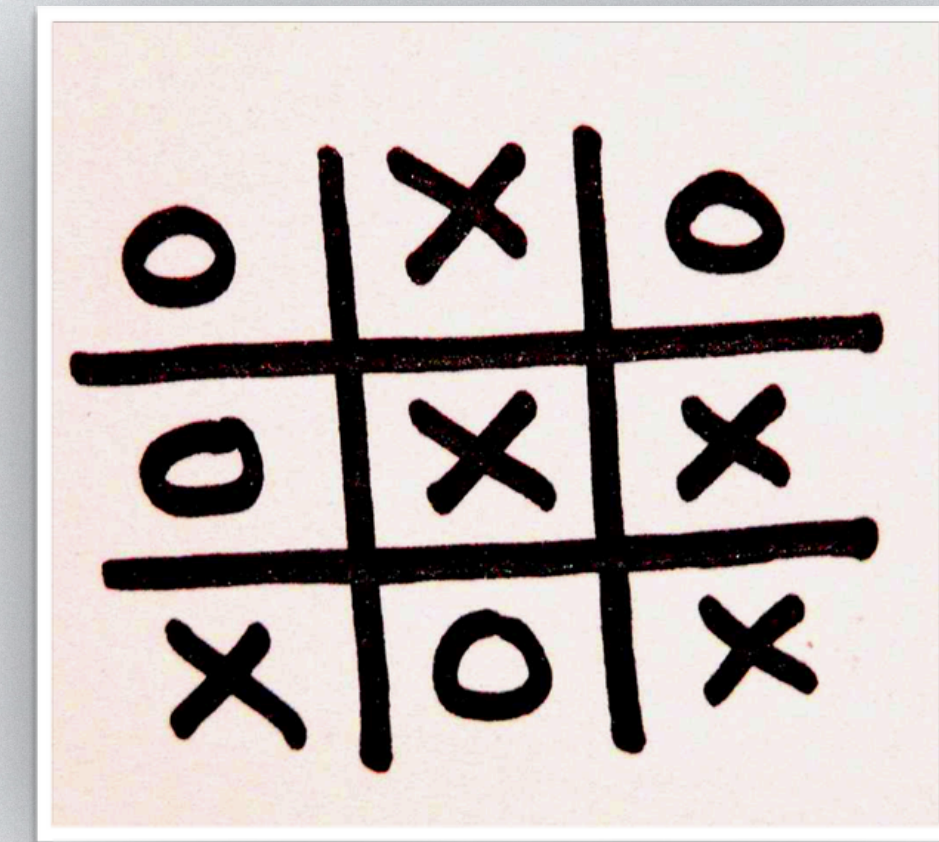
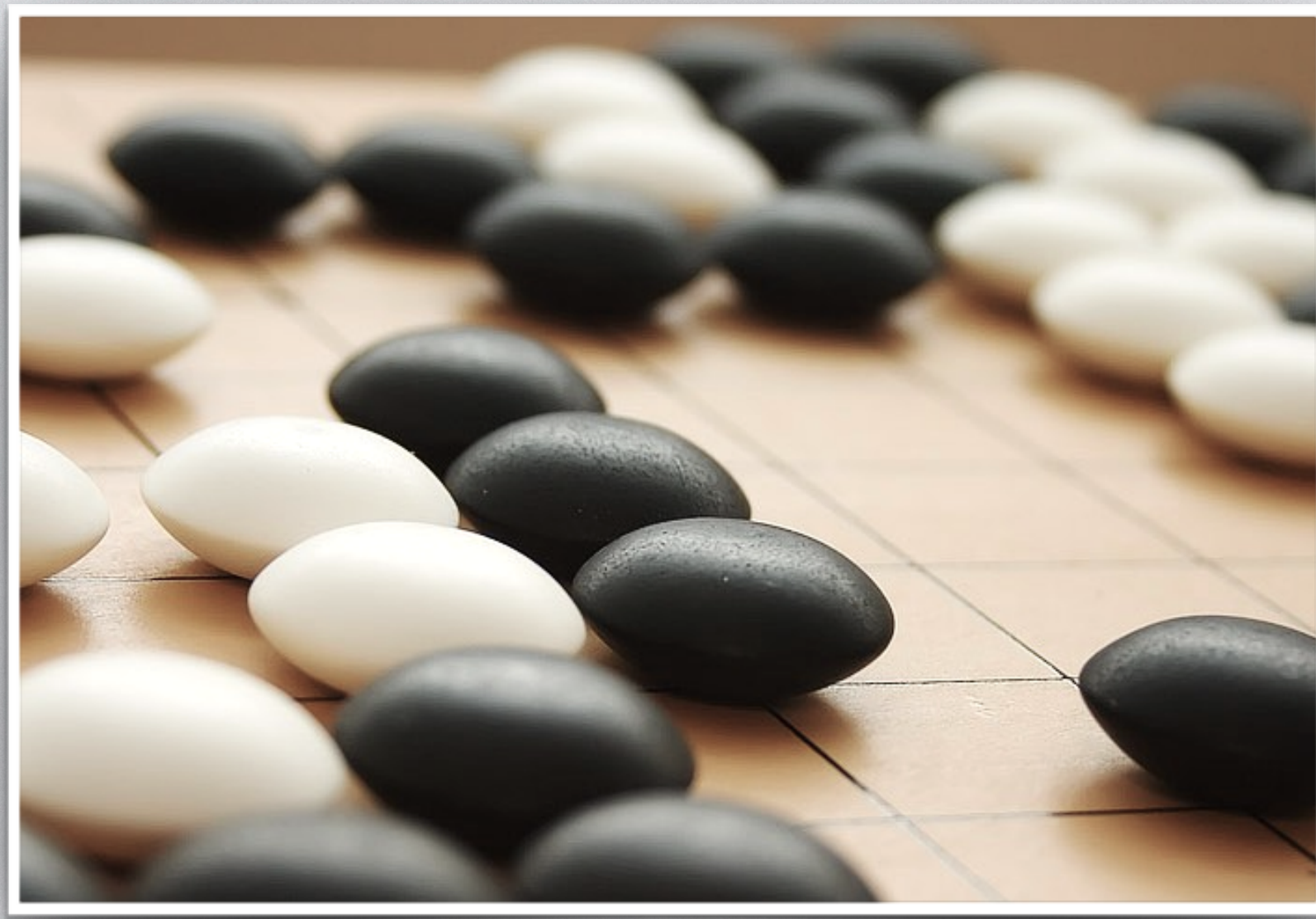


FROM **TIC TAC TOE**



TO

**ALPHA ZERO**





# TOPICS

- **SIMPLE TREE GAME** (TREE SEARCH, MINI-MAX)
- **NOUGHTS AND CROSSES** (PERFECT INFORMATION, GAME THEORY)
- **CHESS** (FORWARD/BACKWARD AND ALPHA/BETA PRUNING)
- **GO** (MONTE CARLO TREE SEARCH, NEURAL NETWORKS)





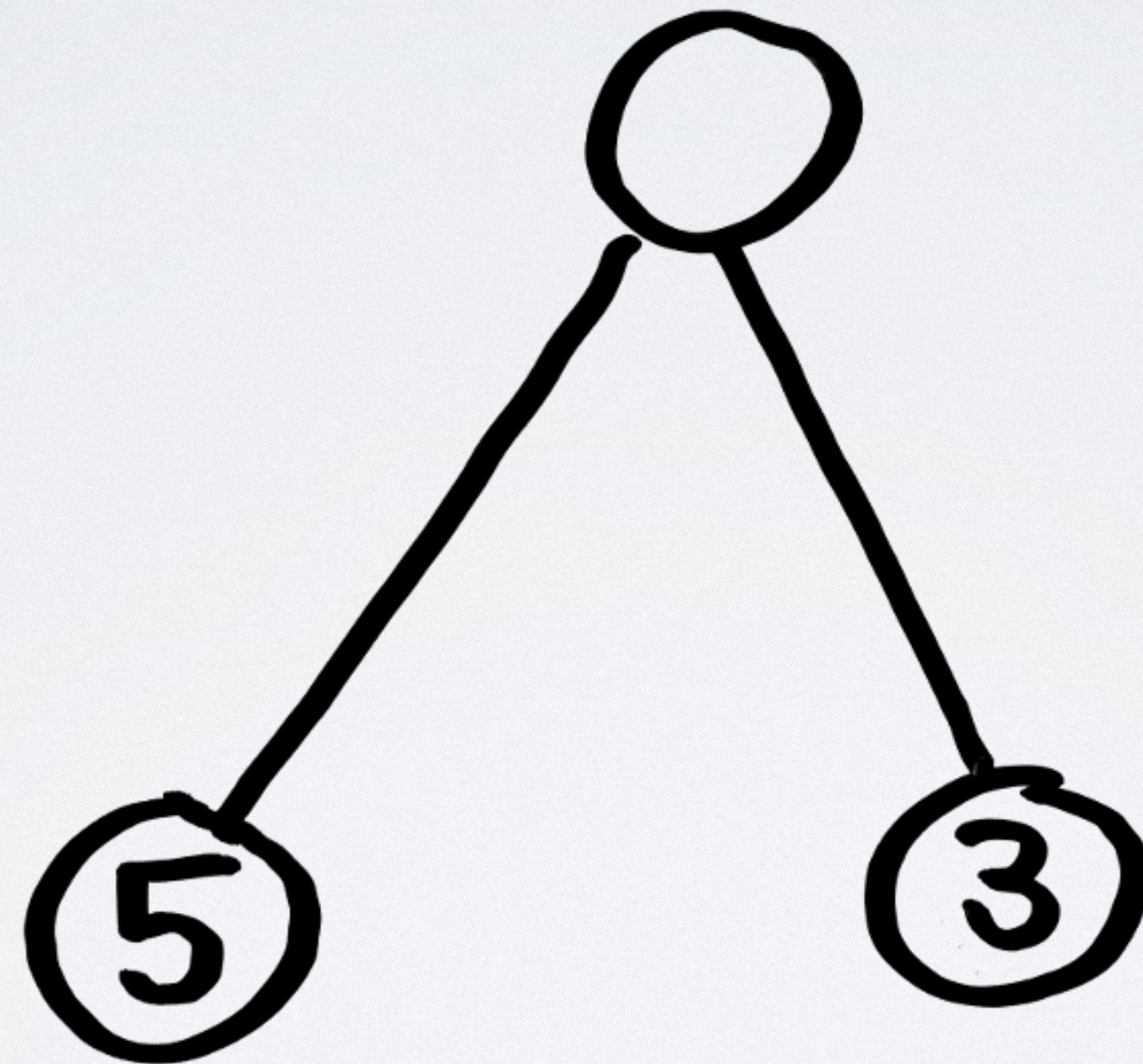
# I WANNA PLAY A GAME...

- TREE-STRUCTURE
- **YOU** ALWAYS START
- HIGHEST SCORE **WINS**



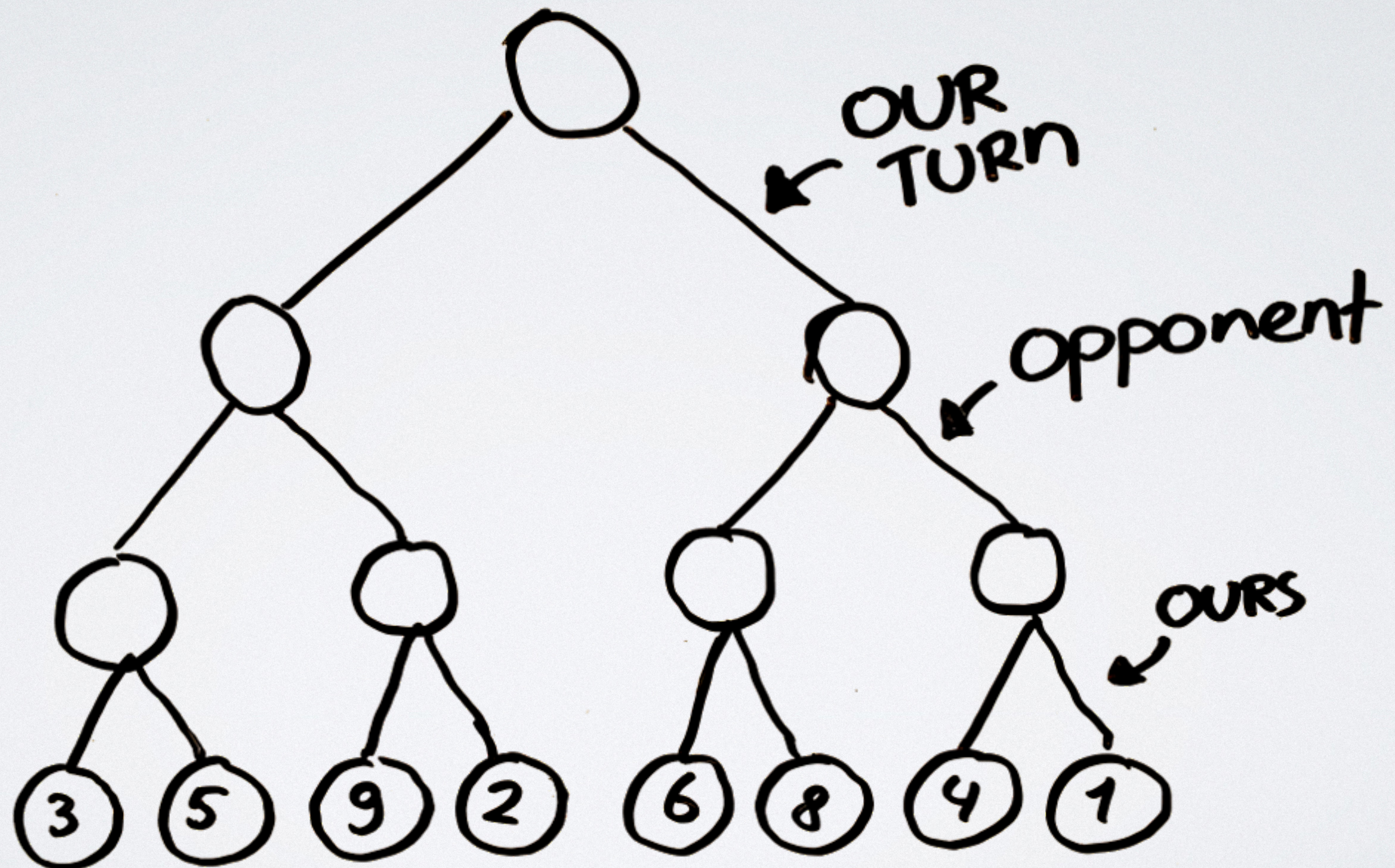


DEPTH:  $N=1$





# DEPTH: $N=3$





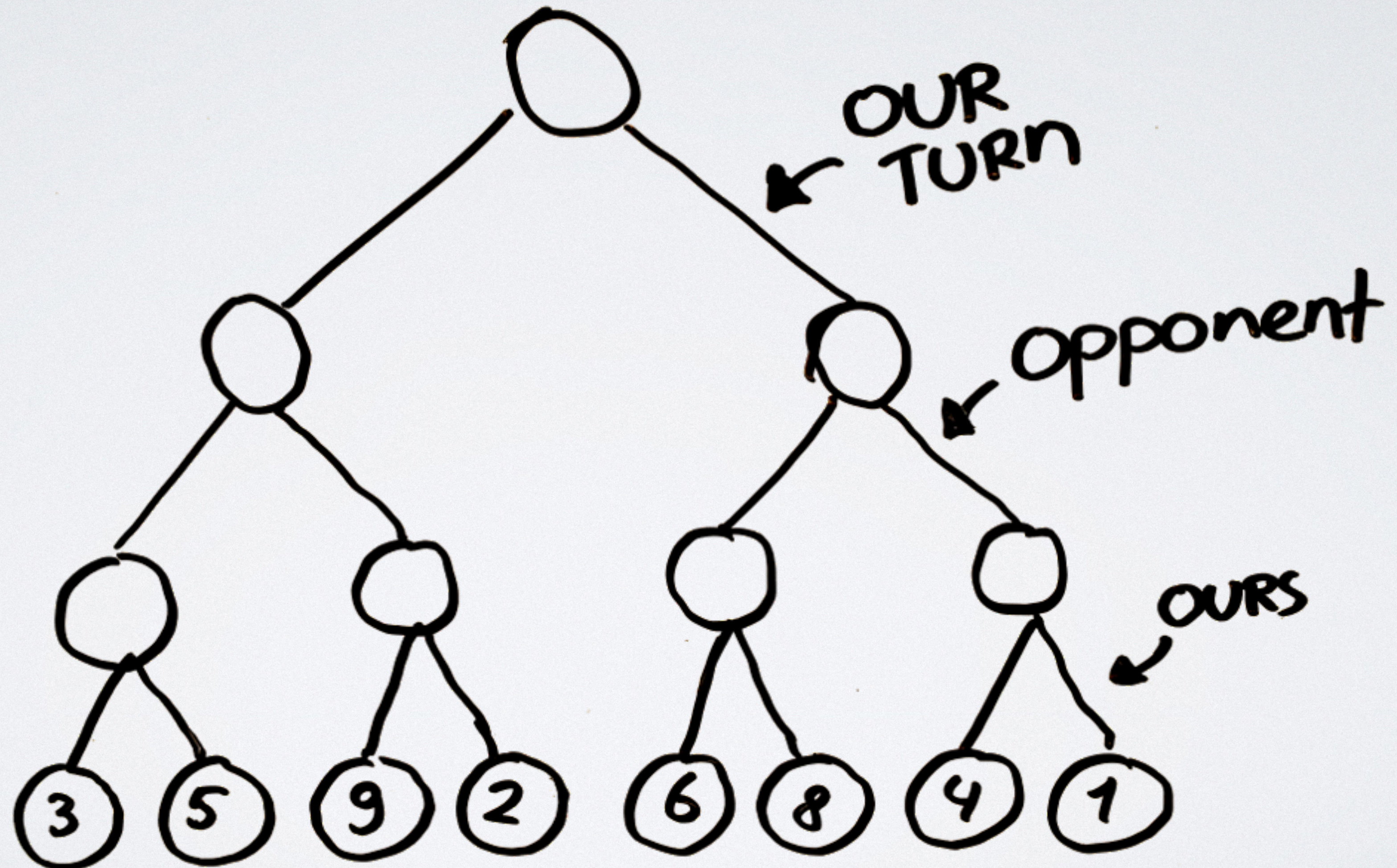
# MINIMAX

- **MINIMISE THE MAXIMUM SCORE** (WHEN IT IS THE OPPONENTS TURN)
- **MAXIMISE THE MINIMUM SCORE** (WHEN IT IS OUR TURN)
- **THIS SIMULATES 'PERFECT PLAY'**



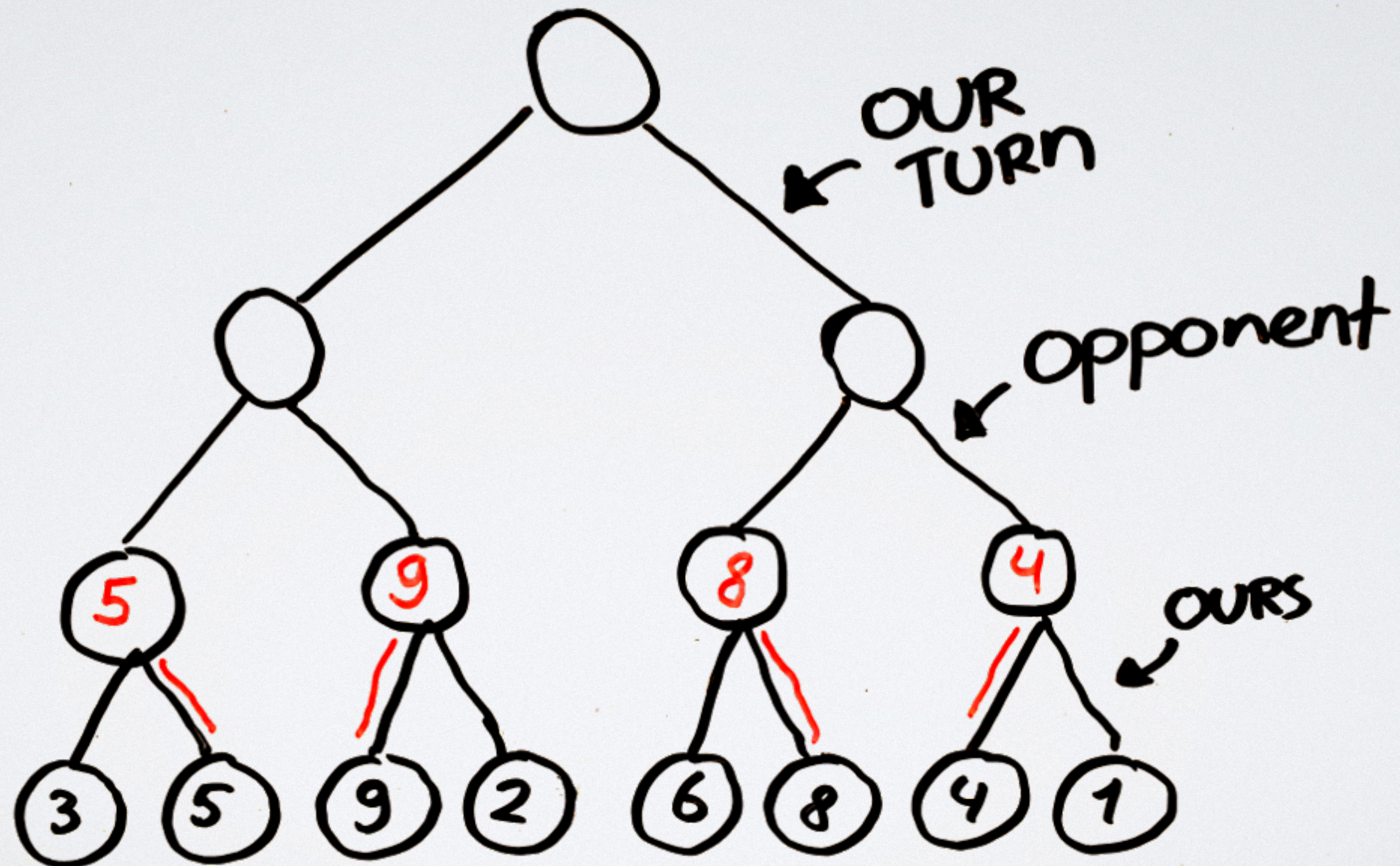


# DEPTH: $N=3$



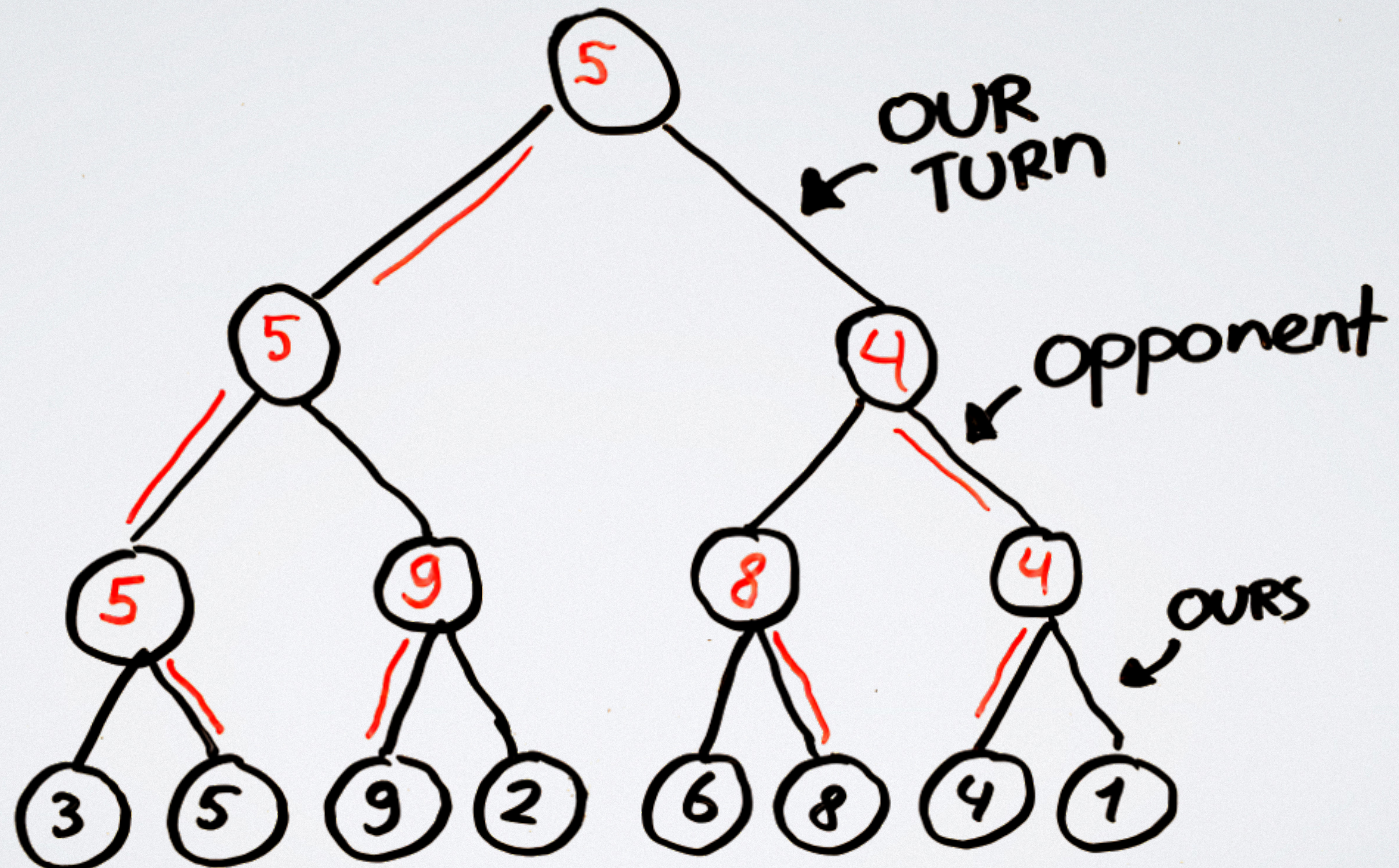


# DEPTH: $N=3$





# DEPTH: N=3





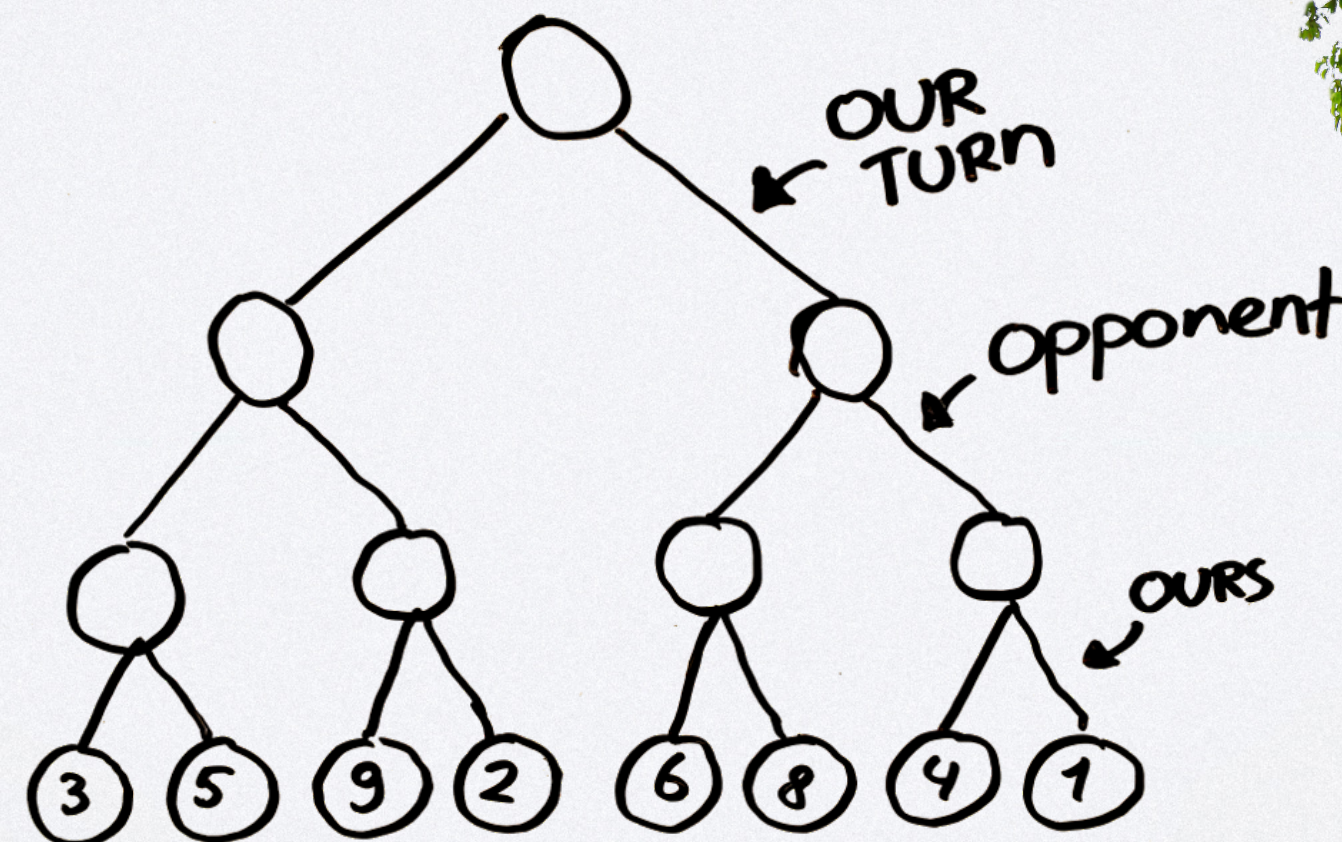
```
int minimax(Node node, boolean maximizingScore) {  
    if(node.isEndNode()) {  
        return node.evaluate();  
    }  
  
    int bestScore = maximizingScore ? Integer.MIN_VALUE : Integer.MAX_VALUE;  
    for(Node child: node.getChildren()) {  
        int score = minimax(child, !maximizingScore);  
        if(maximizingScore) {  
            bestScore = Math.max(score, bestScore);  
        } else {  
            bestScore = Math.min(score, bestScore);  
        }  
    }  
    return bestScore;  
}
```





# SIMPLE TREE GAME

- BRANCHING FACTOR: 2
- GAME DEPTH: 3
- PERFECT INFORMATION





# PLAYING A GAME

using the computer



- A WAY TO **GENERATE** ALL (VALID) **MOVES** (CREATE THE TREE)
- A WAY TO **EVALUATE** NODES
- A WAY TO PICK A **PATH** IN THIS **TREE**



# NOUGHTS AND CROSSES

butter, cheese and eggs?

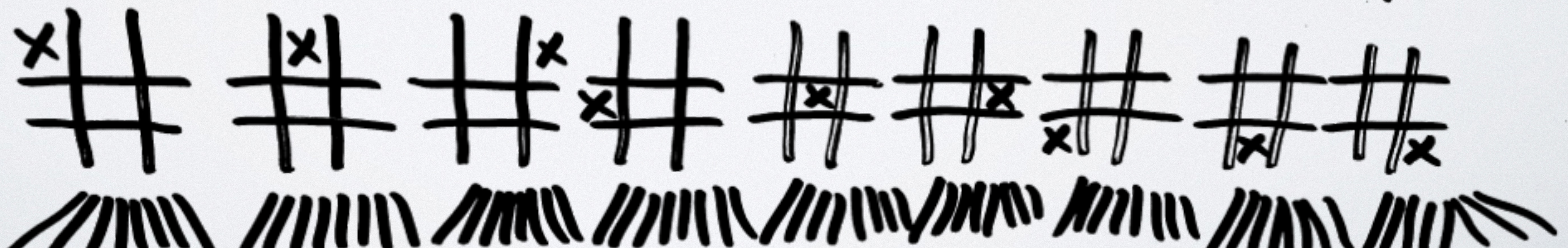
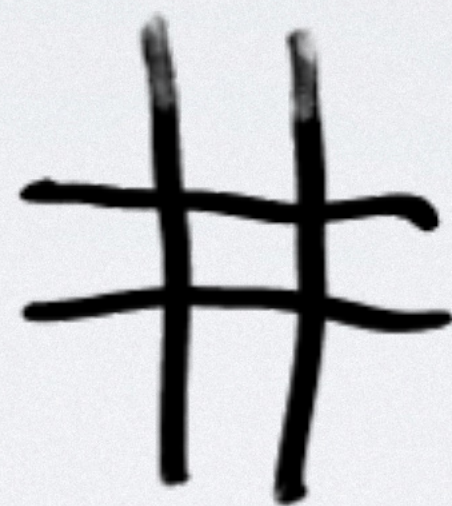


@ROYVANRIJN



# NOUGHTS AND CROSSES

butter, cheese and eggs?



@ROYVANRIJN



# NOUGHTS AND CROSSES

butter, cheese and eggs?



- **BOTTOM NODES HAVE VALUES:**

**WIN: 1**

**TIE: 0**

**LOSE: -1**



# NOUGHTS AND CROSSES

butter, cheese and eggs?



- BRANCHING FACTOR: **5** =  $(9+8+7+6+5+4+3+2+1)/9$
- DEPTH: MAX **9** MOVES
- REMOVING SYMMETRIES THERE ARE **138** TERMINAL POSITIONS
  - × WINS 91 TIMES, ○ WINS 44 TIMES, 3 DRAWS

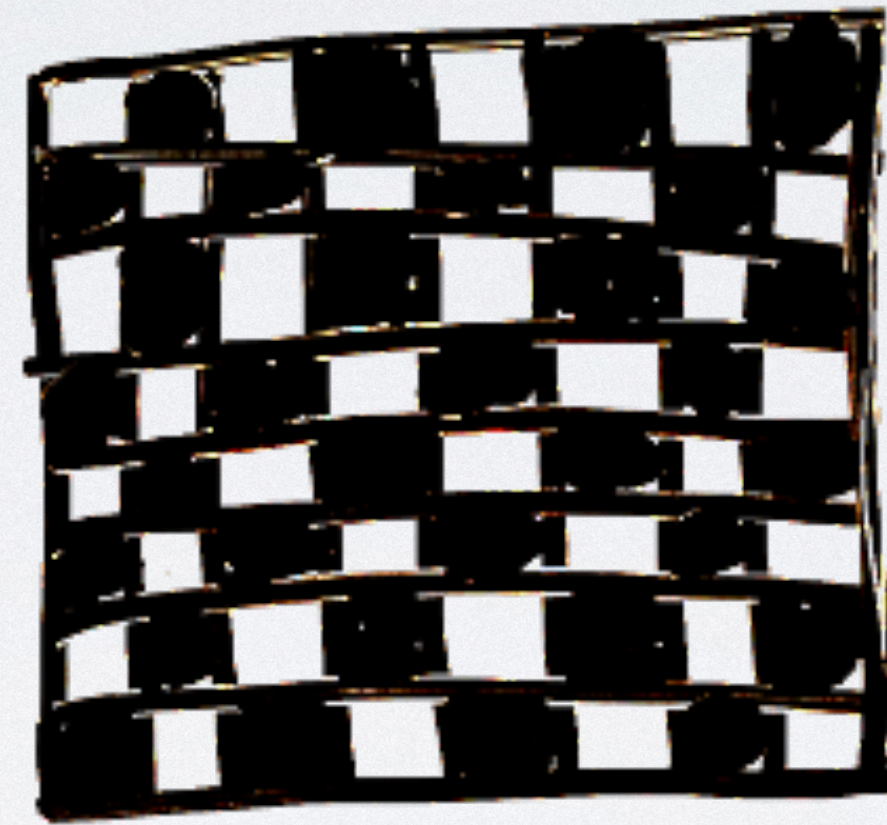




# THE GAME OF CHESS



# CHESS





# 'PERFT'



Depth	Nodes	Captures	E.p.	Castles	Promotions	Checks	Checkmates
0	1	0	0	0	0	0	0
1	20	0	0	0	0	0	0
2	400	0	0	0	0	0	0
3	8,902	34	0	0	0	12	0
4	197,281	1576	0	0	0	469	8
5	4,865,609	82719	258	0	0	27351	347
6	119,060,324	2812008	5248	0	0	809099	10828



# CHESS



- **NO** PERFECT INFORMATION
- HOW DO WE **EVALUATE** A NODE?
  - **COUNT** THE PIECES
  - EVALUATE PIECE **POSITIONS**/LIBERTIES



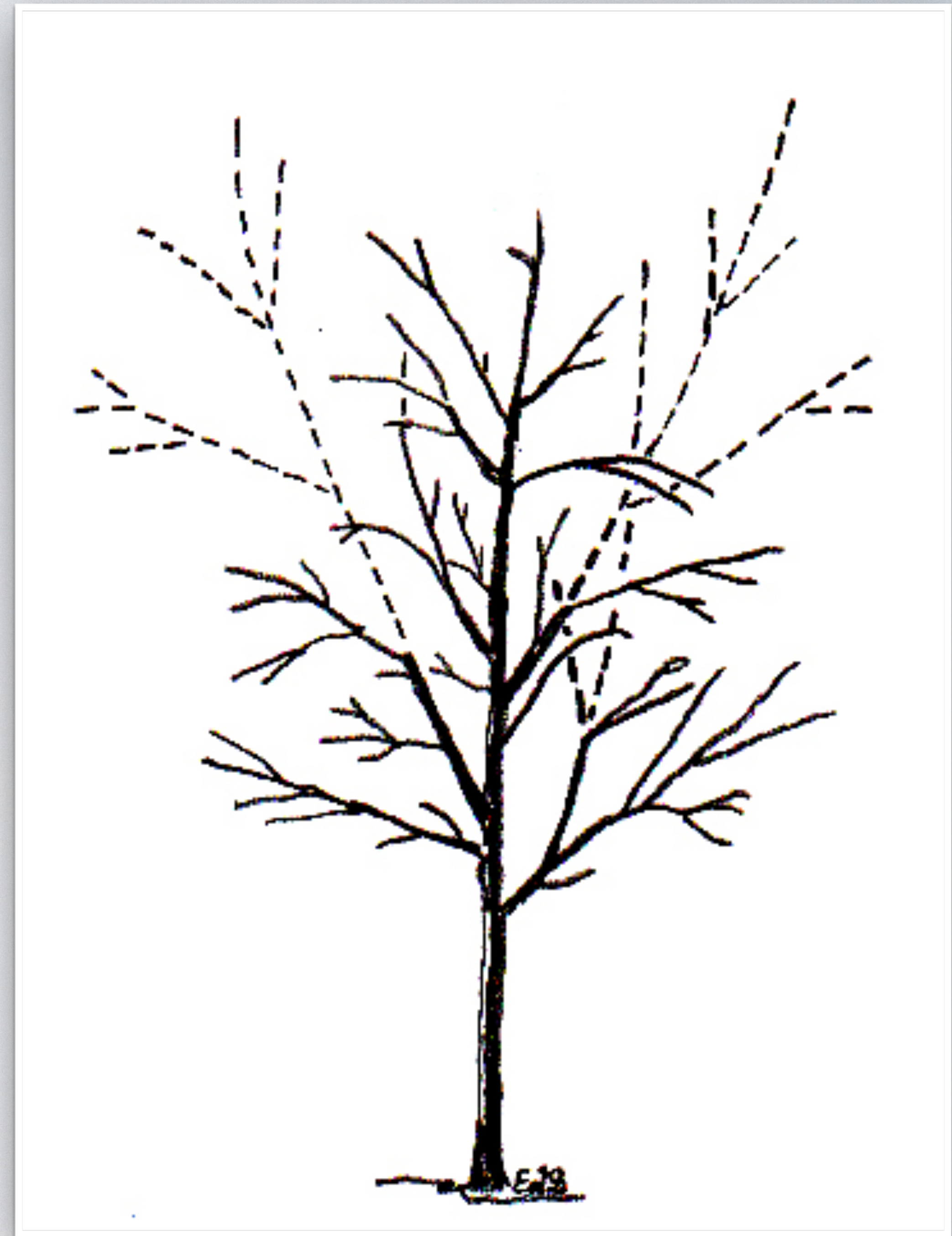
# HORIZON PROBLEM





# PRUNING

- WE NEED TO **CUT** BACK THE TREE
- **FORWARD** PRUNING: RISKY
- **BACKWARD** PRUNING: SAFE





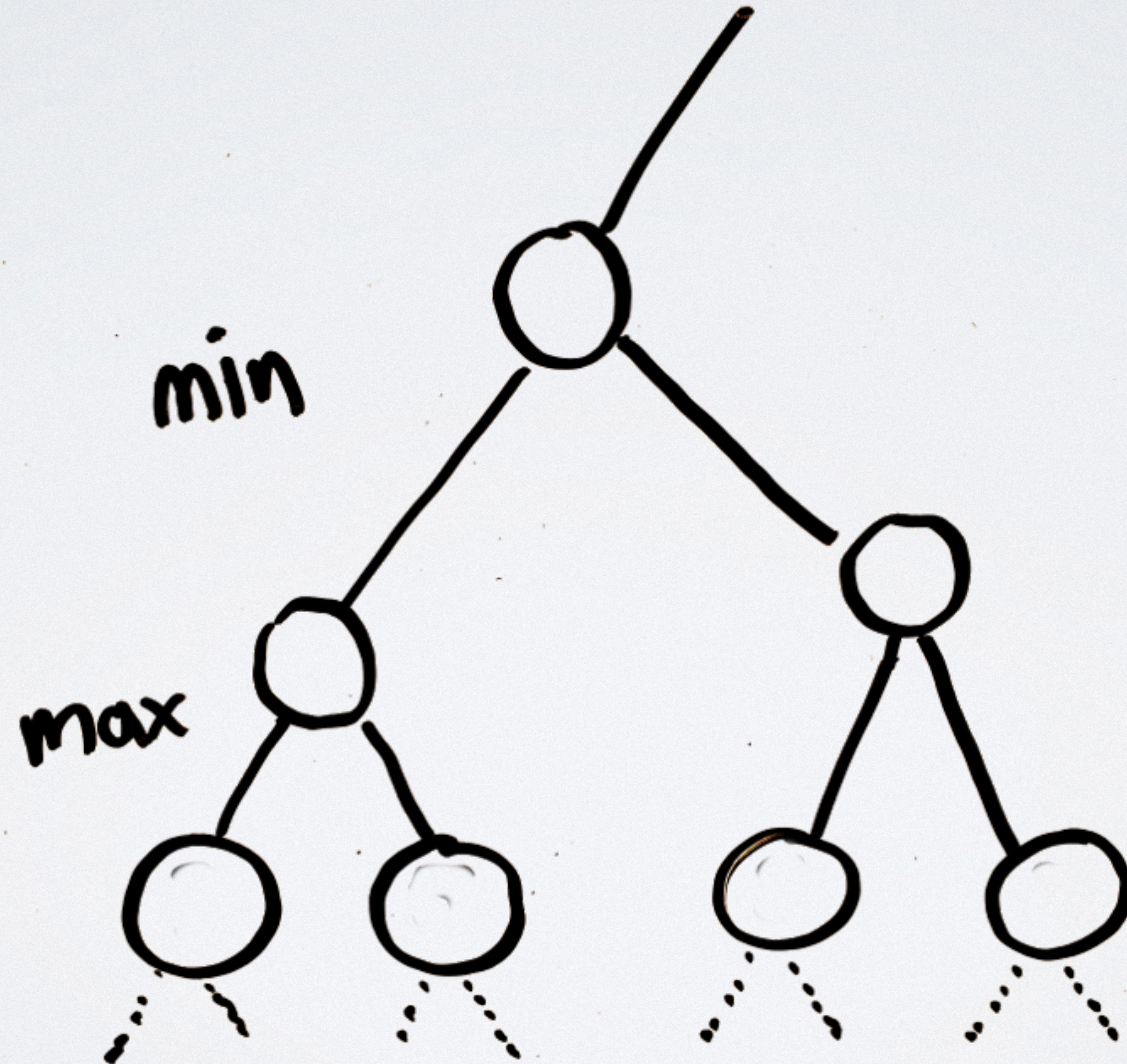
# FORWARD PRUNING

- IF A **MOVE** IS TOO **BAD**: STOP EVALUATING
- IF A **MOVE** IS TOO **GOOD**: STOP EVALUATING



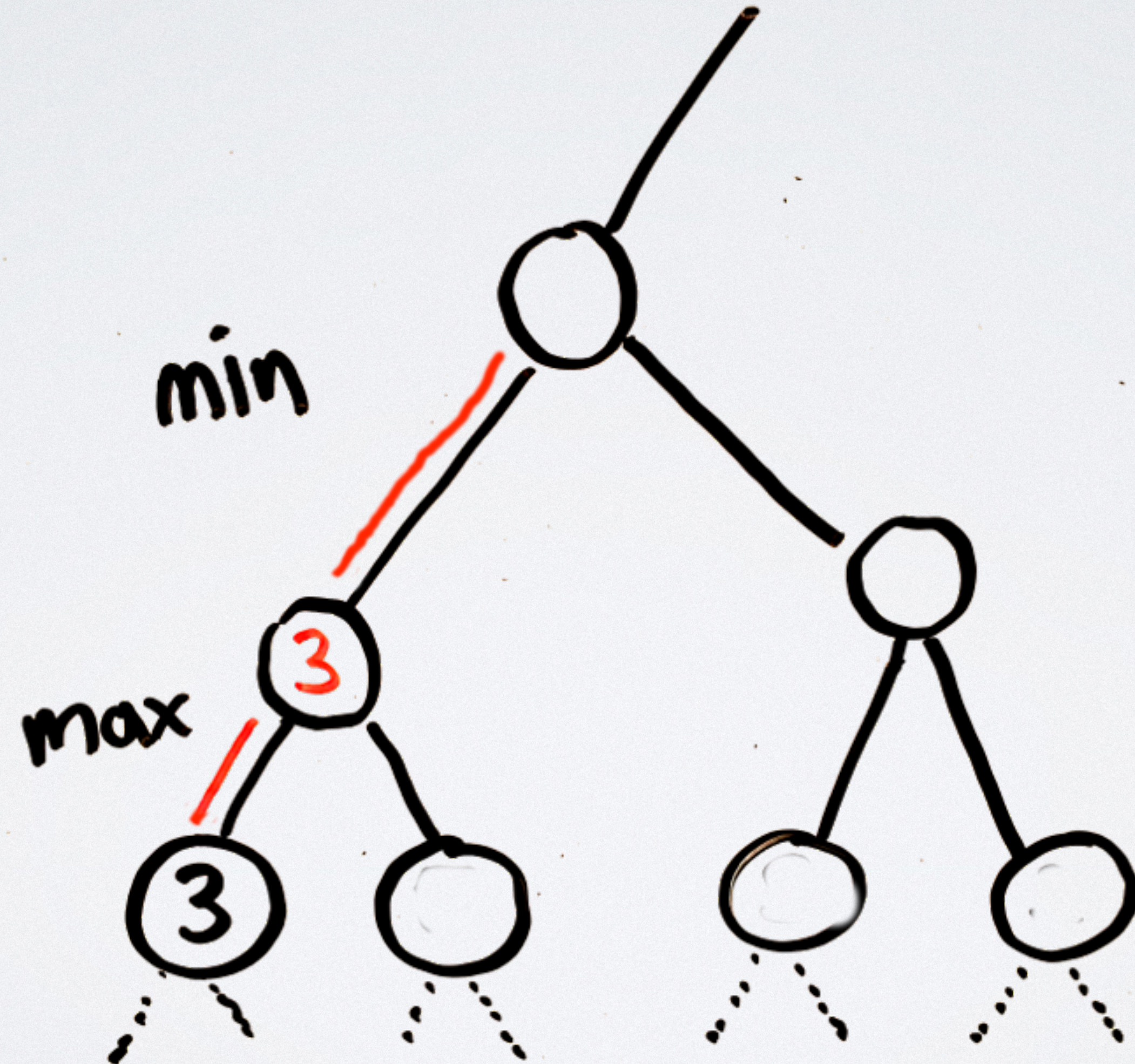


# ALPHA/BETA PRUNING



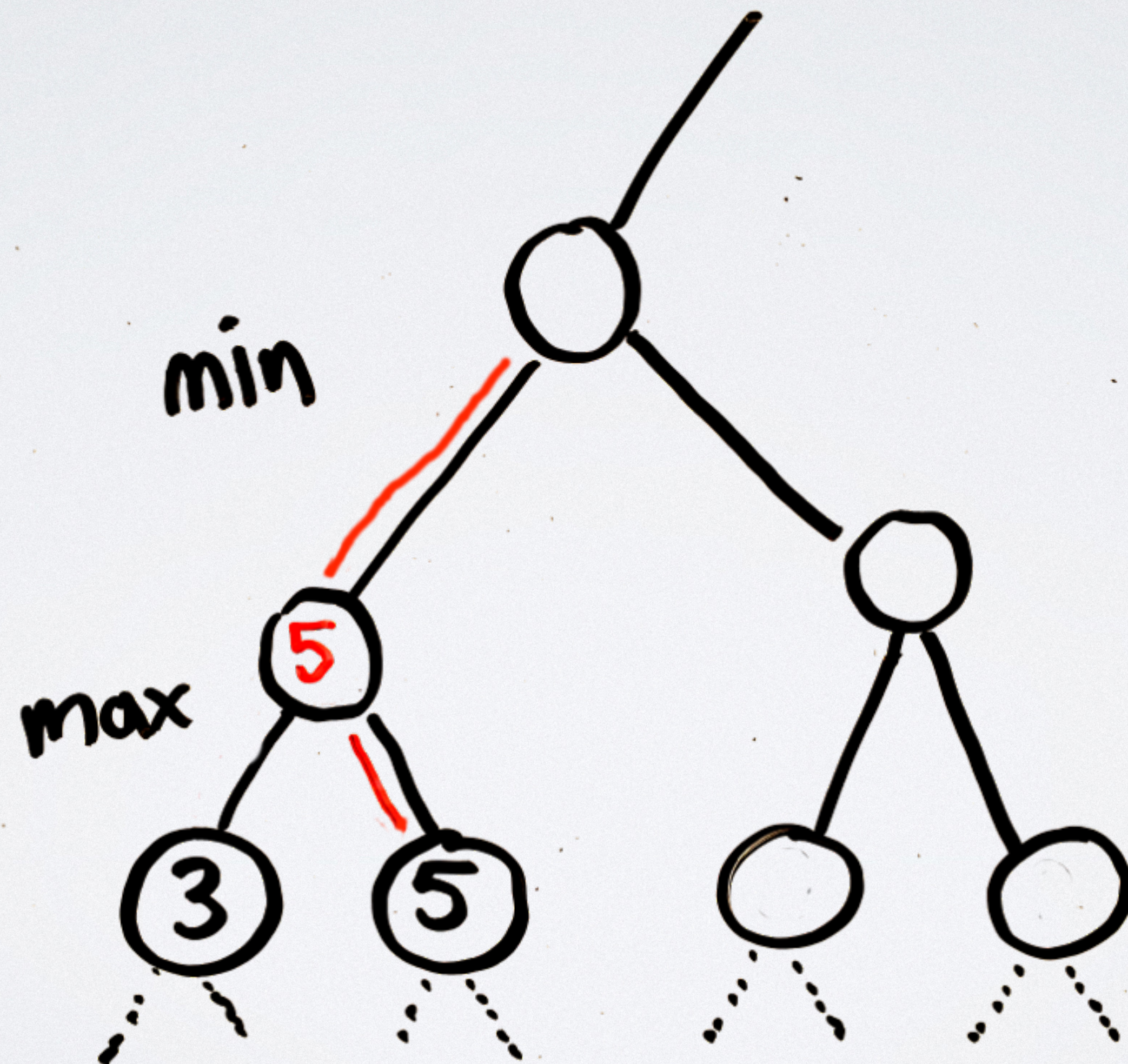


# ALPHA/BETA PRUNING



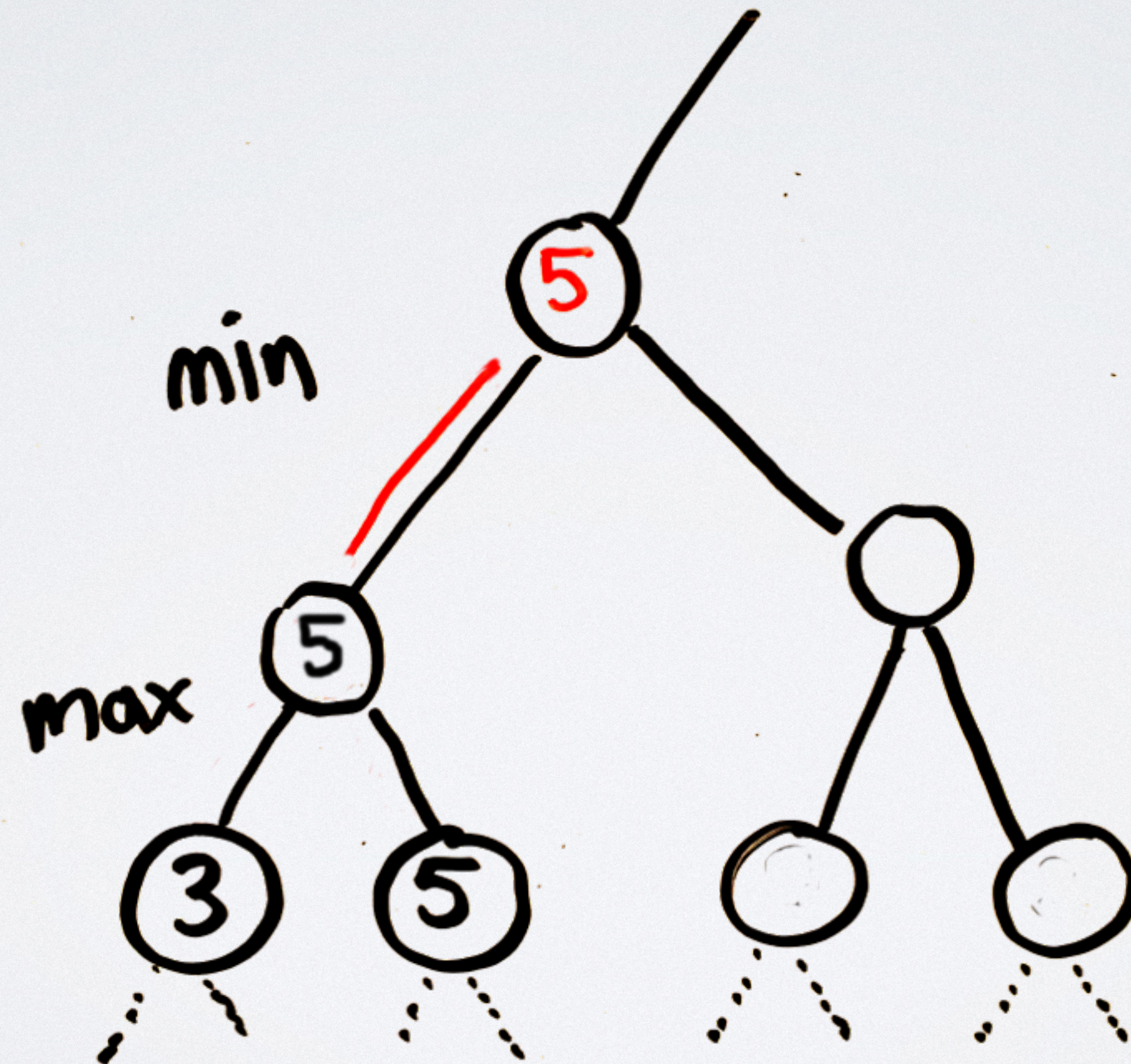


# ALPHA/BETA PRUNING



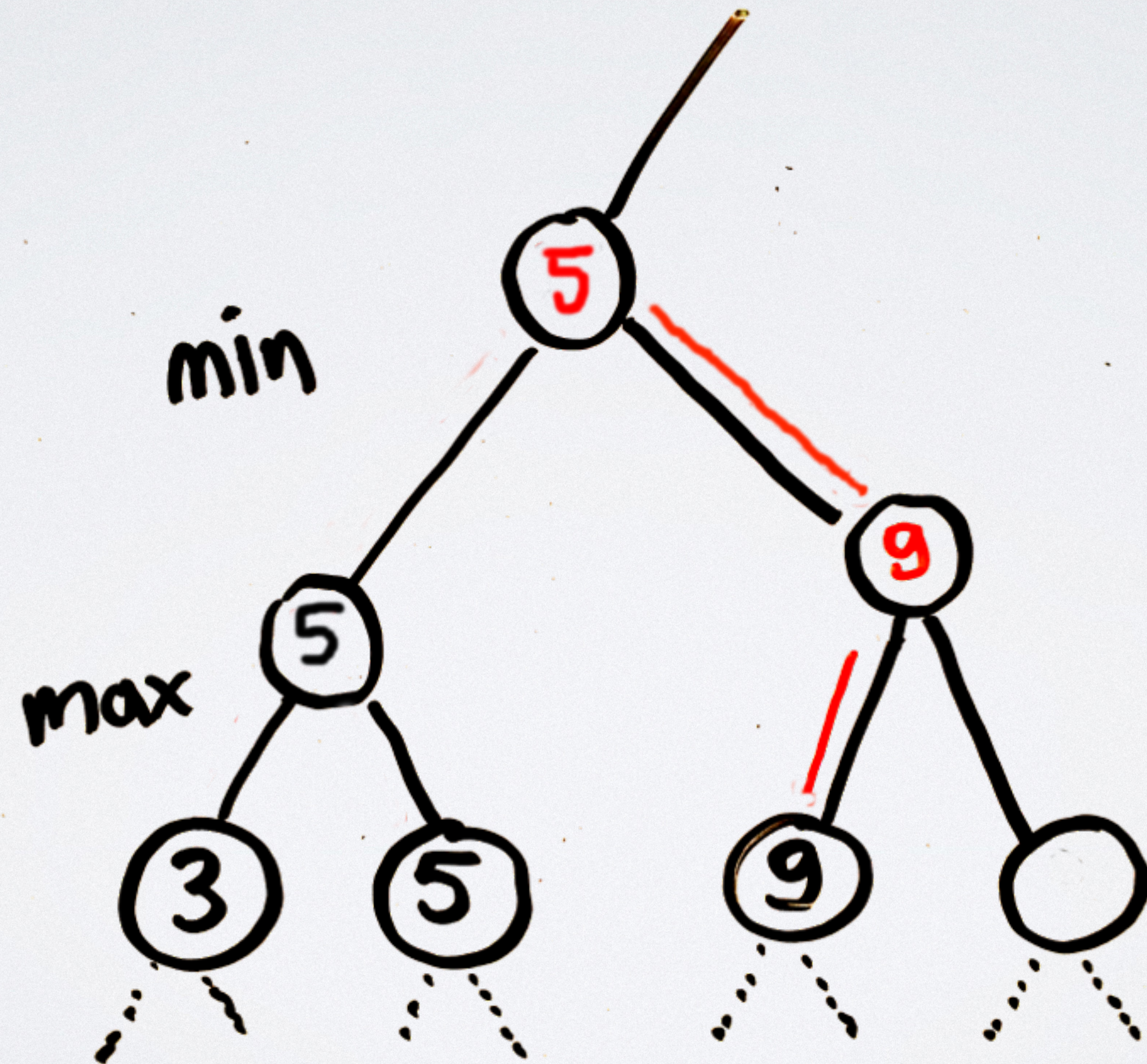


# ALPHA/BETA PRUNING



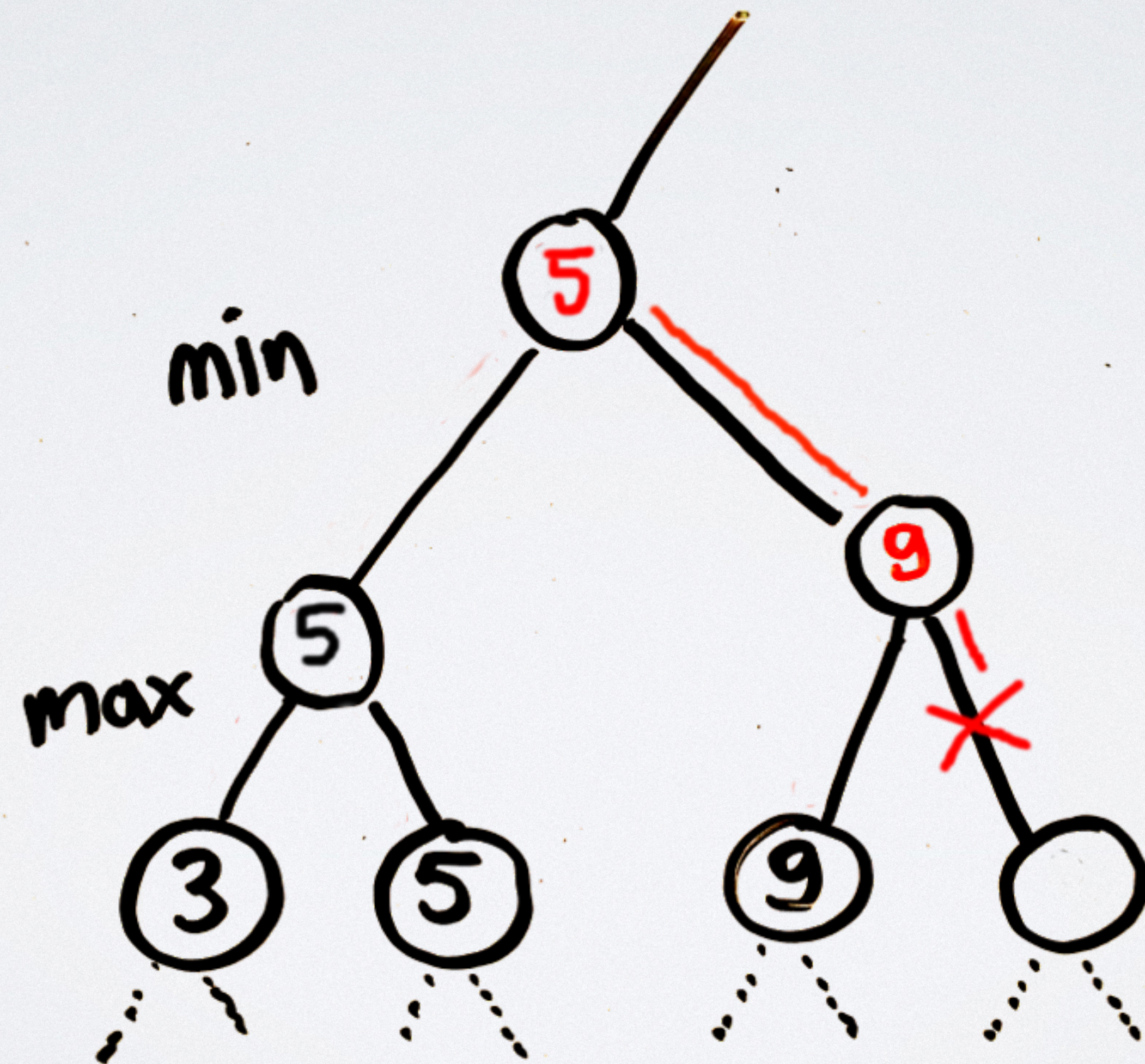


# ALPHA/BETA PRUNING



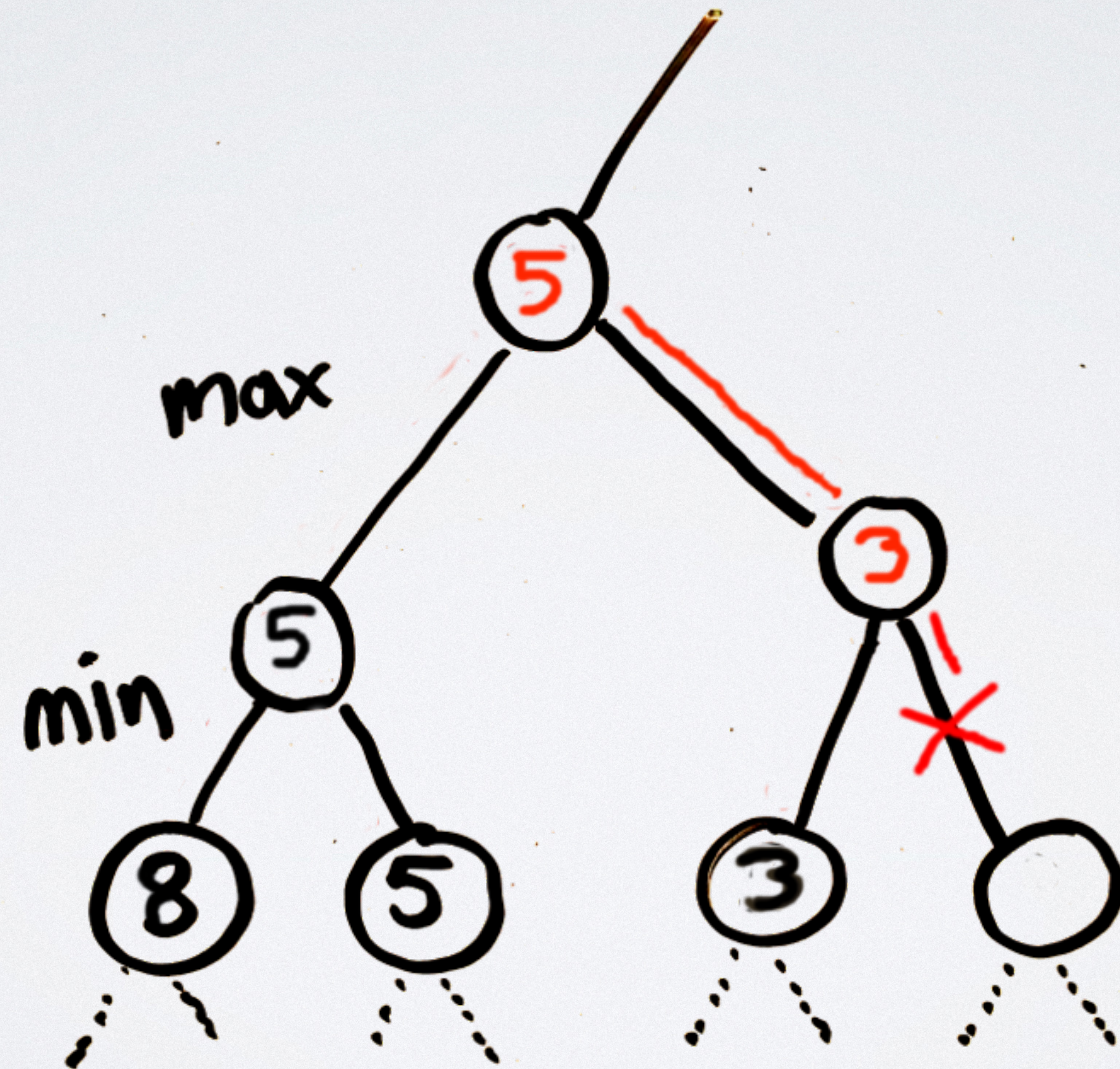


# ALPHA/BETA PRUNING





# ALPHA/BETA PRUNING





```
int minimax(Node node, boolean maximizingScore) {  
    if(node.isEndNode()) {  
        return node.evaluate();  
    }  
  
    int bestScore = maximizingScore ? Integer.MIN_VALUE : Integer.MAX_VALUE;  
    for(Node child: node.getChildren()) {  
        int score = minimax(child, !maximizingScore);  
        if(maximizingScore) {  
            bestScore = Math.max(score, bestScore);  
        } else {  
            bestScore = Math.min(score, bestScore);  
        }  
    }  
    return bestScore;  
}
```





```
int alphaBeta(Node node, [REDACTED] boolean maximizingScore) {  
    if(node.isEndNode()) {  
        return node.evaluate();  
    }  
  
    int bestScore = maximizingScore ? Integer.MIN_VALUE : Integer.MAX_VALUE;  
    for(Node child: node.getChildren()) {  
        int score = alphaBeta(child, [REDACTED] !maximizingScore);  
        if(maximizingScore) {  
            bestScore = Math.max(bestScore, score);  
            [REDACTED]  
        } else {  
            bestScore = Math.min(bestScore, score);  
            [REDACTED]  
        }  
        [REDACTED]  
    }  
    return bestScore;  
}
```



# PLAYING CHESS

using the computer



- A WAY TO **GENERATE** ALL VALID **MOVES** -> CHESS ENGINE
- A WAY TO **EVALUATE** NODES -> COUNT PIECES
- A WAY TO PICK A **PATH** IN THIS **TREE** -> ALPHA/BETA SEARCH



# CHESS



- AVERAGE BRANCHING FACTOR: 35
- AVERAGE GAME DEPTH: 40-50 MOVES
- EVALUATION FUNCTION: RELATIVELY EASY
- ADVANCED CHESS A.I. CAN LOOK 20+ MOVES AHEAD



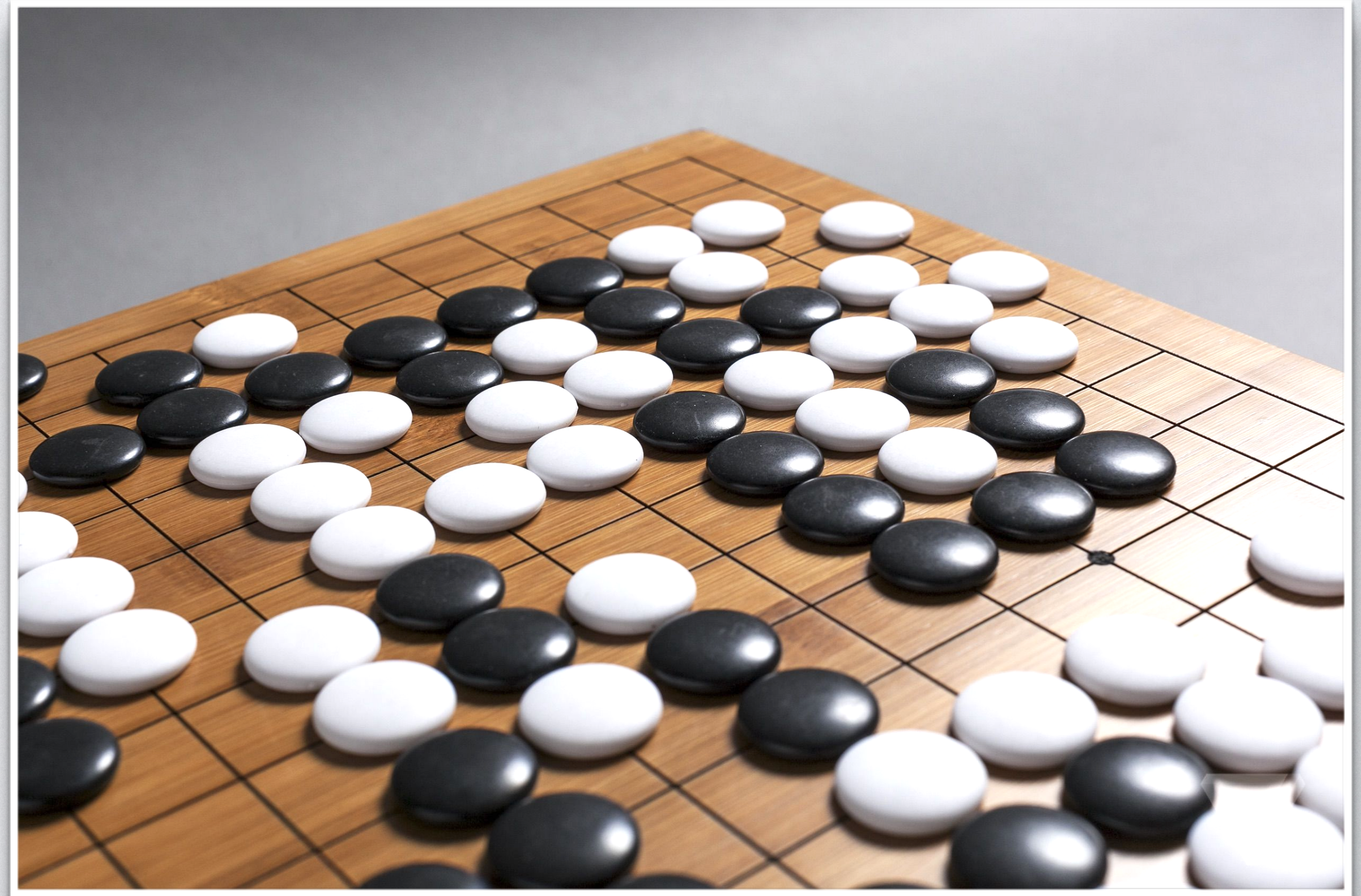


# THE GAME OF GO



# ABOUT THE GAME

- BOARD: **19X19**
- BLACK AND WHITE STONES
- SURROUND AND CAPTURE AREAS





# COMPLEXITY OF GO



- FIRST PROBLEM: BRANCHING FACTOR: +/- 250
- SECOND PROBLEM: GAME DEPTH: 300+ MOVES
- THIRD PROBLEM: EVALUATION FUNCTION: .....



# COMPLEXITY OF GO



**$1.74 \times 10^{172}$**

( LARGER THAN THE AMOUNT OF ATOMS IN THE ENTIRE UNIVERSE )



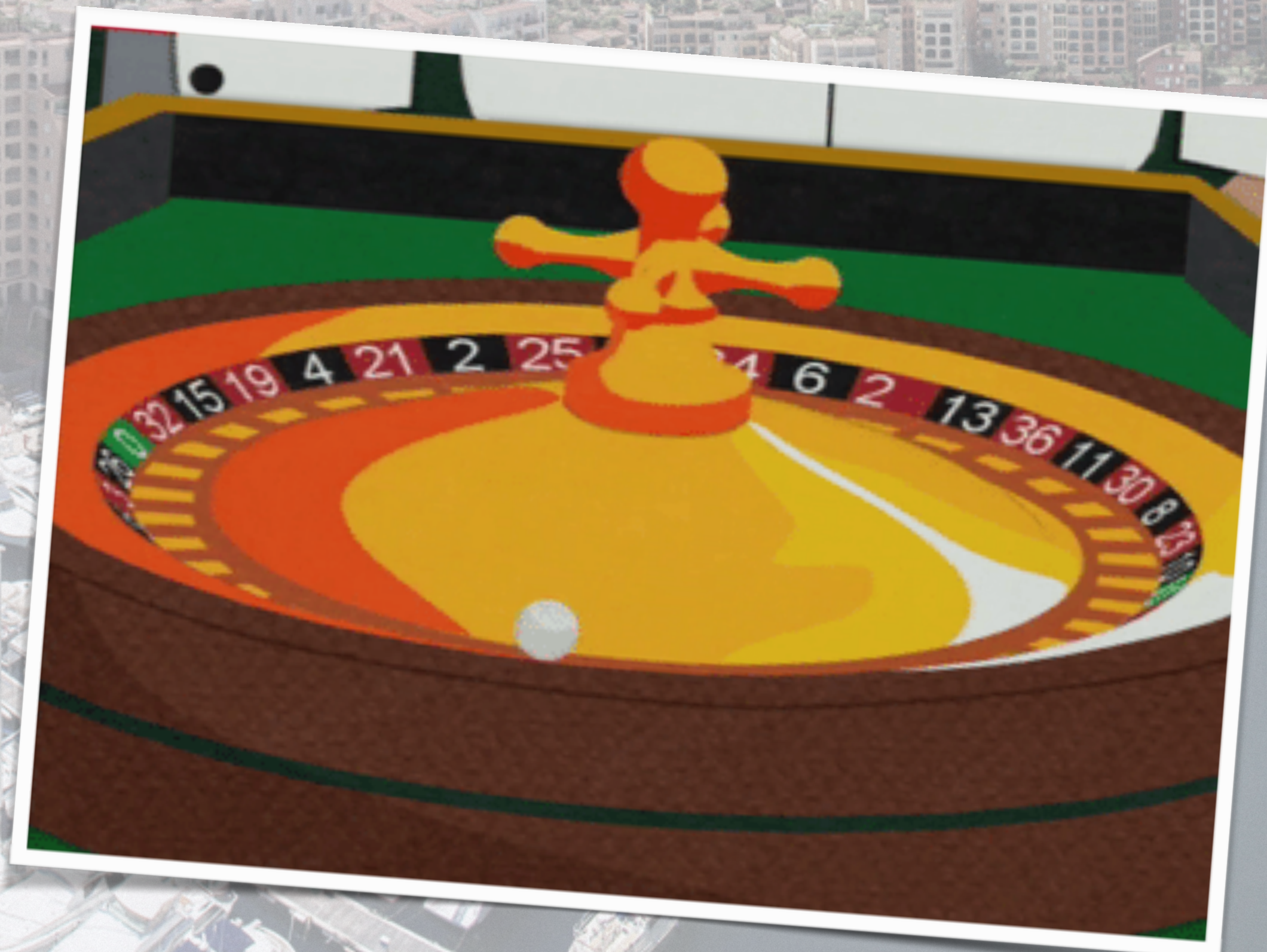
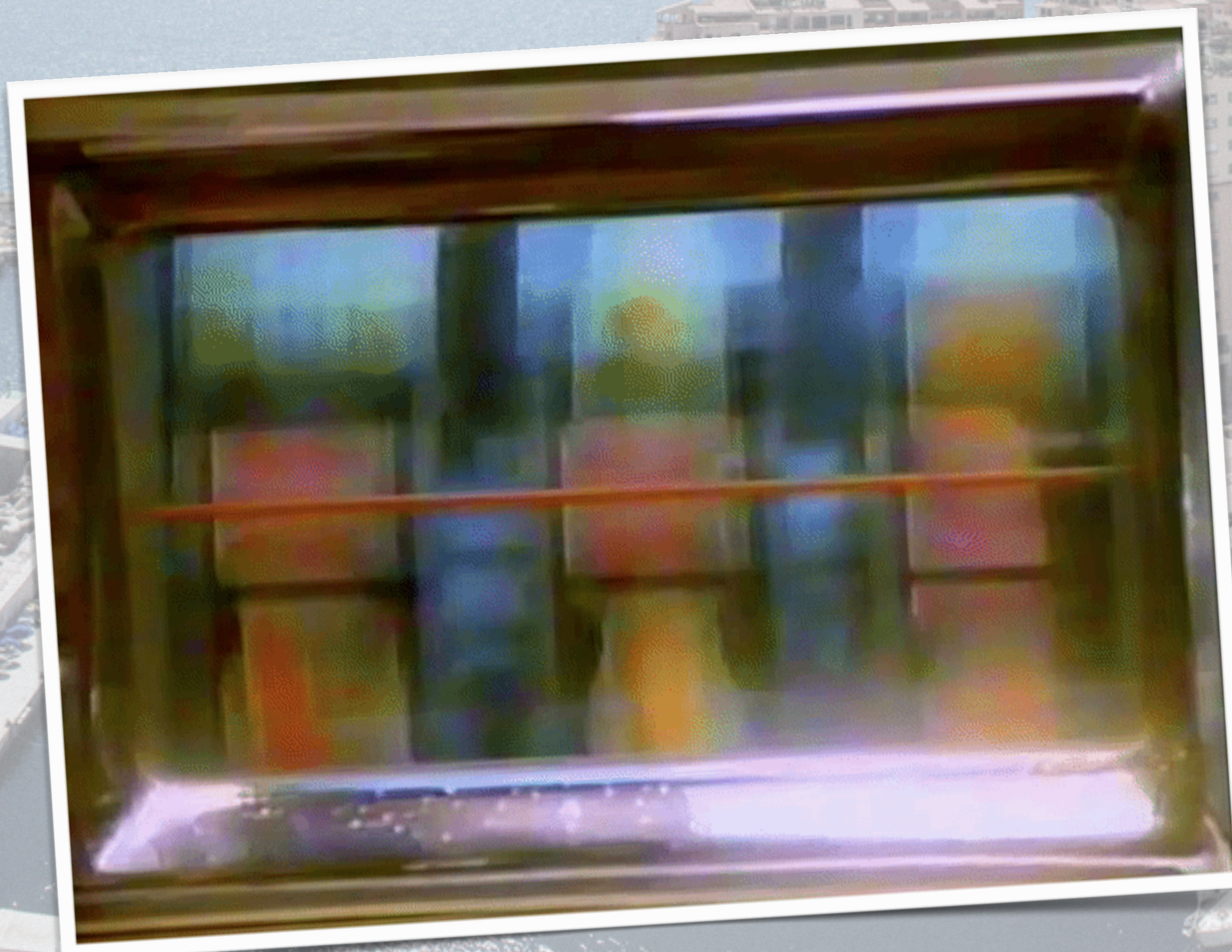
# MONTE CARLO TREE SEARCH



@ROYVANRIJN



# MONTE CARLO TREE SEARCH



@ROYVANRIJN



# MONTE CARLO TREE SEARCH

- PICK A **NODE**
- PLAY (SEMI-) **RANDOM** MOVES TO THE END  
(AS OFTEN AS POSSIBLE)
- THIS GIVES A STRONG **INDICATION** OF THE STRENGTH







**EXPERTS IN 2015:**

**"IT WILL PROBABLY TAKE 10 TO 15 YEARS BEFORE A  
COMPUTER CAN BEAT A PROFESSIONAL GO PLAYER"**









AlphaGo



@ROYVANRIJN



# NEURAL NETWORK

A NEURAL NETWORK IS A **COMPUTER MODEL** DESIGNED TO SIMULATE  
THE BEHAVIOUR OF **BIOLOGICAL NEURAL NETWORKS**





**DEMO TIME**



@ROYVANRIJN



# MORE INFORMATION

- **TENSORFLOW** ([HTTPS://WWW.TENSORFLOW.ORG](https://www.tensorflow.org))
- **DEEPLEARNING4J** ([HTTPS://DEEPLEARNING4J.ORG/](https://deeplearning4j.org/))
- **CAFFE, TORCH, THEANO, ETC**







AlphaGo





# NEURAL NETWORKS IN ALPHAGO

- CONVOLUTIONAL NEURAL NETWORKS
- LEARNING IS SUPERVISED
- HAS HIDDEN **13**-LAYERS





# #1 SUPERVISED LEARNING POLICY NETWORK

- **30** MILLION AMATEUR MATCHES
- GOAL: PREDICT THE **NEXT** MOVE
- RESULT: **57%** CORRECT





# #2 REINFORCED LEARNING POLICY NETWORK

- COPY OF **SUPERVISED** NETWORK
- NEW GOAL: PREDICT THE **\*BEST\*** MOVE
- NETWORK PLAYED ITSELF **1.2** MILLION TIMES (TOOK ONE DAY)
- PLAYS **PACHI** AND WINS: **85%** OF THE TIME (WITHOUT SEARCH!)





# #3 FAST ROLLOUT POLICY NETWORK

- THE **REINFORCED** NETWORK IS SLOW: **3<sub>MS</sub>**
- TOO SLOW FOR **MONTE CARLO** SEARCH
- THIS IS SMALLER, BUT FASTER: **2<sub>μs</sub>** **1500<sub>x</sub>**





# #4 VALUE NETWORK



- TRAINED USING THE SAME **30** MILLION GAMES
- PREDICTS THE **WINNER** BASED ON CURRENT BOARD
- INITIALLY HAD ERROR OF **0.37** (0.5 IS RANDOM)
- AFTER **SELF-PLAY** ERROR CAME DOWN TO **~0.23**



# #4 VALUE NETWORK



- TESTING THE **VALUE NETWORK**
- FOR A GIVEN BOARD, GENERATE **ALL** MOVES
- FOR **ALL** MOVES, EVALUATE AND PICK THE **BEST** NEXT MOVE
- BEATS THE **STRONGEST** KNOWN A.I. STILL WITHOUT TREE-SEARCH (!!!)



# COMBINING ALL THE PIECES

- USE **POLICY** NETWORK TO LOOK AT THE CURRENT **BEST** MOVES
- FOR THOSE MOVES, USE THE **VALUE** NETWORK TO DOUBLE CHECK
- USE **FAST ROLLOUT** NETWORK FOR **MONTE CARLO** TREE SEARCH



# THE CHALLENGE

- LEE SEDOL: THE **BEST** GO PLAYER OF THIS DECADE
- BEST OF **5** GAMES WINS
- WINNER GETS \$**1,000,000.-**





# THE CHALLENGER

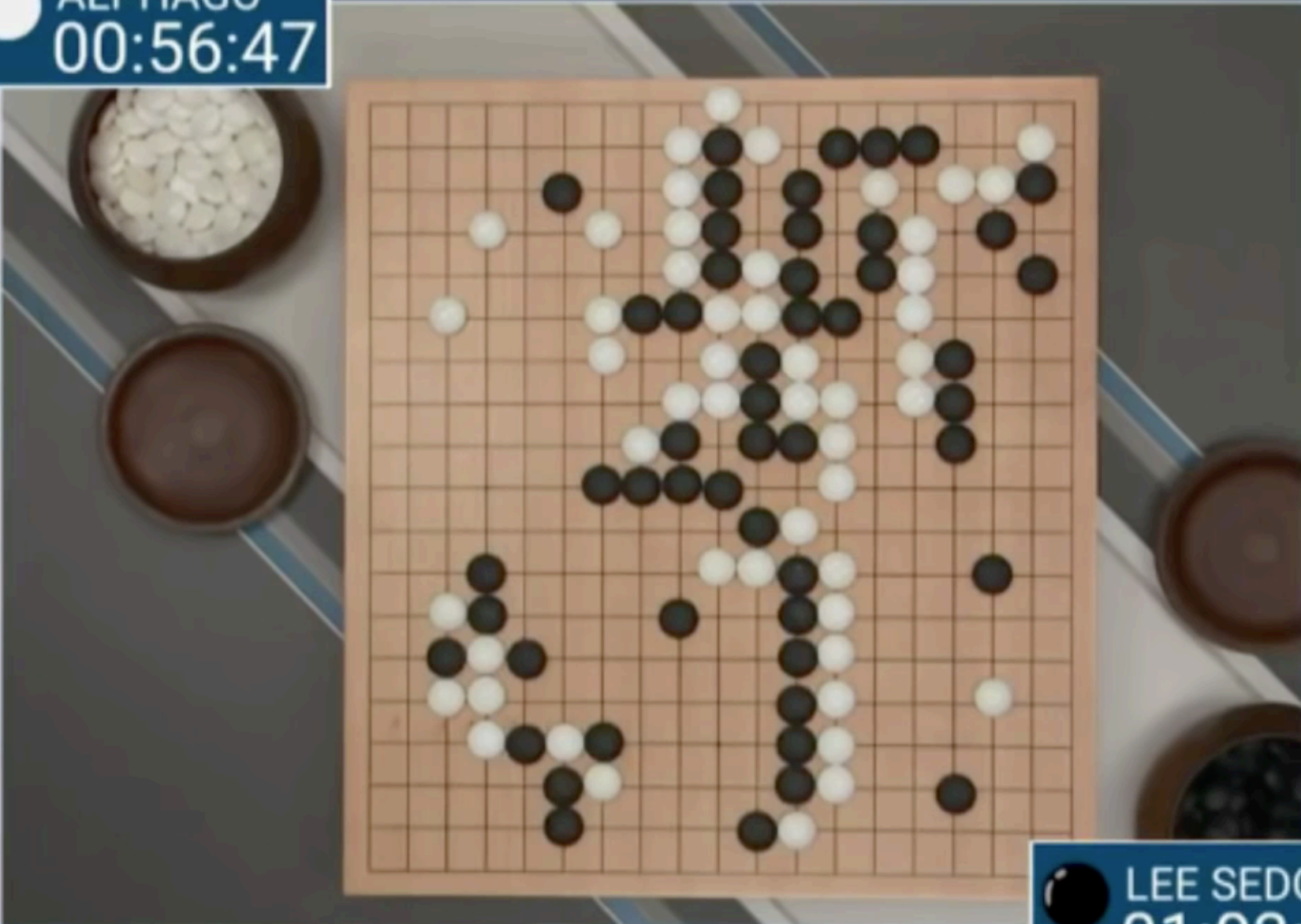
- **DISTRIBUTED** ALPHAGO:
- 1202 CPU's
- 176 GPU's






# GAME 1, MOVE 102


ALPHAGO  
00:56:47



LEE SEDOL  
01:03:43



Google DeepMind  
Challenge Match



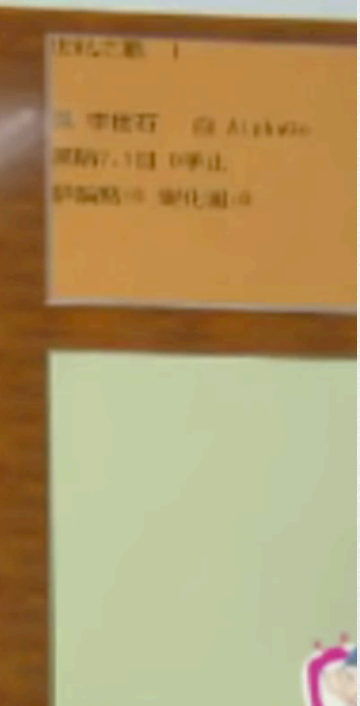
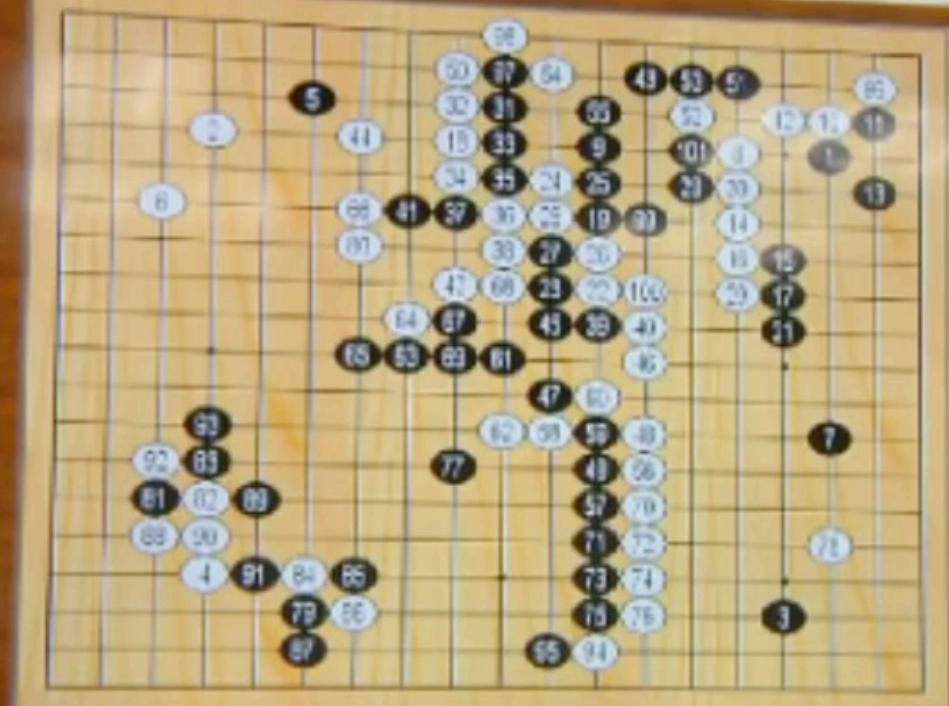
人機  
世紀  
之戰

3/9-3/15

解說員介紹：

林聖賢老師－職業八段 圍棋技藝指導老師

高川格－圍棋評論家 圍棋頻道經營者

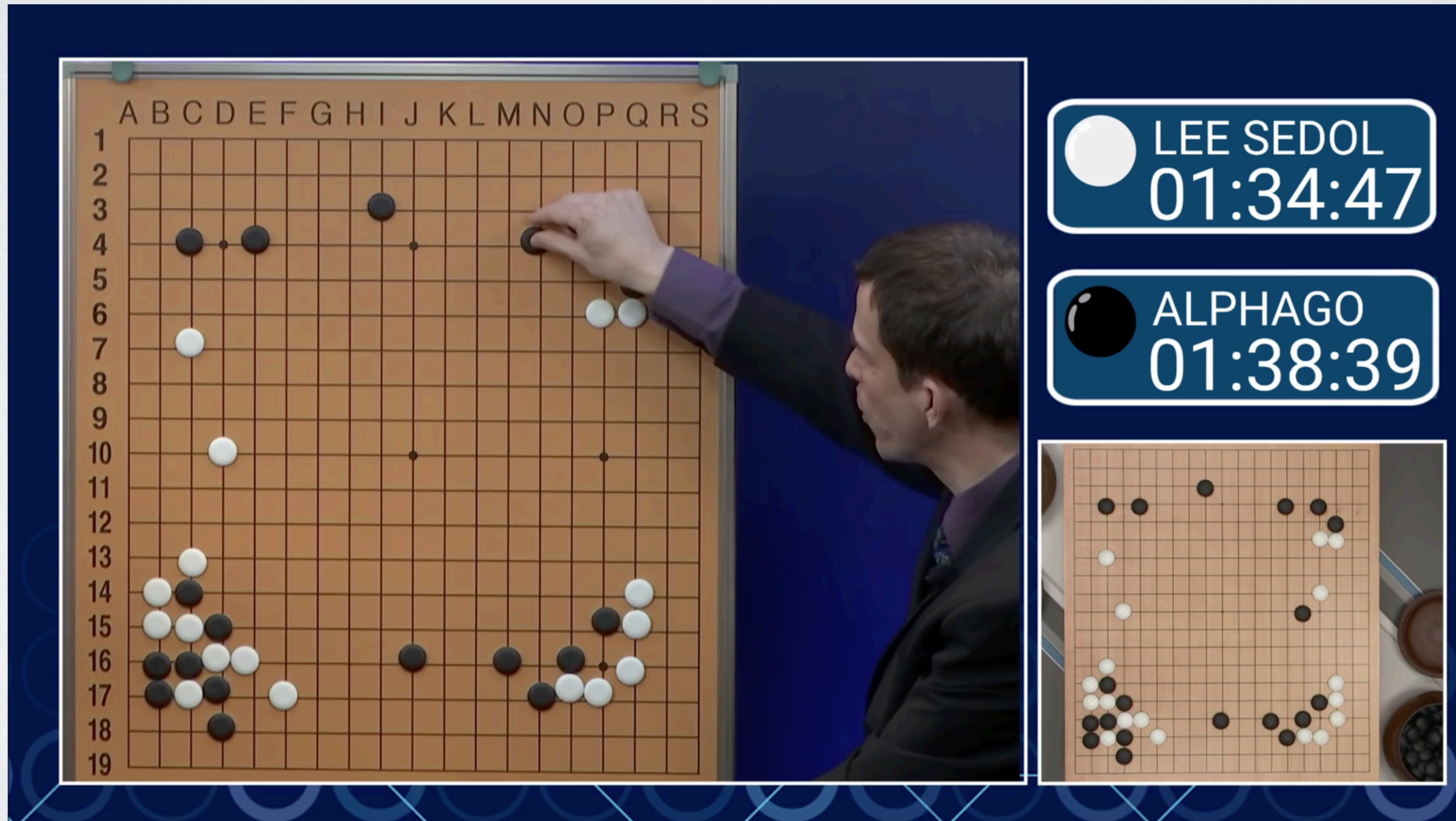




@ROYVANRIJN



# GAME 2, MOVE 37





EUROPEAN CHAMPION **FAN HUI**:

**“IT’S NOT A HUMAN MOVE.  
I’VE NEVER SEEN A HUMAN PLAY THIS MOVE,  
SO **BEAUTIFUL**.”**



# GAME 4, MOVE 78

**GU LI** (LEE'S ARCHRIVAL):

**“THIS MOVE WAS MADE WITH **THE HAND OF GOD.**”**





# RESULTS

ALPHAGO 4 - LEE SEDOL 1









**NOBODY TAUGHT ALPHAGO WHAT A GOOD OR BAD MOVE IS**

**NOBODY PROGRAMMED AN EVALUATION FUNCTION FOR ALPHAGO**

**ALPHAGO ISN'T AN EXPERT SYSTEM**



ALPHAGO **LEARNED** BY WATCHING OTHERS AND SELF-PLAY

USING **GENERAL** MACHINE LEARNING TECHNIQUES  
TO FIGURE OUT FOR **ITSELF** HOW TO WIN AT GO...





The background of the image is a dense, overlapping collage of various US dollar bills. Visible denominations include \$100, \$50, and \$20. The bills are slightly faded and layered, creating a textured, financial-themed backdrop. The text is centered over this background.

**AS A RESPONSE TO THE SUCCESS OF ALPHAGO,  
SOUTH KOREA ANNOUNCED ON 17 MARCH 2016 THAT IT  
WOULD INVEST **\$863 MILLION** IN ARTIFICIAL-  
INTELLIGENCE RESEARCH OVER THE NEXT FIVE YEARS.**



@ROYVANRIJN







- ALPHAGO **ZERO** VERSUS ALPHAGO: **100** – 0
- SUPERHUMAN ABILITIES FOR: **CHESS, SHOGI**
- PROTEIN FOLDING: ALPHA**FOLD**
- STARCRAFT:       ALPHA**STAR**
- ULTIMATE GOAL: DEEPMIND **HEALTH...**





# QUESTIONS?

DON'T FORGET TO VOTE



Jpoint

FOLLOW ME ON TWITTER: [@ROYVANRIJN](#)

WEBSITE: [HTTP://WWW.ROYVANRIJN.COM](http://www.royvanrijn.com)