# Introduction to Artificial Intelligence (INFO8006)

## Exercises 2 – Games and adversarial search

January 4, 2023

## Learning outcomes

At the end of this session you should be able to

- formulate search problems associated to a game with an Initial state, Player function, Actions, Transition model, Terminal test and Utility function (IPATTU);
- define the algorithms to perform game search (Minimax, $\alpha - \beta$ pruning, H-Minimax, Expectminimax, MCTS);
- apply Minimax, $\alpha - \beta$ pruning and H-Minimax in fully observable adversarial environments.

## Exercise 1 21 misery game (January 2019)

The game “21” is played with any number of players who take turns increasing a counter. The counter starts at 1 and each player in turn increases the counter by 1, 2, or 3, but may not exceed 21; the player who says “21” or larger loses.

1. Define the search problem associated with the 2-player version of the “21” game.
2. For the following, consider the game of “5” (still in its 2-player version), which has the same rules has “21” except that you should not say 5 or more. Show the whole game tree.
3. Using the Minimax algorithm, annotate your tree with the backed-up values, and use those values to choose the optimal starting move.
4. Mark the nodes that would be pruned, *i.e.* not evaluated, if $\alpha - \beta$ pruning was applied, assuming the nodes are generated in the optimal order for $\alpha - \beta$ pruning.

## Exercise 2 Tic-Tac-Toe (AIMA, Ex 5.9)

Tic-Tac-Toe is a game for two players, X and O, who take turns marking the cells of a $3 \times 3$ grid. The player who succeeds in placing three of their marks in a straight line (horizontal, vertical or diagonal) wins the game. If neither of the players win before the grid is full, its a draw.

We consider X as the max player and O as the min player. We define $X_n$ as the number of rows, columns or diagonals with exactly $n$ X's and no O's. Similarly, $O_n$ is the number of rows, columns, or diagonals with just $n$ O's. A position $s$ is terminal if $X_3(s) \geq 1$, $O_3(s) \geq 1$ or if the grid is full. The utility function assigns $+1$, $-1$ or $0$ to such position, respectively. For non-terminal positions, we use an evaluation function defined as $eval(s) = 3X_2(s) + X_1(s) - 3O_2(s) - O_1(s)$.

1. Define the search problem associated with the Tic-Tac-Toe game.
2. Approximately how many possible game states of Tic-Tac-Toe are there?
3. Show the whole game tree starting from an empty grid down to depth 2 (one X and one O on the board), taking symmetry into account.
4. Annotate your tree with the evaluations of all the positions at depth 2.
5. Using the H-Minimax algorithm, annotate your tree with the backed-up values for the positions at depths 1 and 0, and use those values to choose the optimal starting move.
6. Mark the nodes that would be pruned, *i.e.* not evaluated, if $\alpha - \beta$ pruning was applied, assuming the nodes are generated in the optimal order for $\alpha - \beta$ pruning.
7. Is this evaluation function a good heuristic? If not, provide one or more states $s$ for which $eval(s)$ is misleading.

## Exercise 3 Quiz

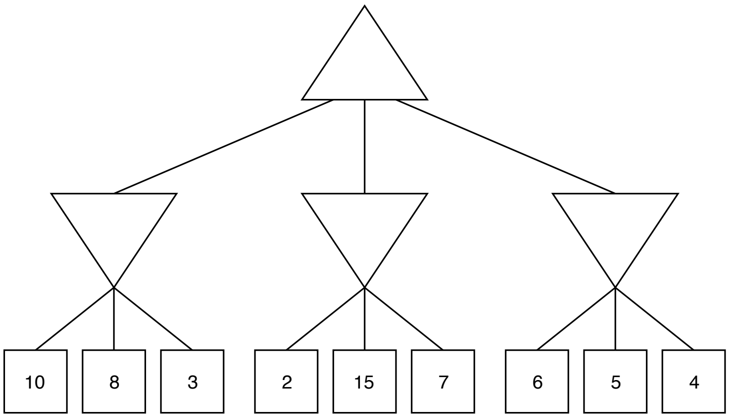
1. In a fully observable, turn-taking, zero-sum game between two perfectly rational players, does it help the first player to know what strategy the second player is using, *i.e.* what actions the second player will take? What if the second player is not rational ?
2. What is a quiescent state?
3. In Monte Carlo Tree Search (MCTS), what is encouraged by each term of the sum in the formula
$$\frac{Q(n',p)}{N(n')} + c\sqrt{\frac{2\log N(n)}{N(n')}},$$
and what is $c$?

## Exercise 4 Chess and transposition table (AIMA, Ex 5.15)

Suppose you have a chess program that can evaluate 16 million nodes per second.

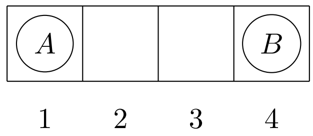
1. Decide on a compact representation of a game state for storage in a transposition table.
2. About how many entries can you fit in a 4 Go in-memory table?
3. Will that be enough for the three minutes of search allocated for one move?
4. How many table lookups can you do in the time it would take to do one evaluation? Suppose that you have a 3.2 GHz machine and that it takes 20 operations to do one lookup on the transposition table.

## Exercise 5 Minimax (UC Berkeley CS188, Fall 2019)




1. Consider the zero-sum game tree shown above. Triangles that point up, such as at the top node (root), represent choices for the maximizing player; triangles that point down represent choices for the minimizing player. Assuming both players act optimally, fill in the Minimax value of each node.
2. Which nodes can be pruned from the game tree above through alpha-beta pruning? If no nodes can be pruned, explain why not. Assume the search goes from left to right; when choosing which child to visit first, choose the left-most unvisited child.
3. Again, consider the same zero-sum game tree, except that now, instead of a minimizing player, we have a chance node that will select one of the three values uniformly at random. Fill in the Expectminimax value of each node. The game tree is redrawn below for your convenience.
4. Which nodes can be pruned from the game tree above through alpha-beta pruning? If no nodes can be pruned, explain why not.

## Exercise 6 Leapfrog (AIMA, Ex 5.8)




Consider the following two-player turn-taking game which initial configuration is shown in the figure above. Player $A$ moves first. Each player must move their token to an adjacent free cell in either direction. If the opponent occupies an adjacent cell, then a player may jump over the opponent to the next free cell, if any. For example, if $A$ is on 3 and $B$ is on 2, then $A$ may move back to 1. The game ends when a player reaches the opposite end of the board. If player $A$ reaches cell 4 first, then the value of the game to $A$ is $+1$; if player $B$ reaches cell 1 first, then the value of the game to $A$ is $-1$.

1. Define the search problem associated with this game.
2. Draw the complete game tree, using the following conventions:
   - Put each terminal state in a square box and annotate it with its game value.
   - Put loop states (states that already appear on the path to the root) in double square boxes. Since their value is unclear, annotate them with a "?" symbol.
3. Explain why the standard minimax algorithm would fail on this game.
4. Annotate each node with its backed-up minimax value. Explain how you handled the "?" values and why.
5. This 4-cell game can be generalized to $n$ cells for any $n > 2$. Prove that $A$ wins if $n$ is even and loses if $n$ is odd.

## Supplementary materials

- Chapter 5 of the reference textbook.