Adversarial Search

Prof. Jacques Savoy
University of Neuchatel

Chapter 6

Some slides from Norvig & Russell

Outline

- Introduction
- Minimax Scheme
- α-β pruning
- Game with random aspect

Why considering games?

- Game is one of the favorite example in AI
  - task well structured
  - intellectual challenge
  - abstraction required
  - performance measure
  - no need for a large amount of knowledge
- Not all games are adapted for AI
  (mainly *zero-sum game with perfect information*)

Ways to solve…

1. Random choice…
2. Analysis, strategy, tactics to select the best move
   How to do? How to combine?
3. Build *if () then* rules. How to evaluate the current board?
   At the end: Not a strong player.
4. Look ahead one level and evaluate…
   Which is the best board for me? Which features are needed? How to combine them? Linear (find the best)?
5. Explore all possibilities?
   Too huge to explore the full space? b=? m=? $O(b^m)$
6. Look ahead *k* levels and evaluate…

Games vs. search problems

- Start with a single game
- Limited to two players (one is MIN, the other MAX)
- We have
  - an initial state
  - state space
  - a function to define the successors
  - test for the end
  - an evaluation function (e.g., 1 = MAX wins, -1 MIN wins)

Games vs. search problems

- "Unpredictable" opponent → specifying a move for every possible opponent reply.
- Time limits → unlikely to find goal, must approximate.
- All information is known and visible.
- With the chess,
  - We have a branching factor *b* = 35
  - We usually have *m/2* =50 game rounds.
  - The space is around $35^{(50+50)} = 35^{100} = 10^{154}$ (Shannon's number)
    number of atoms = 10^{80}
    number of nanosec. (history of universe) = 10^{106}
- For the GO, *b* = 200 with a space is around 10^{350}

Games and AI

- J. Schaeffer writes *Chinook* (*checkers*) and wins the US Open in 1992, 1994 and in 1995 it's world champion.
- Othello: *Logistello* (M. Buro) beats in 1997 the world champion Takeshi Murakami.
- Tesauro, Backgammon in the top 3, in 1992.
- Hsu, Campbell *et al*, *Deeper Blue* vs. Kasparov 1997 (3.5 vs. 2.5).
- *Hydra* vs. M. Adams (7th best chess player) 2005 (5.0 vs. 0).
- Go: we solve the problem; deep learning & reinforcement
  October 2015: AlphaGo beats Fan Hui (European champion)
  March 2016: AlphaGo beats Lee Sedol (World champion)

Type of games

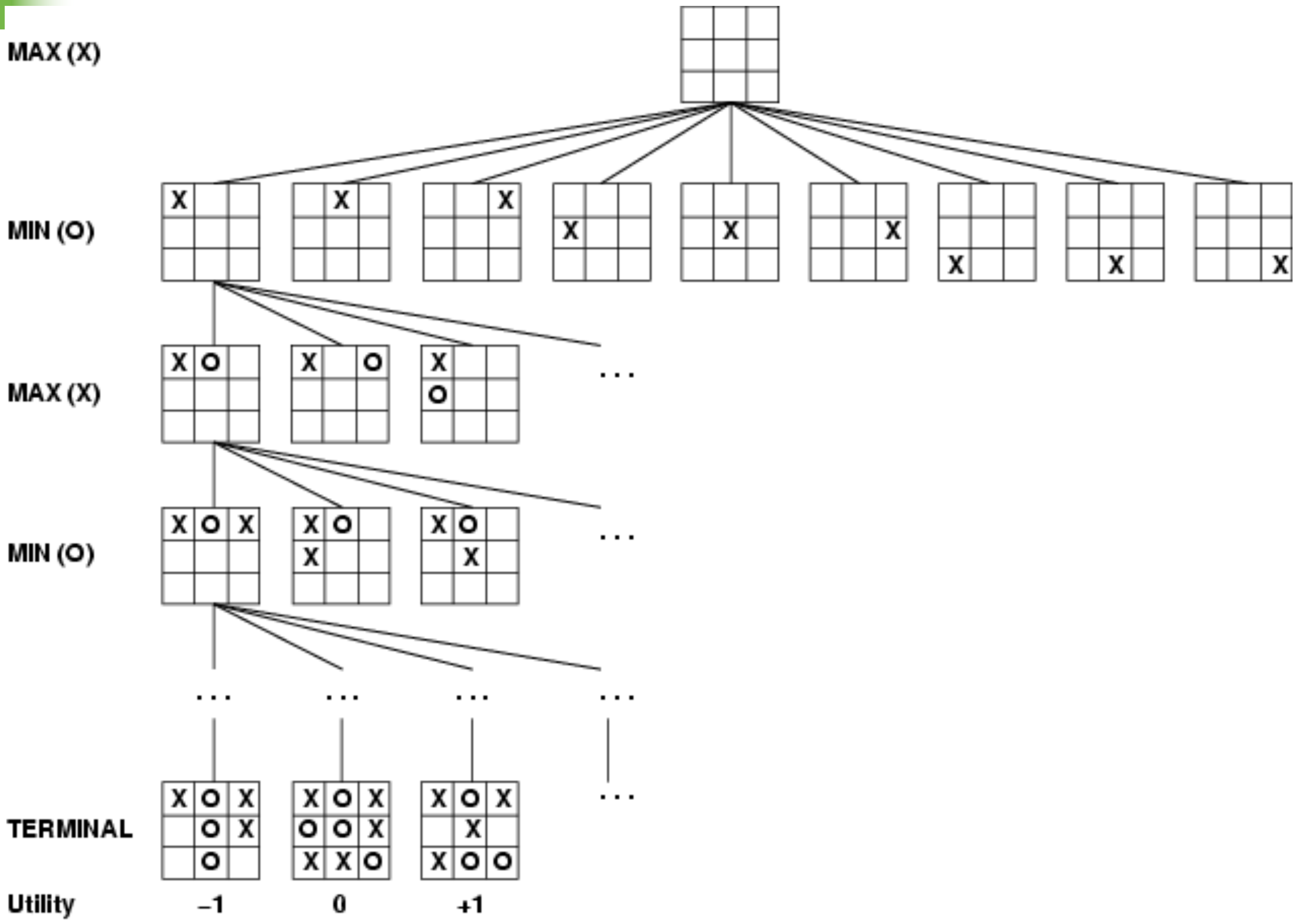
- Is all information available?
- Do we need to take account for a random aspect?

	determinist	with chance
information complete	chess, Othello, checkers	monopoly, backgammon
partial information		bridge, poker, scrabble

Game Tree

- Zero-sum game with two players.
- One is MAX (wants to maximize the utility).
- Other is MIN (wants to minimize the utility).
- We may have the limits
  1 = MAX wins
  -1 = MIN wins
  0 = tie
- If the game is in a final state, only these tree values are possible.

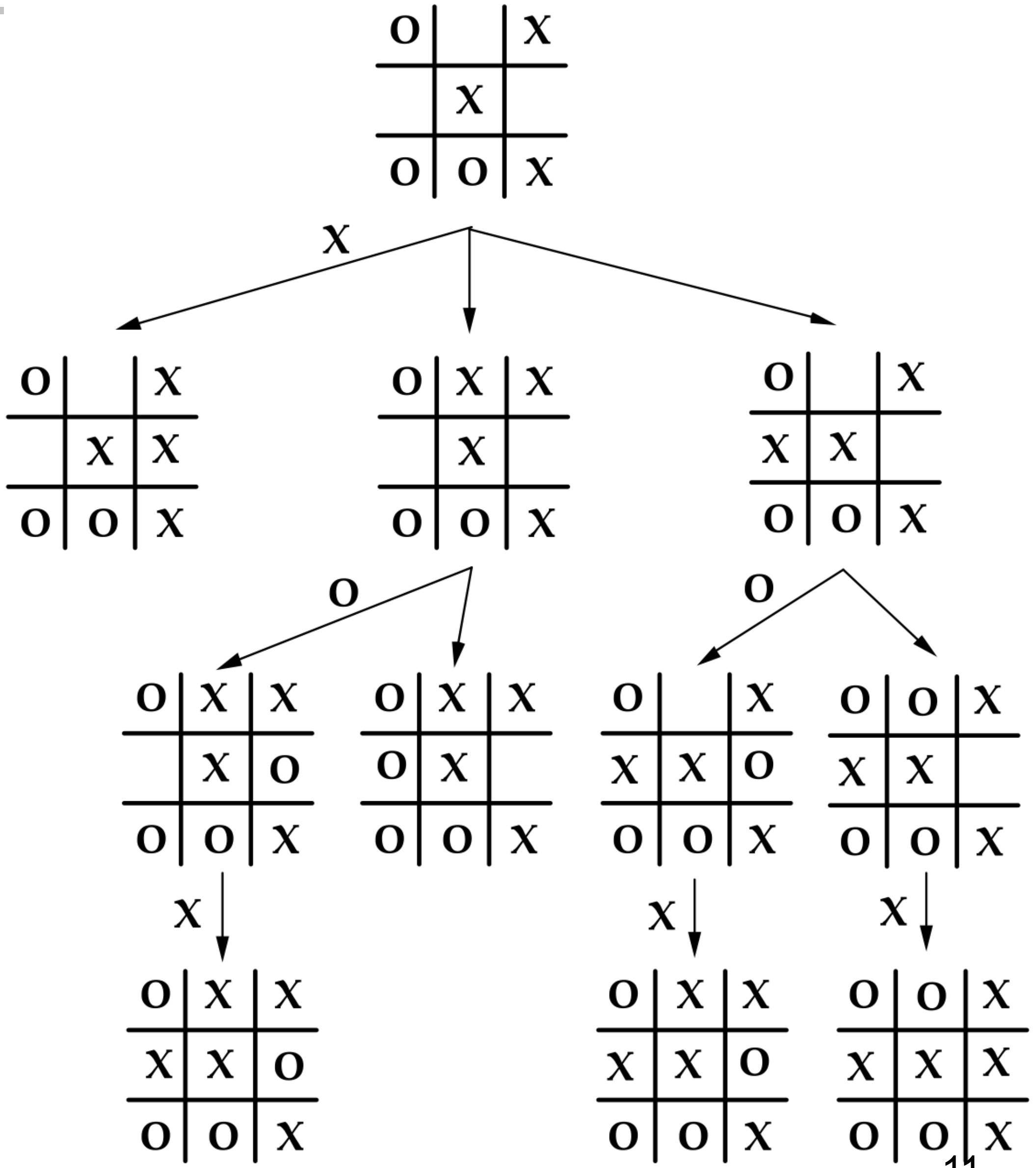
Game Tree



Game Tree

How to determine the value of the leaves and of the internal nodes?

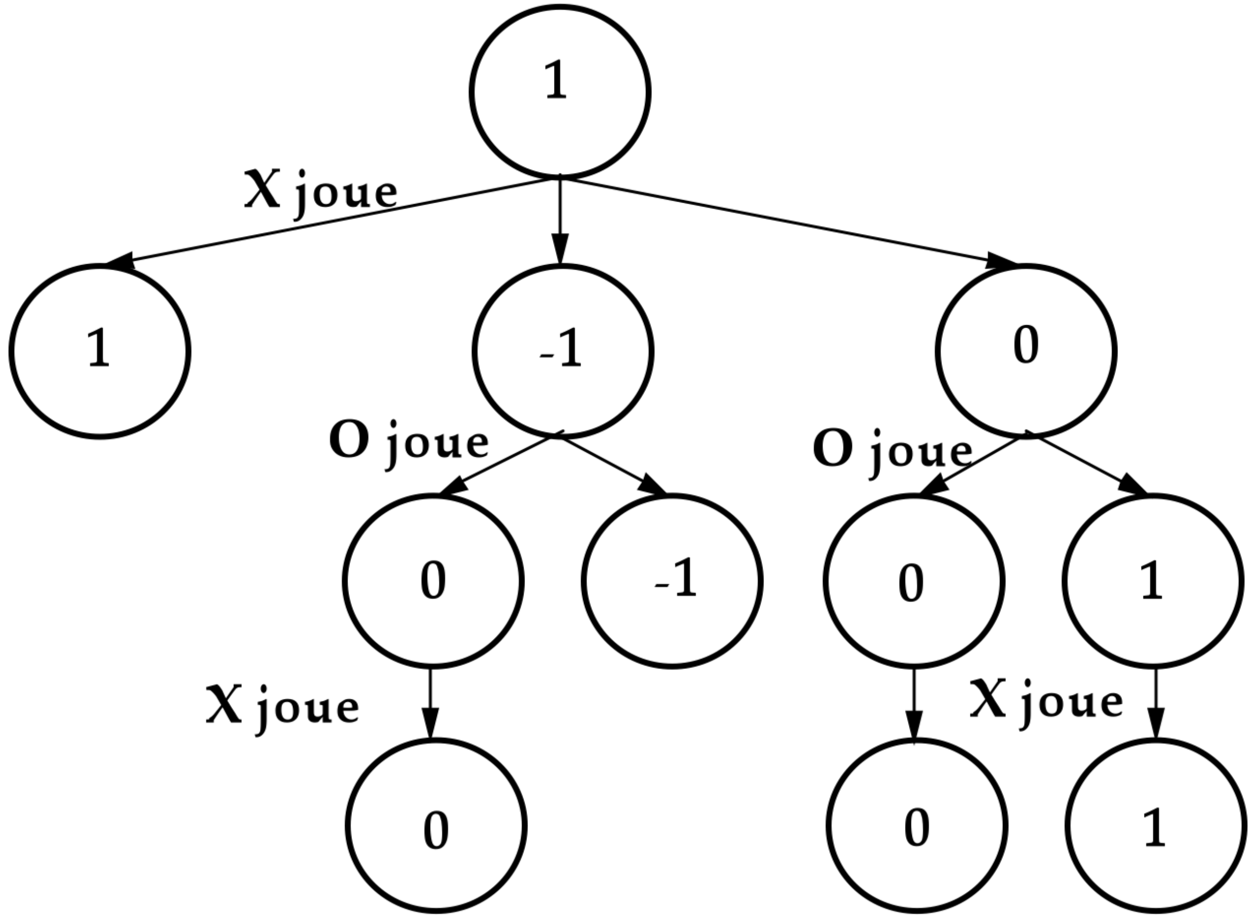
We need to consider who must play. If it is MAX, we need to maximize the utility. If it is MIN, we need to minimize the utility function.



Game Tree

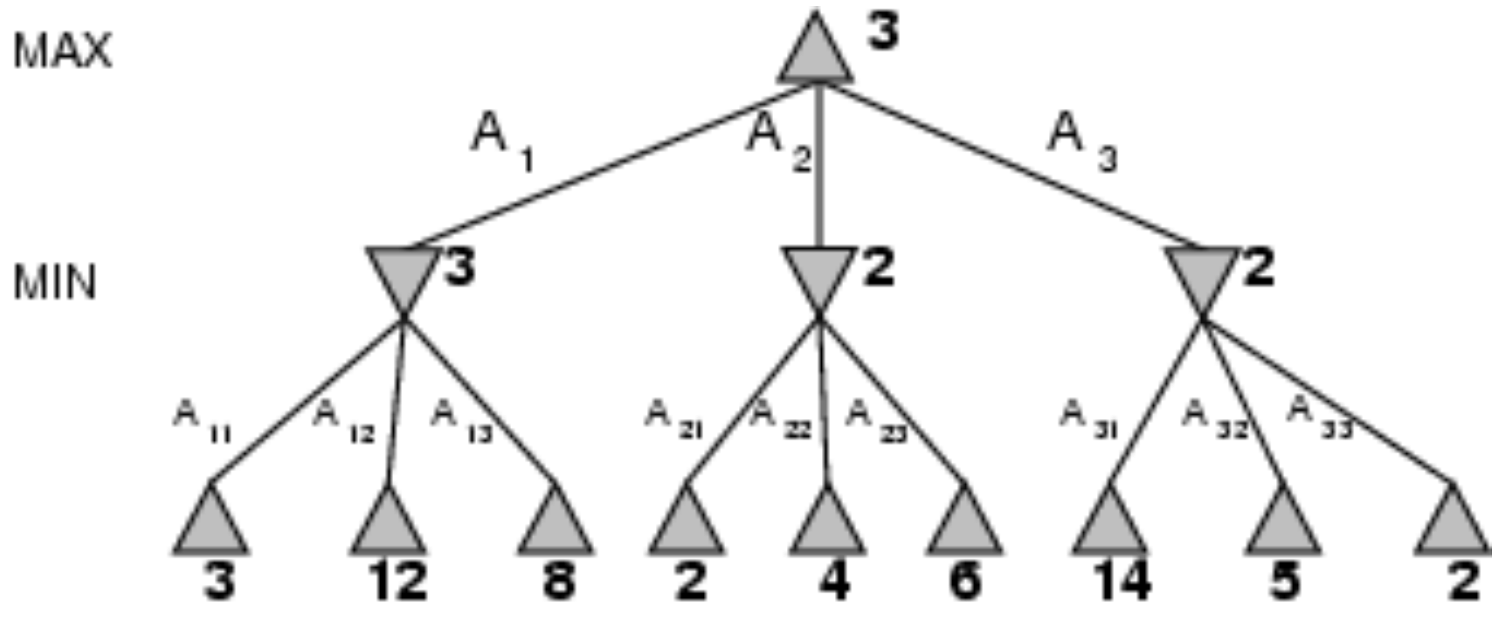
From previous example

with X = MAX
and O = MIN

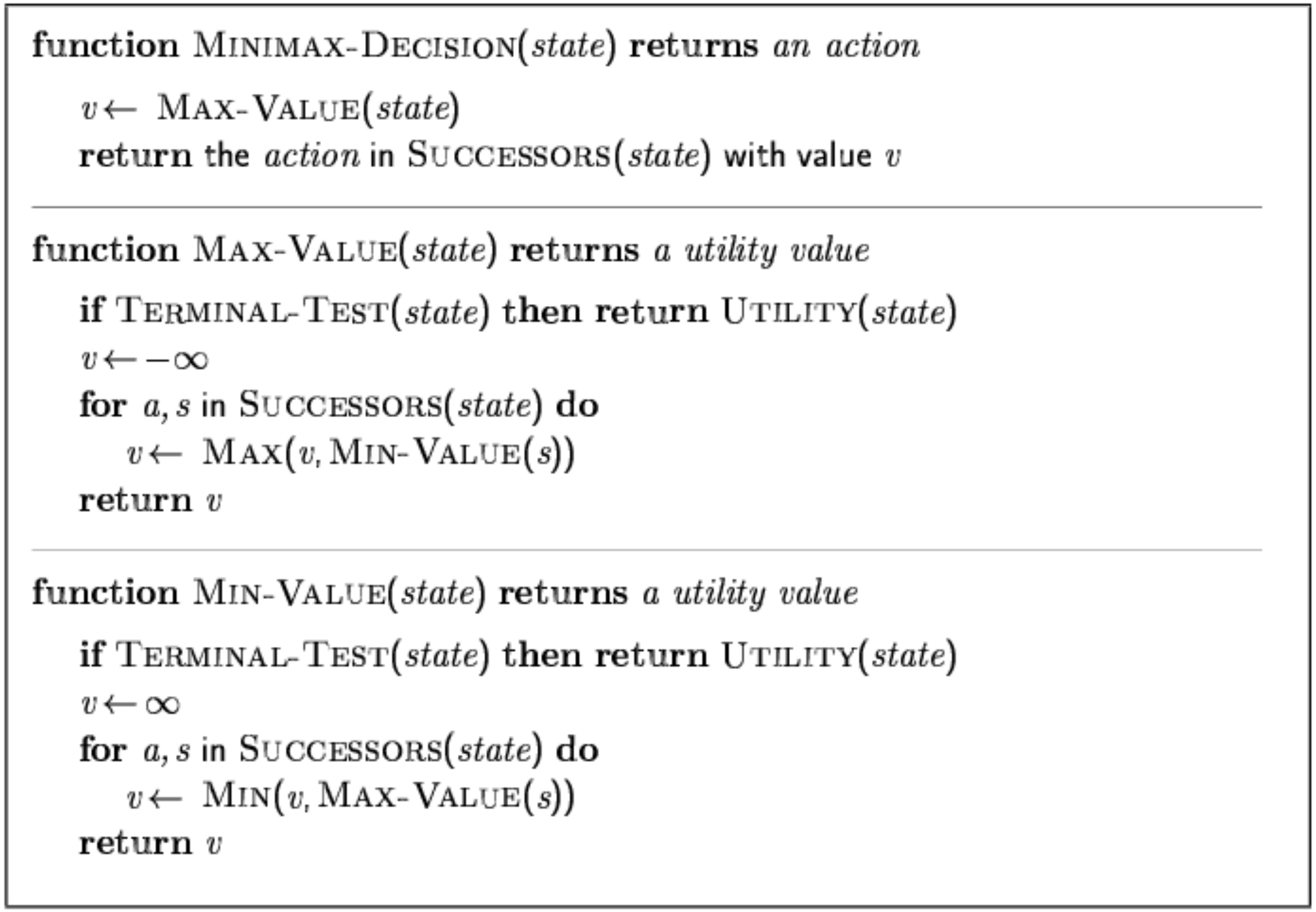


Minimax

- Perfect play for deterministic games
- Idea: choose move to position with highest minimax value
  = best achievable payoff against best play
- E.g., 2-ply game:



Minimax algorithm



```
function MINIMAX-DECISION(state) returns an action
    $v \leftarrow$ MAX-VALUE(state)
    return the action in SUCCESSORS(state) with value $v$

function MAX-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    $v \leftarrow -\infty$
    for $a, s$ in SUCCESSORS(state) do
        $v \leftarrow$ MAX($v$, MIN-VALUE($s$))
    return $v$

function MIN-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    $v \leftarrow \infty$
    for $a, s$ in SUCCESSORS(state) do
        $v \leftarrow$ MIN($v$, MAX-VALUE($s$))
    return $v$
```

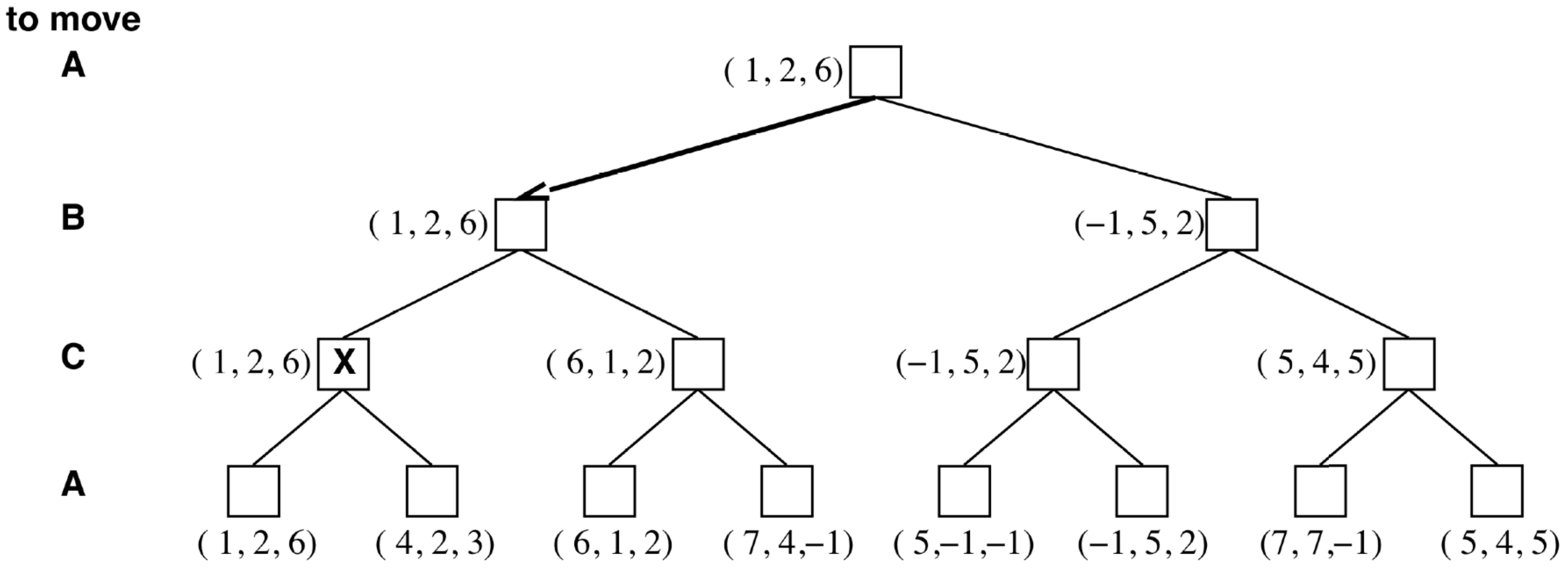
Properties of Minimax

- This is a search scheme in a game space
- Complete? Yes (if tree is finite).
- Optimal? Yes (against an optimal opponent).
- Time complexity? O(b^m).
- Space complexity? O(bm) (depth-first exploration).

- For "reasonable" games
  → exact solution completely infeasible.
- For chess, $b \approx 35$, $m \approx 100$: No

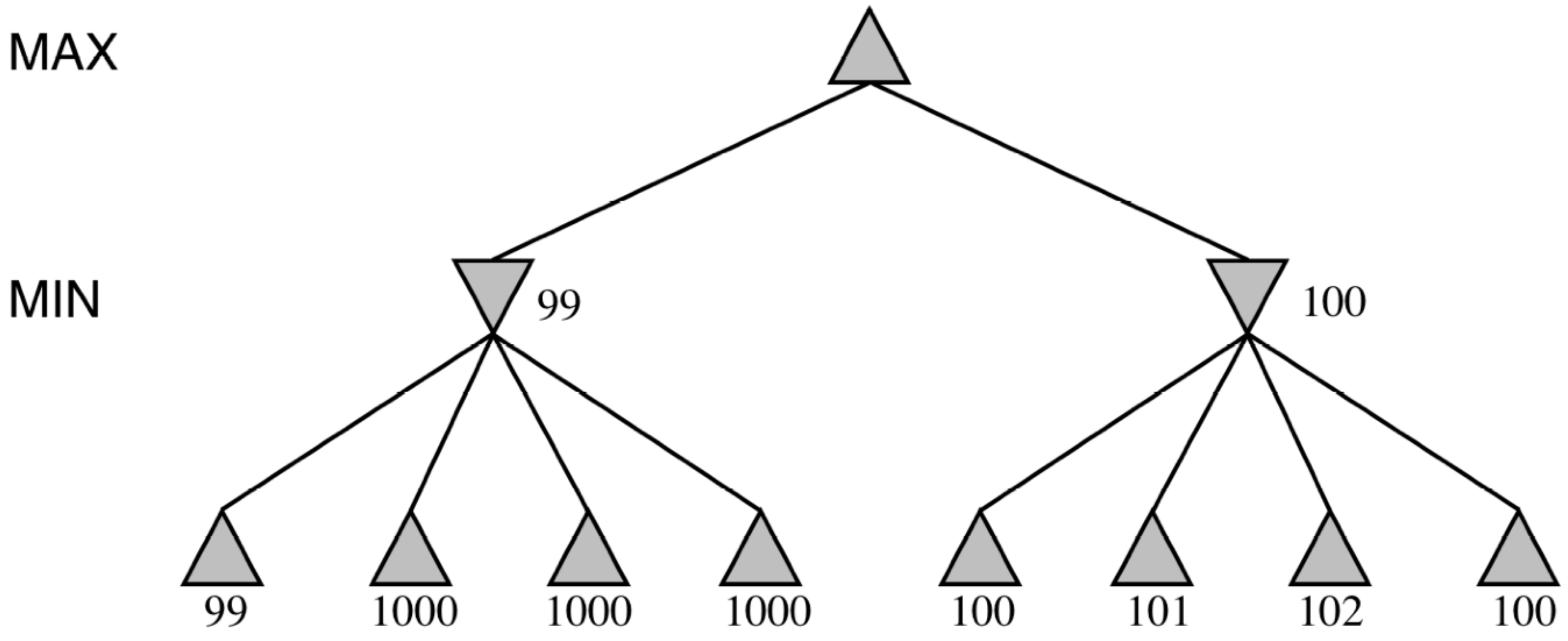
With three players

Use a vector (instead of a Max/Min single value)



Reason vs. Gain

The strict application of the Minimax rule is sometimes questionable.   Cost-benefit analysis.



Properties of Minimax

- But we cannot explore the tree completely.
- Search into the tree until a given level *m* is achieved.
- Compute the utility for each position at the last level.
- Return the values according to the minimax algorithm.

- Need a good heuristic to evaluate the game at a given position (in the *m* level).

Evaluation functions

- For chess, typically linear weighted sum of features

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

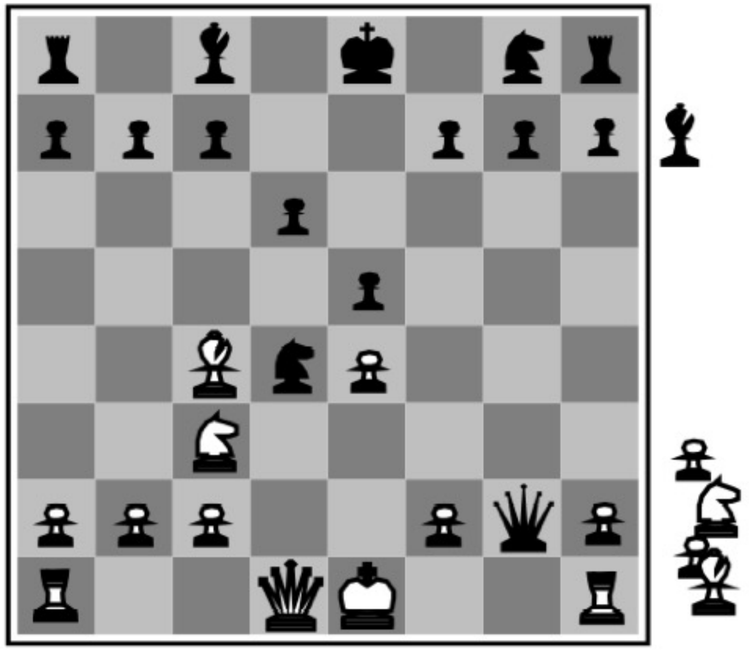
- e.g., $w_1 = 9$ with

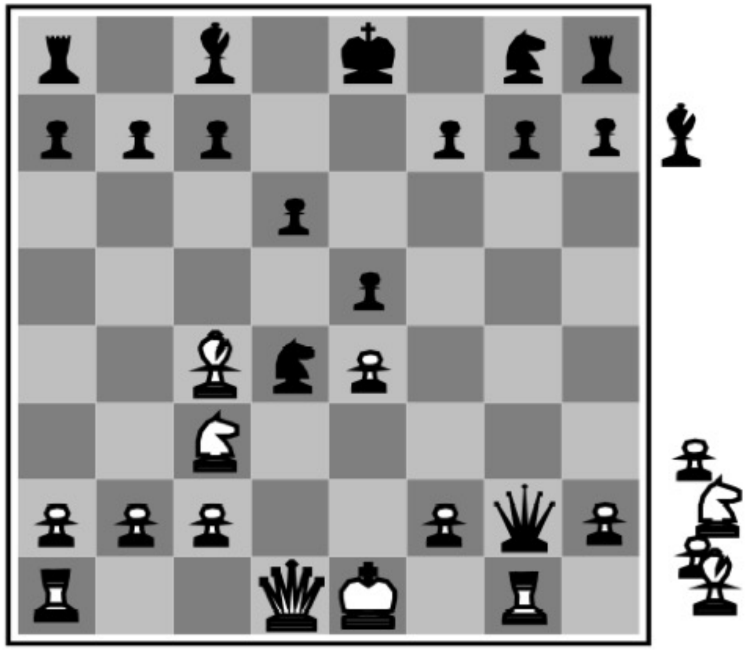
  $f_1(s)$ = (number of white queens) –
  (number of black queens), +
  etc.

- Need to take account for both the number and type of pieces and the position of the pieces.
- Need to avoid cycle.

Evaluation functions

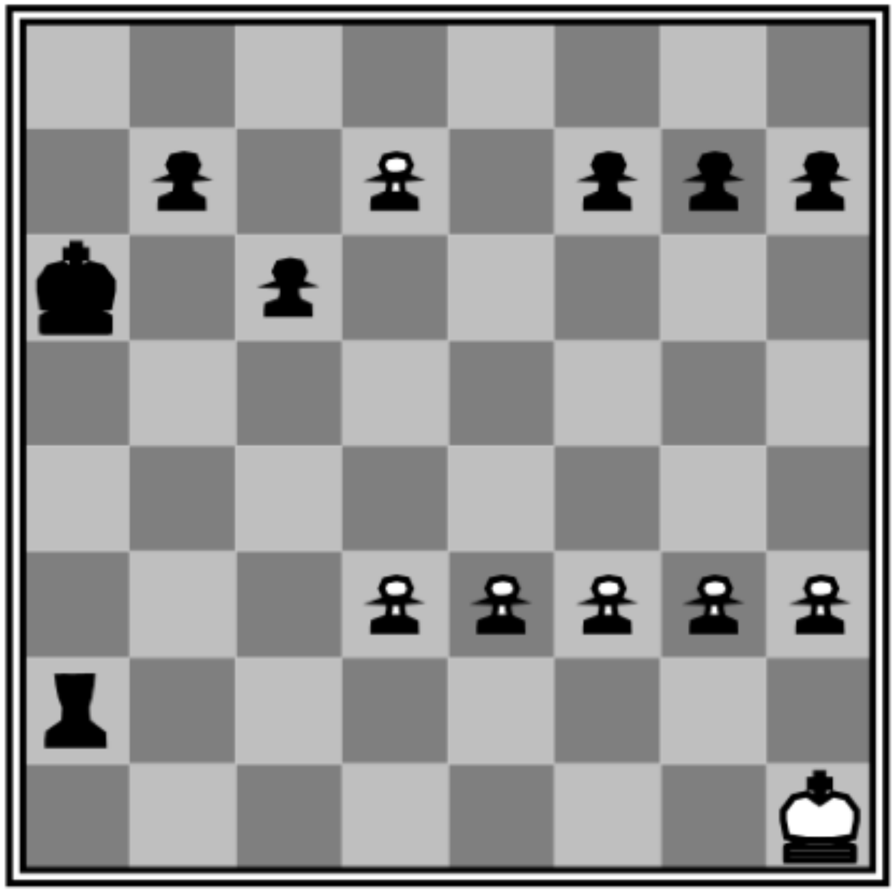
- In a) Black is in advance
- In b) White in advance


(a) White to move


(b) White to move

Evaluation functions

Black seems in advance... but we will have a white queen in a few moves...


Black to move

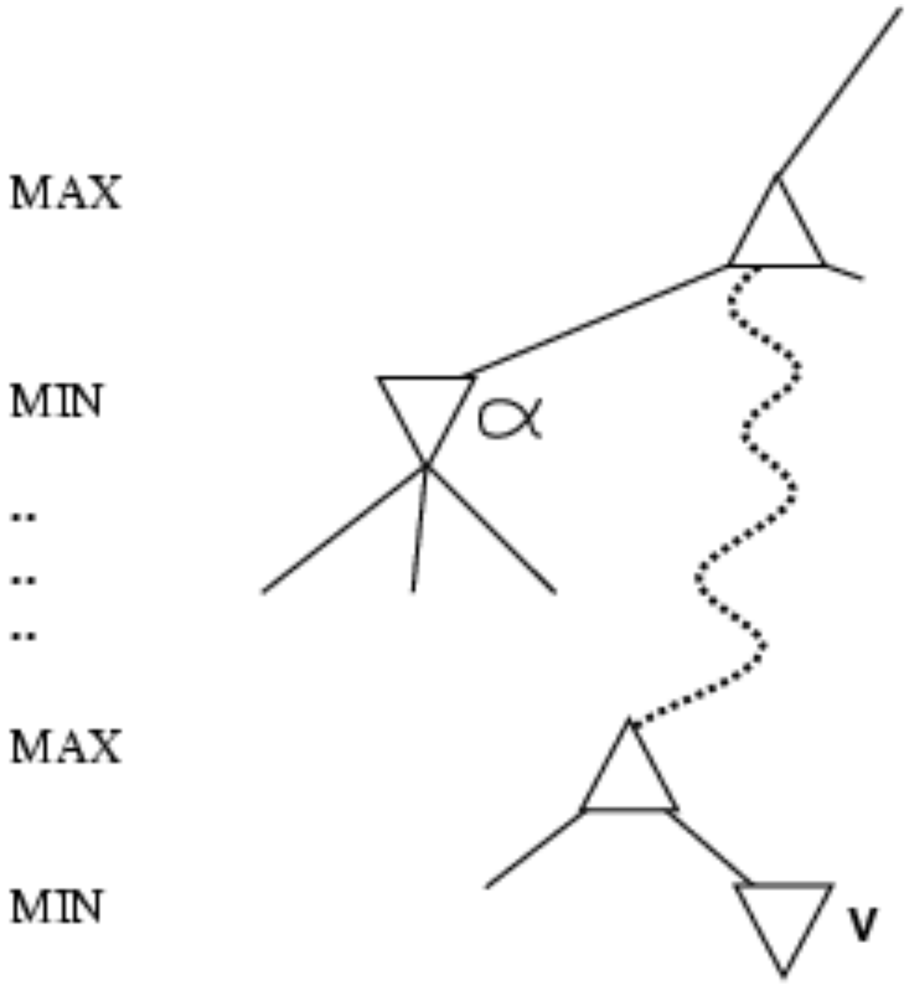
α-β Pruning

- The space space is too large O(b^m), thus need to reduce the search space by pruning.
- The idea: do not generate other solutions as soon as it is evident that the path will not be selected.
- For each Max node: keep trace of the α-value
  = value of the best successor found until now.
- For each Min node: keep trace of the β-value
  = value of the worst successor found until now.
- At the init level α-value = $-\infty$; β-value = $+\infty$
- Rule: Stop the search if:
  - Max node: if its α-value $\geq$ β-value of its parent.
  - Min node: if its β-value $\leq$ α-value of its parent.

Why is it called α-β?

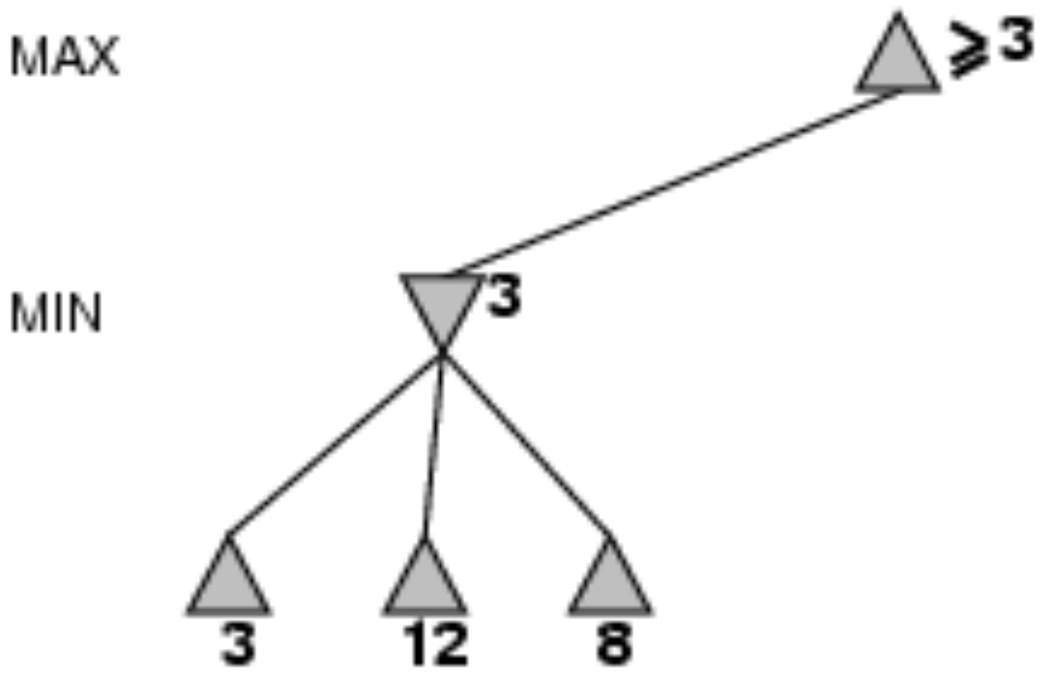
- α is the value of the best (i.e., highest-value) choice found so far at any choice point along the path for *max.*
- If *v* is worse than α, *max* will avoid it.

  → prune that branch
- Define β similarly for *min.*



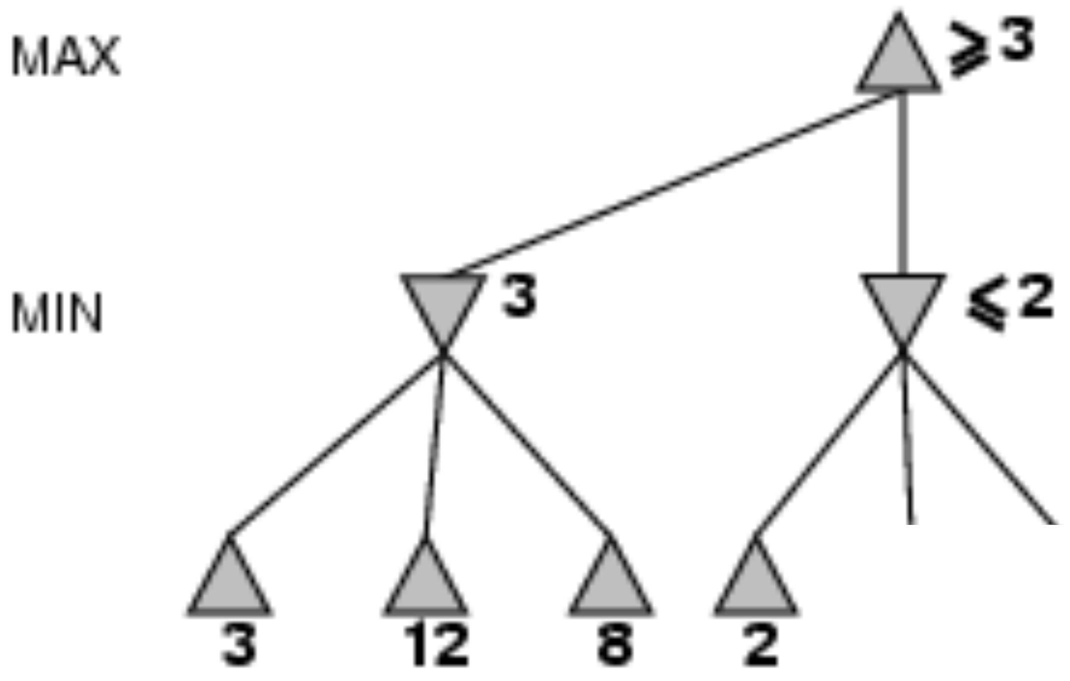
α-β Pruning Example

- We start to explore the possible paths



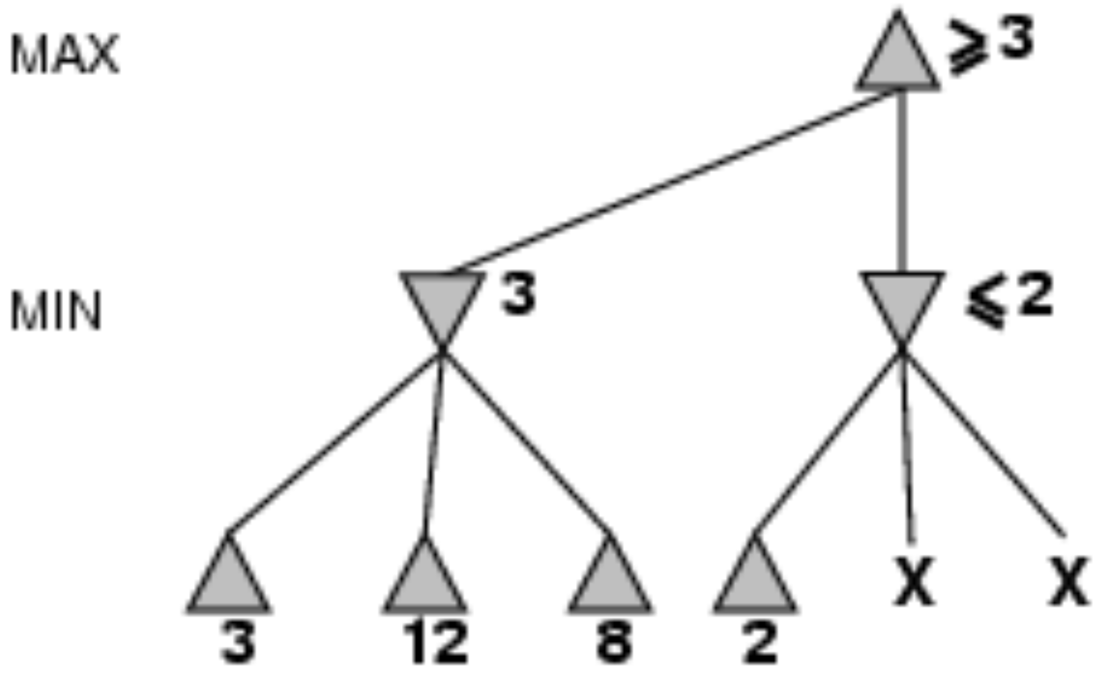
α-β Pruning Example

- After exploring the first child of the second note, does it make sense to continue?



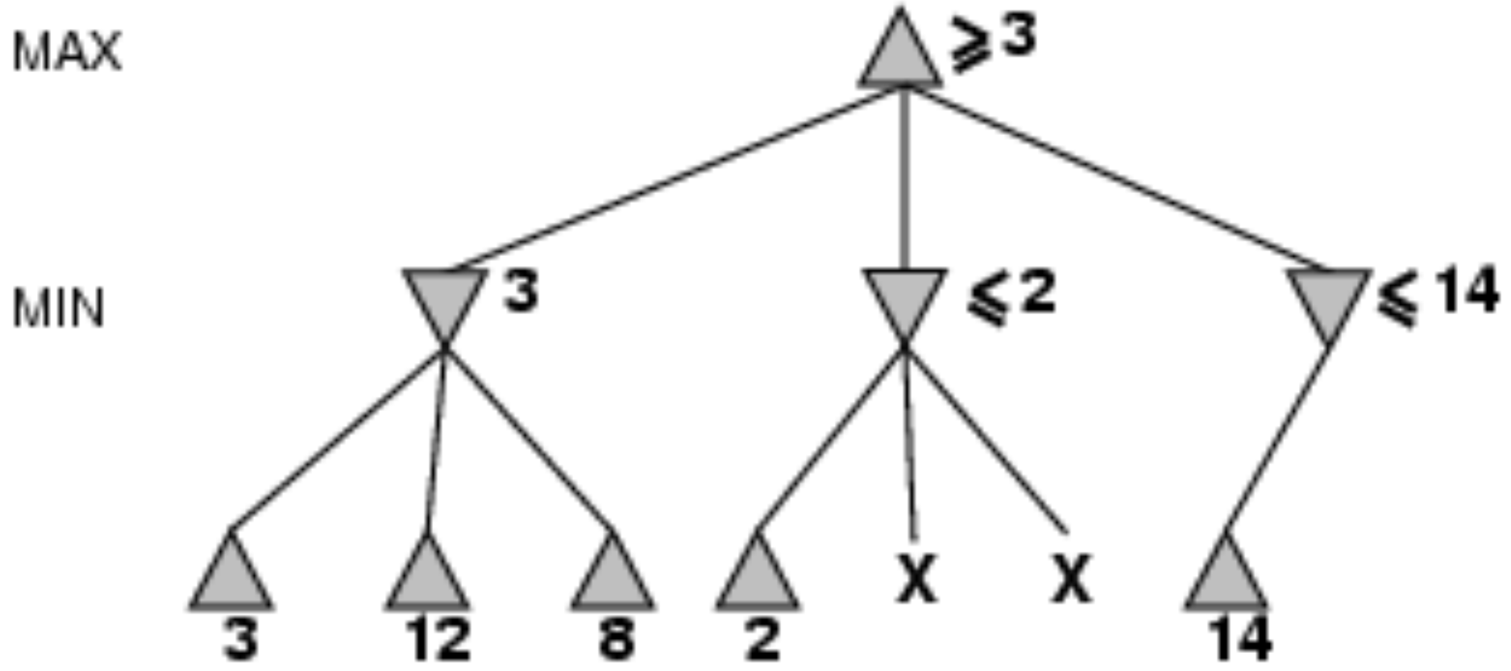
α-β Pruning Example

- Yes for Min, but in the upper level we have Max.



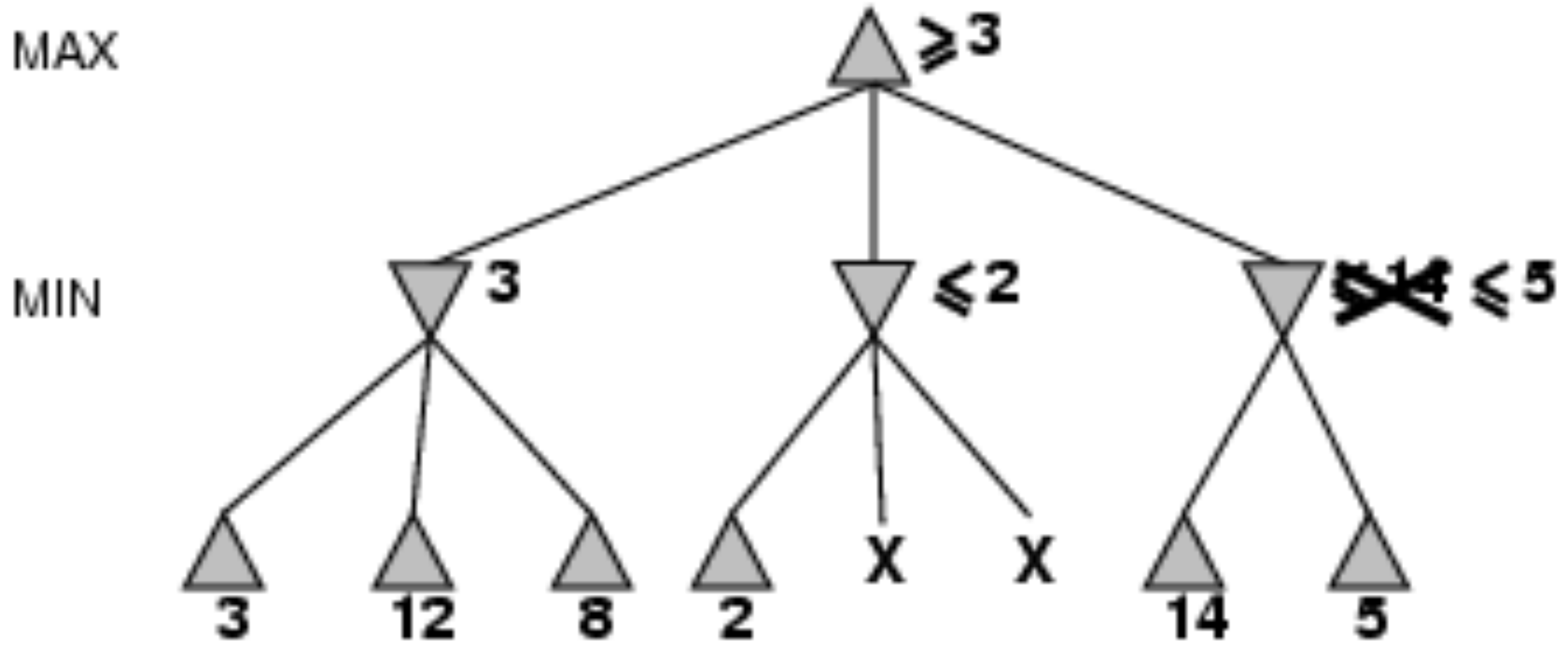
α-β Pruning Example

- The third possible move for Max.



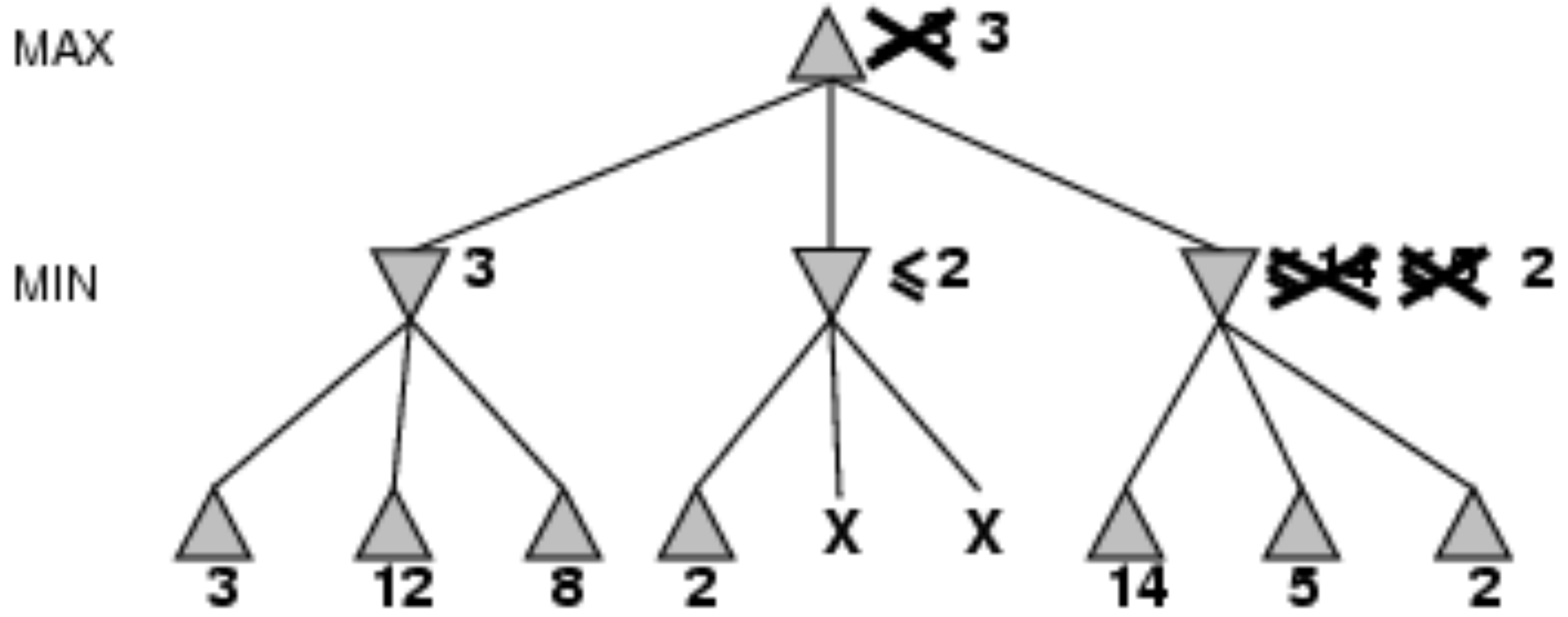
α-β Pruning Example

- The third possible move for Max.
- Do we continue to explore?



α-β Pruning Example

- The third possible move for Max.
- And now do we continue to explore?

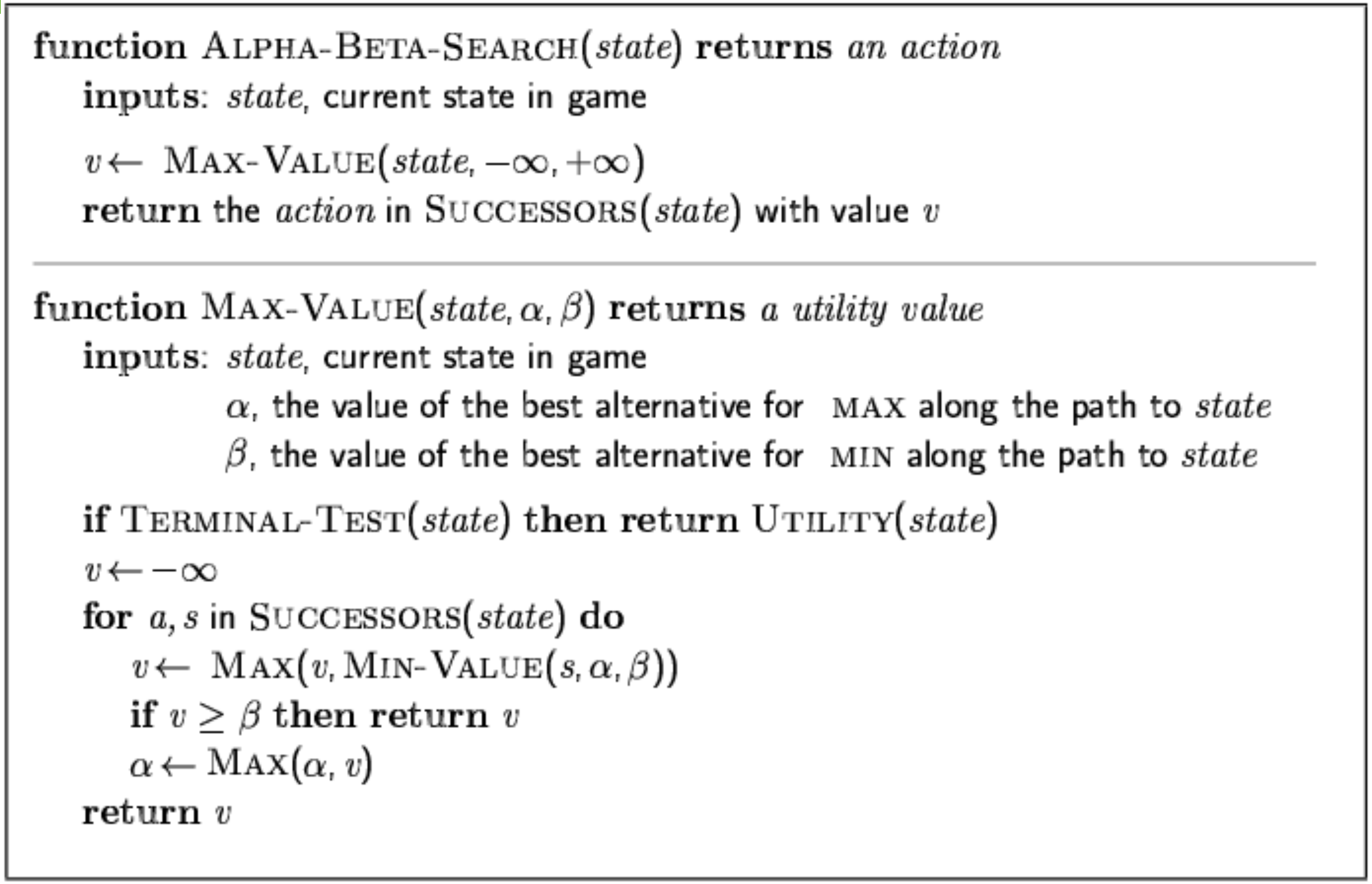


Properties of α-β

- Pruning does not affect final result.
- Good move ordering improves effectiveness of pruning.
- With "perfect ordering," time complexity = $O(2 \cdot b^{m/2})$

  → doubles the depth of search.
- A simple example of the value of reasoning about which computations are relevant (a form of metareasoning).
- And use iterative deepening.
- With transposition table (hash table with hash(board) storing already seen position).

The α-β algorithm



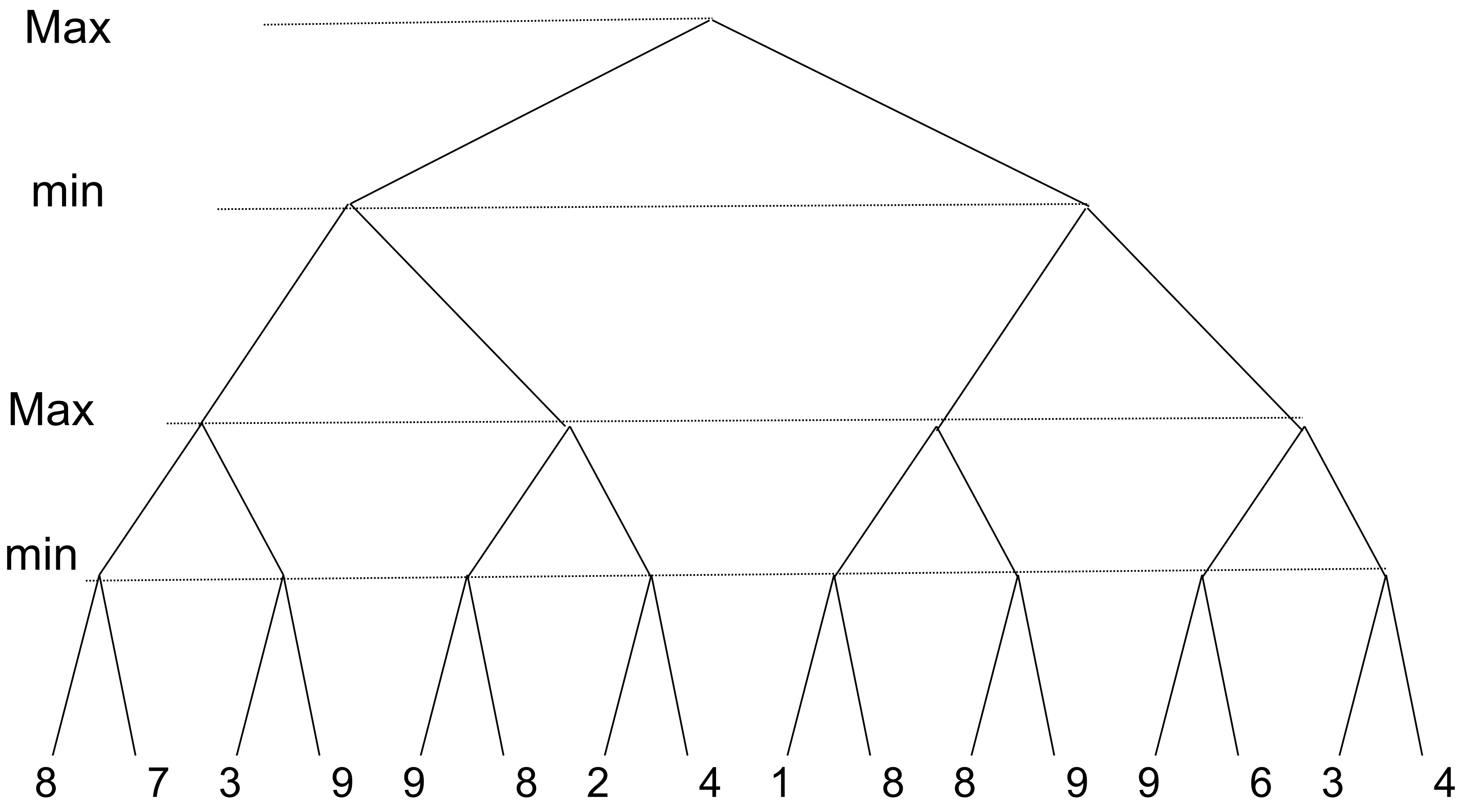
```
function ALPHA-BETA-SEARCH(state) returns an action
    inputs: state, current state in game

    $v \leftarrow$ MAX-VALUE(state, $-\infty$, $+\infty$)
    return the action in SUCCESSORS(state) with value $v$
```

```
function MAX-VALUE(state, $\alpha$, $\beta$) returns a utility value
    inputs: state, current state in game
            $\alpha$, the value of the best alternative for MAX along the path to state
            $\beta$, the value of the best alternative for MIN along the path to state

    if TERMINAL-TEST(state) then return UTILITY(state)
    $v \leftarrow -\infty$
    for $a, s$ in SUCCESSORS(state) do
        $v \leftarrow$ MAX($v$, MIN-VALUE($s, \alpha, \beta$))
        if $v \geq \beta$ then return $v$
        $\alpha \leftarrow$ MAX($\alpha, v$)
    return $v$
```

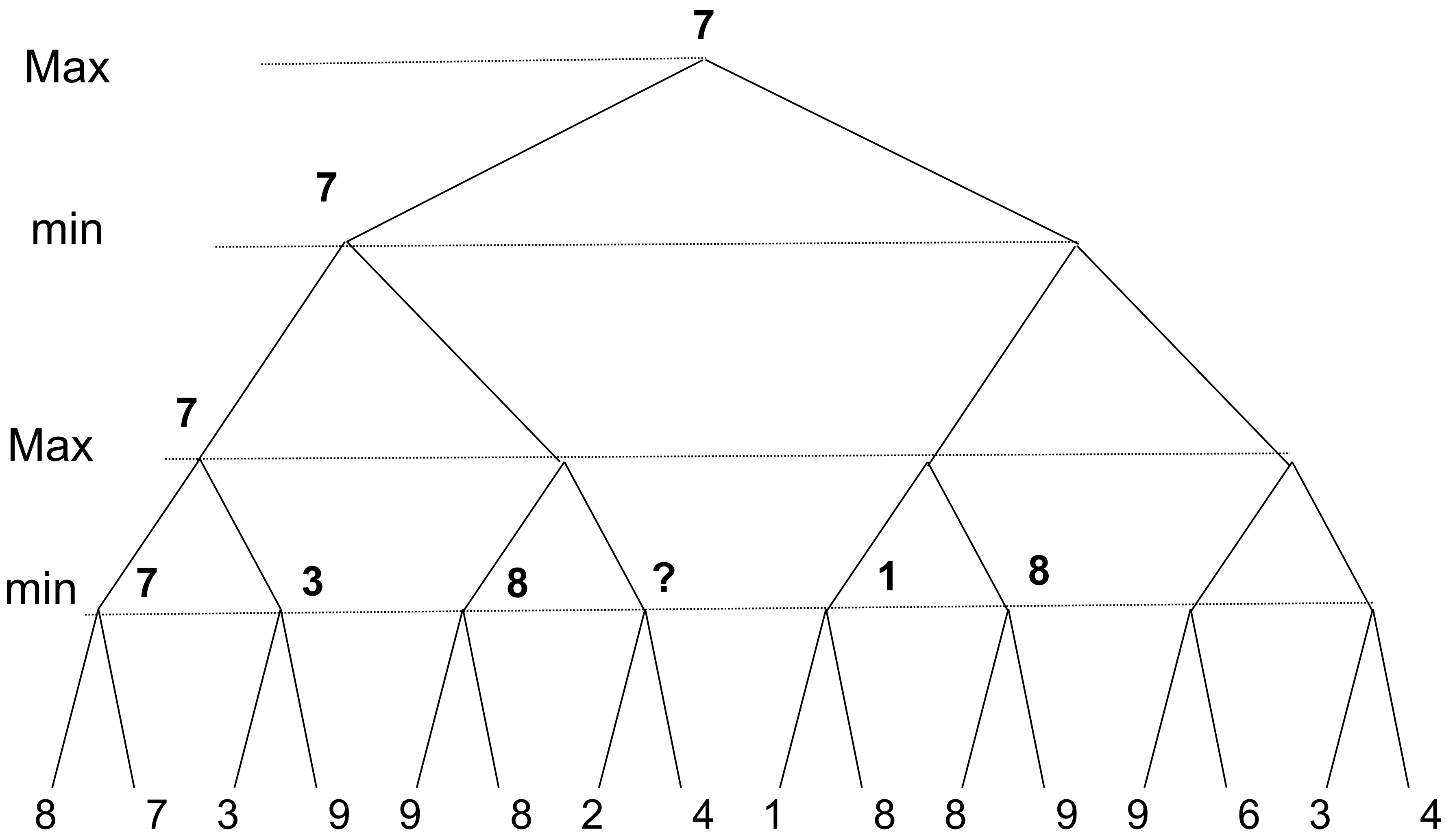
The α-β algorithm

```
function MIN-VALUE($state, \alpha, \beta$) returns a utility value
    inputs: state, current state in game
            $\alpha$, the value of the best alternative for MAX along the path to state
            $\beta$, the value of the best alternative for MIN along the path to state

    if TERMINAL-TEST(state) then return UTILITY(state)
    $v \leftarrow +\infty$
    for $a, s$ in SUCCESSORS(state) do
        $v \leftarrow$ MIN($v$, MAX-VALUE($s, \alpha, \beta$))
        if $v \leq \alpha$ then return $v$
        $\beta \leftarrow$ MIN($\beta, v$)
    return $v$
```

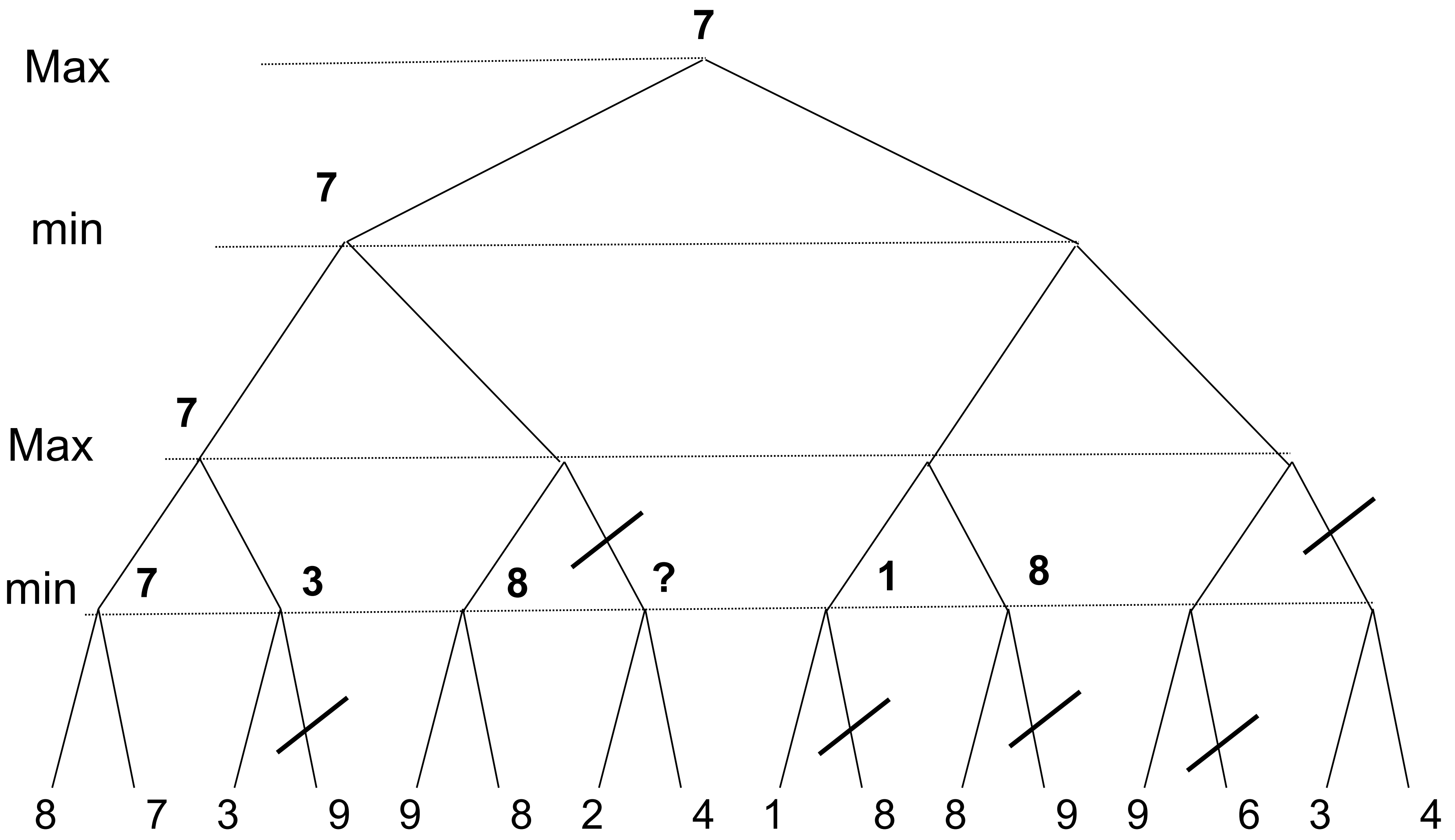
Example



Example



Example

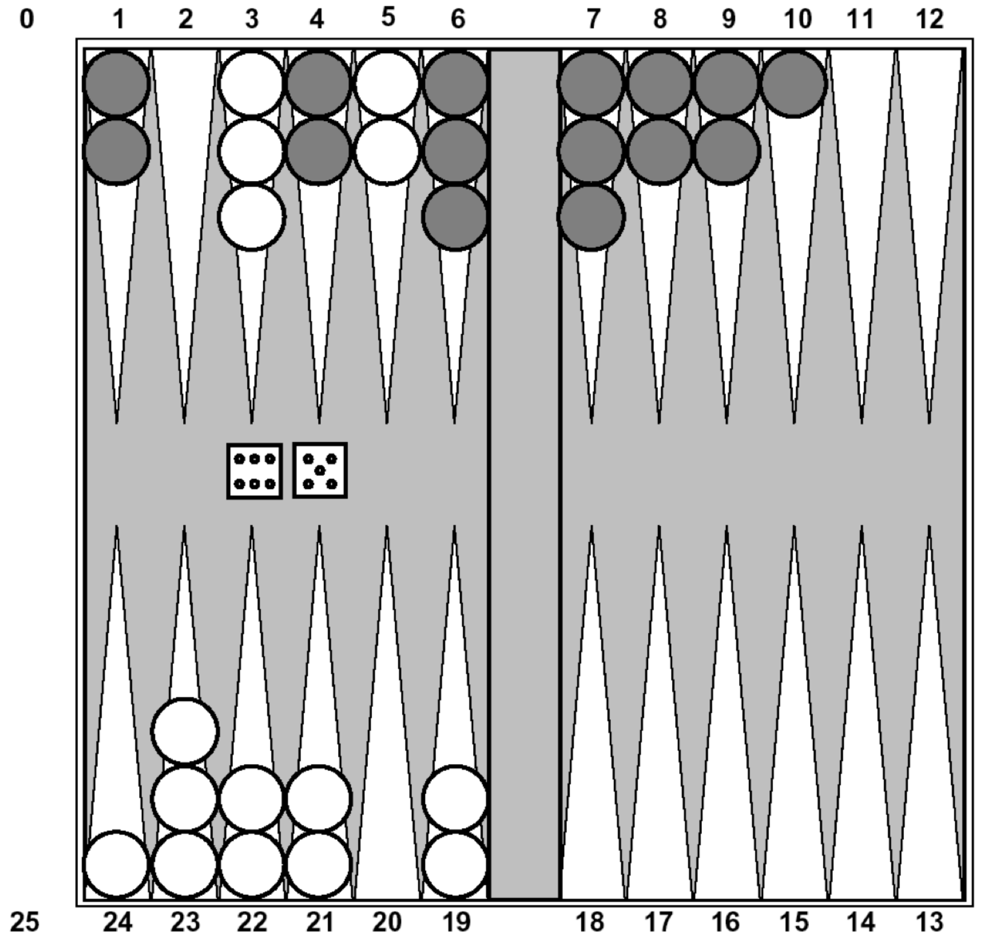


The α-β algorithm

```
function MIN-VALUE($state, \alpha, \beta$) returns a utility value
    inputs: state, current state in game
            $\alpha$, the value of the best alternative for MAX along the path to state
            $\beta$, the value of the best alternative for MIN along the path to state

    if TERMINAL-TEST(state) then return UTILITY(state)
    $v \leftarrow +\infty$
    for $a, s$ in SUCCESSORS(state) do
        $v \leftarrow$ MIN($v$, MAX-VALUE($s, \alpha, \beta$))
        if $v \leq \alpha$ then return $v$
        $\beta \leftarrow$ MIN($\beta, v$)
    return $v$
```

With random aspect

If the game includes a random aspect (dice)?



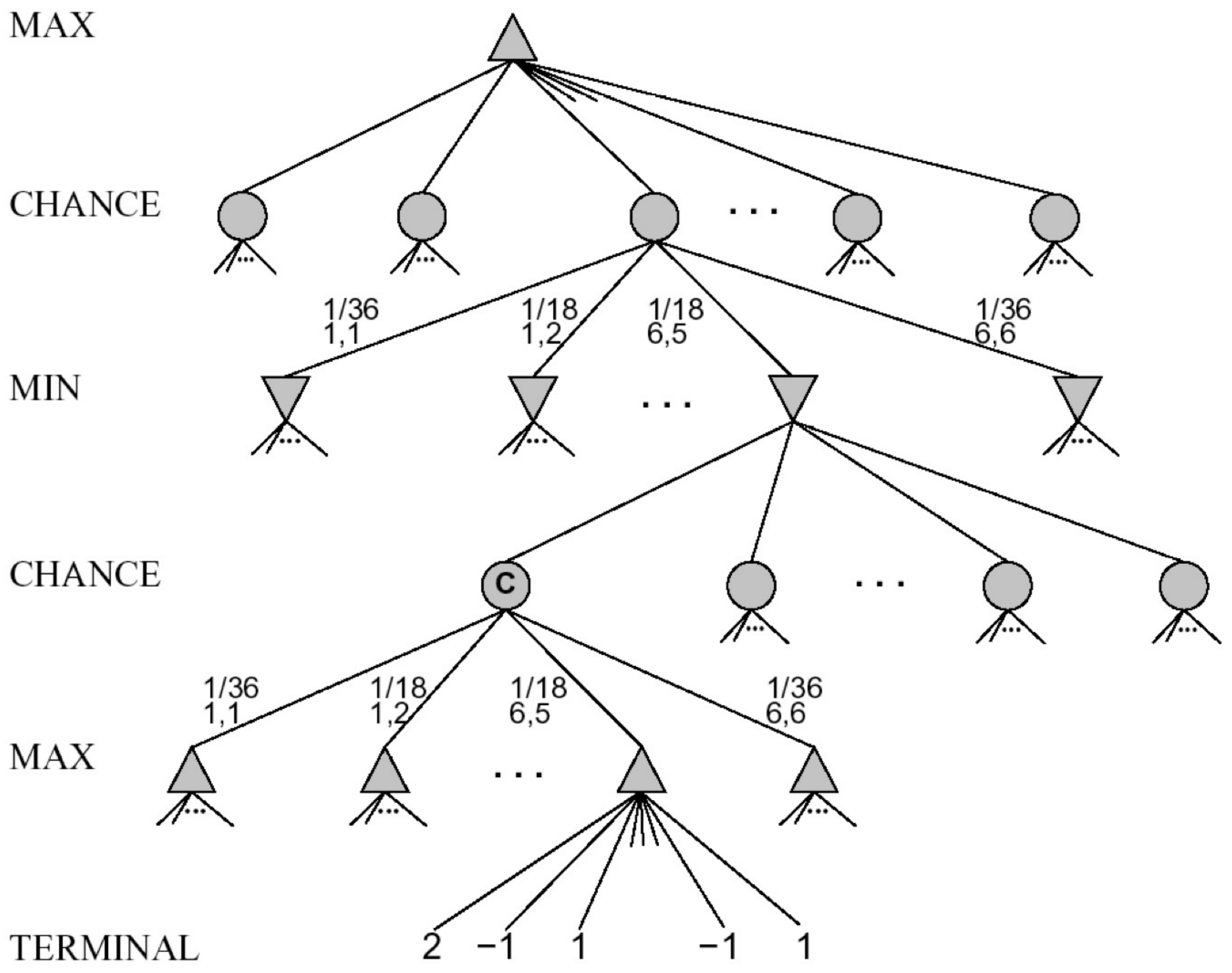
With random aspect

- In the tree, we can include a node representing the random aspect.
- We can include the random distribution in this node.
- We must compute the expected mean (or utility).

$$U(position) = \sum_{k \in Succ(position)} p_k \; U(position_k)$$

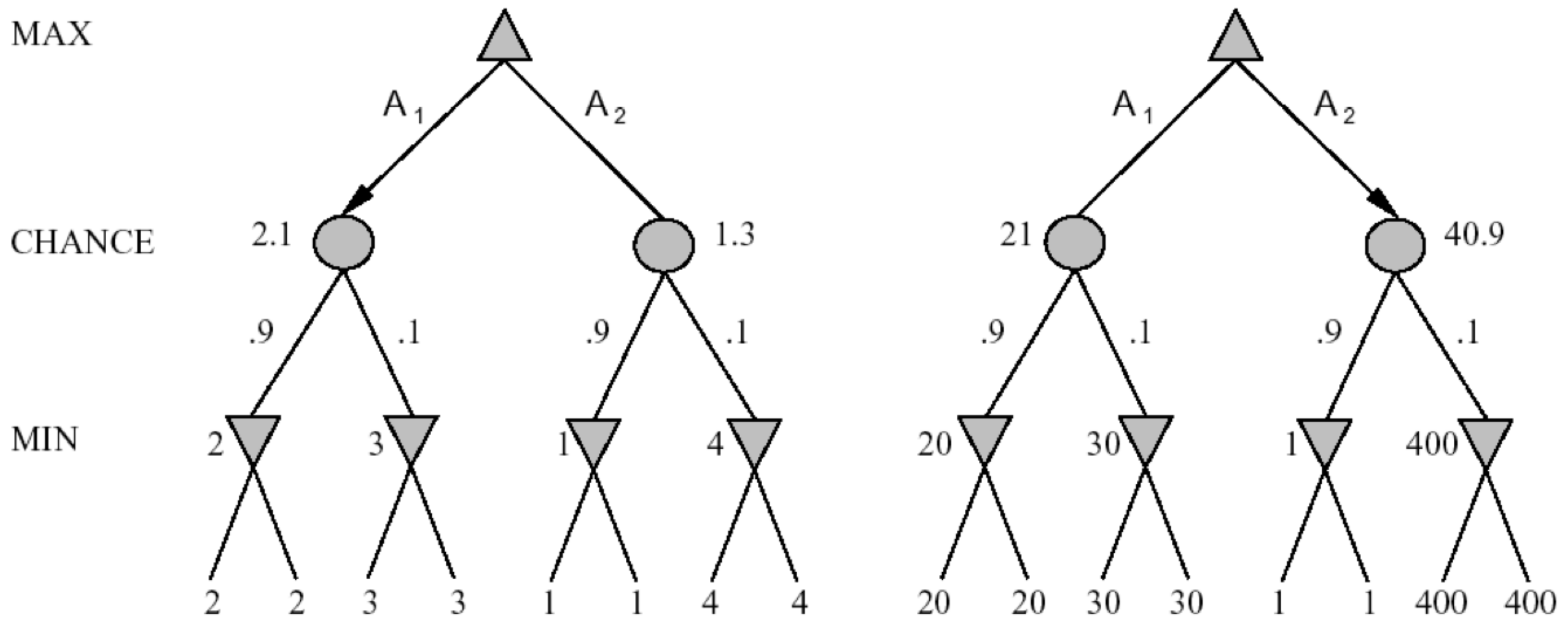
With random aspect

In the tree, we have included a rounded node representing the random aspect.



With random aspect

Which solution to your prefer (root is max)?



With random aspect

- On the left, the best decision is A_1. On the right, it is A_2.
- In both cases, we have the same probabilities. But the mean is sensitive to large (or small) values.
- The complexity function is changing with the random aspect.
- For a game without random aspect $O(b^m)$.
- With random aspect, we need to take account for the n possible outcome $O(b^m \cdot n^m)$.
  With two dices, $n = 21$ (without considering the order $\{(1,1), (1,2), (1,3), \ldots (5,6), (6,6)\}$)
- Backgammon: $b = 20$, $n = 21$, $m = 50$

Conclusion (Chess)

- 1970 : the first program to win the ACM chess contest (based on $\alpha-\beta$ algorithm).
- 1982 : « Belle » the first specialized computer to play chess (a few billions combinaisons per turn).
- 1985 : « Hitech » among the first 800 best chess players in the world (more than 10 M combinaisons per turn).

Games in practice

- Checkers: Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994. Used a precomputed endgame database defining perfect play for all positions involving 8 or fewer pieces on the board, a total of 444 billion positions.
- Chess: Deeper Blue defeated human world champion Garry Kasparov in a six-game match in 1997. Deeper Blue searches 200 million positions per second, uses very sophisticated evaluation, and undisclosed methods for extending some lines of search up to 40 ply.
- Deep Blue = minimax + $\alpha-\beta$ + // computing + opening book + special purposes (end of the game) + uneven tree development + iterative deepening.
- GO: *b > 300*, so most programs (e.g., AlphaGo) use a set of complementary strategies (deep learning (NN), machine learning (reinforcement)). Success with Google DeepMind (2016)
  But it is not working with your PC.

Conclusion

- Are computers intelligent?
- «Saying Deep Blue doesn’t really think about chess is like saying an airplane doesn't really fly because it doesn't flap its wings.» Drew McDermott.
- «Chess is the Drosophila of artificial intelligence. However, computer chess has developed much as genetics might have if the geneticists had concentrated their efforts starting in 1910 on breeding racing Drosophila. We would have some science, but mainly we would have very fast fruit flies.» John McCarthy

Summary

- Games are fun to work on!

- They illustrate several important points about AI.
- Minimax algorithm.
- Usefulness of pruning (according to $\alpha-\beta$ scheme).
- Add progressive deepening.
- Perfection is unattainable → must approximate
- Idea how to work with opponent.
- Some are still difficult. Need the combination with other techniques.