Optimal Decisions in Games

Game definition

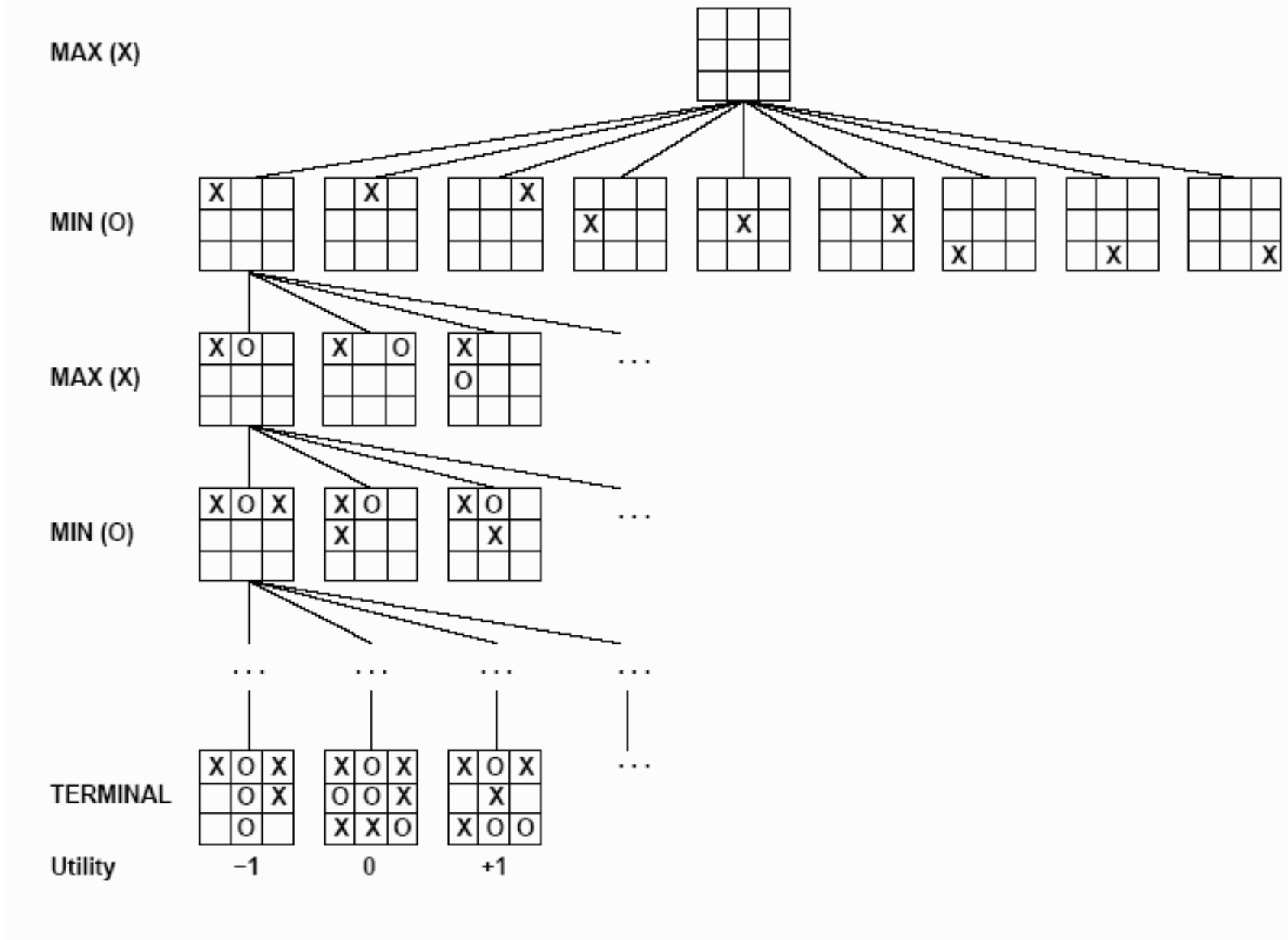
- **initial state** - includes board position and the player to move
- **successor function** - returns a lists of (move, state) pairs, where move is legal
- **terminal test** - game over at some terminal state
- **utility function** (objective or payoff function) - give numeric value of a terminal state
  E. g. win: $+1$, loss: -1, draw: 0

Tic-Tac-Toe

Initial state and legal moves define game tree (see below figure for tic-tac-toe)

- Initial state: board empty, MAX turn
- Successor function: returns the list of 9 pairs (move, state)
  E. g., from the initial state:
  - (0, [X, 1, 2, 3, 4, 5, 6, 7, 8])
  - (1, [0, X, 2, 3, 4, 5, 6, 7, 8])
  - (2, [0, 1, X, 3, 4, 5, 6, 7, 8])
  - (3, [0, 1, 2, X, 4, 5, 6, 7, 8])
  - (4, [0, 1, 2, 3, X, 5, 6, 7, 8])
  - (5, [0, 1, 2, 3, 4, X, 6, 7, 8])
  - (6, [0, 1, 2, 3, 4, 5, X, 7, 8])
  - (7, [0, 1, 2, 3, 4, 5, 6, X, 8])
  - (8, [0, 1, 2, 3, 4, 5, 6, 7, X])

(Partial) Tic-Tac-Toe Search Tree



Tic-Tac-Toe

- The tic-tac-toe layout is
  0 1 2
  3 4 5
  6 7 8
- The initial state of the board without moves is $[0, 1, 2, 3, 4, 5, 6, 7, 8]$
- After an X is moved the the central tile the state is $[0, 1, 2, 3, X, 5, 6, 7, 8]$
- `utility_of(state)` returns $+1$ if winner is X (MAX player), -1 if winner is O (MIN player), or 0 otherwise
- `successors_of(state)` returns a list of pairs (move, state) as shown in the previous slides
- `is_terminal(state)` returns `True` if the state is either a win or a tie (board full)

Questions

1. What is the branching factor at depth 0? At depth 1?
2. What is the maximal depth?
3. Will a MIN move attempt to minimize or maximize the utility?
4. Are states after a terminal state explored?
5. Are all possible states explored to a terminal state?
6. Is this a depth-first or breadth-first search? How do you know? (see Python code)
7. Run the MinMax tic-tac-toe program (you will play O)

Homework

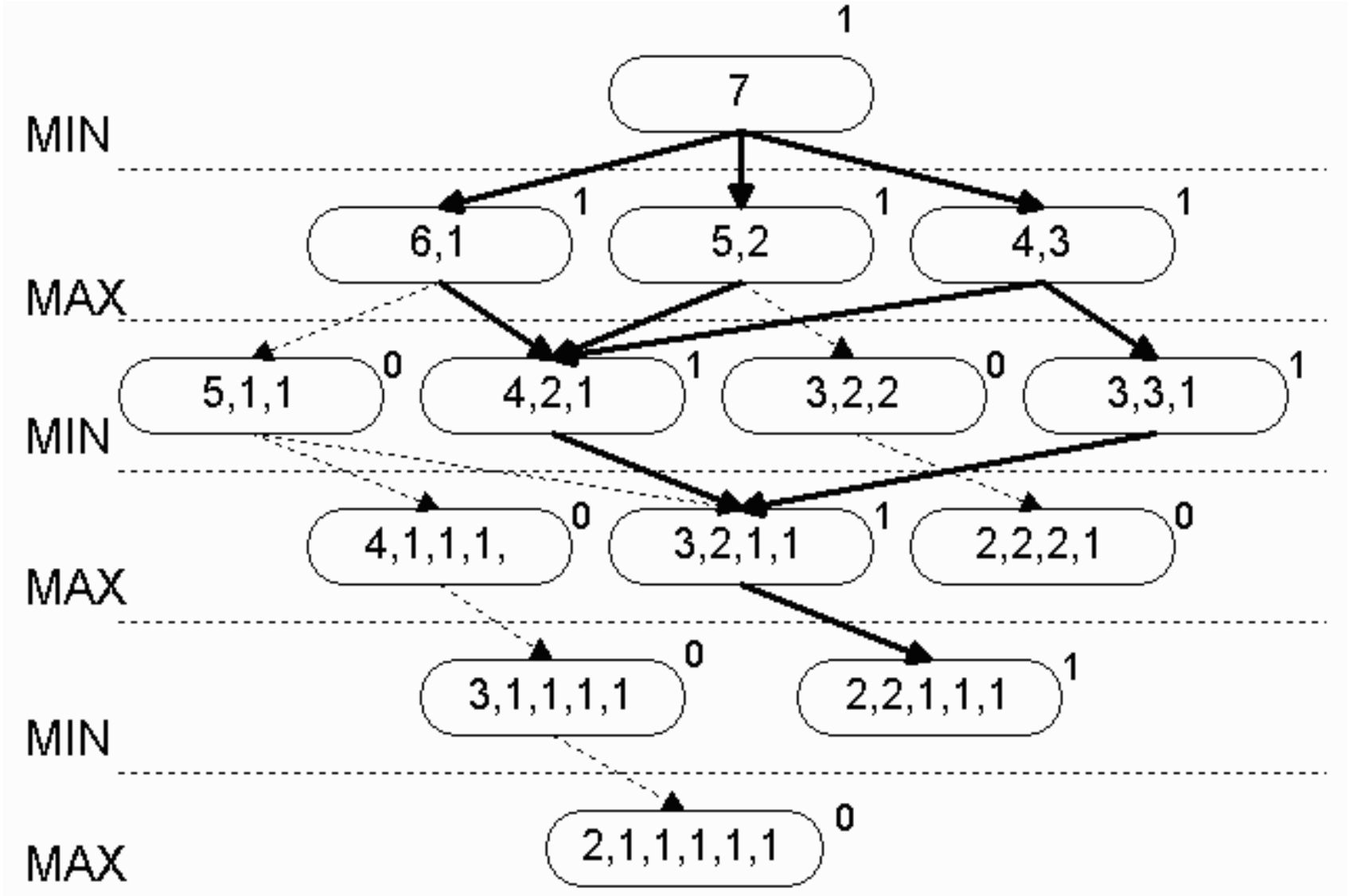
1. Write a nim game using `minmax_decision` from tic-tac-toe program. The state space of nim may be exhaustively searched.
   To play nim, a number of tokens are placed in a pile between two opponents; at each move, the player must divide a pile into two nonempty piles of different sizes. Thus, a pile of six tokens [6] may be divided into two piles of [5,1] or [4,2] but not [3,3]. The first player who can no longer move loses.
   The picture in the following slide illustrates a state space for a game with 7 tokens.
   MIN should start the game and, with 7 tokens is certain to lose if MAX divides the piles following the heavy arrows.
   Test using 15 in the starting pile.

Homework

3. Rewrite nim using `alpha_beta_decision` and compare with minimax over a pile of 20
4. Modify nim to play the MIN position, i. e., `utility_of` should return $+1$ for MIN and -1 for MAX

Note: MAX is stateless, no memory of previous piles is held. Any pile, e. g., [1, 2, 3, 4], can be sent to MAX which will return its move.

Nim Search Tree



Challenge

Solve Breakthrough using alpha beta search

- The game is played on a chessboard, every piece is a pawn.
- A piece may move one space straight or diagonally forward if the target square is empty.
- A piece may move into a square containing an opponent's piece if and only if that square is one step diagonally forward. The opponent's piece is removed and the player's piece replaces it. Note that capturing is not compulsory, nor is it "chained" as in checkers.
- The first player to reach the opponent's home row — the one farthest from the player — is the winner. If all the pieces of a player are captured, that player loses.

Adapted from

`https://en.wikipedia.org/wiki/Breakthrough_(board_game)`