上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

# Lecture 6: Adversarial Search

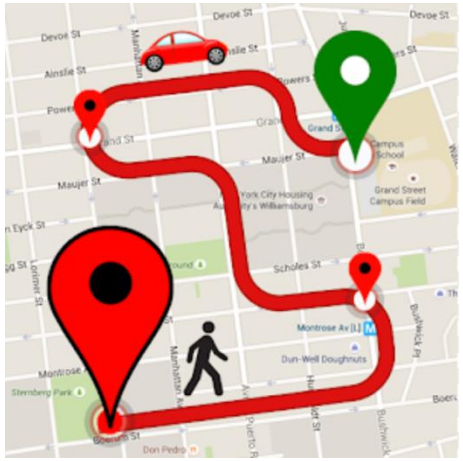
Tao Huang

John Hopcroft Center, School of Computer Science, Shanghai Jiao Tong University

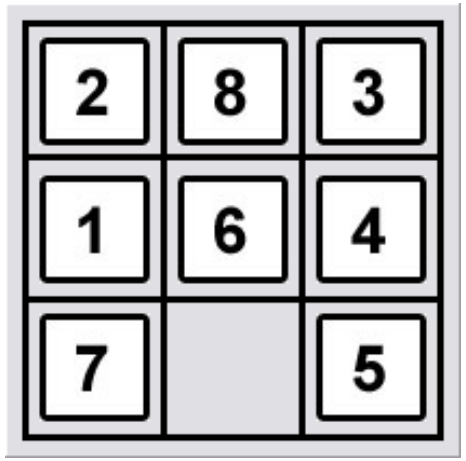
https://taohuang.info/cs3317

https://oc.sjtu.edu.cn/courses/89538

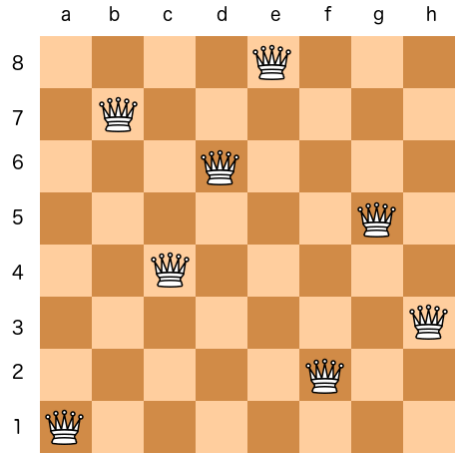
# What's the Difference?



Route Planning

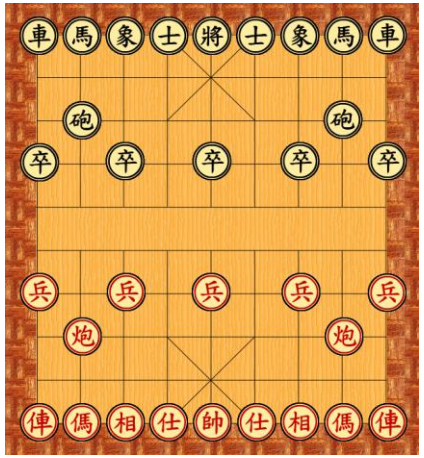



8-Puzzle

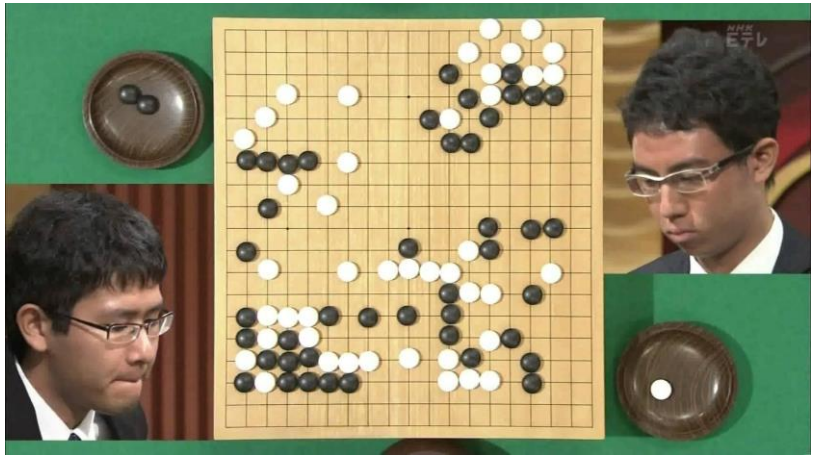



N-Queens

**vs**



Chinese Chess



Go

Robot Battle

# Search with an Opponent

- **Classical search:** Environment is passive.
- **Game search:** Environment contains an intelligent opponent.

Key question:

How should an AI act when another agent is trying to defeat it?

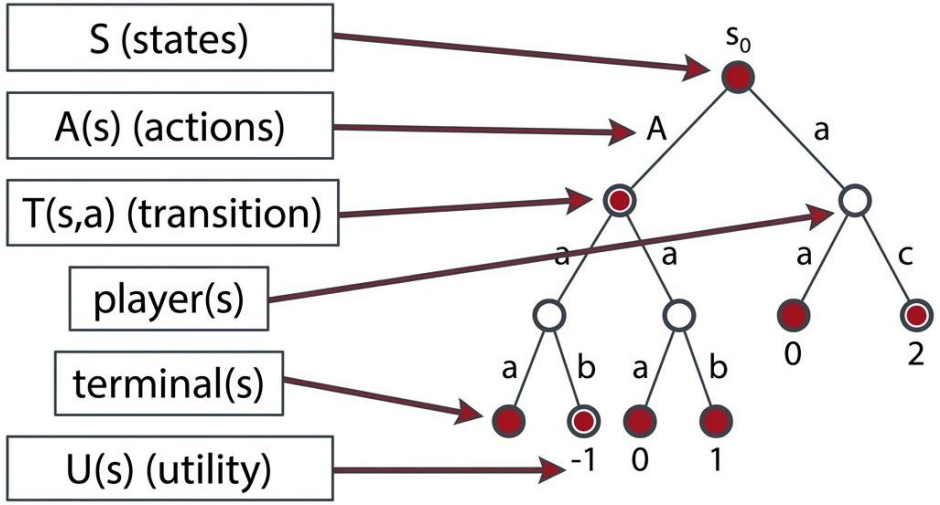
*Examples: Chess, Go, Poker, Strategy games.*

# Formal Definition of a Game

A deterministic two-player game is a tuple:

- S : Set of states (the board configurations)
- A(s) : Legal actions from state s
- T(s, a) : Transition function (result of action a in s)
- player(s) : Whose turn it is (MAX or MIN)
- terminal(s) : Is the game over?
- U(s) : Utility function for terminal states

This formulation defines a Game Tree.

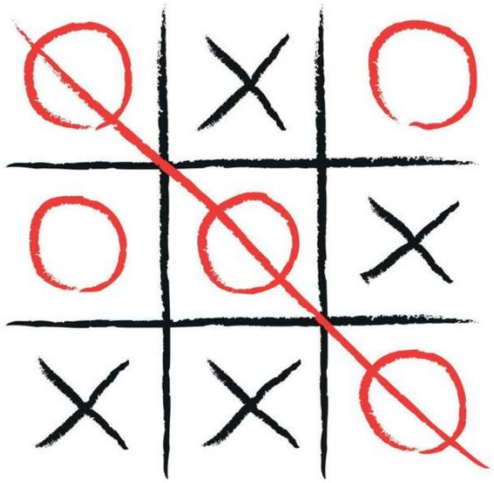


# Properties of Classical Game Models

Most AI game models assume:

- Deterministic (no chance)
- Two-player (adversarial)
- Zero-sum (my gain is your loss)
- Perfect information (observable)

| Game | Branching (b) | Avg. Depth |
|---|---|---|
| Tic-Tac-Toe | ~3 | 9 |
| Chess | ~35 | ~80 |
| Go | ~250 | ~150 |



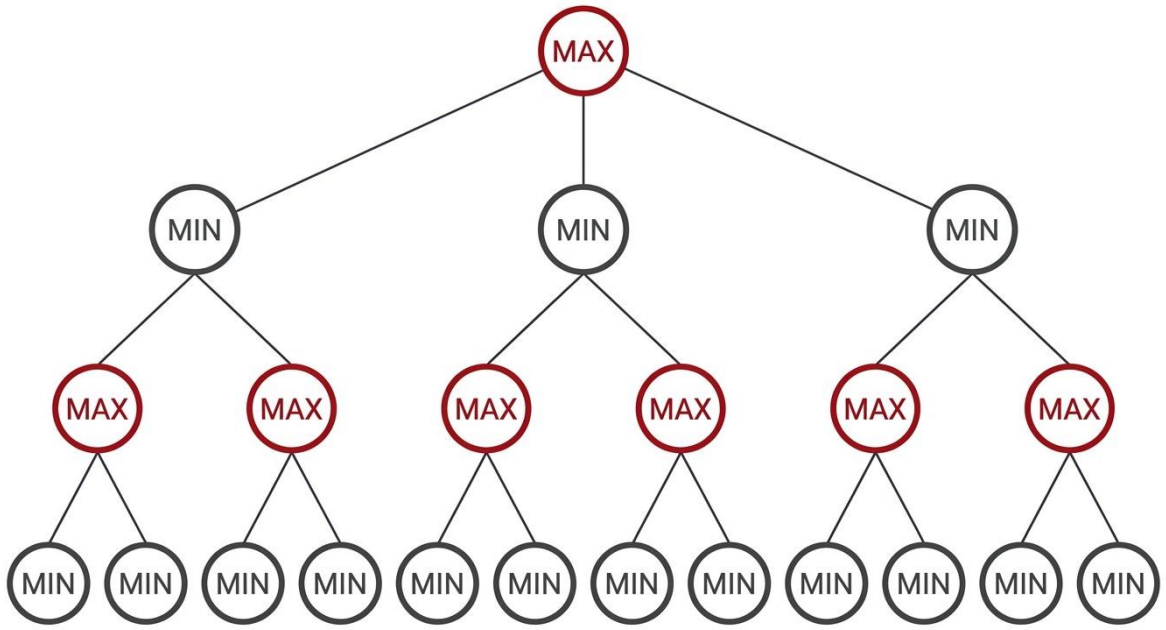
Tic-tac-toe（井字棋）

# Game Tree Representation

- **Nodes**: Game states.
- **Edges**: Legal moves/actions.
- **Layers**: Alternate between players.

**MAX Node:** We want to maximize the outcome.

**MIN Node:** Opponent wants to minimize the outcome.

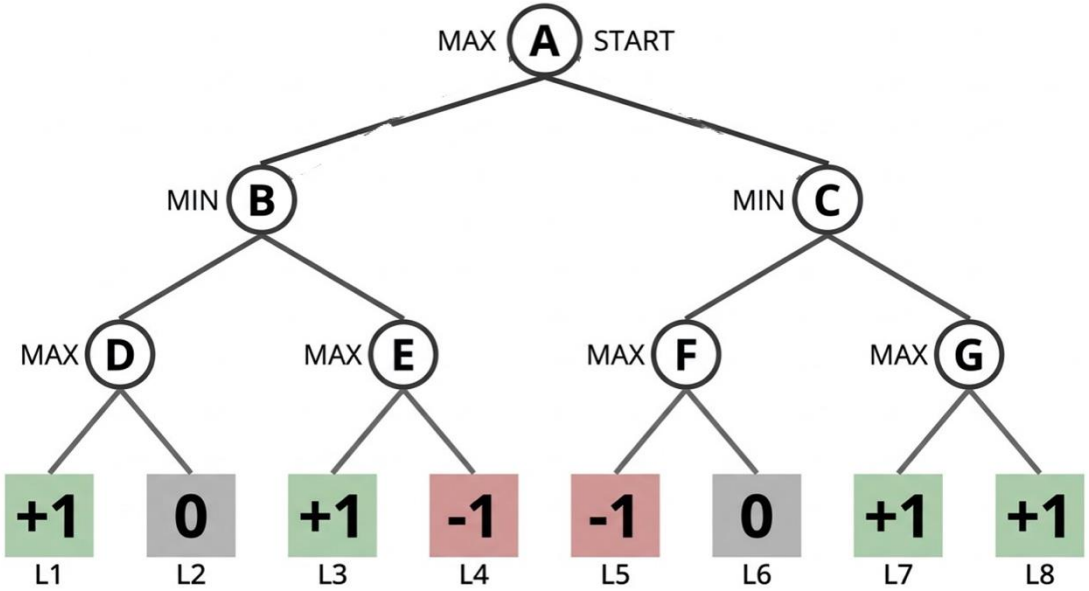


# Utility Values

Terminal states are assigned values.

- **Win = +1**
- **Draw = 0**
- **Loss = -1**

*Goal: Choose actions leading to states with the best possible utility.*

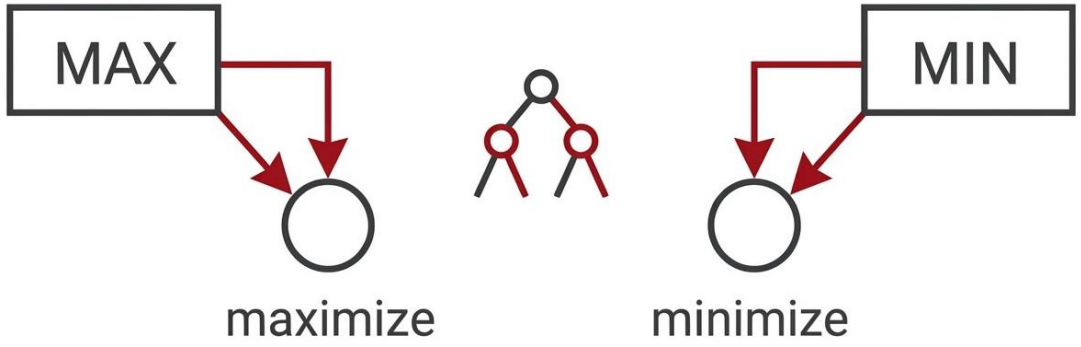


# Rational Opponent Assumption

Both players are assumed to act optimally.

- **MAX** chooses action that maximizes utility.
- **MIN** chooses action that **minimizes** utility.

This symmetry defines the Minimax Principle.



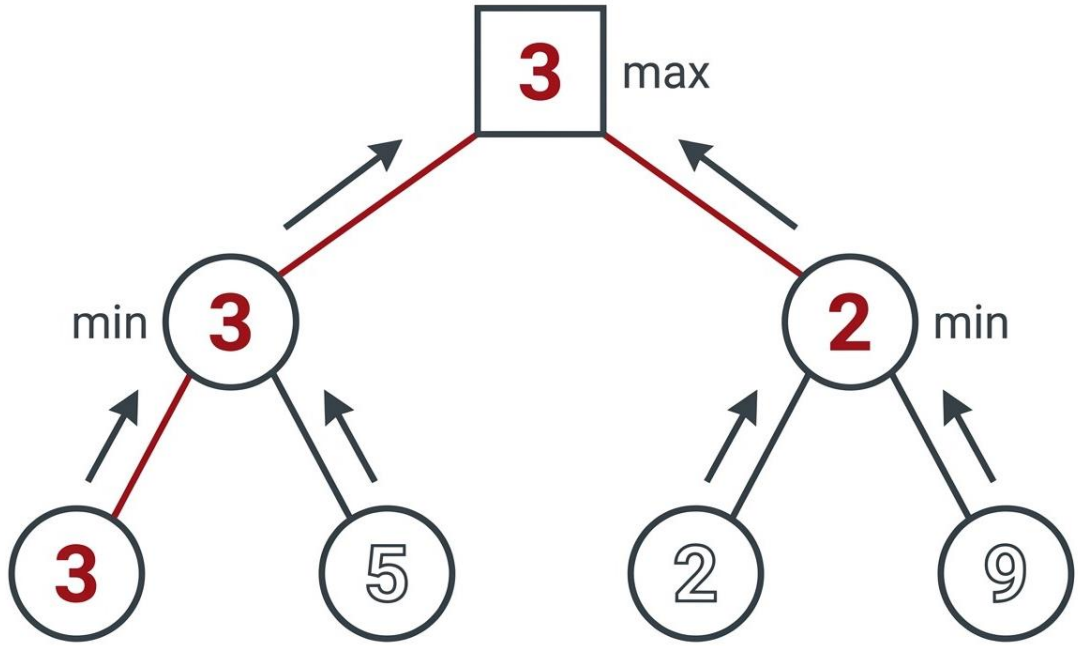
# MiniMax Definition

```
V(s) =
     U(s) if terminal
     max V(T(s, a)) if MAX node
     min V(T(s, a)) if MIN node
```

Values are propagated **bottom-up** from terminal states to the root.

# MiniMax Definition

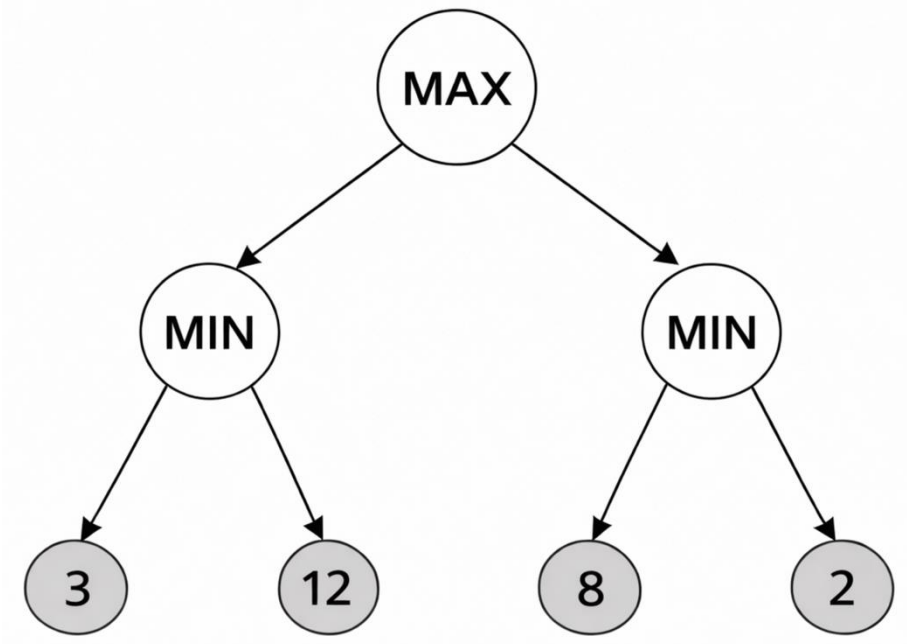
```
V(s) =
    U(s) if terminal
    max V(T(s, a)) if MAX node
    min V(T(s, a)) if MIN node
```




Values are propagated **bottom-up** from terminal states to the root.

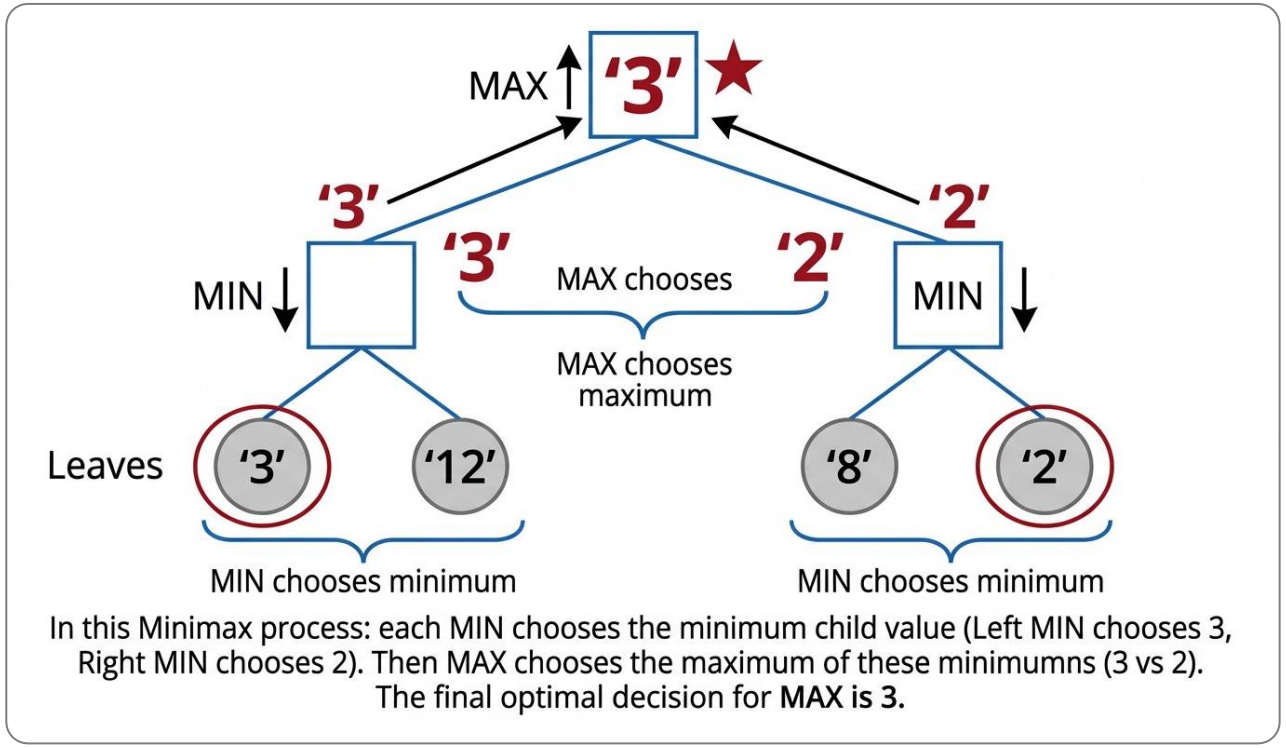
# MiniMax Example

**Interactive Question:** Analyze the game tree on the right.
What value should MAX choose as the root?



# Evaluating the Tree

- **Step 1: MIN** nodes select the minimum of children.
- **Step 2: MAX** node selects the maximum of results.
- **Final Decision:** Left branch (Value 3).



# MiniMax Complexity

**Time Complexity:**

$O(b^d)$

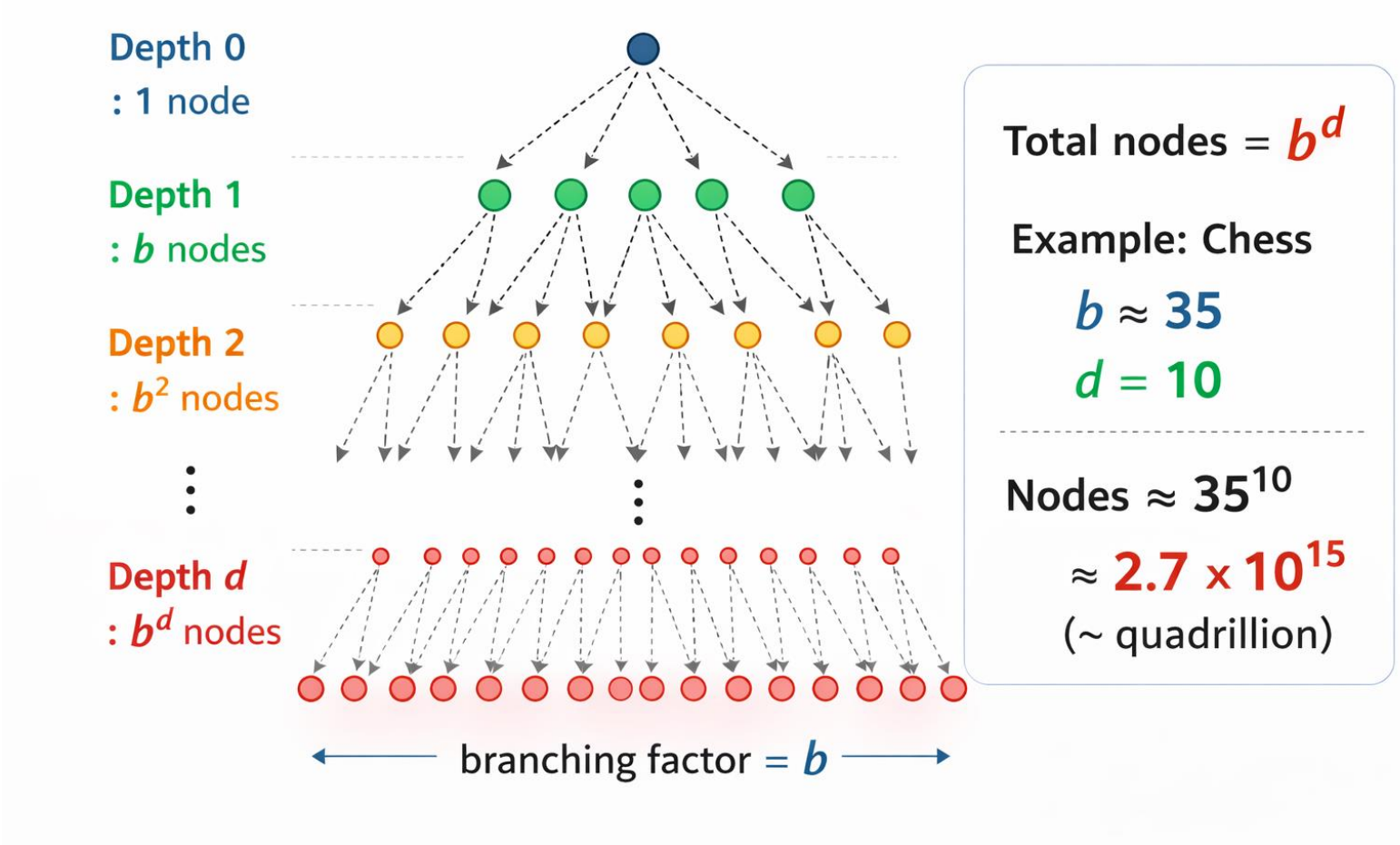
**b** = branching factor

**d** = search depth

**Chess Example:**

b ≈ 35, d = 10

Nodes ≈ $35^{10}$ ≈ 2.7 quadrillion

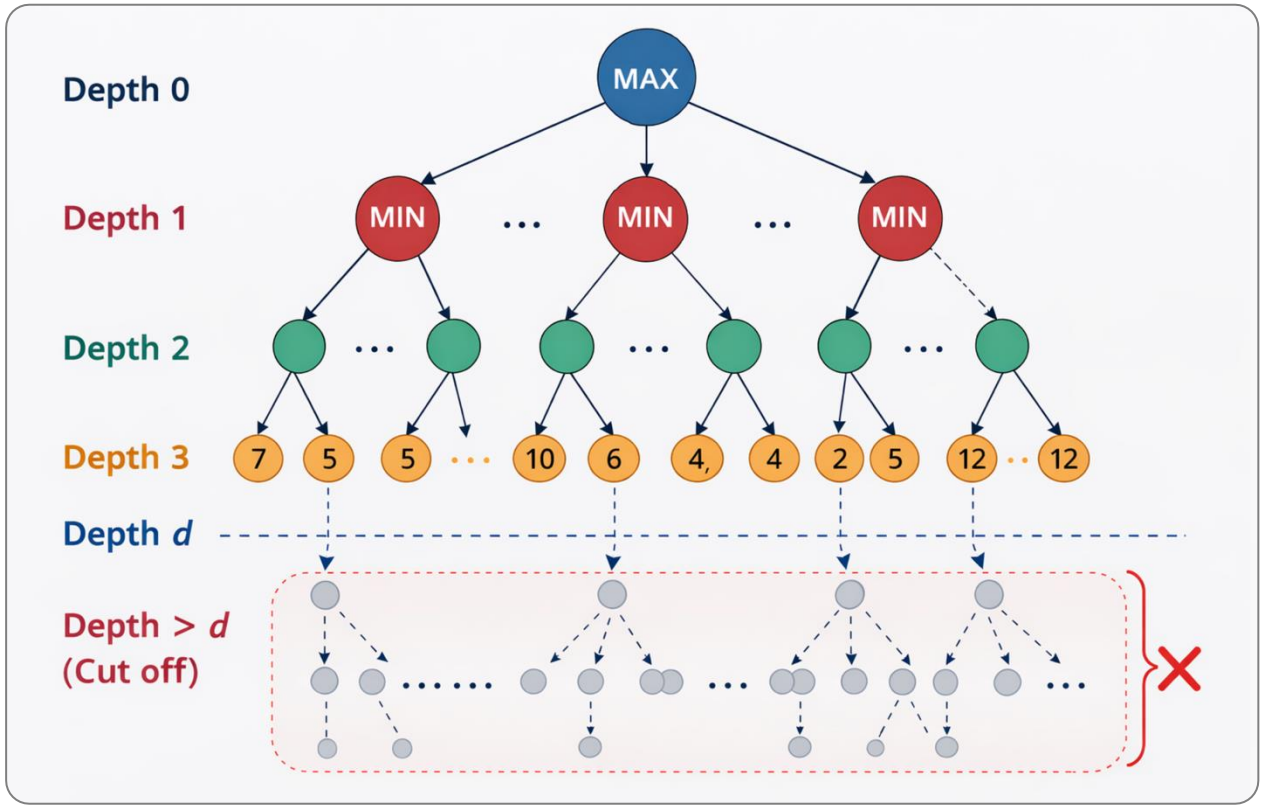


# Depth-Limited Search

Full search is impossible for complex games.

**Solution:** Cut off search at depth *d*.

- Evaluate leaves using **heuristic evaluation**.
- Treat heuristic values as terminal utilities.
- Propagate values back to root.



# Evaluation Functions

Heuristic function **f(s)** estimates the probability of winning.

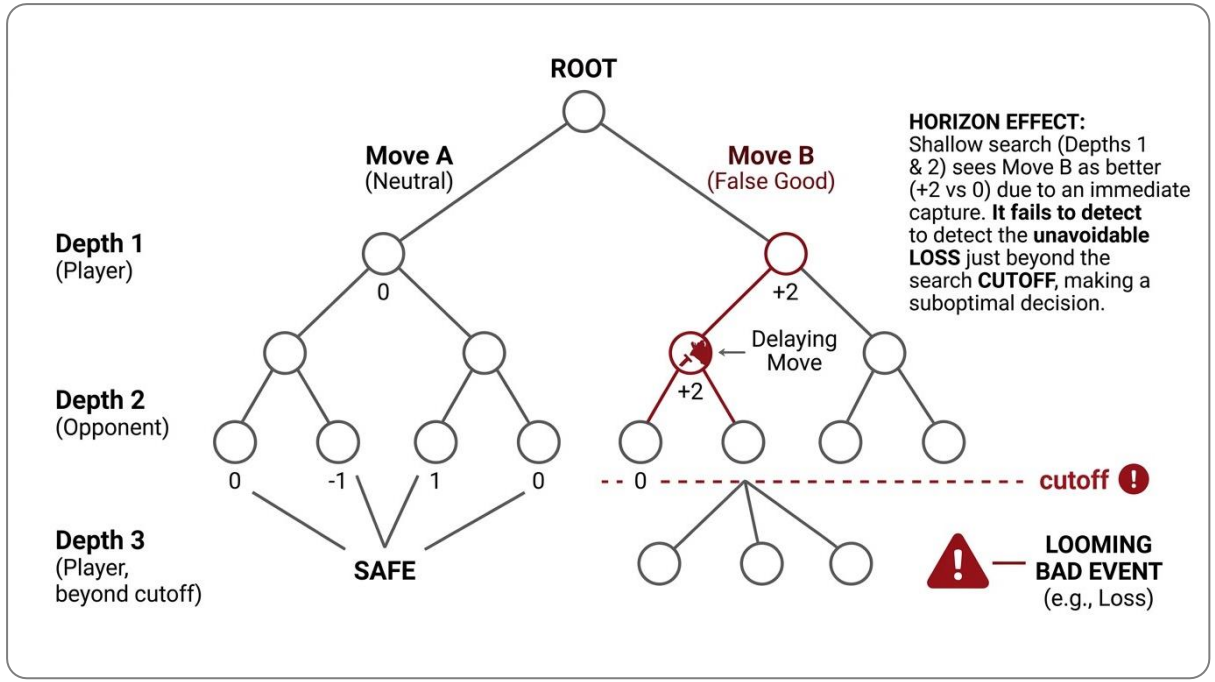
**Chess Features:**

- **Material Balance:** Count pieces
- **King Safety:** Is the king exposed?
- **Mobility:** How many squares can pieces move to?
- **Board Control:** Dominance of the center.

# Horizon Effect

Depth-limited search can be blind to events just beyond the cutoff.

The AI might delay an inevitable loss by making moves that push the event **beyond its search horizon**.



# Alpha-Beta Pruning

- Observation:
  **Some branches cannot influence the final root decision.**

- Why evaluate them?

- Goal: Compute the exact same minimax result while exploring significantly fewer nodes.

# Alpha & Beta Values

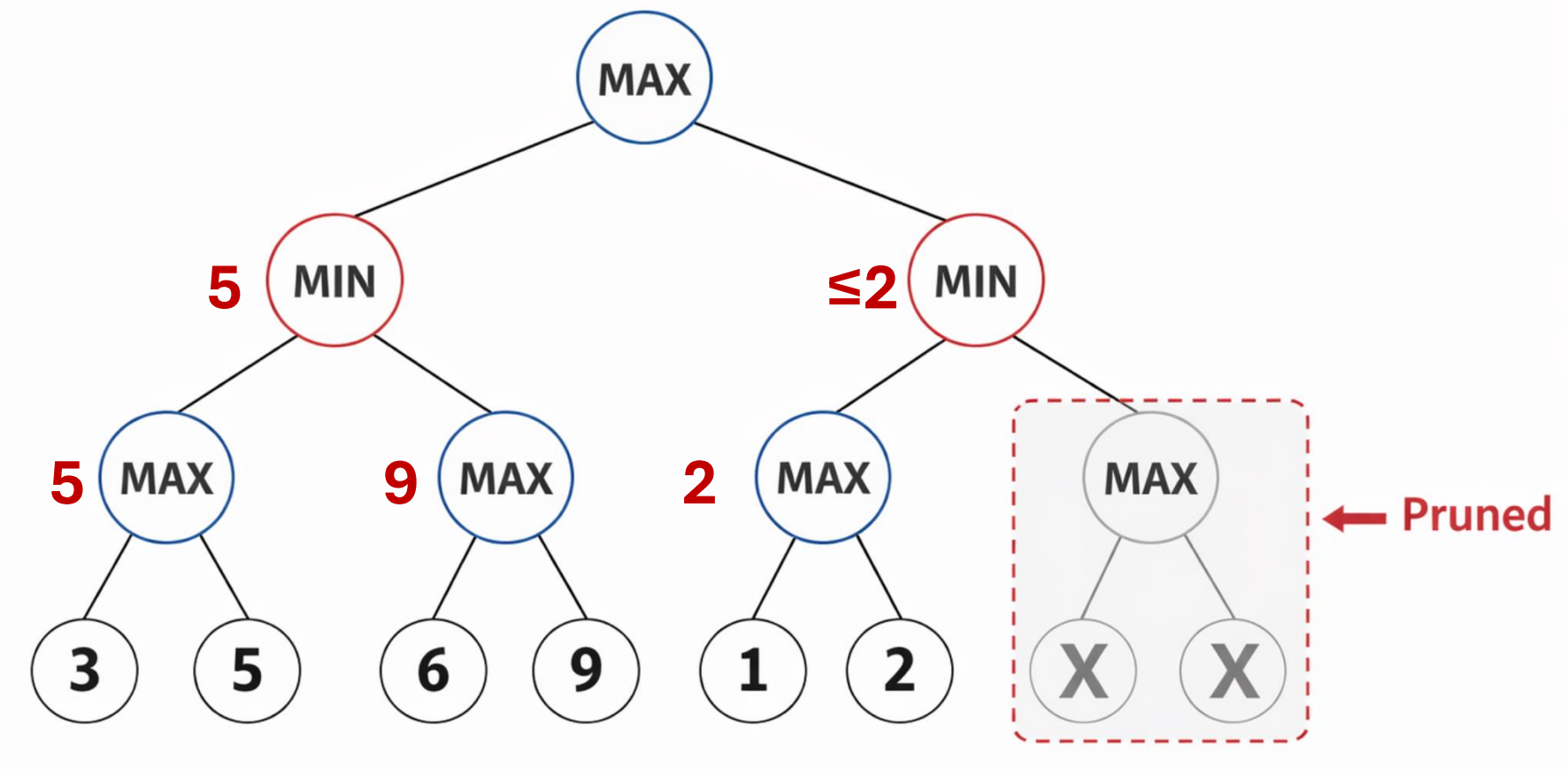
**Alpha (α)**
The **best** value MAX can guarantee so far.

**Beta (β)**
The **best** value MIN can guarantee so far.

**Pruning Condition:**

$$\alpha \geq \beta$$

# Alpha-Beta Pruning Example



# Alpha-Beta Complexity

**Worst Case:** $O(b^d)$

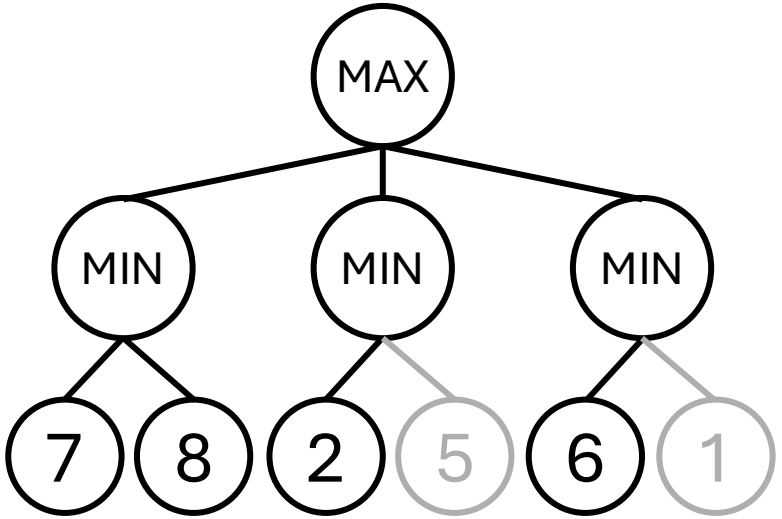
**Best Case:** $O(b^{d/2})$

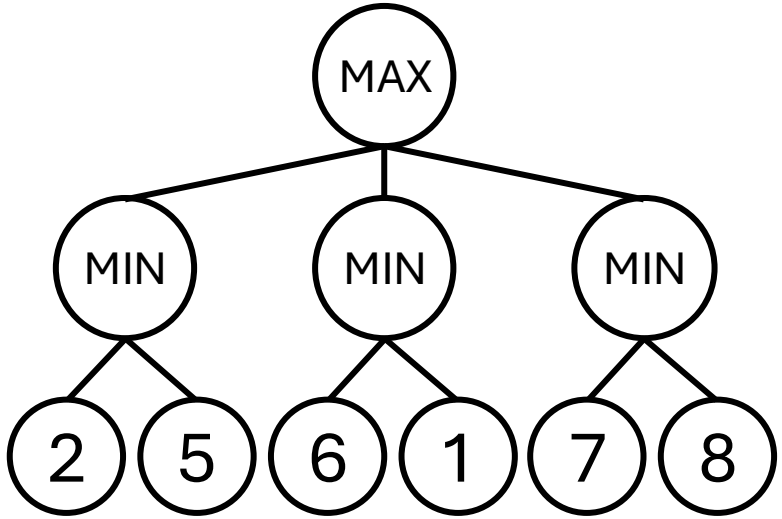
With perfect move ordering, alpha-beta can search **twice as deep** as minimax for the same cost.

Effective branching factor: $\sqrt{b}$

# Move Ordering

- Alpha-beta performance depends heavily on move ordering.
- If best moves are explored first => **Maximum Pruning**.
- **Techniques:**
  - **Heuristic Ordering:** Capture moves first, etc.
  - **Iterative Deepening:** Use previous results to sort moves.



# Iterative Deepening Search

**Procedure:**

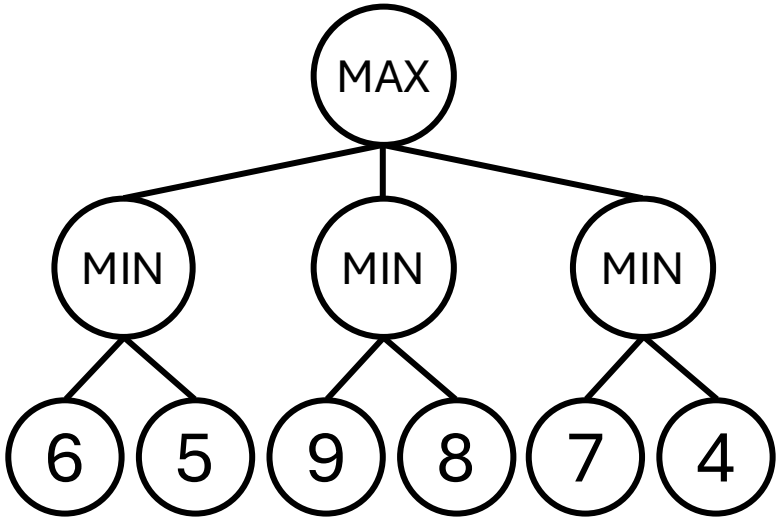
Loop for depth **d = 1, 2, 3…**

**Advantages:**

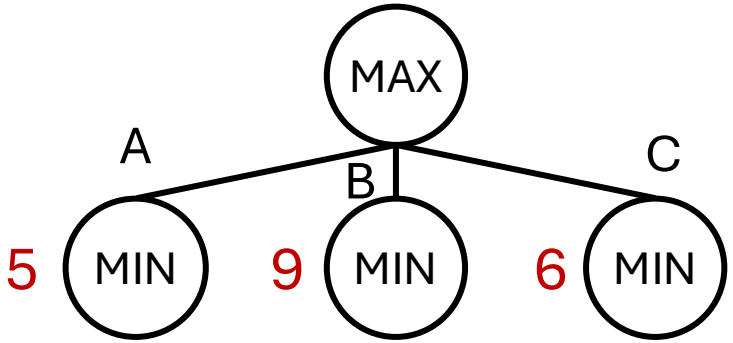
- **Anytime Algorithm:** Always has a move ready.
- **Move Ordering:** Results from depth *d-1* help sort moves at *d*.
- **Time Control:** Can stop when time is up.

# IDS Example

## True Game Tree

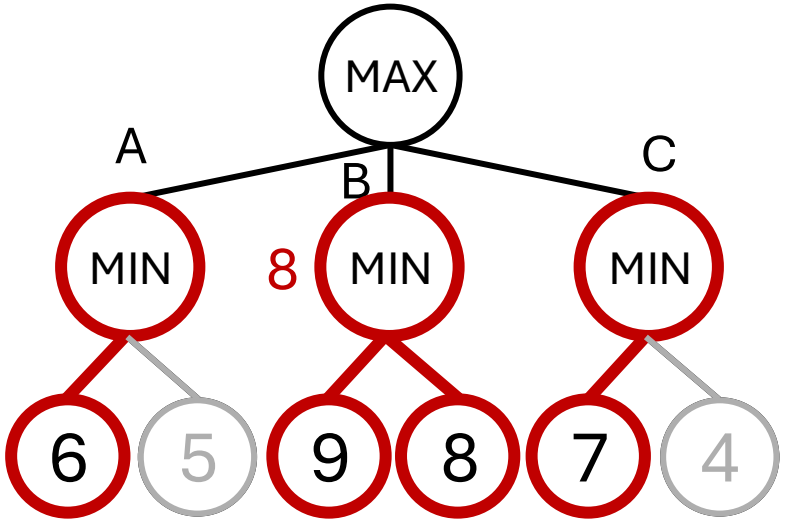



## Depth 1




B > A > C

## Depth 2



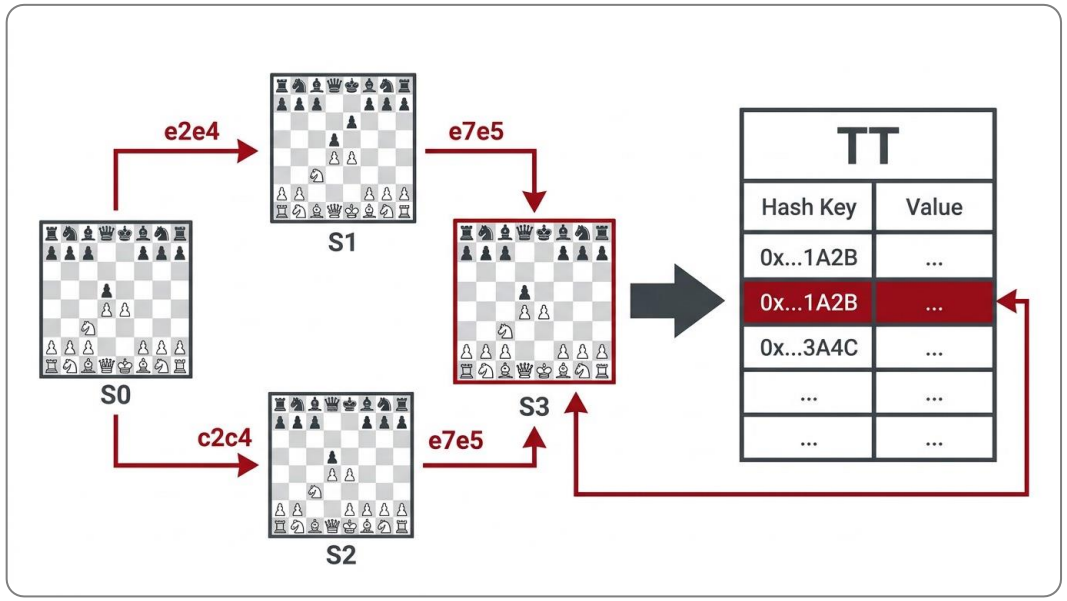
# Transposition Tables

Many game states can be reached via **different paths** (transpositions).

**Hash Table**

Key: Board State Hash

Value: Evaluated Minimax Score




**Benefit:** Avoid recomputing the same subtree multiple times.

# Limitations of Minimax

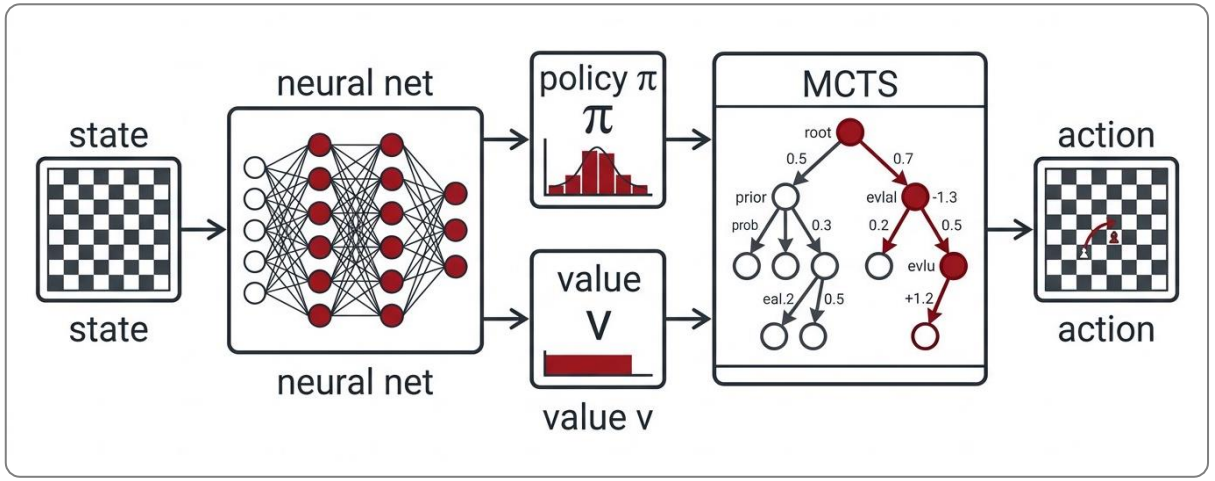
Problems:

- Exponential complexity
- Huge branching factors
- Need strong evaluation functions

Example:

- Branching factor ≈ 250
- Minimax search becomes infeasible

How did AlphaGo succeed? (later lectures)

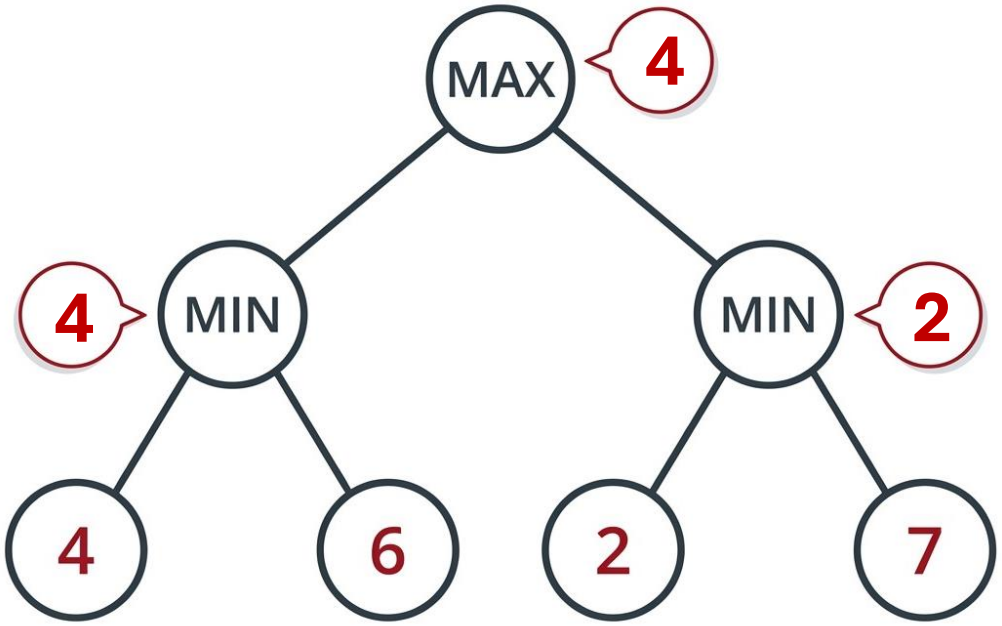



AlphaGo: Deep Learning + Monte-Carlo Tree Search

# Exercise

**Evaluate the tree:**

1. What is the minimax value at the root?
2. Which action should MAX choose?



# Key Takeaways

- Adversarial Search models environments with competing intelligent agents.
- Minimax Principle: Optimal play assumes both players maximize their own utility.
- Alpha-Beta Pruning: Drastically improves search efficiency without changing the result.
- Heuristics: Essential for searching complex games beyond the horizon.

# Next Lecture

Constraint Satisfaction Problems