Homework

Mon Oct Aug Tues

→ Mon Aug 21 Intermediate

Tue

Aug 21 Mon

Tue

Oct 21 Monday

Oct 31 Tue

Schema Design Exercise

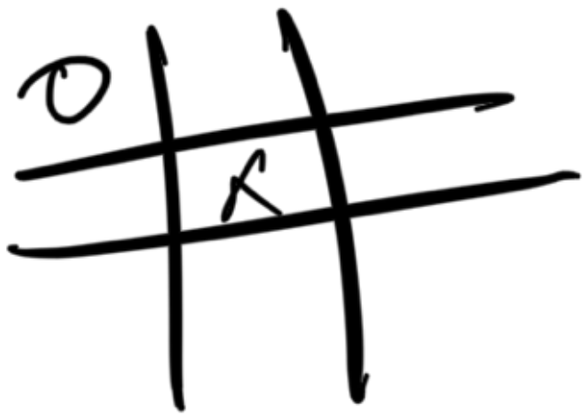
HW

→ Netflix Schema Design

Ctrl + Shift + .

..pdf

LLD Design



Support @ Scaler.com

① Entity

→ Pen

② Game

→ TicTacToe

→ Snake & Ladder

→ Chess

→ Card Game

③ Key Real Systems

→ BMS

→ Parking Lot

Ranking

→ Matching

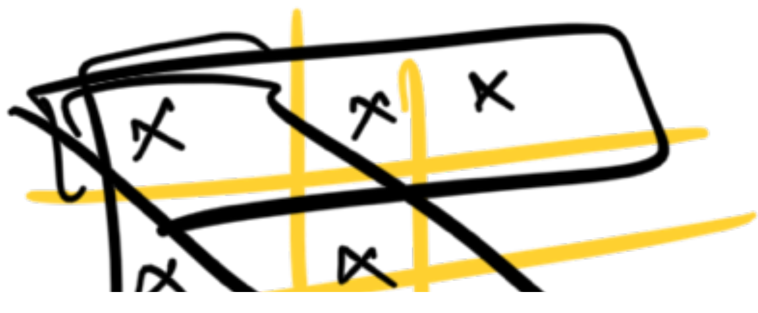
3 classes → → Splitwise

④ Engineering Problem

→ Distributed Cache

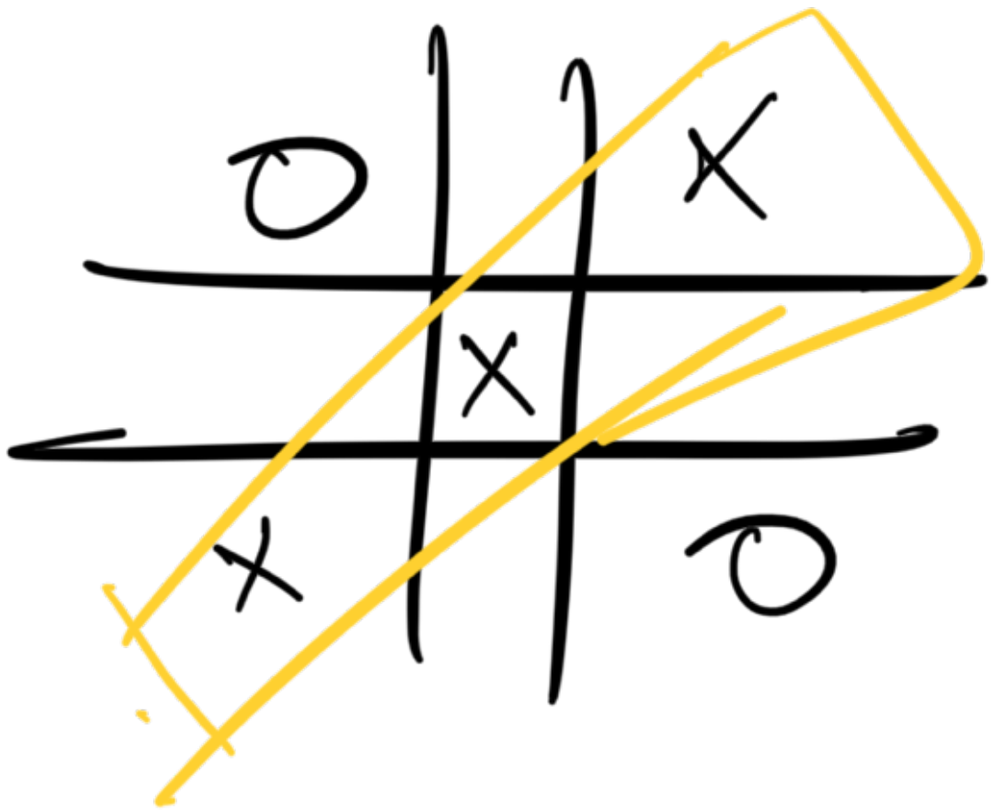
Design Tic Tac Toe

O X



TikTok

Turn 1 X



2 O

3 X

4 O

5 X

Gathering Requirements

⇒ Computer App^n

⇒ There can be any #of players

⇒ ⇒ There can be Bots (Computer Player)

⇒ No timeout support in VO

⇒ Every player chooses a symbol

⇒ There can be different ways to win

→ Always to

→ Support undo feature

implement undo

Features

① Computer Appⁿ

② there can be Any # of players

→ Ratio (computer players)

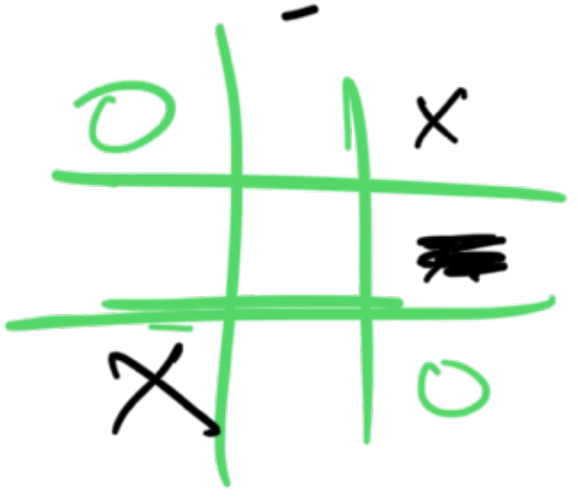
→ Different difficulty levels

③ There can be [illegible]

④ Every player chooses a symbol → any char is a valid symbol

⑤ There can be diff ways to win (not just row/col²/ diag) → One char can be in only one

⑥ Support undo operation → 2 ways to implement undo



⑦ grid can be of any $n \times n$ dimension

Going into details

Clarifying the Requirements

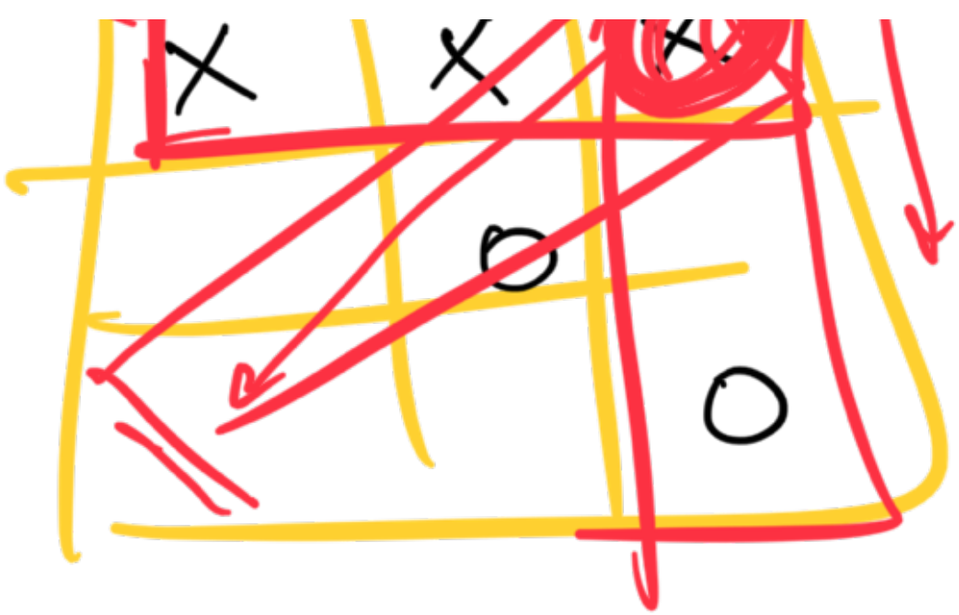
⇒ Undo for only 1 step

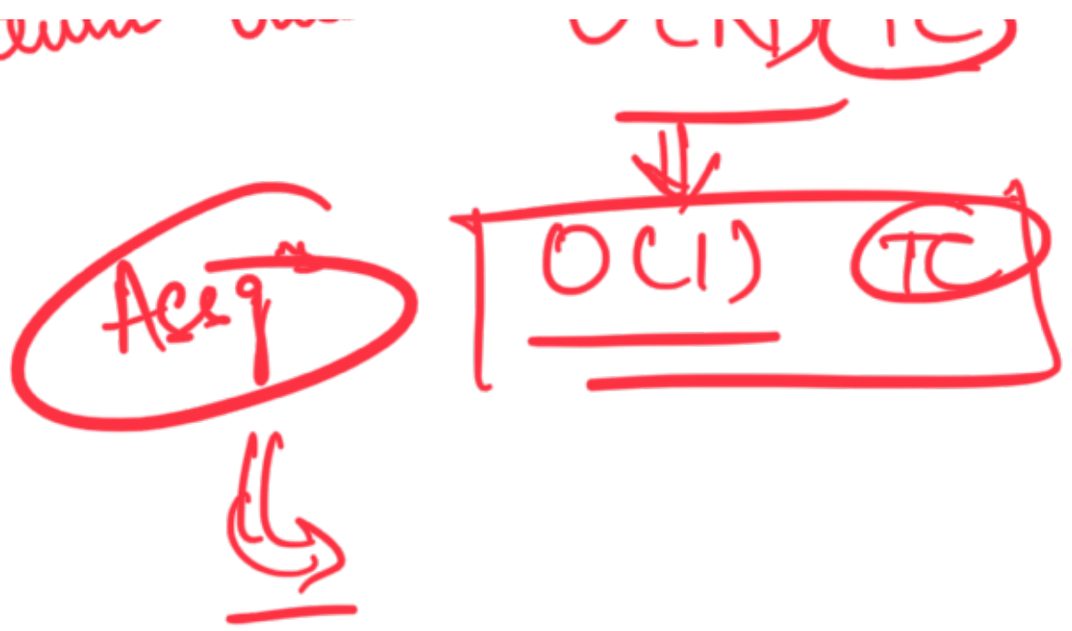
often in LCS problems, there is a Hidden

DAG problem ① TTT

② Splitwise

⇒ Try to implement win detection in O(1)





→ One game can have at max 1 Bot

6 Players (Only 1 Bot Max)

→ Implement whole game via manifunctions

Creat Game

↓

2 Human Players

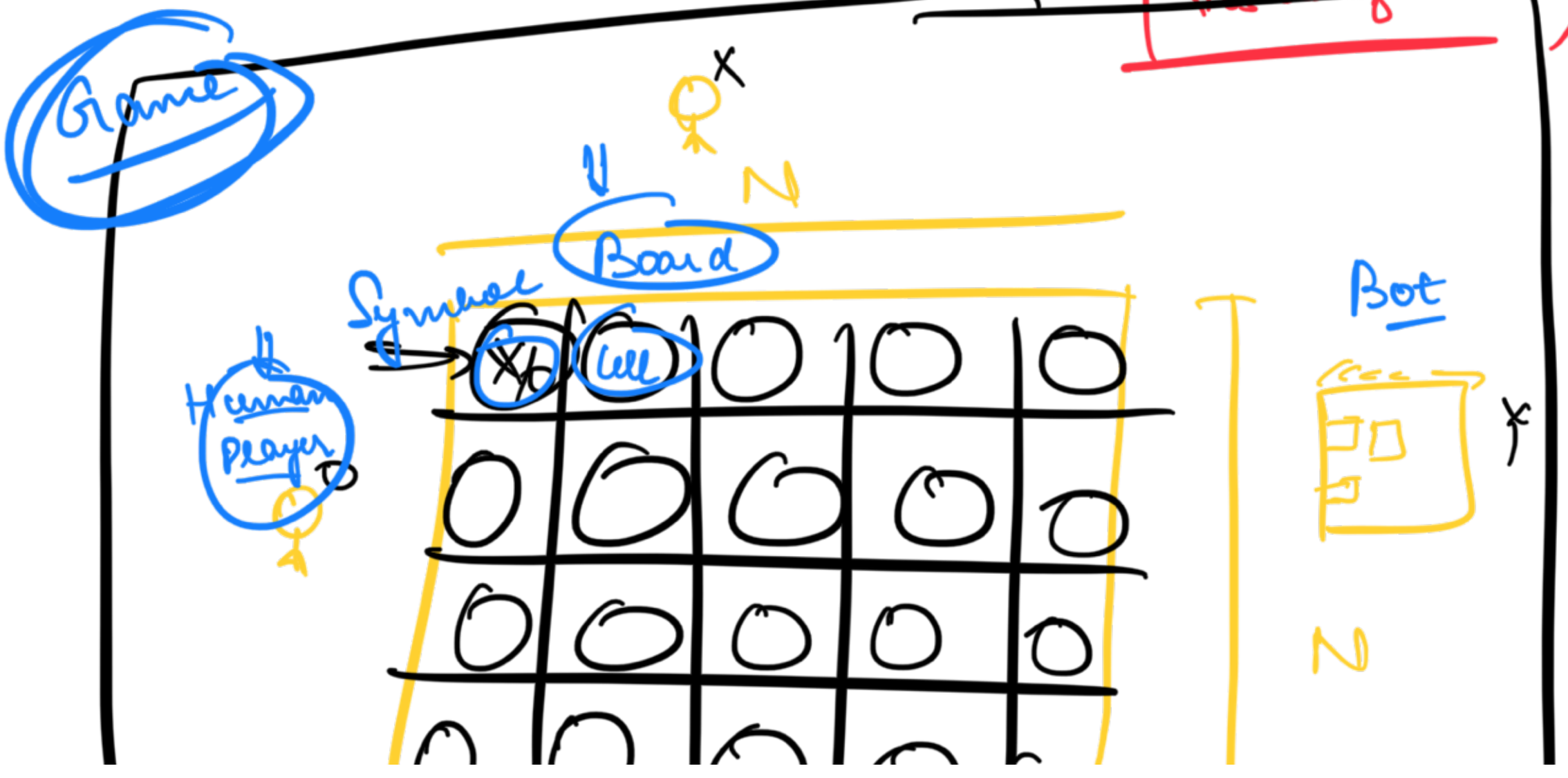
Play on Their Behalf =

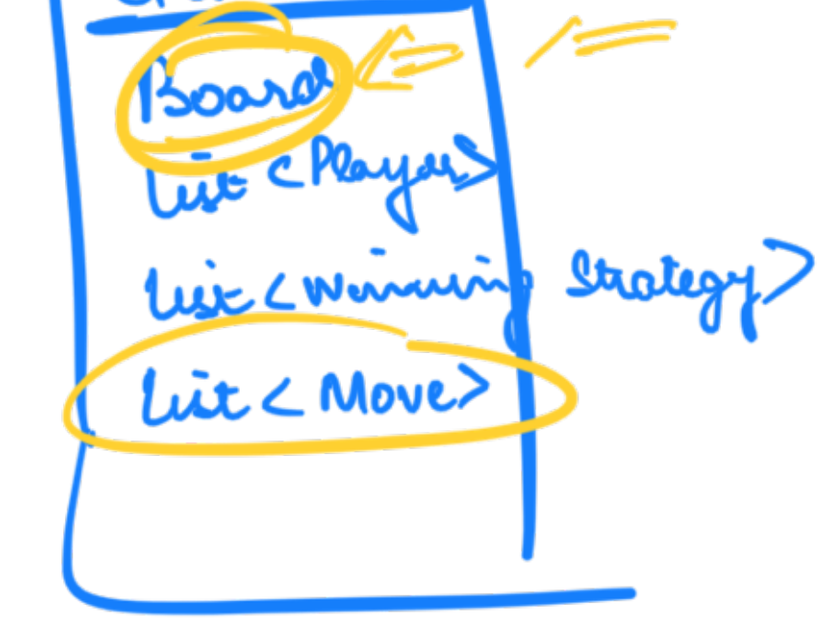
Check who won

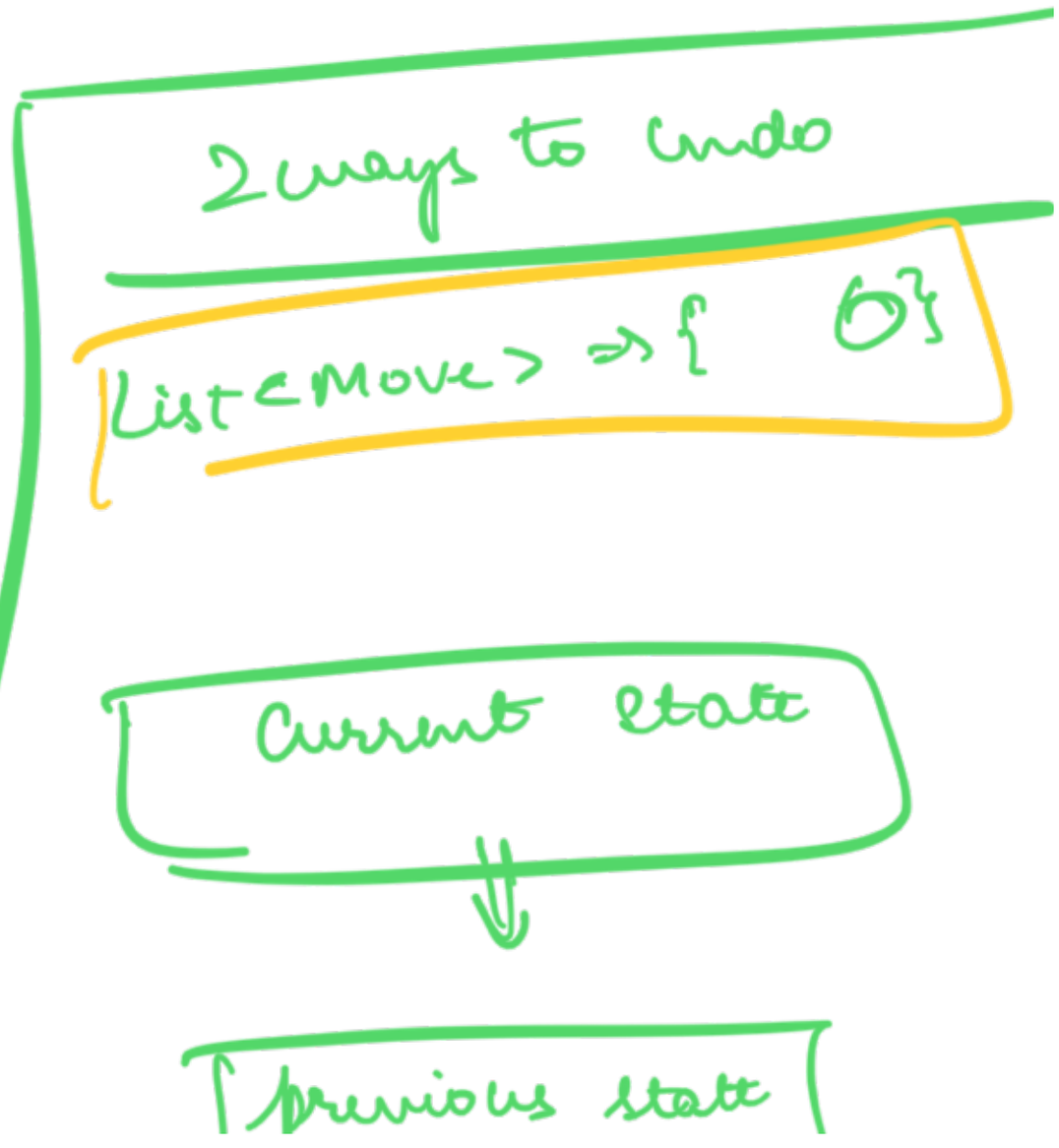
Class Diagram → Know the entities

→ Nouns

→ Verbs



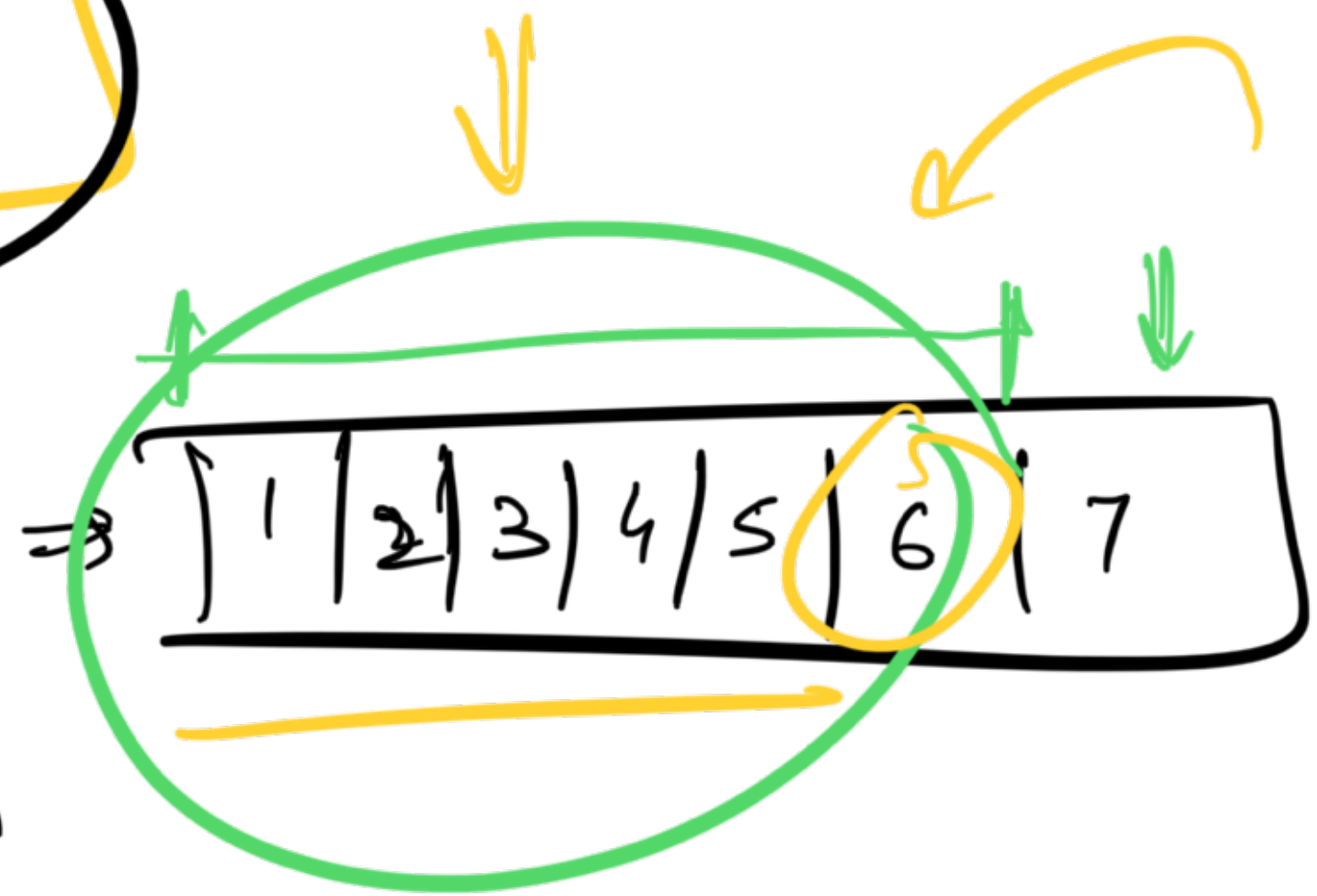

Game
Board
List < Player >
List < Winning Strategy >
List < Move >


2 ways to undo
List < Move > → { }
Current state
Previous state

① Recursion Do reverse of the last move and remove last move from the list.

② Start the game as new and run moves from 0 to n-2

moves = 1, 2, 3, 4, 5, 6, 7

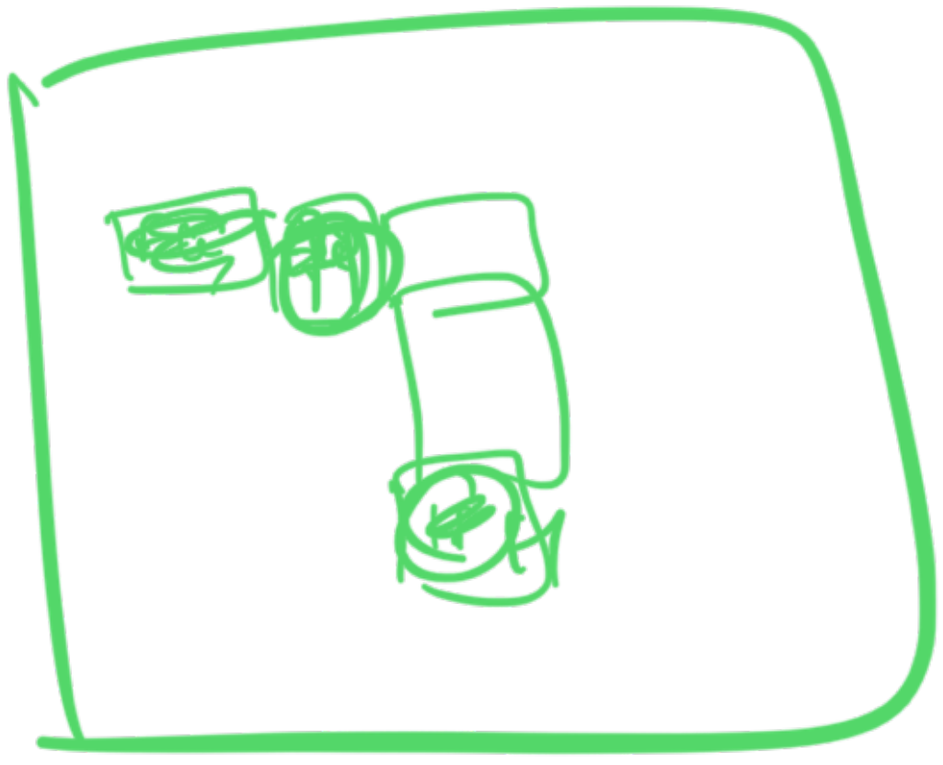


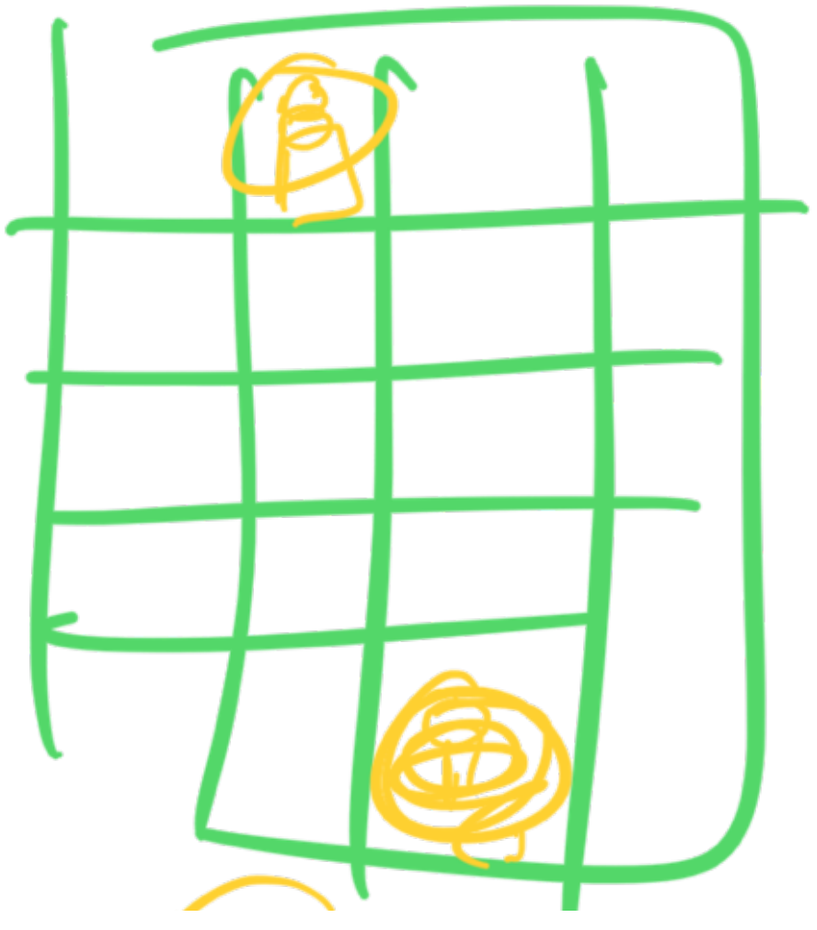
Current Board

undo

Clean Board


Elephant



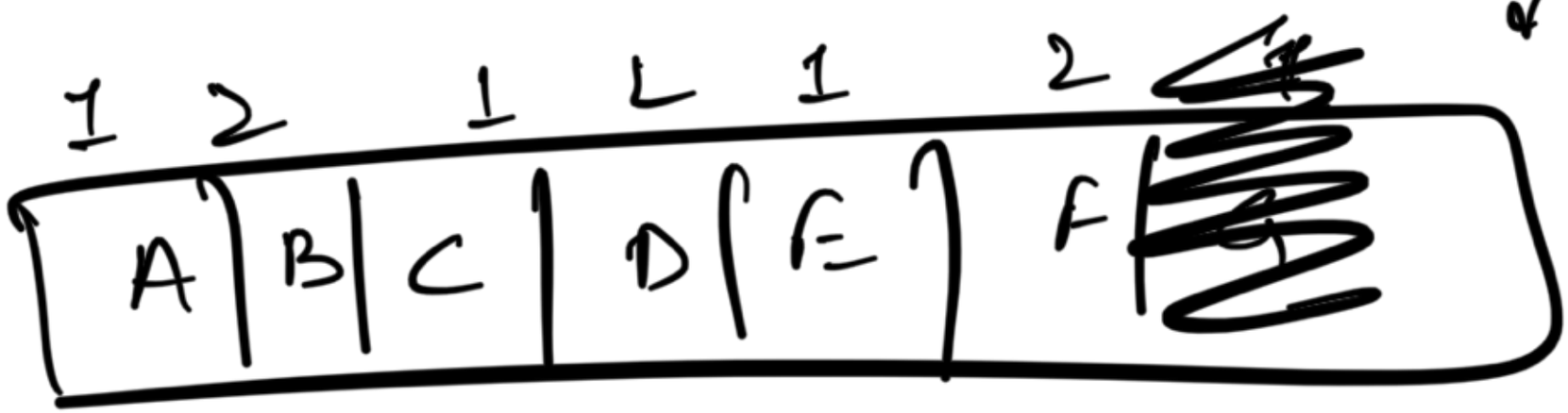


Undo

① Reverse the last op^{n} (eg TT)

② Run all ops other than the last one

(eg Cuts)



30

29

Rebuild

Game

Moves

Board

Status

- IN_PROGRESS
- DRAW
- WON

Game

- List<Player>
- Board
- List<WinningStrategy>
- lastPlayerIndex
- GameStatus
- Winner

Move

- Symbol
- Cell
- Player
- getSymbol()

Player

- Symbol

COMPUTER

Associate

Type : Enums

HUMAN

Human Player

User

Bot

Playing Strategy

Char

User

name

email

phoneNumber

Board

dimension

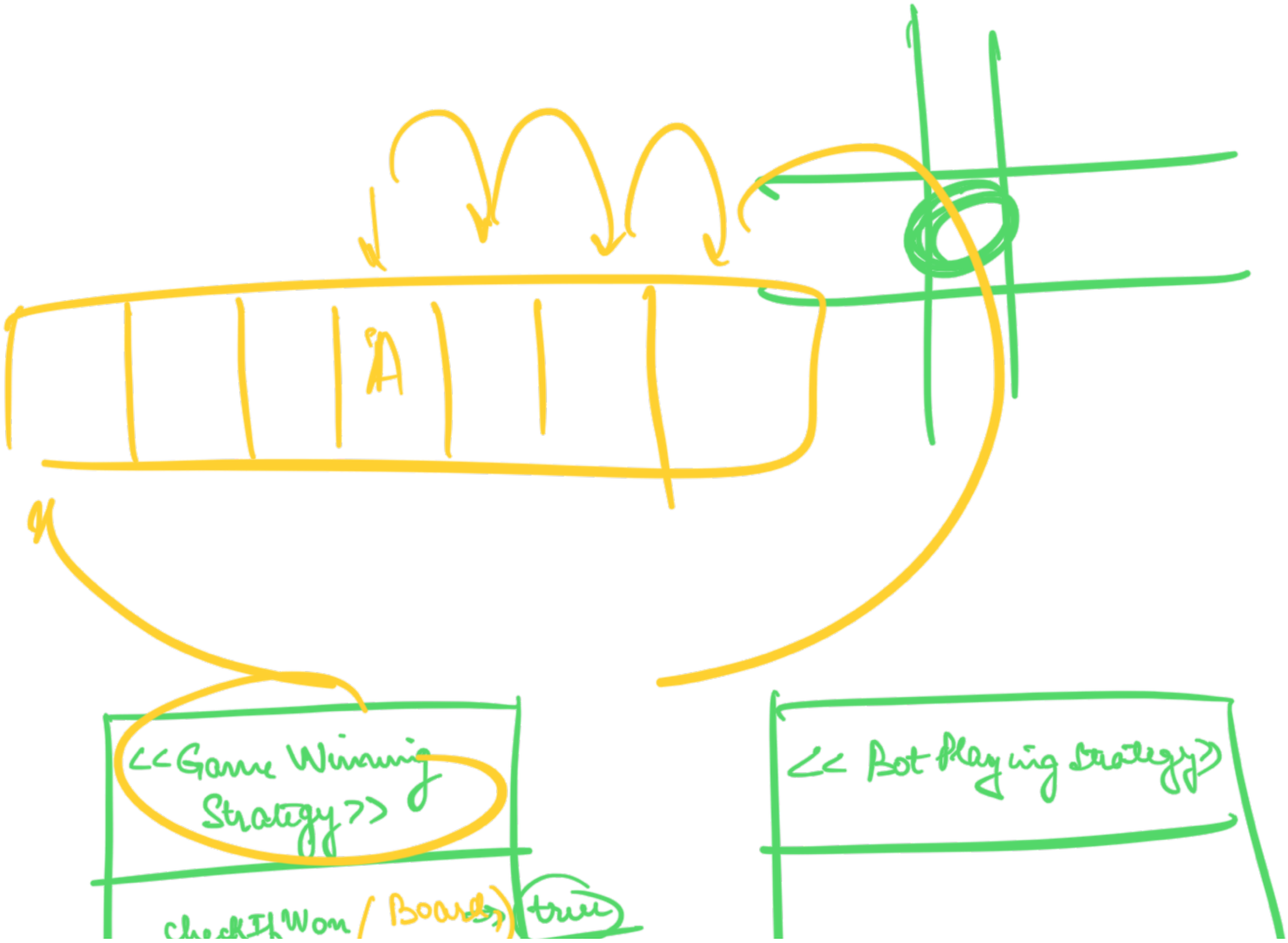
List<List<Cell>> board

Cell

row

col

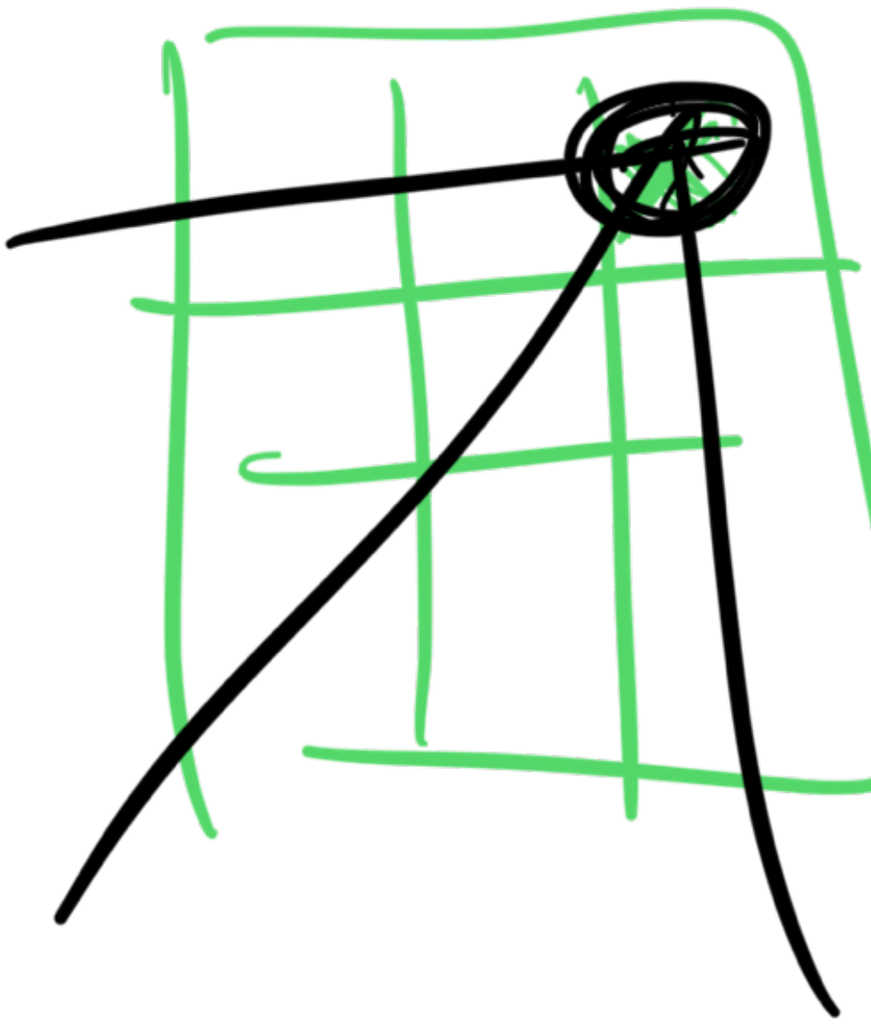
Symbol → can be null

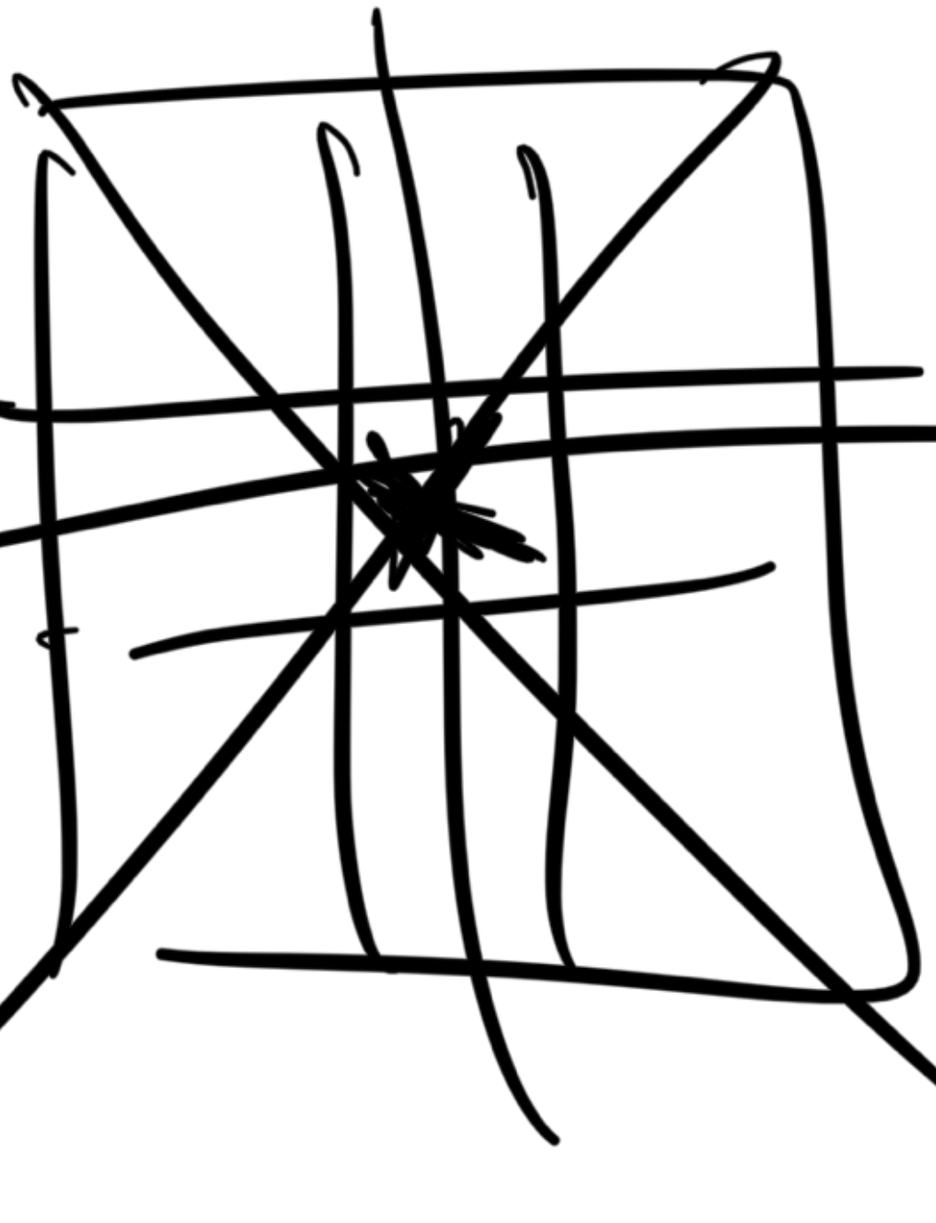

A
<< Game Winning Strategy >>
checkIfWon / Board) true
<< Bot Playing Strategy >>

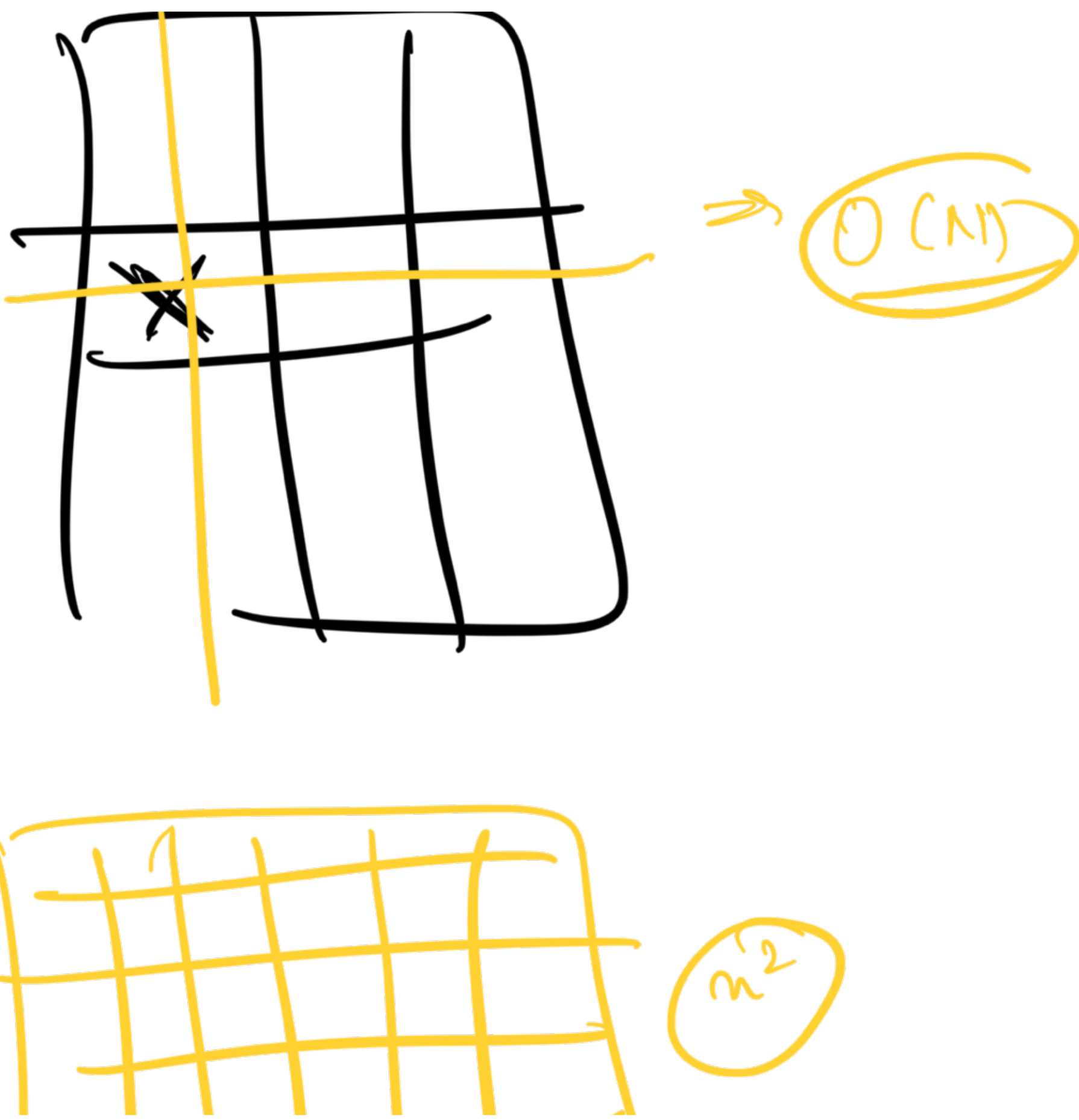
Cell

Cell where last Move Happened

O(1)






O(N)
n²

→ Normal

Strategy

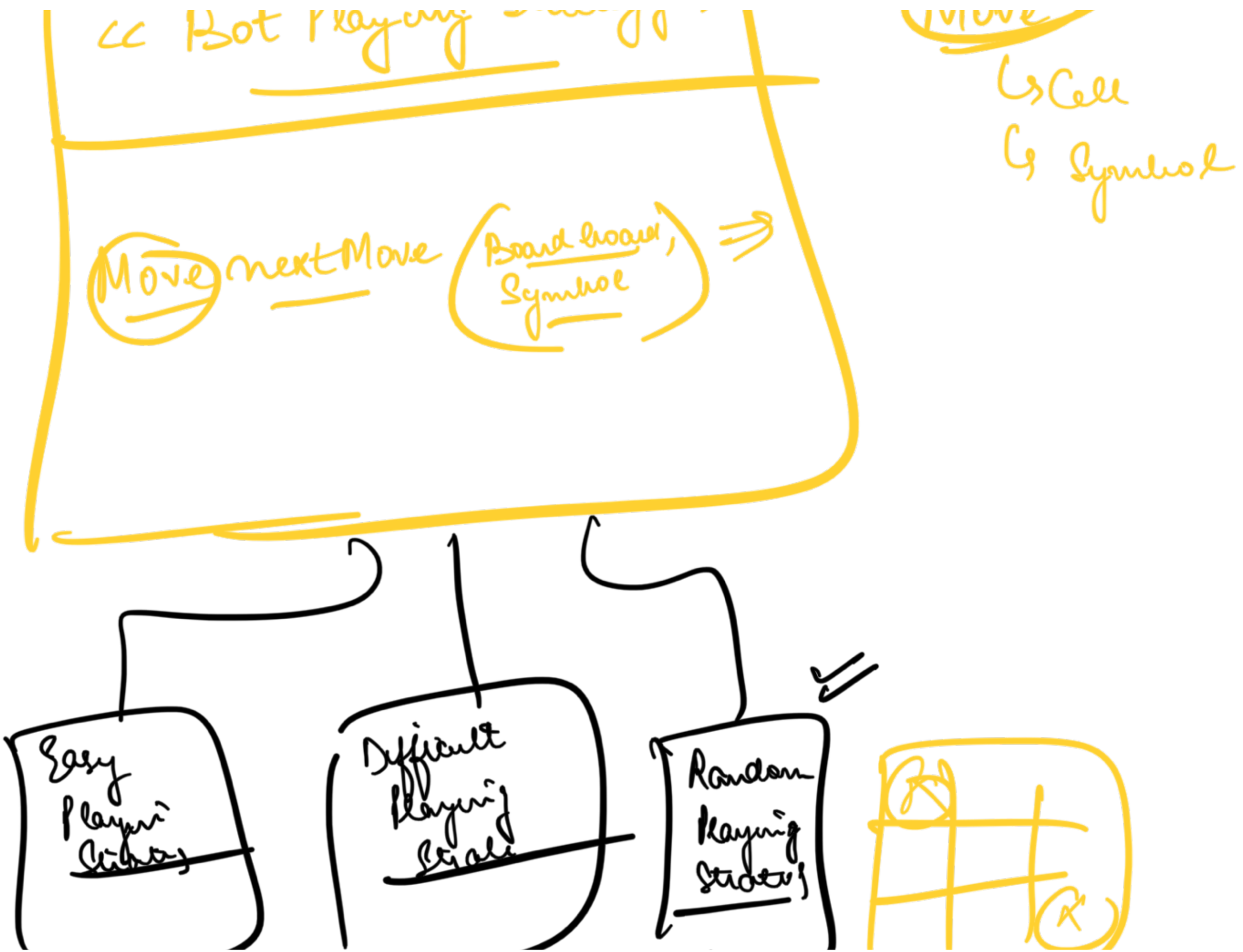
Normal Winning Strategy

Corner winning Strategy

→ Corner

Strategy

Move


Bot Playing
Move
Cell
Symbol
Move nextMove (Board board, Symbol
Easy Playing Strategy
Difficult Playing Strategy
Random Playing Strategy

Assignment

1. Implement models

   models /

   Strategies /

2. For models, only implement attributes

3. For strategies write code for following

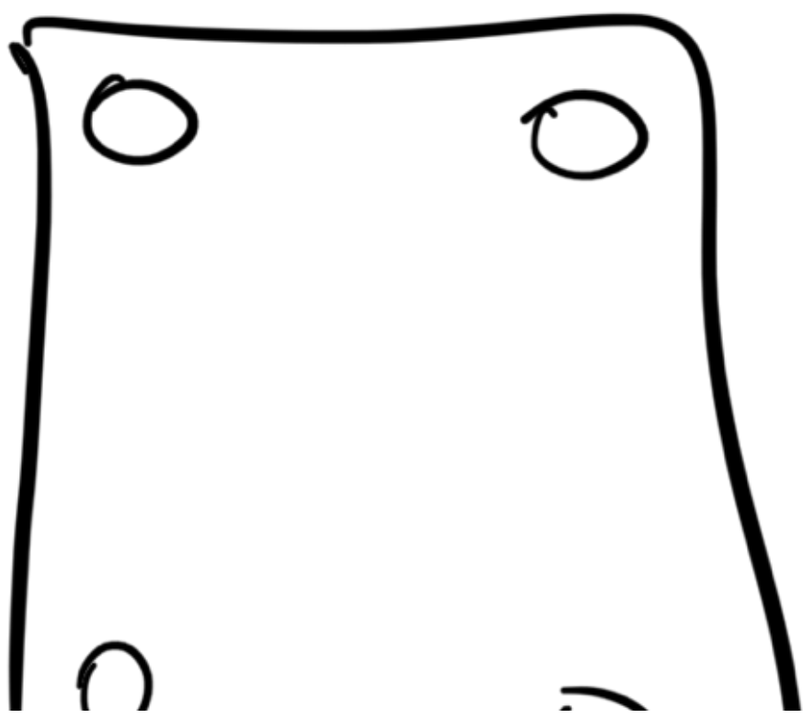
(3) For Strategy

Strategy

(1) Random Move Strategy

(2) Normal Game Winning Strategy

(3) Normal Game Winning Strategy One



(4) In game class implement method called undo()

move()