CSC242: Introduction to Artificial Intelligence

Lecture 1.4

Please put away all electronic devices

Adversarial Search

Games ≠ Toy Problems

8-puzzle: 181,440

15-puzzle: ~1.3×10^{12}

24-puzzle: ~10^{25}

$35^{100} = 10^{154}$

(only 10^{40} distinct)

2×10^{170}

Types of Games

Deterministic (no chance)	Nondeterministic (dice, cards, etc.)
Perfect information (fully observable)	Imperfect information (partially observable)
Zero-sum (total payoff the same in any game)	Arbitrary utility functions

Deciding what to do in a game requires thinking about what the opponent will do, and having a **strategy** that takes that into account

Not simply a sequence of actions, but a contingency plan that specifies what to do depending on what state one finds oneself in (AIMA 4.3)

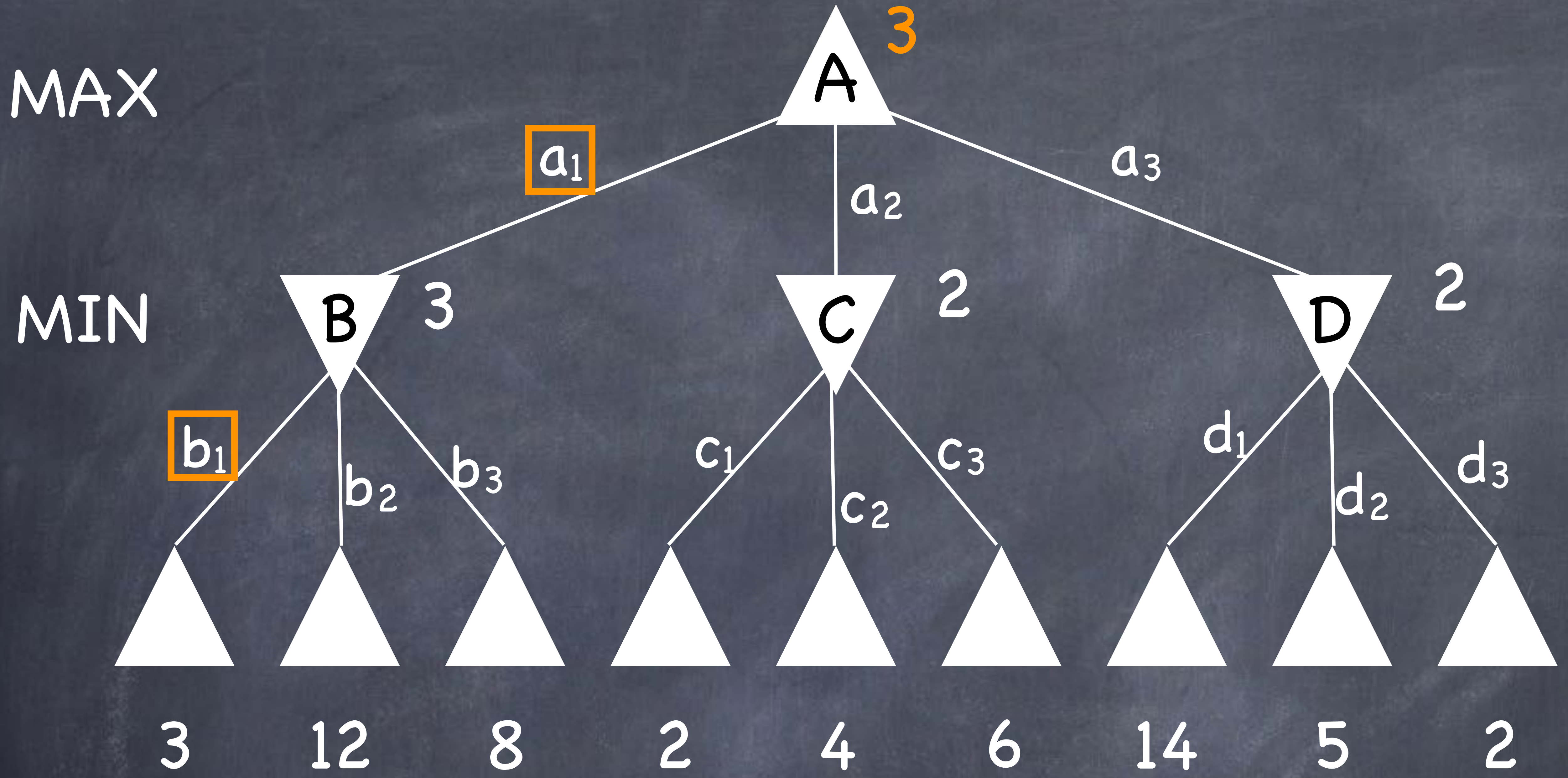
Minimax Algorithm

$$\text{MINIMAX}(s) = \begin{cases} \text{UTILITY}(s) & \text{if TERMINAL-TEST}(s) \\ \max_{a \in \text{ACTIONS}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if PLAYER}(s) = \text{MAX} \\ \min_{a \in \text{ACTIONS}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if PLAYER}(s) = \text{MIN} \end{cases}$$

Minimax Algorithm

$$\textsc{Minimax}(s) =$$

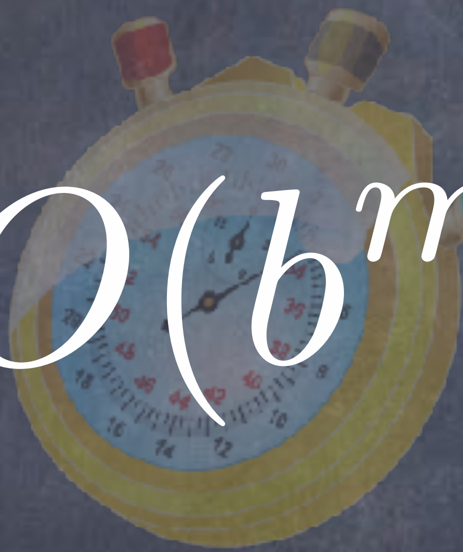
$$\begin{cases} \boxed{\textsc{Utility}(s)} & \boxed{\text{if } \textsc{Terminal-Test}(s)} \\ \max_{a \in \textsc{Actions}(s)} \textsc{Minimax}(\textsc{Result}(s,a)) & \text{if } \textsc{Player}(s) = \textsc{max} \\ \min_{a \in \textsc{Actions}(s)} \textsc{Minimax}(\textsc{Result}(s,a)) & \text{if } \textsc{Player}(s) = \textsc{min} \end{cases}$$


MAX
MIN
A
3
a1
a2
a3
B
3
C
2
D
2
b1
b2
b3
c1
c2
c3
d1
d2
d3
3
12
8
2
4
6
14
5
2

Minimax Summary

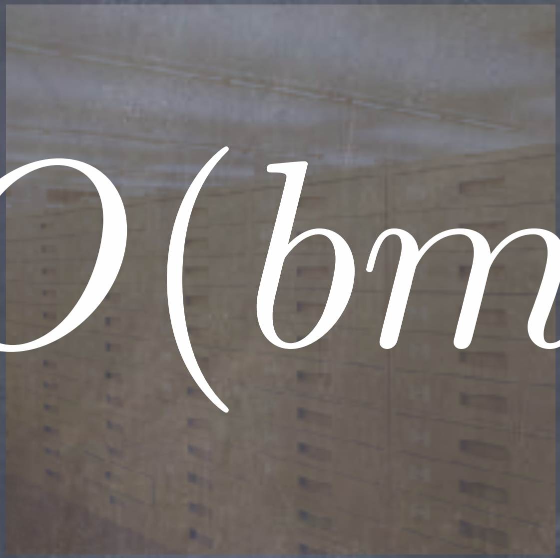
- Computes the optimal move assuming opponent also plays optimally (i.e., worst-case outcome)
- Explores game tree depth-first all the way to terminal states (end of game)
- Backs up utility values through alternating MIN and MAX (what's best for me is worst for you, and vice-versa)

Minimax Analysis



$$O(b^m)$$

Time Complexity



$$O(bm)$$

Space Complexity

"for real games, the time cost is totally impractical"

Heuristic Minimax

$$
\text{H-}\textsc{Minimax}(s) =
\begin{cases}
h(s) & \text{if } \textsc{Cutoff-Test}(s) \\
\max_{a \in \textsc{Actions}(s)} \textsc{Minimax}(\textsc{Result}(s, a)) & \text{if } \textsc{Player}(s) = \textsc{max} \\
\min_{a \in \textsc{Actions}(s)} \textsc{Minimax}(\textsc{Result}(s, a)) & \text{if } \textsc{Player}(s) = \textsc{min}
\end{cases}
$$

H-Minimax

- When to cutoff search?
  - Time, depth, “quiesence”, …
- How evaluate non-terminal states?
  - Depends on game
  - Tradeoff between time and quality

Adversarial Search

- In deterministic, perfect information, zero-sum games:
  - How to find optimal moves
    - MINIMAX (requires utility fn)
  - How to find good moves when time is limited
    - H-MINIMAX (requires cutoff & h fn)

Google/DeepMind AlphaGo

- Monte Carlo tree search (MCTS) with:
  - Move selection (“Policy”)
  - Position evaluation (“Value”)
- Neural networks compute mapping from board positions to moves and evaluations
- Trained using combination of supervised and reinforcement learning

Google/DeepMind AlphaGo

- Monte Carlo tree search (MCTS) with:
    - Move selection ("Policy")
    - Position evaluation ("Value")
- Neural networks compute mapping from board positions to moves and evaluations
- Trained using combination of supervised and reinforcement learning

Google/DeepMind AlphaGo

- Monte Carlo tree search (MCTS) with:
  - Move selection ("Policy")
  - Position evaluation ("Value")
- Neural networks compute mapping from board positions to moves and evaluations
- Trained using combination of supervised and reinforcement learning

Google/DeepMind AlphaGo

- Monte Carlo tree search (MCTS) with:
  - Move selection ("Policy") $\rightarrow$ reduces b
  - Position evaluation ("Value")
- Neural networks compute mapping from board positions to moves and evaluations
- Trained using combination of supervised and reinforcement learning

Google/DeepMind AlphaGo

- Monte Carlo tree search (MCTS) with:
  - Move selection (“Policy”)
  - Position evaluation (“Value”) $\rightarrow h(n)$
- Neural networks compute mapping from board positions to moves and evaluations
- Trained using combination of supervised and reinforcement learning

Google/DeepMind AlphaGo

- Monte Carlo tree search (MCTS) with:
  - Move selection (“Policy”)
  - Position evaluation (“Value”)
- Neural networks compute mapping from board positions to moves and evaluations
- Trained using combination of supervised and reinforcement learning

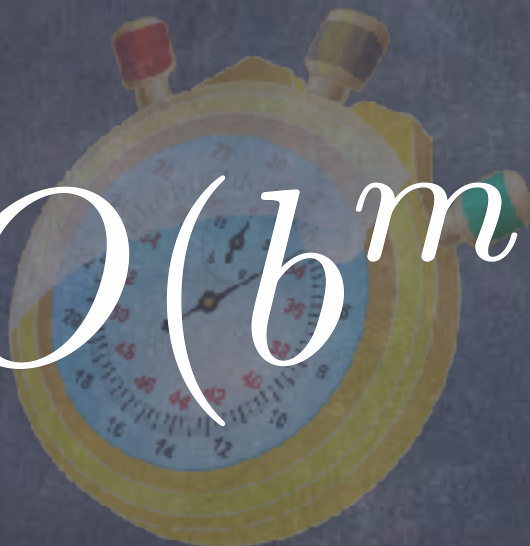
Google/DeepMind AlphaGo

http://deepmind.com/alpha-go.html

Silver, D., et al. (2016). Mastering the game of Go with deep neural networks and tree search. Nature 529, 484–489.

http://doi.org/10.1038/nature16961

Minimax Analysis

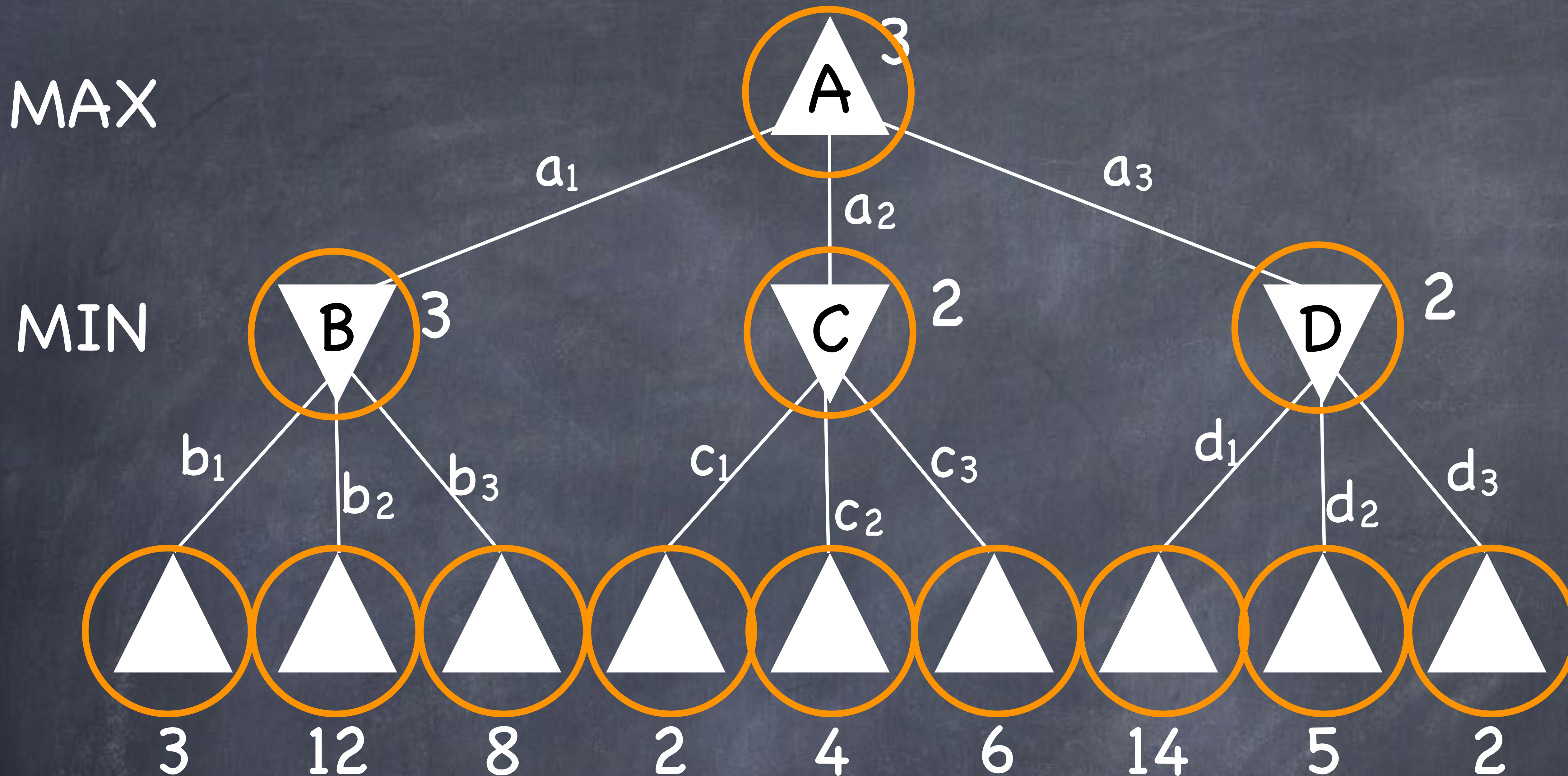


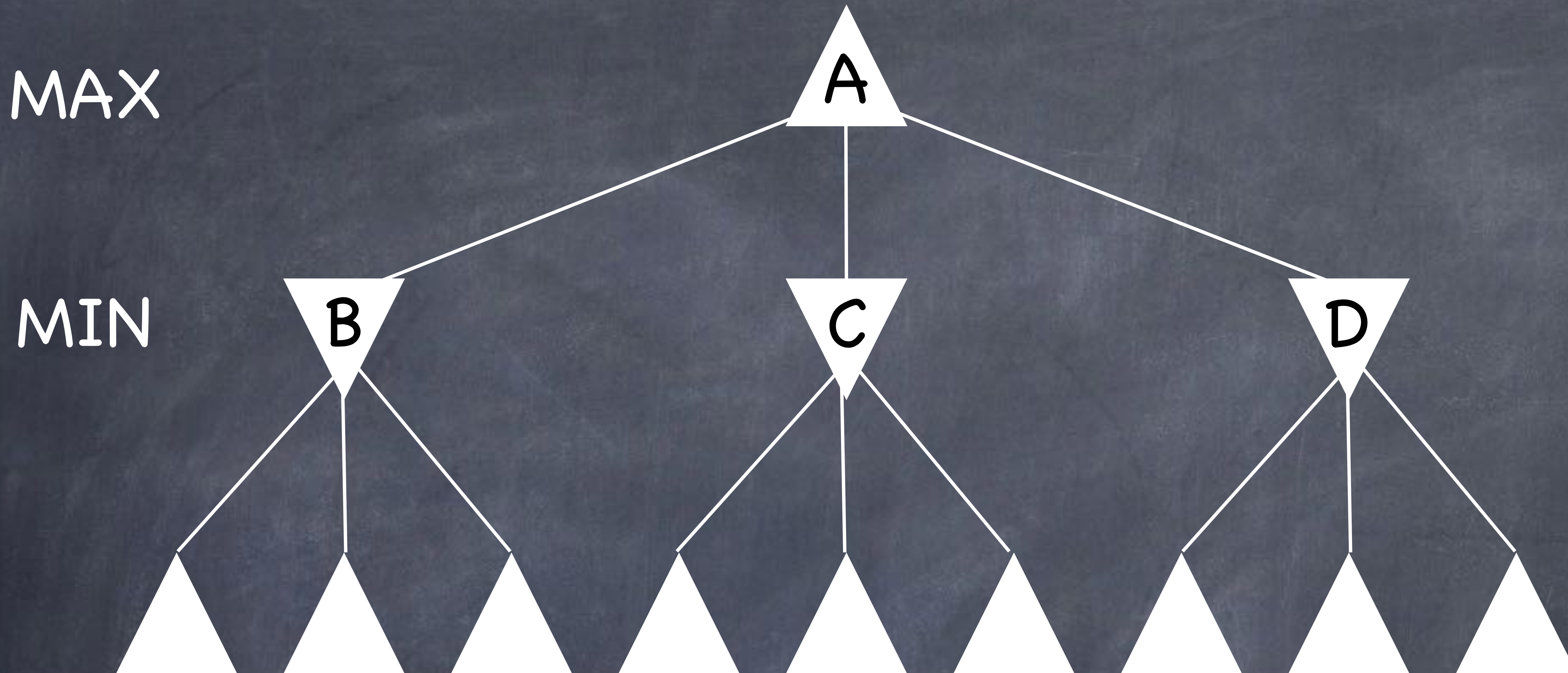
$$O(b^m)$$

Time Complexity

$$O(bm)$$

Space Complexity

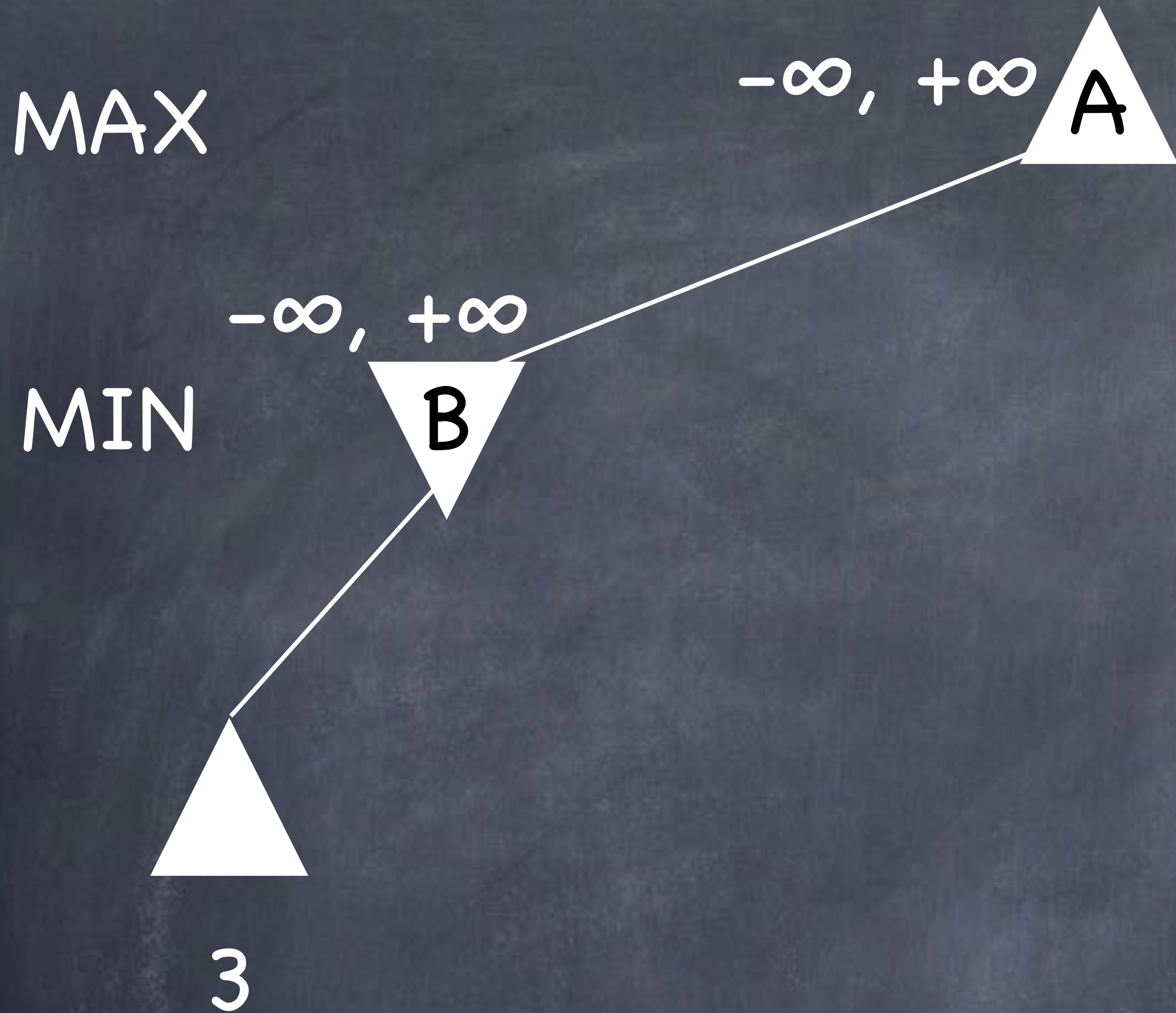

MAX
MIN
A
3
a1
a2
a3
B
3
C
2
D
2
b1
b2
b3
c1
c2
c3
d1
d2
d3
3
12
8
2
4
6
14
5
2

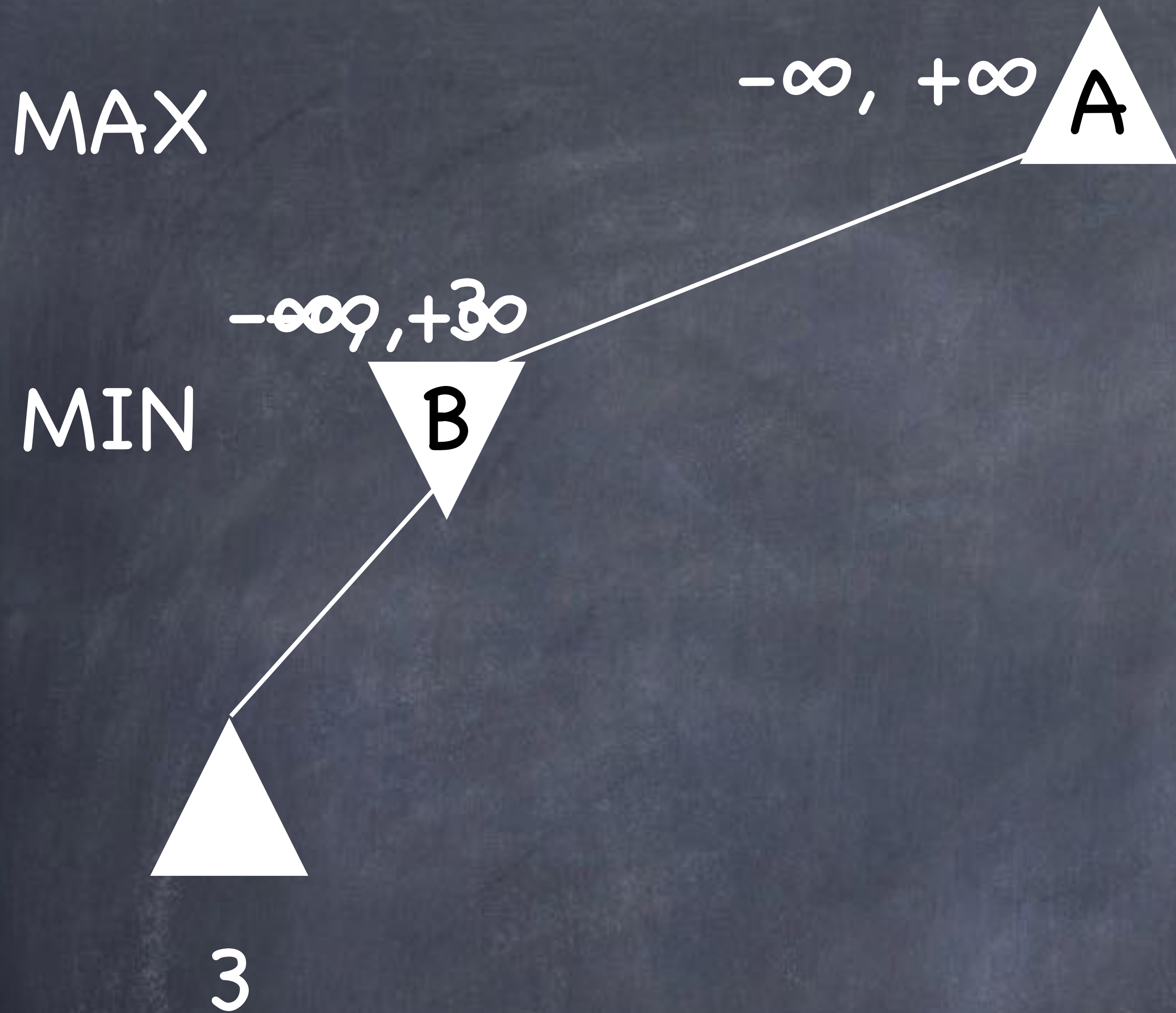


α, β

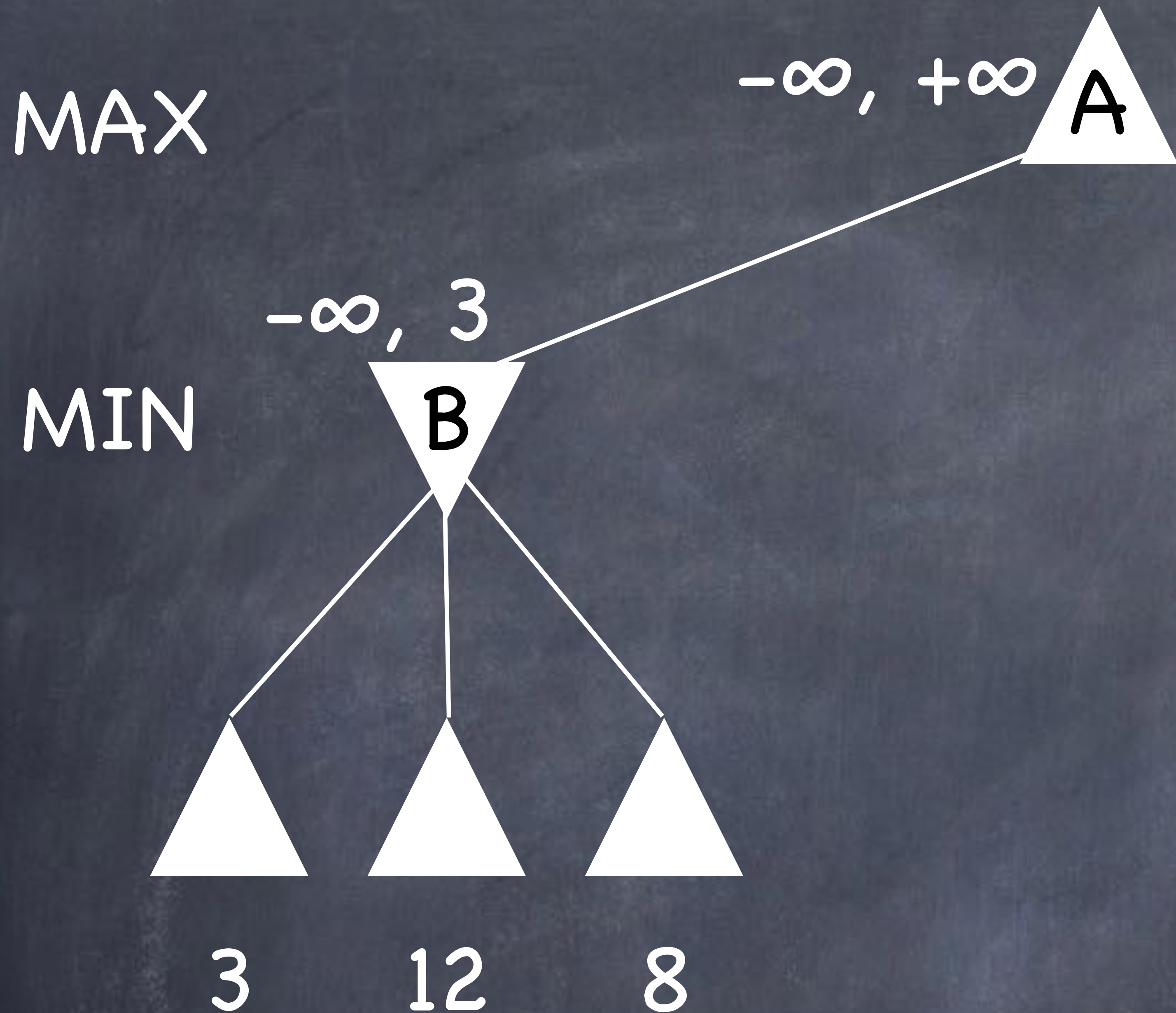
Highest utility seen so far
(lower bound on max value)

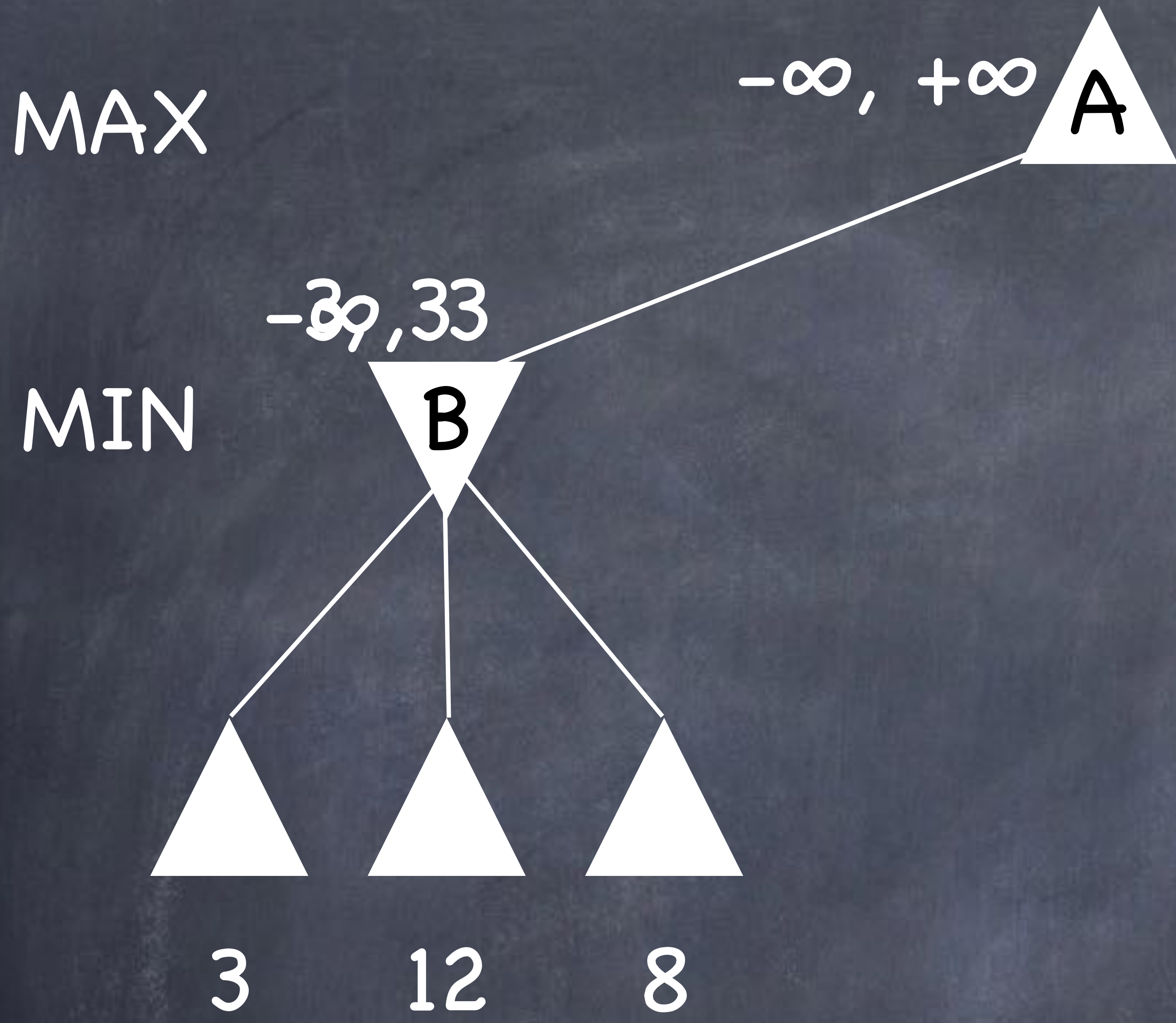
Lowest utility seen so far
(upper bound on min value)

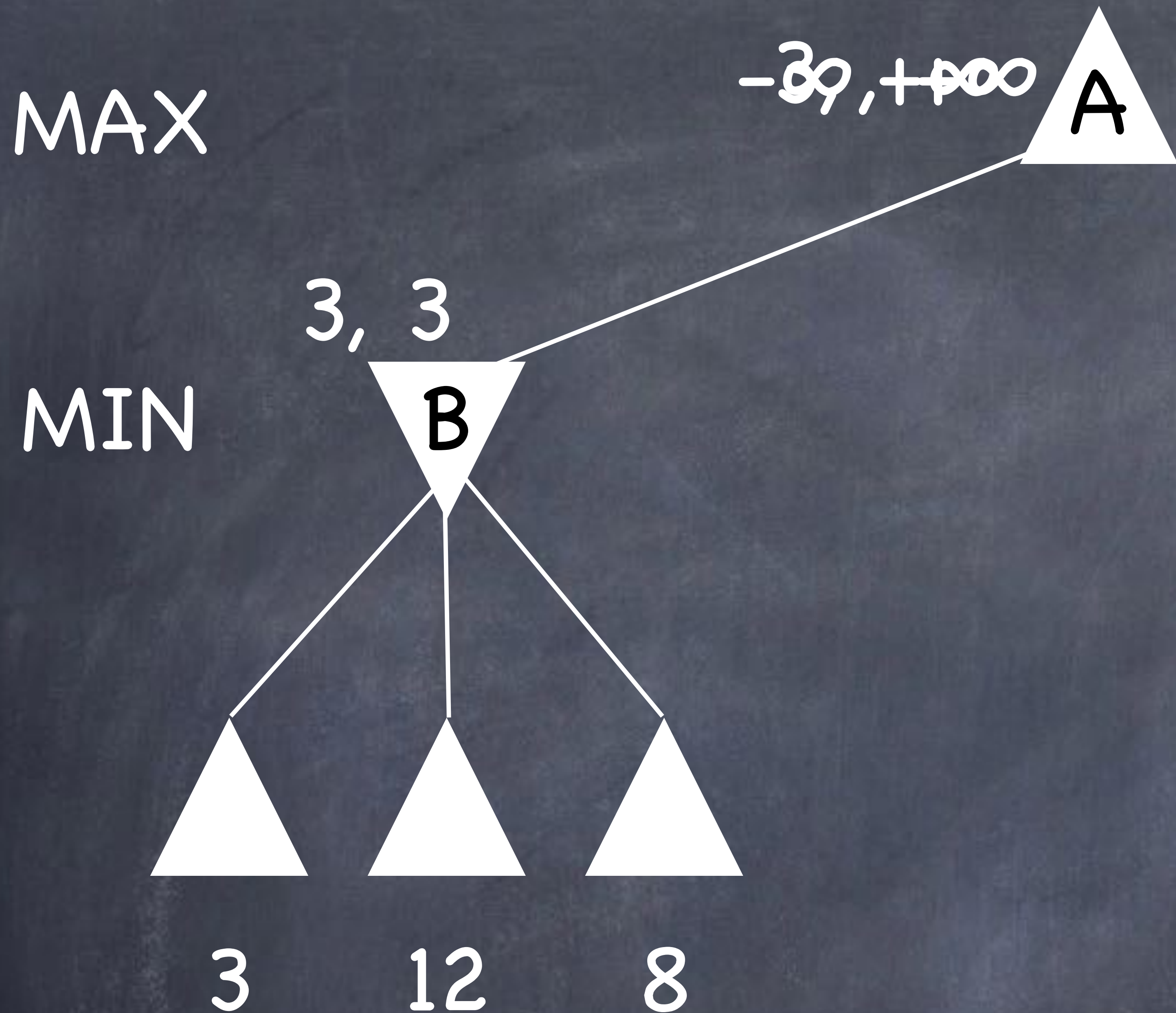

MAX
−∞, +∞
A
MIN
−∞, +∞
B
3

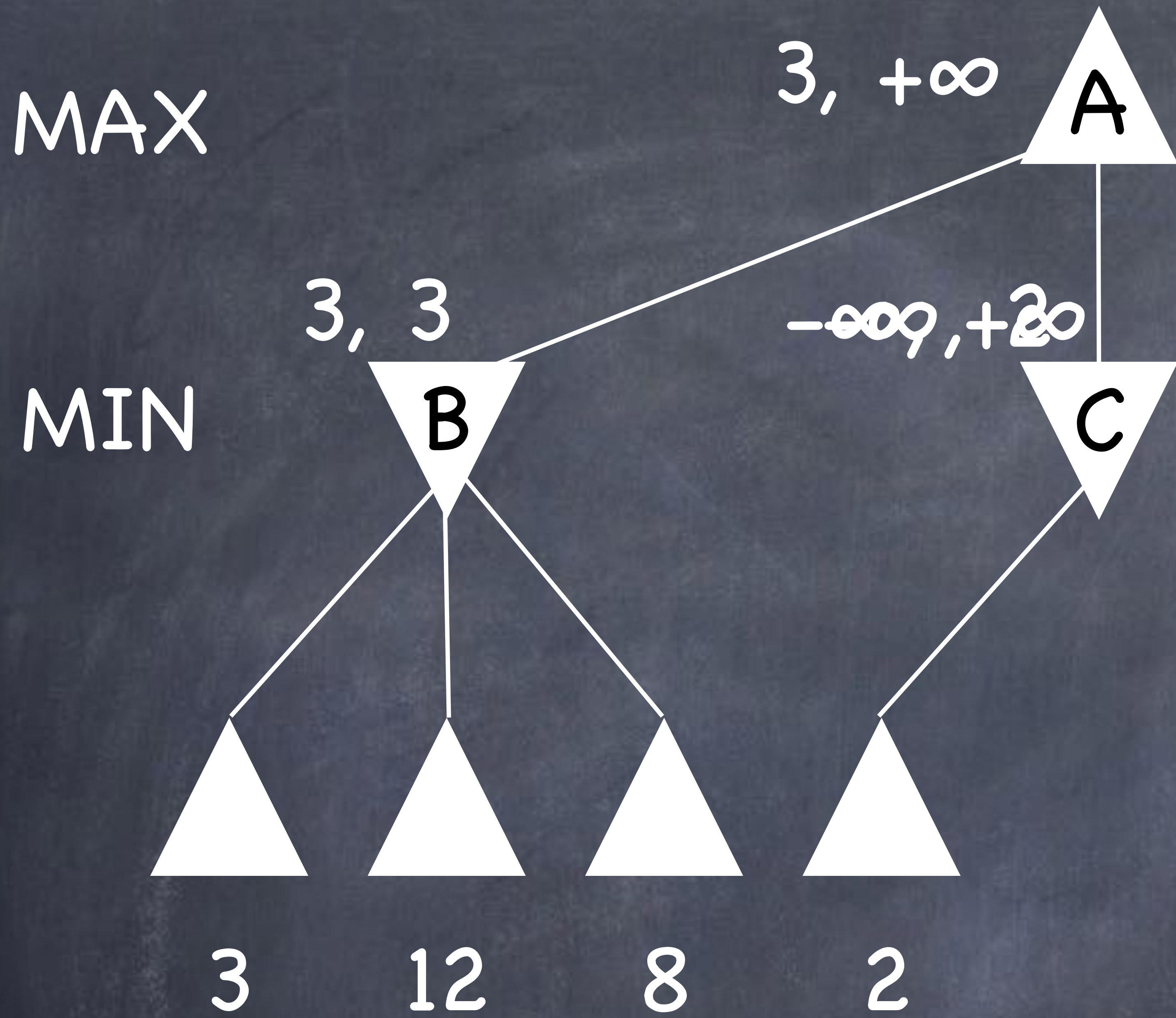


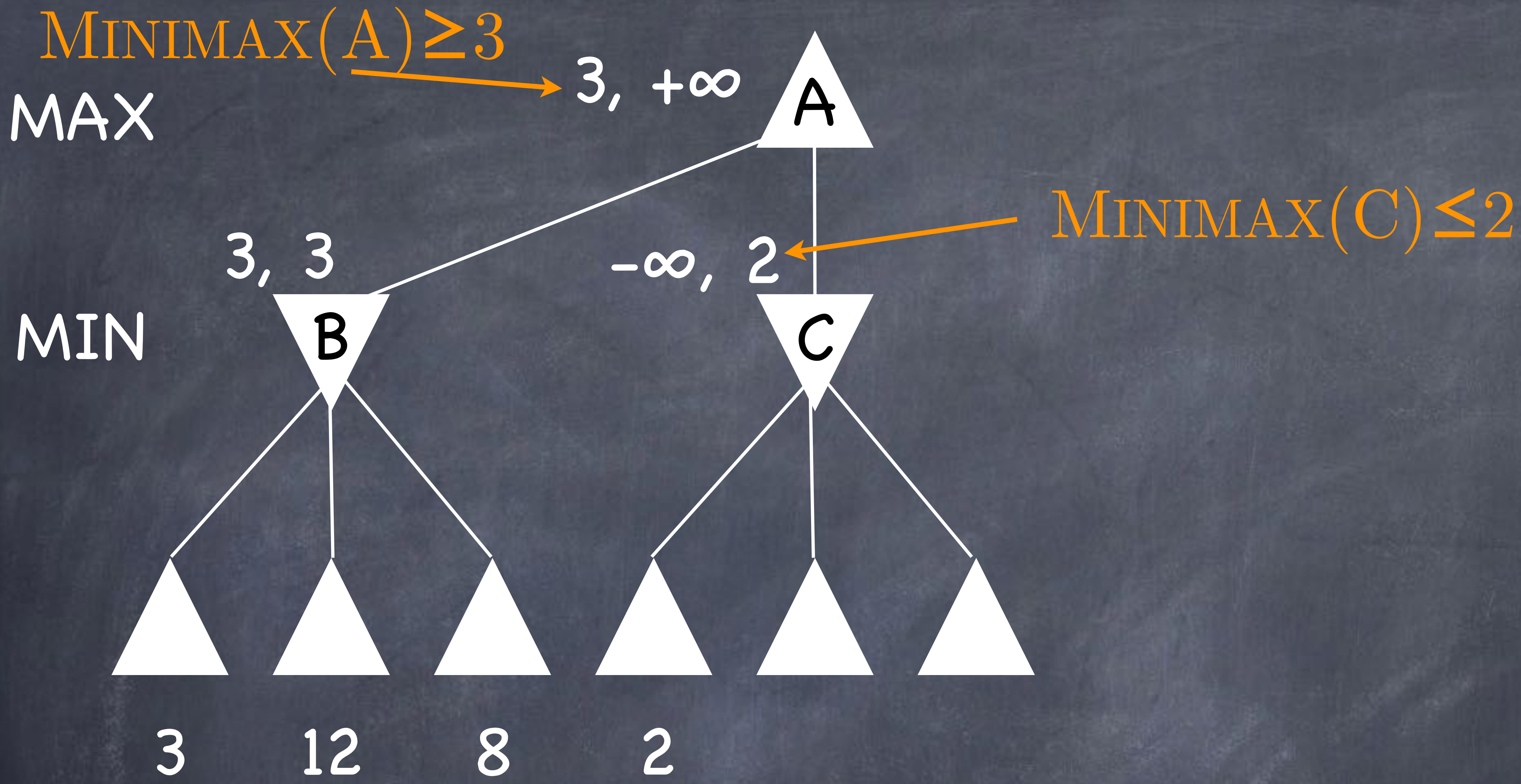
MAX
-∞, +∞
A
MIN
B
3


MAX
MIN
−∞, +∞
A
−∞, 3
B
3
12
8

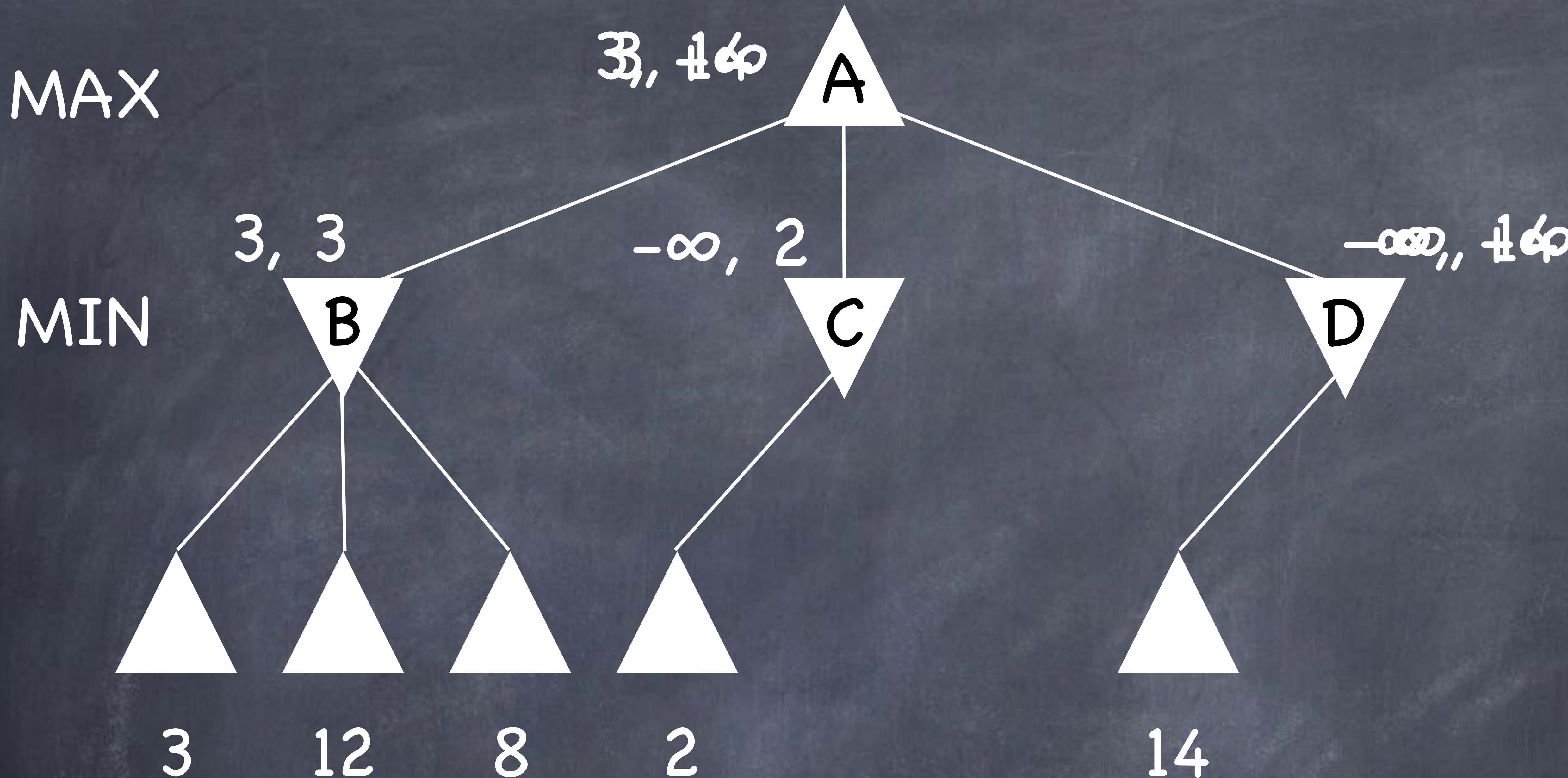

MAX
MIN
−∞, +∞
A
B
3
12
8

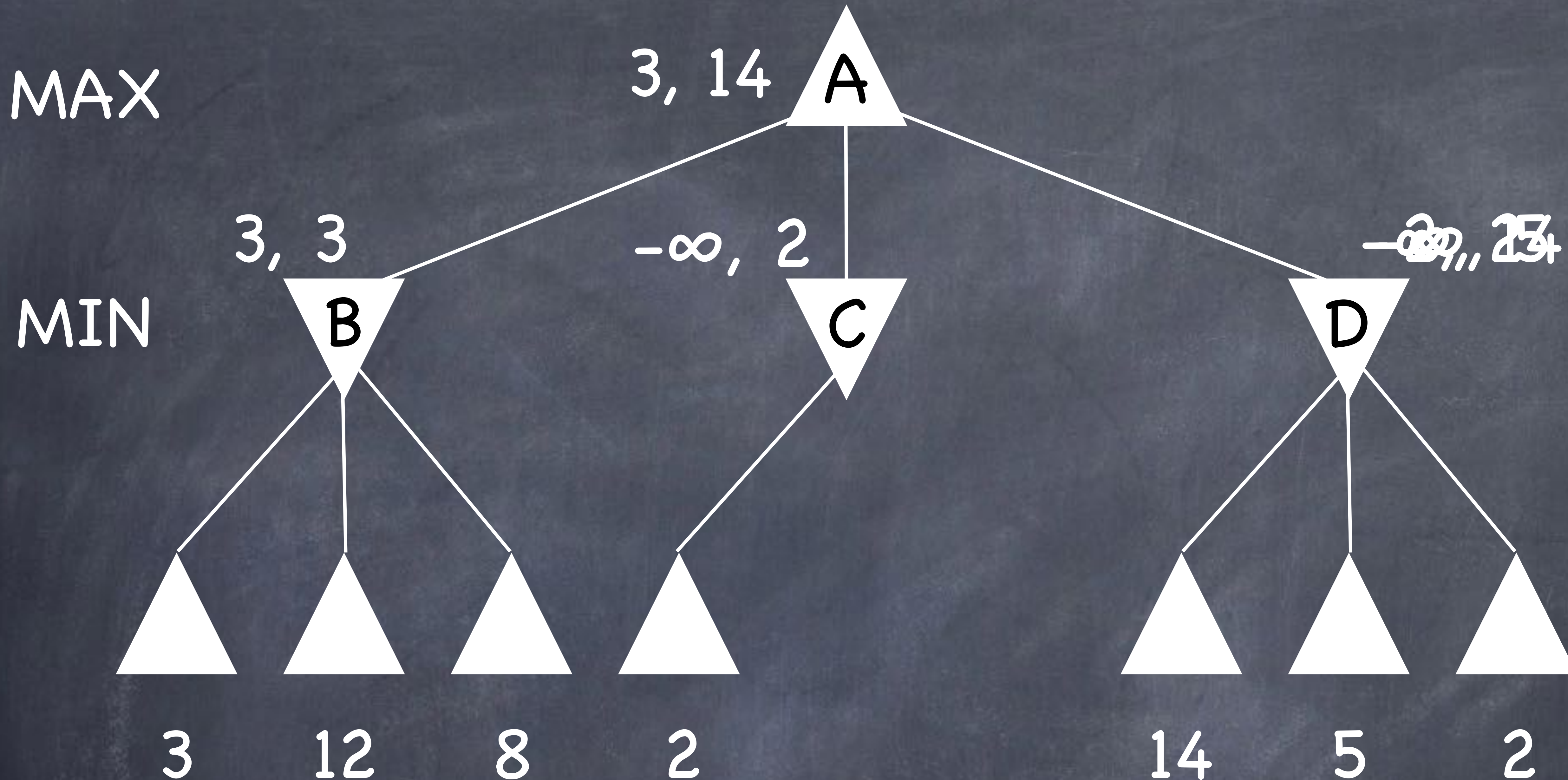

MAX
MIN
A
3, 3
B
3
12
8

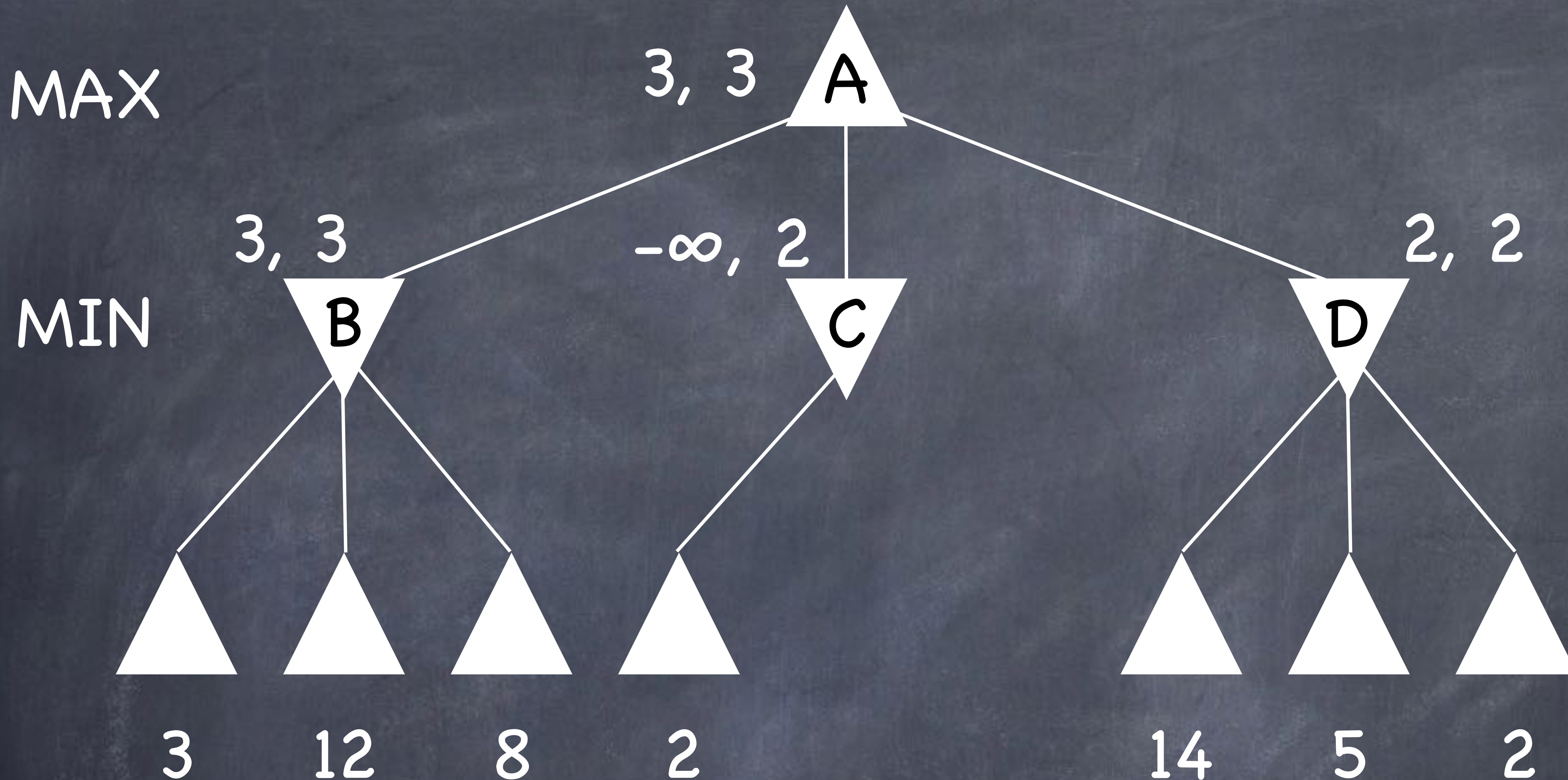

MAX
MIN
3, +∞
A
3, 3
B
−∞, +∞
3, 2
C
3
12
8
2



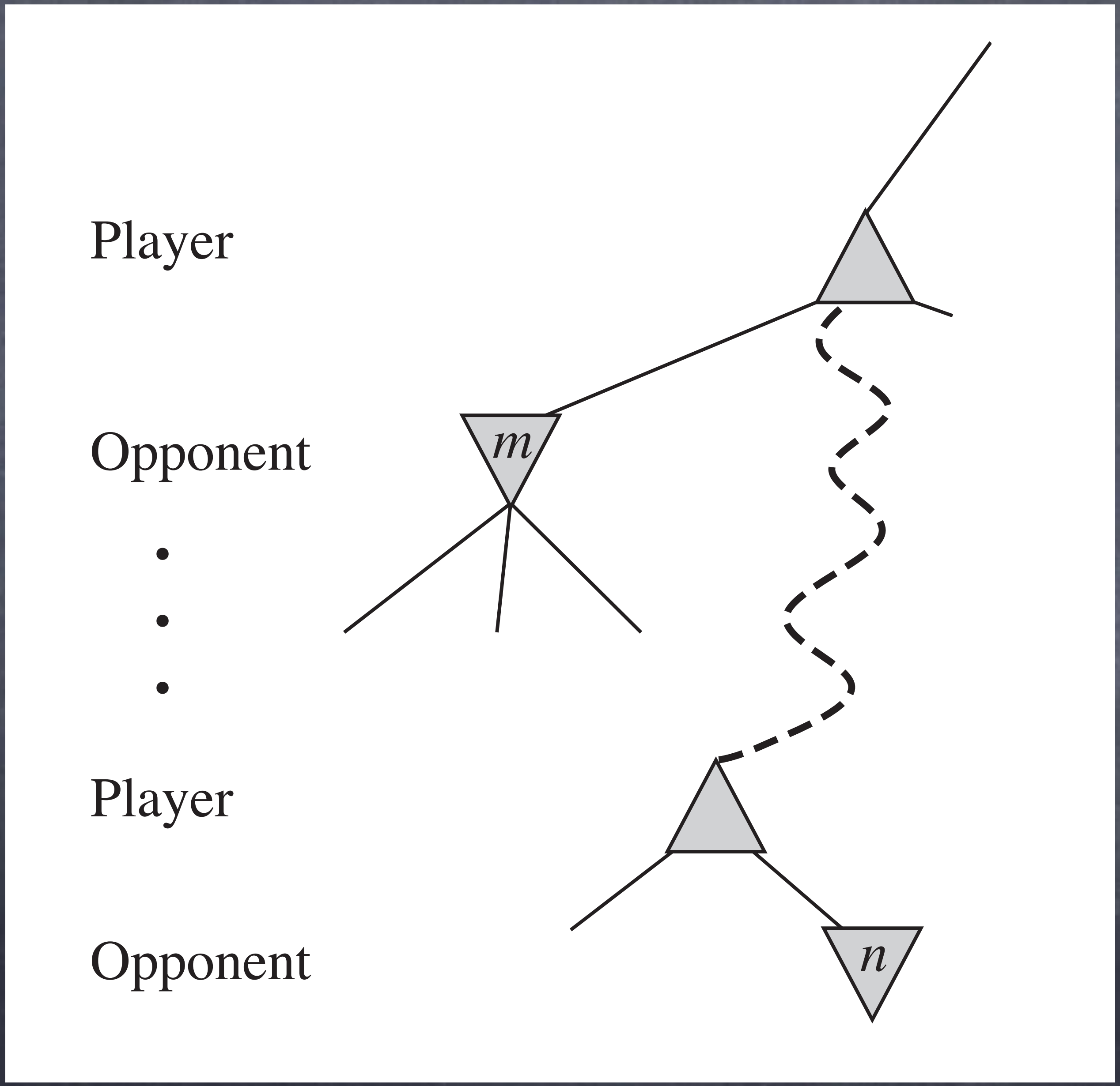
MINIMAX(A)≥3
3, +∞
MAX
A
MINIMAX(C)≤2
3, 3
-∞, 2
MIN
B
C
3
12
8
2


MAX
MIN
A
B
C
D
3, 3
-∞, 2
3
12
8
2
14


MAX
3, 14
A
MIN
3, 3
B
-∞, 2
C
D
3
12
8
2
14
5
2


MAX
MIN
3, 3
A
3, 3
B
-∞, 2
C
2, 2
D
3
12
8
2
14
5
2

Alpha-Beta Pruning



AIMA Fig 5.6

Alpha-Beta Pruning

- During MINIMAX search
  - Keep track of
    - α: value of best choice so far for MAX (lower bound on MAX utility)
    - β: value of best choice so far for MIN (upper bound on MIN utility)
  - Prune when value of node is known to be worse than α (for MAX) or β (for MIN)

Alpha-Beta Pruning

- Still MINIMAX search
  - Optimal (if you search to the terminals)
  - Optimal with respect to your heuristic function otherwise

Alpha-Beta Pruning

- Still MINIMAX search
  - Optimal (if you search to the terminals)
  - Optimal with respect to your heuristic function otherwise

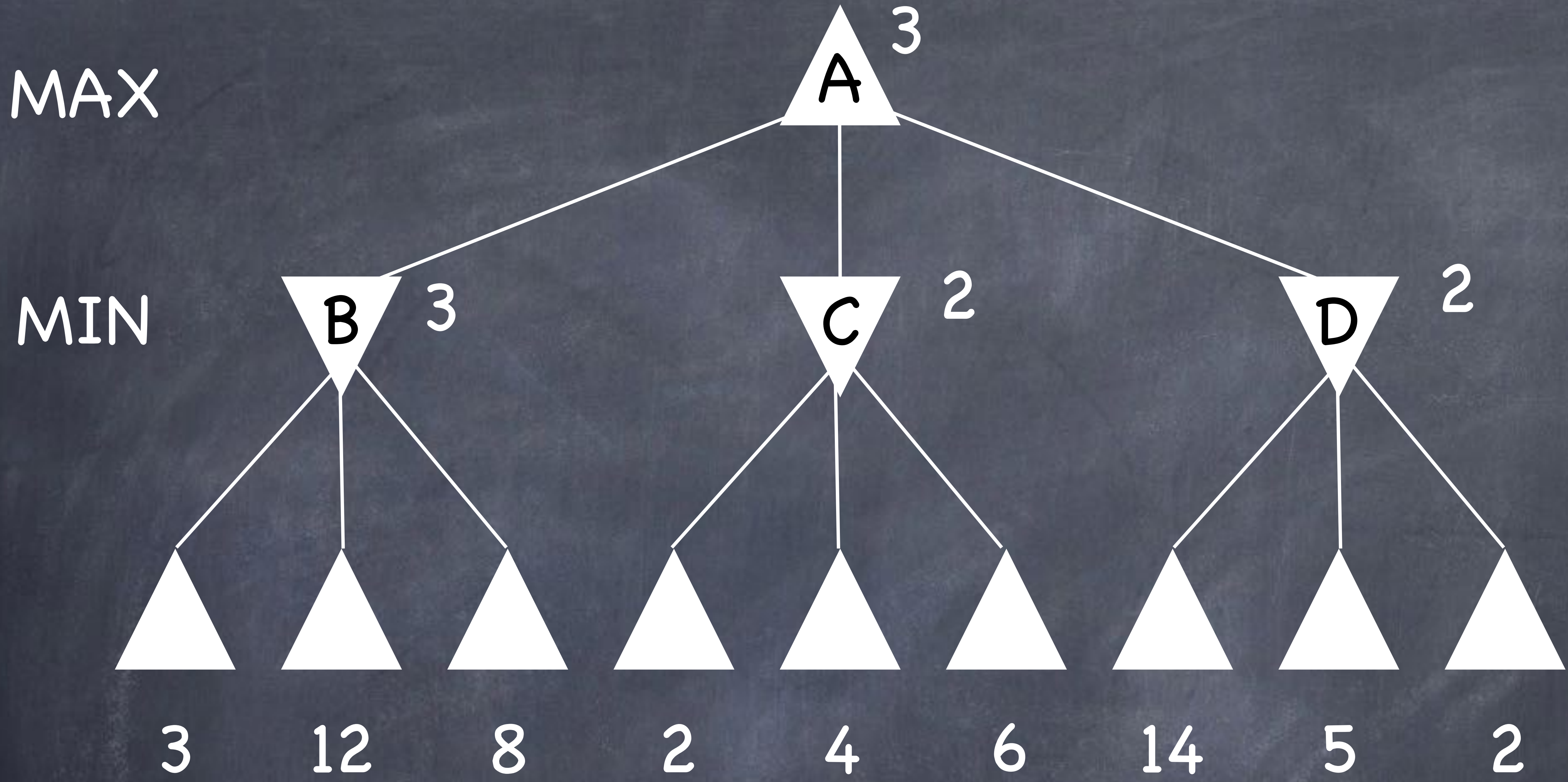
Hang on...

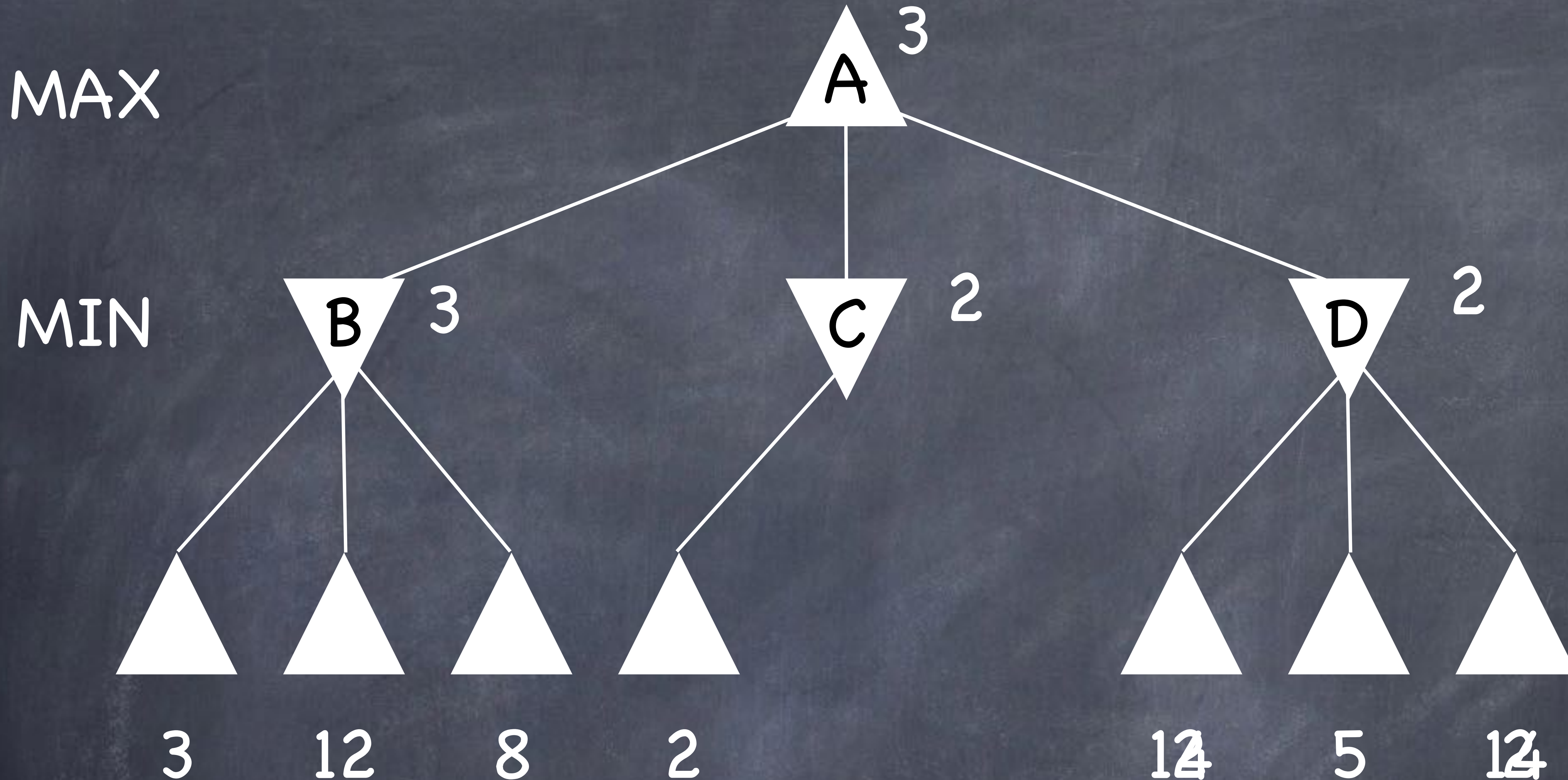
Alpha-Beta Pruning

- Still MINIMAX search
  - Optimal (if you search to the terminals)
  - Optimal with respect to your heuristic function otherwise

DFS with utility function

Alpha-Beta Pruning Analysis


MAX
MIN
A
3
B
3
C
2
D
2
3
12
8
2
4
6
14
5
2


MAX
A
3
MIN
B
3
C
2
D
2
3
12
8
2
14
5
12

Alpha-Beta Pruning Analysis

Ideal case:

Explore the best successor first: $O(b^{m/2})$

Branching factor: $b^{1/2} = \sqrt{b}$

Explore twice as deep a tree in same time

Alpha-Beta Pruning Analysis

Random case:

Explore successors in random order: $O(b^{\frac{3}{4}m})$

Branching factor: $b^{3/4}$

Explore 1/3 deeper tree in same time

Alpha-Beta Pruning Analysis

"Smart" case:

- e.g., for chess order by:
  - Captures
  - Threats of captures
  - Forward moves
  - Backward moves

Alpha-Beta Summary

- Easy bookkeeping modification of basic MINIMAX algorithm
- Not hard to come up with "useful" node orderings
- Even random gets you 33% deeper search
- Works with other ways of improving game tree search

Types of Games

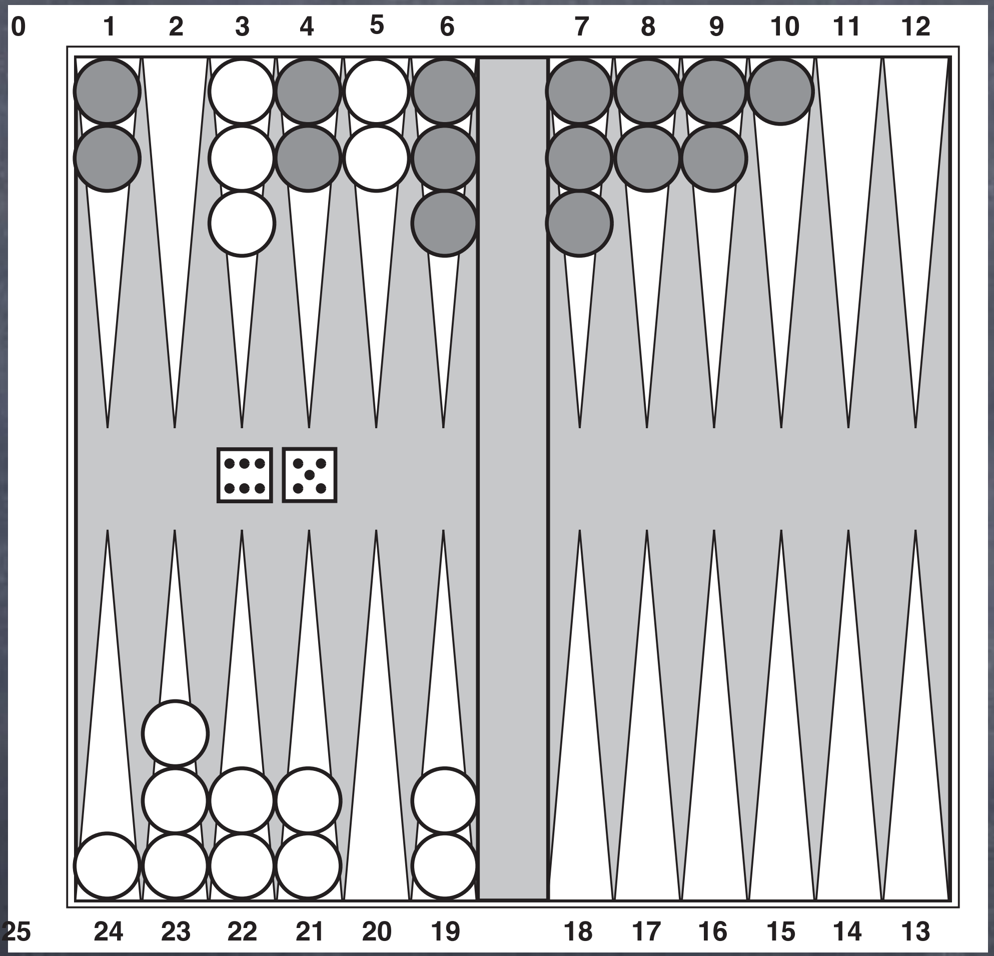
Deterministic (no chance)	Nondeterministic (dice, cards, etc.)
Perfect information (fully observable)	Imperfect information (partially observable)
Zero-sum (total payoff the same in any game)	Arbitrary utility functions

Stochastic and Partially Observable Games

Non-deterministic (Stochastic) Games

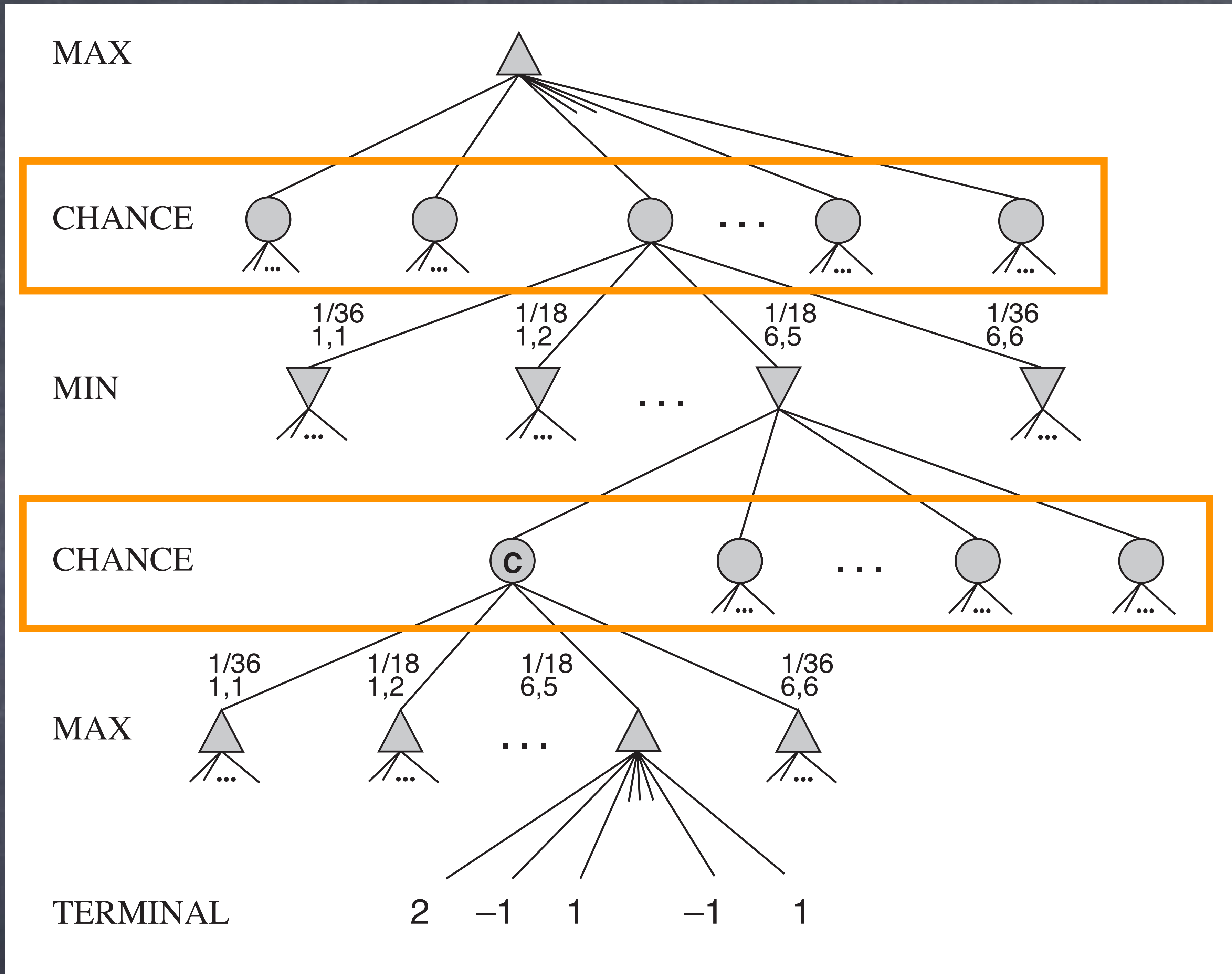
- A player's possible moves depend on chance (random) elements, e.g., dice

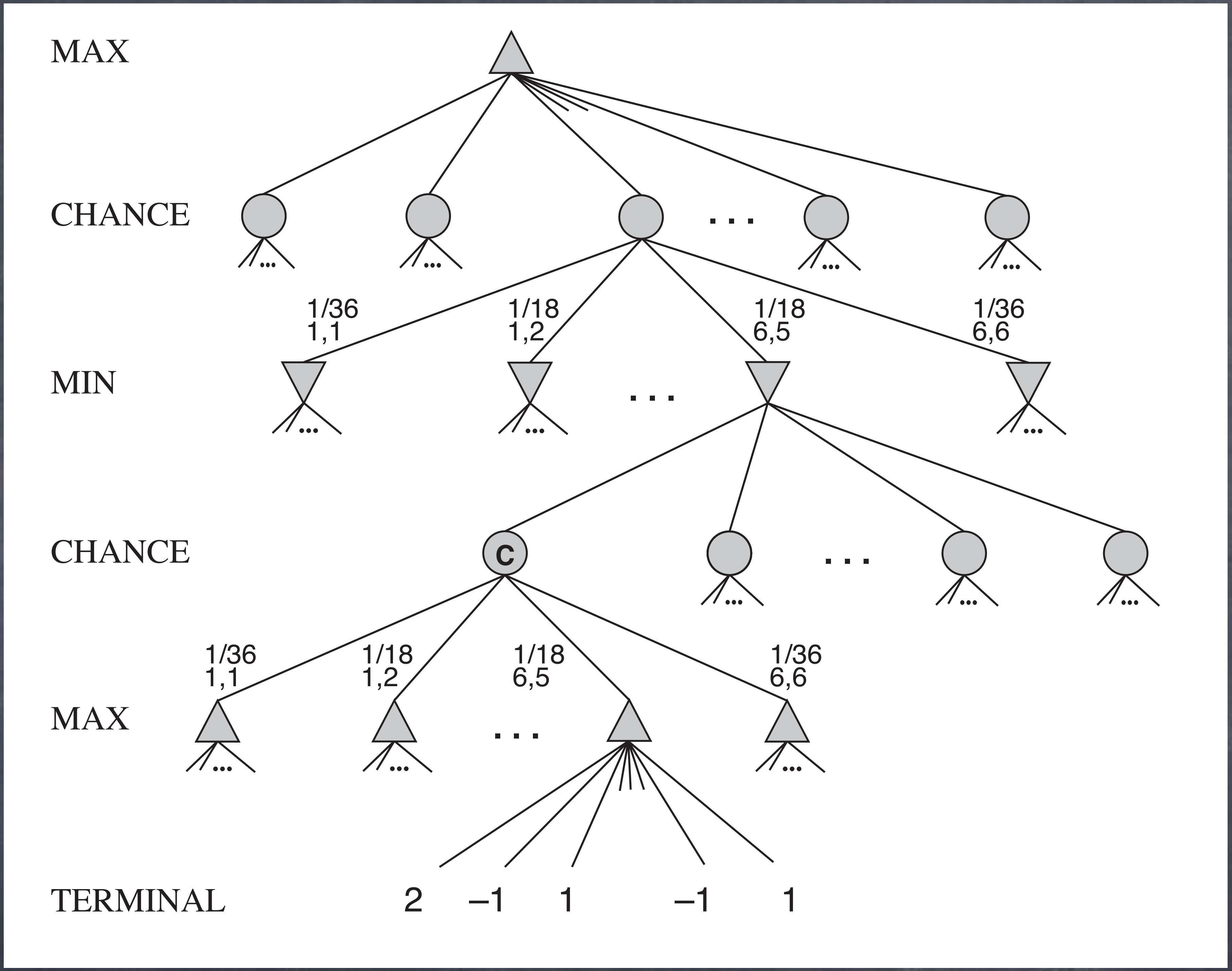
Stochastic Games



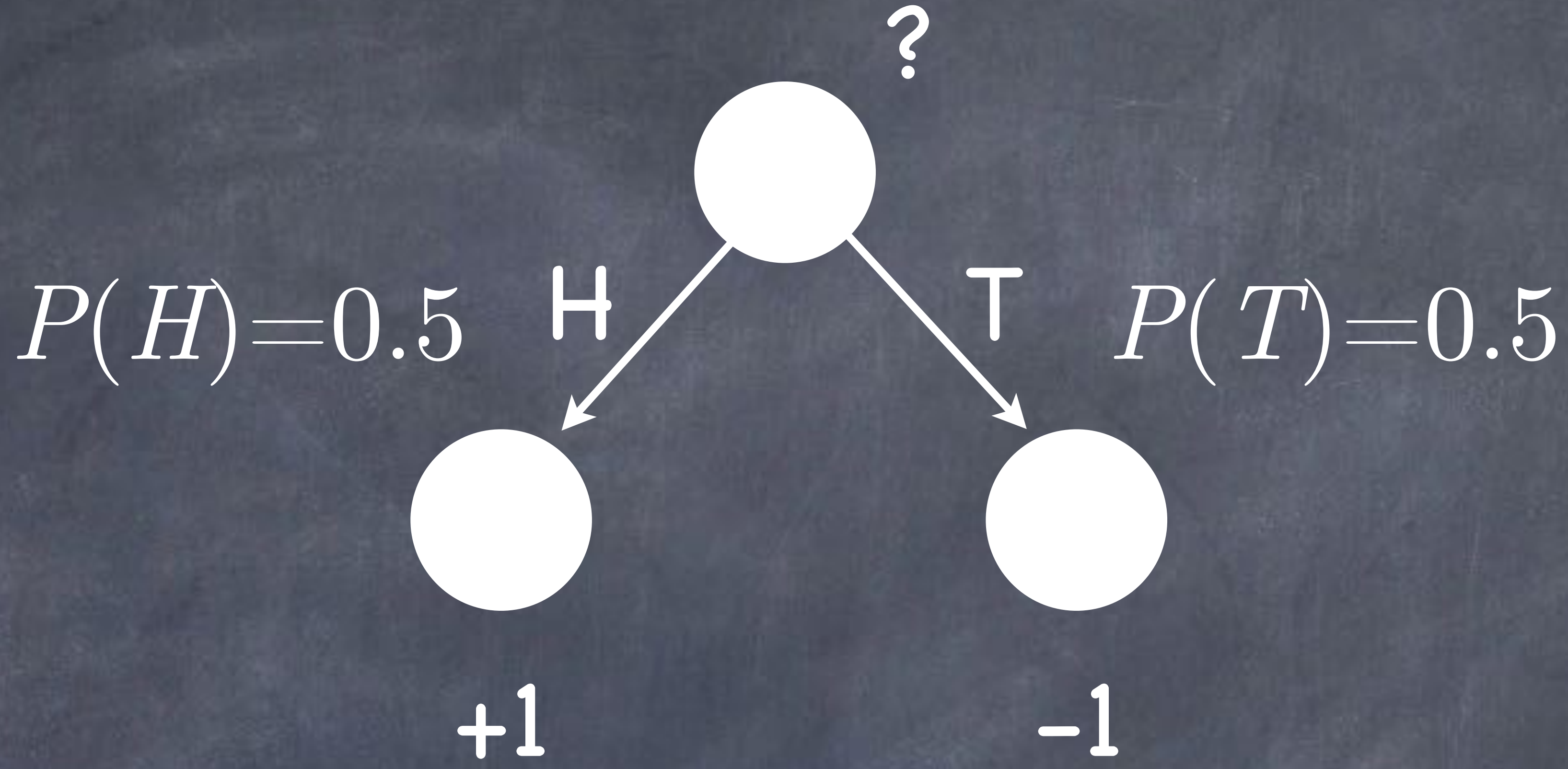
Stochastic Games

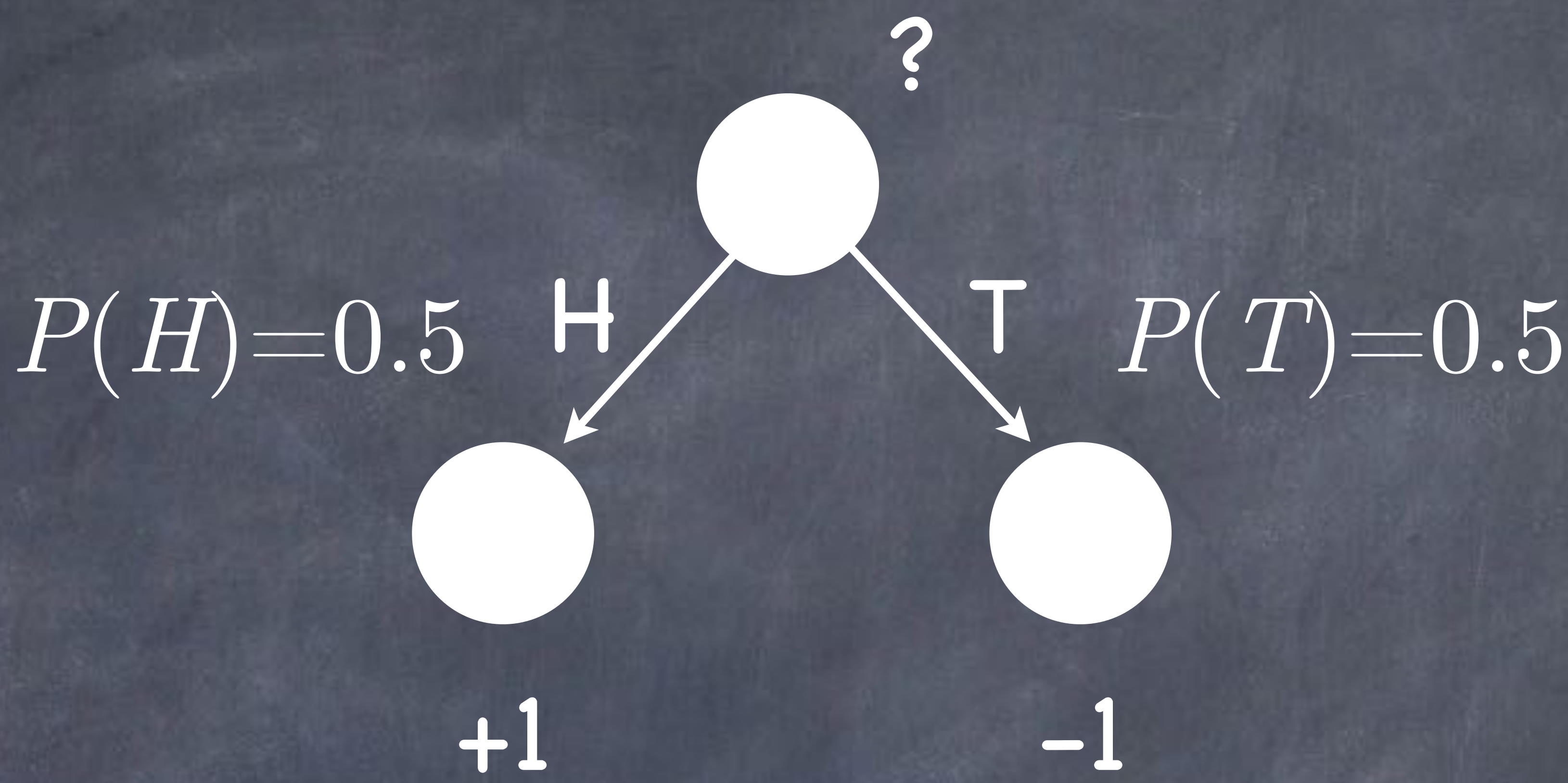
- A player's possible moves depend on chance (random) elements, e.g., dice
- Can't build game tree since don't know what future legal moves will be


MAX
CHANCE
1/36
1,1
1/18
1,2
1/18
6,5
1/36
6,6
MIN
CHANCE
C
1/36
1,1
1/18
1,2
1/18
6,5
1/36
6,6
MAX
TERMINAL
2 –1 1 –1 1

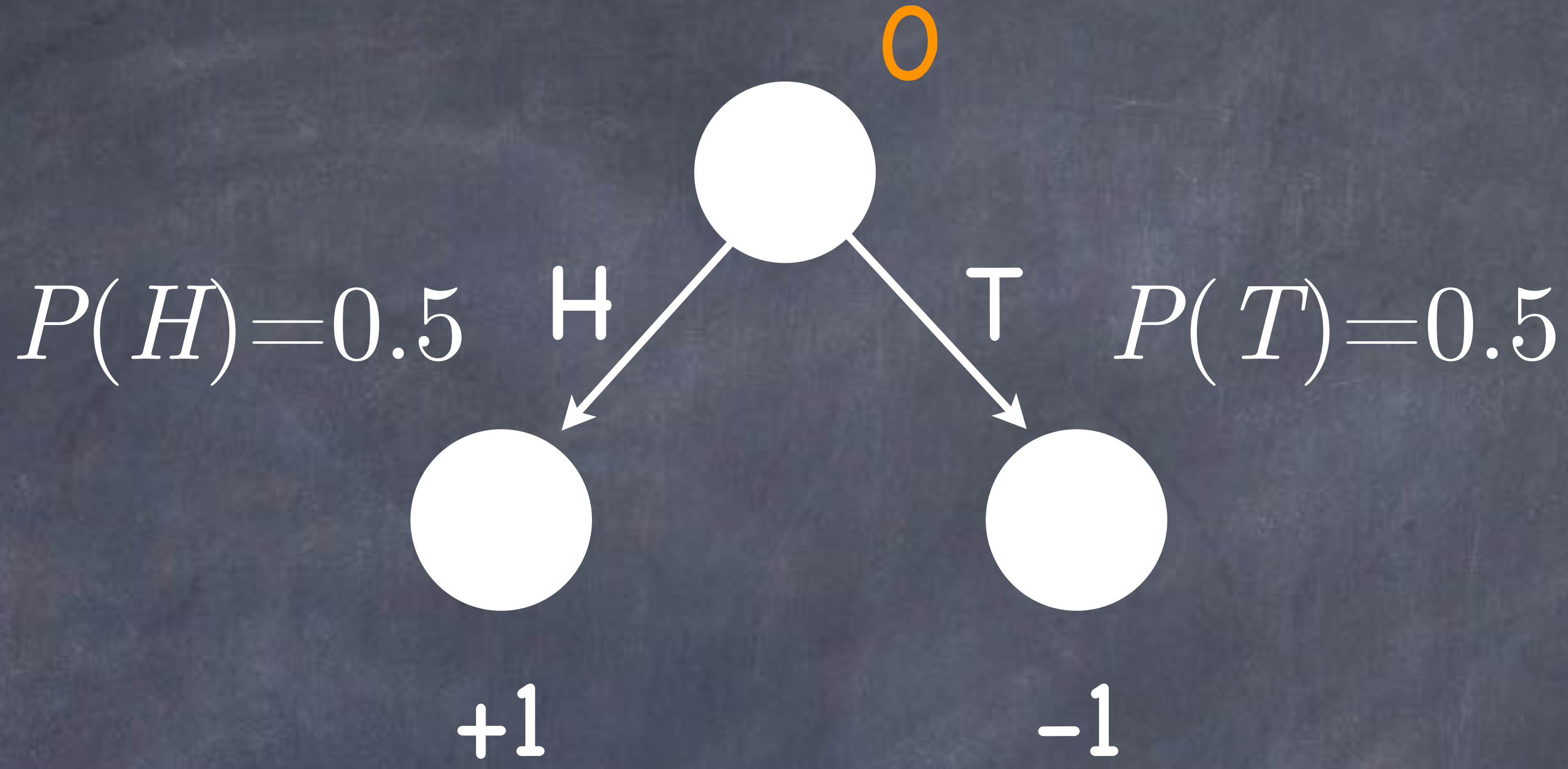

MAX
CHANCE
MIN
CHANCE
MAX
TERMINAL
1/36
1,1
1/18
1,2
1/18
6,5
1/36
6,6
C
2 –1 1 –1 1

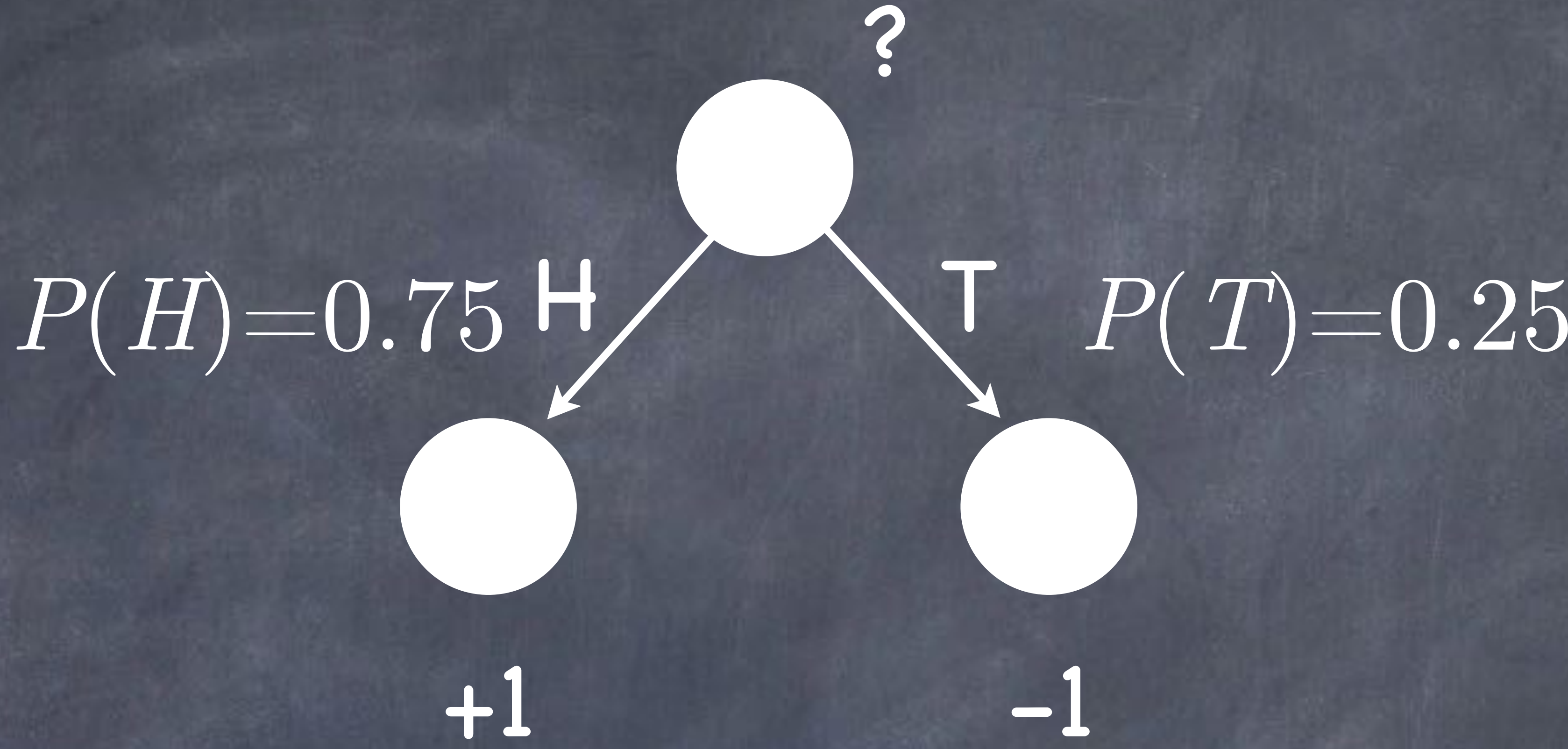
MINIMAX?

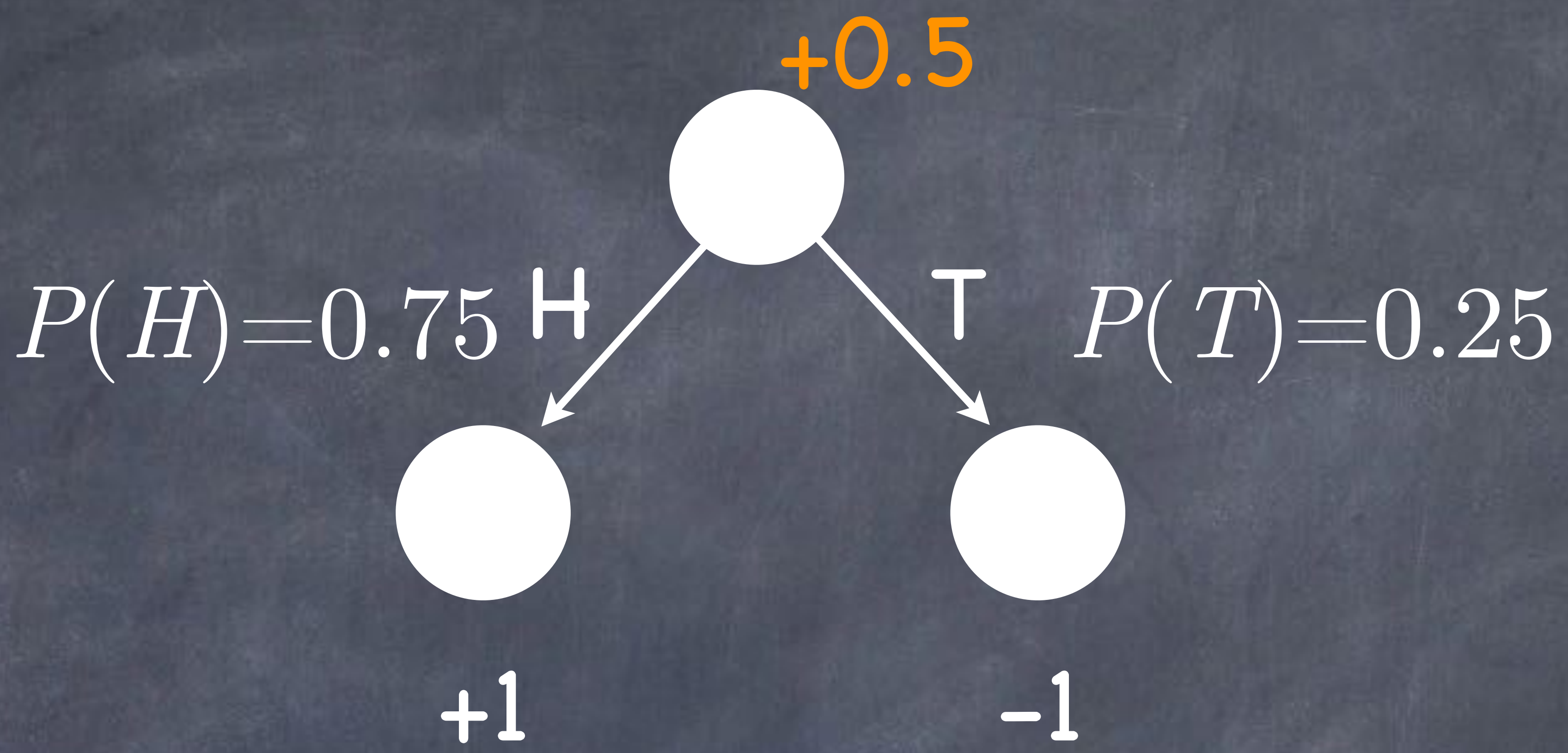

?
H
T
$P(H) = 0.5$
$P(T) = 0.5$
+1
-1



Average over possible outcomes


0
H
T
$P(H) = 0.5$
$P(T) = 0.5$
+1
-1


?
$P(H) = 0.75$
H
T
$P(T) = 0.25$
+1
-1



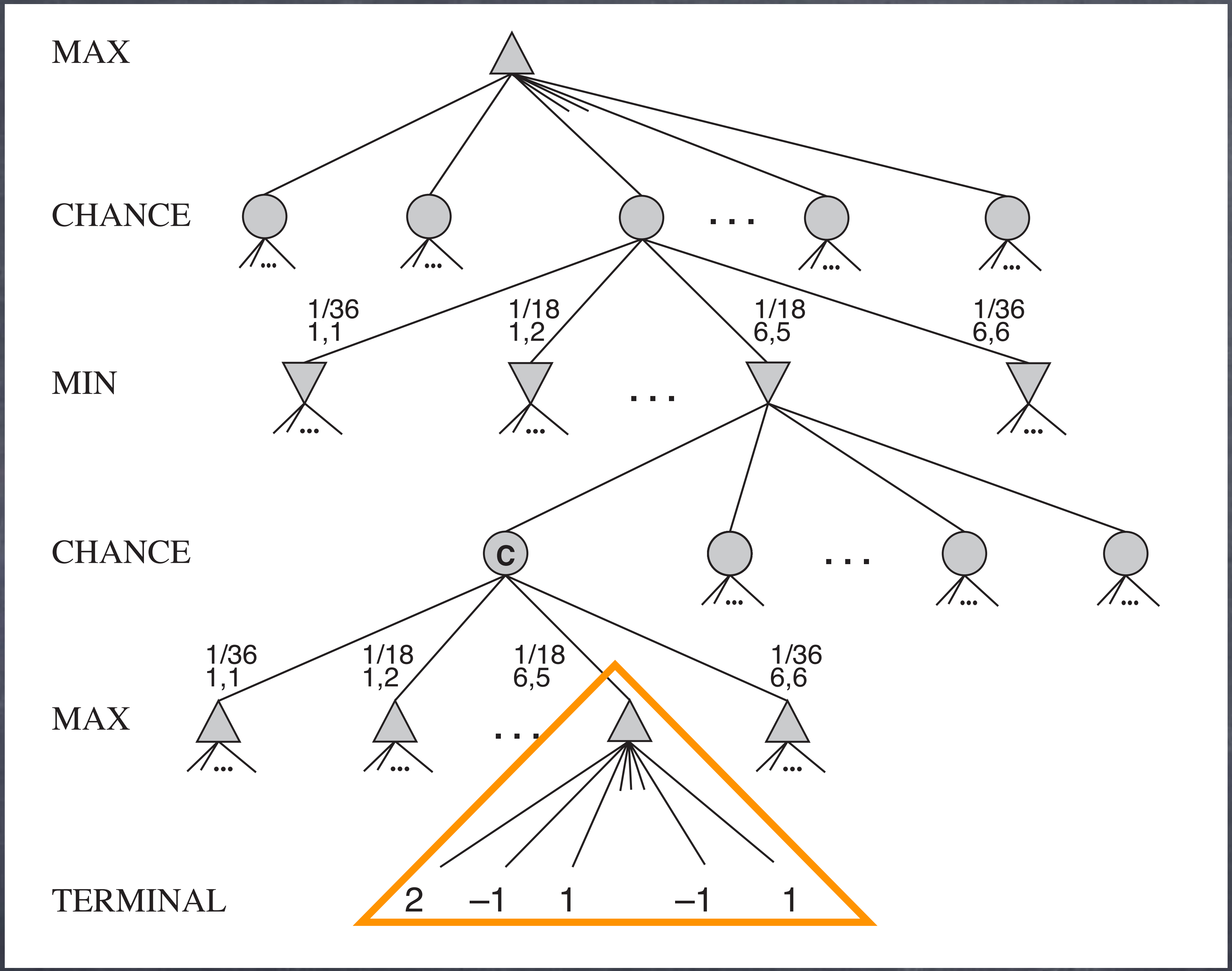
+0.5
$P(H)=0.75$
H
T
$P(T)=0.25$
+1
-1

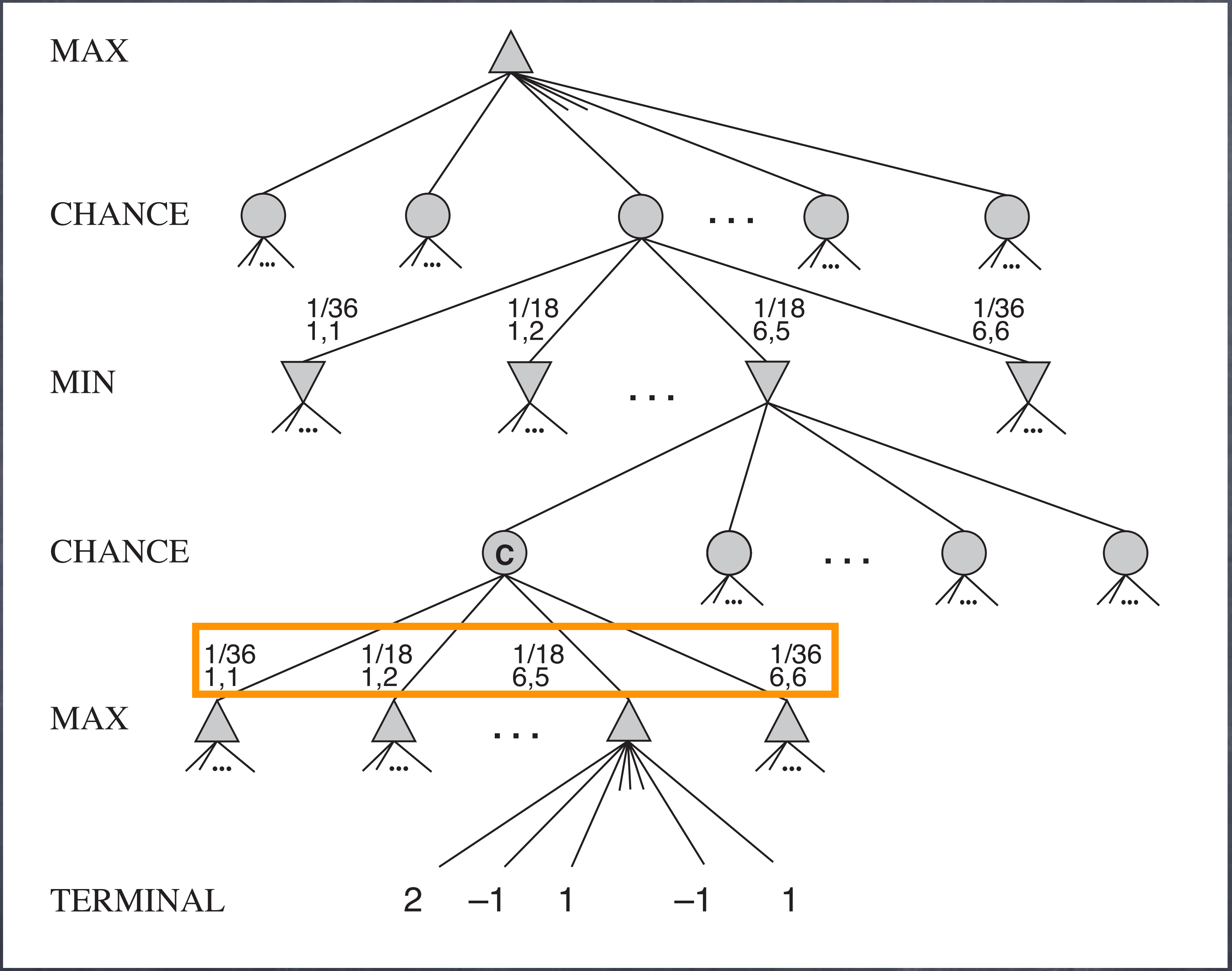
Expectation (Expected Value)

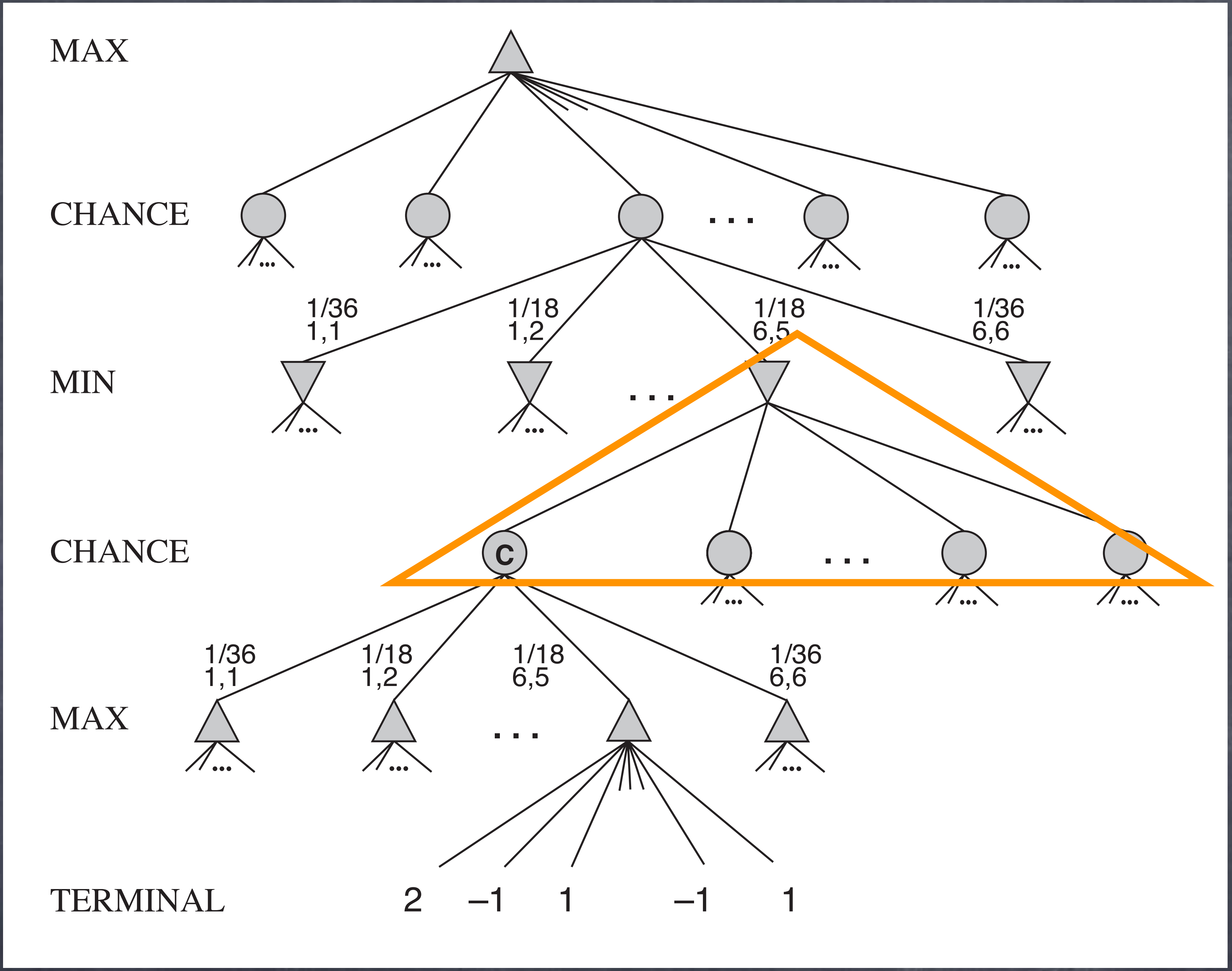
- Weighted average of possibilities
- Sum of the possible outcomes weighted by the likelihood of their occurrence
- What you would expect to win in the long run

Expecti-Minimax

- Same as MINIMAX for MIN and MAX nodes
- Same backing up utilities from terminal nodes
- Take expectation over chance nodes
    - Weighted average of possible outcomes


MAX
CHANCE
MIN
CHANCE
MAX
TERMINAL
1/36
1,1
1/18
1,2
1/18
6,5
1/36
6,6
C
1/36
1,1
1/18
1,2
1/18
6,5
1/36
6,6
2
–1
1
–1
1

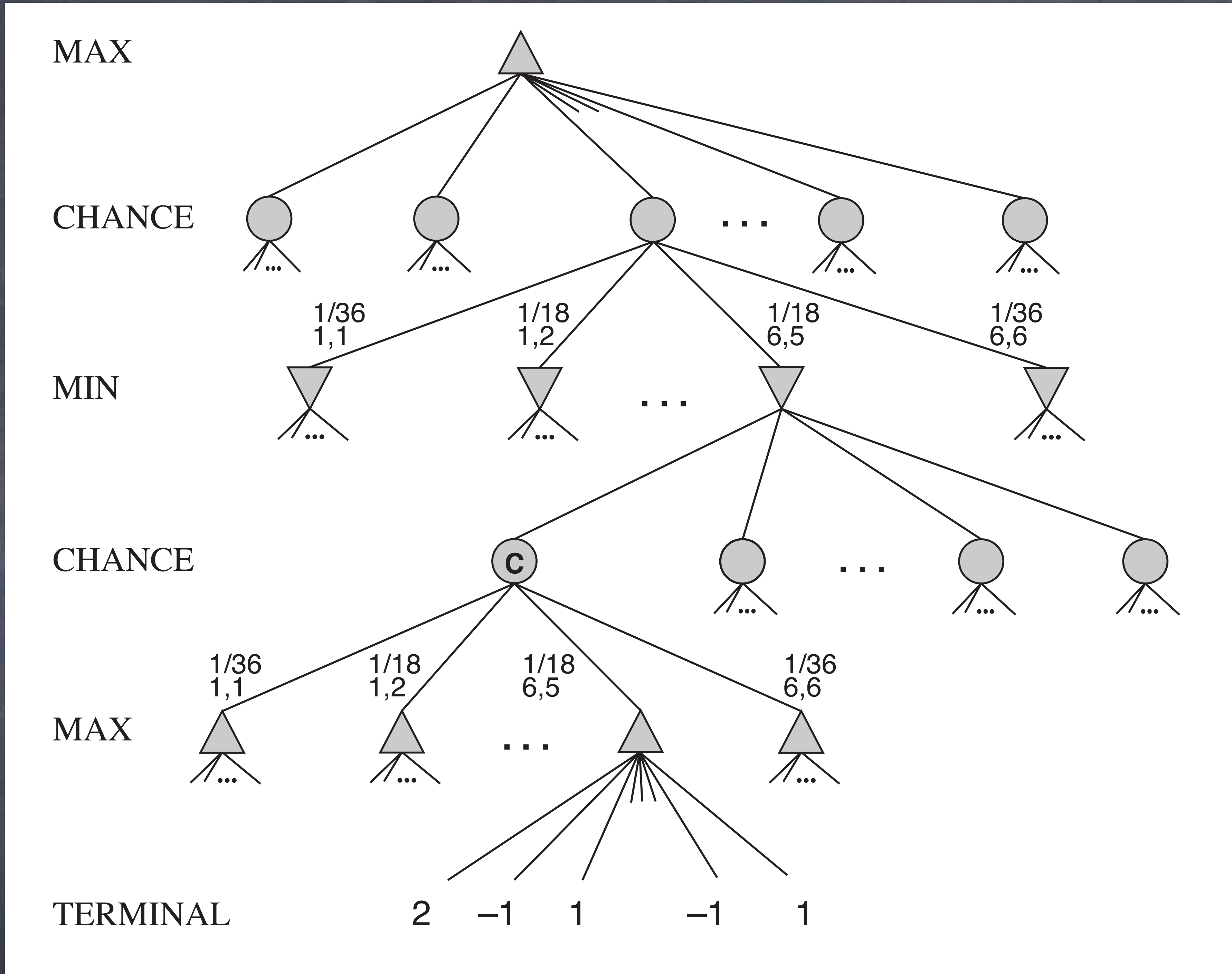

MAX
CHANCE
1/36 1,1
1/18 1,2
1/18 6,5
1/36 6,6
MIN
CHANCE
C
1/36 1,1
1/18 1,2
1/18 6,5
1/36 6,6
MAX
TERMINAL
2 –1 1 –1 1


MAX
CHANCE
MIN
CHANCE
MAX
TERMINAL
1/36
1,1
1/18
1,2
1/18
6,5
1/36
6,6
C
1/36
1,1
1/18
1,2
1/18
6,5
1/36
6,6
2
–1
1
–1
1

Expecti-Minimax

- Weighted average over chance nodes

$$\textsc{EMinimax}(s) = \begin{cases} \textsc{Utility}(s) & \text{if } \textsc{Terminal-Test}(s) \\ \max_a \textsc{EMinimax}(\textsc{Result}(S,a)) & \text{if } \textsc{Player}(s) = \textsc{max} \\ \min_a \textsc{EMinimax}(\textsc{Result}(S,a)) & \text{if } \textsc{Player}(s) = \textsc{min} \\ \sum_r P(r)\textsc{EMinimax}(\textsc{Result}(S,r)) & \text{if } \textsc{Player}(s) = \textsc{chance} \end{cases}$$


MAX
CHANCE
1/36
1,1
1/18
1,2
1/18
6,5
1/36
6,6
MIN
CHANCE
C
1/36
1,1
1/18
1,2
1/18
6,5
1/36
6,6
MAX
TERMINAL
2
–1
1
–1
1

Expecti-Minimax

- Weighted average over chance nodes

$$O(b^m n^m)$$

Stochastic Games

- Expectation to handle uncertainty and randomness
- For example in poker: “Pot Odds”

Partial Observability

- Some of the state of the world is hidden (unobservable)
- There is some uncertainty about the state of the world

Partially-Observable Games

- Some of the state of the game is hidden from the player(s)
- Interesting because:
  - Valuable real-world games (e.g., cards)
  - Partial observability arises all the time in real-world problems

Partially-Observable Games

- Deterministic partial observability
  - Opponent has hidden state
  - Battleship, Stratego, Kriegspiel

Partially-Observable Games

- Deterministic partial observability
  - Opponent has hidden state
  - Battleship, Stratego, Kriegspiel
- Stochastic partial observability
  - Missing/hidden information is random
  - Card games: bridge, hearts, poker (most)

Stochastic Partially Observable Games

Hand	Frequency	Approx. Probability	Approx. Cumulative	Approx. Odds	Mathematical expression of absolute frequency
Royal flush 	4	0.000154%	0.000154%	649,739 : 1	$\binom{4}{1}$
Straight flush (excluding royal flush) 	36	0.00139%	0.00154%	72,192.33 : 1	$\binom{10}{1}\binom{4}{1}-\binom{4}{1}$
Four of a kind 	624	0.0240%	0.0256%	4,164 : 1	$\binom{13}{1}\binom{12}{1}\binom{4}{1}$
Full house 	3,744	0.144%	0.170%	693.2 : 1	$\binom{13}{1}\binom{4}{3}\binom{12}{1}\binom{4}{2}$
Flush (excluding royal flush and straight flush) 	5,108	0.197%	0.367%	507.8 : 1	$\binom{13}{5}\binom{4}{1}-\binom{10}{1}\binom{4}{1}$
Straight (excluding royal flush and straight flush) 	10,200	0.392%	0.76%	253.8 : 1	$\binom{10}{1}\binom{4}{1}^5-\binom{10}{1}\binom{4}{1}$
Three of a kind 	54,912	2.11%	2.87%	46.3 : 1	$\binom{13}{1}\binom{4}{3}\binom{12}{2}\binom{4}{1}^2$
Two pair 	123,552	4.75%	7.62%	20.03 : 1	$\binom{13}{2}\binom{4}{2}^2\binom{11}{1}\binom{4}{1}$
One pair 	1,098,240	42.3%	49.9%	1.36 : 1	$\binom{13}{1}\binom{4}{2}\binom{12}{3}\binom{4}{1}^3$
No pair / High card 	1,302,540	50.1%	100%	.995 : 1	$\left[\binom{13}{5}-10\right]\left[\binom{4}{1}^5-4\right]$
Total	**2,598,960**	**100%**	**100%**	1 : 1	$\binom{52}{5}$

Weighted Minimax

- For each possible deal s:
  - Assume s is the actual situation
  - Compute Minimax or H-Minimax value of s
  - Weight value by probability of s
- Take move that yields highest expected value over all the possible deals

Weighted Minimax

$$\operatorname*{argmax}_a \sum_s P(s)\,\textsc{Minimax}(\textsc{Result}(s,a))$$

Weighted Minimax

$$\operatorname*{argmax}_a \sum_s P(s)\textsc{Minimax}(\textsc{Result}(s,a))$$

$$\binom{26}{13} = 10,400,600$$

$$\binom{47}{25} = 1.48338977 \times 10^{13}$$

Monte Carlo Methods

- Use a “representative” sample to approximate a events from an underlying probability distribution

Monte Carlo Minimax

$$\operatorname*{argmax}_a \frac{1}{N} \sum_{i=1}^{N} \textsc{Minimax}(\textsc{Result}(s_i, a))$$

Summary

- Non-deterministic games
  - Expecti-MINIMAX: Compute expected MINIMAX value over chance nodes
- Partially observable games
  - Weighted MINIMAX: Compute expected value over possible hidden states
- Naive approaches impractical

For Next Time:
Chapter 4.0-4.1