Game Trees & Path Planning

Group Members:
Daniel Biskup, Mohammad Asif Khan, Nilesh Chakraborty, Nofel Mahmood, Saptarishi Bhattacharya, Saurav Ganguli

Overview

Game Tree

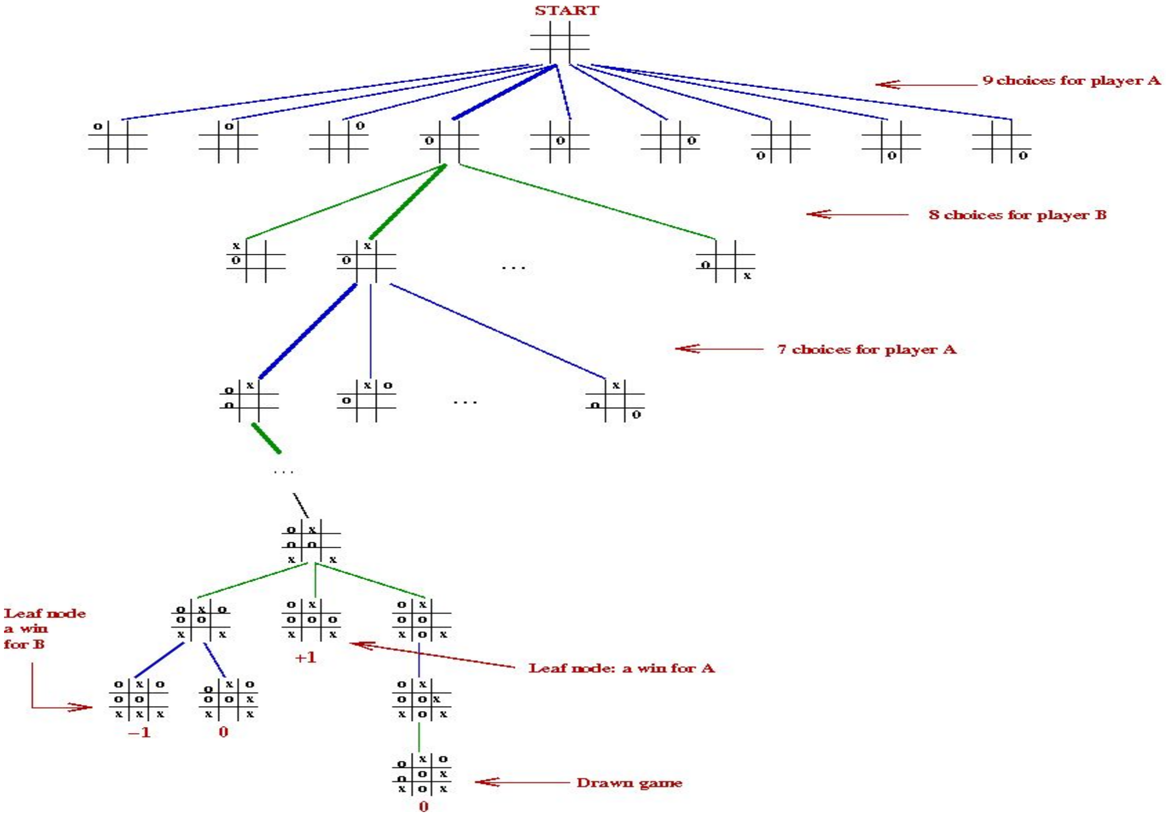
Minimax Algorithm

Connect 4

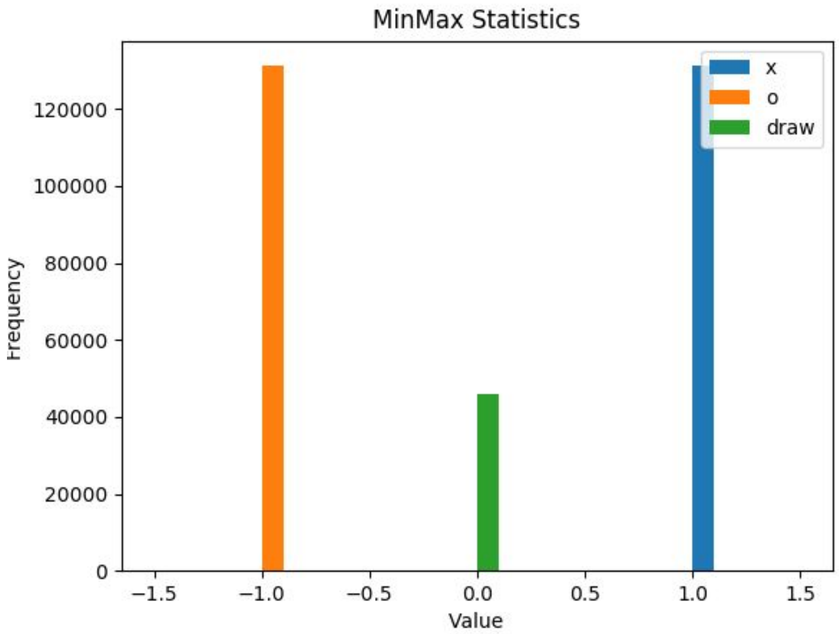
Breakout

Path Planning

Game Tree


START
9 choices for player A
8 choices for player B
7 choices for player A
Leaf node a win for B
Leaf node: a win for A
Drawn game
+1
−1
0
0

Game Tree



MinMax Statistics
x
o
draw
Frequency
Value
0
20000
40000
60000
80000
100000
120000
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

Game Tree

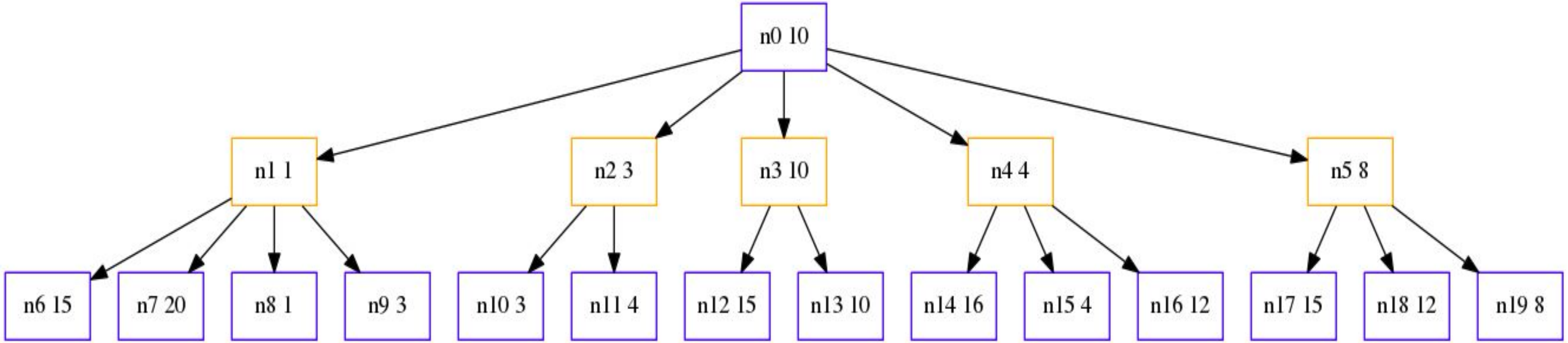
Upper Bound on Number of Nodes:	**255168.**
Wins of player X:	**131184.**
Wins of player O:	**77904.**
Draws:	**46080.**
Number of Nodes:	**549945.**
Number of Leaf Nodes:	**255168.**
Number of Branches:	**549936.**
Branching Factor:	**0.99998**

Game Tree

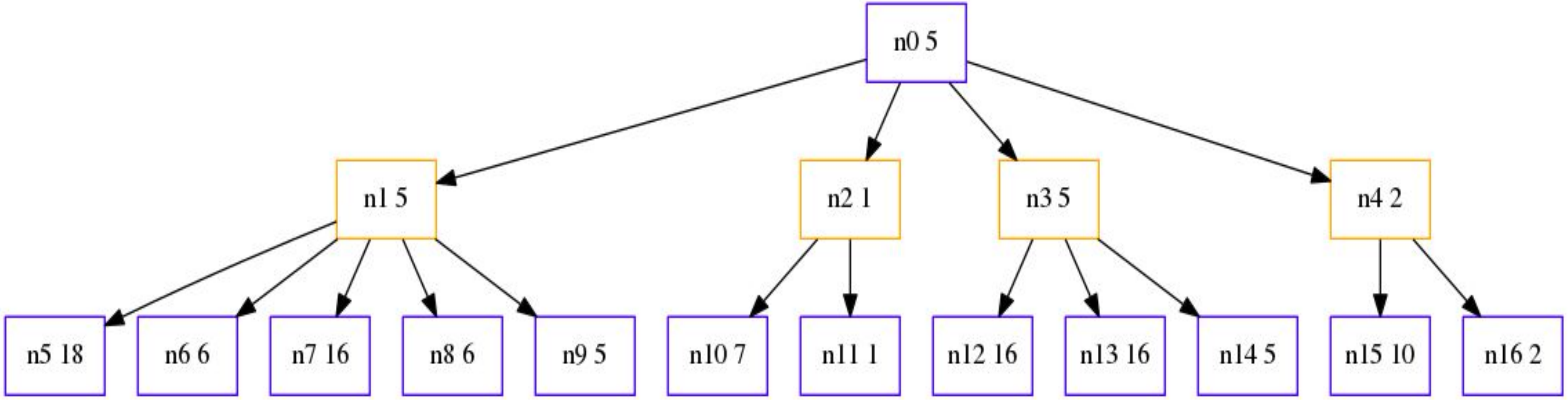
Upper Bound Combinatorial Argument

Games ending on 5th Move:	8*3!*6*5	=1440.
Games ending on 6th Move:	8*3!*6*5*4 -6*3!*2*3!	=5328.
Games ending on 7th Move:	8*3*6*3!*5*4*3 -6*3*6*3!*3!	=47952.
Games ending on 8th Move:	8*3*6*3!*5*4*3*2 - 6*3*6*3!*2*4!	=72576.
Games ending on 9th Move:	2*3*8*4!*4! + 6*3*4*4!*4! + 22*1*4!*4! + 16*5!*4!	=127872.
Total		**255158**

MiniMax


n0 10
n1 1
n2 3
n3 10
n4 4
n5 8
n6 15
n7 20
n8 1
n9 3
n10 3
n11 4
n12 15
n13 10
n14 16
n15 4
n16 12
n17 15
n18 12
n19 8

MiniMax


n0 5
n1 5
n2 1
n3 5
n4 2
n5 18
n6 6
n7 16
n8 6
n9 5
n10 7
n11 1
n12 16
n13 16
n14 5
n15 10
n16 2

Tie-breaking Heuristics

- How to resolve the tie between n1 and n3?
  - Randomly
  - Recursive arg-max to pick the node having a child with highest score (i.e. n1)
- Outcome does not change assuming both players are rational (make optimal moves at each step)

Connect 4

Recursive MiniMax Algorithm

DFS Game Tree traversal

Game Tree not explicitly but still traversed

Required:

- Reward Function for terminal states
- Evaluation Function for non-terminal states

Game State Representation

						X
		O		X		O
	O	X	O	O	O	X
X	X	O	O	X	X	O

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	1
0	0	-1	0	1	0	-1
0	-1	1	-1	-1	-1	1
1	1	-1	-1	1	1	-1

Evaluation Function

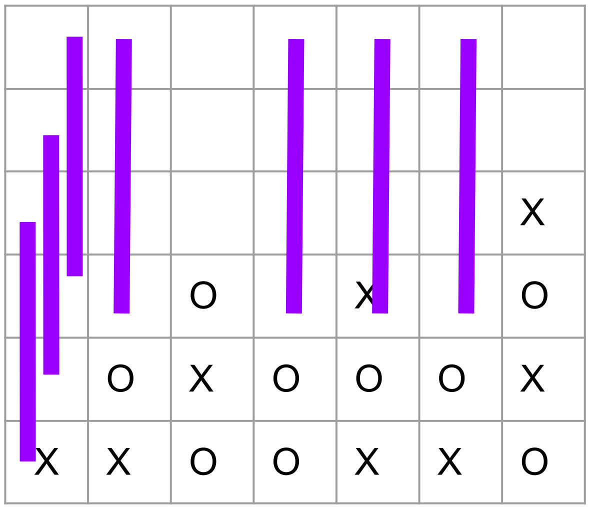
Define a function on the Game State:

f(A, GameState) = number of possible 4-in-a-row configurations left for player A in GameState.

eval(A, GameState) =
f(A, GameState) - f(B, GameState)

Intuition:
It's easier for A to win, if there are more possibilities left for A to win than for B.

Example for vertical lines. Also count horizontal and diagonal ones to add them all up.



f(X,GameState) = 3 + 1 + 0 + 1 + 1 + 1 + 0 = 7

Reward for Terminal States

Win: 1000

Draw: 0

Loss: -1000

Evaluation of non terminal states will always lie between -1000 and 1000, so, so that information about terminal stats weights heavier.

Make it more human-like

AI is predictable / makes same moves every time:

The Fix:

At the root, pick the Max move out out all maximum moves at random.

AI doesn't win as fast as possible

This is NOT a bug, but it would look more human, to:

Win as fast as possible.

Delay losing as long as possible.

The Fix: In the MinMax-Tree:

Pick the shortest path to a **win**:

Because it looks more human.

Pick the longest path to a **loss**:

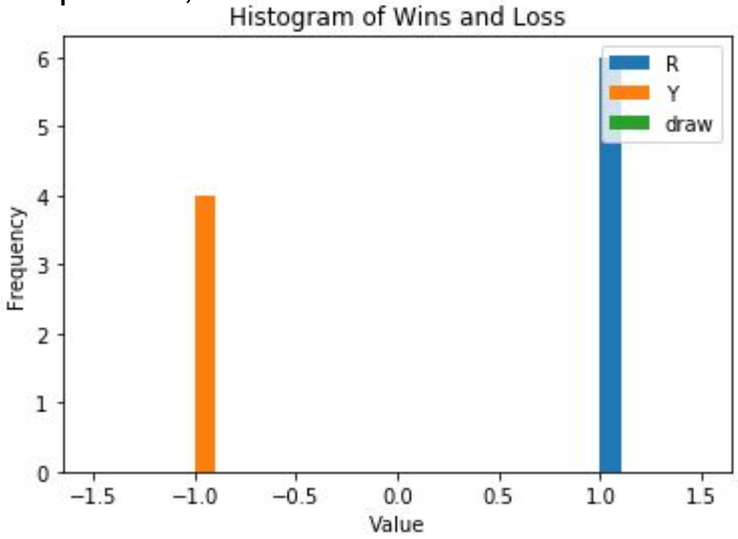
Because in games against humans or players that make radom moves this might change the outcome of the game.

Implementation in Reward for terminal states:

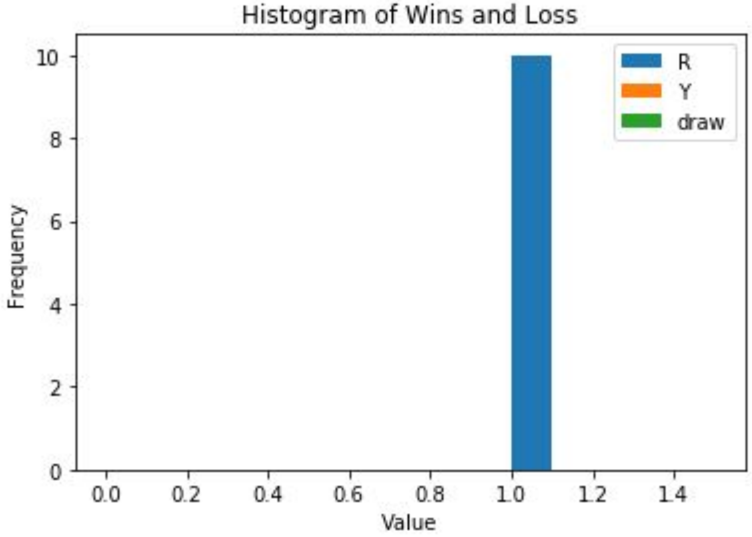
Win: 1000 - depth Draw: 0 Loss: -1000 + depth

Statistics

Depth of 4, 10 rounds:



Both use MinMax



R use MinMax, Y moves a random

Breakout

- Implemented breakout mechanics using PyGame. Strategy to move paddle is random.

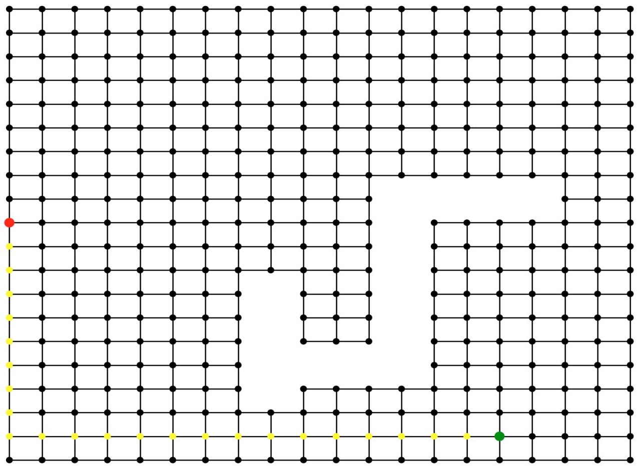
- Increasing speed of paddle over time.

Path Planning

Graph Reading

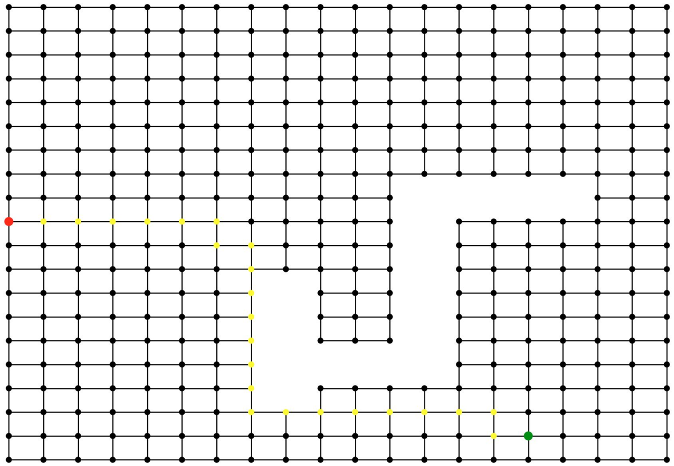
- Read data into a 2d array.
- Create a 2d grid graph using networkx.
- Create a mapping to relabel coordinates in the grid graph.
- Iteratively remove nodes with values “1”.

Path Planning



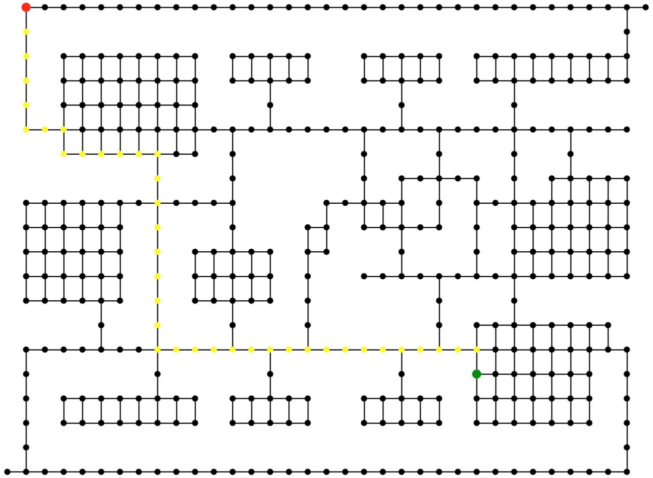
Dijkstra Path

(0,10) to (15, 1)



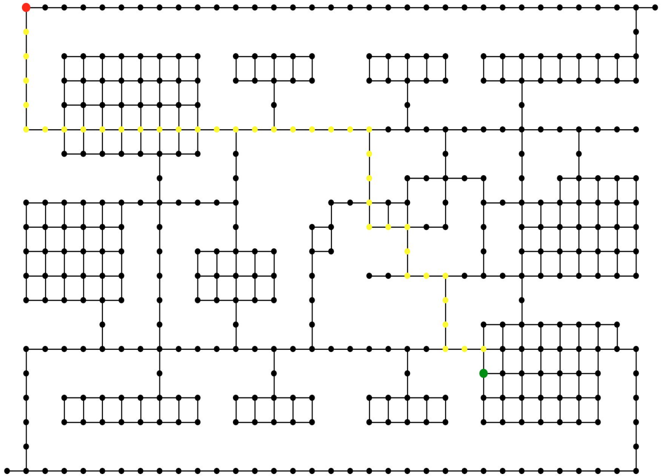
A* Path

Path Planning



Dijkstra Path

(1,20) to (25, 5)



A* Path

Conclusion

- GameTree
  - In Tic-Tac-Toe player 1 has an advantage over player 2.
  - However, if both player are the optimal game will always result in a draw.
- MiniMax
  - Exponential state space complexity.
- Connect-4
  - Player 1 has an advantage over player 2.
- Path-Planning
  - In an unweighted graph A* and Dijkstra find a path with same cost.

Questions.