

# Connect 4 'in Java' { [Single Player Game]

< By: Pranav Sitaraman and Alan Shi >

}

# Table Of 'Contents' {

## 01 Basics

< Intro to Project >

## 02 Computer Algorithm

< Minimax & Alpha-Beta Pruning >

## 03 Frontend

< Built with JavaFX >

}

01 {

[Basics]

< Intro to Project >

}

## Game Play < /1 > {



< Connect 4 is a simple game in which the player & computer take turns dropping pieces down a 6×7 board in order to connect 4 in a row >

}

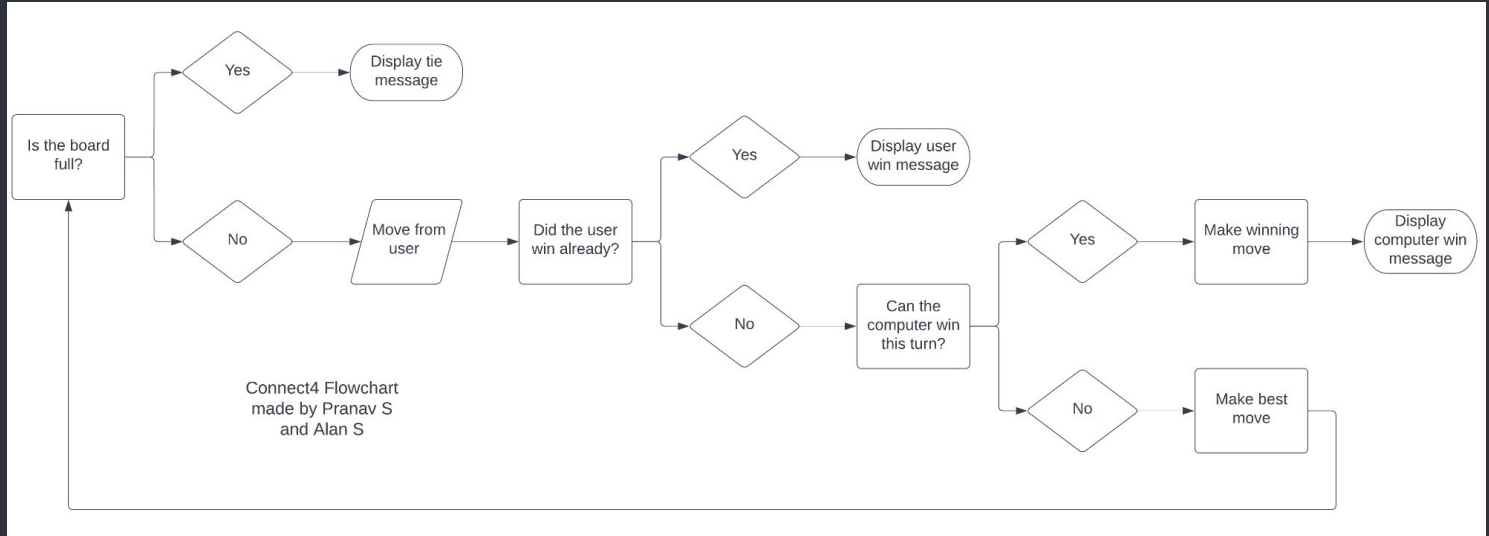
## Outline < /2 > {



< Written in Java, with a JavaFX frontend >

}

# Flowchart; {




}

# Project 'Breakdown' {

## Time Spent

Backend  60%

< Developing minimax  
algorithm to calculate  
computer move >

Frontend  40%

< Create JavaFX  
interface and board  
animation >

## Features



Relatively  
quick



Played with  
keyboard



Easy to use  
UI

}

# Code Layout; {

}



02 {

[Computer Algorithm]

< Minimax & Alpha-Beta  
Pruning >

}



# Computer Algorithm: 'Minimax' {

## Step 01

Depth-first-search (DFS) evaluation structure that alternates maximizing computer evaluation and minimizing player evaluation

## Step 02

Currently pruning game-tree below a depth of 9 to produce suitable runtime, but fares better in C++

## Step 03

Requires robust evaluation function: current evaluation function is state-based

## Step 04

Theoretically perfect but limited by game tree depth and would likely be better if heuristic

}

# Computer Algorithm: 'Minimax' {

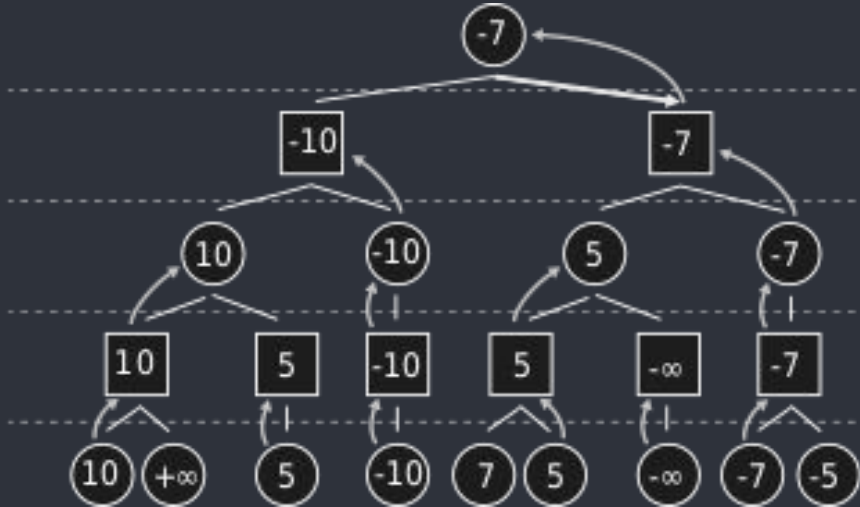
0 (max)

1 (min)

2 (max)

3 (min)

4 (max)



```
1
2
3
4 4,531,985,219,092 {
5
6
```

```
7 |
8   < Potential Game States (Complexity) >
9 |
```

```
10 }
11
12
13
14
```

# Computer Algorithm: ' $\alpha$ - $\beta$ Pruning' {

## Step 01

Pruning of the game tree to increase speed if a minimizing node minimizes below its parent's previous maximum, or vice versa

## Step 02

Any smaller values would obviously not be maximized by parent, while larger values would not be minimized by child node

## Step 03

Extremely beneficial for performance, results below show comparison at depth 7

With Alpha-Beta: 242733

With Alpha-Beta: 684904

With Alpha-Beta: 711719

With Alpha-Beta: 363515

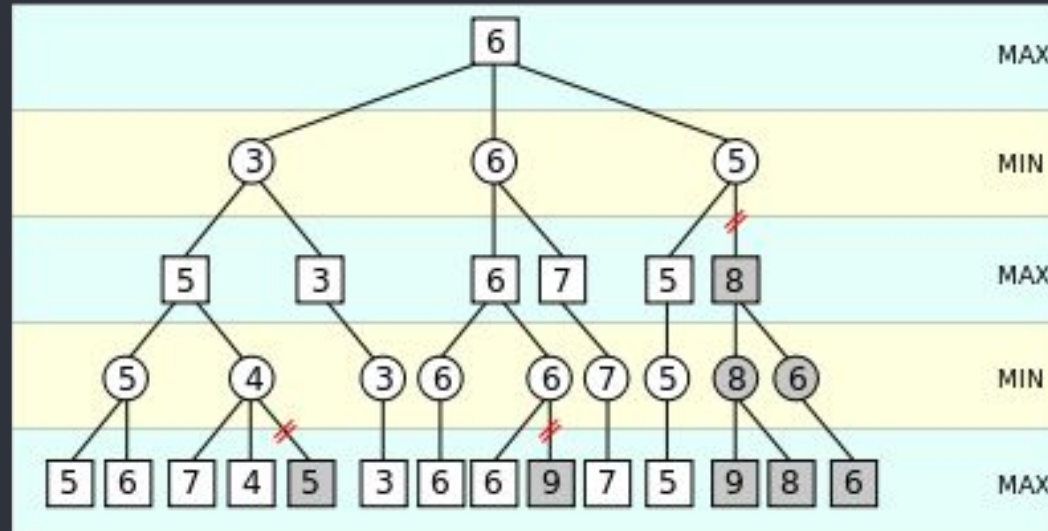
Without Alpha-Beta: 25981394

Without Alpha-Beta: 23967904

Without Alpha-Beta: 23908150

Without Alpha-Beta: 22278365

# Computer Algorithm: ' $\alpha$ - $\beta$ Pruning' {



03 {

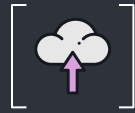
[Frontend]

< Built with JavaFX >

}

# Features of 'The Frontend' {

## Column Select



< Switching between columns with the arrow keys >

## Game Play



< Enter to drop pieces, with smooth animations >

## Eval Mode



< Computer evaluation updated and displayed >

## Game Lines



< Shows predicted computer 'lines' >

}

# The Animation; {

## Difficulties



< Had a lot of trouble with smooth animations while calculating computer move, as delays blocked the JavaFX thread >

## Methods Tried



< Attempted to bind the computer move algorithm to the end of the transition but was unsuccessful >

## Solution

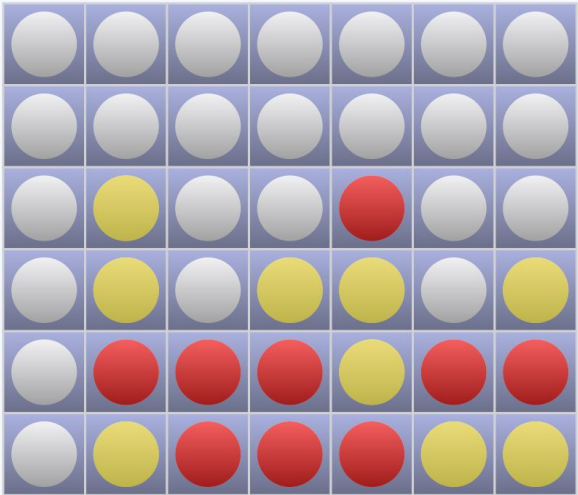


< Use of Java CompletableFuture< > to calculate computer move asynchronously on different thread pool >

}



1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14



-9	-9	-9	-9	-9	-9	-9
----	----	----	----	----	----	----

## Connect 4

Test your Connect 4 skills against a computer!

Use the arrow keys to navigate between columns and enter to drop your piece!

Click the button below to turn on engine mode!

Gameplay Mode

(1, 1) - [(3, 3)]  
(2, 5) - [(3, 3)]  
(3, 3) - [(3, 4)]  
(4, 4) - [(3, 3)]  
(5, 5) - [(3, 3)]  
(6, 3) - [(3, 3)]  
(7, 4) - [(3, 3)]

04 {

[Summary]

< Summary and Demonstration  
of our Project >

}

# Future Goals {

< Future considerations to improve performance of the computer algorithm >

- < /1 > \* Implement more heuristic evaluation algorithm
- \* Use the null-move heuristic algorithm to prune tree
- < /2 > \* Use iterative deepening of the minimax DFS to explore deeper depths
- \* Cache upper and lower bounds of board states in a transposition table

}

< and finally, our app >

Connect4 Game in Java  
Pranav Sitaraman and Alan Shi