COMP 472 Artificial Intelligence: Adversarial Search part 4
Minimax video #1

- Russell & Norvig: Sections 5.1, 5.2, 5.3

Today

敌对的

- Adversarial Search 

1. Minimax
2. Alpha-beta pruning next video

Motivation

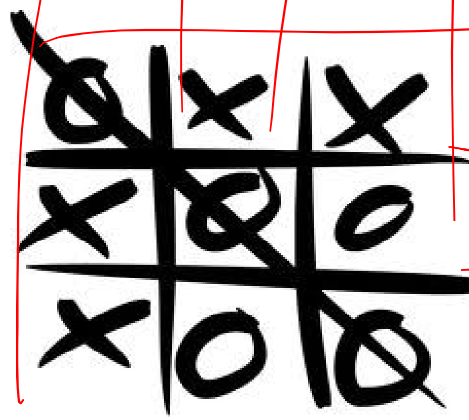
有对手在抗争

GO

black

white

chess



tic-tac-toe

State Space Search for Game Playing

2 players

- Classical application for heuristic search
  - simple games: exhaustibly searchable 类似uninformed，我们能够搜索entire tree
    可被用尽的
  - complex games: only partial search possible
  - additional problem: playing against opponent
- Type of game:
  - 2 person adversarial games
    - win, lose or tie
  - Perfect information
    - both players know the state of the game and all possible moves
  - No chance involved
    - outcome of the game is only dependent on player's moves
  - zero-sum game
    - If the total gains of one player are added up, and the total losses are subtracted, they will sum to zero.
    - a gain by one player must be matched by a loss by the other player
  - eg. chess, GO, tic-tac-toe, …

eg. poker; battleship, …

扑克，有几率问题

不共享信息

Today

- Adversarial Search
  1. Minimax

  2. Alpha-beta pruning

Minimax Search

- Game between two opponents, MIN and MAX

  player#1 player#2
  - MAX tries to win, and
  - MIN tries to minimize MAX's score
- Existing heuristic search methods do not work
  - would require a helpful opponent
  - need to incorporate "hostile" moves into search strategy

- 2 flavors:

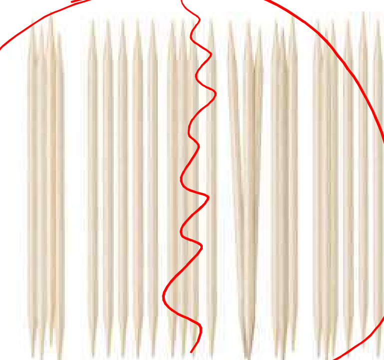
1. exhaustive Minimax  ch
2. n-ply Minimax with Heuristic  st.

Exhaustive Minimax Search

- For small games where exhaustive search is feasible 可行的
- Procedure:
  1. build complete game tree
  2. label each level according to player's turn (MAX or MIN) 实用函数
  3. label leaves with a utility function to determine the outcome of the game end of the game
     - MAX loses MAX wins
     - e.g., (0, 1) or (-1, 0, 1)
     - max loses Max wins
  4. propagate this value up:
     - if parent=MAX, give it max value of children
     - if parent=MIN, give it min value of children
  5. Select best next move for player at the root as the move leading to the child with the highest value (for MAX) or lowest values (for MIN)

Example: Game of Nim

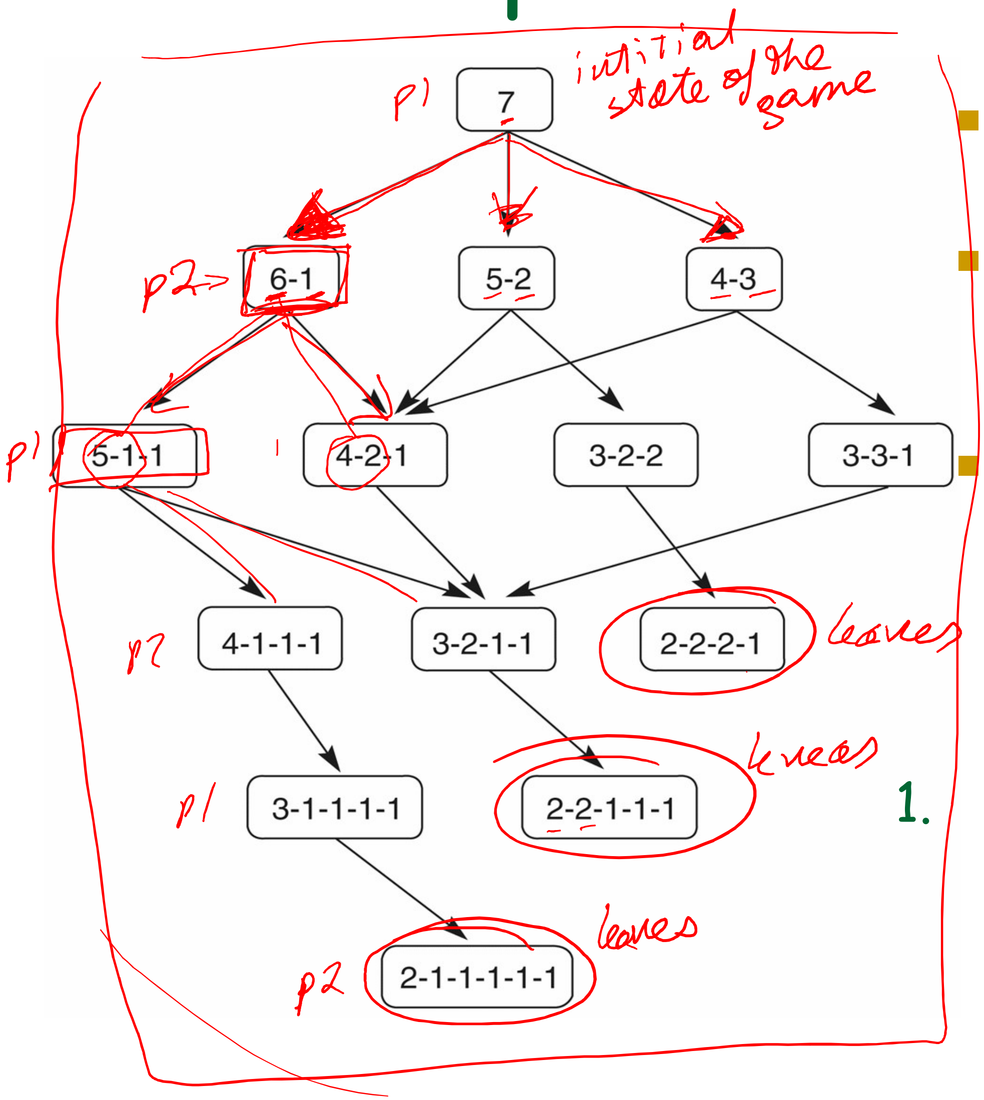
- Rules
  - 2 players start with a pile of tokens
  - move: split (any) existing pile into two non-empty differently-sized piles
  - game ends when no pile can be unevenly split
  - player who cannot make their move loses



toothpicks
coins
rocks

https://www.scholastic.com/parents/kids-activities-and-printables/printables/math-worksheets/game-nim.html

State Space of Game Nim

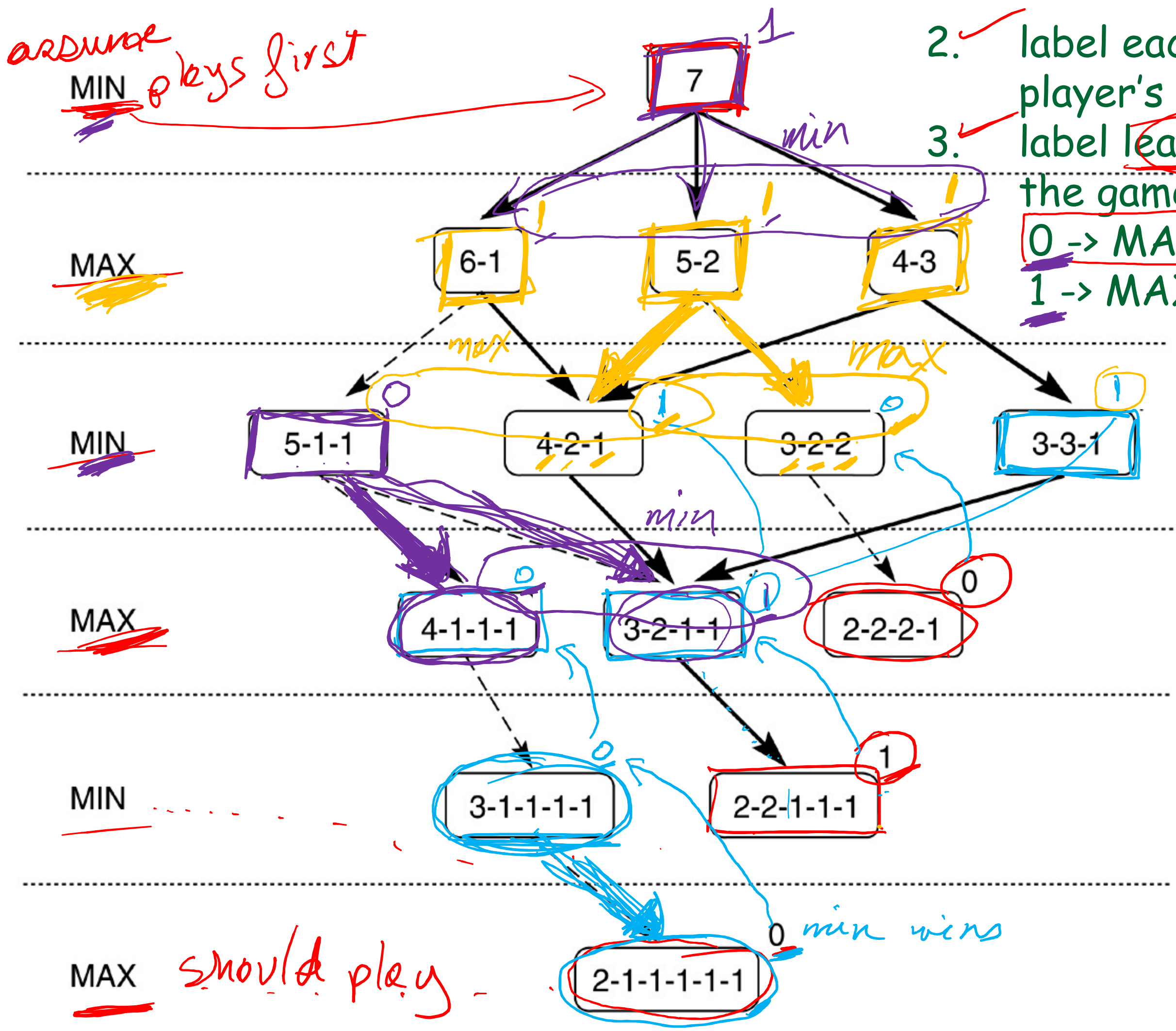


- eg. start with one pile of 7 tokens
- each step has to divide one pile into 2 non-empty piles of different sizes
- player without a move left loses game

1. build complete game tree

source: G. Luger (2005)

Exhaustive Minimax for Nim



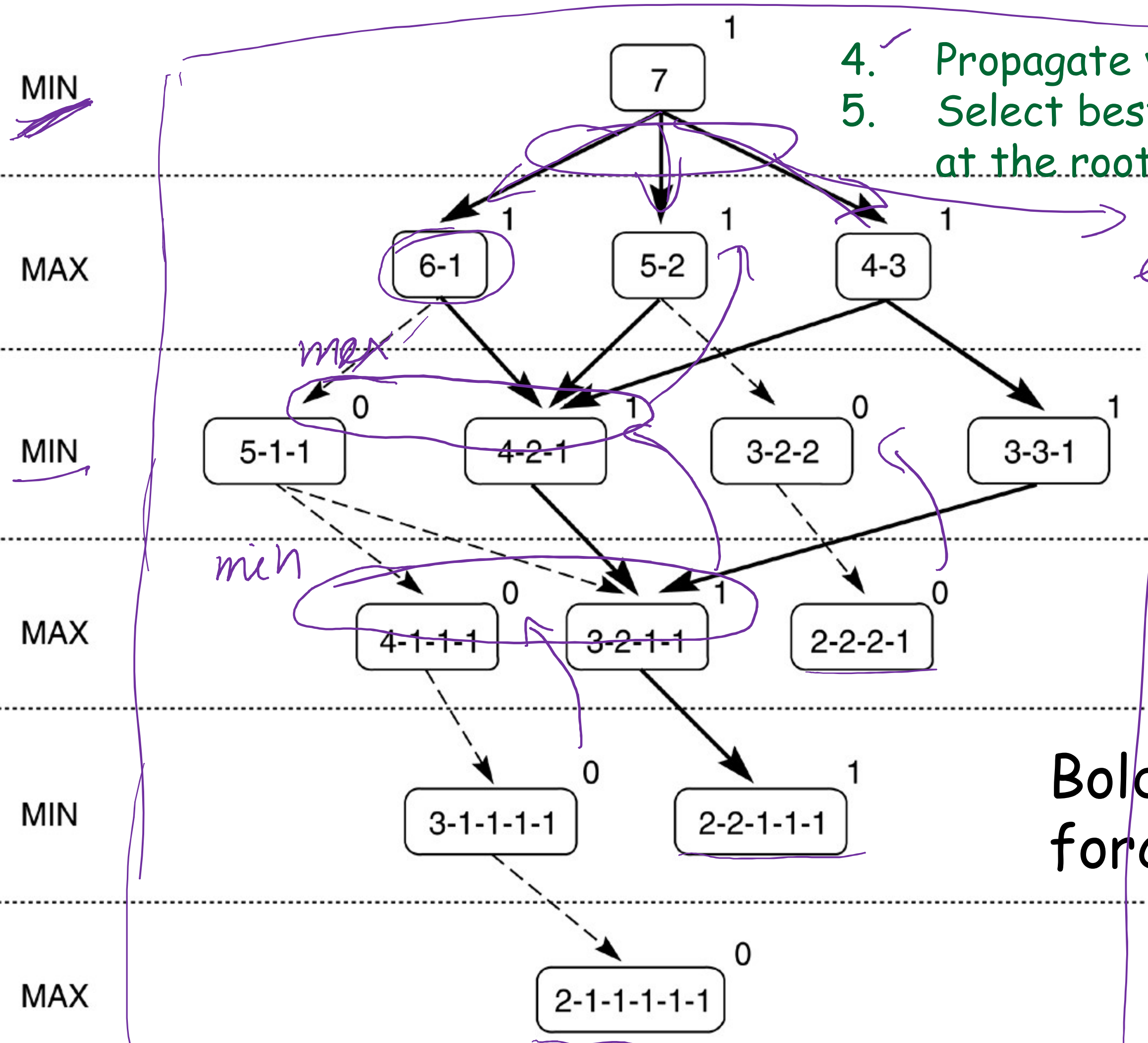
2. label each level according to player's turn (MAX or MIN)
3. label leaves with the outcome of the game
0 -> MAX loses / MIN wins
1 -> MAX wins / MIN loses

123都好说，列树，label每一层的TREE，结果记录谁赢了（取决于在哪层ENDING）

第四步propogate（往上聚合，就是compiler design那个）， 如果没有选择（蓝线），那自然是不变。如果发现他的父亲有选择了，看他的父亲是MAX还是MIN，如果是MIN，取0（紫线），如果是MAX，取1（橙线）

source: G. Luger (2005)

Exhaustive Minimax for Nim



4. Propagate values up.
5. Select best next move for player at the root

all 3 moves are equally bad for MIN

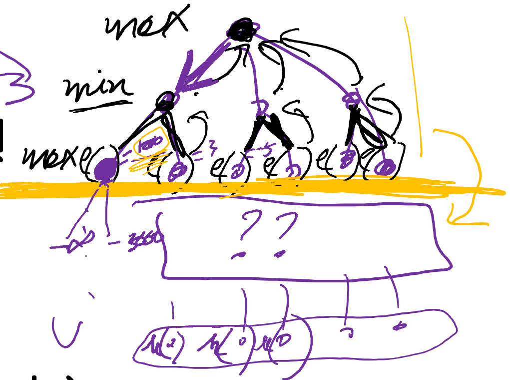
第五步，ROOT的player选对自己最好的那步

Bold lines indicate forced win for MAX

source: G. Luger (2005)

horizon effect:
假设是3-ply,左边是1000，其他都是垃圾，但是因为我们只能看三层，
有可能是个陷阱
有可能1000successor全是垃圾，局面就崩了

n-ply Minimax with Heuristic



- problem with exhaustive Minimax...
  - state space for interesting games is too large!
- solution:
  - search only up to a predefined level
  - called n-ply look-ahead (n = max number of levels)
  - not an exhaustive search

看往前n格的（黄色那层），用e(n)评估，
然后propogate

  - nodes cannot be evaluation with win/loss/tie
  - internal nodes at level n are evaluated with heuristics function e(n)
  - e(n) indicates how good a state seems to be for MAX compared to MIN
  - suffers from the horizon effect

e = 30

How better Max compared to min

对当前状态进行比较

绿的只剩三颗子了

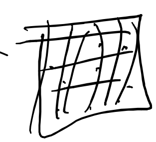
MIN

右上角那个

https://www.wallpaperflare.com/beach-shore-sun-sunrise-sea-horizon-ship-water-sky-sunset-wallpaper-sbhsq

Heuristic Function for 2-player games

- simple strategy:
  - try to maximize difference between MAX's game and MIN's game
- typically called *e(n)*
- *e(n)* is a heuristic that estimates how favorable a node n is for MAX with respect to MIN
  - *e(n)* > 0 --> n is favorable to MAX
  - *e(n)* < 0 --> n is favorable to MIN
  - *e(n)* = 0 --> n is neutral



Choosing a Heuristic Function *e(n)*

- Usually *e(n)* is a weighted sum of various features:

$$e(n) = \sum w_i \, f_i(n)$$

dependent on actual game

- E.g. of features:
  - f_1 = number of pieces left on the game for MAX
  - f_2 = number of possible moves left for MAX
  - f_3 = -(number of pieces left on the game for MIN)
  - f_4 = -(number of possible moves left for MIN)

f_5 : queen still alive (1/0)

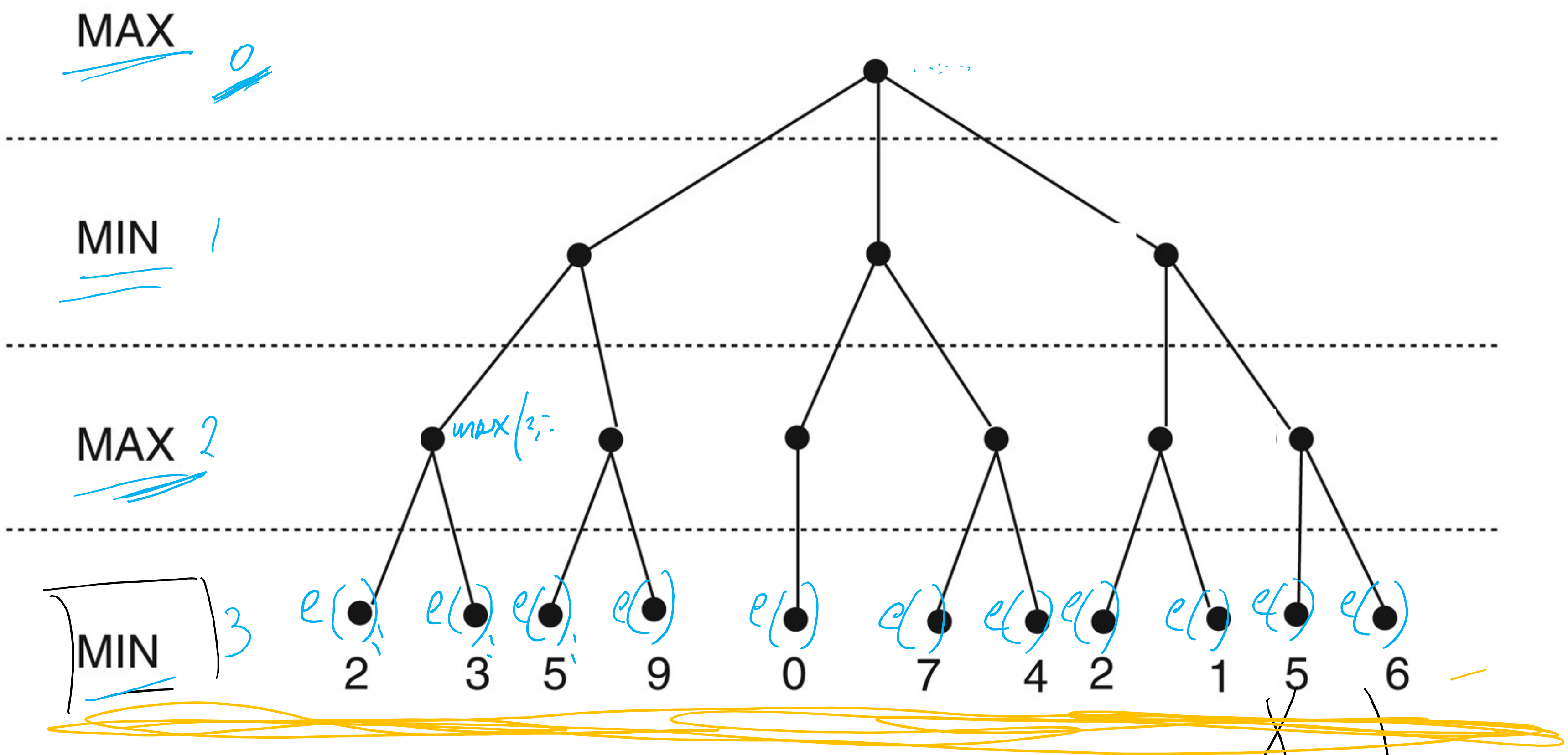
f_6 : # of bishops left

- E.g. of weights:
  - w_1 = 0.5 (0.1) // f_1 is a very important feature
  - w_2 = 0.2 // f_2 is not very important
  - w_3 = 0.2 // f_3 is not very important
  - w_4 = 0.1 // f_4 is really not important

$w_5 = 0.3$

Minimax with 3-ply look-ahead

不包括自己往下看三层

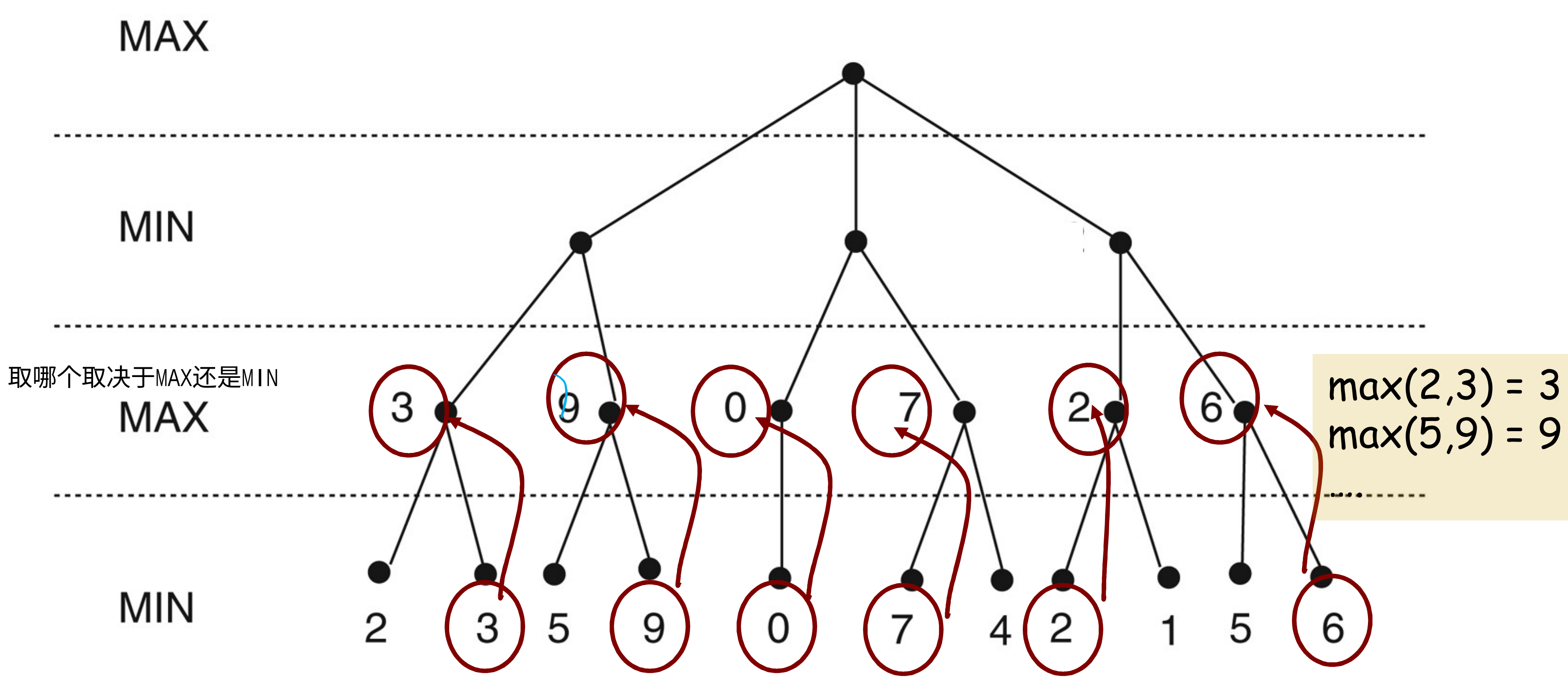


Leaf nodes show the actual heuristic value e(n)

leaves = end of the game

source: G. Luger (2005)

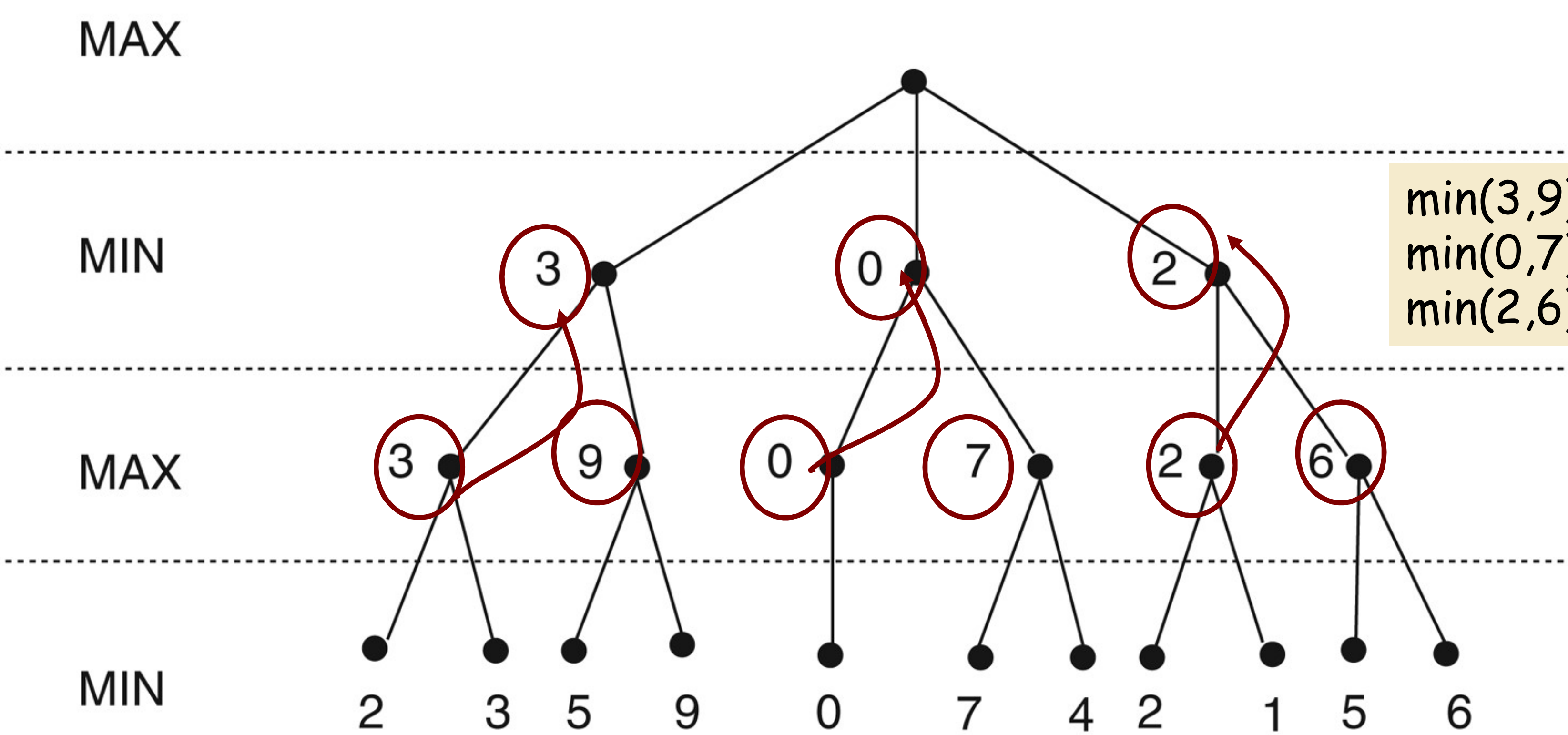
Minimax with 3-ply look-ahead



Leaf nodes show the actual heuristic value *e(n)*
Internal nodes show back-up heuristic value

source: G. Luger (2005)

Minimax with 3-ply look-ahead

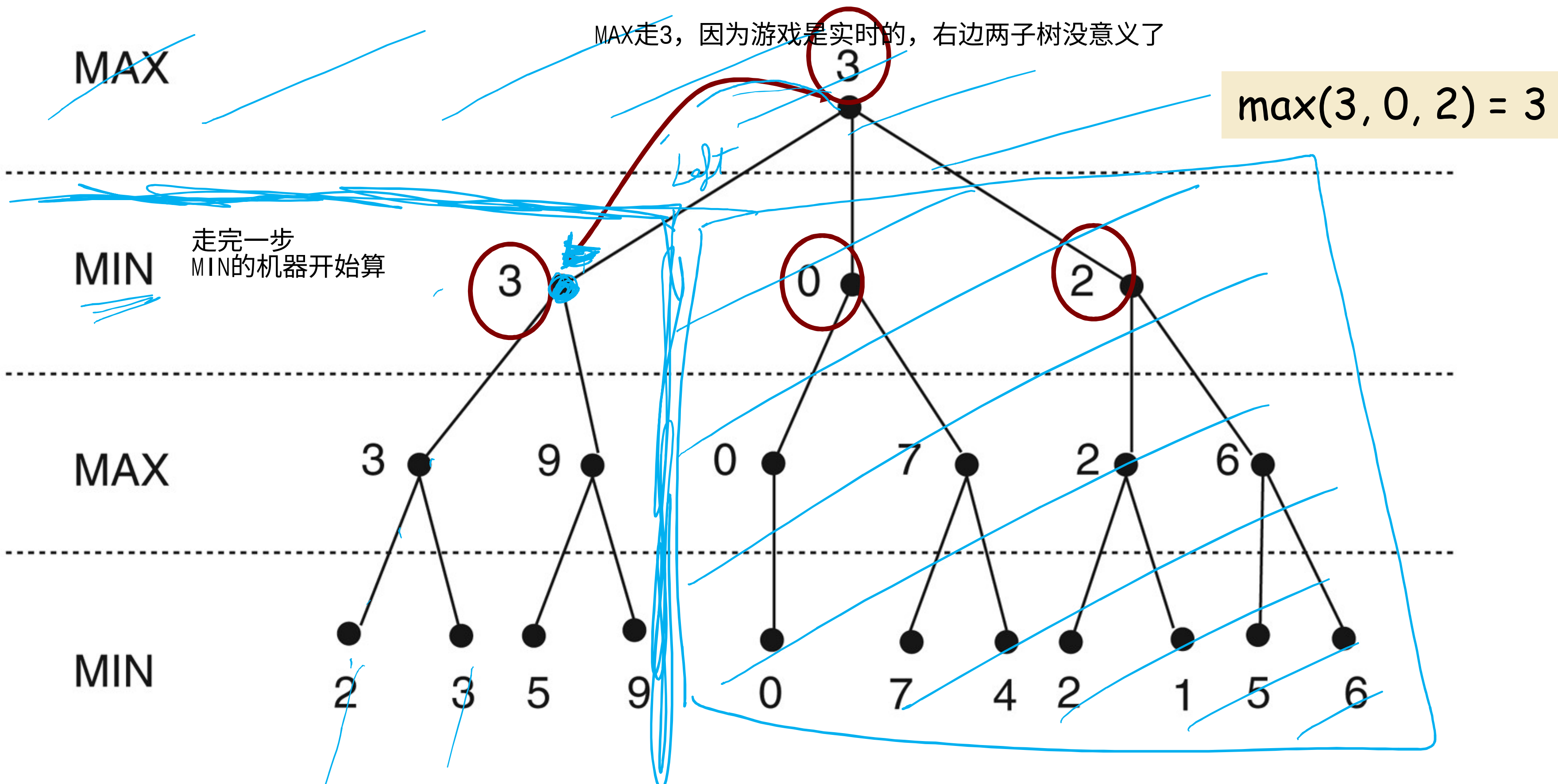


min(3,9) = 3
min(0,7) = 0
min(2,6) = 2

Leaf nodes show the actual heuristic value *e(n)*
Internal nodes show back-up heuristic value

source: G. Luger (2005)

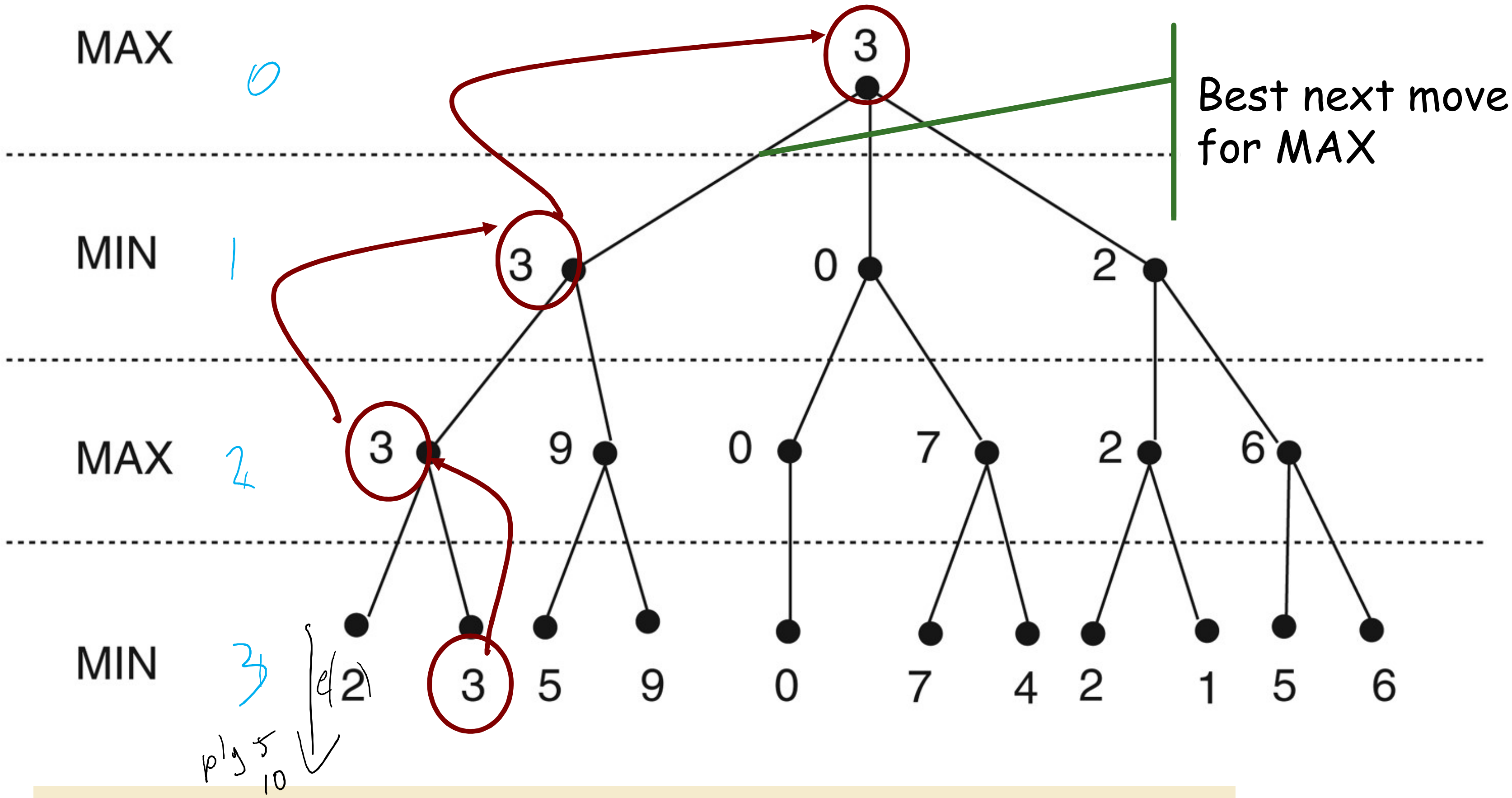
Minimax with 3-ply look-ahead



Leaf nodes show the actual heuristic value e(n)
Internal nodes show back-up heuristic value

source: G. Luger (2005)

Minimax with 3-ply look-ahead

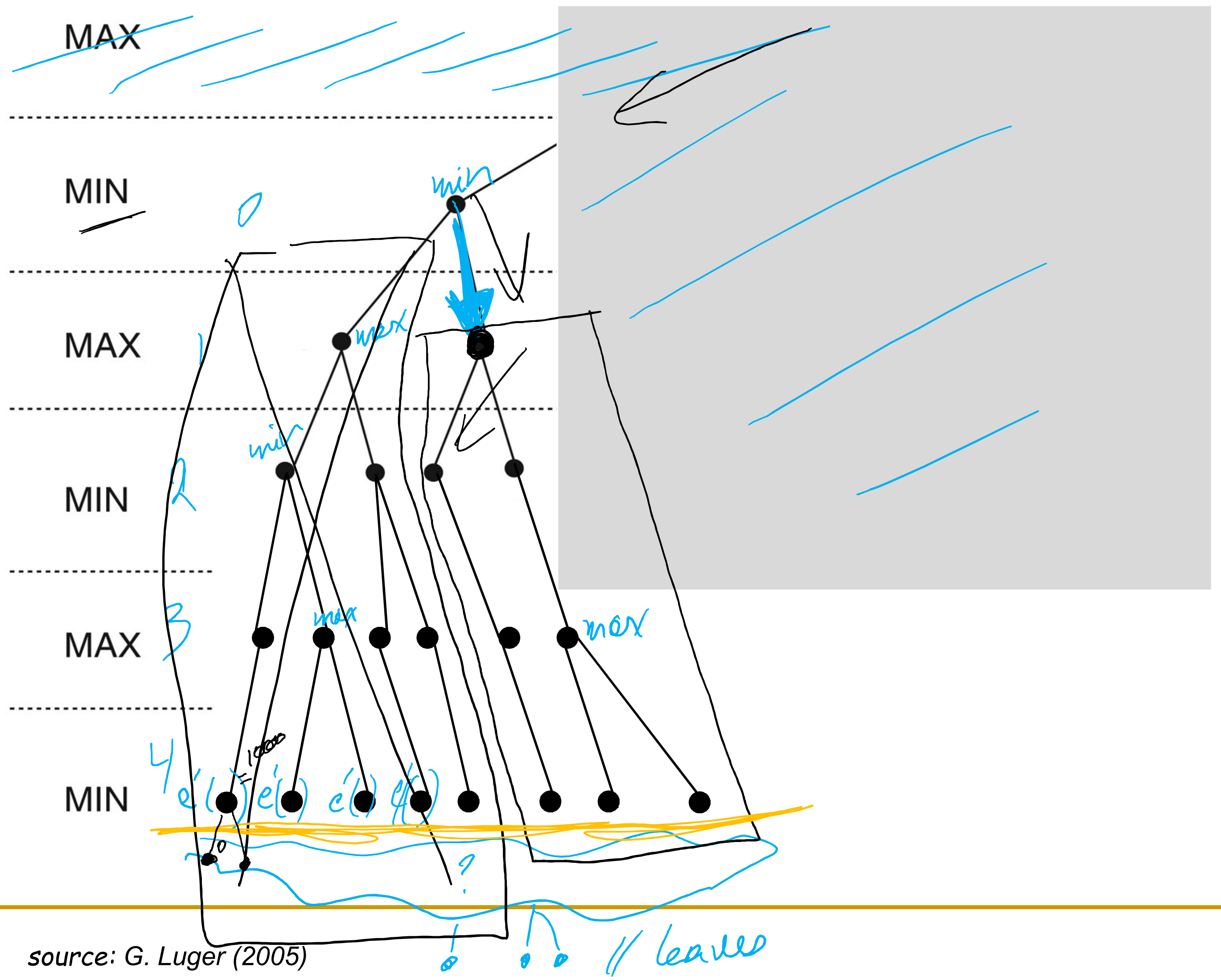


Leaf nodes show the actual heuristic value *e(n)*
Internal nodes show back-up heuristic value

source: G. Luger (2005)

and then, MIN will play

assume MIN can afford to look 4 ply ahead



source: G. Luger (2005)

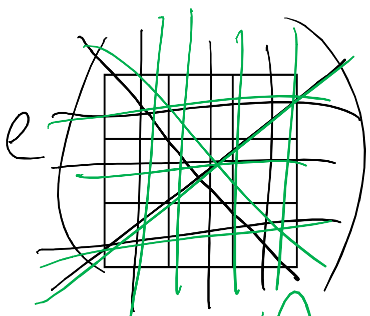
Example: $e(n)$ for Tic-Tac-Toe

- assume MAX plays X
- possible $e(n)$

en不是越informed越好，informed代表慢，代表n-ply深度浅，这是个trade off

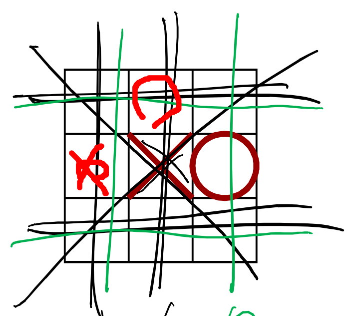
但是，如果我们算力足够，那么我们是要10-ply(相对慢)，还是3-ply呢，答案是10ply,越DEEP越好

$$e(n) = \begin{cases} \text{number of rows, columns, and diagonals open for MAX} \\ \quad - \text{number of rows, columns, and diagonals open for MIN} \\ +\infty, \text{ if n is a forced win for MAX} \\ -\infty, \text{ if n is a forced win for MIN} \end{cases}$$



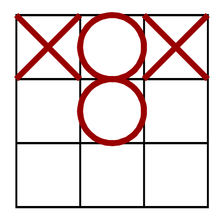
$e(n) = 8 - 8 = 0$

max min



$e(n) = 6 - 4 = 2$

max min



$e(n) = 3 - 3 = 0$

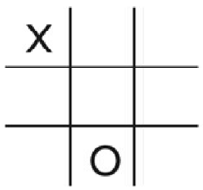
我们没在数可能的步数，我们在数可能的赢的路线，

8就是上下左右3+3+斜线2

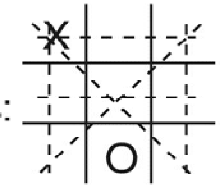
X这里被圈挡掉两个所以是6

0被×多挡了斜线两个所以是4

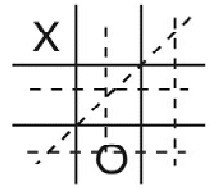
More examples…



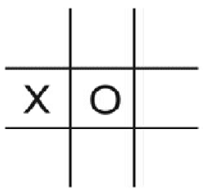
X has 6 possible win paths:



O has 5 possible wins:

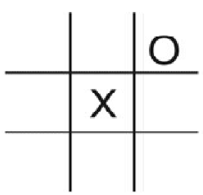


$E(n) = 6 - 5 = 1$



X has 4 possible win paths;
O has 6 possible wins

$E(n) = 4 - 6 = -2$



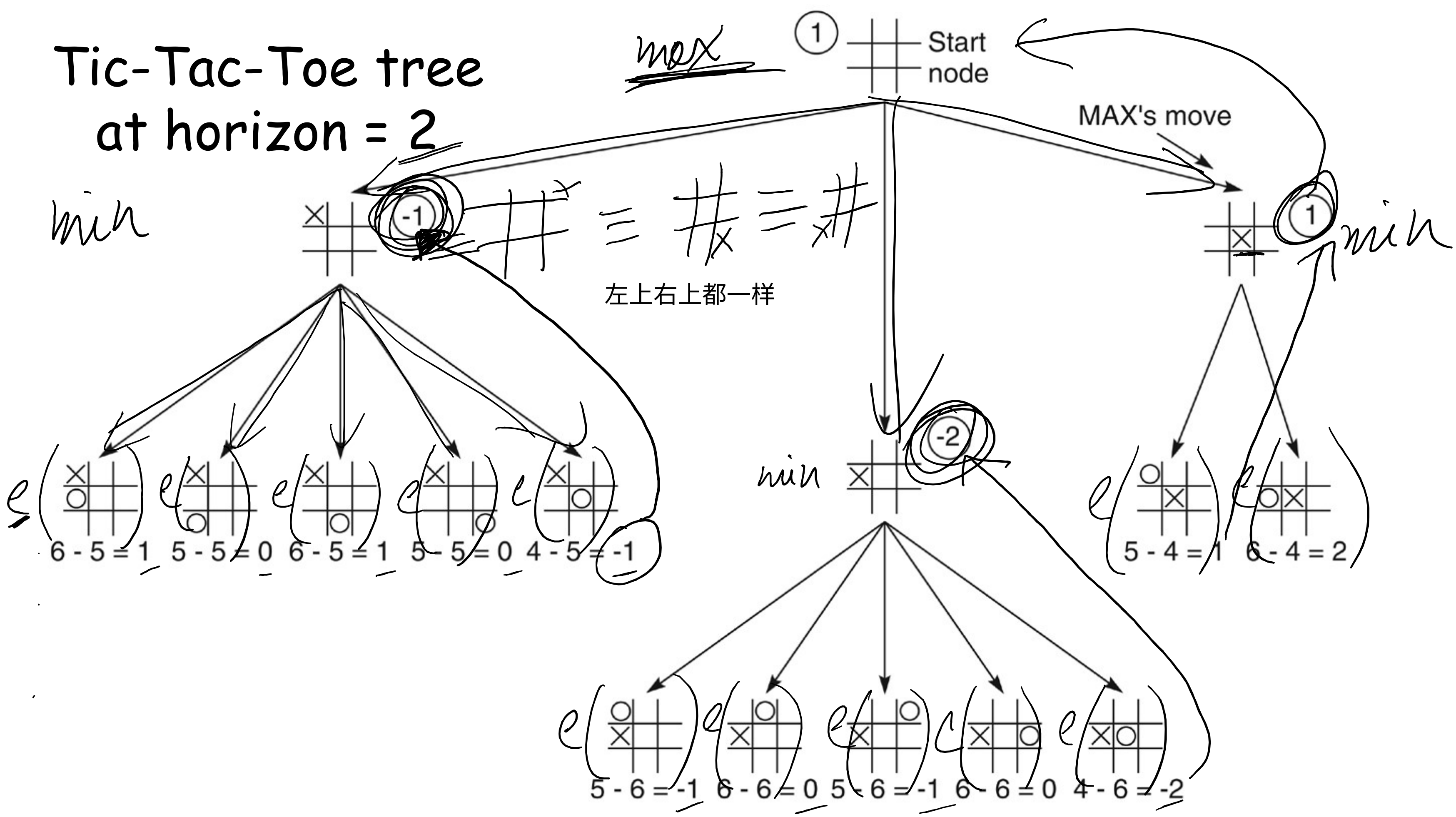
X has 5 possible win paths;
O has 4 possible wins

$E(n) = 5 - 4 = 1$

source: G. Luger (2005)

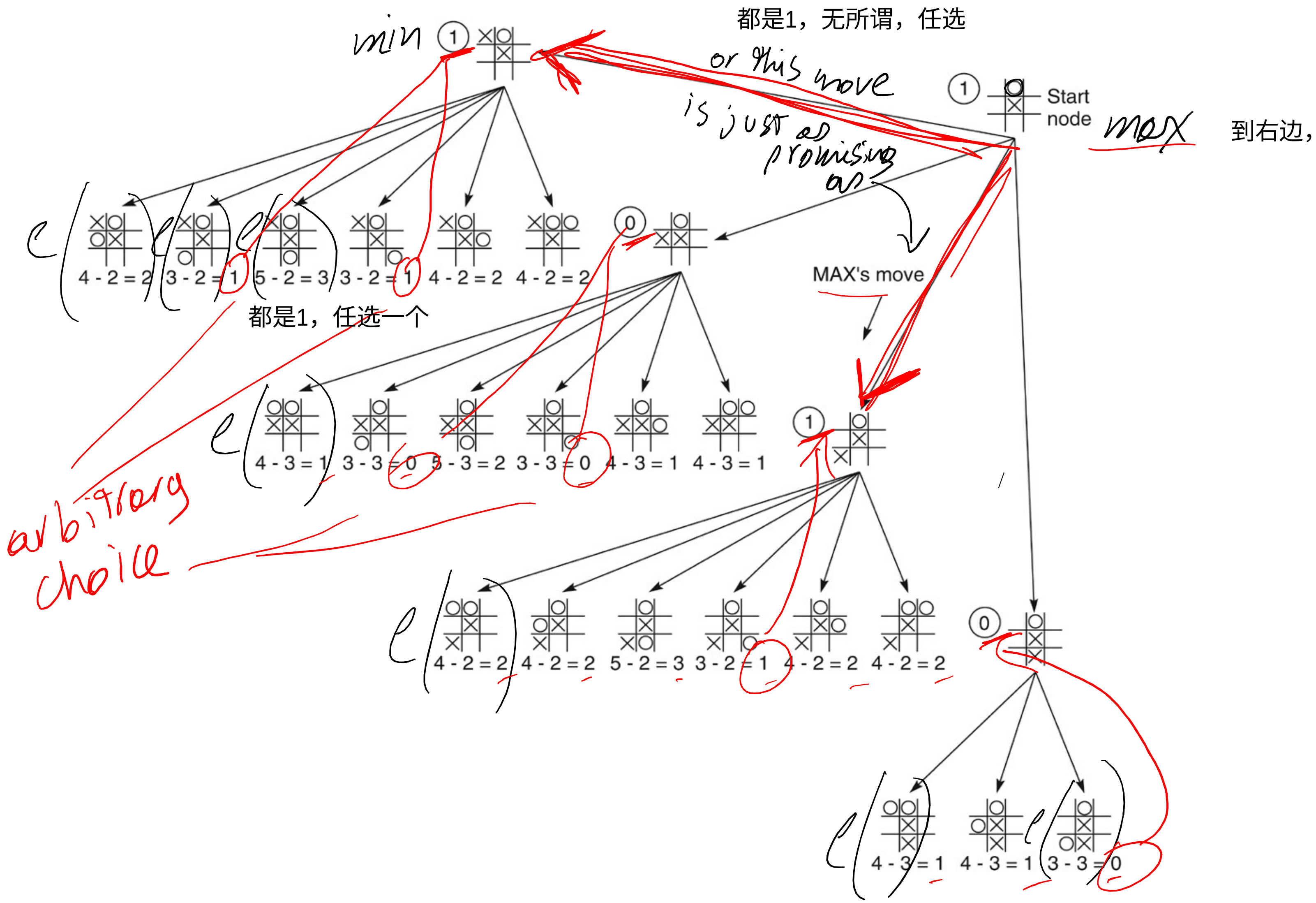
2-ply Minimax for Opening Move

Tic-Tac-Toe tree
at horizon = 2



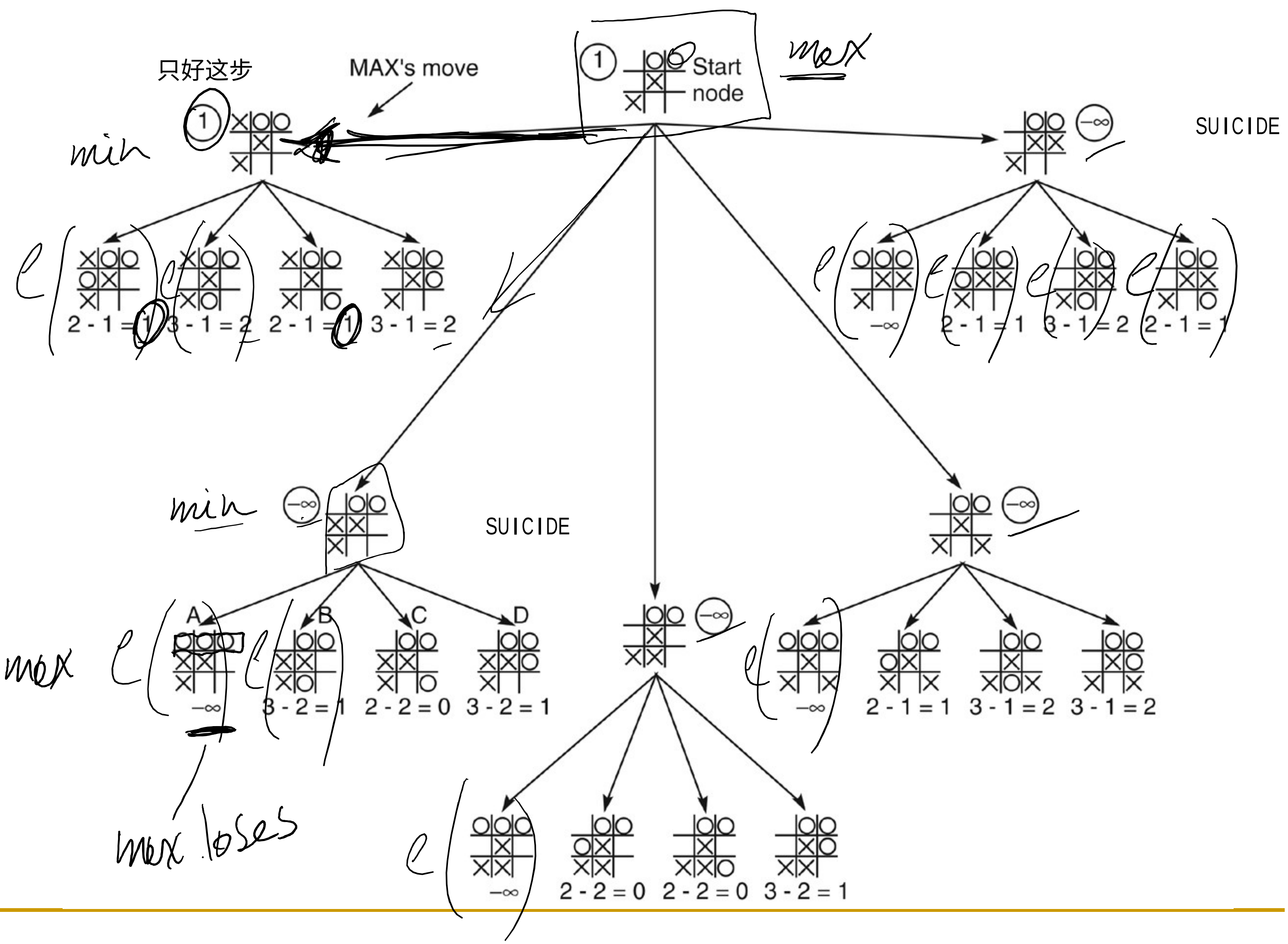
source: G. Luger (2005)

2-ply Minimax: MAX's possible 2nd moves



source: G. Luger (2005)

2-ply Minimax: MAX's move at end



source: G. Luger (2005)

Today

- Adversarial Search

1. Minimax
2. Alpha-Beta Pruning

Up Next

- Adversarial Search
  1. Minimax
  2. Alpha-Beta Pruning