

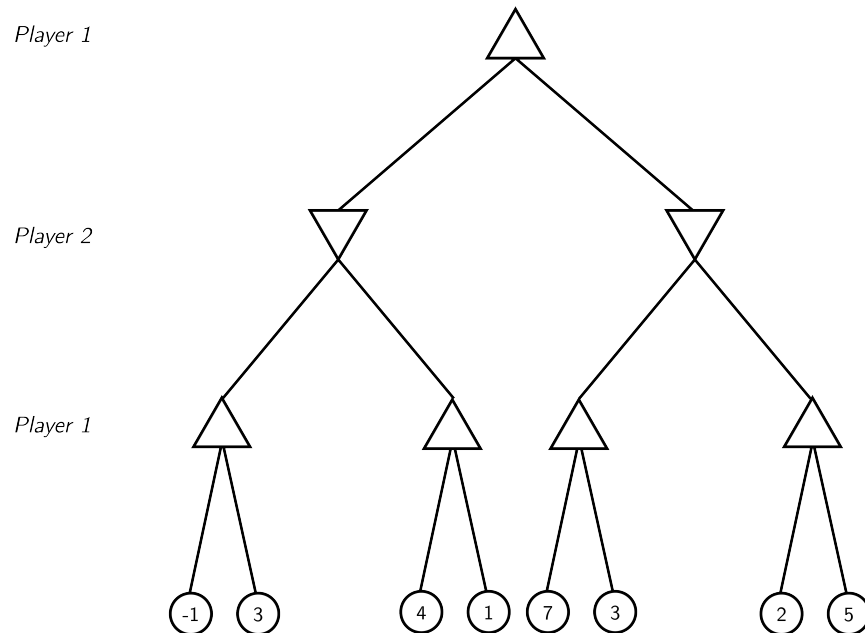
# Lab session 4: Adversarial search

Pierre-Alexandre Murena

April 26, 2024

## 1 Applying the Minimax algorithm

**Question 1.** What is the optimal strategy for Player 1 in the following sequential game?



## 2 Nim

Nim is a 2 player game, where the players have to remove objects from a pile initially containing  $N$  objects. Players take turn, and can remove only 1, 2 or 3 objects at a time. We call the player who makes the first move MAX, and the other one MIN. The winner is the one removing the last object.

**Question 2.** In the case where  $N = 4$ , write the game tree and show that MAX is guaranteed to lose if MIN plays rationally.

In the file `nim.py`, there is an implementation of Nim.

**Question 3.** Implement the MINIMAX algorithm by completing the functions `minimax_search`, `max_value` and `min_value`. Play some rounds of the game against the MINIMAX-opponent by calling the function `interactive_game`. Can you win? Also try different values of  $N$ .

**Question 4.** Implement another function `maximin_search`, which should return an optimal action for MIN. Use your implementation of `minimax_search` as a basis. Let the two rational agents play against each other by calling the function `optimal_game`. Who wins? Try it for different values of  $N$  and look for a pattern.

**Question 5** (optional, 1 + 1 bonus points).

- a) Show that, whenever  $N$  is a multiple of 4, MAX is guaranteed to lose if MIN plays rationally.
- b) Show that, in all other cases, MIN is guaranteed to lose if MAX plays rationally.

*Submission deadline for question 5: Wednesday, May 8, 2024*

### 3 Tic-tac-toe

Another well-known two-player game is tic-tac-toe. Both players play on a  $3 \times 3$  grid, with MAX placing crosses ( $\times$ ) and MIN placing noughts ( $\circ$ ). An implementation is found in `tic_tac_toe.py`.

**Question 6.** Complete the functions `alpha_beta_search`, `max_value` and `min_value`. Play against the agent by executing the `interactive_game` function. Can you win?

After each move by MAX, the number of explored nodes is printed. By commenting the lines responsible for breaking the for-loop over all actions, you will see that the number of explored nodes is substantially larger and, consequentially, the search becomes slower.

**Question 7.** Alter the code of `max_value` and `min_value` to change the order that the actions are searched in. You can make use of `random.shuffle()` from the Python standard library for this. Does it have an effect on the number of recursive calls? If so, why?