

## Cairo University Faculty of Computers and Artificial Intelligence Computer Science Department



**CS361: Artificial Intelligence Connect 4 Al agent Project.** 

Submitted to:

**Prof. Khaled Wassif** 

Submitted by:

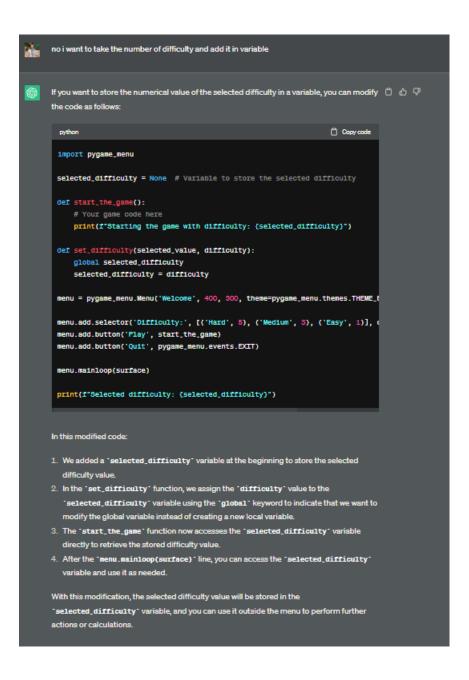
Ahmed Mohamed Sallam	20210614
Omar Walid Ahmed	20201126
Mohamed Gamal Abdelaziz	20201146
Ahmed Hany	20200055

## • GitHub Repo:

ahmedm-sallam/AI-Connect4-Game-Agent (github.com)

## • LinkedIn Post :

https://www.linkedin.com/posts/ahmedm-sallam\_artificialintelligence-boardgames-aiagent-activity-7065071740229419009-DyQu?utm\_source=share&utm\_medium=member\_desktop





The code provided is a Connect 4 game implemented in Python using the Pygame library. It allows the player to play against an Al opponent. The Al opponent uses the minimax algorithm with alpha-beta pruning to make its moves.

0 4

To measure the performance of the two algorithms and create a graph to compare them, you can track the following metrics:

- Execution time: Measure the time it takes for each algorithm to make a move. You can use
  the 'time' module in Python to record the start and end times for each move and
  calculate the difference.
- Number of nodes explored: Count the number of nodes explored by each algorithm during the search process. This can give you an idea of the search space and the algorithm's efficiency.
- 3. Win rate: Run multiple games between the AI and the player using both algorithms and keep track of the number of wins for each algorithm. This will give you an indication of their effectiveness in winning the game.

Once you have collected the data for each metric, you can create a graph to compare the performance of the two algorithms. You can use Python libraries such as Matplotlib or Seaborn to plot the data and visualize the comparison.



Mesure Execution time for this code



To measure the execution time of the code, you can use the 'time' module in Python. Here's  $\Box$   $\Box$   $\Box$   $\Box$  an example of how you can modify the code to measure the execution time:

```
import time

# ...

def start_the_game():
    print(f"the difficulty = {selected_difficulty}")

    start_time = time.time()
    # the GUI of the board open when RunGame start
    RunGame(selected_difficulty)
    # Do the job here !
    end time = time.time()
```

```
# Do the job here !
end_time = time.time()

execution_time = end_time - start_time
print(f"Execution time: {execution_time} seconds")
# ...
```

By adding the `start\_time` and `end\_time` variables and calculating the difference between them, you can obtain the execution time in seconds. The `print` statement displays the execution time at the end of the game.

Remember to import the 'time' module at the beginning of your code for this to work.



Count the number of nodes explored by each algorithm during the search process. in this



To count the number of nodes explored by each algorithm during the search process in the Connect 4 game code, you can mo Regenerate response m implementation. Here's an example of how you can do it: