Search In Games

Artificial Intelligence
DD2380

Content

- Zero Sum Game
- MiniMax Search
- NegaMax
- Alpha-Beta Pruning
- Repeated States
- Scoring Function/Heuristics
- Tips

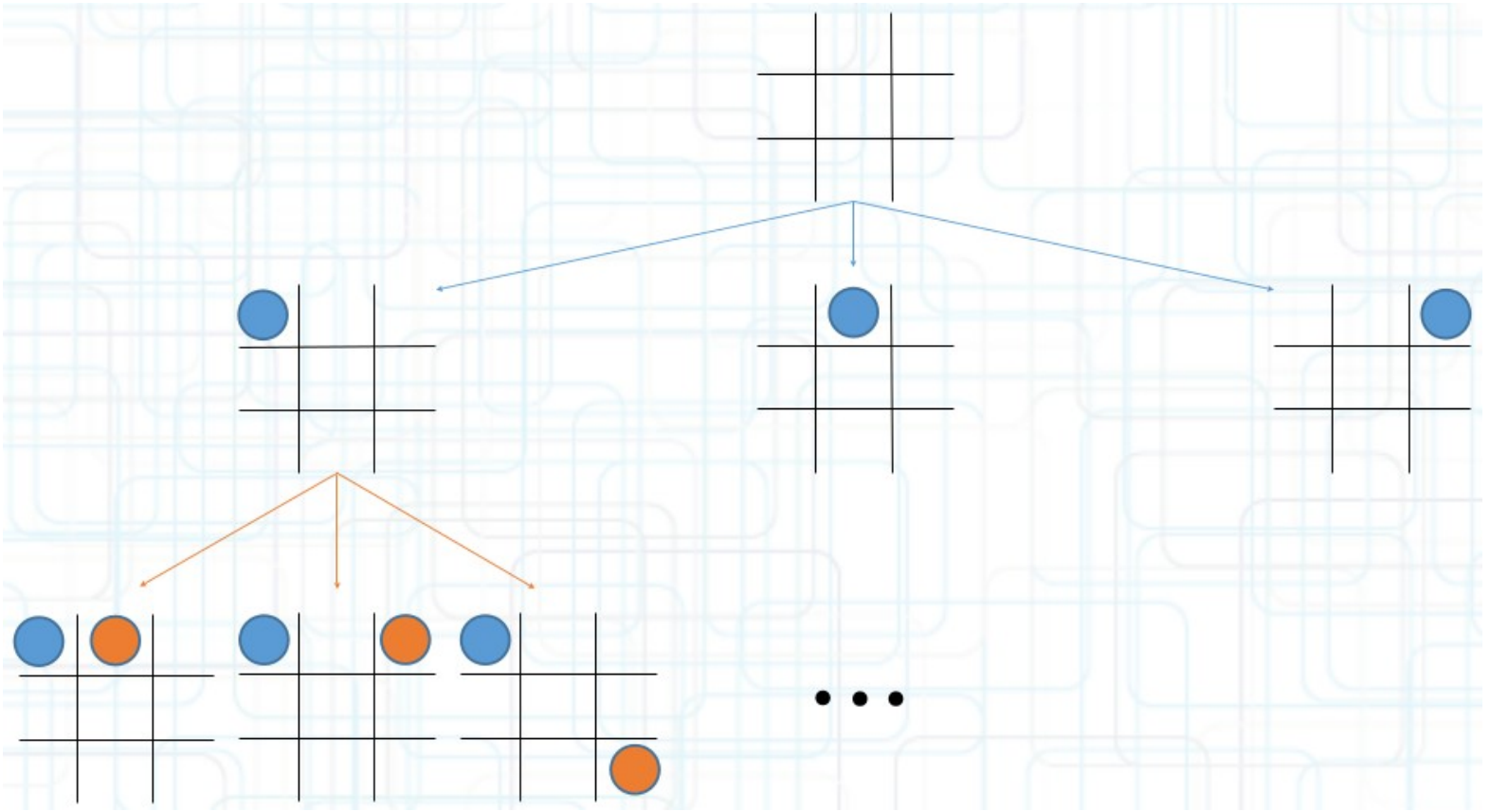
Content

- Zero Sum Game
- MiniMax Search
- NegaMax
- Alpha-Beta Pruning
- Repeated States
- Scoring Function/Heuristics
- Tips

Zero Sum Game

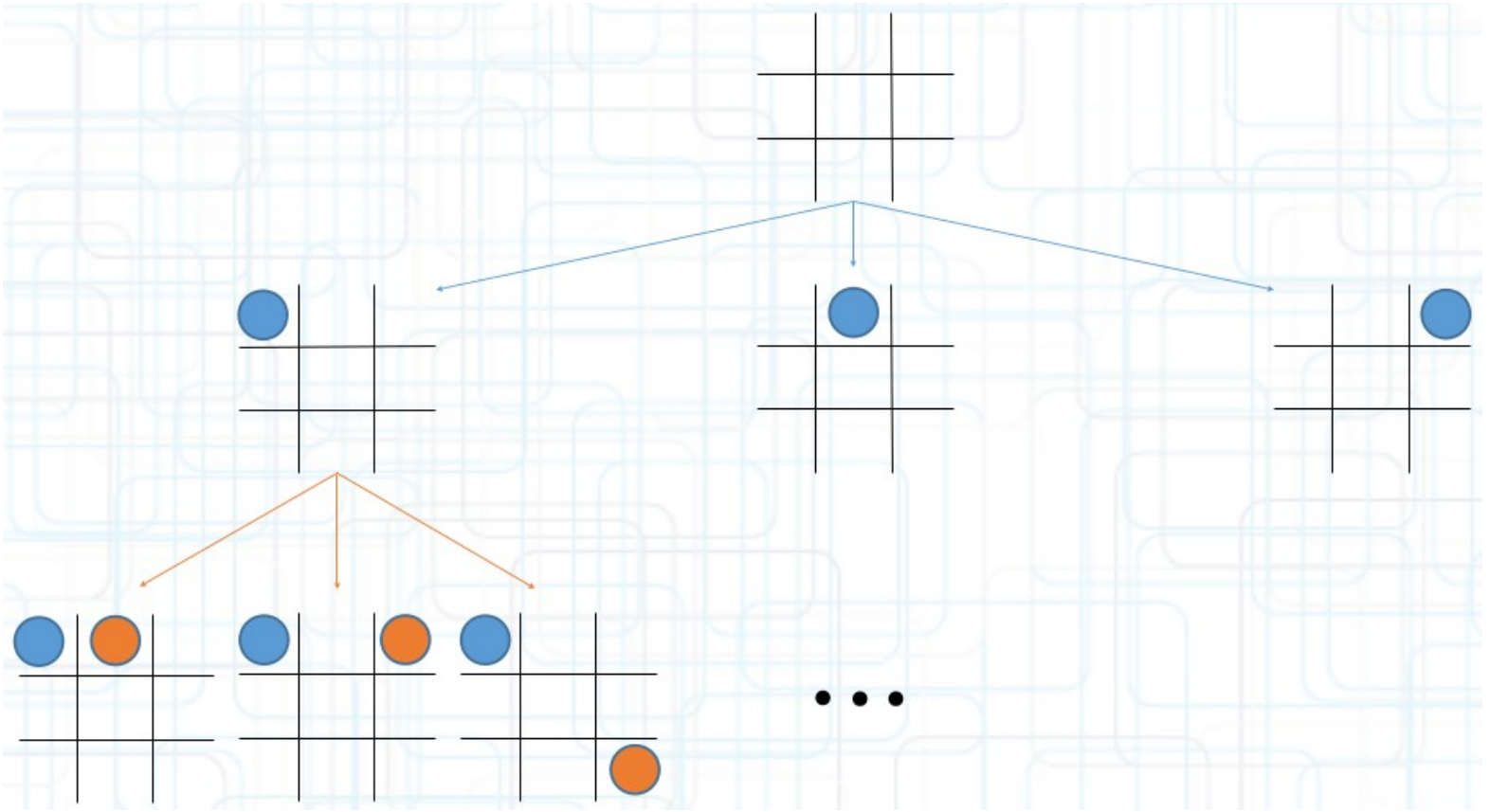
- If one player wins, the other player loses
  - No collaboration!
- Gain/loss of player 1 = loss/gain of the player 2
  - Total gains/losses of the participants will **sum to zero**
- How to measure gains and losses?
  - Scoring Function/ Heuristic.
- Examples: Chess, Checkers, Tic-Tac-Toe etc.

Zero Sum Game



- Tree Representation
  - Nodes: States
  - Edges: Moves / Actions

Zero Sum Game



- Optimize your move
  - Create the game tree in the background
  - See the opponent as smart as you are

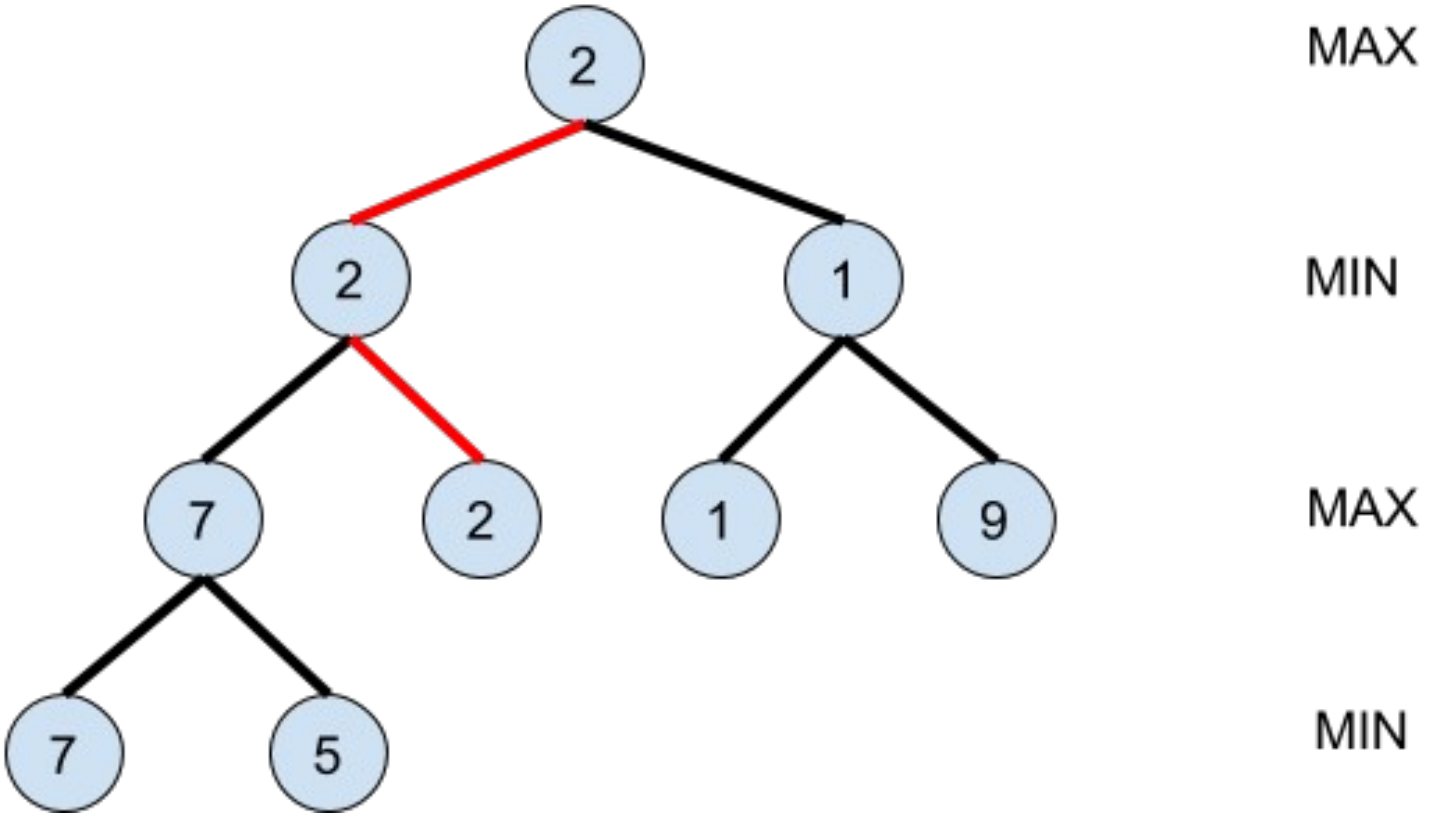
Content

- Zero Sum Game
- MiniMax Search
- NegaMax
- Alpha-Beta Pruning
- Repeated States
- Scoring Function/Heuristics
- Tips

MiniMax Search

- Strategy to find the best move
- Two players, taking turns: Max & Min
  - **Max** wants to **maximize** it's own gain
  - **Min** wants to **minimize** Max's gain (as smart as you)
- So, we want to look further in the game to see what will probably happen – **Search**.
- DFS or Iterative Deepening
  - WHY? Only leaves are scored!

MiniMax Search



- Two layer types:
  - Max: Take the best possible move
  - Min: Also take the best possible, but for the opponent
  - Depth first

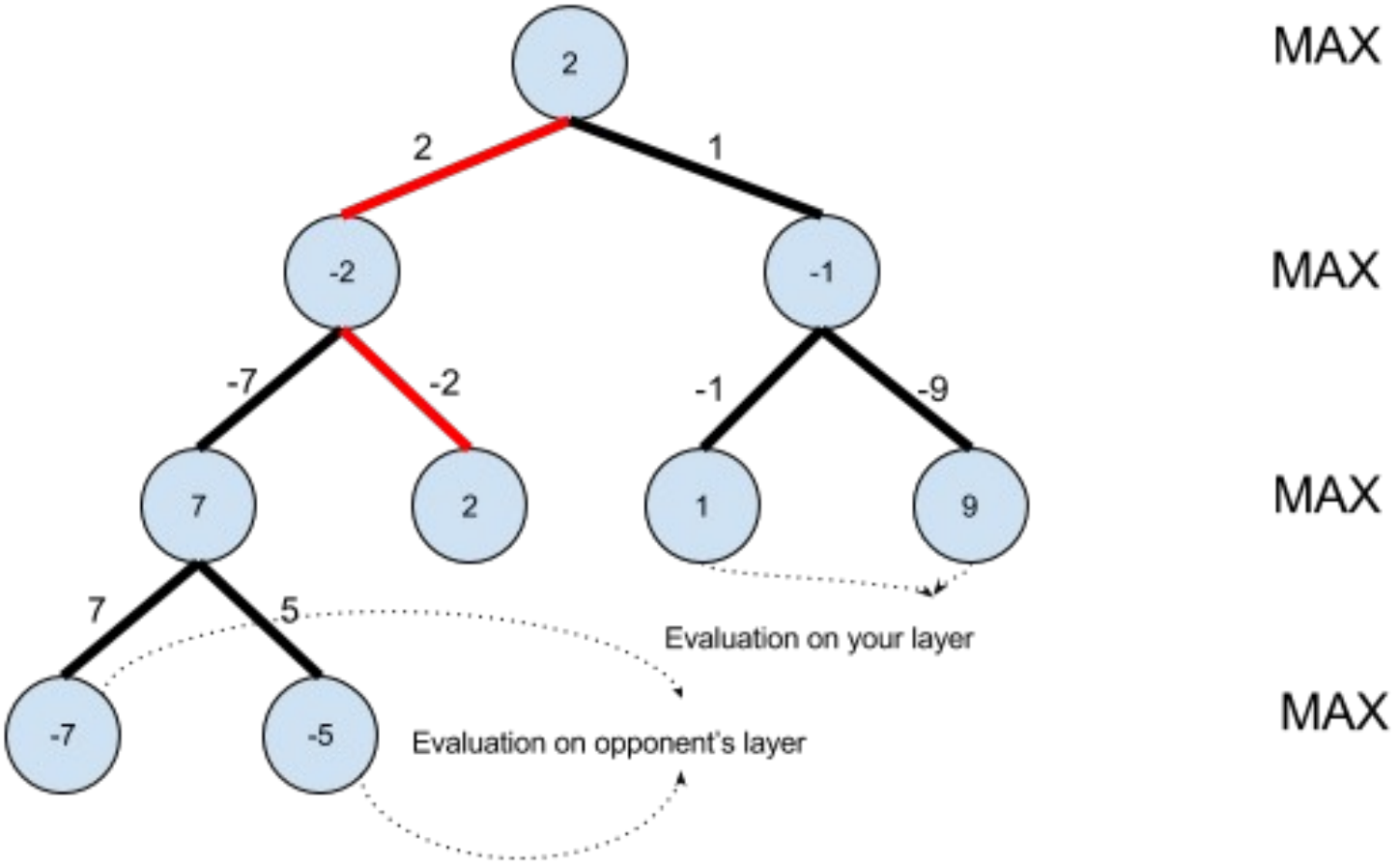
Content

- Zero Sum Game
- MiniMax Search
- NegaMax
- Alpha-Beta Pruning
- Repeated States
- Scoring Function/Heuristics
- Tips

NegaMax Search

- Same theory with MiniMax
  - Back propagate the max value of children nodes multiplied by -1.
- Always take the maximum from children nodes
- Save a function

NegaMax Search



- Multiply by -1 when back-propagate
- Multiply by -1 the evaluation is on opponent's layer

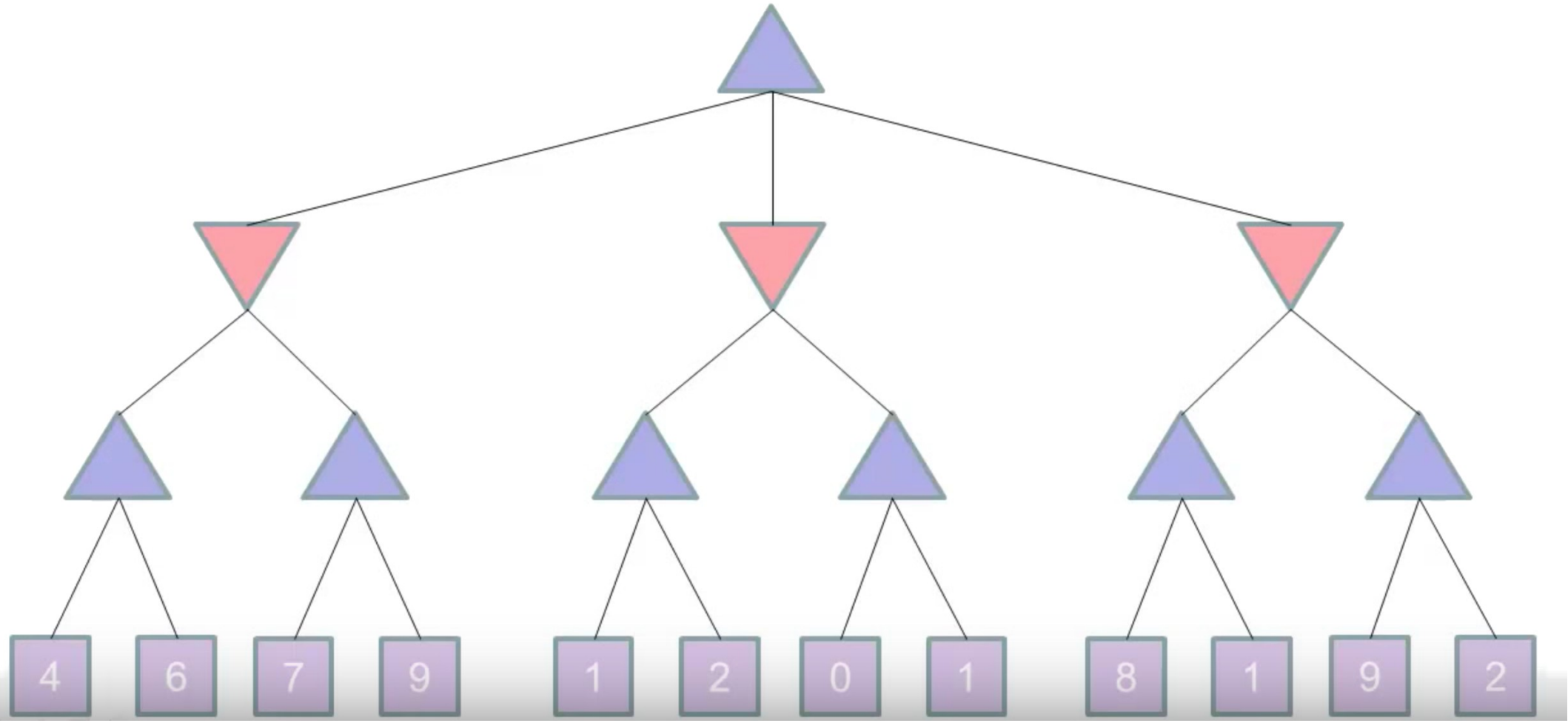
Content

- Zero Sum Game
- MiniMax Search
- NegaMax
- Alpha-Beta Pruning
- Repeated States
- Scoring Function/Heuristics
- Tips

Alpha-Beta Pruning

- Some states are useless to be checked
  - Because their values will not make effects
- Alpha: Highest value found along path to the root for Max
  - Max's Lower Bound
- Beta: Lowest value found along path to the root for Min
  - Min's Upper Bound
- Prune when Beta ≤ Alpha

Alpha-Beta Pruning



- CS188 Artificial Intelligence, by Prof. Pieter Abbeel

Example

3x3 Tic-Tac-Toe

Content

- Zero Sum Game
- MiniMax Search
- NegaMax
- Alpha-Beta Pruning
- Repeated States
- Scoring Function/Heuristics
- Tips

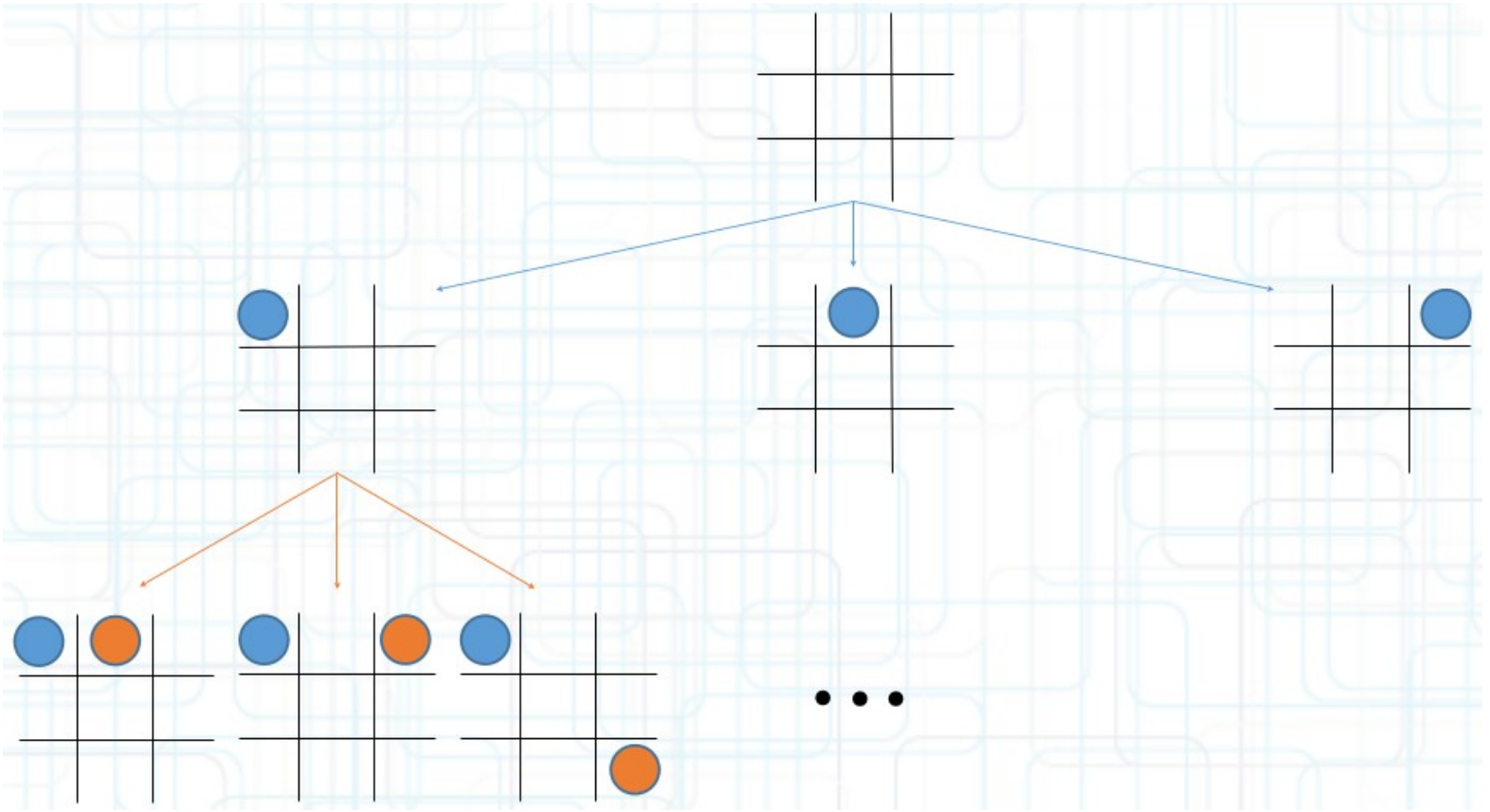
Repeated States

- Search algorithms can visit same states via different sequences of moves
  - Computationally expensive
  - Downgrade the performance

Repeated States

- We can implement a data structure to avoid calculating the same states
  - Hash function
- What are equivalent states?
  - Same placements of pieces
  - **Symmetric** placements
  - **OBS!** Who made the move (when it matters)?

Repeated States



- How are the repeated states created?

Content

- Zero Sum Game
- MiniMax Search
- NegaMax
- Alpha-Beta Pruning
- Repeated States
- Scoring Function/Heuristics
- Tips

Scoring Function / Heuristics

- **Experience-based** techniques for problem solving, learning, and discovery that give a solution which is not guaranteed to be optimal.
- Score the leaf states.
- Different people have different strategies, there is no best one.

Scoring Function / Heuristics

- Count the number of aligned markers
  - Simple and fast to compute
  - But would introduce false positives
  - What else... ...

Content

- Zero Sum Game
- MiniMax Search
- NegaMax
- Alpha-Beta Pruning
- Repeated States
- Scoring Function/Heuristics
- Tips

Tips and Questions

- HW
  1. Make sure your MiniMax/NegaMax works
  2. AB-Pruning
  3. Repeated states checking
  4. Symmetry breaking
  5. Move ordering for AB-Pruning
  6. Ending states lookup table

Tips and Questions

- Kattis has a time limit, return the best in hand when time is up
- Almost EOG? Search to the end!

Have Fun!