Artificial Intelligence: Adversarial Search part 4 Minimax

- Russell & Norvig: Chapter 5

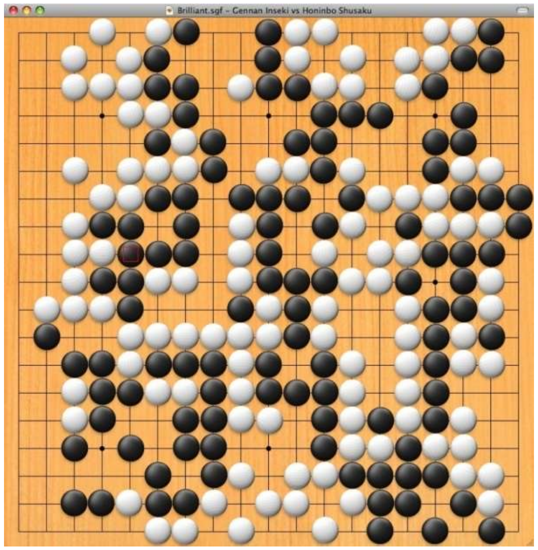
Today

- Adversarial Search YOU ARE HERE!

chap 5

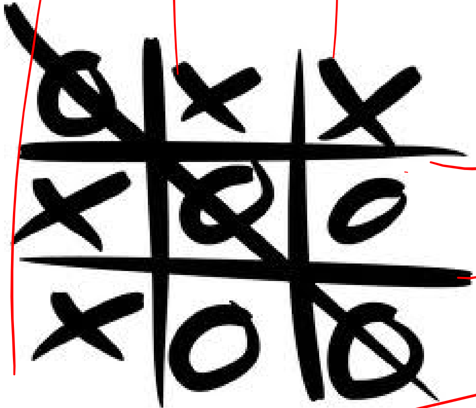
1. Minimax 4.1
2. Alpha-beta pruning 4.2
3. Other Adversarial Search 4.3
    1. Multiplayer Games
    2. Stochastic Games
    3. Monte Carlo Tree Search

Motivation



GO

chess



tic-tac-toe

Games

4.1 & 4.

4.3

v1

1. ## Assumptions
   1. Zero-sum game:
      - no player has a game advantage. If the total gains of one player are added up, and the total losses are subtracted, they will sum to 0.
   2. Players play rationally (i.e. to win)

2. ## Characteristics of Games

	Deterministic	Stochastic
Perfect information	chess, checkers, go	backgammon, monopoly
Imperfect information	battleship	scrabble, poker, bridge

- ☐ Number of players? ~~1~~, 2, 3+

ze

- ☐ Deterministic:
  - the outcome of the game is only dependent on the moves of the players
- ☐ Stochastic:
  - chance involved
- ☐ Perfect information:
  - all players know the state of the game and all possible moves a.k.a *fully observable*
- ☐ Imperfect information:
  - eg. a player is hiding their game aka. *partially observable*

Techniques

Nb of players	Deterministic / Stochastic ?	Perfect / Imperfect information?	Example	Technique
1	deterministic	perfect info	8-puzzle	heuristic search part 3 ✓
2	deterministic	perfect info	chess, checkers, go video 4.1	minimax & alpha-beta 4.2
3+	deterministic	perfect info	Chinese Checkers	max^n
1	stochastic	yes	2048	expectimax
2	stochastic	yes	backgammon	expectiminimax
3+	stochastic	yes		modified expectimax

video 4.3

Techniques

Nb of players	Deterministic / Stochastic ?	Perfect / Imperfect information?	Example	Technique
1	deterministic	perfect info	8-puzzle	heuristic search
2	deterministic	perfect info	chess, checkers, go	minimax & alpha-beta
3+	deterministic	perfect info	Chinese Checkers	max^n
1	stochastic	yes	2048	expectimax
2	stochastic	yes	backgammon	expectiminimax
3+	stochastic	yes		modified expectimax

Today

- Adversarial Search
  1. Minimax

  2. Alpha-beta pruning
  3. Other Adversarial Search
     1. Multiplayer Games
     2. Stochastic Games
     3. Monte Carlo Tree Search

Minimax Search

player#1 player#2

- Game between two opponents, MIN and MAX
  - 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

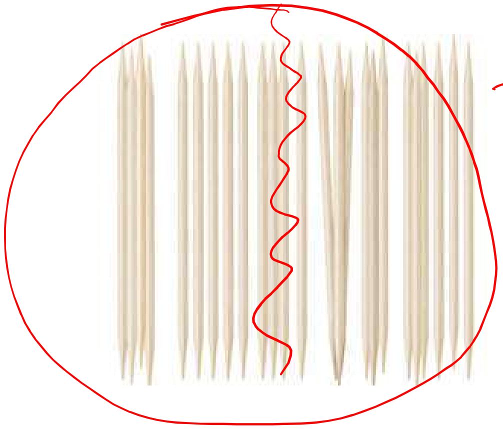
- For small games where exhaustive search is feasible
- Procedure:
  1. build complete game tree
  2. label each level according to player's turn (MAX or MIN)
  3. label leaves with a utility function to determine the outcome of the game

     end of the game
     - e.g., (0, 1) or (-1, 0, 1)

       MAX loses MAX wins 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

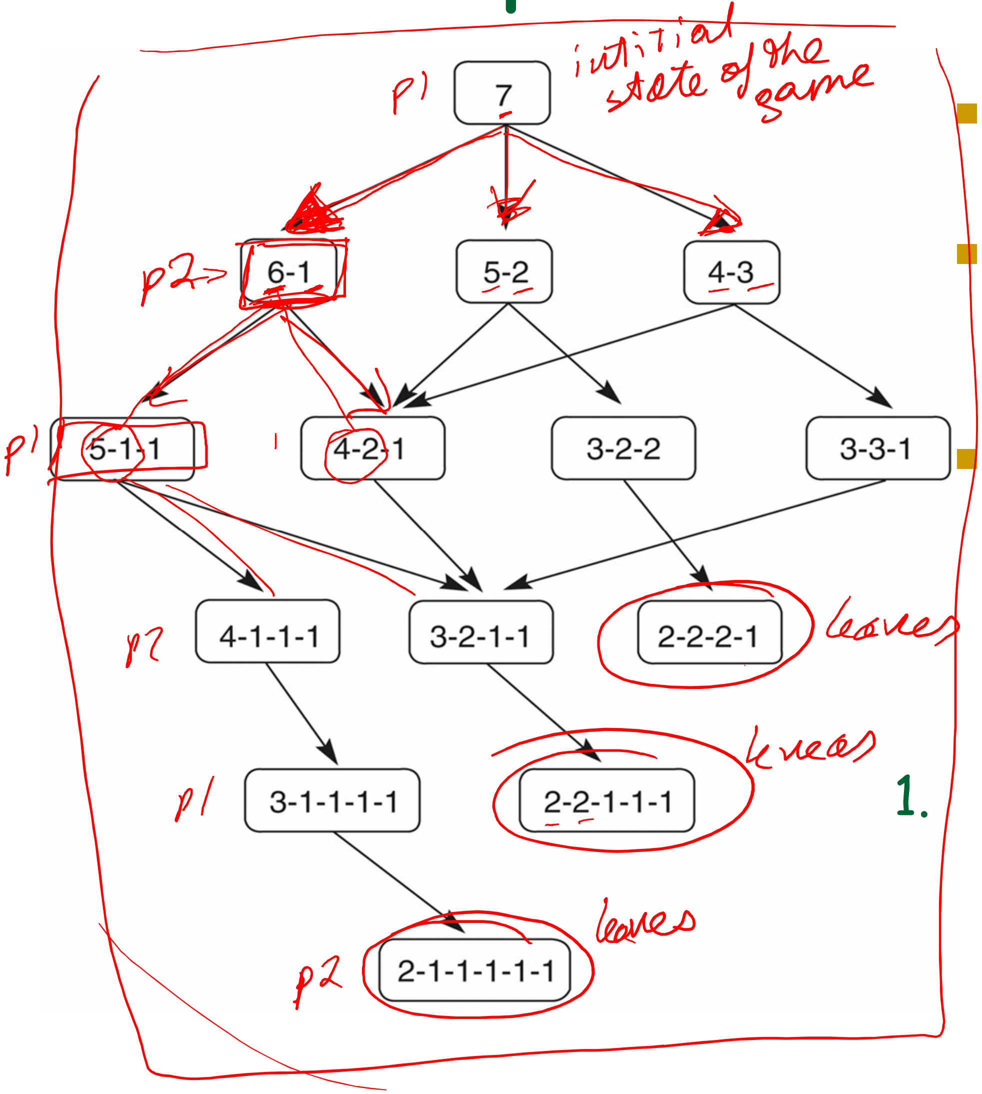
- Rules
  - 2 players start with a pile of tokens
  - move: split (any) existing pile into two non-empty differently-sized piles
  - game ends when no pile can be unevenly split
  - player who cannot make their move loses



toothpicks
coins
rocks

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

State Space of Game Nim

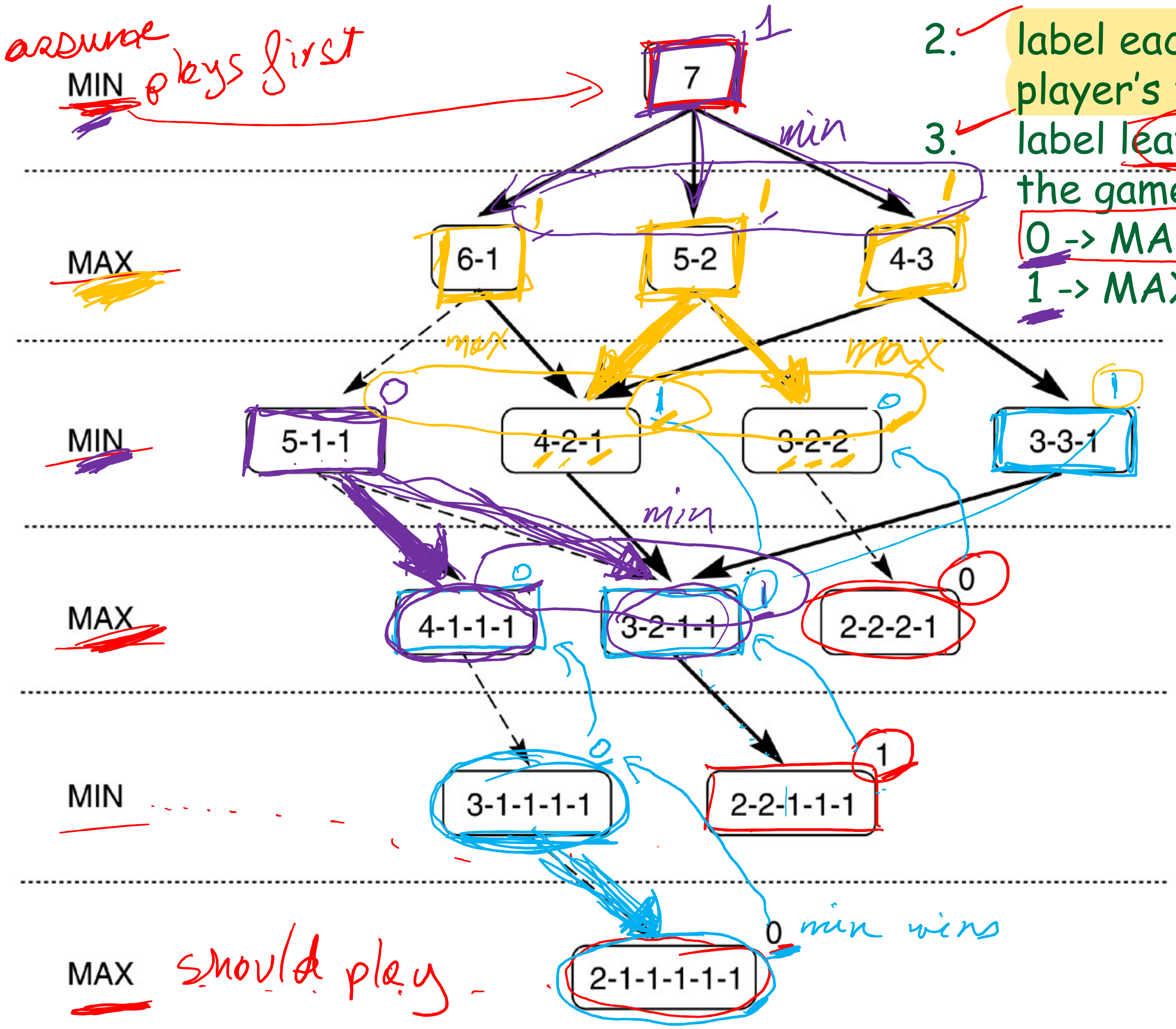


- eg. start with one pile of 7 tokens
- each step has to divide one pile into 2 non-empty piles of different sizes
- player without a move left loses game

1. build complete game tree ✓

source: G. Luger (2005)

Exhaustive Minimax for Nim



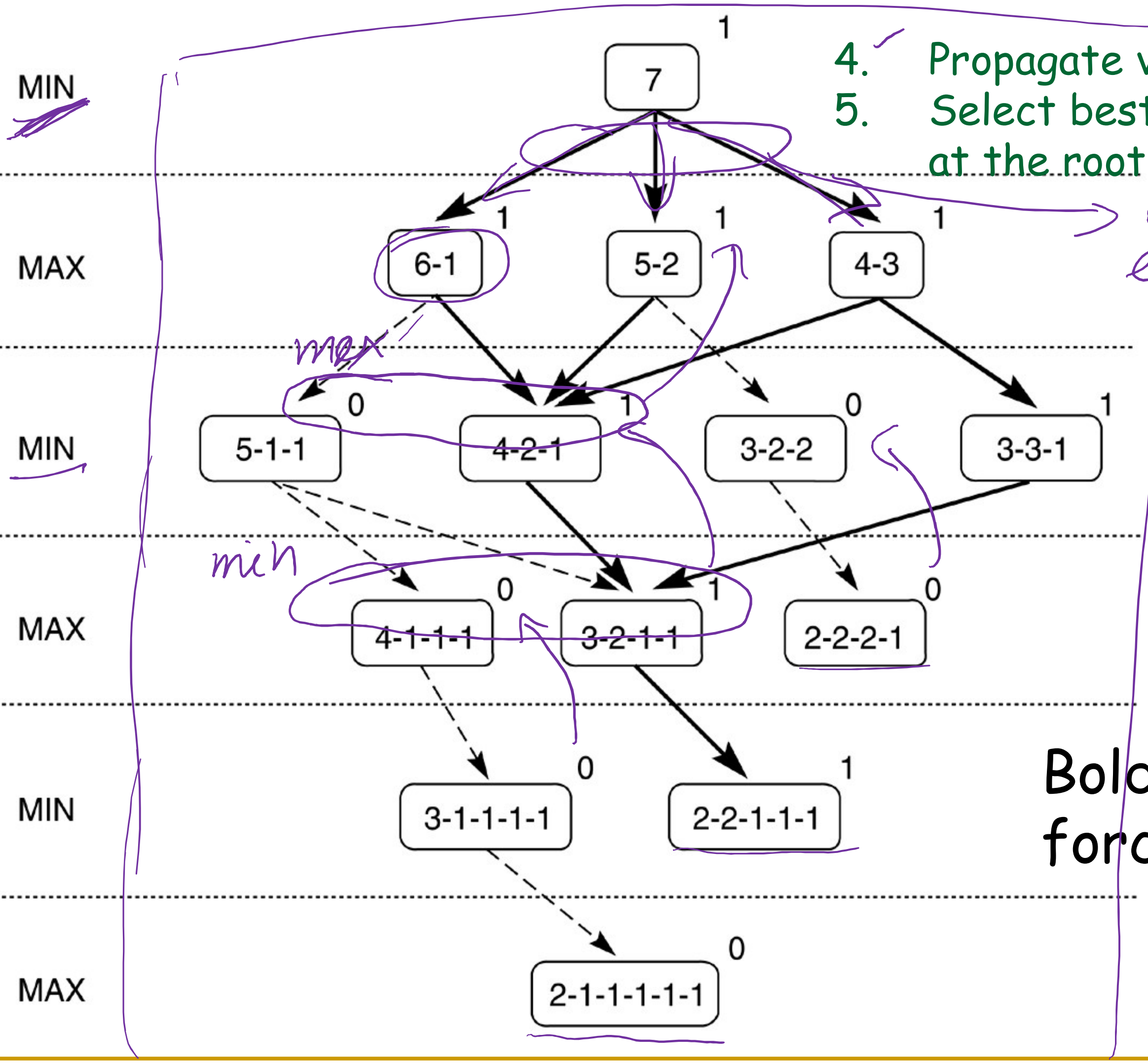
2. label each level according to player's turn (MAX or MIN)
3. label leaves with the outcome of the game

0 -> MAX loses / MIN wins

1 -> MAX wins / MIN loses

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

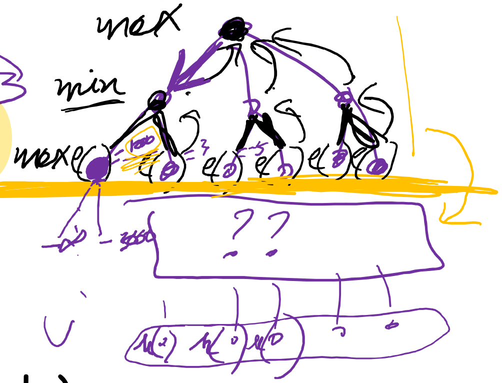
Bold lines indicate forced win for MAX

source: G. Luger (2005)

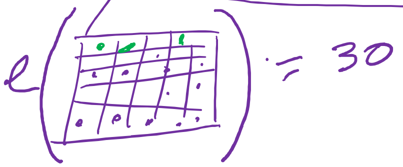
n-ply Minimax with Heuristic

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

horizon



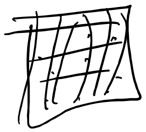
exact value of the leafs



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)

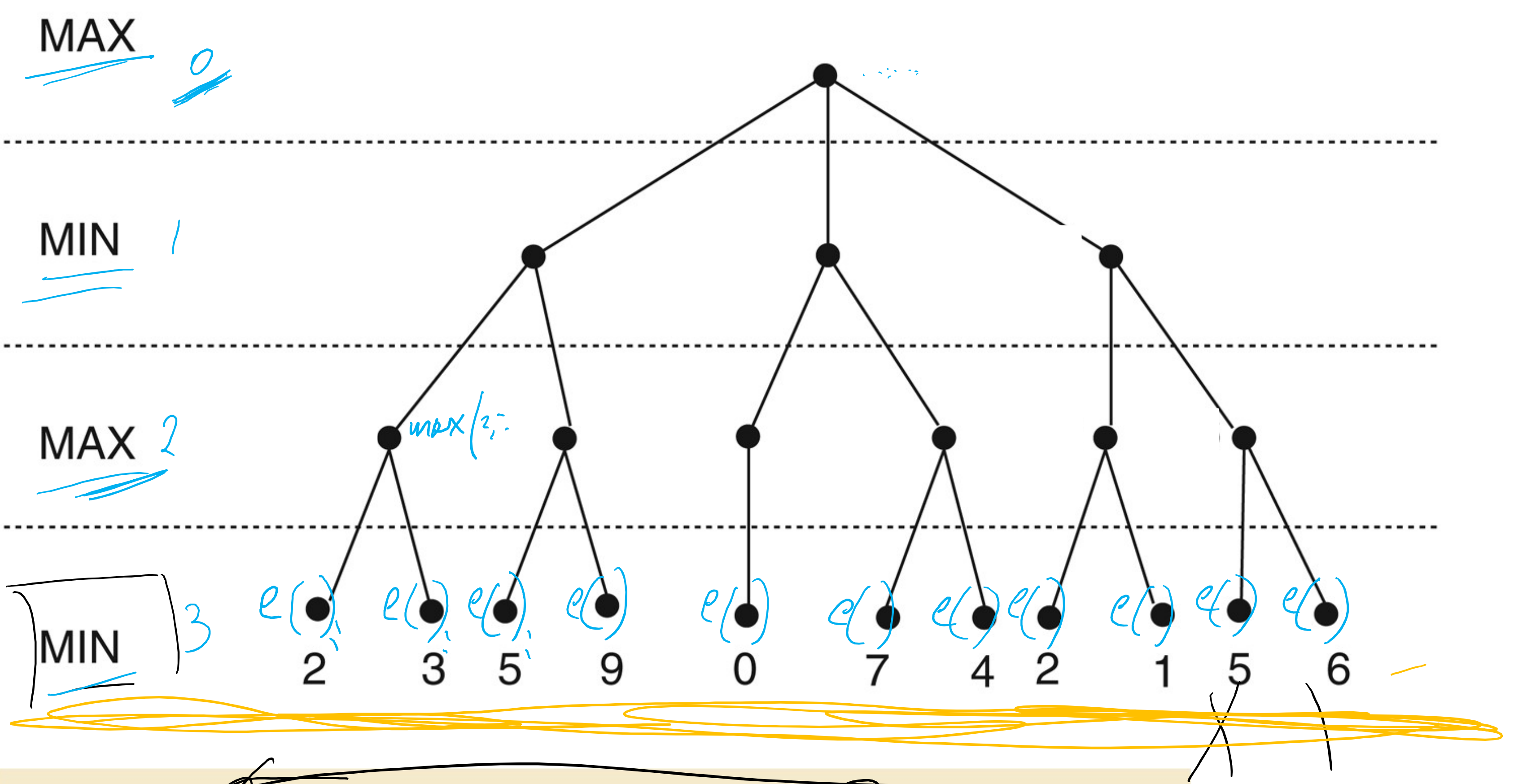
f_5 : queen still alive (1/0)

f_6 : # of bishops left

- E.g. of weights:
  - ❑ w_1 = $\overset{0.1}{0.5}$ // 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

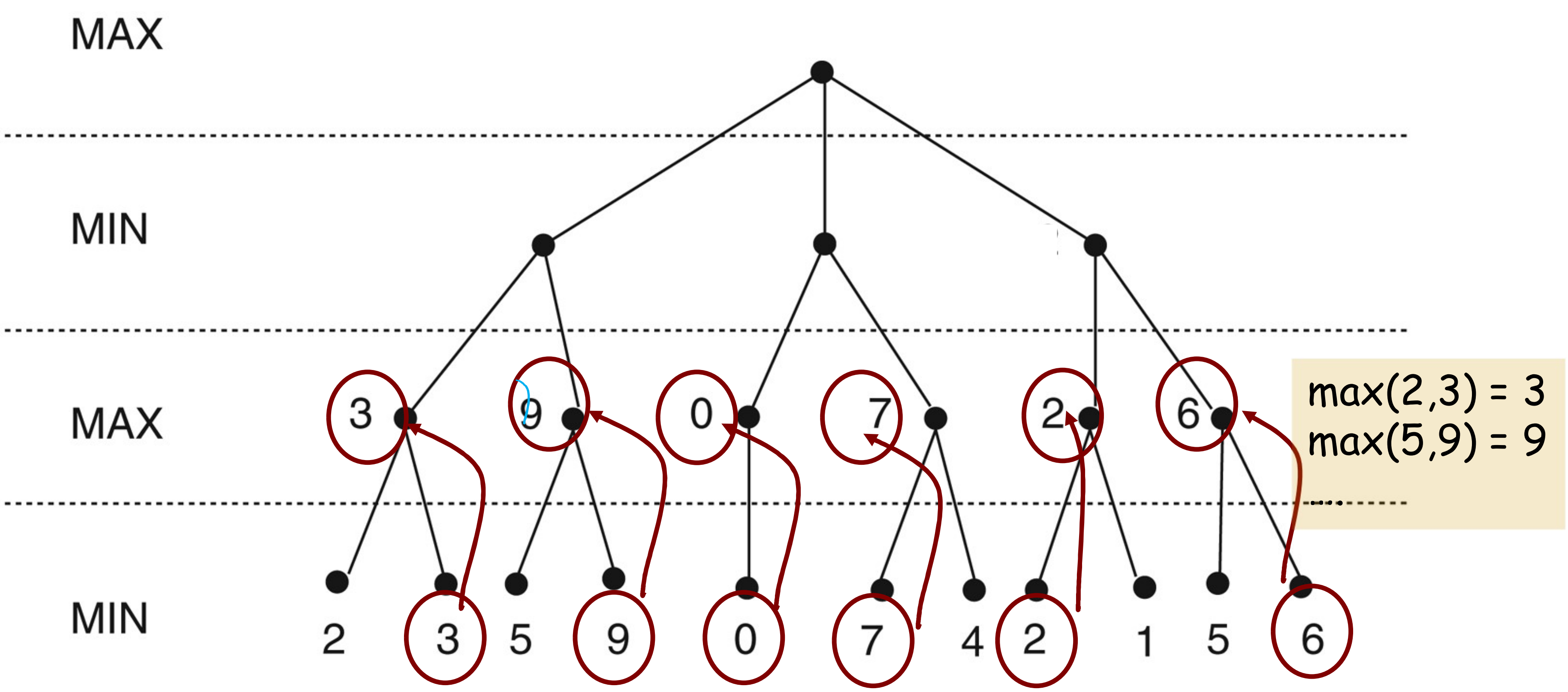


Leaf nodes show the actual heuristic value e(n)

leaves = end of the game

source: G. Luger (2005)

Minimax with 3-ply look-ahead

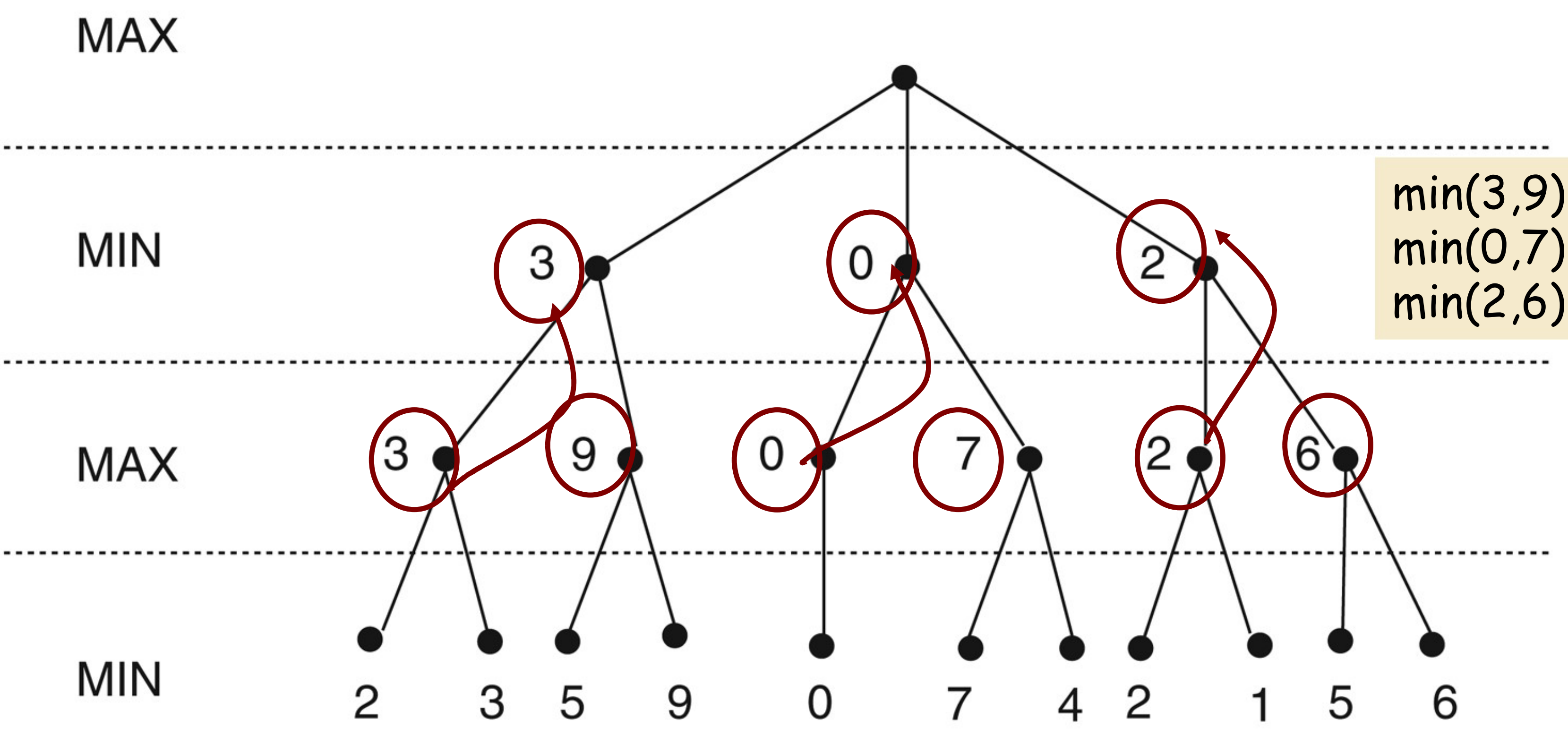


max(2,3) = 3
max(5,9) = 9
….

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

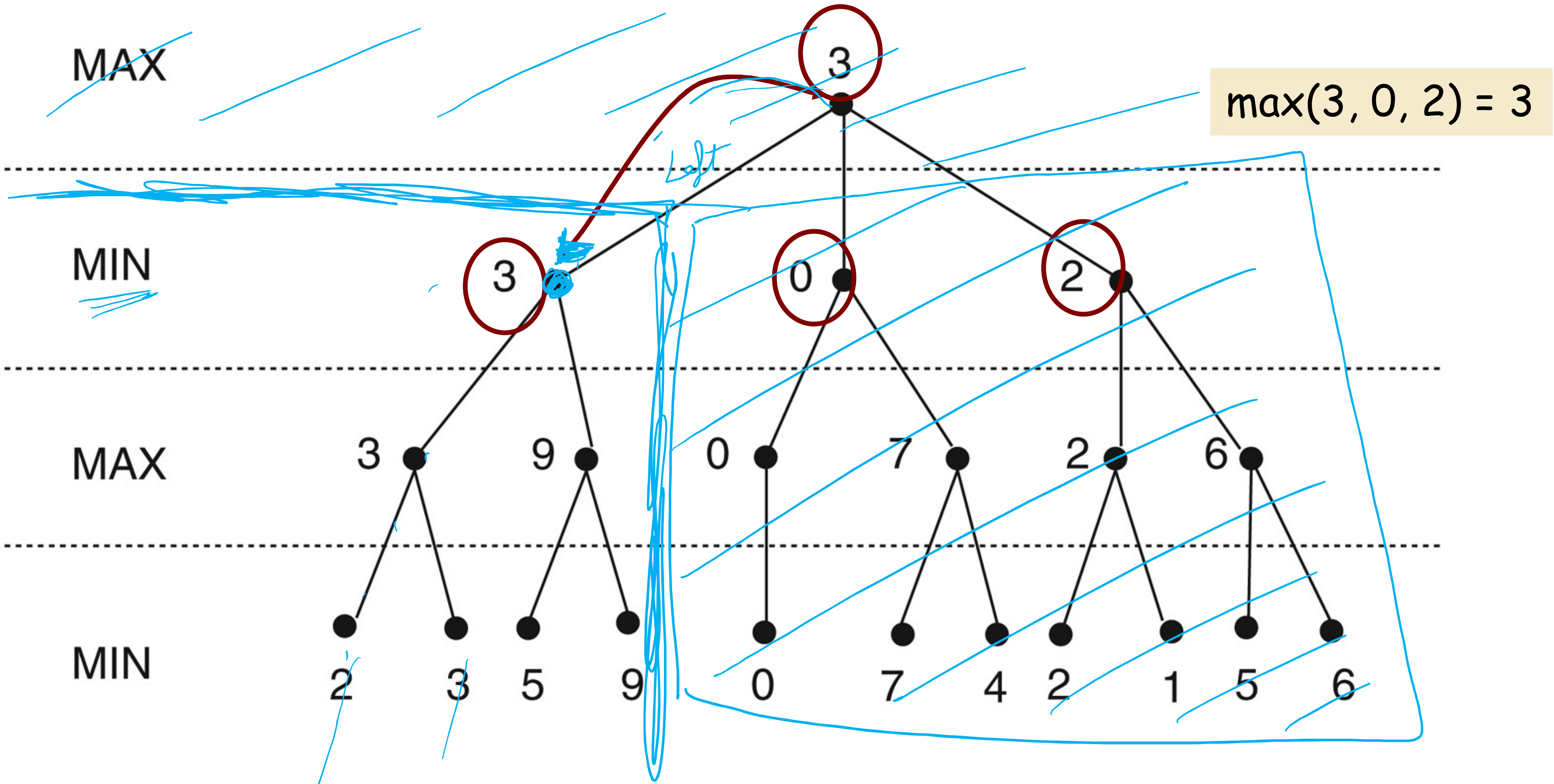


min(3,9) = 3
min(0,7) = 0
min(2,6) = 2

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

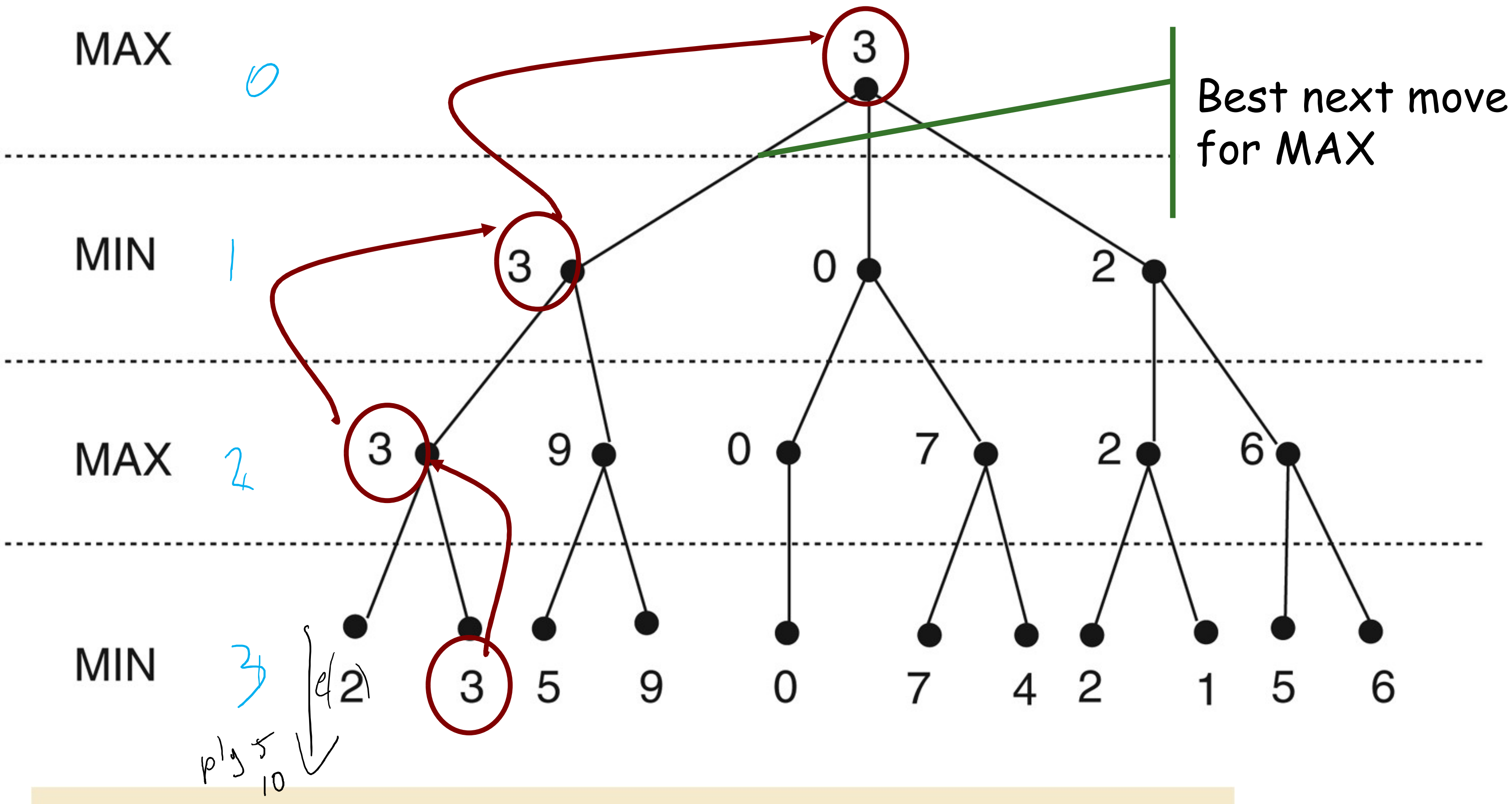


max(3, 0, 2) = 3

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



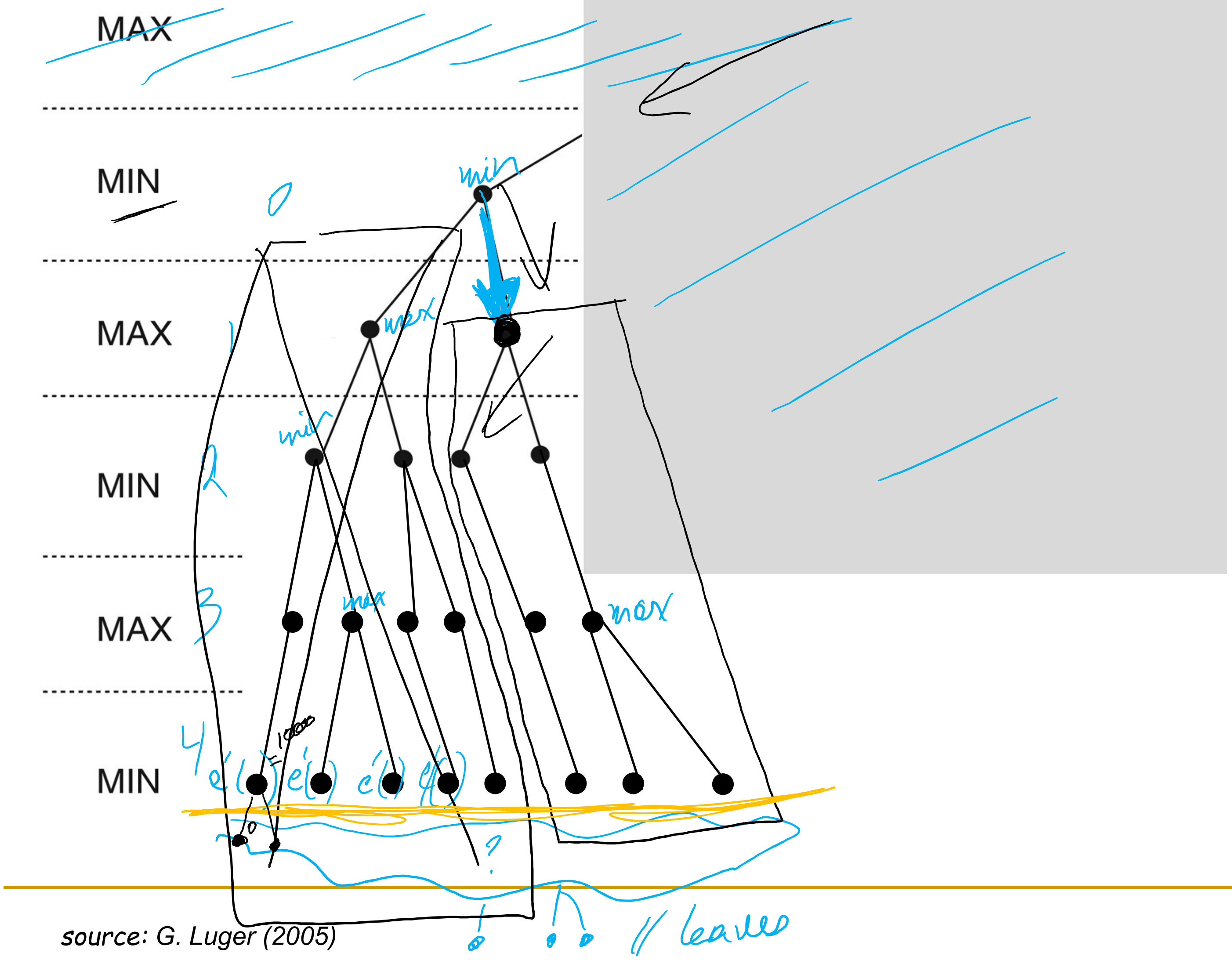
Best next move for MAX

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

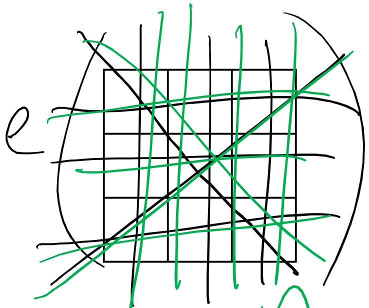


source: G. Luger (2005)

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

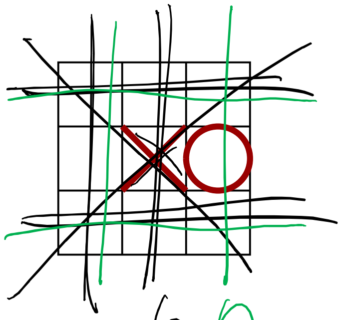
- assume MAX plays X
- possible *e(n)*

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



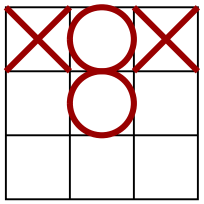
$e(n) = 8 – 8 = 0$

max min



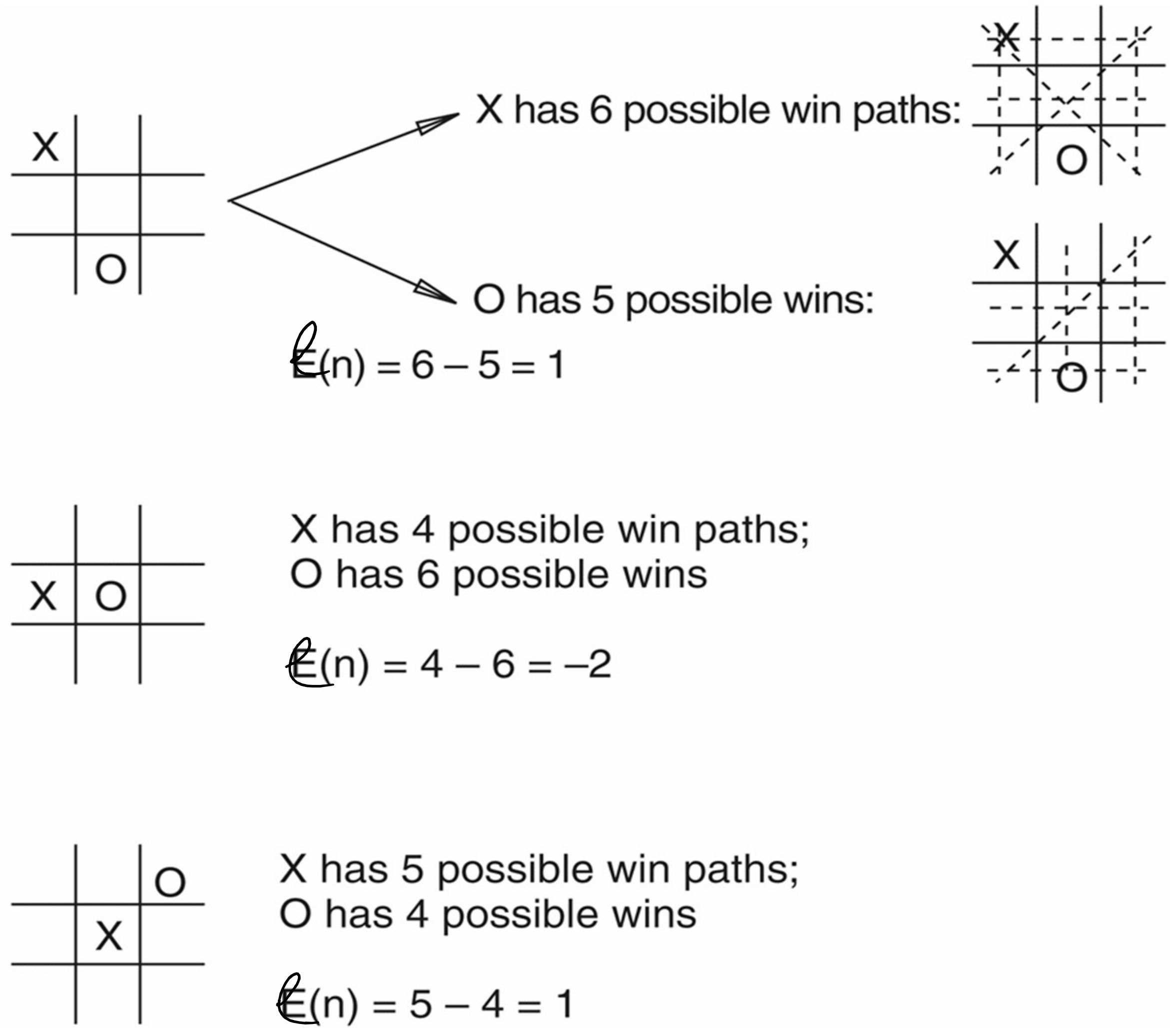
$e(n) = 6 – 4 = 2$

max min



$e(n) = 3 – 3 = 0$

More examples…



X has 6 possible win paths:

O has 5 possible wins:

~~E~~ℓ(n) = 6 – 5 = 1

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

~~E~~ℓ(n) = 4 – 6 = –2

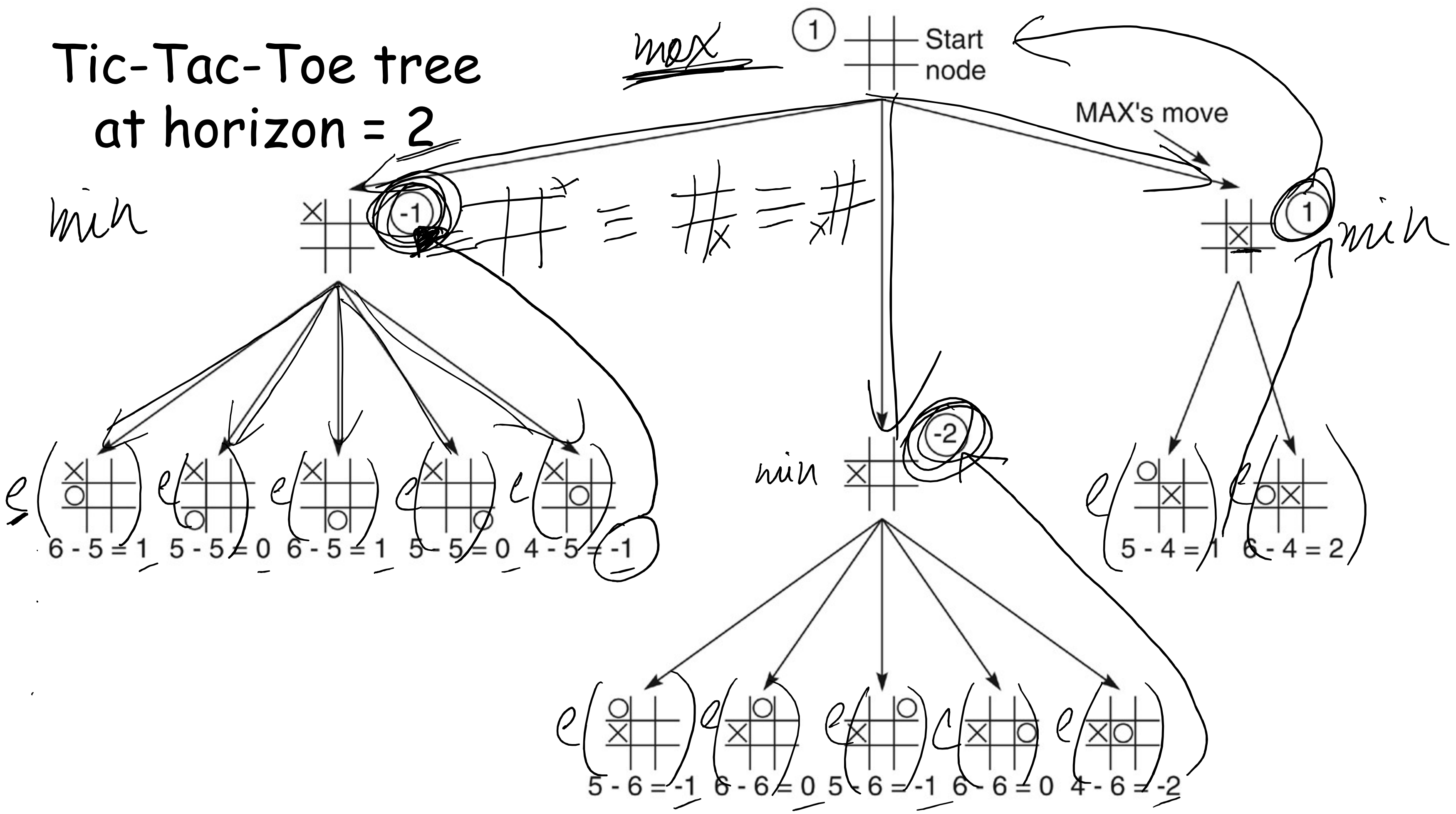
X has 5 possible win paths;
O has 4 possible wins

~~E~~ℓ(n) = 5 – 4 = 1

source: G. Luger (2005)

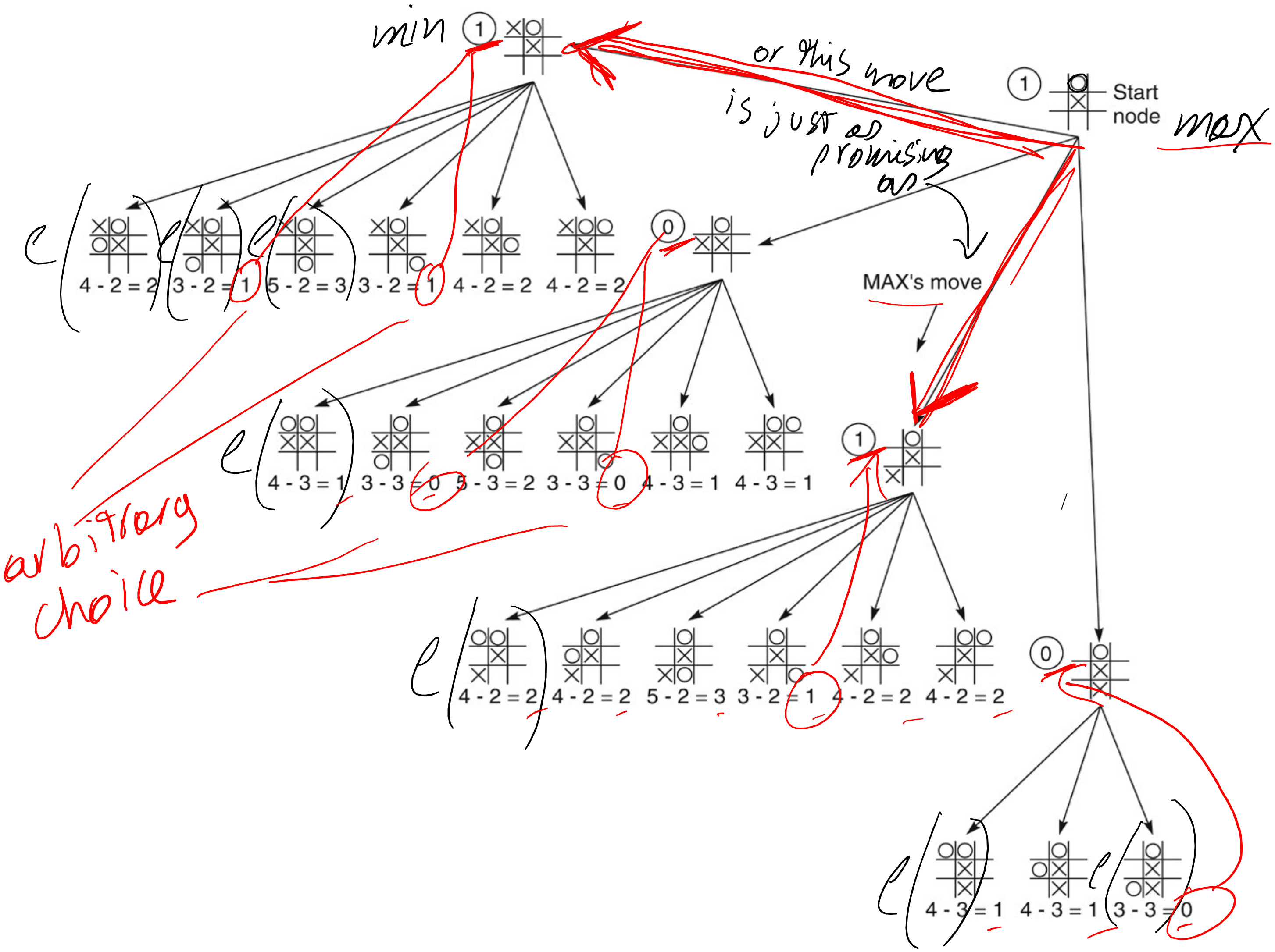
2-ply Minimax for Opening Move

Tic-Tac-Toe tree
at horizon = 2



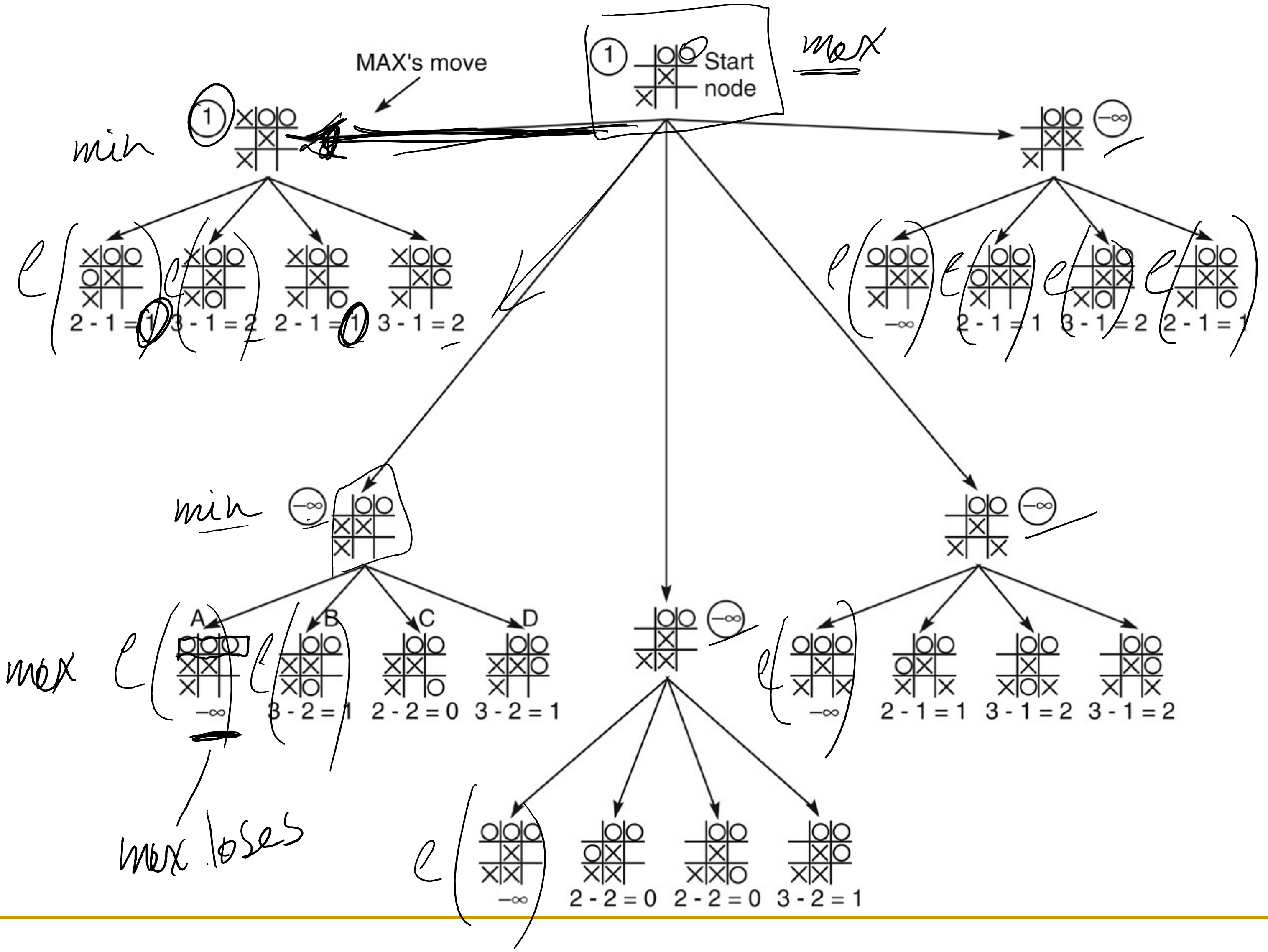
source: G. Luger (2005)

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
  3. Other Adversarial Search
     1. Multiplayer Games
     2. Stochastic Games
     3. Monte Carlo Tree Search

Up Next

- Adversarial Search
  1. Minimax
  2. Alpha-Beta Pruning
  3. Other Adversarial Search
     1. Multiplayer Games
     2. Stochastic Games
     3. Monte Carlo Tree Search