Introduction to Artificial Intelligence

Dr. Thomas Keller
C. Büchner, C. Grundke, A. Kauffmann

University of Zürich
Spring Semester 2022

Exercise Sheet 5

Due: March 30, 2022

Points total: 20 marks

Exercise 5.1 – Minimax — 8 marks

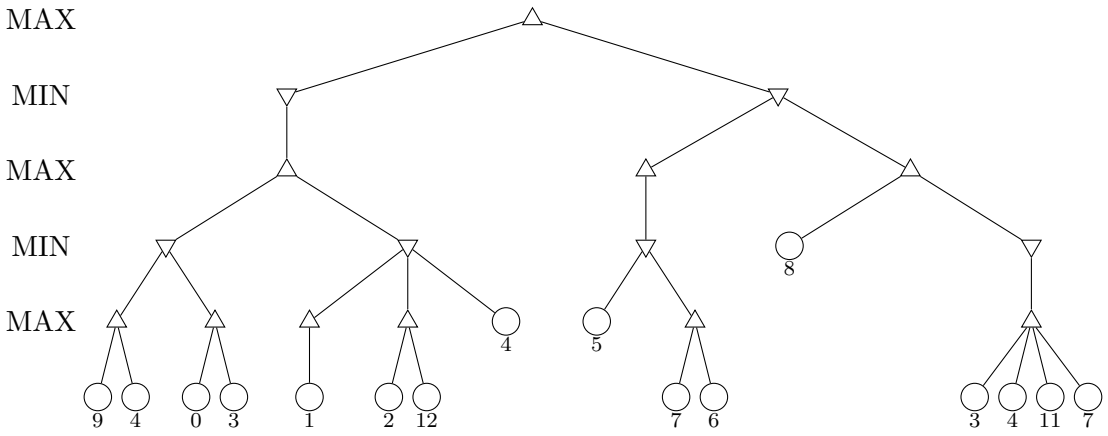
A state in the *Nim* game consist of $n > 0$ stacks of coins where $c_i \geq 0$ is the number of coins on stack i. Players MAX and MIN alternate (with MAX starting) choosing a nonempty stack i and removing between 1 and c_i coins from it. Whichever player removes the last coin from the last stack wins the game.

(a) Draw the game tree for a game starting with two stacks, one with three coins and one with one coin. Denote leaves with 1 if they represent a winning state for MAX and with -1 if they represent a winning state for MIN. (3 marks)

(b) Apply Minimax in order to determine the value of the root. Which player has a winning strategy? (4 marks)

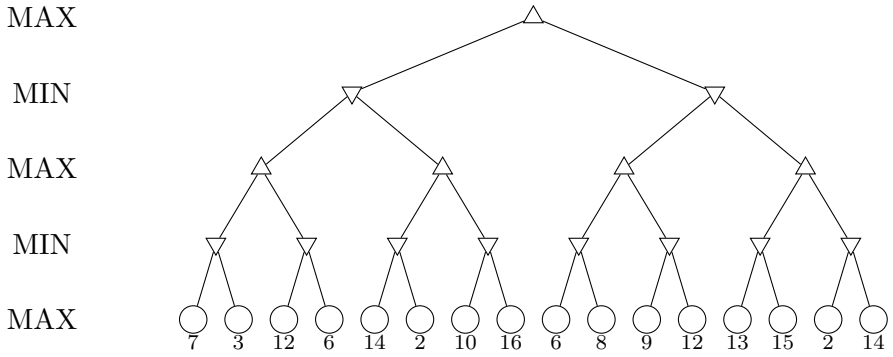
(c) Who has the winning strategy if the initial state is (4, 1)? Justify your answer by describing the winning strategy. (1 mark)

Exercise 5.2 – Alpha–Beta search — 6 marks



Apply alpha-beta search to the game tree depicted above, considering successor nodes from left to right. Annotate all considered nodes with the returned value as well as the (last) alpha and beta values, and show which parts of the tree can be pruned (e.g., by drawing cut lines through edges which lead to subtrees that do not need to be considered).

Exercise 5.3 – Monte Carlo Tree Search **6 marks**



Apply the first 4 iterations of Monte Carlo Tree Search with the following policies:

- tree policy: For MAX nodes select the successor with higest utility, for MIN nodes select the successor with lowest utility.
- expansion: Always pick the left child first.
- default policy: Always pick the right child.

Estimate updates calculate the average of all utilities that were backpropagated trough that node. Given a node n with (old) utility u, (updated) visit count v and r being utility of the terminal node that was reached in the simulation phase, we can efficiently calculate the updated utility for n with $u + (r - u)/v$.

Submission rules:

- Exercise sheets must be submitted in groups of three students. Please submit a single copy of the exercises per group (only one member of the group does the submission).
- Create a single PDF file (ending `.pdf`) for all non-programming exercises. Use a file name that does not contain any spaces or special characters other than the underscore "_". If you want to submit handwritten solutions, include their scans in the single PDF. Make sure it is in a reasonable resolution so that it is readable, but ensure at the same time that the PDF size is not astronomically large. Put the names of all group members on top of the first page. Make sure your PDF has size A4 (fits the page size if printed on A4). Submit your single PDF file to the corresponding exercise assignment in MOODLE.
- For programming exercises, only create those code text files required by the exercise. Put your names in a comment on top of each file. Make sure your code compiles and test it. Code that does not compile or which we cannot successfully execute will not be graded. Create a ZIP file (ending `.zip`, `.tar.gz`, or `.tgz`; *not* `.rar` or anything else) containing the code text file(s) (ending `.py`) and nothing else. Do not use directories within the ZIP, i.e., zip the files directly.
- Do not upload several versions to MOODLE, i.e., if you need to resubmit, use the same file name again so that the previous submission is overwritten.