CECS545-Artificial Intelligence

Project 4

Dr. Roman Yampolskiy

Project 4: TSP – Genetic Algorithm

- Learning objectives. At the completion of this project, you should be able to:
  - Implement a genetic algorithm for an intractable problem.
  - Be able to discuss aspects of the GA that control success in finding good solutions.

Problem

- You will be expected to implement a GA for a TSP dataset with up to 100 cities
- Data for each problem will be supplied

Hints

- Analyze the results of two different settings for two different parameters. For example you can use two different types of crossover methods and two different mutation rates. In that case, you need to collect four sets of data:

	Crossover A	Crossover B
Mutation 1	Dataset 1A	Dataset 1B
Mutation 2	Dataset 2A	Dataset 2B

Hints

- For each of the four datasets, you will need to run your code several times to develop performance statistics.
- I use the example of two crossovers and two mutations. You can select any modification of GA so long as you can have (at least) 2 options for each modification. For example you could choose two different sizes of populations. You could choose two different selection methods, etc.
- Don’t forget, if you run your code a 100 times on a large dataset, you will likely wind up with nearly 100 different solutions.

Deliverables

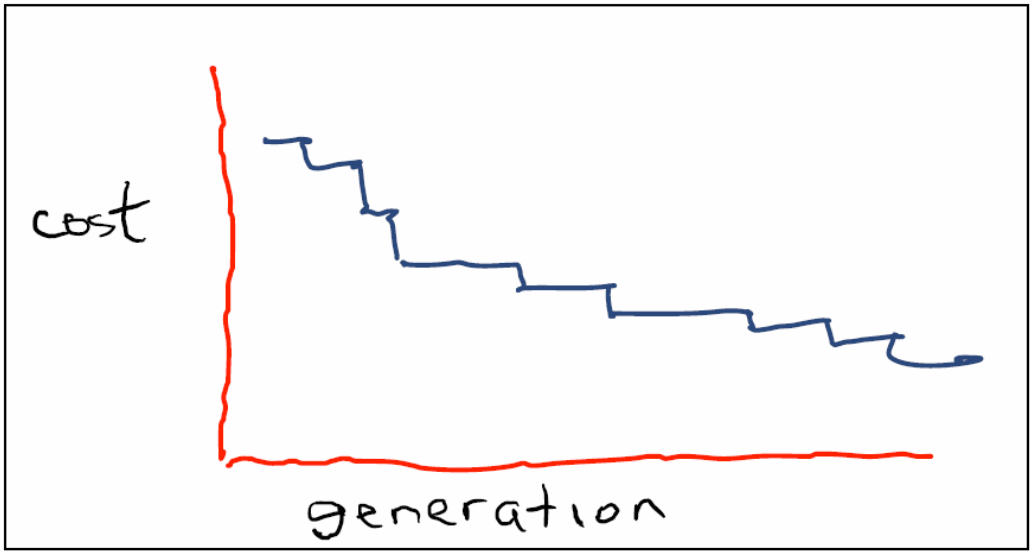
- Well-commented source code for your project. You can use any language you like, but I reserve the right to ask you to demo performance of your algorithm on a new dataset.
- Include a GUI with visual (and dynamic) representation of the solutions for this project.
- Project report (3-4 pages).

Deliverables

- Define your crossover and mutation methods. Or else define whichever two GA elements you chose to investigate. Clearly identify if your methods were your idea or else the source from which you got the idea.
- Define your stopping criteria.
- Clearly identify your best solution for the solved problem.
- Analyze the effectiveness of the two chosen variations in GA. To do this you will likely need to run your code several times and provide the min, max, average and standard deviation of each set.

Deliverables

- Briefly discuss how long a typical run took for each of the datasets.
- Graphically present your improvement curves.



Critical Thinking

- Describe the biggest problem that you had in the implementation and analysis of this project.
- Describe the one thing you would change if you did this project again.
- Please elaborate on what you learned from using GA. At least one paragraph, please.
- Give your overall impressions of GA as problem solving technique.

CECS545-Artificial Intelligence

Adversarial Search

Dr. Roman Yampolskiy

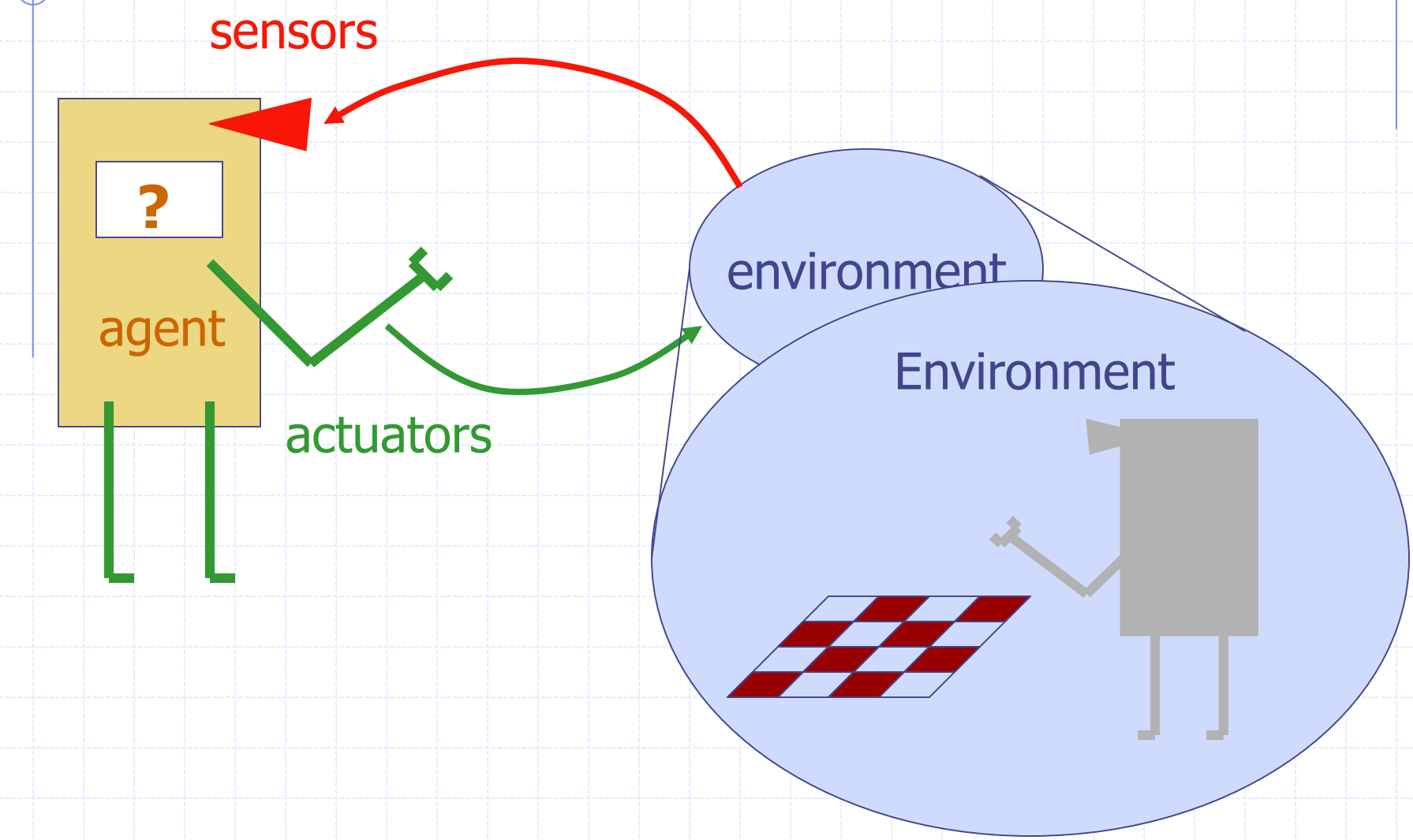
Topics

- Game playing
- Game trees
  - Minimax
  - Alpha-beta pruning
- Examples

What & Why

- Games are a form of multi-agent environment
  - What do other agents do and how do they affect our success?
  - Cooperative vs. competitive
  - Competitive multi-agent environments give rise to **adversarial search** a.k.a. ***games***
- Why study games?
  - Fun; historically entertaining
  - Interesting subject of study because they are hard
  - Easy to represent and agents restricted to small number of actions

Game-Playing Agent



Relation of Games to Search

- Search – no adversary
  - Solution is (heuristic) method for finding goal
  - Heuristics and CSP techniques can find *optimal* solution
  - Evaluation function: estimate of cost from start to goal through given node
  - Examples: path planning, scheduling activities
- Games – adversary
  - Solution is strategy (strategy specifies move for every possible opponent reply).
  - Time limits force an *approximate* solution
  - Evaluation function: evaluate "goodness" of game position
  - Examples: chess, checkers, Othello, backgammon

Types of games

	deterministic	chance
perfect information	chess, checkers, go, othello	backgammon monopoly
imperfect information	battleships, blind tictactoe	bridge, poker, scrabble nuclear war

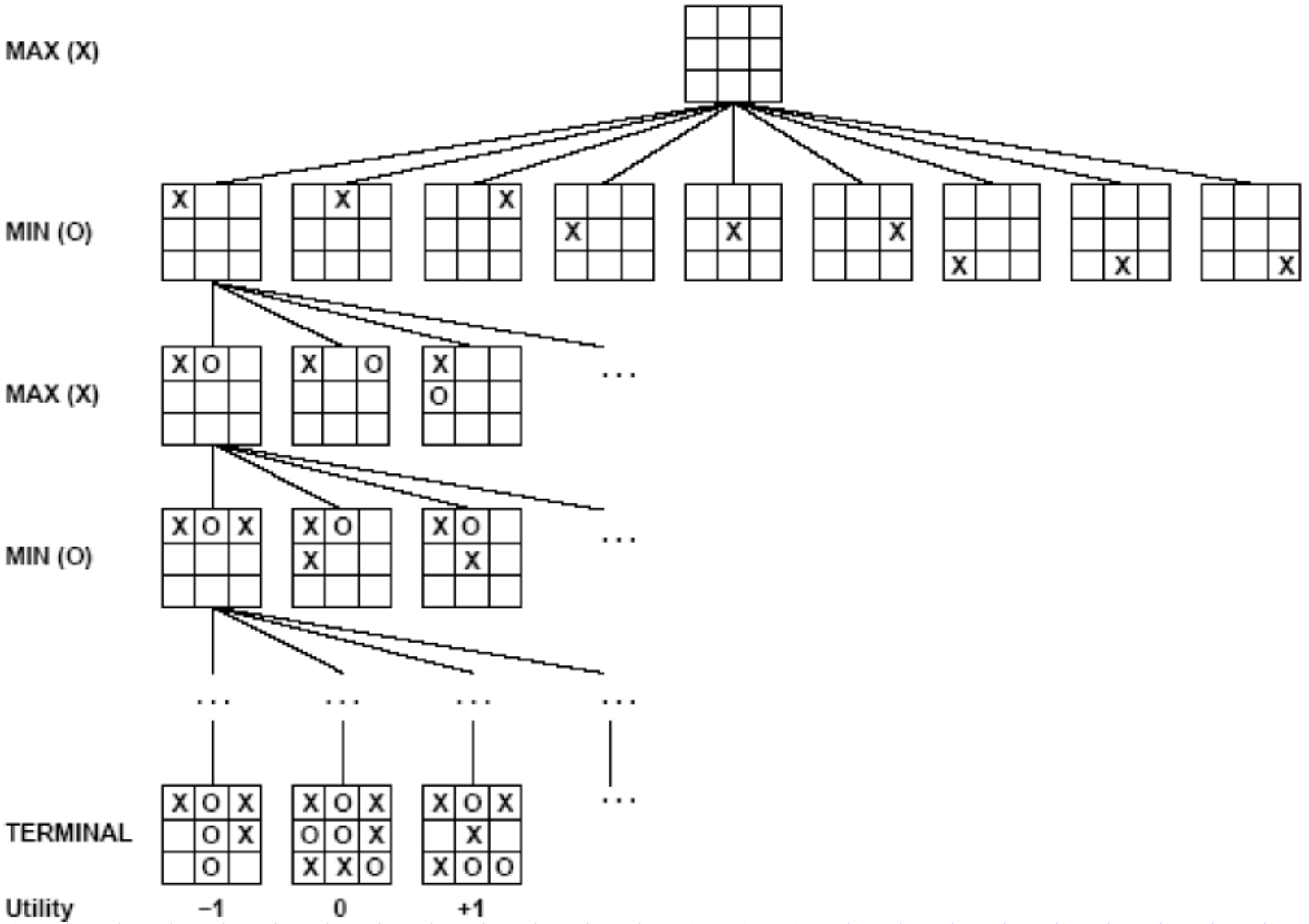
Typical case

- 2-person game
- Players alternate moves
- **Zero-sum**: one player's loss is the other's gain
- **Perfect information**: both players have access to complete information about the state of the game. No information is hidden from either player.
- No chance (e.g., using dice) involved
- Examples: Tic-Tac-Toe, Checkers, Chess, Go, Nim, Othello
- Not: Bridge, Solitaire, Backgammon, ...

Game Search Formulation

- Two players MAX and MIN take turns (with MAX playing first)
- State space
- Initial state
- Successor function
- Terminal test
- Utility function, that tells whether a terminal state is a win (for MAX), a loss, or a draw
- MAX uses search tree to determine next move.

Partial Game Tree for Tic-Tac-Toe



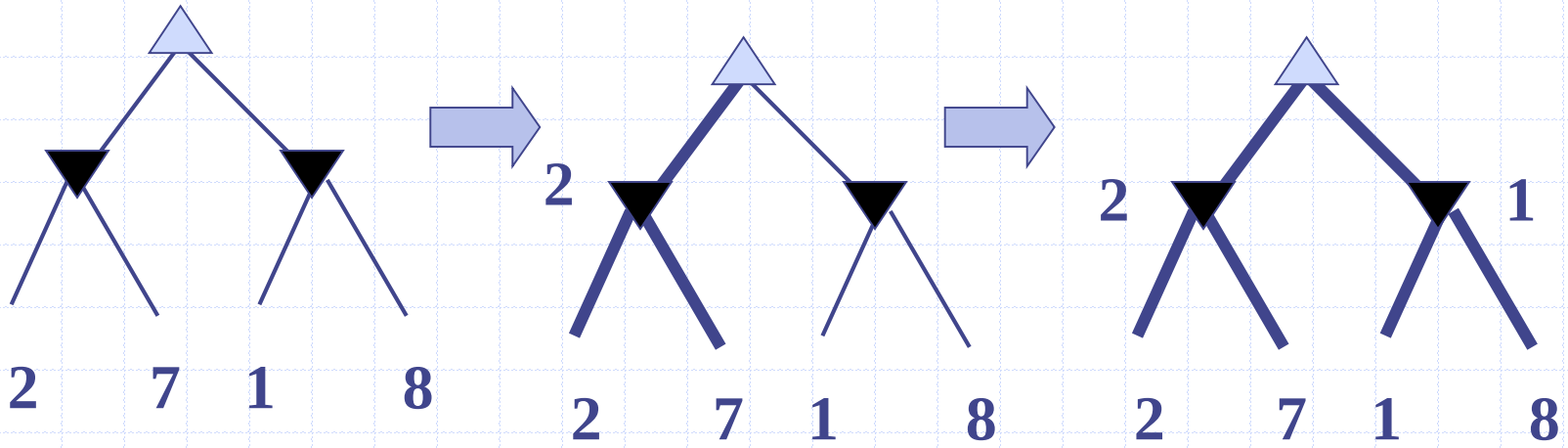
Optimal strategies

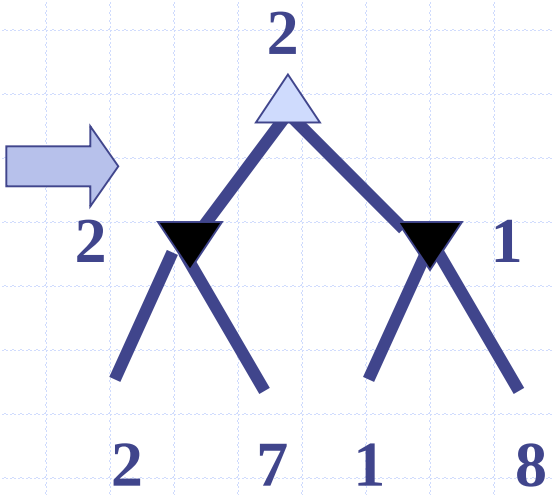
- Find the contingent **strategy** for MAX assuming an infallible MIN opponent.
- Assumption: Both players play optimally !!
- Given a game tree, the optimal strategy can be determined by using the minimax value of each node:

MINIMAX-VALUE(n)=

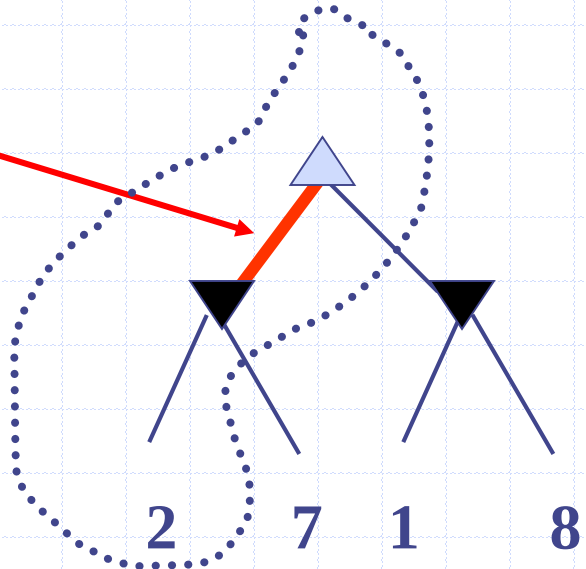
UTILITY(n)	If n is a terminal
$\max_{s \in successors(n)}$ MINIMAX-VALUE(s)	If n is a max node
$\min_{s \in successors(n)}$ MINIMAX-VALUE(s)	If n is a min node

Optimal Play





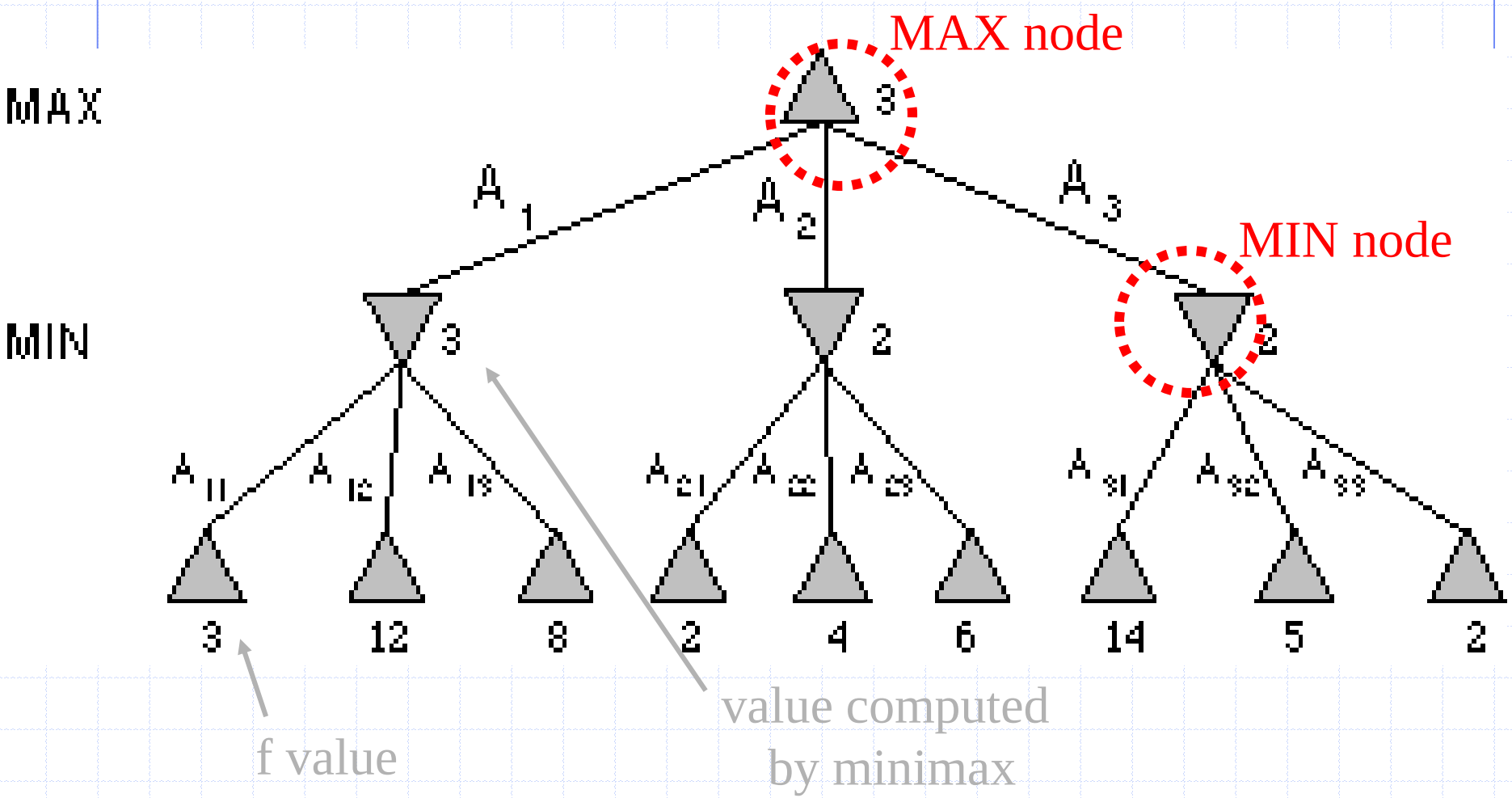
This is the optimal play



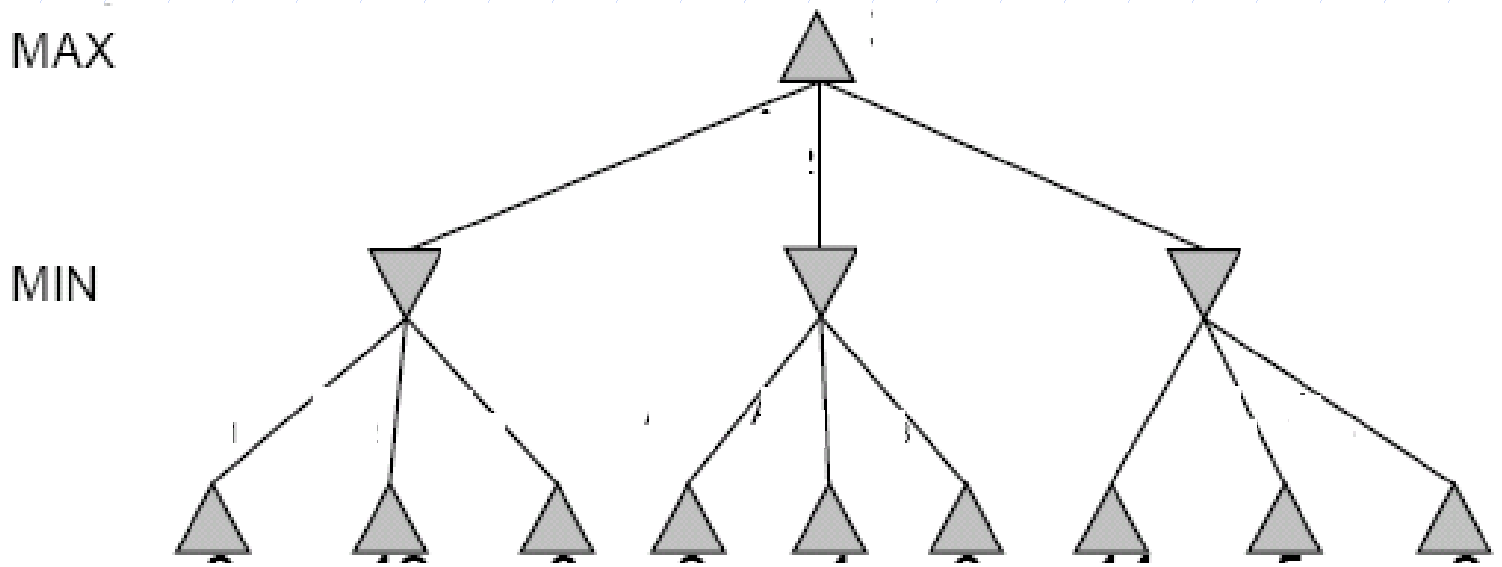
MAX

MIN

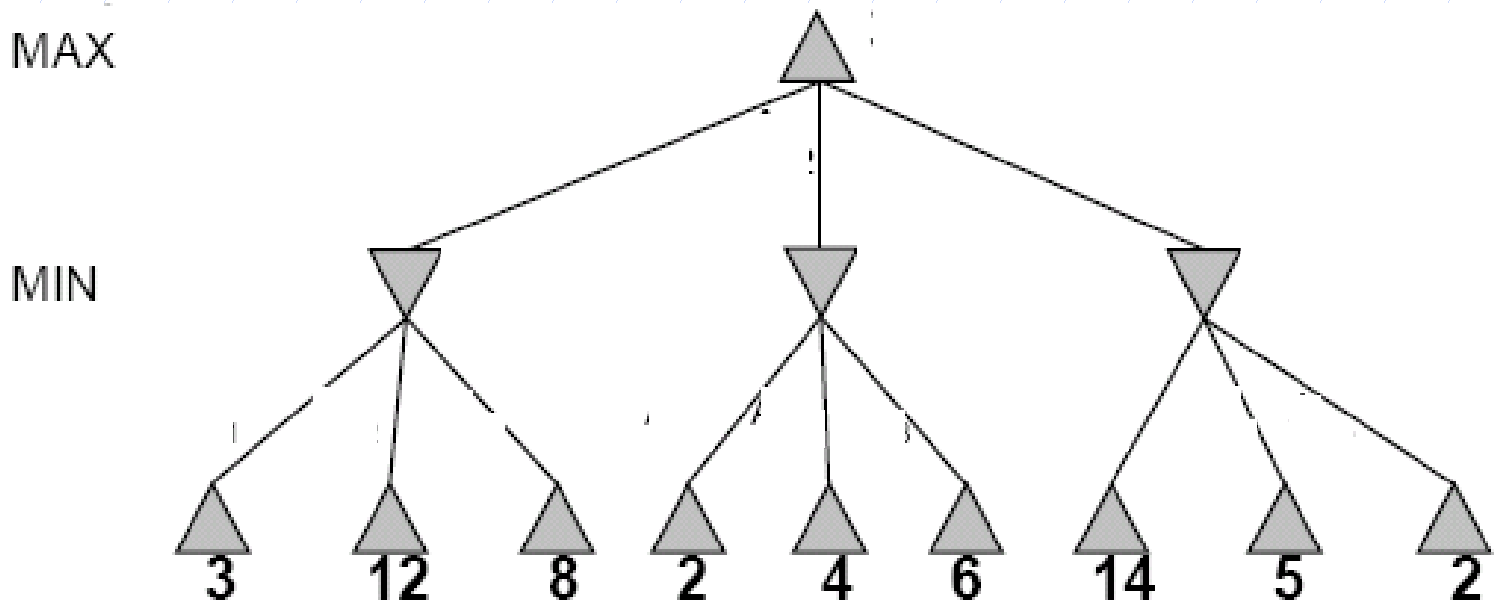
Minimax Tree



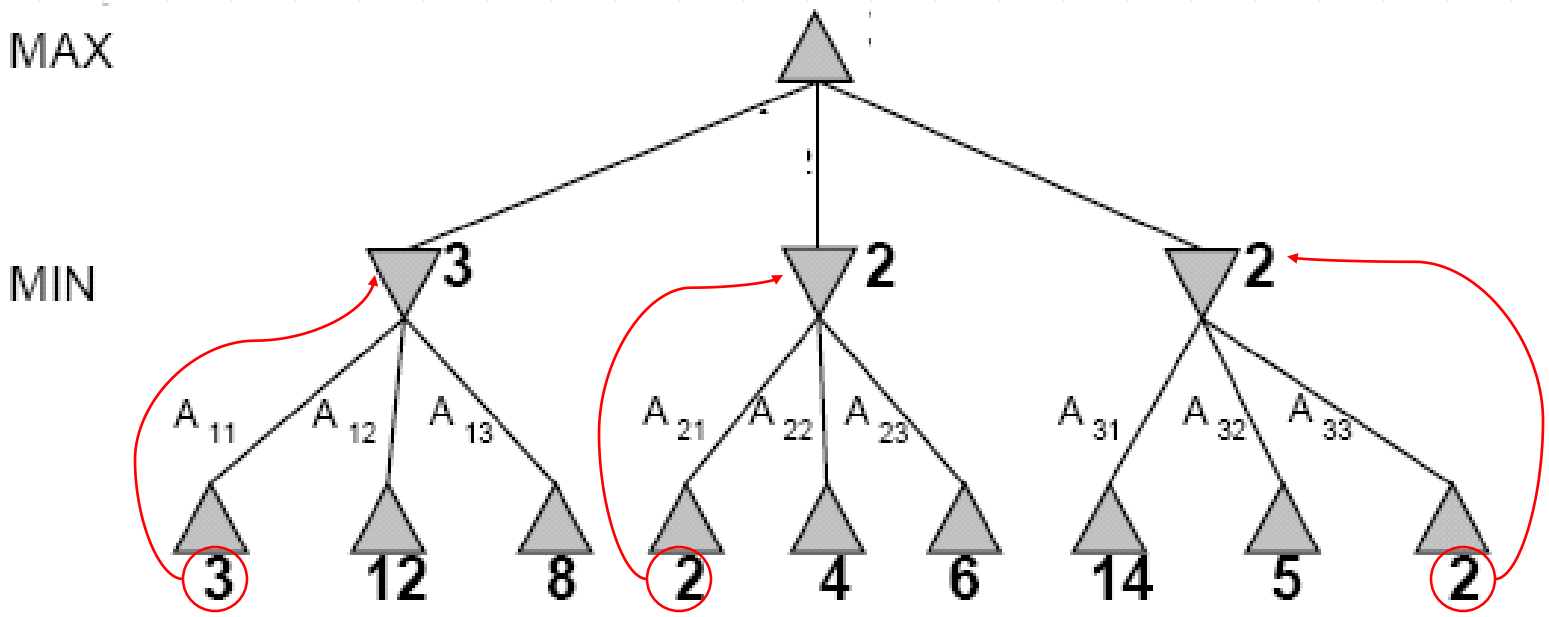
Two-Ply Game Tree



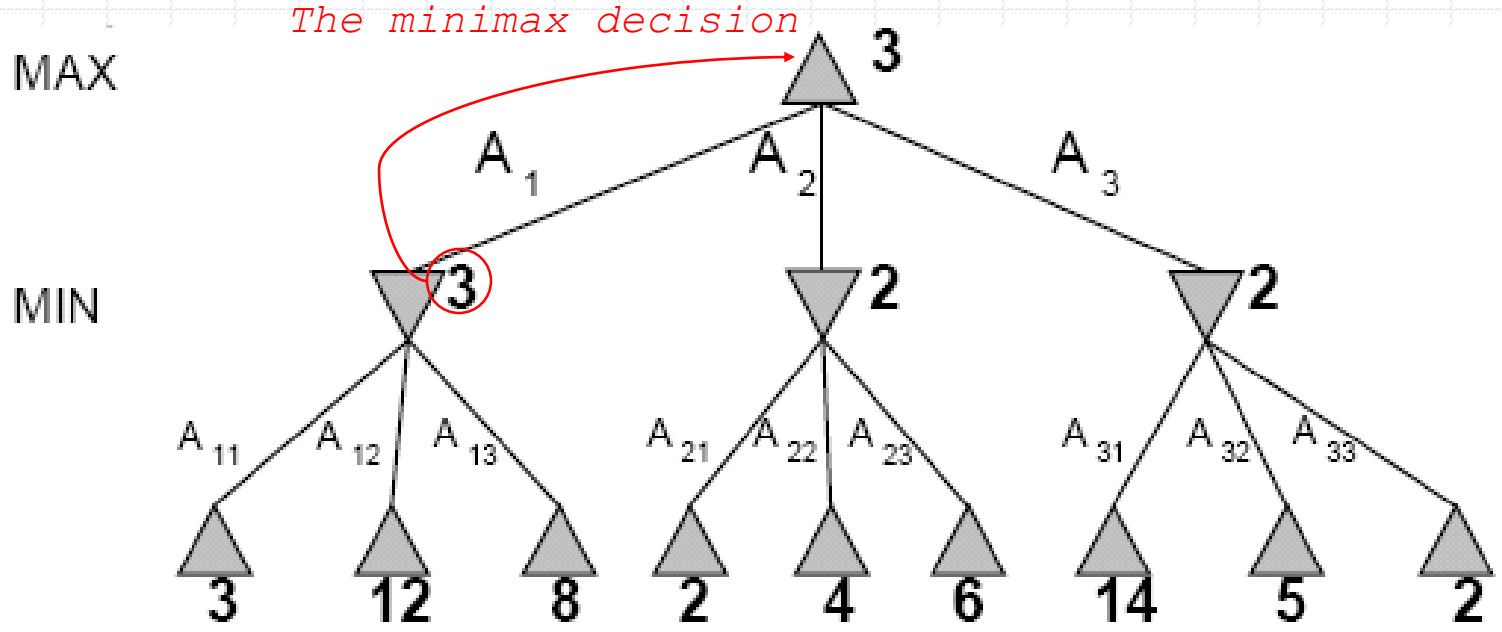
Two-Ply Game Tree



Two-Ply Game Tree



Two-Ply Game Tree



Minimax maximizes the worst-case outcome for max.

What if MIN does not play optimally?

- Definition of optimal play for MAX assumes MIN plays optimally: maximizes worst-case outcome for MAX.
- But if MIN does not play optimally, MAX can do even better.

Properties of Minimax

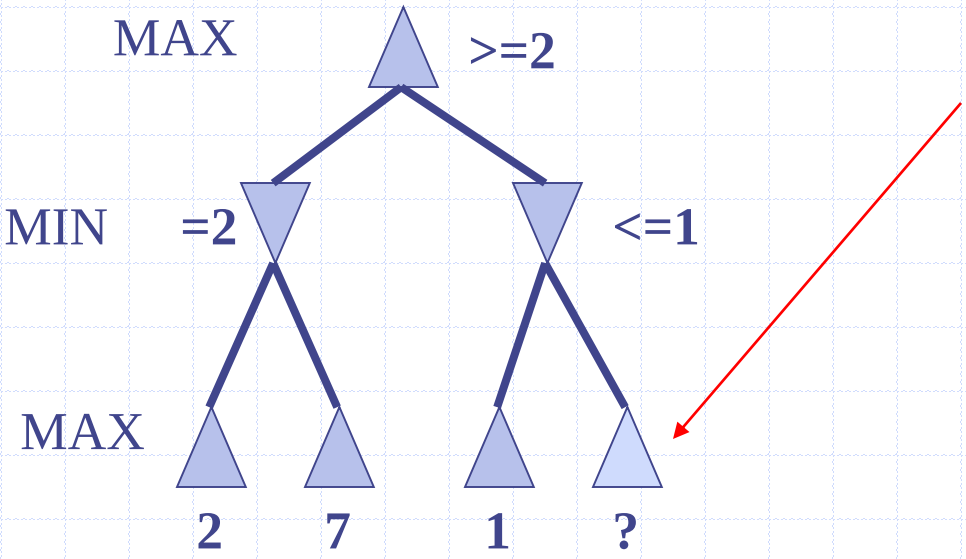
Criterion	Minimax
Complete?	yes, ☺
Optimal?	yes, ☺
Time	$O(b^m)$, ☹
Space	O(bm), ☺

Problem of minimax search

- Number of games states is exponential to the number of moves.
  - Solution: Do not examine every node
  - ==> Alpha-beta pruning
    - Remove branches that do not influence final decision

Alpha-beta pruning

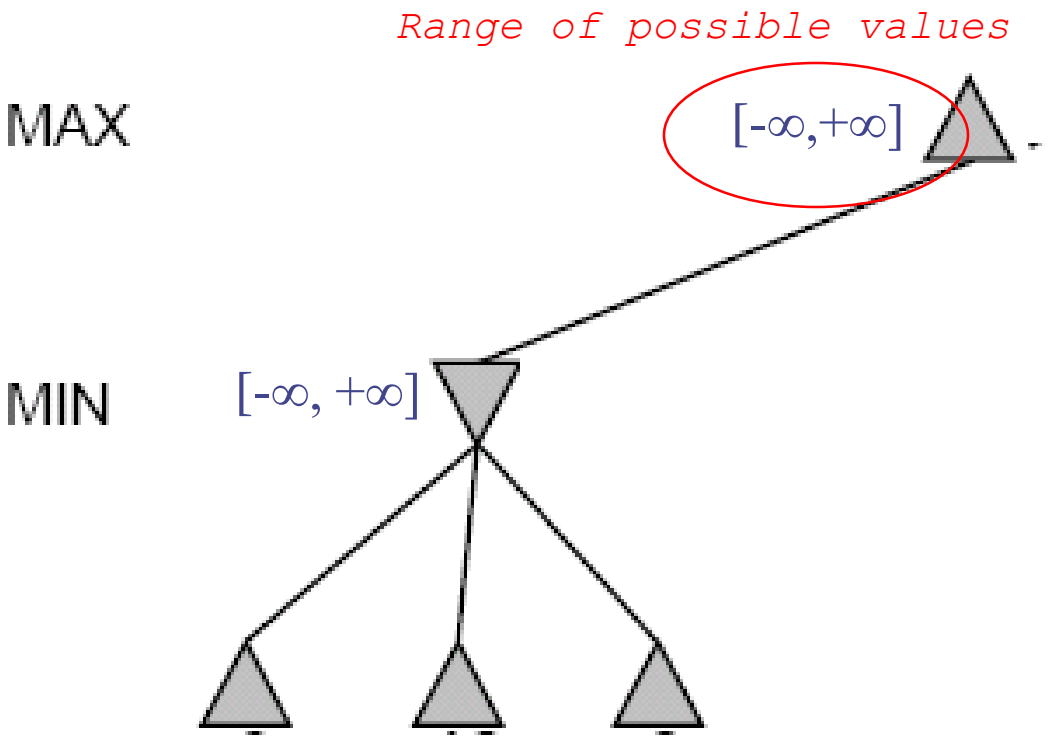
- We can improve on the performance of the minimax algorithm through **alpha-beta pruning**
- Basic idea: *"If you have an idea that is surely bad, don't take the time to see how truly awful it is."* -- Pat Winston



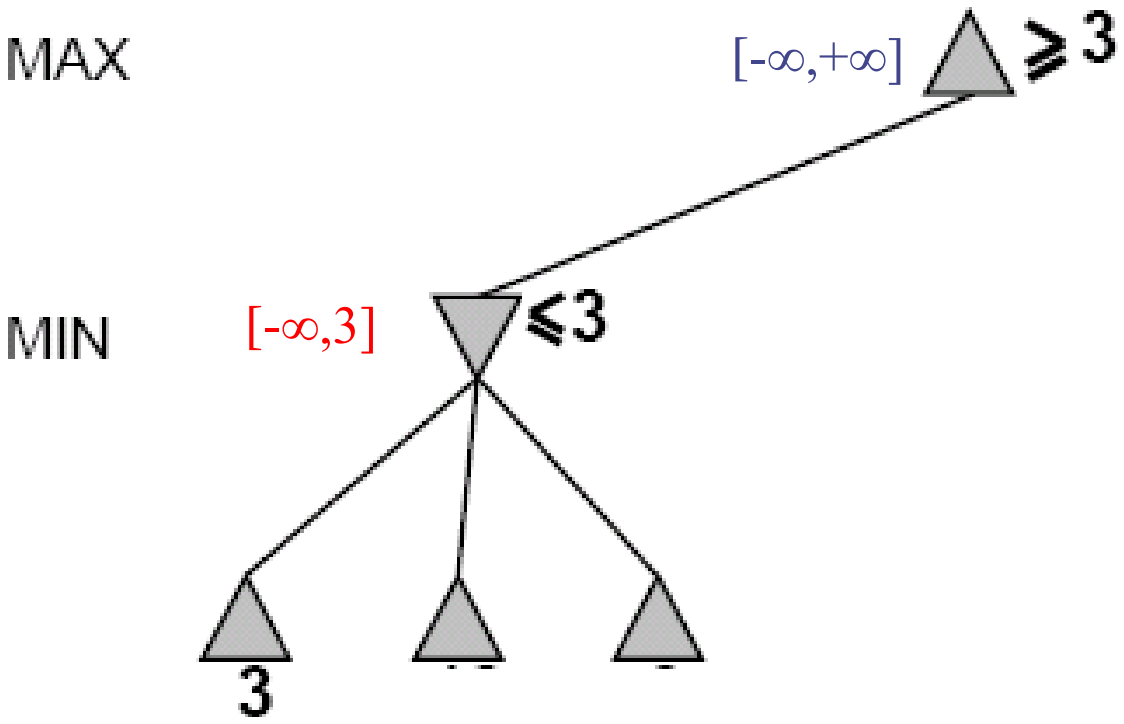
- We don't need to compute the value at this node.
- No matter what it is, it can't affect the value of the root node.

Alpha-Beta Example

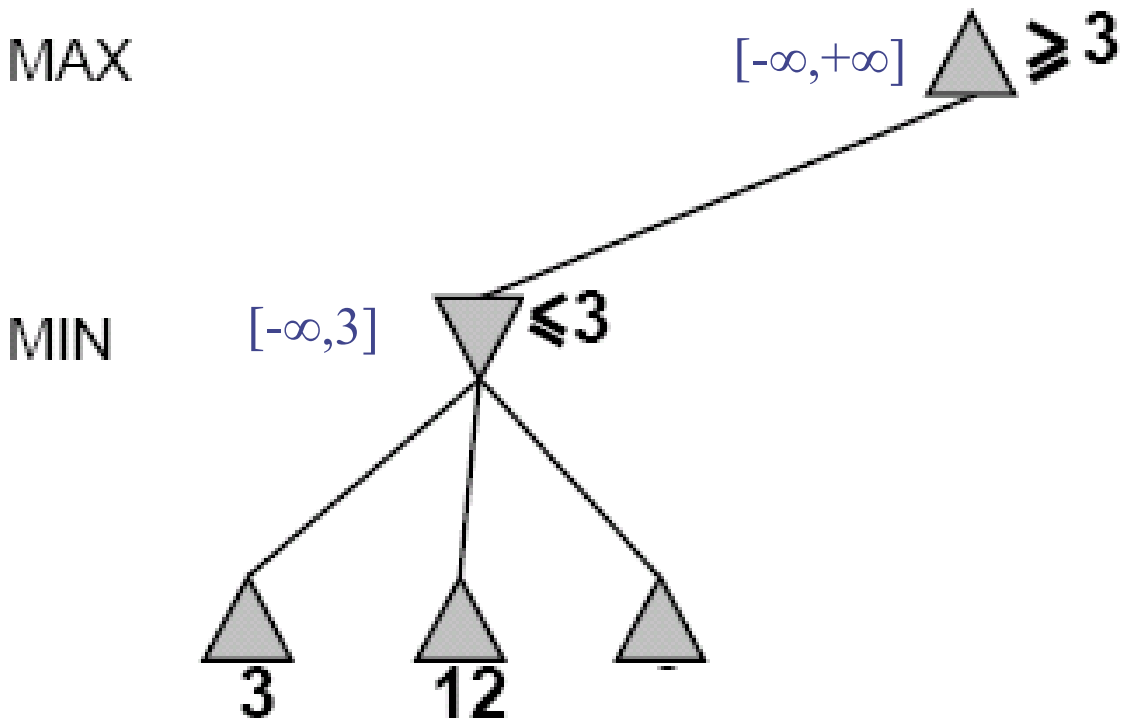
Do DF-search until first leaf



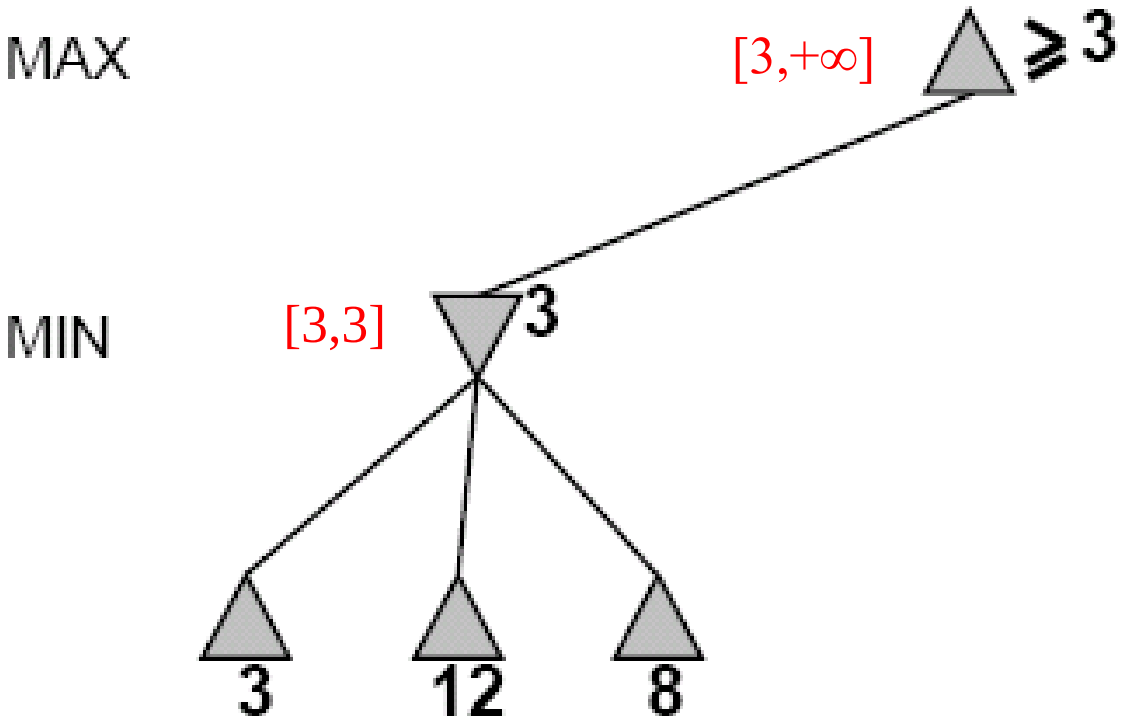
Alpha-Beta Example (continued)



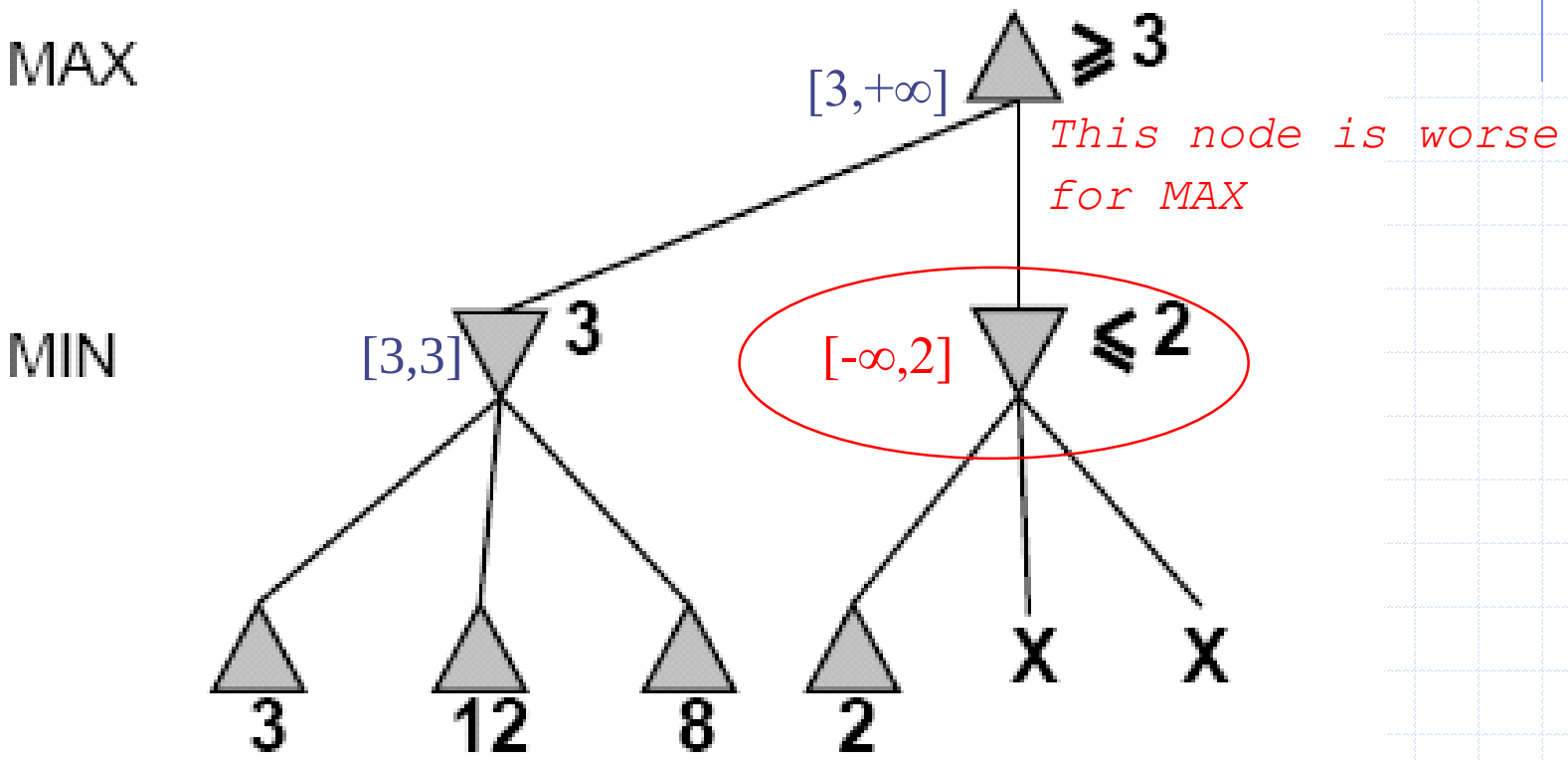
Alpha-Beta Example (continued)



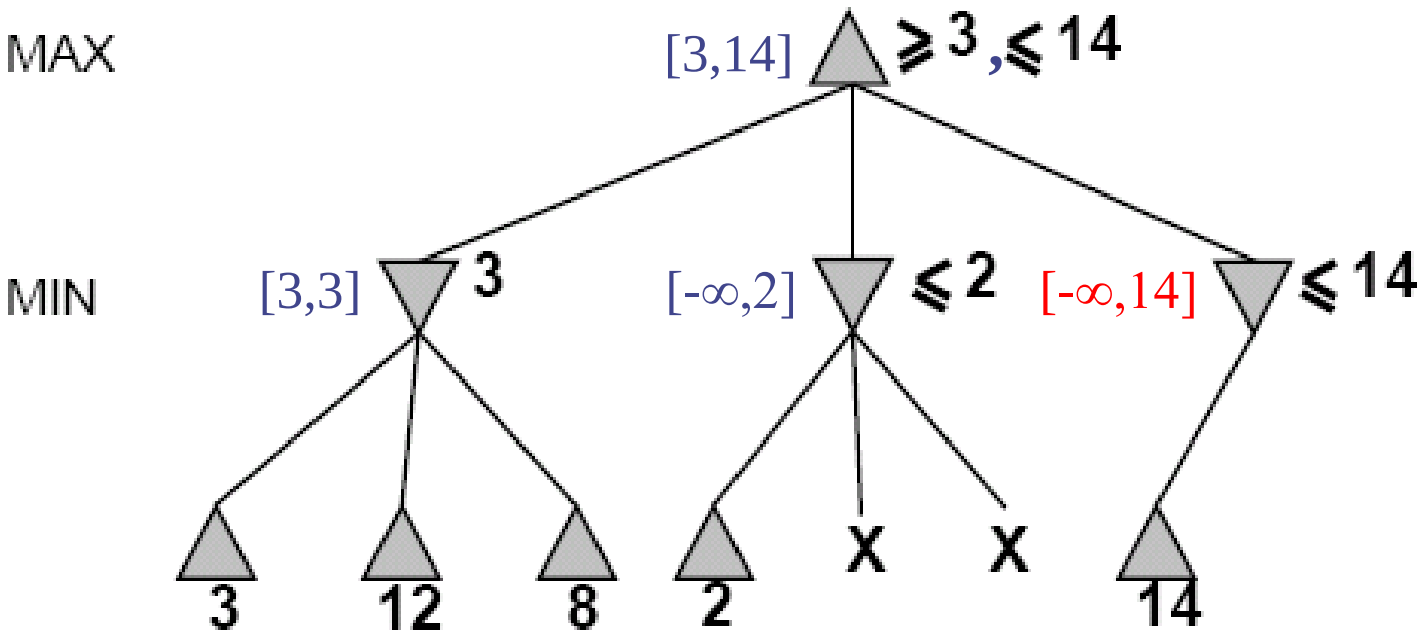
Alpha-Beta Example (continued)



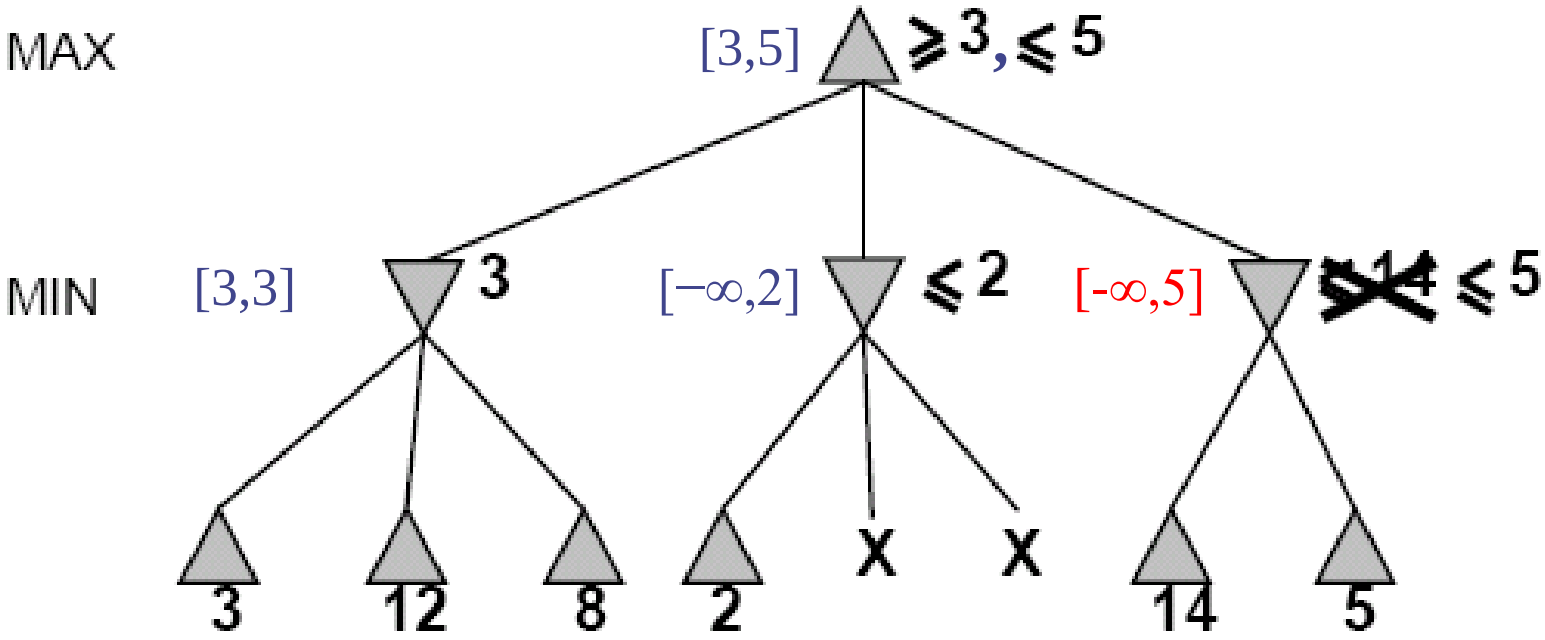
Alpha-Beta Example (continued)



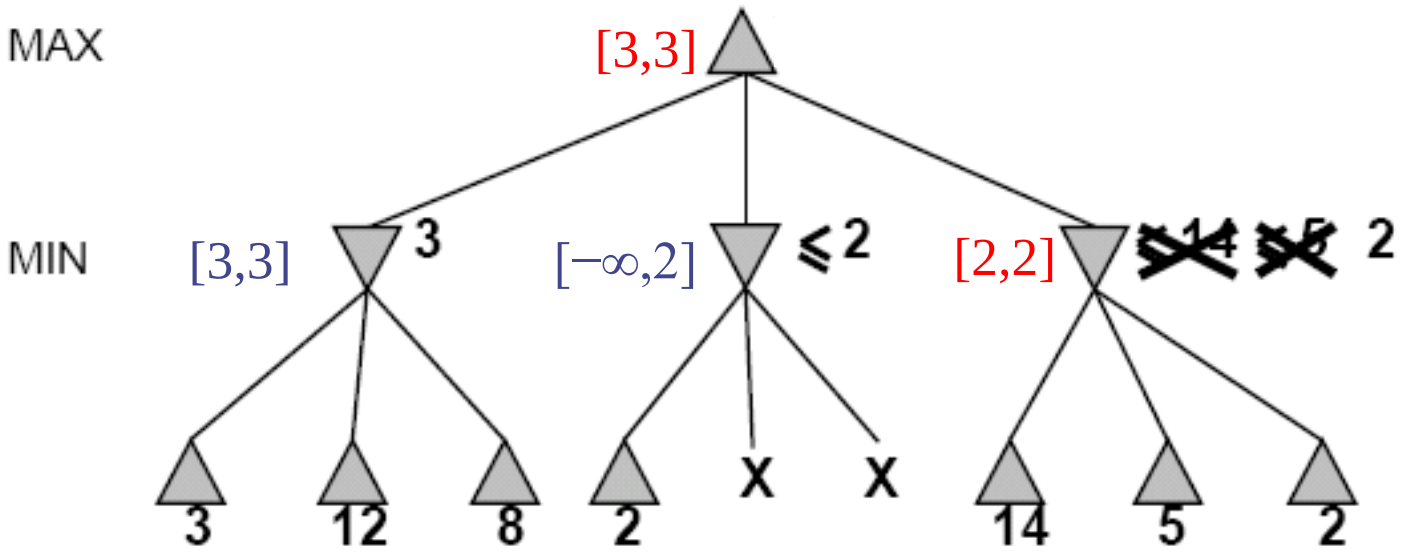
Alpha-Beta Example (continued)



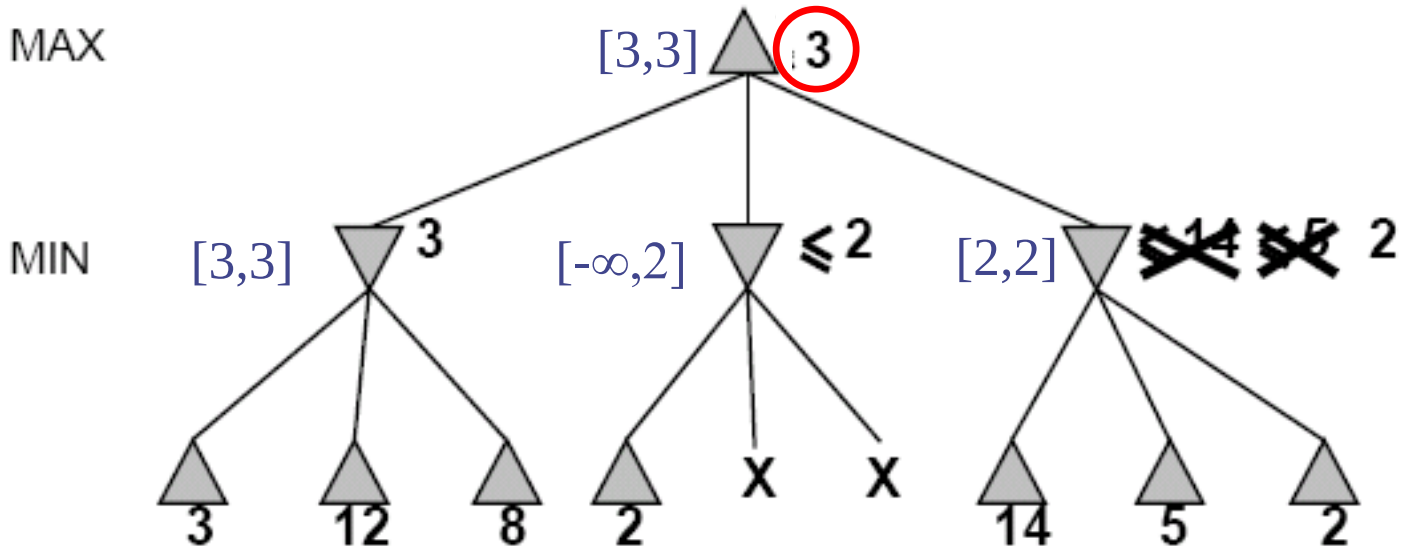
Alpha-Beta Example (continued)



Alpha-Beta Example (continued)

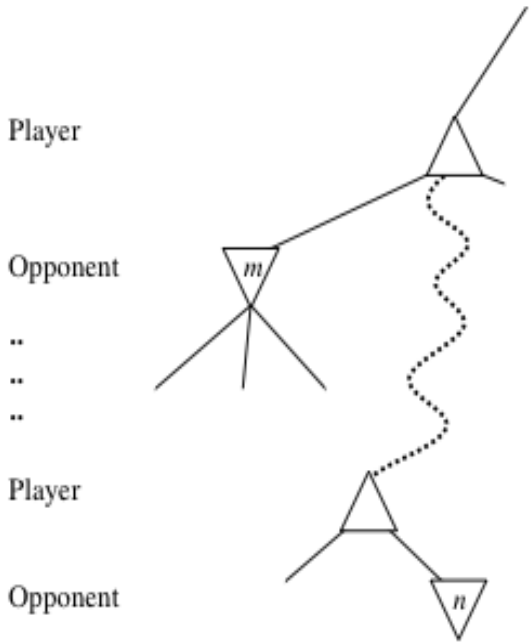


Alpha-Beta Example (continued)



General alpha-beta pruning

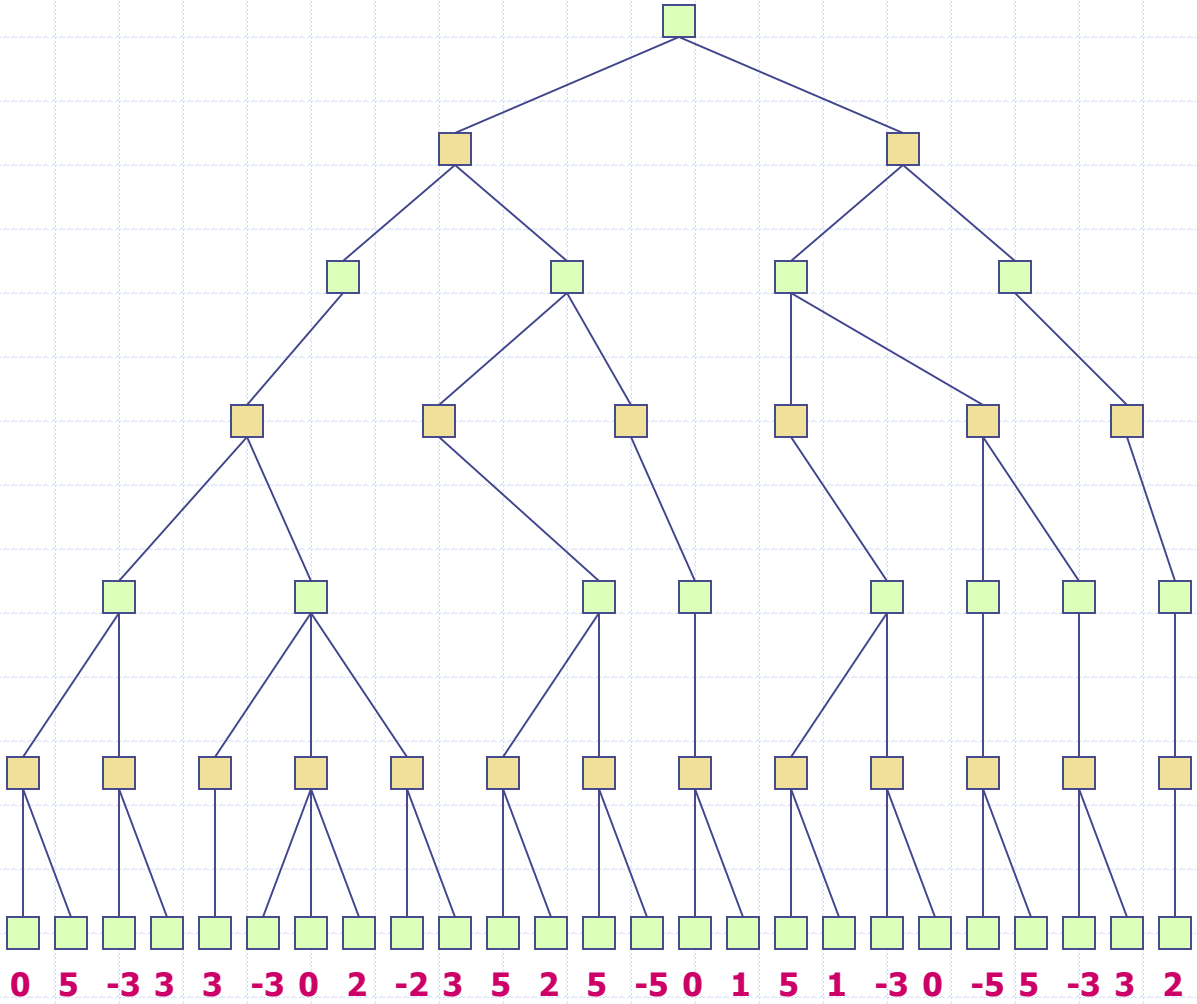
- Consider a node *n* somewhere in the tree
- If player has a better choice at
  - Parent node of n
  - Or any choice point further up
- *n* will **never** be reached in actual play.
- Hence when enough is known about *n*, it can be pruned.



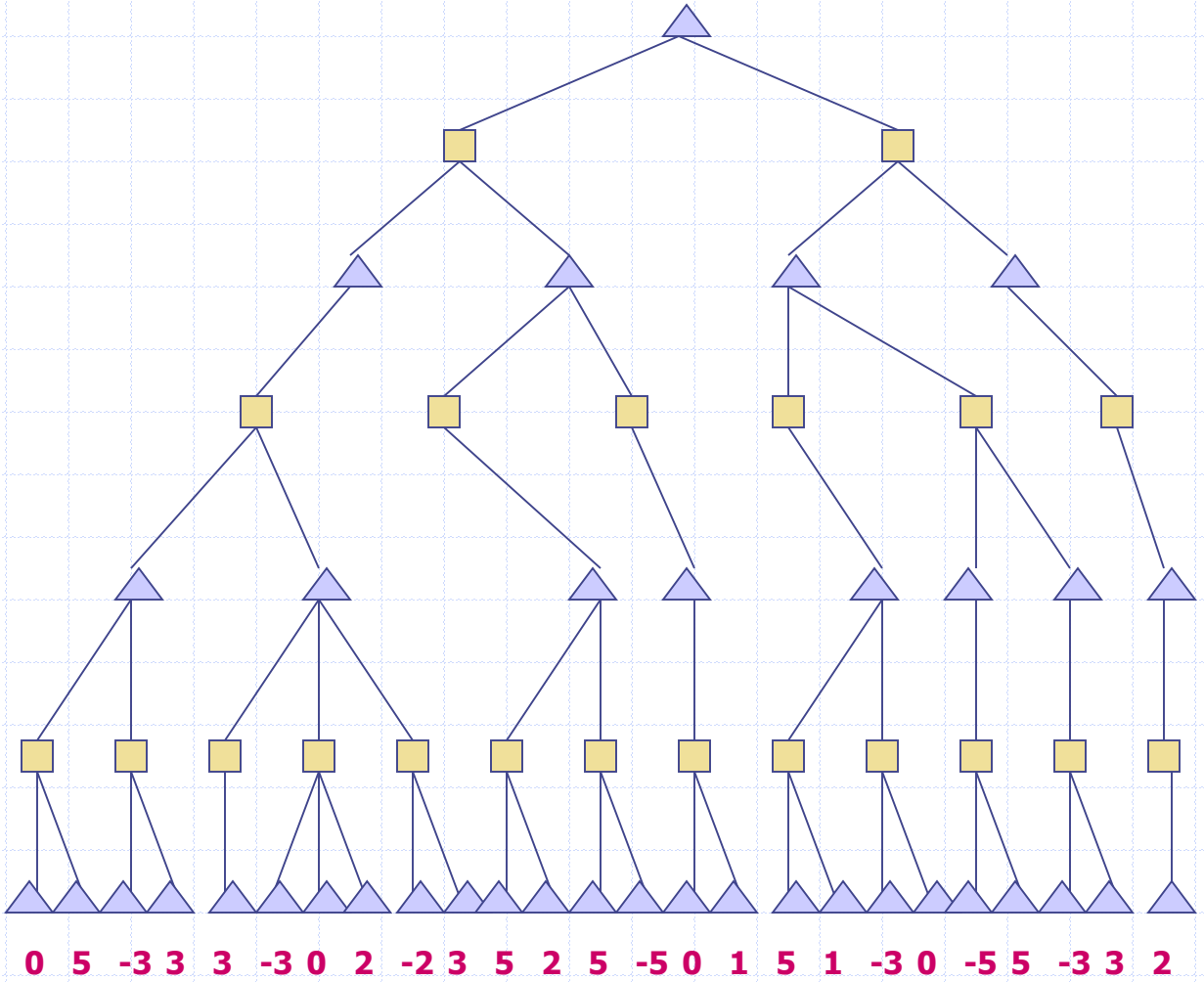
Comments: Alpha-Beta Pruning

- Pruning does not affect final results
- Entire subtrees can be pruned.
- Good move *ordering* improves effectiveness of pruning
- With "perfect ordering," time complexity is $O(b^{m/2})$
  - Branching factor of sqrt(b) !!
  - Alpha-beta pruning can look twice as far as minimax in the same amount of time
- Repeated states are again possible.
  - Store them in memory = transposition table

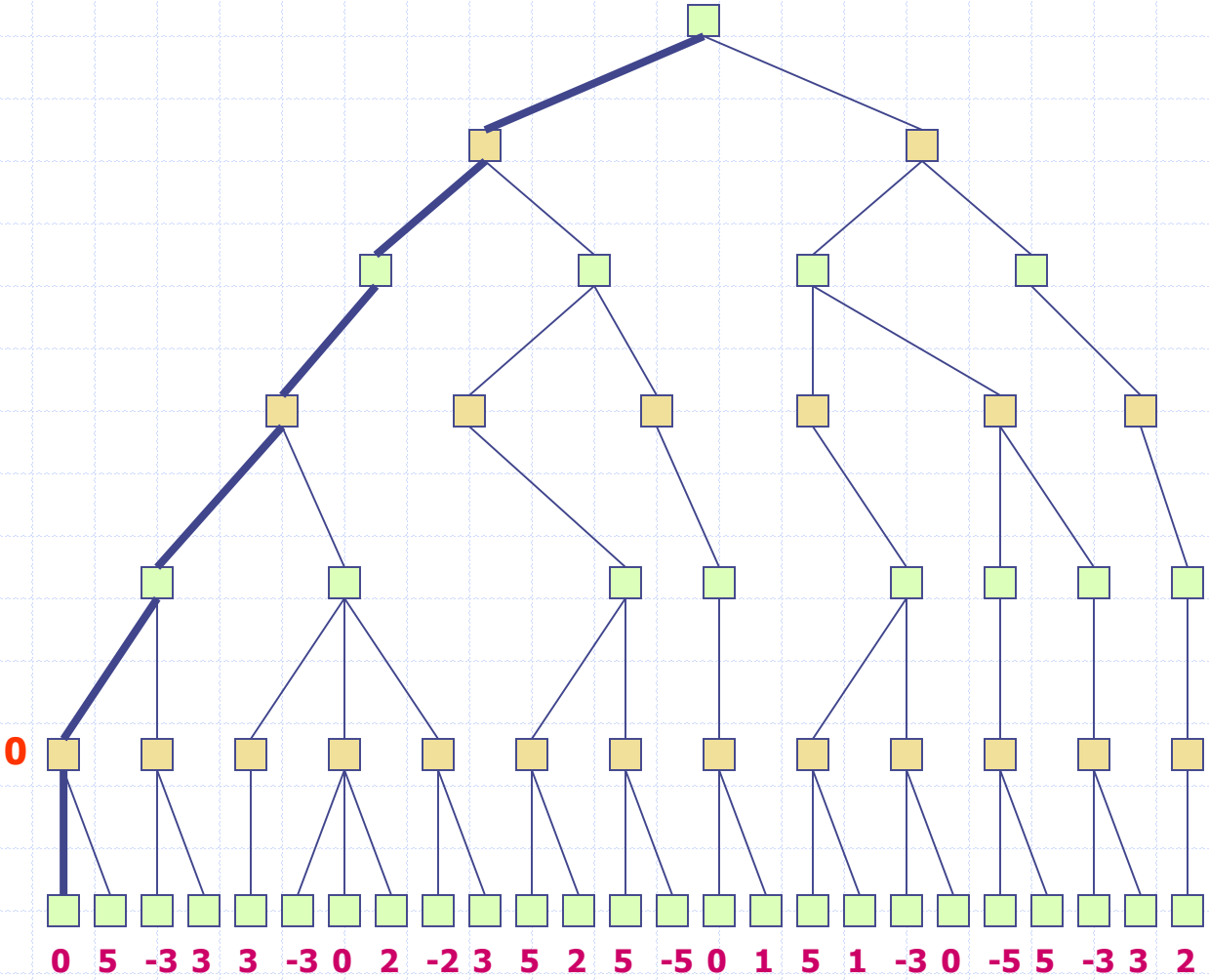
Alpha-Beta Example



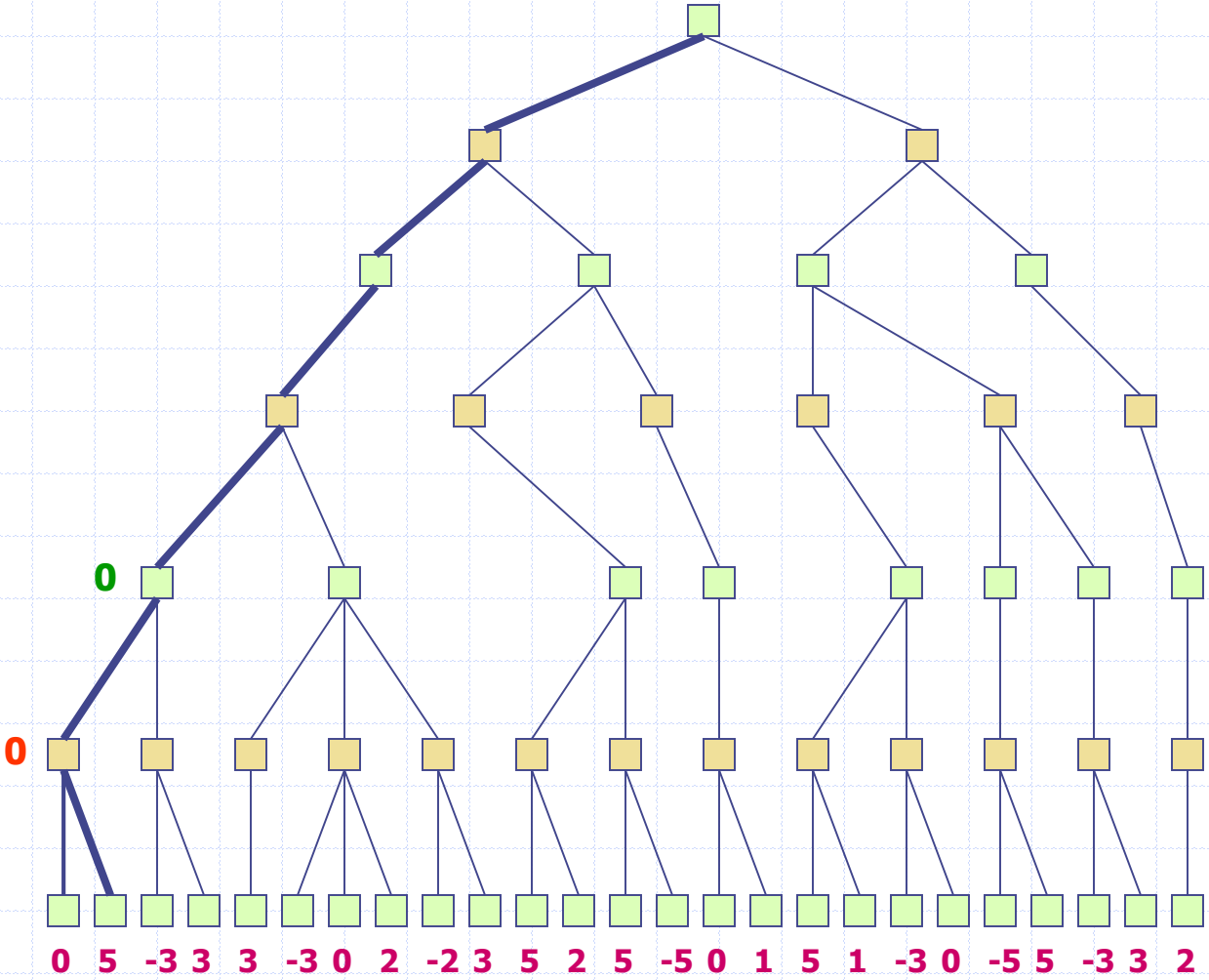
Alpha-Beta Example



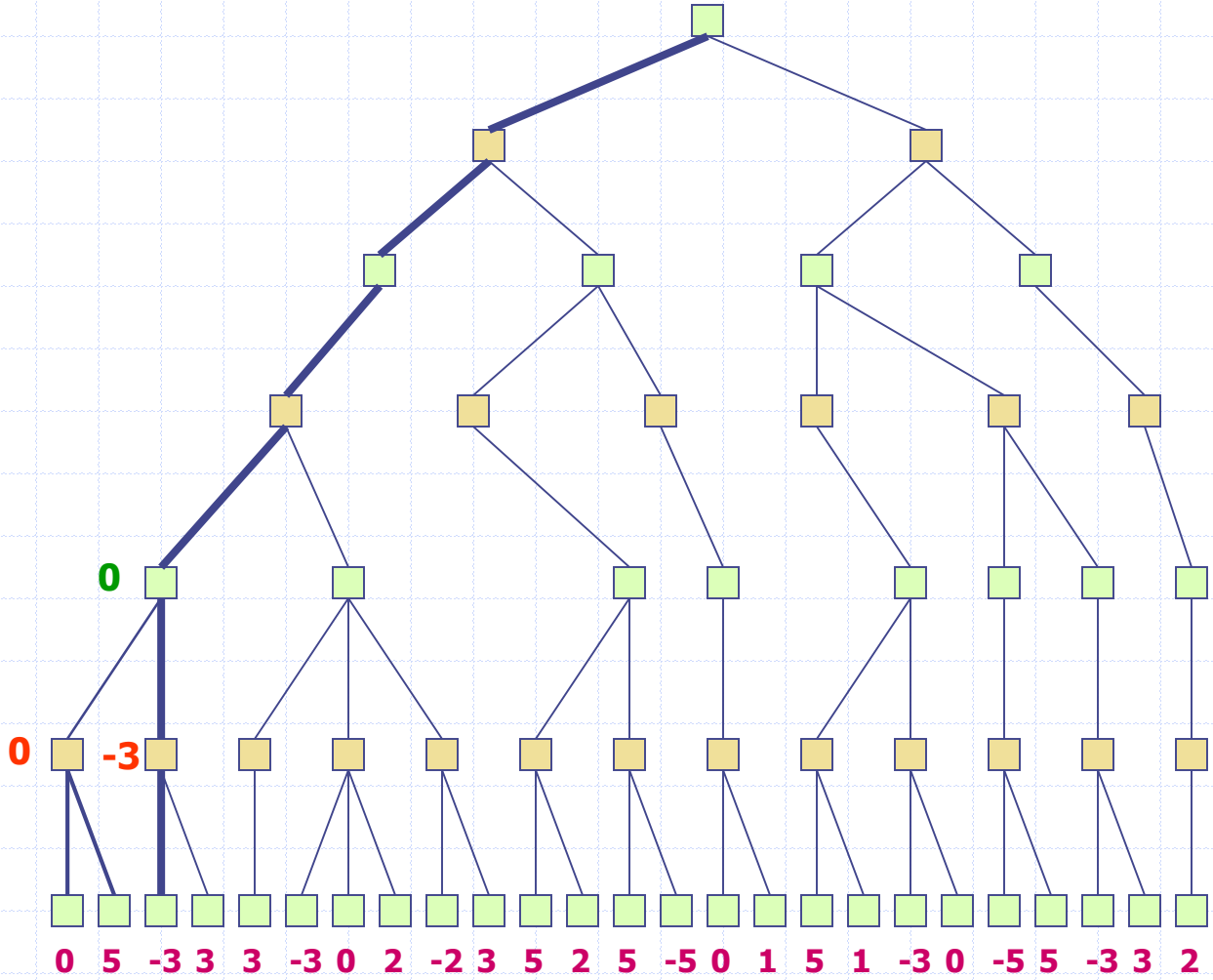
Alpha-Beta Example



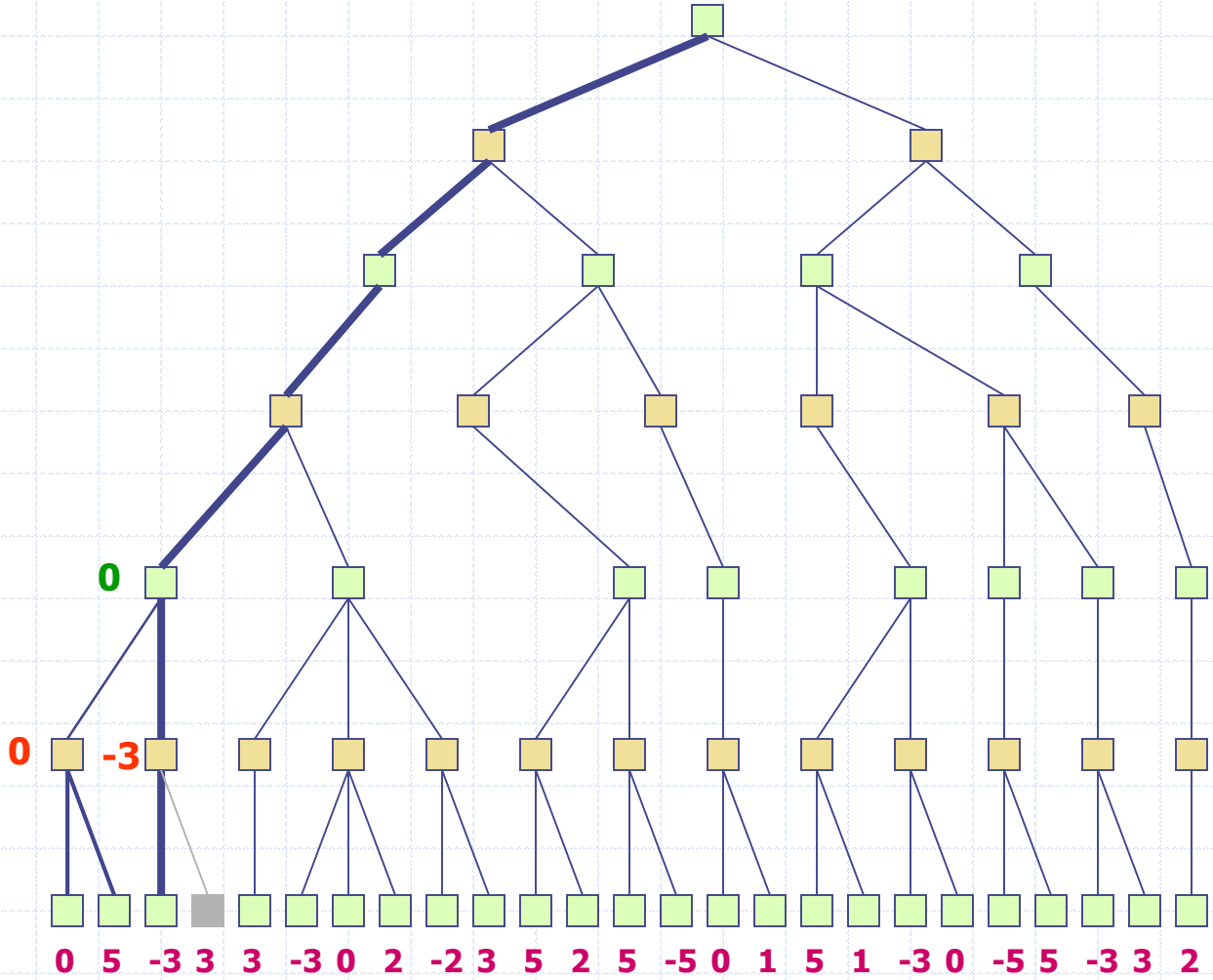
Alpha-Beta Example



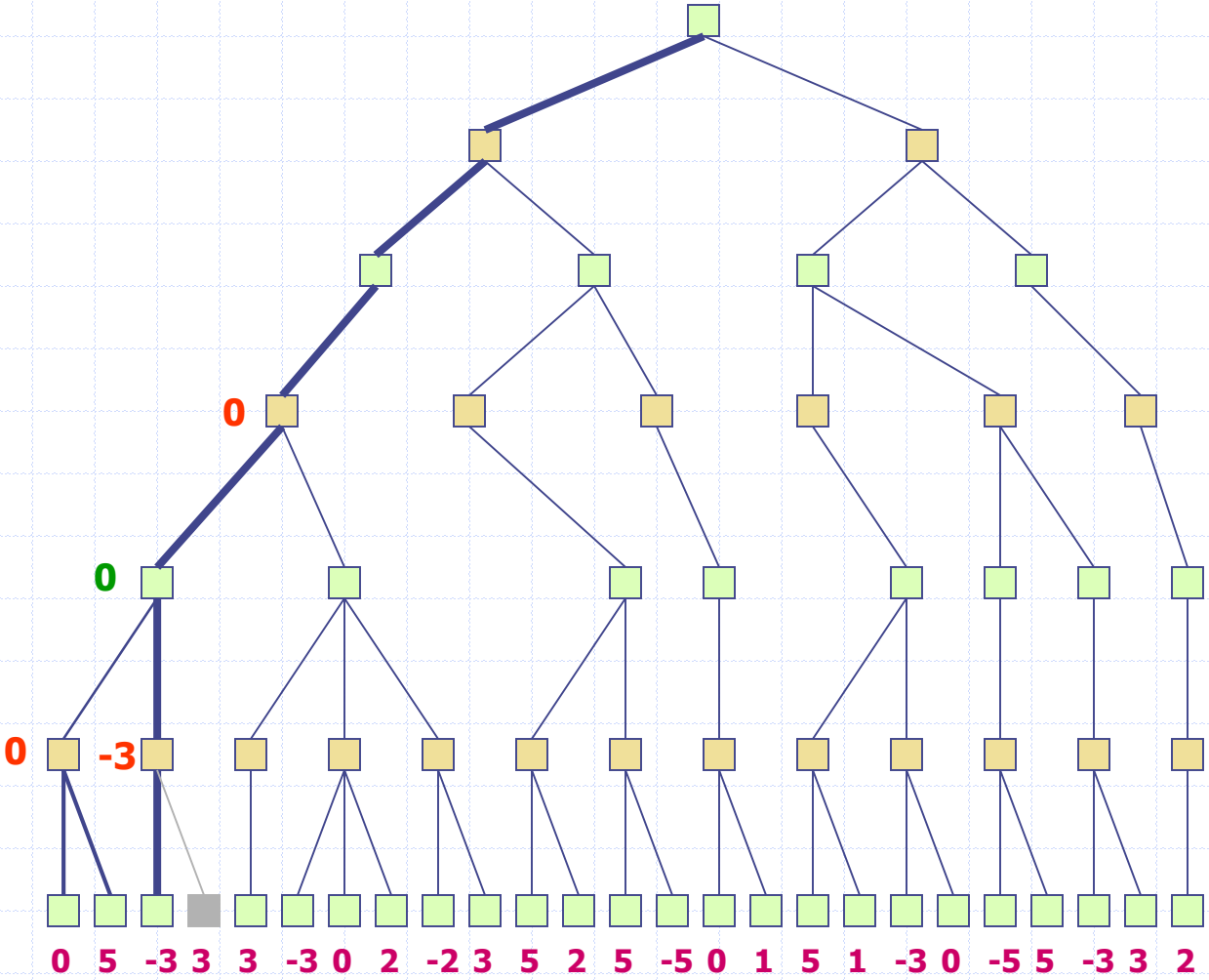
Alpha-Beta Example



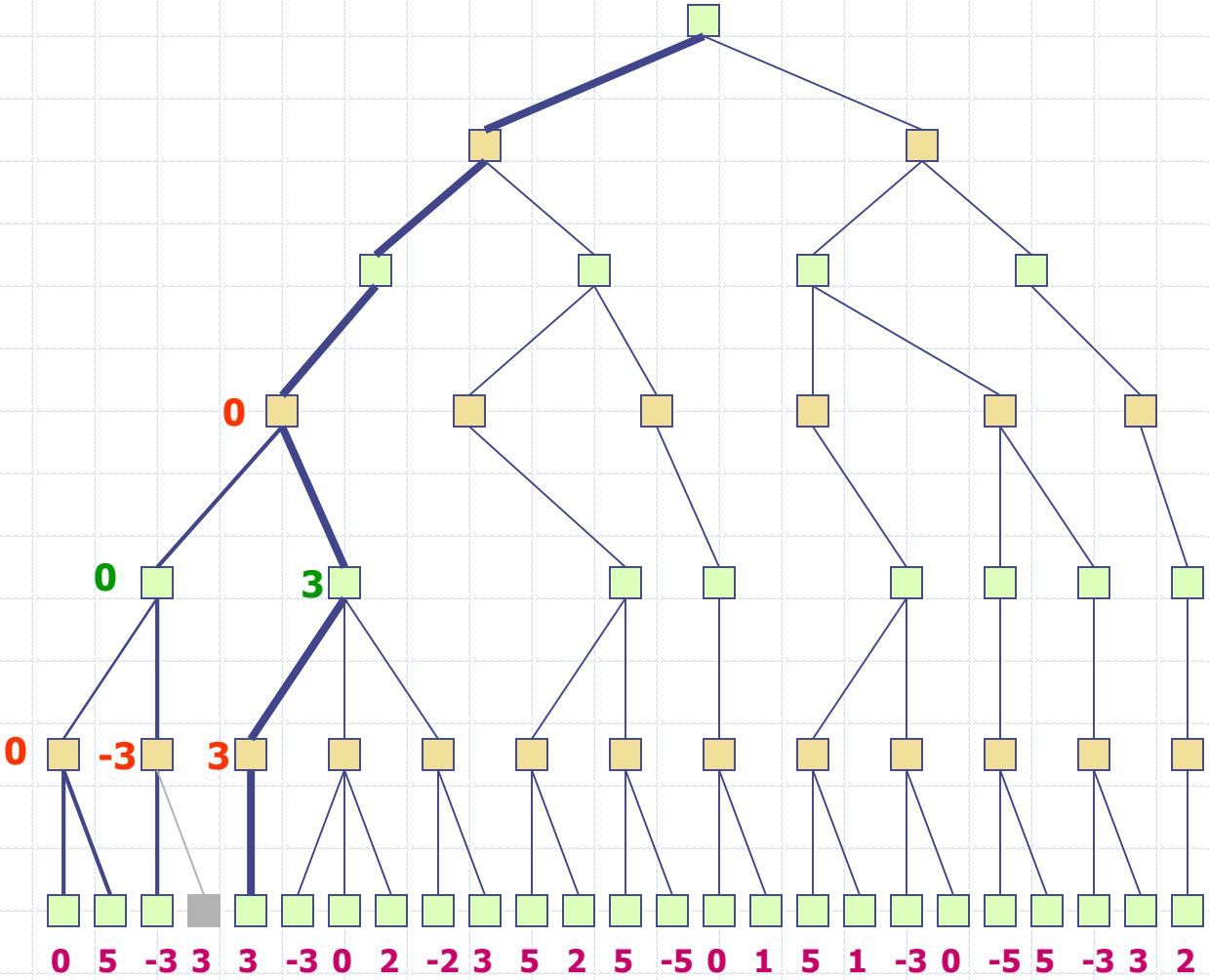
Alpha-Beta Example



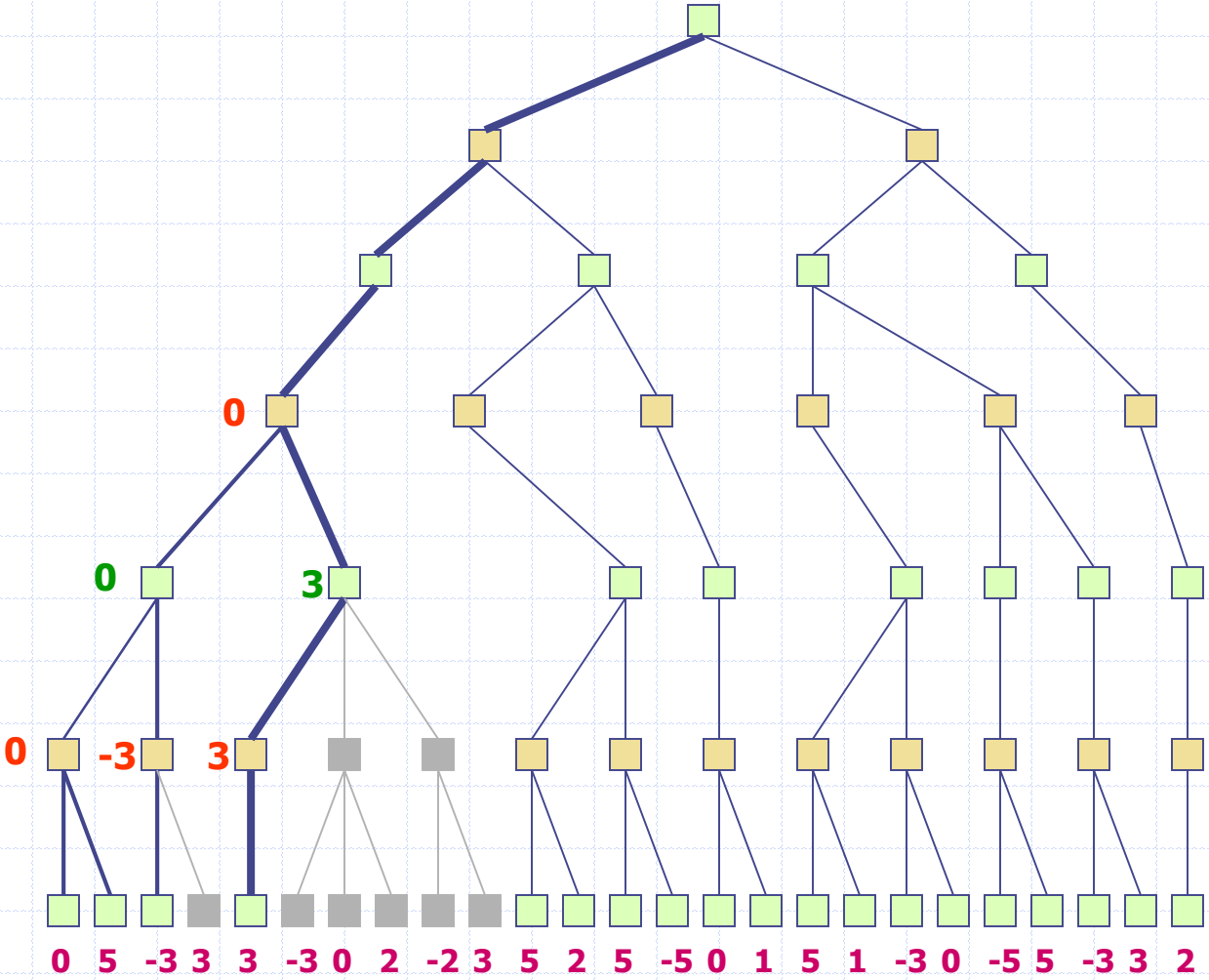
Alpha-Beta Example



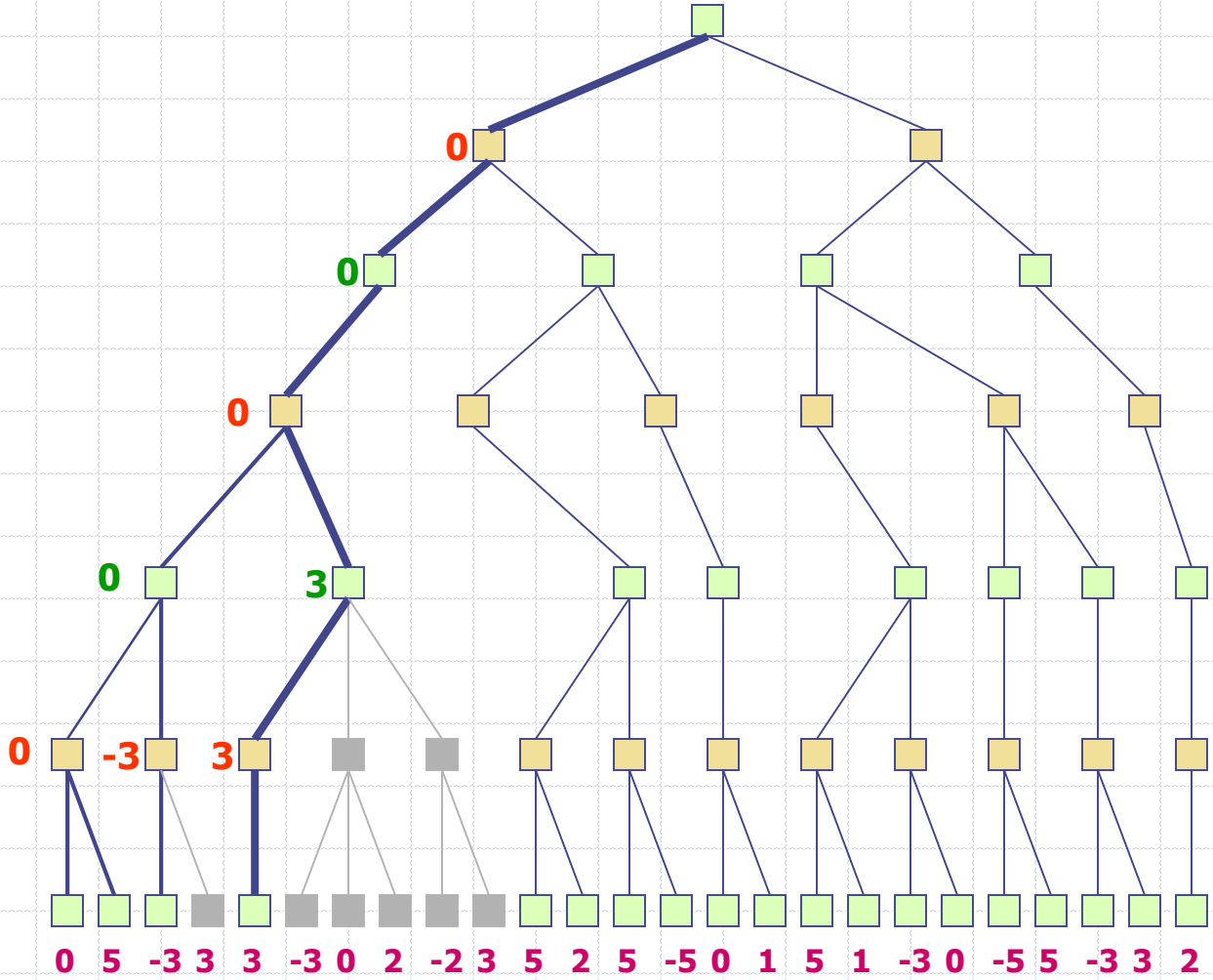
Alpha-Beta Example



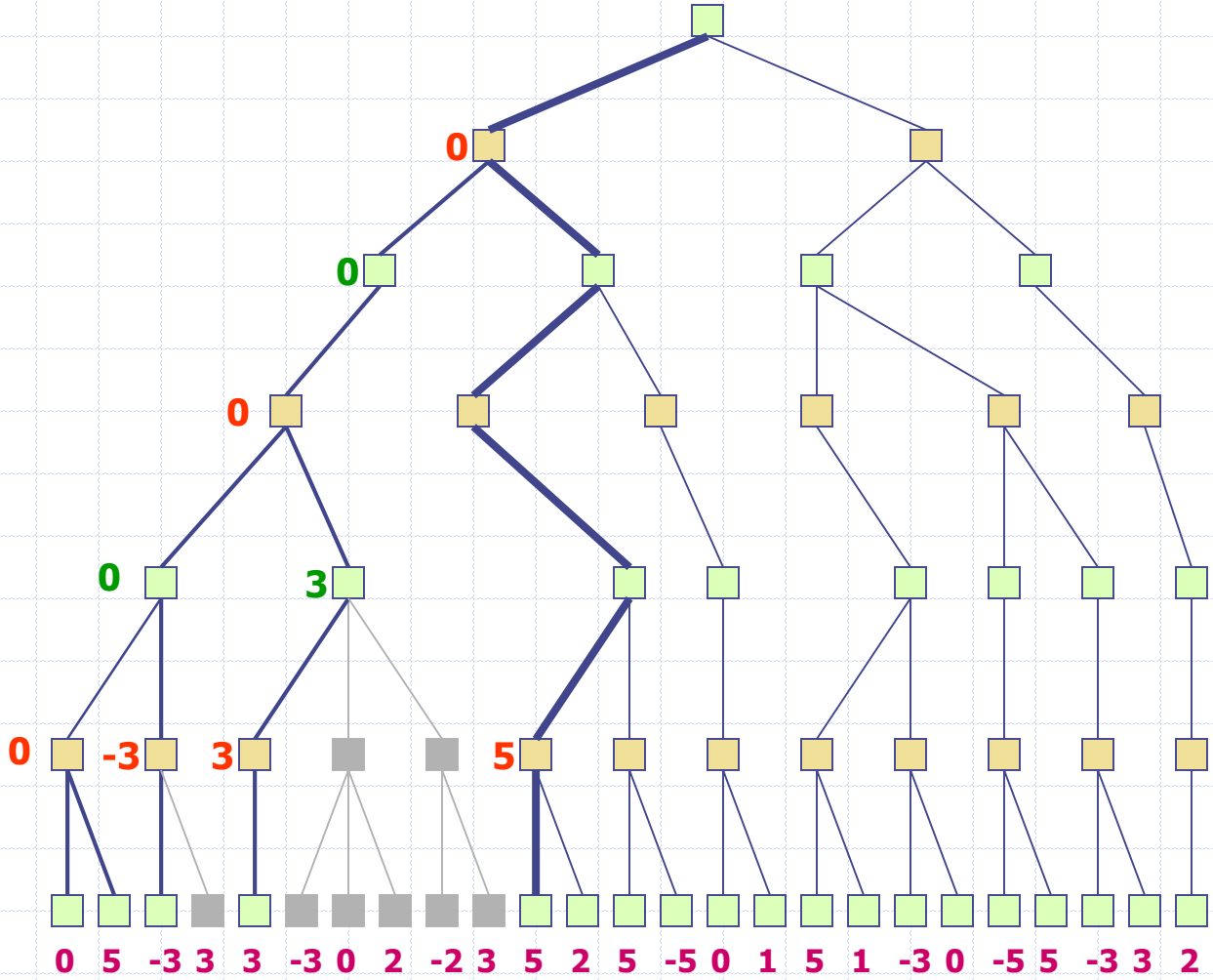
Alpha-Beta Example



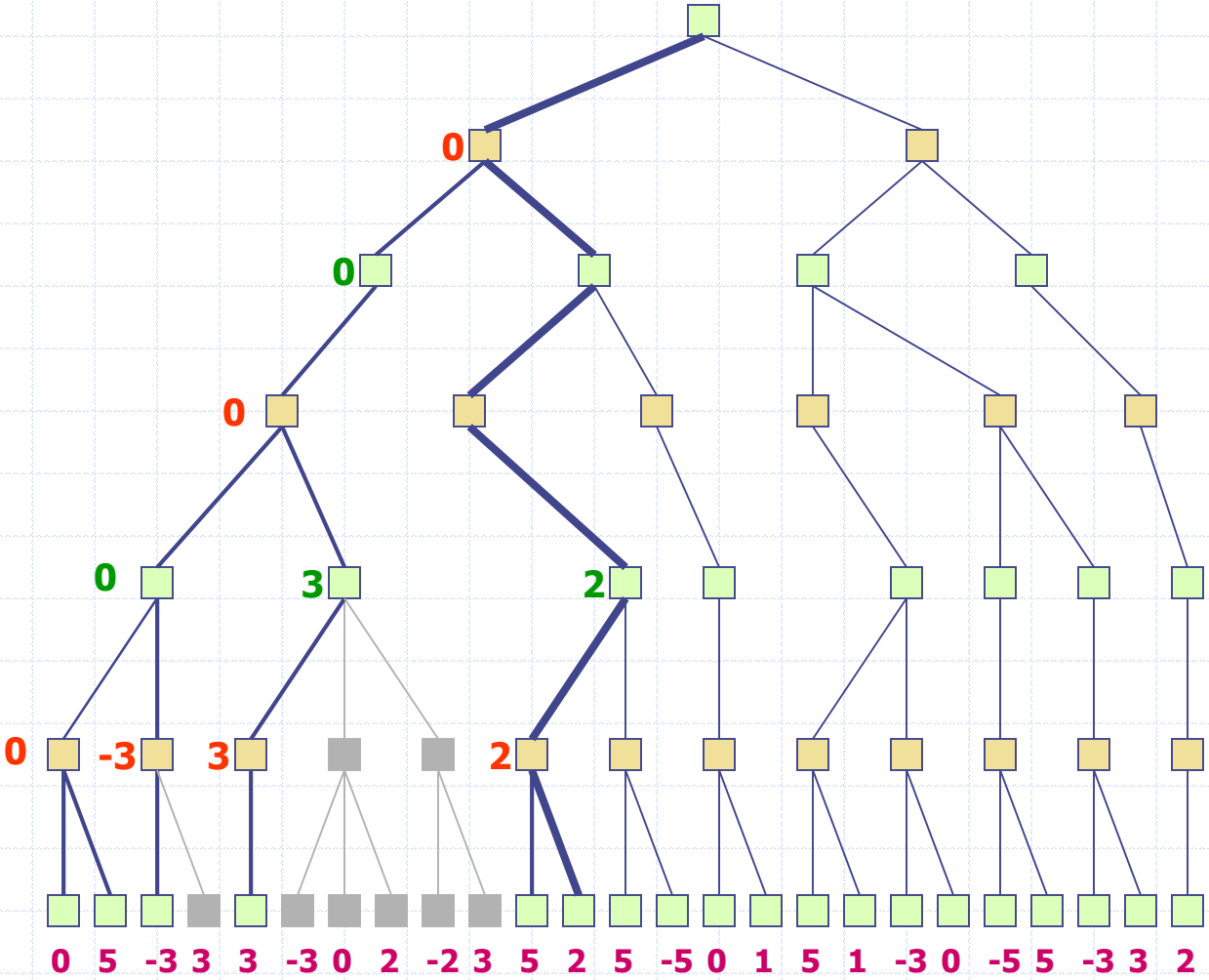
Alpha-Beta Example



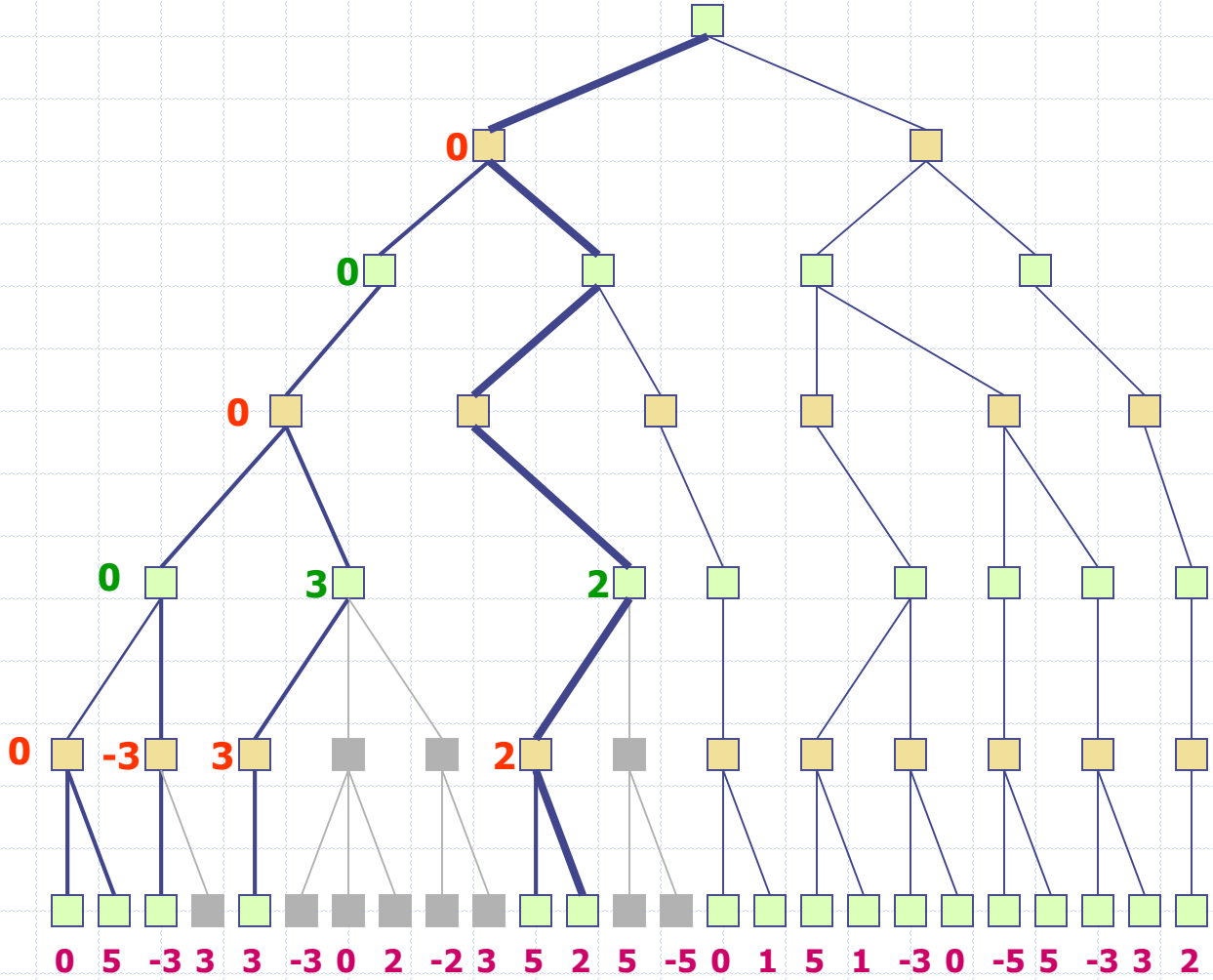
Alpha-Beta Example



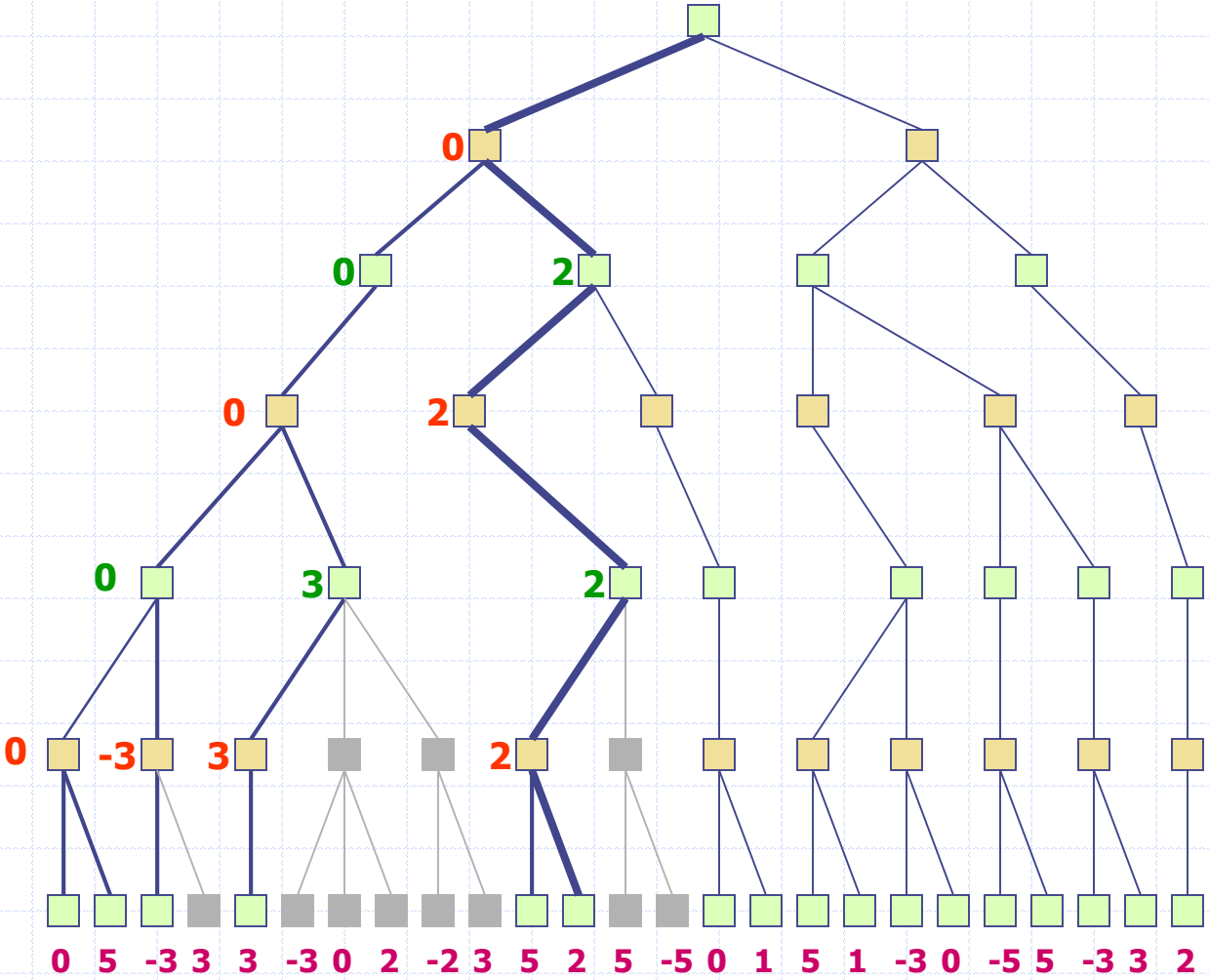
Alpha-Beta Example



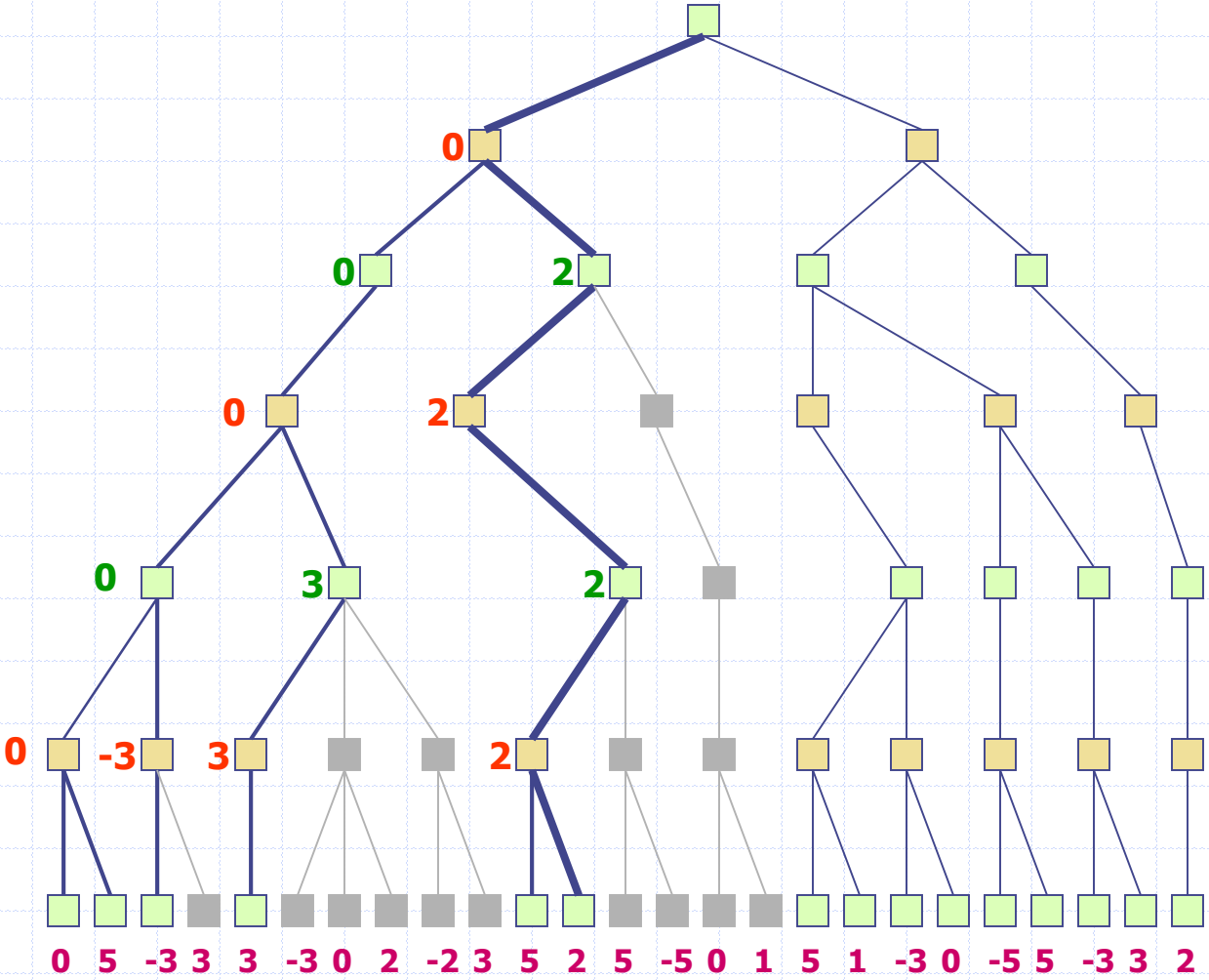
Alpha-Beta Example



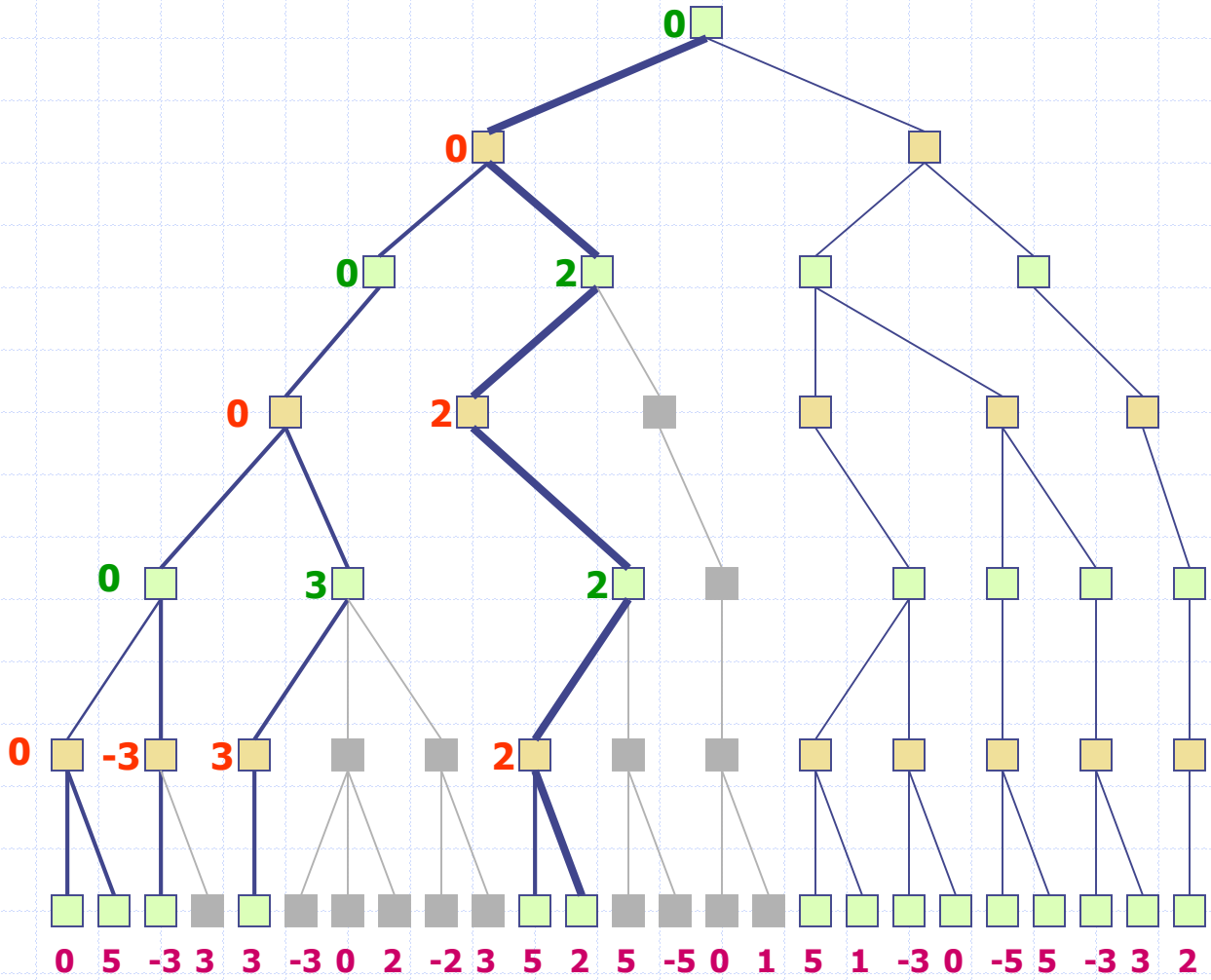
Alpha-Beta Example



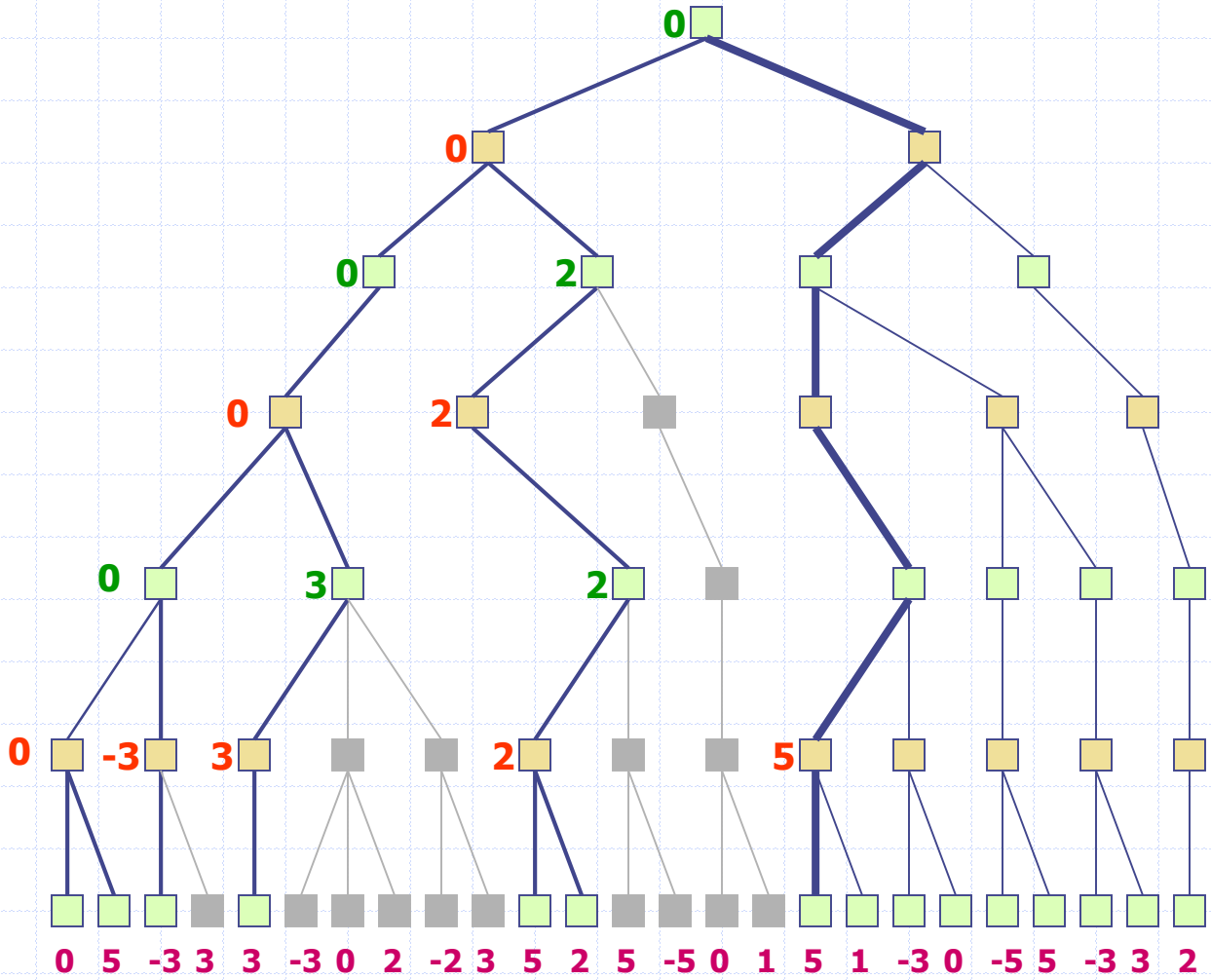
Alpha-Beta Example



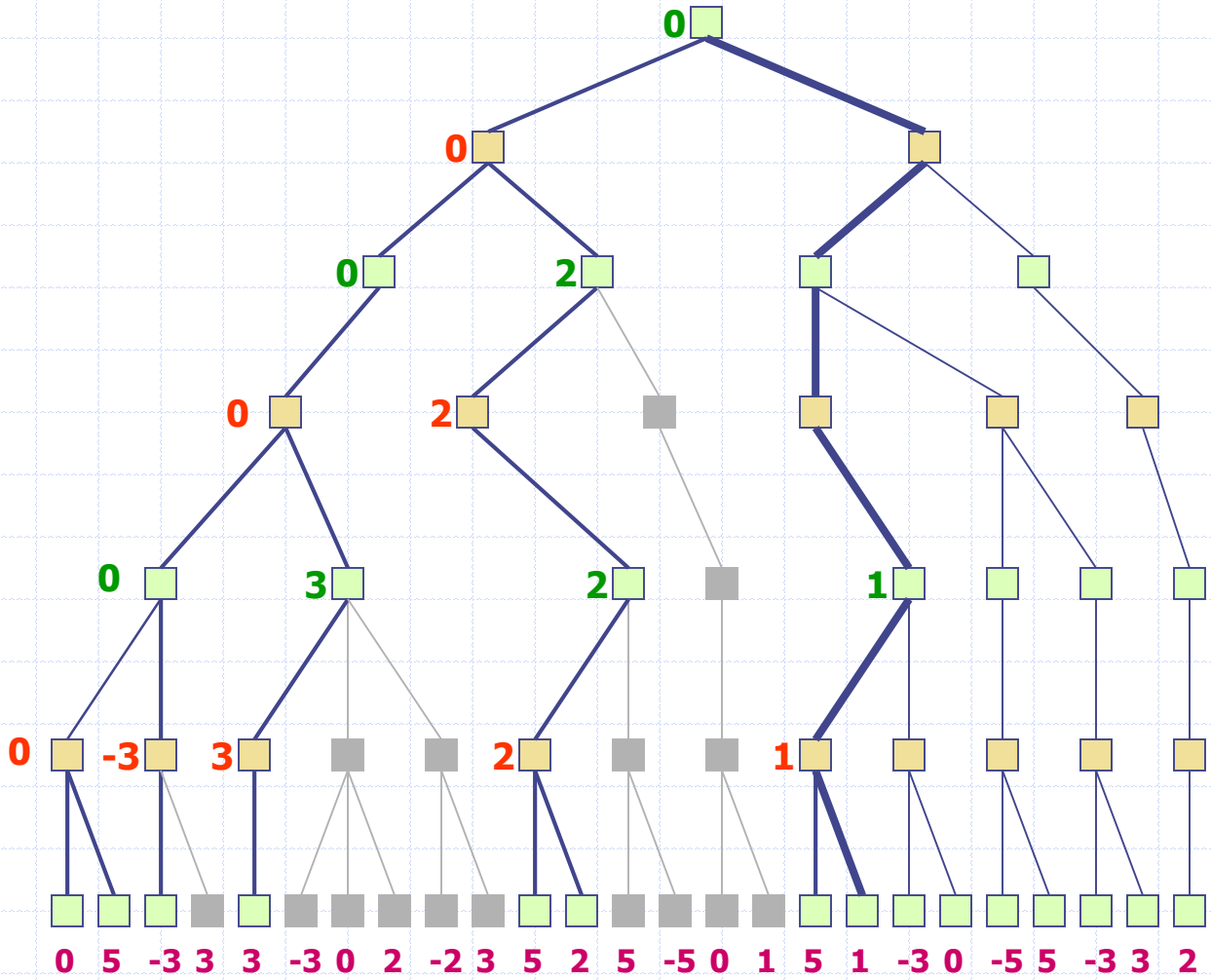
Alpha-Beta Example



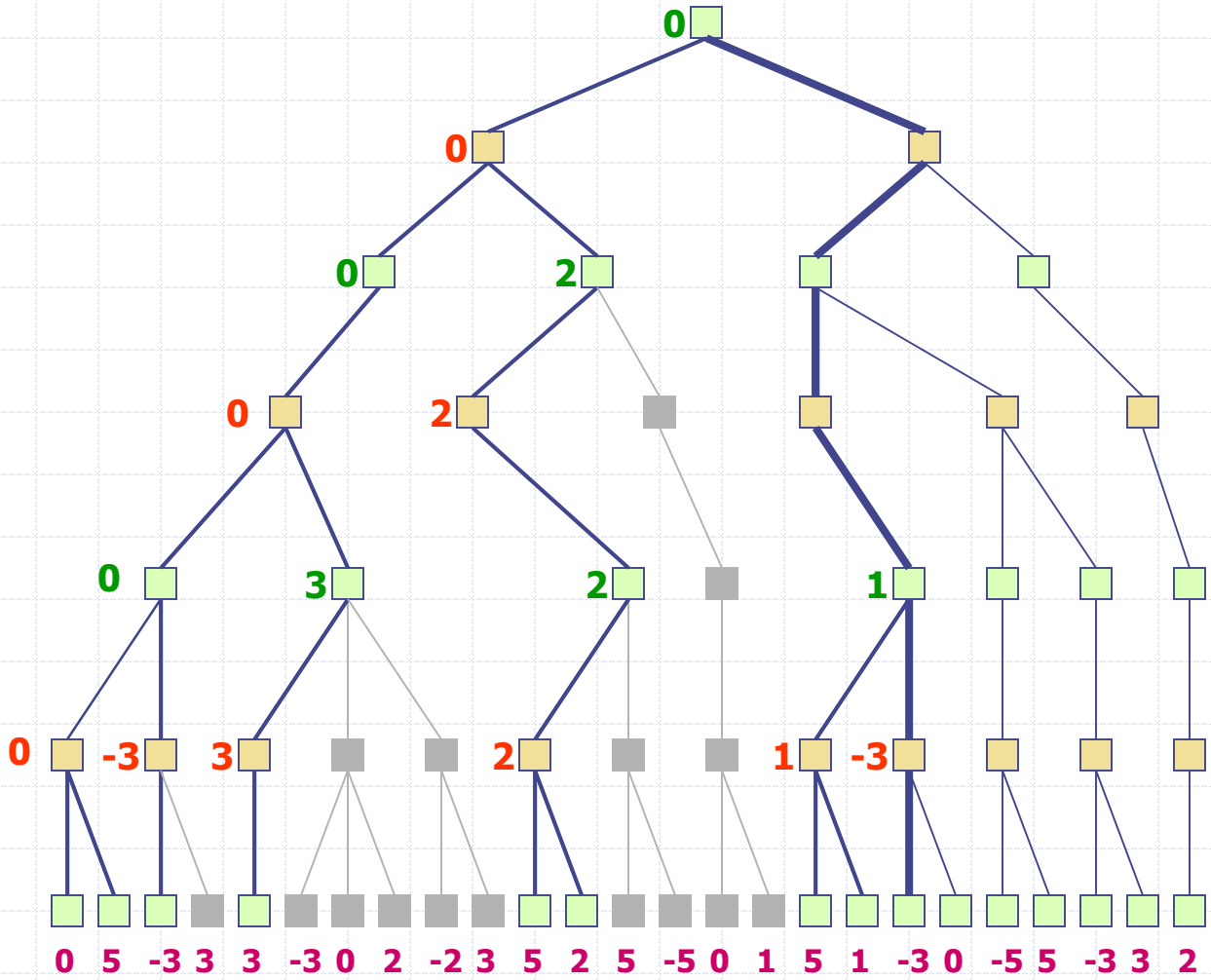
Alpha-Beta Example



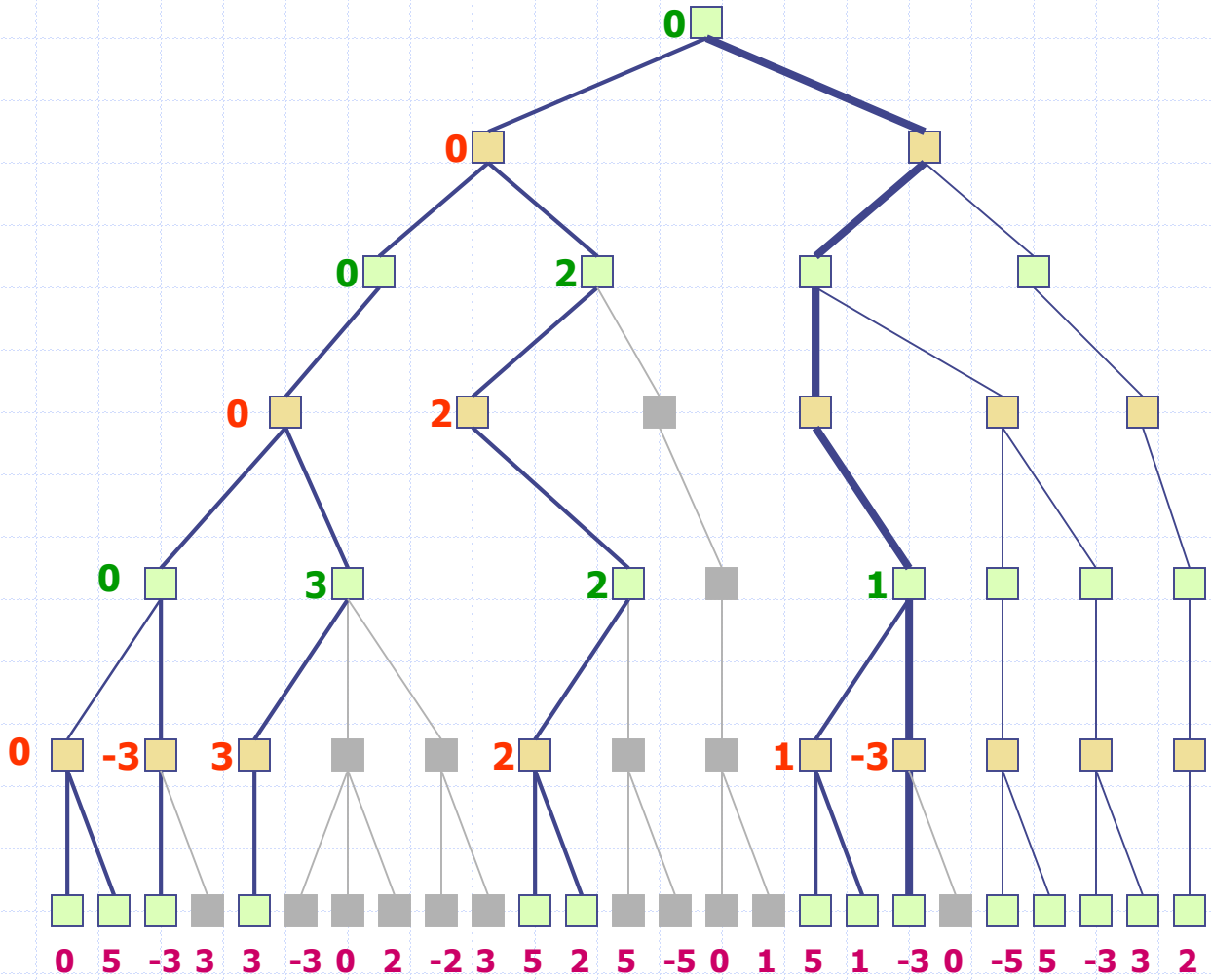
Alpha-Beta Example



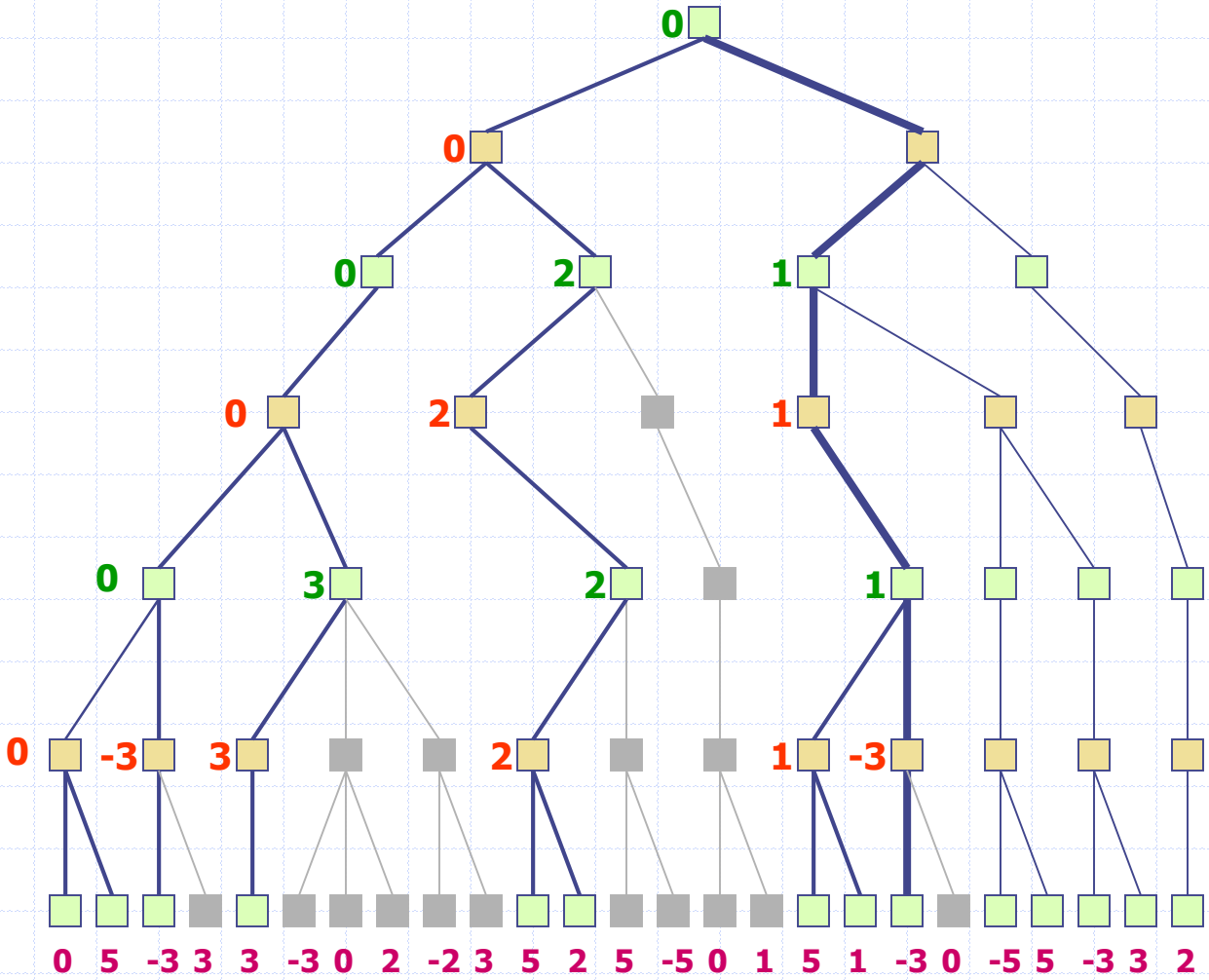
Alpha-Beta Example



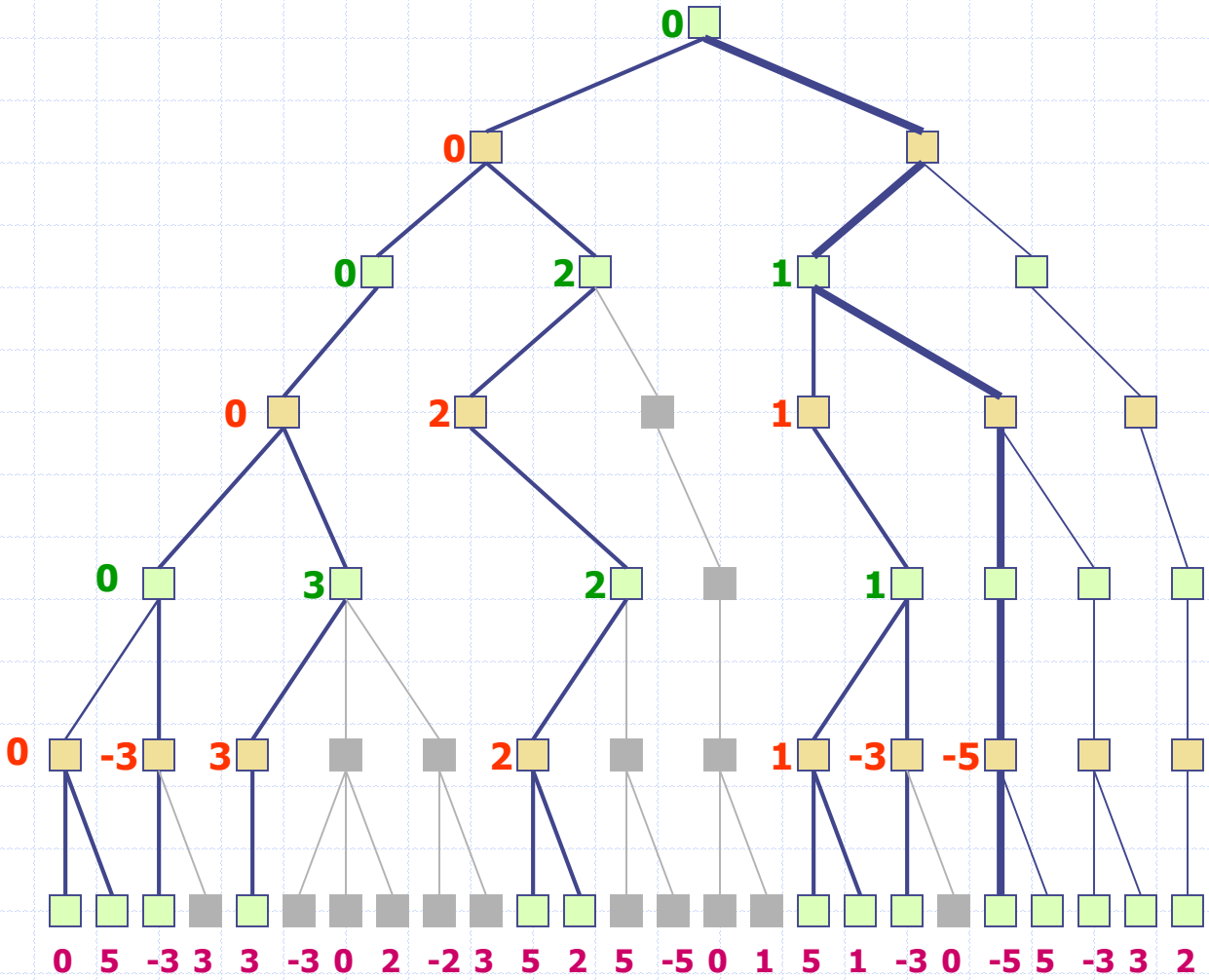
Alpha-Beta Example



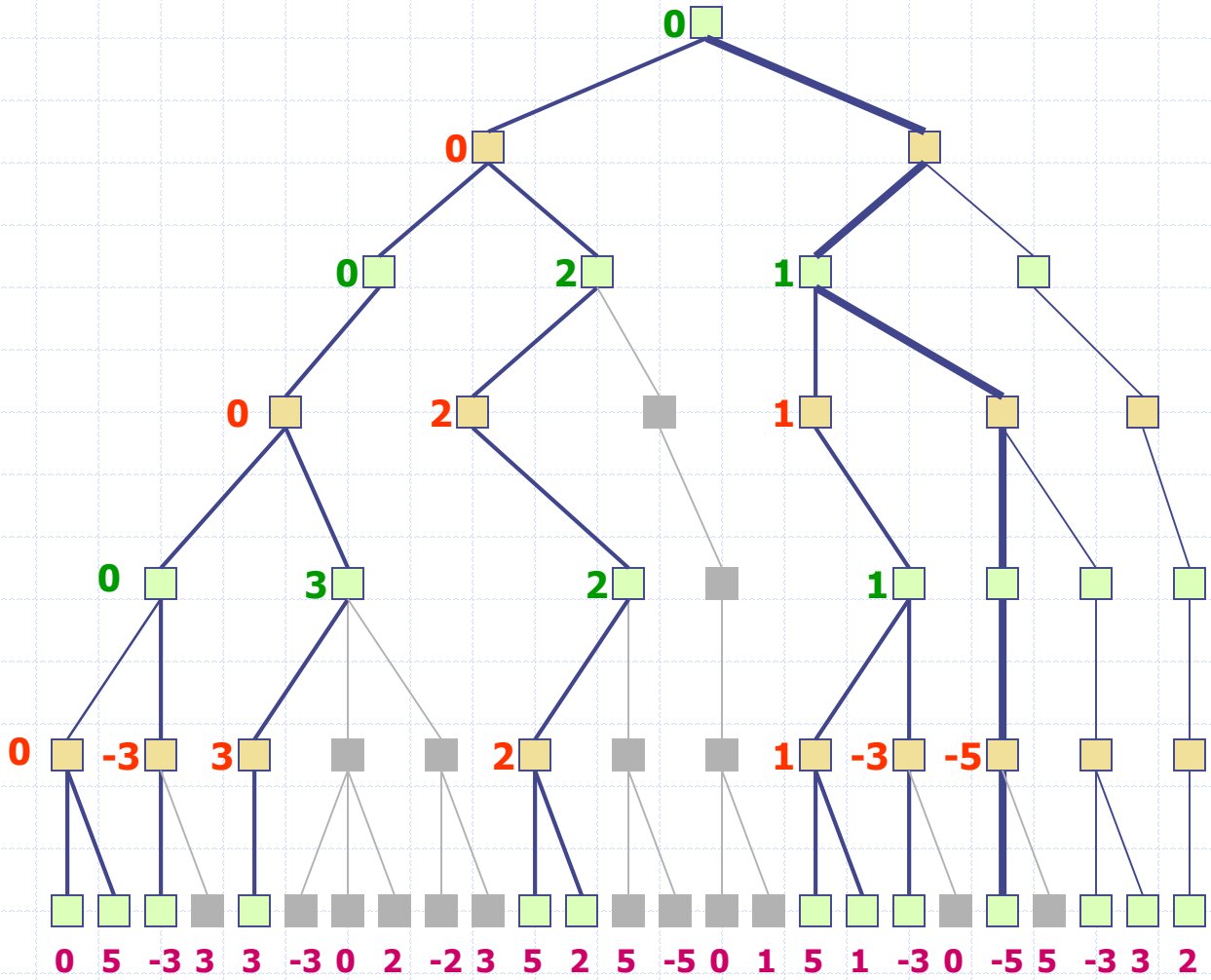
Alpha-Beta Example



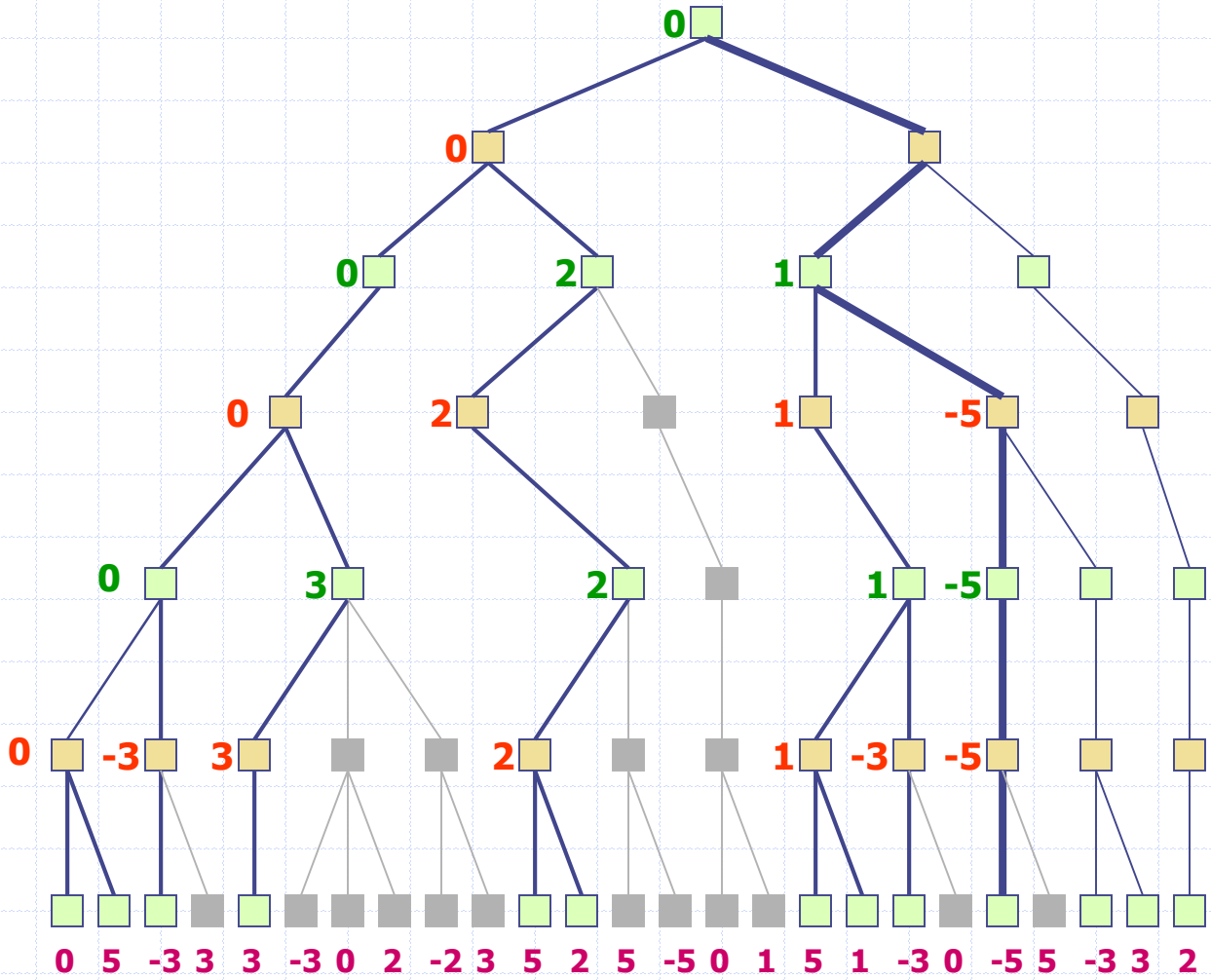
Alpha-Beta Example



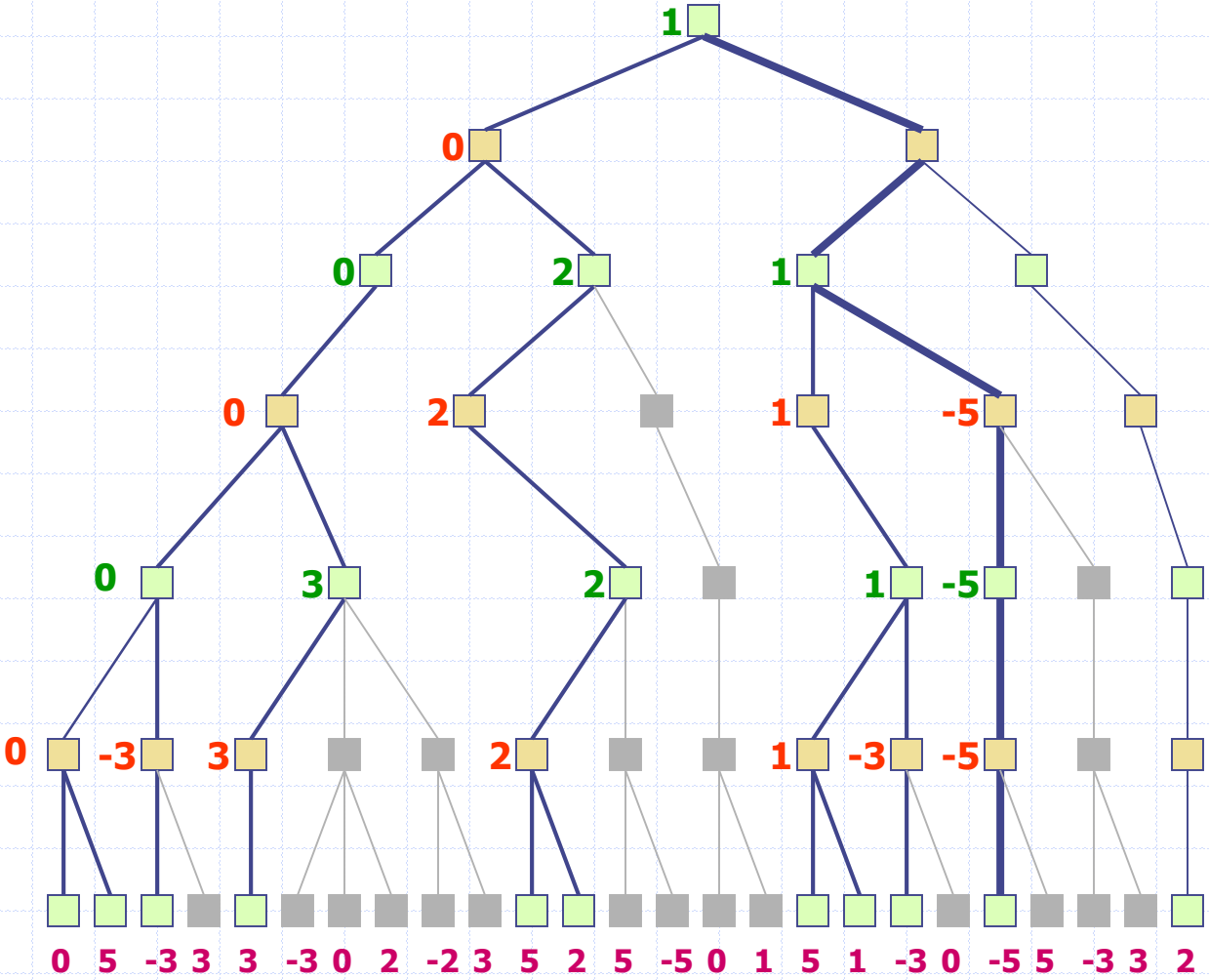
Alpha-Beta Example



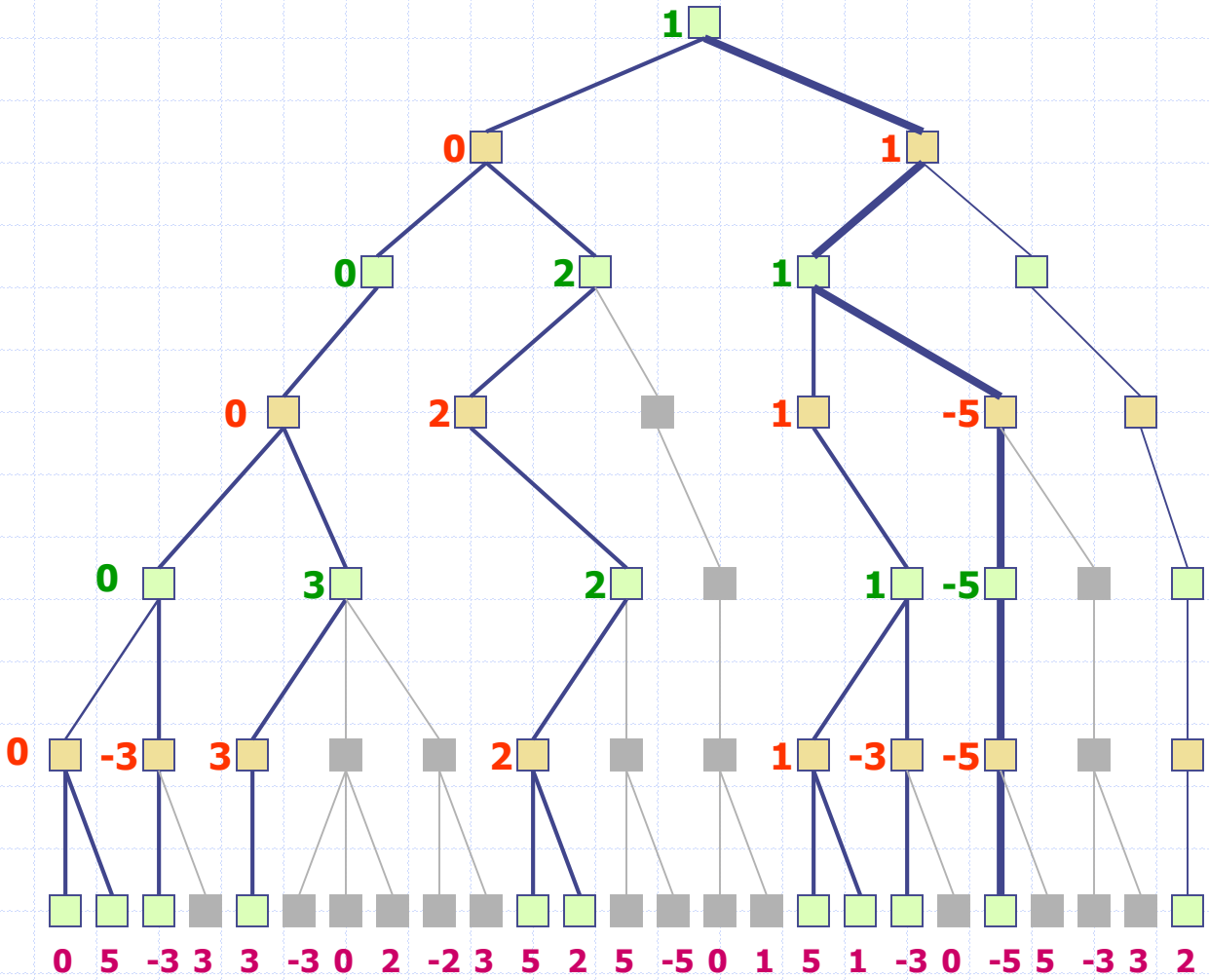
Alpha-Beta Example



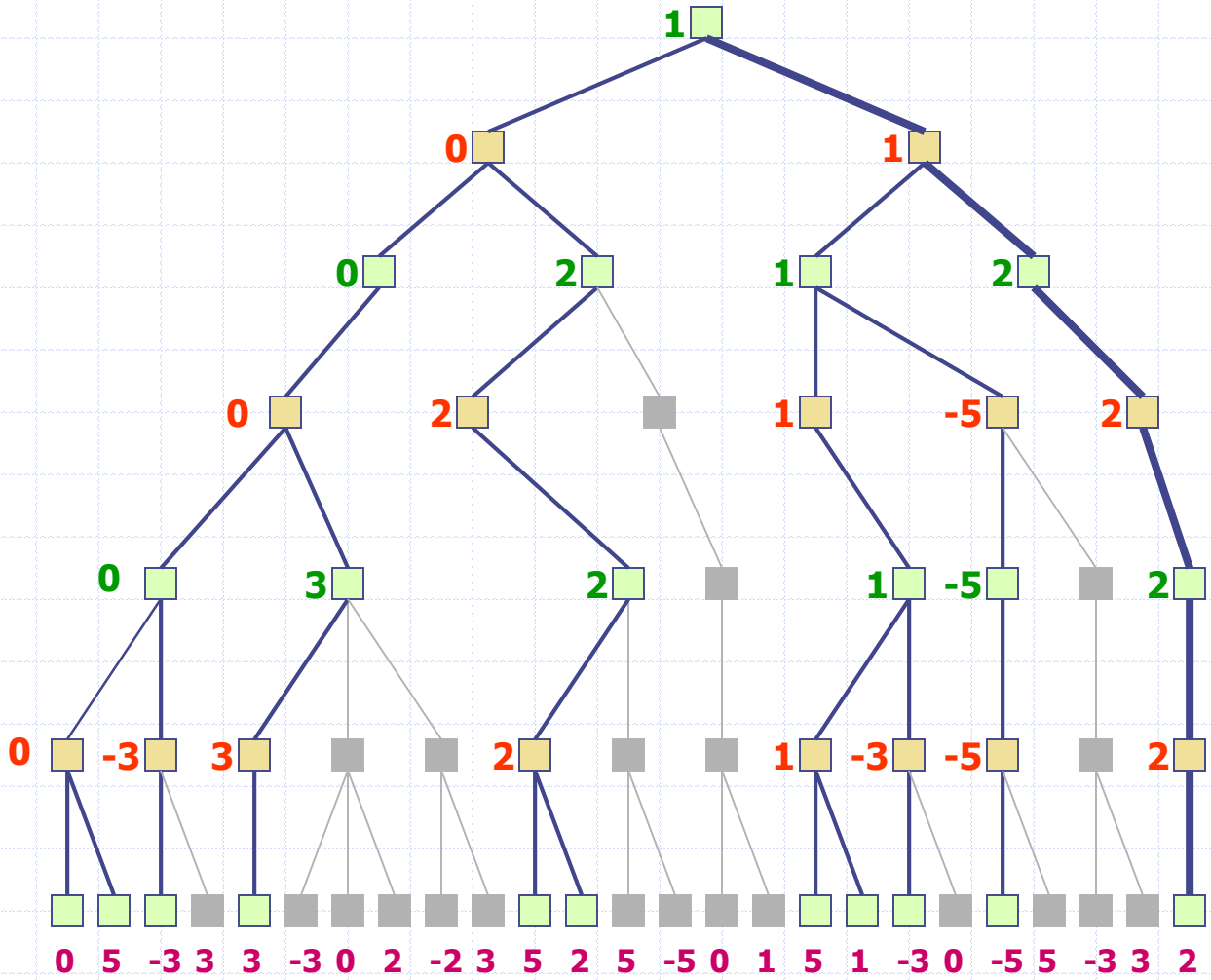
Alpha-Beta Example



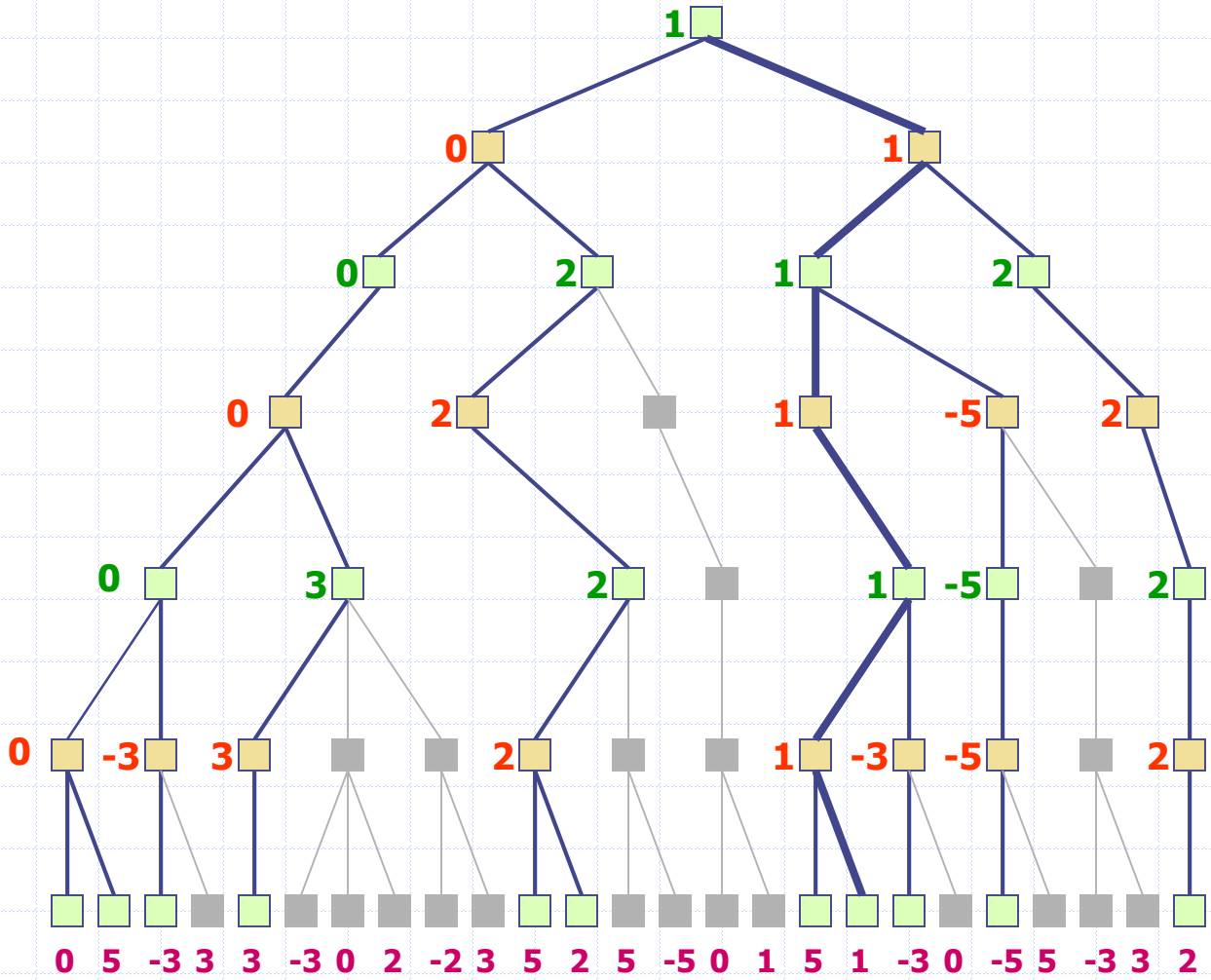
Alpha-Beta Example



Alpha-Beta Example



Alpha-Beta Example



Imperfect Real-time Decisions

In general the search tree is too big to make it possible to reach the terminal states!

Examples:

- Checkers: $\sim 10^{40}$ nodes
- Chess: $\sim 10^{120}$ nodes

SHANNON (1950):

Cut off search earlier (replace TERMINAL-TEST by CUTOFF-TEST)

Apply heuristic evaluation function EVAL (replacing utility function of alpha-beta)

Cutting off search

- Change**:**
  - **if** TERMINAL-TEST(*state*) **then return** UTILITY(*state*)

  into
  - **if** CUTOFF-TEST(*state,depth*) **then return** EVAL(*state*)

- Introduces a fixed-depth limit *depth*
  - Is selected so that the amount of time will not exceed what the rules of the game allow.
- When cuttoff occurs, the evaluation is performed.

Evaluation function

- **Evaluation function** or **static evaluator** is used to evaluate the "goodness" of a game position.
  - Contrast with heuristic search where the evaluation function was a non-negative estimate of the cost from the start node to a goal and passing through the given node
- The zero-sum assumption allows us to use a single evaluation function to describe the goodness of a board with respect to both players.
  - **f(n) >> 0**: position n good for me and bad for you
  - **f(n) << 0**: position n bad for me and good for you
  - **f(n) near 0**: position n is a neutral position
  - **f(n) = +infinity**: win for me
  - **f(n) = -infinity**: win for you

Evaluation function examples

- Example of an evaluation function for Tic-Tac-Toe:

  f(n) = [# of 3-lengths open for me] - [# of 3-lengths open for you]

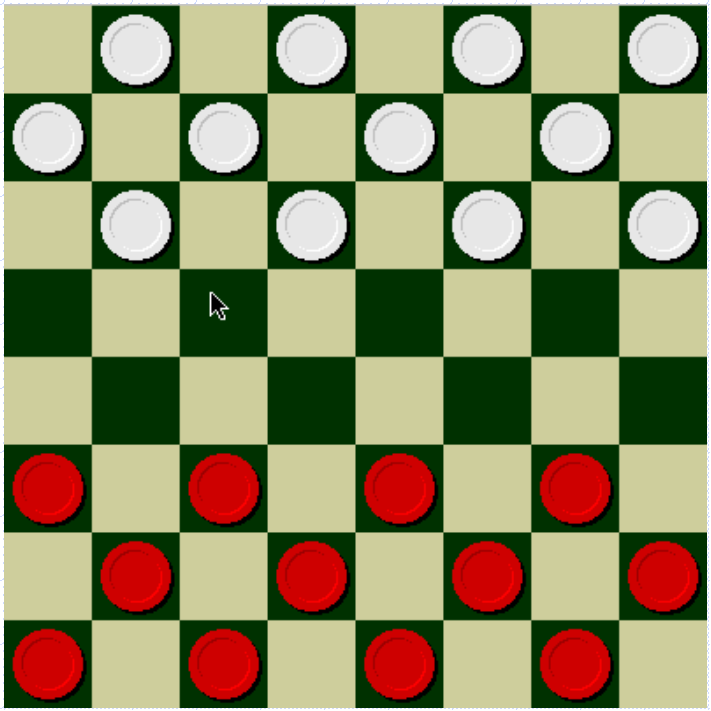
  where a 3-length is a complete row, column, or diagonal

- Alan Turing's function for chess
  - **f(n) = w(n)/b(n)** where w(n) = sum of the point value of white's pieces and b(n) = sum of black's
- Most evaluation functions are specified as a weighted sum of position features:

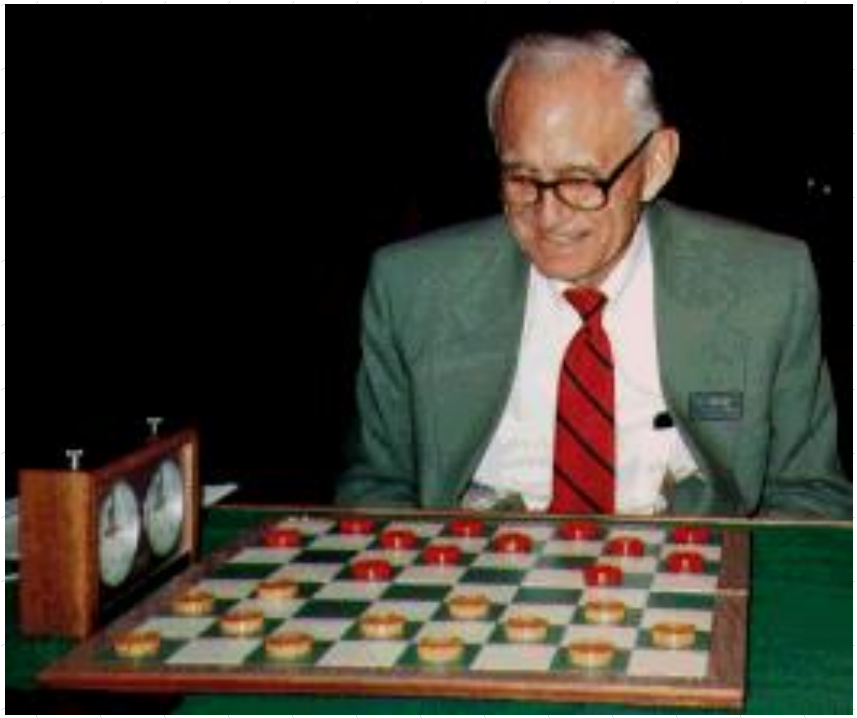
  $f(n) = w_1*feat_1(n) + w_2*feat_2(n) + \ldots + w_n*feat_k(n)$

- Example features for chess are piece count, piece placement, squares controlled, etc.
- Deep Blue has about 6000 features in its evaluation function

Some examples….**Checkers**

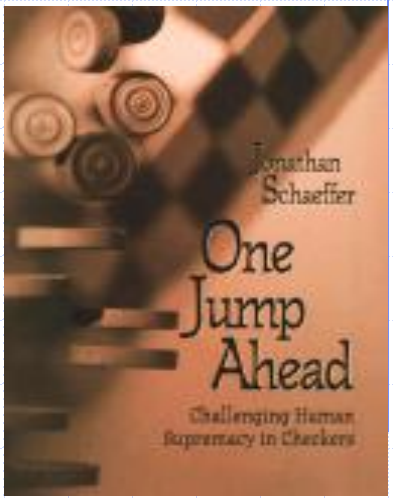


Chinook vs. Tinsley



Name: Marion Tinsley
World Checkers Champ
Record: Over 42 years
loses only 3 (!)
games of checkers

Chinook

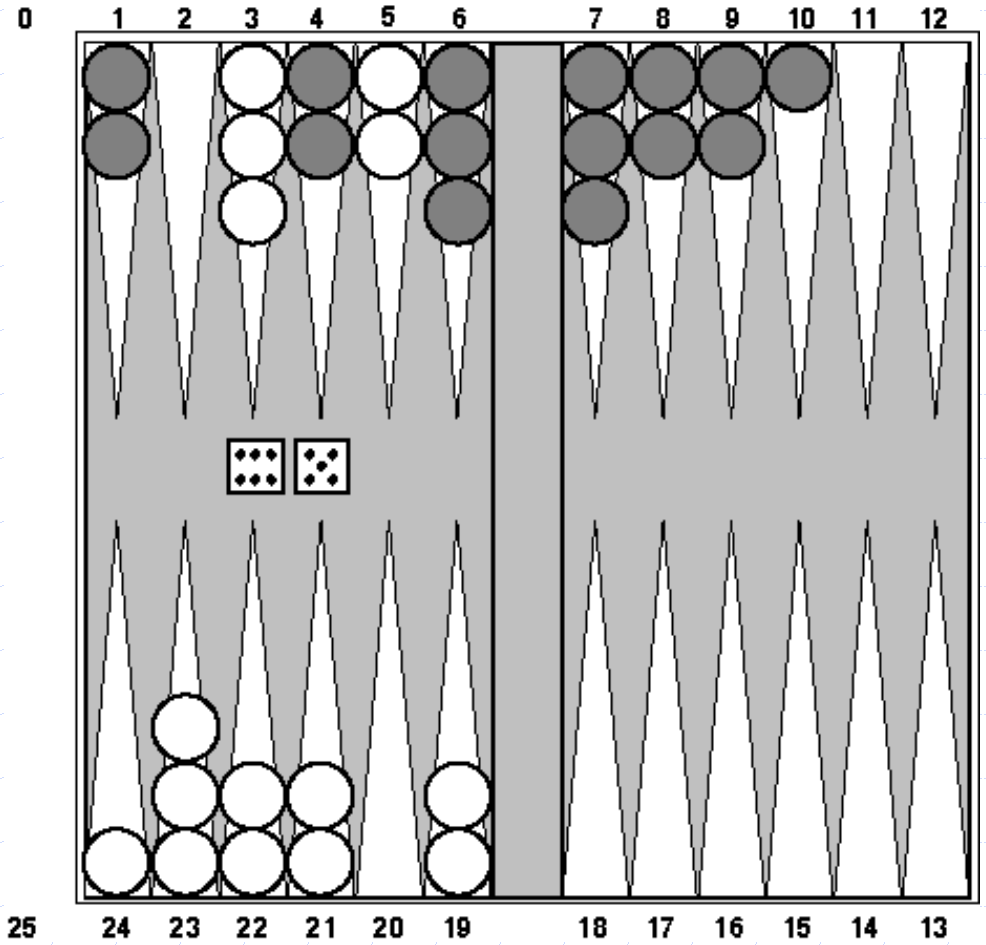


First computer to win human world championship!

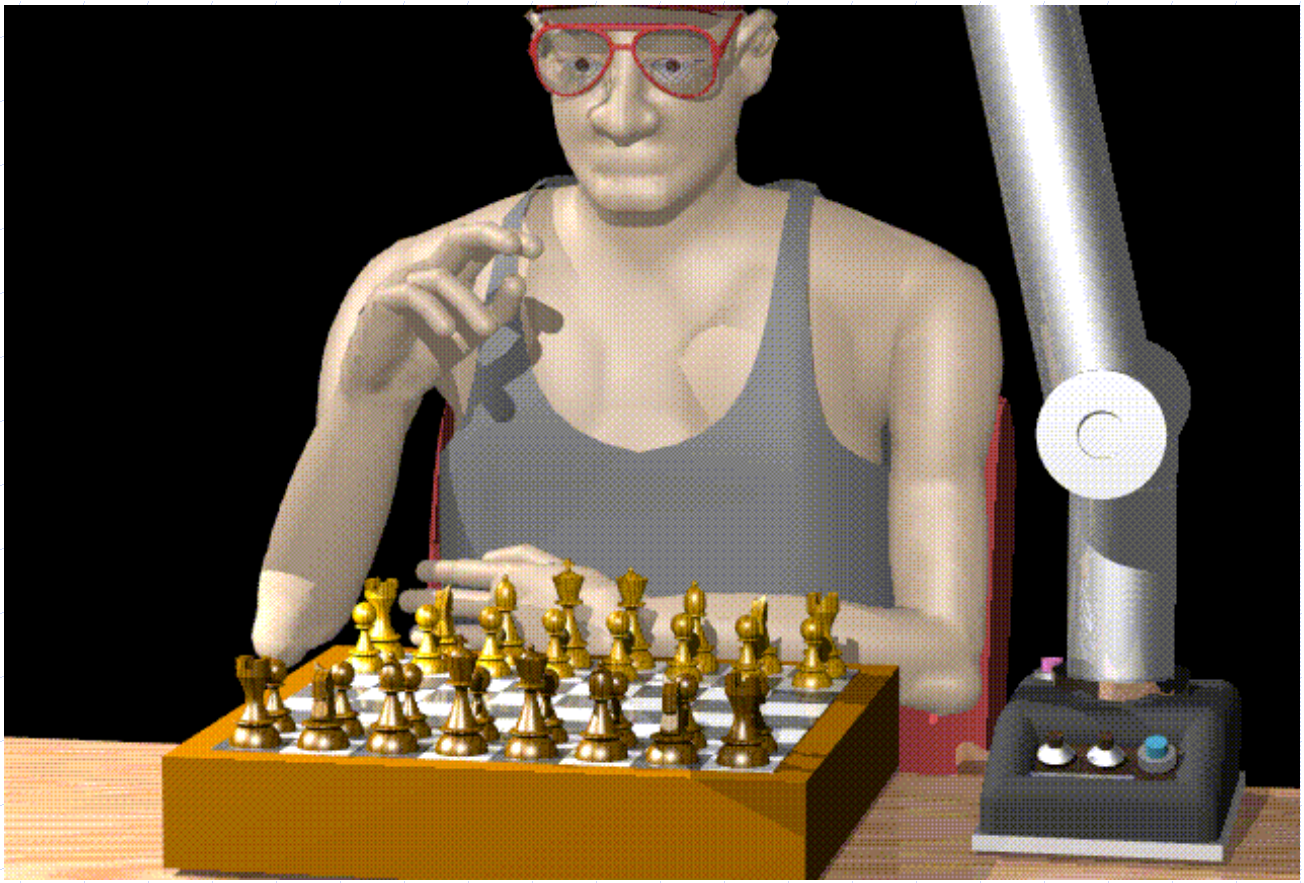
Visit http://www.cs.ualberta.ca/~chinook/ to play a version of Chinook over the Internet.

Backgammon

- branching factor several hundred
- TD-Gammon v1 – 1-step lookahead, learns to play games against itself
- TD-Gammon v2.1 – 2-ply search, does well against world champions
- TD-Gammon has changed the way experts play backgammon.



Chess



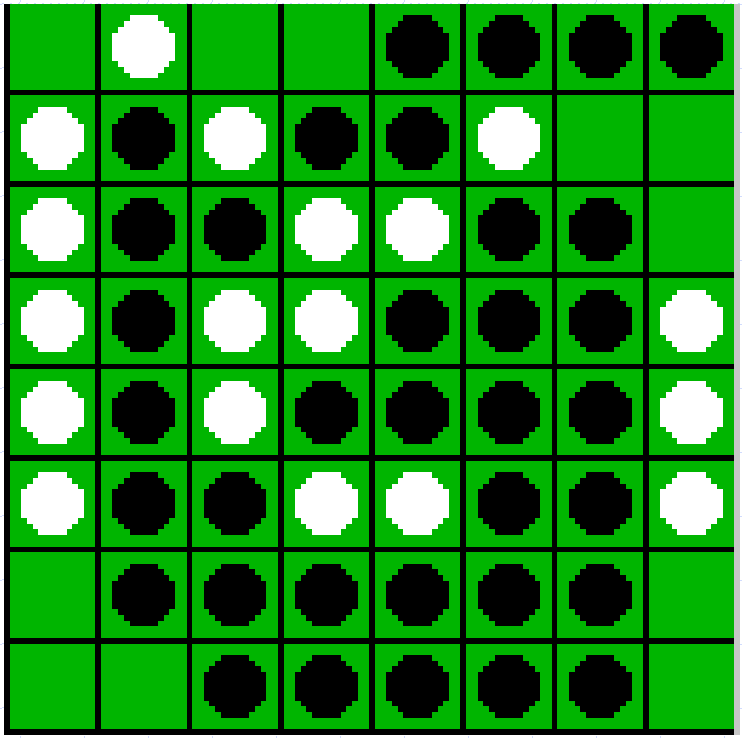
Man vs. Machine

Kasparov	Name	Deep Blue
5’10”	Height	6’ 5”
176 lbs	Weight	2,400 lbs
34 years	Age	4 years
50 billion neurons	Computers	512 processors
2 pos/sec	Speed	200,000,000 pos/sec
Extensive	Knowledge	Primitive
Electrical/chemical	Power Source	Electrical
Enormous	Ego	None

Chess

Name: Garry Kasparov
Title: World Chess Champion
Lost Chess Crown to a Machine ☹

Reversi/Othello



Othello

Name: Takeshi Murakami
Title: World Othello Champion
Crime: Man crushed by machine

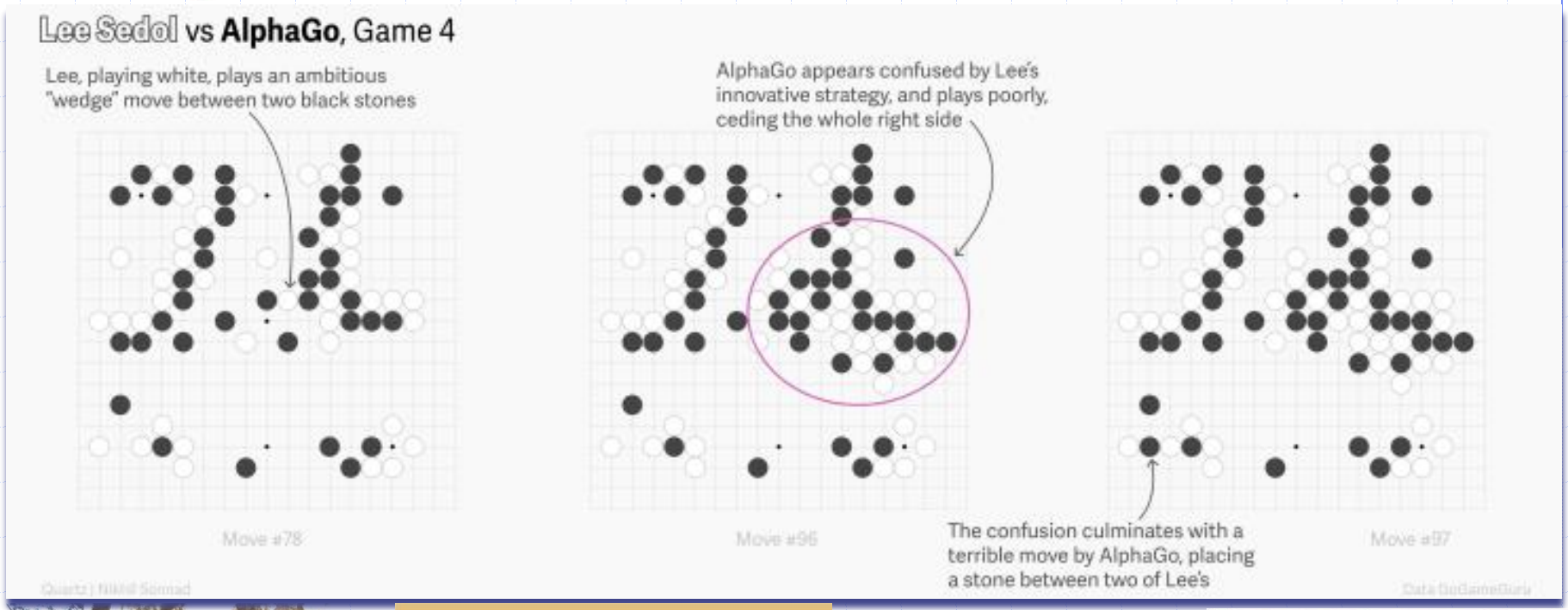
Computer Go:

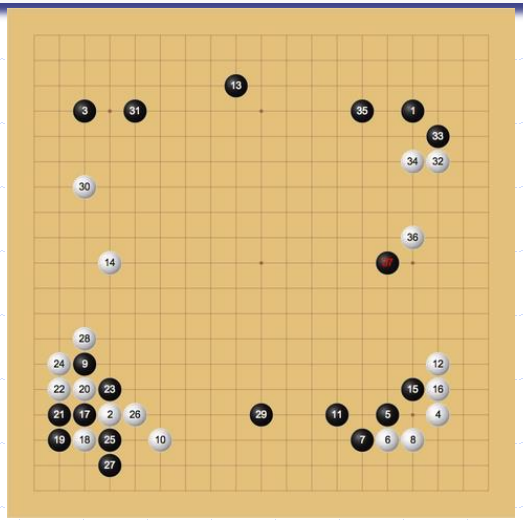
Name: Chen Zhixing
Author: Handtalk (Goemate)
Profession: Retired
Computer skills: self-taught assembly language programmer
Accomplishments: dominated computer go for 4 years.

Go VS. Human Opponents

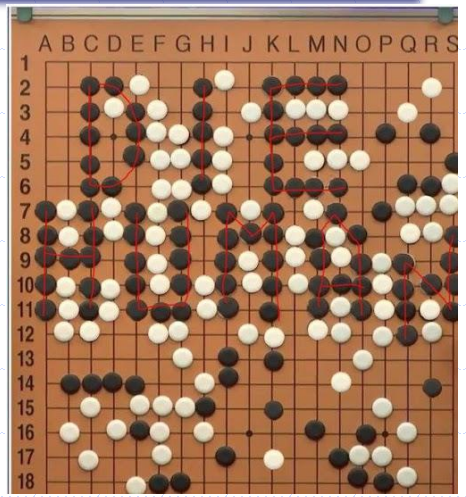
Gave Handtalk a 9 stone handicap and still easily beat the program, thereby winning $15,000

Moves 78, 37

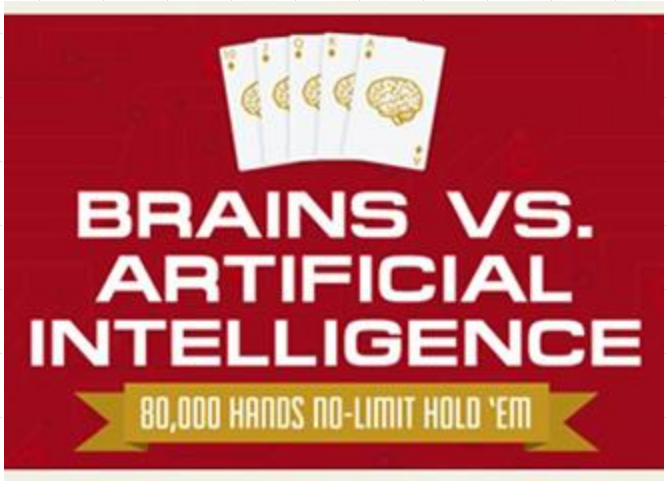




Move 37 →



IBM Watson


BRAINS VS.
ARTIFICIAL
INTELLIGENCE
80,000 HANDS NO-LIMIT HOLD 'EM


PIENSE
THINK
ΣΚΕΨΟΥ
DENKE
PENSER
$300,000
Who is Stoker?
$1,000
$1,000,000
Who is Bram Stoker?
$17,973
$200,000
$5600


THINK
Who is Stoker?
(I FOR ONE WELCOME OUR NEW COMPUTER OVERLORDS)
$18,200

Deterministic games in practice

Checkers: Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994. Used an endgame database defining perfect play for all positions involving 8 or fewer pieces on the board, a total of 443,748,401,247 positions.

Chess: Deep Blue defeated human world champion Gary Kasparov in a six-game match in 1997. Deep Blue searches 200 million positions per second, uses very sophisticated evaluation, and undisclosed methods for extending some lines of search up to 40 ply.

Othello: human champions refuse to compete against computers, who are too good.

Go: human champions refuse to compete against computers, who are too bad. In go, $b > 300$, so most programs use pattern knowledge b
suggest plausible moves.

The End!