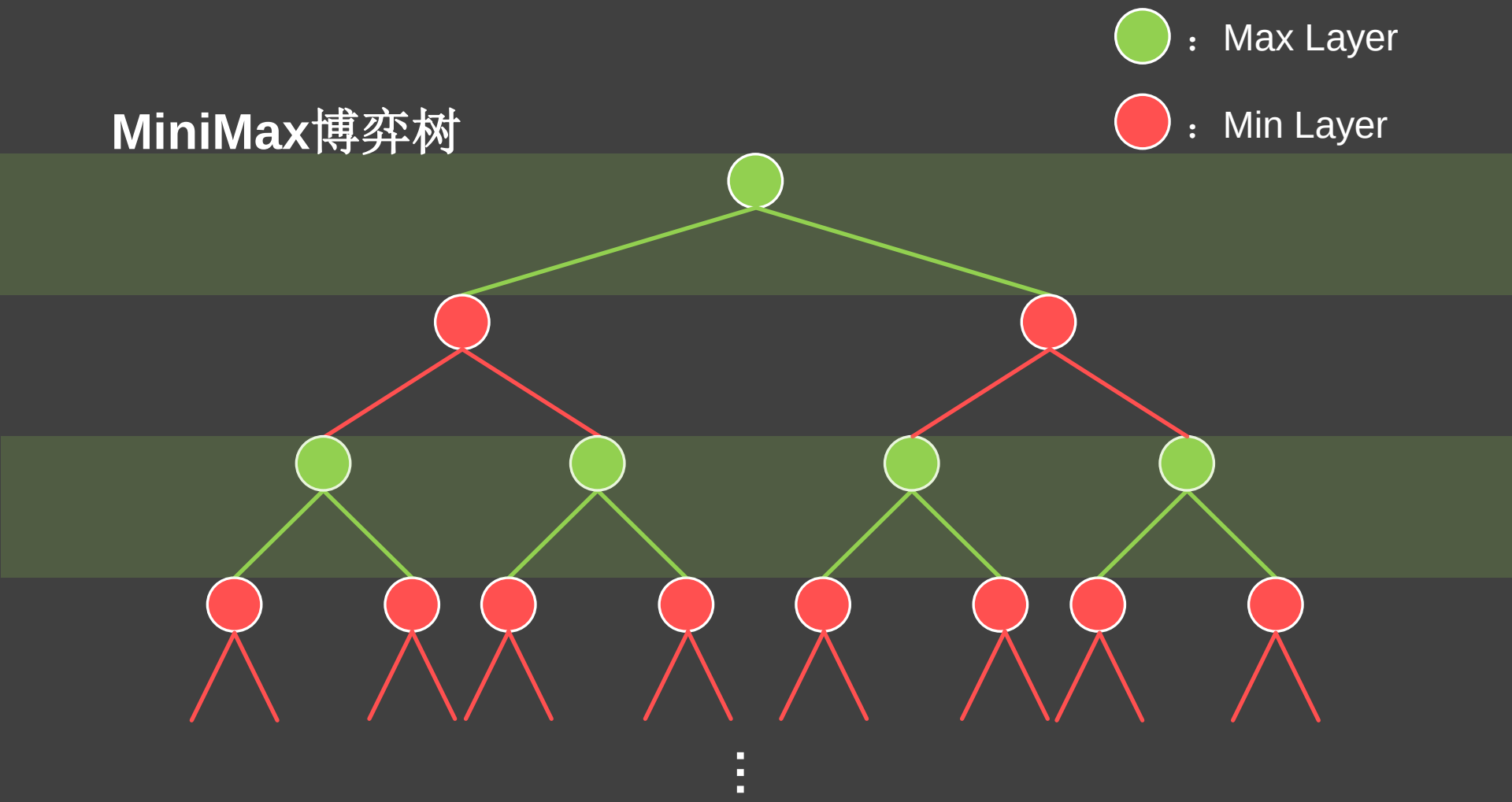
$v \leftarrow \text{MIN}(v, \text{MAX} - \text{VALUE}(\text{Result}(s, a), \alpha, \beta))$

$\beta \leftarrow MIN(\beta, v)$

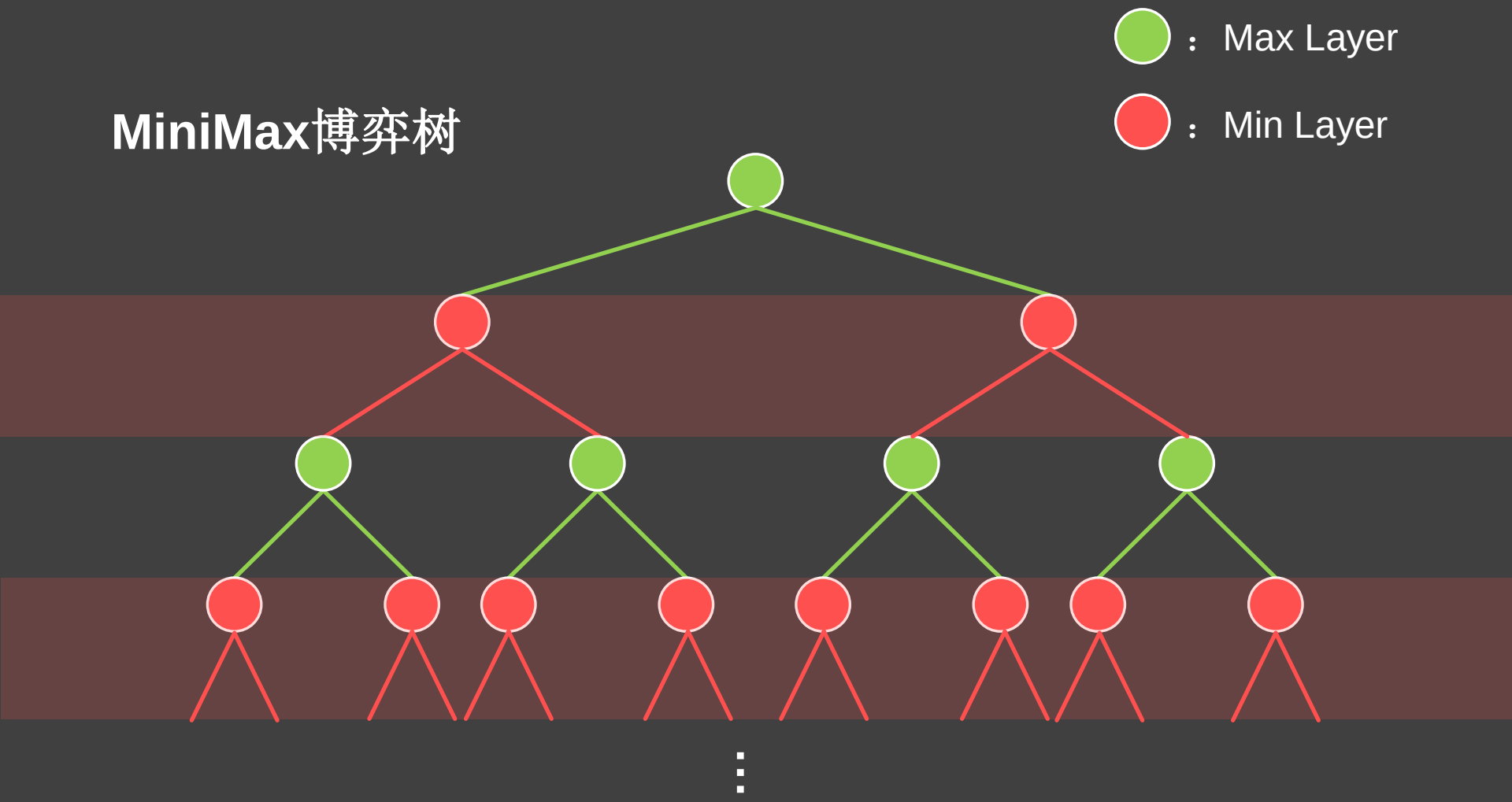
if $\beta \leq \alpha$ **then** α -pruning

return v

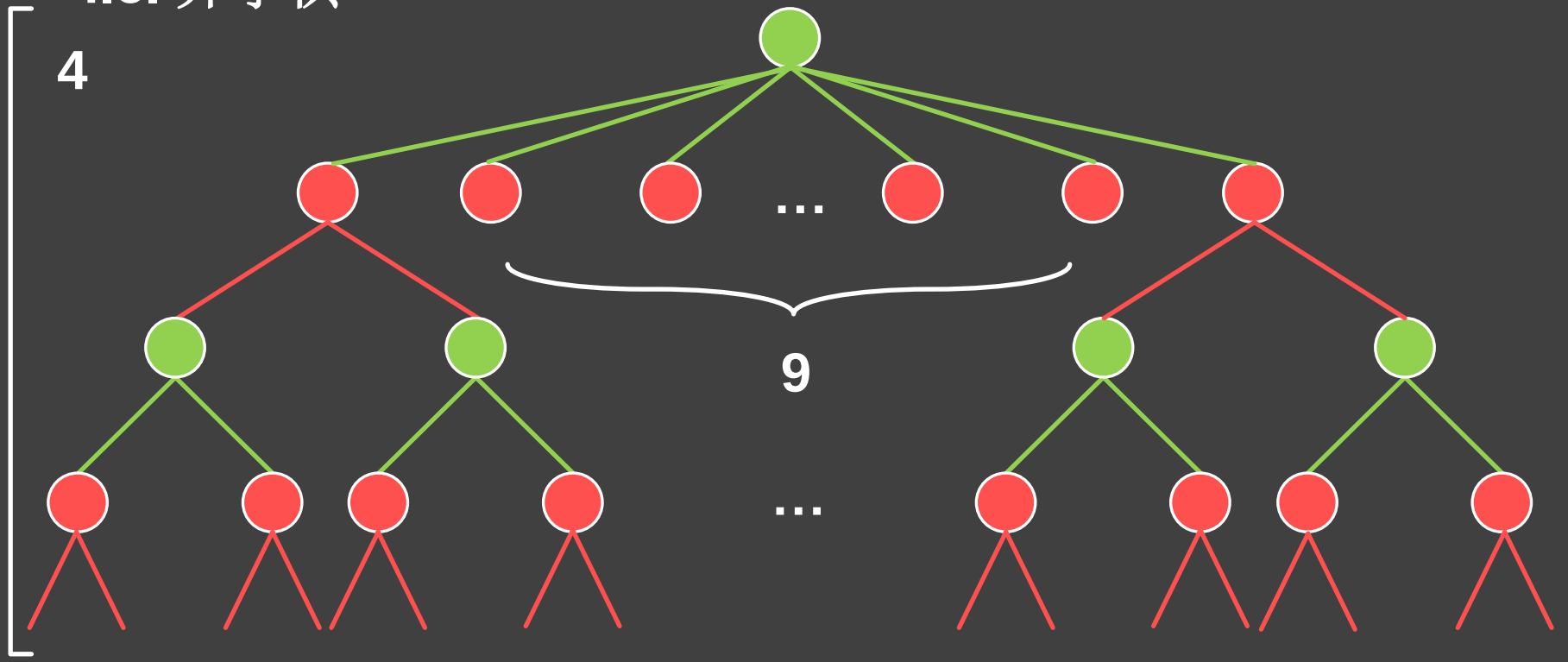
MiniMax博弈树

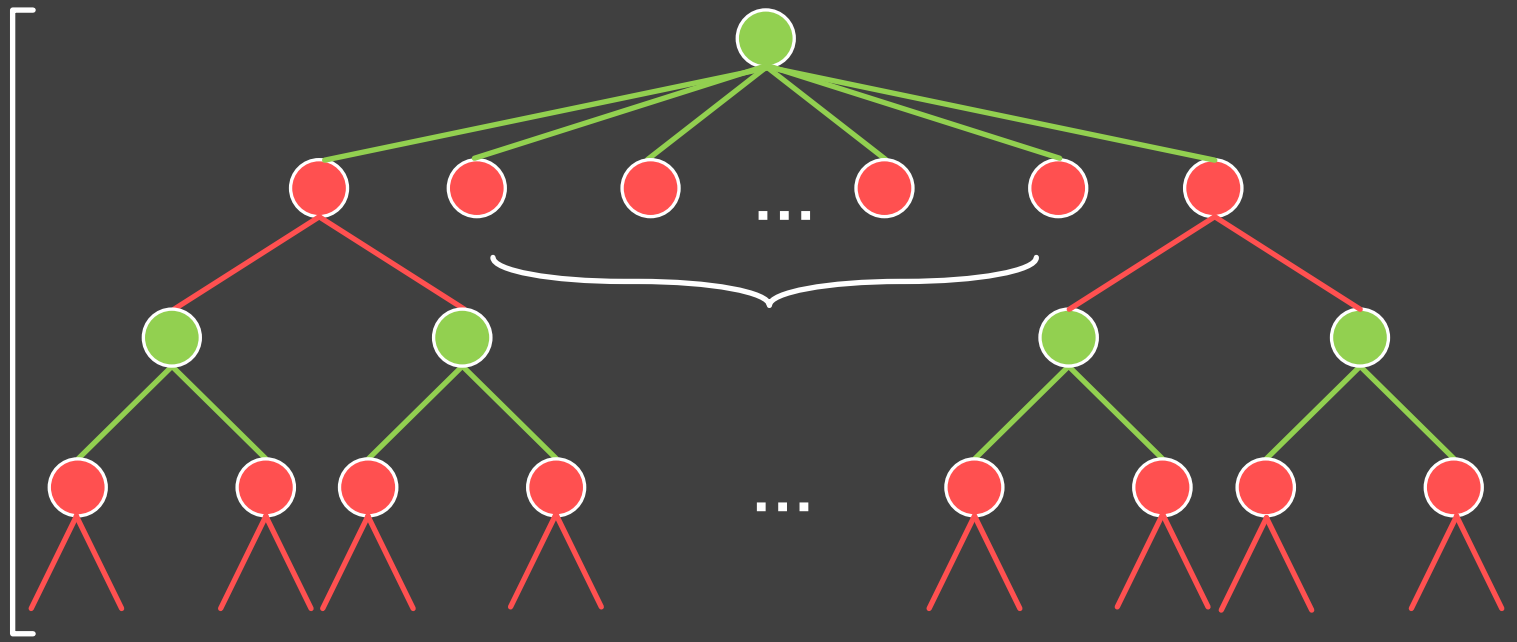


MiniMax博弈树

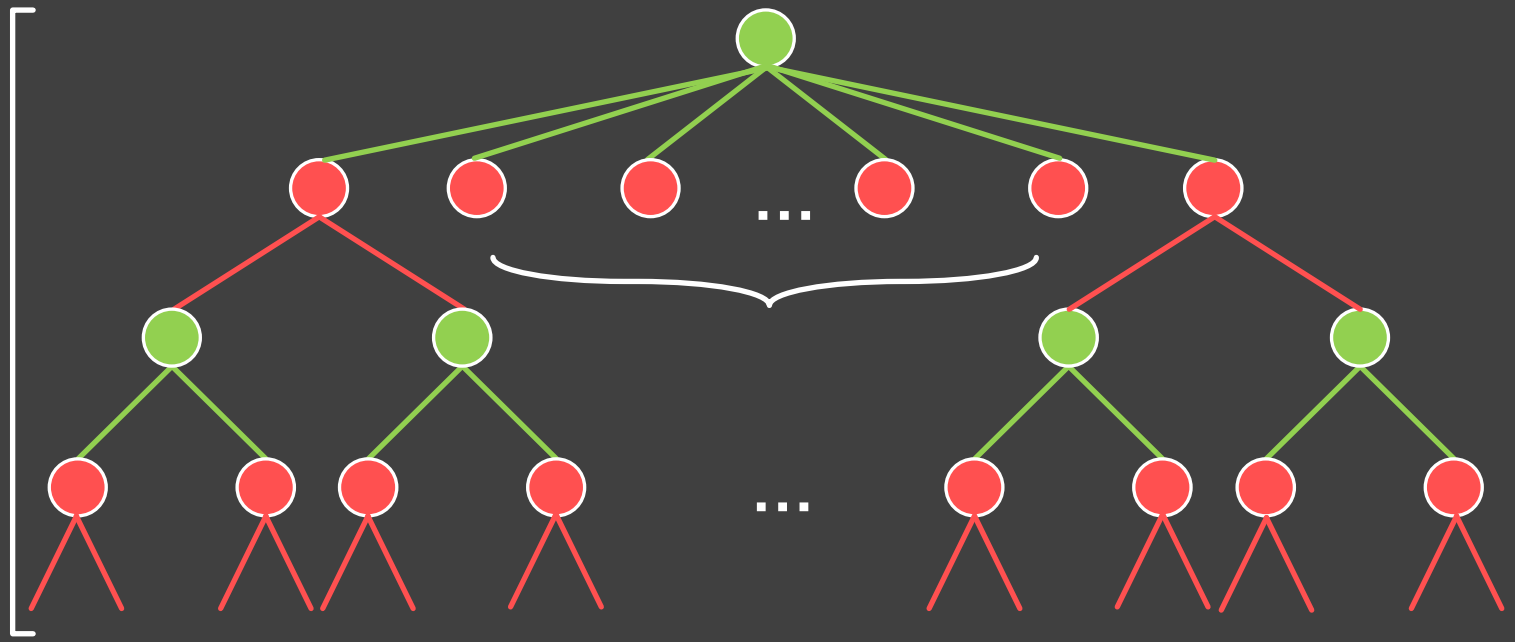


i.e. 井字棋





查询次数：A_9^4



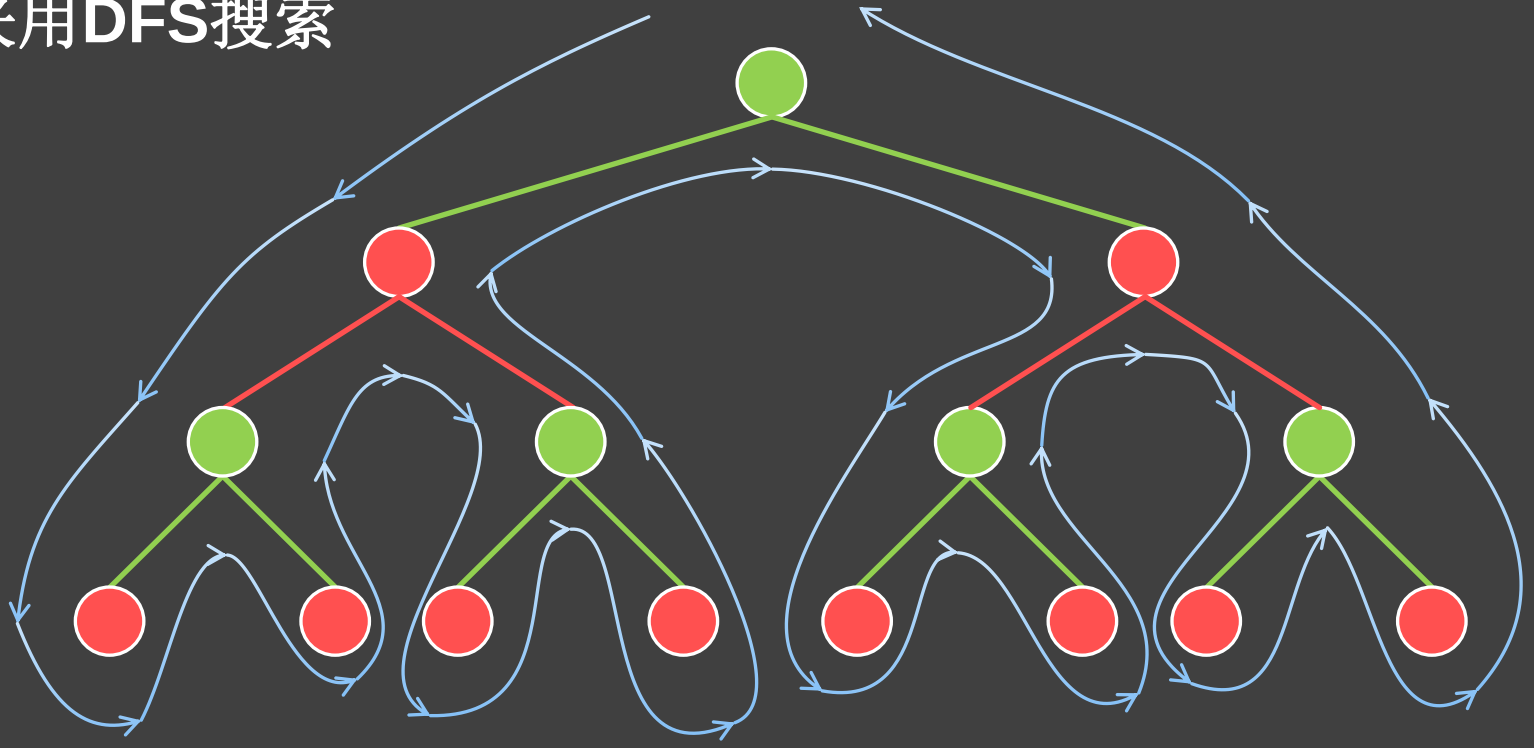
查询次数：A_n^m

m：给定搜索深度

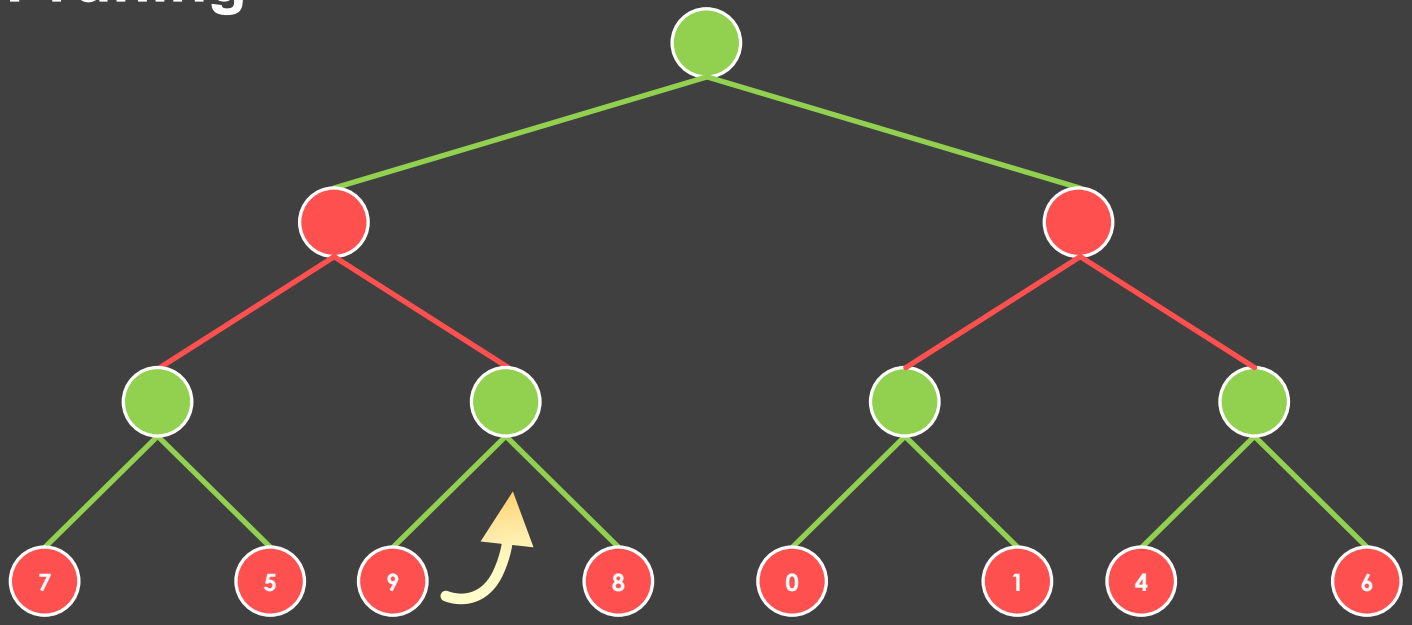
n：初始解空间规模

运算成本巨大

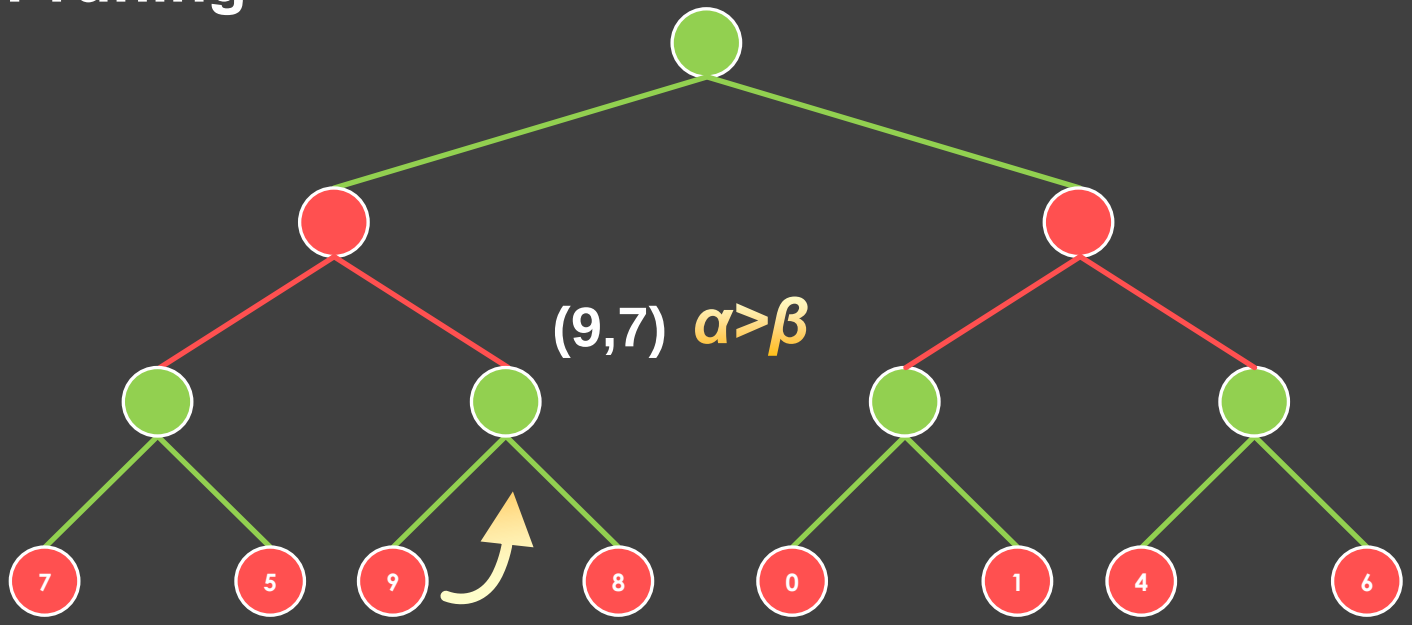
采用DFS搜索



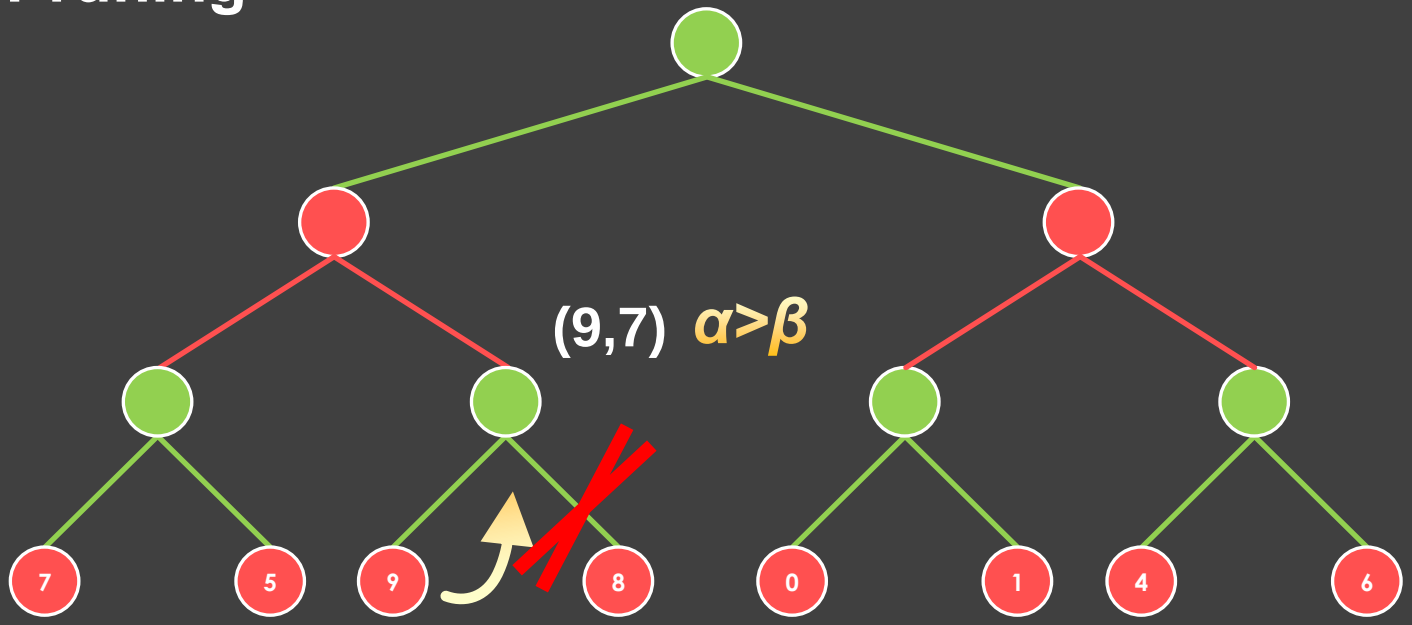
α-β Pruning



α-β Pruning

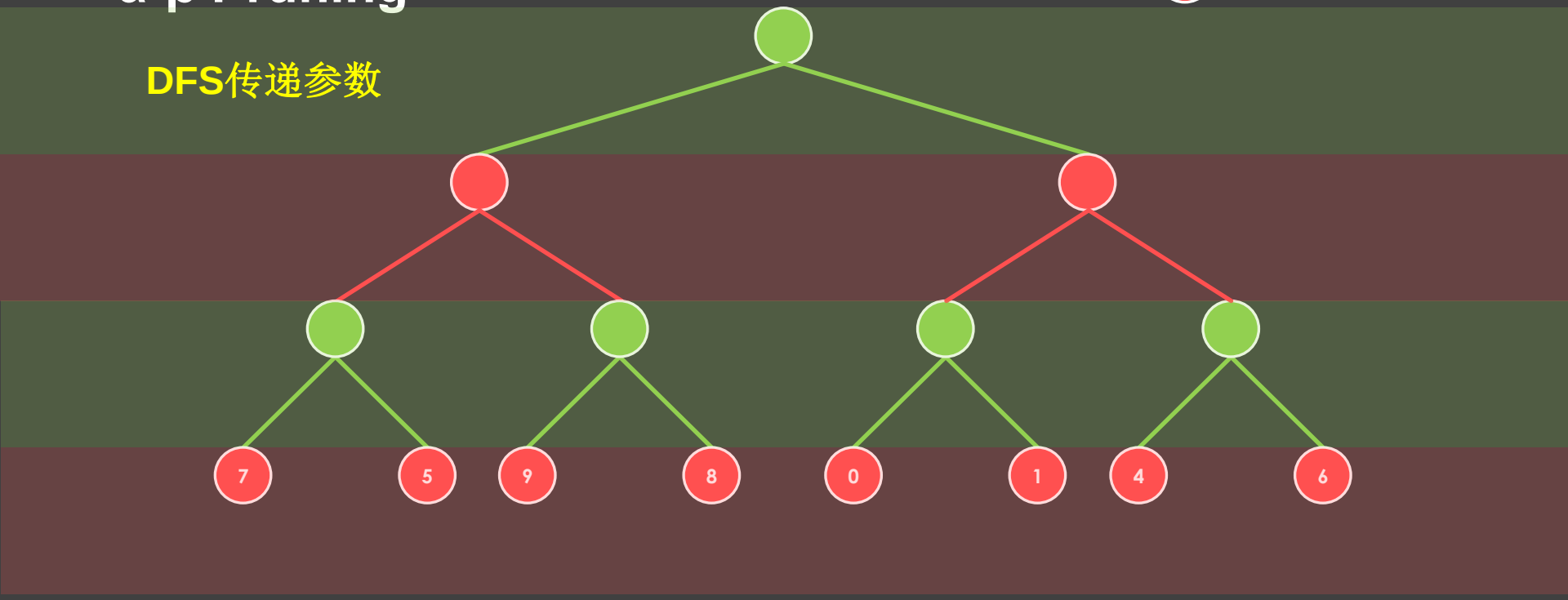


α-β Pruning

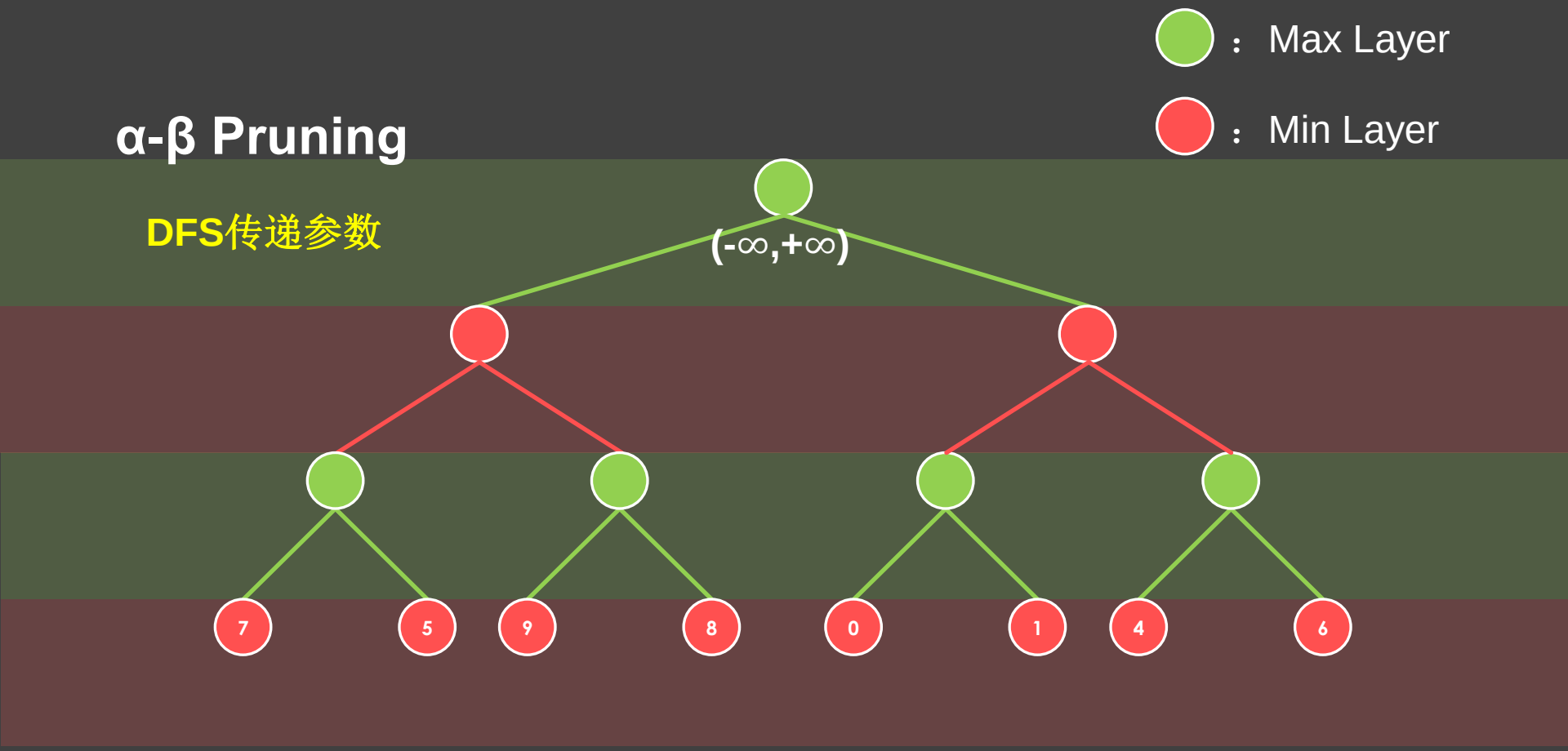


α-β Pruning

DFS传递参数

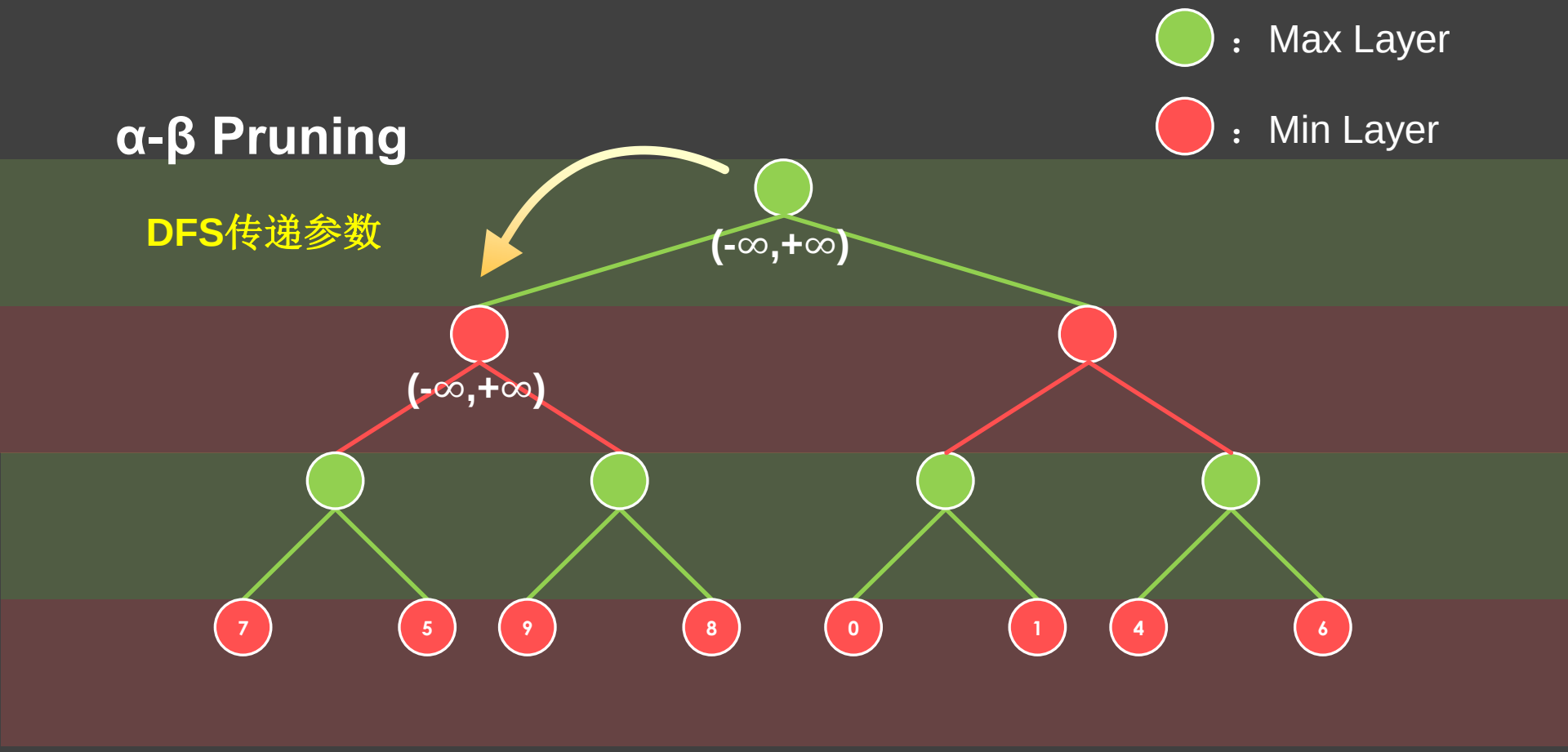


α-β Pruning

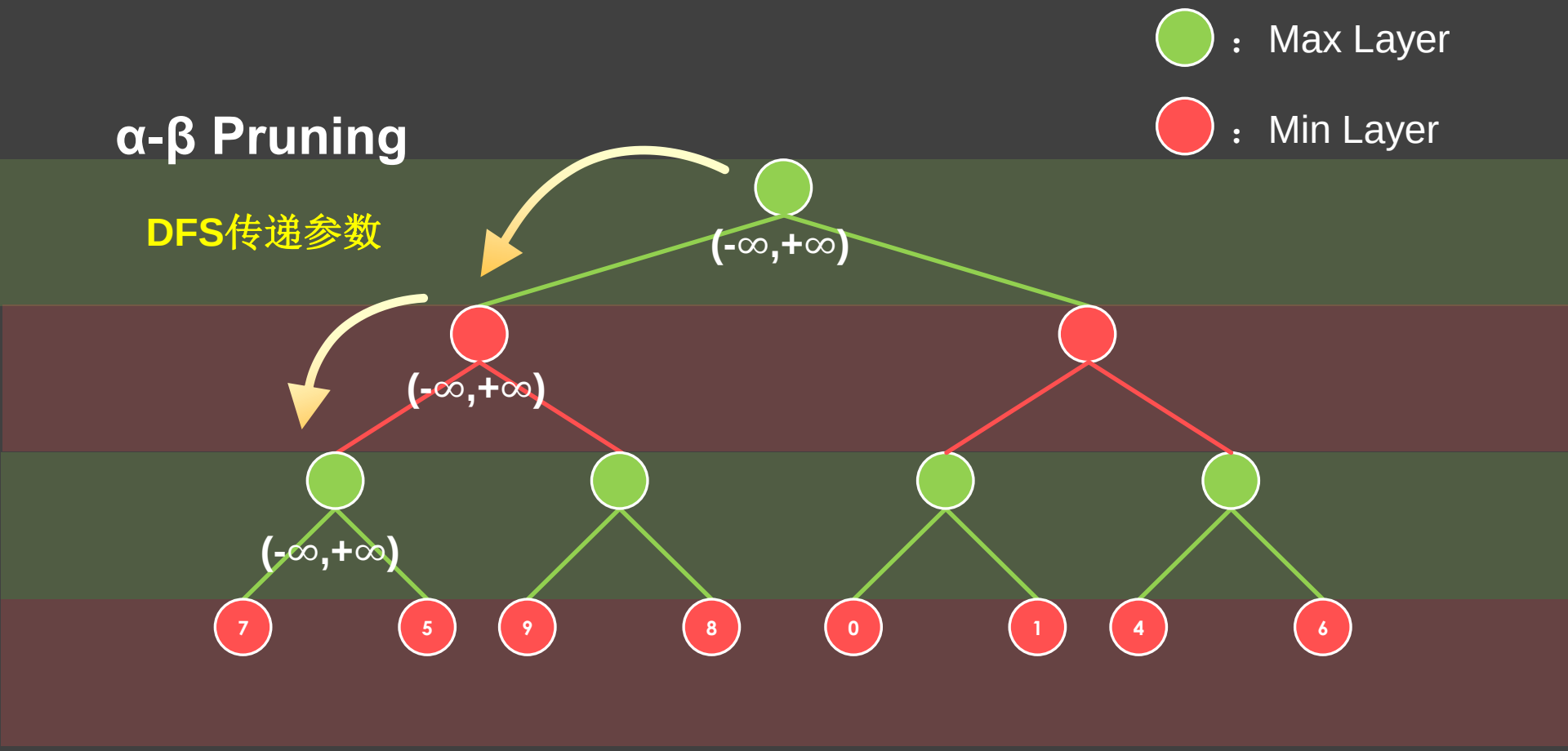


α-β Pruning

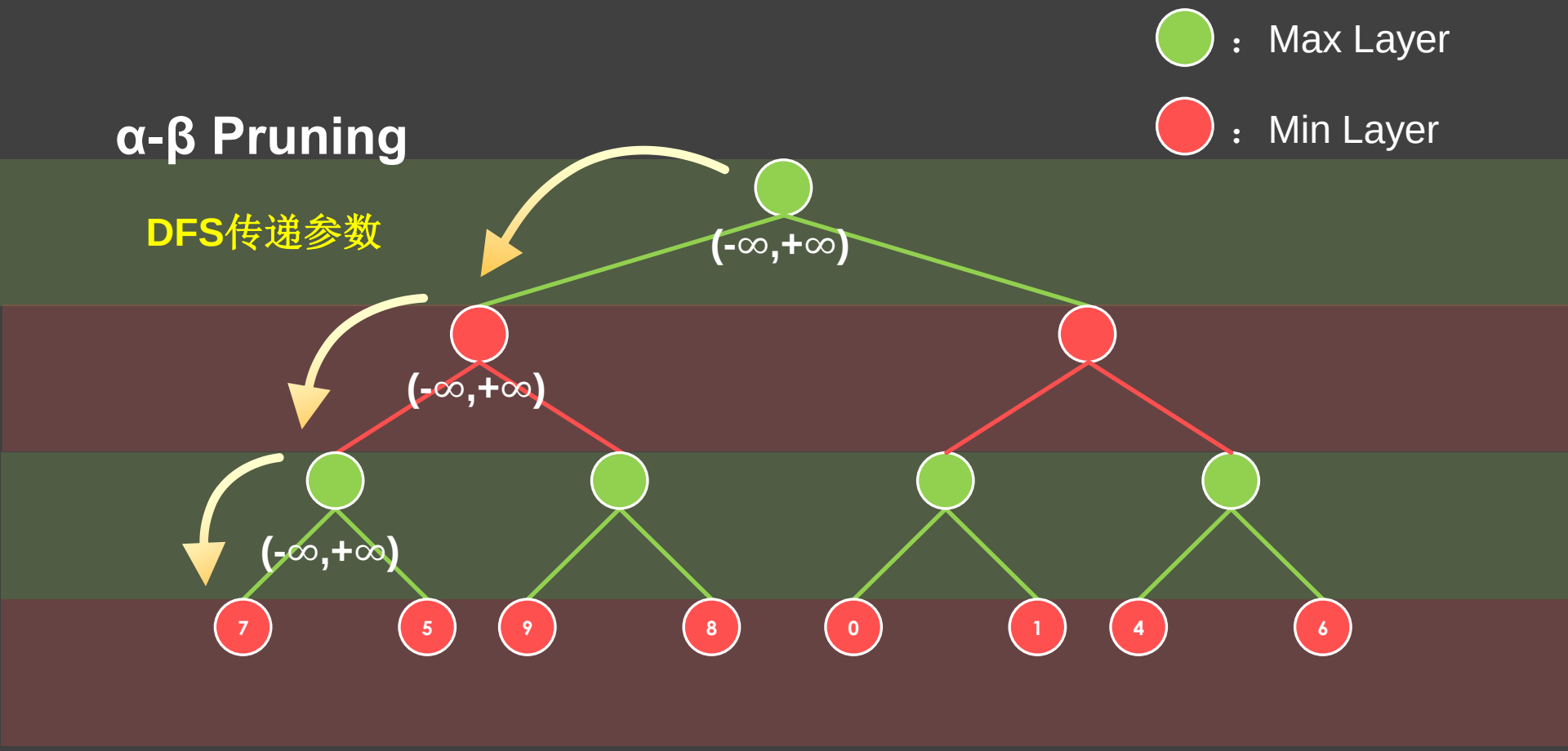
DFS传递参数



α-β Pruning



α-β Pruning

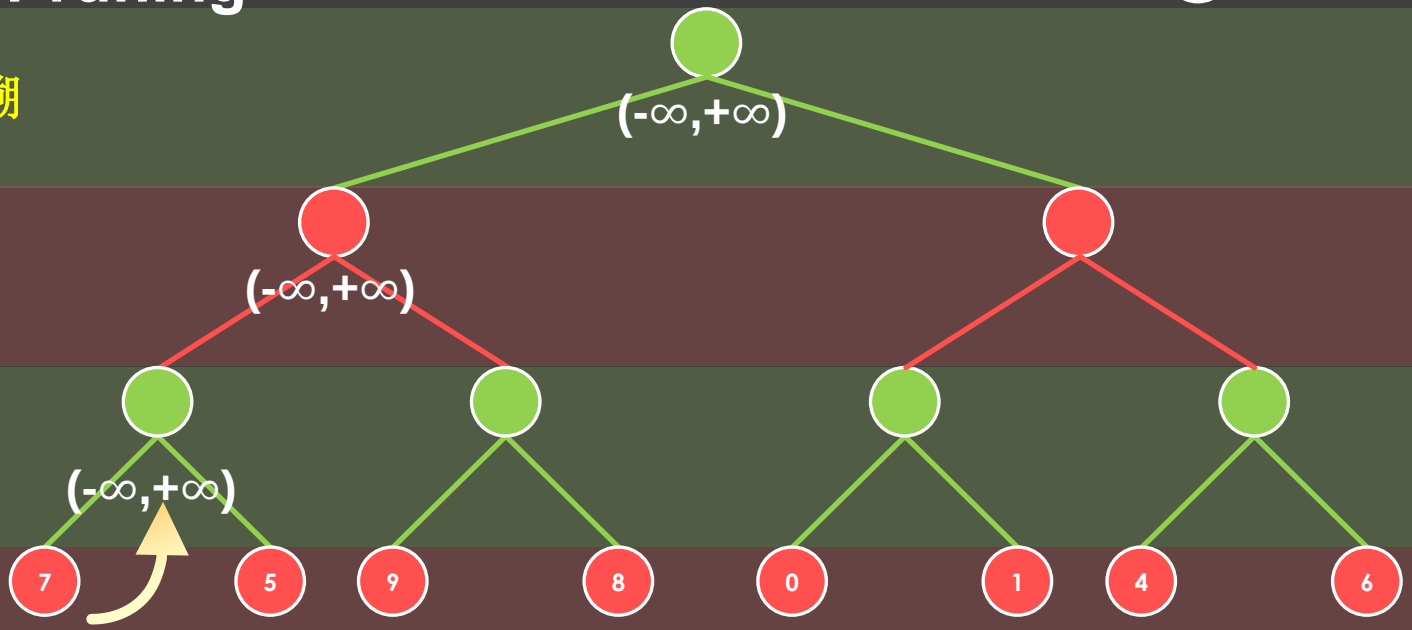


α-β Pruning

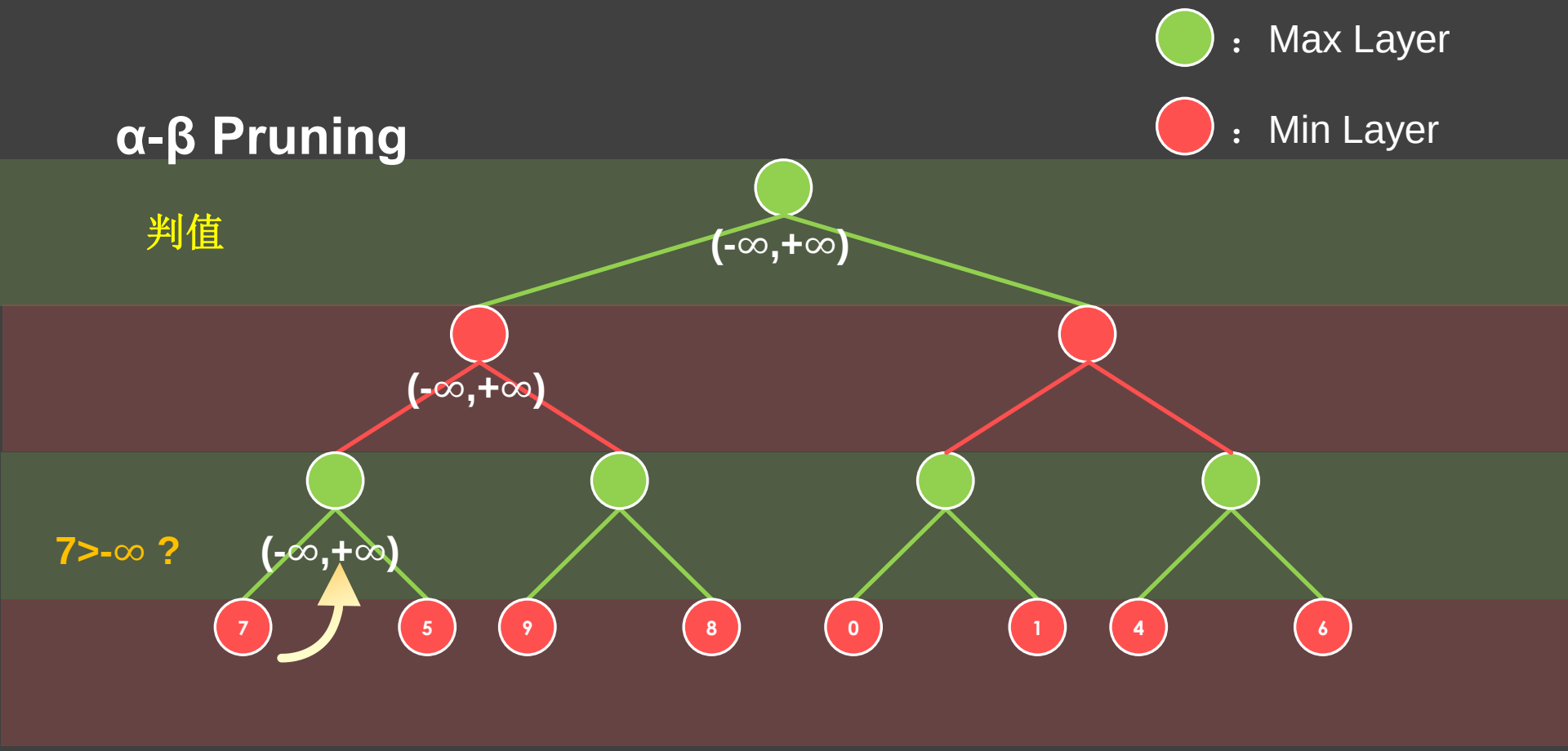
回溯

：Max Layer

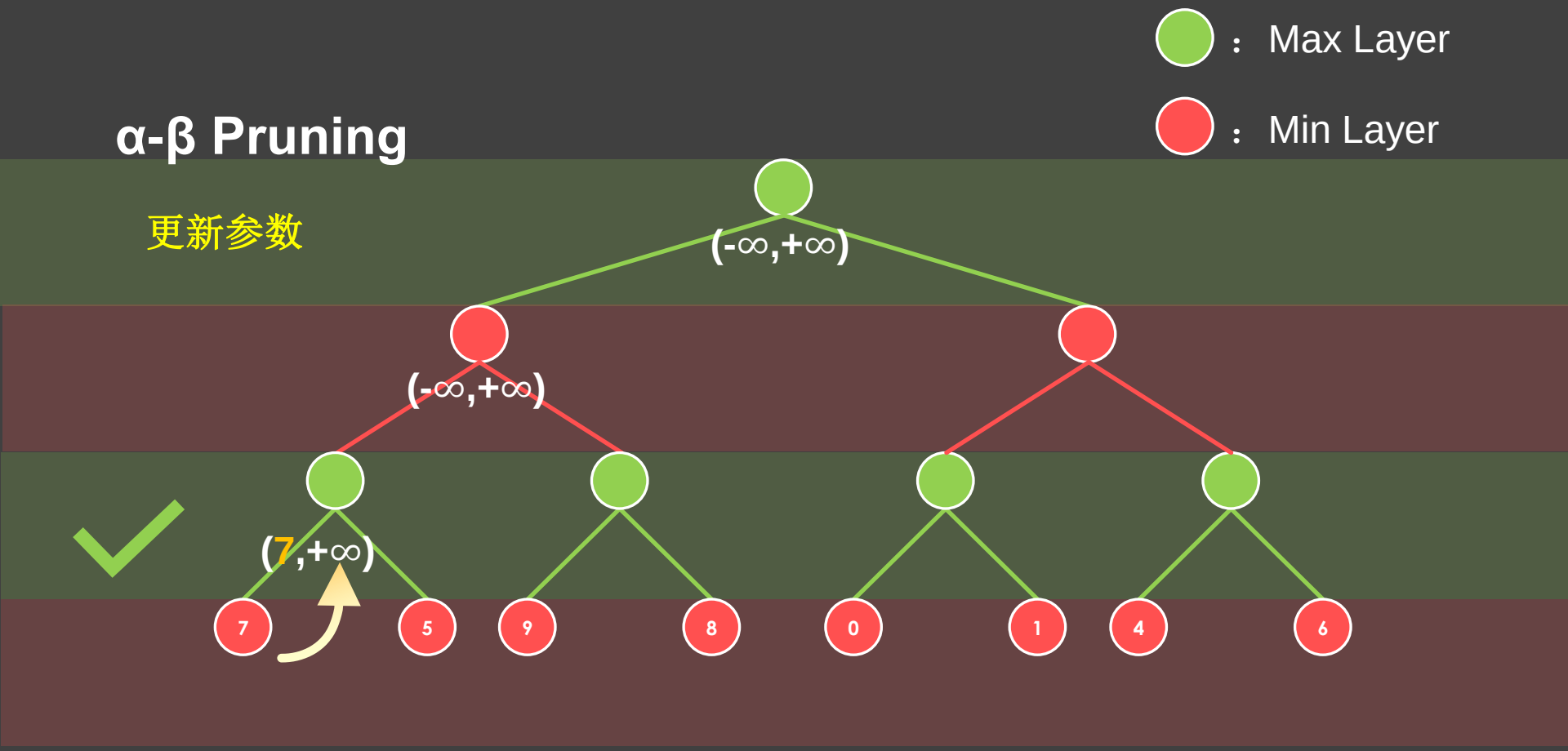
：Min Layer



α-β Pruning

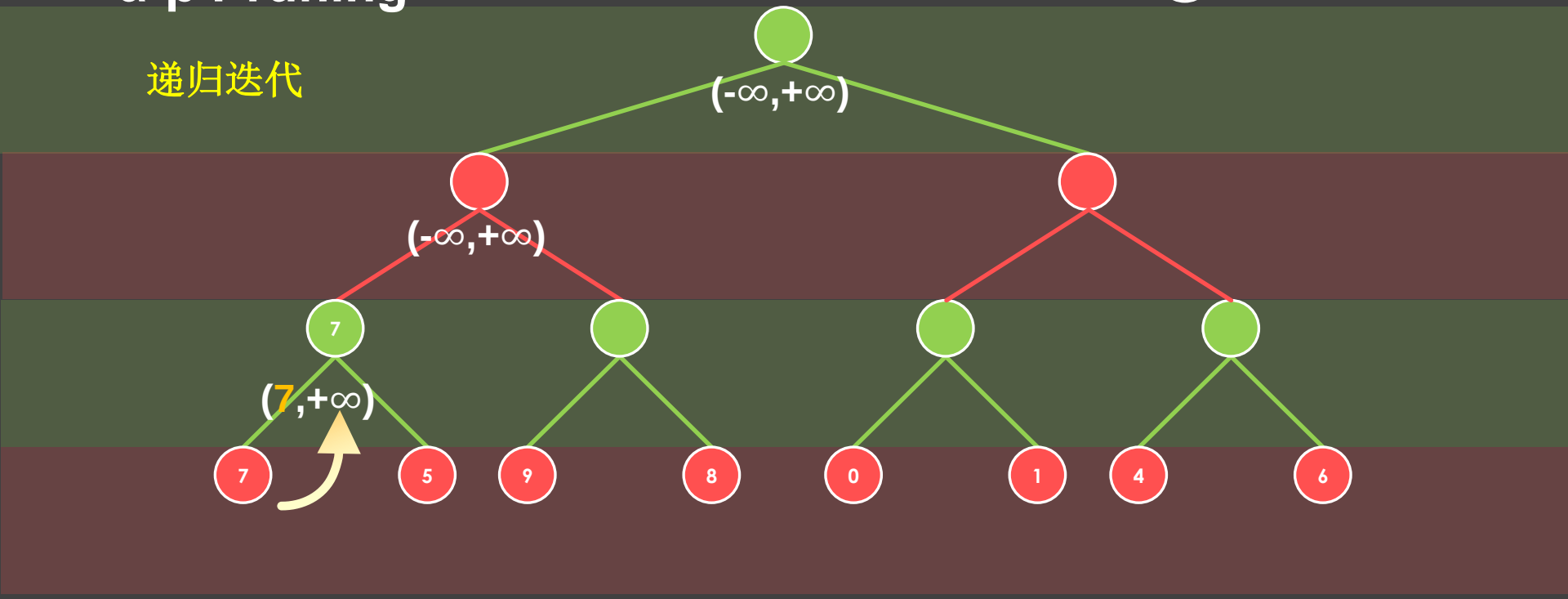


α-β Pruning



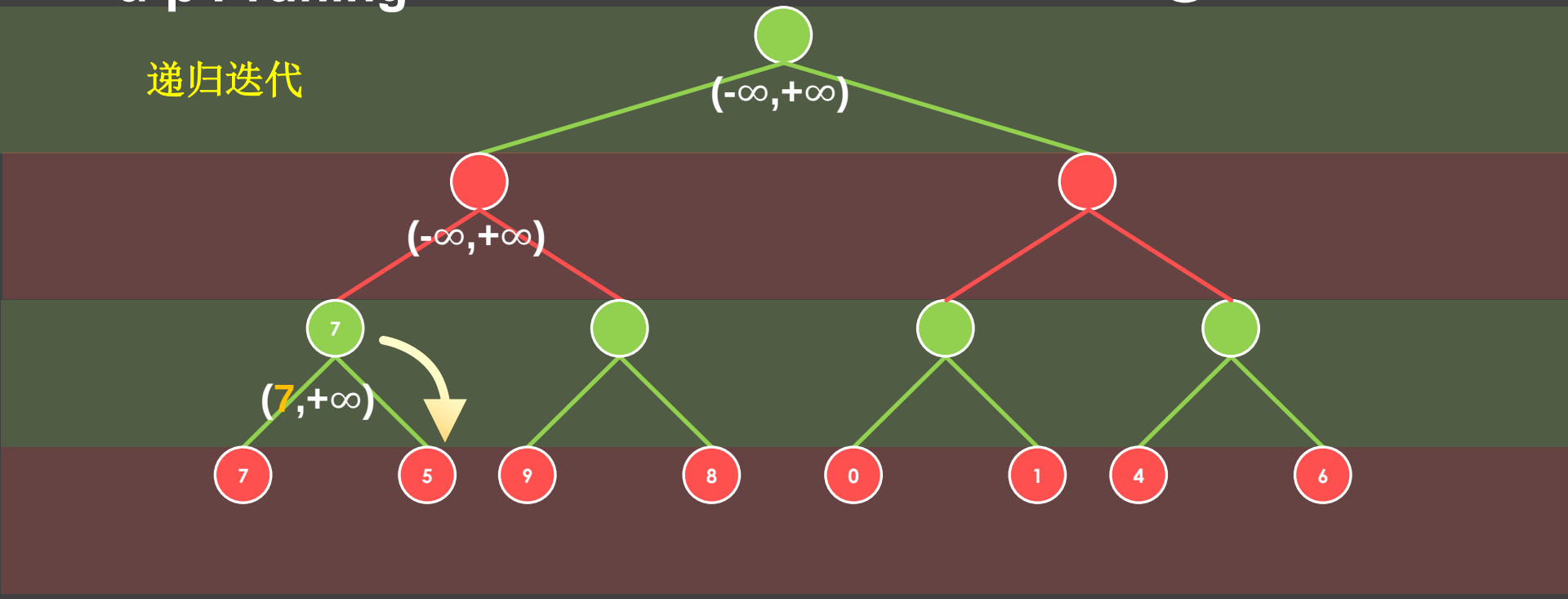
α-β Pruning

递归迭代



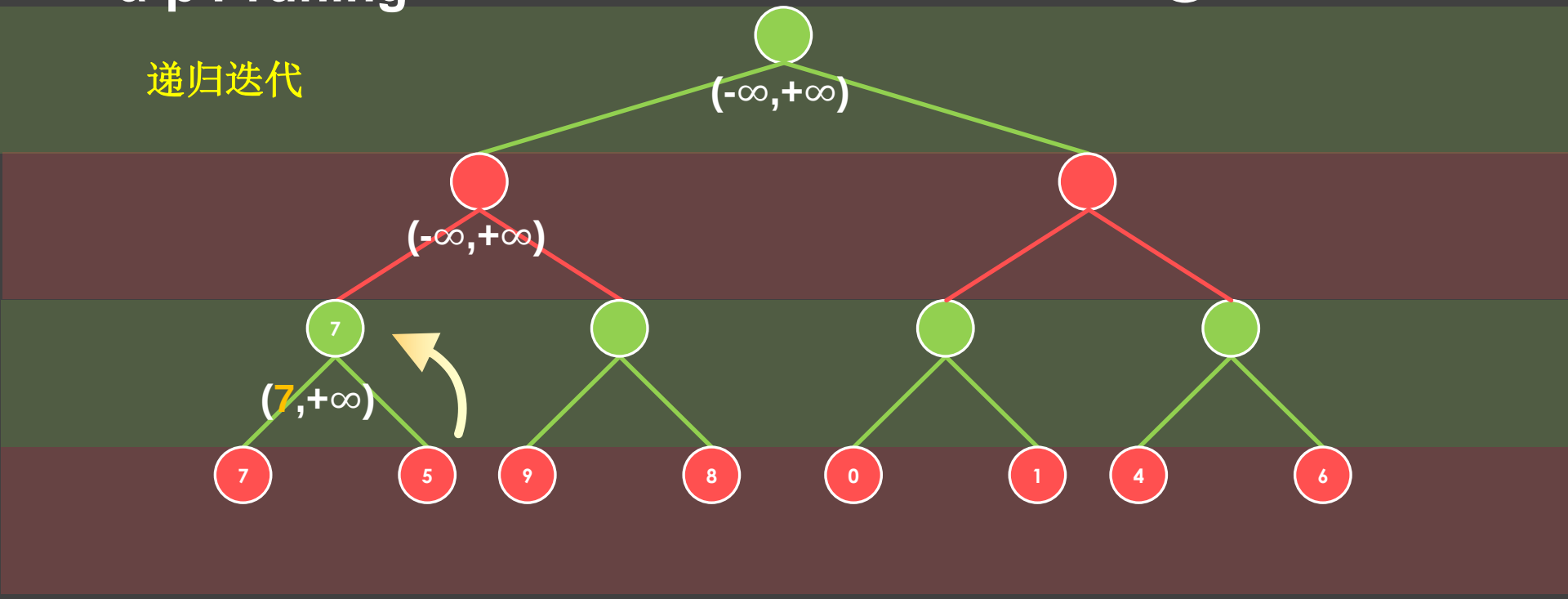
α-β Pruning

递归迭代

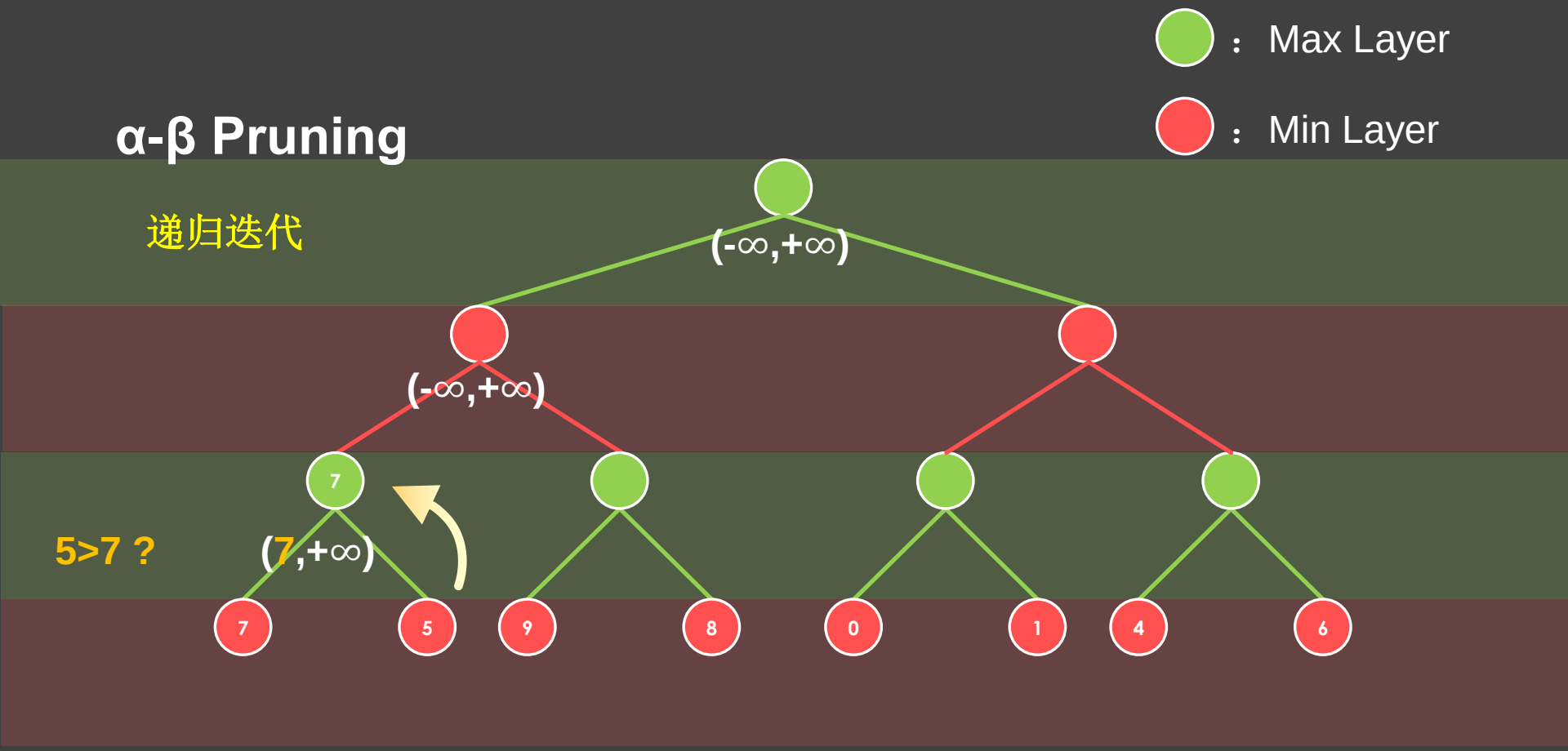


α-β Pruning

递归迭代

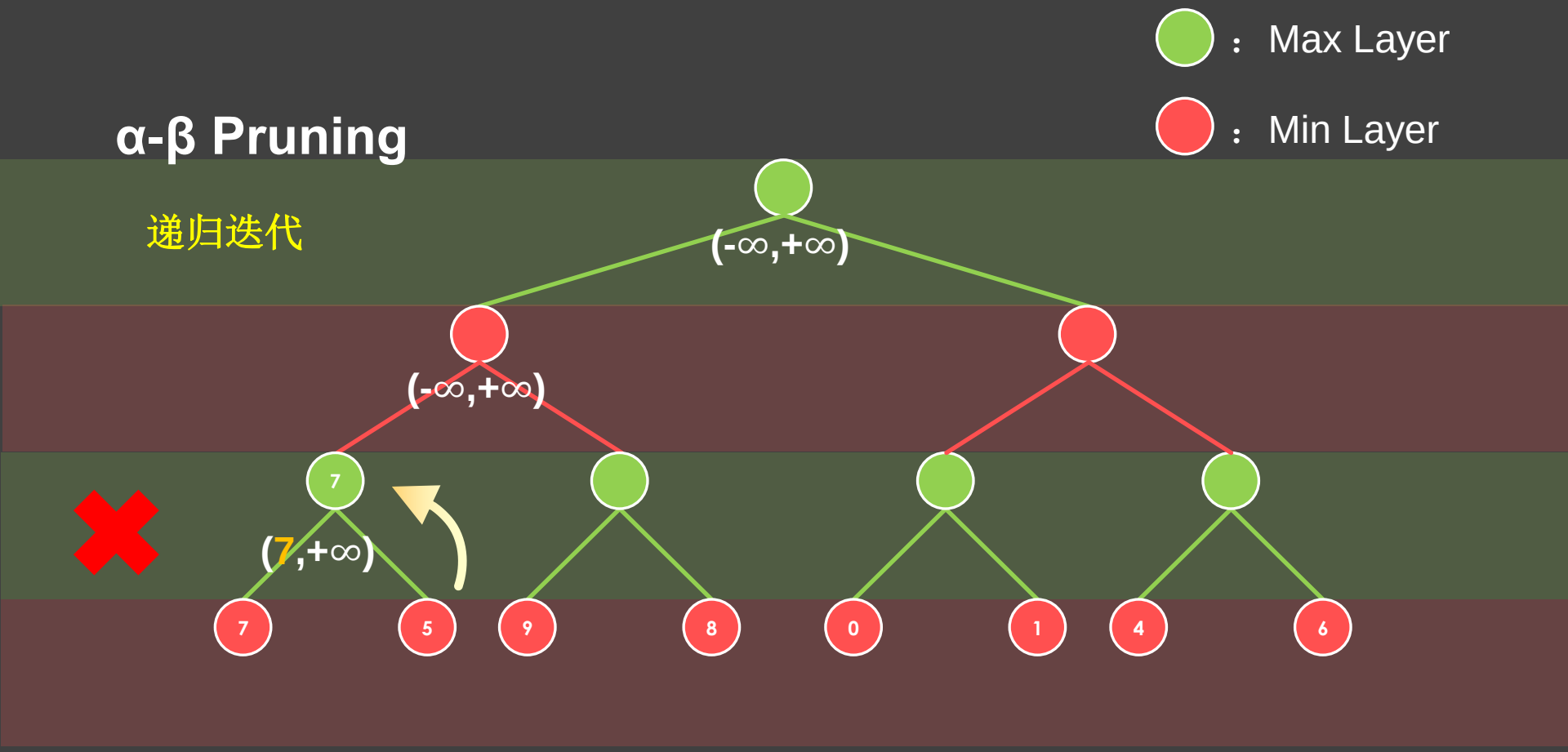


α-β Pruning



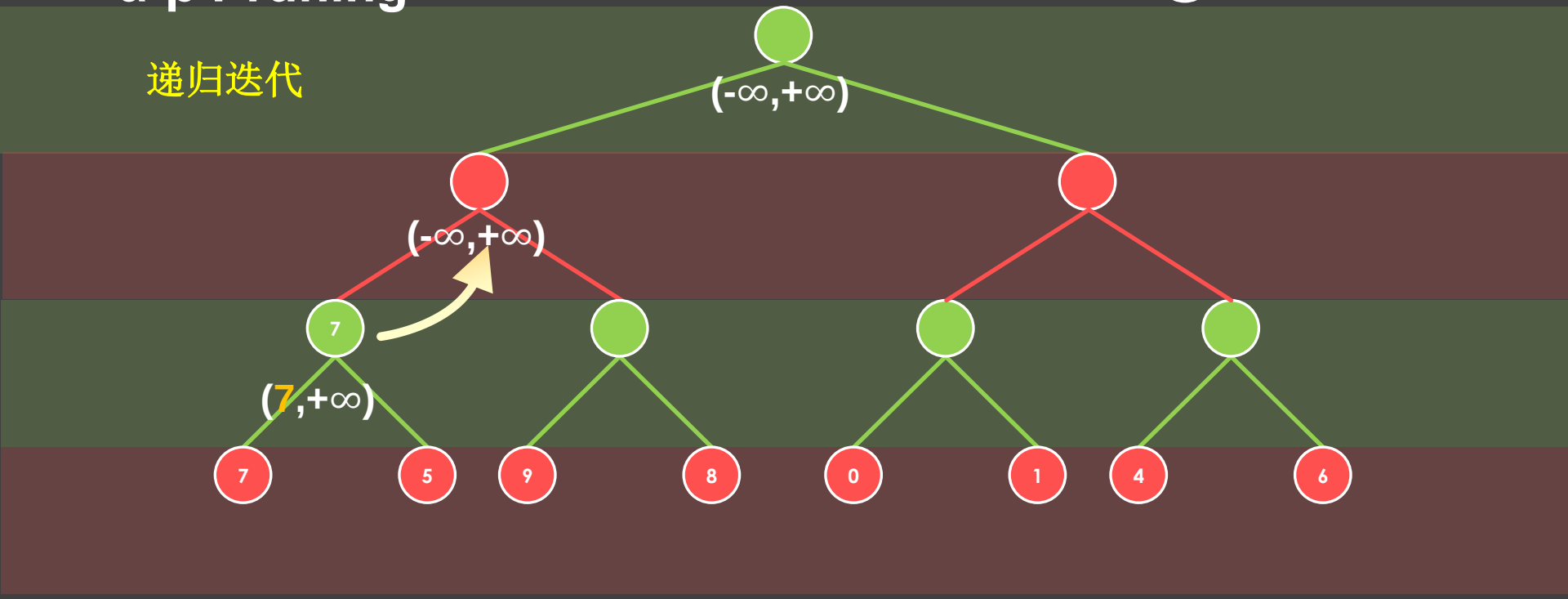
α-β Pruning

递归迭代

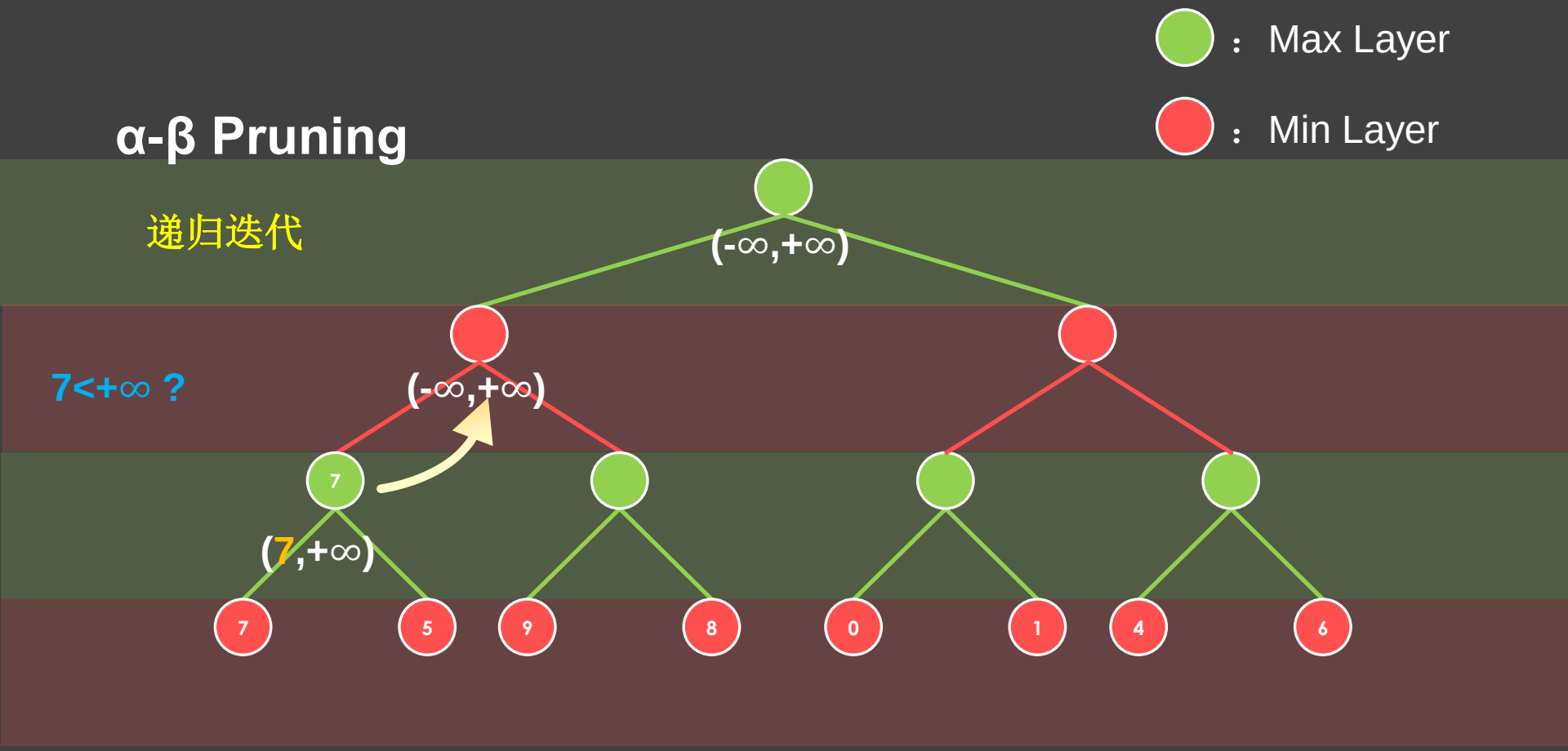


α-β Pruning

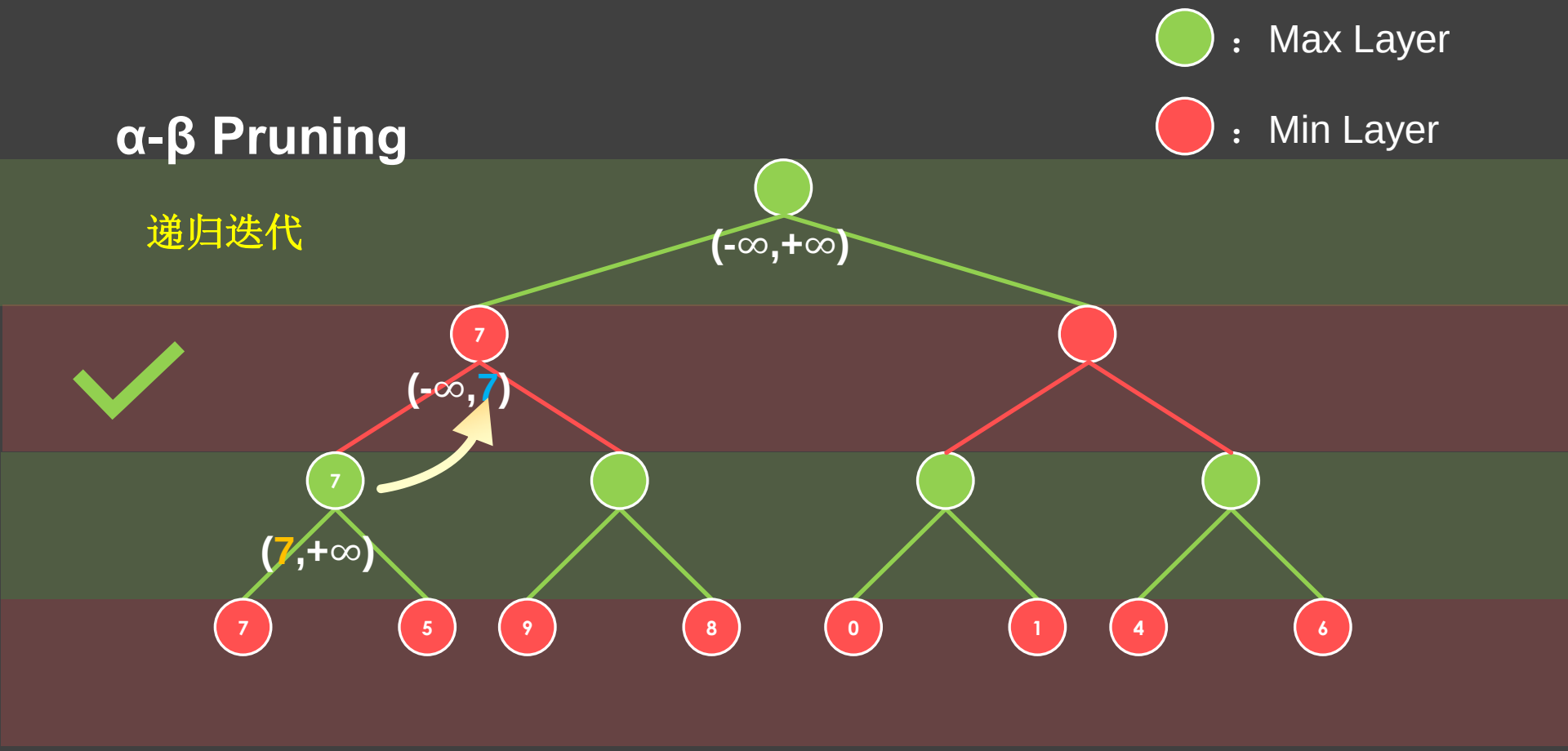
递归迭代



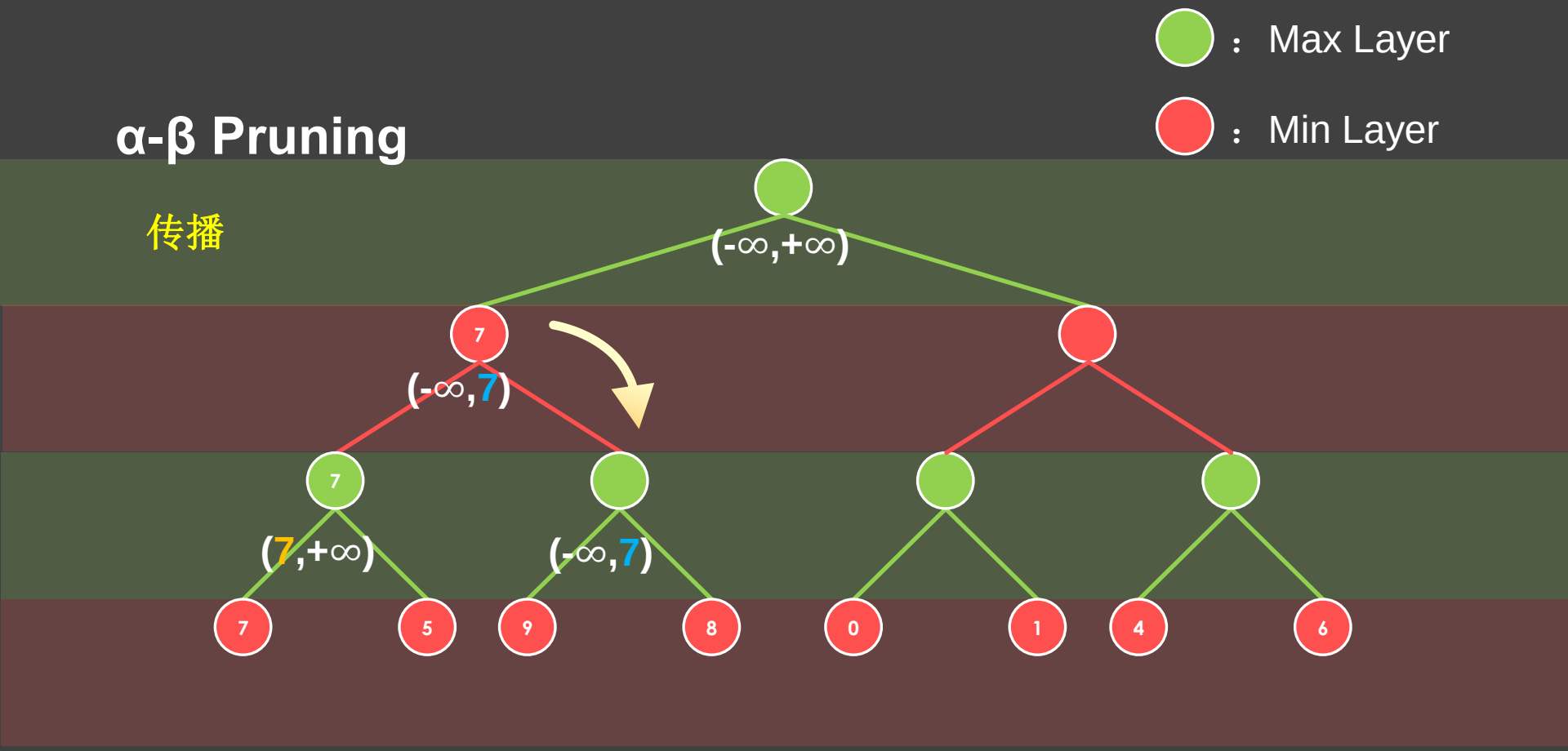
α-β Pruning



α-β Pruning

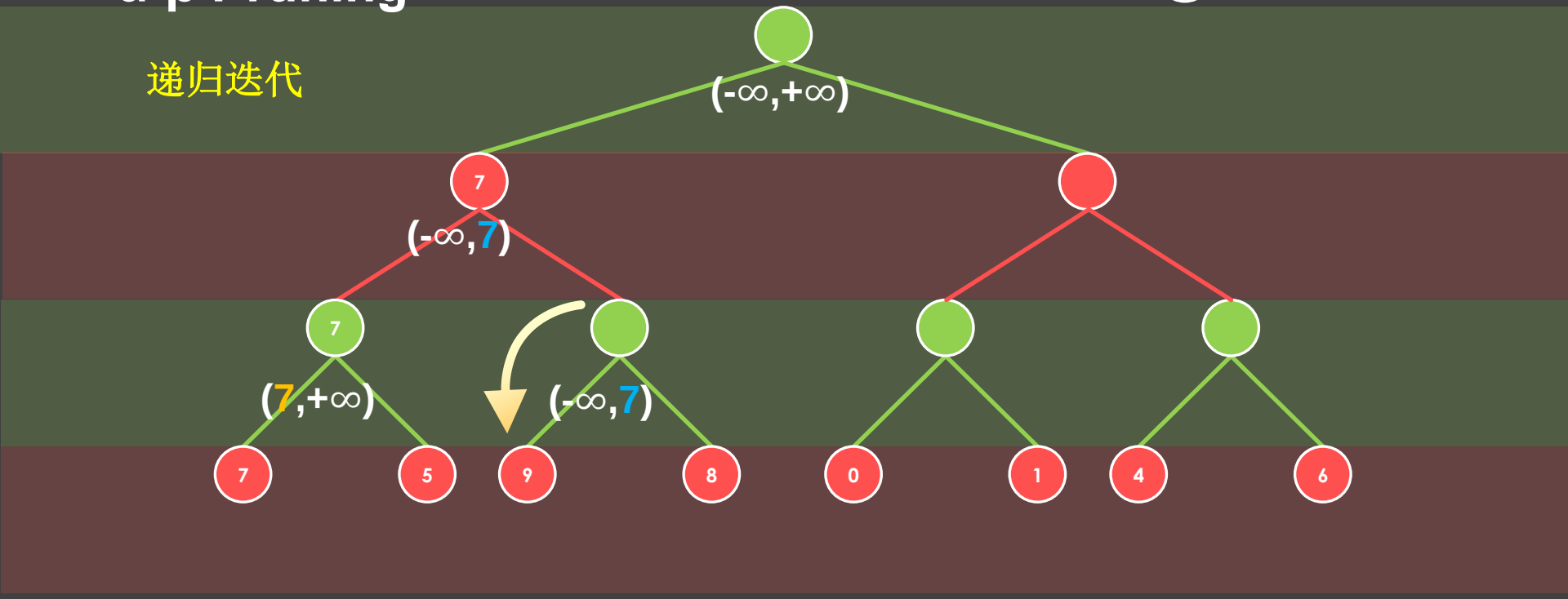


α-β Pruning

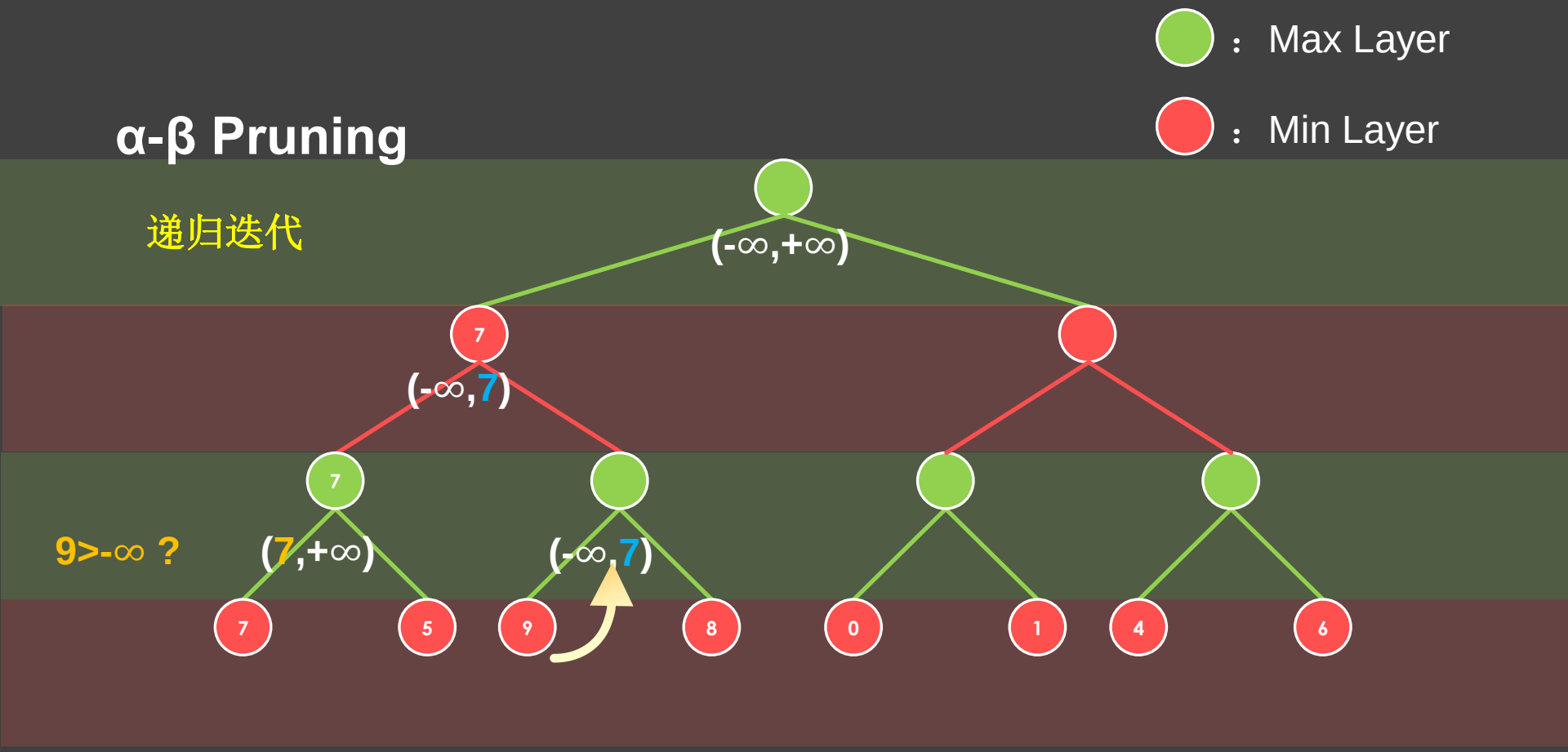


α-β Pruning

递归迭代



α-β Pruning

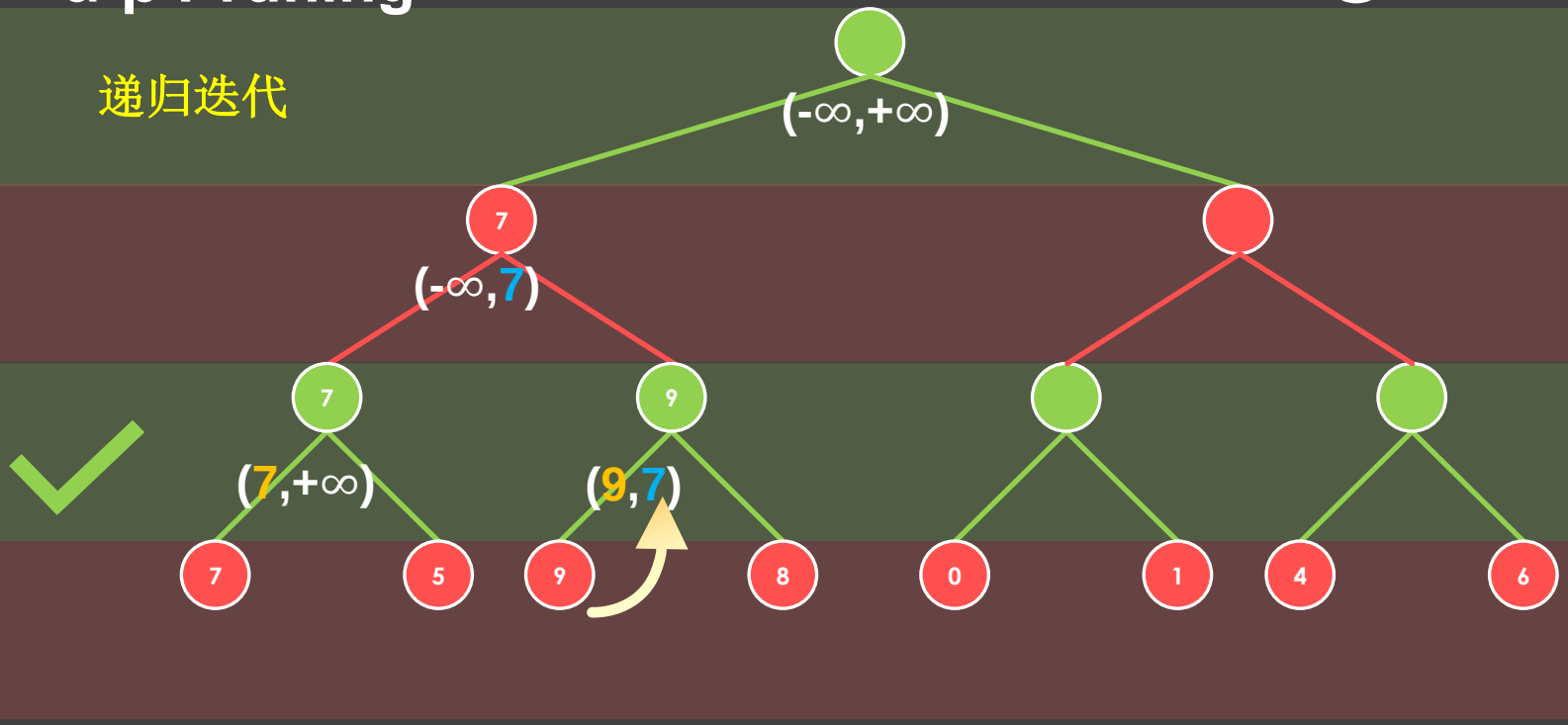


α-β Pruning

：Max Layer

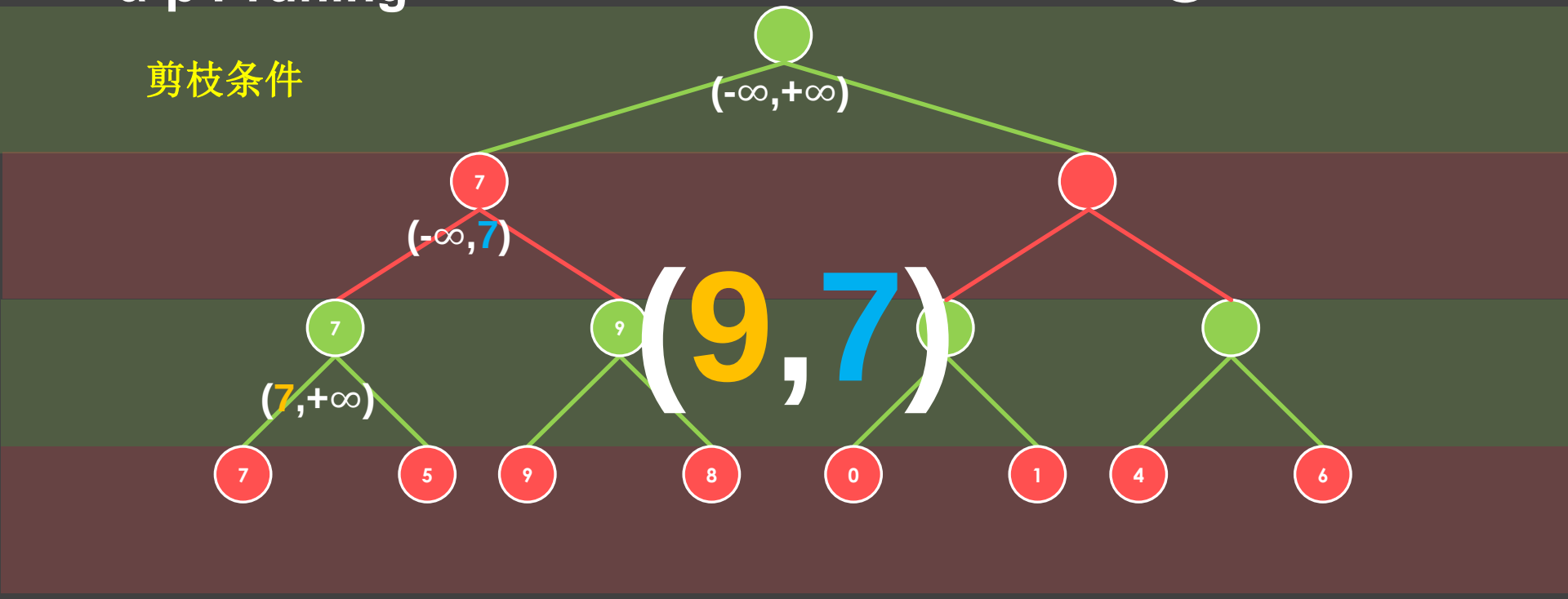
：Min Layer

递归迭代



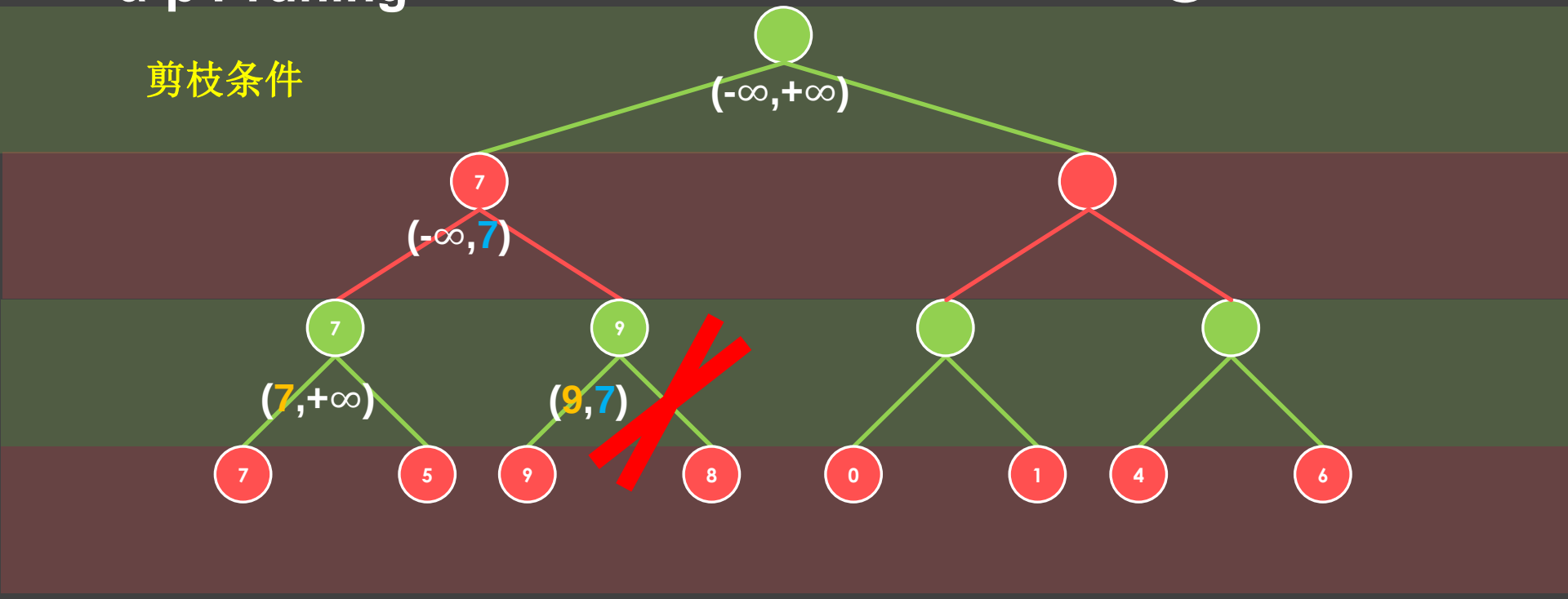
α-β Pruning

剪枝条件



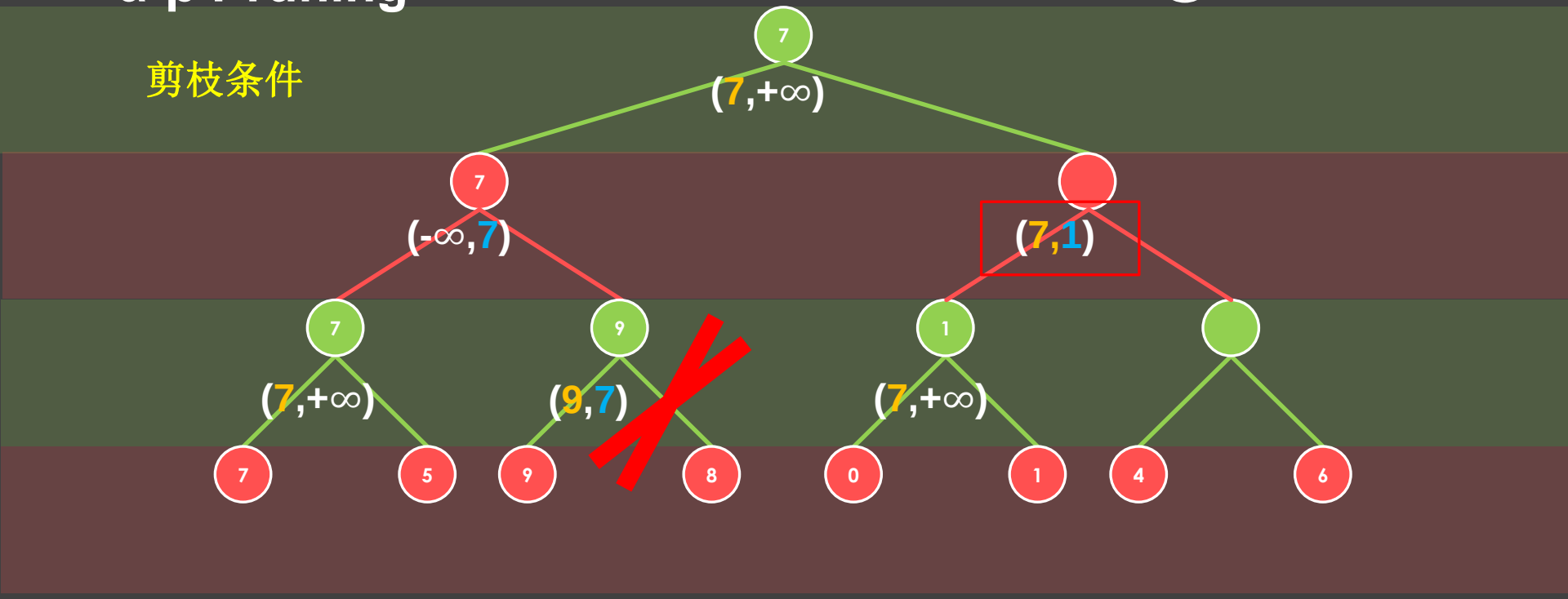
α-β Pruning

剪枝条件



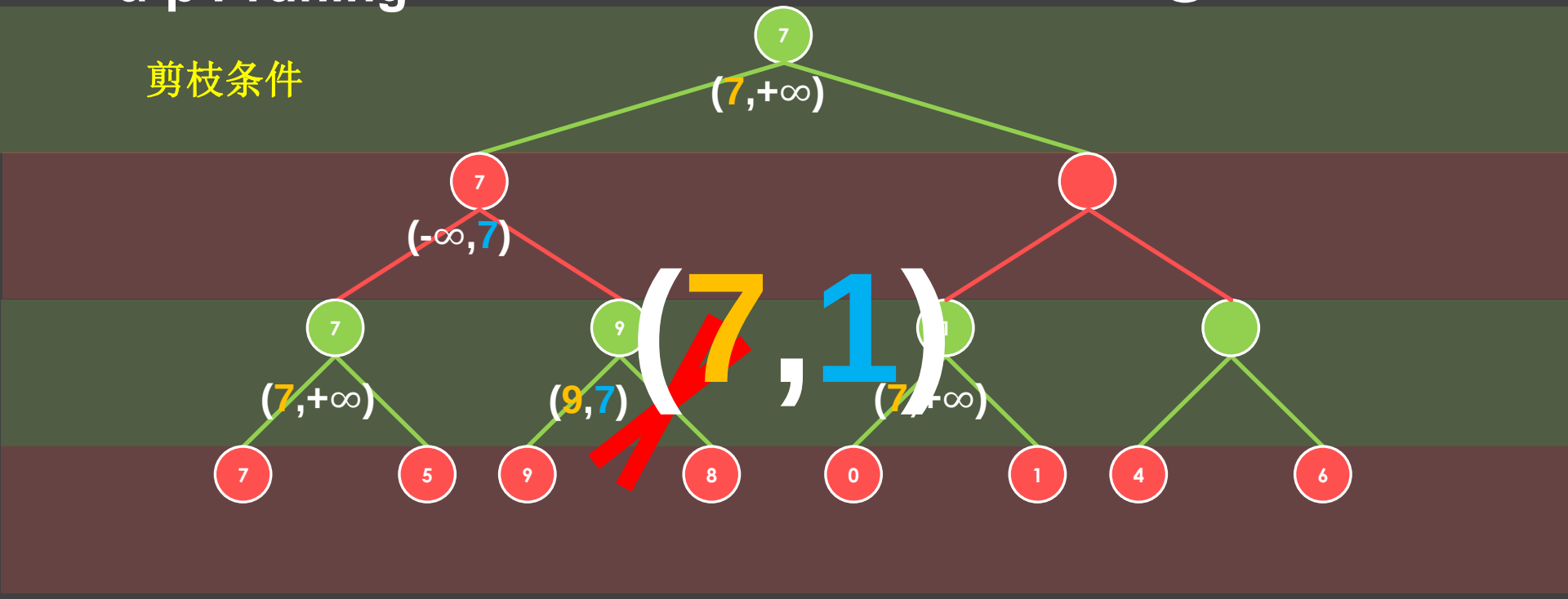
α-β Pruning

剪枝条件



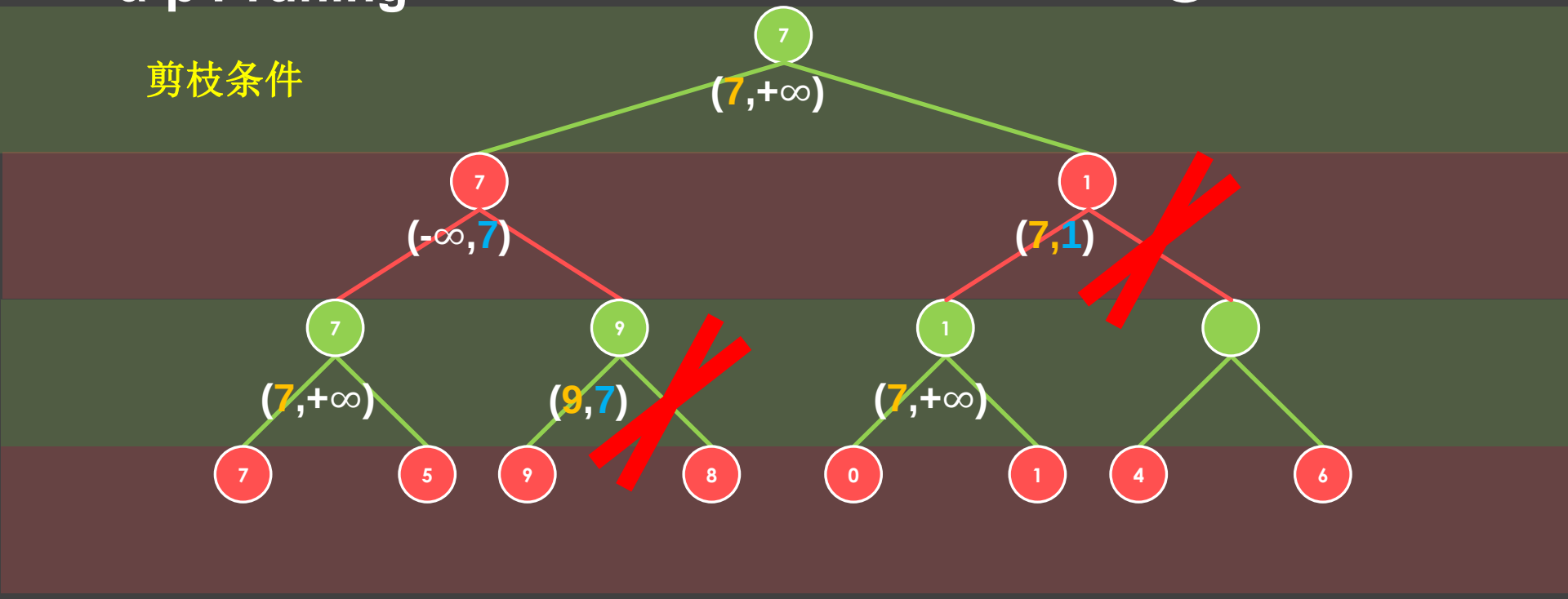
α-β Pruning

剪枝条件



α-β Pruning

剪枝条件

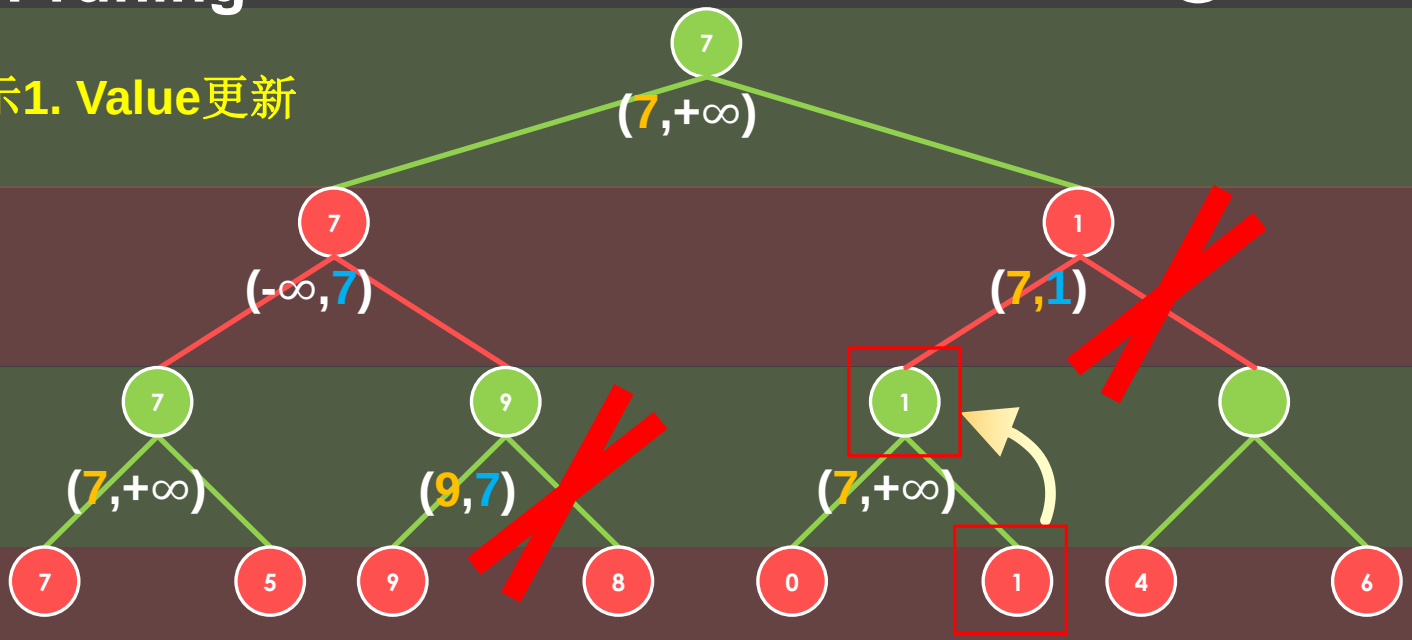


α-β Pruning

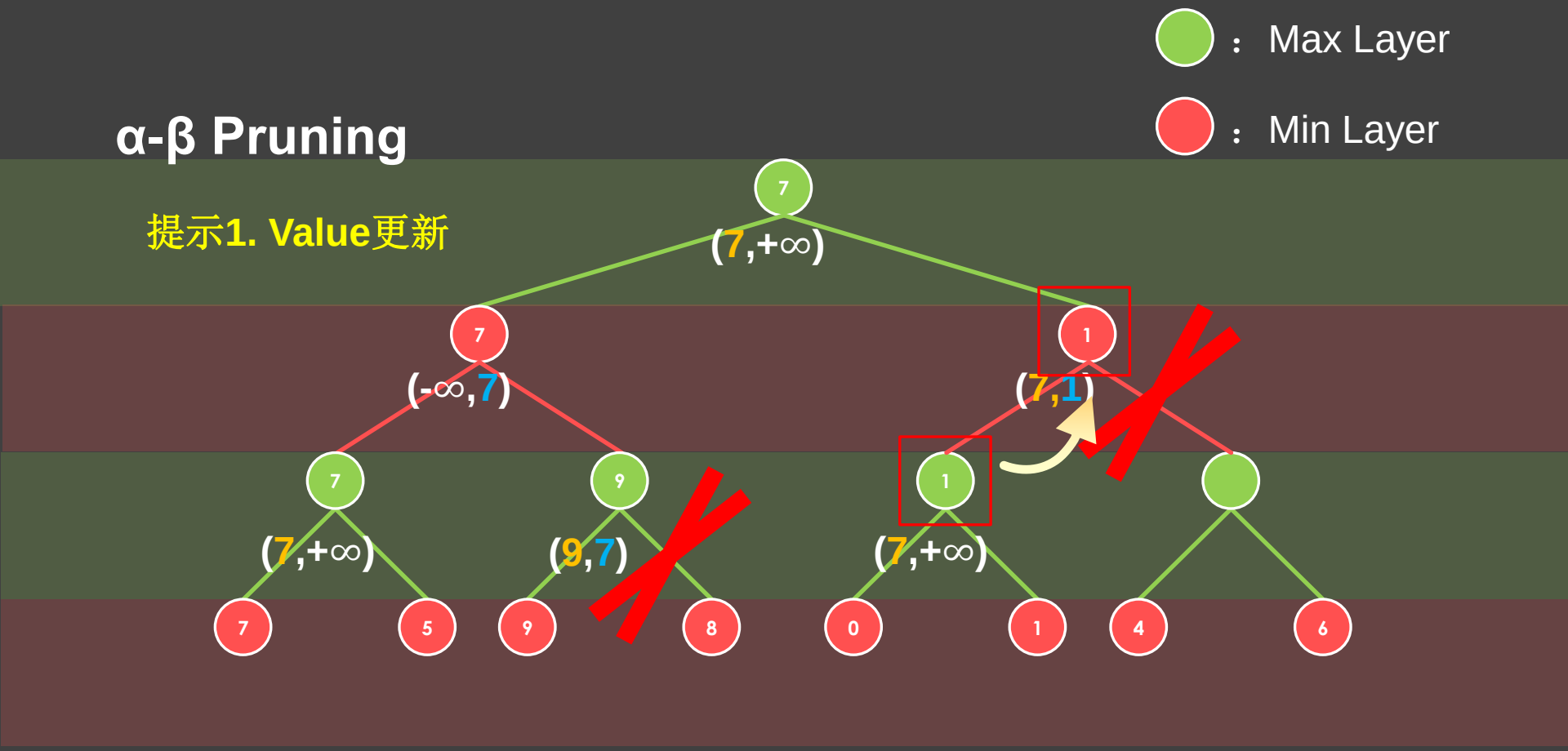
：Max Layer

：Min Layer

提示1. Value更新

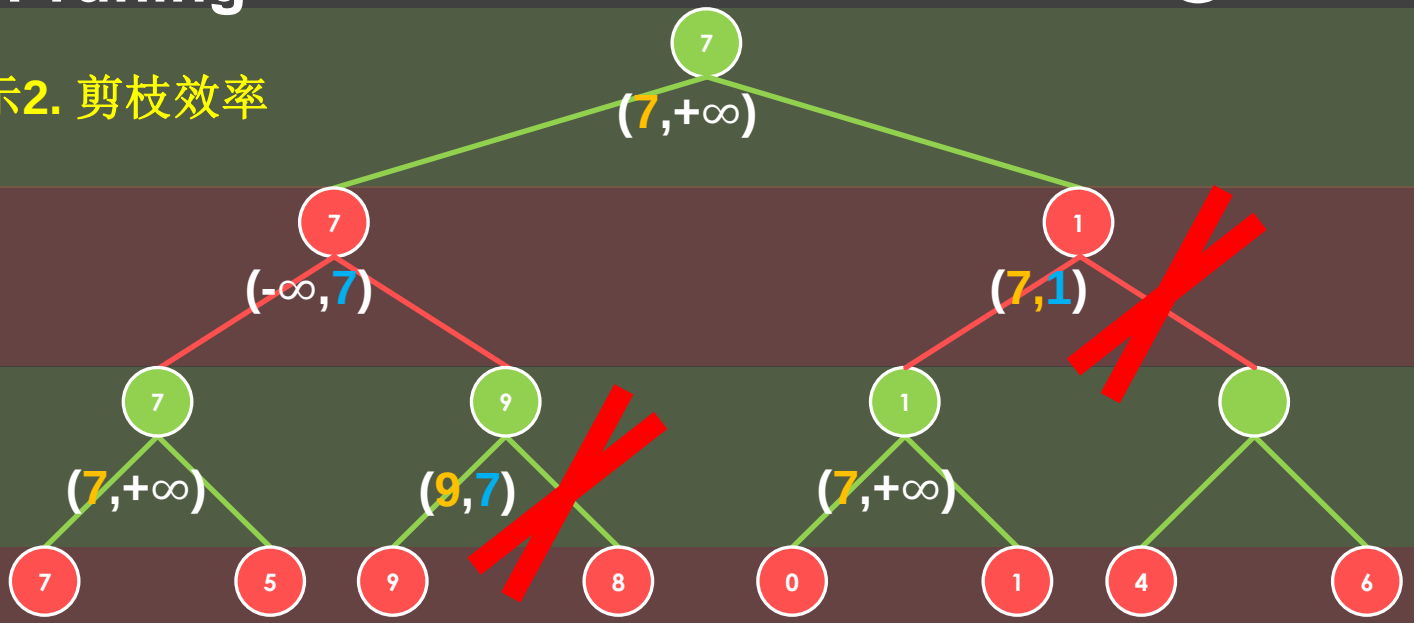


α-β Pruning



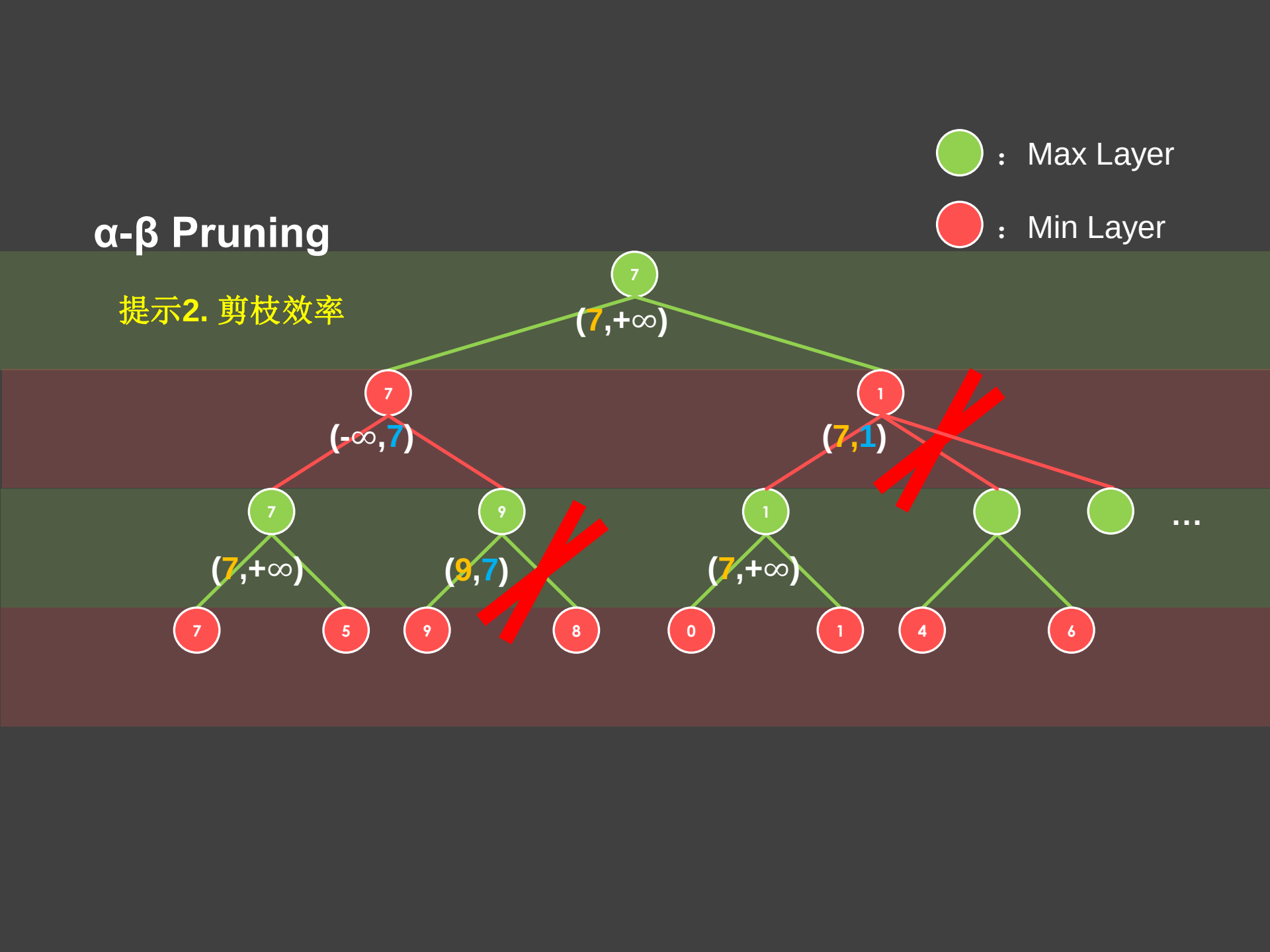
α-β Pruning

提示2. 剪枝效率



α-β Pruning

提示2. 剪枝效率



第5讲 博弈问题求解

5.4 博弈中的优化决策： α-β剪枝

- Alpha-Beta剪枝是最小最大方法的优化
- 两种方法产生的结果相同
- Alpha-Beta剪枝运行效率高

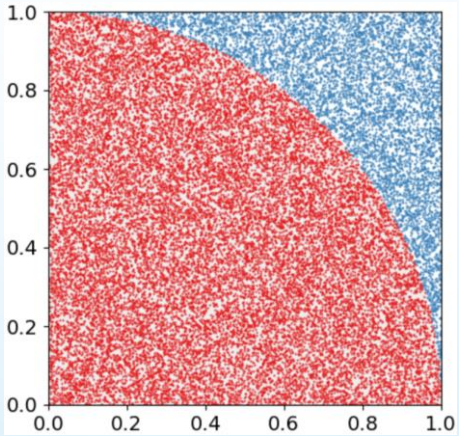
第5讲 博弈问题求解

5.4 博弈中的优化决策：蒙特卡罗树搜索

- 蒙特卡罗方法

  以概率统计理论为基础，通过重复随机采样来获得数值结果。

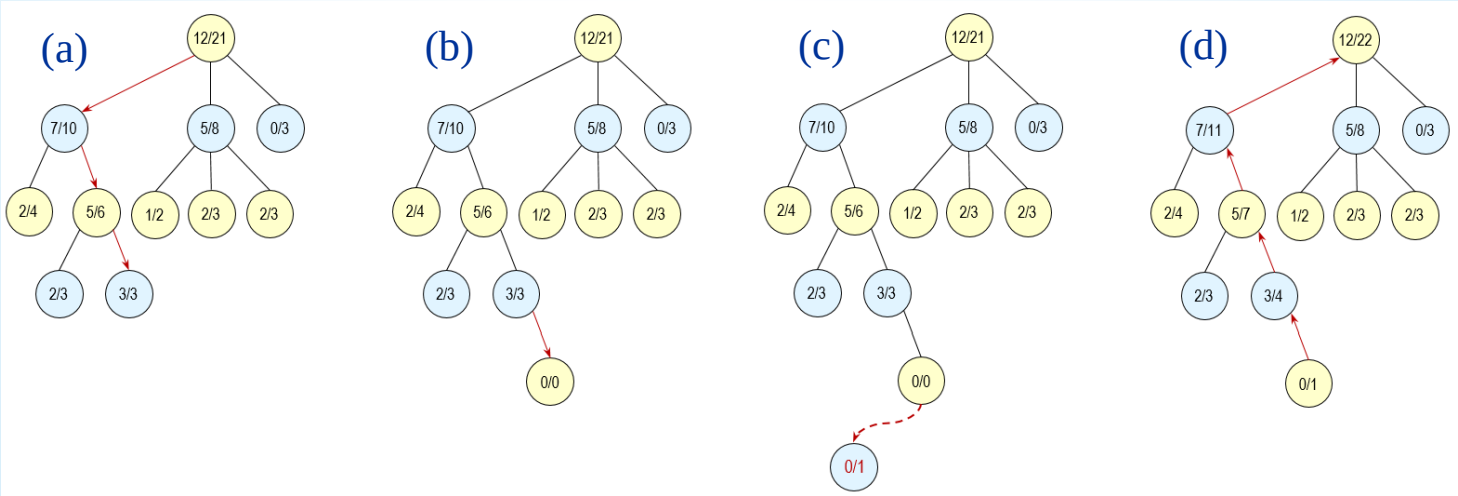
- 用蒙特卡罗方法求圆周率
  - 用蒙特卡罗方法模拟某一过程，需要产生各种概率分布的随机变量（如在正方形内散落小圆珠）
  - 再用统计方法把变量的数值特征估计出来，从而得到实际问题的数值解（如统计圆弧内和正方形内小圆珠的数量，并依此计算圆周率）



第5讲 博弈问题求解

5.4 博弈中的优化决策：蒙特卡罗树搜索

- 图(a) 选择（Selection）：从根节点出发，按照某种策略，选择一个给定节点的子节点。
- 图(b) 扩展（Expansion）：在搜索树中创建一个新的节点N_n作为N的一个新的子节点。
- 图(c) 模拟（Simulation）：进行蒙特卡罗模拟，直到得到一个结果，作为N_n的初始评分。
- 图(d) 反向传播（Backpropagation）：更新N_n的父节点N及反向传播路径上每个节点的状态。



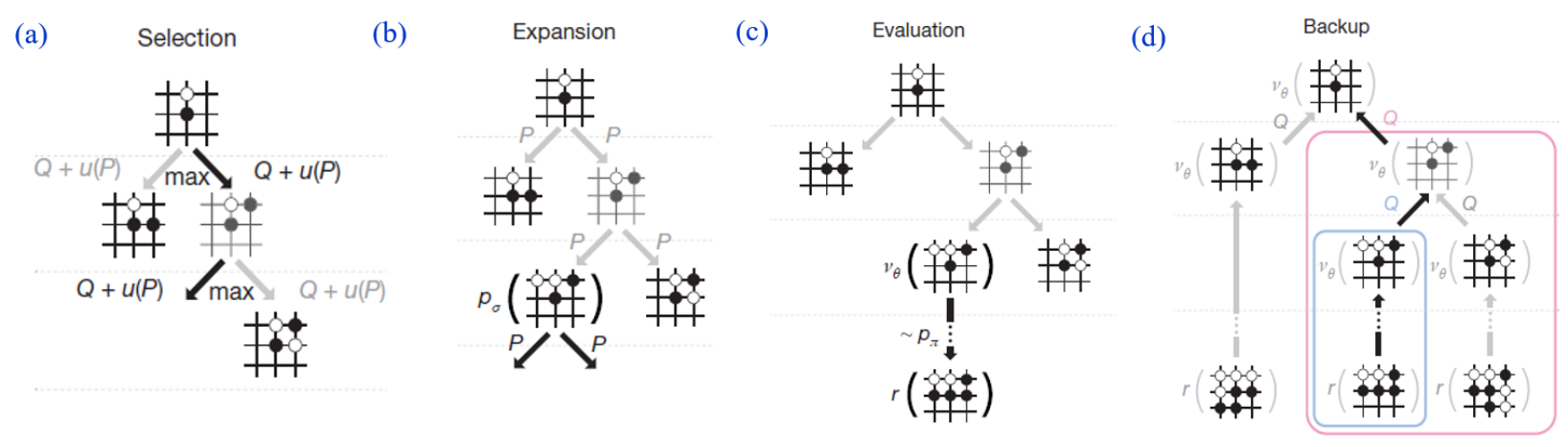
随机模拟+ 搜索树

第5讲 博弈问题求解

5.4 博弈中的优化决策：蒙特卡罗树搜索

AlphaGo中的蒙特卡罗树搜索： 对经典的蒙特卡罗树搜索进行了改进，

将第三步改为评估（Evaluation）、将第四步称为回溯（Backup）。



第 5 讲 博弈问题求解

5.1 博弈问题

5.2 博弈问题的类型

5.3 博弈问题的求解

5.4 博弈中的优化决策

5.5 博弈中的实时决策

第5讲 博弈问题求解

5.5 博弈中的实时决策：

- 博弈中的优化决策
  - 最小最大算法生成整个博弈搜索空间
  - α-β剪枝算法允许剪裁掉其中的一部分，但仍然需要搜索抵达终端状态的所有途径，至少是搜索空间中的一部分

    ——这样的搜索深度通常难以满足在合理时间内完成的要求！

- 博弈中的实时决策

  在博弈搜索中对状态应用启发式评估函数，有效地将非终端节点转换为终端叶节点——
  - 用启发式评估函数（EVAL）取代效用函数（UTILITY）
  - 用截断测试（CUTOFF-TEST）取代终止测试（TERMINAL-TEST）

第5讲 博弈问题求解

5.5 博弈中的实时决策：评估函数

- 评估函数EVAL：一个启发式函数，用于估计状态的期望效用值
- 评估函数的需求
  - 对终止状态的排序应该和真实的效用函数的排序结果一致
  - 计算的时间一定不能太长
  - 对于非终止状态，评估函数应该和取胜几率密切相关
- 评估函数的设计
  - 考虑状态的不同特征参数（features）
  - 以期望值（expected value）形式表达的状态的合理评价

第5讲 博弈问题求解

5.5 博弈中的实时决策：评估函数

- 评估函数设计举例：一种加权线性函数

$$EVAL(s) = w_1 f_1(s) + w_2 f_2(s) + \cdots + w_n f_n(s) = \sum_{i=1}^{n} w_i f_i(s)$$

其中，

- w_i：权重(棋子的单位价值)
- f_i：棋局的某个特征（棋子的数目）。

对于国际象棋

- w_i：每种棋子的单位价值。如“兵”为1，“象”为3，“车”为5，等
- f_i：棋盘上每种棋子的数目

第5讲 博弈问题求解

5.5 博弈中的实时决策：截断搜索

- 截断测试：
  - 用来代替终止测试
  - 确定何时应用EVAL函数

$$MINIMAX(s)=\begin{cases} UTILITY(s) & \text{即}TERMINAL-TEST(s)\text{为真}\ s\text{为终止状态} \\ max_{a\in Actions(s)}MINIMAX\big(RESULT(s,a)\big) & s\text{为}MAX\text{节点} \\ min_{a\in Actions(s)}MINIMAX\big(RESULT(s,a)\big) & s\text{为}MIN\text{节点} \end{cases}$$

$$H-MINIMAX(s,d)=\begin{cases} EVAL(s) & \text{若}CUTOFF-TEST(s,d)\text{为真} \\ max_{a\in Actions(s)}MINIMAX(RESULT(s,a),d+1) & s\text{为}MAX\text{节点} \\ min_{a\in Actions(s)}MINIMAX(RESULT(s,a),d+1) & s\text{为}MIN\text{节点} \end{cases}$$

其中，s 为状态，d 为最大深度（与计算时间相关）

作业

已知底层节点值的博弈树如下，方框表示MAX，圆圈表示MIN

1. 用最小最大博弈算法决策A的走步选择
2. 用alpha-beta剪枝，哪些节点不再需要检查