Games, logical agents

UI V

Mária Markošová

Summary of the last lecture

1. CSP problem, definition.
2. CSP examples.
3. Backtracking for CSP.
4. CSP heuristics and controls.

Outline

1. Definition of games, one agent game, two agents game.
2. Minimax algorithm.
3. $\alpha - \beta$ prunning.
4. Evaluation functions.
5. Games and probability.
6. How to find a good strategy.
7. Introduction to logical agents

One player (agent) game

- It is basically a searching problem
  - Find a sequence of moves to get a required state
  - Example: Rubik cube
  - Solution by searching with an appropriate heuristics

Rubik cube

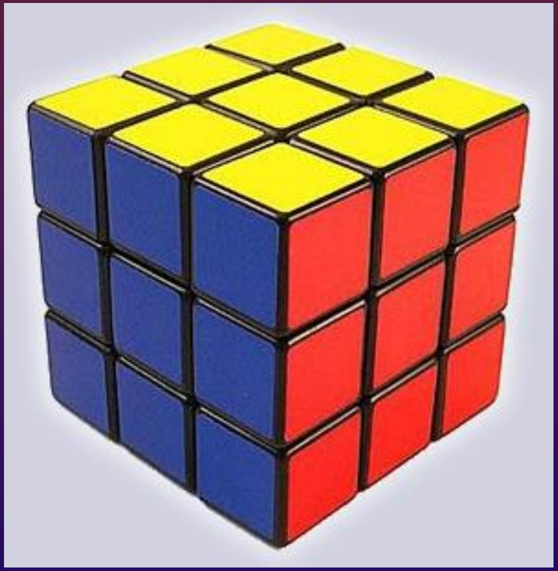
Problem: Find a desirable state, on the picture from an arbitrary initial state.

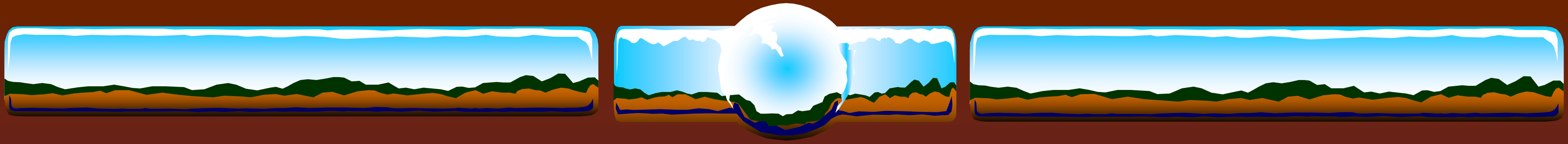
Partial problems:

-Find a move sequence which exchanges position of two small cubes not violating others.

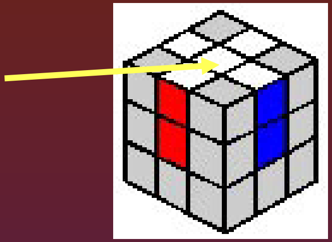
-example: there is known 14 ply sequence how to exchange a position of 2 corner cubes

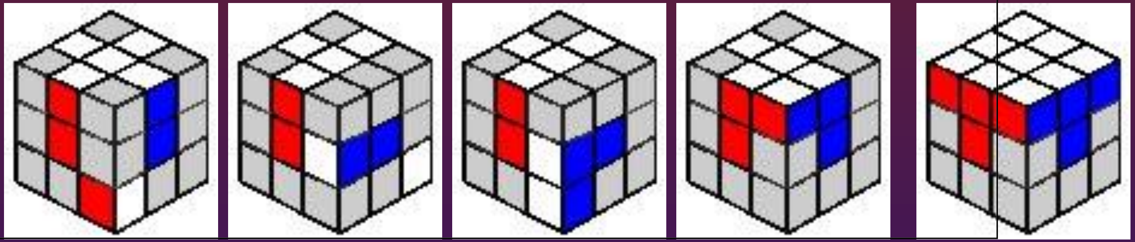
-It is a difficult searching problem

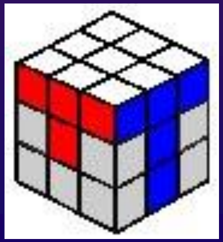




Heuristic steps:

1. Find a cross: 

2. Place first corner cubes on the first level, then complete the first level: 

3. Complete T shapes : 

Multiplayer game : multiagent environment

Multiagent environment types

1. Agents are cooperative
2. Agents are competitive

$\downarrow$

game

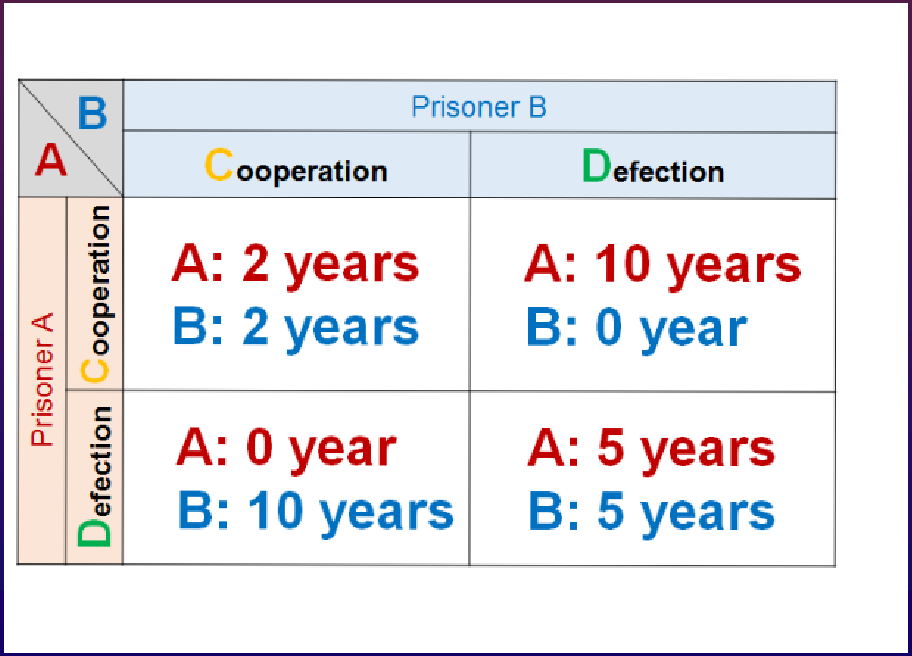
Game : consists of competitive agents in the multiagent environment.

Simultaneous games

Players make decisions at the same time.

Example: Prisoner's dilemma – described by the payoff matrix

Both prisoners A and B are interrogated at the same time and they are both told:
If you do not defect your partner, and you cooperate with him, the number of years in prison are: if your opponent cooperates too you both get 2 years in prison. If you cooperate, but your opponent defect's you, you get 10 years and he zero. If you both defect you both get five years.



A \ B		Prisoner B	
		Cooperation	Defection
Prisoner A	Cooperation	A: 2 years B: 2 years	A: 10 years B: 0 year
	Defection	A: 0 year B: 10 years	A: 5 years B: 5 years

Prisoner’s dilemma strategies

1. There is only one interrogation. What is the best strategy?

   To defect.

2. There are more interrogation and you are informed what your opponent did previously, the strategy changes.

   Tit for tat is the best strategy.

Sequential games

The players do not decide simultaneously, but sequentially. First player makes a move and the second player answers the move.

If there are more then two players, they make alternate moves.

Example: chess

These games use evaluation function.

The rest of the lecture is devoted to the sequential games.

Game vs. searching problems

- „Unpredictable" opponent → player has to find an answer to each opponent move
- Time limit → makes a perfect solution difficult, often improbable. Player has to approximate.

What is a game? Summary

Game for us will be: idealized world, in which states are well defined, rules are well defined. In this world two (most simple case) competitive agents fight to gain as much utility (grades) as possible on the cost of the partner.

a) Environment is fully observable, deterministic, multiagent. In our case we have first two (later more) agents.

Episodic ?

b) Actions of two competitive agents alternate and are done with respect to the game rules.

a) Utility values at the end of a game can be equal but with opposite sign. (e.g win +1, loss –1, no loss no win 0 = zero sum games).

Example: chess, tic tac toe, checkers

Zero sum game:

In the zero sum game gain of one agent and loss of the other (or more others) agent is exactly balanced . If one gains *x* grades, the other gains –*x*.

Example: chess, checkers, etc.

We are going to speak about zero sum games here.

Typical properties of real games

-A lot of game steps for both agents (chess about 50)

-A great branching factor (chess typically about 35)

-Great searching trees (chess $35^{100} \approx 10^{50}$ nodes)

-Agent needs to decide not knowing whether his decision is good or bad

-Bad decision are quickly penalized and agent has troubles.

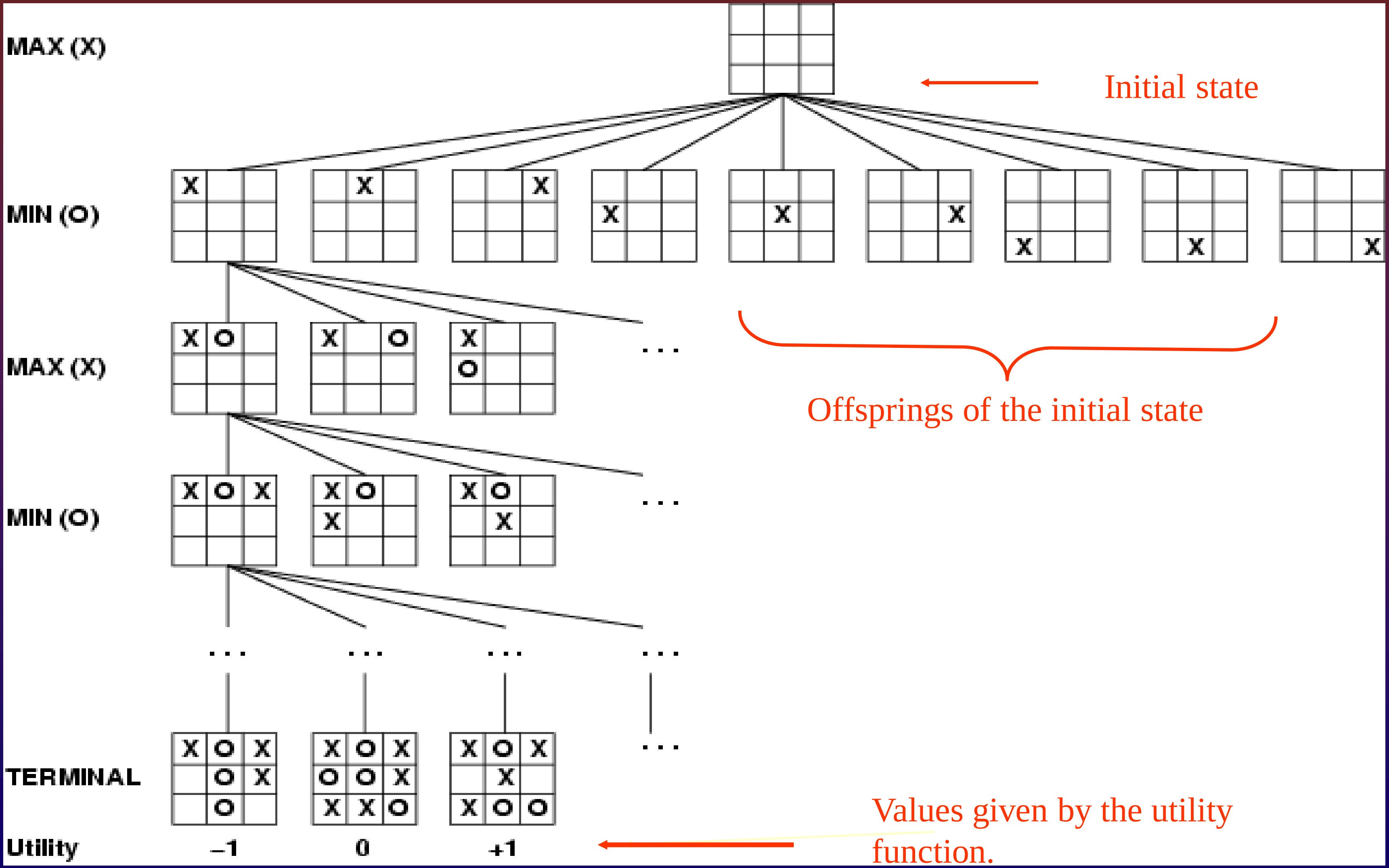
Game of two players

Agents: MAX makes the first step, MIN the second one, then alternate.

MINIMAX: Searching problem consisting of:

- ***initial state;*** board position and specification of player which is going to make a step
- ***operators (successor function);*** ***list of connections (ply, state)***
- ***final test;*** is it an end of the game? Is the game in one of the terminal states?
- ***utility function (payoff f.) ;*** gives numerical value as a game result (1,-1,0 in chess)

Tic tac toe: Game tree (2-player, deterministic, turns)


MAX (X)
MIN (O)
MAX (X)
MIN (O)
TERMINAL
Utility
–1
0
+1
Initial state
Offsprings of the initial state
Values given by the utility function.

Strategy: All correct moves for MAX as an answer to MIN moves leading to the potential victory.

Problem in a game: To find an optimal strategy.

Optimal strategy: Finding MINIMAX values of each node *n* of the searching tree.

Minimax-Value(n): It is an utility function value (from the MAX point of view) in the node *n*. Therefore MAX chooses moves maximizing his utility.

Minimax-Value(n)=Utility(n) if *n* is a terminate state

Utility function:

-Usually it is a price which agent pays for solving the task.

-In games it is a value, points, grades which agent MAX gains or

pays when the game is over.

-For zero sum games agent MIN gain such value which sums to zero with MAX value

Let us create an artificial game called MINIMAX.

Rules of simple game called MINIMAX

a) First ply (move): MAX possibilities are A_1, A_2, A_3.

b) Second ply: possible answers for A_1 are A_{11}, A_{12}, A_{13}, for A_2 are A_{21}, A_{22}, A_{23} etc.

c) Game finishes after one move of MAX and MIN, searching tree is thus two plies deep.

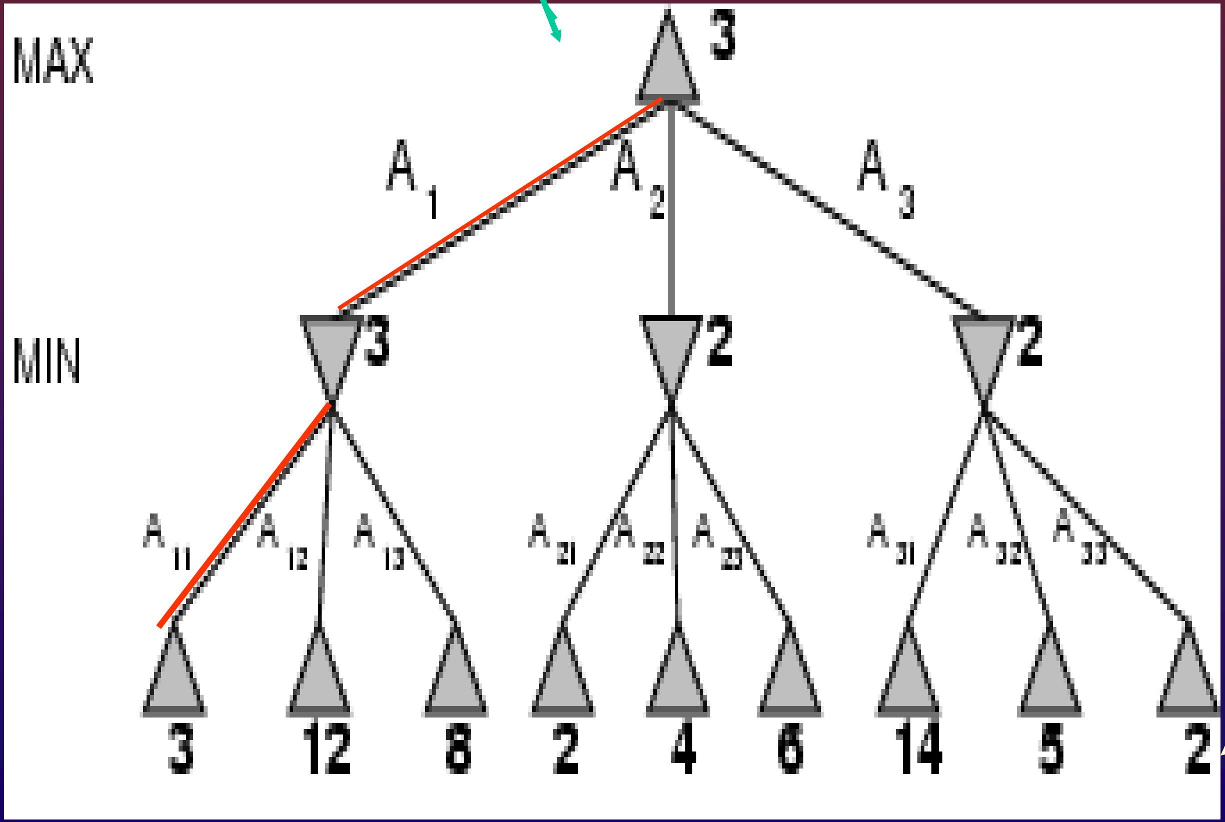
d) *Minimax-Values* of terminal states are in the interval *2 – 14*.

We suppose:

1. Both players exchange their moves.
2. MinMax values of terminal states are bounded.
3. Both players play optimally, that means both try to gain as much as possible on the cost of the other player.
4. To gain something means to gain the best possible value at the end of the game.
5. These conditions are necessary conditions for MINIMAX algorithm to find the best possible move sequence.

MINIMAX example

Best MAX move



▲ ← ply MAX

▼ ← ply MIN

Utility (MinMax), values from the MAX viewpoint.

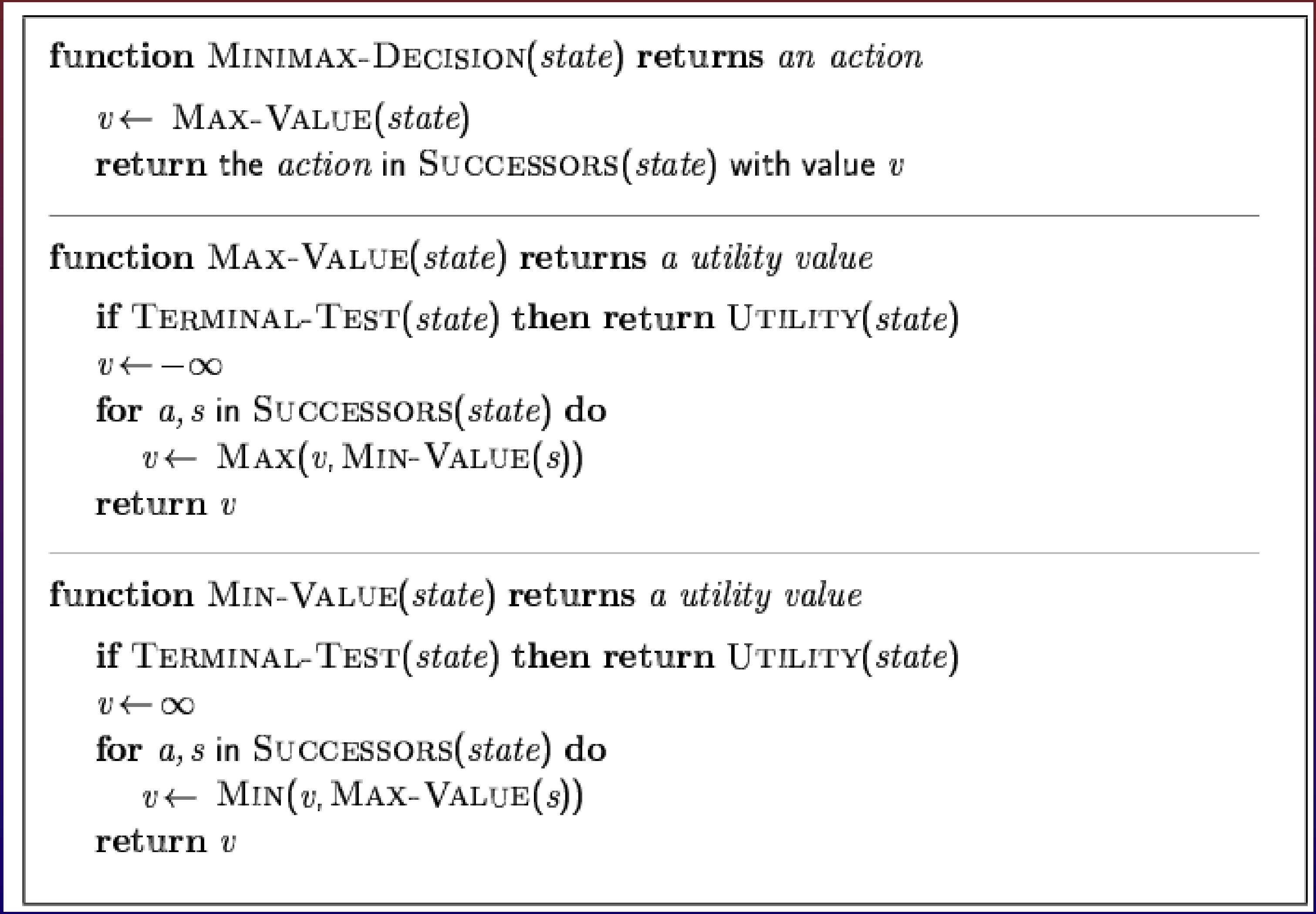
-3 -12 -8 from the MIN viewpoint

MINIMAX algorithm

Finds optimal strategy for MAX; supposes that both MAX and MIN play moves which are the best for them, that mean worst for the opponent. Algorithm:

- Generates complete game tree by a depth first method.
- Evaluates terminal states.
- Propagates MinMax values from the tree leaves to the root.
- To the nodes where MAX is moving the maximum of values is given, to those where MIN is moving the minimum.

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 $v$
```

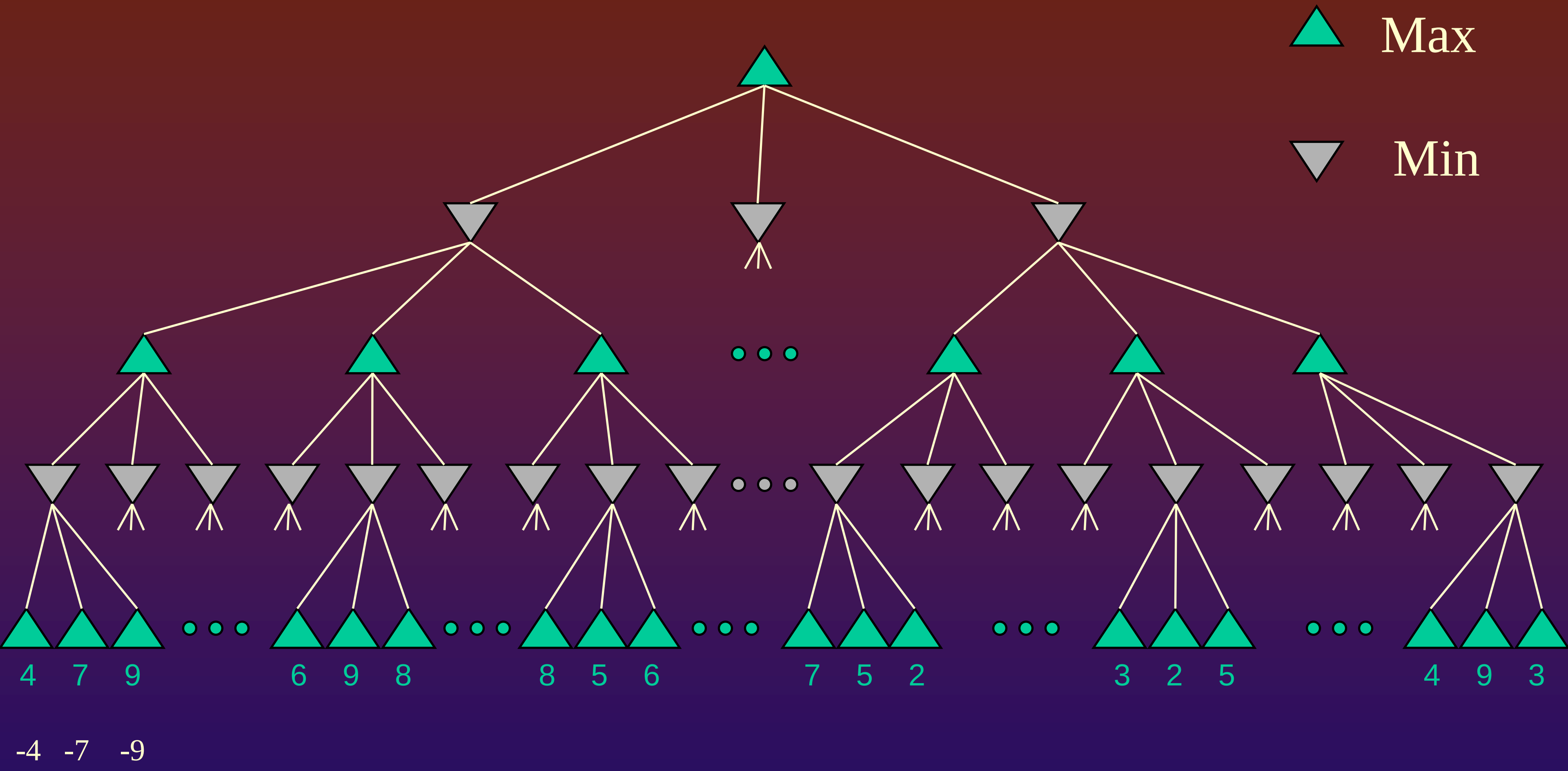
MINMAX algorithm properties

- Complete? Yes (for the finite game tree)
- Optimal? Yes (if both players make an optimal moves, maximizing their profit)
- Time complexity? O(b^m)
- Memory complexity? O(bm) (if depth first is used)

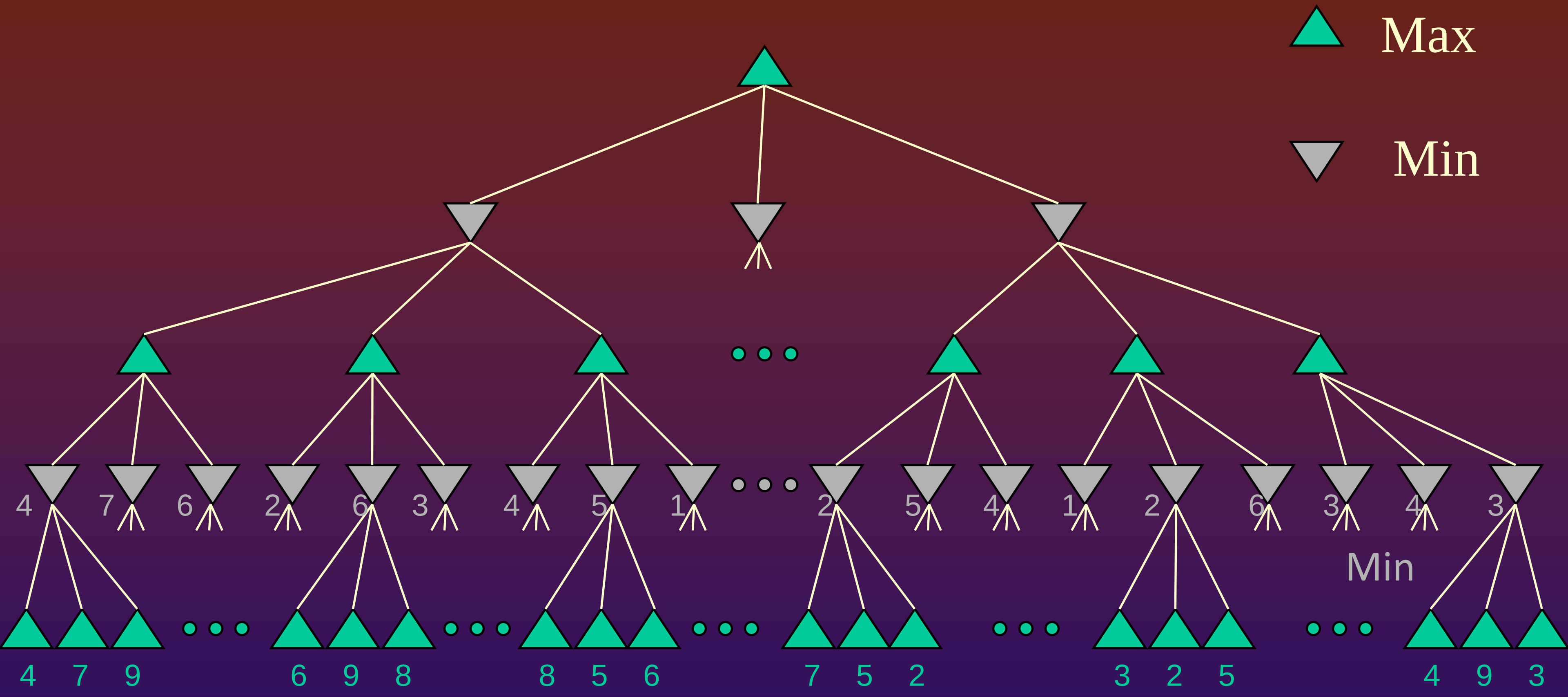
- Chess, $b \approx 35$, $m \approx 100$ for “standard" manner of playing
  → exact solutions are in fact unreachable

MINIMAX example

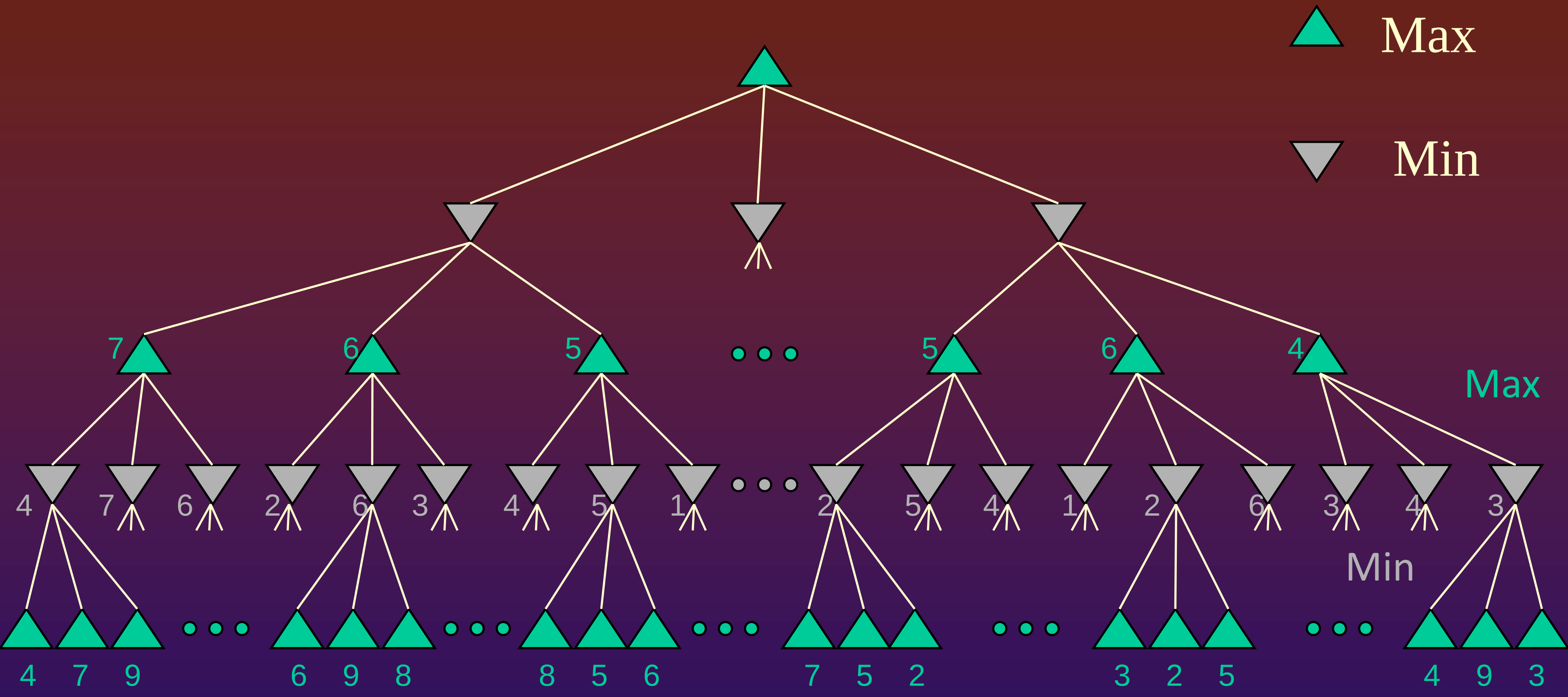
It is up to you to tell me the values of nodes

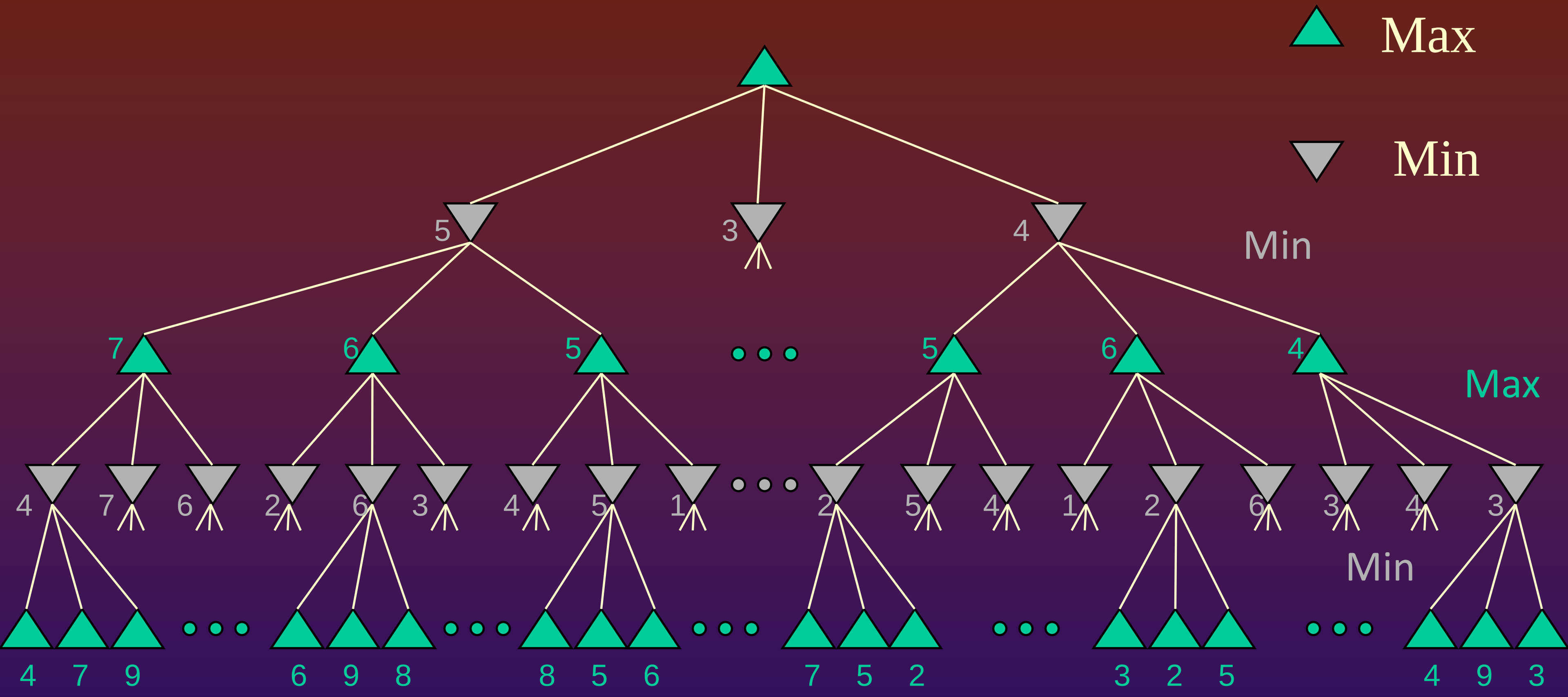


Terminal nodes and their utilities (MinMax values): the numbers are given by the utility function.

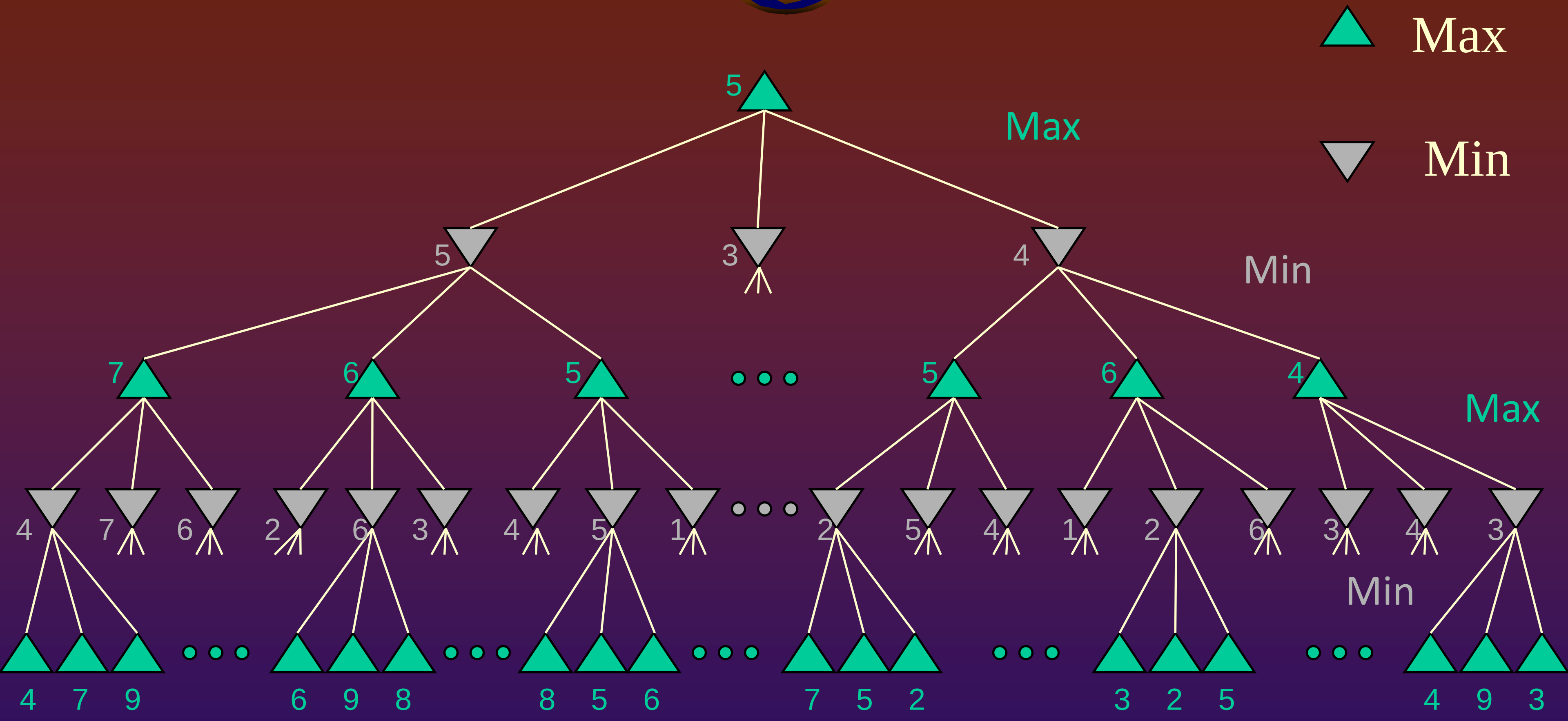


other nodes: values calculated via minimax algorithm

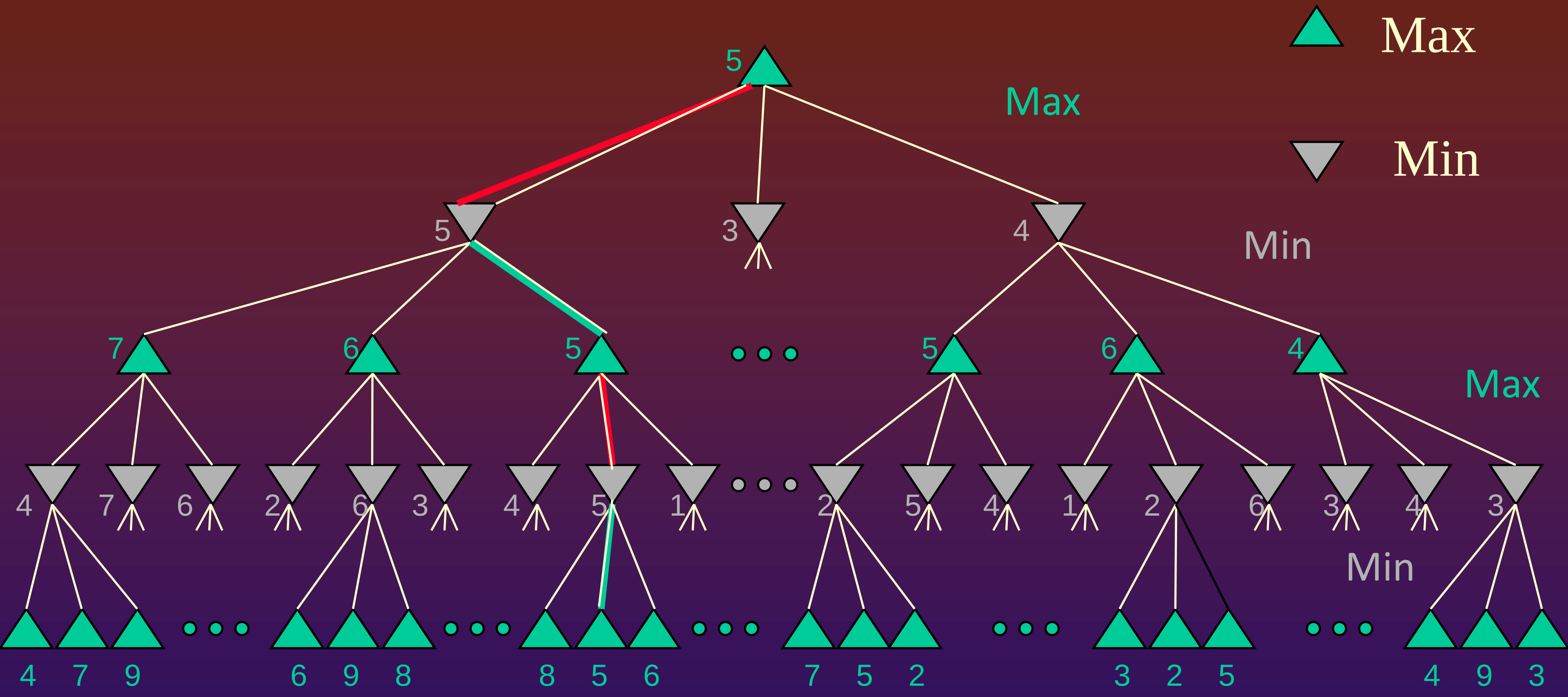

Max
Min
7
6
5
5
6
4
Max
4
7
6
2
6
3
4
5
1
2
5
4
1
2
6
3
4
3
Min
4 7 9
6 9 8
8 5 6
7 5 2
3 2 5
4 9 3



Max
Min
5
3
4
Min
7
6
5
5
6
4
Max
4
7
6
2
6
3
4
5
1
2
5
4
1
2
6
3
4
3
Min
4 7 9
6 9 8
8 5 6
7 5 2
3 2 5
4 9 3



Max
Min
5
Max
5
3
4
Min
7
6
5
5
6
4
Max
4
7
6
2
6
3
4
5
1
2
5
4
1
2
6
3
4
3
Min
4
7
9
6
9
8
8
5
6
7
5
2
3
2
5
4
9
3



▲ Max
▼ Min
Max
Min
Max
Min
5
5
3
4
7
6
5
5
6
4
4
7
6
2
6
3
4
5
1
2
5
4
1
2
6
3
4
3
4 7 9
6 9 8
8 5 6
7 5 2
3 2 5
4 9 3

Minimax for more then two players (Maxn algoritmus)

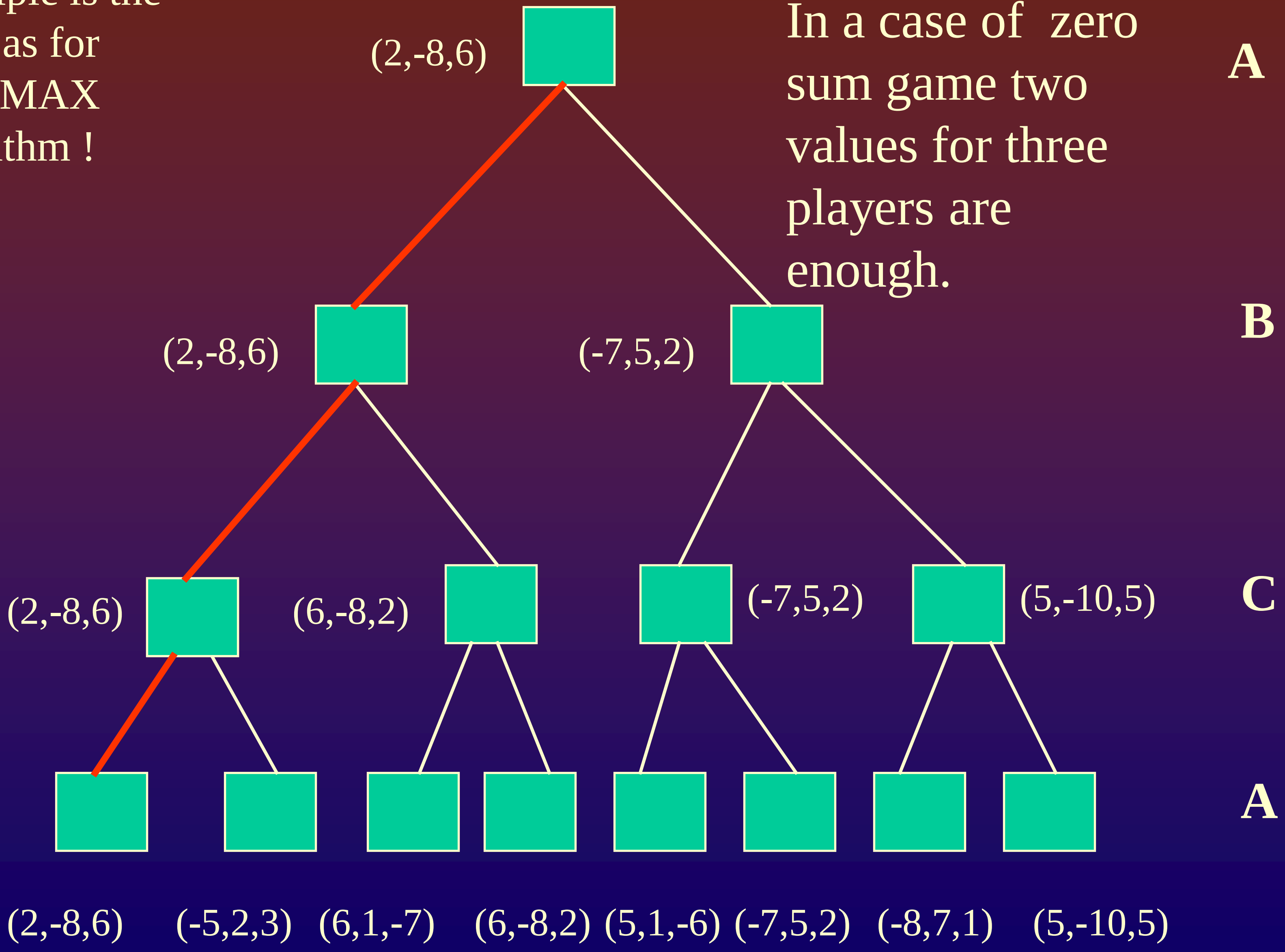
Let us have three optimally playing agents A, B, C. Their moves alternate

Our game is one step deep ABCA.

Utility function output is a vector: $\left(v_A, v_B, v_C\right)$

Principle is the same as for MINIMAX algorithm !

In a case of zero sum game two values for three players are enough.



MINIMAX variants

If the game tree is too deep:

1. Minimize number of searched branches.
2. Cut the tree at some depth.
3. Use heuristic function to evaluate nodes.

cutt – off search

$\alpha - \beta$ pruning

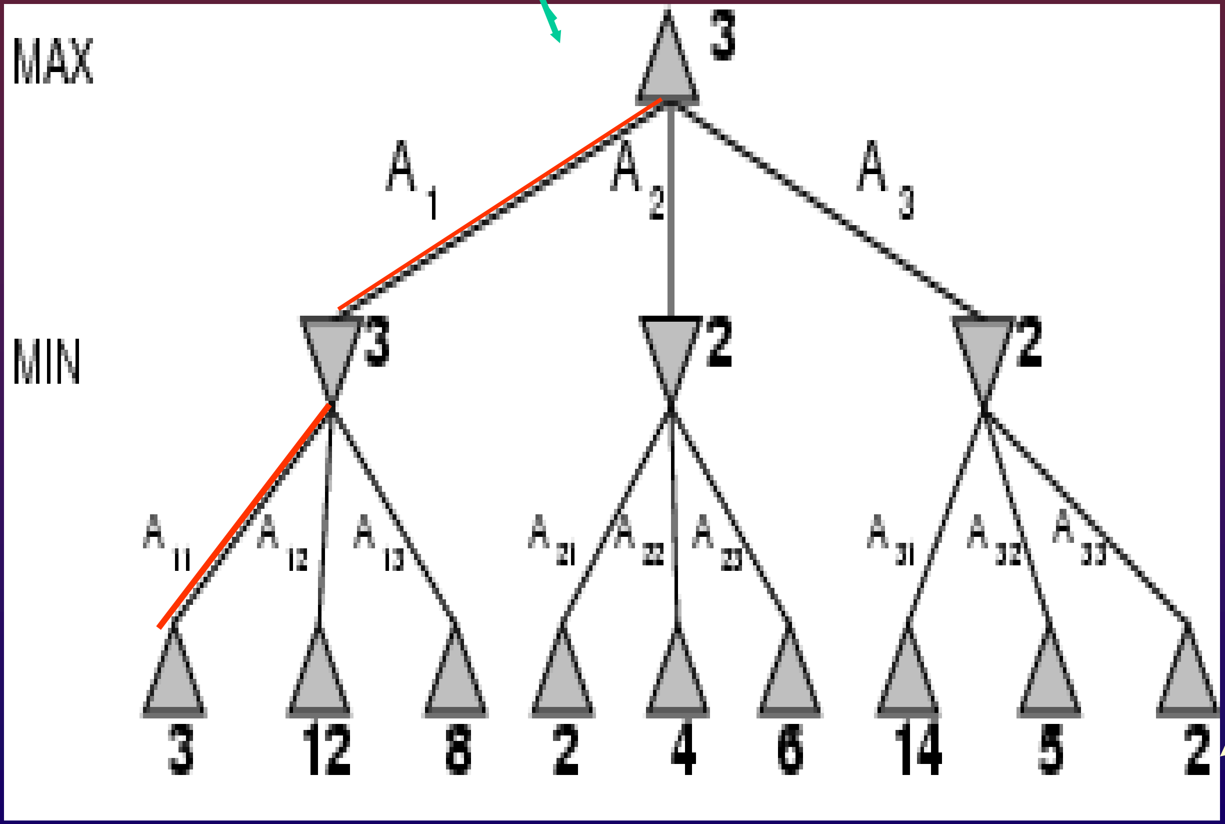
$\alpha - \beta$ pruning

Method : gives the same moves as MINIMAX, but prunes searching unpromising branches.

MINIMAX is usually implemented as $\alpha - \beta$ pruning.

MINIMAX example

Best MAX move



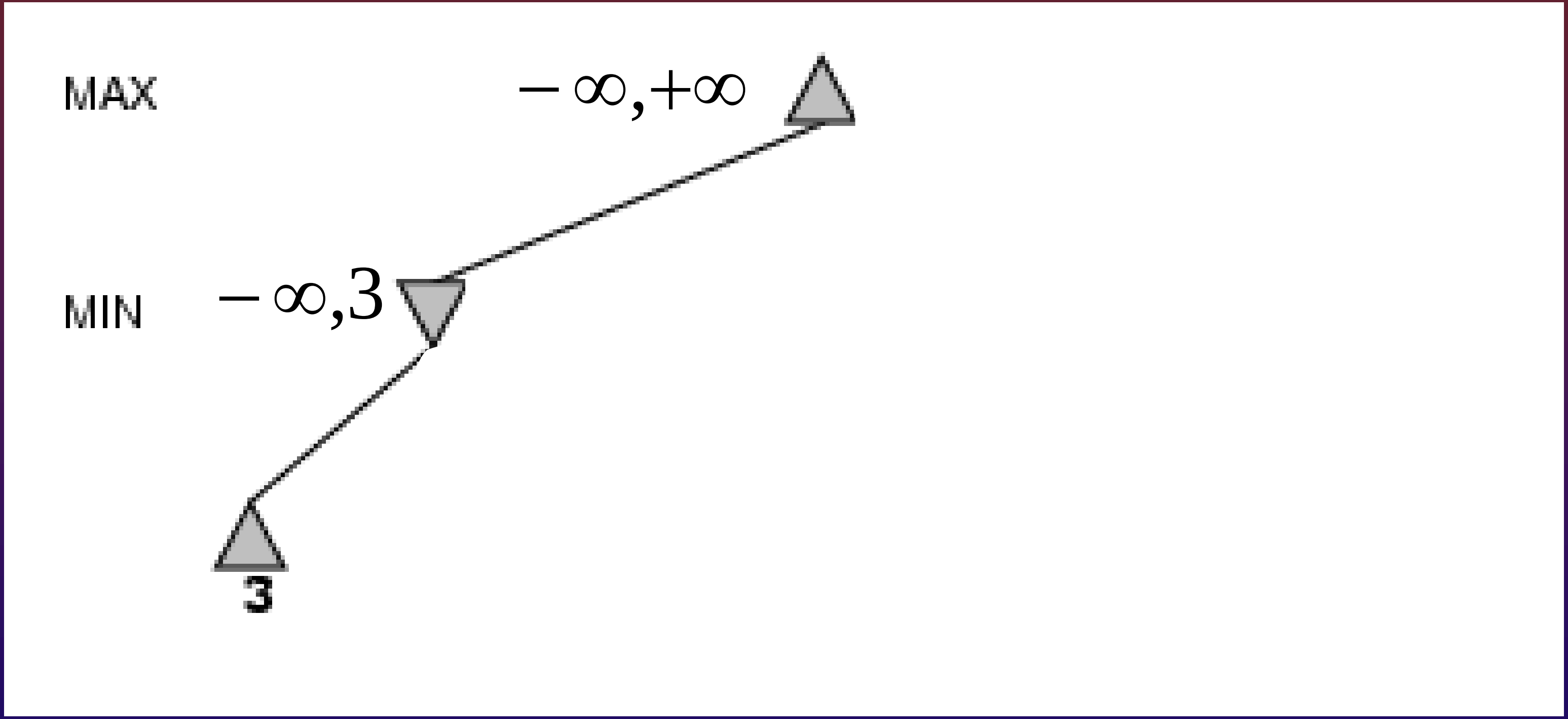
▲ ← ply MAX

▼ ← ply MIN

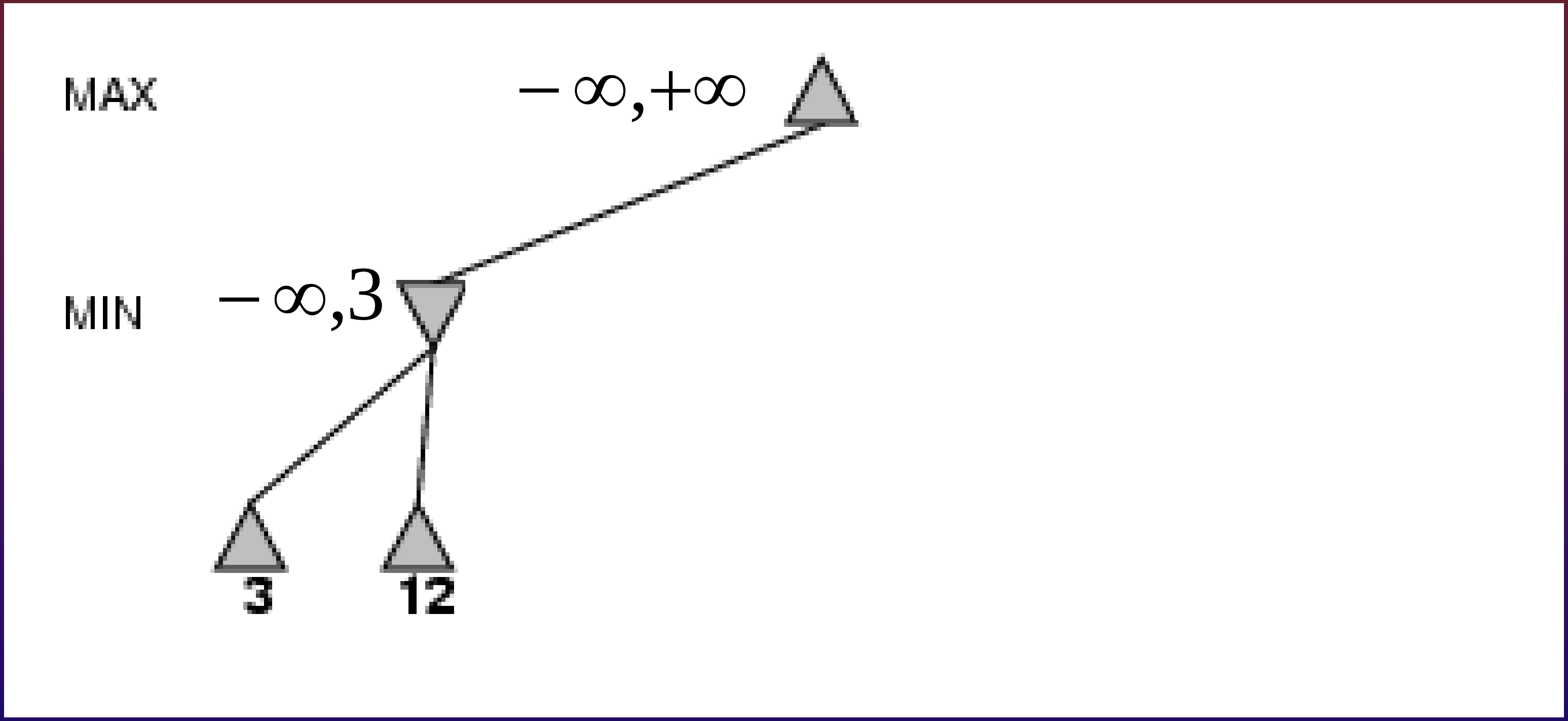
Utility (MinMax), values from the MAX viewpoint.

-3 -12 -8 from the MIN viewpoint

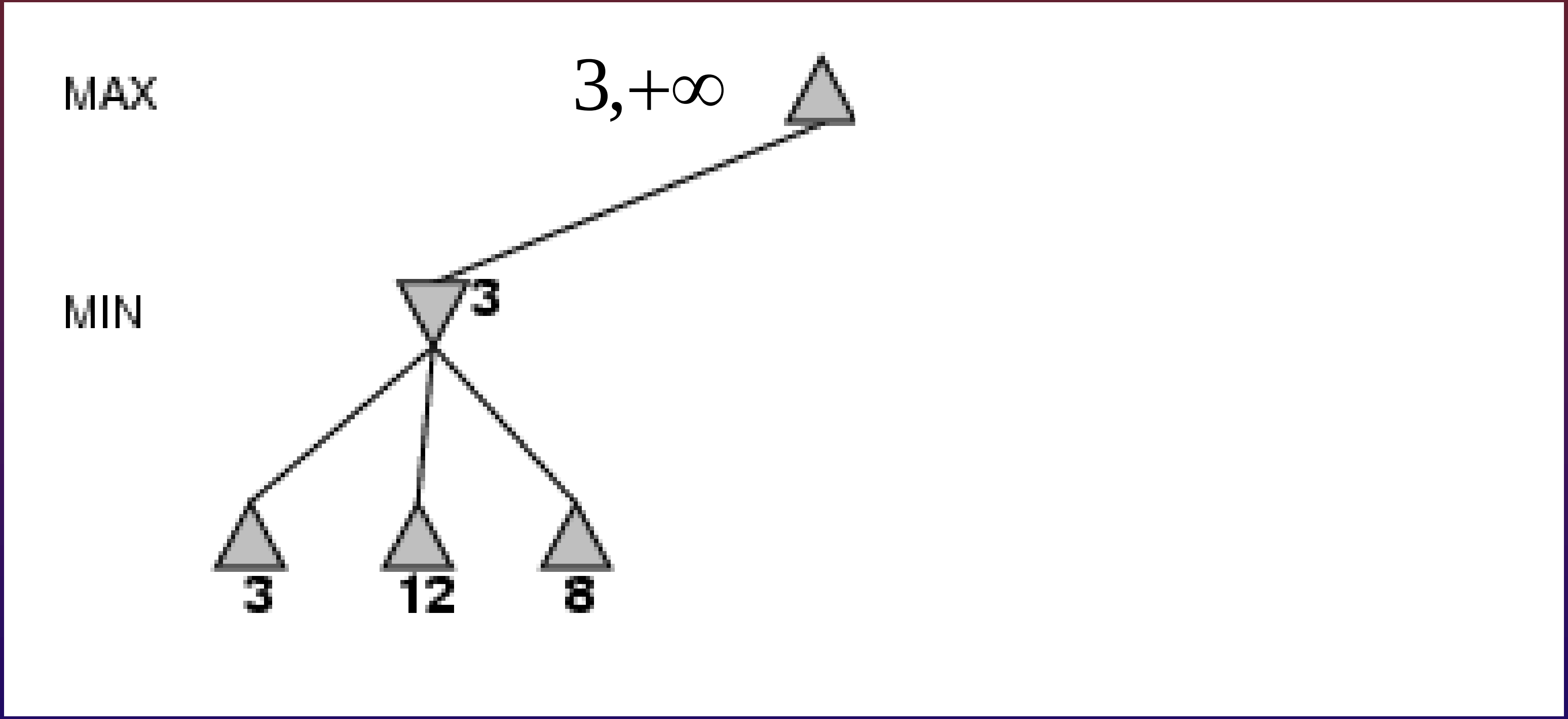
α-β pruning example (from Russel Norwig book)



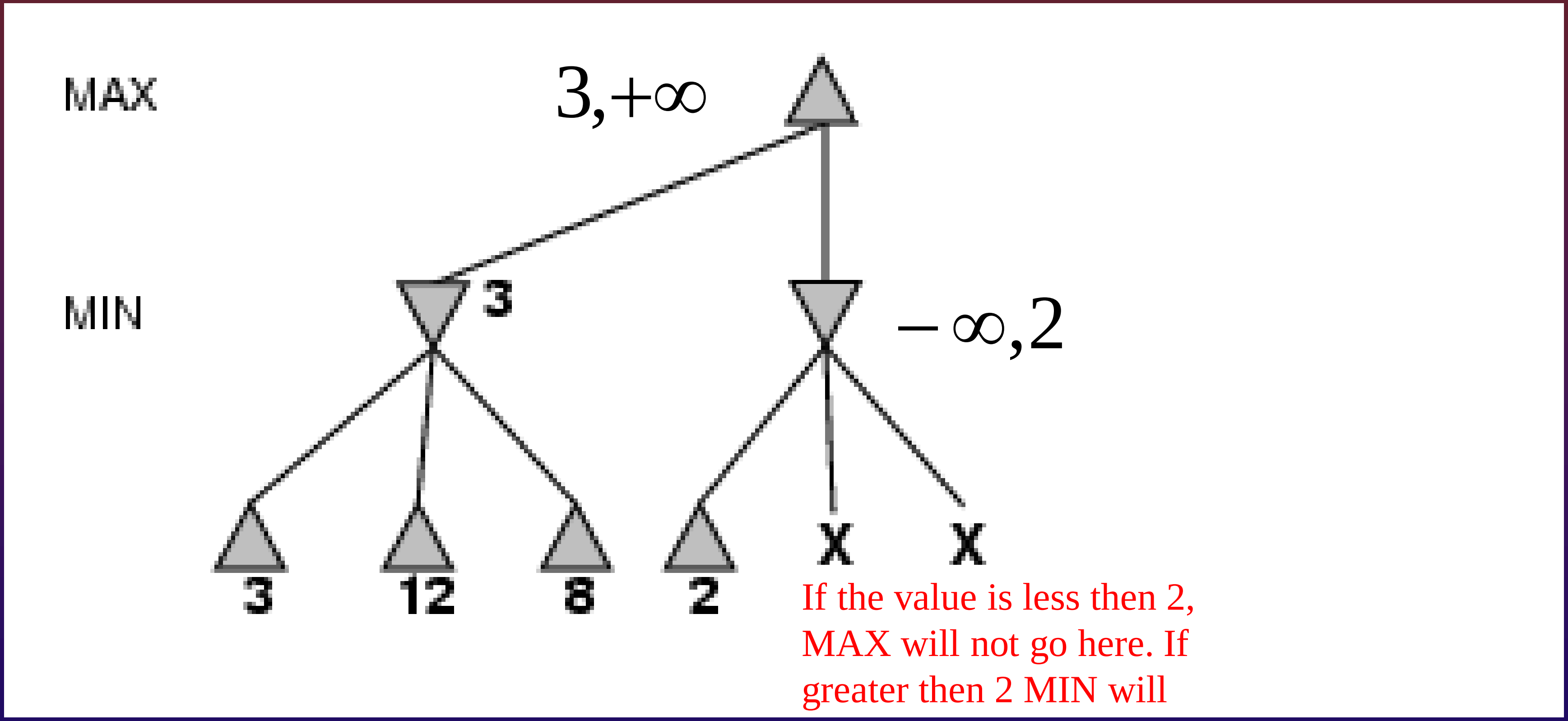
α-β pruning example



α-β pruning example

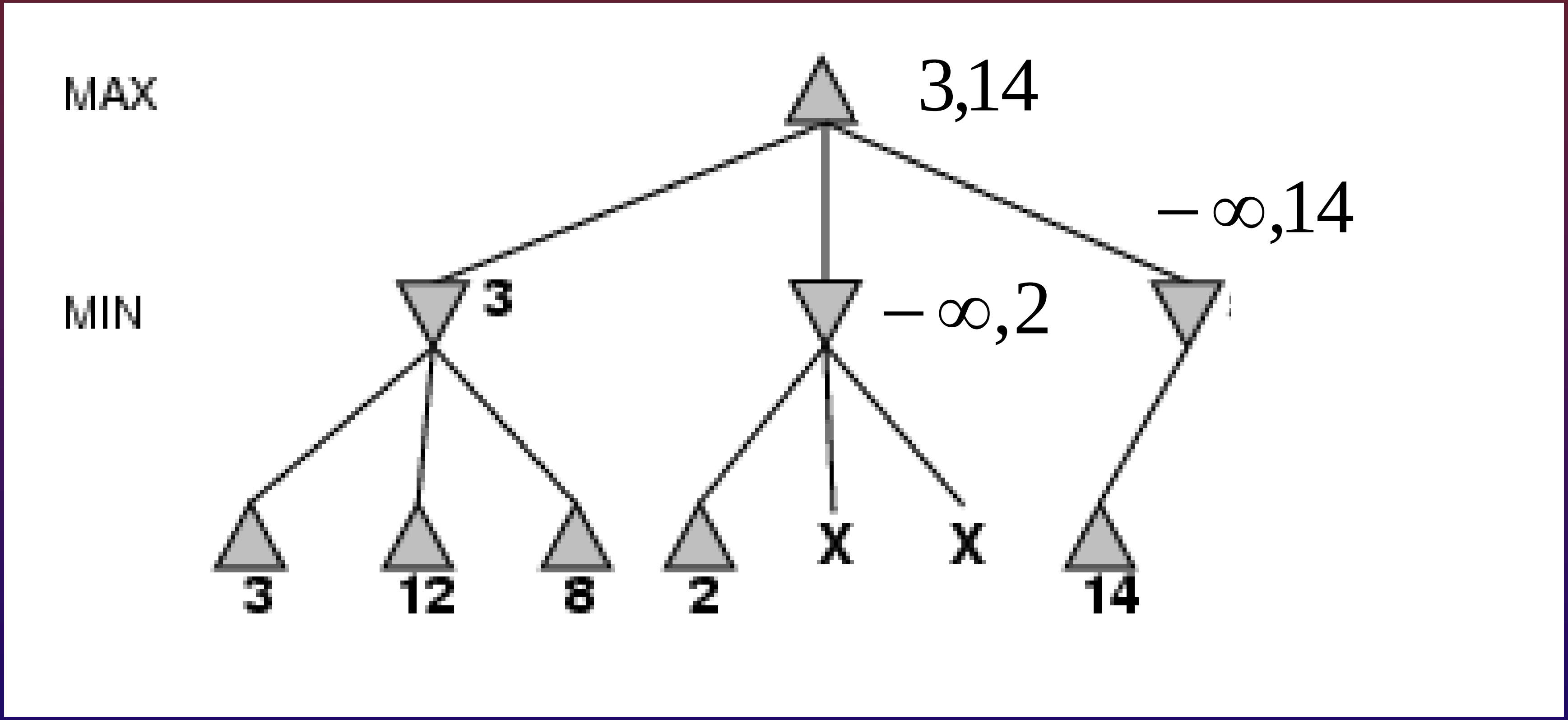


α-β pruning example

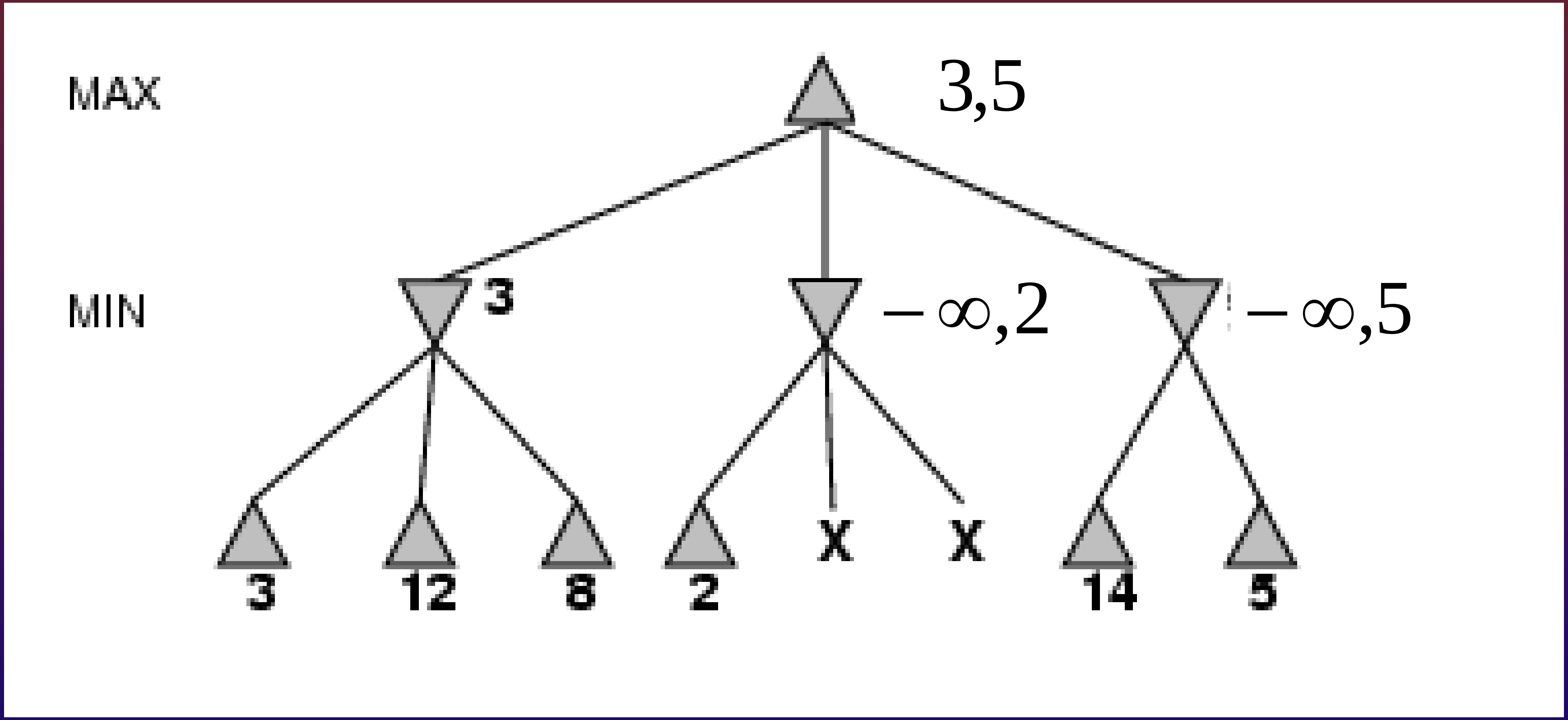


If the value is less then 2, MAX will not go here. If greater then 2 MIN will not choose it.

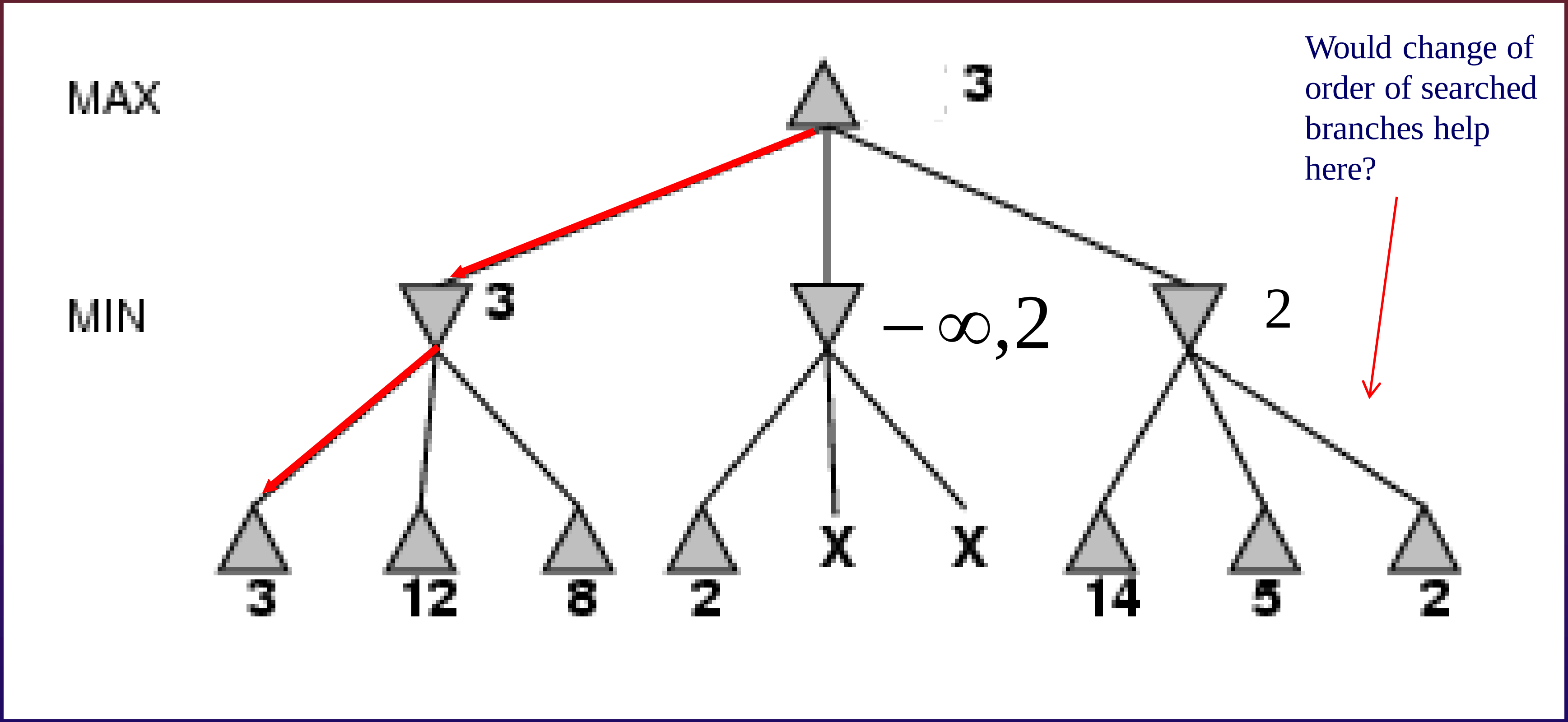
α-β pruning example



α-β pruning example



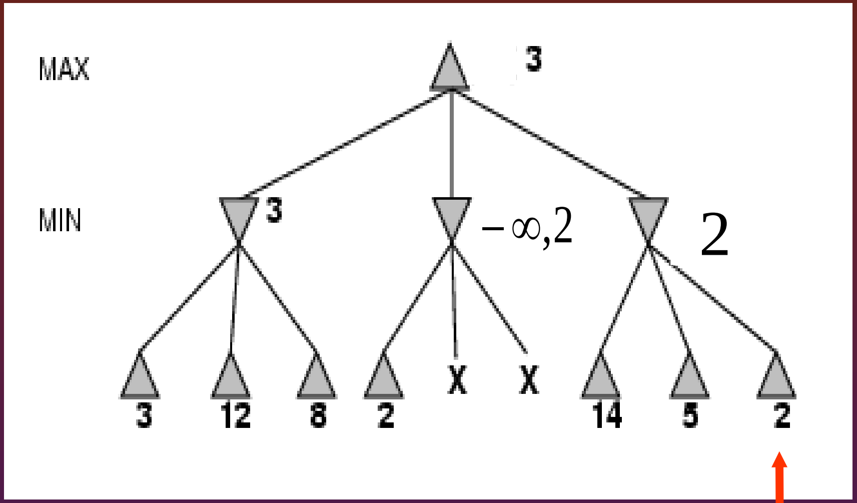
α-β pruning example



Minmax-value(root)=max(min(3,2,12), min(2,x,x), min(4,5,2)) = max(3, min(2,x,y),2) = 3

Properties of α-β pruning

- Pruning does not change the final result.
- Branch order has a great influence on the pruning algorithm.
- Perfect order in pruning gives the time complexity $= O(b^{m/2})$
  - → for the same time available we are able to search to the double depth.

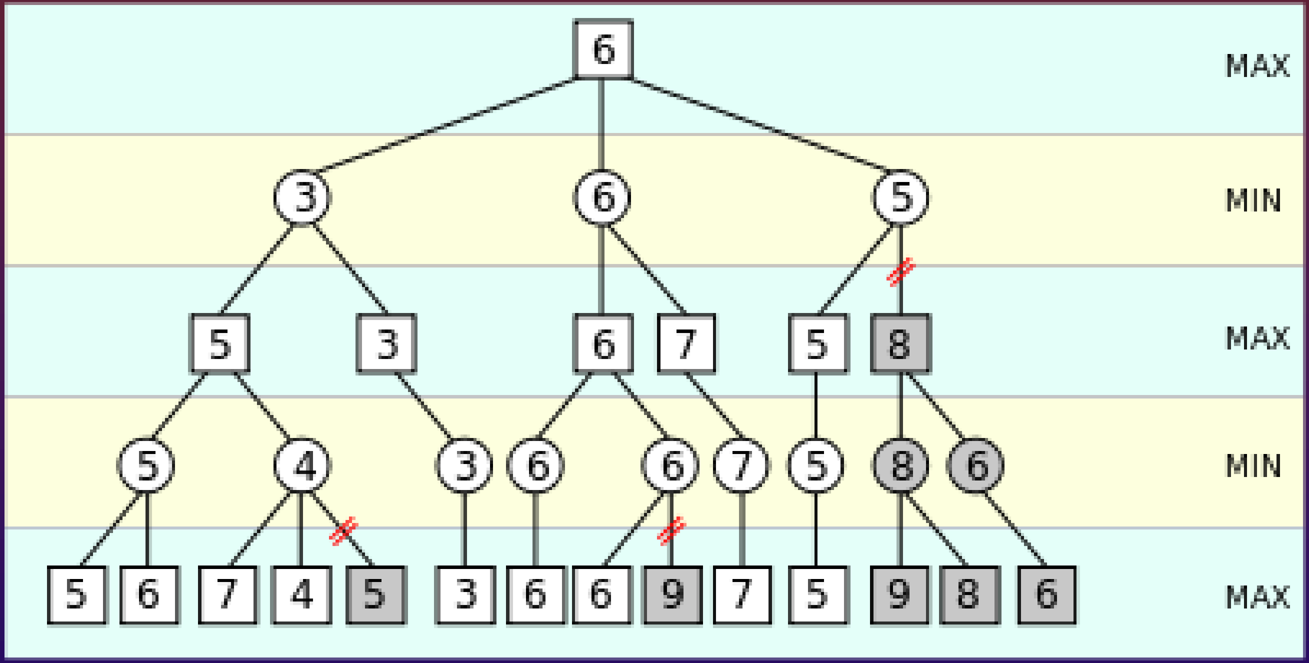


Order of successor generation influences algorithm effectivity.

Heuristics for ordering is important.

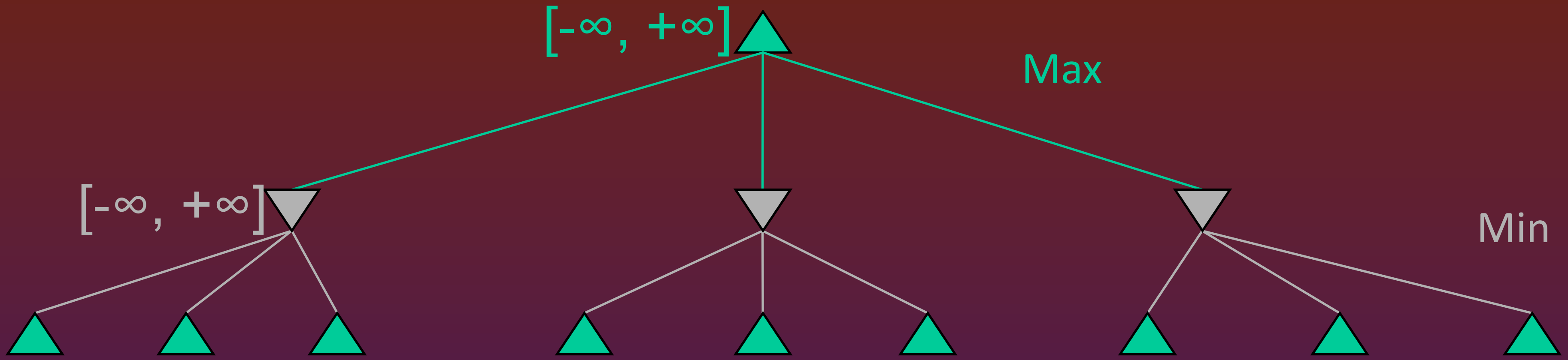
Generating this node as a first one, two next branches are pruned.

Another example of alpha –beta pruning



Alfa beta pruning, example.

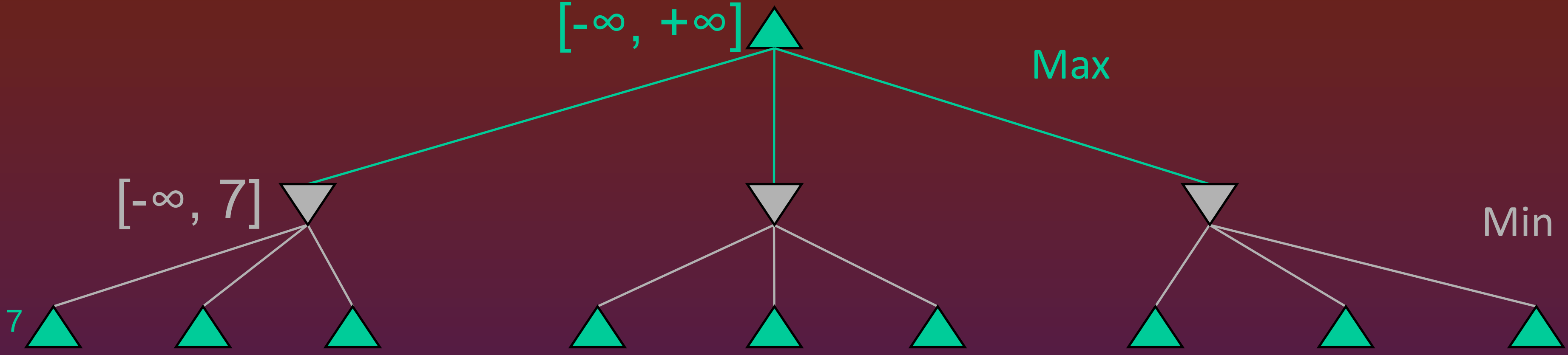
Tell me the node values.



α best choice for Max ?

β best choice for Min ?

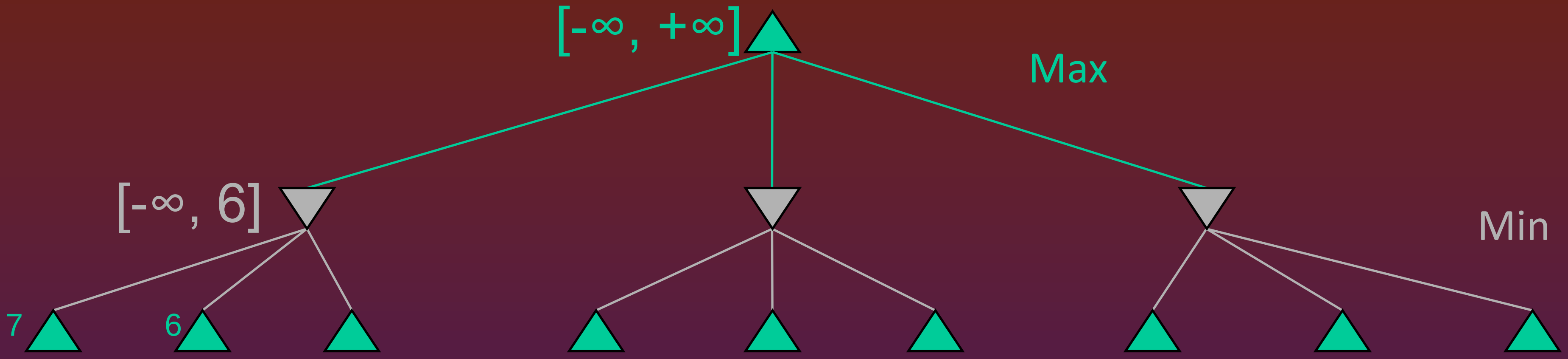
- we assume a depth-first, left-to-right search as basic strategy
- the range of the possible values for each node are indicated
  - initially [-∞, +∞]
  - from Max's or Min's perspective
  - these *local* values reflect the values of the sub-trees in that node; the *global* values α and β are the best overall choices so far for Max or Min



α best choice for Max ?

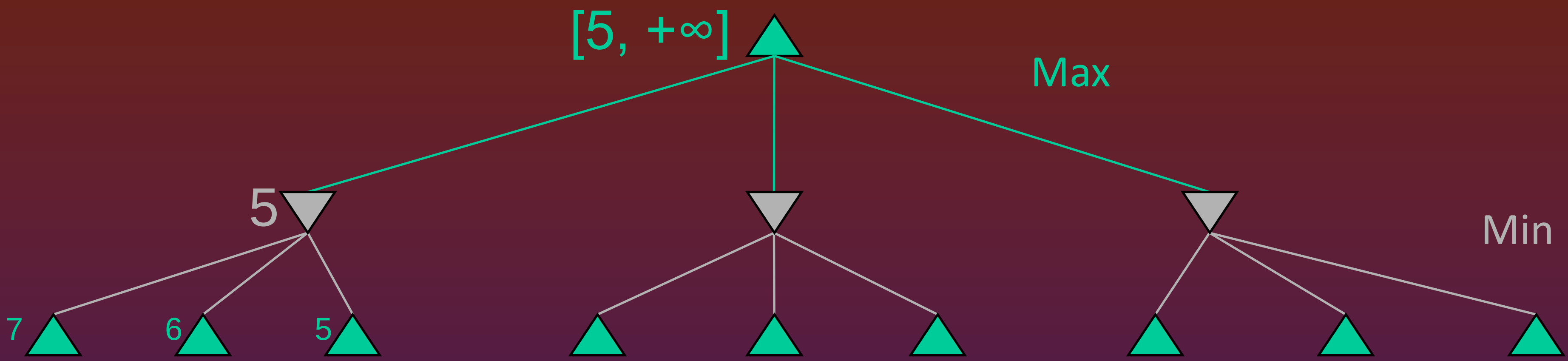
β best choice for Min 7

❖ Min obtains the first value from a successor node



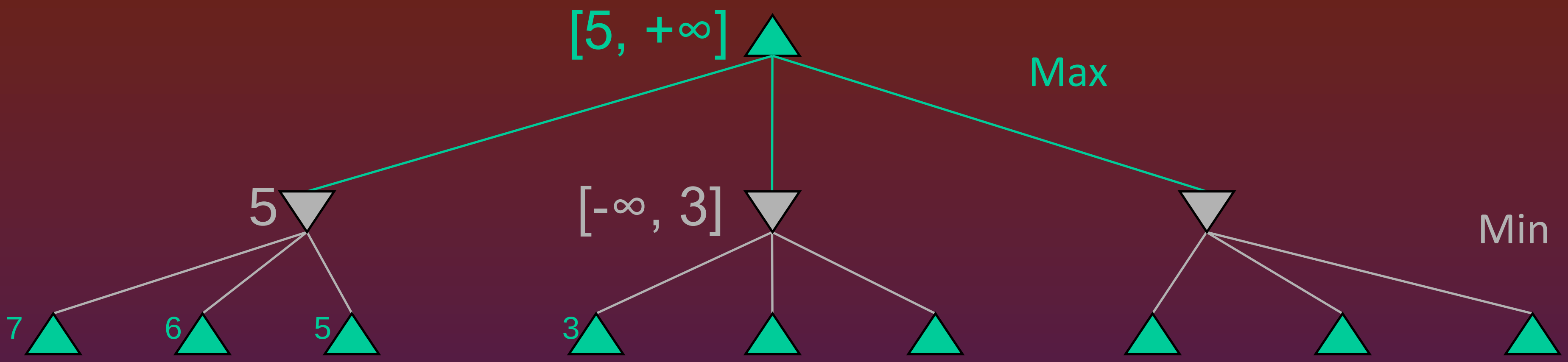
α best choice for Max ?
β best choice for Min 6

- Min obtains the second value from a successor node



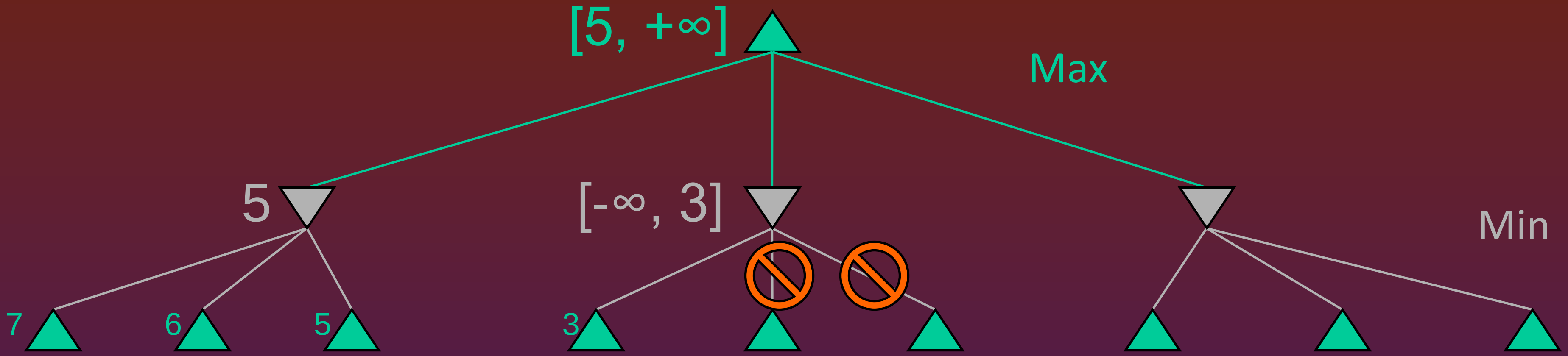
α best choice for Max	5
β best choice for Min	5

- Min obtains the third value from a successor node
- this is the last value from this sub-tree, and the exact value is known
- Max now has a value for its first successor node, but hopes that something better might still come



α best choice for Max	5
β best choice for Min	3

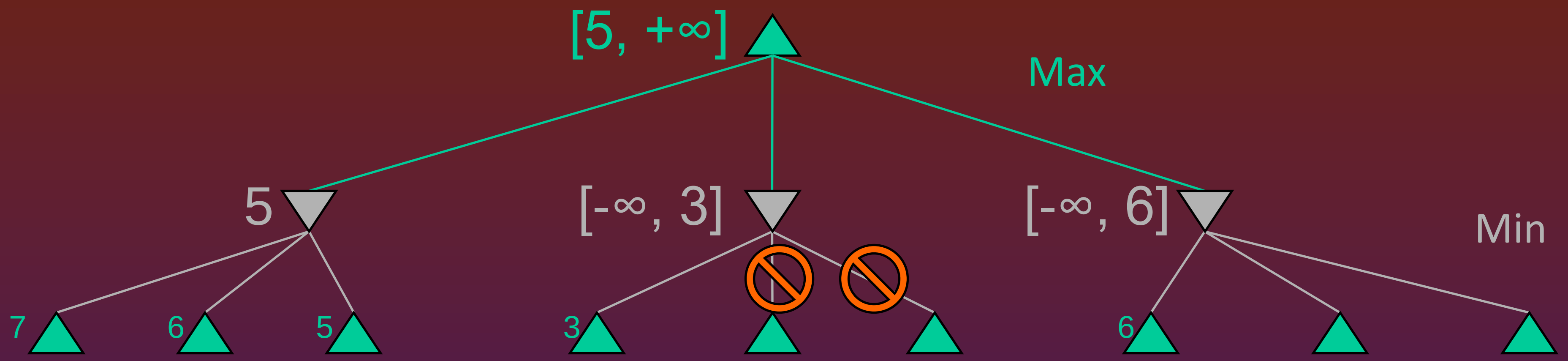
- Min continues with the next sub-tree, and gets a better value
- Max has a better choice from its perspective, however, and will not consider a move in the sub-tree currently explored by Min
  - initially [-∞, +∞]



α best choice for Max 5

β best choice for Min 3

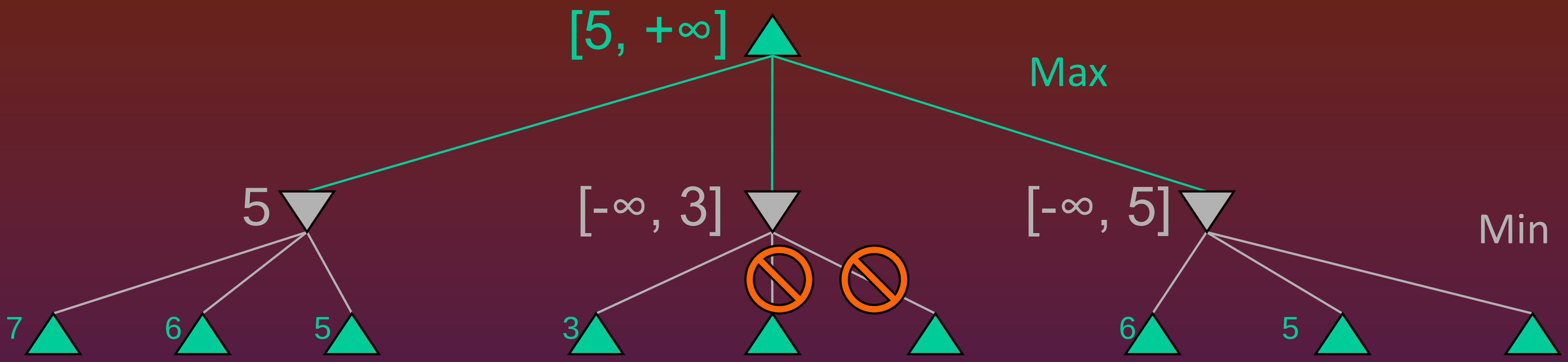
- Min knows that Max won’t consider a move to this sub-tree, and abandons it
- this is a case of *pruning*, indicated by 🚫



α best choice for Max 5

β best choice for Min 3

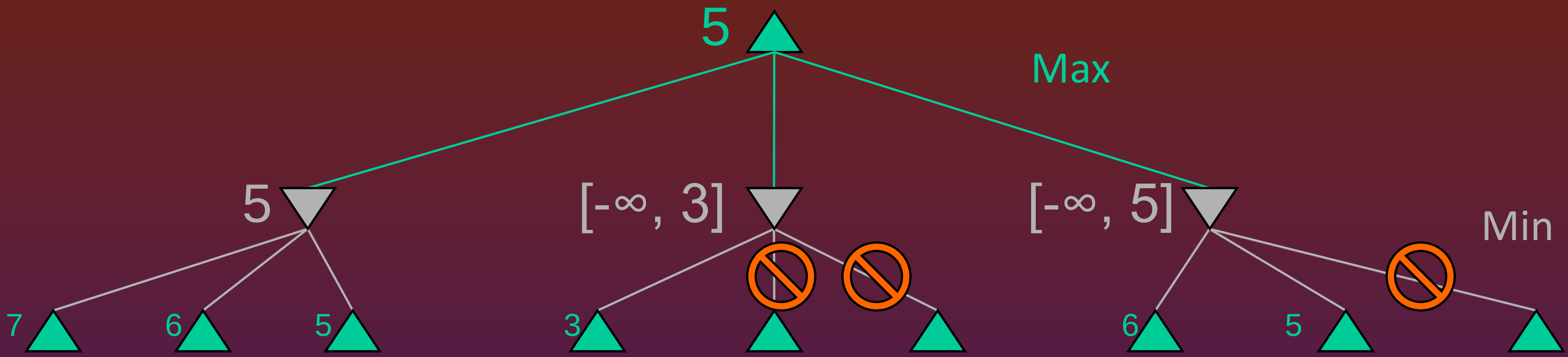
- Min explores the next sub-tree, and finds a value that is worse than the other nodes at this level
- if Min is not able to find something lower, then Max will choose this branch, so Min must explore more successor nodes



α best choice for Max 5

β best choice for Min 3

- Min is lucky, and finds a value that is the same as the current worst value at this level
- Max can choose this branch, or the other branch with the same value

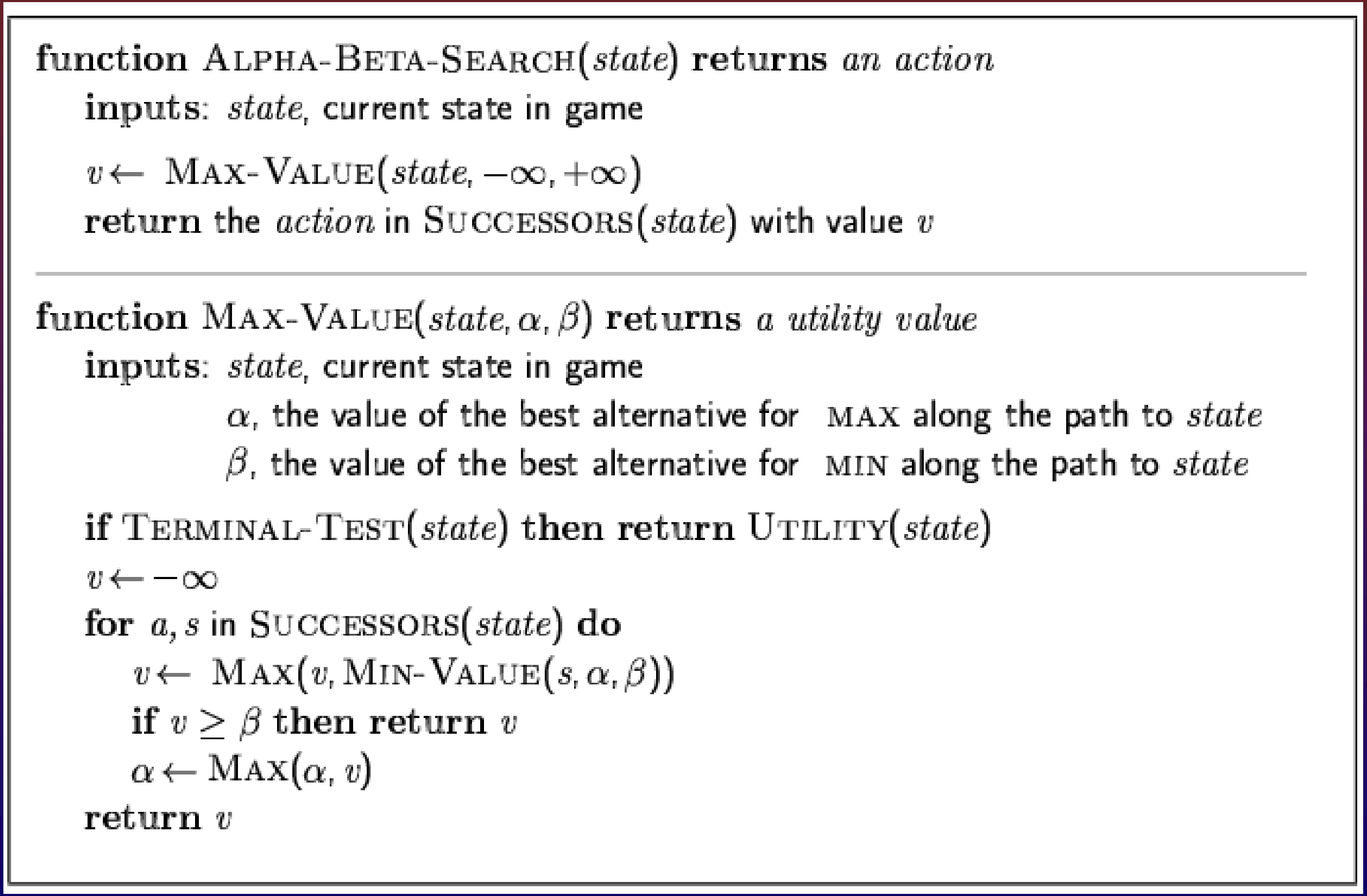


α best choice for Max 5

β best choice for Min 3

- Min could continue searching this sub-tree to see if there is a value that is less than the current worst alternative in order to give Max as few choices as possible
  - this depends on the specific implementation
- Max knows the best value for its sub-tree

The α-β algorithm



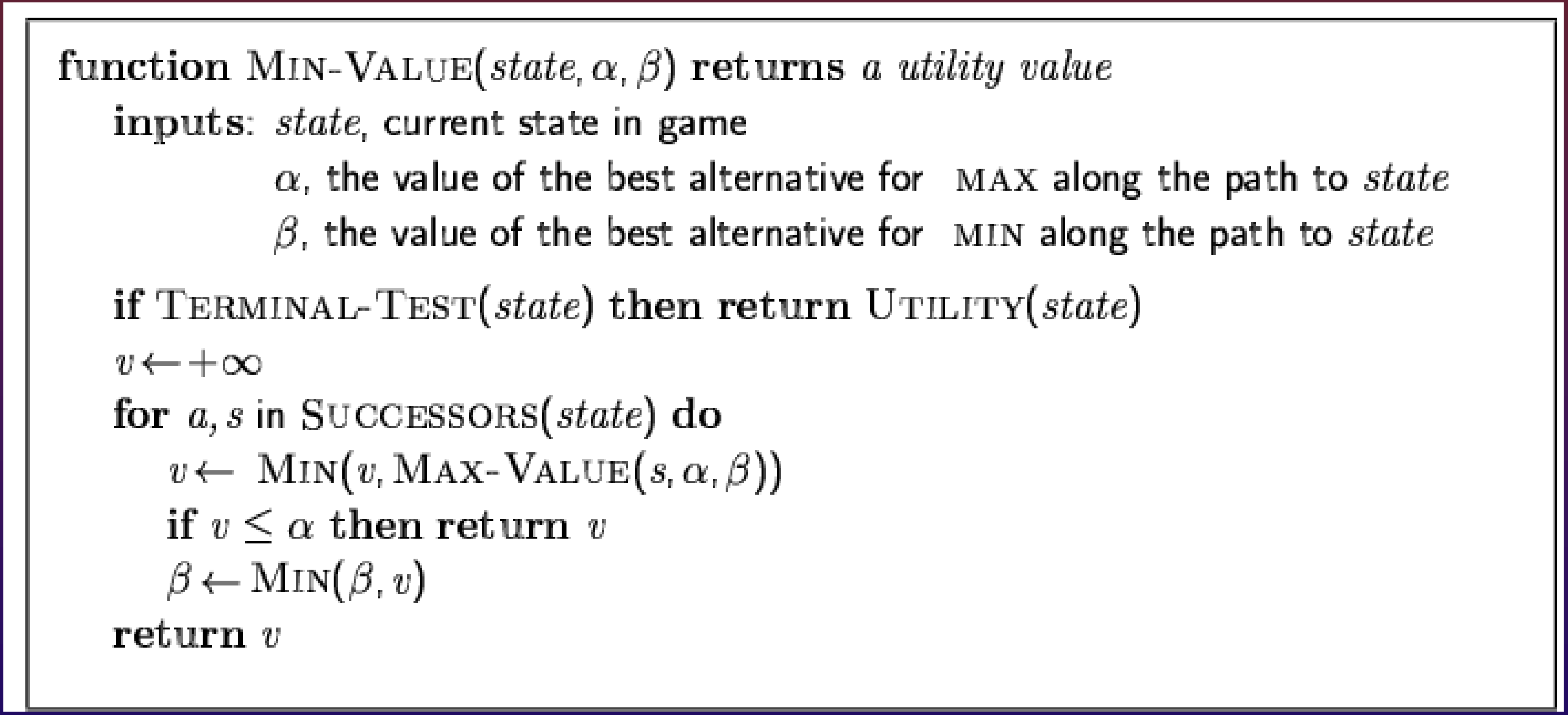
```
function ALPHA-BETA-SEARCH($state$) returns $an\ action$
   inputs: $state$, current state in game

   $v \leftarrow$ 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 $v$
```

Cons of α-β pruning

1. Not always optimal node ordering.
2. At least for a part of the tree we have to reach bottom level of the tree. This can be a problem for a very deep game trees.

If the game tree is too deep

1. Cut the tree at some appropriate depth.

2. Except of real minimax values, we used estimated heuristic values. Values are outputs of the heuristic evaluation function.

Evaluation function : gives values of nodes at the cut offed level of the tree.

If good: -orders the nodes the same way as the exact utility function

-heuristic values calculation cannot be too time consuming

-heuristic values should give a good estimate of the win chance

Cutting off search

Can be used as a variant of α-β pruning. If the branch is cut off sooner then at the final tree depth, evaluation function estimates its value. Estimated values are further taken as MinMax values.

Evaluation functions – some remarks

Weighted game features, for example. Features are chosen and functions created by experts.

Chess

Stone values, strengths

Number of certain stone types at certain positions

- 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.

Cons of linear evaluation functions

- they are based on the anticipation that features are independent, which is not always true (for example strengths of chess stone is independent of the strengths and configuration of other chess stones – definitely not true)

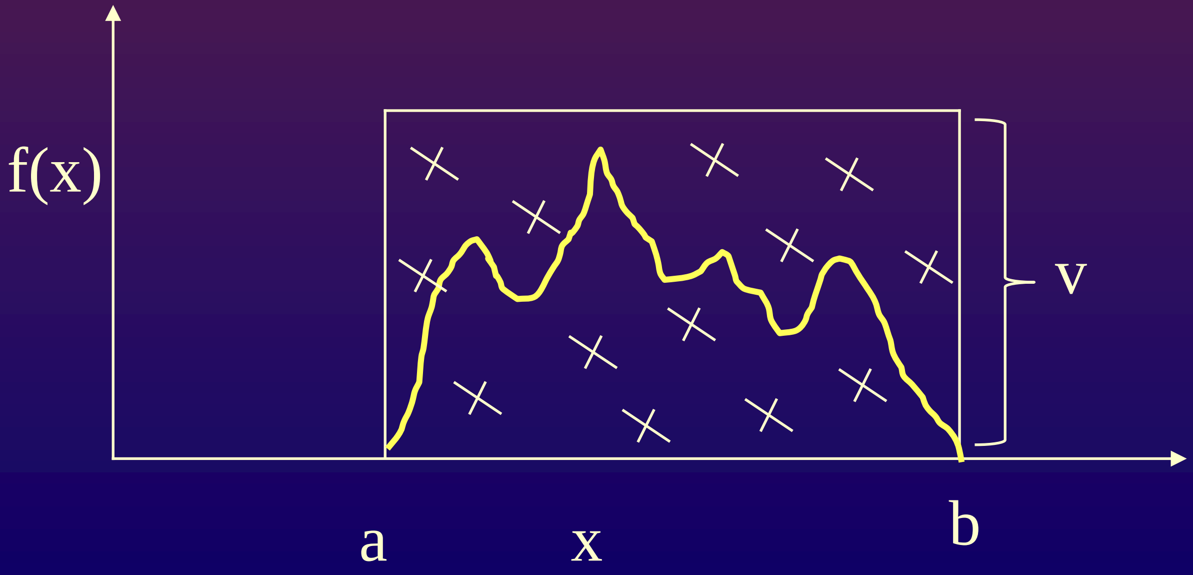
$\downarrow$

Therefore complicated nonlinear evaluation functions were created for various games, taking into account strength of the feature combination.

Using Monte Carlo methods in the games

Monte Carlo principle:

We want to calculate a complicated definite integral which is difficult to do analytically.



1. Definite integral is an area under the curve.
2. Close it into the rectangle.
3. Generate random points in the rectangle.
4. Estimate the integral I.

$$P_{rect} = (b-a)v$$

$$P_{rec} / I \approx N_{rec} / N_I$$

$$I \approx (N_I / N_{rec})(b-a)v$$

Using Monte Carlo methods in the games

Monte Carlo method are based on sampling.

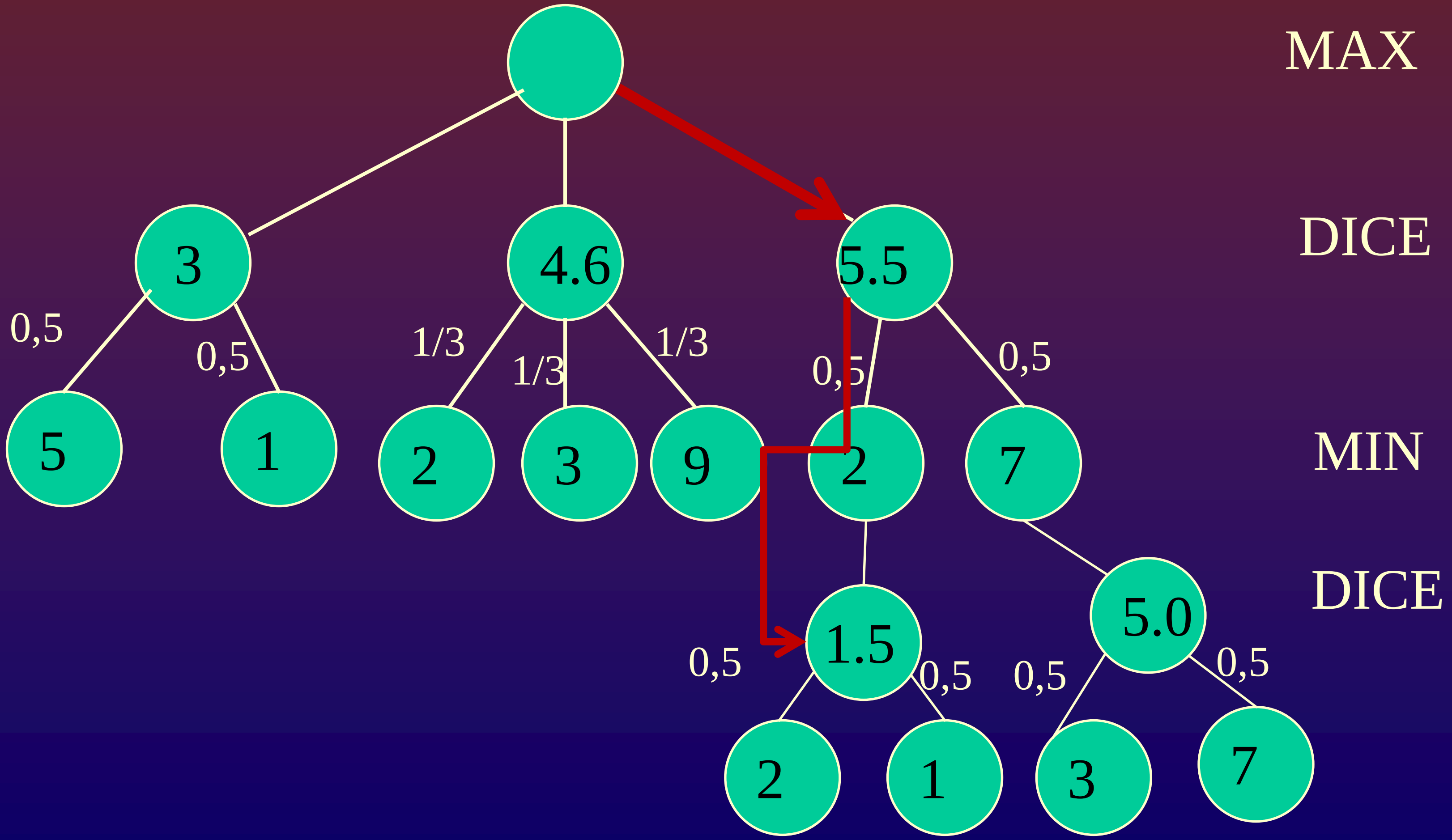
1. Let players play the game. At the first step then can make random moves.
2. This way the search tree is sampled.
3. Evaluate number of wins and losses for the games going through certain nodes.

More about it in the master course of AI.

Expectiminimax

- Algorithm searching for the optimal player moves, if they depend on some uncertainty, for example dice throwing.
- Example: backgammon.
- Minimax tree contains MIN, MAX and DICE levels.

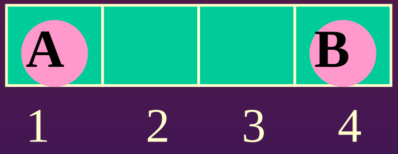
Expectiminimax tree example



Questions

1. If MinMax values are multiplied by some constant, does it influence the best moves? If yes , why? If no, why?

2. Let us have a game:



-A, B players exchange moves. Each have to move to a closest empty square independently of the direction.

-if opponent occupies the closest square, player can jump over to the next closest one

- (1, ak prvý skončí A, -1 ak B, 0 ak obaja naraz) game finishes if A is at the position of B and vice versa (1, if A finishes first, -1 if B, 0 if both)

Tasks:

Draw a complete game tree. State is a vector containing positions of both agents.

Evaluate terminal states.

Generalize for n squares and prove that n>2 A wins if n is even and B wins if n is odd.

Summary

1. Game and „zero sum“ game definition.
2. MINIMAX algorithm and its variants.

   -alfa-beta pruning

   -cutoff search

3. Evaluation function remarks
4. Expectiminimax. Monte Carlo in games.

Logical Agents I

UI VI

Mária Markošová

Summary from the last lecture

1. Game in AI, definition, one player games
2. Multiplayer games, simultaneous games, sequential games.
3. MINIMAX algorithm and zero sum games.
4. Alpha beta pruning.
5. Cut off search and evaluation functions.
6. Monte Carlo method and games.

Outline

1. Logical agents
2. Wumpus world and knowledge base.
3. Propositional logic.
4. Inference methods in propositional knowledge base.

Logical agents

Knowledge base agent: KB agent

KB agent has a knowledge base (KB) implemented in which knowledges about the environment and its changes are stored. KB agent has inference methods implemented, together with learning unit. KB agent is able to learn, inference a new knowledges from the old ones. For logical agents KB consists of logical sentences. Propositional KB means, that KB is written in the propositional logic.

Knowledge based agent

What contains KB agent?

1. KB, knowledge base in which knowledges about the world in which agent acts are stored. .
2. Background knowledge base; can be empty
3. Inference mechanism.

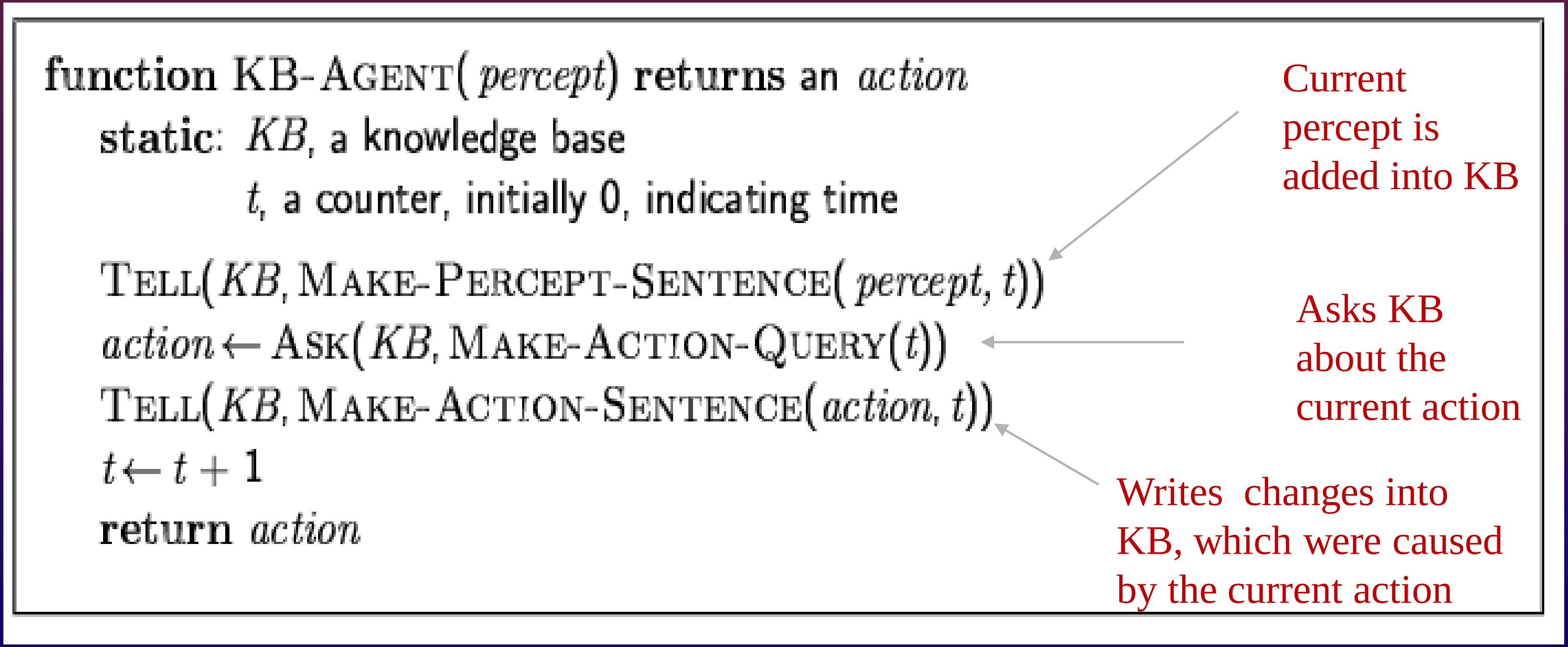
How the KB agent works?

a) It tells KB, what it currently percepts using the **TELL function**.

b) It asks KB what is the best current action. **ASK function**.

Tell: Sentence adding function, adds new sentences into KB

Ask: Function asking about the knowledges stored in the KB, function wich chooses the best possible agent action.



```
function KB-AGENT(percept) returns an action
    static: KB, a knowledge base
            t, a counter, initially 0, indicating time

    TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
    action ← ASK(KB, MAKE-ACTION-QUERY(t))
    TELL(KB, MAKE-ACTION-SENTENCE(action, t))
    t ← t + 1
    return action
```

Current percept is added into KB

Asks KB about the current action

Writes changes into KB, which were caused by the current action

Wumpus world

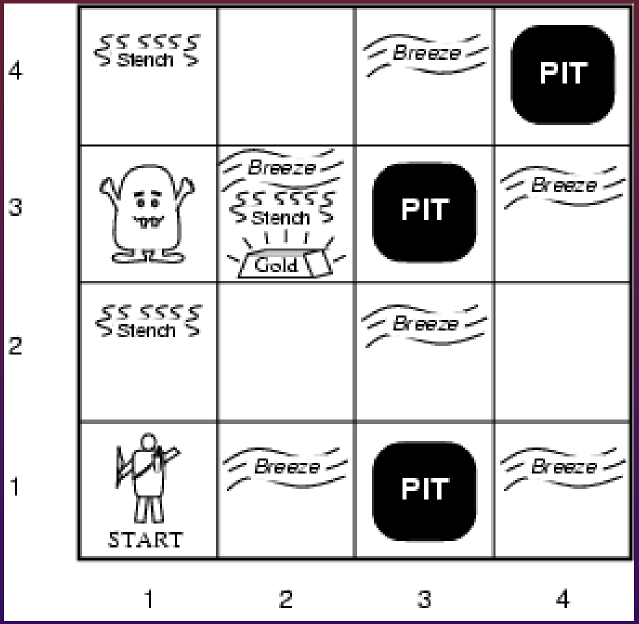
Environment: Wumpus world is a square lattice. Agent is placed at certain square, wumpus, gold and pits are environment features. ***Start*****:** agent at the position [1,1]

Goal**:** To find a gold, to avoid a pit, not to be eaten by wumpus, bring the gold to the position [1,1]

*Agent***:** agent does not percept its own position, agent knows the state of the current square

Wumpus World PEAS description (Russel, Norwig)

- **P**erformance measure
  - gold +1000, death -1000
  - -1 per step, -10 for using the arrow
- **E**nvironment
  - Squares adjacent to wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter if gold is in the same square
  - Shooting kills wumpus if you are facing it
  - Shooting uses up the only arrow
  - Grabbing picks up gold if in same square
  - Releasing drops the gold in same square



agent bumped the wall

wmpus is killed

- **S**ensors: Stench, Breeze, Glitter, Bump, Scream
- **A**ctuators: Left turn, Right turn, Forward, Grab, Release, Shoot

Percept: five position vector

(stench, breeze,glitter,bumb,scream)

(zápach, vánok,ligot,buchnutie,výkrik)

Agent percepts gold

Agent bumps into thr wall

Agent hits wumpus by arrow

Simple percept representation?

binary vector of the length 5

Current percept: example

(stench, breeze, none, none ,none)

or

(1, 1, 0, 0 ,0)

„Wumpus world“ features

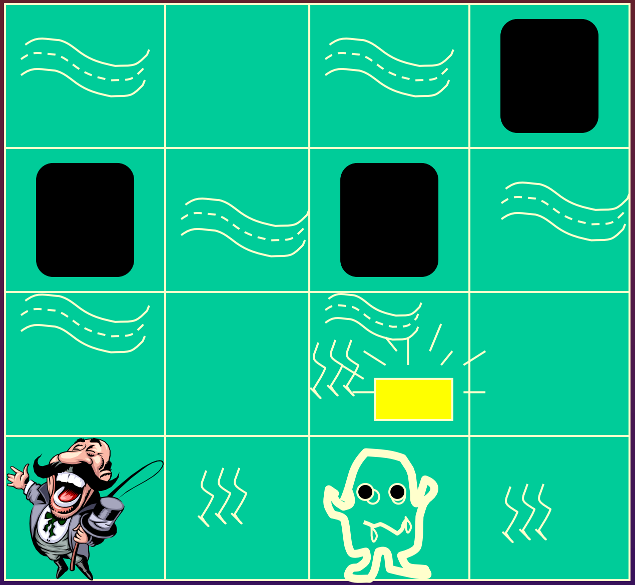
- Observable? No – percept of the agent is local
- Deterministic? Yes
- Epizodic? No
- Static Yes – wumpus and pits are features, not agents
- Discret? Yes
- Single agent? Yes, wumpus is a property of the environment not an agent

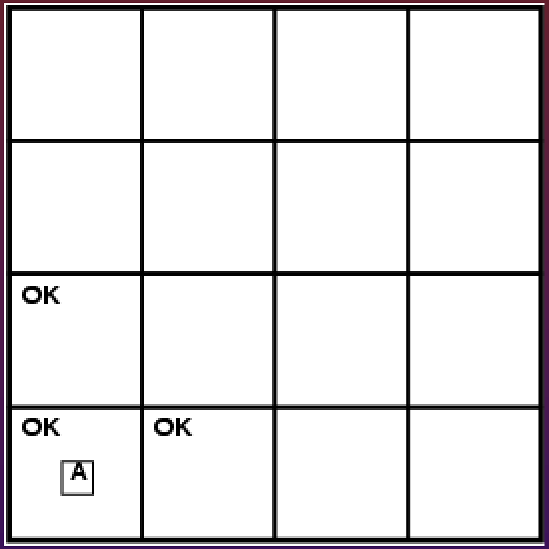
Background KB for the „wumpus world“

Background KB contains knowledges common for all wumpus worlds, for all configurations of pits, gold, wumpus.

This KB contains sentences as:
-neighbour squares of wumpus are „smelly“
-squares near pits are „breezy“
-in the wumpus worlds is exactly one wumpus and exactly one agent
-agent starts at the position (1,1)
-etc

Exploring wumpus world





breeze (vánok)

stench (zápach)

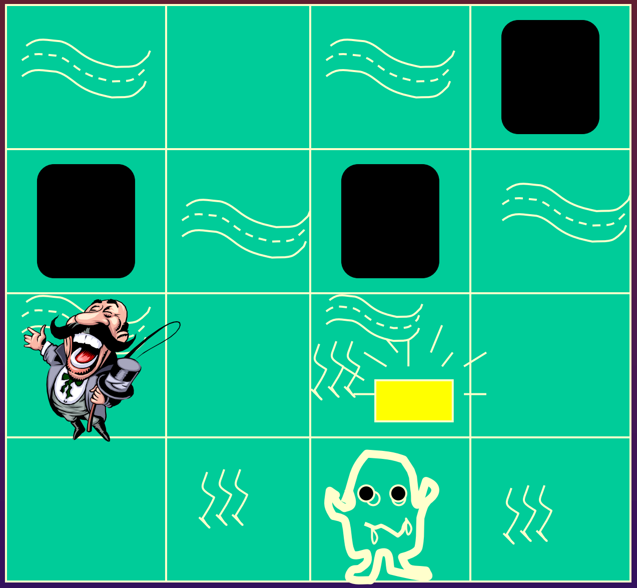
(stench, breeze,glitter,bumb,scream)

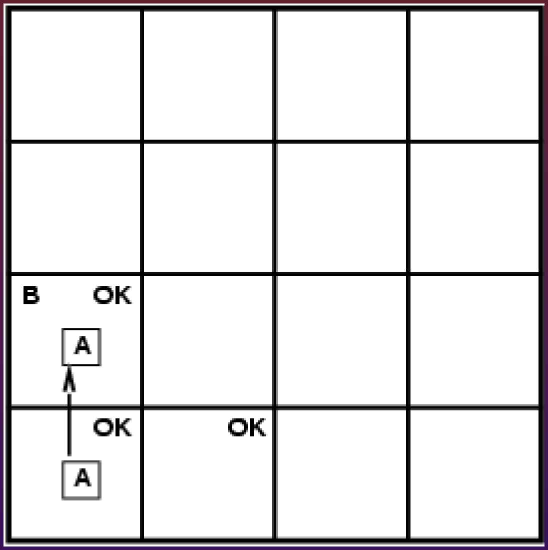
(none,none,none,none,none)

(0,0,0,0,0)

Current percept

Exploring a wumpus world





breeze

stench

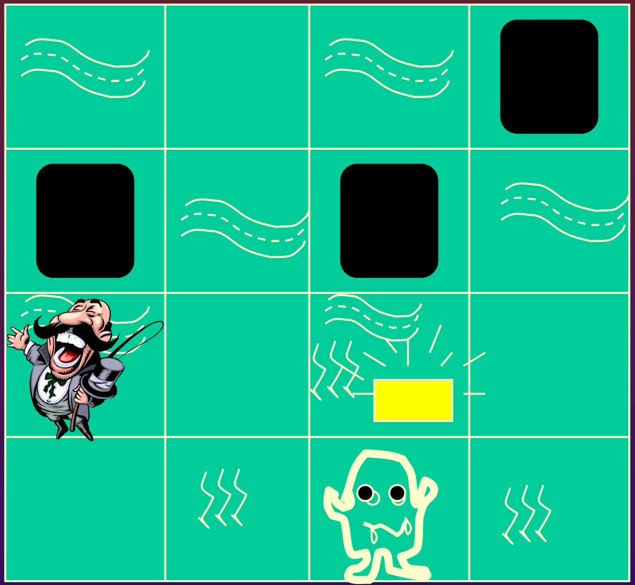
(stench, breeze,glitter,bumb,scream)

(none,breeze,none,none,none)

(0,1,0,0,0)

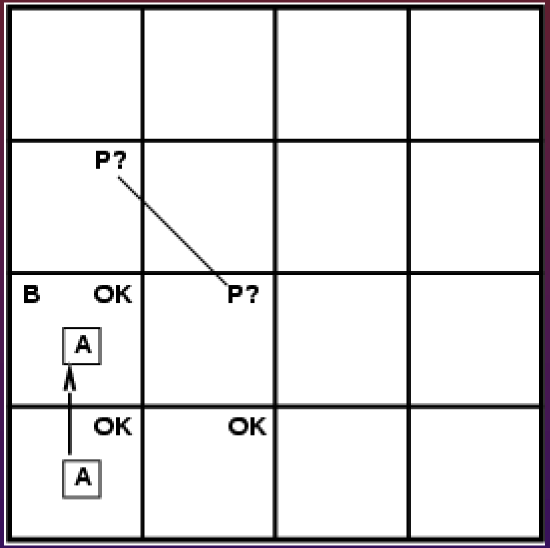
Current percept

Exploring a wumpus world



breeze

stench

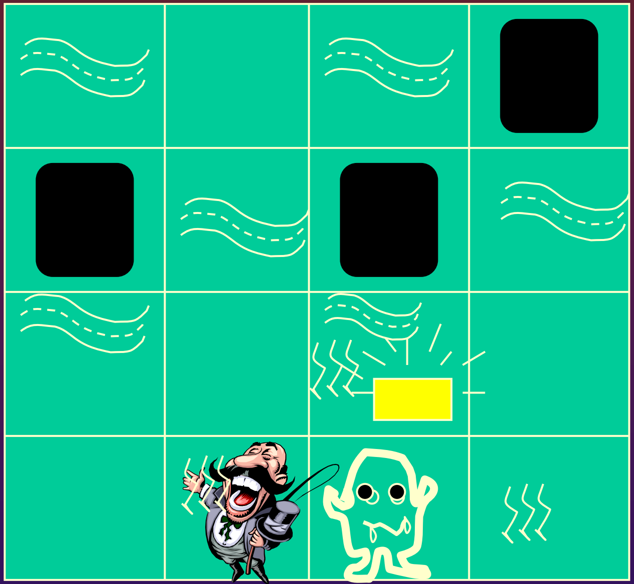


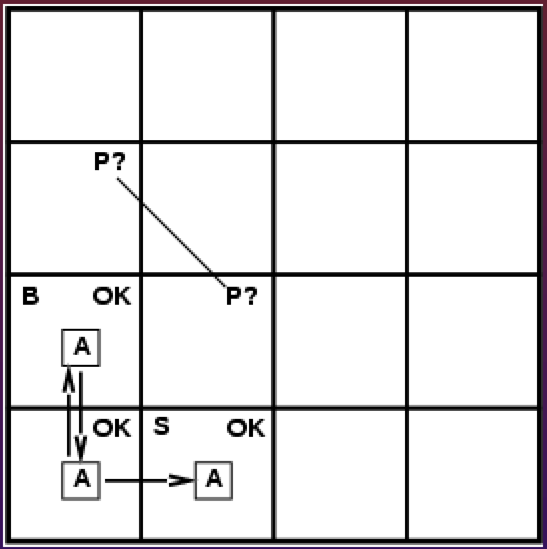
(stench, breeze,glitter,bumb,scream)

(none,breeze,none,none,none)

Current percept

Exploring a wumpus world





breeze

stench

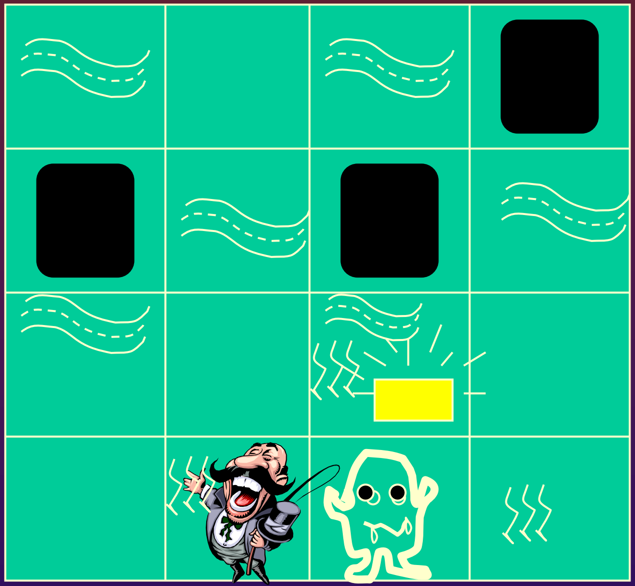
(stench, breeze,glitter,bumb,scream)

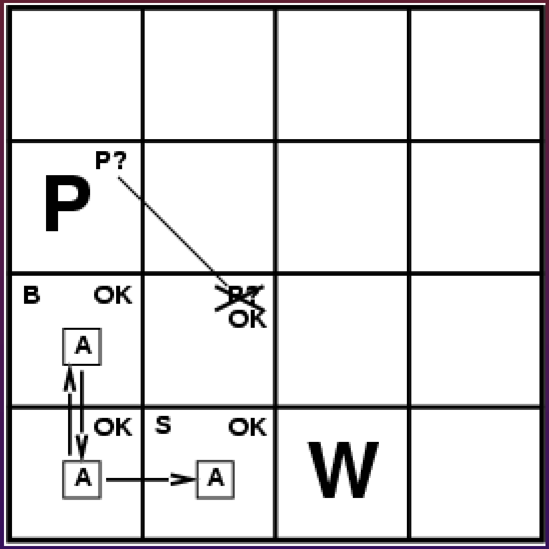
(stench,none,none,none,none)

(1,0,0,0,0)

Current percept

Exploring a wumpus world

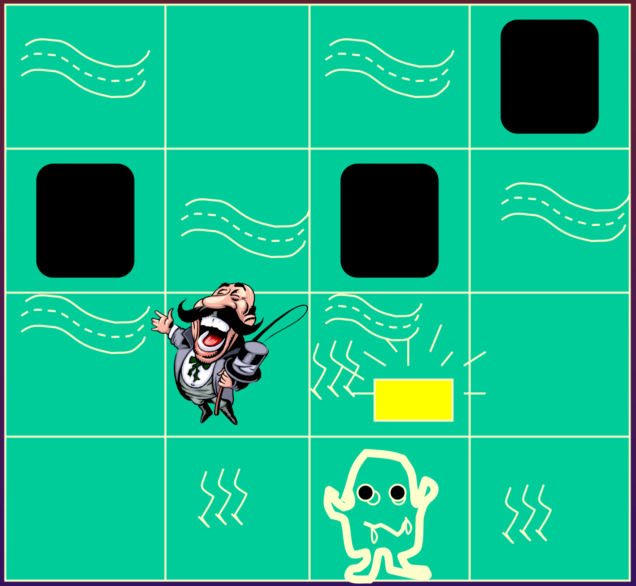


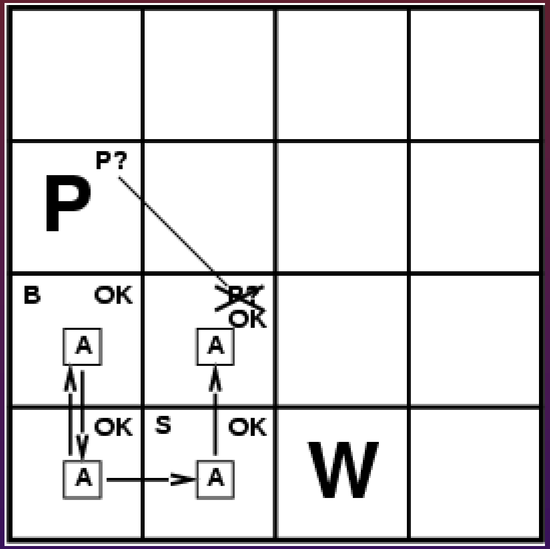


breeze

stench

Exploring a wumpus world





breeze

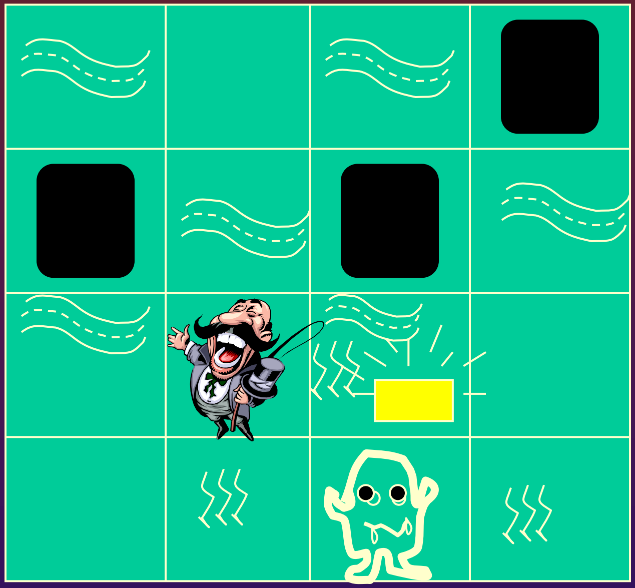
stench

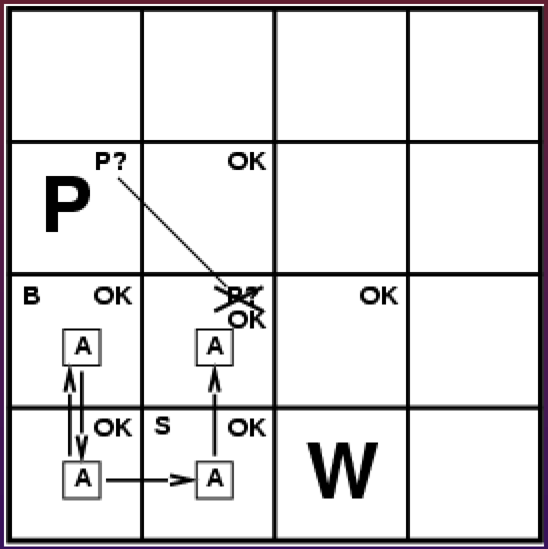
(stench, breeze,glitter,bumb,scream)

(0,0,0,0,0)

Current percept

Exploring a wumpus world

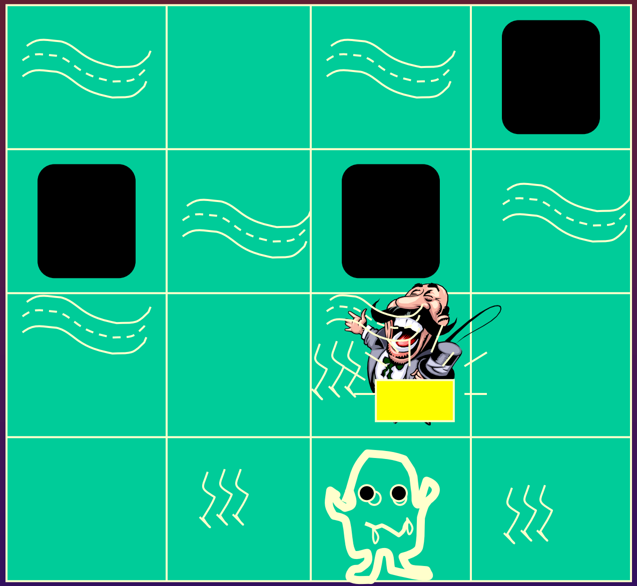


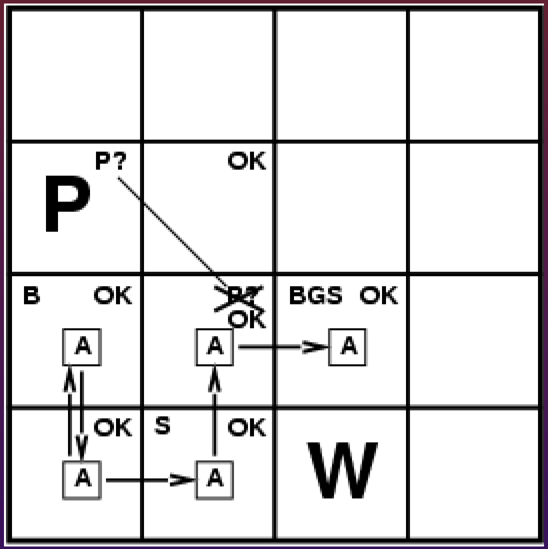


breeze

stench

Exploring a wumpus world





breeze

stench

(stench, breeze,glitter,bumb,scream)

(1,1,1,0,0)

aktuálny vnem

How to tell KB new percepts and knowledges? By what kind of language?

$$\downarrow$$

Logical agent: the language is logic

What is logic?

It is a science about correct reasoning on the symbolic level. The concrete content of sentences is ignored.

-Propositional logic
-First order logic

Logic

Logic consists of:

a) Formal problem describing system syntax and semantics

b) Theory of logical proofs and logical reasoning, rules of logical reasoning.

Logical reasoning: On the basis of logical rules and relations we derive from the known sentences the unknown ones. We are dealing with propositional logic in this lecture.

Interpretation: It is an assignment of meaning to the logical symbols. In propositional logic it is a mathematical abstraction of the real world in which we know the truth values of each sentence.

Model: Mathematical abstraction of the real world in which the sentence has a value True or 1 which means true.

Example: logical sentence

$$A \wedge (B \Rightarrow D) \wedge (A \vee D) \wedge (\neg C) \wedge (C \vee A)$$

Model of the sentence

1 1 1 1 1 1 0 1