Foundations of Artificial Intelligence

M. Helmert
S. Eriksson
Spring Term 2022

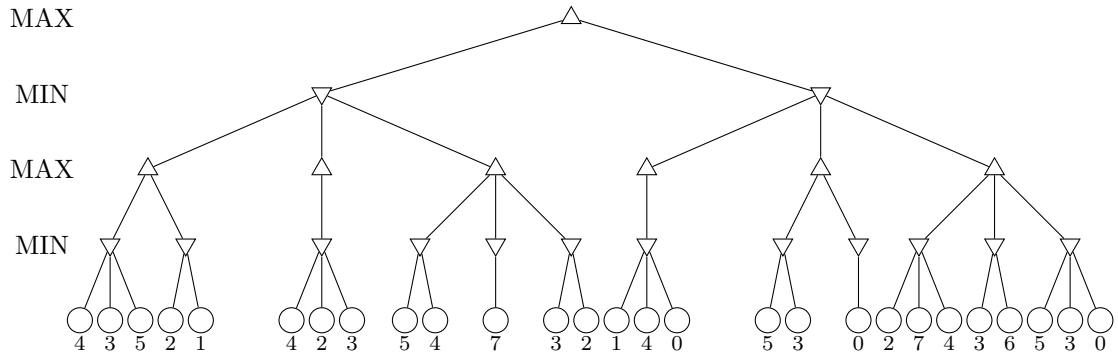
University of Basel
Computer Science

Exercise Sheet 12

Due: May 22, 2022

Important: for submission, consult the rules at the end of the exercise. Non-adherence to the rules will lead to your submission not being corrected.

For exercises 12.2 to 12.4, we consider the following game tree T:



Exercise 12.1 (2 marks)

Consider the following state in a tic tac toe game, where the "circle" player goes next:

```
o o
x
xx
```

Draw the game tree (including utility of terminal nodes) of the tic tac toe game rooted at this state, with MAX being the "circle" player and MIN the "cross" player. Your nodes should contain a representation of the game state.

Exercise 12.2 (1.5+0.5 marks)

(a) Use the minimax algorithm to annotate T.

(b) According to the values computed by the minimax algorithm, what is a the playout if both players play optimal?

Exercise 12.3 (2.5 + 0.5 marks)

(a) Use alpha-beta search for T. Consider 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).

(b) How many nodes does alpha-beta search need to consider? Discuss the difference compared to Exercise 12.2.

Exercise 12.4 (3 marks)

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

- tree policy: For MAX nodes select the successor with highest utility, for MIN nodes select the successor with lowest utility.
- expansion: Pick the leftmost unexplored child.
- default policy: Pick the rightmost child.

For each iteration, draw the search tree in a similar fashion as on slide 23 of chapter 43 (print version). Namely, show the entire visited search tree, the updated utility estimate and visit counter for each visited node, and the value of the terminal node that the simulation phase explored.
Note: For updating utility estimates, we need to 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 the 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 two 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. Either use page numbers on all pages or put your names on each page. Make sure your PDF has size A4 (fits the page size if printed on A4).
- For programming exercises, only create those code textfiles 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.
- For the submission: if the exercise sheet does not include programming exercises, simply upload the single PDF. If the exercise sheet includes programming exercises, upload a ZIP file (ending .zip, .tar.gz or .tgz; *not* .rar or anything else) containing the single PDF and the code textfile(s) and nothing else. Do not use directories within the ZIP, i.e., zip the files directly.
- Do not upload several versions to ADAM, i.e., if you need to resubmit, use the same file name again so that the previous submission is overwritten.