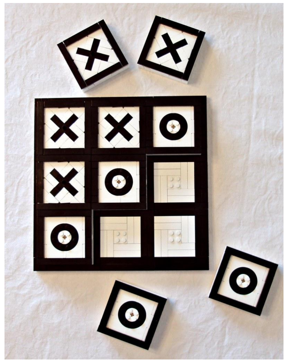
PROJECT WORK IN FAIKR

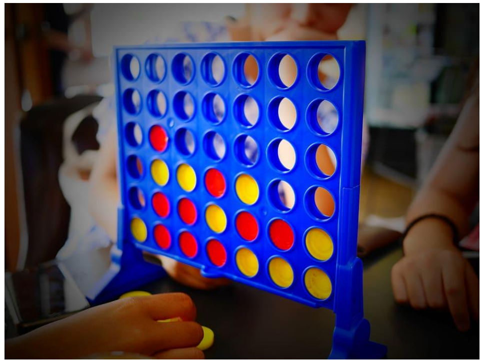
MIN-MAX ALGORITHM INTO BOARD GAMES: CONNECT 4 AND TIC TAC TOE

IN THIS PRESENTATION:

- Describe the traditional algorithm used in board game AI: Min-Max algorithm and Alpha-Beta cuts to reduce the search space.

-Apply this same algorithm to two games taken as examples: Tic Tac Toe and Connect 4. Then explain the two used strategies.





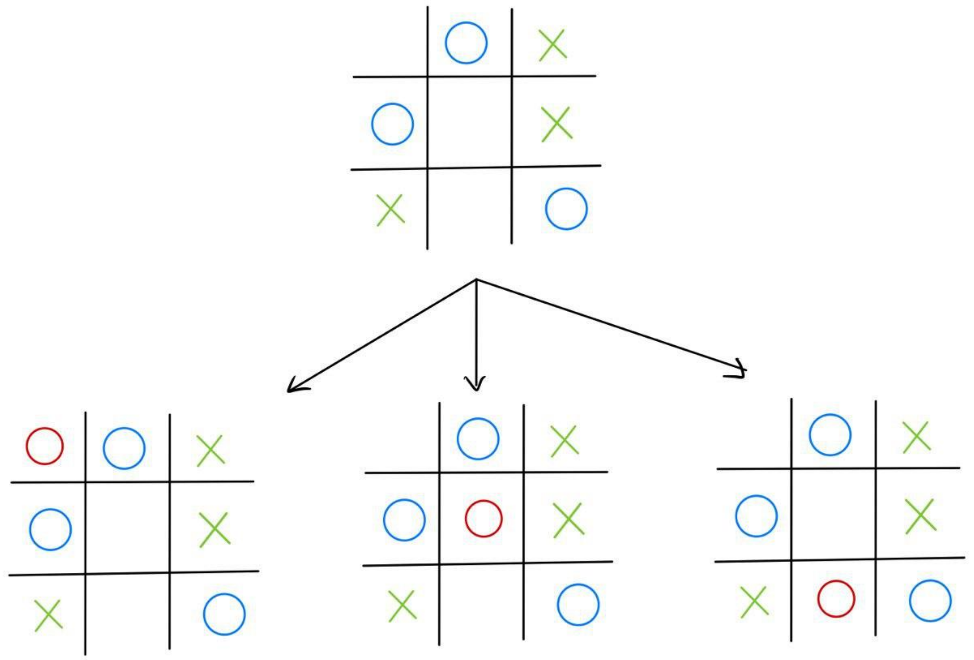
USED SETUP

-The whole project was developed using python.

-The graphic interface of the games has been implemented with the use of pygame library.

Everything can be downloaded and performed using the instructions indicated at this link.

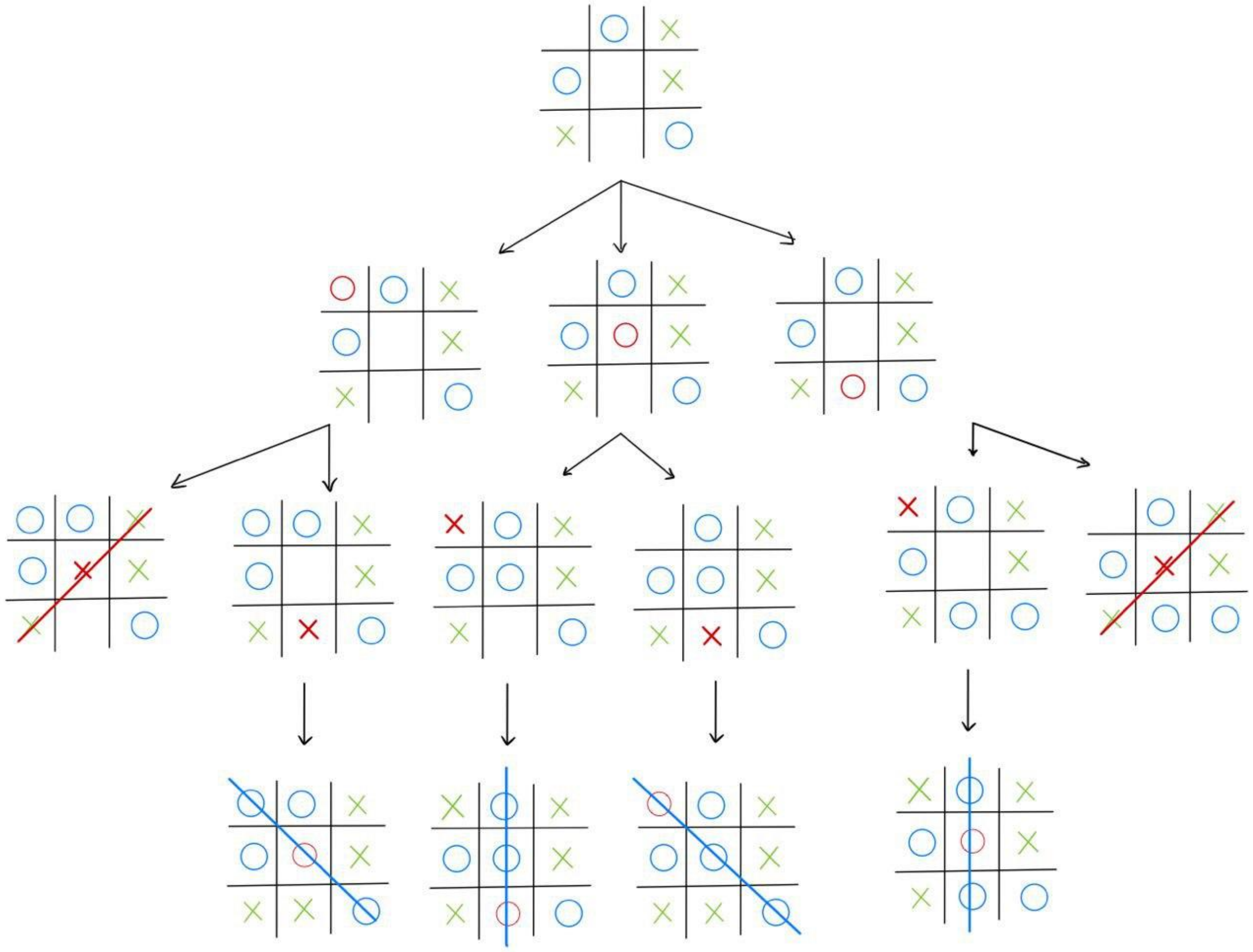
TIC TAC TOE

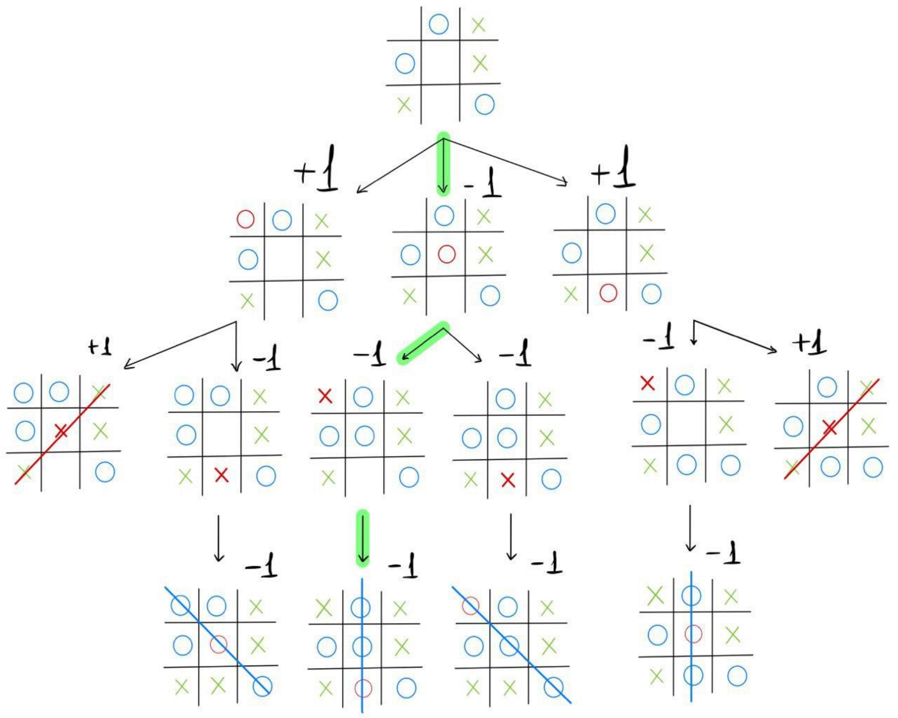


O MINIMIZE

X MAXIMIZE

TERMINAL CASES	EVALUATION
O WINS	-1
X WINS	+1
DRAW	0



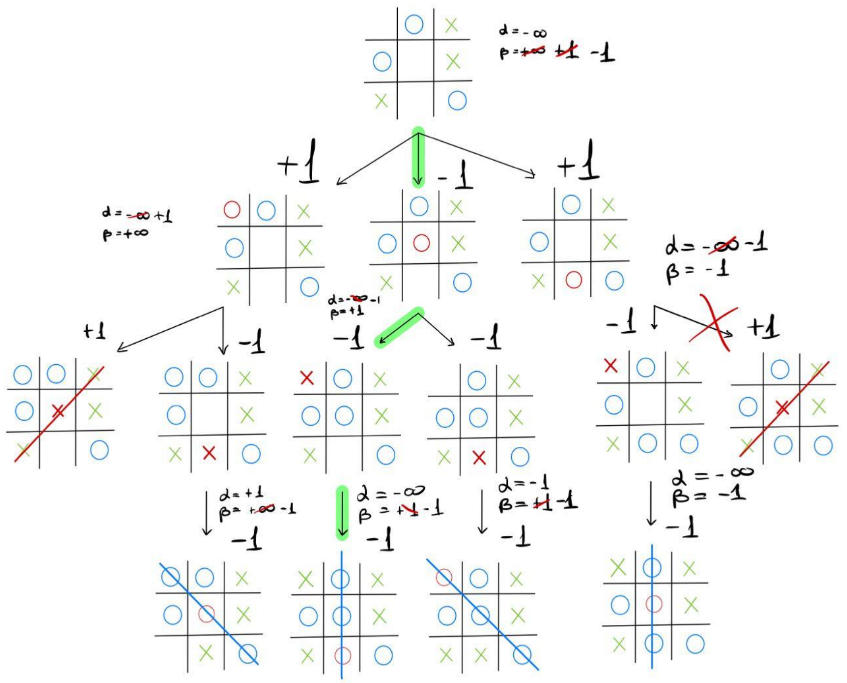


+1
-1
+1
+1
-1
-1
-1
-1
+1
-1
-1
-1
-1

MIN

MAX

MIN



α = -∞
β = ~~+∞~~ ~~+1~~ -1
+1
-1
+1
α = ~~-∞~~ +1
β = +∞
α = ~~-∞~~ -1
β = -1
α = ~~-∞~~ -1
β = +1
+1
-1
-1
-1
-1
+1
α = +1
β = ~~+∞~~ -1
-1
α = -∞
β = ~~+1~~ -1
-1
α = -1
β = ~~+1~~ -1
-1
α = -∞
β = -1
-1

TIC TAC TOE RESULTS

Obviously the simpler of the two.

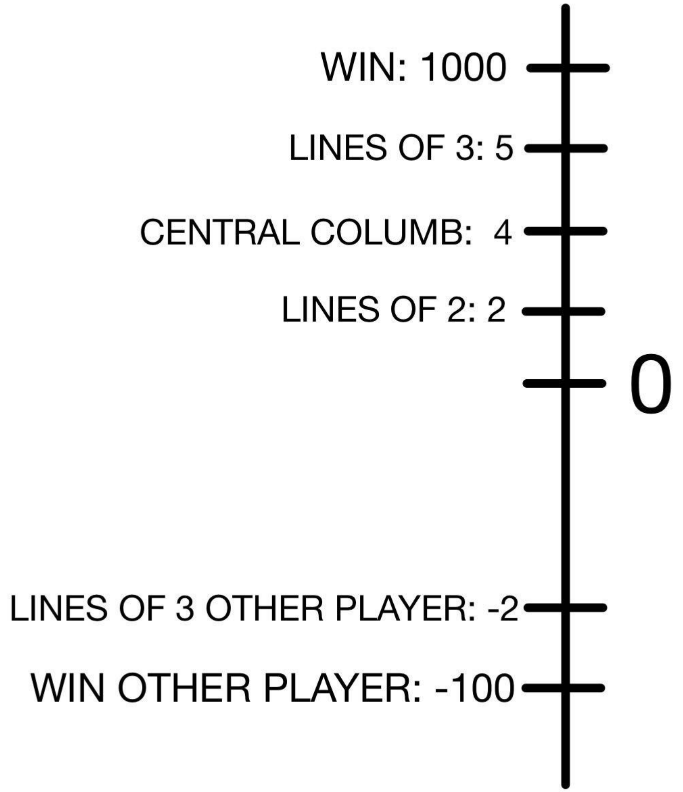
By applying the Min-Max algorithm to this game, it is possible to explore the entire search space (at most 9 * 8 * 7 * .. * 1 nodes), thus managing to obtain the unbeaten, at most you can draw.

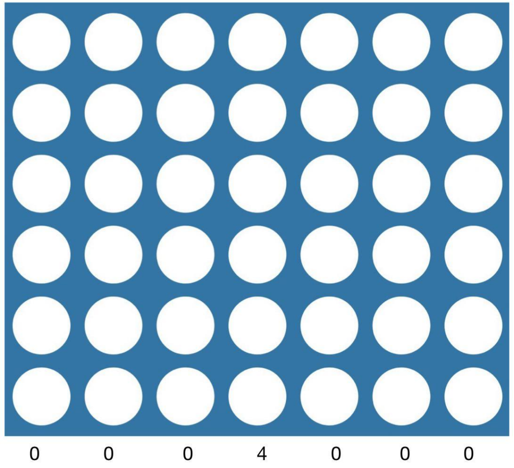
The alpha-beta cuts, is not absolutely necessary, since, without it, the first move (if made by the AI) is made in about 4 seconds, while the following ones in even less time and from the third onwards negligible.

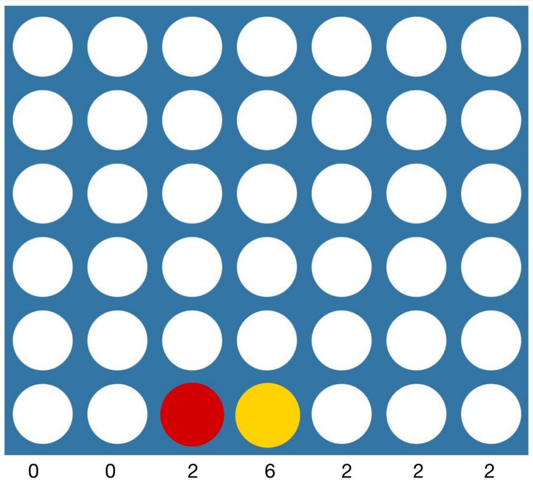
The addition of alpha-beta cuts improves these already low times.

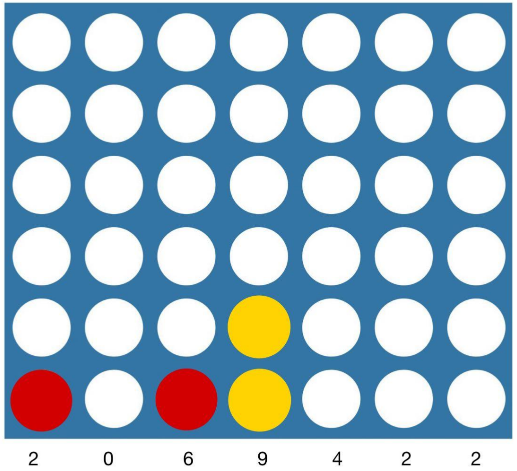
CONNECT 4

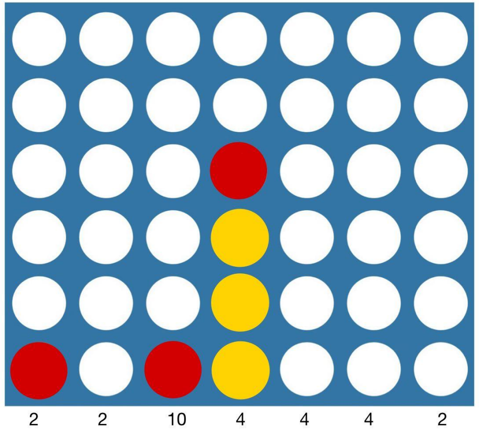
Before seeing how the Min-Max algorithm has been applied to this game, we must first explain the move evaluation mechanism.

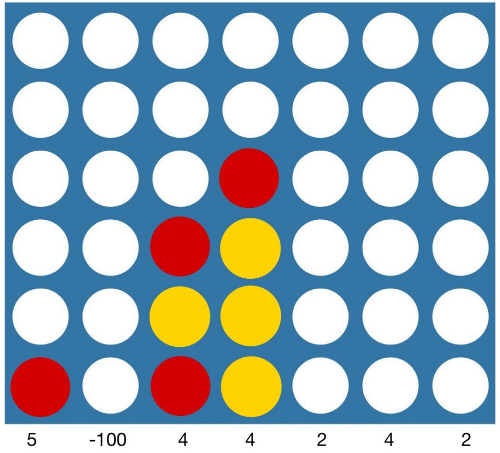


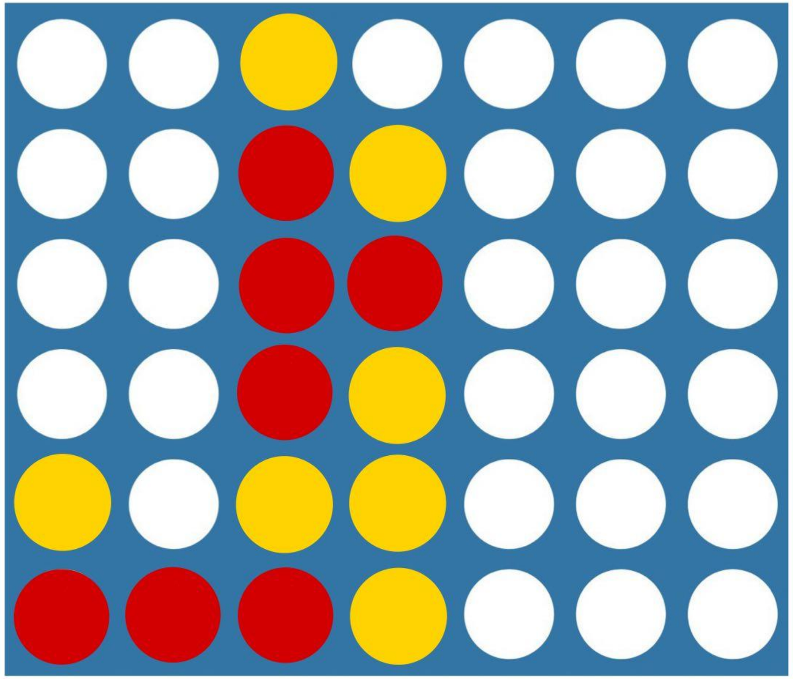

0 0 0 4 0 0 0

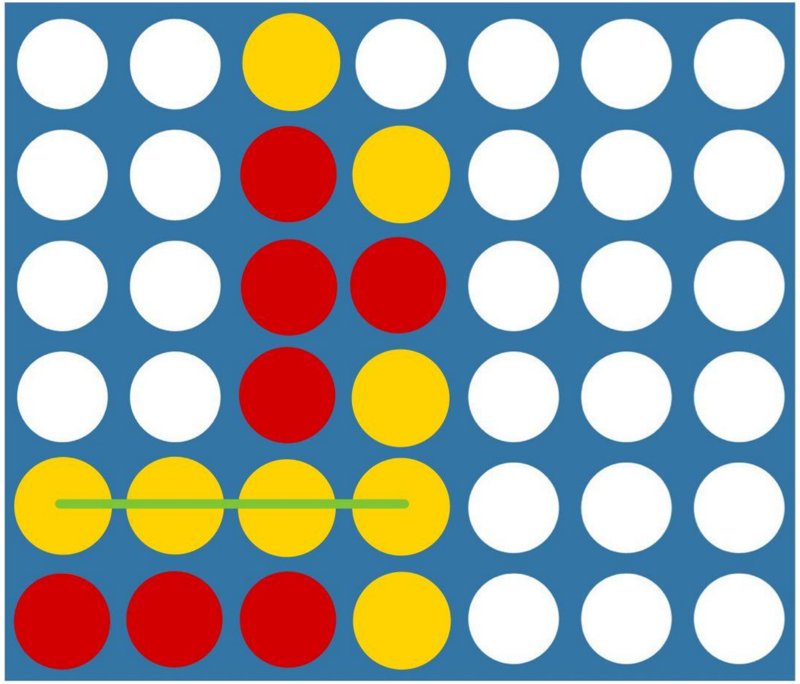

0 0 2 6 2 2 2

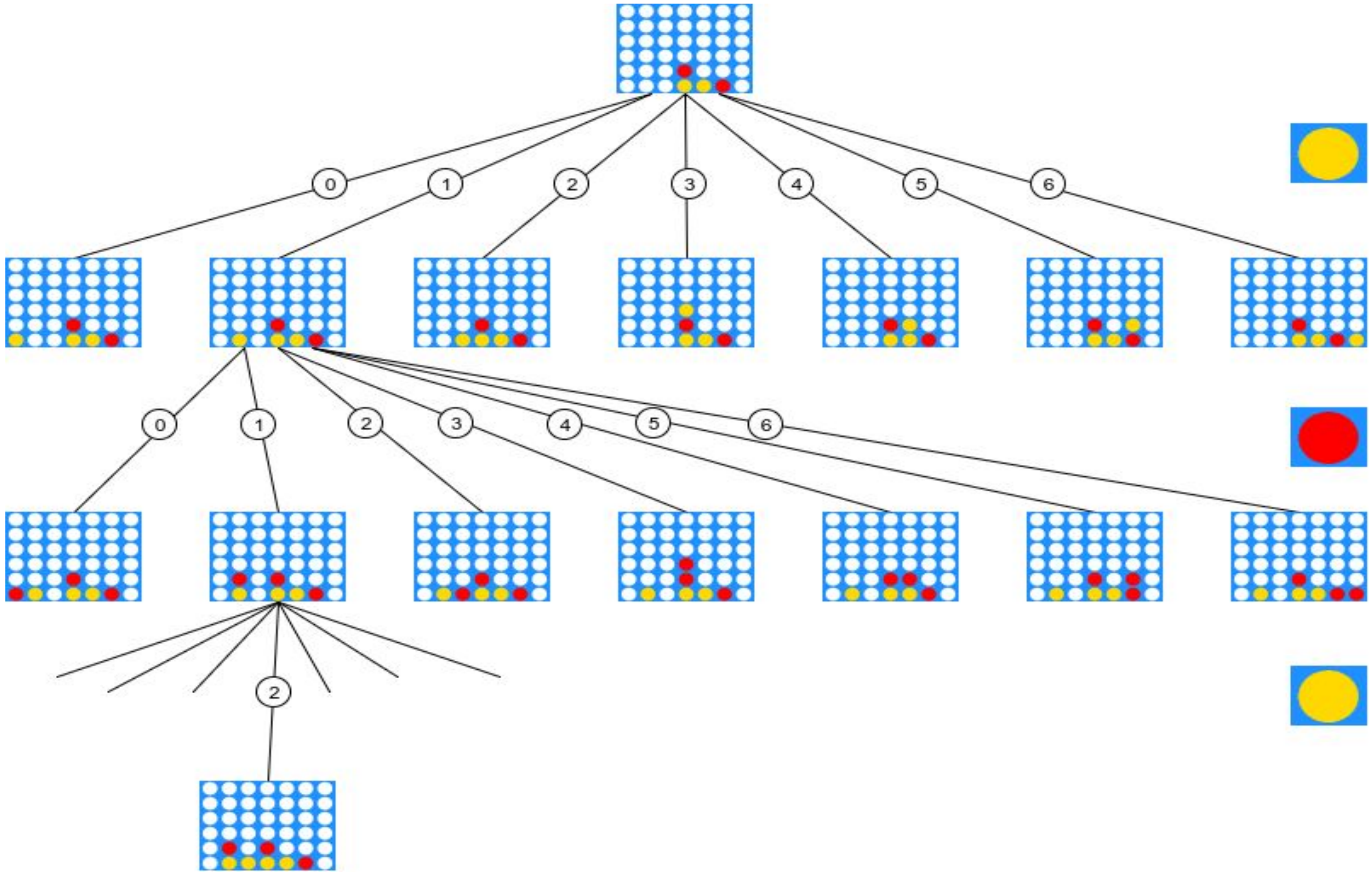

2 0 6 9 4 2 2


2 2 10 4 4 4 2


5 -100 4 4 2 4 2


1000


1000


0
1
2
3
4
5
6
0
1
2
3
4
5
6
2

CONNECT 4 RESULTS

Obviously the more difficult of the two.

Here it was necessary to include a more complex strategy than the one used in Tic Tac Toe, which simply evaluated victory, defeat and draw (1, -1 and 0) as seen in the previous slides.

Here the entire search space cannot be explored (at most 6^42 nodes) in fact in order not to wait superhuman times (or saturate the RAM of the pc) I decided to enter a maximum depth (6) so that the waiting time is only a few seconds (in the first few moves, the further you go the more negligible it becomes). The node that is at this maximum depth is worth zero if it is not the victory node for either player (added, of course, to the path node to get there).

Alpha-Beta cuts is absolutely necessary, compared to Tic Tac Toe, as it allows us to get to a greater depth in less time, and thus greatly improve the performance of the algorithm (from depth 4 to 6 with the same properties).

THE END

THANKS FOR LISTENING