Assignment 2: Artificial intelligence.

Group 2392. Pair 1.

Kevin de la Coba and Juan Luis Sanz.

1. Minimax and alpha-beta pruning
   1. **Implementation**
      1. **Which tests have been designed and applied to determine whether the implementation is correct?**

In the demos of TicTacToe and Reversi, we created players which were using the MinimaxAlphaBetaStrategy and the minimaxStrategy. First, we did matches between the minimaxStrategy players, then we did the same matches with the MinimaxAlphaBetaStrategy. We came to the conclusion both strategies should choose the same movements, but the ones of alpha-beta should be faster.

So, we checked that the movements were always the same.

* + 1. **Design.**

We followed the already implemented minimaxStrategy class design and we used the assignment pseudocode for the MinimaxAlphaBetaStrategy. The differences between these 2 strategies pseudocodes are the pruning and “alpha” and “beta” values, so our design focused on implementing the pruning and naming variables correctly for the min\_value and max\_value functions.

* + 1. **Implementation.**

We tried to follow as strictly as we could the pseudocode given. We can see that the algorithm is the same until the pruning, so we copied the minimax implementation until that part and then we implemented the pruning. Small changes were made on that copy.

To be more specific, in the function next\_move, we created the variables alpha and beta, and we initialized them to -∞, ∞ respectively. As we are the max, we have to update the alpha by checking the successor’s min.

In the min\_value and max\_value functions we have to update beta and alpha respectively and for the pruning we must compare if the biggest value taken is bigger than beta (max\_value case) and if the smallest value taken is smaller than alpha (min\_value case).

* 1. **Efficiency of alpha-beta pruning.**
     1. **Complete description of the evaluation protocol.**

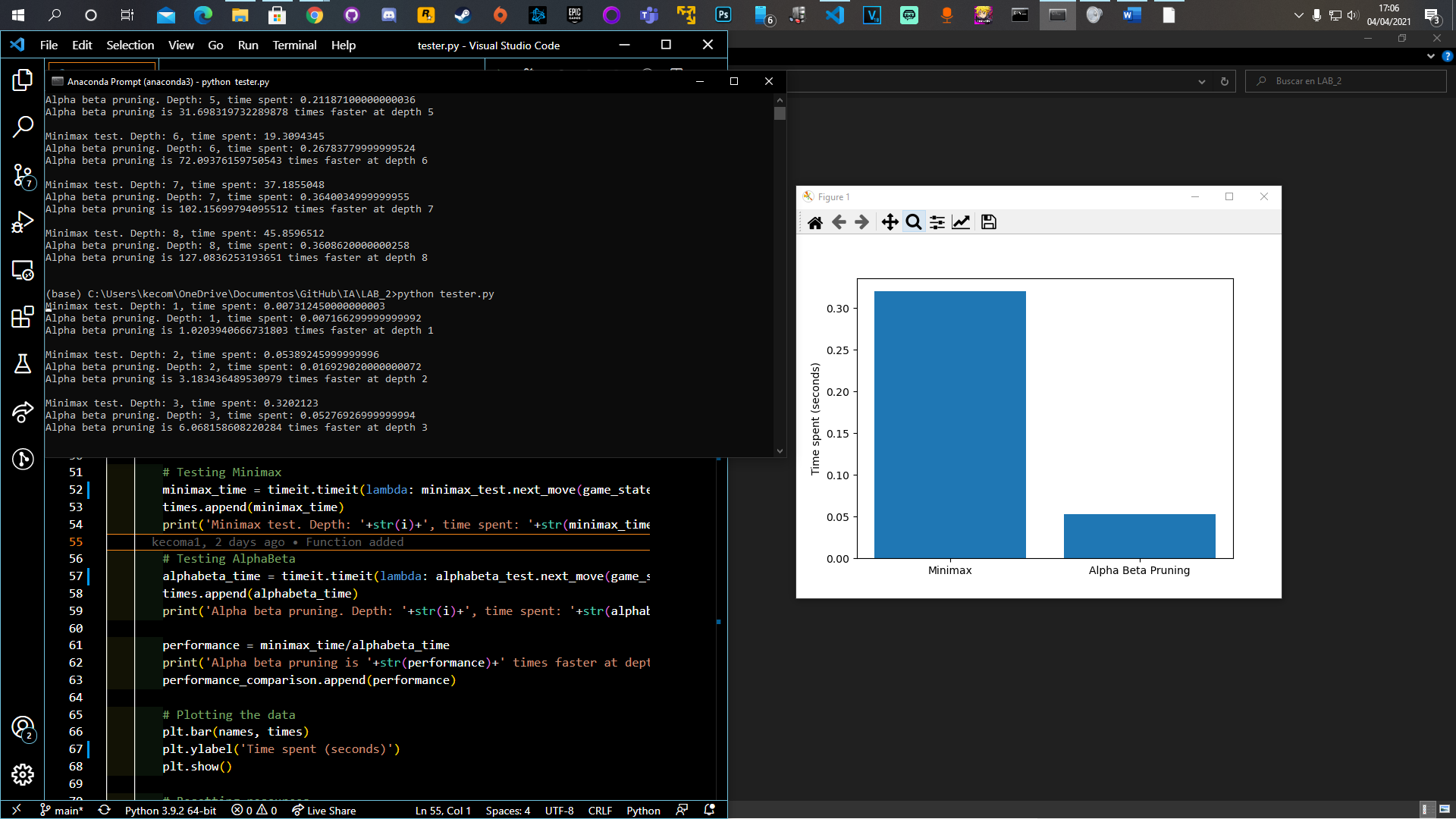
We created the file tester.py, in this one we first define the heuristic used for testing and then we buil a game state from the TicTacToe game in which the board is empty. We chose this state (empty board state) because is the one in which can choose more movements. Once we built the game state, we started testing the strategies using the timeit library.

We test each strategy 10 times, from depths 1 to 4. After each depth, we plot a histogram with the time spent by minimaxStrategy and MinimaxAlphaBetaStrategy. After all the depths, we plot how many times is alpha-beta faster than minimax in each depth. This last test is the computer independent test.

* + 1. **Tables in which times with and without pruning are reported.**

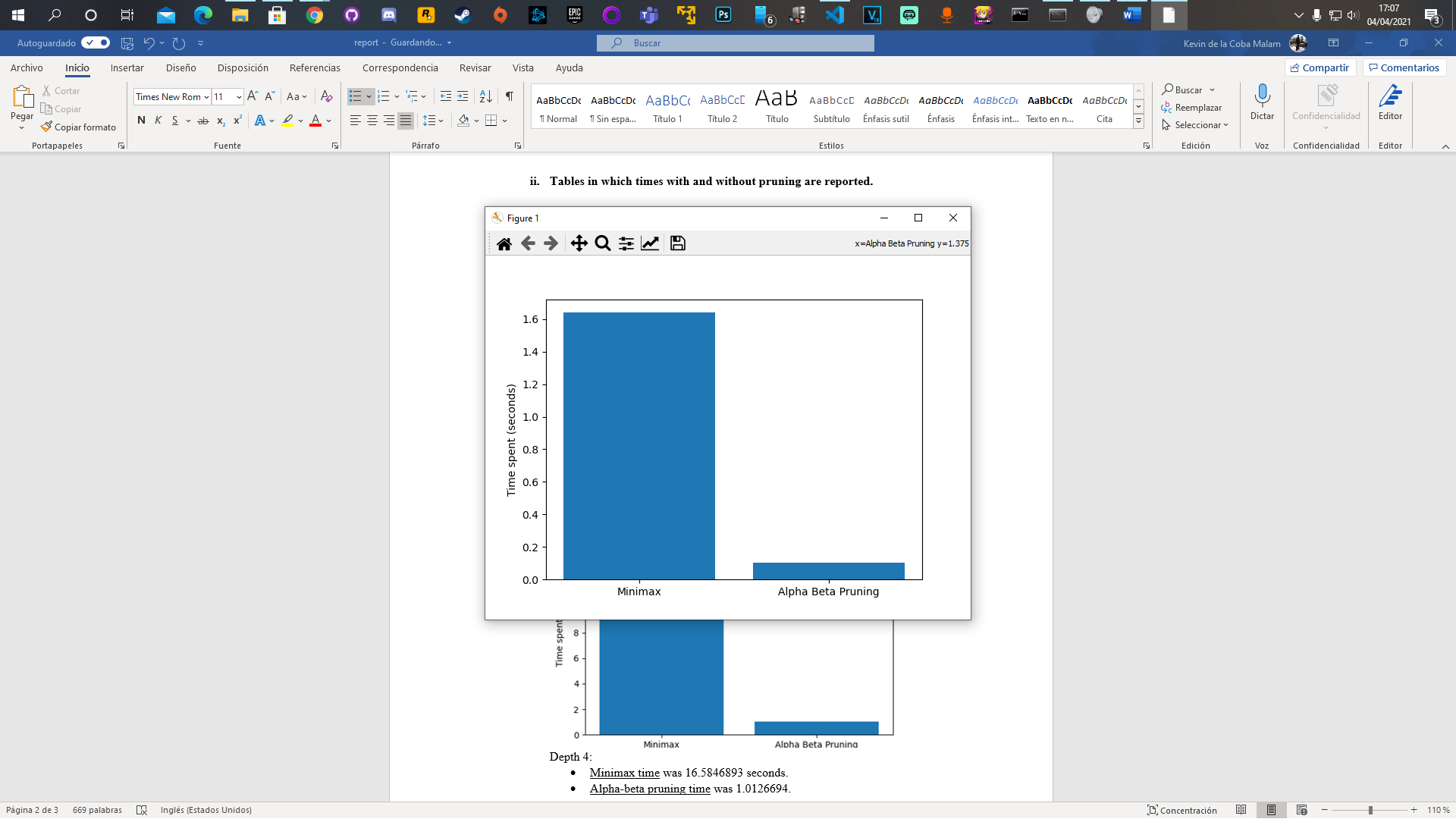
In the following histograms we can see both strategies and the time spent by each one in the Y axis.

Depth 3:



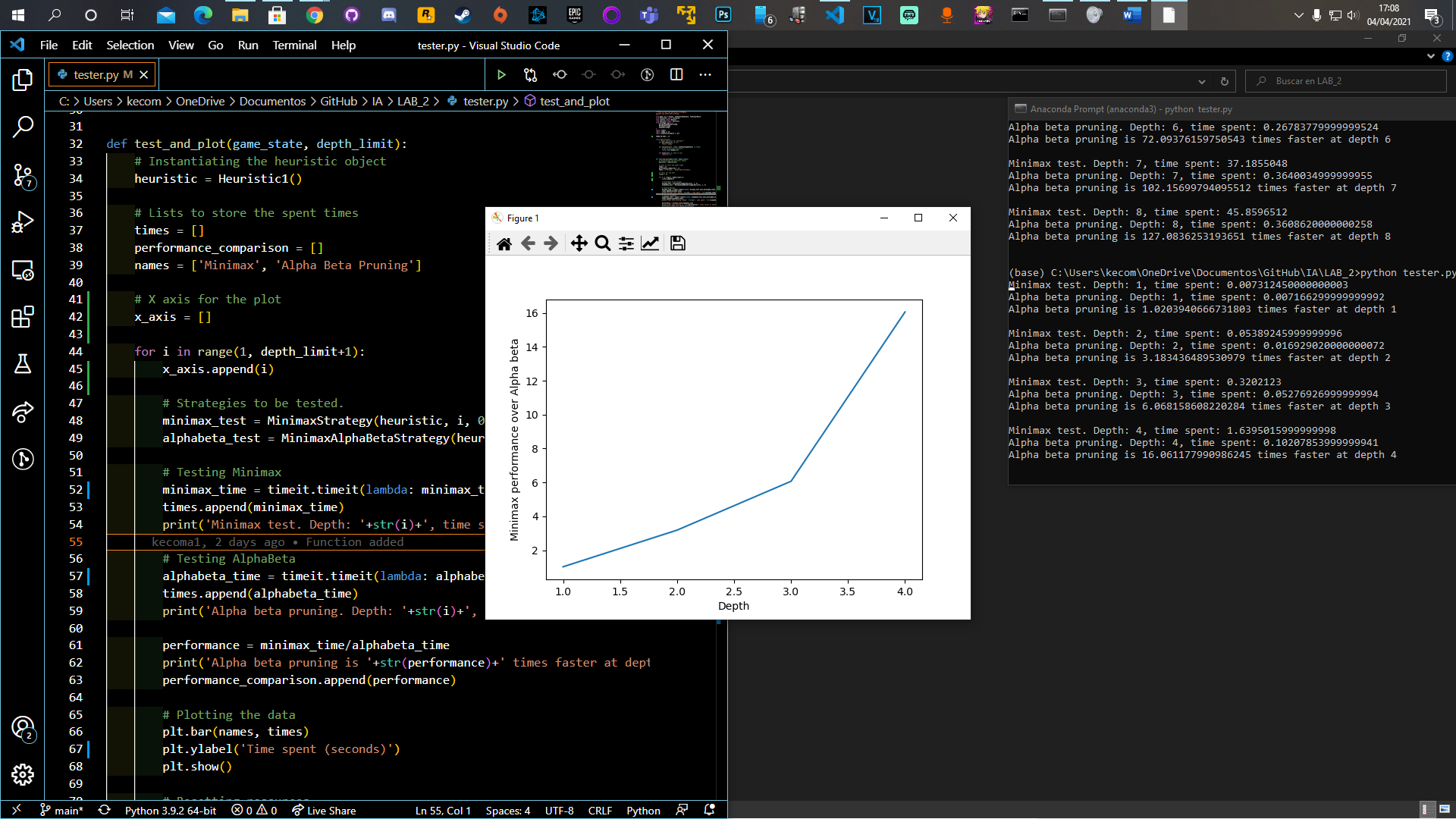
* Minimax time was 0.3202123seconds.
* Alpha-beta pruning time 0.05276926999999994 seconds.

Depth 4:

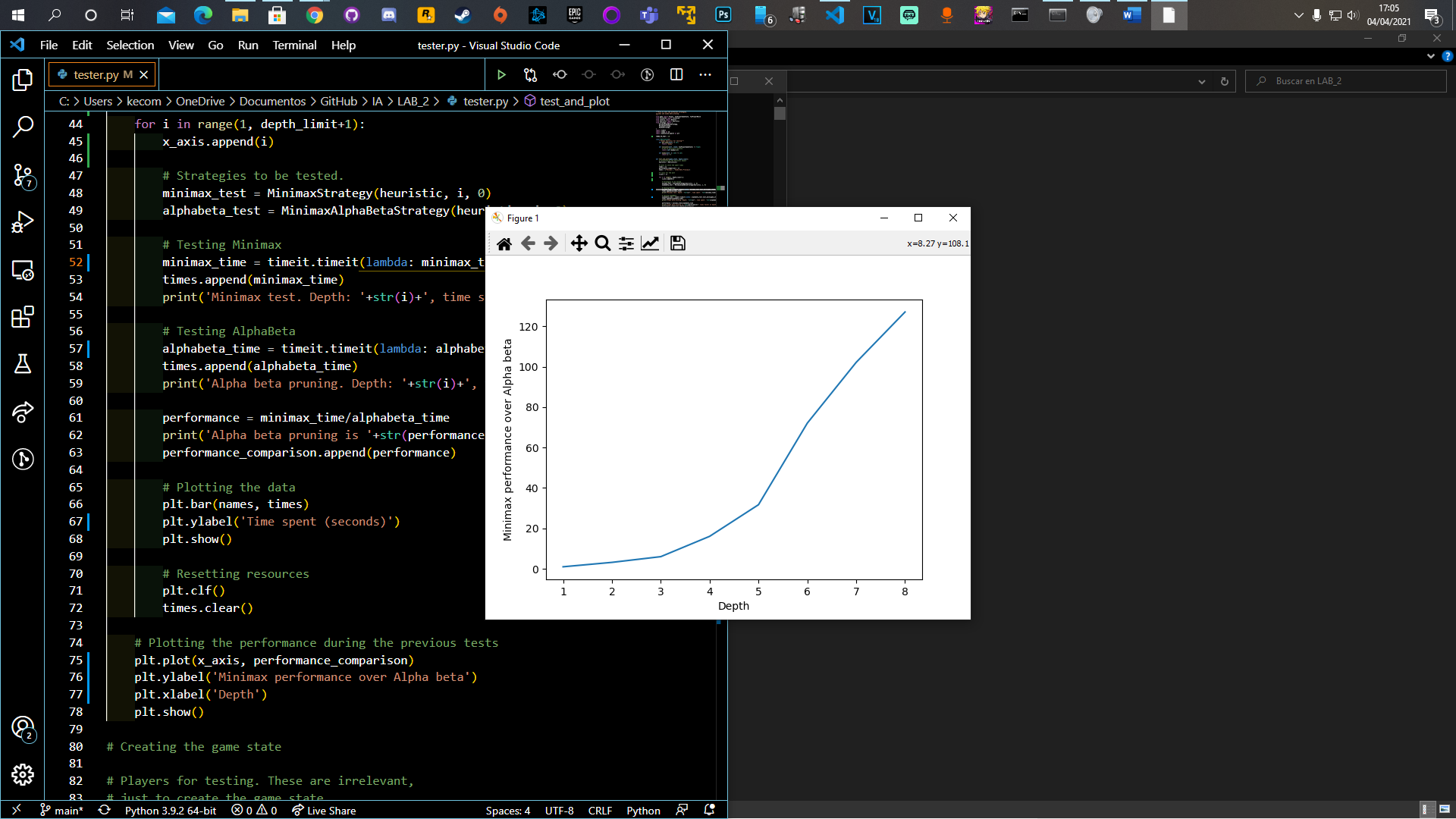


* Minimax time was 1.6395015999999998 seconds.
* Alpha-beta pruning time was 0.10207853999999941 seconds.
  + 1. **Computer independent measures of improvement.**

On the Y axis we can see how many times is alpha beta faster than minimax. On the X axis we can see the depths for each value.



From depths 1 to 4, the time spent of minimax divided by the time spent of alpha-beta.



From depths 1 to 8, the value of the division is getting bigger, this means that the gap between these 2 strategies is growing.

* + 1. **Correct, clear, and complete analysis of the results.**

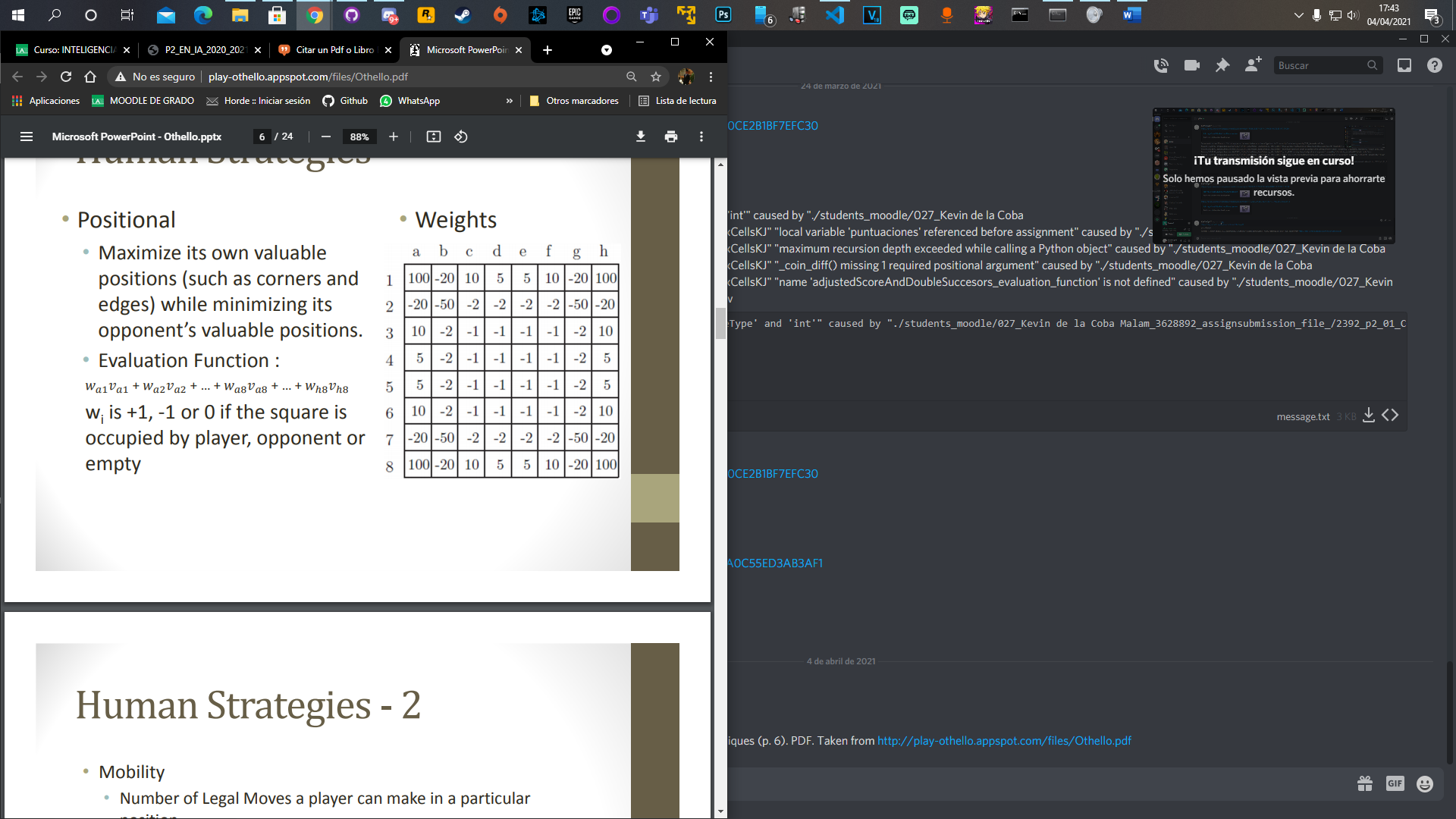
For depths 3 and 4 we could see that the bigger the depth, the bigger the difference between the strategies. This is because the “tree” gets bigger with the depth and minimax has to compare more nodes than alpha-beta (thanks to the pruning).

* + 1. **Other relevant information.**

We saw that just in the depth 1, alpha-beta pruning has a worst performance than minimax, this is because at that depth, the time spent to check if we have to prune is bigger than the overall time spent by minimax only comparing nodes.

1. Documentation of the design of the heuristic
   1. **Review of previous work on Reversi strategies, including references in APA format.**
   2. **Description of the design process:**
      1. **How was the design process planned and realized?**
      2. **Did we have a systematic procedure to evaluate the heuristics designed?**
      3. **Did we take advantage of strategies developed by others? If these are publicly available, provide references in APA format; otherwise, include the name of the person who provided the information and give proper credit of the contribution as “private communication”.**

For the Reversi strategies we mainly focused on the idea of having a weighted board, searching from the internet we found a weighted board like this:



*Parekh, P., Singh Bhatia, U., & Sukthankar, K. Othello/Reversi using Game Theory techniques (p. 6). PDF. Taken from* [*http://play-othello.appspot.com/files/Othello.pdf*](http://play-othello.appspot.com/files/Othello.pdf)

From this document we started playing with the idea of giving more value to some of the cells such as corners, walls...

* 1. **Description of the final heuristic submitted.**
  2. **Other relevant information**