Part 2 Option B: 60 Marks - Multi - Game Playing Agent

The aim of this assignment is for you to better understand computer game playing by implementing a multi - game playing agent with the minimax algorithm and alpha-beta pruning.

Task 1 (50 Marks)

The agent will need to play multiple games, including the Tic-Tac-Toe, the game of Nim with 4 heaps in configuration of (1,3,5,7), the game of Tiger vs Dogs. The detailed description will be given at the end of this option.

In particular, you will need to:

(a) Implement the minimax algorithm and alpha-beta pruning: (30 marks)

- (i) using complete tree search. (Call them $Minimax_{complete}$, $AB_{complete}$)
  You may use Depth-first search (DFS).
- (ii) using depth limited tree search and an evaluation function.(Call them $Minimax_{limited}$, $AB_{limited}$)
  For depth limited tree search, you just need to introduce a depth variable and when it decreases to 0, either the evaluation value or the terminal value is returned. For Tic-Tac-Toe game, you can use the evaluation function in the workshop 2. For Nim and Tiger vs Dogs, you need to come up with your own evaluation function.

A Pseudocode can be found at `https://en.wikipedia.org/wiki/Minimax,https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning`

You need to

- Report on the details of your program design (data structures, and functions).
- Provide an evaluation of the players you've developed. Let two algorithms compete against each other and report on the performance difference, e.g., win-draw-lose rate, and resource consumption. A random player (selecting moves randomly) is needed to serve as baseline.

(b) Implement an intuitive user interface (10 marks)

This allows the human player to know the game state/actions to play. In the start of games, the human player should be able to choose who plays first and which version of the algorithm to use (in the case of depth limited, allowing to set the search depth).

Report on the program design and some sample plays.

(c) Scalability Study (10 marks)

Your agent is resource bounded, so the response of the player will be slower if the game gets larger. Design and conduct experiments to study the scalability of your game playing agent by using scaled up versions of Tic-Tac-Toe and Nim.

Specifically, here are some important aspects to consider:

- Game complexity.
  In Tic-Tac-Toe, you can scale up by increasing the number in (m,n,k)-game. In Nim, you can scale up by increasing the number of heaps X and size of each heap Y.
  E.g., one question on scalability will be: what is the maximal size of the game variant that your agent (running in your hardware) can handle (producing a response in a reasonable amount of time, say 10 or 100 seconds?)
- Algorithm efficiency.
  You can compare minimax, alpha-beta pruning (more efficient than minimax), and varying the limit of search depth.

Report on your experiment design and results. Your hardware and software configurations should also be noted as they are very relevant to the results.

Task 2 (10 Marks)

Implement a game player for a game of your own interest. Here are the requirements.

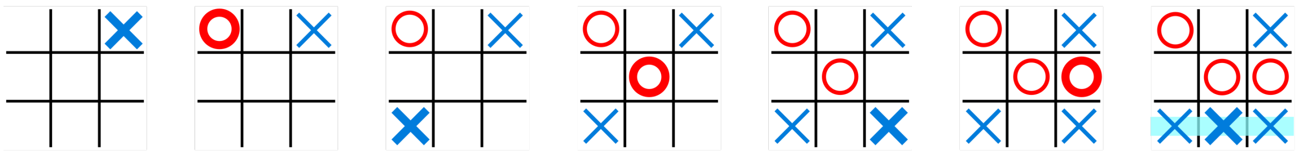
- Describe the game clearly, including the players, actions, game dynamics, terminal conditions and goals.
- Provide an evaluation.

Games

(1) Tic-Tac-Toe

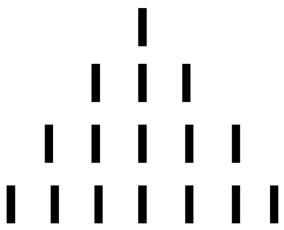
Tic-Tac-Toe is played on a three-by-three grid by two players, who alternately place the marks X and O in one of the nine spaces in the grid. The We call the first player Xplayer, and the second player Oplayer. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row is the winner. Tic-tac-toe is an instance of an (m,n,k)-game, where two players alternate taking turns on an $m \times n$ board until one of them gets k in a row.

Here is a sample play from `https://en.wikipedia.org/wiki/Tic-tac-toe`



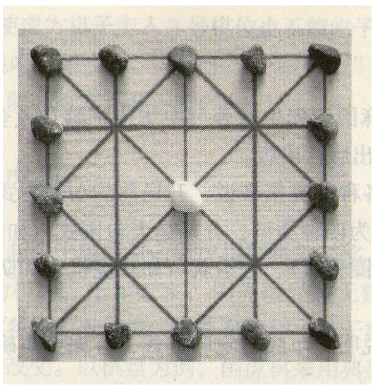
(2) Nim (1,3,5,7)

Nim (1,3,5,7) is a game played on 4 heaps (rows) with 1, 3, 5 and 7 sticks on each heap by two players who alternately remove 1 or more sticks in the same row. The player who removes the last sticks loses the game, and the other player wins the game. This game can be scaled up with more heaps and sticks.



(3) Tiger vs Dogs

Here is a very ancient game originated from China: Tiger vs. Dogs.



In the above 5×5 board, there are one tiger (represented by a white stone in the center) and 16 dogs (represented by black stones in the perimeter).

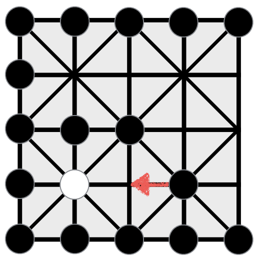
The tiger is controlled by the tiger player and the dogs are controlled by the dog player. The tiger player goes first and then they take turns. Each player can go one step along the line to an adjacent position that is not occupied.

When the tiger enters a position such that the following condition hold "two dogs are adjacent to this position such that they three are in the same line, and also these two dogs have no adjacent dogs in the same line", then these two dogs are killed by the tiger. If 6 dogs are killed, then the tiger player wins and the dog player loses.

When the dogs surrounded the tiger such that there is no unoccupied adjacent position for the tiger to move, then the tiger player loses and the dog player wins.

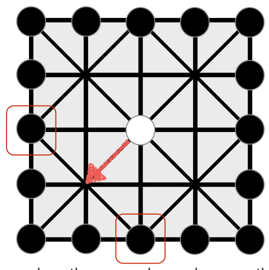
To help you to understand this game, here are a few examples:

Example:



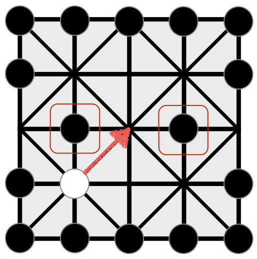
When the dog makes the move by red arrow, the tiger is surrounded.

Example:



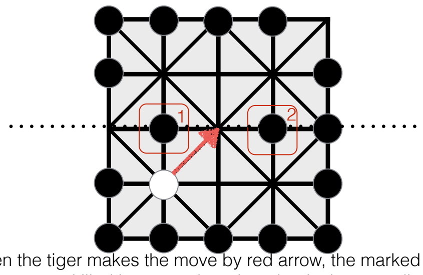
When the tiger makes the move by red arrow, the marked two dogs are killed.

Example:



When the tiger makes the move by red arrow, the marked two dogs are killed.

Example:



When the tiger makes the move by red arrow, the marked two dogs are not killed because there is a dog in the same line that is next to dog 2 (as marked).

Submission

You can team up with a classmate to work on Part 2 of this assignment. One submission per team for this part.

You are required to provide:

(1) Your solution to Base Problems (individual submission), and

(2) Your report for Task 1 (Limited to 4 A4 pages, 12 points Times New Roman),

(3) Your report for Task 2 (Limited to 2 A4 pages, 12 points Times New Roman).

(4) The source code and the compiled program of your game playing agent (with notes on what library it may depends on and instructions on how to run it). There is no restriction on your programming language. Note that the you can use some code already exists but you need to clearly state in your report and emphasise particularly your own original contribution.

We prefer that (1) in a single pdf file (file name format AI2023A1P1X.pdf, where X is your student ID), (2) and (3) are in another single PDF file (file name format AI2023A1P2BXY.pdf, where X,Y are your student ID(s)). The submission is via Canvas.