Red Blue Nim

Assignment 2

Game Playing Problem

Name - MAYANK VEKARIYA

UTA ID - 1002078999

NetID - mxv8999

Git hub link - https://github.com/bapu-1777/ai_2

The game consists of two players taking turns to remove either one red or one blue token from the board. The player who removes the last token wins the game. The program allows the user to play against the computer. The user can specify the number of red and blue tokens and the depth of the decision tree. The program implements the minimax algorithm to find the best move for the computer.

Installation

need python 3 >

To Run

To run the program, use the following command:

```
red_blue_nim.py <num-red> <num-blue> <first-player> <depth>
```

- num-red: the number of red tokens (an integer)

- num-blue: the number of blue tokens (an integer)

- first-player: the first player (either "human" or "computer")

- depth: the depth of the decision tree (an integer)

If the first-player argument is not specified, the default value is "computer". If the depth argument is not specified or is negative, the program sets the depth to the sum of the number of red and blue tokens.

During the game, the program displays the current state and the available moves. If it is the turn of the computer, the program selects the best move based on the minimax algorithm. If it is the turn of the human, the program asks the user to select a move, just need to select a or b. The user can also specify a new depth for the decision tree or 0(if not want to specify). The program ends when there are no red or blue tokens left. The program displays the winner and the number of points earned by the winner.

Functions

- children(red, blue)
This function takes two arguments, the number of red and blue tokens, and returns a list of all possible moves for the current player. Each move is a list of two elements: the number of red and blue tokens left after the move.

- future_eval(r, b, turn)
This function takes three arguments, the number of red tokens, the number of blue tokens, and a boolean turn that indicates which player is making the move. The function returns a score for the given state based on the difference between the number of red and blue tokens. If it is the turn of the computer, the function returns 2 if the difference is even and -2 if it is odd. If it is the turn of the human, the function returns 2 if the difference is even and -2 if it is odd.

- minimax(red, blue, depth, max_turn, alpha, beta)
This function implements the minimax algorithm to find the best move for the computer. It takes five arguments:

red: the number of red tokens left

blue: the number of blue tokens left

depth: the depth of the decision tree

max_turn: a boolean that indicates whether it is the turn of the computer or not

alpha: the alpha value for alpha-beta pruning

beta: the beta value for alpha-beta pruning

The function returns the best score for the current state. If the game is over (there are no red or blue tokens left), the function returns a score based on the number of remaining tokens. If the depth of the decision tree has been reached, the function returns a score based on the future_eval function. If it is the turn of the computer, the function returns the maximum score of all possible moves. If it is the turn of the human, the function returns the minimum score of all possible moves. The function also implements alpha-beta pruning to improve the performance of the minimax algorithm.

Demo

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
step 0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* num_red : 2
* num_blue : 2
* first_player : computer
* next_player : human
* tree_depth : 2
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
computer have two choices.
(a) red:1, blue 2 (b) red:2 blue:1
computer select -> red : 2, blue : 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
          |
          |
          v

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
step 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* num_red : 2
* num_blue : 1
* first_player : human
* next_player : computer
* tree_depth : 3
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
human have two choices.
(a) red:1, blue 1 (b) red:2 blue:0
select (a) or (b) : a
depth or put it 0 : 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

|
|
v

~~~~~~~~~~~~~~~~~~~~~~~~~~~~
step 2
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* num_red : 1
* num_blue : 1
* first_player : computer
* next_player : human
* tree_depth : 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
computer have two choices.
(a) red:0, blue 1 (b) red:1 blue:0
computer select -> red : 1, blue : 0
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
|
|
v

~~~~~~~~~~~~~~~~~~~~~~~~~~~~
step 3
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* num_red : 1
* num_blue : 0
* first_player : human
* next_player : computer
* tree_depth : 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
!! HUMAN WIN !! POINTS -> 2
~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Reference -
https://realpython.com/python-min-and-max/#:~:text=Use%20Python's%20min()%20and,the%20key%20and%20default%20arguments

----

Authors

- [@Mayank](https://www.hackerrank.com/mayankpv2001)