南京大學
NANJING UNIVERSITY

人工智能导论

对抗搜索

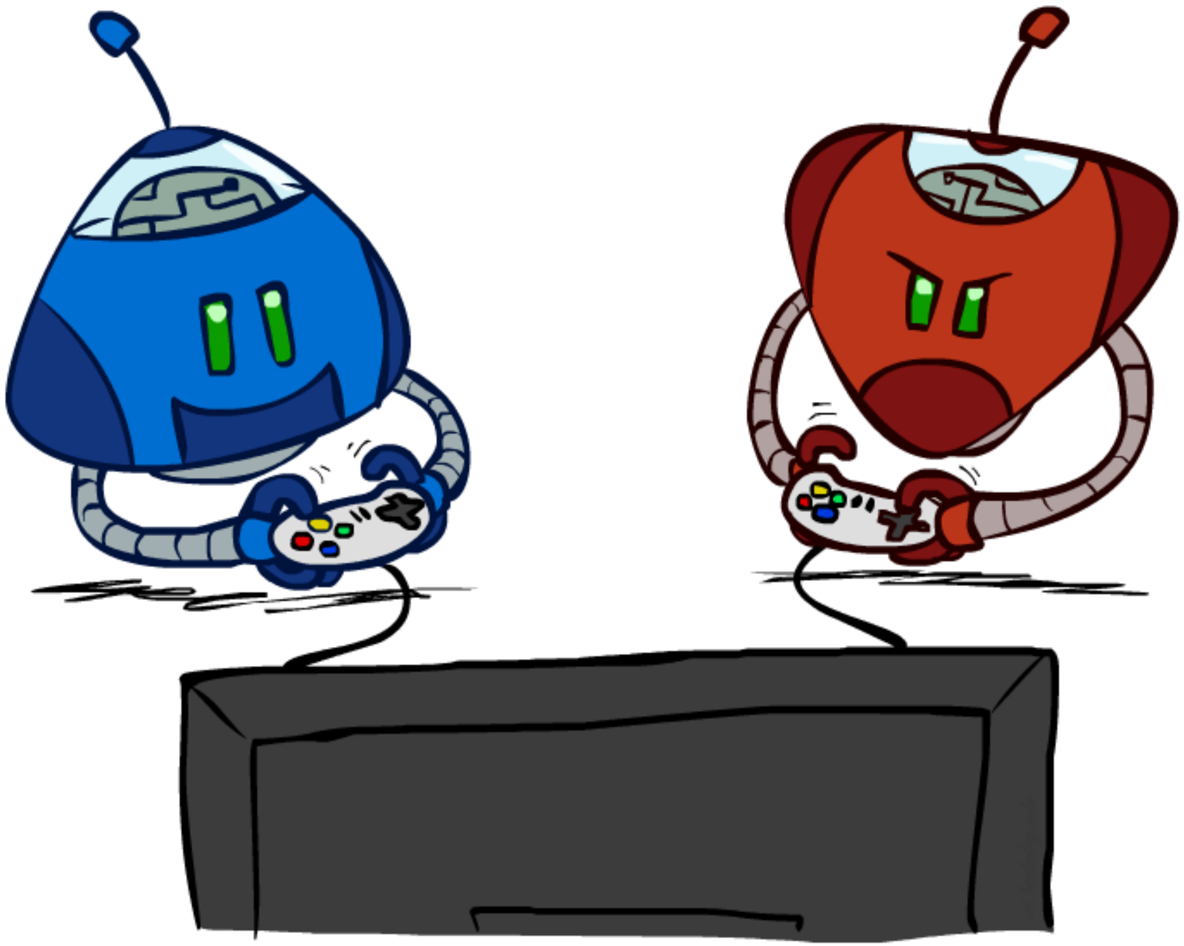
郭兰哲

南京大学 智能科学与技术学院

Homepage: www.lamda.nju.edu.cn/guolz

Email: guolz@nju.edu.cn

对抗搜索



提纲

- 对抗博弈
  - 双人零和博弈
- 确定性搜索
  - 最大最小搜索
  - Alpha-beta 剪枝
- 基于模拟的搜索
  - 蒙特卡洛树搜索

提纲

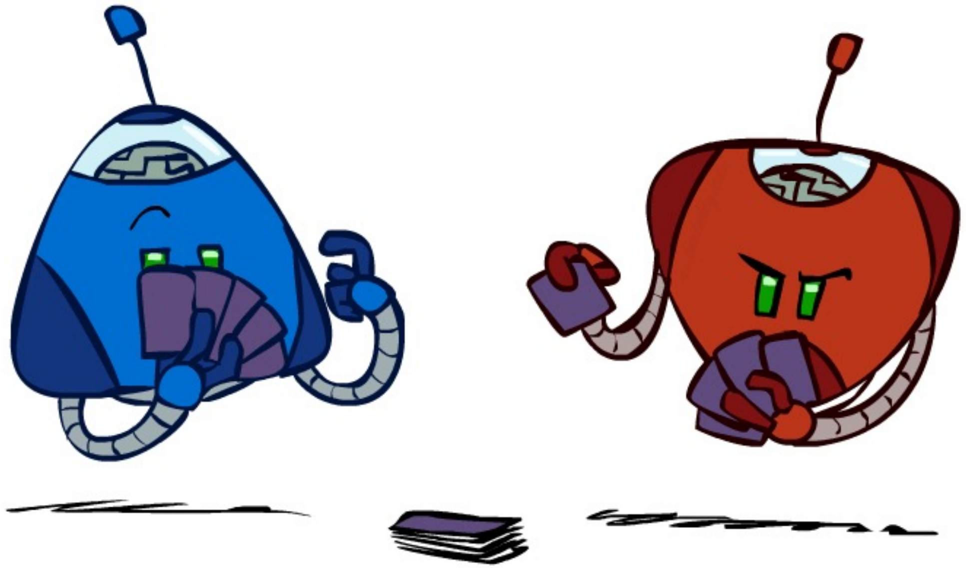
- 对抗博弈
  - 双人零和博弈
- 确定性搜索
  - 最大最小搜索
  - Alpha-beta 剪枝
- 基于模拟的搜索
  - 蒙特卡洛树搜索

博弈

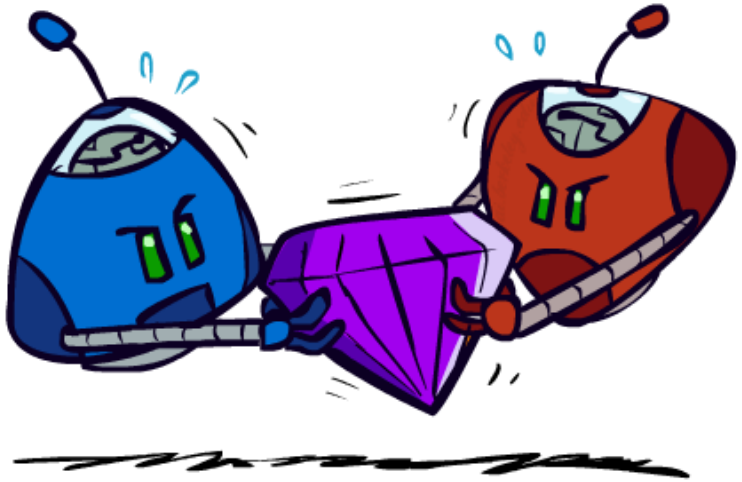
对抗搜索（Adversarial Search）也称为博弈搜索（Game Search）

博弈的种类：

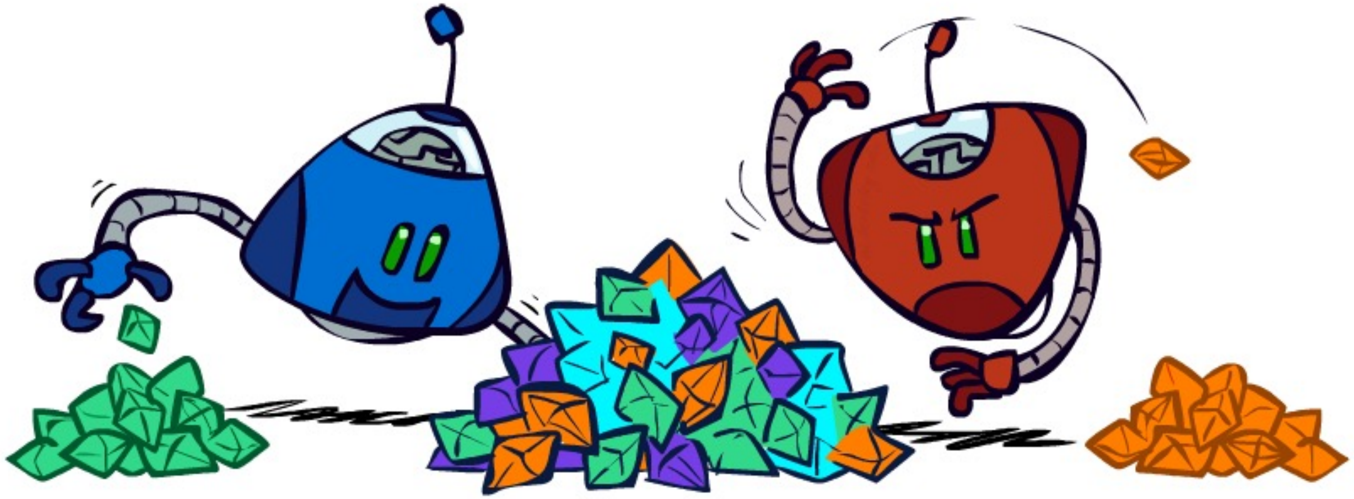
- ✓ 确定的、有随机性的
- ✓ 是否有完整信息？
- ✓ 几个玩家？
- ✓ 是不是零和博弈？



零和博弈



一个玩家赢了，则对手一定输了



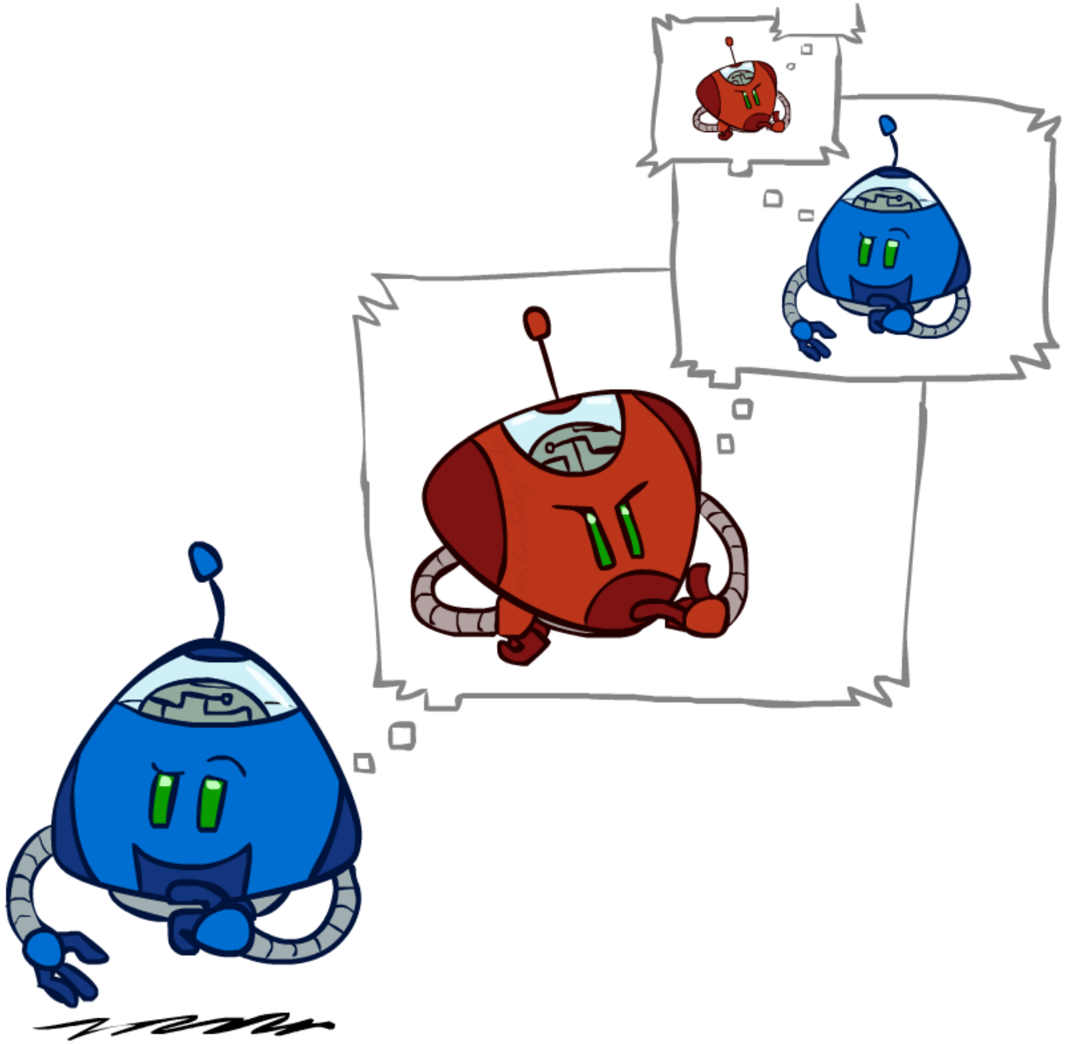
你可能赚了，但我也不亏

双人零和博弈

我们考虑**信息确定、全局可观察、竞争对手轮流行动、输赢收益零和**假设下的双人博弈问题

vs.

对抗搜索



Two-Step Game

- Two step game: 首先，Alice选择第 i 行，然后，**Bob**选择第 j 列
- 结果：**Alice**输（**Bob**赢）第 i 行第 j 列的元素值
- A MinMax Game

3	5	2
6	8	4
7	10	9

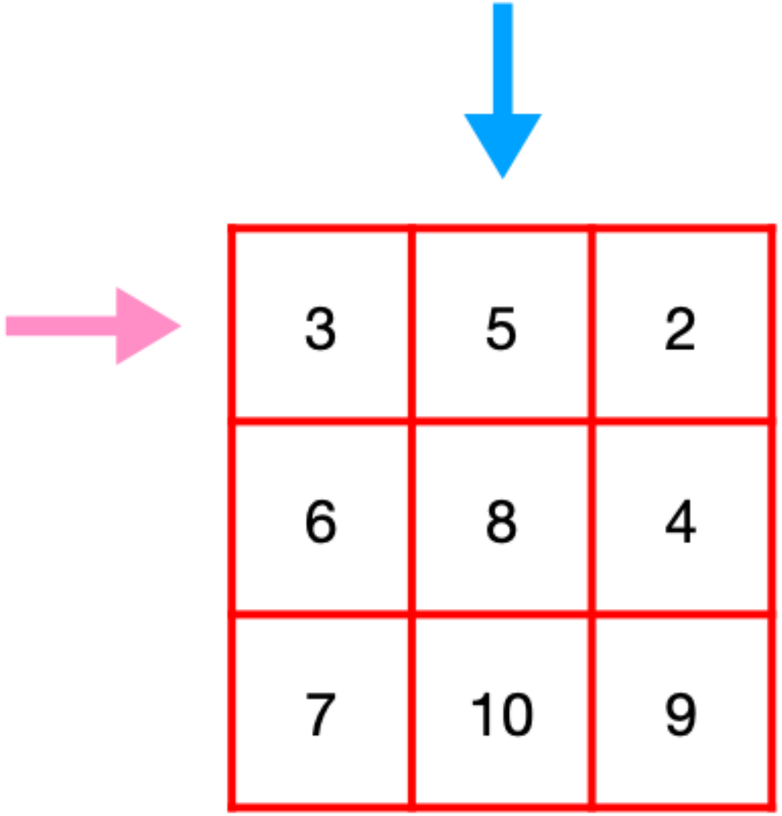
Two-Step Game

- Two step game: 首先，**Alice**选择第 i 行，然后，**Bob**选择第 j 列
- 结果：**Alice**输（**Bob**赢）第 i 行第 j 列的元素值
- A MinMax Game

3	5	2
6	8	4
7	10	9

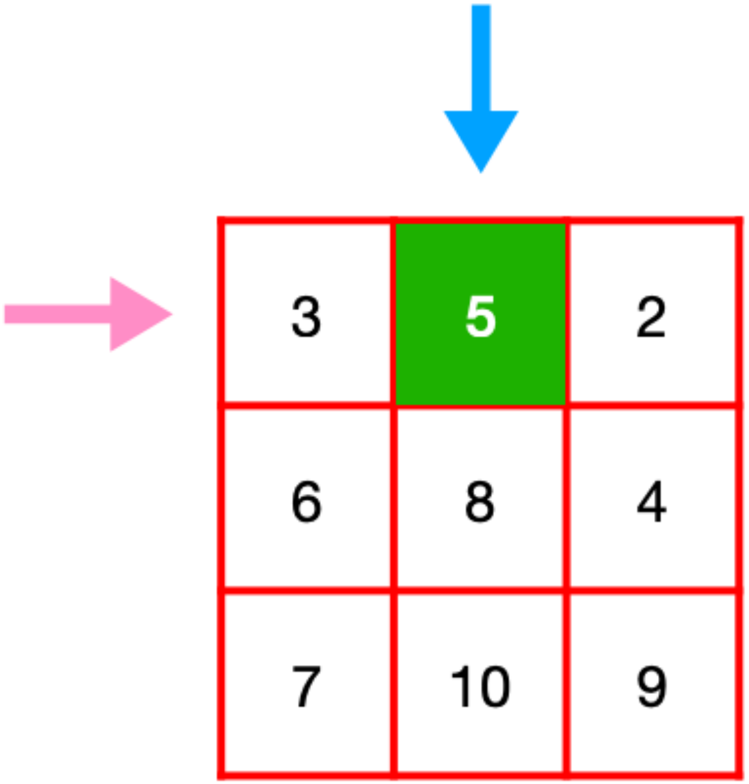
Two-Step Game

- Two step game: 首先，Alice选择第 i 行，然后，Bob选择第 j 列
- 结果：Alice输（Bob赢）第 i 行第 j 列的元素值
- A MinMax Game



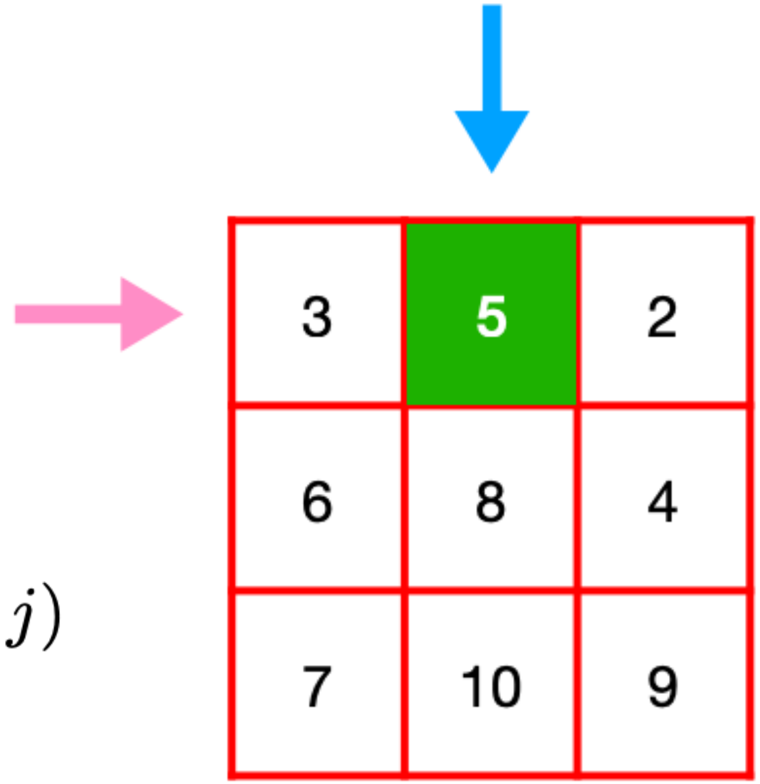
Two-Step Game

- Two step game: 首先，**Alice**选择第 i 行，然后，**Bob**选择第 j 列
- 结果：**Alice**输（**Bob**赢）第 i 行第 j 列的元素值
- A MinMax Game



Two-Step Game

- Two step game: 首先，Alice选择第 i 行，然后，Bob选择第 j 列
- 结果：Alice输（Bob赢）第 i 行第 j 列的元素值
- A MinMax Game



$$U(i^*, j^*) = \min_i \max_j U(i,j)$$

Two-Step Game

- Two step game: 首先，**Bob**选择第 j 列，然后，**Alice**选择第 i 行
- 结果：**Alice**输（**Bob**赢）第 i 行第 j 列的元素值
- A MaxMin Game

3	5	2
6	8	4
7	10	9

Two-Step Game

- Two step game: 首先，**Bob**选择第 j 列，然后，**Alice**选择第 i 行
- 结果：**Alice**输（**Bob**赢）第 i 行第 j 列的元素值
- A MaxMin Game

3	5	2
6	8	4
7	10	9

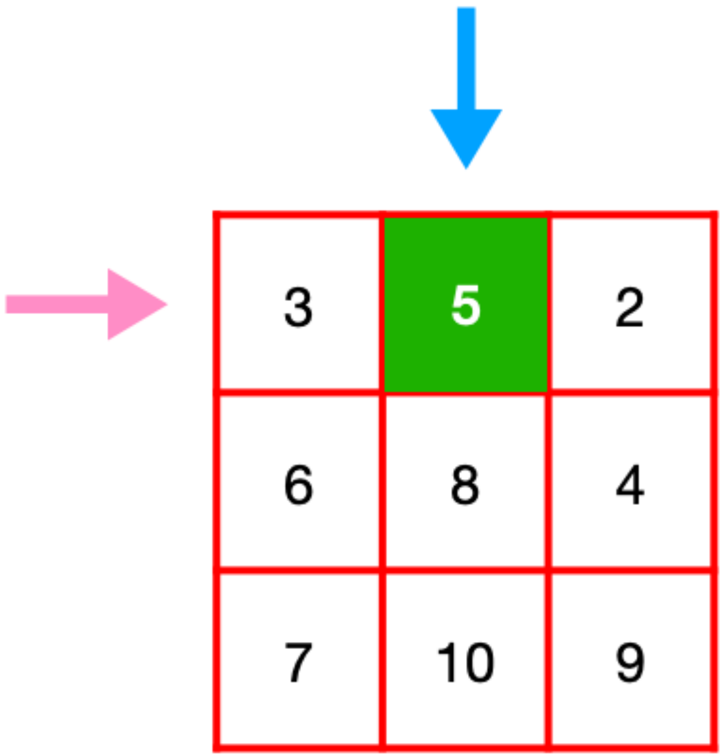
Two-Step Game

- Two step game: 首先，**Bob**选择第 j 列，然后，**Alice**选择第 i 行
- 结果：**Alice**输（**Bob**赢）第 i 行第 j 列的元素值
- A MaxMin Game

3	5	2
6	8	4
7	10	9

Two-Step Game

- Two step game: 首先，**Bob**选择第 j 列，然后，**Alice**选择第 i 行
- 结果：**Alice**输（**Bob**赢）第 i 行第 j 列的元素值
- A MaxMin Game



Two-Step Game

- Two step game: 首先，**Bob**选择第 j 列，然后，**Alice**选择第 i 行
- 结果：**Alice**输（**Bob**赢）第 i 行第 j 列的元素值
- A MaxMin Game

3	5	2
6	8	4
7	10	9

$$U(i^*, j^*) = \max_j \min_i U(i, j)$$

Quiz : MinMax = MaxMin?

3	5	2
6	8	4
7	10	9

Theorem:

$$\max_j \min_i U(i,j) \leq \min_i \max_j U(i,j)$$

确定性策略和非确定性策略

	0.1	0.4	0.5
0.3	3	5	2
0.3	6	8	4
0.4	7	10	9

Extra： 确定性策略和非确定性策略

	0.1	0.4	0.5
0.3	3	5	2
0.3	6	8	4
0.4	7	10	9

Von Neumann's Minimax Theorem:

$$\min_{P_i} \max_{P_j} \mathbb{E}_{P_i, P_j}\left[U(i,j)\right] = \max_{P_j} \min_{P_i} \mathbb{E}_{P_i, P_j}\left[U(i,j)\right]$$

Extra：纳什均衡（Nash Equilibrium）

$$\min_{P_i}\max_{P_j}\mathbb{E}_{P_i,P_j}\left[U(i,j)\right]=\max_{P_j}\min_{P_i}\mathbb{E}_{P_i,P_j}\left[U(i,j)\right]$$

在纳什均衡条件下，任何一个玩家更改策略都不会使结果更好

注意，纳什均衡点并不代表最优解。

Extra： 囚徒困境

两个嫌疑犯作案后被警察抓住，分别关在不同的屋子里接受审讯。警察知道两人有罪，但缺乏足够的证据。警察告诉每个人：

如果两人都抵赖，各判刑 1年；

如果两人都坦白，各判 3年；

如果两人中一个坦白而另一个抵赖，坦白的放出去，抵赖的判 5年

		A	
		认罪	不认罪
B	认罪	(3，3)	0，5
	不认罪	5，0	1，1

提纲

- 对抗博弈
  - 双人零和博弈
- 确定性搜索
  - 最大最小搜索
  - Alpha-beta 剪枝
- 基于模拟的搜索
  - 蒙特卡洛树搜索

单一Agent搜索树

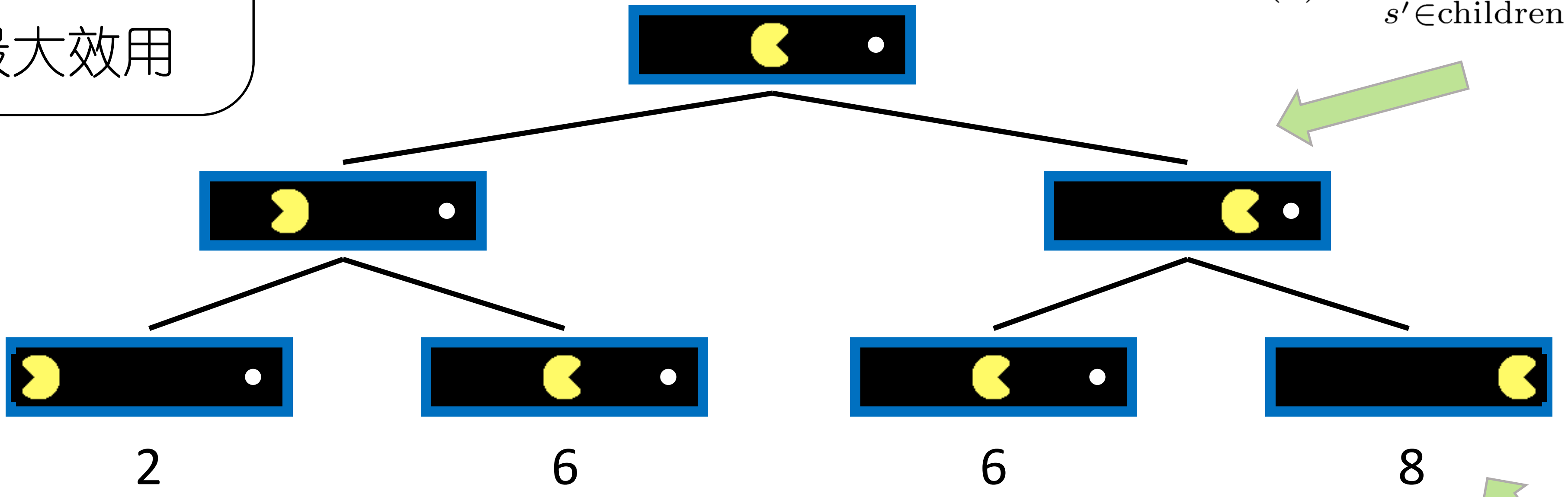
状态的价值 V

从当前状态出发能

获得的最大效用

Non-Terminal States:

$$V(s) = \max_{s' \in \text{children}(s)} V(s')$$



Terminal States:

$$V(s) = \text{known}$$

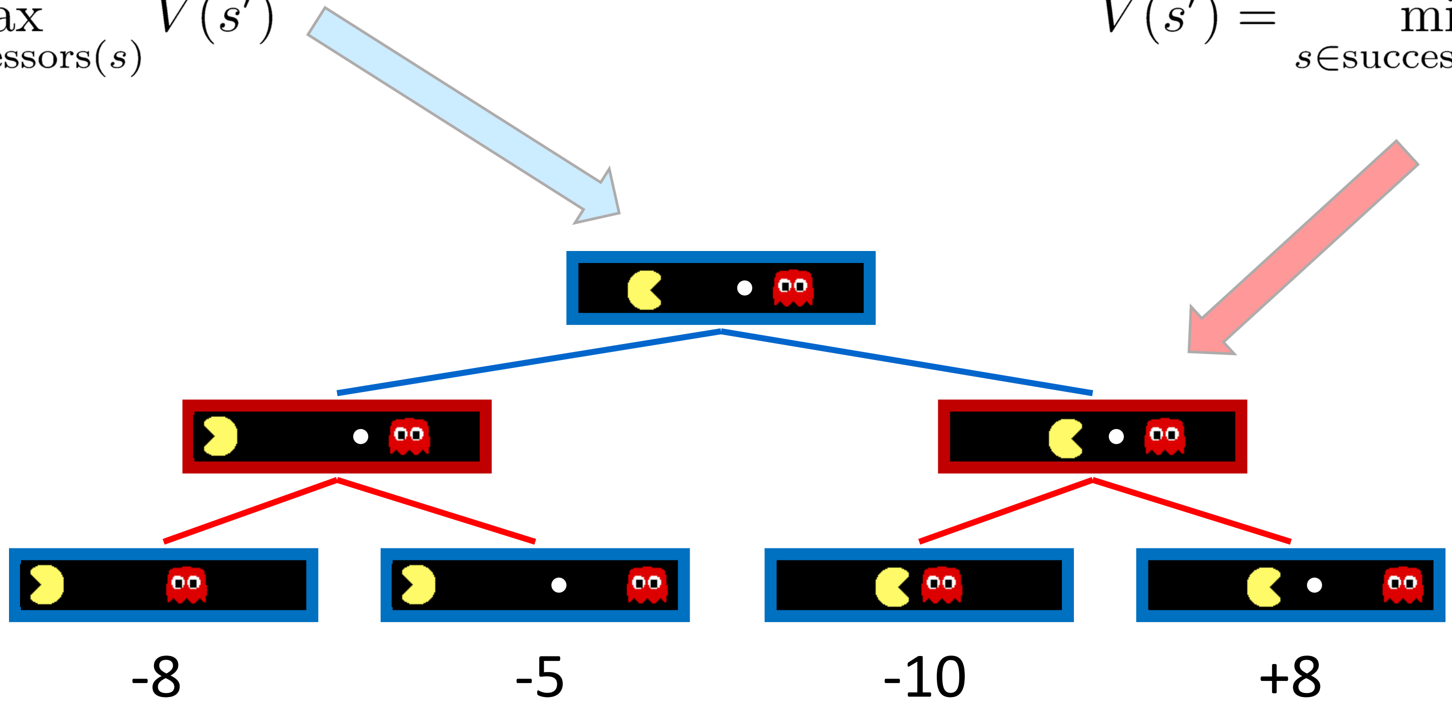
博弈搜索树

States Under Agent’s Control:

$$V(s) = \max_{s' \in \text{successors}(s)} V(s')$$

States Under Opponent’s Control:

$$V(s') = \min_{s \in \text{successors}(s')} V(s)$$

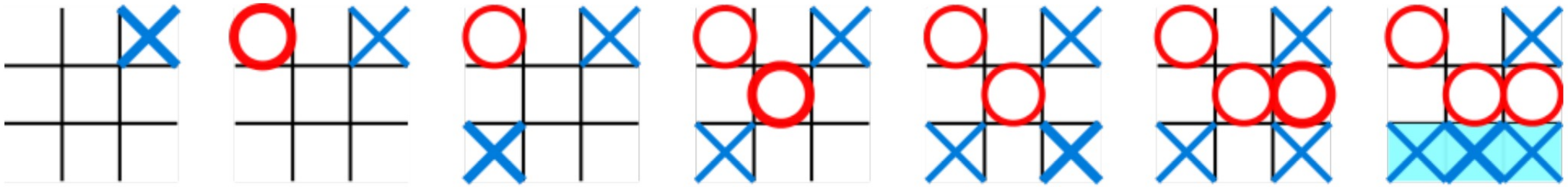


Terminal States:

$$V(s) = \text{known}$$

多步搜索

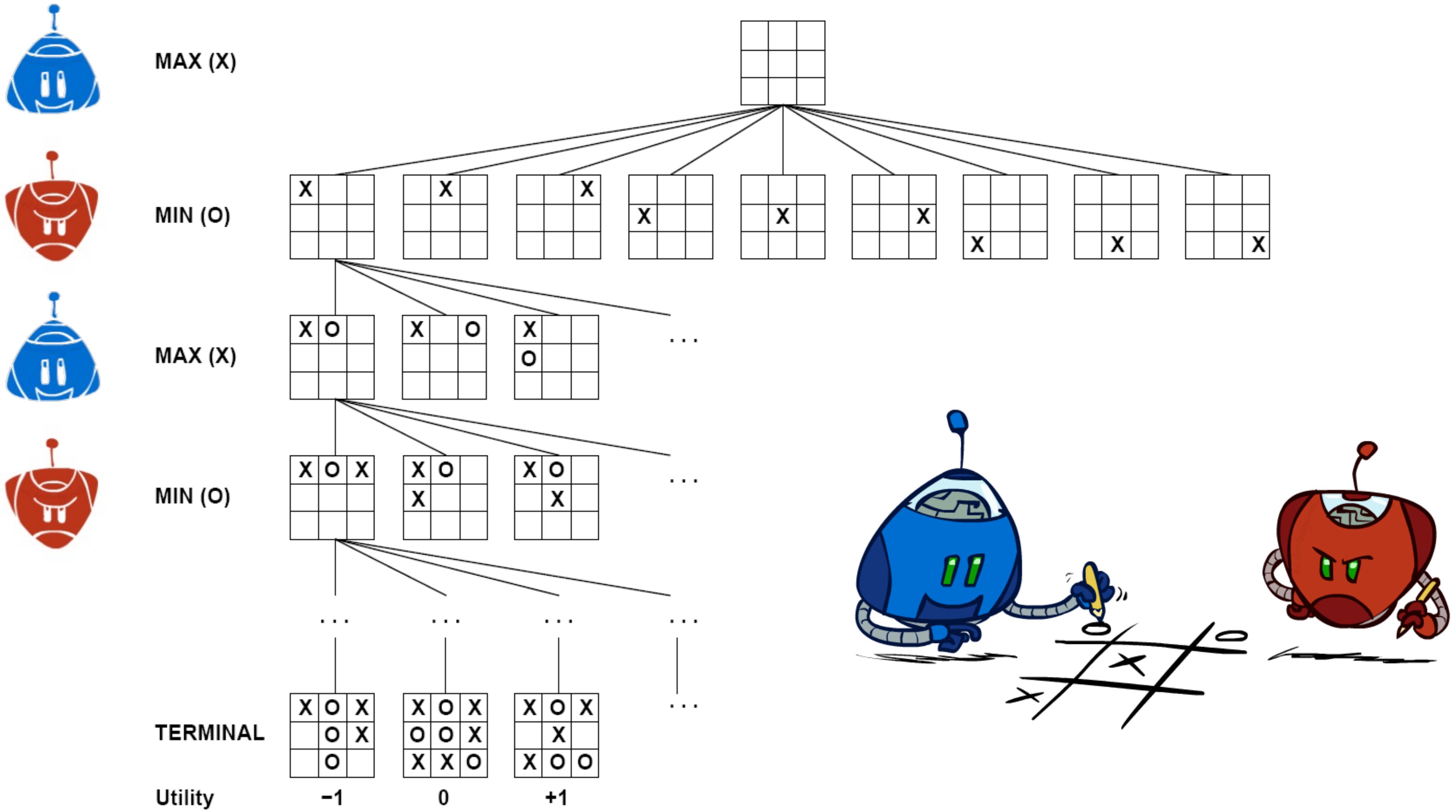
两人轮流在一有九格方盘上划加字或圆圈，谁先把三个同一记号排成横线、直线、斜线，即是胜者



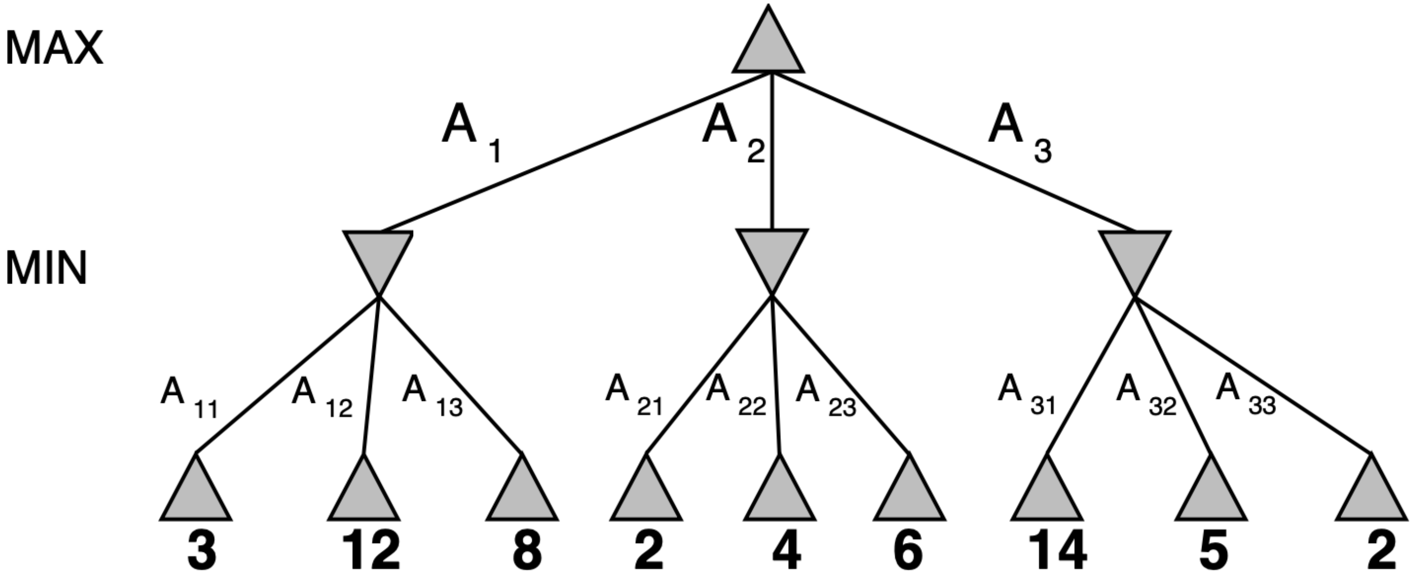
问题定义

- **状态**：状态s包括当前的游戏局面和当前行动的玩家
- **动作**：给定状态s，动作指的是$player(s)$在当前局面下可以采取的操作a，记动作集合为$actions(s)$
- **状态转移**：给定状态s和动作$a \in actions(s)$，状态转移函数$result(s,a)$决定了在s状态采取a动作后所得后继状态
- **终局状态检测**：终止状态检测函数$terminal_test(s)$用于测试游戏是否在状态s结束
- **终局得分**：终局得分$utility(s,p)$表示在终局状态s时玩家p的得分

搜索树

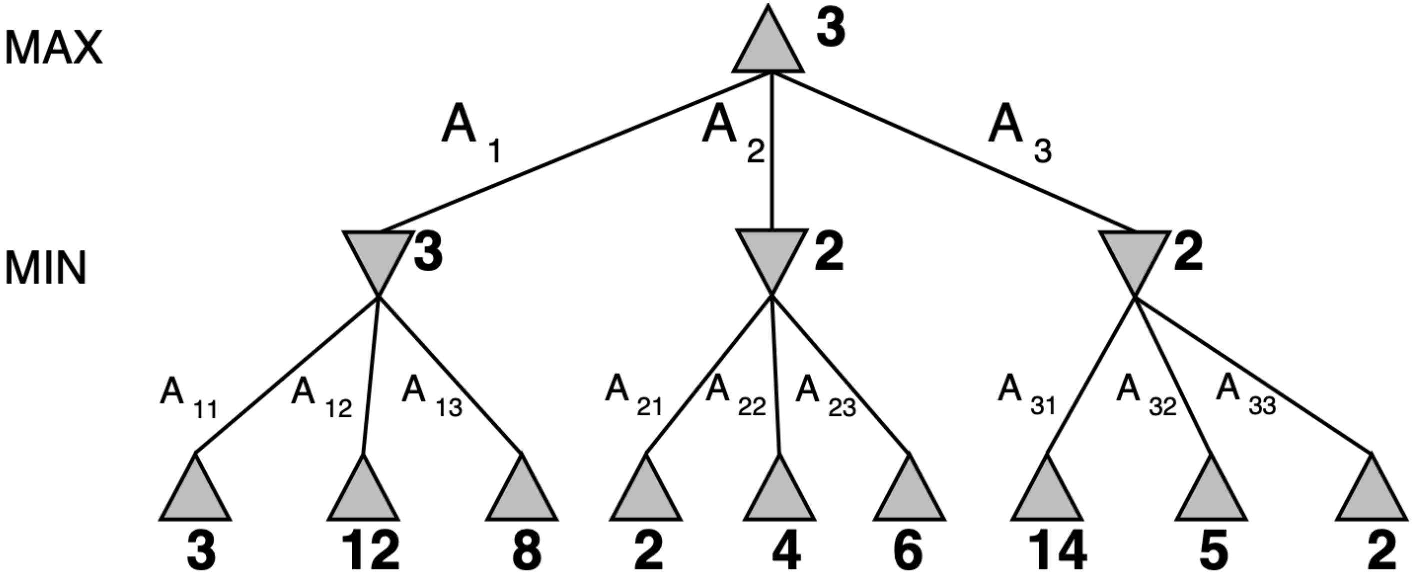


最优策略



MAX
MIN
A 1
A 2
A 3
A 11
A 12
A 13
A 21
A 22
A 23
A 31
A 32
A 33
3
12
8
2
4
6
14
5
2

最优策略



给定一棵博弈树，最优策略可以通过检查每个节点的极小极大值来决定，记为minimax(n)

$$\text{minimax(s)} = \begin{cases} \text{utility}(s), & \text{if terminal\_test}(s) \\ \max_{a \in \text{actions}(s)} \text{minimax}\big(\text{result}(s,a)\big), & \text{if player}(s) = MAX \\ \min_{a \in \text{actions(s)}} \text{minimax}\big(\text{result}(s,a)\big), & \text{if player}(s) = MIN \end{cases}$$

最小最大搜索

```
def max-value(state):
    initialize v = -∞
    for each successor of state:
        v = max(v, min-value(successor))
    return v
```

```
def min-value(state):
    initialize v = +∞
    for each successor of state:
        v = min(v, max-value(successor))
    return v
```

$$V(s) = \max_{s' \in \text{successors}(s)} V(s')$$

$$V(s') = \min_{s \in \text{successors}(s')} V(s)$$

最小最大搜索

```
def value(state):
    if the state is a terminal state: return the state's utility
    if the next agent is MAX: return max-value(state)
    if the next agent is MIN: return min-value(state)
```

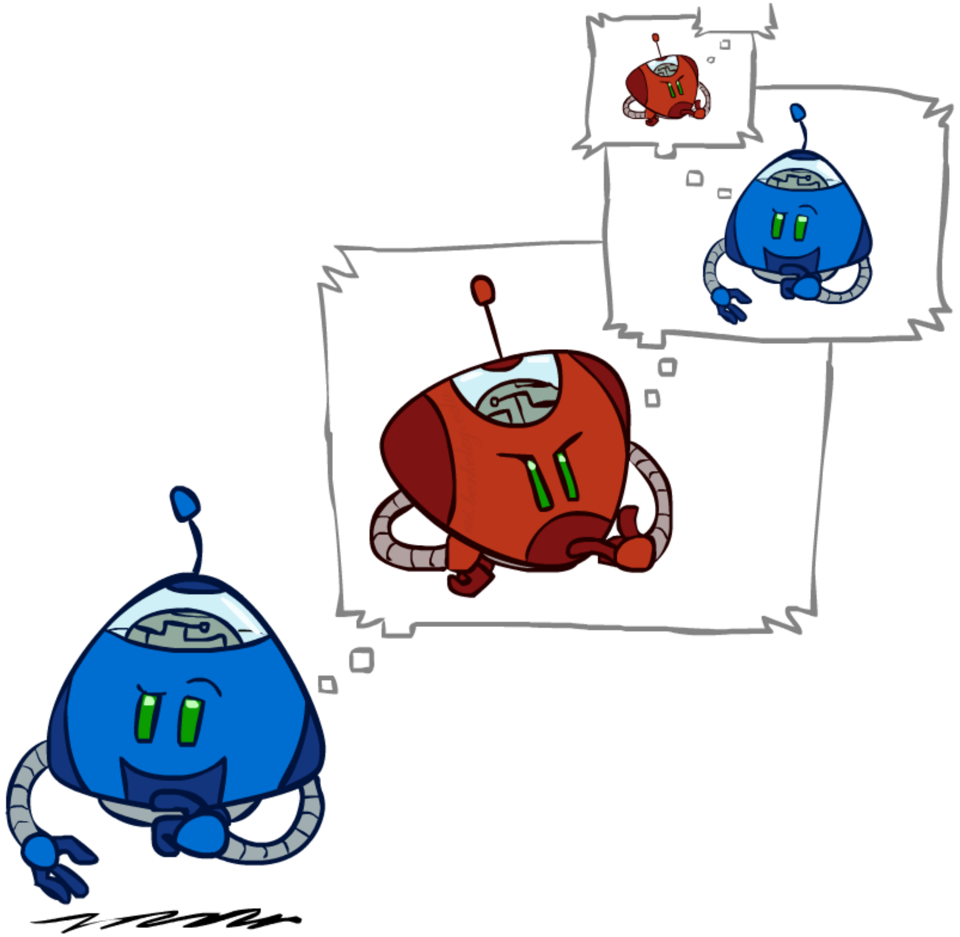
```
def max-value(state):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor))
    return v
```

```
def min-value(state):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor))
    return v
```

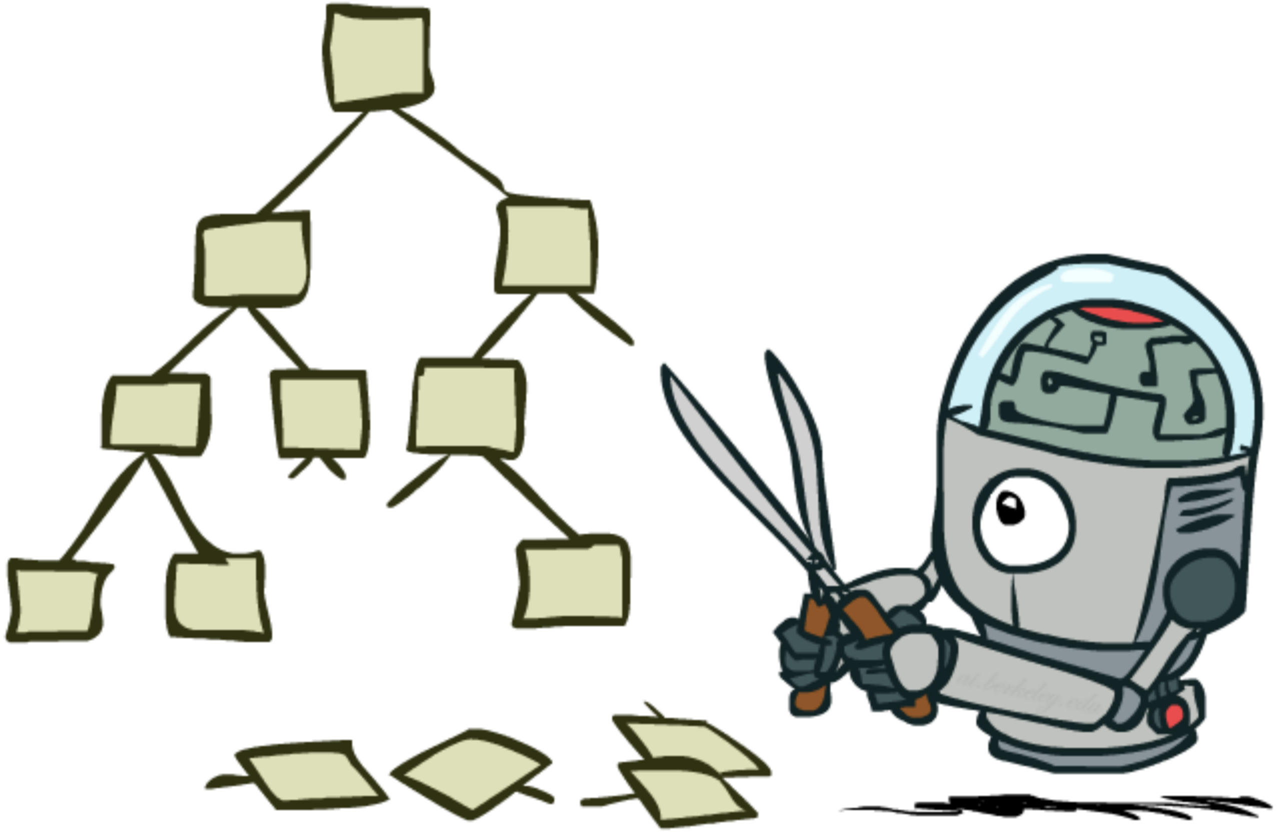
性能分析

- 时间和空间?
  - 和 DFS 类似
  - 时间复杂度: $O(b^m)$
  - 空间复杂度: $O(bm)$

- Example: For chess, $b \approx 35$, $m \approx 100$
  - 精确的搜索几乎是不可行的



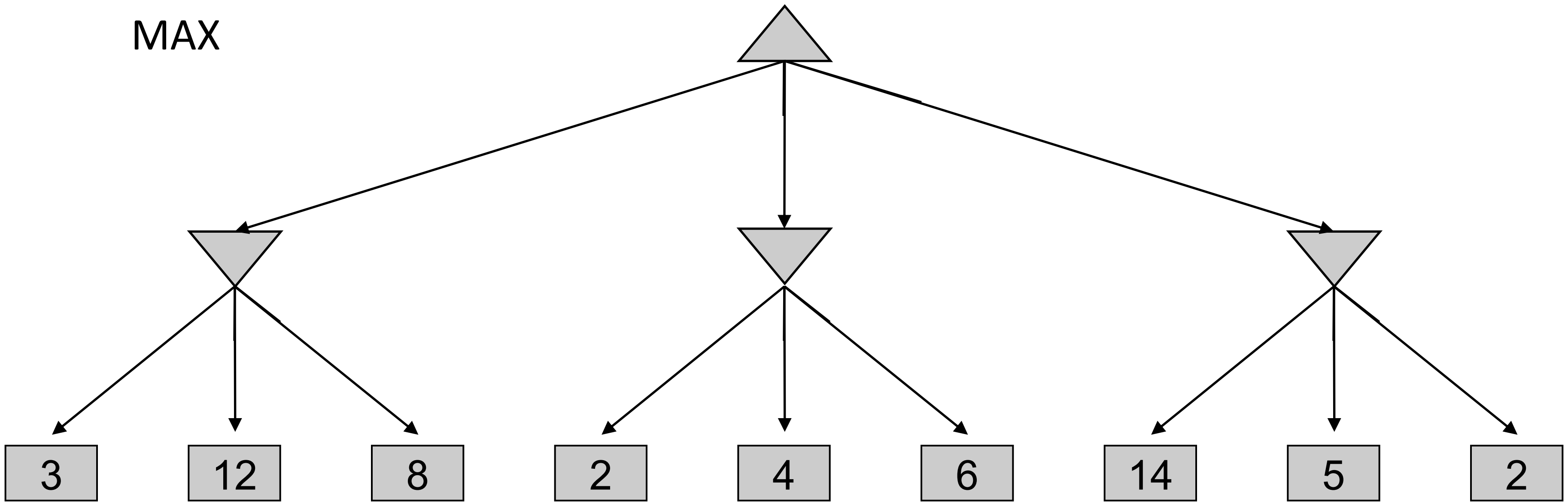
树剪枝



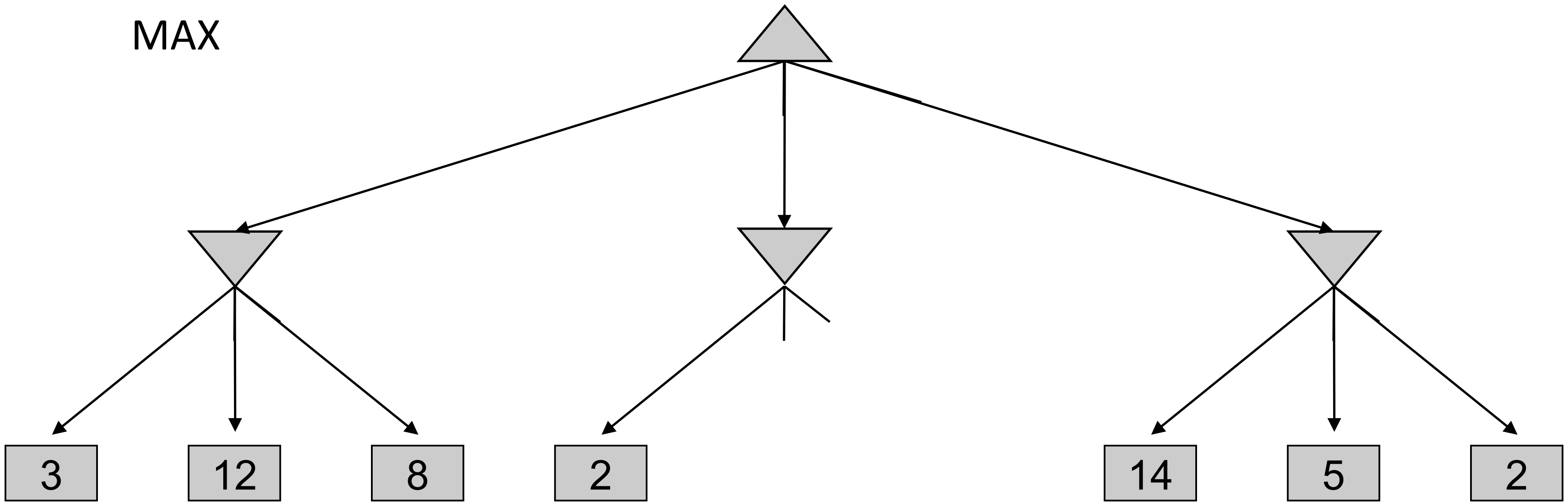
提纲

- 对抗博弈
  - 双人零和博弈
- 确定性搜索
  - 最大最小搜索
  - Alpha-beta 剪枝
- 基于模拟的搜索
  - 蒙特卡洛树搜索

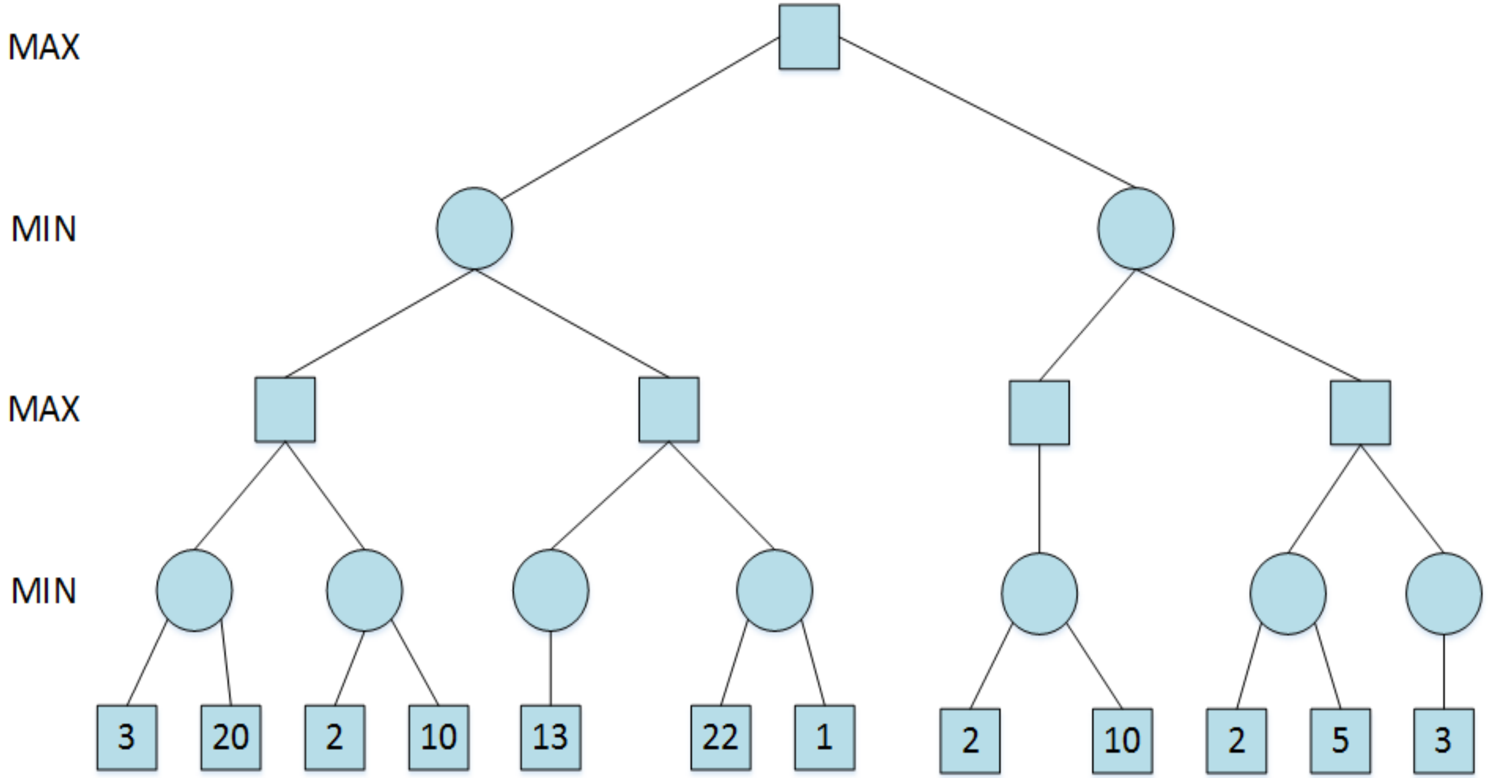
树剪枝



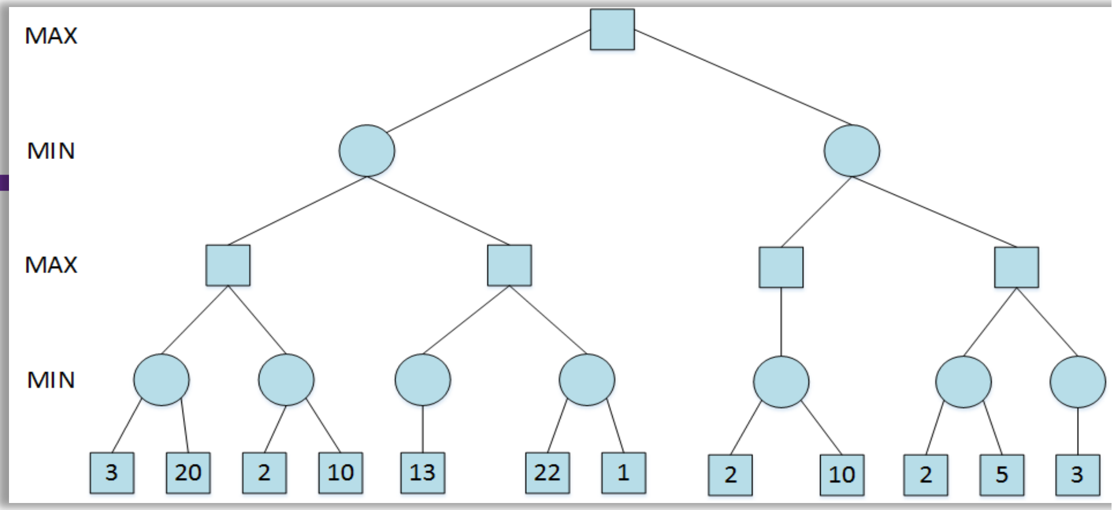
树剪枝

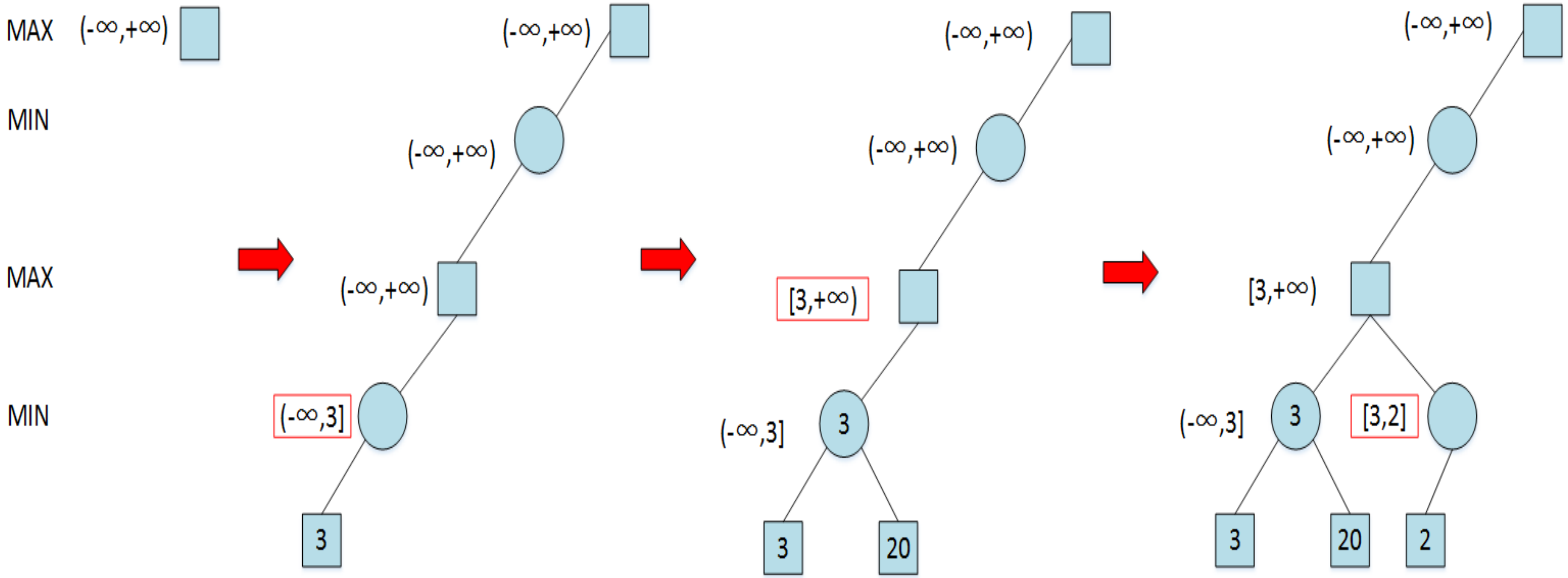


Alpha-Beta剪枝

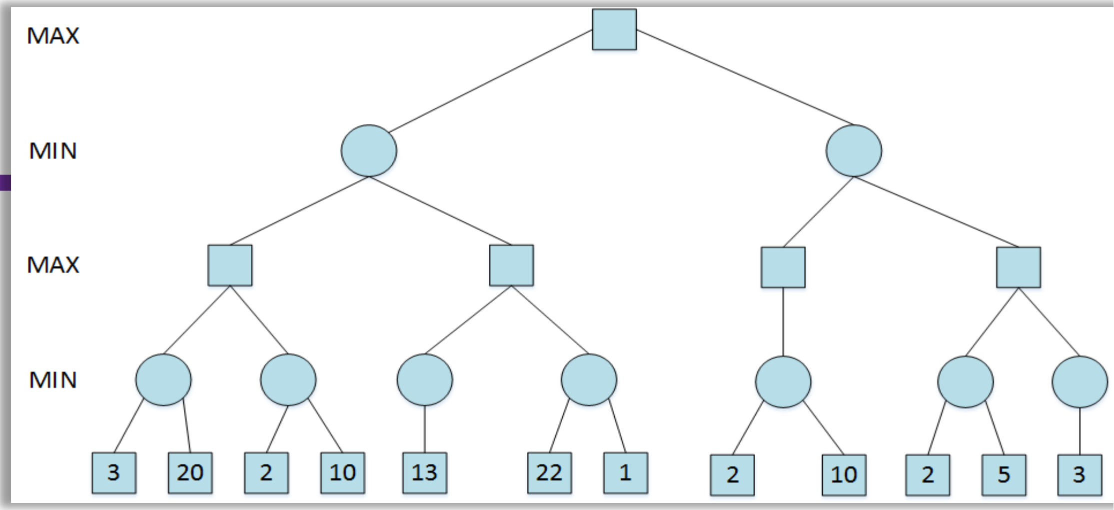


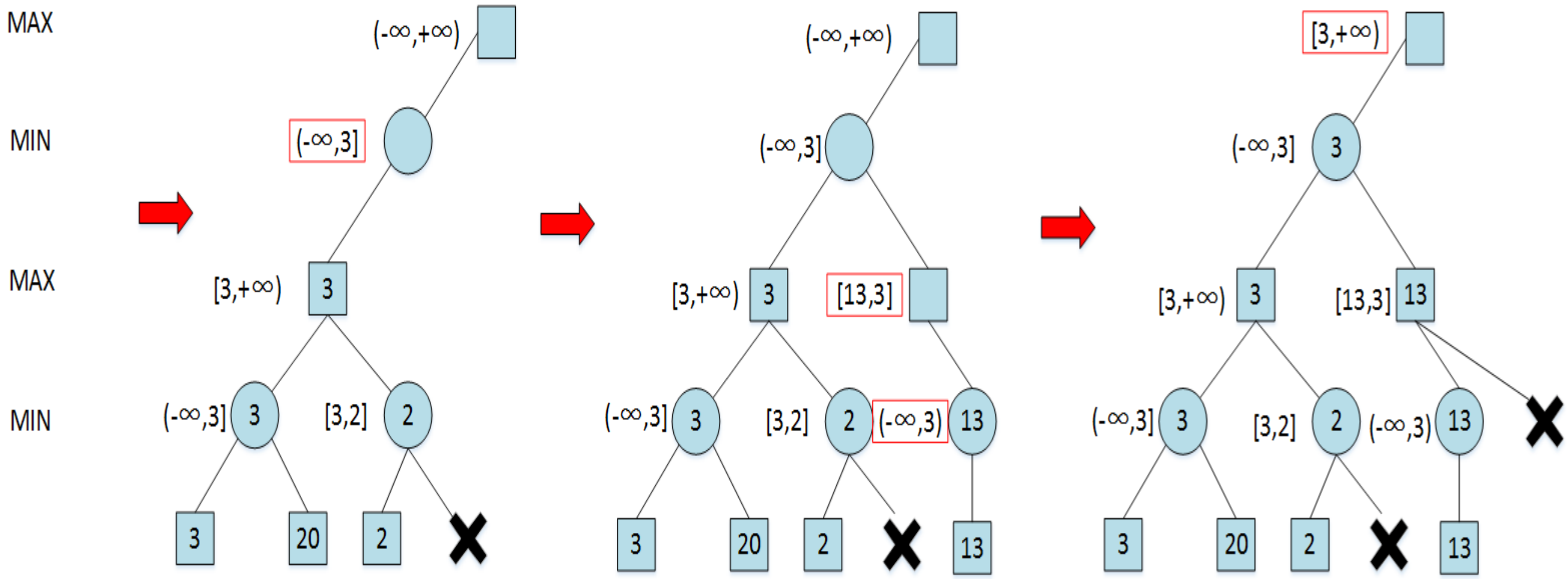
Alpha-Beta剪枝



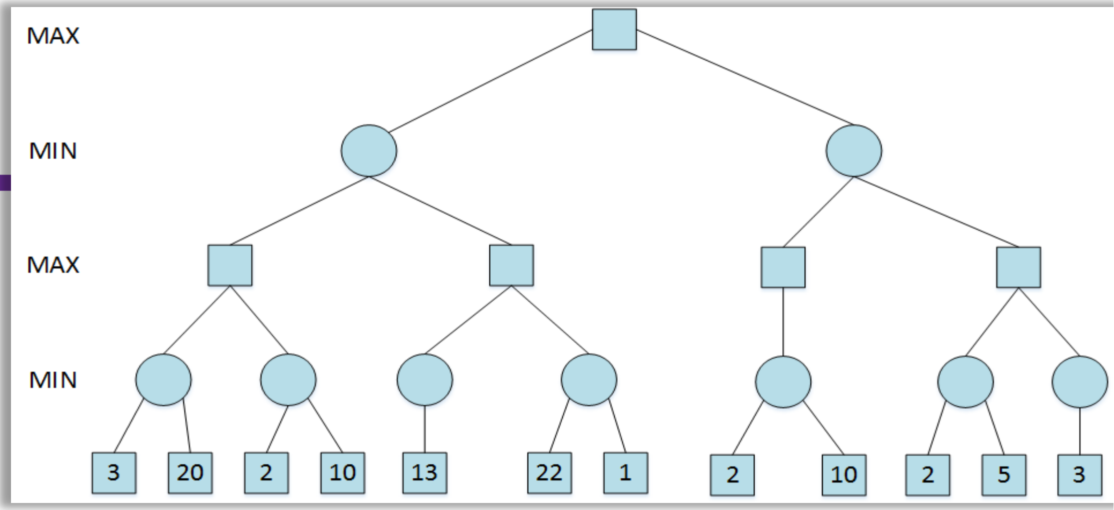


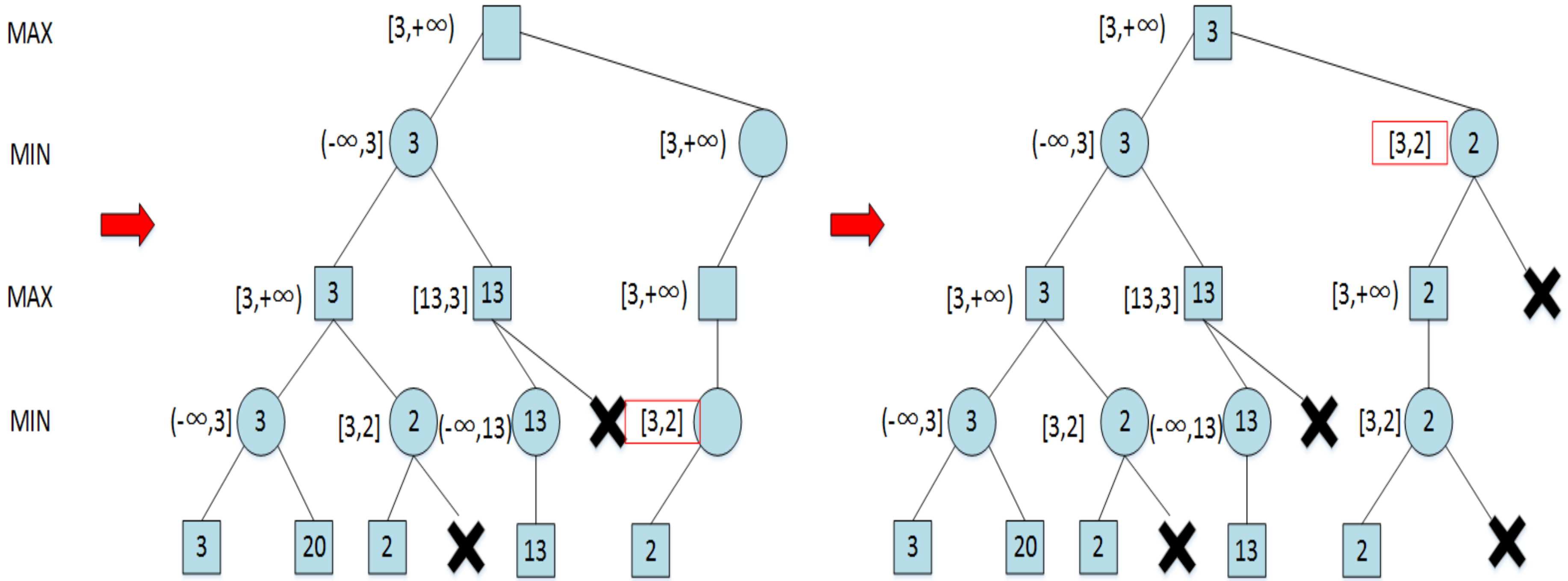
Alpha-Beta剪枝



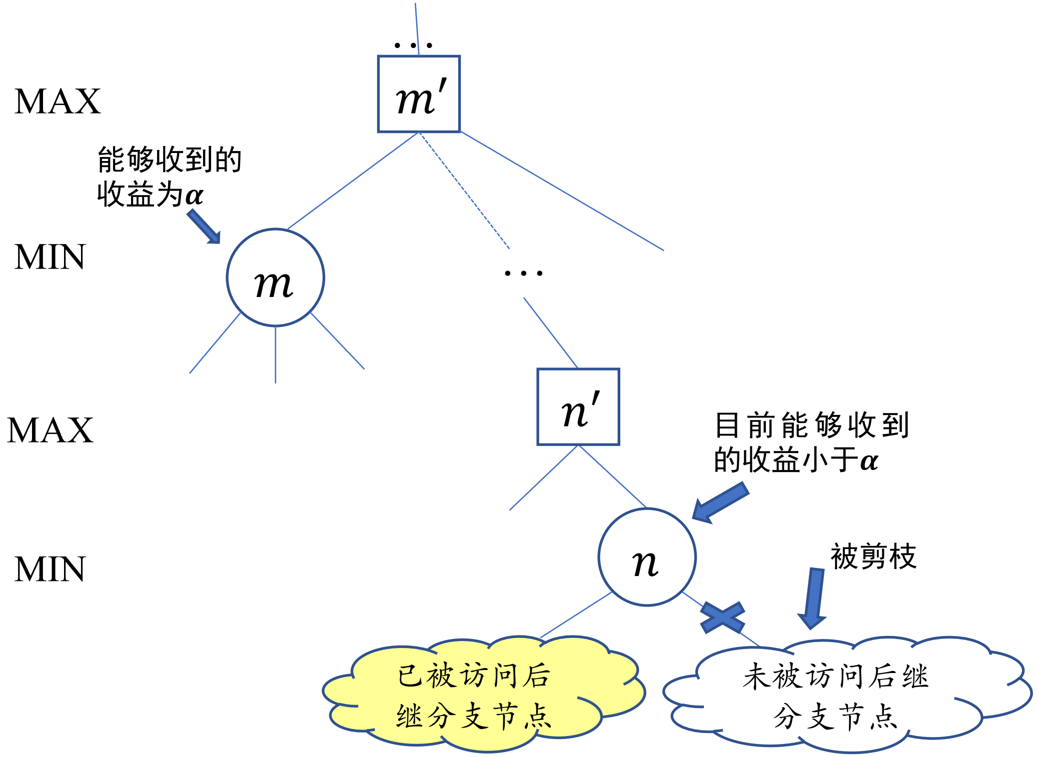


Alpha-Beta剪枝





Alpha-Beta剪枝

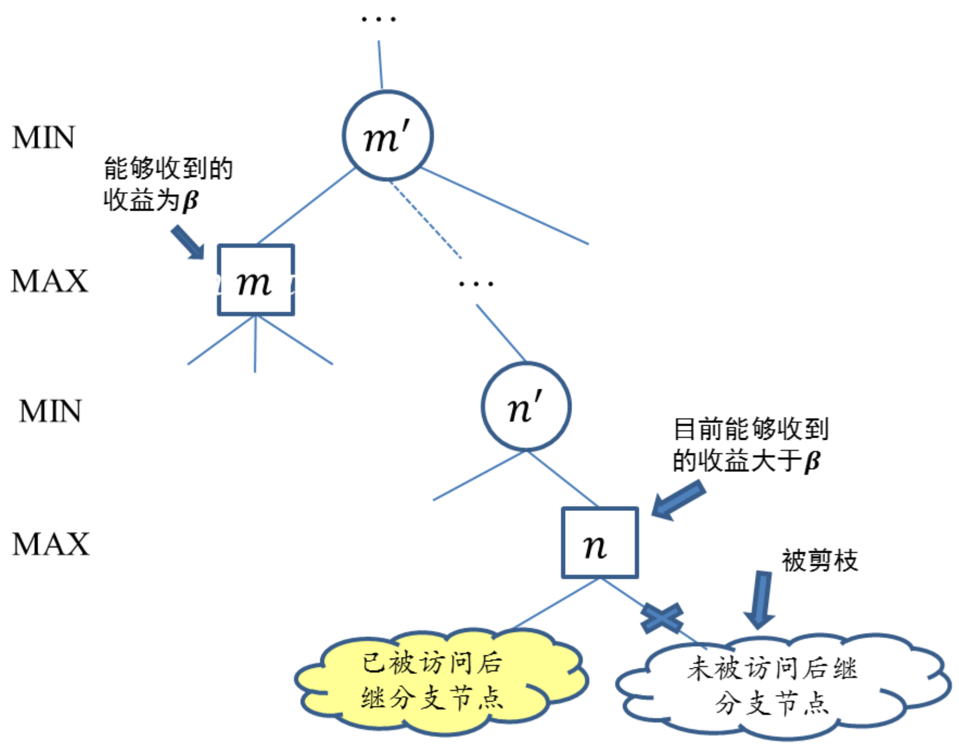


基于MIN结点反馈收益进行剪枝（alpha剪枝）

α：目前为止路径上发现的 MAX的最佳（即极大值）选择

假设有一个位于MIN层的结点m，已知该结点能够向其上MAX结点反馈的收益为α（alpha）。n是与结点m位于同一层的某个兄弟（sibling）结点的后代结点。如果在结点 n 的后代结点被访问一部分后，知道结点n能够向其上一层MAX结点反馈收益小于α，则结点n的未被访问孩子结点将被剪枝。

Alpha-Beta剪枝



基于MAX结点反馈收益进行剪枝（beta剪枝）

β：目前为止路径上发现的MIN的最佳（即极小值）选择

考虑位于MAX层的结点m，已知结点m能够从其下MIN层结点收到的收益为β，结点n是结点m上层结点m'的位于MAX层的后代结点，如果目前已知结点n能够收到的收益大于β，则不再扩展结点n的未被访问后继结点，因为位于MIN层的结点m'只会选择收益小于或等于β的结点来采取行动。

Alpha-Beta剪枝

- 对于MAX节点，如果其孩子结点（MIN结点）的收益大于当前的α值，则将α值更新为该收益；对于MIN结点，如果其孩子结点（MAX结点）的收益小于当前的β值，则将β值更新为该收益。根结点（MAX结点）的α值和β值分别被初始化为$-\infty$和$+\infty$

- 随着搜索算法不断被执行，每个结点的α值和β值不断被更新。大体来说，每个结点的$[\alpha, \beta]$从其父结点提供的初始值开始，取值按照如下形式变化：α逐渐增加、β逐渐减少。不难验证，如果一个结点的α值和β值满足$\alpha > \beta$的条件，则该结点尚未被访问的后续结点就会被剪枝，因而不会被智能体访问

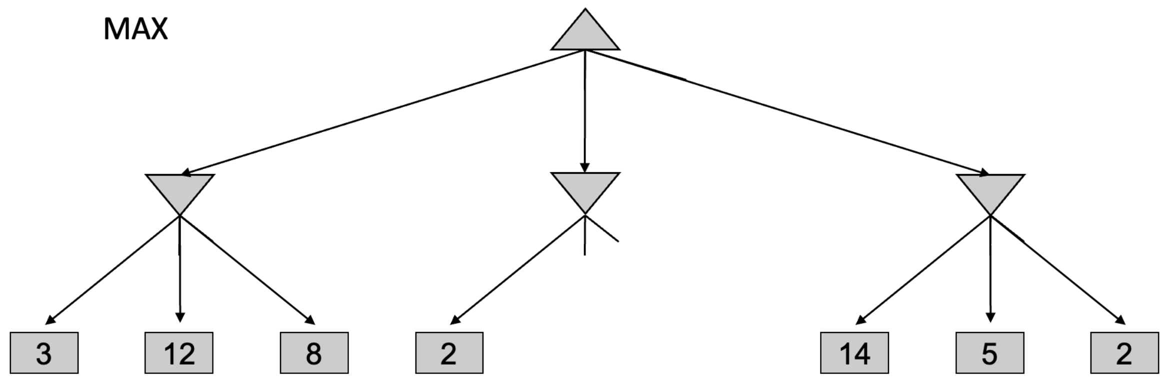
Alpha-Beta剪枝

α:目前为止路上发现的 MAX的最佳（即极大值）选择
β:目前为止路径上发现的MIN的最佳（即极小值）选择

```
def max-value(state, α, β):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor, α, β))
        if v ≥ β return v
        α = max(α, v)
    return v
```

```
def min-value(state, α, β):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor, α, β))
        if v ≤ α return v
        β = min(β, v)
    return v
```

搜索顺序很重要！

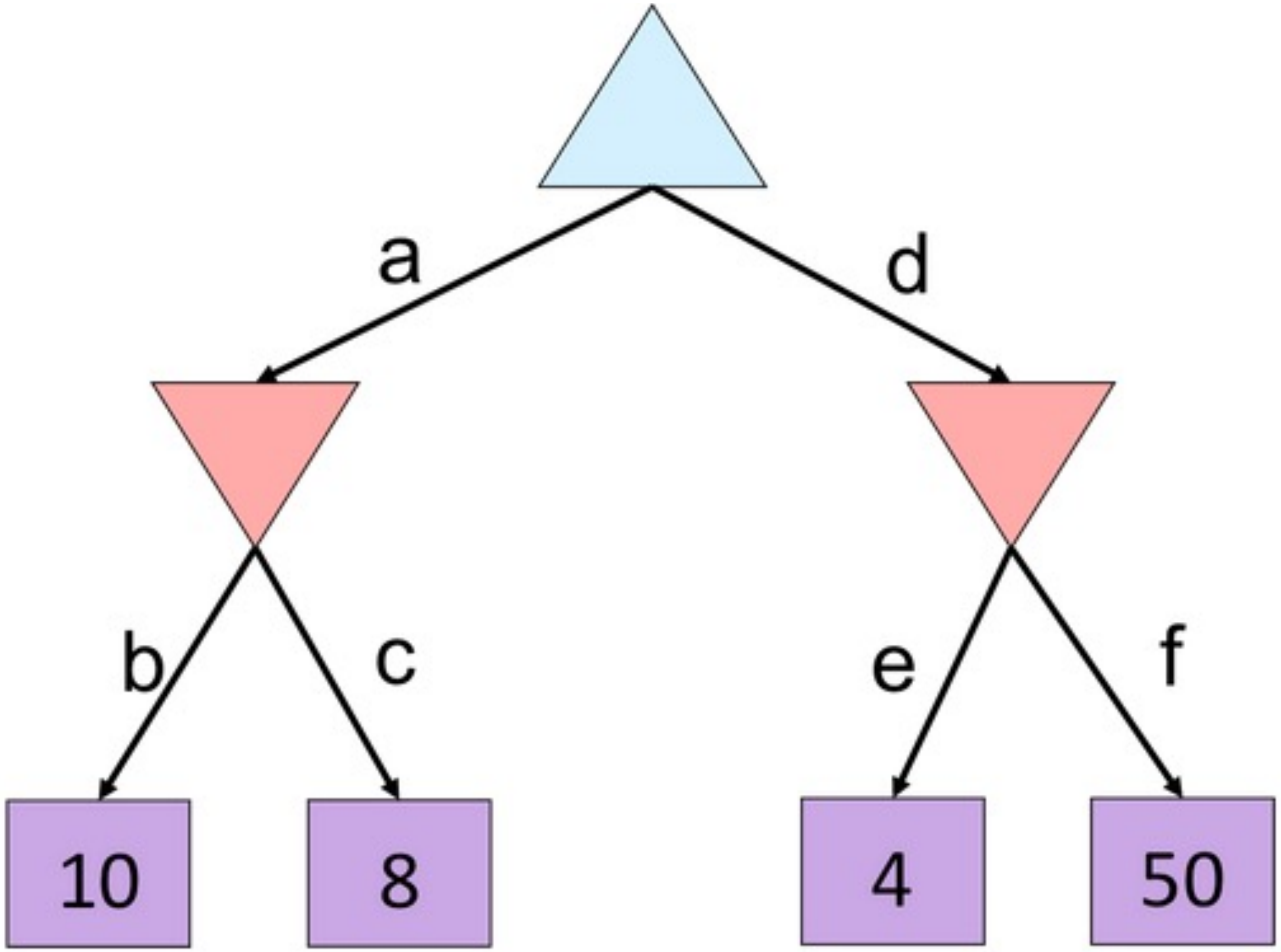


启发？

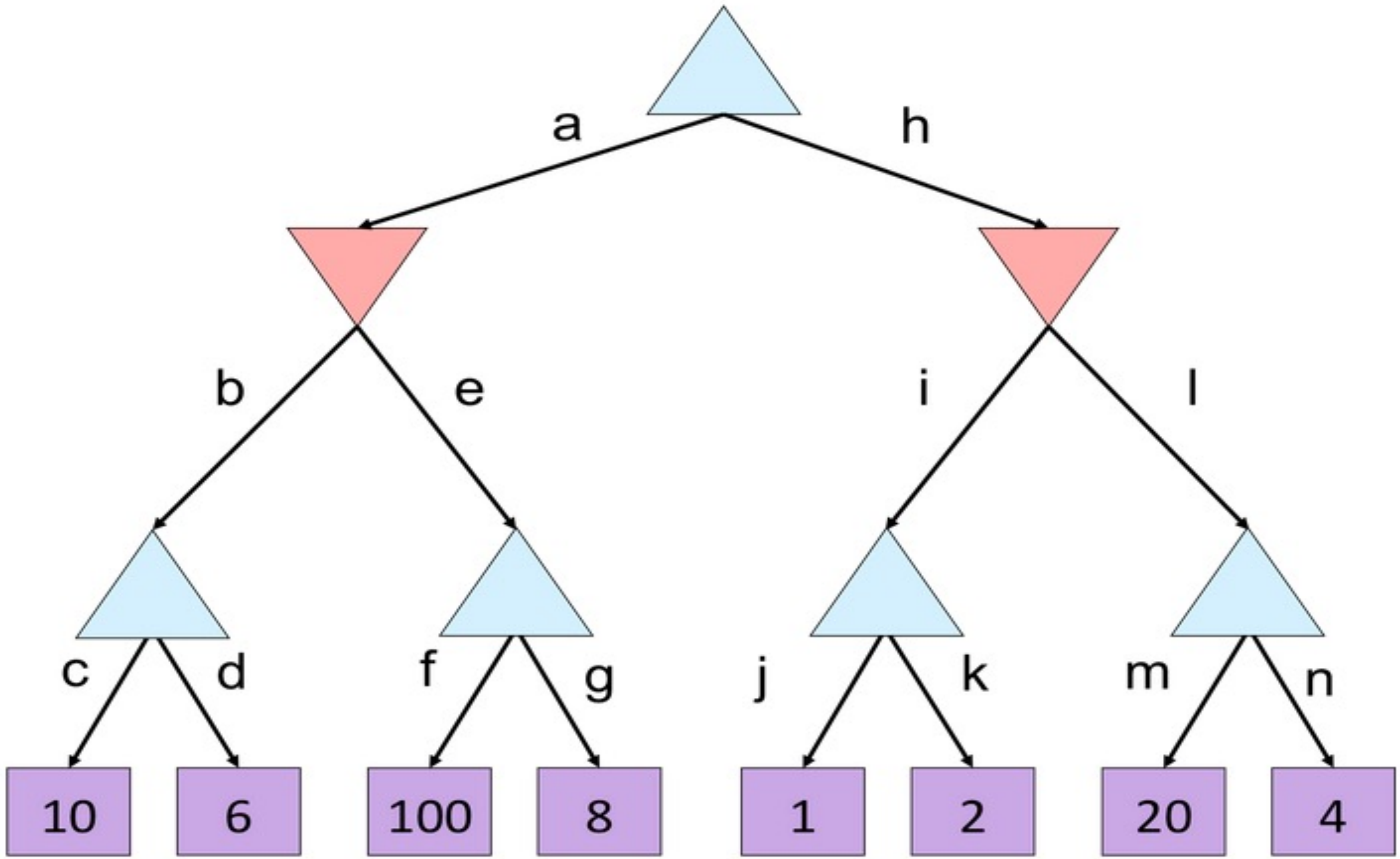
可以设计方案对后继状态进行排序，

例如，对于国际象棋，可以设计排序规则：吃子>威胁>前进>后退

Quiz1: Alpha-Beta剪枝



Quiz2: Alpha-Beta剪枝

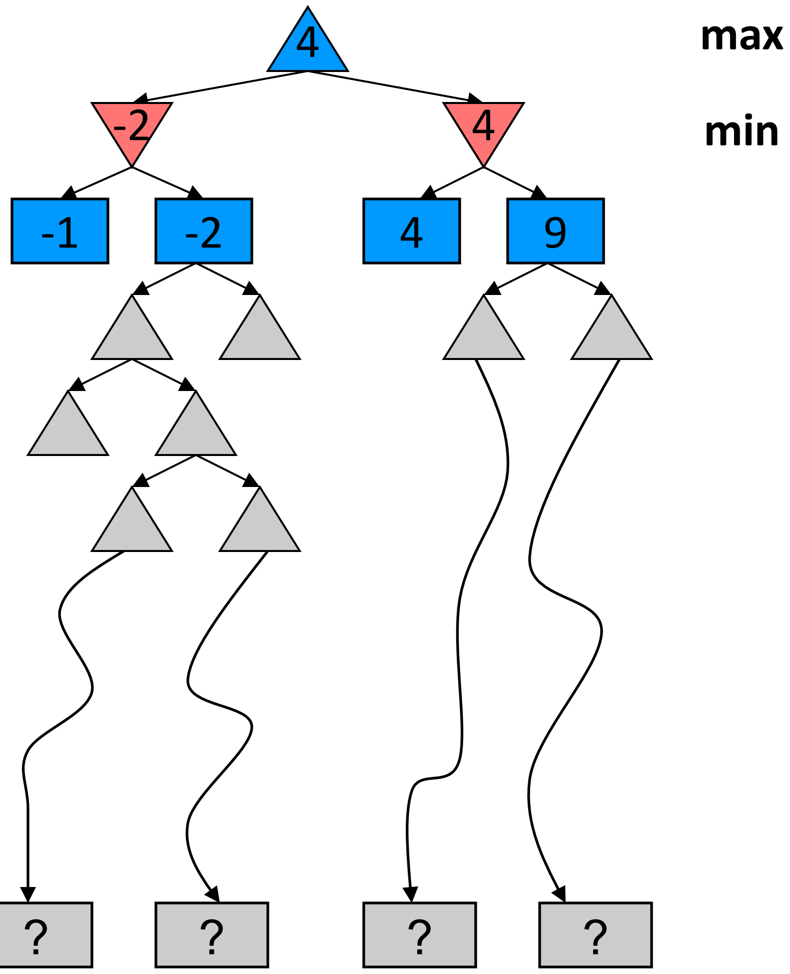


资源受限

尽管alpha-beta剪枝能够避免搜索完整的空间，但是仍然要**搜索部分空间直至终止状态**，这样的搜索深度也是不现实的

一个可行的思路：

参考启发式搜索，设计评估函数用于搜索中的状态，有效地把非终止节点变成终止节点



评估函数

大多数评估函数都需要考虑状态的不同特征参数

以象棋为例，要考虑兵的数目、车的数目、马的数目等等

形式化描述：加权线性函数

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

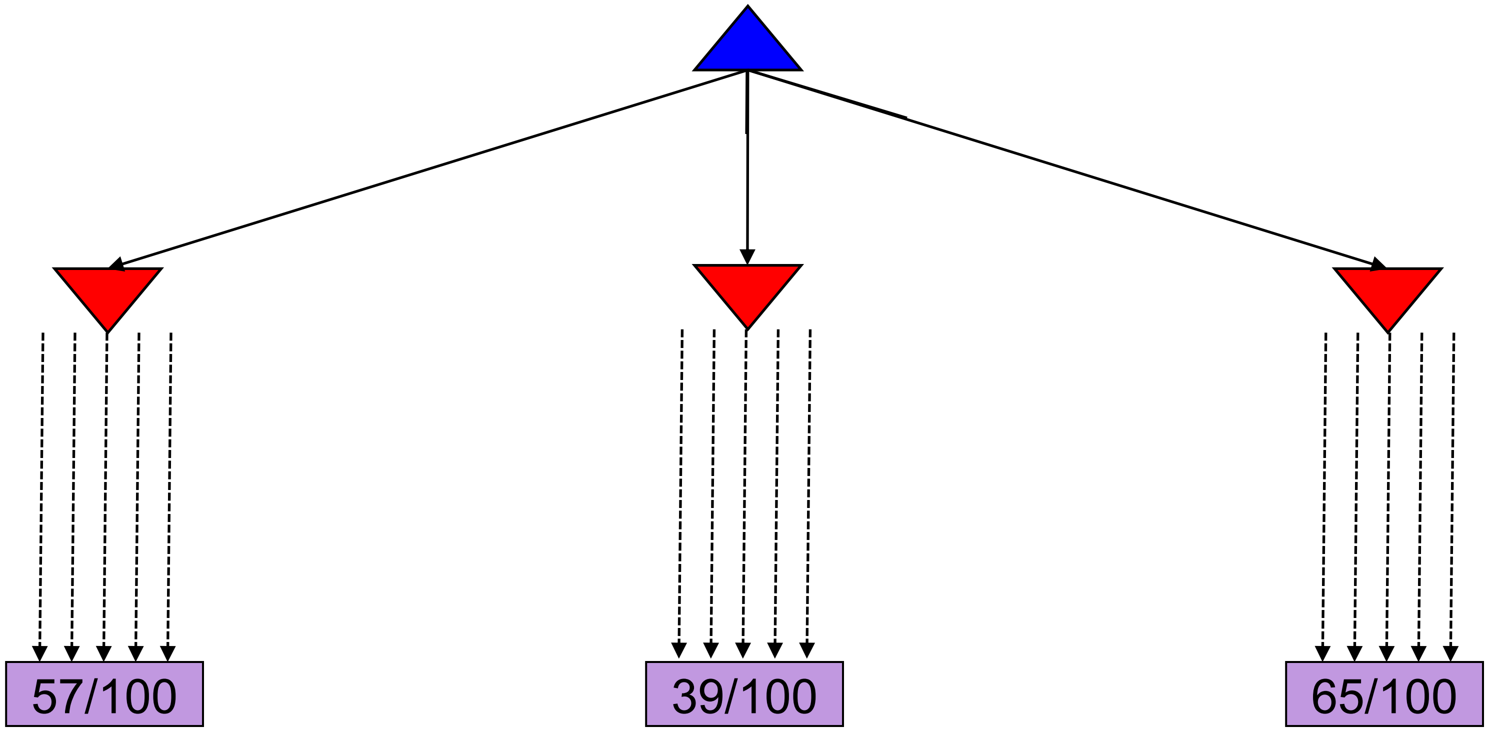
w_i表示权重，f_i是棋局的某个特征

提纲

- 对抗博弈
  - 双人零和博弈
- 确定性搜索
  - 最大最小搜索
  - Alpha-beta 剪枝
- 基于模拟的搜索
  - 蒙特卡洛树搜索

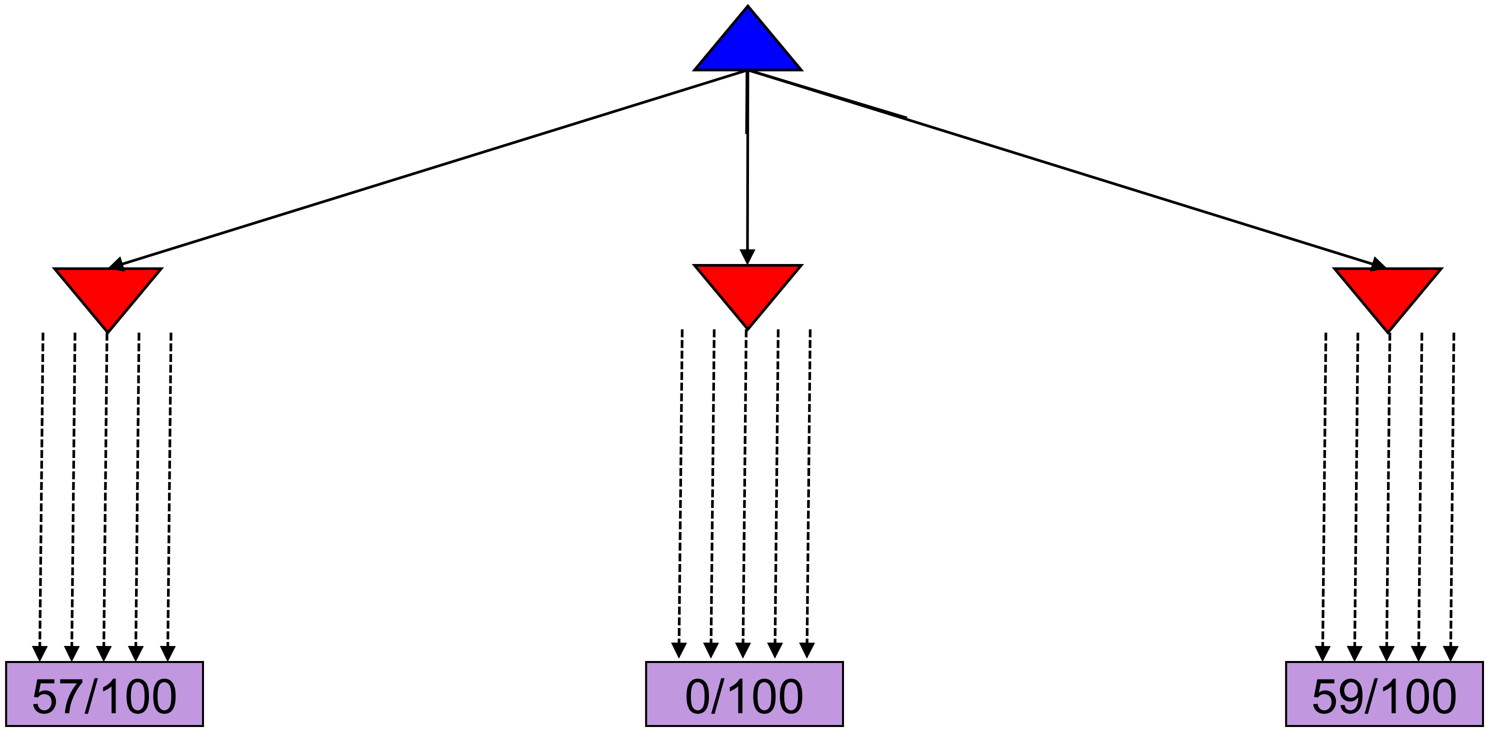
蒙特卡洛搜索

模拟：从一个节点出发，进行大量模拟，记录赢的次数



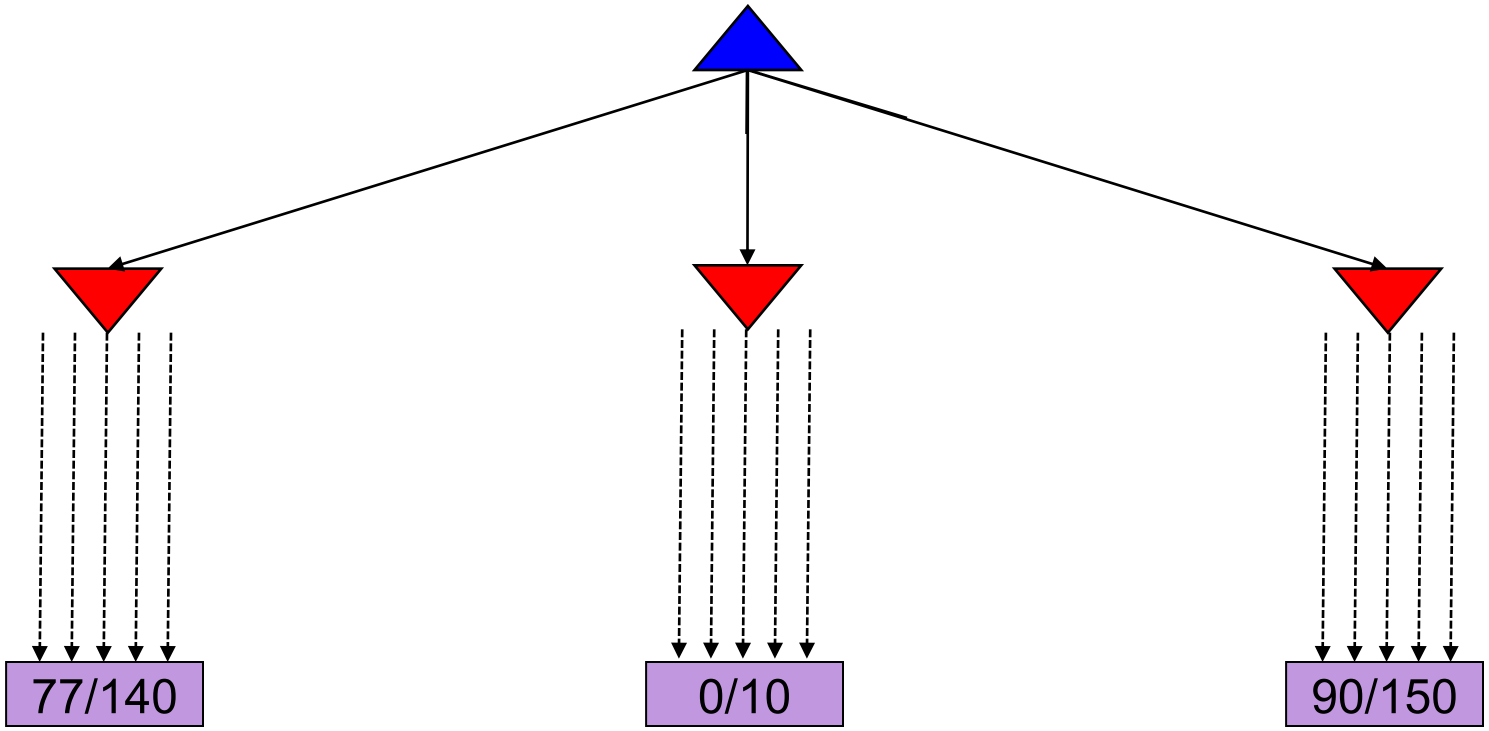
蒙特卡洛搜索

模拟：从一个节点出发，进行大量模拟，记录赢的次数



蒙特卡洛搜索

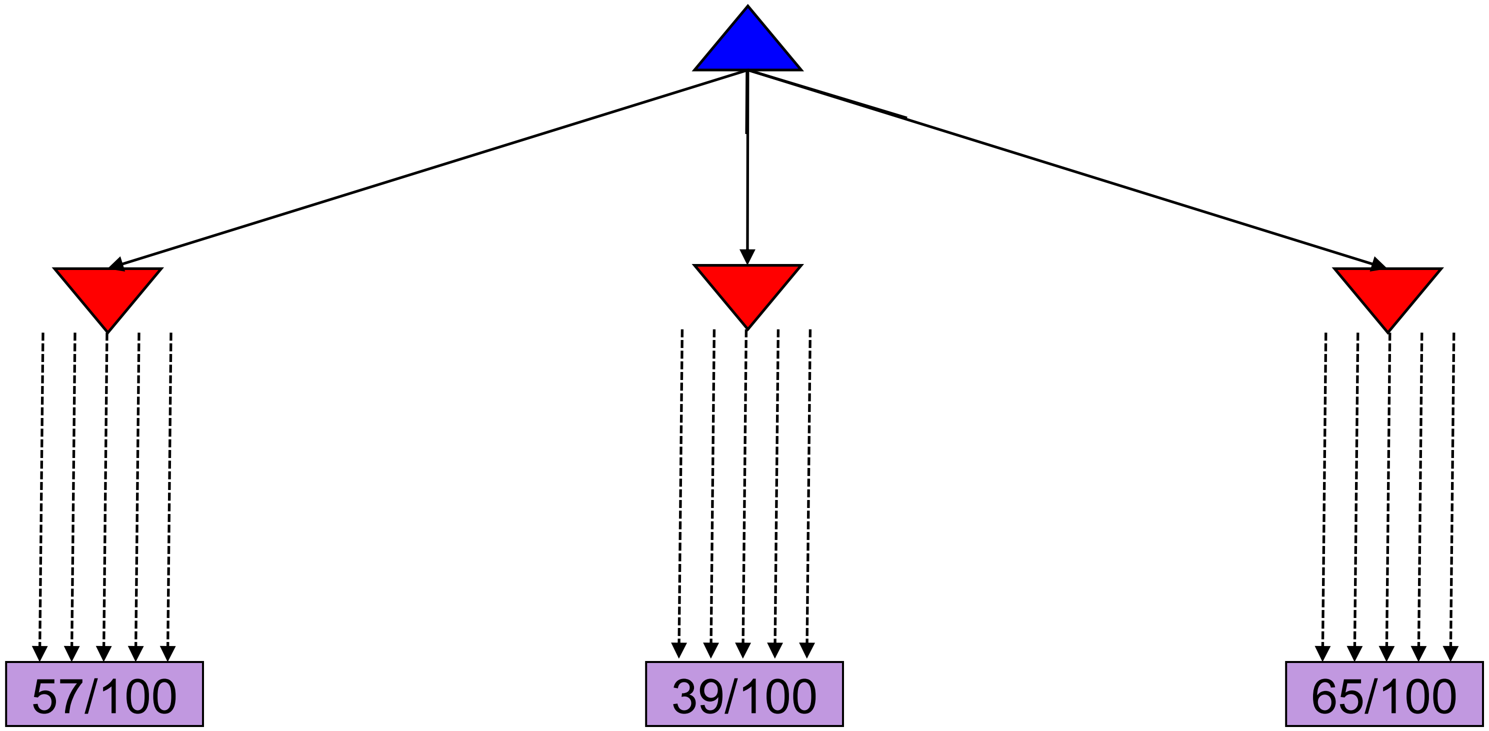
模拟： 从一个节点出发，进行大量模拟，记录赢的次数



蒙特卡洛搜索

资源是有限的，选择哪些节点进行模拟？

选择更有希望的节点进行模拟

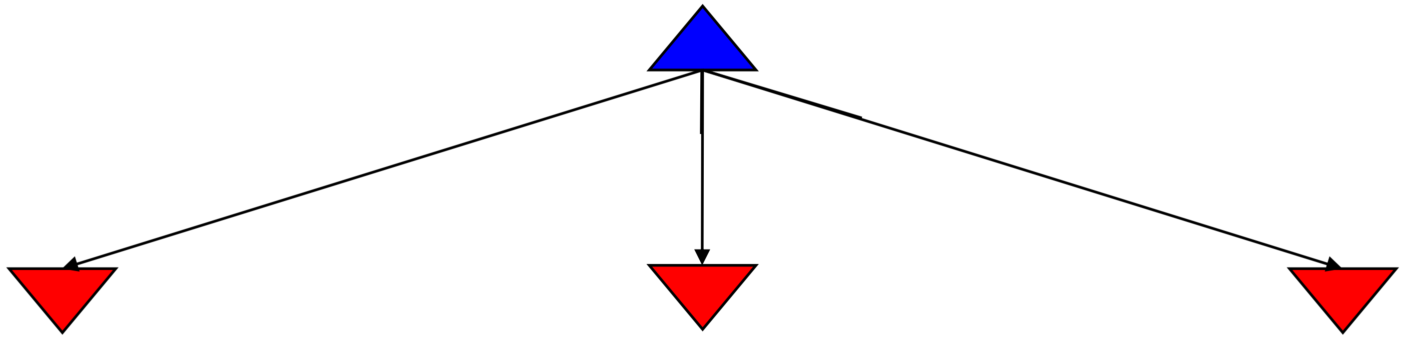


蒙特卡洛搜索

如何评估节点的价值？

大量的模拟

先有鸡还是先有蛋？



多臂老虎机

☐ Multi-Armed Bandits

赌博机有 K 个摇臂，每次转动一个赌博机摇臂，赌博机则会随机吐出一些硬币

如何在有限次数的尝试中使收益最大化？

多臂老虎机

如何在有限次数的尝试中使收益最大化？

- 仅探索 (Exploration-only)

每个摇臂摇动 T / K 次

不足：浪费次数在收益较差的摇臂上

- 仅利用 (Exploitation-only)

1. 每个摇臂摇动一次，记录收益
2. 剩余的T-K次全部用在收益最大的摇臂上

不足：一次估计的结果不可靠

探索利用窘境

Exploration-Exploitation Dilemma

ϵ-贪心算法

ϵ−贪心算法：在探索与利用之间进行平衡的搜索算法

在第t步，ϵ−贪心算法按照如下机制来选择摇动赌博机：

- 以 **$1-\epsilon$ 的概率**，选择在过去$t-1$次摇动赌博机所得**平均收益最高的摇臂**进行摇动；

- 以 **ϵ 的概率**，**随机选择一个摇臂**进行摇动。

不足：没有考虑每个摇臂被探索的次数

输入： 摇臂数 K;
奖赏函数 R;
尝试次数 T;
探索概率 ϵ.

过程:

$r = 0$;
$\forall i = 1, 2, \ldots K: \ Q(i) = 0, \ \mathrm{count}(i) = 0$;
for $t = 1, 2, \ldots, T$ **do**
　if rand()$< \epsilon$ **then**
　　$k =$ 从 $1, 2, \ldots, K$ 中以均匀分布随机选取
　else
　　$k = \arg\max_i Q(i)$
　end if
　$v = R(k)$;
　$r = r + v$;
　$Q(k) = \frac{Q(k) \times \mathrm{count}(k) + v}{\mathrm{count}(k) + 1}$;
　$\mathrm{count}(k) = \mathrm{count}(k) + 1$;
end for

输出： 累积奖赏 r

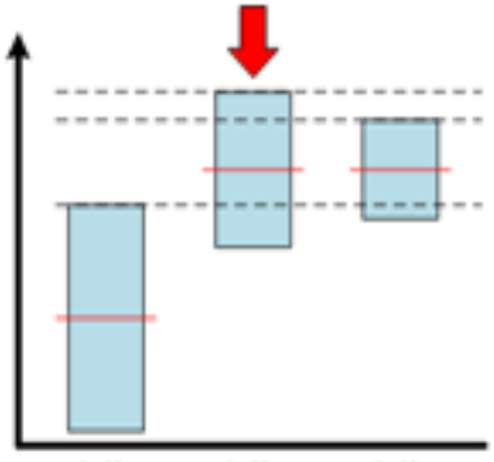
上限置信区间 （Upper-confidence bound）

上限置信区间算法（Upper Confidence Bounds, UCB）：为每个动作的奖励期望计算一个估计范围，优先采用估计范围上限较高的动作

假设每个摇臂的均值为$Q(k)$，估计的偏差为$\delta(k)$

每次根据$Q(k)+\delta(k)$选择摇臂

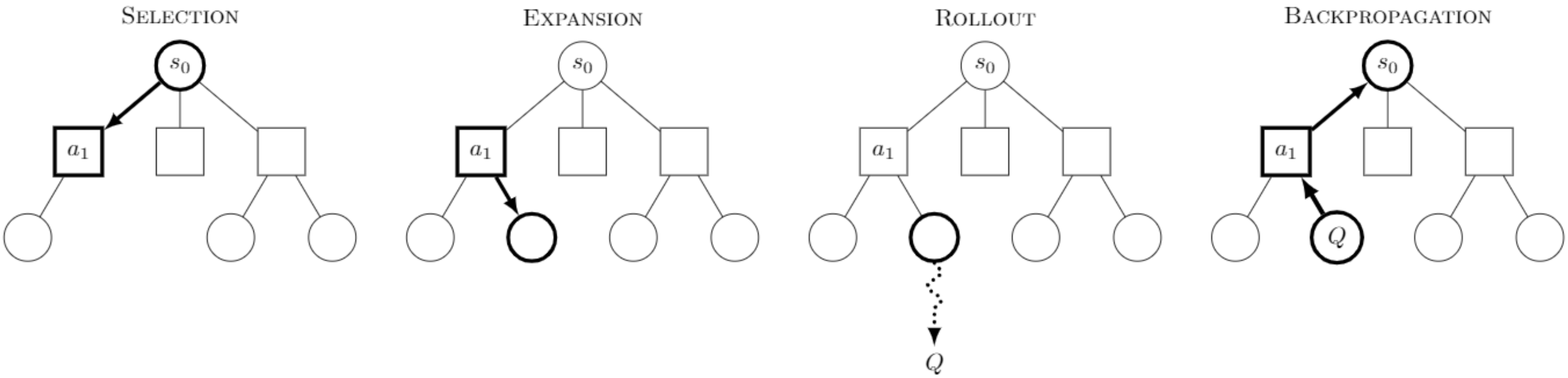
$$\boldsymbol{Q(k)} + \mathbf{2}\ln\sqrt{T/T_k}$$



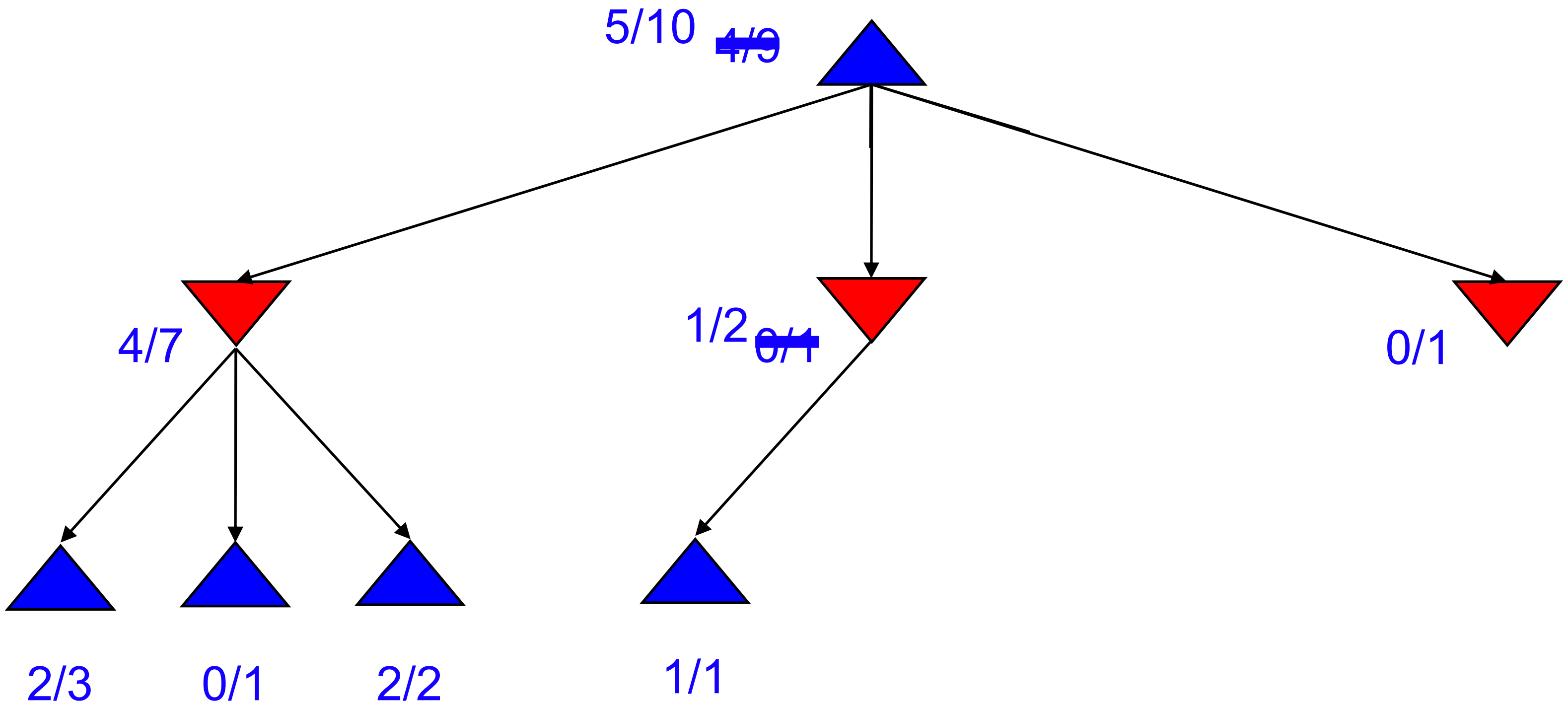
蒙特卡洛搜索

- **选择（selection）**：选择指算法从搜索树的根节点开始，向下递归选择子节点，直至到达叶子节点或者到达具有还未被扩展过的子节点的节点L，可以采用UCB算法
- **扩展（expansion）**：如果节点 L 不是一个终止节点（或对抗搜索的终局节点），则随机扩展它的一个未被扩展过的后继边缘节点M
- **模拟（simulation）**：从节点M出发，模拟扩展搜索树，直到找到一个终止节点
- **回溯（backpropagation）**：用模拟所得结果（终止节点的代价或游戏终局分数）回溯更新模拟路径中M以上（含M）节点的奖励均值和被访问次数

蒙特卡洛搜索

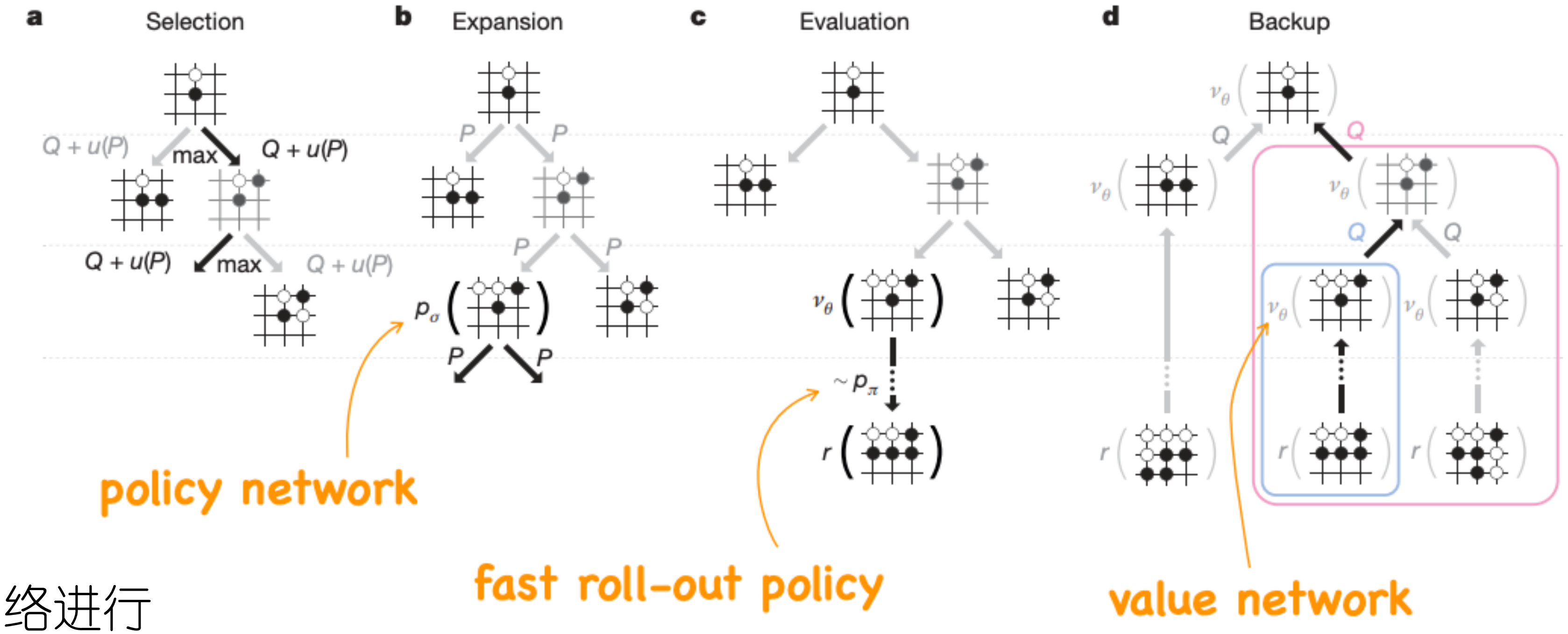

Selection
s_0
a_1
Expansion
s_0
a_1
Rollout
s_0
a_1
Q
Backpropagation
s_0
a_1
Q

蒙特卡洛搜索



AlphaGO

综合使用蒙特卡洛搜索、神经网络、强化学习等技术



训练策略网络进行
节点扩展

训练神经网络直接评估当前节点，无需模拟

博弈AI的发展现状

- **跳棋：** 1990年战胜人类冠军，使用alpha-beta搜索和存有390000亿个残局的数据库表现趋于完美

- **国际象棋：** IBM的深蓝国际象棋程序，1997年击败世界冠军Garry Kasparov，每步棋搜索最多至300亿个棋局，常规搜索深度是14步，某些情况下搜索深度可以达到40层，评估函数考虑了超过8000个特征

- **围棋：** AlphaGO，采用蒙特卡洛搜索+深度强化学习，AlphaZero，无需人类棋谱数据进行训练

博弈AI的发展现状

- **星际争霸：** DeepMind 团队基于多智能体深度强化学习推出的AlphaStar在星际争霸II中达到了人类大师级的水平，并且在《星际争霸II》的官方排名中超越了 99.8%的人类玩家

- **DOTA2：** OpenAI推出的“OpenAI Five”击败世界冠军

- **王者荣耀：** 腾讯推出的觉悟AI，可以击败97%的玩家，并且多次击败顶尖职业团队

	GO	MOBA
Action Space	$250^{150} \approx 10^{360}$ (250 pos available, 150 decisions per game in average)	10^{1500} (10 options, 1500 actions per game)
State Space	$3^{360} \approx 10^{170}$ (361 pos, 3 states each)	10^{20000} (10 heroes, 2000+pos * 10+states)

本章小结

- 双人零和博弈：两个rational agent之间的游戏
- 最大最小搜索：类似于DFS，通过递归实现
- Alpha-beta剪枝：减去不会影响上层节点的分支
- 蒙特卡洛搜索：通过模拟判断节点的价值

搜索部分小结

- 掌握常见的无信息搜索算法，能够编程实现DFS， BFS
- 掌握常见的启发式搜索算法，能够编程实现A*搜索
- 了解博弈搜索算法的基本思想，能够综合运用，解决现实博弈问题，如象棋、黑白棋等。