A03 - Assignment 3: Adversarial Search

Students:

- Deidier Simone - student number: *133020*
- Esposito Davide - student number: *132667*

Program's Output

```
simonedeidier@dhcp-10-24-148-134 src % python3 halving_game.py
The number is 5 and it is P1's turn
P1's action: --
The number is 4 and it is P2's turn
P2's action: --
The number is 3 and it is P1's turn
P1's action: /2
The number is 1 and it is P2's turn
P2's action: --
The number is 0 and P1 won
simonedeidier@dhcp-10-24-148-134 src %
```

Figure 1: *Output of our implementation of the minimax algorithm for the halving game.*

```
simonedeidier@dhcp-10-24-148-134 src % python3 bucket_game.py
The state is (0, ['A', 'B', 'C']) and it is P1's turn
P1's action : B
The state is (1, [3, 1]) and it is P2's turn
P2's action : 1
The state is (0, [1]) and P1's utility is 1
simonedeidier@dhcp-10-24-148-134 src %
```

Figure 2: *Output of our implementation of the minimax algorithm for the bucket game.*

Tic Tac Toe Minimax Variant

In this scenario, the Minimax algorithm may choose the middle square (*instead of the bottom-left, which would be the winning move for X*) because of the way the algorithm evaluates game states. Minimax tries to optimize the worst-case scenario and plays conservatively to minimize potential losses, this means it doesn't necessarily recognize the immediate opportunity for a win if it's prioritizing a defensive strategy, especially if it's programmed to focus on preventing

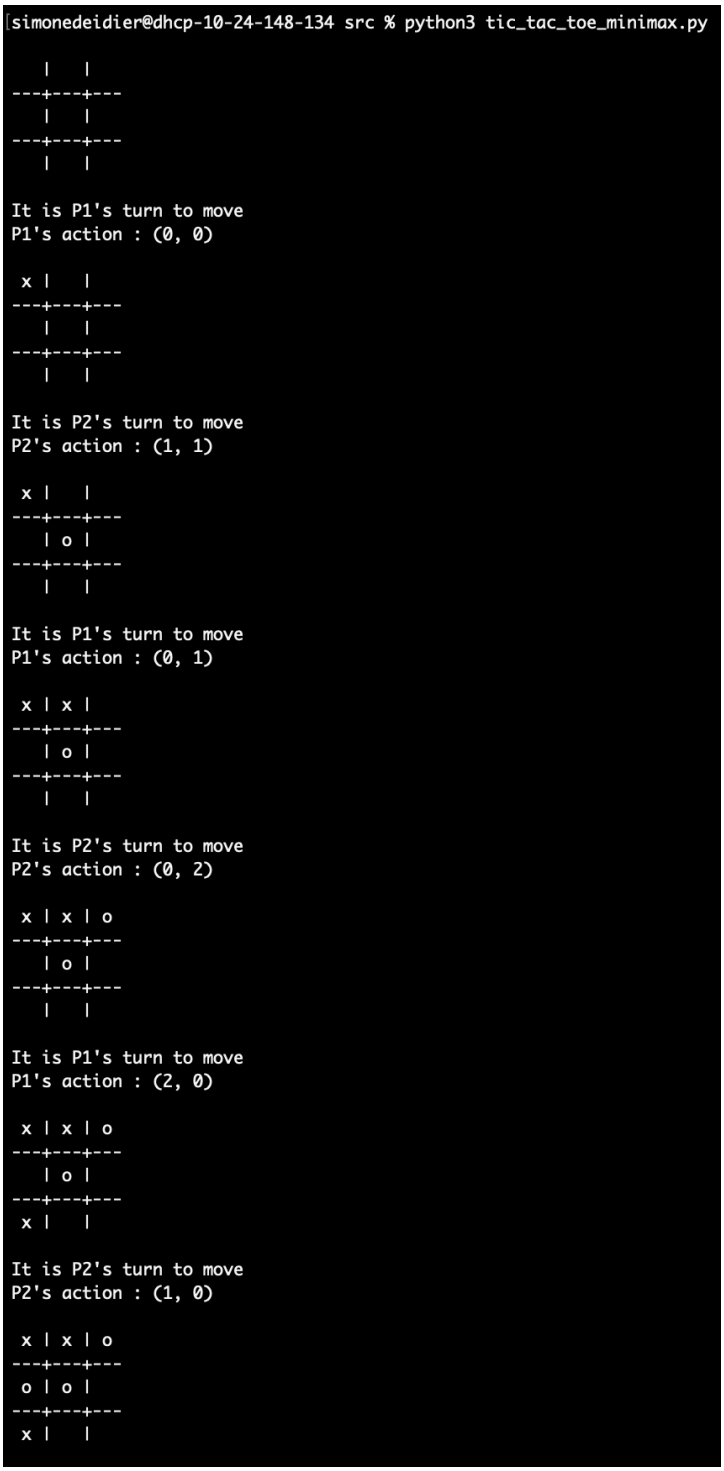

Figure 3: *Output of our implementation of the minimax algorithm for the tic tac toe game - part 1.*

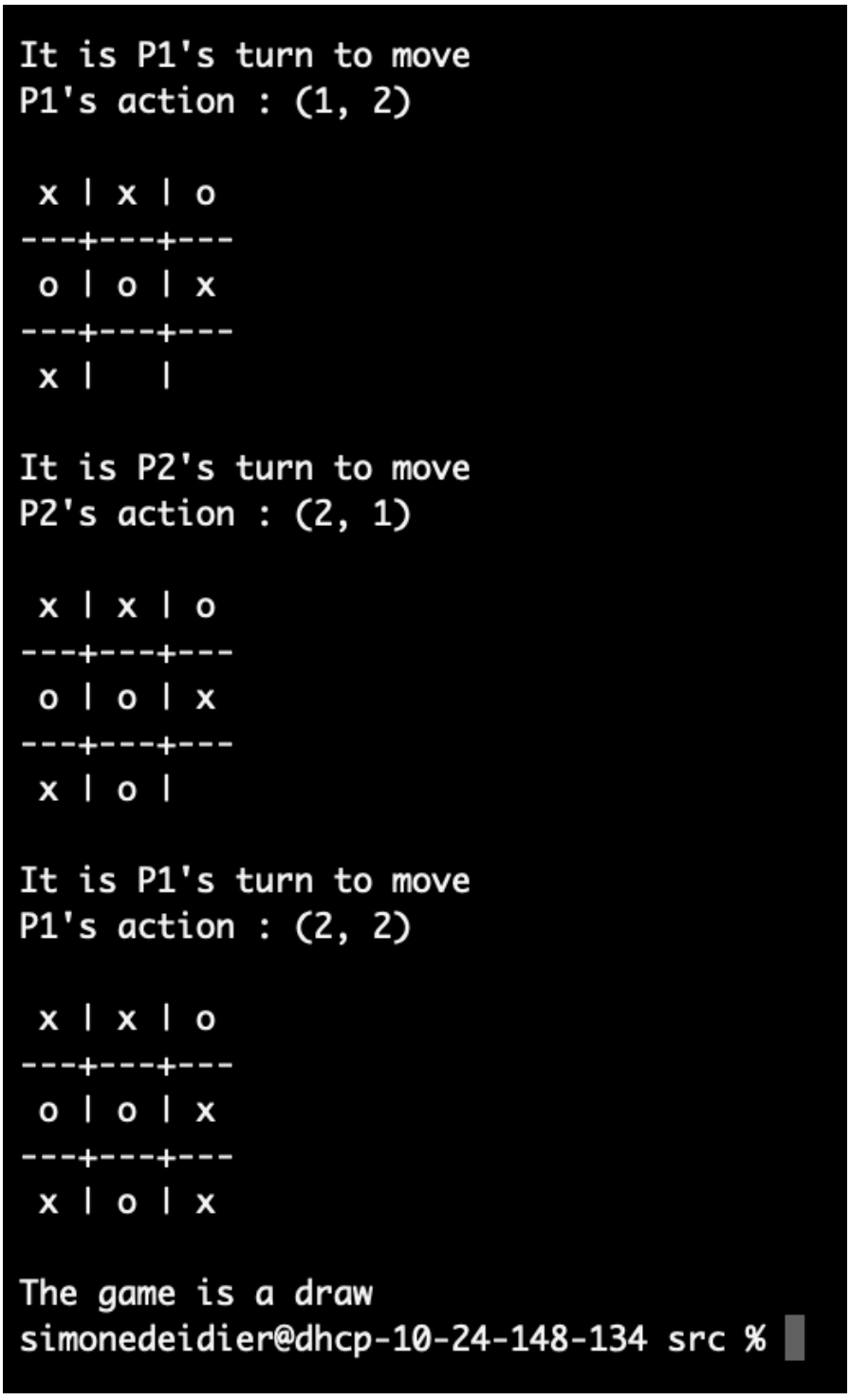

Figure 4: *Output of our implementation of the minimax algorithm for the tic tac toe game - part 2.*

future losses rather than immediate victories.

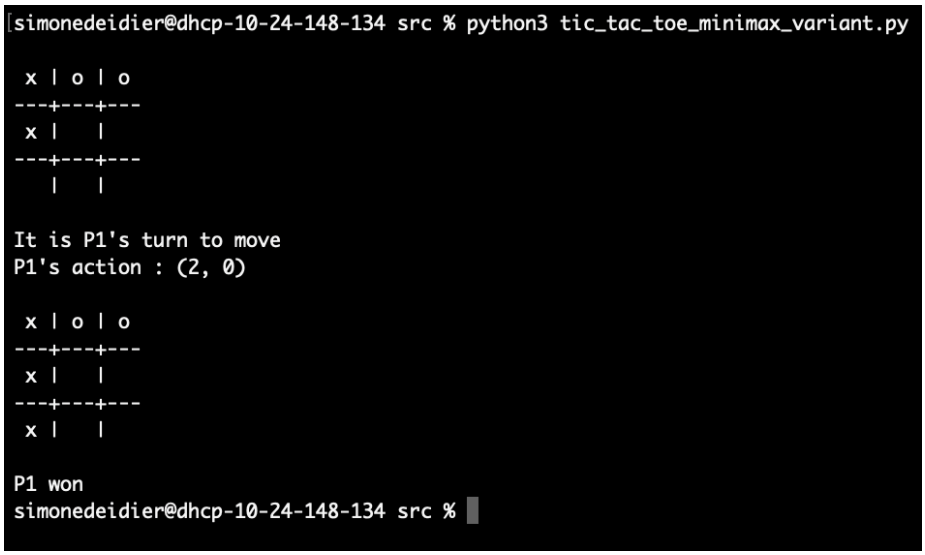

Figure 5: *Output of our implementation of the variant minimax algorithm for the tic tac toe game to prioritize winning moves.*

Runtime to Find the First Move

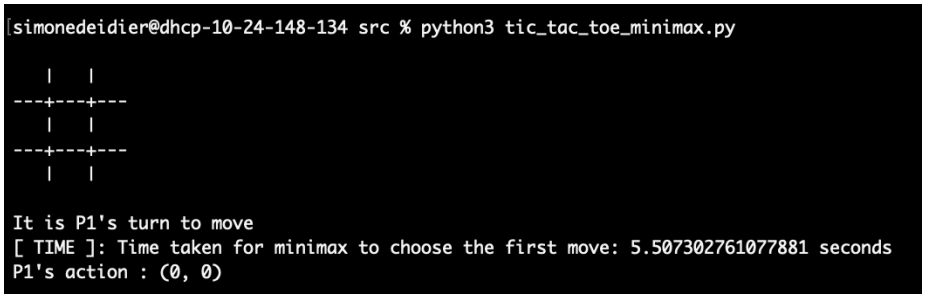

Figure 6: *Runtime of our minimax algorithm to find the first move.*

$$\text{Speedup} = \frac{T_{\text{minimax}}}{T_{\alpha\beta\text{-pruning}}} = \frac{5.5073}{0.1855} \approx 29.68733$$

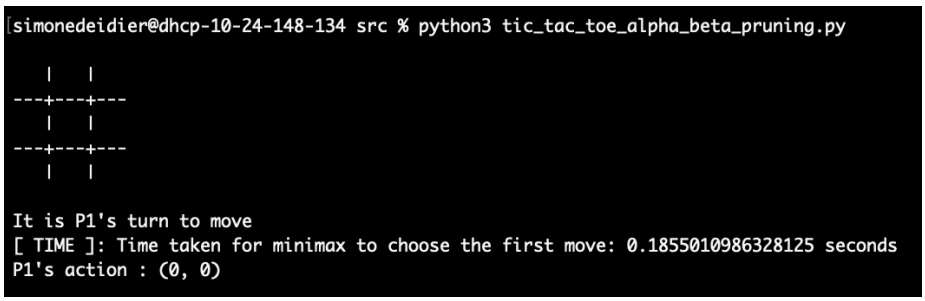

Figure 7: *Runtime of our alpha-beta pruning algorithm to find the first move.*