TIC-TAC-TOE with AI opponent

Mentored by
Prof. Abhishek Singh
Adarsh Prasad Behra
Manas Gogoi

Done by
Ranjith Kalingeri - MIT2020017
Sushmanth Reddy - MIT2020061

TIC TAC TOE RULES

1. The game is played on a grid that has n squares by n squares.

2. First player use sign **X** and opponent (the computer in this case) is sign **O**. Players complete their turns by putting their sign in empty squares.

3. The first player to get n of her marks in a row (up, down, across, or diagonally) is the winner.

4. When all n^2 squares are full, the game is over. If no player has n marks in a row, the game ends in a tie.

x	x	x
	o	
		o

x	o	
x		o
x		

x	o	
	x	o
		x

SAMPLE CASES WHERE PLAYER WINS

x		
o	o	o
	x	

	x	o
x		o
		o

x		o
	o	x
o		

SAMPLE CASES WHERE AI WINS

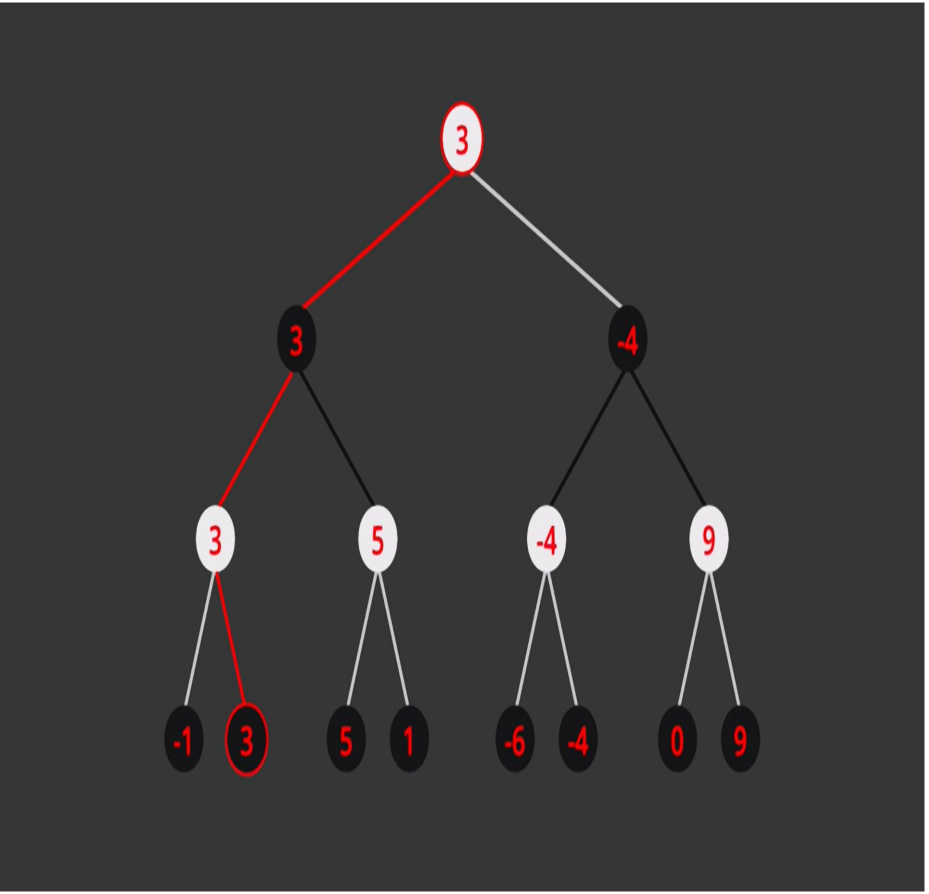
o	x	o
o	x	o
x	o	x

x	x	o
o	o	x
x	o	x

o	o	x
x	o	o
o	x	x

SAMPLE CASES FOR DRAW

Min - Max algorithm



At any point of time board can have a minmax value ranging from [min,max], if board value is max white player wins, if board value is min black player wins. If value is in between min and max game is in progress

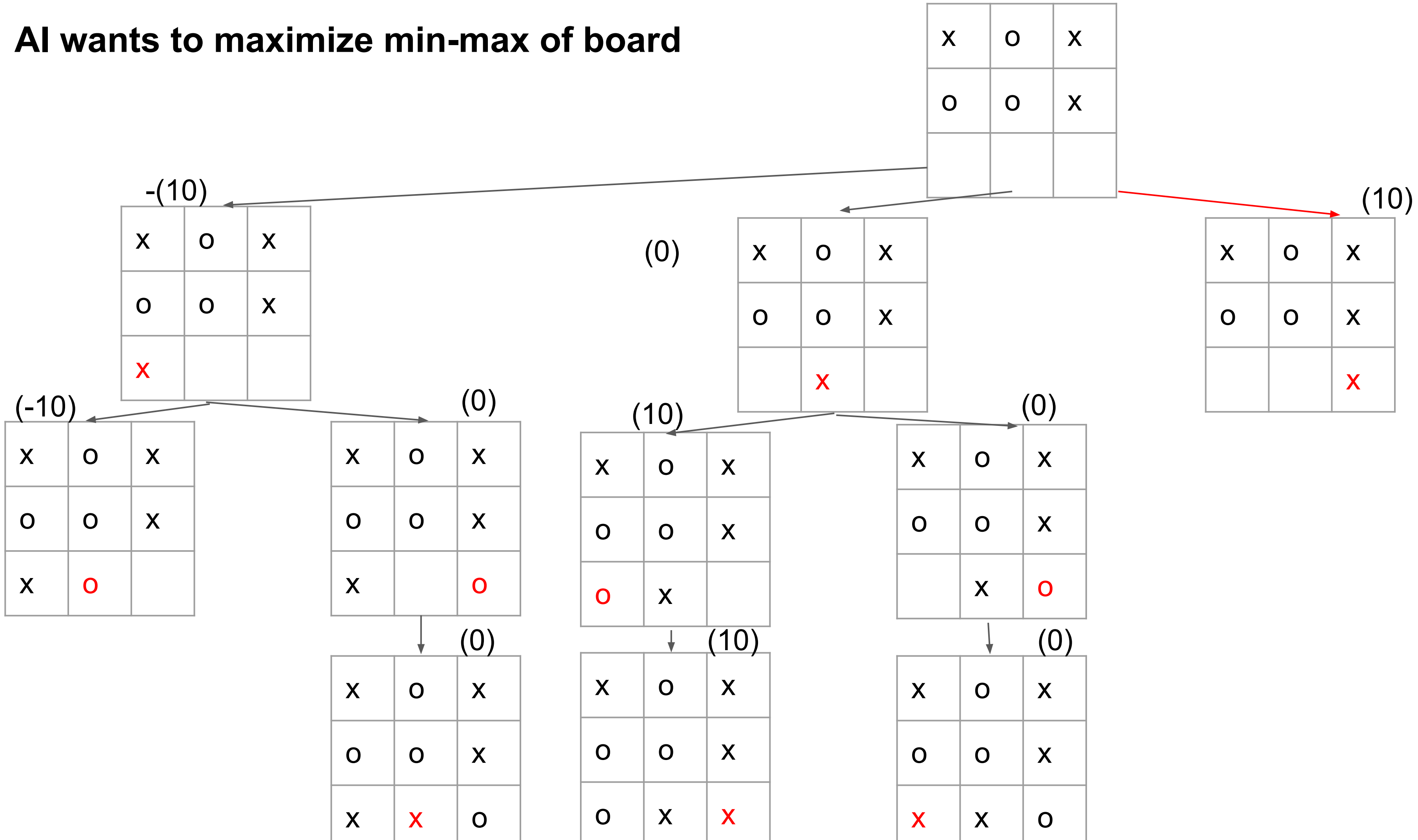
White players tries to maximize game state minmax value and black player tries to minimize game state minmax value by their respective moves

Here root node indicates that the minmax values of initial state of board that is 3.

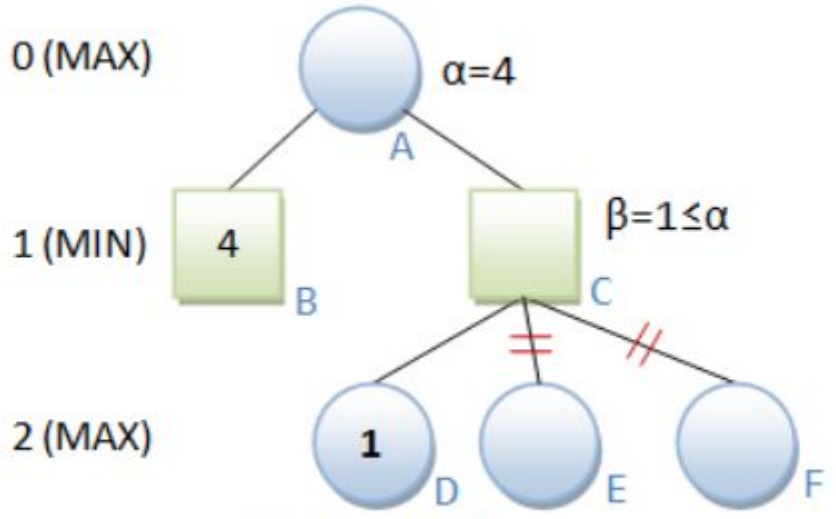
At root node it is white players turn he has two possibilities whether to make a left move or right move which modifies the minmax value of board to 3 and -4 respectively as white player tries to maximize min-max value of board state he makes a left move.

Similarly black player tries to minimize the min-max of board state so he makes left move

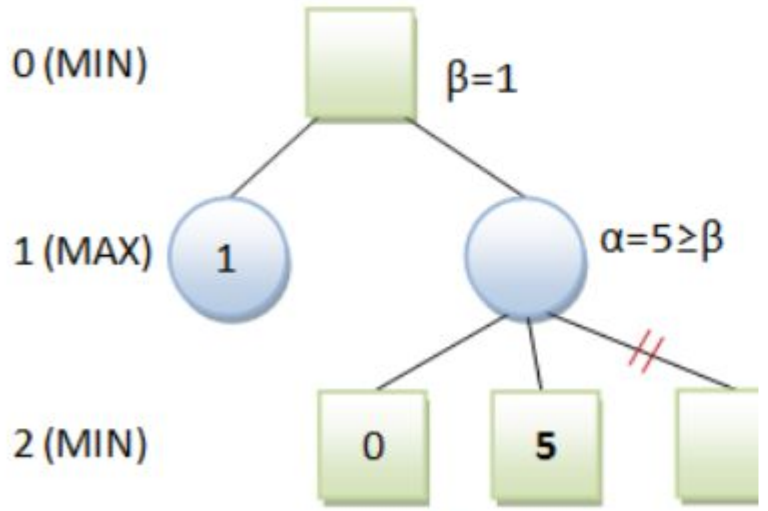
AI wants to maximize min-max of board



Min-max with Alpha-beta Pruning



Alpha Cut-off

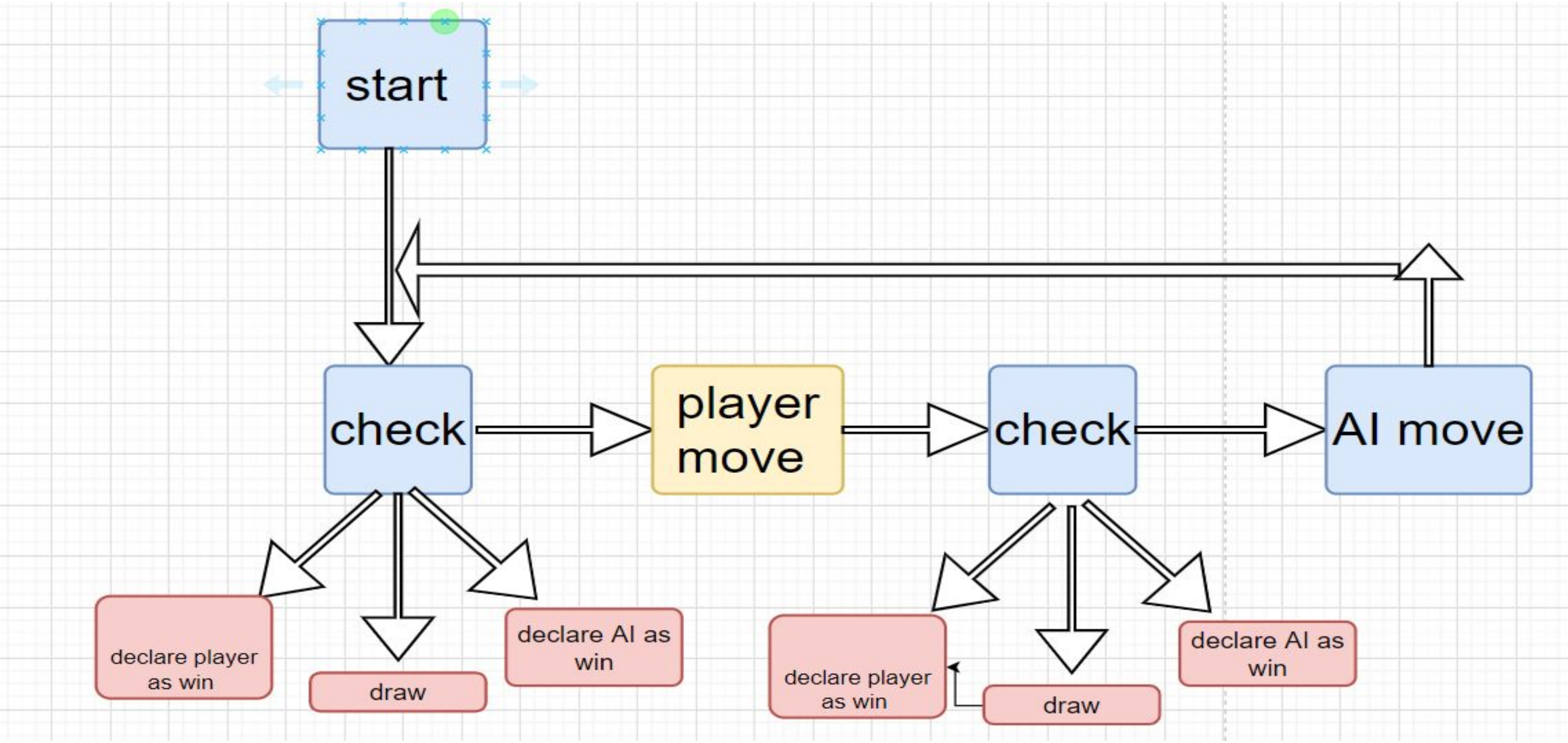


Beta Cut-off

this alpha cut-off, since node D returns 1, node C (MIN) cannot be more than 1. But node B is 4. There is no need to search the other children of node C, as node A will certainly pick node B over node C.

in the beta cut-off, since node E returns 5, node C (MAX) cannot be less than 5. But node B is 1. There is no need to search the other children of node C, as node A will certainly pick node B over node C.

FLOW CHART


start
check
player move
check
AI move
declare player as win
draw
declare AI as win
declare player as win
draw
declare AI as win

Class, score computation and Documentation

Class	**TicTacToe**
Class	**TicTacToeDriver**
Class	**Move**
Class	**WinUtil**

x	o	x
o	o	x
x	x	

Score: 200

- Java Documentation is created for the classes.
- Need for methods, parameters of methods and their return values are described
- Link to Documentation:
  - https://github.com/sushmanthreddymandi/TicTacToe/tree/master/documentation

Deployment and Individual Contribution

Individual contribution

- UI, 2 player game, winstate() and alpha-beta pruning : Sushmanth
- getScore(),Min-Max() and findbestmove() for TicTacToe AI : Ranjith

Total hours dedicated - 48 (18 hours for research, 4 hours for design and 16 hours for implementation).

Guide to run and install

- **Prerequisites:** Java SE 8
- **Packages used:** javax.swing, java.awt, java.awt.event
- **To compile:** javac *.java
- **To run:** Go to command prompt or terminal
    - execute **java TicTacToeDriver ai n** to play against a computer
    - execute **java TicTacToeDriver n** to play against a human

Where, n is size of grid

Best practices implemented & Future Plan

- Object oriented Programming, Abstraction, Encapsulation, Polymorphism, Code decoupling, Exception Handling, modularity, documentation, version control system are implemented to work with the project
- Project URL: https://github.com/sushmanthreddymandi/TicTacToe
- Minmax consumes lot of time and resources when size of grid is large
- Have to avoid patterns of the game where minmax fails
- Reinforcement learning algorithms can help us in solving these two issues
- Deploy this application as web application
- Implement save and quit option
- **To Design an online algorithm that learns from the game patterns**

Thank You