



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



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

Submitted to :

**Prof. Khaled Wassif**

Submitted by:

**Ahmed Mohamed Sallam**  
**Omar Walid Ahmed**  
**Mohamed Gamal Abdelaziz**  
**Ahmed Hany**

**20210614**  
**20201126**  
**20201146**  
**20200055**

- GitHub Repo :

[ahmedm-sallam/AI-Connect4-Game-Agent \(github.com\)](https://github.com/ahmedm-sallam/AI-Connect4-Game-Agent)

- LinkedIn Post :

[https://www.linkedin.com/posts/ahmedm-sallam\\_artificialintelligence-boardgames-aiagent-activity-7065071740229419009-DyQu?utm\\_source=share&utm\\_medium=member\\_desktop](https://www.linkedin.com/posts/ahmedm-sallam_artificialintelligence-boardgames-aiagent-activity-7065071740229419009-DyQu?utm_source=share&utm_medium=member_desktop)

no i want to take the number of difficulty and add it in variable

If you want to store the numerical value of the selected difficulty in a variable, you can modify the code as follows:

```
python
Copy code

import pygame_menu

selected_difficulty = None # Variable to store the selected difficulty

def start_the_game():
    # Your game code here
    print(f"Starting the game with difficulty: {selected_difficulty}")

def set_difficulty(selected_value, difficulty):
    global selected_difficulty
    selected_difficulty = difficulty

menu = pygame_menu.Menu('Welcome', 400, 300, theme=pygame_menu.themes.THEME_1)

menu.add.selector('Difficulty:', [('Hard', 5), ('Medium', 3), ('Easy', 1)], callback=set_difficulty)
menu.add.button('Play', start_the_game)
menu.add.button('Quit', pygame_menu.events.EXIT)

menu.mainloop(surface)

print(f"Selected difficulty: {selected_difficulty}")
```

In this modified code:

1. We added a `selected_difficulty` variable at the beginning to store the selected difficulty value.
2. In the `set_difficulty` function, we assign the `difficulty` value to the `selected_difficulty` variable using the `global` keyword to indicate that we want to modify the global variable instead of creating a new local variable.
3. The `start_the_game` function now accesses the `selected_difficulty` variable directly to retrieve the stored difficulty value.
4. After the `menu.mainloop(surface)` line, you can access the `selected_difficulty` variable and use it as needed.

With this modification, the selected difficulty value will be stored in the `selected_difficulty` variable, and you can use it outside the menu to perform further actions or calculations.



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



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

1. 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.
2. 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.

© 2023 OpenAI. All rights reserved.



Mesure Execution time for this code



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

python

Copy code

```
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 code



To count the number of nodes explored by each algorithm during the search process in the Connect 4 game code, you can modify the code to include a counter variable. Here's an example of how you can do it:

Regenerate response