Minimax Algorithm

AI+X
INSTITUTE FOR ARTIFICIAL INTELLIGENCE

Tic-Tac-Toe

Can we use the search algorithms seen so far to decide our next move in a tic-tac-toe match?

Tic-Tac-Toe

Tic-Tac-Toe

Can we use the search algorithms seen so far to decide our next move in a tic-tac-toe match?

We miss the idea that our adversaries are actively trying to defeat us!

Tic-Tac-Toe

Tic-Tac-Toe

Can we use the search algorithms seen so far to decide our next move in a tic-tac-toe match?

We miss the idea that our adversaries are actively trying to defeat us!

"To defeat your enemy, you must become your enemy."

Tic-Tac-Toe

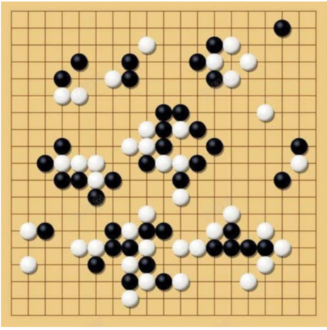
Competitive Environments

- Environments where two or more agents have conflicting goals
- Adversarial search problems

Tic-Tac-Toe

Two-Player Zero-Sum Games

- Deterministic, two-player, turn-taking, perfect information, zero-sum games
  - “perfect information” → “fully observable”
  - “zero-sum” → what is good for one player is just as bad for the other
- We will concentrate on such games because:
  - the state of a game is easy to represent
  - agents are usually restricted to a small number of actions whose effects are defined by precise rules

Two-Player Zero-Sum Games

- 1952: Arthur Samuel’s checkers playing program

Figure 1. Playing checkers on the 701

1952

Two-Player Zero-Sum Games

- 1994: Chinook ended 40-year-reign of human champion Marion Tinsley using complete 8-piece endgame

Figure 2. Chinook team (August 1992). From left to right: Duane Szafron, Joe Culberson, Paul Lu, Brent Knight, Jonathan Schaeffer, Rob Lake, and Steve Sutphen. Our checkers expert, Norman Treloar, is missing.

1952

1994

Two-Player Zero-Sum Games

- 1997: IBM's Deep Blue beats world chess champion

Figure 3. Garry Kasparov in a 1997 game against Deep Blue.

1997

1952

1994

Two-Player Zero-Sum Games

- 2007: Chinook – the unbeatable checkers player

Figure 4. Chinook team (August 2012). From left to right Steve Sutphen, Duane Szafron, Jonathan Schaeffer, Rob Lake, and Paul Lu.

1997

1952 1994 2007 2017

Two-Player Zero-Sum Games

- 2017: Google DeepMind's AlphaGo defeats Go champions

Figure 5. World champion Ke Jie struggles against AlphaGo, a product of Alphabet's (formerly Google's) subsidiary DeepMind.

1997

1952 1994 2007 2017

Adversarial Search

- For you to play optimally, your opponent must play optimally

Figure 5. World champion Ke Jie struggles against AlphaGo, a product of Alphabet's (formerly Google's) subsidiary DeepMind.

Adversarial Search

- Maximize the utility of your moves, minimize the utility of your opponent's moves
    - Utility function gives a numeric value for a terminal state
    - Example: win, loss, or draw, with values 1, -1, or 0

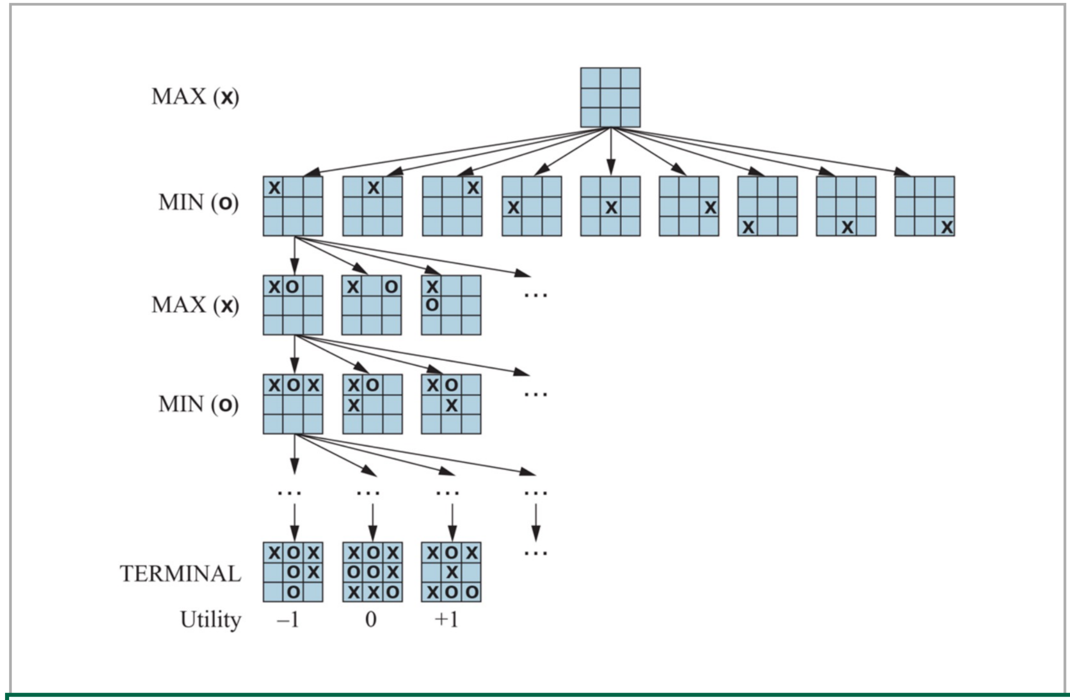


Figure 6. A (partial) game tree for the game of tic-tac-toe.

The Minimax Algorithm

- *minimax*(**s**) =
  - *utility*(**s**) for the MAX player if **s** is terminal
  - $\max_{a \in Actions(s)}$ *minimax*(*move*(s,a)) if MAX is playing
  - $\min_{a \in Actions(s)}$ *minimax*(*move*(s,a)) if MIN is playing

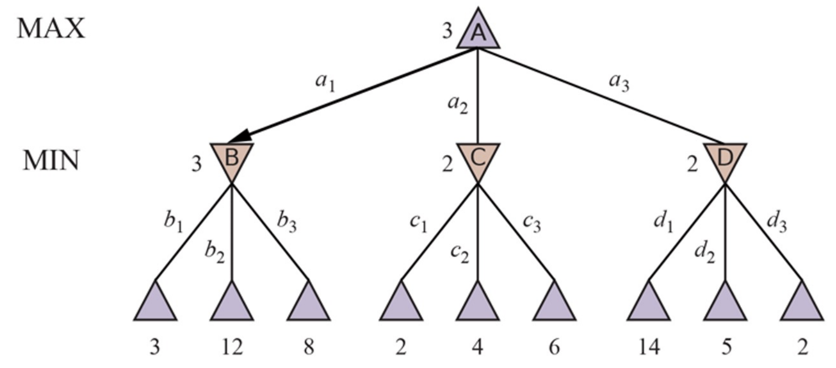


Figure 7. Minimax search tree.

The Minimax Algorithm - Pseudocode

```
(action) minimax(s):
    a, val = max_player(s)
    return a

(action,utility) max_player(s):
    if s is terminal:
        return null, utility_function(s)
    max_a, max_val = null, -∞
    for a in actions:
        b, val = min_player(move(s,a))
        if val > max_val:
            max_a, max_val = a, val
    return max_a, max_val
```

```
(action,utility) min_player(s):
    if s is terminal:
        return null, utility_function(s)
    min_a, min_val = null, ∞
    for a in actions:
        b, val = max_player(move(s,a))
        if val < min_val:
            min_a, min_val = a, val
    return min_a, min_val

(utility) utility_function(s):
    return utility of state s for the max_player

(state) move(s,a):
    return state reached from s after taking action a
```

Knowledge Check 1

What will be the values of the nodes A, B, C, D, E, F, and G in the minimax search tree on the left?

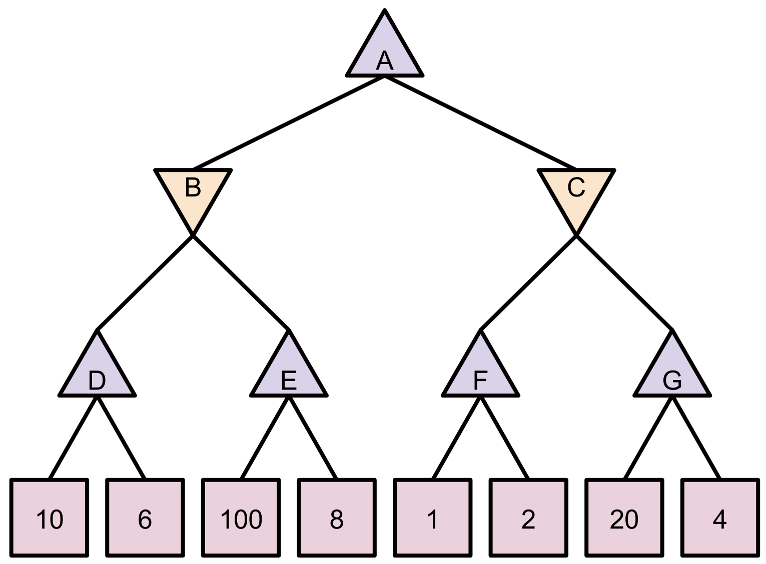


Figure 8. Minimax search tree.

- A 4, 8, 4, 6, 8, 1, 4
- B 100, 100, 20, 10, 100, 2, 20
- C 10, 10, 2, 10, 100, 2, 20
- D 6, 6, 1, 6, 8, 1, 4

Minimax Efficiency

- How efficient is minimax?
  - Just like (exhaustive) DFS
  - Time: $O(b^m)$
  - Space: $O(bm)$
- Example: For chess, $b \approx 35$, $m \approx 100$
  - Exact solution is completely infeasible
  - But, do we need to explore the whole tree?

Alpha-Beta Pruning

- The number of game states is exponential in the depth of the tree
- No algorithm can completely eliminate the exponent, but we can sometimes cut it in half by pruning large parts of the tree that make no difference to the outcome
- The particular technique we will examine is called alpha–beta pruning

Alpha-Beta Pruning

- Alpha–beta pruning gets its name from the two extra parameters that describe bounds along the search path:
  - α = the value of the best choice we have found so far along the path for MAX (α = “at least”)
  - β = the value of the best choice we have found so far along the path for MIN (β = “at most”)

Alpha-Beta Pruning

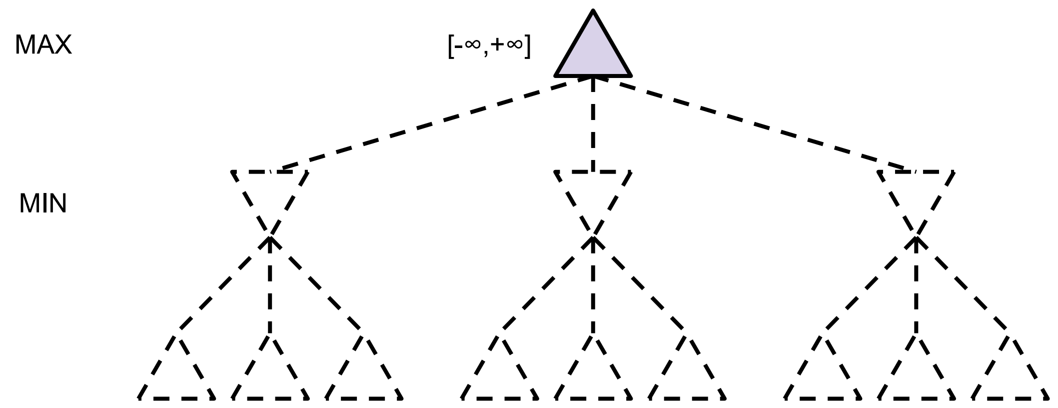


Figure 9. Alpha-Beta Pruning example.

Alpha-Beta Pruning

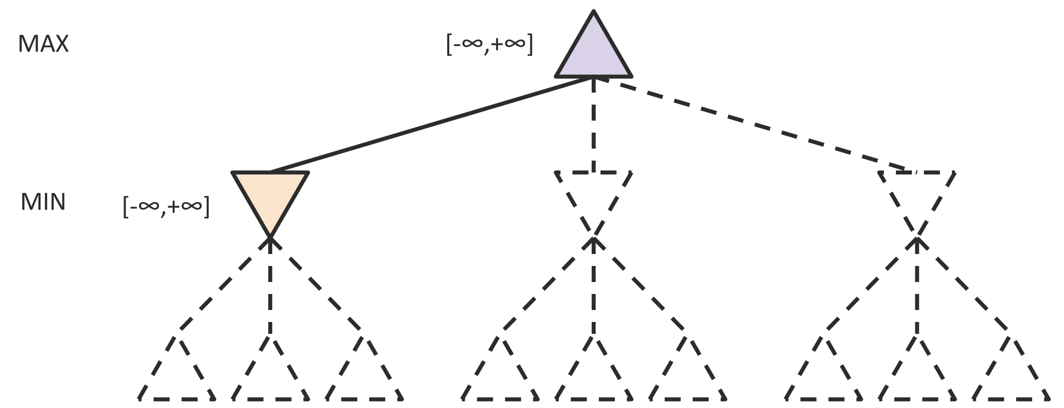


Figure 9. Alpha-Beta Pruning example.

Alpha-Beta Pruning

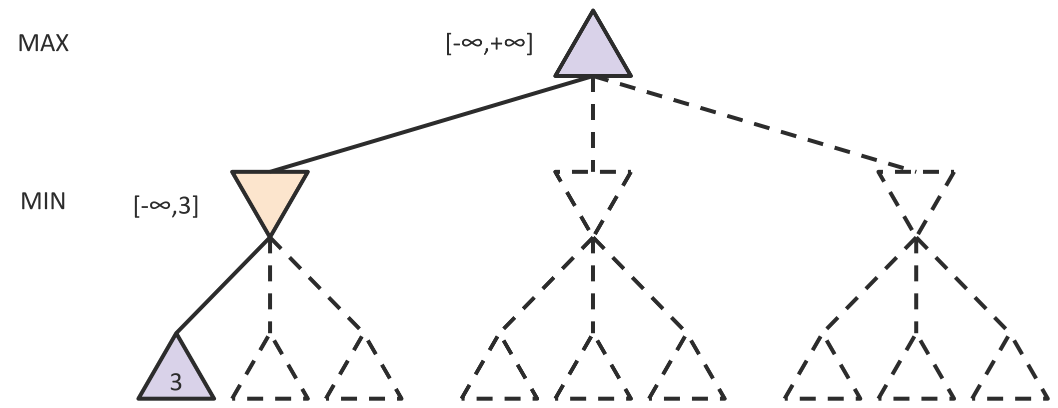


Figure 9. Alpha-Beta Pruning example.

Alpha-Beta Pruning

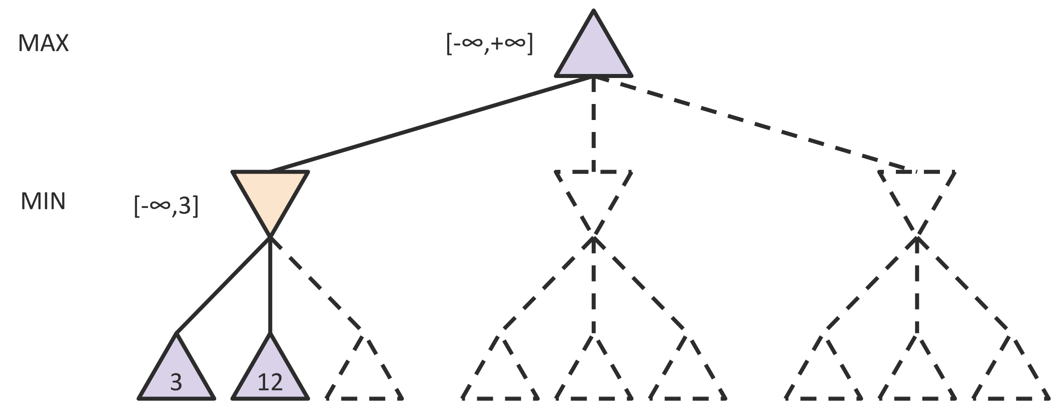


Figure 9. Alpha-Beta Pruning example.

Alpha-Beta Pruning

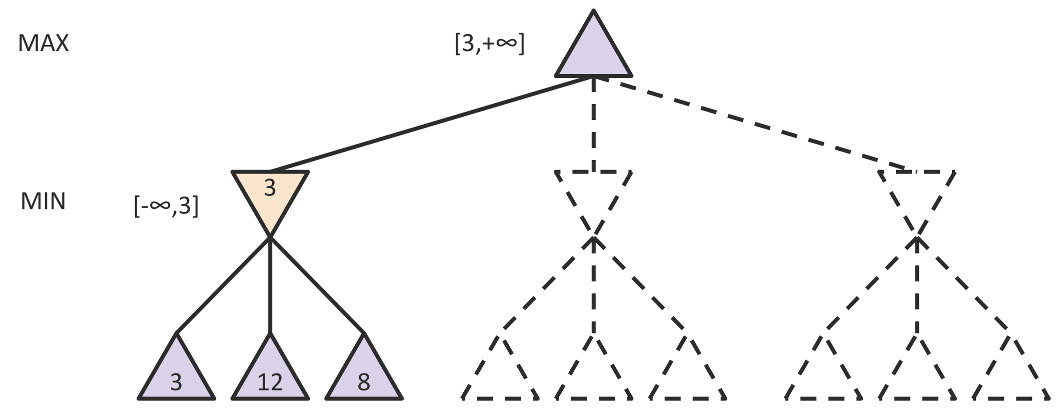


Figure 9. Alpha-Beta Pruning example.

Alpha-Beta Pruning

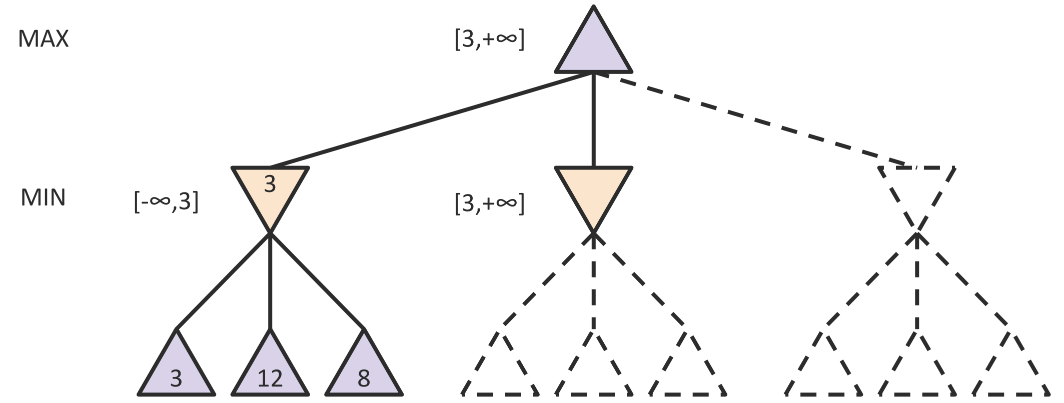


Figure 9. Alpha-Beta Pruning example.

Alpha-Beta Pruning

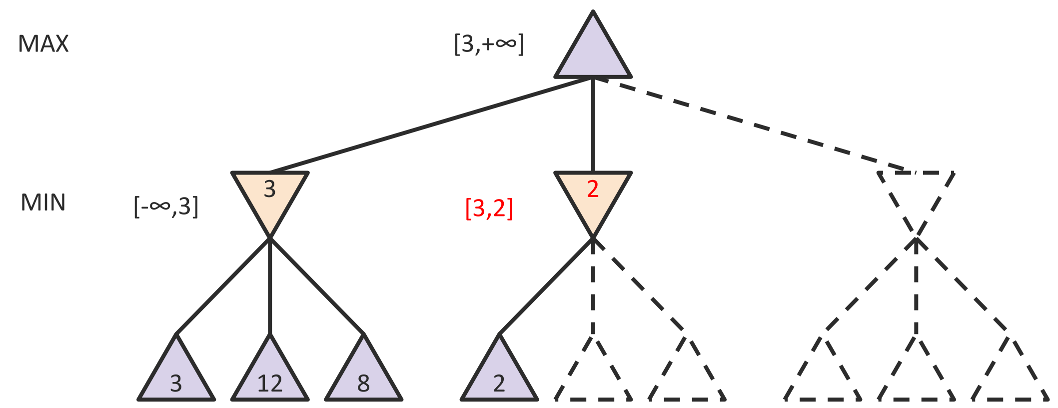


Figure 9. Alpha-Beta Pruning example.

Alpha-Beta Pruning

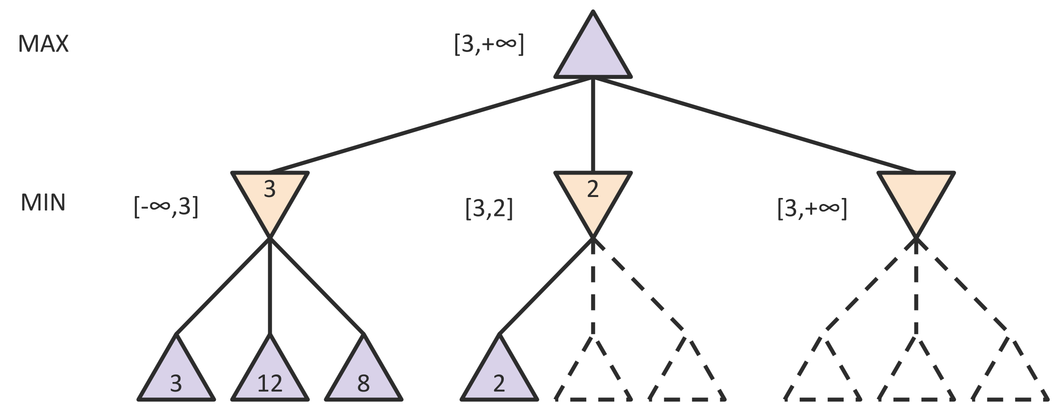


Figure 9. Alpha-Beta Pruning example.

Alpha-Beta Pruning

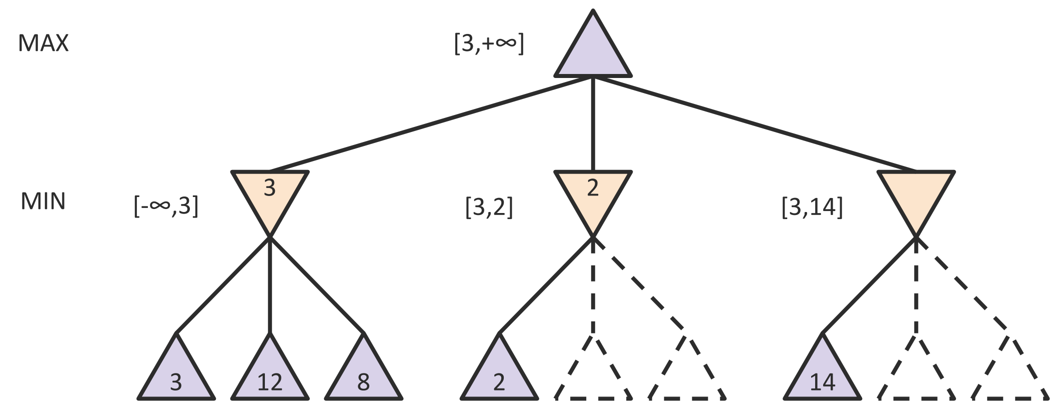


Figure 9. Alpha-Beta Pruning example.

Alpha-Beta Pruning

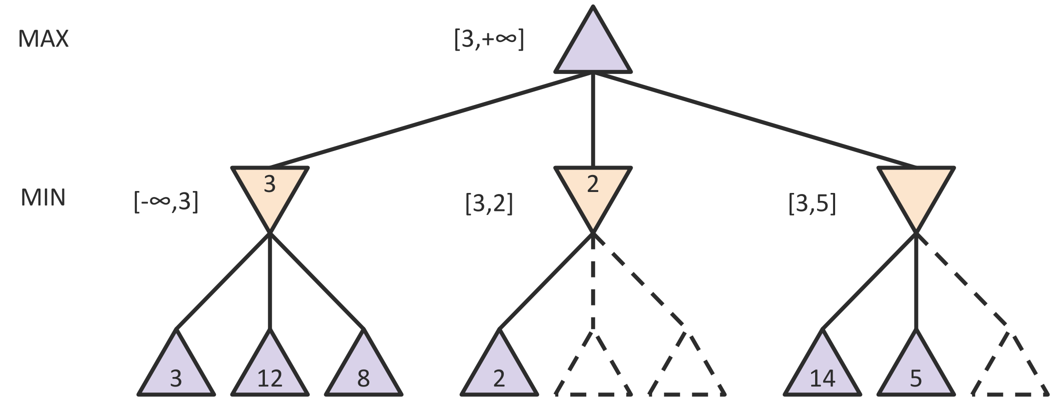


Figure 9. Alpha-Beta Pruning example.

Alpha-Beta Pruning

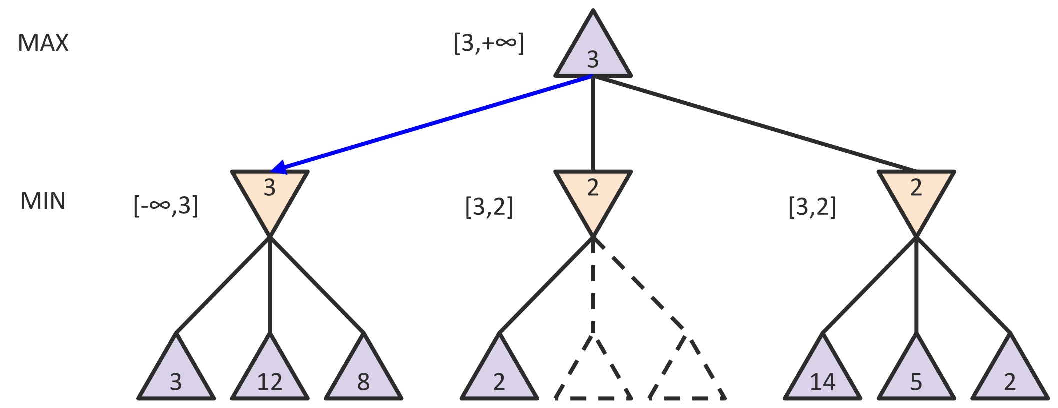


Figure 9. Alpha-Beta Pruning example.

Alpha-Beta Pruning - Pseudocode

```
(action) alpha_beta(s):
    a, val = max_player(s,-∞,+∞)
    return a

(action,utility) max_player(s,α,β):
    if s is terminal:
        return null, utility_function(s)
    max_a, max_val = null, -∞
    for a in actions:
        b, val =
min_player(move(s,a),α,β)
        if val > max_val:
            max_a, max_val = a, val
            α = max(α,val)
        if max_val ⩾ β:
            break
    return max_a, max_val
```

```
(action,utility) min_player(s,α,β):
    if s is terminal:
        return null, utility_function(s)
    min_a, min_val = null, ∞
    for a in actions:
        b, val = max_player(move(s,a),α,β)
        if val < min_val:
            min_a, min_val = a, val
            β = min(β,val)
        if min_val ⩽ α:
            break
    return min_a, min_val
```

Knowledge Check 2

Which of the internal nodes A-G and terminal nodes (a)-(h) will not be expanded when using Alpha-Beta pruning?

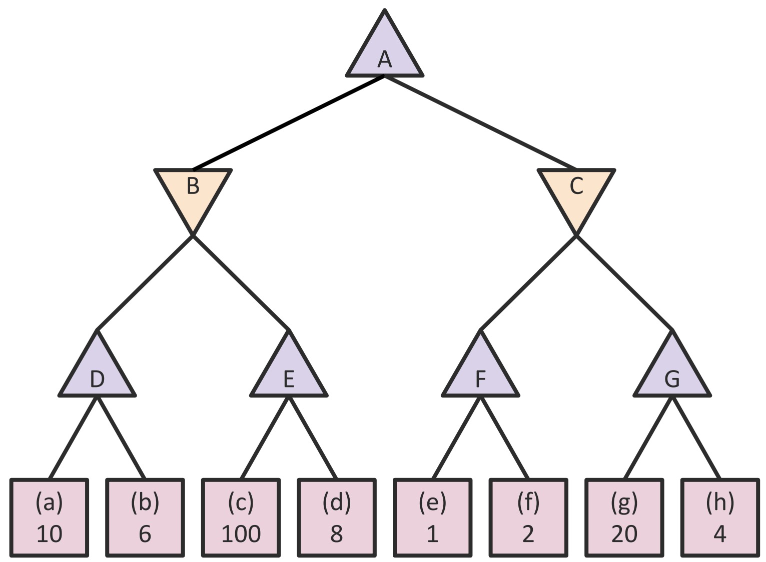


Figure 10. Minimax search tree.

- **A** E, (c), (d), (f), (h)
- **B** E, G, (c), (d), (f), (g), (h)
- **C** G, (d), (g), (h)
- **D** (d), (h)

Alpha-Beta Pruning Properties

- This pruning has no effect on minimax value computed for the root!
- Values of intermediate nodes might be wrong
- Good child ordering improves effectiveness of pruning
- With “perfect ordering”:
  - Time complexity drops to $O(b^{m/2})$
  - Doubles solvable depth!
  - Full search of complex games, e.g. chess, is still hopeless…

You have reached the end of the lecture.

Image/Figure References

Figure 1. Playing checkers on the 701. Source: https://www.ibm.com/ibm/history/ibm100/us/en/icons/ibm700series/impacts/
Figure 2. Chinook team (August 1992). From left to right: Duane Szafron, Joe Culberson, Paul Lu, Brent Knight, Jonathan Schaeffer, Rob Lake, and Steve Sutphen. Our checkers expert, Norman Treloar, is missing. Source: http://jonathanschaeffer.blogspot.com/2012/08/chinook-twenty-years-later.html
Figure 3. Garry Kasparov in a 1997 game against Deep Blue. Source: https://www.businessinsider.com/how-ibm-watson-is-transforming-healthcare-2015-7
Figure 4. Chinook team (August 2012). From left to right Steve Sutphen, Duane Szafron, Jonathan Schaeffer, Rob Lake, and Paul Lu. Source: http://jonathanschaeffer.blogspot.com/2012/08/chinook-twenty-years-later.html
Figure 5. World champion Ke Jie struggles against AlphaGo, a product of Alphabet's (formerly Google's) subsidiary DeepMind. Source: https://www.theneweconomy.com/technology/alphago-defeats-world-go-champion-ke-jie
Figure 6. A (partial) game tree for the game of tic-tac-toe. Source: Russell & Norvig, Artificial Intelligence: A Modern Approach, 4th edition, Pearson, 2021.
Figure 7. Minimax search tree. Source: Russell & Norvig, Artificial Intelligence: A Modern Approach, 4th edition, Pearson, 2021.
Figure 8. Minimax search tree. Source: Russell & Norvig, Artificial Intelligence: A Modern Approach, 4th edition, Pearson, 2021.
Figure 9. Alpha-Beta Pruning example. Source: Russell & Norvig, Artificial Intelligence: A Modern Approach, 4th edition, Pearson, 2021.
Figure 10. Minimax search tree. Source: Russell & Norvig, Artificial Intelligence: A Modern Approach, 4th edition, Pearson, 2021.
Other images were purchased from Getty Images and with permission to use.