Davis Janechokpinyo
109006201
周志偉

Report

In this project, we have to implement the method five_in_a line. To make sure that there are five stones in a consecutive way either horizontally, vertically or diagonally , then a player will win. A Monte Carlo method is a computerized mathematical technique that allows people to quantitatively account for risk in forecasting and decision-making. At its core, the Monte Carlo method is a way to use random samples of parameters to explore the behavior of a complex system. A Monte Carlo Tree Search (MCTS) is a search technique in the field of Artificial Intelligence (AI). It is a probabilistic and heuristic driven search algorithm that combines the classic tree search implementations alongside machine learning principles of reinforcement learning. Every action might not be the most optimal for example when we can win just by 5 moves it takes 10 moves. MCTS algorithm is sometimes useful as it continues to evaluate other alternatives periodically during the learning phase by executing them, instead of the current perceived optimal strategy.

Inside def IsFiveInLine(self, x, y), there will be two if statements one for when player one moves and one for player two move. For each player horizontally and vertically will be controlled by self.board[i][j]. In this case i will be for consecutive horizontal pieces and j will be for vertical.

```
if self.board[i][j] == 2 and self.board[i][j + 1] == 2 and self.board[i][j + 2] == 2 and self.board[i][j + 3] == 2 and self.board[i][j + 4] == 2:
    return 1
if self.board[j][i] == 2 and self.board[j + 1][i] == 2 and self.board[j + 2][i] == 2 and self.board[j + 3][i] == 2 and self.board[j + 4][i] == 2:
    return 1
```

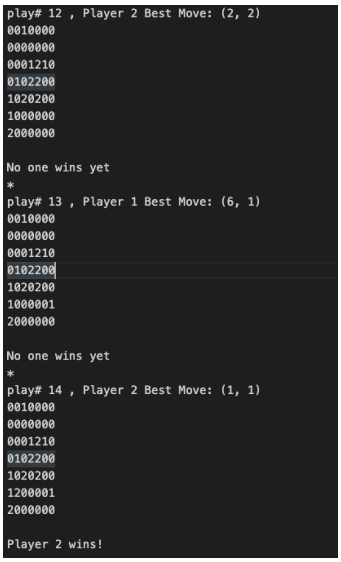
This is for the diagonal.

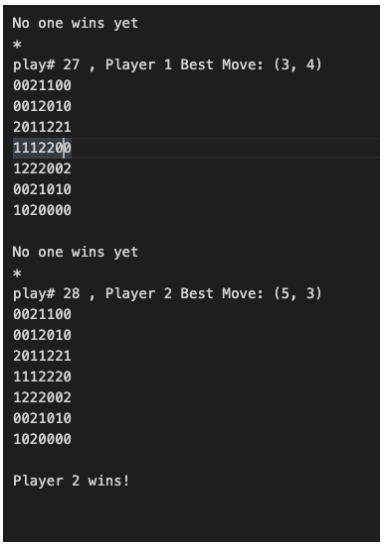
```
r i in range(3):
  for j in range(4,7):
      if self.board[i][j] == 2 and self.board[i + 1][j - 1] == 2 and self.board[i + 2][j - 2] == 2 and self.board[i + 3][j - 3] == 2 and self.board[i + 4
          return 1
r i in range(3):
  for j in range(3):
      if self.board[i][j] == 2 and self.board[i + 1][j + 1] == 2 and self.board[i + 2][j + 2] == 2 and self.board[i + 3][j + 3] == 2 and self.board[i + 4
          return 1
```

I

The range can't be starting from zero like horizontal and vertical since as much as you are going down from the top, is as much as you are going to the left or right.

RESULT





Program Performance

I think that the level of performance of the game is very low. Since not every move is optimal, there are many times where both players can win with the better move. However the player picks the other one and it takes much more rounds, that doesn't really make any sense. This can be improved by using other heuristic algorithms that can make the player smarter. By implementing double-free three-in-line, four-in-line-missing-one, free-three-in-line can also make the program better and more efficient. Since it can distinguish the good and bad moves.