Programming Assignment 3: Two player Checker Game

Problem Description

A two player checker game is designed with placement of coins on triangular structures arranged as shown in Figure 1. Each player has 10 coins in the beginning. The coins appear on the board arranged randomly for both players. A player can move in at most 6 directions always following a straight line. A coin can move to its neighboring vacant circle protecting itself from being wiped out from the board by the opponent (Type 1 move). Each player can also jump after an opponent's coin in its neighborhood and wipe out the crossed over coin from the board (Type 2 move). The jump is feasible only if there is a vacant circle [Fig. 2]. Also, a player cannot jump over its own coin. The player, who wipes out more coins than those of opponent's before the termination of the game, wins. The maximum branching factor is 120 considering maximum possible 12 moves by one coin, for initial 10 coins on the board. As the coins are wiped out, the branching factor reduces.

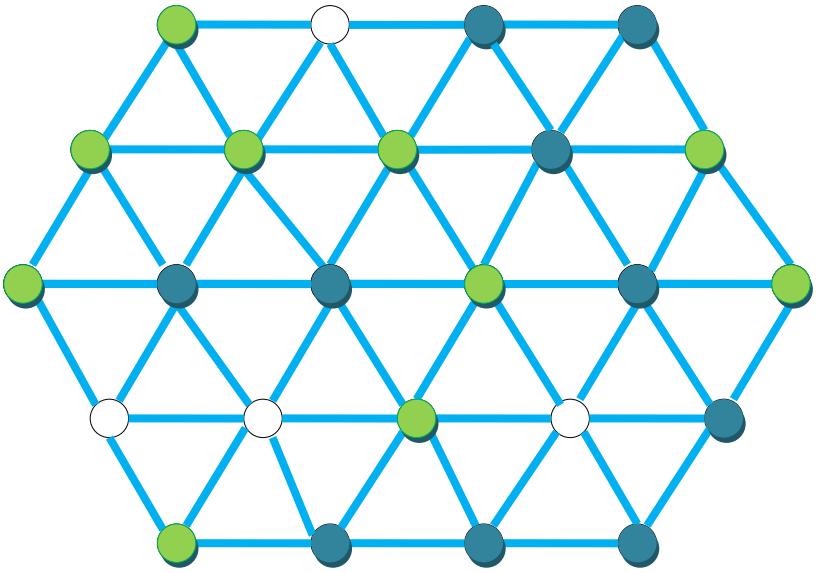
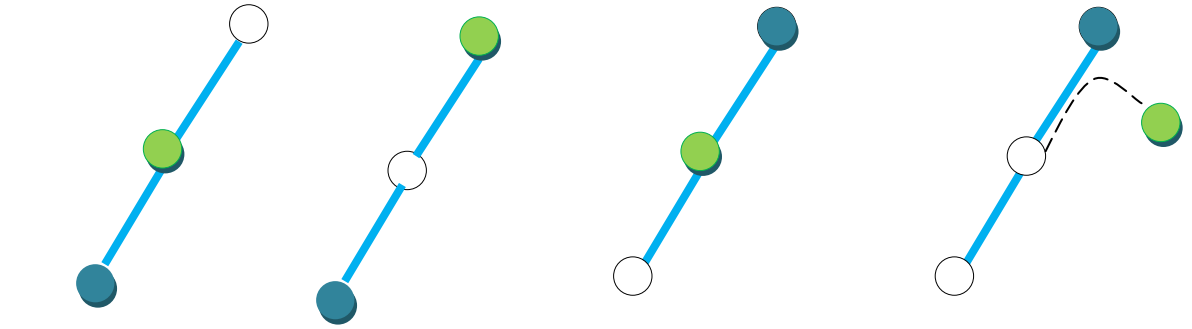


Figure 1: Initial board arrangement

The game terminates if one player's all coins are wiped out (making the other player win with points equal to the number of coins on board) or both players are not able to move anywhere in two consecutive turns. In the case, no player is able to move any coin in two consecutive turns, the player with more coins on board wins by the difference in points compared with the one

whose less coins remain on the board. The terminating states are those with exclusively one, two, ...or ten coins of player 1 in any combination. Similarly of player 2 with exclusively one, two, ...or ten coins of player 2 in any combination. Also, the terminating states are the states when no player is able to move in two consecutive turns. This also involves all possible combinations appropriately.



(a) Before a move is made

(b) After a move is made by green coin to a neighboring vacant circle (Type 1 move)

(c) After a move is made by the blue coin on the state shown in (a) using jump (Type 2 move)

(d) The vacant space in the middle after using jump as in (c). The green coin is wiped out

Figure 2. Moves illustration

The corners are safe as no coin can jump over opponent's coins placed in any of these 6 corners as depicted using red circles in Figure 3. However, any coin placed in the corner is free to move in other directions.

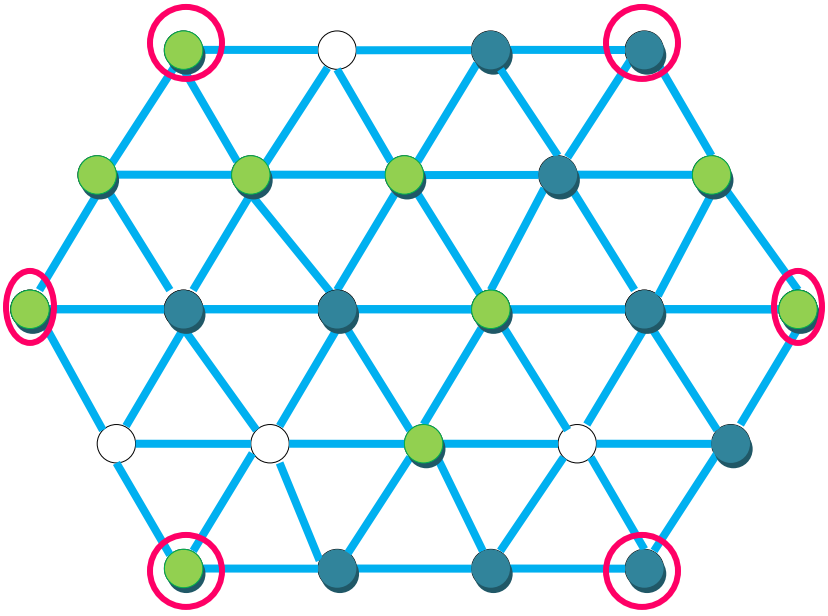


Figure 3: Safe corners (in red circles)

Represent the problem as state space search problem to compute a strategy for the intelligent agent (player 2: Intelligent agent) to win. A strategy consists of finding the initial move for player 2, next move of player 2 based on a move by player 1, next move of player 2 based on another move by player 1, and so on till the end of the game. Students must understand that if the board is not represented as a state and the problem is not represented as state space search problem, then no credit will be given mere on solving the problem. Identify the terminal states and associate utility values as discussed in the class. Implement Minimax algorithm and Alpha Beta pruning to solve the above problem.

Implementation

Use Python version 2.7.13 (Windows 10) for implementing your solution. Only standard Python libraries should be used. Support from external sources or libraries such as github will not be accepted in your submissions. Each student must design own solution and write own code. [Refer handout to understand the malpractice policies.]

Modules

Implement the following modules. The names are self explanatory and refer to the class discussions.

1. Initial state generator()
2. Successor_function(state s) returns a state.
3. Terminal_test(Hashtable H, state s) returns a boolean value. [Hint:Use hashing]
4. Utility_value(Hashtable H, state s) returns a number. [Hint: Use hashing for O(1) time complexity]
5. MIN_VALUE(state s) returns utility_value.
6. MAX_VALUE(state s) returns utility_value.
7. Minimax algorithm
8. Alpha Beta pruning algorithm along with its Min_value() and Max_value () functions equivalent.

You are free to write functions to generate combinations of the terminal states to be populated in the hashtable H as above while you are also free to work out the combinations manually and populate the hash table appropriately.

Graphics

The game is expected to be played by the machine with a human player. Produce the graphics as shown in figures above, calibrate the upper left corner of the board and use cursor position to pick up the move by human player, use that as input to generate the strategy for player 2's next move till the termination of the game. Turtle/ Tkinter graphics is used to display the board and the coins. ***You must first display graphically the moves by intelligent agent as you compute the strategy and then show the data R1-R12 as asked below.*** Also illustrate the WIN or LOSS of player 2 (the intelligent agent) appropriately and the points with which it wins, or illustrate the draw on the screen. Prompt the player 1 (human) for a new game once the previous game is over.

If the player 1 wishes not to play, exit. List all the computed values of R1- R12 on the left hand side of the board.

Analysis Module

Produce the following analyses and display the resultant values.

(a) Minimax algorithm based analysis

i. Compute the number of nodes generated till the problem is solved. **[R1]**
ii. Compute the amount of memory allocated to one node. **[R2]**
iii. Compute the maximum growth of the implicit stack (if recursion is used) or of explicit stack used with the search tree. **[R3]**
iv. Compute the total time to play the game. **[R4]**
v. Compute the number of nodes created in one micro second **[R5]**

(b) Alpha Beta pruning based analysis

i. Compute the number of nodes generated till the problem is solved. **[R6]**
ii. Compute the ratio (R1 - R6)/R1 as saving using pruning. **[R7]**
iii. Compute the total time to play a game. **[R8]**

(c) Comparative analysis

i. Compare the memory used in both the techniques (Minimax and Alpha Beta pruning). **[R9]**
ii. Play the game 10 times and compute average time to play the game **[R10]**
iii. Play the game 10 times and compute the number of times player 2 wins **[R11]**
iv. Repeat (iii) 20 times and compute the average number of times player 2 wins. **[R12]**
v. Compare R4 and R8.

Driver

The driver must integrate all functionalities and execute the functions appropriately using these options

Option 1: Display the empty board
Option 2: Play the game using Minimax algorithm
Option 3: Play the game using Alpha Beta pruning
Option 4: Show all results R1-R12.

Writeup, evaluation and submission

Write up details will be made available one day before the submission date. Evaluation will be out of 16 marks (8% weight). Students are advised to inform me immediately if any discrepancy exists in this document. The assignment is due for submission on October 20, 2018 (Saturday) by

6:30 p.m. The total duration of the assignment is 10 days after excluding the mid semester test duration. Students are advised to work out the details of the solution, discuss with me their complete understanding of the concepts individually and plan the implementation appropriately.

In case of any difficulty, please feel free to contact me.

Vandana
October 1, 2018