
问题解答

【大作业】二〇四八赛制

决策树原理与实践

# 数据结构与算法（Python）-19/0519

陈斌 gischen@pku.edu.cn 北京大学地球与空间科学学院

# 目录





# 问题解答



## 慕课相关

第11、12周慕课的解答（不用上课讲，canvas没有就反馈一下）

复习

## 大作业相关

方法的返回值是什么

程序架构　？异步协程？

大作业能否给个样例代码，文字说明有点难懂

天梯赛什么时候开始

测试能测试些什么啊？不同模块运行多少轮的时间？

天梯赛的时间及赛制

推荐一些便于小白自学的机器学习/人工智能网课

# 期末大作业时间进度



- **各组组长加入大作业组长群**
- **即日开始实习作业**

  开发算法，编程测试，热身天梯赛，撰写报告

  注意组员分工明确，协同合作

- **（不早于）5月26日在代码竞技场专区开展小组天梯赛**

  可以多次修改和提交代码，争取更高积分排名；

  5月31日晚结束小组天梯赛

- **（W16）6月2日（周二）课上进行算法竞赛决赛**
- **（W17）6月9日（周二）前提交完整作业**

  包括代码、实验报告

# 小组算法开发指南



## 详见github代码仓库

github: https://github.com/pkulab409/sessdsa.2048

国内码云镜像：https://gitee.com/chbpku/sessdsa.2048/

码云镜像国内高速访问，适合没有梯子的同学

## 各组必看文档

规则文档，用户文档，赛制文档

## 代码竞技场（免入校VPN）

http://gis4g.pku.edu.cn/ai_arena/

# 现场决赛赛制的介绍

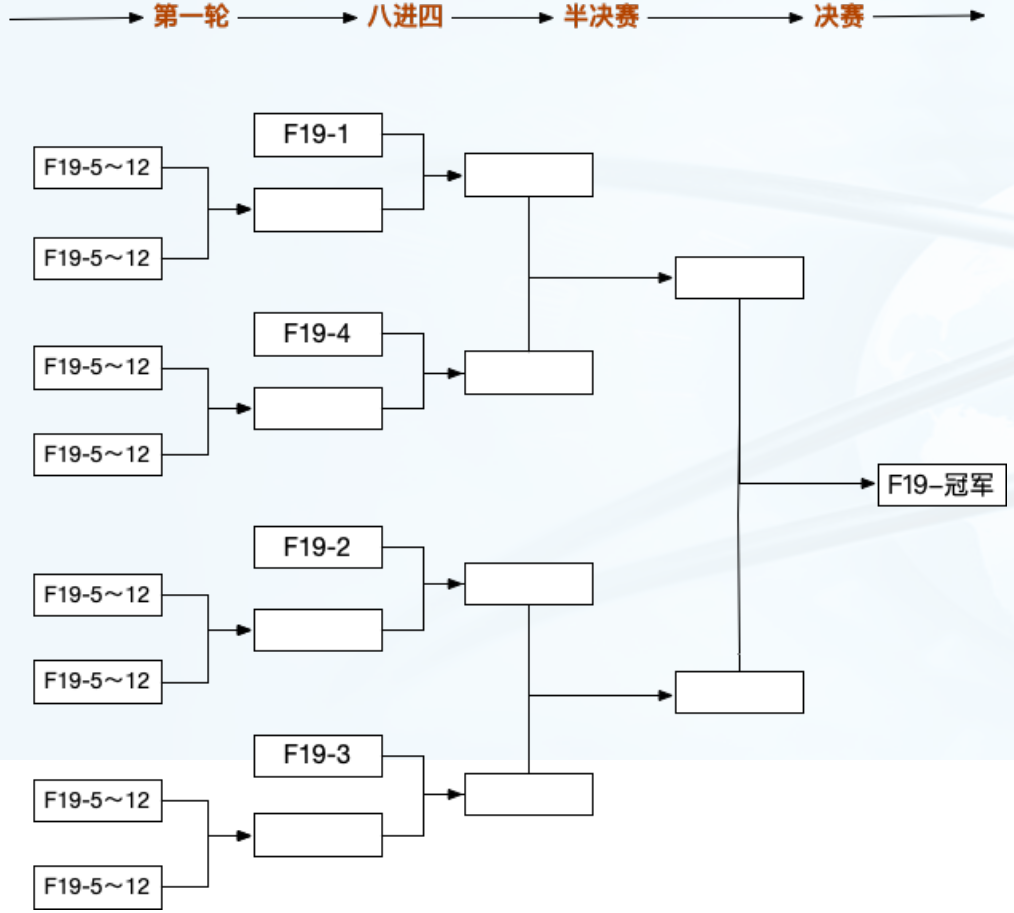


- **F19联盟：35组**
  12组出线，产生冠军1，亚军1，季军2
- **N19联盟：20组**
  8组出线，产生冠军1，亚军1，季军1
- **分为F19和N19两个联盟进行比赛**
- **所有单败淘汰赛对战双方先后手各对战10局，共20局**
- **比赛必须分出胜负，如果20局打平的话，则双方抽签决定先后手，用可视化对战一局分胜负。**

# 现场决赛赛制的介绍：F19联盟前12名出线



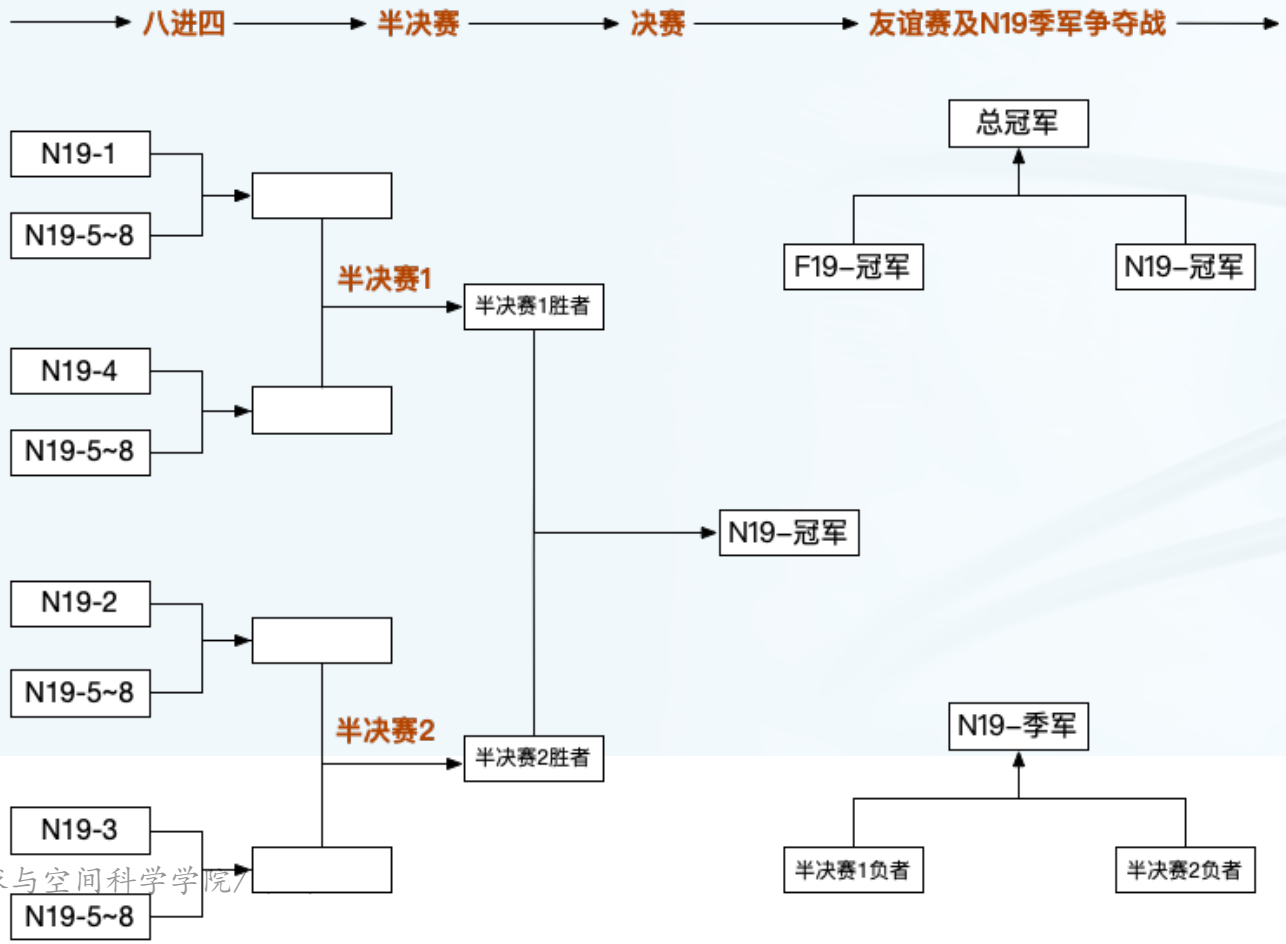
- 其中前4名第一轮轮空，如图分配直接进入8强赛上下半区
- 5～12名随机分配进入第一轮单败淘汰赛，胜者进入8强赛上下半区开展对战。



# 现场决赛赛制的介绍：N19联盟前8名出线



- 其中前4名按如图分配直接进入8强赛上下半区
- 5～8名随机分配进入8强赛上下半区开展对战。



# 大作业相关算法策略



## 完全信息的零和博弈

完全信息：双方都知道关于棋局的所有信息，以及查询下棋历史

零和博弈：只需要使对手收益最小化，自己的收益即可实现最大化

## 在对战的每一个回合中考虑两个阶段策略

## 下棋阶段

是选择下在对方棋盘实现封堵？

还是选择现在己方棋盘实现合并？

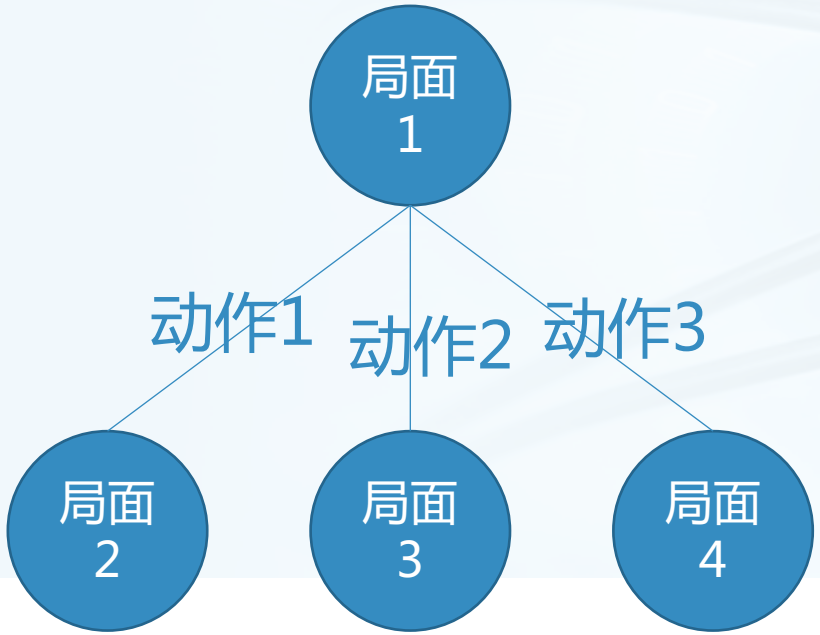
## 合并阶段

成功的棋子合并：己方内部合并；边界上的吞并对方

# 博弈树构建



- 博弈树的每个节点对应于每一个**局面**，每条边对应于一个**动作**
- 在完全信息零和博弈的条件下，能够构建简单的博弈树
- 如果在不完全信息、非零和博弈的情况下，博弈树较为复杂

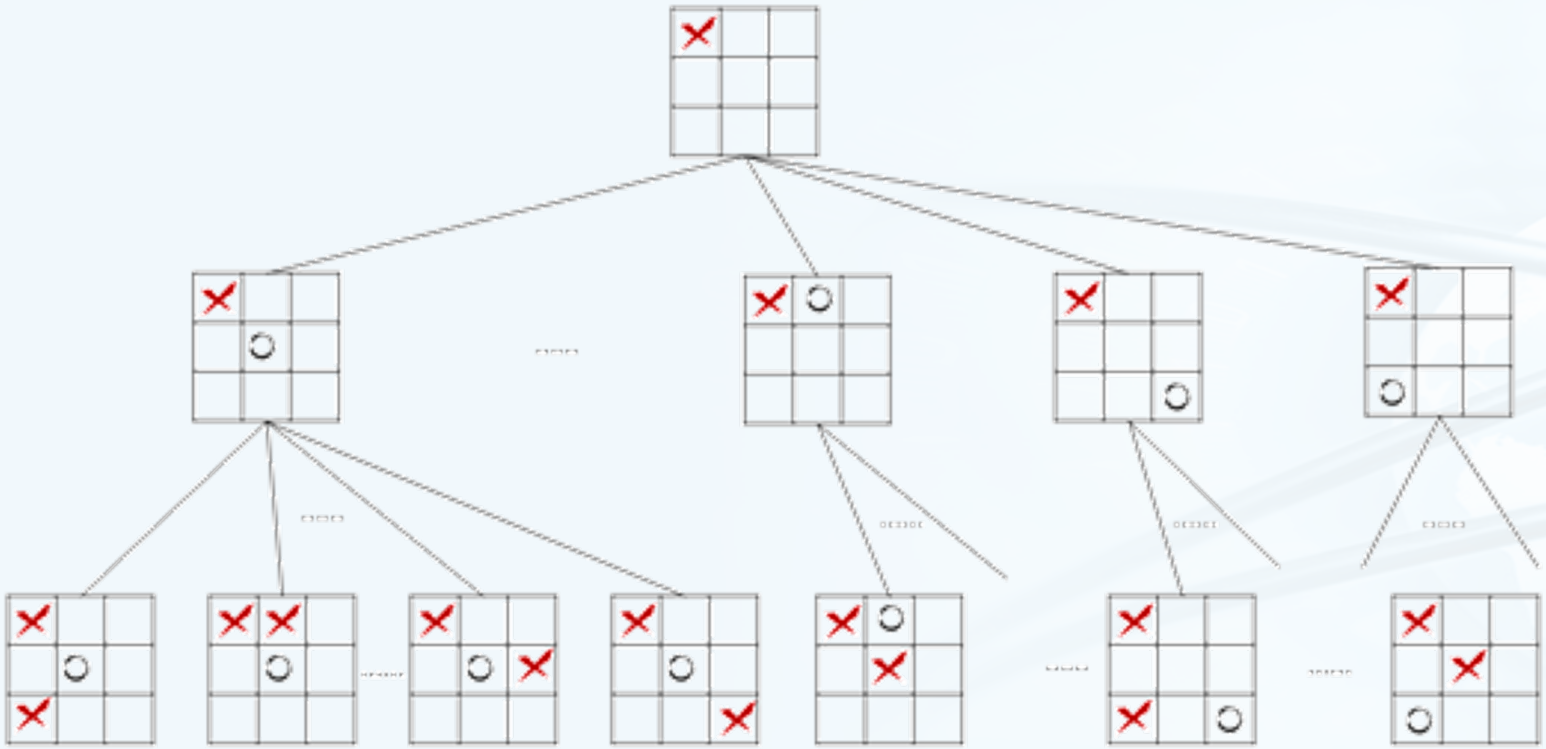


# 博弈树构建



## 博弈树（例子：井字棋）

最高有9层

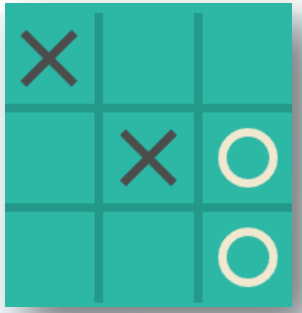


# 如何判断哪个动作最有利？



## 估值函数

对每一种局面给出一个估值

$f($  $) = ?$

## 一般的依据

静态子力、数量

动态特征：不同位置、时间阶段价值不同

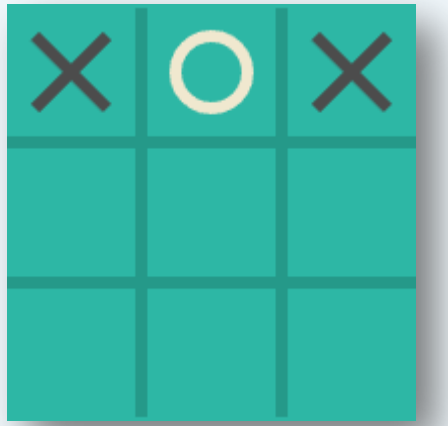
| | | 中国象棋 | | 国际象棋 | |
|---|---|---|---|---|---|
| | | 价值 | 数量 | 价值 | 数量 |
| | 兵(卒) | 1 | 5 | 2 | 8 |
| 守子 | 仕(士) | 2 | 2 | - | - |
| | 相(象) | 2 | 2 | - | - |
| 轻子 | 马 | 5 | 2 | 6 | 2 |
| | 炮 | 5 | 2 | - | - |
| | 象 | - | - | 6 | 2 |
| 重子 | 车 | 10 | 2 | 10 | 2 |
| | 后 | - | - | 18 | 1 |
| 总价值(双方) | | 106 | 32 | 156 | 32 |
| 棋盘大小 | | 90 | | 64 | |

| 棋子名称 | | 活动范围 | | | 子力价值 | | |
|---|---|---|---|---|---|---|---|
| 红 | 黑 | 中央 | 边线 | 角落 | 开局 | 中局 | 残局 |
| 帅 | 将 | 4 | 3 | 2 | - | - | - |
| 仕 | 士 | 4 | - | 1 | 1 | 2 | 2 |
| 相 | 象 | 4 | 2 | - | 2 | 2 | 3 |
| 马 | 马 | 8/6/4 | 4/3 | 2 | 4 | 5 | 5 |
| 车 | 车 | 17 | 17 | 17 | 10 | 10 | 10 |
| 炮 | 炮 | 17/13 | 17/14 | 17/15 | 5 | 5 | 6 |
| 兵 | 卒 | 1/3 | 1/2 | 1 | 2 | 1/3/5 | 3/2/1 |

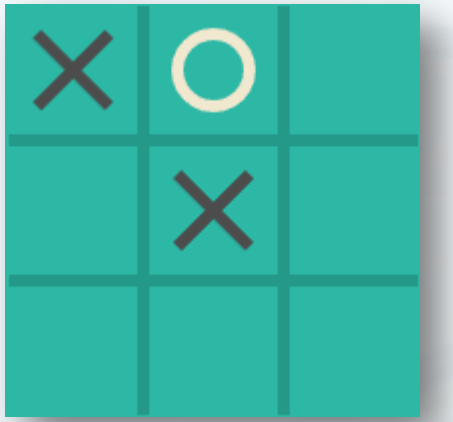
# 井字棋的估值函数



- 玩家X还存在可能性的行、列、斜线数减去
- 玩家O还存在可能性的行、列、斜线数



6-3=3

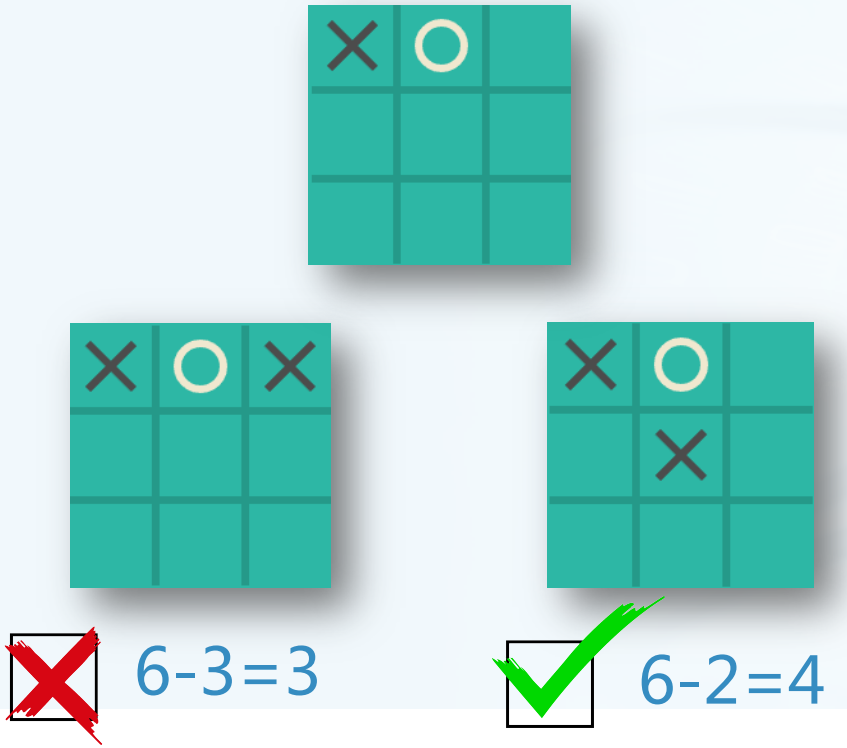


6-2=4

# 选取策略



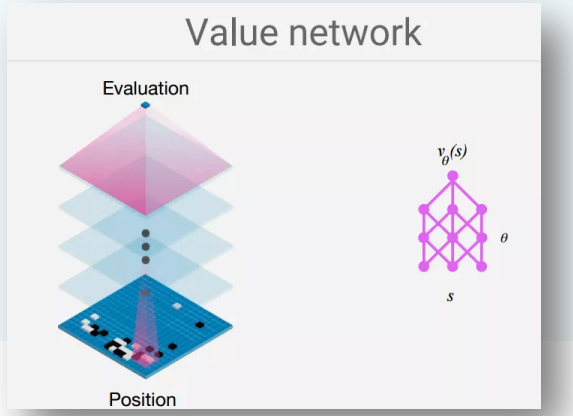
- 根据估值函数选择最优策略
- 最佳行动就是能够使得下一个状态的评估值**最大**的行动

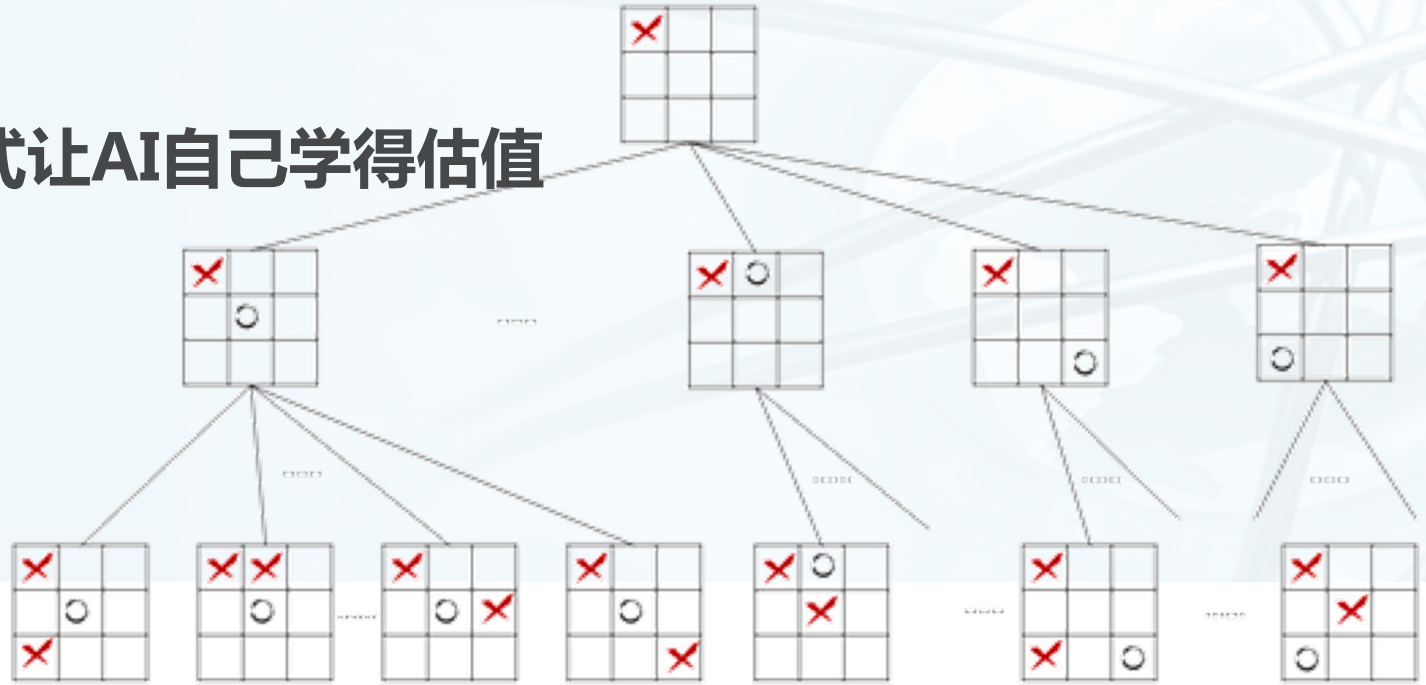


# 进一步改进



- 采用**多步**搜索策略
- 提高搜索的**深度**，尽量接近搜索过程的终止状态
- 搜索配合**剪枝**提高效率
- 改进估值函数
- 通过神经网络等方式让AI自己学得估值



# 最大最小值法



## 零和游戏中

玩家在可选的选项中选择将其优势**最大化**的选择

也就是说要选择令**对手**优势**最小化**的方法

## 回合制的游戏

双方都很**聪明**，都会采用最优策略

用**最大最小值法**统一表示

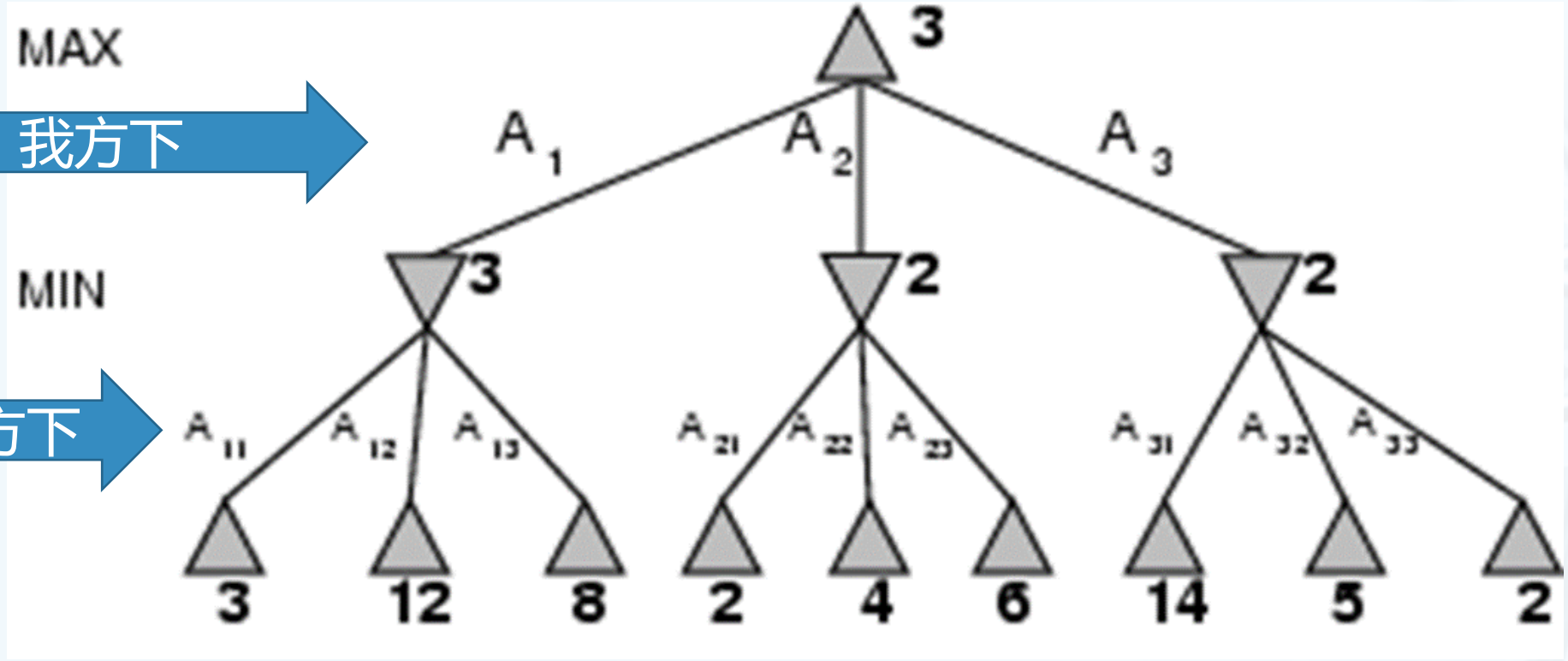
轮到我方下，选择我方的**最大**估值

轮到对方下，选择我方的**最小**估值

# minimax值



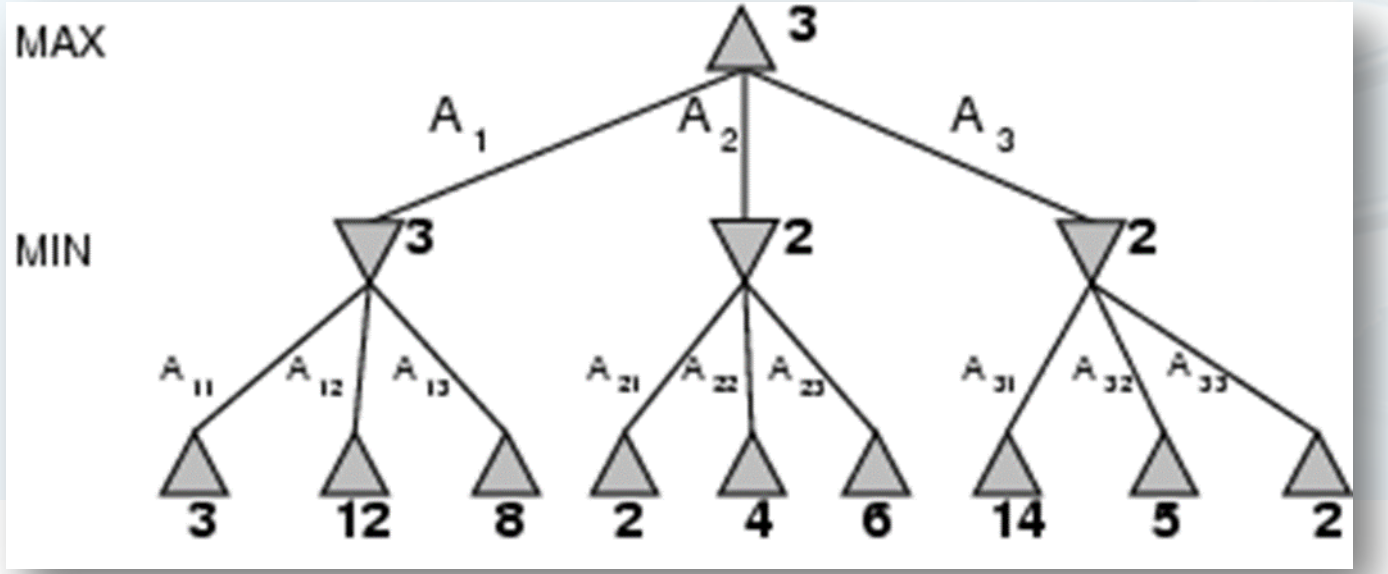
› 一个minimax决策树，包括max结点、min结点和终止结点



# minimax值：表示决策树上结点的估值



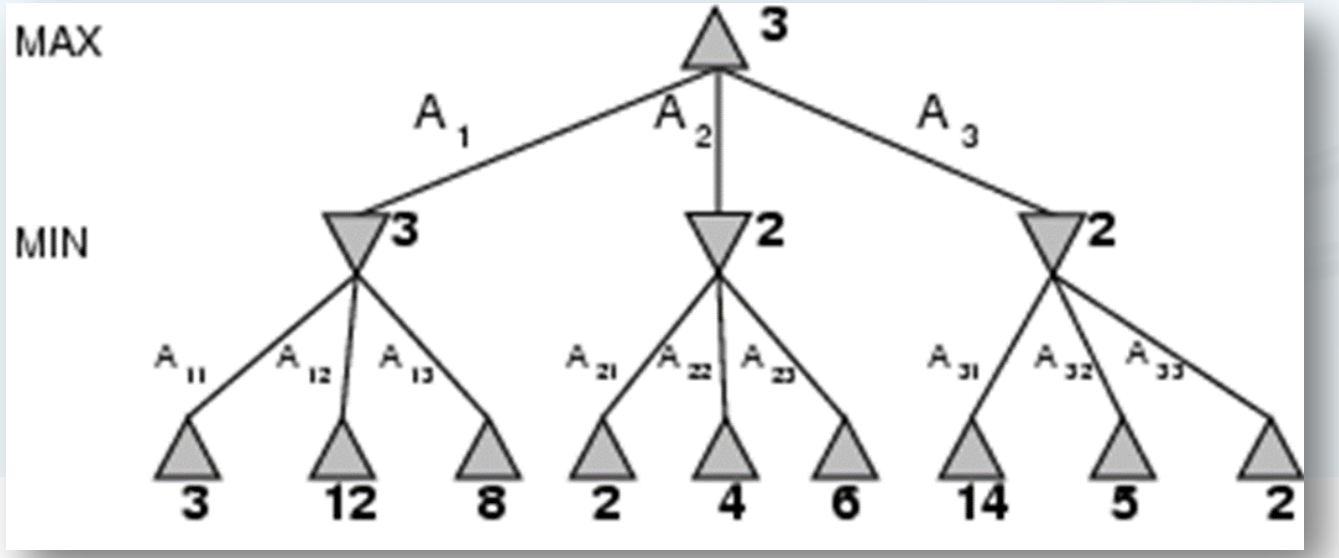
- 对于终止结点， minimax值等于直接对局面的估值
- 对于MAX结点，选择minimax值最大的子结点的值作为MAX结点的值
- 对于MIN结点，选择minimax值最小的子结点的值作为MIN结点的值



# 算法过程

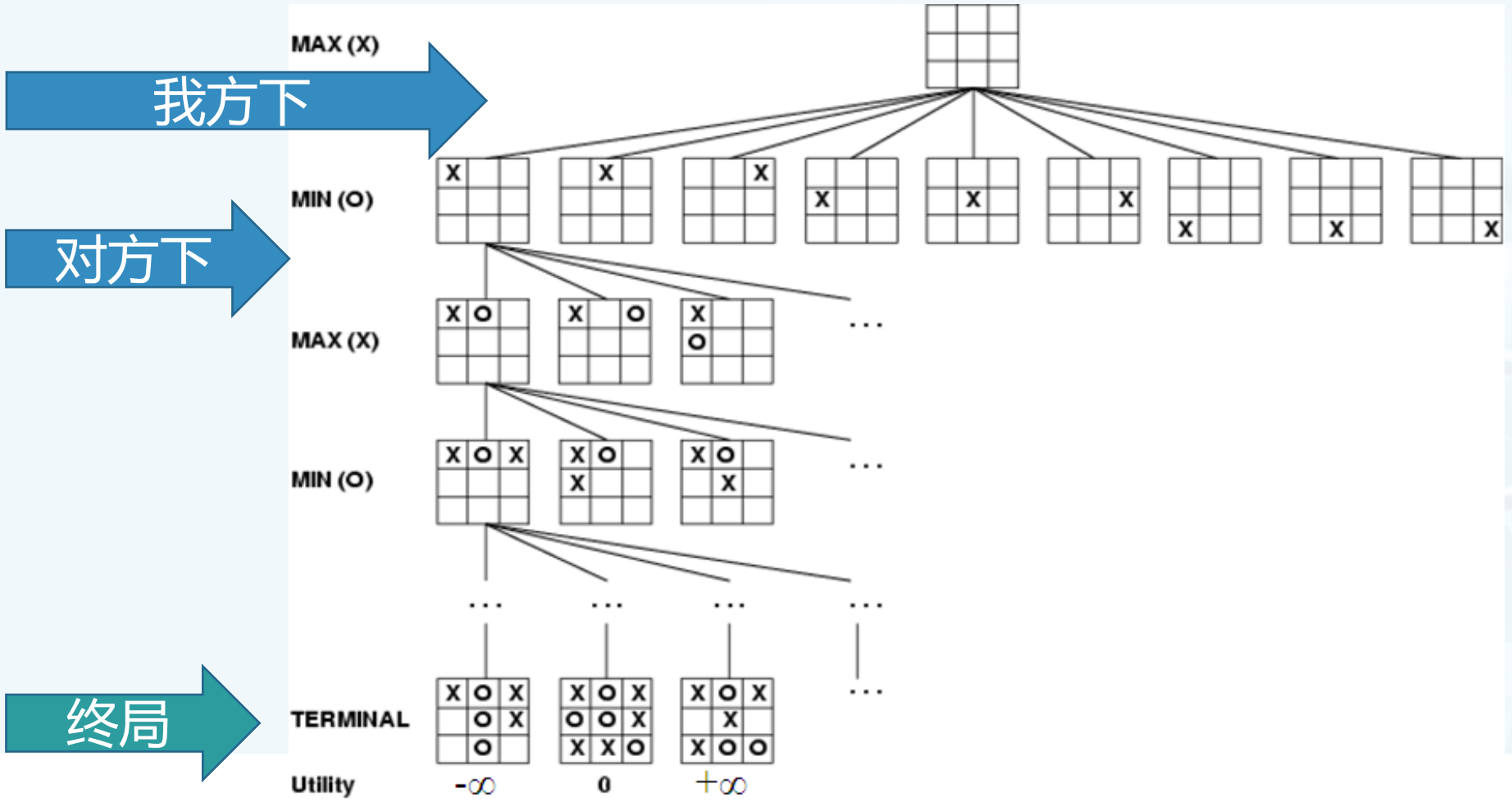


1. 构建决策树
2. 将评估函数应用于终局的**叶子**结点
3. 自底向上计算每个结点的minimax值
4. 从根结点选择minimax值**最大**的分支，作为**行动策略**



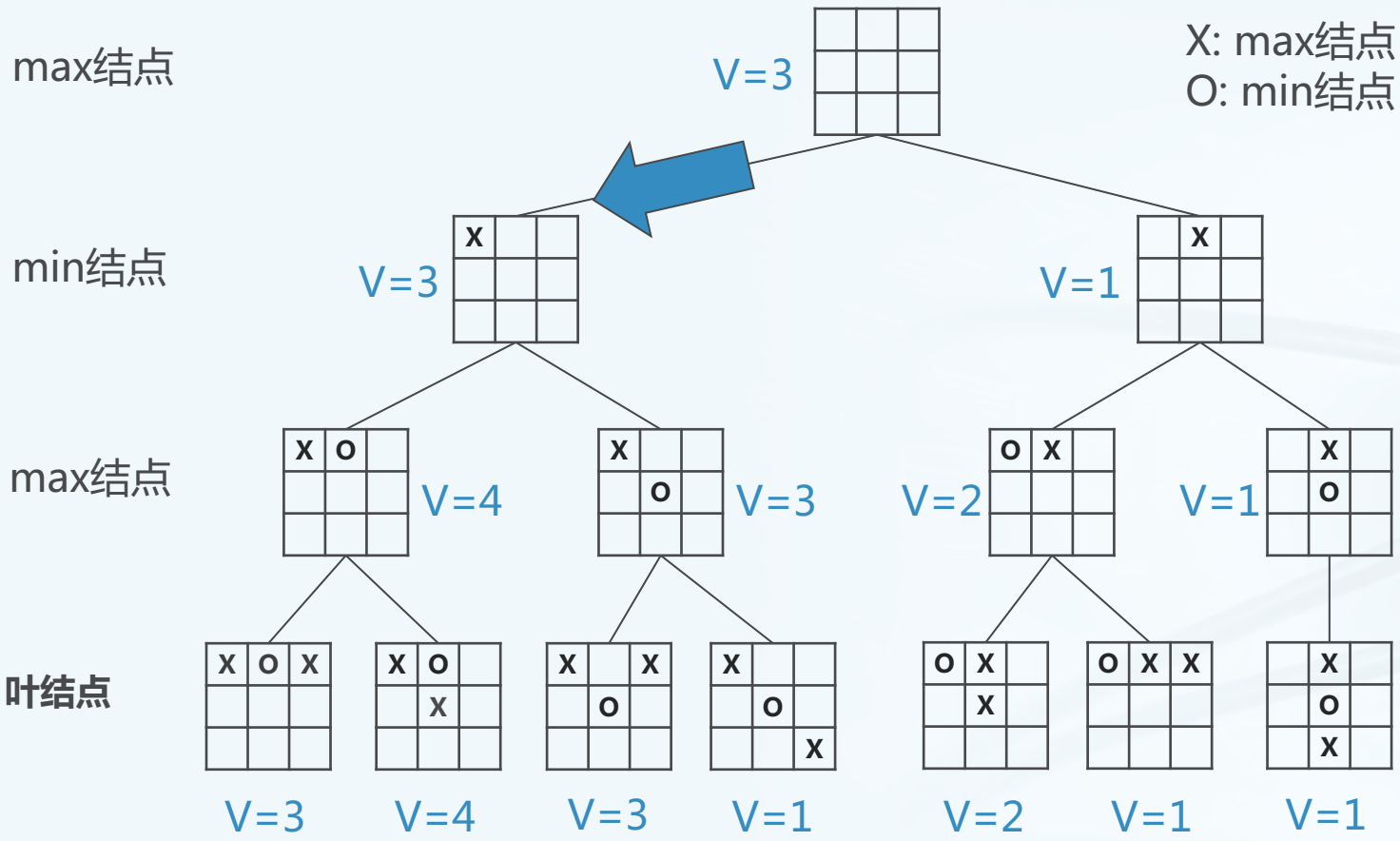
# 构建决策树，叶子结点估值





# 算法过程演示

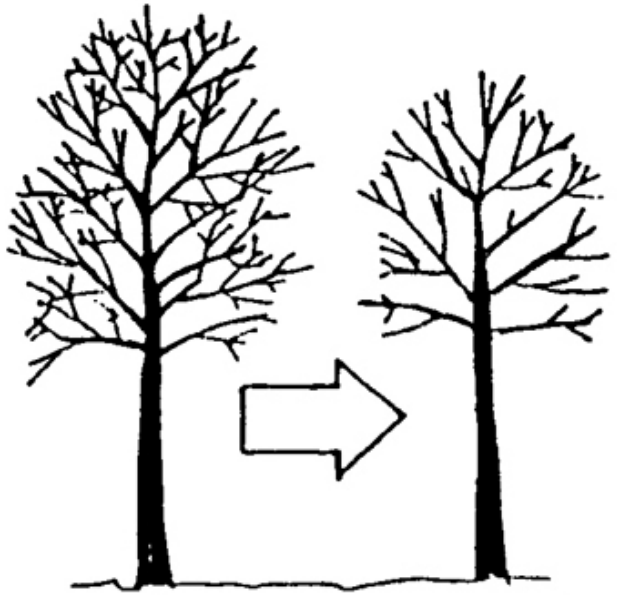



max结点
min结点
max结点
叶结点
X: max结点
O: min结点
V=3
V=3
V=1
V=4
V=3
V=2
V=1
V=3
V=4
V=3
V=1
V=2
V=1
V=1

# Minimax的优化：剪枝



- Minimax需要展开**整个**决策树
- 对于局面复杂的问题，需要**很大**的空间
- 有**部分结点**跟最后的结果**无关**，无需展开局面和计算估值
- 不计算这些结点可节省算法搜索的时间
- 去掉这些节点的过程叫做“剪枝”



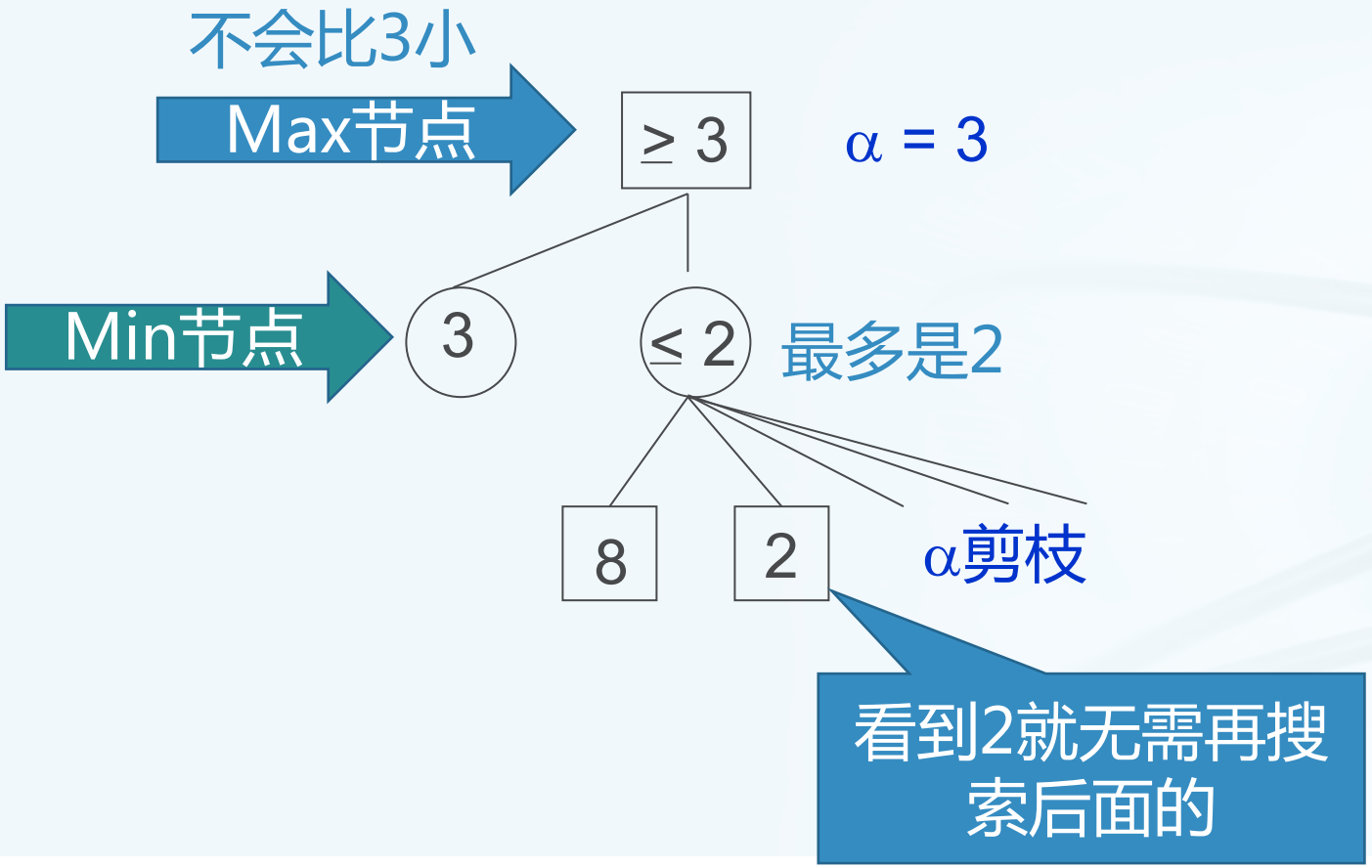
# Alpha-Beta剪枝



- **加速**minimax搜索过程
- 每个结点存储局面估值之外，还存储**可能取值**的上下界
- 估值：minimax值
- 下界（最小可能值）：Alpha值
- 上界（最大可能值）：Beta值

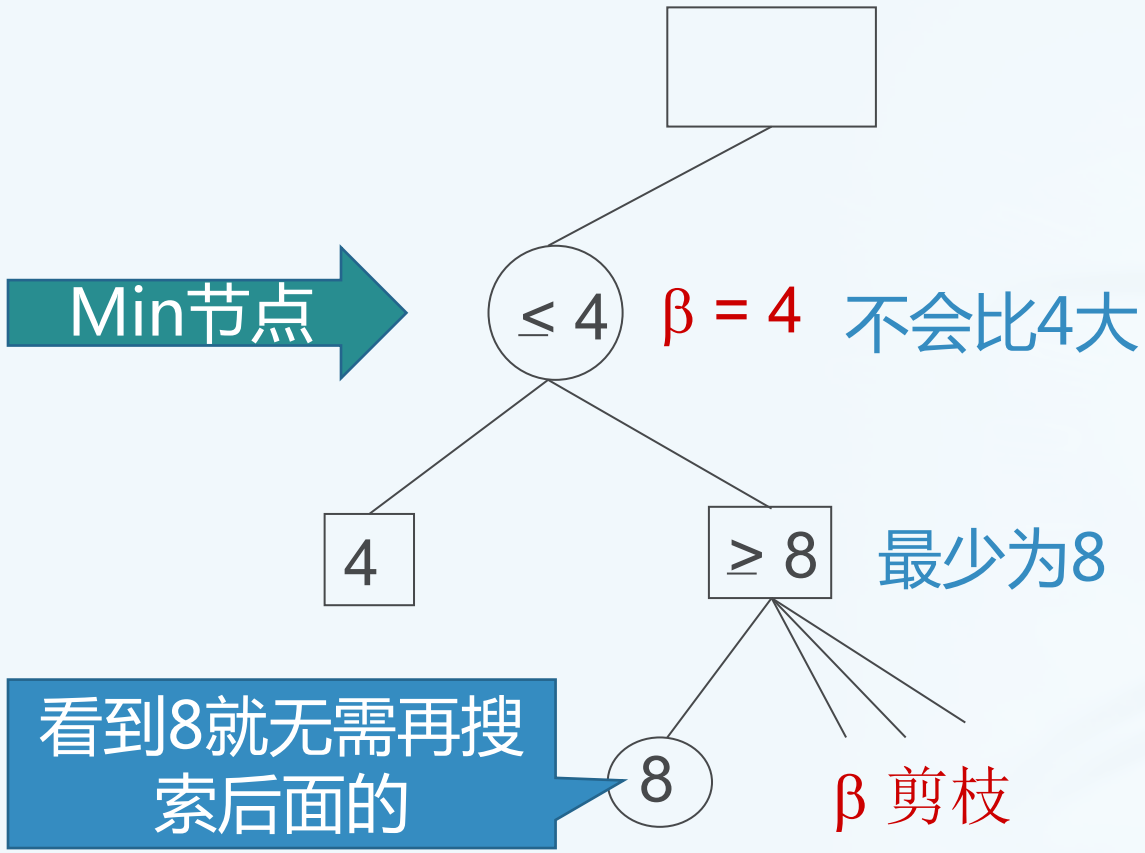
# Alpha剪枝：搜索MAX节点的子节点时





# Beta剪枝：搜索Min节点的子节点时





# 算法过程



1. 开始构建决策树
2. 将估值函数应用于叶子结点
3. 深度优先搜索，传递并更新α、β、结点值

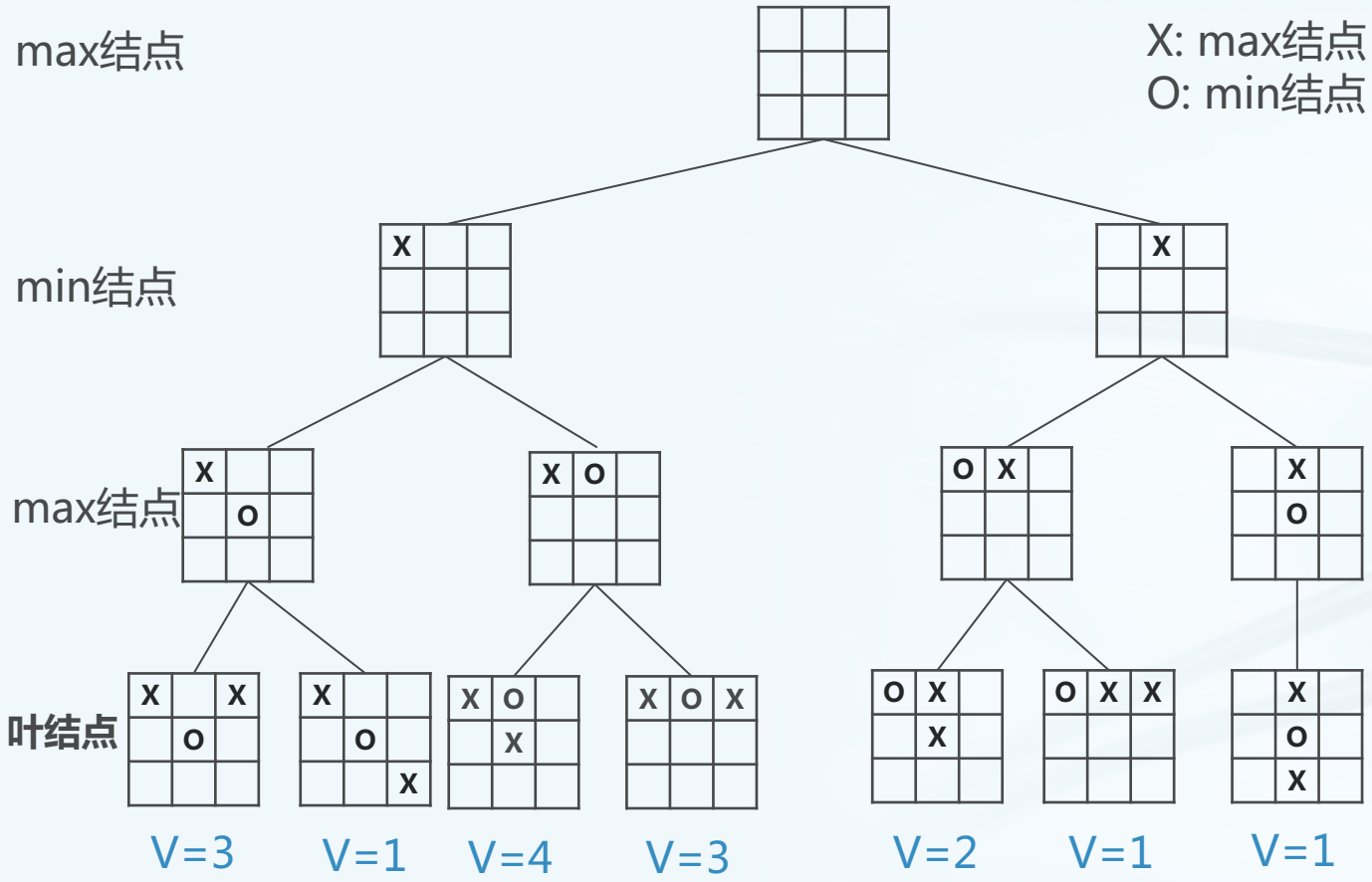
   Max结点更新α值(下限)

   Min结点更新β值(上限)

4. 从根结点选择**评估值最大**的分支，作为行动策略

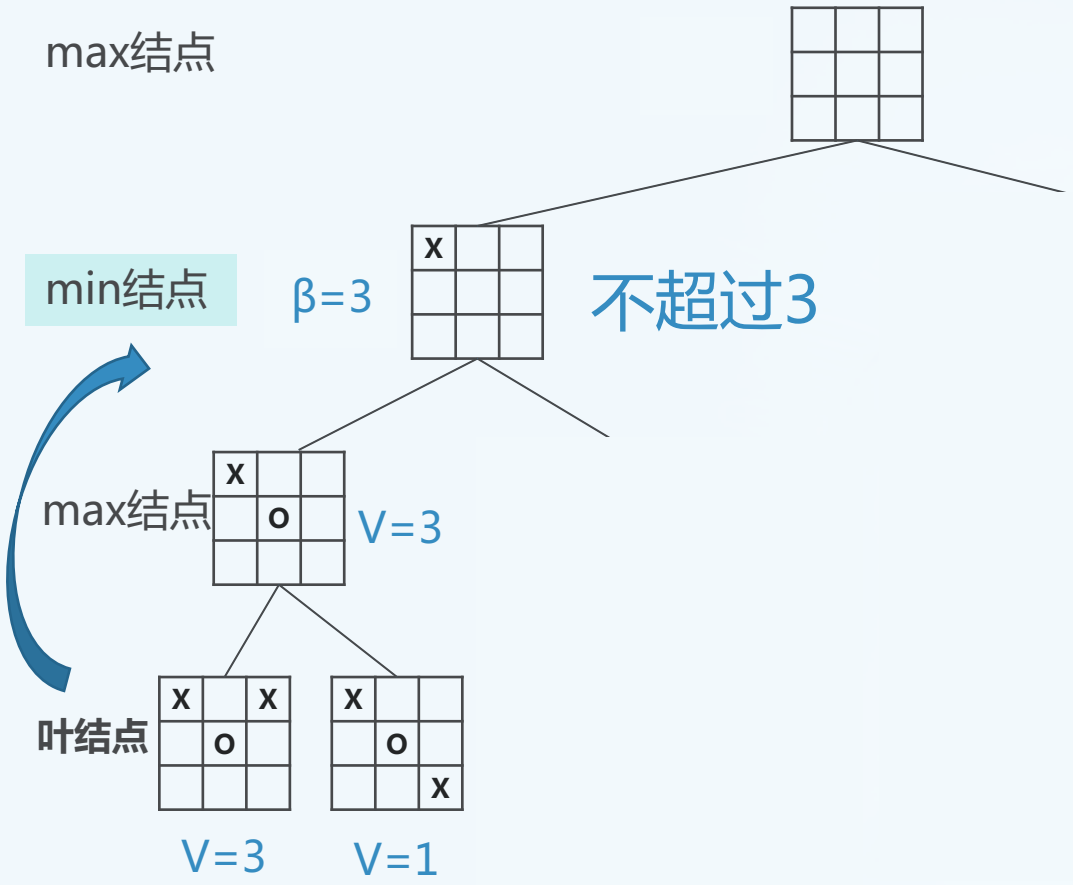
# 算法过程：未剪枝





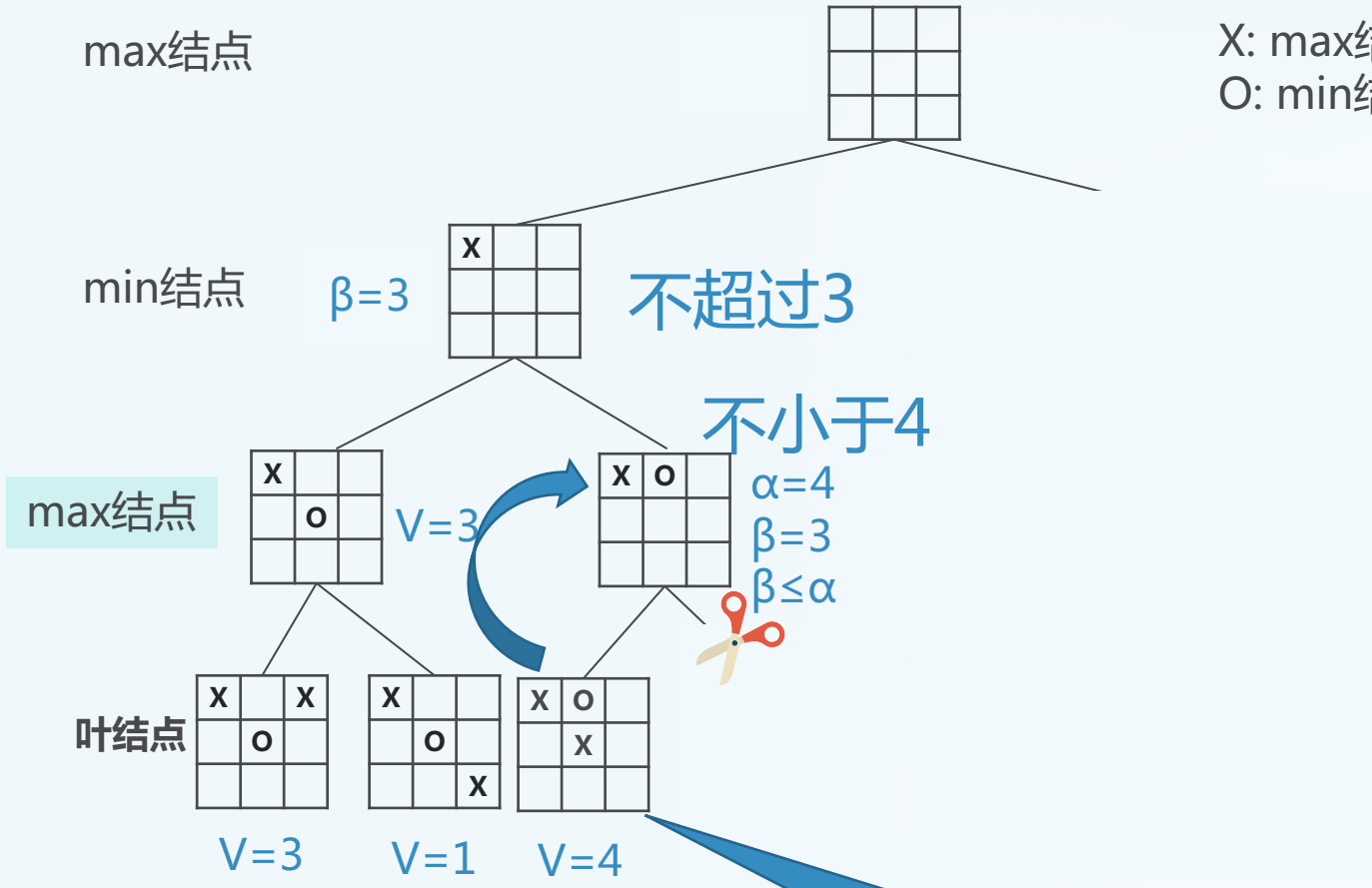
# 算法过程





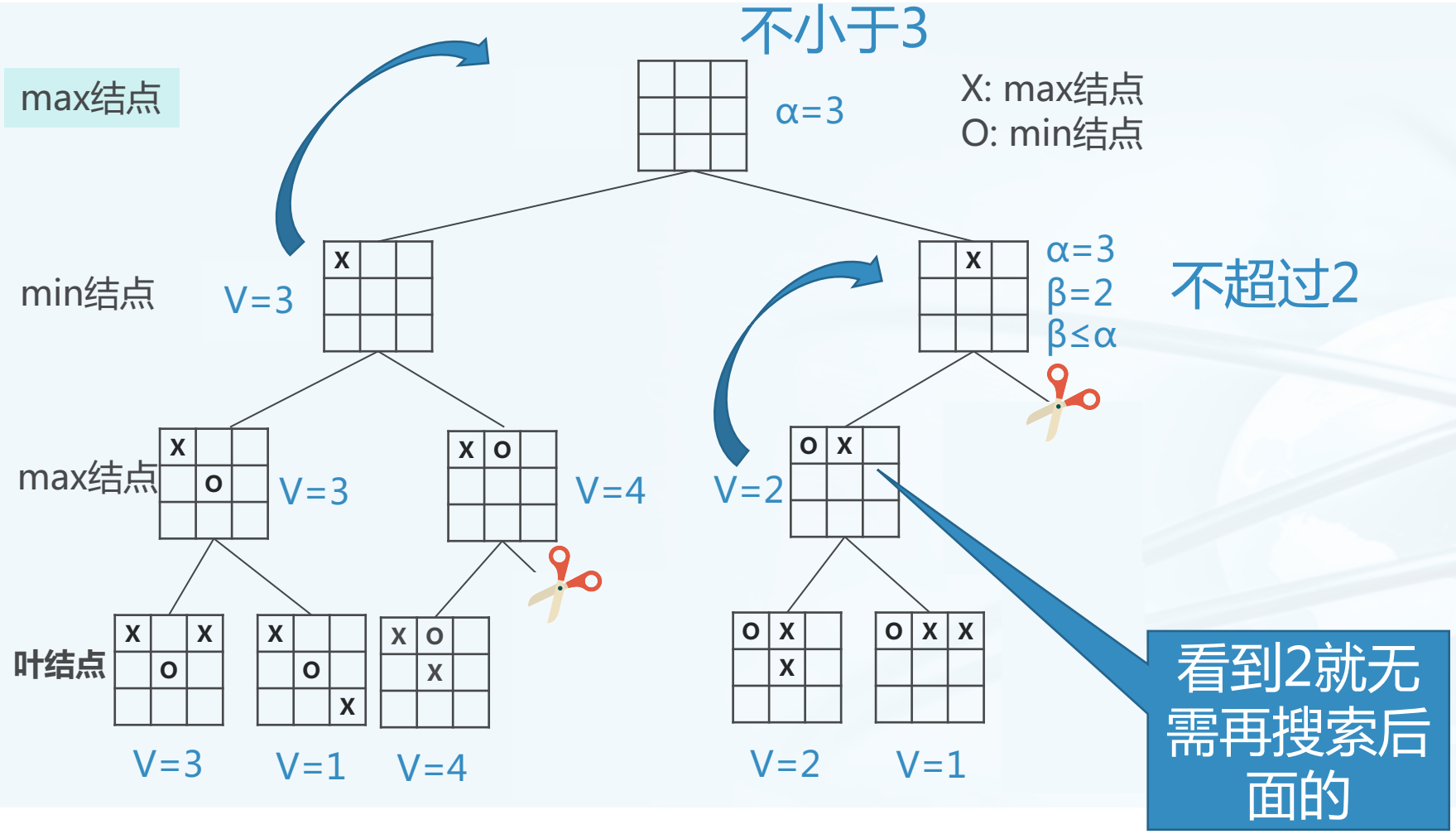
# 算法过程：Beta剪枝




max结点
min结点
β=3
不超过3
不小于4
α=4
β=3
β≤α
max结点
V=3
叶结点
V=3
V=1
V=4
X: max结点
O: min结点
看到4就无需再搜索后面的

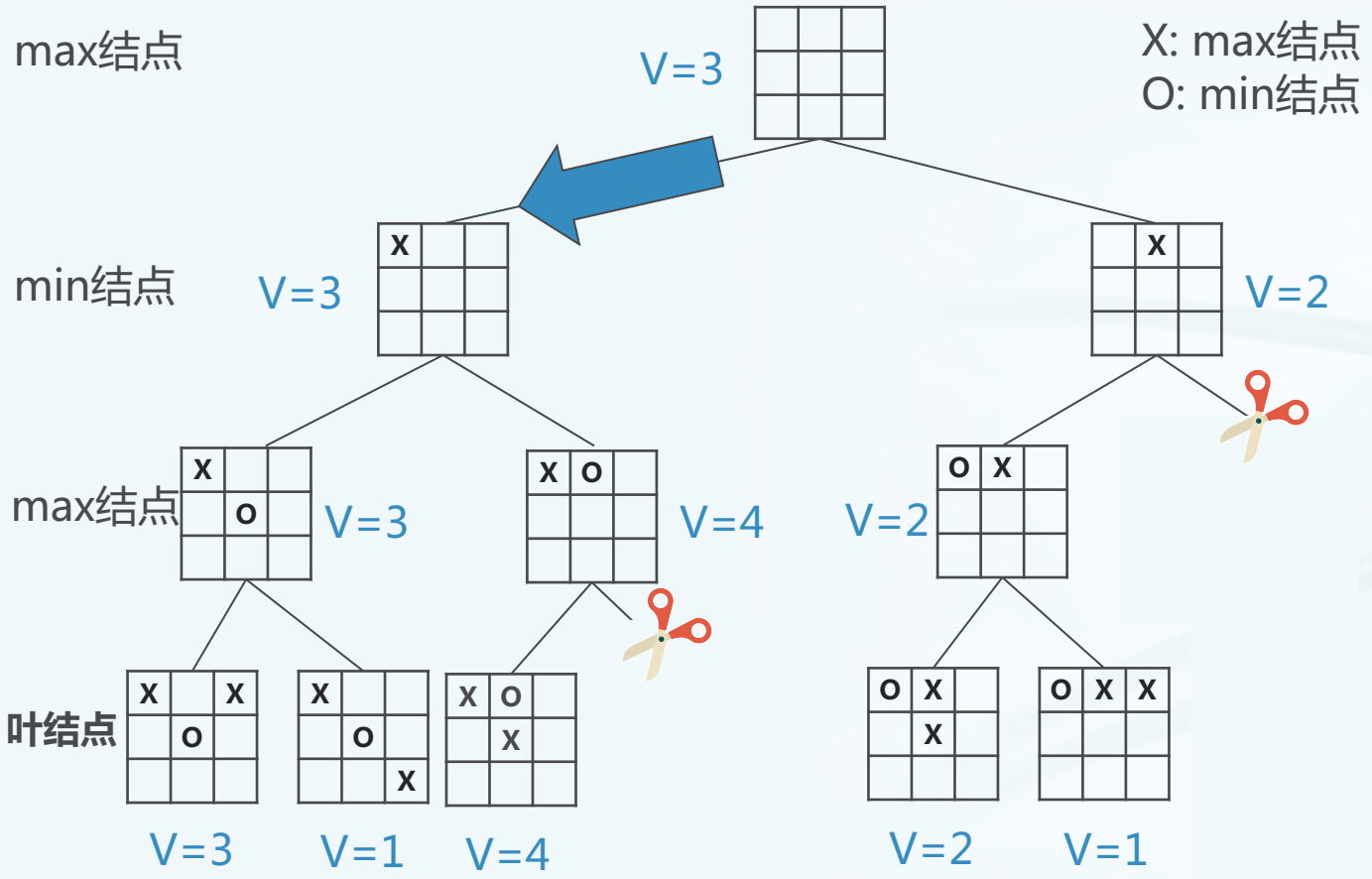
# 算法过程：Alpha剪枝





# 算法过程




max结点
min结点
max结点
叶结点
V=3
V=3
V=2
V=3
V=4
V=2
V=3
V=1
V=4
V=2
V=1
X: max结点
O: min结点

# Alpha-Beta剪枝的局限性



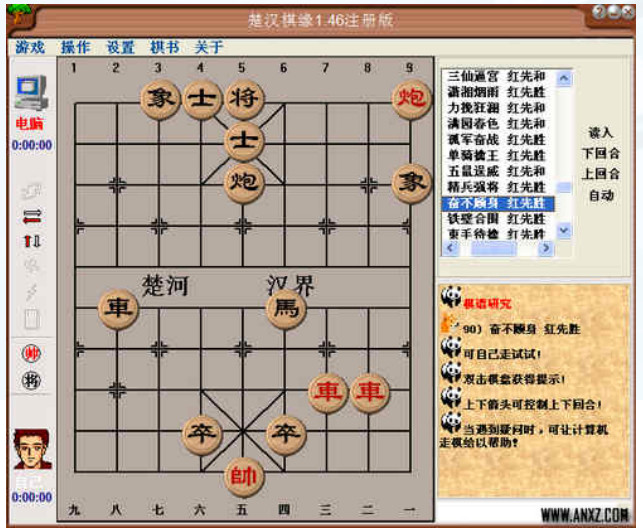
- 受到搜索次序的影响很大
- 并不都能实现有效剪枝

# 启发式算法



- 在搜索过程中，启发式算法被定义成一系列额外的规则
- 经验法则，利用一些特定的知识

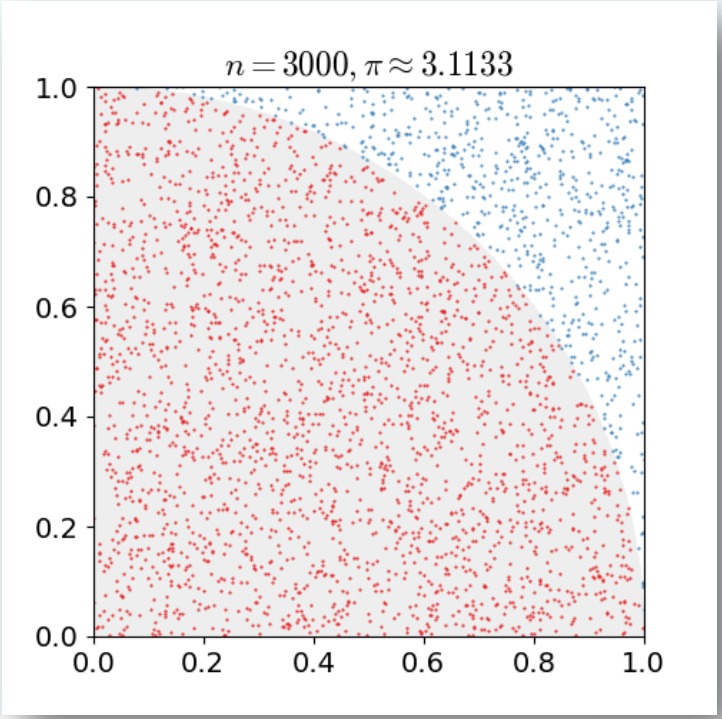
  “高手怎么下，我也怎么下”
- 它常能发现很不错的解，但也没办法证明它不会得到较坏的解
- 它通常可在合理时间解出答案，但也没办法知道它是否每次都可以这样的速度求解



# 蒙特卡洛方法



- 通过**随机采样**计算得到近似结果
- 一种通用的计算方法

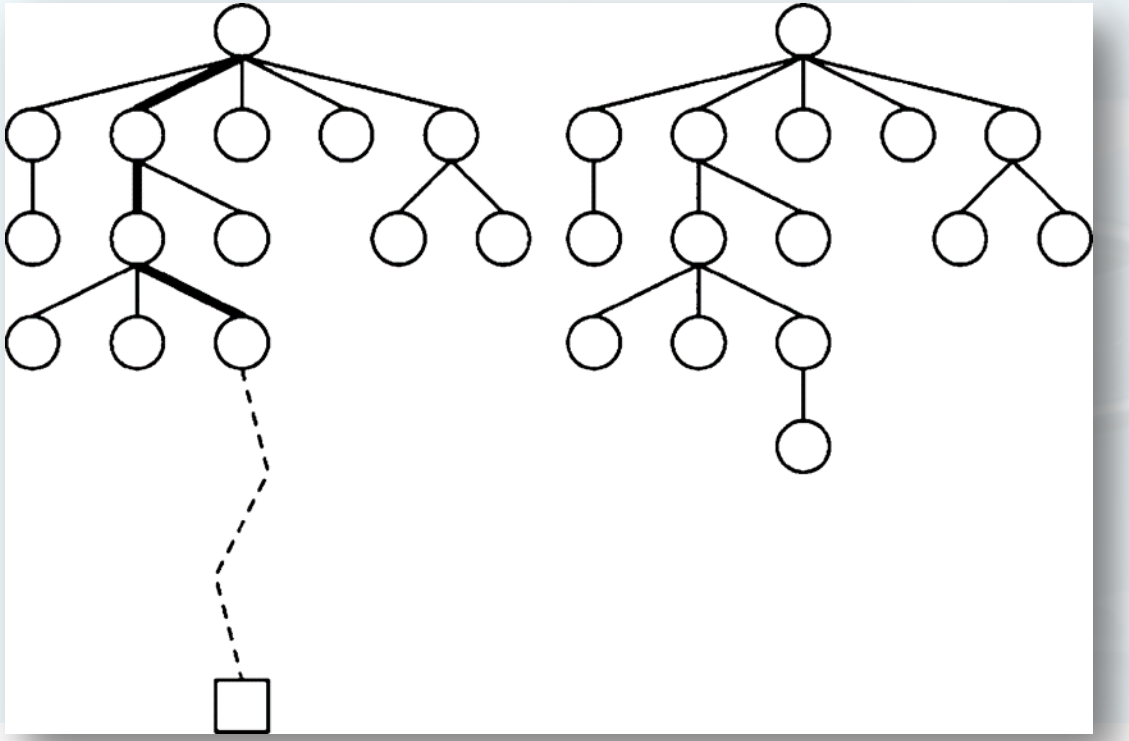


# 蒙特卡洛树搜索（MCTS）



- 一种通过在决策空间中**随机采样**并根据结果构建决策树来寻找最优策略的方法
- 决策树的构建

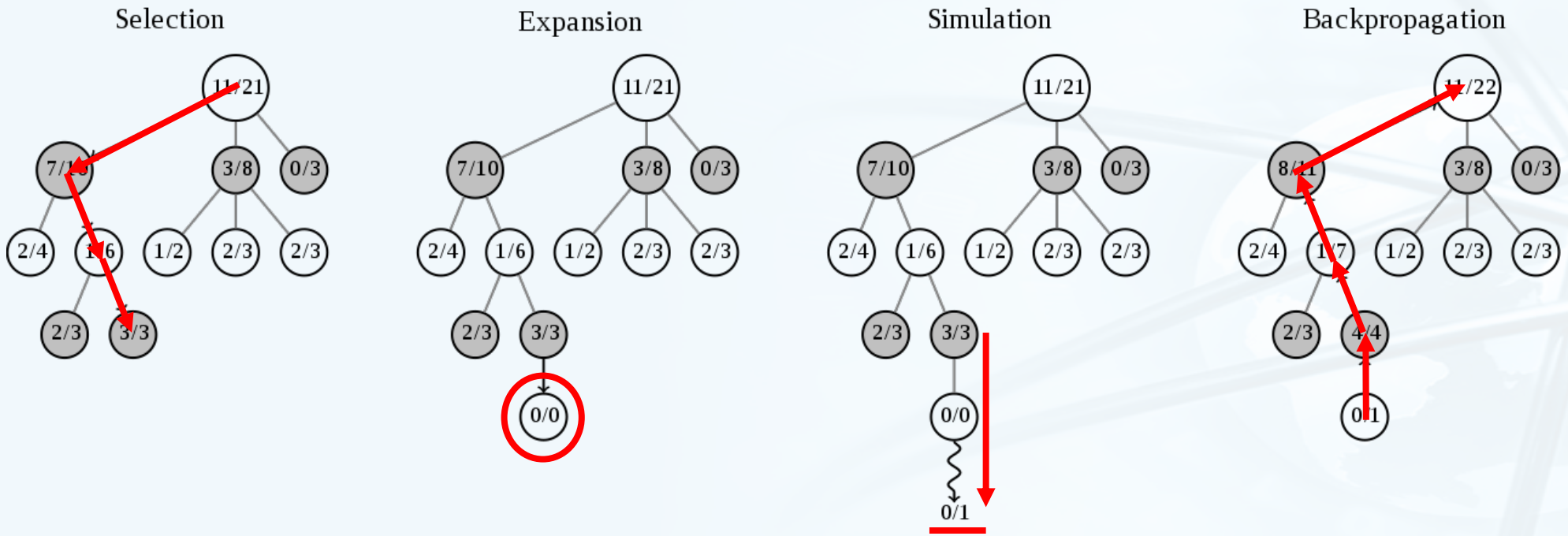
  选择、扩张、模拟、反馈



# MCTS例子



› 可以用来优化估值函数，以及得到更好的选择策略



# 小结



- 棋局的估值函数
- 决策树：最大最小法
- Alpha-Beta剪枝
- 启发式规则
- 蒙特卡洛树搜索