**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI**

**DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION SYSTEMS**

**Artificial Intelligence (BITS F444/ CS F407)**

**I Semester 2019-20**

**Programming Assignment-3**

**Coding Details**

**(October 17, 2019)**

*Instruction: Type the details precisely and neatly*

1. ID \_2016A7PS0150P\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name \_\_Patel\_Parth\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Mention the names of Submitted files :
   1. utils.py
   2. algo.py
   3. GUI.py
   4. coding details PA3.docx
   5. b.png
   6. g.png
2. Total number of submitted files: \_6\_\_\_\_\_\_
3. Name of the folder :\_\_2016A7PS0150P\_\_\_\_
4. Have you checked that all the files you are submitting have your name in the top?(yes/no) Yes
5. Have you checked that all the files you are submitting are in the folder as specified in 4 (and no subfolder exists)?(yes/no) Yes
6. Problem formulation:
   1. State representation: The state representation consists of following attributes:

1. Size of board: N

2. Actual board state: a bytearray of size N\*N where each byte is 0 (empty cell), 1 (green coin/ Machine M) or 2 (blue coin/ Human H).

3. Variables - game\_over and game\_util\_val, denoting whether the game is over or not and if it’s over, what is the utility value.

* 1. Pseudo code of your successor function:

state next\_state(state, action, coin){

// coin is 1 (Machine M) or 2 (Human H).

// action = (x, y) i.e. coordinates on which to place the coin.

nxt \_state = clone\_state(state)

nxt\_state.place\_coin(action, coin)

return nxt\_state

}

* 1. Terminal states generation process:

Terminal states are not enumerated and stored somewhere because it’s not a good practice as we will have anyways have to match a particular state with all possible terminal states (to know whether this state is terminal or not) and this will be more time-and-space consuming rather than checking if the state satisfies any of the conditions (3 aligned same-colored-coins or no more empty squares) to be called a terminal state ; Instead, each state is checked if its terminal state or not, before expanding the tree downward from this particular state.

* 1. Data structure to store terminal states: N/A
  2. Method to access terminal states and corresponding utility values: If we are able to identify if a state is terminal or not (by checking if it satisfies any of the terminal conditions - 3 aligned same-colored-coins or no more empty squares), then we can easily assign it a utility value.

1. Minimax Technique details
   1. Node structure: Alongwith current state information, maintain which all successor states have been visited and for which the tree has been expanded and examined. Backtrack if all successor states have been expanded and examined.

* 1. Method to ensure the correctness of terminal test (describe in maximum 4 lines): Check for 3 same-colored coins aligned vertically, horizontally, at 45o or at 135o to detect win or loss. If none of these is satisfied, check if there are more empty squares (i.e. game is not over) or not (i.e. a draw).
  2. Total number of nodes generated to play one game: O(b^m), where b is branching factor and m is max depth of tree. For example, it took 617282 nodes for one game.
  3. Write the statistics here as asked

R1 = 617282 R2 = 64 R3 = 16

R4 = 24.969320536 secs. R5= 0.024721618

* 1. Code status (implemented fully/ partially/ not done):Implemented fully.

1. Alpha Beta technique details:
   1. Explain the logic used for pruning (in maximum four lines): As utility values are -1, 0, and +1, alpha and beta are initialised to -2 and +2 respectively. Alpha-beta search updates the values of alpha and beta as it goes along and prunes the remaining branches at a node (i.e., terminates the recursive call) as soon as the value of the current node is known to be worse than the current alpha or beta value for MAX or MIN, respectively.
   2. Total number of nodes generated to play one game: Roughly, we can say O(b^(m/2)), where b is branching factor and m is max depth of tree. For example, it took 8472 nodes for one game.
   3. Write the statistics here as asked

R6 = 8472 R7 = 0.986275317 R8 = 0.37395587 secs.

* 1. Code status (implemented fully/ partially/ not done): implemented fully

1. Comparative analysis:

R9 = 1.12 R10 = 62.932871593 R11= 8 R12= 7.25

Fill in the following information based of 10 independent games

|  |  |  |
| --- | --- | --- |
|  | Minimax Algorithm | Alpha Beta Pruning |
| Average number of nodes created | 625492 | 8834 |
| Average time taken (by machine) | 25.63 secs | 0.42 secs |
| Number of times machine wins (player M) | 8/10 | 8/10 |

1. GUI details
   1. Created the GUI (yes/ No): Yes
   2. Have created it according to the specifications?(yes/No) Yes
   3. Which module of Python is used for creating graphics? PyQt5
   4. Is this under the standard Python library or not? yes
   5. If not, why? N/A
2. Graphics details:
   1. Is graphics working fine for displaying the board and coins? Yes
   2. How have you calibrated the board and accepted human input to play the game? Yes, if human clicks on any valid move/square (as specified in the document), a blue coin is placed at that square.
   3. How are you showing the base line? A maroon line shown on top of the board.
   4. How are you showing the move of the machine? Machine places a green coin (at a valid coordinate decided by it algorithm) after each move by human.
   5. How are you showing the move of the human player? Human player can click on a square to put his/her blue coin at that particular coordinate.
3. Compilation Details:
   1. Code Compiles (Yes/ No):\_Yes\_\_\_\_\_\_\_\_\_\_
   2. Mention the .py files that do not compile:\_\_N/A\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   3. Any specific function that does not compile:\_N/A\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   4. Ensured the compatibility of your code with the specified Python version(yes/no)\_\_Yes\_\_\_\_\_
   5. Instructions for compilation of your files mentioning the multi file compilation process used by you (We may use the replica of these for compiling your files while evaluating your code): On Ubuntu terminal, type: **python GUI.py**
4. Driver Details: Does it take care of the options specified earlier(yes/no):\_\_Yes\_\_\_\_
5. Execution status (describe in maximum 2 lines): All the submitted code works. The GUI is as specified - R1 to R13 are displayed, and buttons are provided to select different combinations of options (such as board size N, algorithm to be used-Minimax or AlphaBetaPruning, etc.) and play the game with the machine.
6. Declaration: I, \_\_Patel\_Parth\_\_\_\_\_\_\_\_\_\_ (name) declare that I have put my genuine efforts in creating the python code for the given programming assignment and have submitted only the code developed by me. I have not copied any piece of code from any source. If the code is found plagiarized in any form or degree, I understand that a disciplinary action as per the institute rules will be taken against me and I will accept the penalty as decided by the department of Computer Science and Information Systems, BITS, Pilani.

ID\_\_\_2016A7PS0150P\_\_\_\_\_\_\_\_\_ Name:\_\_\_Patel\_Parth\_\_\_\_\_\_\_

Date: \_\_17/10/2019\_\_

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Should not exceed three pages \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*