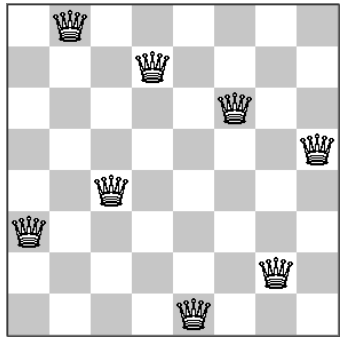
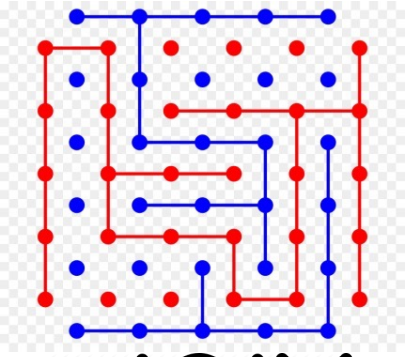
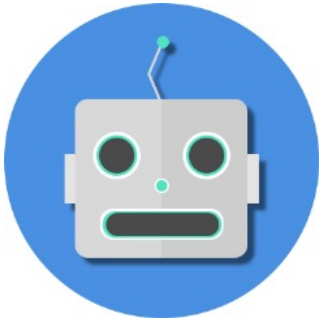
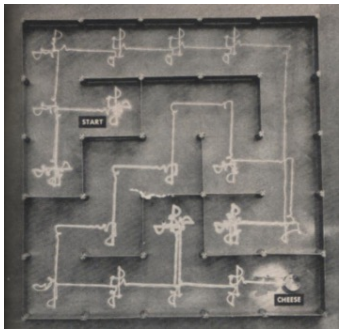
Artificial Intelligence:

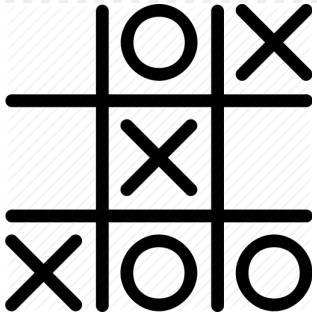
Past, Present and Future



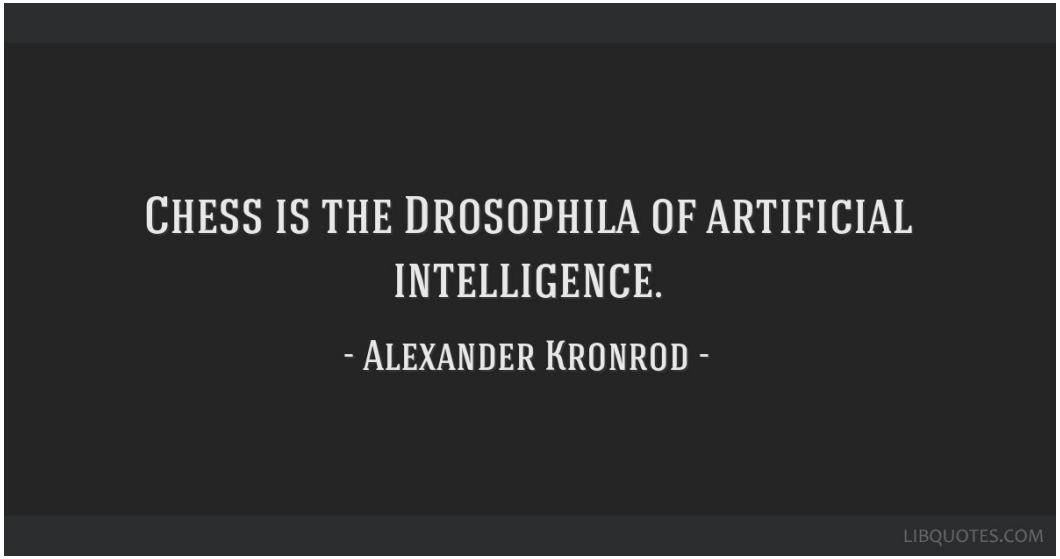








Chee Wei Tan



Aleksandr S. Kronrod

- This lecture looks at two-player AI games of strategy that *stand the test of time*
- These AI games of strategy are positional games or chess-like
- A key of AI games is *to automatically search for the right path (sequence of moves)*
- This lecture informs you that AI game algorithms are driven by questions such as: How many paths? Which is *right*? *A*nd how do you even define *right*?
- To *automate the process of answering these questions* requires clever algorithms and understanding the art of *trial-and-error*

http://jmc.stanford.edu/articles/drosophila/drosophila.pdf

https://en.wikipedia.org/wiki/Alexander_Kronrod

tive feedback system for hunting for particular solutions of the logical equations without an exhaustive search through all possible combinations. This is achieved by elements which sense whether or not a particular logical relation is satisfied. If not, the truth variables involved in this relation are caused to oscillate between their two possible values. Thus, variables appearing in unsatisfied relations are continually changing, while those appearing only in satisfied relations do not change. If ever all relations are simultaneously satisfied the machine stops at that particular solution. Changing only the variables in unsatisfied relations tends, in a general way, to lead to a solution more rapidly than methodical exhaustion of all cases, but, as is usually the case when feedback is introduced, leads to the possibility of continual oscillation. McCallum and Smith point out the desirability of making the changes of the variables due to the feedback unbalance as random as possible, to enable the machine to escape from periodic paths through various states of the relays.

Game Playing Machines

The problem of designing game-playing machines is fascinating and has received a good deal of attention. The rules of a game provide a sharply limited environment in which a machine may operate, with a clearly defined goal for its activities. The discrete nature of most games matches well the digital computing techniques available without the cumbersome analog-digital conversion necessary in translating our physical environment in the case of manipulating and sensing machines.

Game playing machines may be roughly classified into types in order of increasing sophistication:

1. Dictionary-type machines. Here the proper move of the machine is decided in advance for each possible situation that may arise in the game and listed in a "dictionary" or function table. When a particular position arises, the machine merely looks up the move in the dictionary. Because of the extravagant memory requirements, this rather uninteresting method is only feasible for exceptionally simple games, e.g., tic-tac-toe.
2. Machines using rigorously correct playing formulas. In some games, such as Nim, a complete mathematical theory is known, whereby it is possible to compute by a relatively simple formula, in any position that can be won, a suitable winning move. A mechanization of this formula provides a
3. Machines applying general principles of approximate validity. In most games of interest to humans, no simple exact solution is known, but there are various general principles of play which hold in the majority of positions. This is true of such games as checkers, chess, bridge, poker and the like. Machines may be designed applying such the principles are not infallible, neither are the machines, as indeed, neither are humans.
4. Learning machines. Here the machine is given only the rules of the game and perhaps an elementary strategy of play, together with some method of improving this strategy through experience. Among the many methods that have been suggested for incorporation of learning we have:

   a) trial-and-error with retention of successful and elimination of unsuccessful possibilities;

   b) imitation of a more successful opponent;

   c) "teaching" by approval or disapproval, or by informing the machine of the nature of its mistakes; and finally

   d) self-analysis by the machine of its mistakes in an attempt to devise general principles.

Many examples of the first two types have been constructed and a few of the third. The fourth type, learning game-players, is reminiscent of Mark Twain's comment on the weather. Here is a real challenge for the programmer and machine designer.

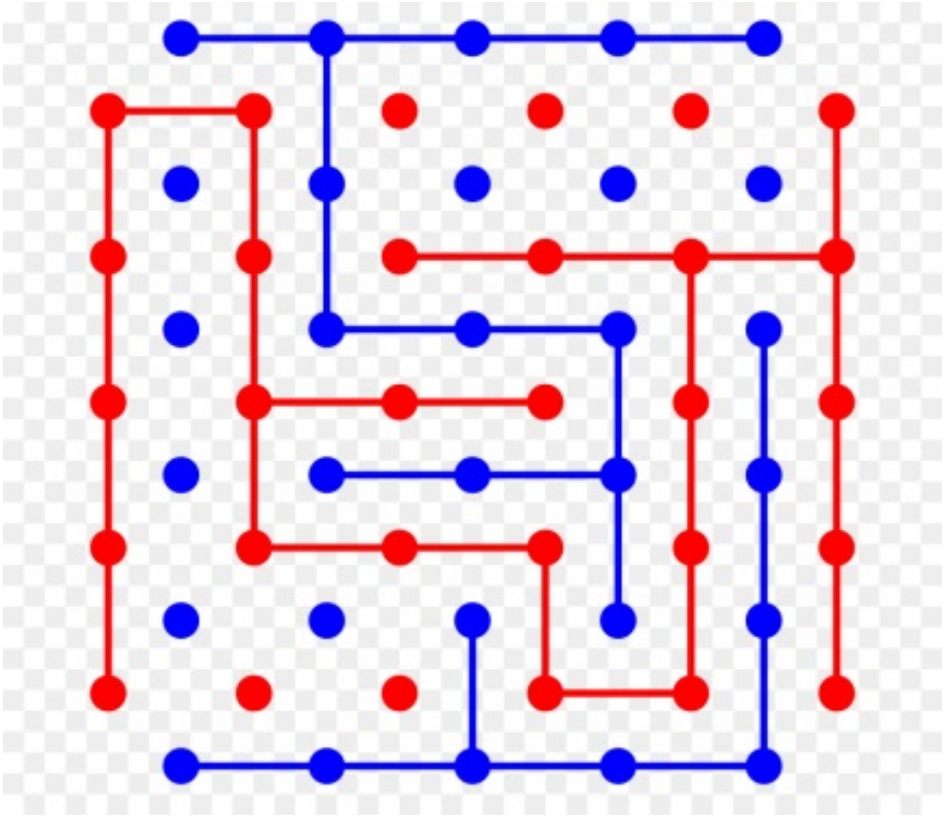
Two examples of the third category, machines applying general principles, may be of interest. The first of these is a machine designed by E. F. Moore and the writer for playing a commercial board game known as Hex. This game is played on a board laid out in a regular hexagon pattern, the two players alternately placing black and white pieces in unoccupied hexagons. The entire board forms a rhombus and Black's goal is to connect the top and bottom of this rhombus with a continuous chain of black pieces. White's goal is to connect the two sides of the rhombus with a chain of white pieces. After a study of this game, it was conjectured that a reasonably good move could be made by the following process. A two-dimensional potential field is set up corresponding to the playing board, with white pieces as positive charges and black pieces as negative charges. The top and bottom of the board are negative and the two sides positive. The move to be made corresponds to a certain specified saddle point in this field.

To test this strategy, an analog device was constructed, consisting of a resistance network and gadgetry to locate the saddle points. The general principle, with some improvements suggested by experience, proved to be reasonably sound. With first move, the machine won about seventy per cent of its games against human opponents. It frequently surprised its designers by choosing odd-looking moves which, on analysis, proved sound. We normally think of computers as expert at long involved calculations and poor in generalized value judgments. Paradoxically, the positional judgment of this machine was good; its chief weakness was in end-game combinatorial play. It is also curious that the Hex-player reversed the usual computing procedure in that it solved a basically digital problem by an anlog machine.

The game of checkers has recently been programmed into a general-purpose computer, using a "general prin-

C. E. Shannon, Computers and Automata, Proc. of IRE, vol. 41, pp. 1234-1241, 1953

Bridg-It: From Hex to Game of Gale to Shannon Switching Game



Piet Hein

John Nash

David Gale

Claude Shannon

Bridg-It Game and AI

The rules of Bridg-It

- There are two players. The goal is to connect the pair of opposite sides by building bridges across two dots, say Player A top to bottom and Player B left to right.

- Players build bridges to form a connected path of bridges from one side of the board to the opposite side while blocking their opponent from doing the same.

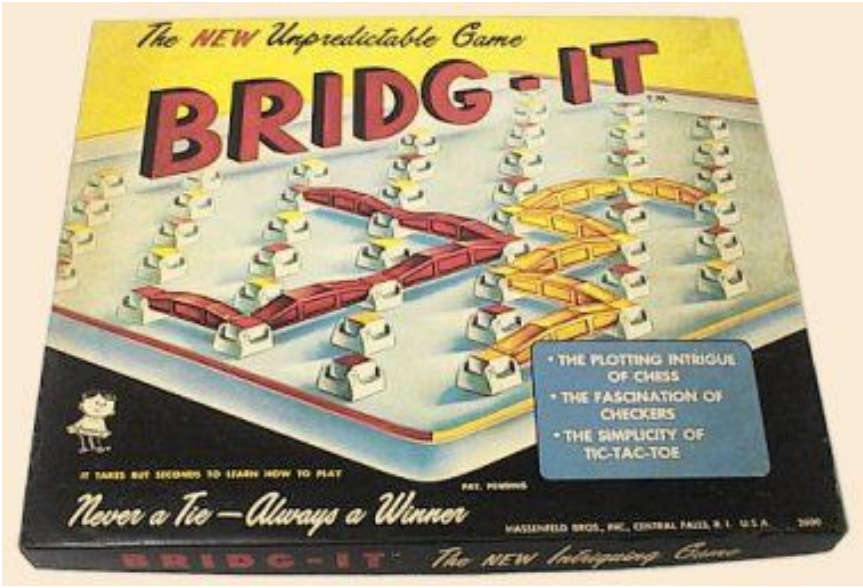
- The winner is the player who first connect his pair of sides.

Bridg-It: First-Mover Advantage

following reason. Although no "decision procedure" is known which will assure a win on a standard board, there is an elegant *reductio ad absurdum* "existence proof" that there is a winning strategy for the first player on a field of any size! (An existence proof merely proves the existence of something without telling you how to go about finding it.) The following is a highly condensed version of the proof (it can be formulated with much greater rigor) as it was worked out in 1949 by John Nash:

1. Either the first or second player must win, therefore there must be a winning strategy for either the first or second player.
2. Let us assume that the second player has a winning strategy.
3. The first player can now adopt the following defense. He first makes an arbitrary move. Thereafter he plays the winning second-player strategy assumed above. In short, he becomes the second player, but with an extra piece placed somewhere on the board. If in playing the strategy he is required to play on the cell where his first arbitrary move was made, he makes another arbitrary move. If later he is required to play where the second arbitrary move was made, he makes a third arbitrary move, and so on. In this way, he plays the winning strategy with one extra piece always on the field.
4. This extra piece cannot interfere with the first player's imitation of the winning strategy, for an extra piece is always an asset and never a handicap. Therefore the first player can win.
5. Since we have now contradicted our assumption that there is a winning strategy for the second player, we are forced to drop this assumption.
6. Consequently there must be a winning strategy for the first player.

- That the first-player has a winning strategy was proved by John Nash in 1949 – an existence proof.
- How to find this winning strategy is however computationally hard
- Popular game by Hasbro in 1964 – *Never a Tie – Always a Winner*!

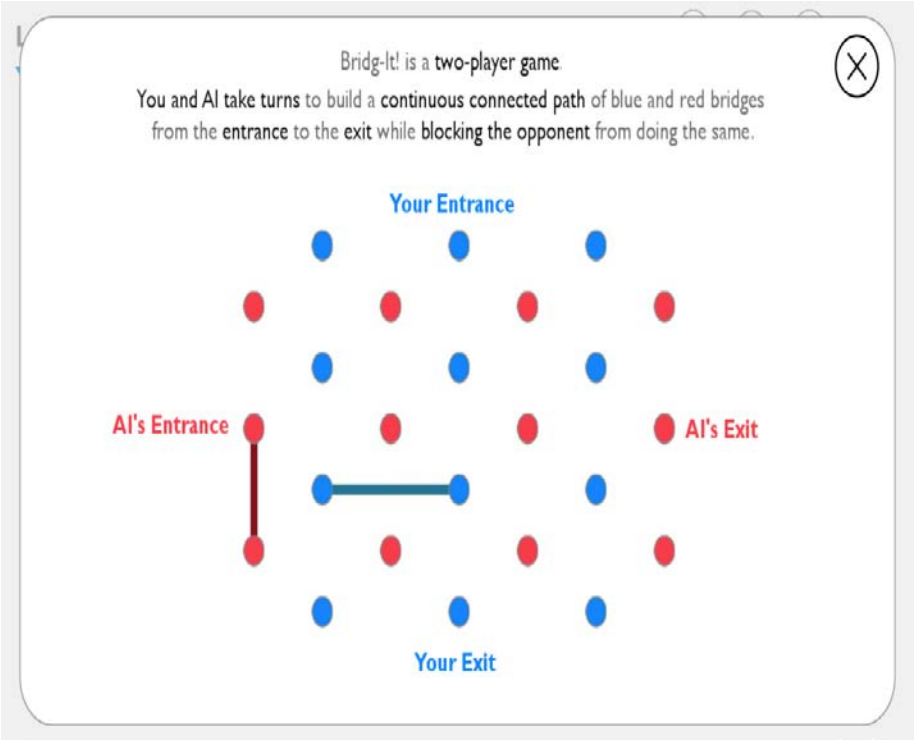


M. Gardner, The Scientific American Book of Mathematical puzzles & Diversions, Simon & Schuster, pp. 73-83, 1959.

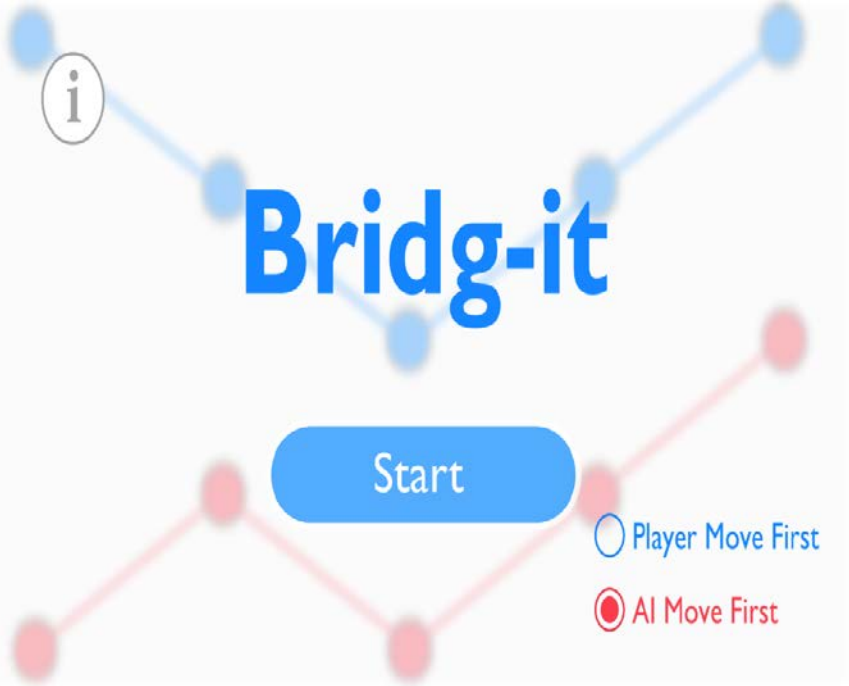
Bridg-It Game and AI

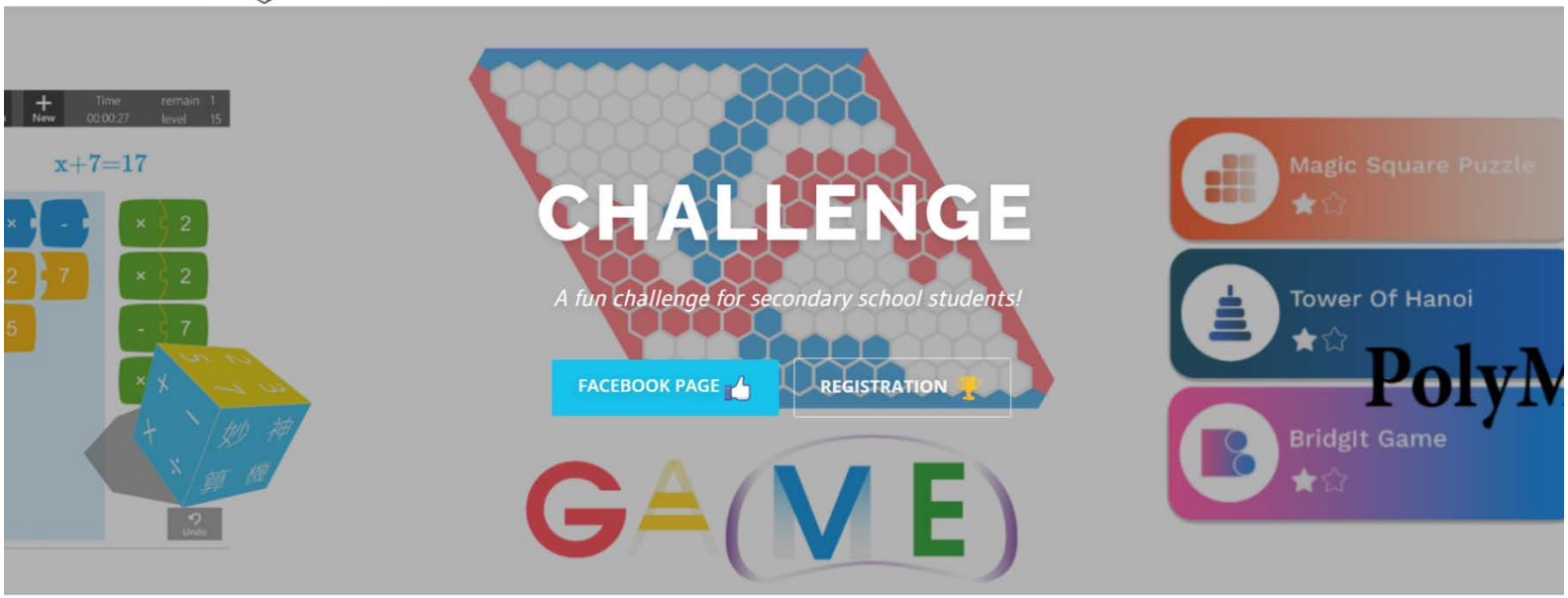
You and AI take turns to build a continuous connected path of blue and red bridges from the entrance to the exit while blocking the opponent from doing the same.

Screen 1: Introduction



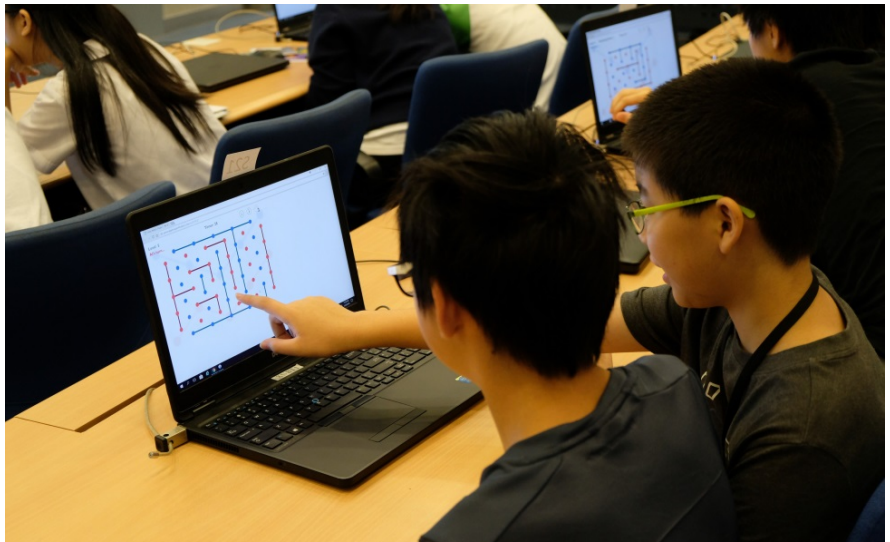
Screen 2: Start menu to select first move by either human player or AI



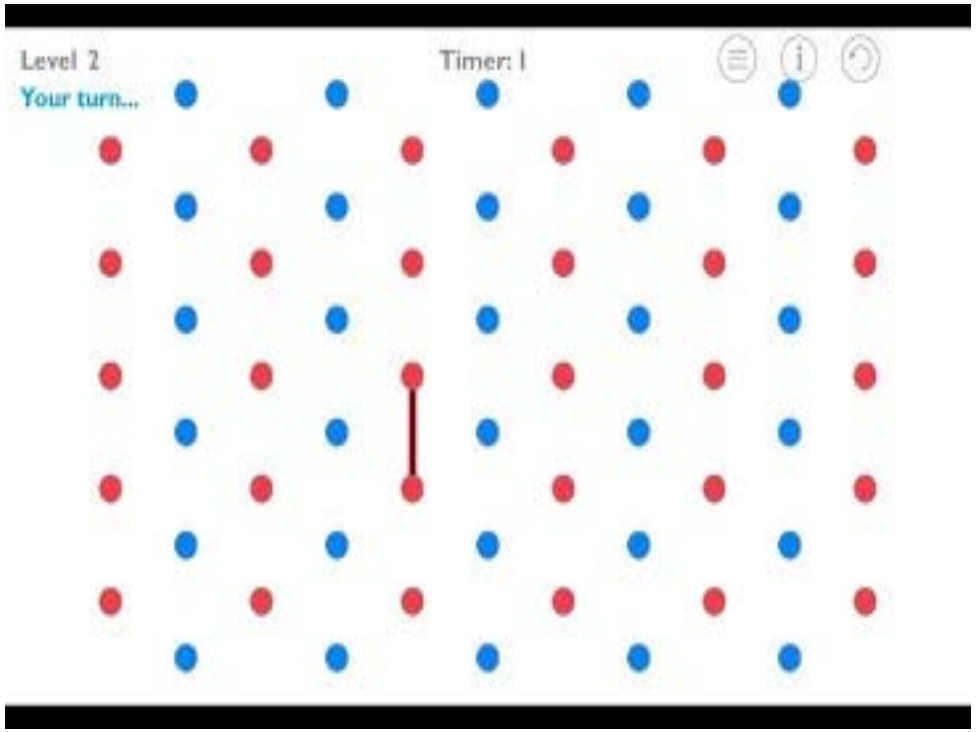


ABOUT 關於

COMPUTER SCIENCE CHALLENGE
電腦科學大挑戰

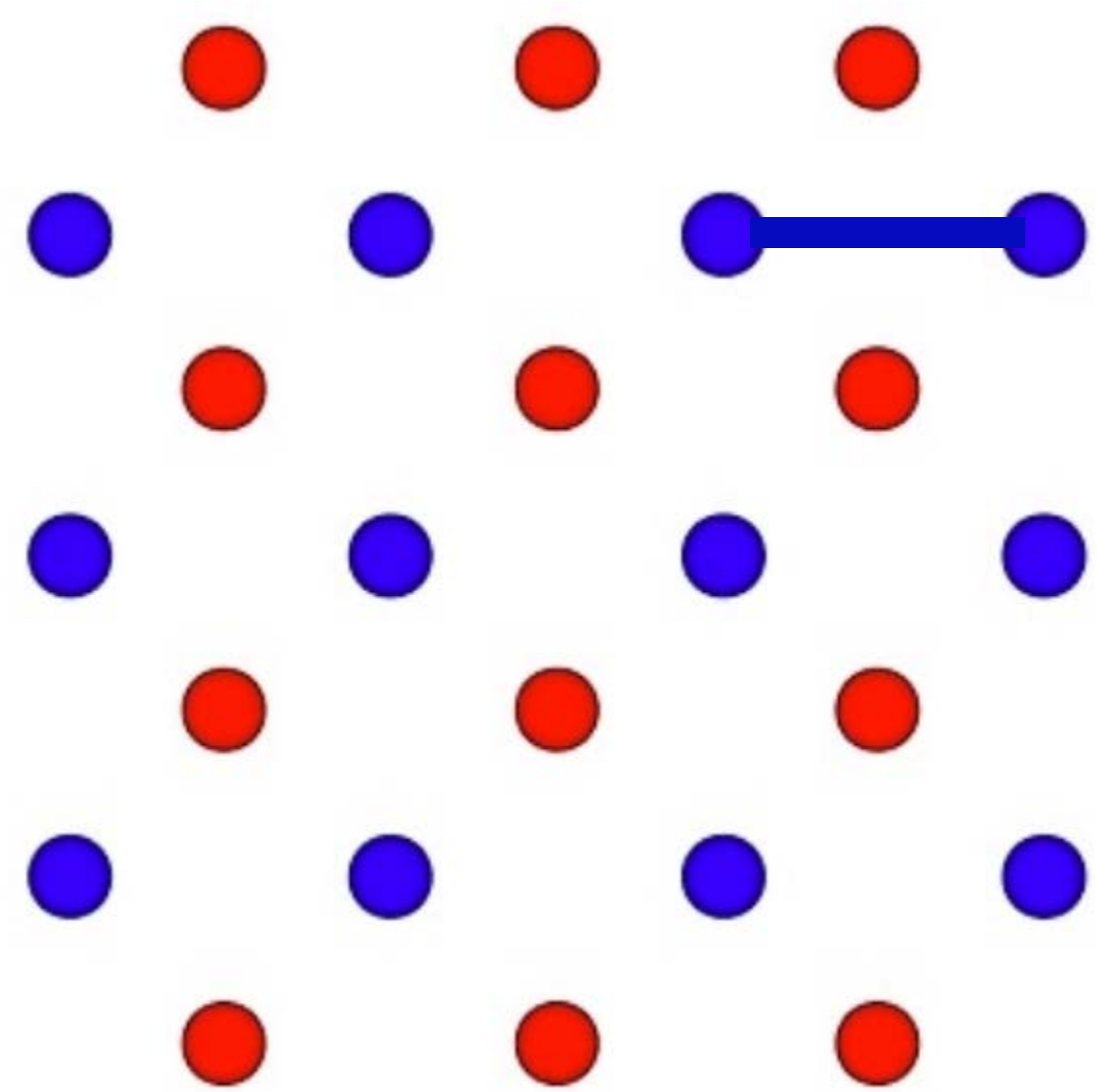


Activity: Try Bridg-It! Game

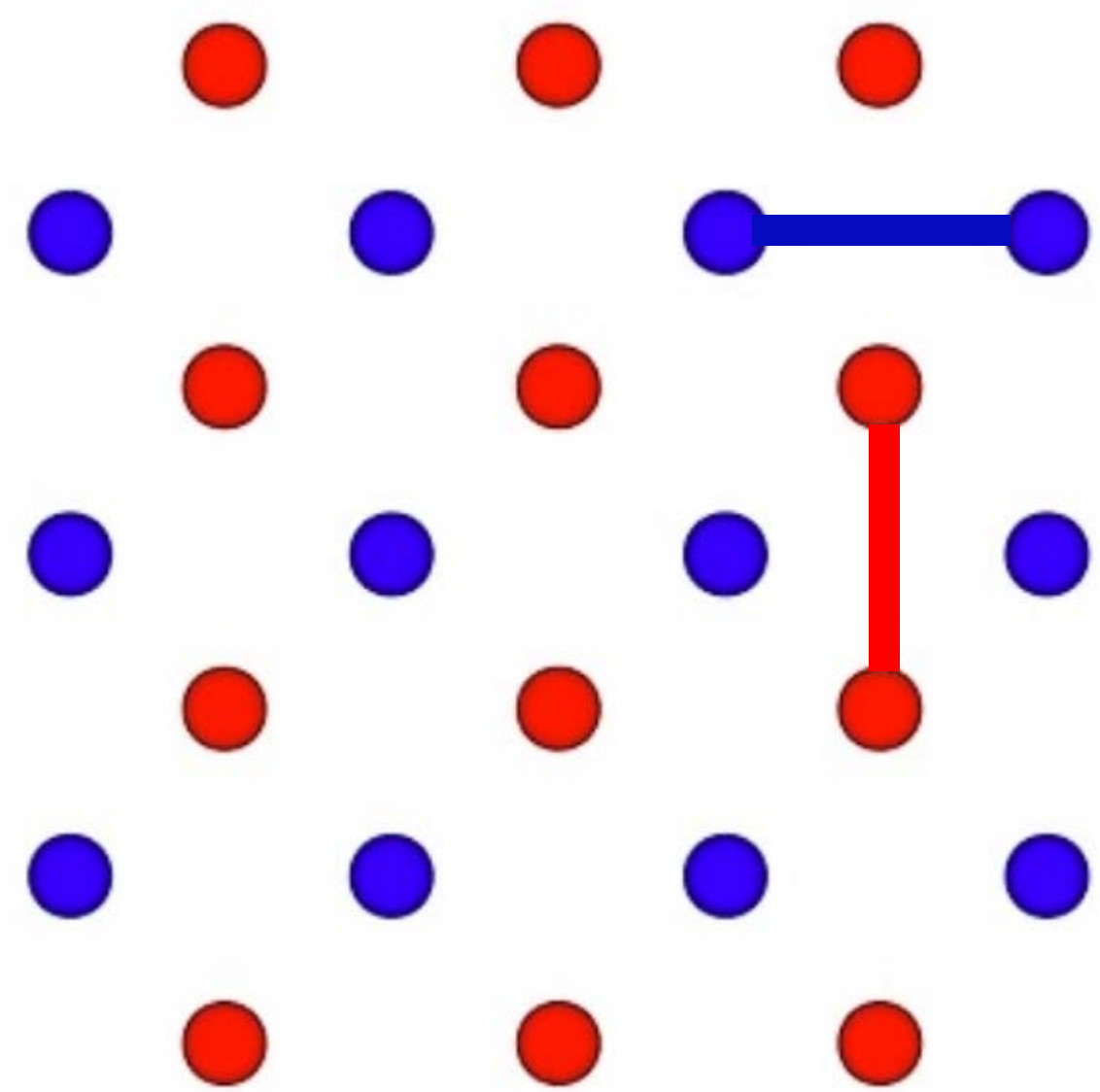


- What is a winning strategy?
- Can there be a draw?
- How would you know you are losing or winning?
- Can you make variants of the Bridg-It game?

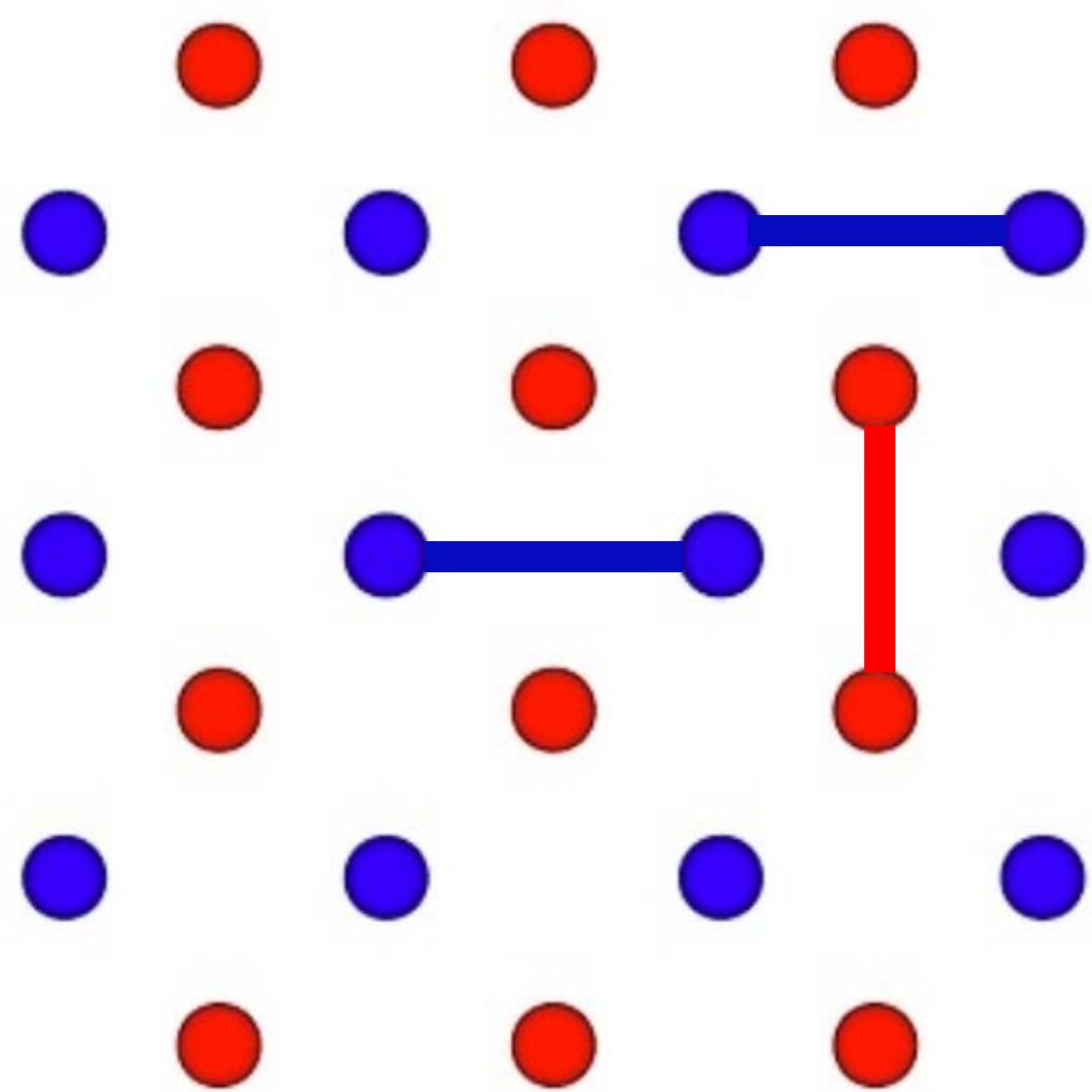
Bridg-It Game: Example 1



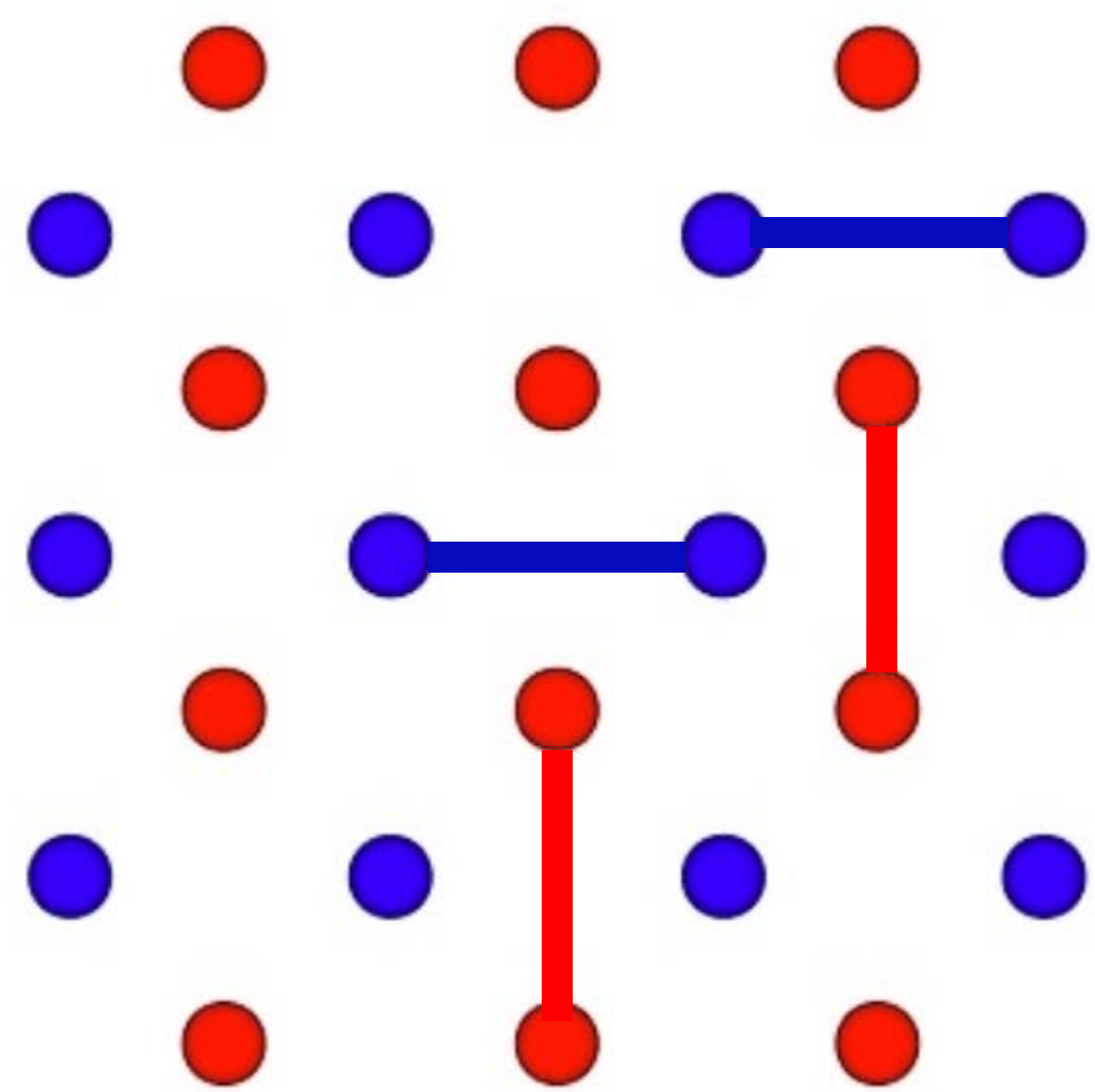
Bridg-It Game: Example 1



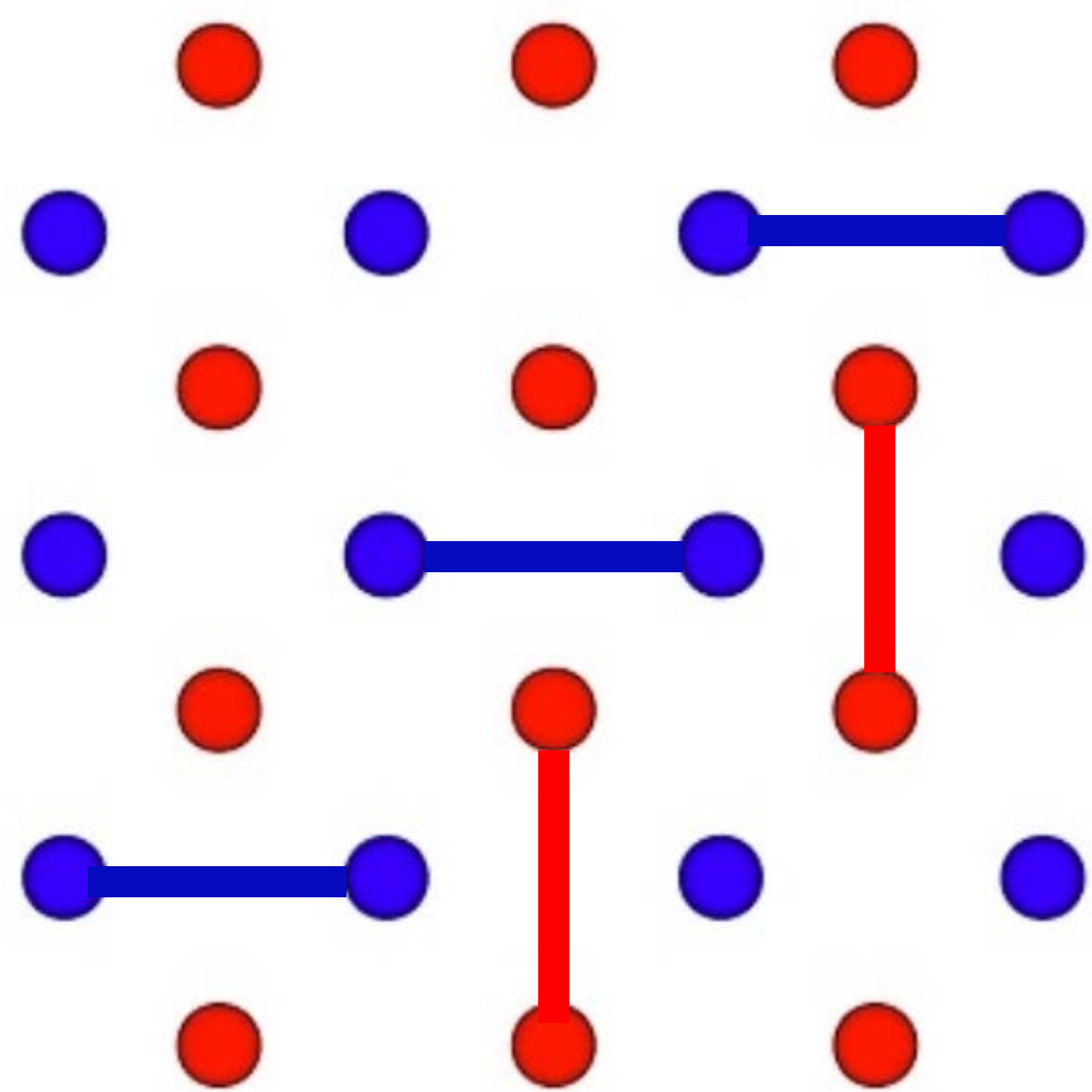
Bridg-It Game: Example 1



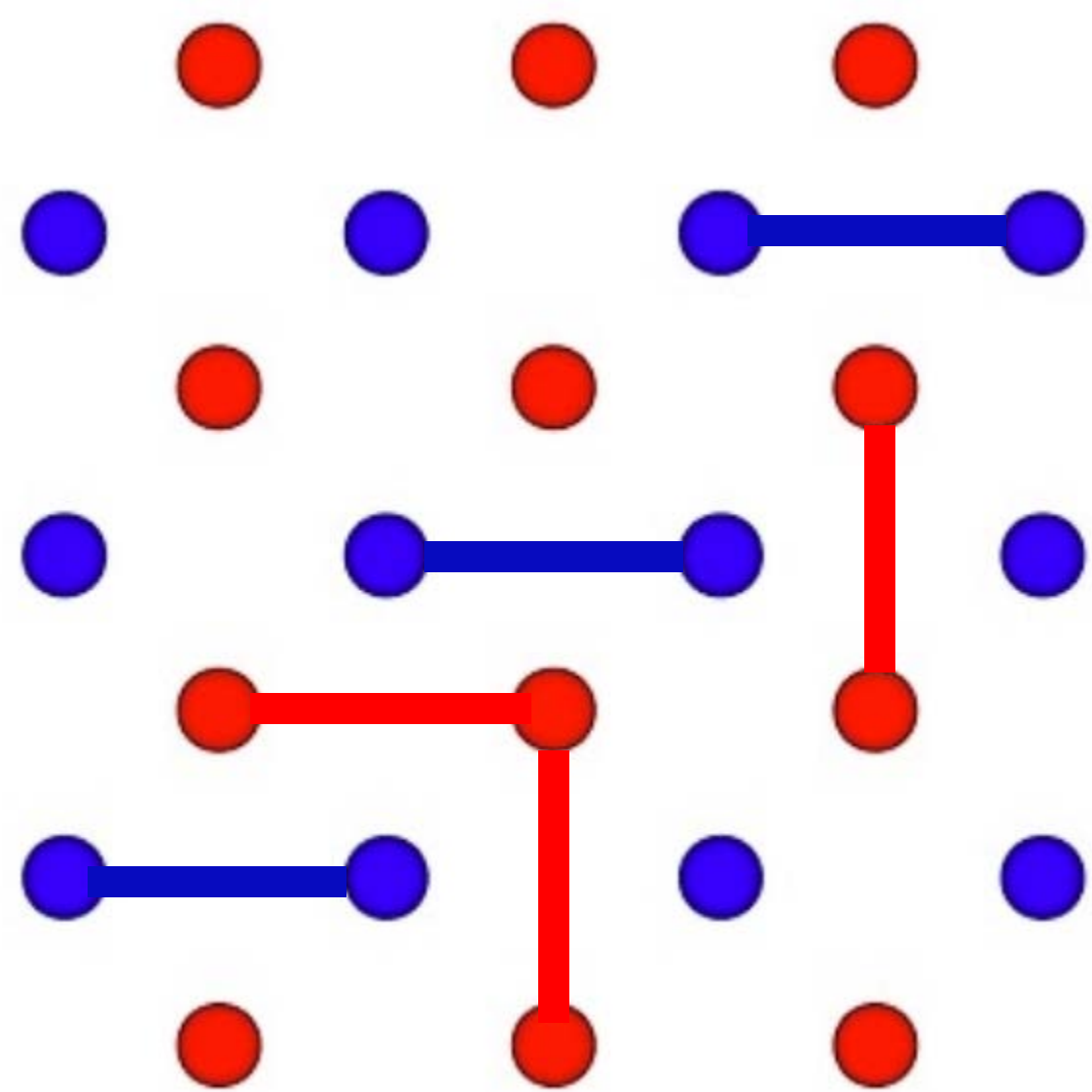
Bridg-It Game: Example 1



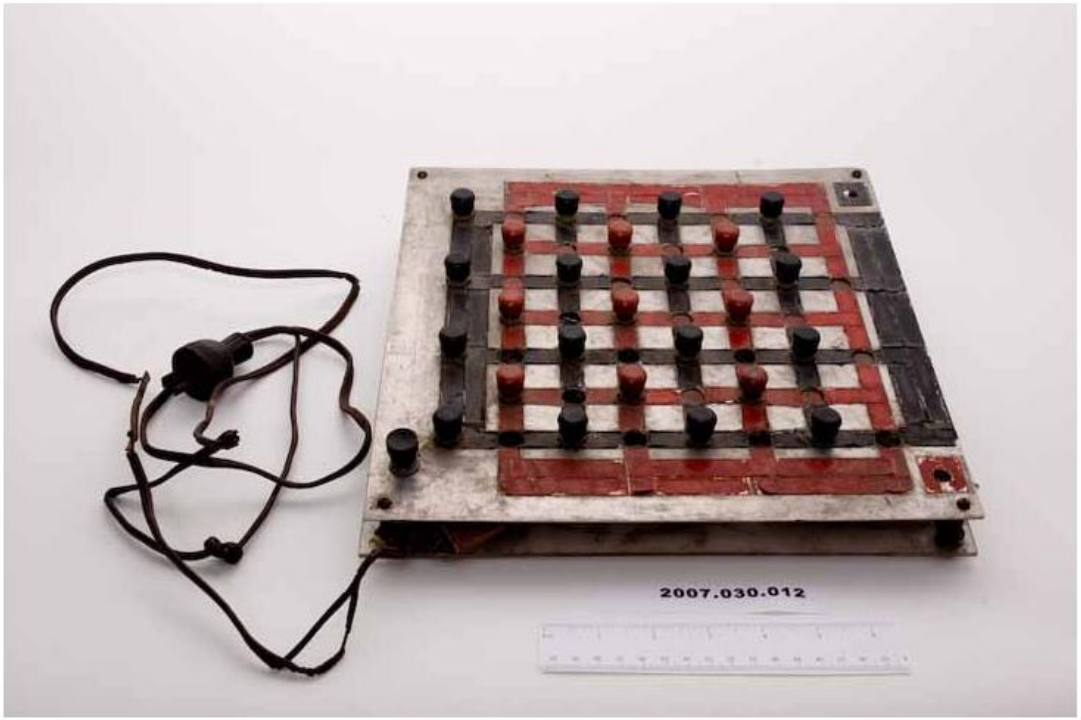
Bridg-It Game: Example 1



Bridg-It Game: Example 1



Bridg-It Game: Shannon's Machine



- Built by Shannon and Edward R. Moore in 1953 at AT&T Bell Labs
- With first move, the machine won about *seventy per cent of its games against human opponents*. It frequently *surprised its designers* by choosing odd-looking moves which, on analysis, proved sound. …its chief weakness was in end-game combinatorial play…..it *solved a basically digital problem by an analog machine*
- The Shannon's heuristic opens door to new results in many fields such as probability, graph theory and AI applications.

MIT Museum
https://webmuseum.mit.edu/detail.php?term=Shannon&module=objects&type=keyword&x=0&y=0&kv=75604&record=16&page=1

Bridg-It Game: Shannon's Machine

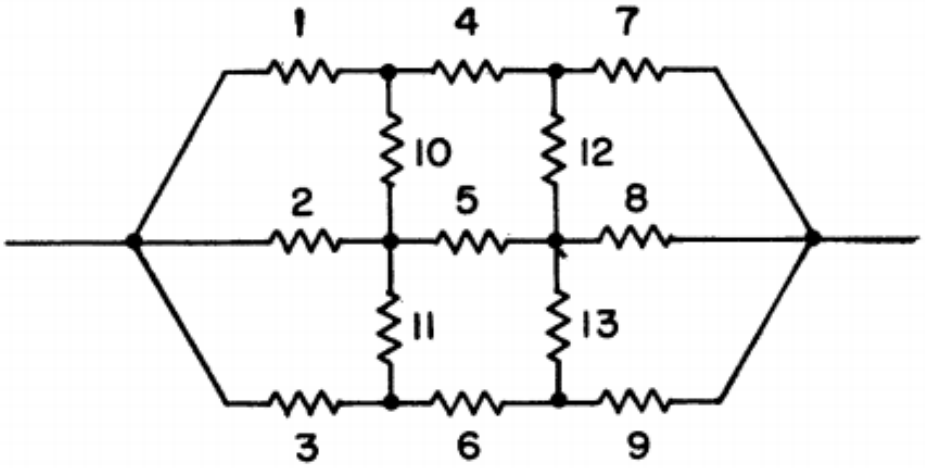
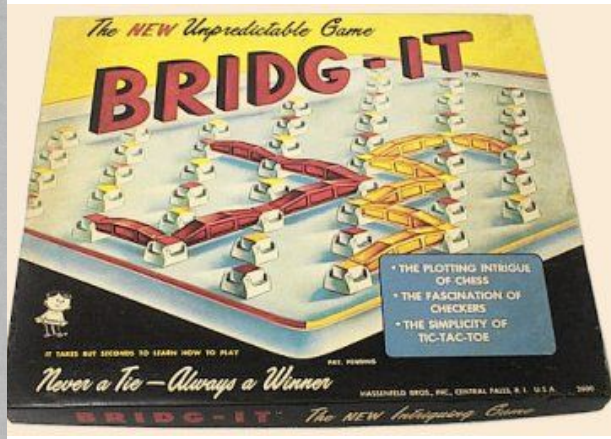


Fig. 11—This board game (due to C. E. Shannon) is played on a network of equal resistors. The first player's goal is to open the circuit between the endpoints; the second player's goal is to short the circuit. A move consists of opening or shorting a resistor. If the first player begins by opening resistor 1, the second player might counter by shorting resistor 4, following the strategy described in the text. The remaining move pairs (if *both* players use that strategy) would be (5, 8) (9, 13) (6, 3) (12, 10 or 2) (2 or 10 *win*). In this game the first player should be able to force a win, and the maximum-current strategy seems always to do so, even on larger networks.

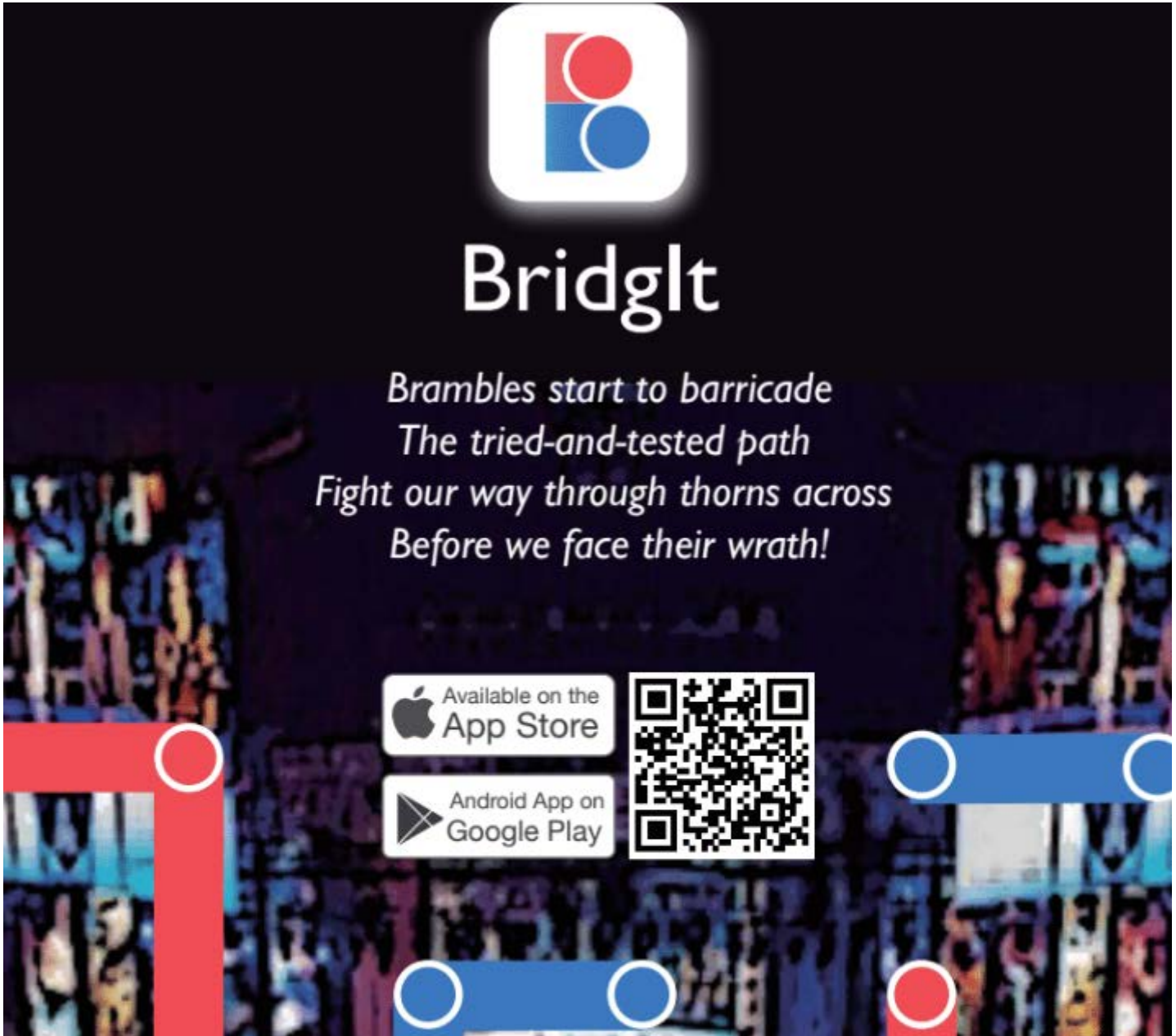
J. McCarthy, Steps towards Artificial Intelligence, Proceedings of the IRE, Vol 49, No: 1 1961
https://courses.csail.mit.edu/6.803/pdf/steps.pdf

Bridg-It Game: Shannon's Machine



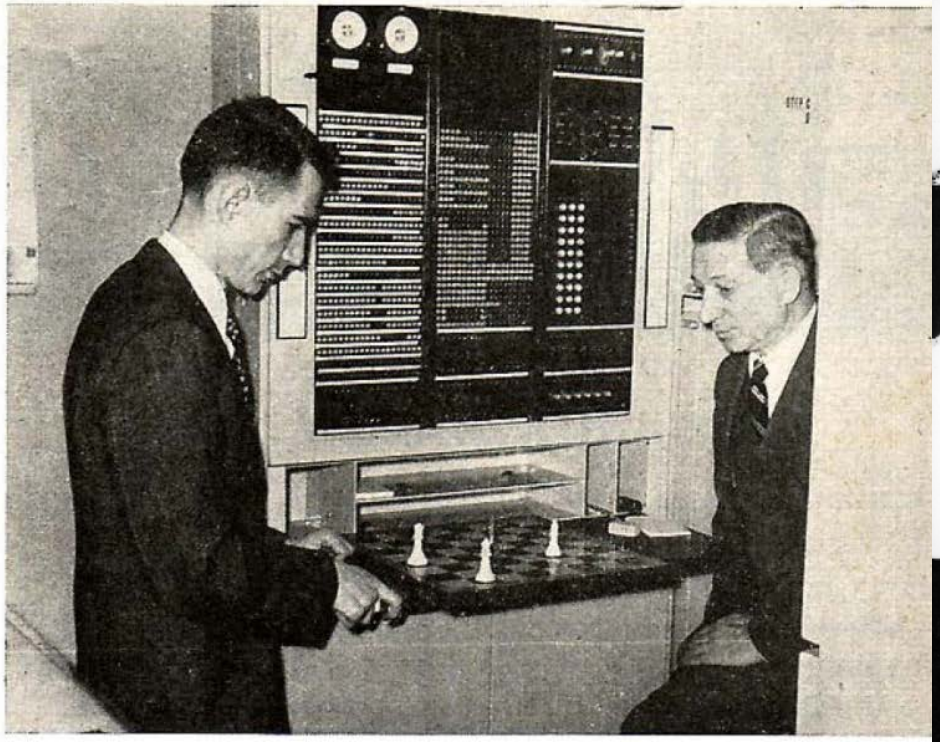
- Can you draw the circuit of resistors for the Shannon's heuristic?
- First play a 3x3 board and then play a 5x5 board. What insights do you get?

How about a Tournament with AI?

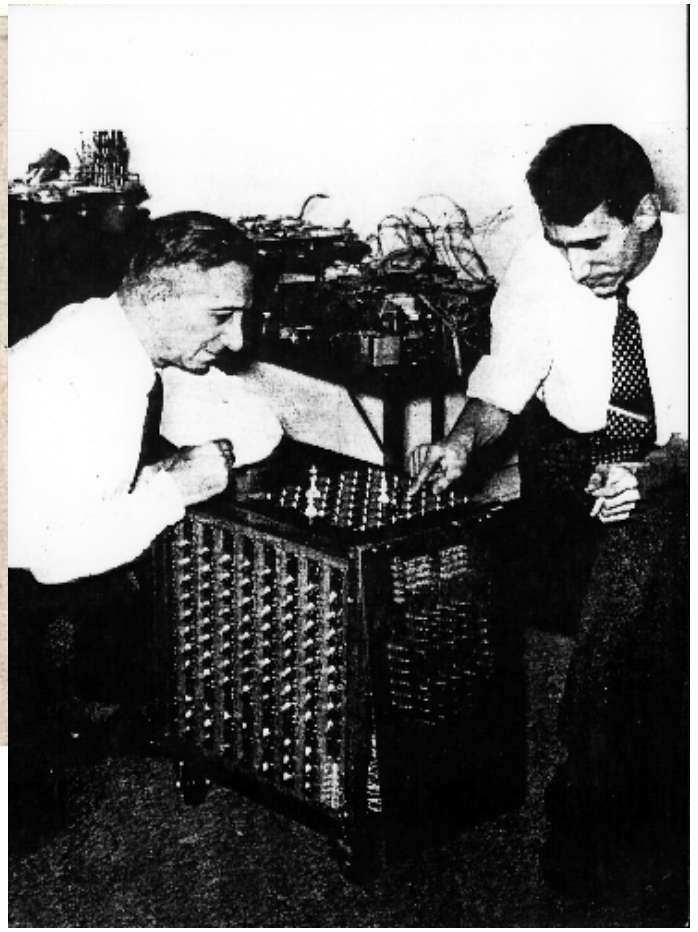


Programming a Computer for Playing Chess

Chapter XI on "The Electronic Chess Player" explains how Dr. Claude E. Shannon of the Bell Telephone Laboratories "programs" an electronic or a mechanic computer for chess play—and his conclusions in regard to the ability of a "chess robot."



Self-photo by Dr. E. Lasker (right)



https://www.computerhistory.org/chess/doc-431614f453dde/

Philosophical Magazine, Ser.7, Vol. 41, No. 314 - March 1950.

XXII. Programming a Computer for Playing Chess[1]

By CLAUDE E. SHANNON

Bell Telephone Laboratories, Inc., Murray Hill, N.J.[2]

[Received November 8, 1949]

1. INTRODUCTION

This paper is concerned with the problem of constructing a computing routine or "program" for a modern general purpose computer which will enable it to play chess. Although perhaps of no practical importance, the question is of theoretical interest, and it is hoped that a satisfactory solution of this problem will act as a wedge in attacking other problems of a similar nature and of greater significance. Some possibilities in this direction are: -

(1)Machines for designing filters, equalizers, etc.
(2)Machines for designing relay and switching circuits.
(3)Machines which will handle routing of telephone calls based on the individual circumstances rather than by fixed patterns.
(4)Machines for performing symbolic (non-numerical) mathematical operations.
(5)Machines capable of translating from one language to another.
(6)Machines for making strategic decisions in simplified military operations.
(7)Machines capable of orchestrating a melody.
(8)Machines capable of logical deduction.

It is believed that all of these and many other devices of a similar nature are possible developments in the immediate future. The techniques developed for modern electronic and relay type computers make them not only theoretical possibilities, but in several cases worthy of serious consideration from the economic point of view.

Machines of this general type are an extension over the ordinary use of numerical computers in several ways. First, the entities dealt with are not primarily numbers, but rather chess positions, circuits, mathematical expressions, words, etc. Second, the proper procedure involves general principles, something of the nature of judgement, and considerable trial and error, rather than a strict, unalterable computing process. Finally, the solutions of these problems are not merely right or wrong but have a continuous range of "quality" from the best down to the worst. We might be satisfied with a machine that designed good filters even though they were not always the best possible.

---

1 First presented at the National IRE Convention, March 9, 1949, NewYork, U.S.A.
2 Communicated by the Author

first move. This is in contrast with card games, backgammon, etc. Furthermore, in chess each of the two opponents has "perfect information" at each move as to all previous moves (in contrast with Kriegspiel, for example). These two facts imply (von Neumann and Morgenstern, 1944) that any given position of the chess pieces must be either: -

(1) A won position for White. That is, White can force a win, however Black defends.
(2) A draw position. White can force at least a draw, however Black plays, and likewise Black can force at least a draw, however White plays. If both sides play correctly the game will end in a draw.
(3) A won position for Black. Black can force a win, however White plays.

This is, for practical purposes, of the nature of an existence theorem. No practical method is known for determining to which of the three categories a general position belongs. If there were chess would lose most of its interest as a game. One could determine whether the initial position is a won, drawn, or lost for White and the outcome of a game between opponents knowing the method would be fully determined at the choice of the first move. Supposing the initial position a draw (as suggested by empirical evidence from master games [1]) every game would end in a draw.

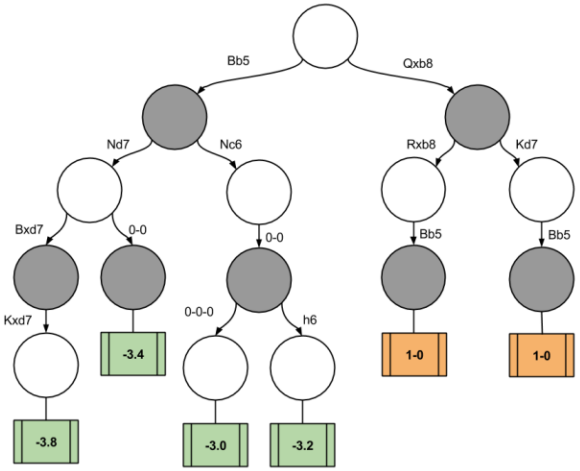
It is interesting that a slight change in the rules of chess gives a game for which it is provable that White has at least a draw in the initial position. Suppose the rules the same as those of chess except that a player is not forced to move a piece at his turn to play, but may, if he chooses, "pass". Then we can prove as a theorem that White can at least draw by proper play. For in the initial position either he has a winning move or not. If so, let him make this move. If not, let him pass. Black is not faced with essentially the same position that White has before, because of the mirror symmetry of the initial position [2]. Since White had no winning move before, Black has none now. Hence, Black at best can draw. Therefore, in either case White can at least draw.

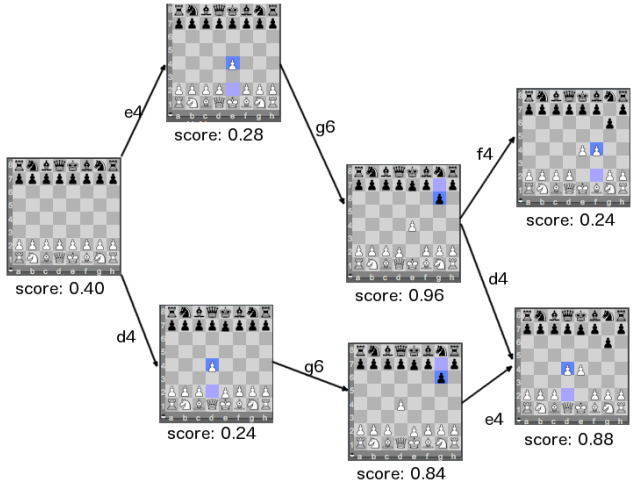
In some games there is a simple *evaluation function* f(P) which can be applied to a position P and whose value determines to which category (won, lost, etc.) the position P belongs. In the game of Nim (Hardy and Wright, 1938), for example, this can be determined by writing the number of matches in each pile in binary notation. These numbers are arranged in a column (as though to add them). If the number of ones in each column is even, the position is lost for the player about to move, otherwise won.

If such an evaluation function f(P) can be found for a game is easy to design a machine capable of perfect play. It would never lose or draw a won position and never lose a drawn position and if the opponent ever made a mistake the machine would capitalize on it. This

Game Tree

- A game tree is a type of recursive search function that examines all possible moves of a strategy game in order to ascertain an optimal sequence of moves. They are the workhorse in AI games that have a moderate number of possible choices per play.

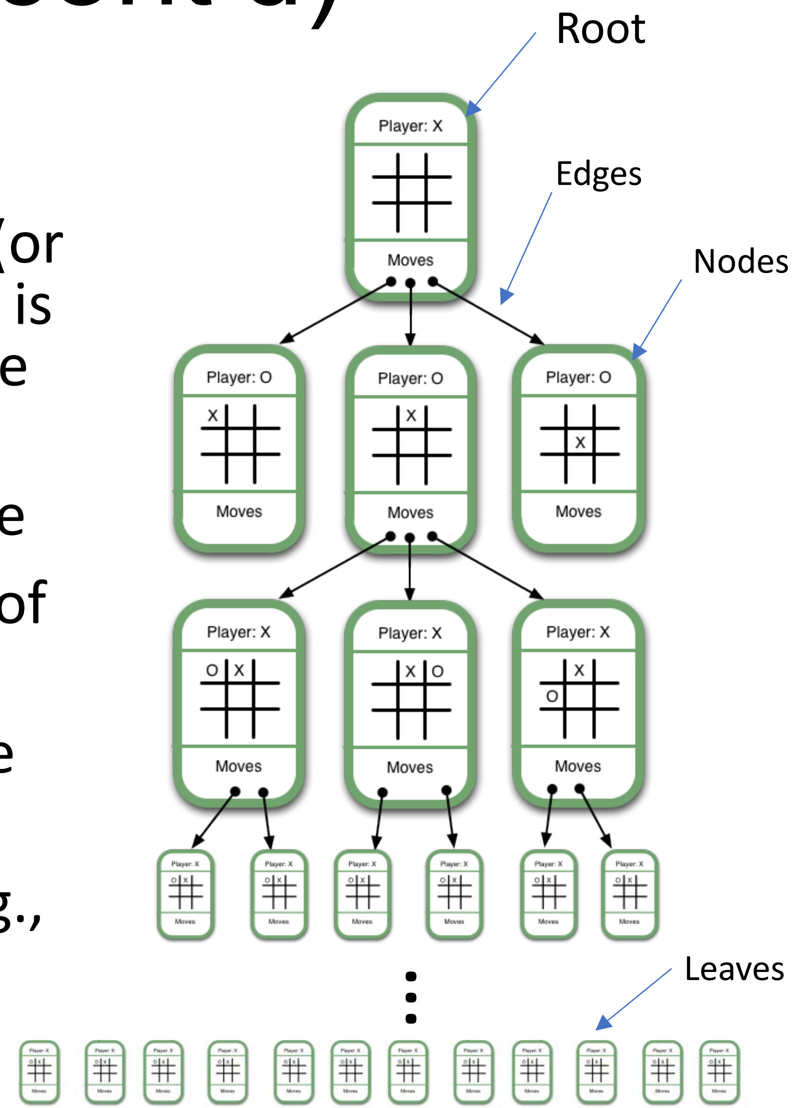




Game Tree (cont'd)

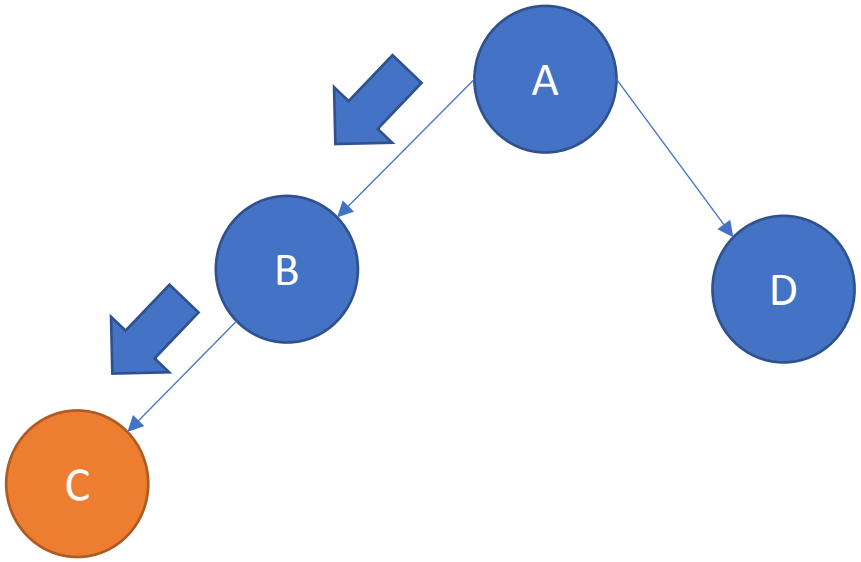
Each game consists of a problem space, an initial state, and a single (or a set of) goal states. The game tree is an abstraction of the problem space visualized as a rooted tree:

1. Root node represents initial state
2. Nodes represent feasible states of the game
3. Branching edges represent move options
4. Leaves represent final states (e.g., win, loss or draw)



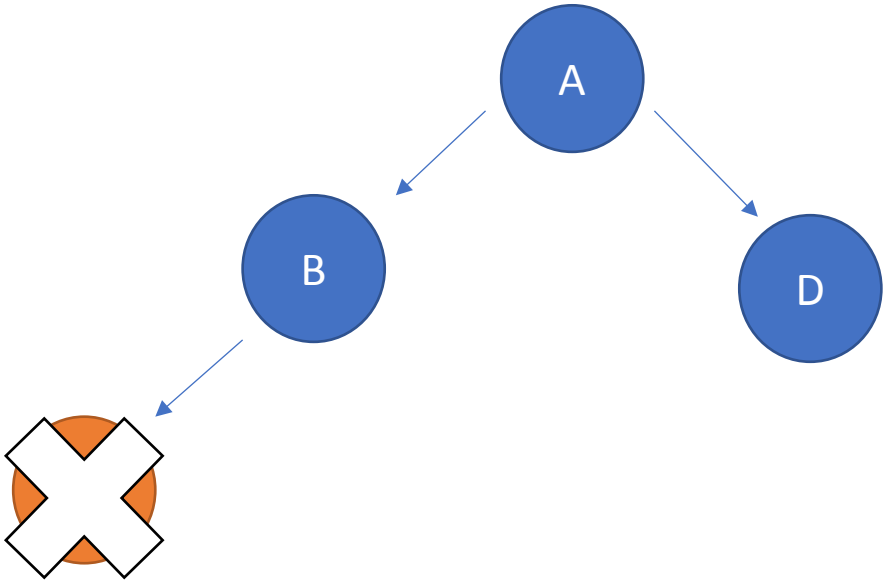
What is Backtracking?

- When you reach a dead-end



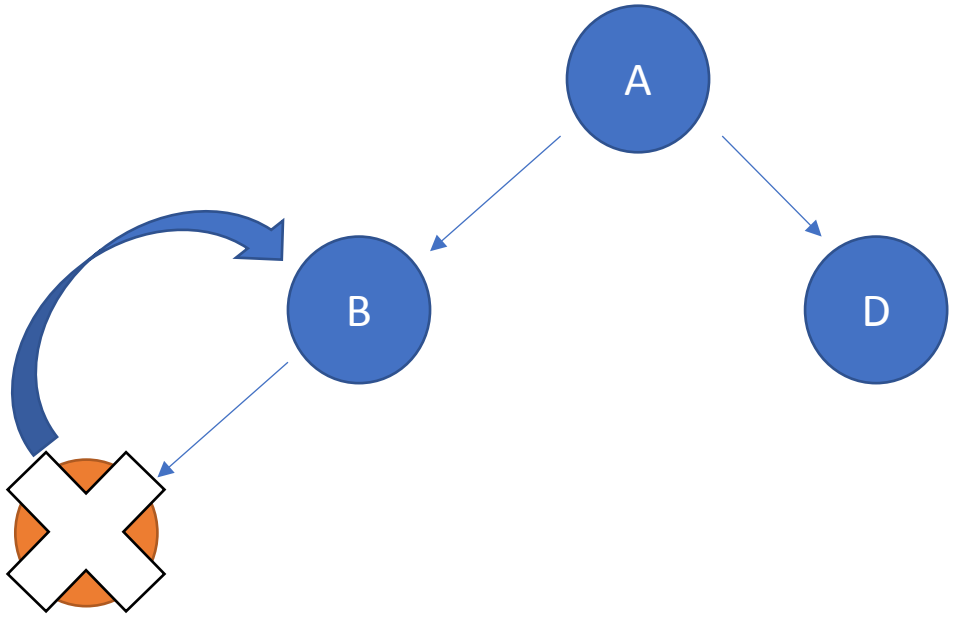
What is Backtracking?(cont.)

- When you reach a dead-end
  - “Backtrack” from the most recent choice



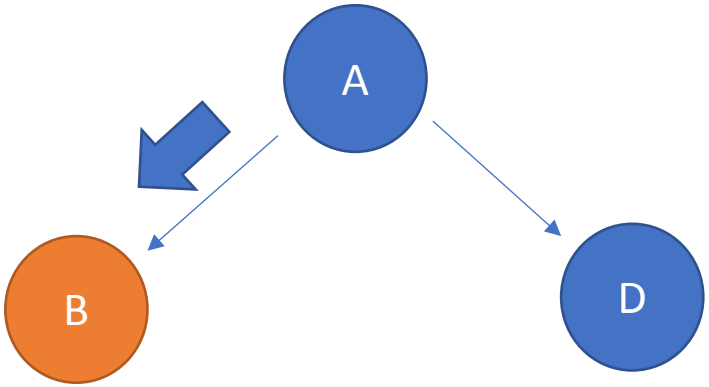
What is Backtracking?(cont.)

- When you reach a dead-end
  - “Backtrack” from the most recent choice
    - Undo its consequences



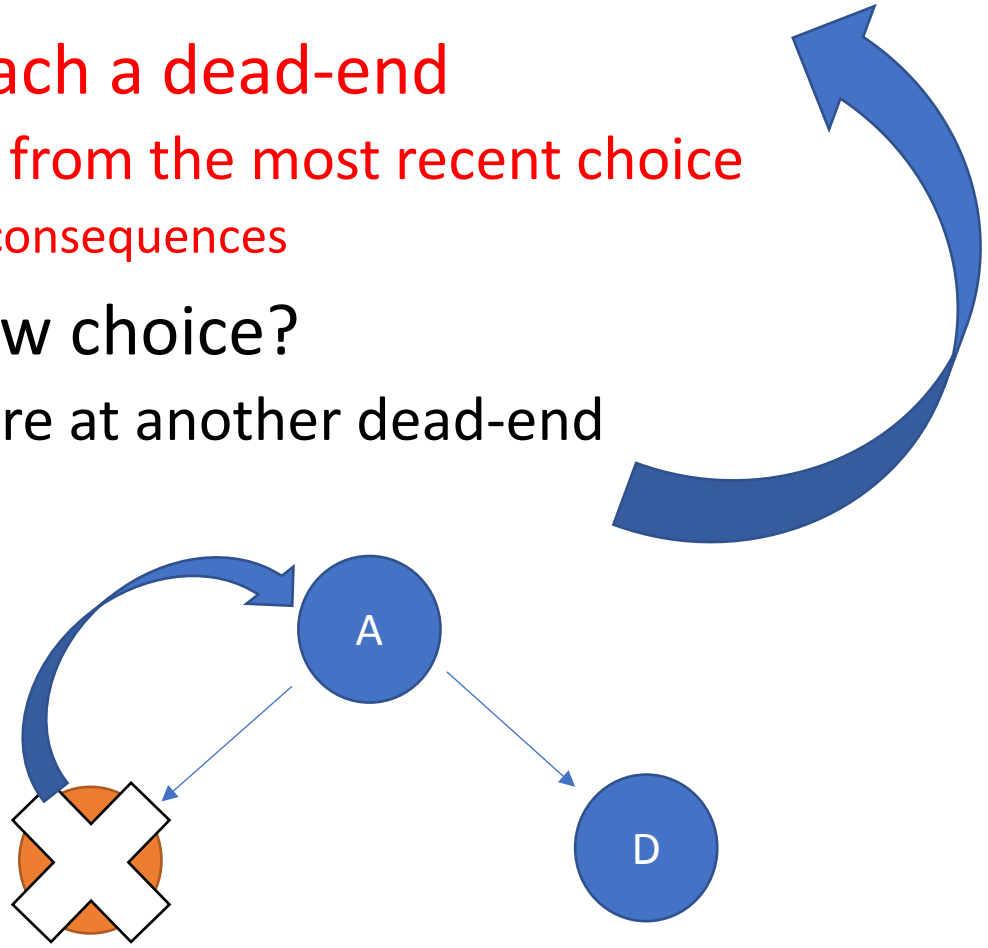
What is Backtracking?(cont.)

- When you reach a dead-end
  - “Backtrack” from the most recent choice
    - Undo its consequences
- "Is there a new choice?”
  - If not, you are at another dead-end



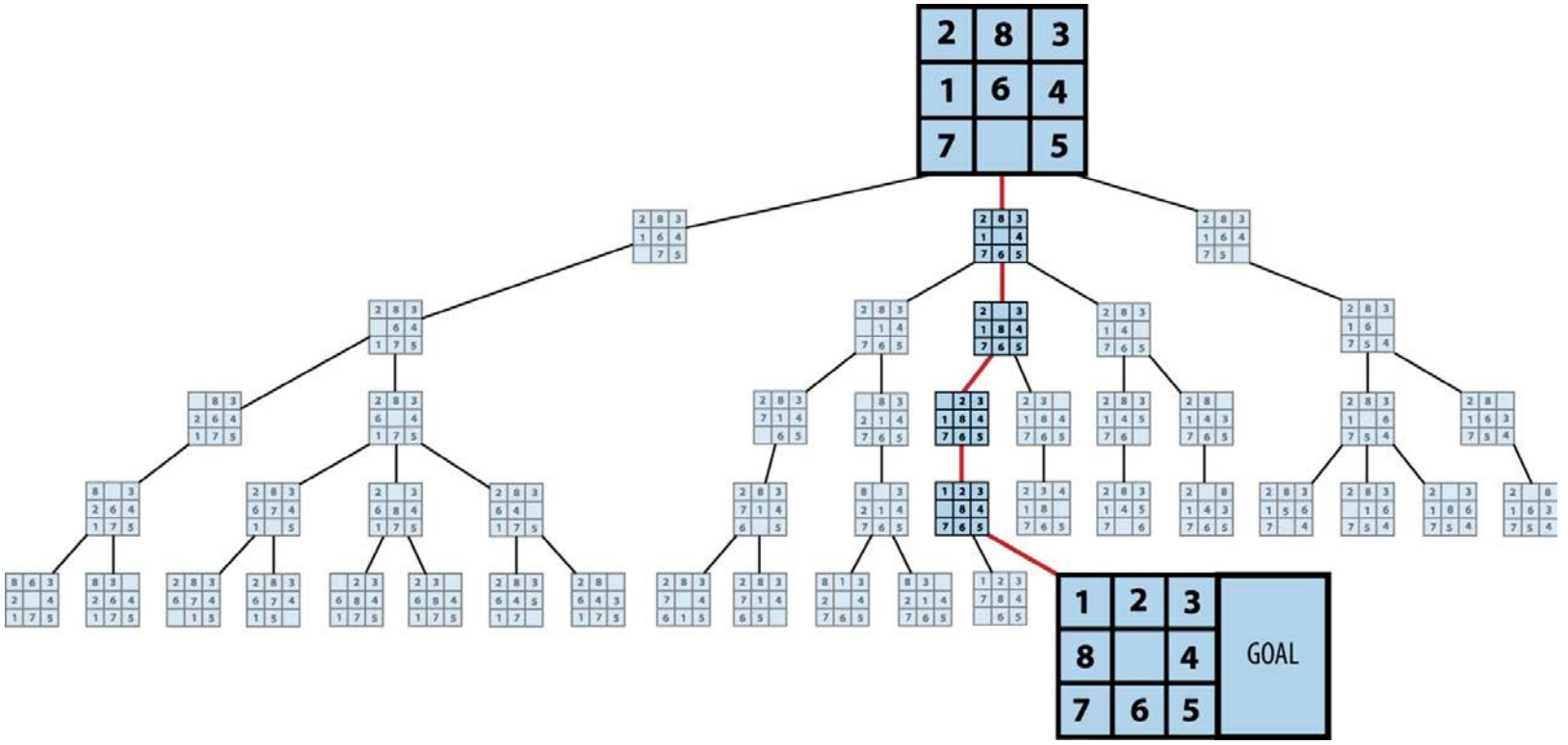
What is Backtracking?(cont.)

- When you reach a dead-end
  - “Backtrack” from the most recent choice
    - Undo its consequences
- "Is there a new choice?
  - If not, you are at another dead-end



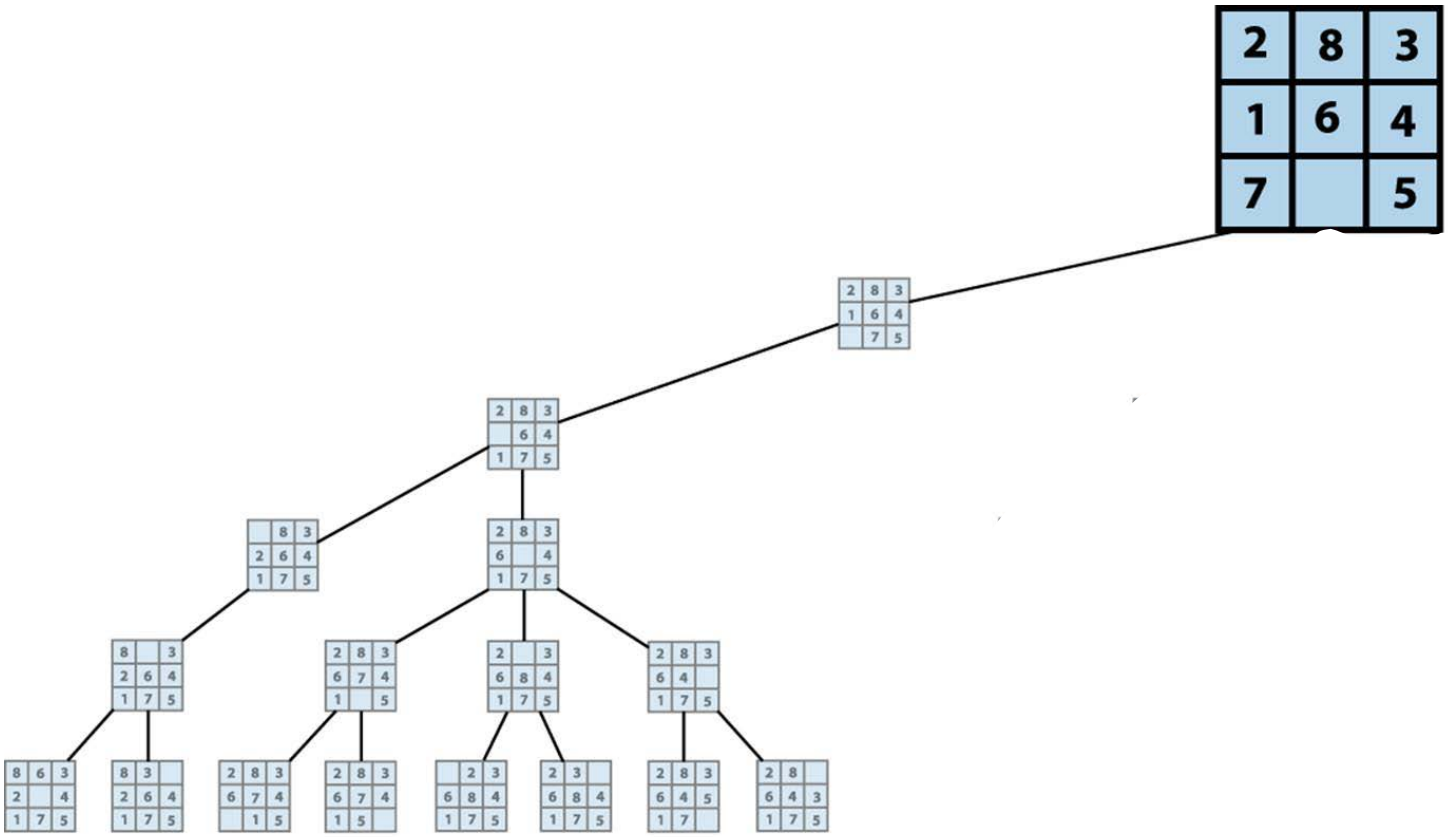
Example 1: 8-Puzzle game

- Nodes: the different permutations of the tiles.
- Edges: moving the blank tile up, down, right or left.
- There are three options: move 7, move 6 or move 5



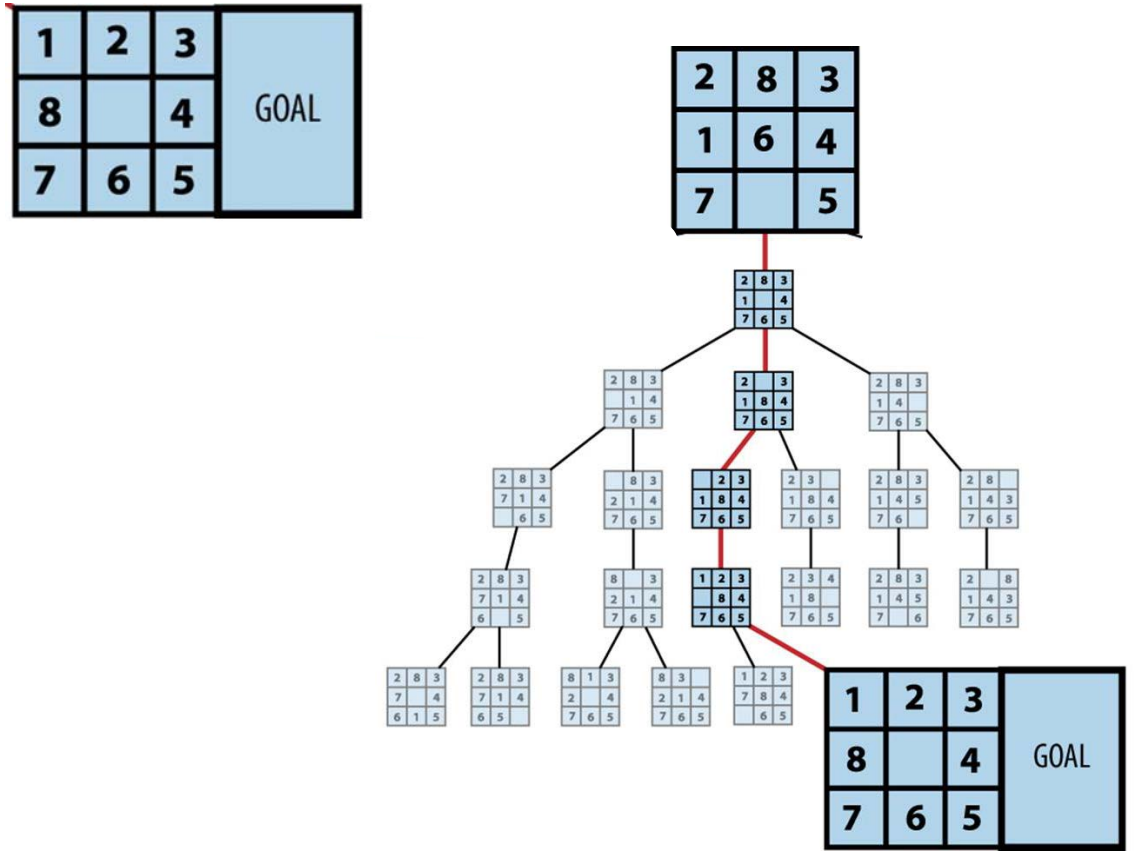
Example 1: 8-Puzzle game

If we move 7, what are the states you can move to?



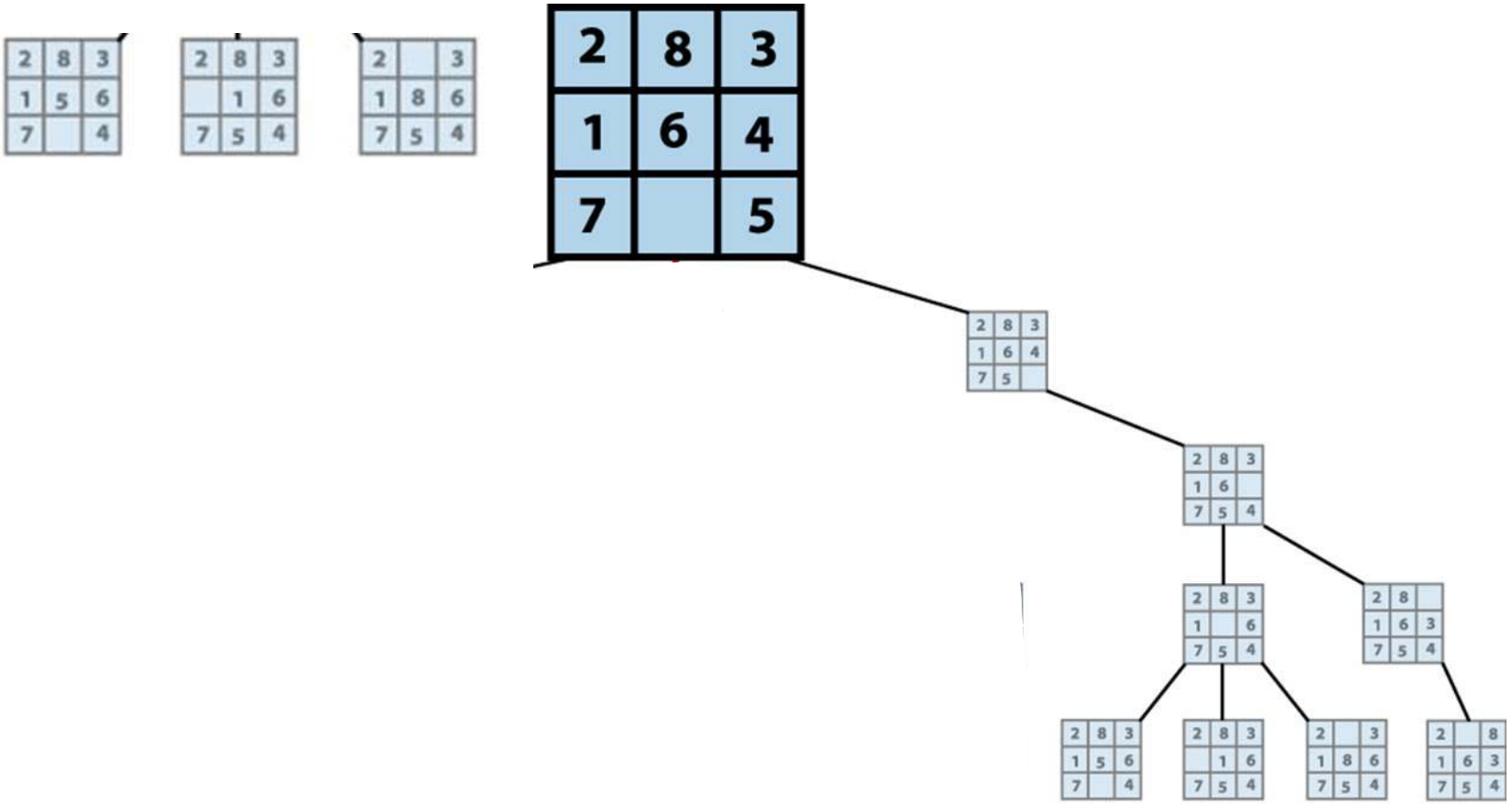
Example 1: 8-Puzzle game

If we move 6, what are the states you can move to?



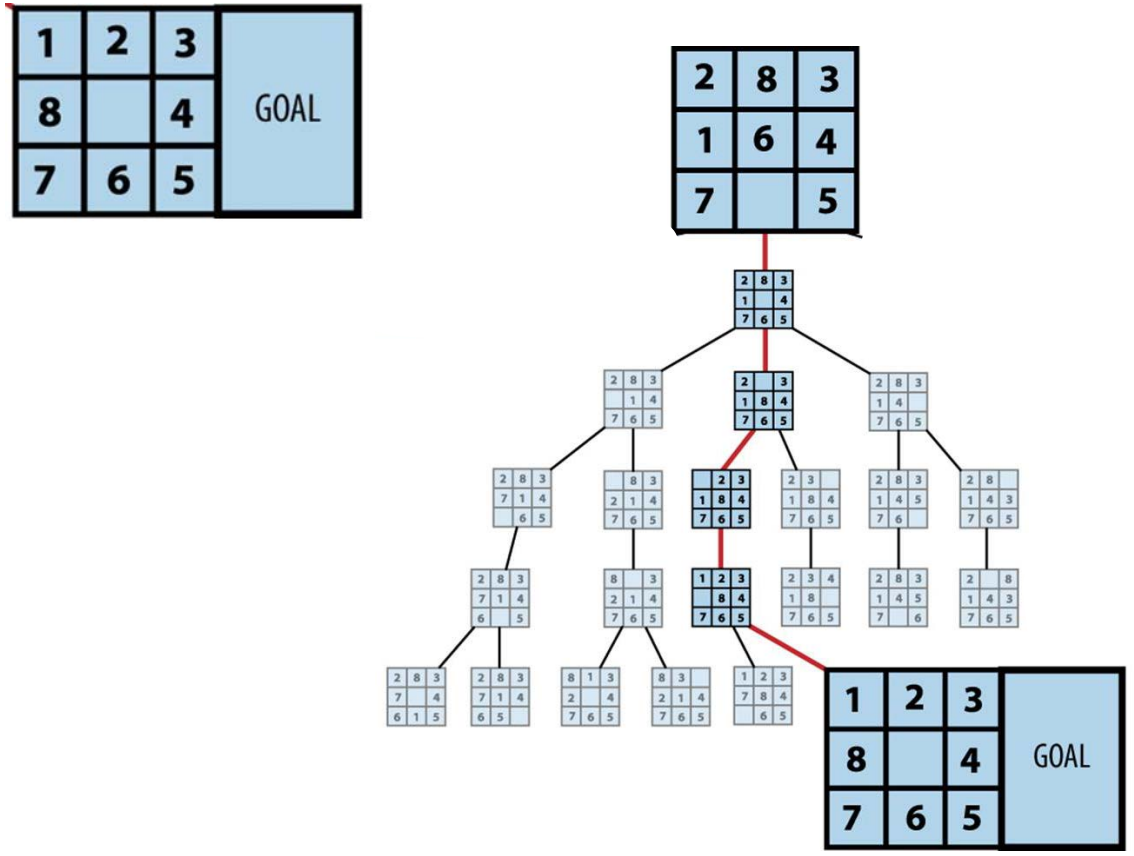
Example 1: 8-Puzzle game

If we move 5, what are the states you can move to?

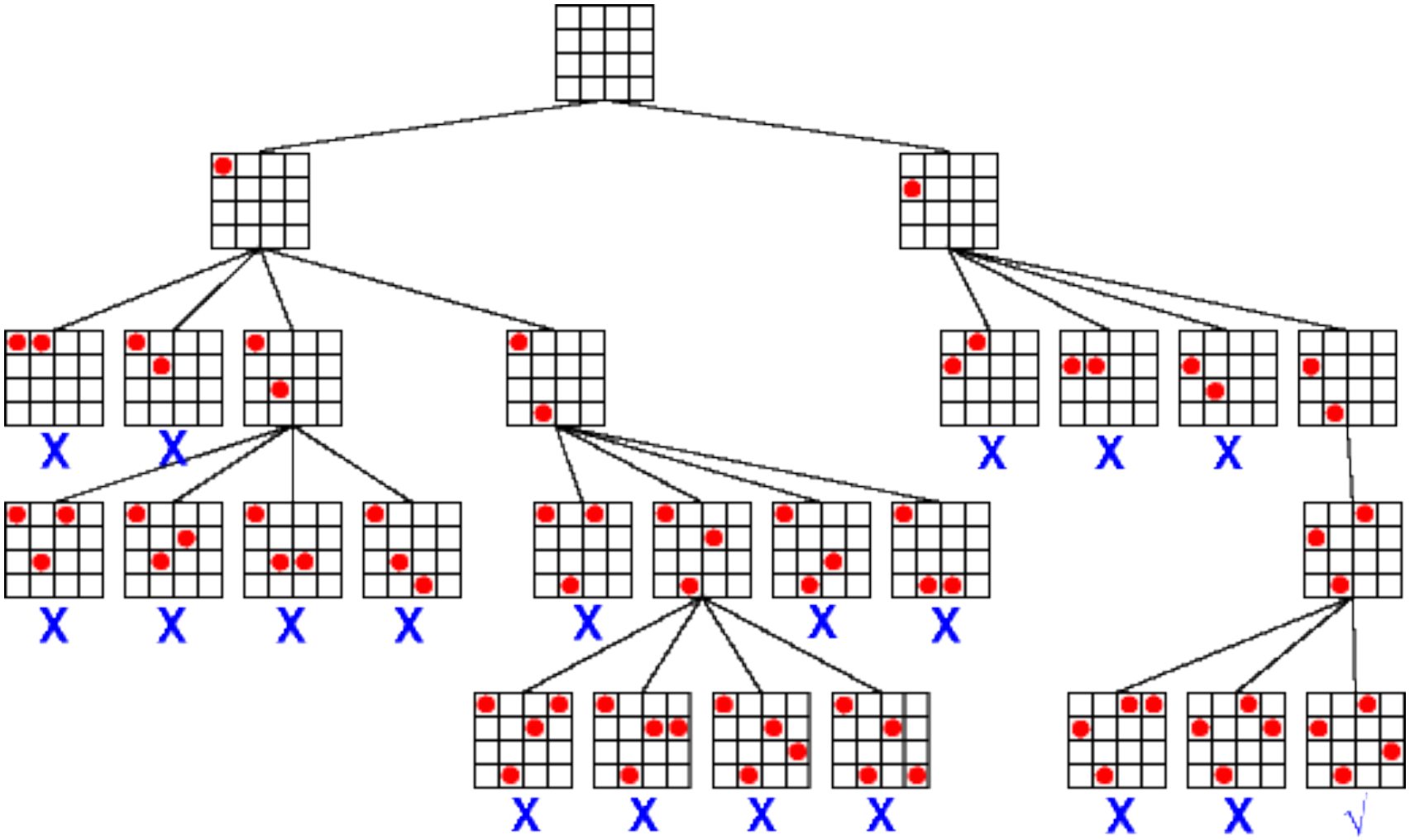


Example 1: 8-Puzzle game

Based on this analysis, move 6 would be the best choice



Example 2: The 4-Queens problem



The idea is to place queens one by one in different columns, starting from the leftmost column. When we place a queen in a column, we check for constraint violation with previously-placed queens. In the current column, if we find a row for which there is no violation, we record its row and column and set it as a partial solution.

Example 3: Fish-Flavored Lollipops

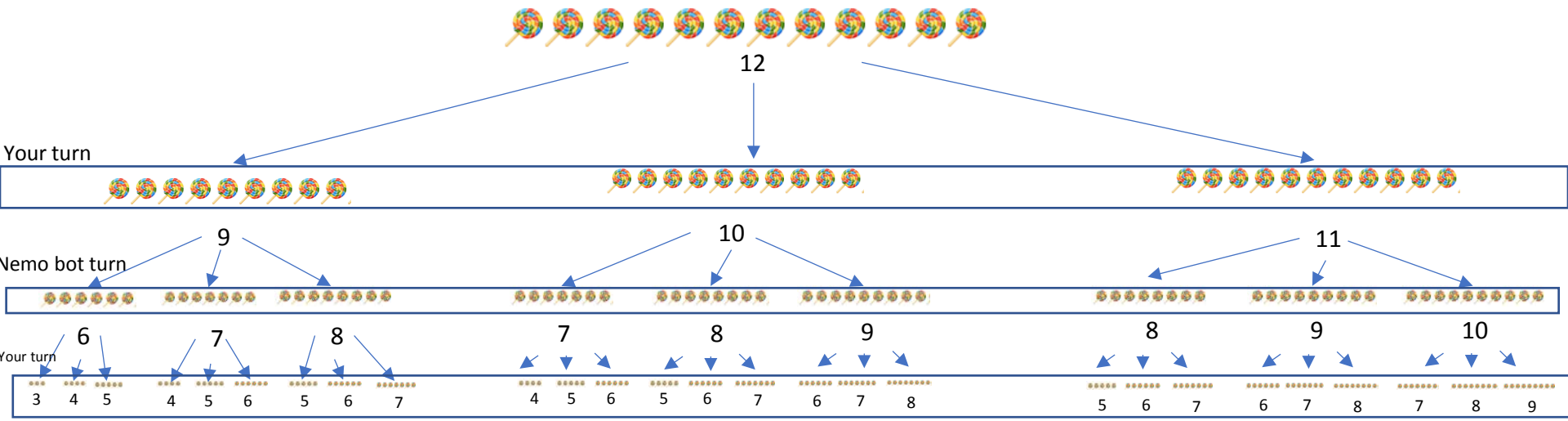
- Draw the game tree to win Nemo bot!

Fish-Flavored Lollipops is a variant of Nim, an ancient math puzzle

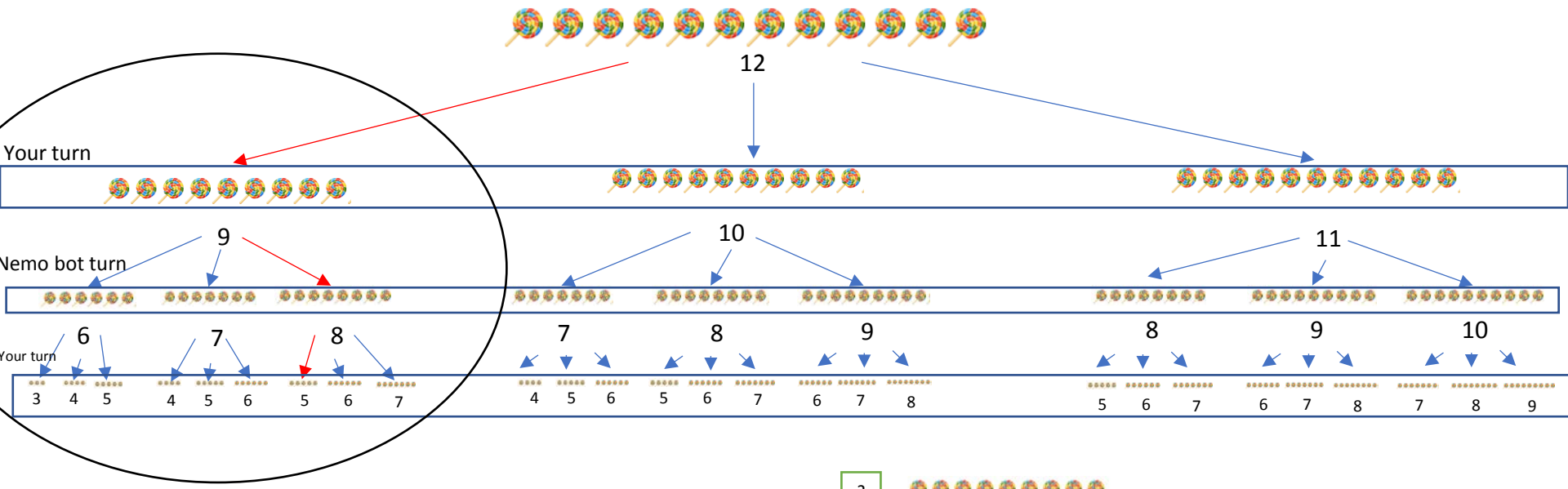
When the game starts, I will show you 12 lollipops, where the last one of them is fish-flavored. It tastes so disgusting that nobody wants to eat it.

The lollipops will be placed in one line, and you and I will take turns to take lollipops from the row. You can't take more than 3 lollipops at a time, and you can't skip your turn. Whoever takes the last lollipop (the fish-flavored one) lose the game.

Example 3: Fish-Flavored Lollipops



Example 3: Fish-Flavored Lollipops

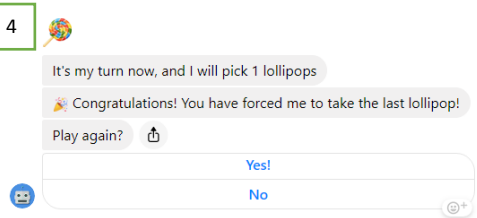


1

2

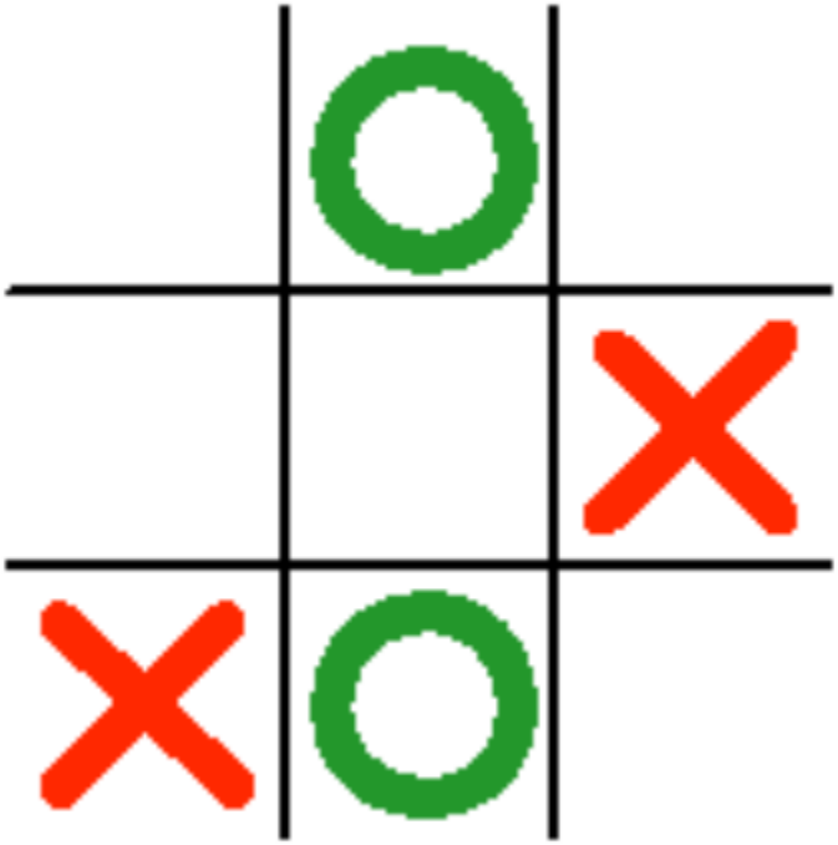
3

4



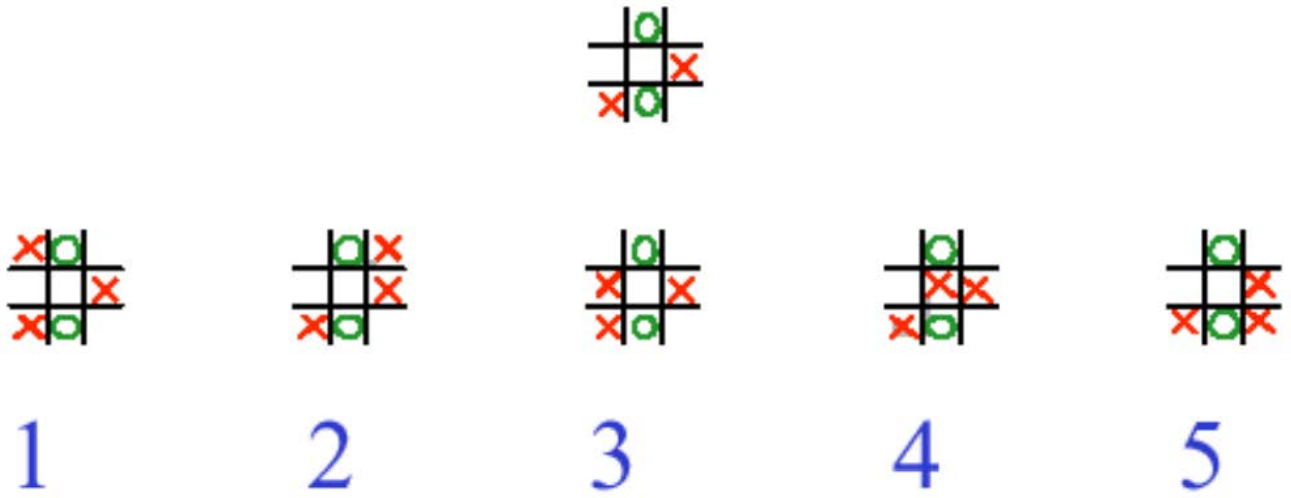
Example 4: Tic-Tae-Toe

- We are playing X, and it is now our turn.



Example 4: Tic-Tae-Toe

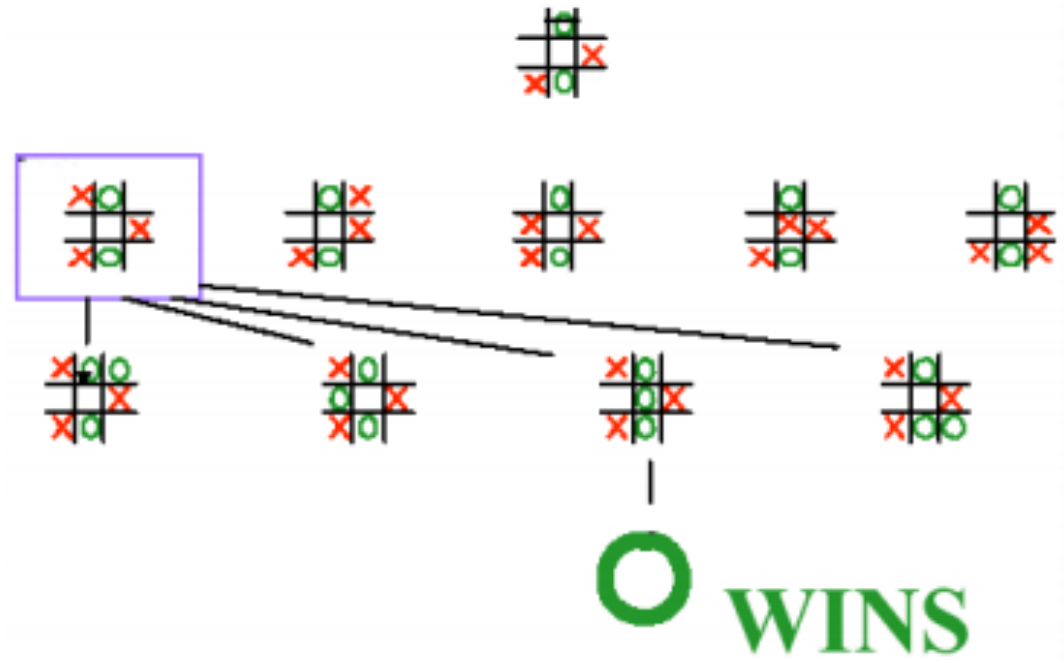
- Let's write out all possibilities



Each number represents a position after each legal move we have.

Example 4: Tic-Tae-Toe

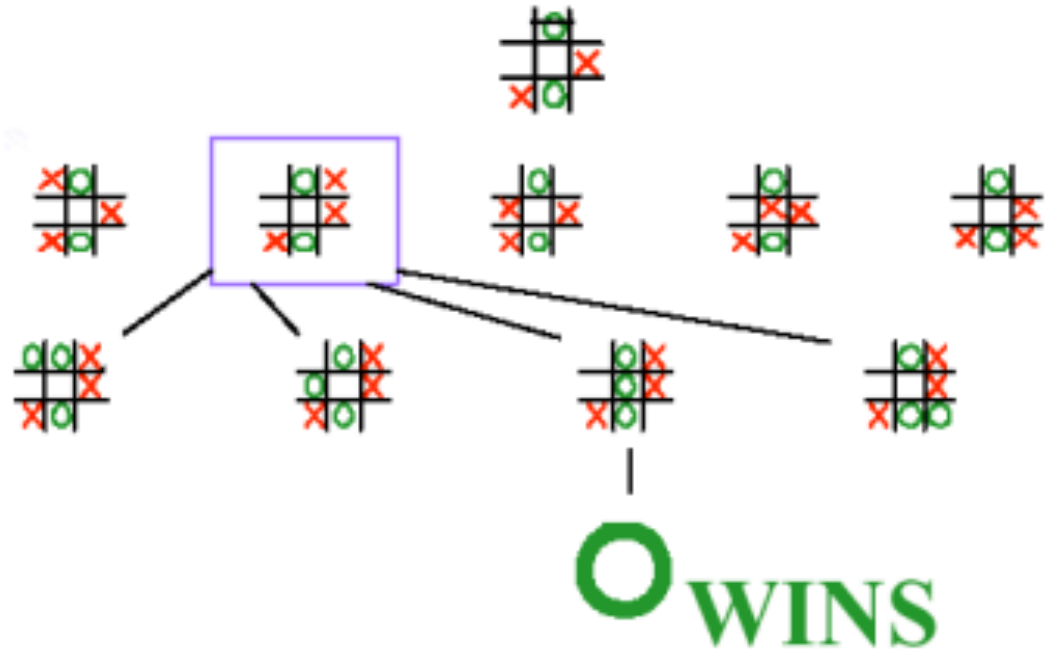
- Now let's look at the opponent's options



Here we are enumerating all the opponent's responses to the first possible move we could make.

Example 4: Tic-Tae-Toe

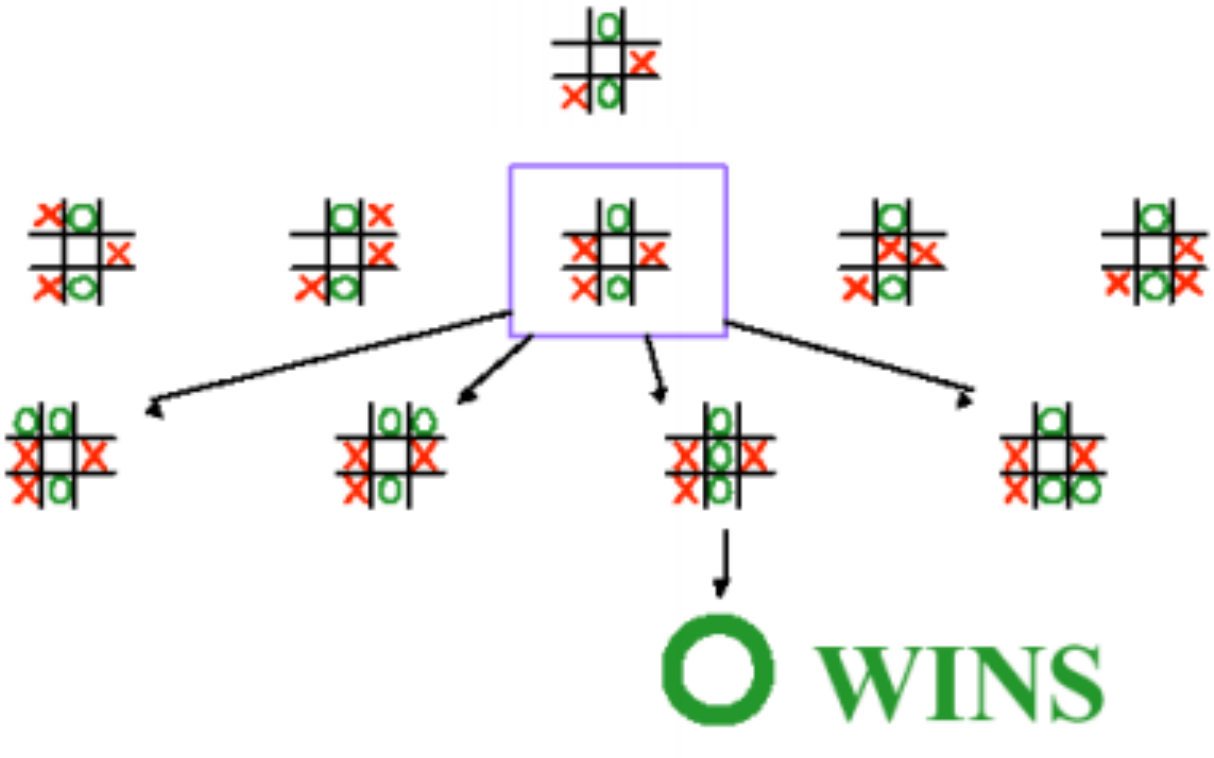
- Now let's look at the opponent's options



Opponent options after our second possibility. Not good again…

Example 4: Tic-Tae-Toe

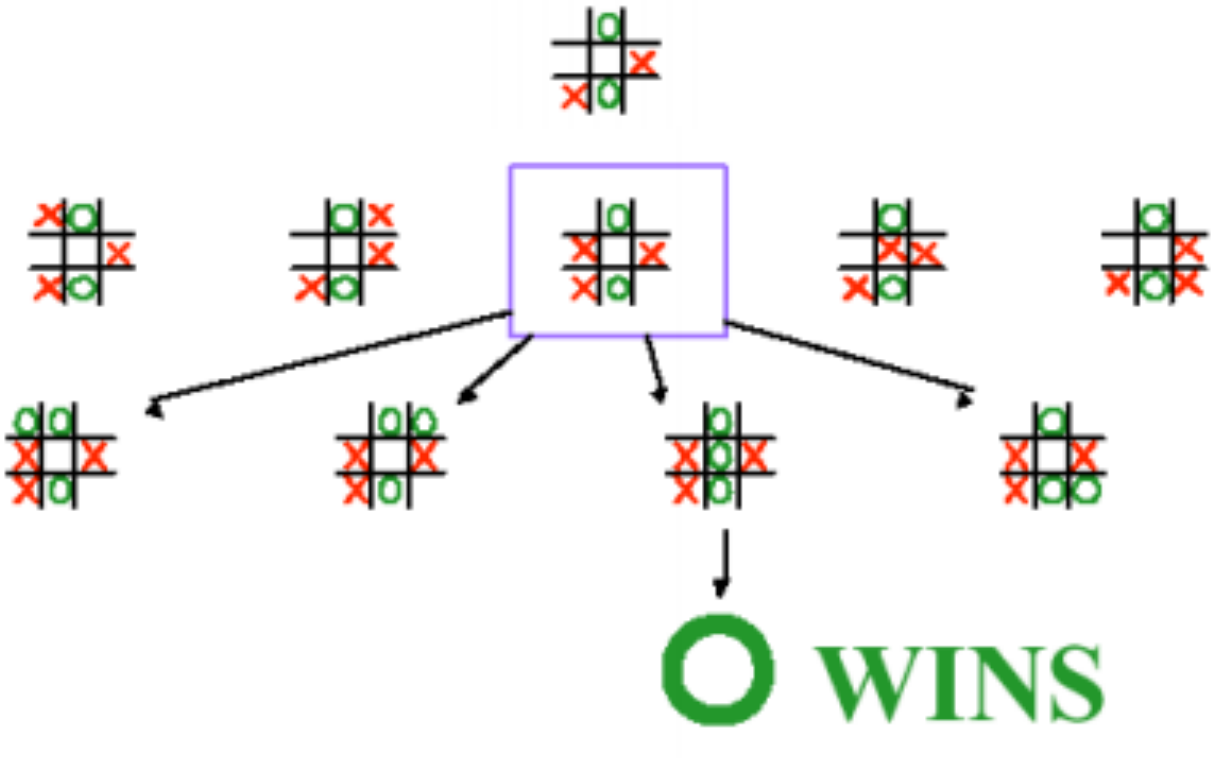
- Now let's look at the opponent's options



Struggling…

Example 4: Tic-Tae-Toe

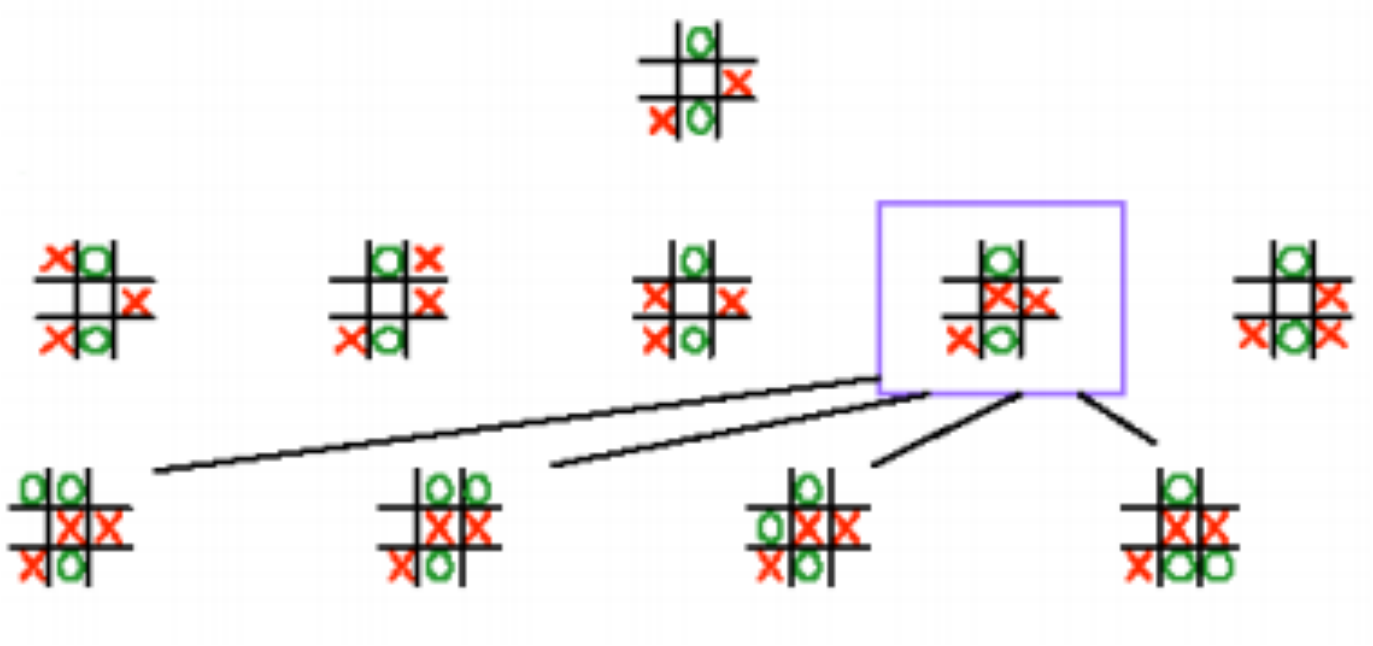
- Now let's look at the opponent's options



Struggling…

Example 4: Tic-Tae-Toe

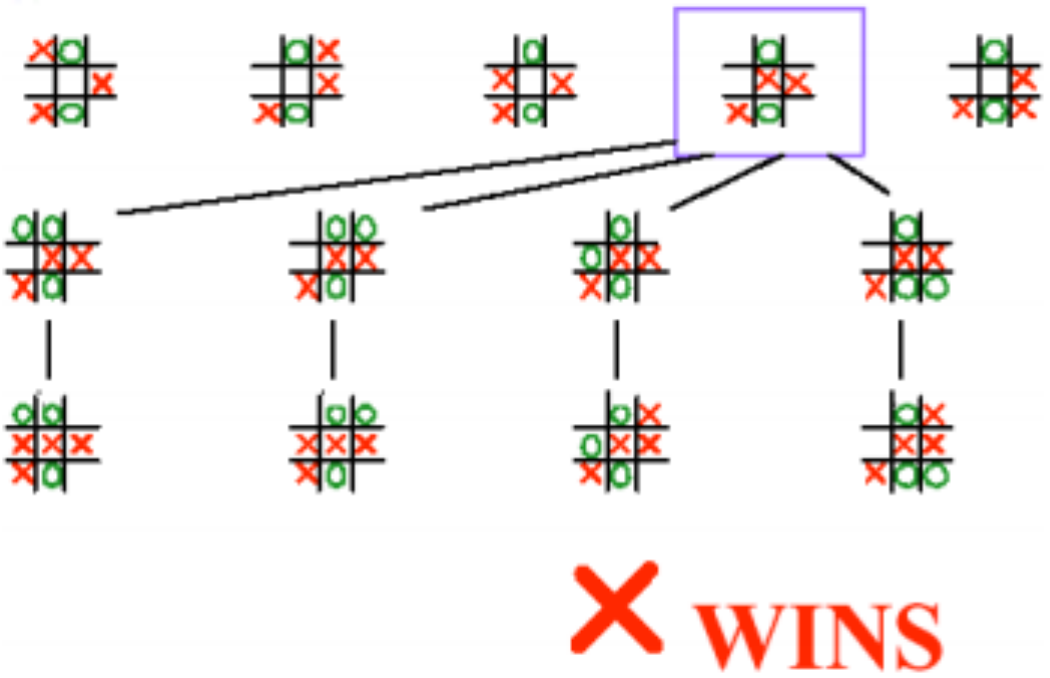
- Inspecting the next level



As the opponent does not win on the next move, we go deeper into the next level of the game tree to consider our moves in response to the opponent's move.

Example 4: Tic-Tae-Toe

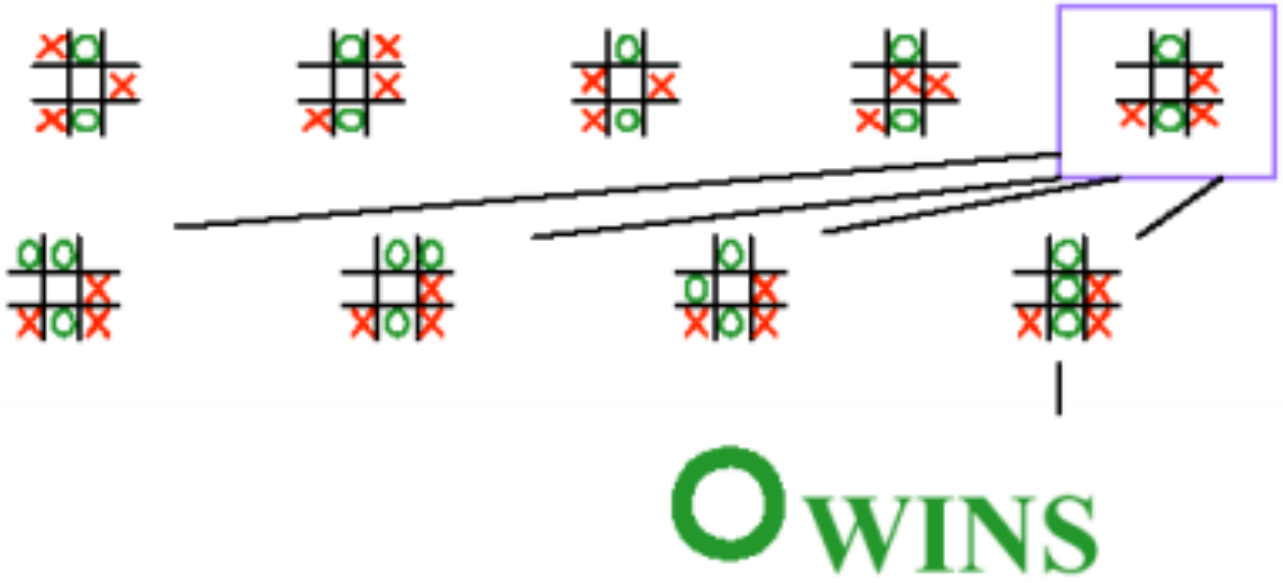
- Searching deeper into the game tree (depth first)



Since we have a win for any move the opponent makes, the position in purple box is an X win.

Example 4: Tic-Tae-Toe

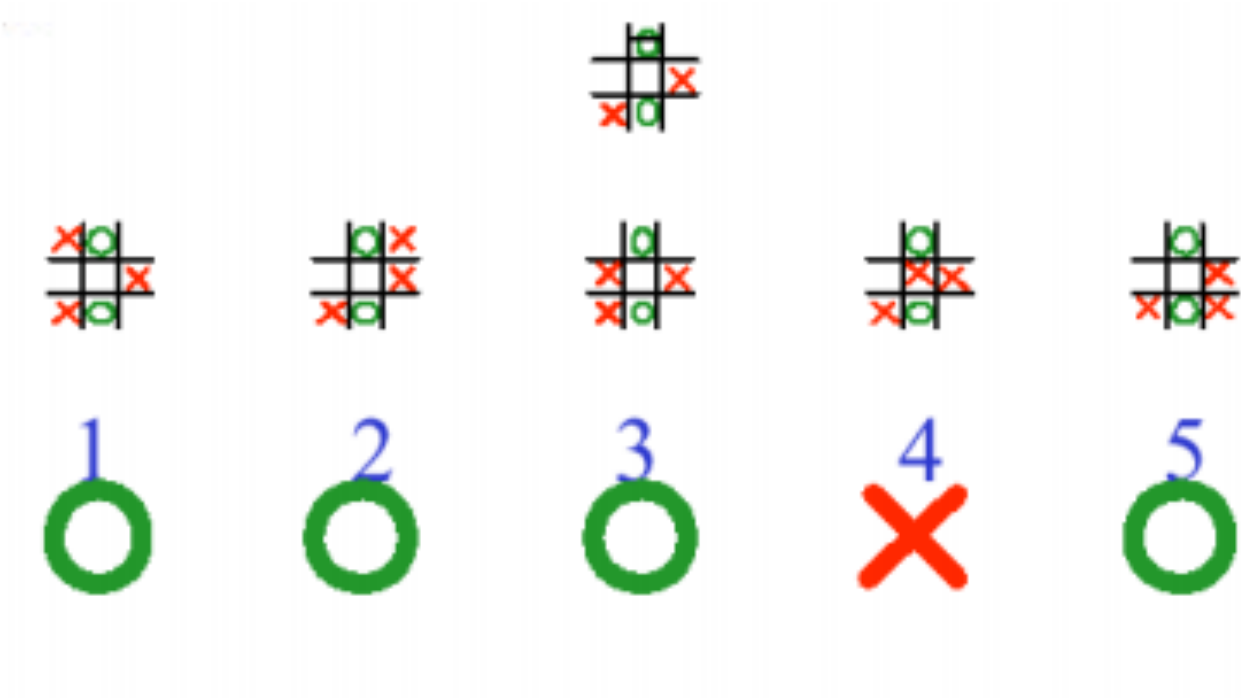
- Inspecting other branches of the game tree



The opponent wins if we take our fifth option to move.

Example 4: Tic-Tae-Toe

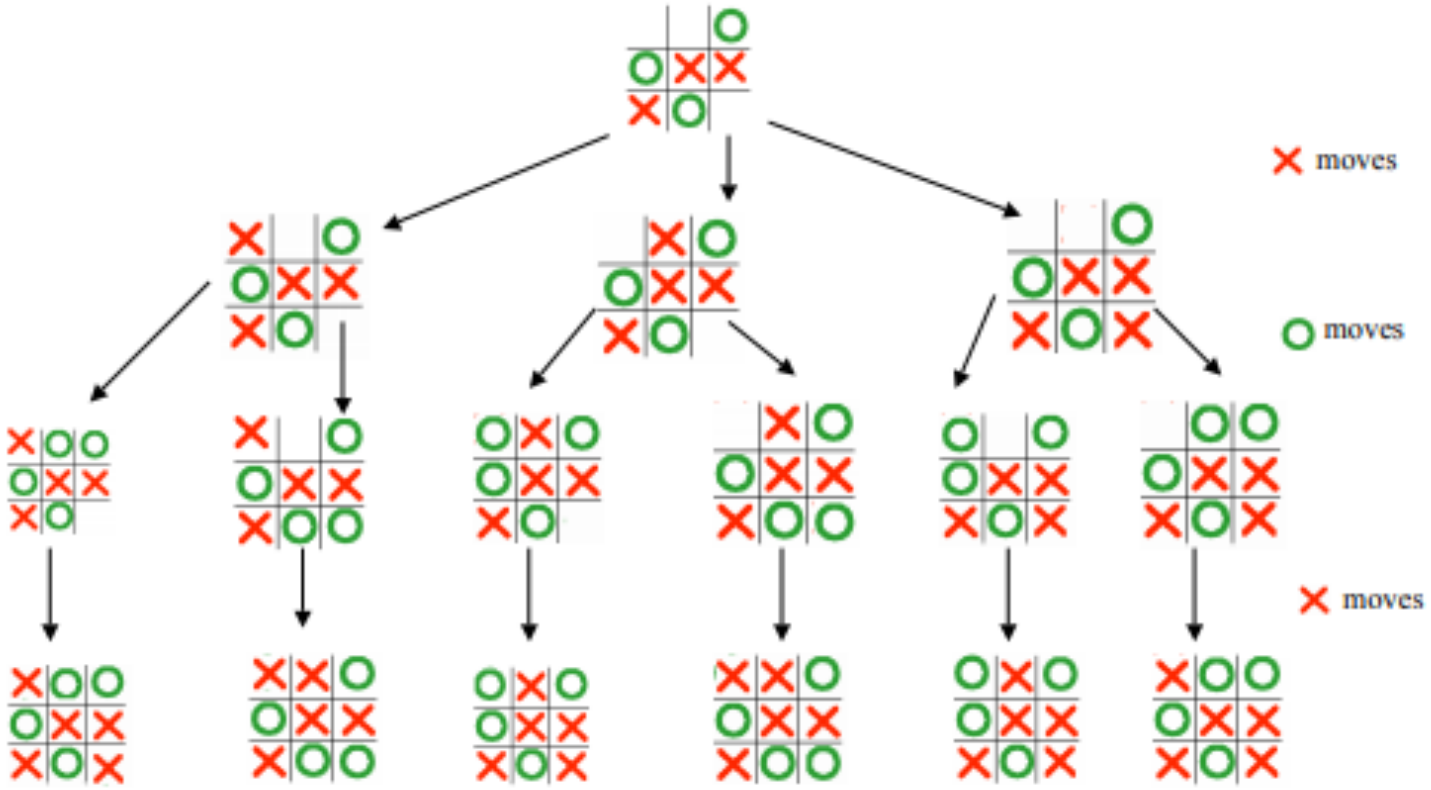
- Summary of the game tree analysis



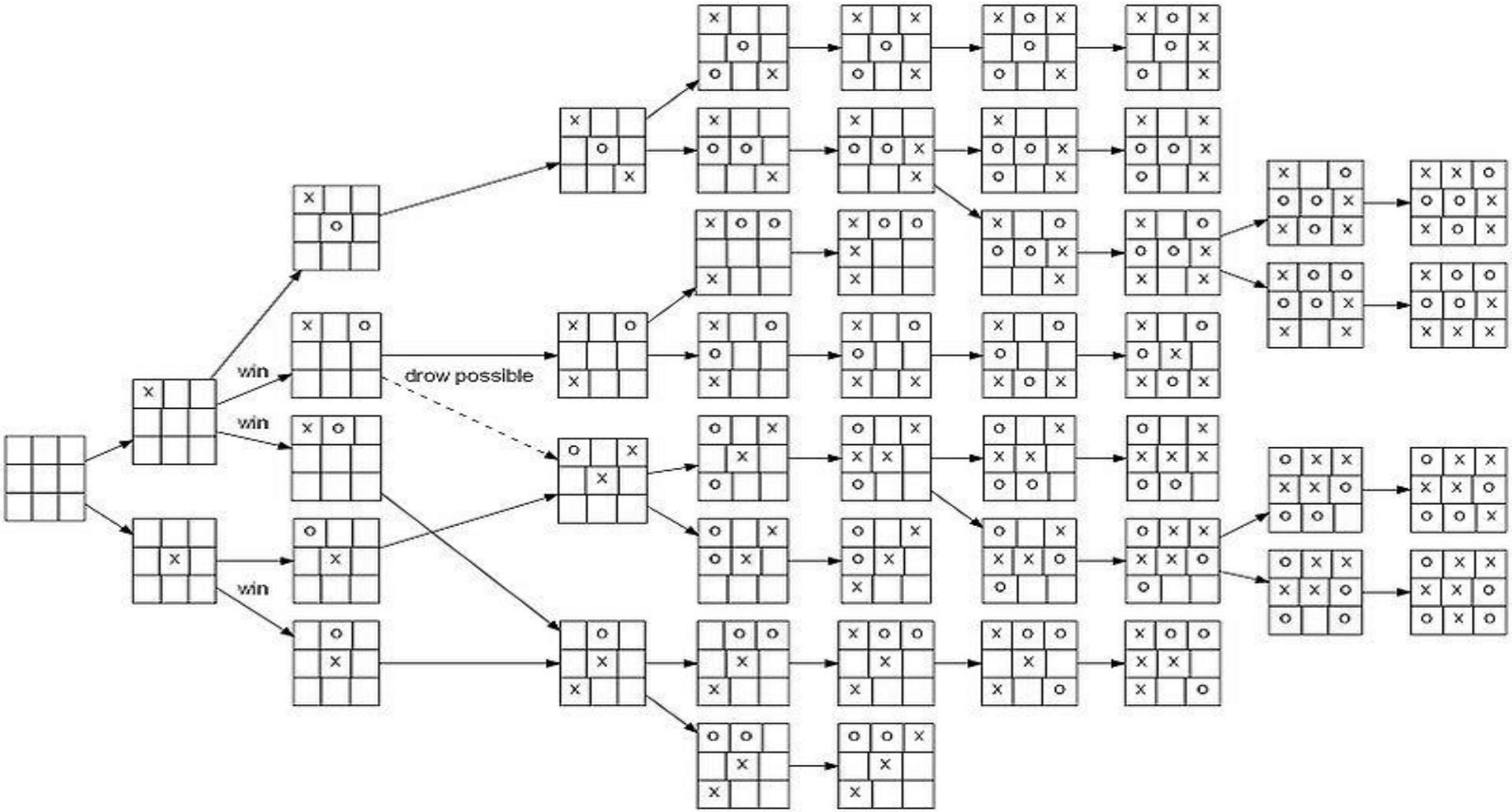
So what move do we make?

Example 4: Tic-Tae-Toe

- Overview of the game tree



Tic-Tac-Toe Game Tree



The game tree starts with the initial position and consecutive nodes contain all possible distinct ways that the game can be played from each position. Overall a complete game tree for Tic-Tac-Toe has 255,168 leaf nodes.

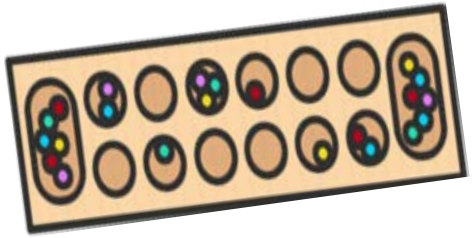
Mancala (Awari) Game and AI

Invented between the 2nd and 3rd centuries, the oldest known mancala relics existed in a Roman bathhouse in Gedera, Israel. Upon its creation, mancala began to travel quickly throughout the world. It first arrived in Africa, and then eventually made its way up to Europe. From there, it spread to North America, South America, Asia, and Australia. Consisting of a board with 12 pits as well as a collection of stones, it can be played in a variety of different ways.

https://www.timesofisrael.com/spa-and-game-room-with-mancala-found-at-ancient-pottery-workshop-in-israel/

Mancala (Awari) Game

1. Two-player turn-based strategy board games



2. Played with small stones, beans, or seeds and rows of holes or pits in the earth, a board or other playing surface.

3. The objective is usually to capture all or some set of the opponent's pieces.

https://en.wikipedia.org/wiki/Mancala

Mancala (Awari) Game Rules

1. The game begins with one player picking up all of the pieces in any one of the pockets on his/her side.

2. Moving counter-clockwise, the player deposits one of the stones in each pocket until the stones run out.

3. If you run into your own Mancala (store), deposit one piece in it. If you run into your opponent's Mancala, skip it and continue moving to the next pocket.

4. If the last piece you drop is in your own Mancala, you take another turn.

5. If the last piece you drop is in an empty pocket on your side, you capture that piece and any pieces in the pocket directly opposite.

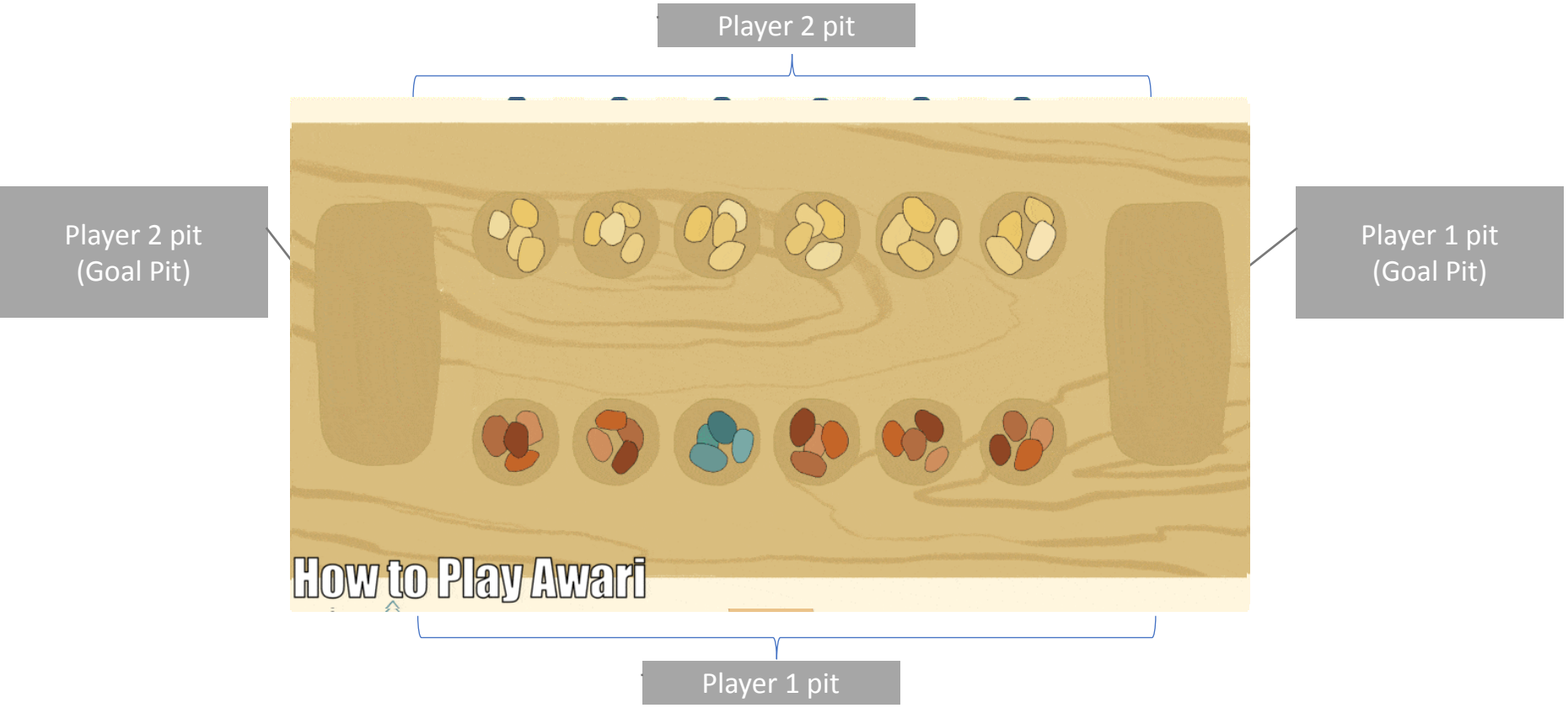
6. Always place all captured pieces in your Mancala (store).

7. The game ends when all six pockets on one side of the Mancala board are empty.

8. The player who still has pieces on his/her side of the board when the game ends captures all of those pieces.

9. Count all the pieces in each Mancala. The winner is the player with the most pieces.

Mancala(Awari) Game Board



Mancala (Awari) Game in NemoBot

1. The move option numbered 0 to 5 means choose the pit at bottom row from left(0) to right(5) 2. The move option numbered 7 to 12 means choose the pit at top row from right(7) to left(12).

Top board

. ⑫ ⑪ ⑩ ⑨ ⑧ ⑦ .

⑬ ⑥

Bottom board

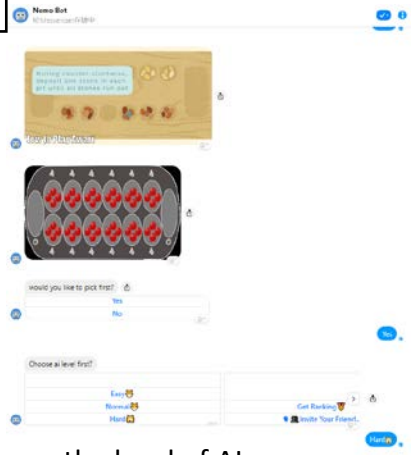
. ⓪ ① ② ③ ④ ⑤ .

2. Position ⑬ is the mancala for the top board, while position ⑥ is the mancala for the bottom board player.

3. If you choose to move first, you are a bottom row player; if AI moves first, you are a top row player.

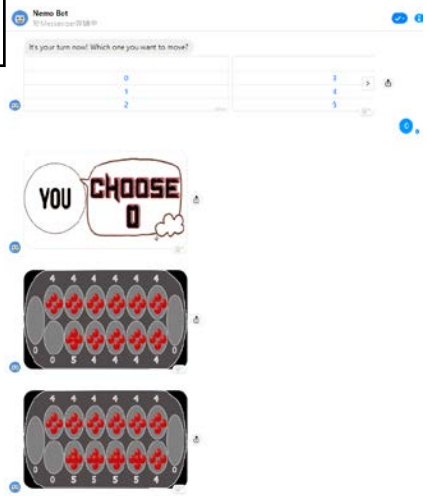
Example

1.



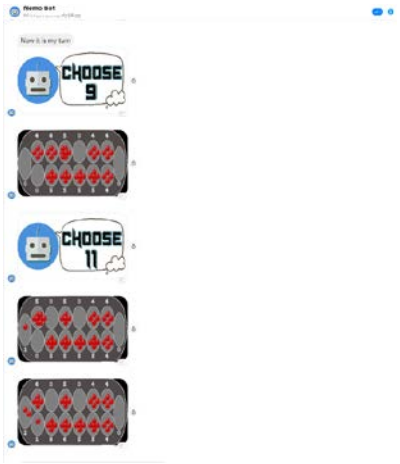
Choose the level of AI

2.



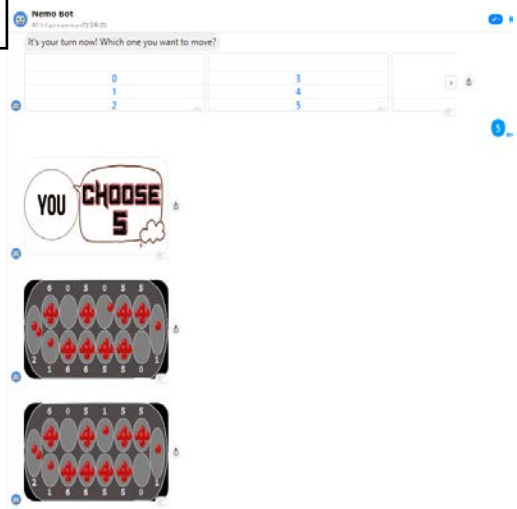
Nemobot : 0, Player : 0,
player choose position 0

3.



Nemobot : 2, Player : 0,
Nemobot choose position 9 , 11

4.

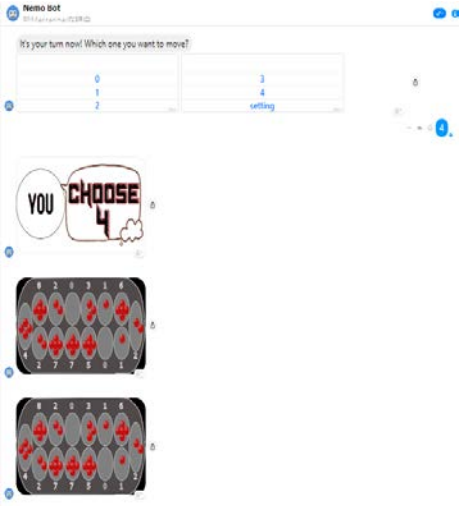


Nemobot : 2, Player : 1,
player choose position 5

5.

Nemobot : 4, Player : 1,
Nemobot choose position 8 , 10

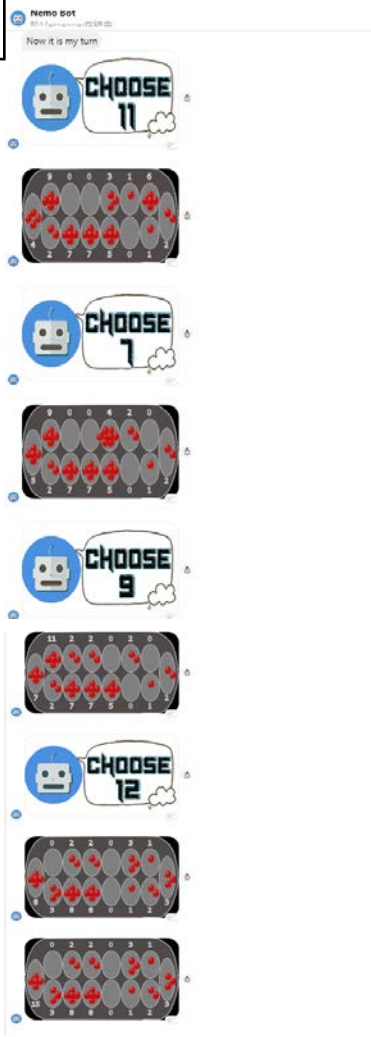
6.



Nemobot : 4, Player : 2,
player choose position 4

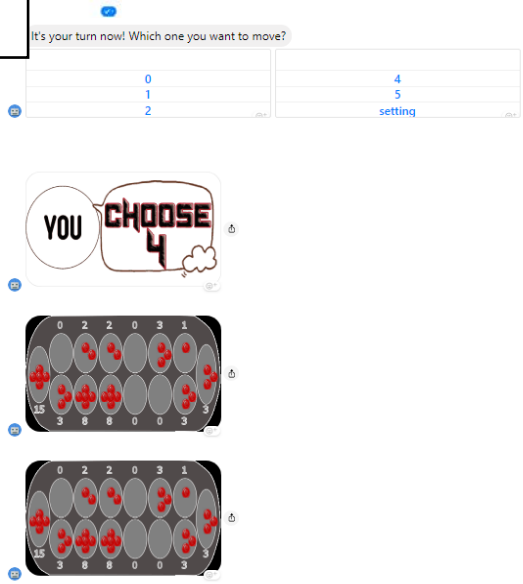
Example

7.



Nemobot : 15, Player : 3,
Nemobot choose position 11,7,9,12

8.



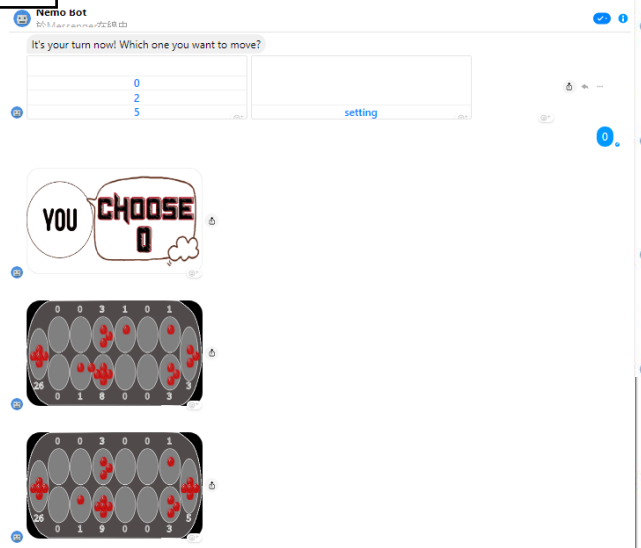
Nemobot : 15, Player : 3,
player choose position 4

9.

Nemobot : 26, Player : 3,
Nemobot choose position 11,12,8

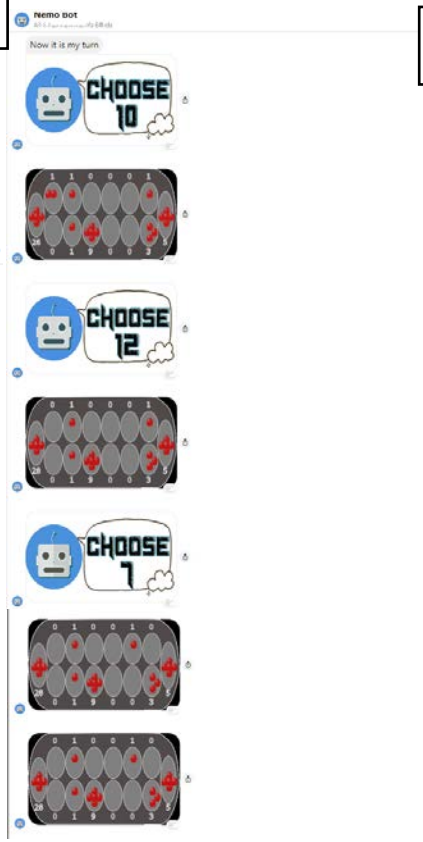
Example

10.



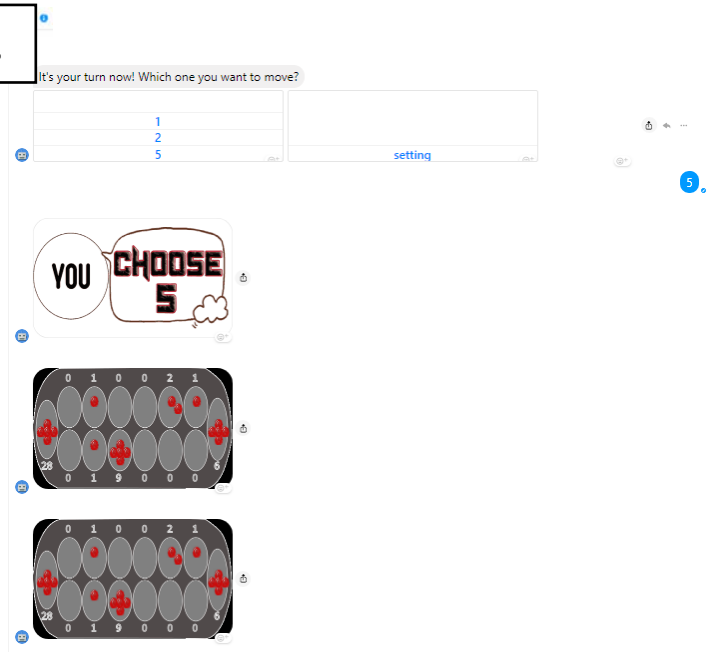
Nemobot : 26, Player : 5,
Player choose position 0

11.



Nemobot : 28, Player : 5,
Player choose position 10,12,7

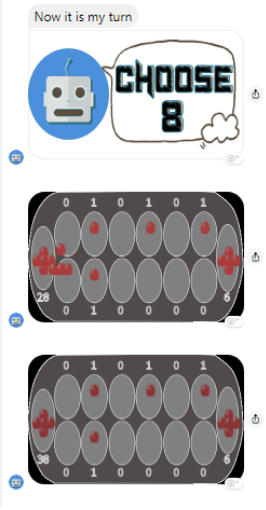
12.



Nemobot : 28, Player : 6,
Player choose position 0

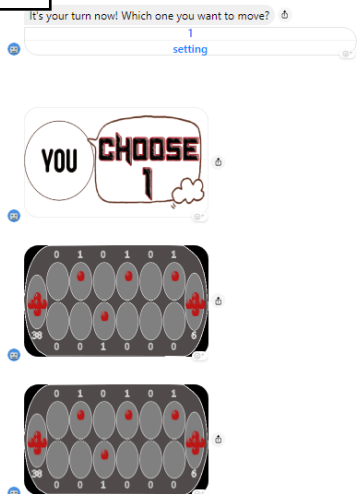
Example

13.



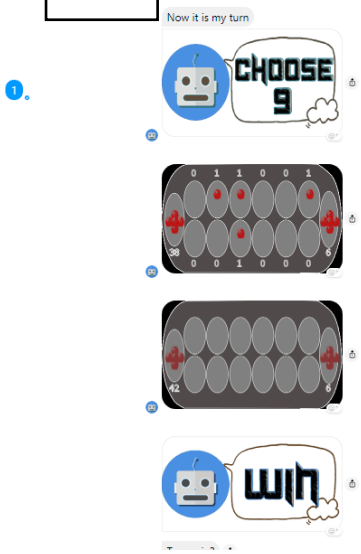
Nemobot : 38, Player : 6,
Nemobot choose position 8

14.



Nemobot : 38, Player : 6,
Player choose position 1

15.



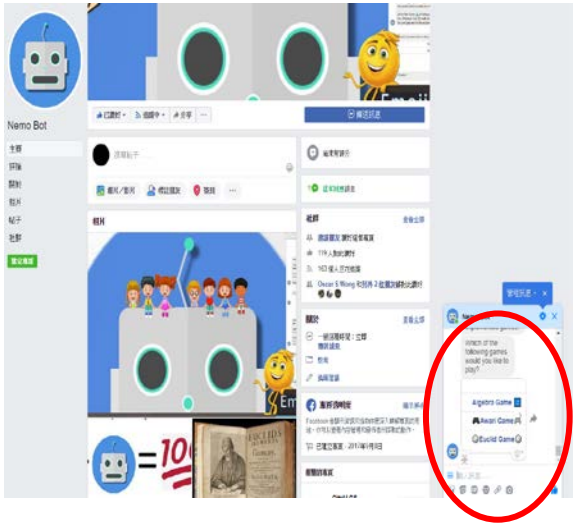
Nemobot : 42, Player : 9,
Nemobot choose position 9 and get player stone, player side has no more stone and so game finishes. Nemobot wins the game

Play in Nemobot

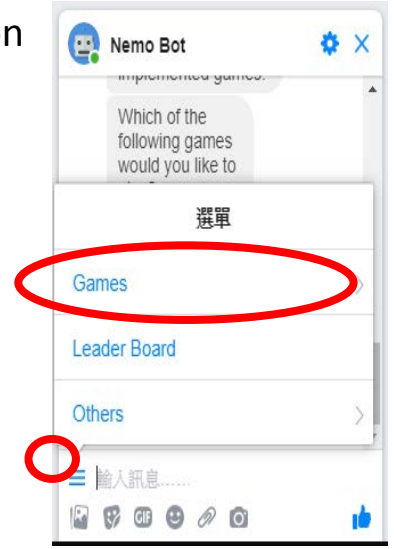
Scan QR Code

https://www.facebook.com/Nemo-Bot-454163798317367/

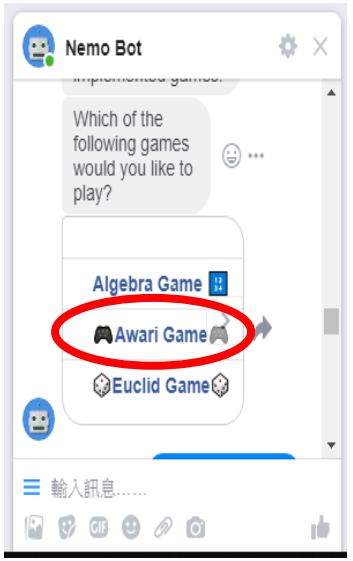
Talk to Nemo Bot on Messenger



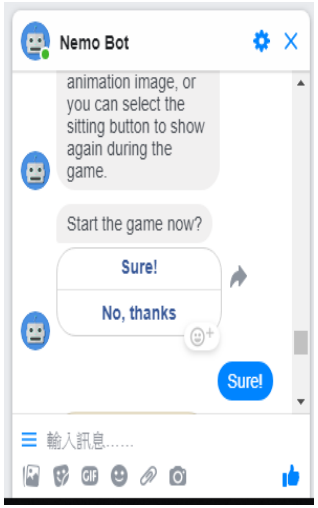
Open menu and click “Official Games” button



Click “Awari Game - Mancala” button

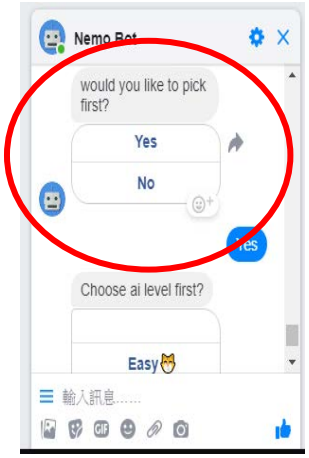


Click “Sure” button

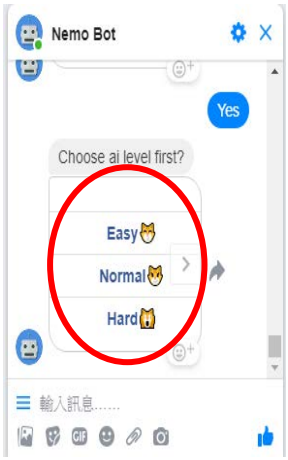


Play in Nemobot

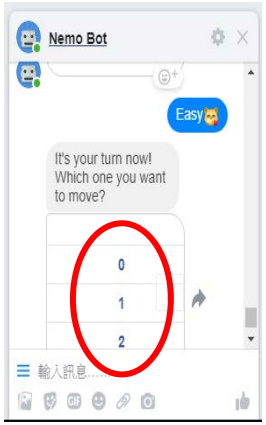
Choose who goes first



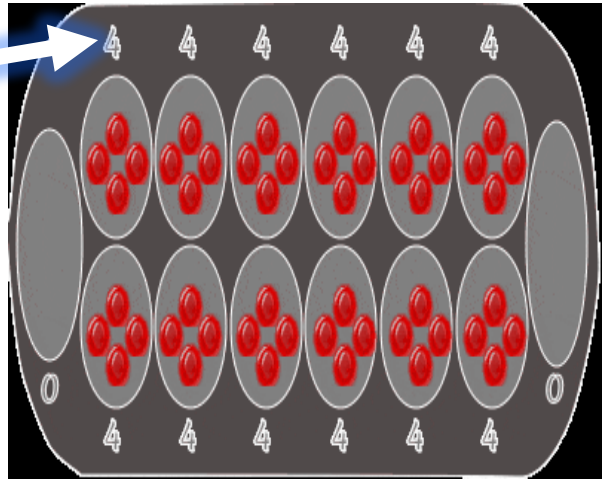
Choose AI level

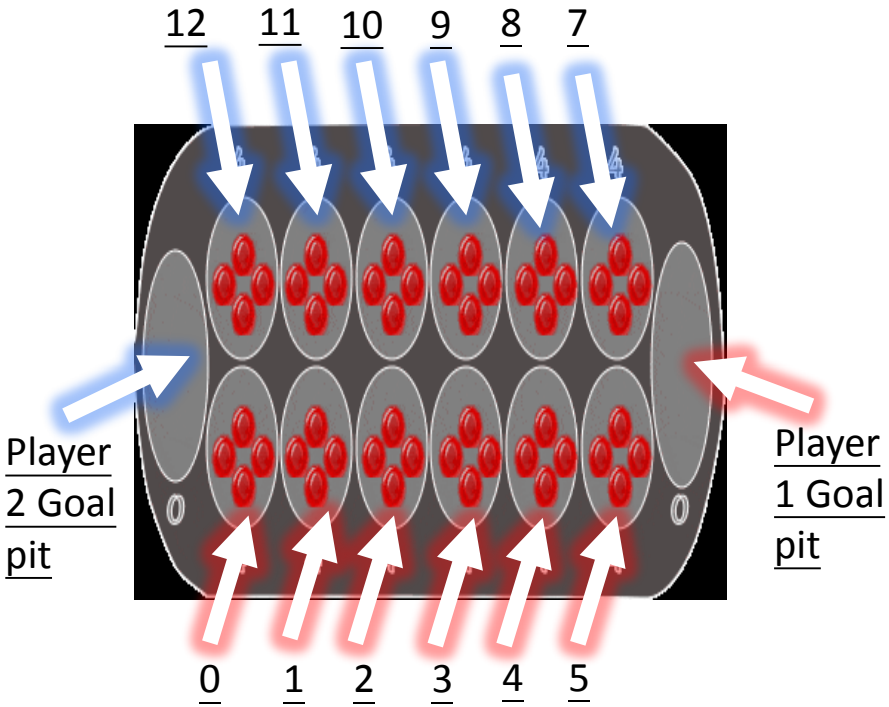


Choose which pit you want to move



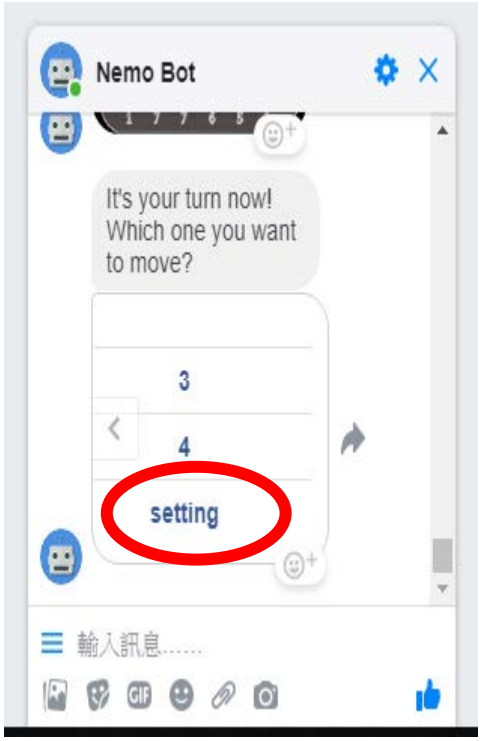
Number of stones in pit



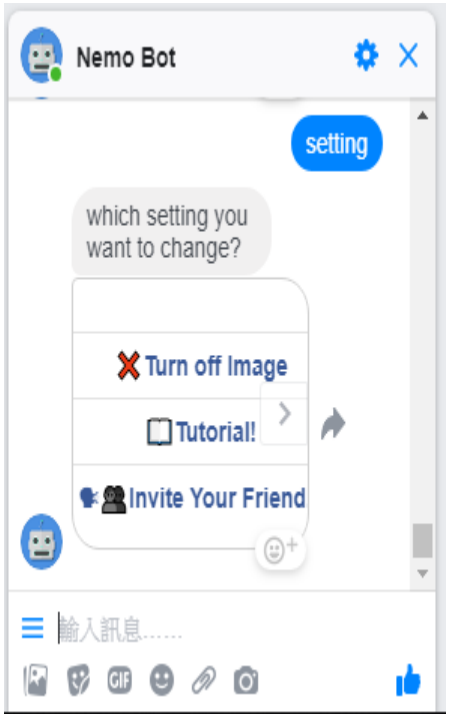


Play in Nemobot

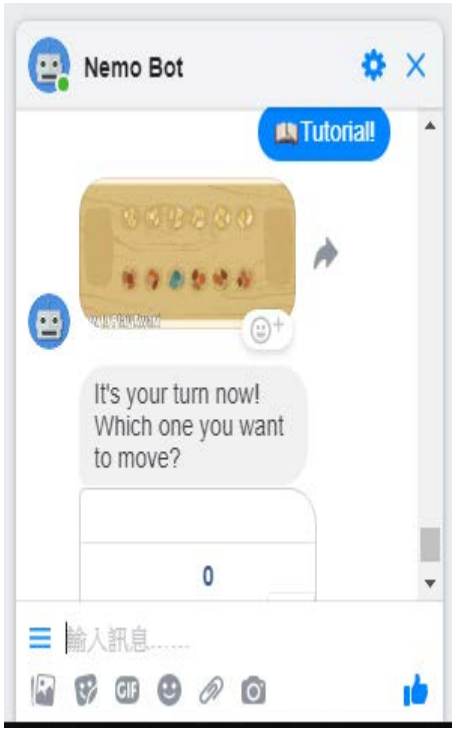
Choose “setting” located as last option to configure game anytime



Choose “Tutorial!” in setting menu

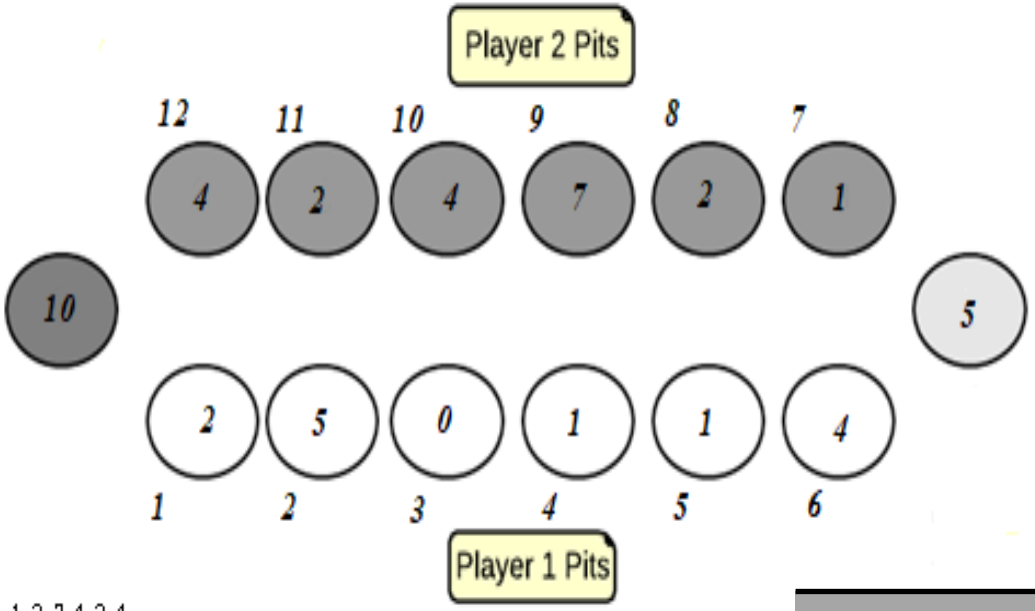


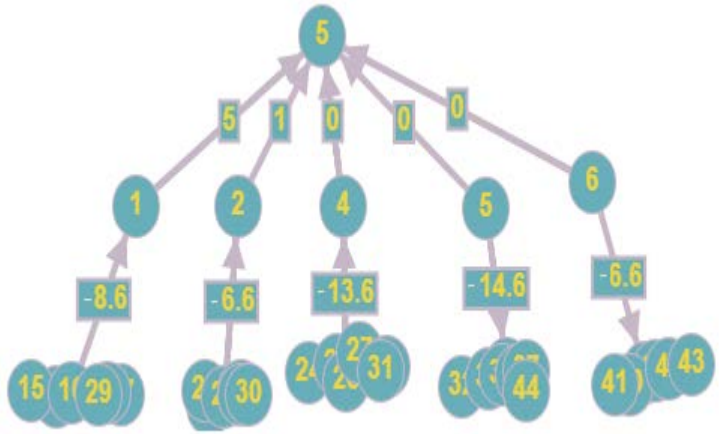
Show the tutorial animation



Awari GameGame AI in Nemobot

Minimax Algorithm

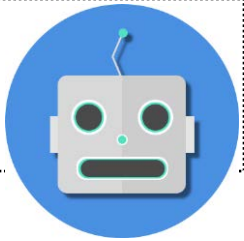




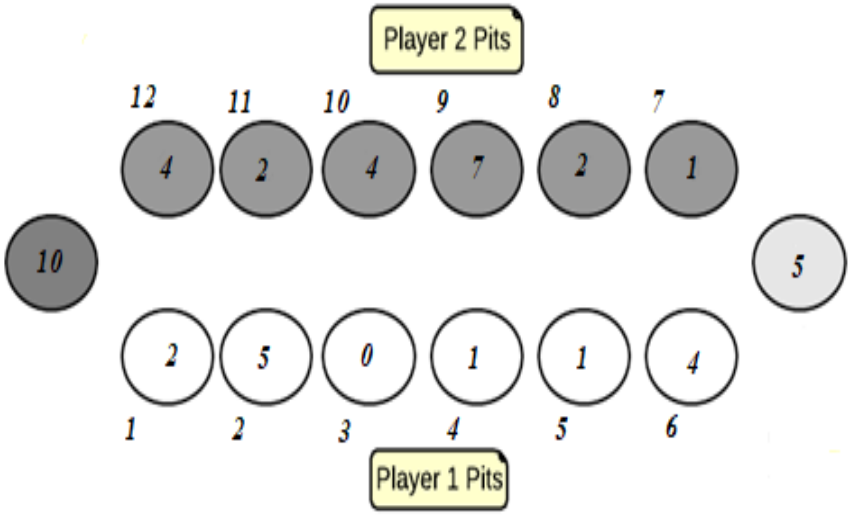

```
  1 2 7 4 2 4
10              5
  2 5 0 1 1 4
Top player's turn? true
NEW xxxMOVE 5 -6.6
NEW xxxMOVE 4 -14.6
NEW xxxMOVE 3 -13.6
NEW xxxMOVE 1 -6.6
NEW xxxMOVE 0 -8.6
best move 5 -6.6
```

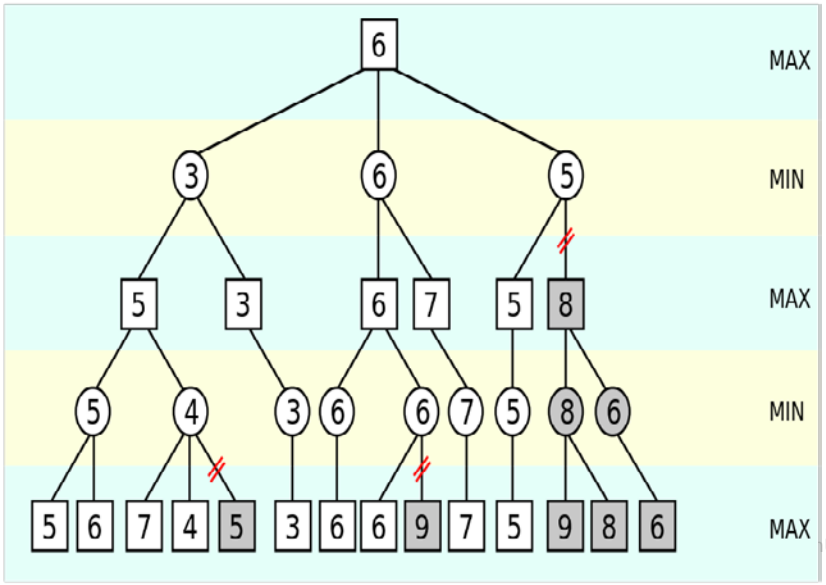
Available select	
1	➔ number of stones in the player's holding + 5
2	➔ number of stones in the player's holding +1, extra round
3	Not available
4	➔ 5
5	➔ 6

Awari GameGame AI in Nemobot

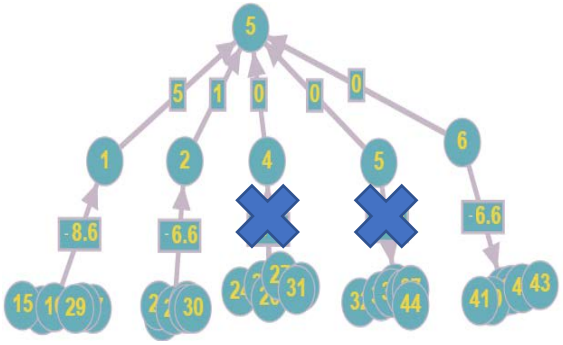


Alpha-Beta Pruning Minimax





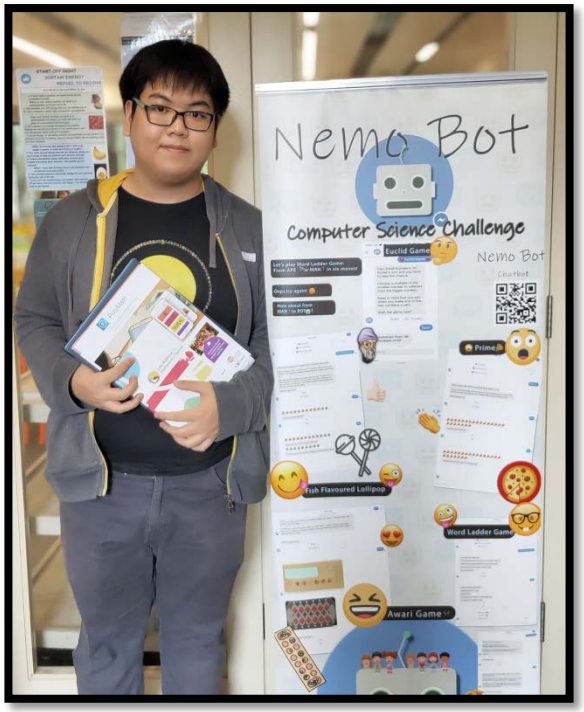

```
  1 2 7 4 2 4
10            5
  2 5 0 1 1 4
Top player's turn? true
NEW xxxMOVE 5 -6.6
NEW xxxMOVE 4 -7.4
NEW xxxMOVE 3 -7.4
NEW xxxMOVE 1 -6.6
NEW xxxMOVE 0 -7.6
best move 5 -6.6
```



Available	
select	
1	➔ number of stones in the player's holding + 5
2	➔ number of stones in the player's holding +1, extra round
3	Not available
4	➔ 5

Awari GameGame AI in Nemobot

Case 1		
Greedy Algorithm	Minimax Algorithm	Alpha Beta Pruning Minimax
Execution time: 0.840416 NEW xxxMOVE 5 -0.4 NEW xxxMOVE 4 -0.4 NEW xxxMOVE 3 -0.4 NEW xxxMOVE 2 0.4 NEW xxxMOVE 1 -1.4 NEW xxxMOVE 0 -1.4 best move 2 0.4 Nemo choose 9 . ⑤ ⑤ ⑤ ⓪ ④ ④ . ① ① . ⓪ ⑤ ① ⑥ ⑥ ⑤ . Execution time: 0.35646 NEW xxxMOVE 5 0.6 NEW xxxMOVE 4 0.6 NEW xxxMOVE 3 0.6 NEW xxxMOVE 1 -0.4 NEW xxxMOVE 0 -0.4 best move 5 0.6 Nemo choose 12 . ⓪ ⑤ ⑤ ⓪ ④ ④ . ② ① . ① ⑥ ② ⑦ ⑥ ⑤ .	Execution time: 14.606176877 NEW xxxMOVE 5 3.4 NEW xxxMOVE 4 4.4 NEW xxxMOVE 3 4.4 NEW xxxMOVE 2 6.6 NEW xxxMOVE 1 6.6 NEW xxxMOVE 0 6.6 best move 2 6.6 Nemo choose 9 . ⑤ ⑤ ⑤ ⓪ ④ ④ . ① ① . ⓪ ⑤ ① ⑥ ⑥ ⑤ . Execution time: 18.111569207 NEW xxxMOVE 5 3.4 NEW xxxMOVE 4 5.6 NEW xxxMOVE 3 5.6 NEW xxxMOVE 1 5.6 NEW xxxMOVE 0 5.6 best move 4 5.6 Nemo choose 11 . ⑥ ⓪ ⑤ ⓪ ④ ④ . ② ① . ① ⑥ ② ⑥ ⑥ ⑤ .	Execution time: 1.130538269 NEW xxxMOVE 5 2.6 NEW xxxMOVE 4 3.6 NEW xxxMOVE 3 3.6 NEW xxxMOVE 2 4.6 NEW xxxMOVE 1 4.6 NEW xxxMOVE 0 3.6 best move 2 4.6 Nemo choose 9 . ⑤ ⑤ ⑤ ⓪ ④ ④ . ① ① . ⓪ ⑤ ① ⑥ ⑥ ⑤ . Execution time: 2.-446701222 NEW xxxMOVE 5 2.6 NEW xxxMOVE 4 3.6 NEW xxxMOVE 3 4.4 NEW xxxMOVE 1 4.4 NEW xxxMOVE 0 4.4 best move 3 4.4 Nemo choose 10 . ⑥ ⑥ ⓪ ⓪ ④ ④ . ② ① . ① ⑥ ① ⑥ ⑥ ⑤ .

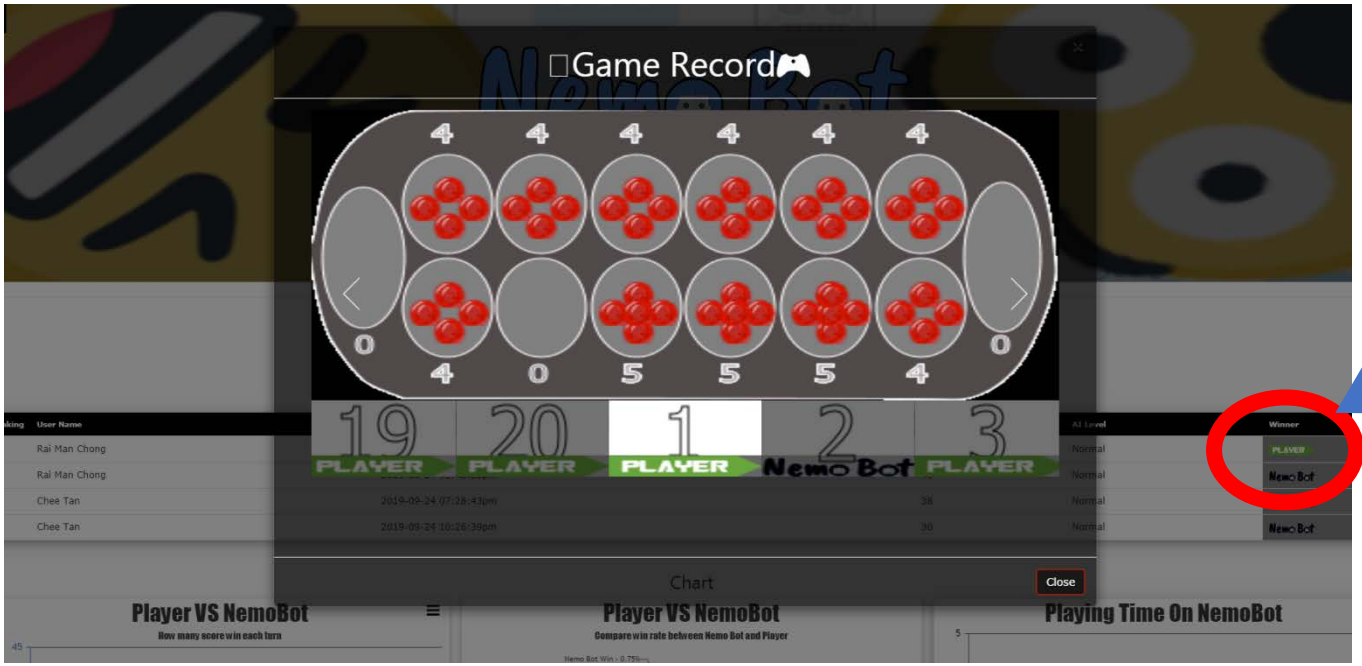


“Twitch”: Play to Learn

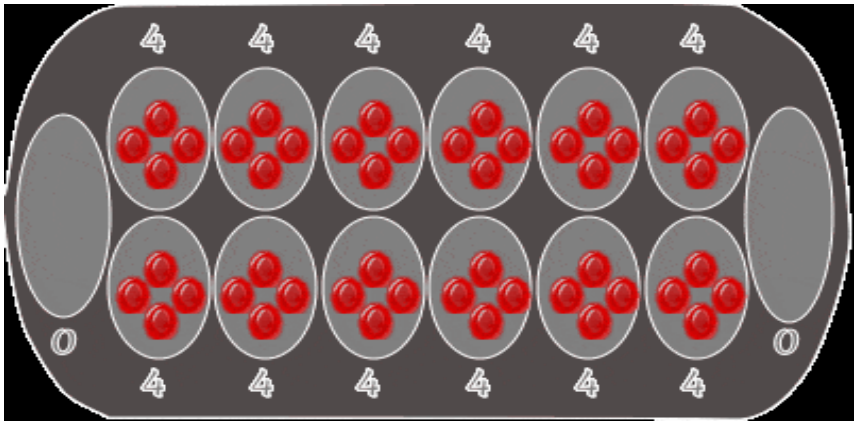
Browse the moves of yours and opponent to learn from errors made and find strategies

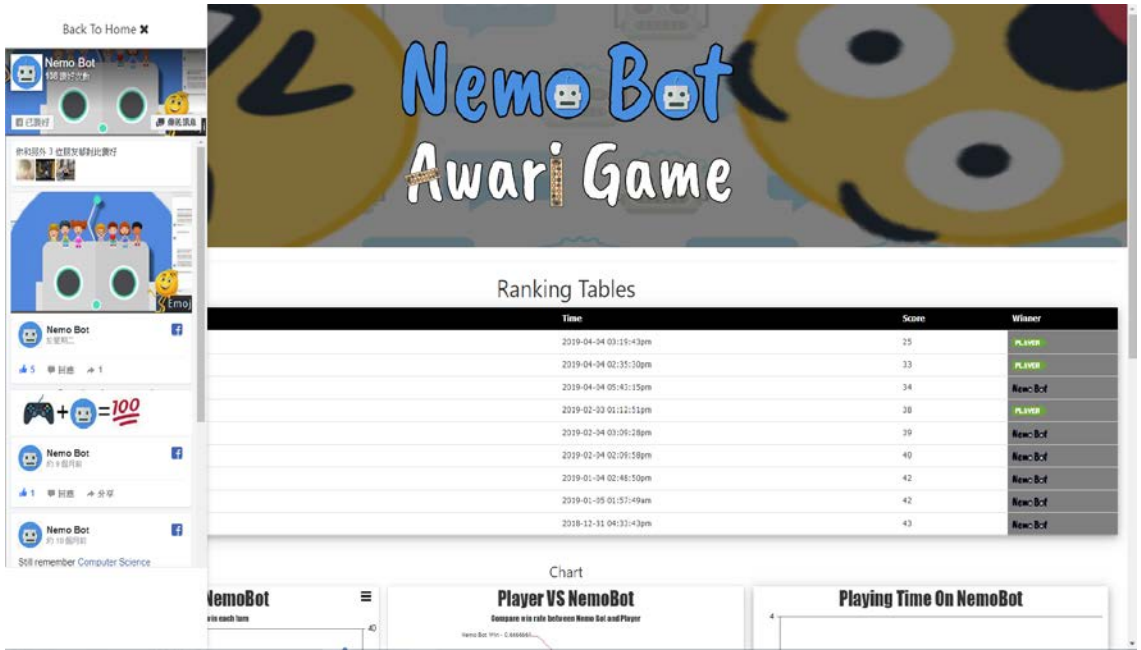
As you play, you traverse the Game Tree and think of a right path

https://awari.algebragame.app/rating



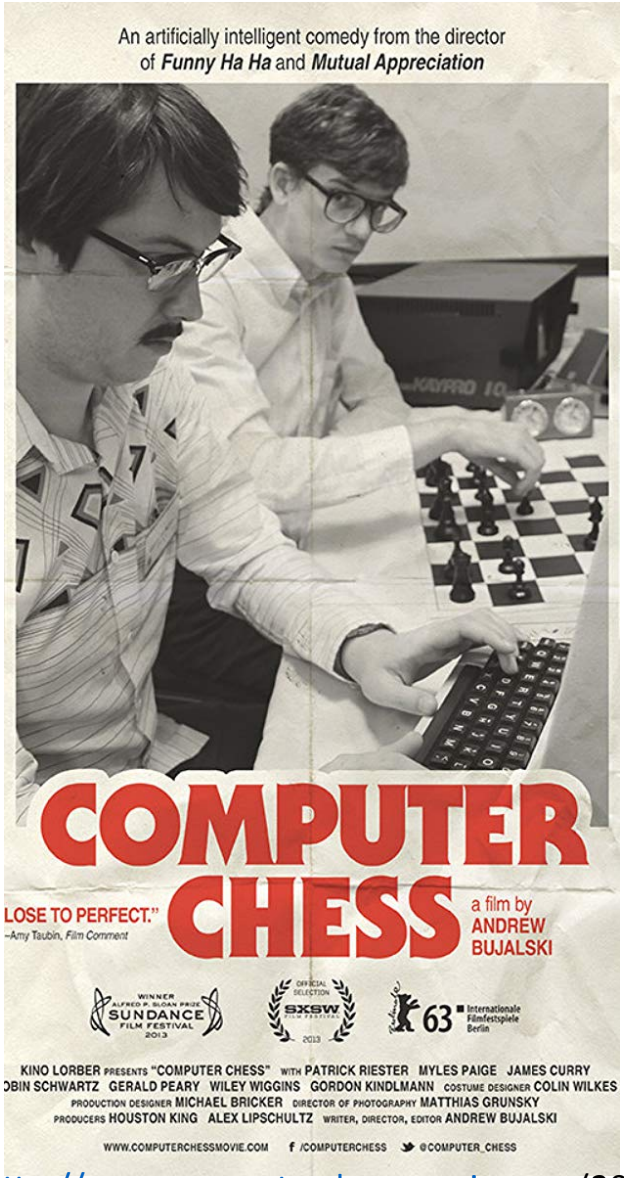
How about a Tournament with AI?





Summary of AI Games

- Games of strategy raise questions about the nature of intelligence and, by virtue of its sharply limited environment and clearly defined goal, serve as the ideal *drosophila* to evaluate approaches to artificial intelligence
- Search in two-player AI games
  - Game Tree
  - Backtracking
  - Trial-and-Error and evaluation of moves
- Curation of AI games (Bridg-It, Mancala, Awari) and their human-computer interface (HCI)
  - Human aspect of an AI game is always important!



http://www.computerchessmovie.com (2013)