Explanations :

Minimax algorithm:
It wasn’t anything special.

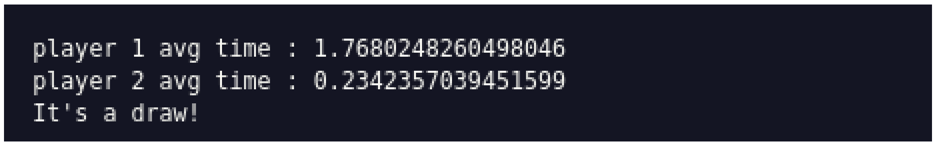
Minimax + Alpha beta pruning:
I think it is possible to write this function with only one variable acting as both alpha and beta, but i failed to prove that so i just stuck with two variables.
Otherwise this was pretty much the same as minimax.

Minimax + evaluation function:
It also uses alpha beta pruning because why not!
I made the algorithm to not go any further than depth 4 in search of a move. It may not have been the best decision but I wanted to rely more on my evaluation function rather than minimaxing.
The evaluation function works like this:
It gives the player who has the turn 1 point.
It scans the field and gives one point to a player who is one move away from winning (not regarding turn)

MCTS with UCB1:
This one was tricky. I didn’t like the fact that MCTS was based on hopes and dreams. Whenever the board was against it (e.g. it had to play for either draw or loss. The possibility of winning is very slim), it would play a move that would result in its loss.
I tried to fix that with an enhanced reward policy. I made the reward for draw the same as victory, which is 8 points. Loss has the reward of -10. If reward for draw becomes 7, the tree starts to lose instead of drawing!
It made things worse as it started to make random moves.
I fixed that problem with tuning the constant c in ucb function from sqrt (2) to sqrt(250).
After all, with only playing 100 games for deciding its next move, it can draw minimax.
I DID IT! (it actually took so long to do it and it wasn’t worth it)

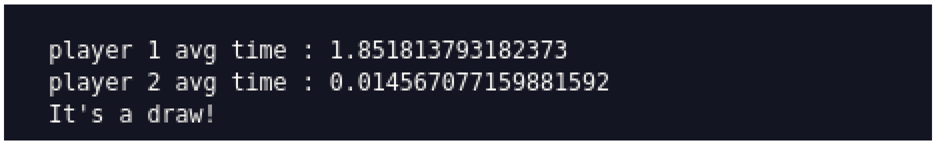
MATCHDAY:

Minimax vs minimax:



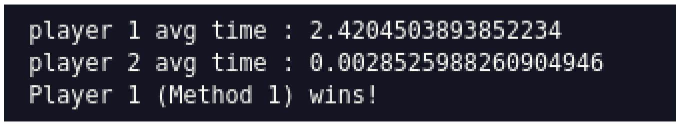
```
player 1 avg time : 1.7680248260498046
player 2 avg time : 0.2342357039451599
It's a draw!
```

Minimax vs alphabeta:



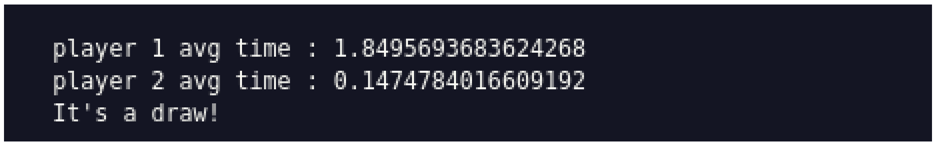
```
player 1 avg time : 1.851813793182373
player 2 avg time : 0.014567077159881592
It's a draw!
```

Minimax vs eval:



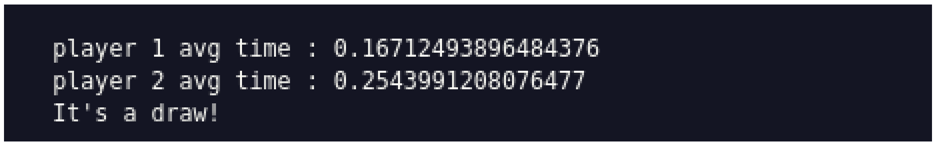
```
player 1 avg time : 2.4204503893852234
player 2 avg time : 0.0028525988260904946
Player 1 (Method 1) wins!
```

Minimax vs MCTS:



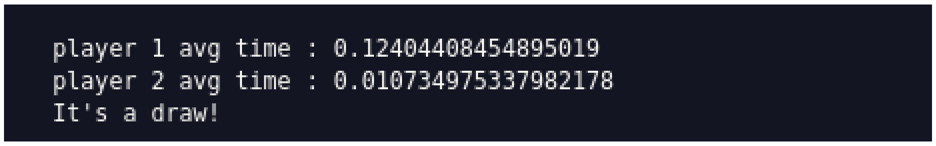
```
player 1 avg time : 1.8495693683624268
player 2 avg time : 0.1474784016609192
It's a draw!
```

Alphabeta vs minimax:



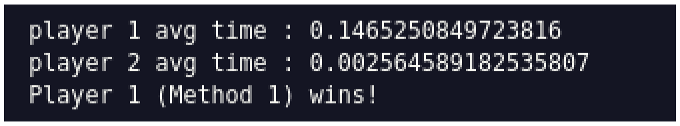
```
player 1 avg time : 0.16712493896484376
player 2 avg time : 0.2543991208076477
It's a draw!
```

Alphabeta vs alphabeta:



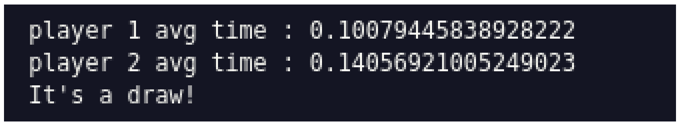
```
player 1 avg time : 0.12404408454895019
player 2 avg time : 0.010734975337982178
It's a draw!
```

Alphabeta vs eval:



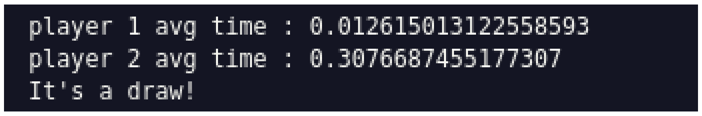
```
player 1 avg time : 0.1465250849723816
player 2 avg time : 0.002564589182535807
Player 1 (Method 1) wins!
```

Alphabeta vs MCTS:



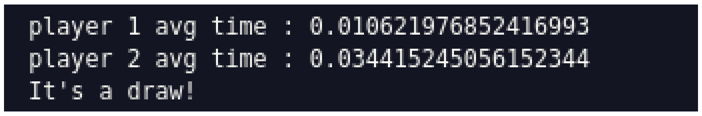
```
player 1 avg time : 0.10079445838928222
player 2 avg time : 0.14056921005249023
It's a draw!
```

Eval vs minimax:



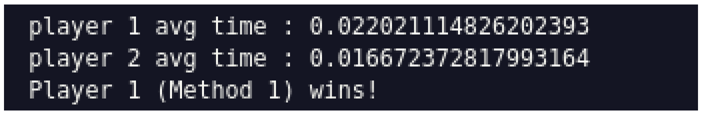
```
player 1 avg time : 0.012615013122558593
player 2 avg time : 0.3076687455177307
It's a draw!
```

Eval vs alphabeta:



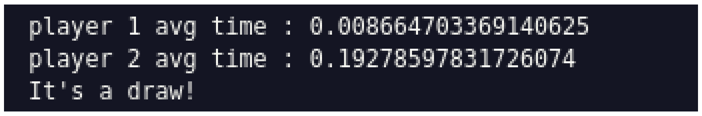
```
player 1 avg time : 0.010621976852416993
player 2 avg time : 0.034415245056152344
It's a draw!
```

Eval vs eval:



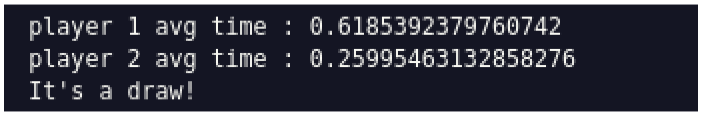
```
player 1 avg time : 0.022021114826202393
player 2 avg time : 0.016672372817993164
Player 1 (Method 1) wins!
```

Eval vs MCTS:



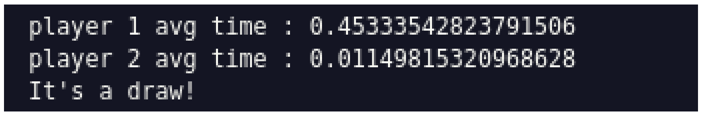
```
player 1 avg time : 0.008664703369140625
player 2 avg time : 0.1927859783172607 4
It's a draw!
```

MCTS vs minimax:



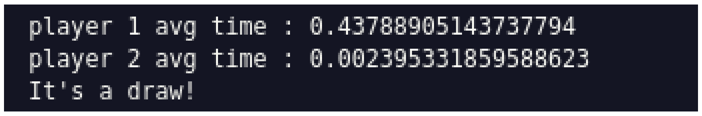
```
player 1 avg time : 0.6185392379760742
player 2 avg time : 0.25995463132858276
It's a draw!
```

MCTS vs alphabeta:



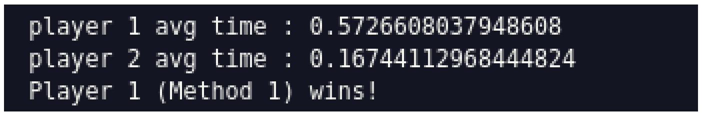
```
player 1 avg time : 0.45333542823791506
player 2 avg time : 0.01149815320968628
It's a draw!
```

MCTS vs eval:



```
player 1 avg time : 0.43788905143737794
player 2 avg time : 0.002395331859588623
It's a draw!
```

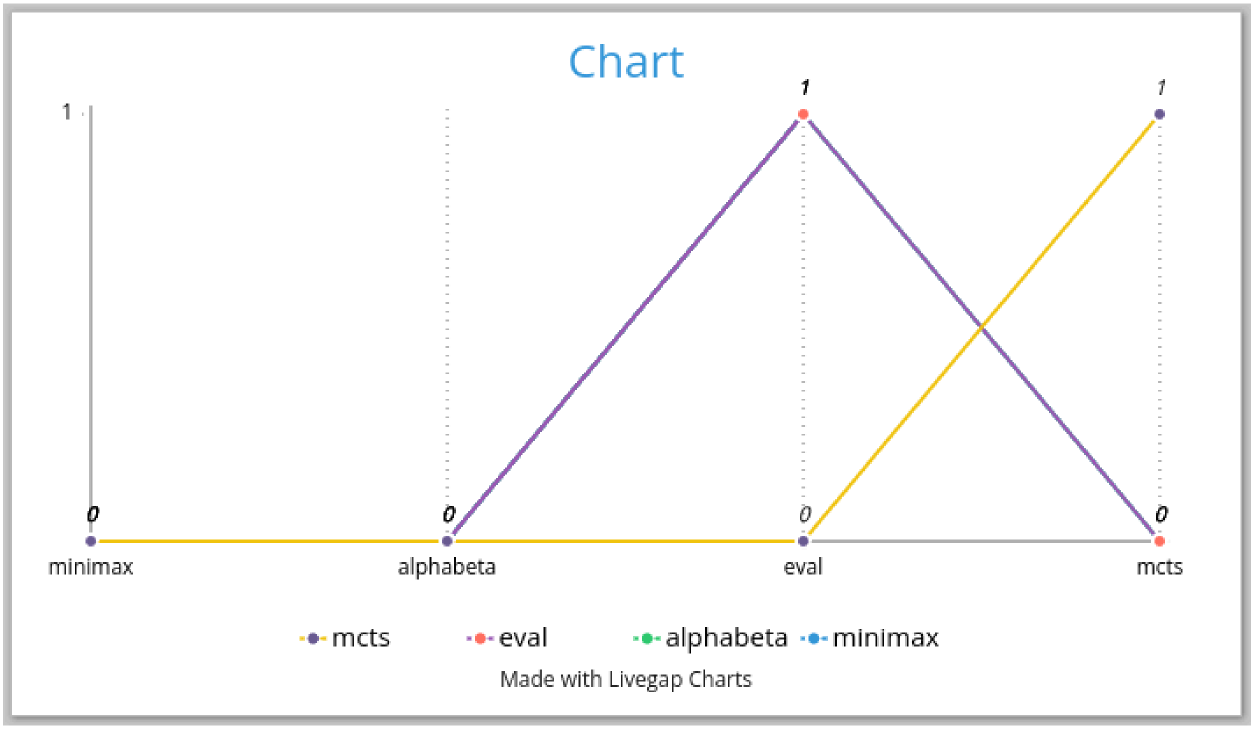
Eval played perfectly! MCTS didn't abuse evals weaknesses
MCTS vs MCTS:

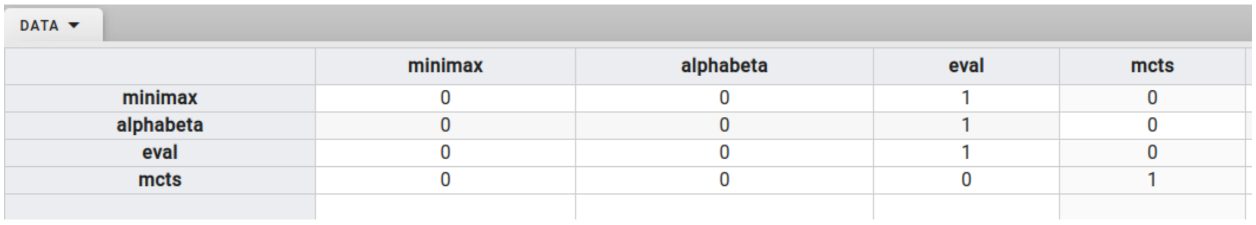


```
player 1 avg time : 0.5726608037948608
player 2 avg time : 0.16744112968444824
Player 1 (Method 1) wins!
```

The only one who beat MCTS is MCTS!

Overall:
Minimax is very heavy.
Alpha beta is minimax with an optimization. It is less heavy but it is still heavy! You can't see the heaviness in this game because this game is very restricted.
Their hardware usage is almost the same.
Evaluation based minimax is as good as its evaluation function. My evaluation is a little bit lazy. For a game like tic tac toe, you could come up with an eval function that would be super accurate. Evaluation based minimax heaviness is dependent on the depth. 3 or 4 depth is good enough for tic tac toe.
MCTS is in my opinion the best algorithm. For this game it is a bit underrated for sure but i believe in it! I managed to find the best values for it to work!




DATA

	minimax	alphabeta	eval	mcts
minimax	0	0	1	0
alphabeta	0	0	1	0
eval	0	0	1	0
mcts	0	0	0	1