AI

Contents:

Optimal Solution 最优解

- In normal search 普通搜索
  - The optimal solution would be a sequence of actions leading to a goal state (terminal state) that is a win.

    最优解将是导致获胜的目标状态（终端状态）的一系列动作。
- In adversarial search 对抗搜索
  - Both of MAX and MIN could have an optimal strategy.

    MAX和MIN都会有一个最优策略。
    - In initial state, MAX must find a strategy to specify MAX's move,

      在初始状态，MAX必须找到一个策略来确定MAX的动作，
    - then MAX's moves in the states resulting from every possible response by MIN, and so on.

      然后MAX针对MIN的每个合理的对应采取相应的动作，以此类推。

Minimax Theorem 最小最大定理

For every two-player, zero-sum game with finitely many strategies, there exists a value V and a mixed strategy for each player, such that

对于两个玩家、具有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) Given player 2’s strategy, the best payoff possible for player 1 is V,

给定玩家2的策略，则玩家1可能的最好收益是V，

(b) Given player 1's strategy, the best payoff possible for player 2 is −V.

给定玩家1的策略，则玩家2可能的最好收益是-V。

- For a zero sum game, the name minimax arises because each player *minimizes the maximum payoff* possible for the other, he also *minimizes his own maximum loss*.
  对于零和博弈来说，其名称minimax的由来是因为每个玩家会使对手可能的最大收益变得最小，还会使自己的最大损失变得最小。

Optimal Solution in Adversarial Search 对抗搜索的最优解

- Given a game tree, the optimal strategy can be determined from the minimax value of each node, write as MINIMAX(n).
  给定一棵博弈树，则最优策略可以由每个节点的minimax值来确定，记作MINIMAX(n)。
- Assume that both players play optimally from there to the end of the game.
  假设两个玩家博弈自始至终都发挥得很好。

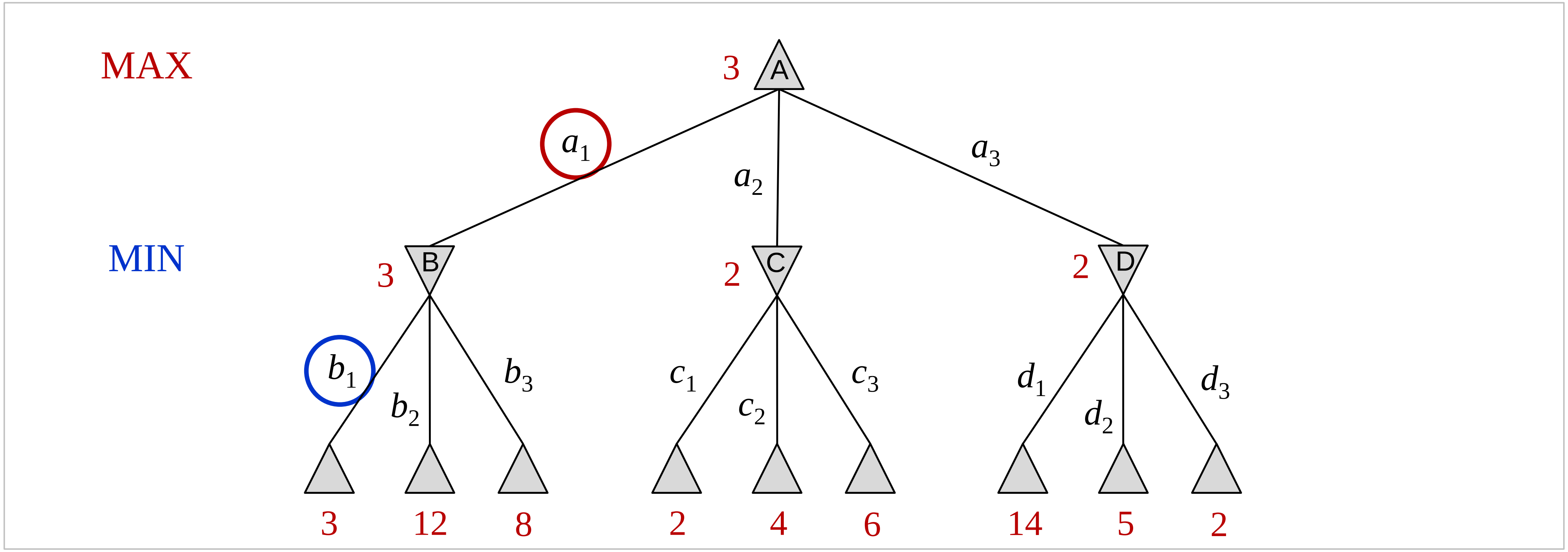
function MINIMAX(s) **returns** an action
  if TERMINAL-TEST(s) **then return** UTILITY(s)
  if PLAYER(s) = MAX **then return** $\max_{a \in \text{ACTIONS}(s)}$ MINIMAX(RESULT(s, a))
  if PLAYER(s) = MIN **then return** $\min_{a \in \text{ACTIONS}(s)}$ MINIMAX(RESULT(s, a))

The minimax value of a terminal state is just its utility.

MAX prefers to move to a state of maximum value, MIN prefers a state of minimum value.

终端状态的minimax值只是其效用。MAX倾向于移动到一个最大值状态，MIN则倾向于一个最小值状态。

Minimax Decision -- A Two-player Game Tree 一个双人玩家的博弈树



MAX's best move at root is a_1 (with the highest minimax value)

根节点处MAX的最佳移动是a_1（具有最高的minimax值）

MIN's best reply at B is b_1 (with the lowest minimax value)

B节点处MIN的最佳应对是b_1（具有最低的minimax值）

Minimax Algorithm 最小最大算法

```
function MINIMAX-DECISION(state) returns an action
    return argmax_{a ∈ ACTIONS(s)} MIN-VALUE(RESULT(state, a))

function MAX-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← −∞
    for each a in ACTIONS(state) do v ← MAX(v, MIN-VALUE(RESULT(state, a)))
    return v

function MIN-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← +∞
    for each a in ACTIONS(state) do v ← MIN(v, MAX-VALUE(RESULT(state, a)))
    return v
```

Properties of Minimax Decision 最小最大决策的性质

The minimax algorithm performs a depth-first exploration of the game tree.
最小最大算法表现为博弈树的深度优先探索。

- Time complexity 时间复杂性

  $O(b^m)$

- Space complexity 空间复杂性

  $O(bm)$ -- The algorithm generates all actions at once
  算法同时生成所有动作

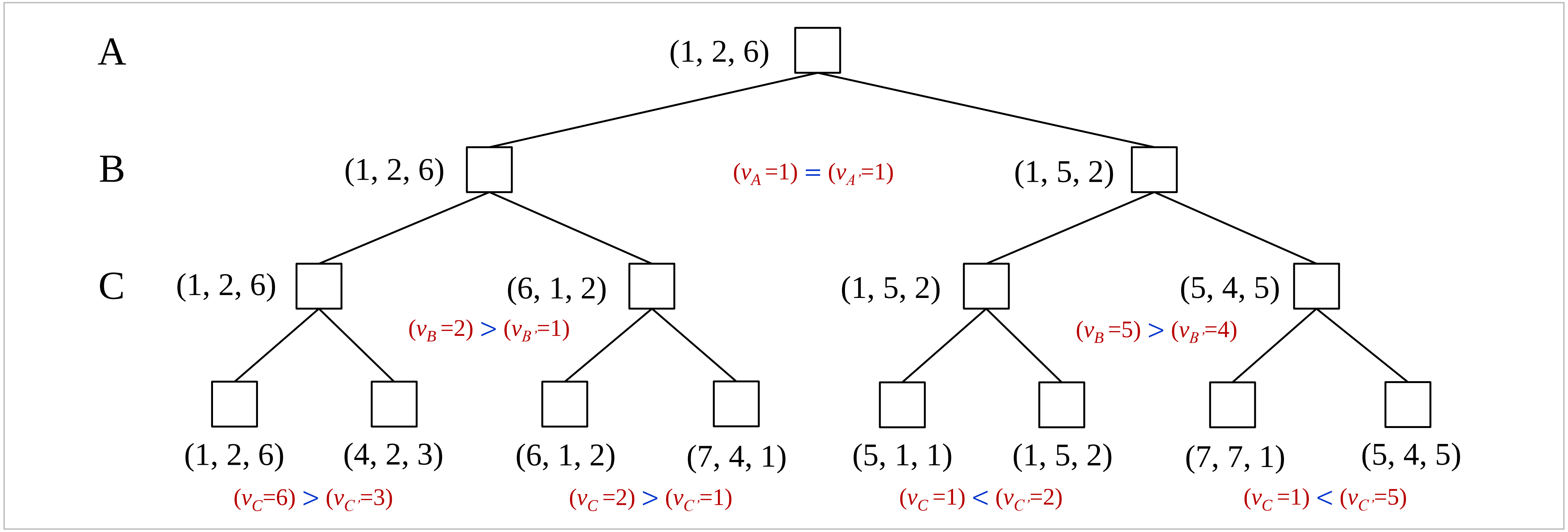
  $O(m)$ -- The algorithm generates actions one at a time
  算法一次生成一个动作

where
- b -- The branching factor (legal moves at each point)
  分支因子（每个点的合法走子）
- m -- The maximum depth of any node
  任一节点的最大深度

Optimal Decisions in Multi-player Games 多玩家博弈中的最优决策

- Let us examine how to extend the minimax idea to multiplayer games.
  让我们考察一下如何将minimax思想扩展到多玩家博弈中。
- We need to replace single value for each node with a *vector* of values.
  我们需要将每个节点的单一值替换为一个值的向量。
  - E.g., in a three-player game with players A, B, and C, a vector (v_A, v_B, v_C) is associated with each node.
    例如，对于具有A、B、C三个玩家的博弈，将向量(v_A, v_B, v_C)与每个节点相关联。
- For terminal states, this vector gives the utility of the state from each player's viewpoint. The simplest way to implement this is to have the UTILITY function return a vector of utilities.
  对于终端状态，从每个玩家的角度来看，这个向量给定该状态的效用。实现的最简单方法是拥有一个返回效用向量的UTILITY函数。

Optimal Decisions in Multi-player Games 多玩家博弈中的最优决策



Multiplayer games usually involve formal or informal alliances among the players.

多玩家博弈通常在玩家之间涉及正式的或非正式的联盟。

Alliances are made and broken as the game proceeds.

联盟随着博弈收益的变化建立或者破裂。