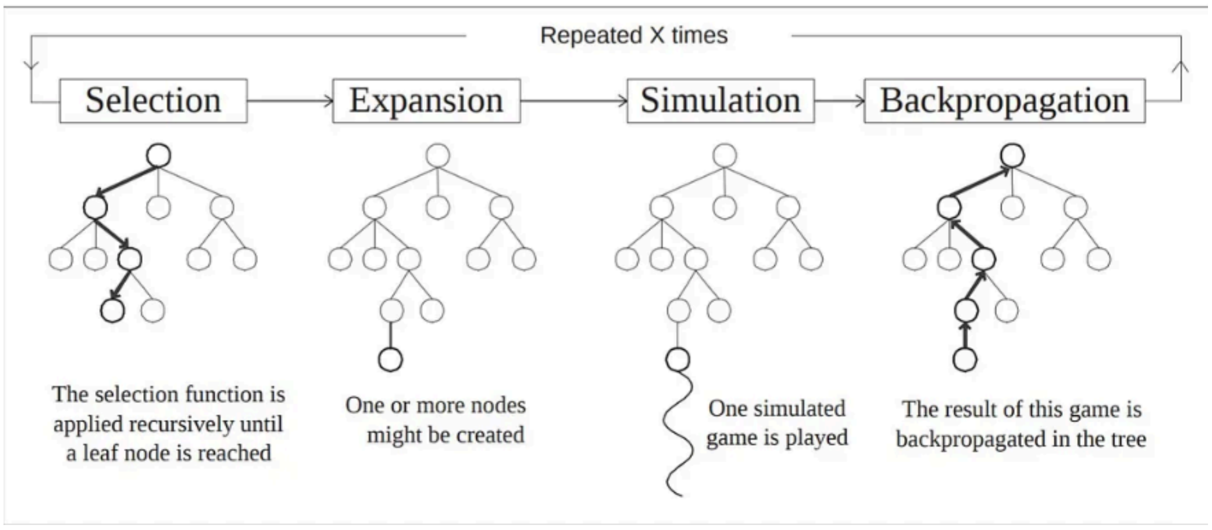
PSA Final Project Report

Submitted By:
Omkar Vaity : 002836207
Mihir Adelkar : 002810839

Introduction:

MONTE CARLO TREE SEARCH -

Monte Carlo tree search (MCTS) is a general game-playing algorithm to find the best move from any given game state of any game.


Repeated X times
Selection
Expansion
Simulation
Backpropagation
The selection function is applied recursively until a leaf node is reached
One or more nodes might be created
One simulated game is played
The result of this game is backpropagated in the tree

MCTS works in the following steps -

1) **Selection**: The algorithm begins at the root node and selects subsequent child nodes until a leaf node is reached. The selection is based on a policy that trades off between exploration of new nodes and exploitation of nodes with high win rates. A common policy used is the Upper Confidence bound applied to Trees (UCT), which considers both the total value of each node and the number of visits to each node.
2) **Expansion**: In this phase, one or more children are added to expand the search tree, based on the possible moves from the current game state. Expansion occurs until a preset state space limit is reached or the game reaches a conclusion.
3) **Simulation**: Starting from the new nodes, the algorithm simulates random gameplays to the end, using either a random policy or a light-weight default policy. The objective is to estimate the value of the new nodes based on the outcomes of these simulations.
4) **Backpropagation**: The results of the simulation are propagated back through the tree. During backpropagation, the algorithm updates the nodes with the results of the simulated endgame, improving the estimated win rate of the nodes.

These steps are repeated for a large number of iterations, often determined by computational constraints such as time or memory. After completing these iterations, MCTS selects the move associated with the child node of the root with the highest win rate.

IMPLEMENTATION:

1) TIC-TAC-TOE

The Monte Carlo Tree Search (MCTS) algorithm is particularly well-suited for turn-based games like Tic Tac Toe. The algorithm's exploration of potential future moves allows it to make informed decisions at each turn. Here's how MCTS is implemented for Tic Tac Toe:

Selection:
The selection phase begins at the root of the tree, which represents the current state of the Tic Tac Toe board. From there, the algorithm selects child nodes down the tree, representing potential future moves. It uses the Upper Confidence bound applied to Trees (UCT) to balance exploration of less-visited nodes (to discover potentially stronger strategies) with exploitation of nodes that have historically led to wins. The algorithm proceeds down the tree until it reaches a node that represents an unexplored move (a leaf node).

Expansion:
Once an unexplored node is reached, the expansion phase begins. If the node corresponds to a non-terminal state (the game is not over), one or more child nodes are added to the tree, representing the possible moves that could follow. In the context of Tic Tac Toe, this would involve adding a node for each empty cell on the board that could be filled by the current player's symbol (either 'X' or 'O').

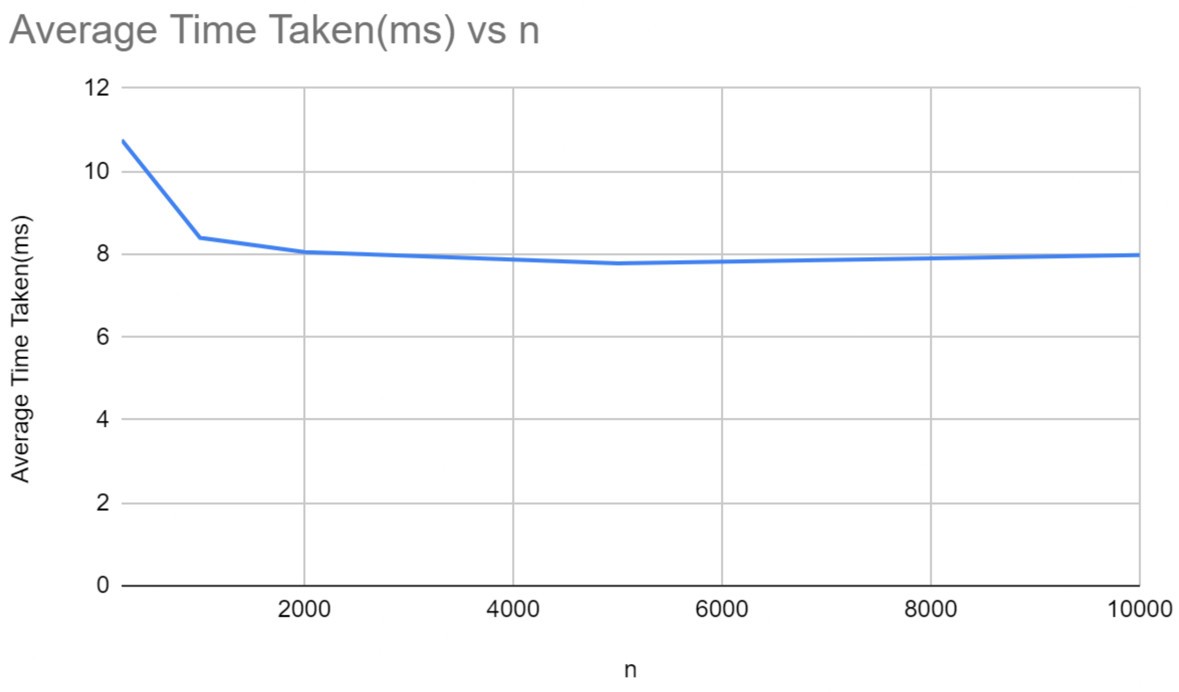
Simulation:
From the newly expanded nodes, the algorithm simulates random games of Tic Tac Toe until a terminal state is reached — either a win, a loss, or a draw. The simulations may be completely random, with each player placing their symbol in a random empty spot, or they may follow a heuristic to more closely approximate optimal play.

Backpropagation:
After a simulation reaches its conclusion, the results are backpropagated up the tree. Each node along the path from the simulated node up to the root node is updated with the outcome of the simulation. This typically involves updating statistics such as the number of wins and the number of simulations for each node.

The MCTS algorithm repeats these four phases for a predetermined number of iterations or until a certain amount of computational time has elapsed. After the final iteration, the algorithm selects the move associated with the child node of the root that has the highest win ratio. This move is then played on the actual Tic Tac Toe board.

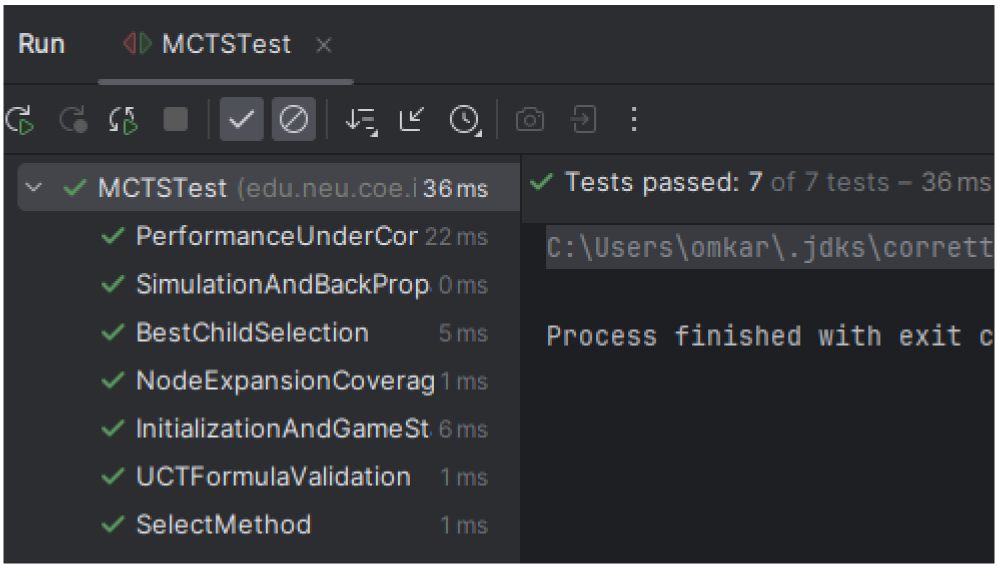
Results Table -

n	Average Time Taken(ms)
250	10.756
1000	8.395
2000	8.049
5000	7.7756
10000	7.9796



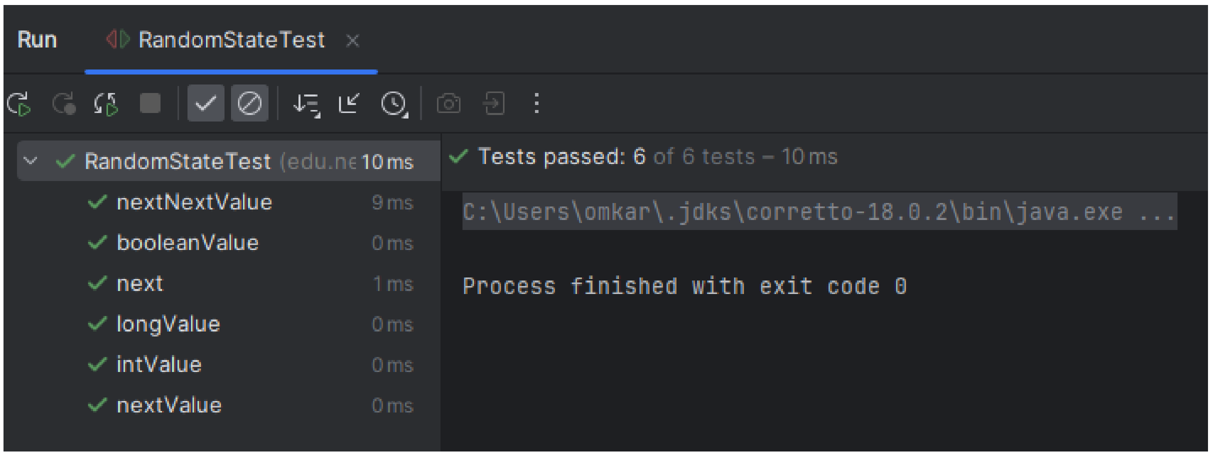
The graph above illustrates the average duration required for the Monte Carlo Tree Search (MCTS) algorithm to secure a victory in a Tic-Tac-Toe game. It compares this duration against the number of iterations the algorithm completes.

UNIT TEST Result Images -



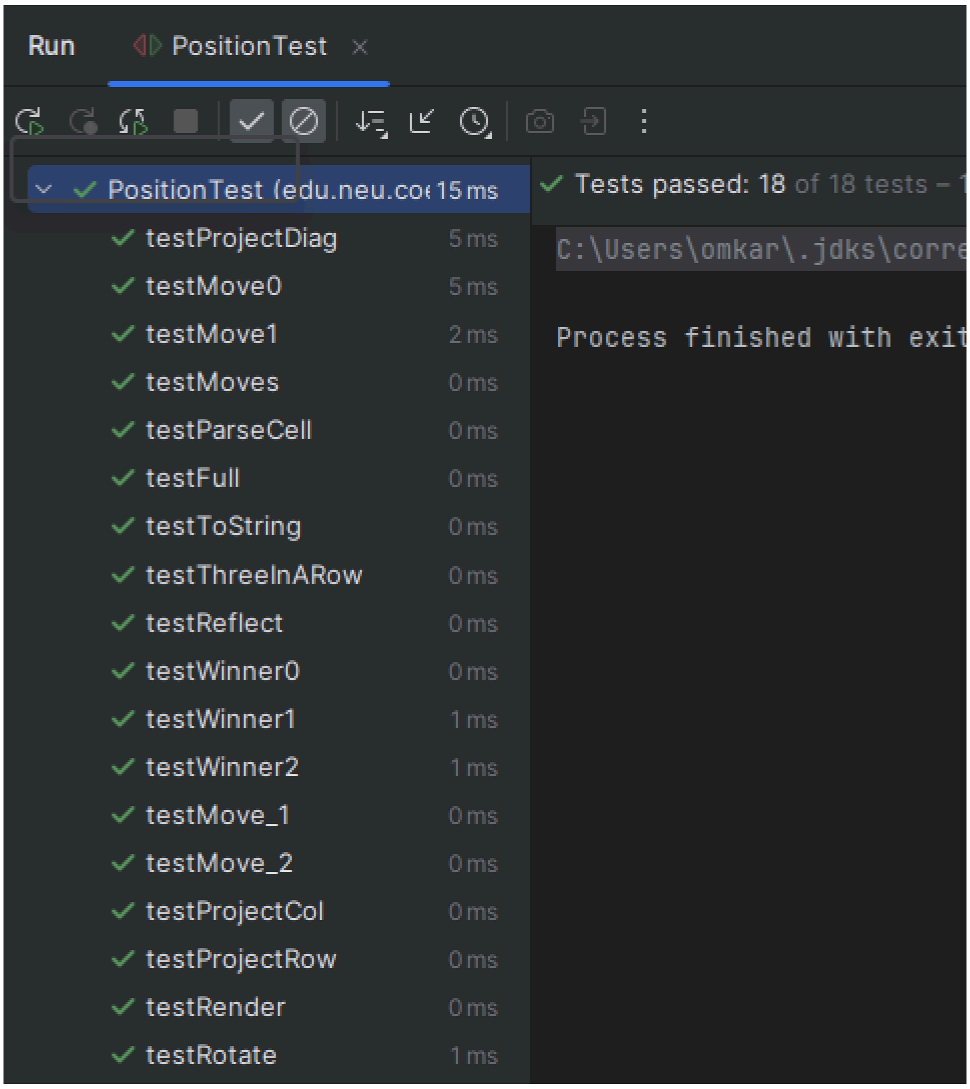
```
✓ MCTSTest (edu.neu.coe.i 36 ms
    ✓ PerformanceUnderCor 22 ms
    ✓ SimulationAndBackProp 0 ms
    ✓ BestChildSelection 5 ms
    ✓ NodeExpansionCoverag 1 ms
    ✓ InitializationAndGameSt 6 ms
    ✓ UCTFormulaValidation 1 ms
    ✓ SelectMethod 1 ms
```

```
✓ Tests passed: 7 of 7 tests – 36 ms
C:\Users\omkar\.jdks\corrett

Process finished with exit c
```



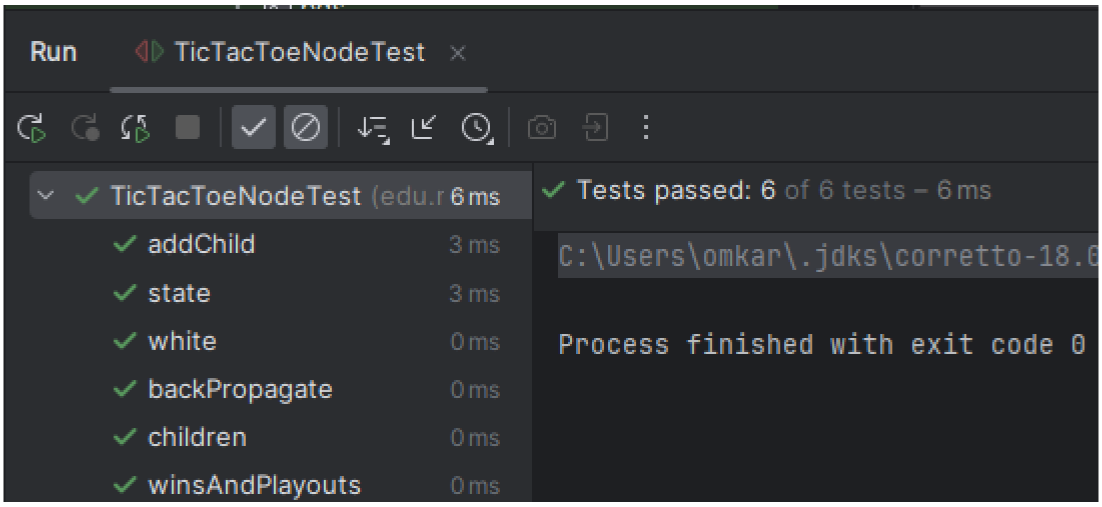
```
✓ RandomStateTest (edu.ne 10 ms
    ✓ nextNextValue 9 ms
    ✓ booleanValue 0 ms
    ✓ next 1 ms
    ✓ longValue 0 ms
    ✓ intValue 0 ms
    ✓ nextValue 0 ms
```

```
✓ Tests passed: 6 of 6 tests – 10 ms
C:\Users\omkar\.jdks\corretto-18.0.2\bin\java.exe ...

Process finished with exit code 0
```



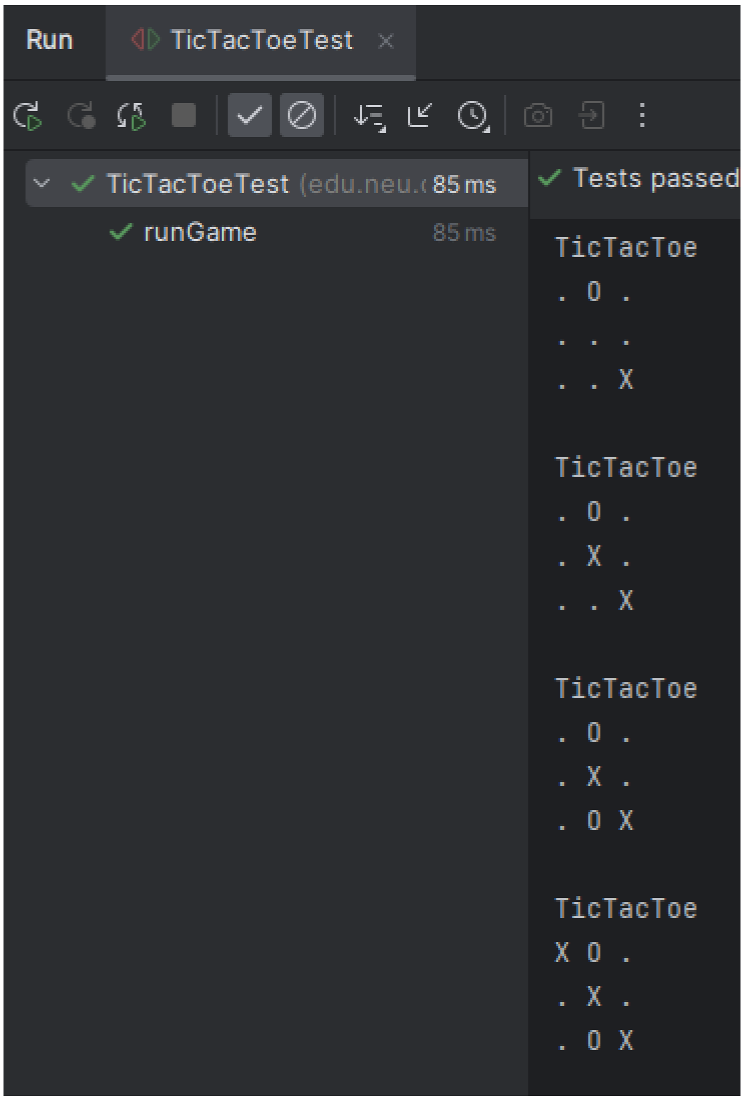
```
✓ PositionTest (edu.neu.co 15 ms
    ✓ testProjectDiag     5 ms
    ✓ testMove0           5 ms
    ✓ testMove1           2 ms
    ✓ testMoves           0 ms
    ✓ testParseCell       0 ms
    ✓ testFull            0 ms
    ✓ testToString        0 ms
    ✓ testThreeInARow     0 ms
    ✓ testReflect         0 ms
    ✓ testWinner0         0 ms
    ✓ testWinner1         1 ms
    ✓ testWinner2         1 ms
    ✓ testMove_1          0 ms
    ✓ testMove_2          0 ms
    ✓ testProjectCol      0 ms
    ✓ testProjectRow      0 ms
    ✓ testRender          0 ms
    ✓ testRotate          1 ms
```

```
✓ Tests passed: 18 of 18 tests – 1

C:\Users\omkar\.jdks\corre

Process finished with exit
```



```
✓ TicTacToeNodeTest (edu.r 6 ms
    ✓ addChild            3 ms
    ✓ state               3 ms
    ✓ white               0 ms
    ✓ backPropagate       0 ms
    ✓ children            0 ms
    ✓ winsAndPlayouts     0 ms
```

```
✓ Tests passed: 6 of 6 tests – 6 ms

C:\Users\omkar\.jdks\corretto-18.0

Process finished with exit code 0
```

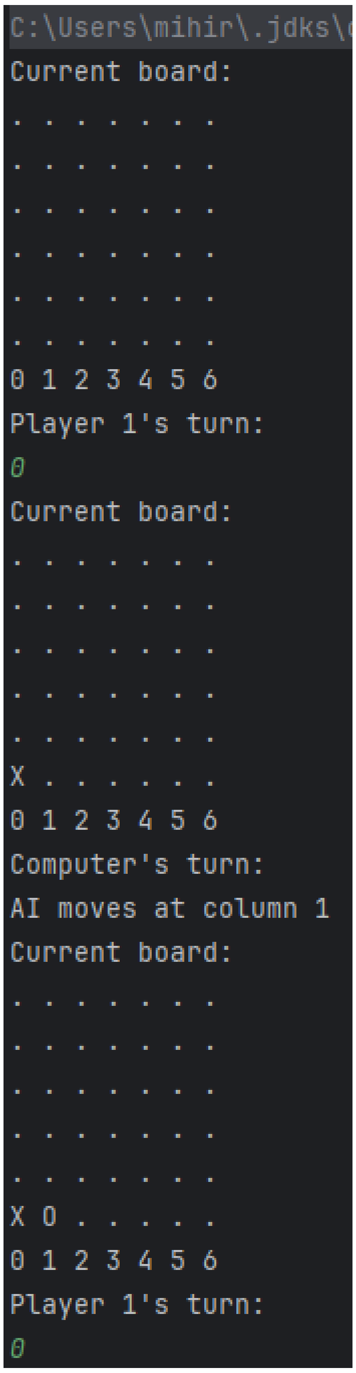


```
TicTacToe
. O .
. . .
. . X

TicTacToe
. O .
. X .
. . X

TicTacToe
. O .
. X .
. O X

TicTacToe
X O .
. X .
. O X
```

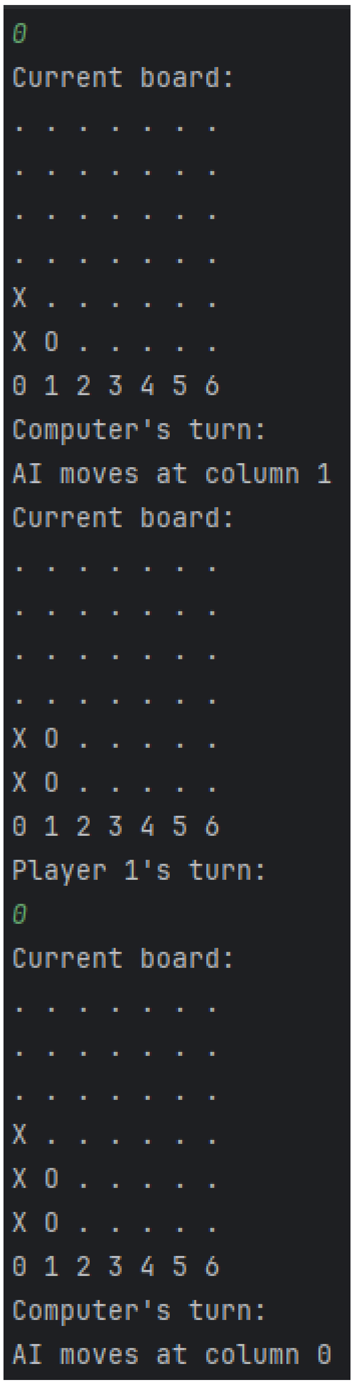
2) Connect Four :

Connect Four is a two-player connection game in which the players first choose a colour and then take turns dropping coloured discs into a seven-column, six-row vertically suspended grid. The pieces fall straight down, occupying the lowest available space within the column. The objective of the game is to be the first to form a horizontal, vertical, or diagonal line of four of one's own discs.

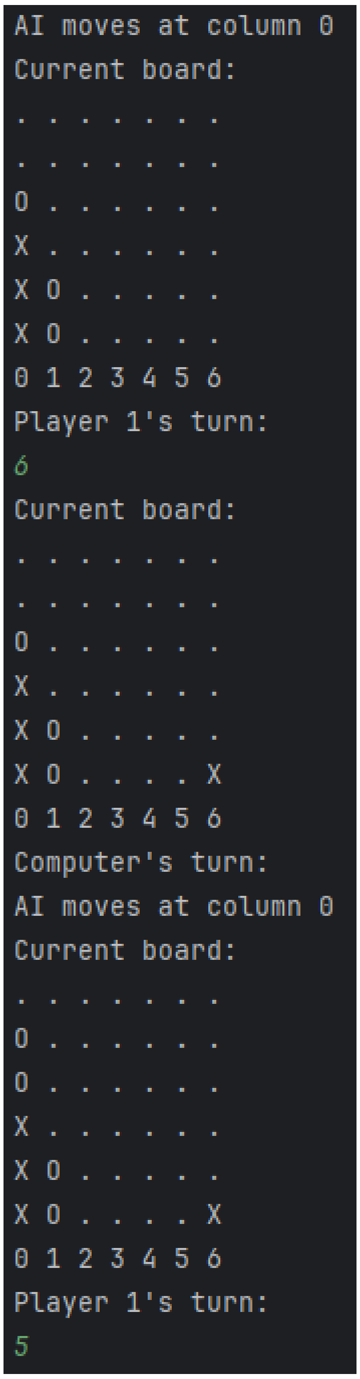
GAME SIMULATION -



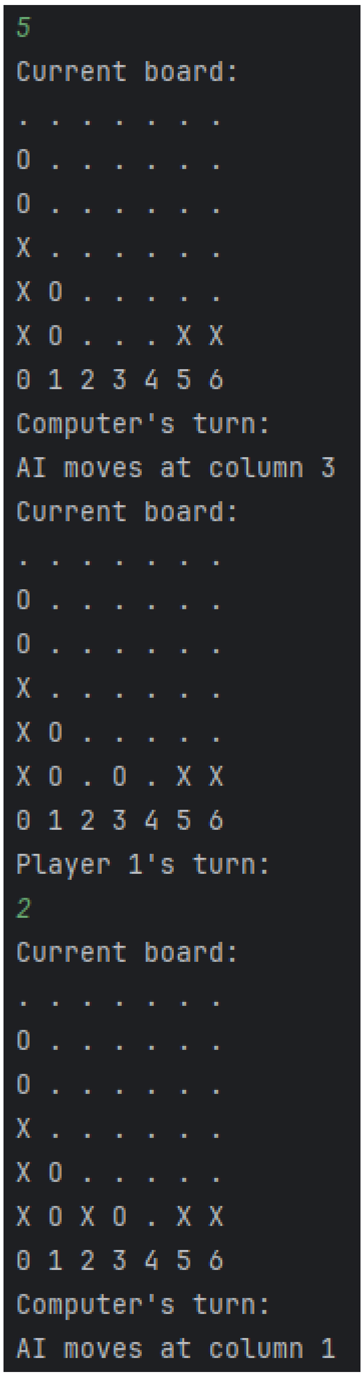
```
C:\Users\mihir\.jdks\
Current board:
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
0 1 2 3 4 5 6
Player 1's turn:
0
Current board:
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
X . . . . . .
0 1 2 3 4 5 6
Computer's turn:
AI moves at column 1
Current board:
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
X O . . . . .
0 1 2 3 4 5 6
Player 1's turn:
0
```



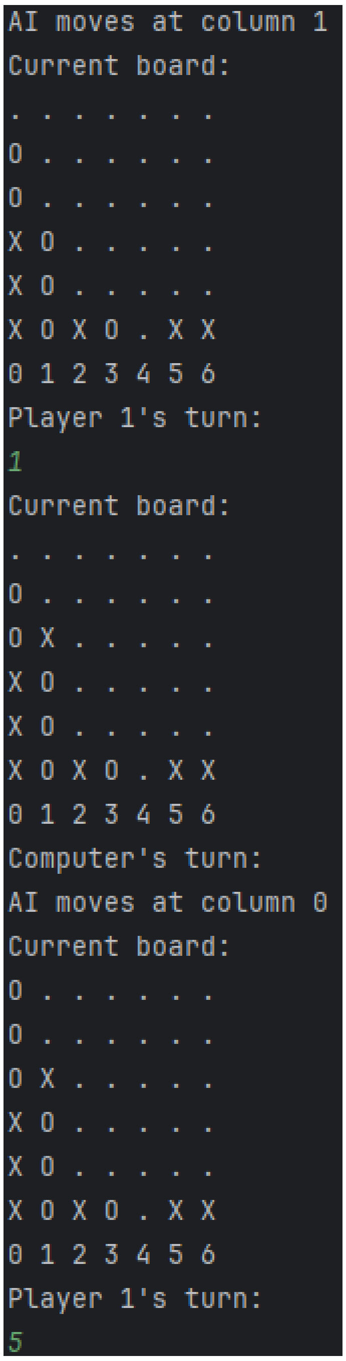
```
0
Current board:
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
X . . . . . .
X O . . . . .
0 1 2 3 4 5 6
Computer's turn:
AI moves at column 1
Current board:
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
X O . . . . .
X O . . . . .
0 1 2 3 4 5 6
Player 1's turn:
0
Current board:
. . . . . . .
. . . . . . .
. . . . . . .
X . . . . . .
X O . . . . .
X O . . . . .
0 1 2 3 4 5 6
Computer's turn:
AI moves at column 0
```



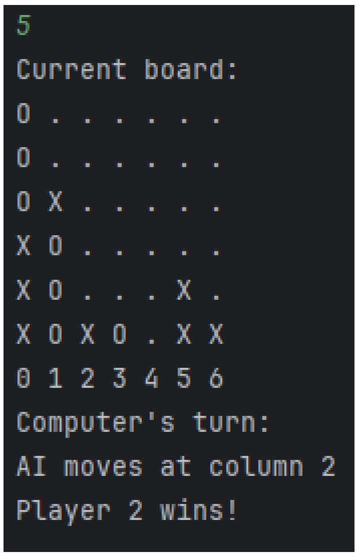
```
AI moves at column 0
Current board:
. . . . . . .
. . . . . . .
O . . . . . .
X . . . . . .
X O . . . . .
X O . . . . .
0 1 2 3 4 5 6
Player 1's turn:
6
Current board:
. . . . . . .
. . . . . . .
O . . . . . .
X . . . . . .
X O . . . . .
X O . . . . X
0 1 2 3 4 5 6
Computer's turn:
AI moves at column 0
Current board:
. . . . . . .
O . . . . . .
O . . . . . .
X . . . . . .
X O . . . . .
X O . . . . X
0 1 2 3 4 5 6
Player 1's turn:
5
```



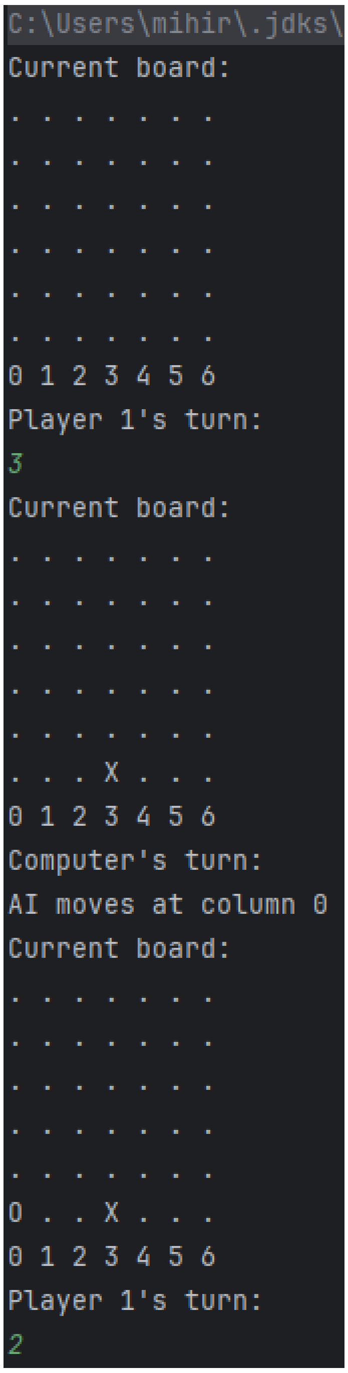
```
5
Current board:
. . . . . . .
O . . . . . .
O . . . . . .
X . . . . . .
X O . . . . .
X O . . . X X
0 1 2 3 4 5 6
Computer's turn:
AI moves at column 3
Current board:
. . . . . . .
O . . . . . .
O . . . . . .
X . . . . . .
X O . . . . .
X O . O . X X
0 1 2 3 4 5 6
Player 1's turn:
2
Current board:
. . . . . . .
O . . . . . .
O . . . . . .
X . . . . . .
X O . . . . .
X O X O . X X
0 1 2 3 4 5 6
Computer's turn:
AI moves at column 1
```



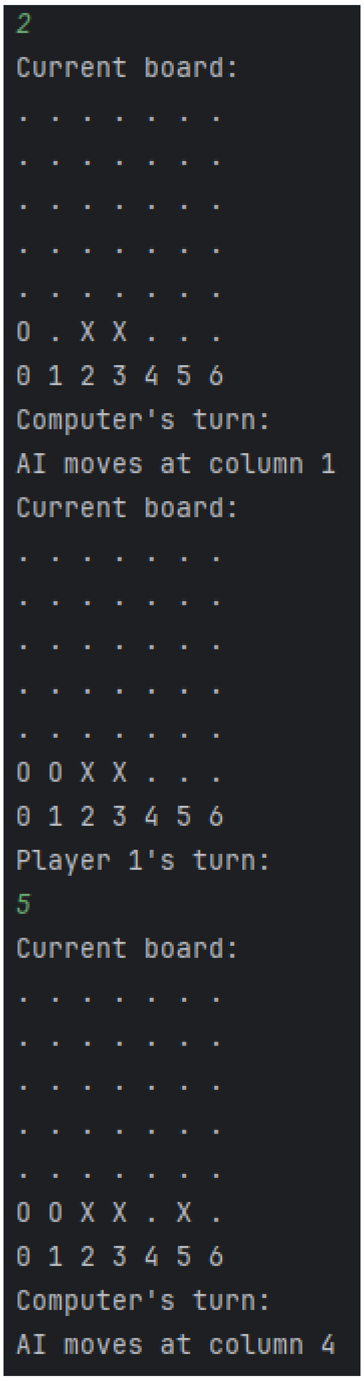
```
AI moves at column 1
Current board:
. . . . . . .
O . . . . . .
O . . . . . .
X O . . . . .
X O . . . . .
X O X O . X X
0 1 2 3 4 5 6
Player 1's turn:
1
Current board:
. . . . . . .
O . . . . . .
O X . . . . .
X O . . . . .
X O . . . . .
X O X O . X X
0 1 2 3 4 5 6
Computer's turn:
AI moves at column 0
Current board:
O . . . . . .
O . . . . . .
O X . . . . .
X O . . . . .
X O . . . . .
X O X O . X X
0 1 2 3 4 5 6
Player 1's turn:
5
```



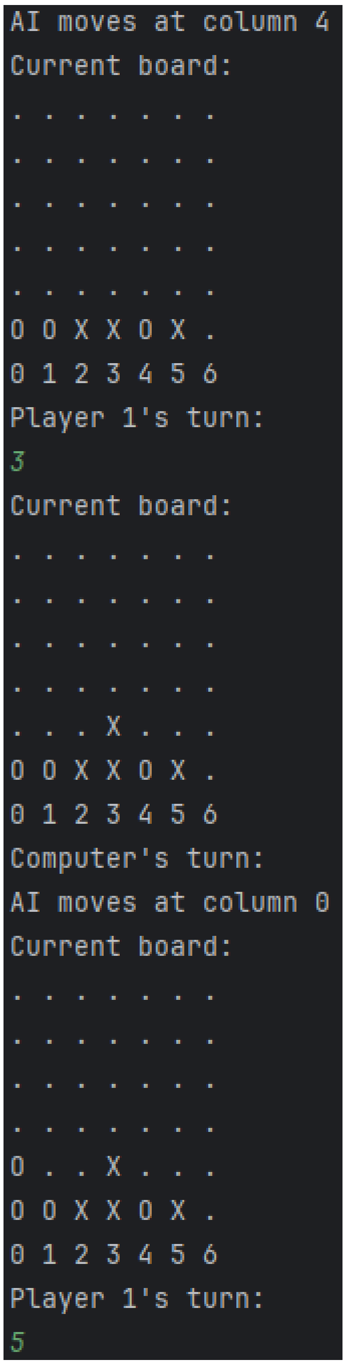
```
5
Current board:
O . . . . . .
O . . . . . .
O X . . . . .
X O . . . . .
X O . . . X .
X O X O . X X
0 1 2 3 4 5 6
Computer's turn:
AI moves at column 2
Player 2 wins!
```



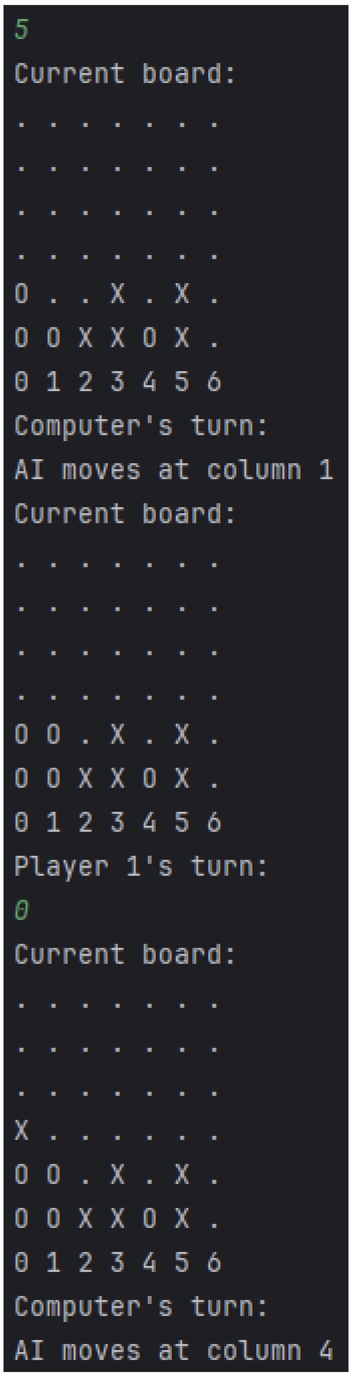
```
C:\Users\mihir\.jdks\
Current board:
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
0 1 2 3 4 5 6
Player 1's turn:
3
Current board:
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . X . . .
0 1 2 3 4 5 6
Computer's turn:
AI moves at column 0
Current board:
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
O . . X . . .
0 1 2 3 4 5 6
Player 1's turn:
2
```



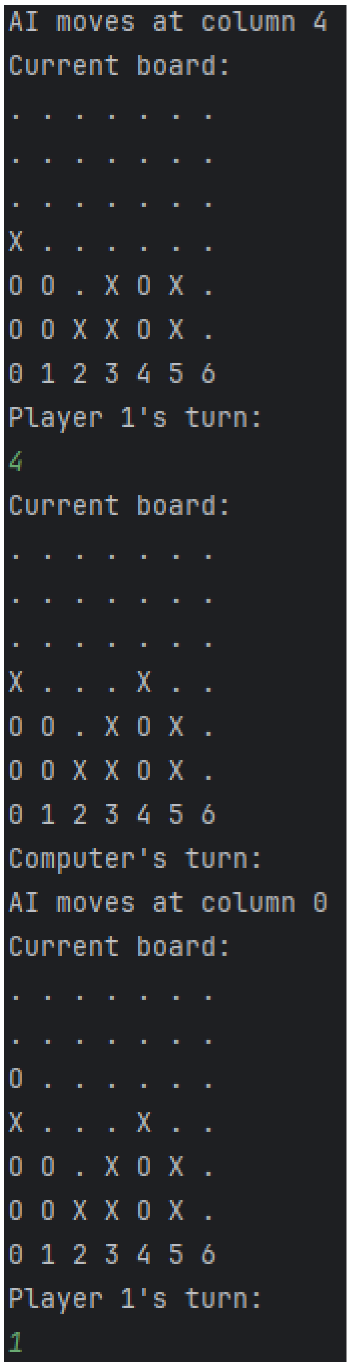
```
2
Current board:
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
O . X X . . .
0 1 2 3 4 5 6
Computer's turn:
AI moves at column 1
Current board:
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
O O X X . . .
0 1 2 3 4 5 6
Player 1's turn:
5
Current board:
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
O O X X . X .
0 1 2 3 4 5 6
Computer's turn:
AI moves at column 4
```



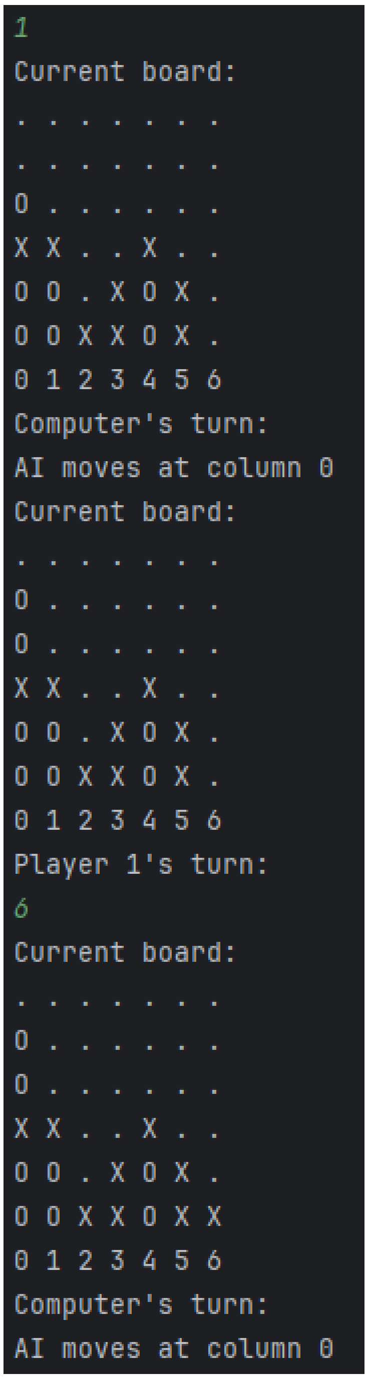
```
AI moves at column 4
Current board:
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
O O X X O X .
0 1 2 3 4 5 6
Player 1's turn:
3
Current board:
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
. . . X . . .
O O X X O X .
0 1 2 3 4 5 6
Computer's turn:
AI moves at column 0
Current board:
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
O . . X . . .
O O X X O X .
0 1 2 3 4 5 6
Player 1's turn:
5
```



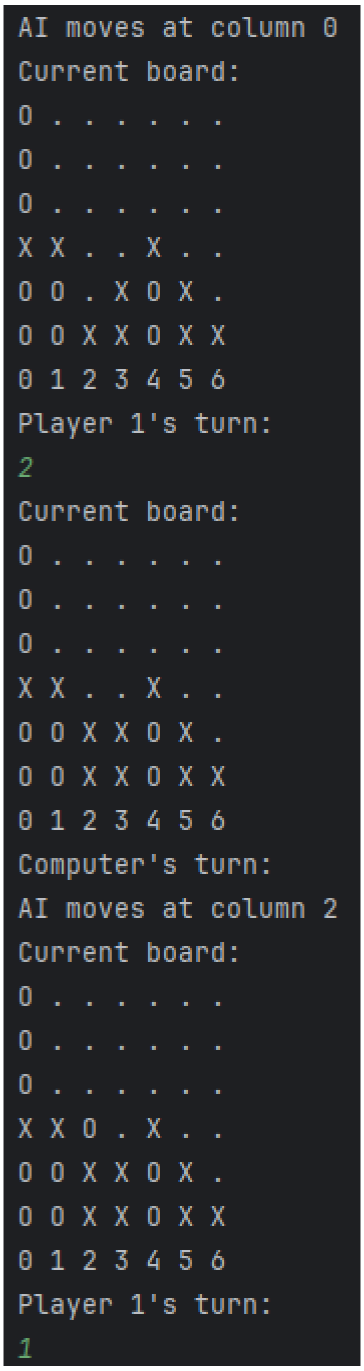
```
5
Current board:
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
O . . X . X .
O O X X O X .
0 1 2 3 4 5 6
Computer's turn:
AI moves at column 1
Current board:
. . . . . . .
. . . . . . .
. . . . . . .
. . . . . . .
O O . X . X .
O O X X O X .
0 1 2 3 4 5 6
Player 1's turn:
0
Current board:
. . . . . . .
. . . . . . .
. . . . . . .
X . . . . . .
O O . X . X .
O O X X O X .
0 1 2 3 4 5 6
Computer's turn:
AI moves at column 4
```



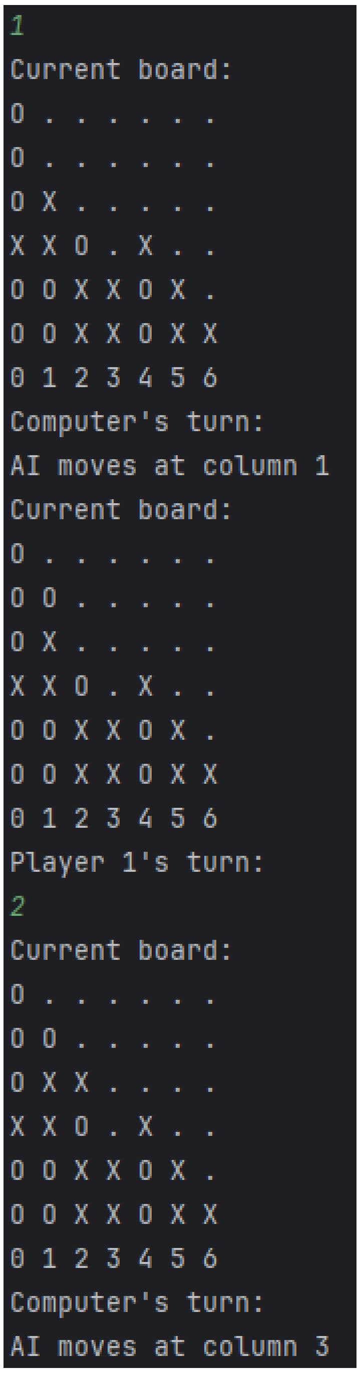
```
AI moves at column 4
Current board:
. . . . . . .
. . . . . . .
. . . . . . .
X . . . . . .
O O . X O X .
O O X X O X .
0 1 2 3 4 5 6
Player 1's turn:
4
Current board:
. . . . . . .
. . . . . . .
. . . . . . .
X . . . X . .
O O . X O X .
O O X X O X .
0 1 2 3 4 5 6
Computer's turn:
AI moves at column 0
Current board:
. . . . . . .
. . . . . . .
O . . . . . .
X . . . X . .
O O . X O X .
O O X X O X .
0 1 2 3 4 5 6
Player 1's turn:
1
```



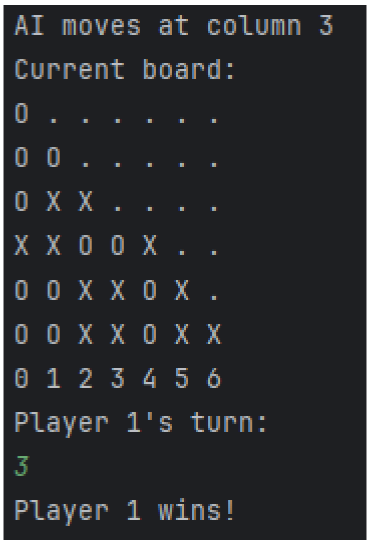
```
1
Current board:
. . . . . . .
. . . . . . .
O . . . . . .
X X . . X . .
O O . X O X .
O O X X O X .
0 1 2 3 4 5 6
Computer's turn:
AI moves at column 0
Current board:
. . . . . . .
O . . . . . .
O . . . . . .
X X . . X . .
O O . X O X .
O O X X O X .
0 1 2 3 4 5 6
Player 1's turn:
6
Current board:
. . . . . . .
O . . . . . .
O . . . . . .
X X . . X . .
O O . X O X .
O O X X O X X
0 1 2 3 4 5 6
Computer's turn:
AI moves at column 0
```



```
AI moves at column 0
Current board:
O . . . . . .
O . . . . . .
O . . . . . .
X X . . X . .
O O . X O X .
O O X X O X X
0 1 2 3 4 5 6
Player 1's turn:
2
Current board:
O . . . . . .
O . . . . . .
O . . . . . .
X X . . X . .
O O X X O X .
O O X X O X X
0 1 2 3 4 5 6
Computer's turn:
AI moves at column 2
Current board:
O . . . . . .
O . . . . . .
O . . . . . .
X X O . X . .
O O X X O X .
O O X X O X X
0 1 2 3 4 5 6
Player 1's turn:
1
```

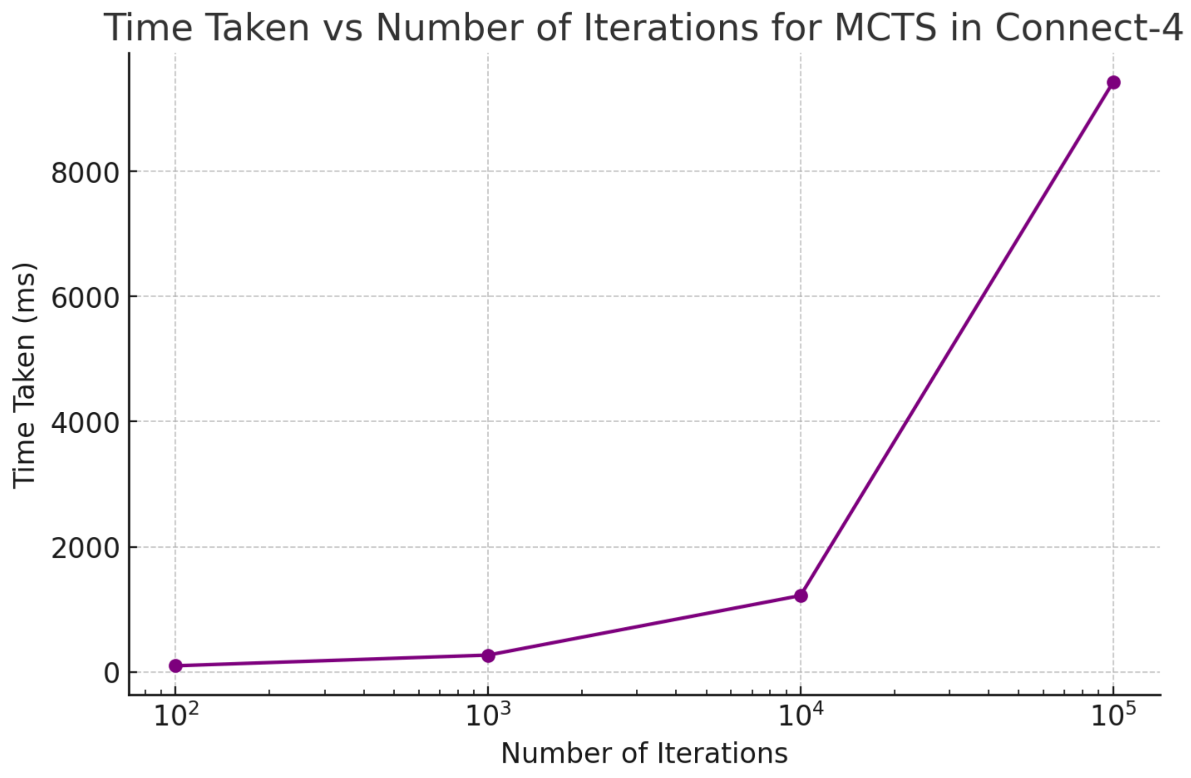


```
1
Current board:
O . . . . . .
O . . . . . .
O X . . . . .
X X O . X . .
O O X X O X .
O O X X O X X
0 1 2 3 4 5 6
Computer's turn:
AI moves at column 1
Current board:
O . . . . . .
O O . . . . .
O X . . . . .
X X O . X . .
O O X X O X .
O O X X O X X
0 1 2 3 4 5 6
Player 1's turn:
2
Current board:
O . . . . . .
O O . . . . .
O X X . . . .
X X O . X . .
O O X X O X .
O O X X O X X
0 1 2 3 4 5 6
Computer's turn:
AI moves at column 3
```



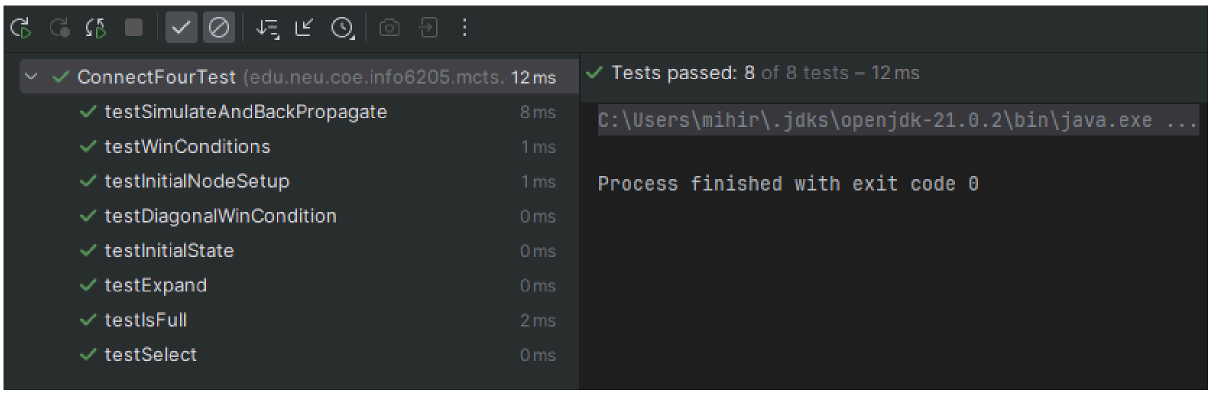
```
AI moves at column 3
Current board:
O . . . . . .
O O . . . . .
O X X . . . .
X X O O X . .
O O X X O X .
O O X X O X X
0 1 2 3 4 5 6
Player 1's turn:
3
Player 1 wins!
```

Results -



Here's the graph showing the time taken for MCTS to consider all moves to win a Connect-4 game, plotted against the number of iterations. The time taken increases significantly as the number of iterations increases, with a logarithmic scale on the x-axis to better illustrate the relationship across a wide range of iteration counts.

UNIT TEST Result Images -



```
✓ ConnectFourTest (edu.neu.coe.info6205.mcts. 12 ms
    ✓ testSimulateAndBackPropagate      8 ms
    ✓ testWinConditions                 1 ms
    ✓ testInitialNodeSetup              1 ms
    ✓ testDiagonalWinCondition          0 ms
    ✓ testInitialState                  0 ms
    ✓ testExpand                        0 ms
    ✓ testIsFull                        2 ms
    ✓ testSelect                        0 ms
```

```
✓ Tests passed: 8 of 8 tests – 12 ms

C:\Users\mihir\.jdks\openjdk-21.0.2\bin\java.exe ...

Process finished with exit code 0
```