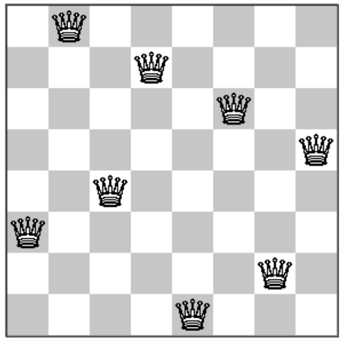
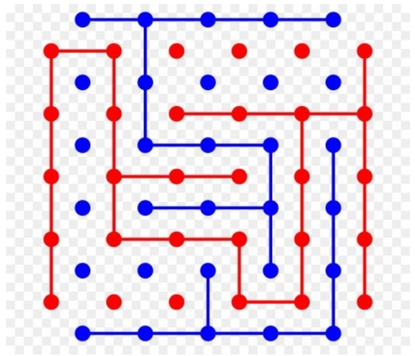
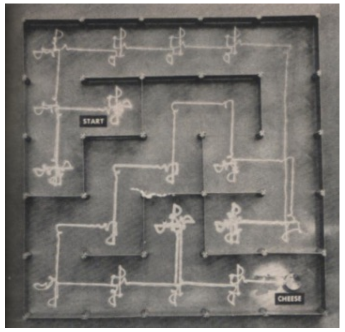
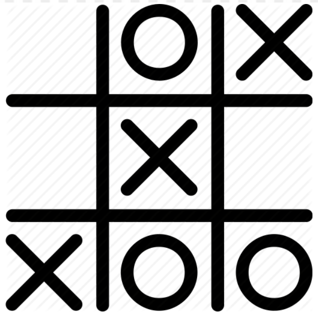
Artificial Intelligence:

Past, Present and Future









TensorFlow

Chee Wei Tan

Bridg-It Game and AI

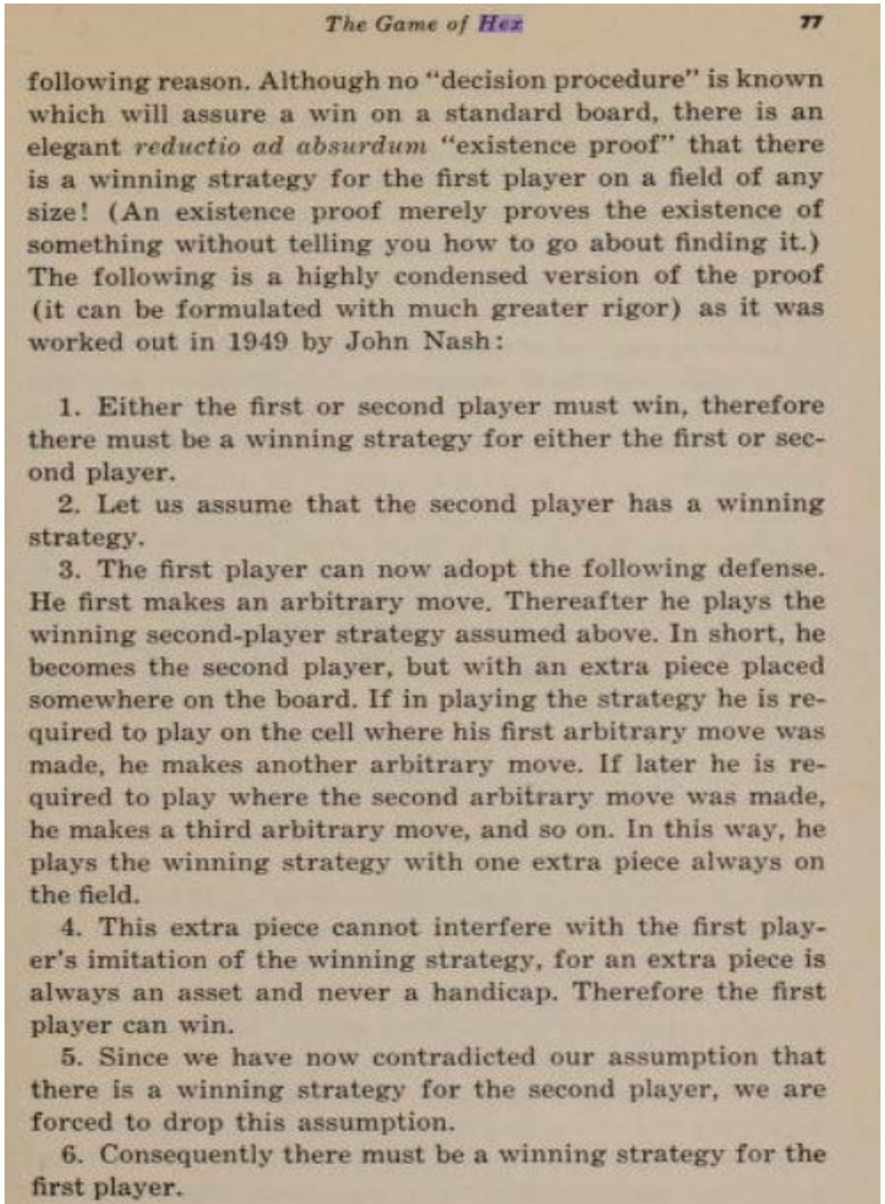
The rules of Bridg-It

- 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.

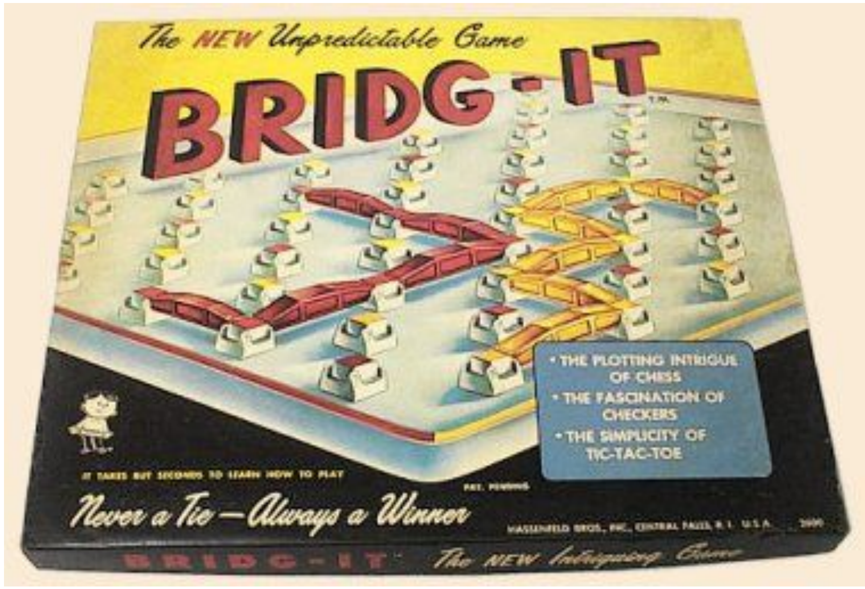
Brídg-It: First-Mover Advantage



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:

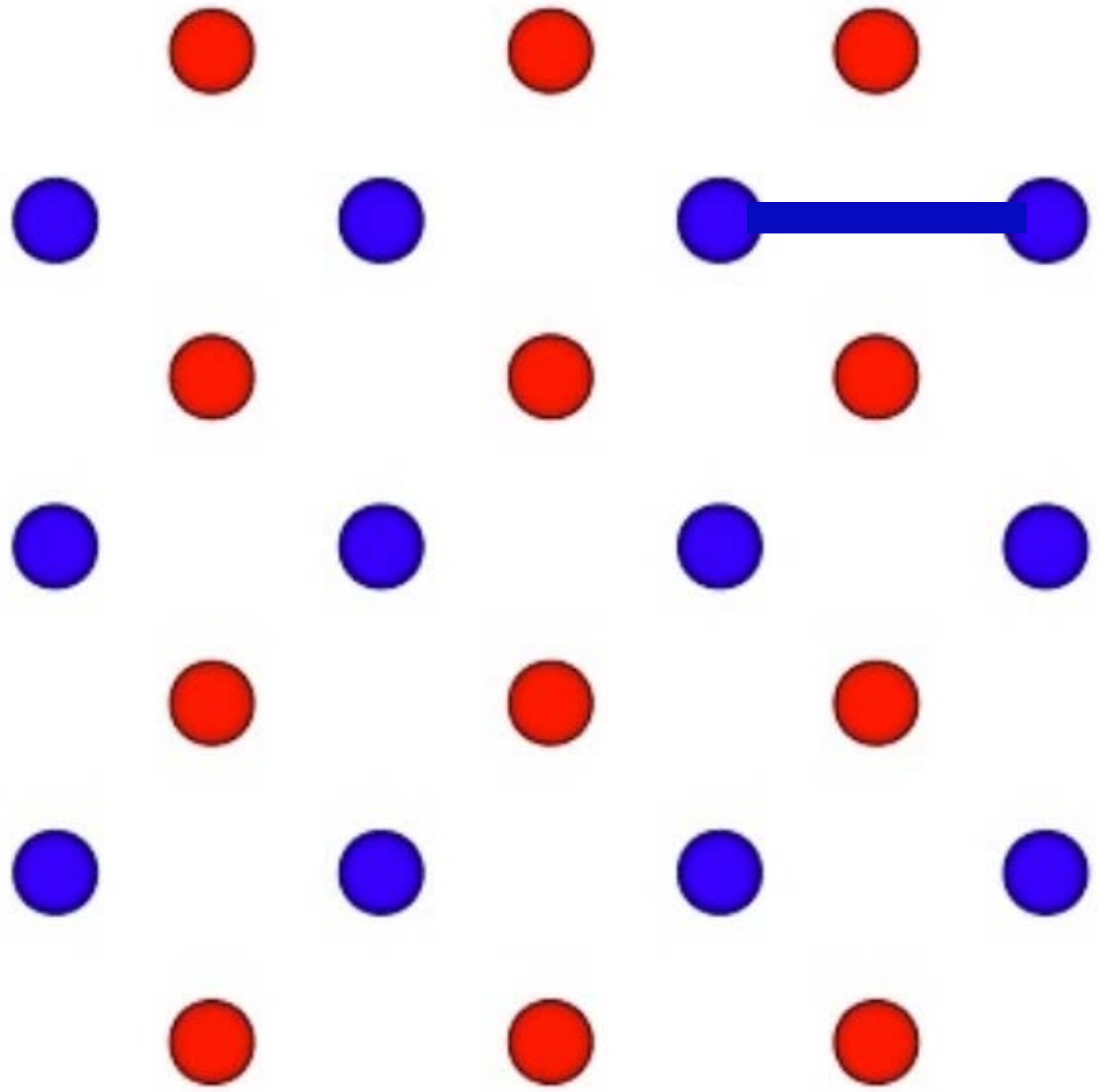
1. Either the first or second player must win, therefore there must be a winning strategy for either the first or second player.
2. Let us assume that the second player has a winning strategy.
3. The first player can now adopt the following defense. He first makes an arbitrary move. Thereafter he plays 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*!

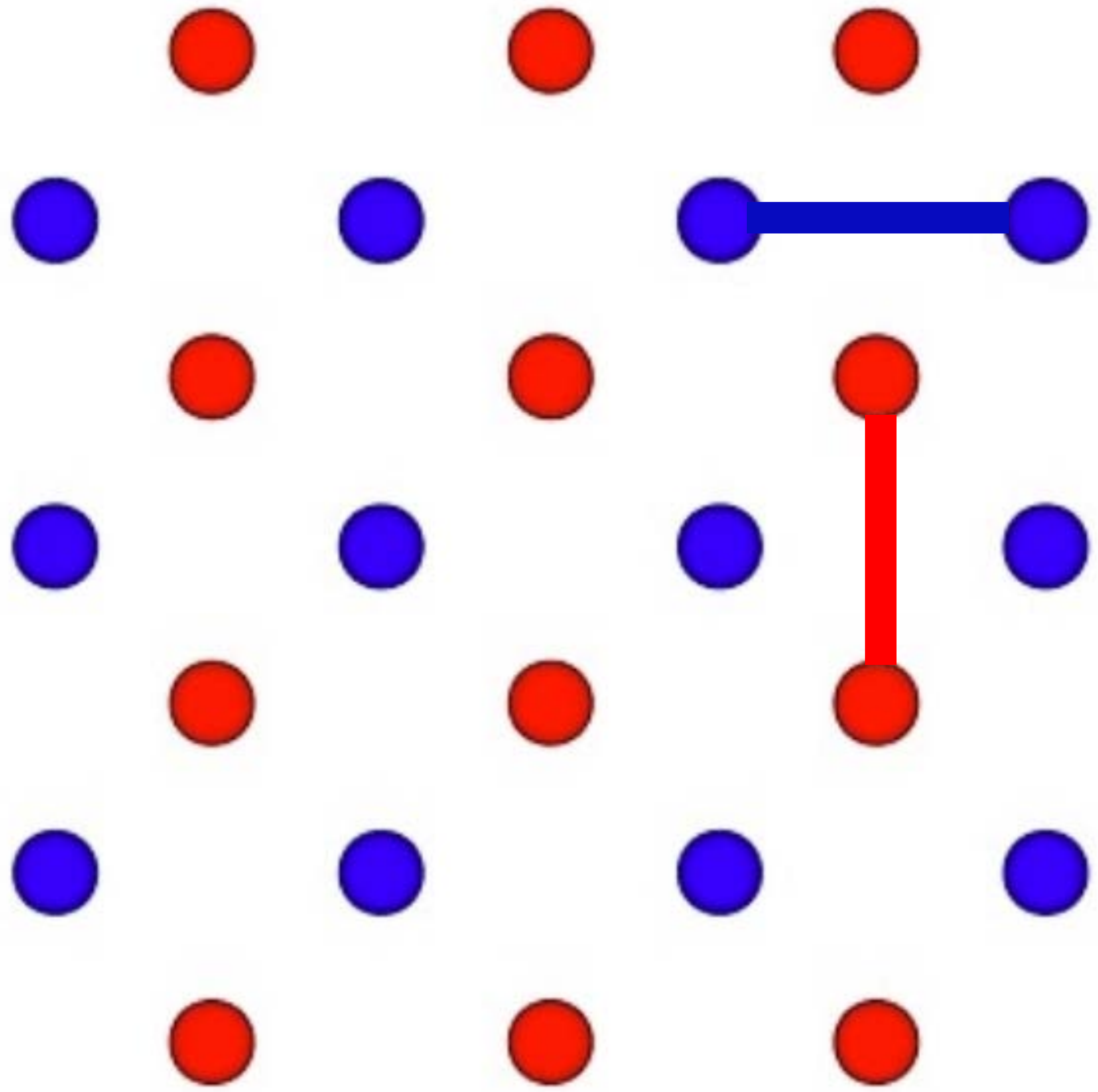

The NEW Unpredictable Game
BRIDG-IT
• THE PLOTTING INTRIGUE OF CHESS
• THE FASCINATION OF CHECKERS
• THE SIMPLICITY OF TIC-TAC-TOE
IT TAKES BUT SECONDS TO LEARN HOW TO PLAY
PAT. PENDING
Never a Tie—Always a Winner
BRIDG-IT
The NEW Intriguing Game

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

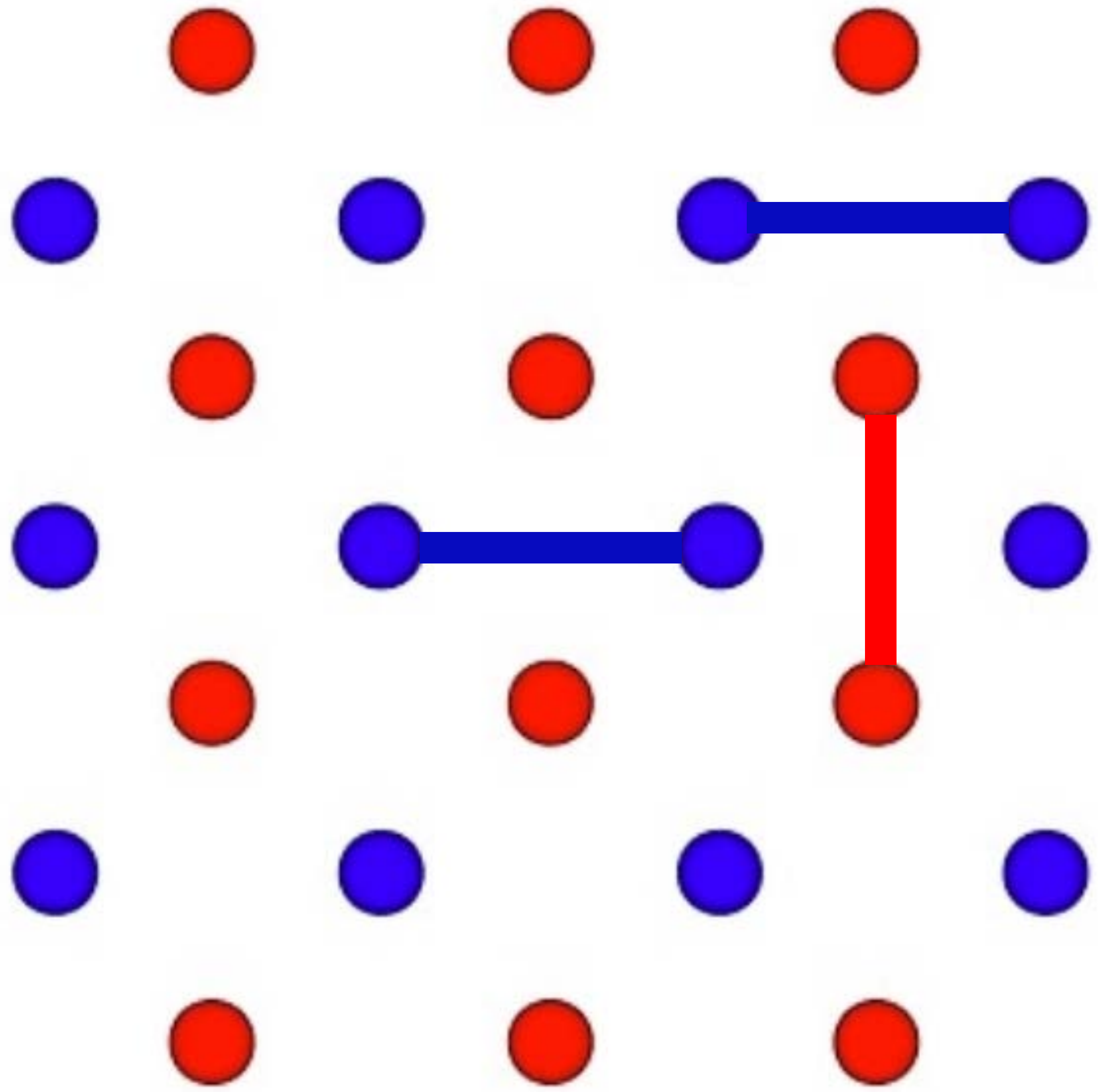
Bridg-It Game: Example 1



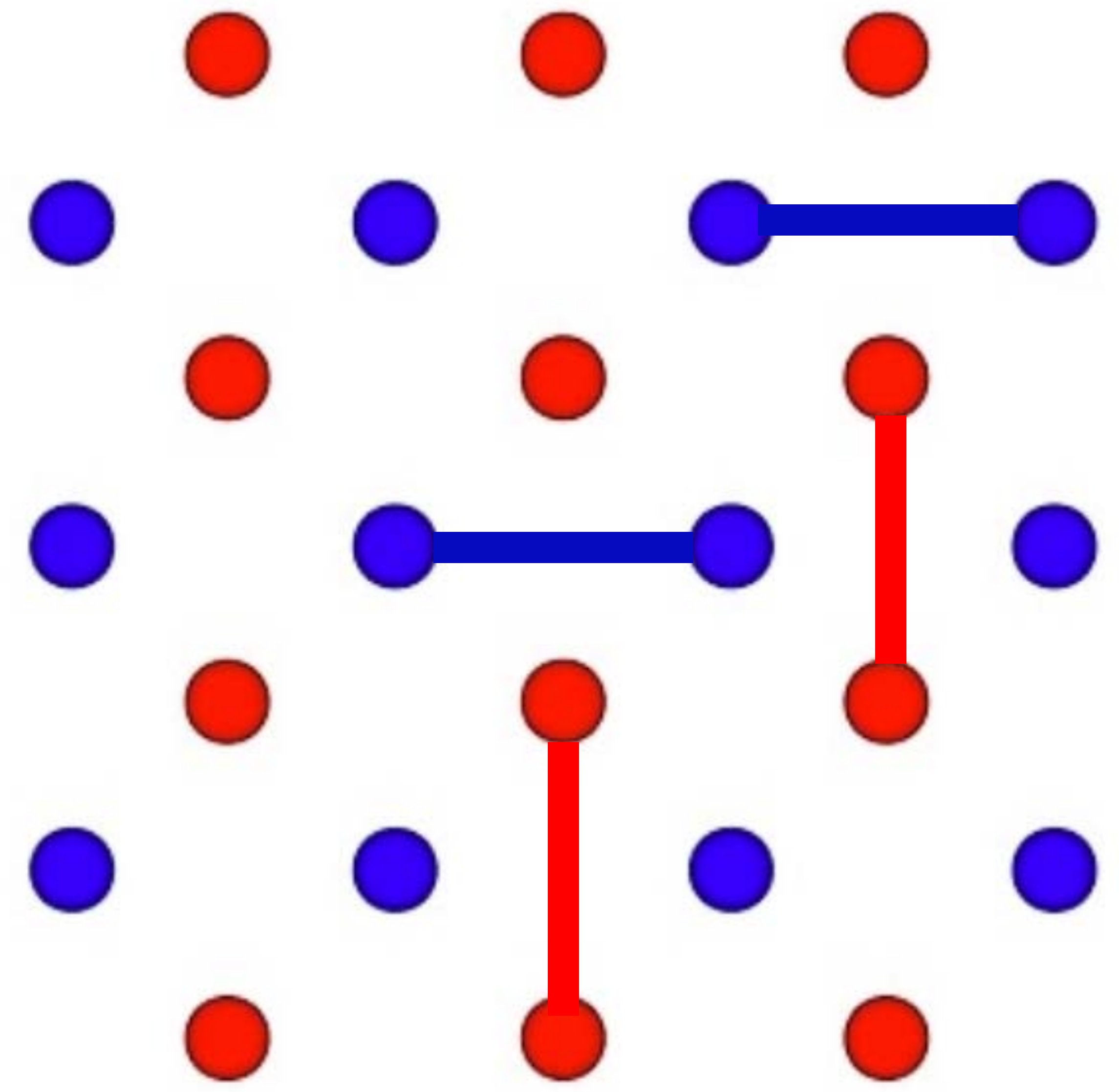
Bridg-It Game: Example 1



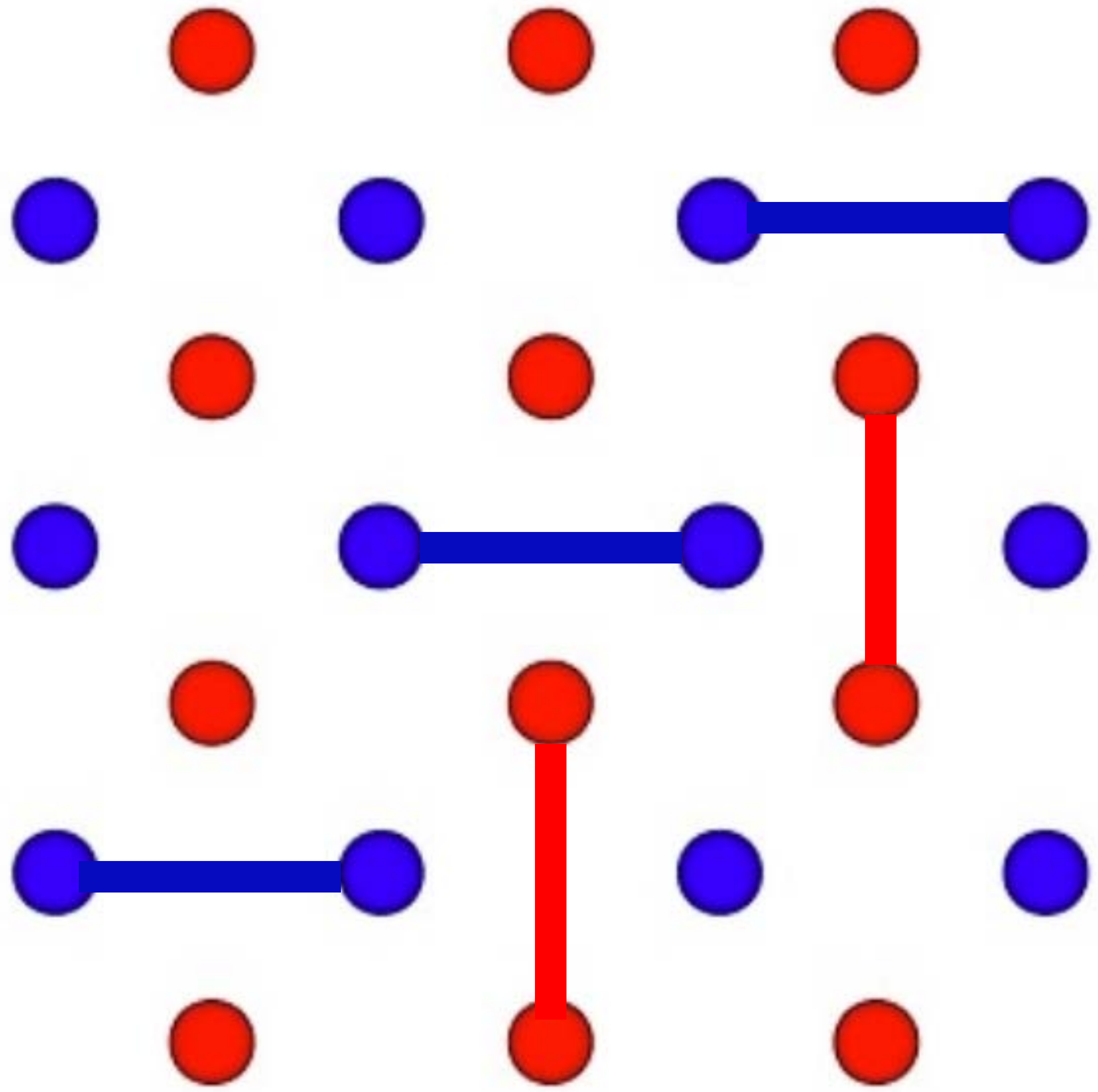
Bridg-It Game: Example 1



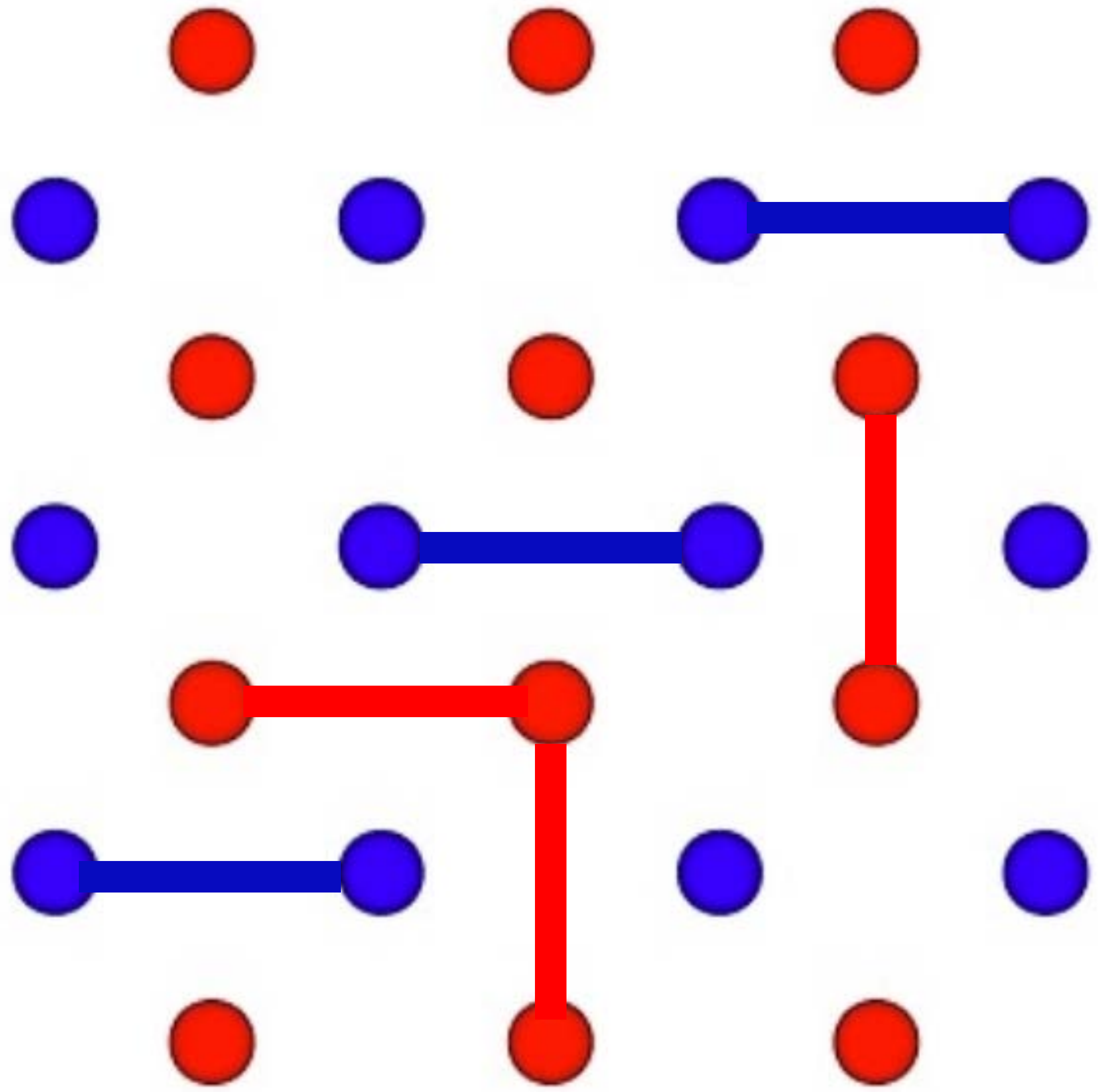
Bridg-It Game: Example 1



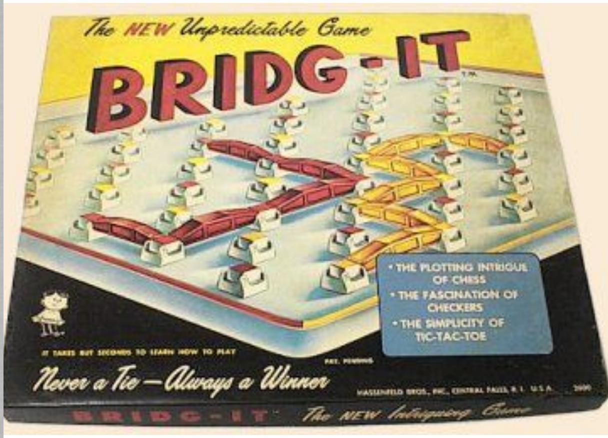
Bridg-It Game: Example 1



Bridg-It Game: Example 1



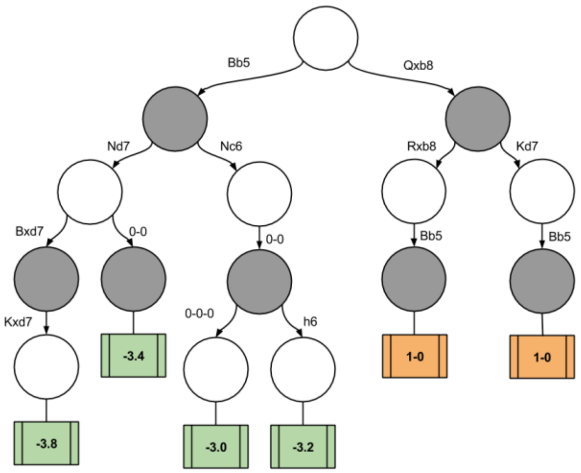
Bridg-It Game: Shannon's Machine

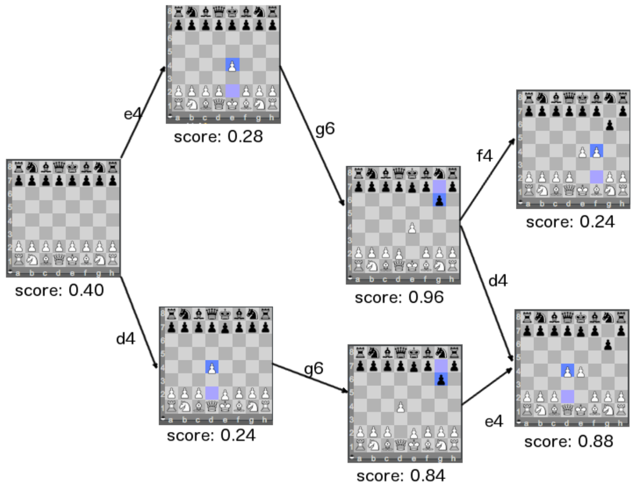

The NEW Unpredictable Game
BRIDG-IT
• THE PLOTTING INTRIGUE OF CHESS
• THE FASCINATION OF CHECKERS
• THE SIMPLICITY OF TIC-TAC-TOE
IT TAKES BUT SECONDS TO LEARN HOW TO PLAY
Never a Tie—Always a Winner
HASSENFELD BROS., INC. CENTRAL FALLS, R.I. U.S.A.
BRIDG-IT The NEW Intriguing Game

- 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?

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.

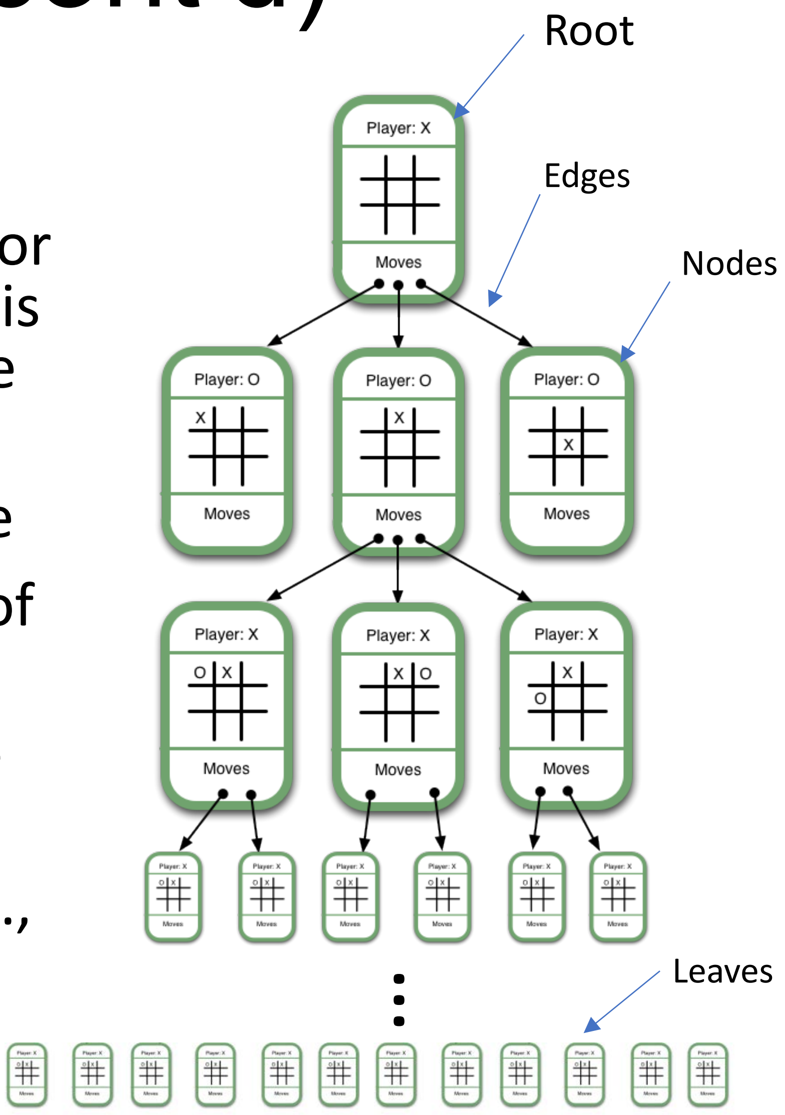




Game Tree (cont'd)

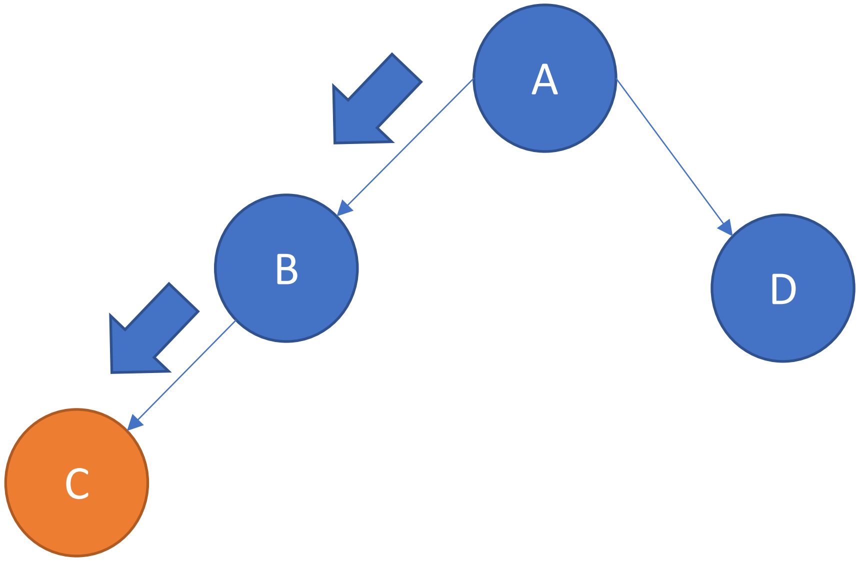
Each game consists of a problem space, an initial state, and a single (or a set of) goal states. The game tree is an abstraction of the problem space visualized as a rooted tree:

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)



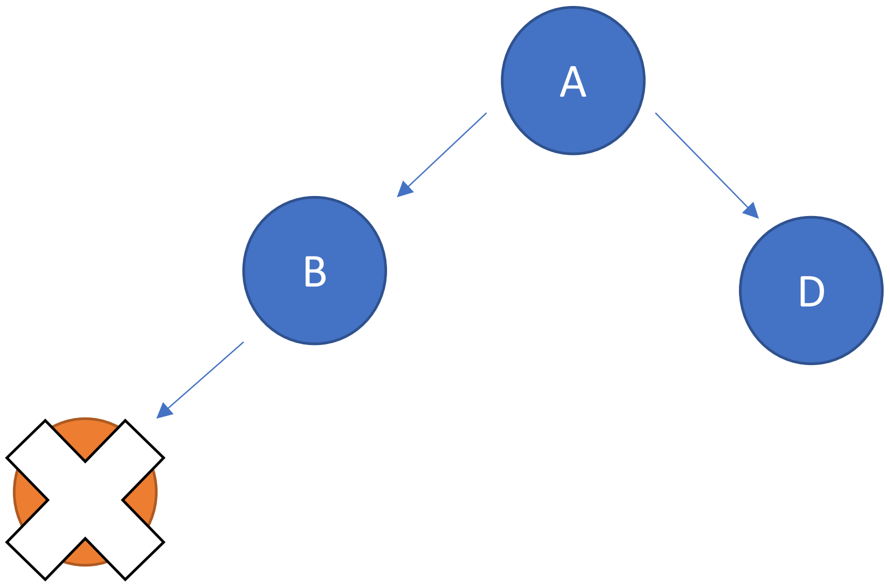
What is Backtracking?

- When you reach a dead-end



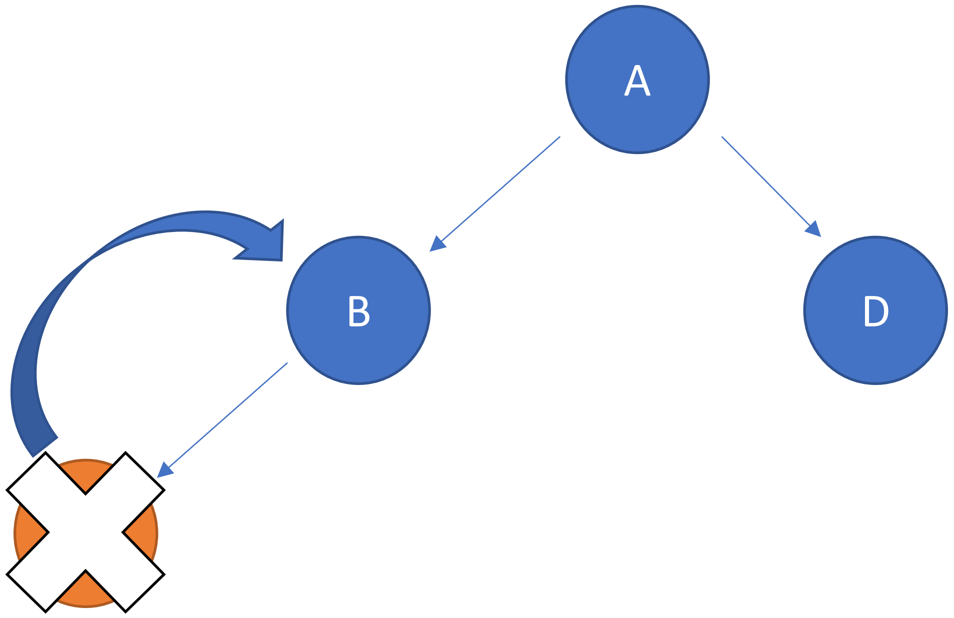
What is Backtracking?(cont.)

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



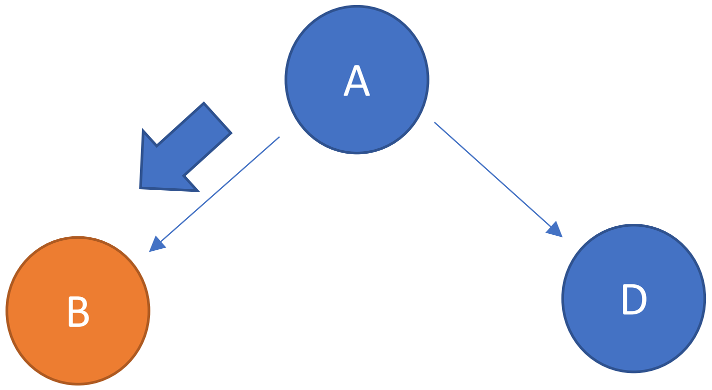
What is Backtracking?(cont.)

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



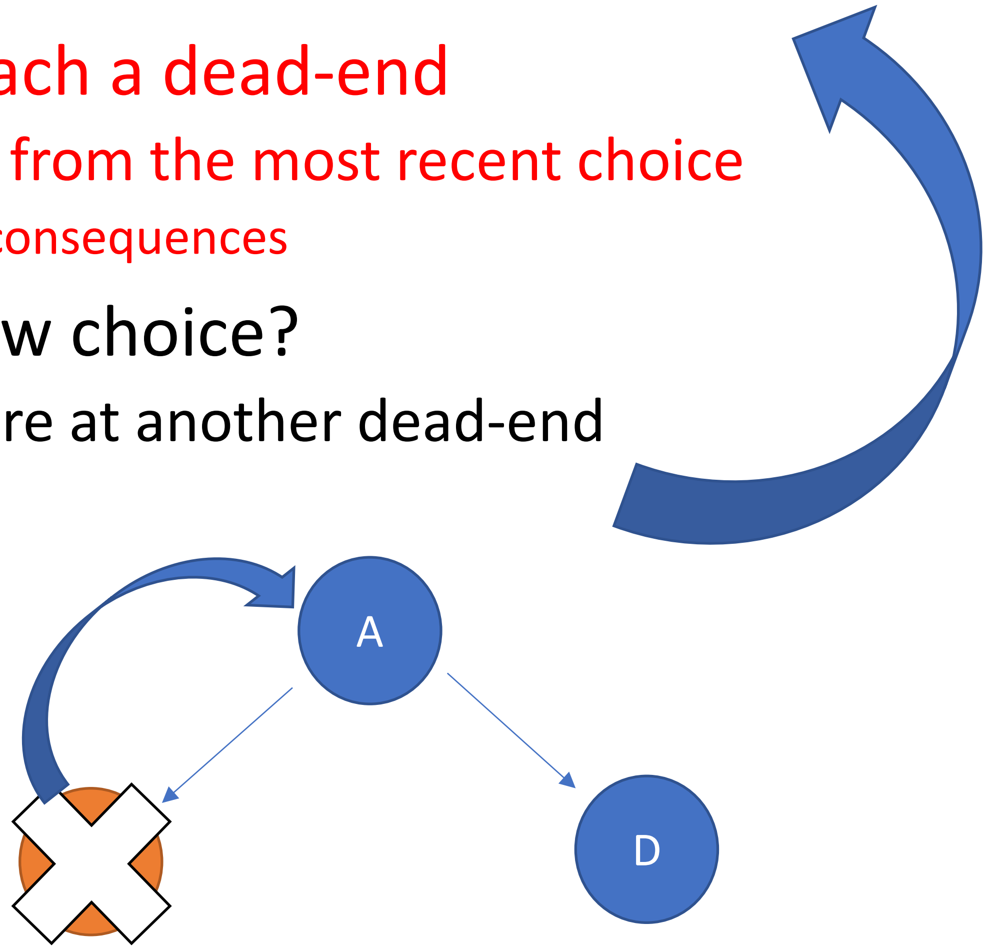
What is Backtracking?(cont.)

- 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



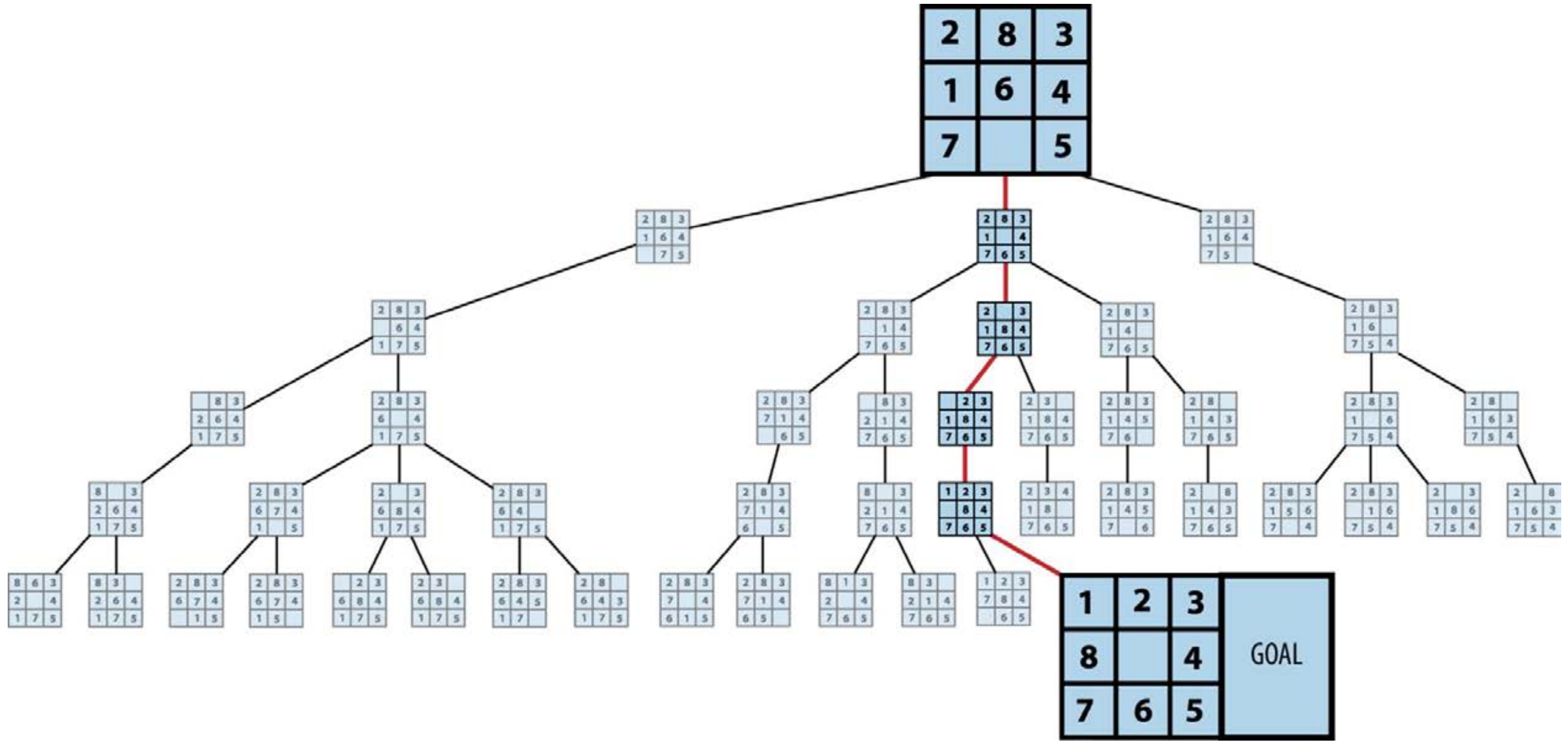
What is Backtracking?(cont.)

- 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



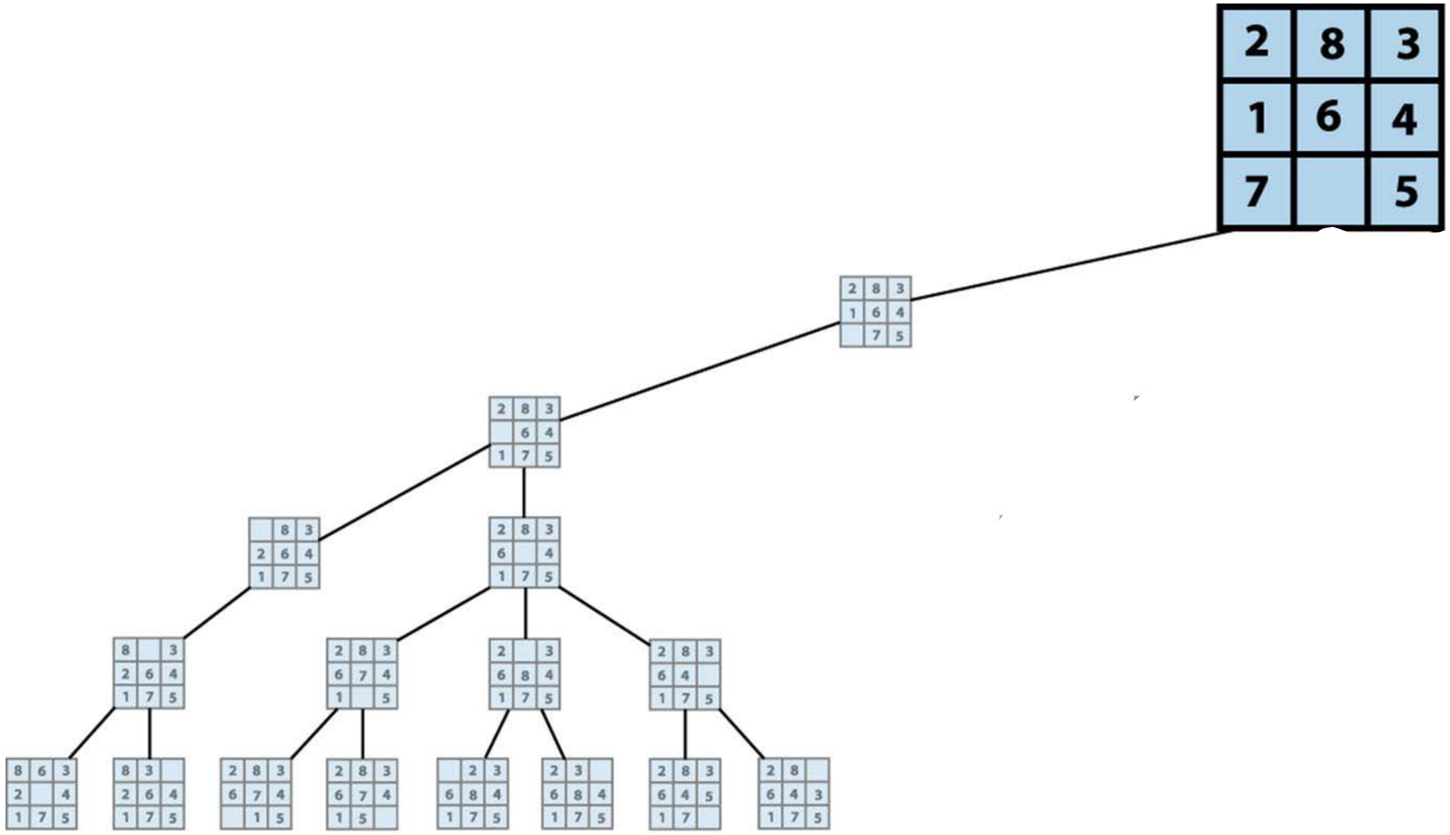
Example 1: 8-Puzzle game

- 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



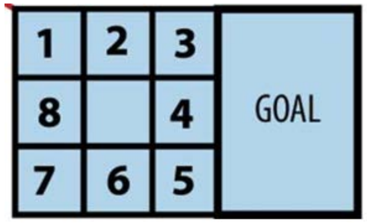
Example 1: 8-Puzzle game

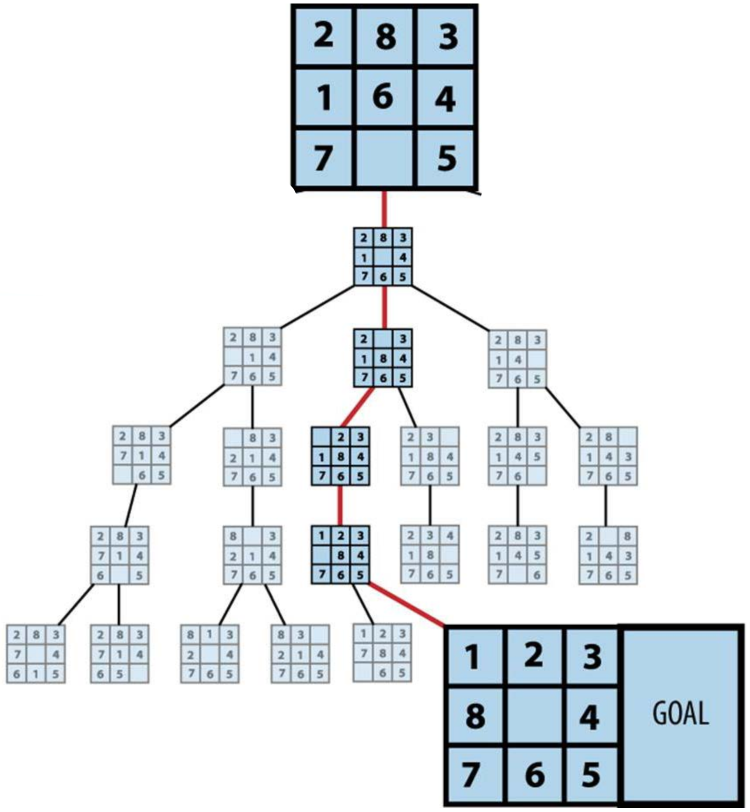
If we move 7, what are the states you can move to?



Example 1: 8-Puzzle game

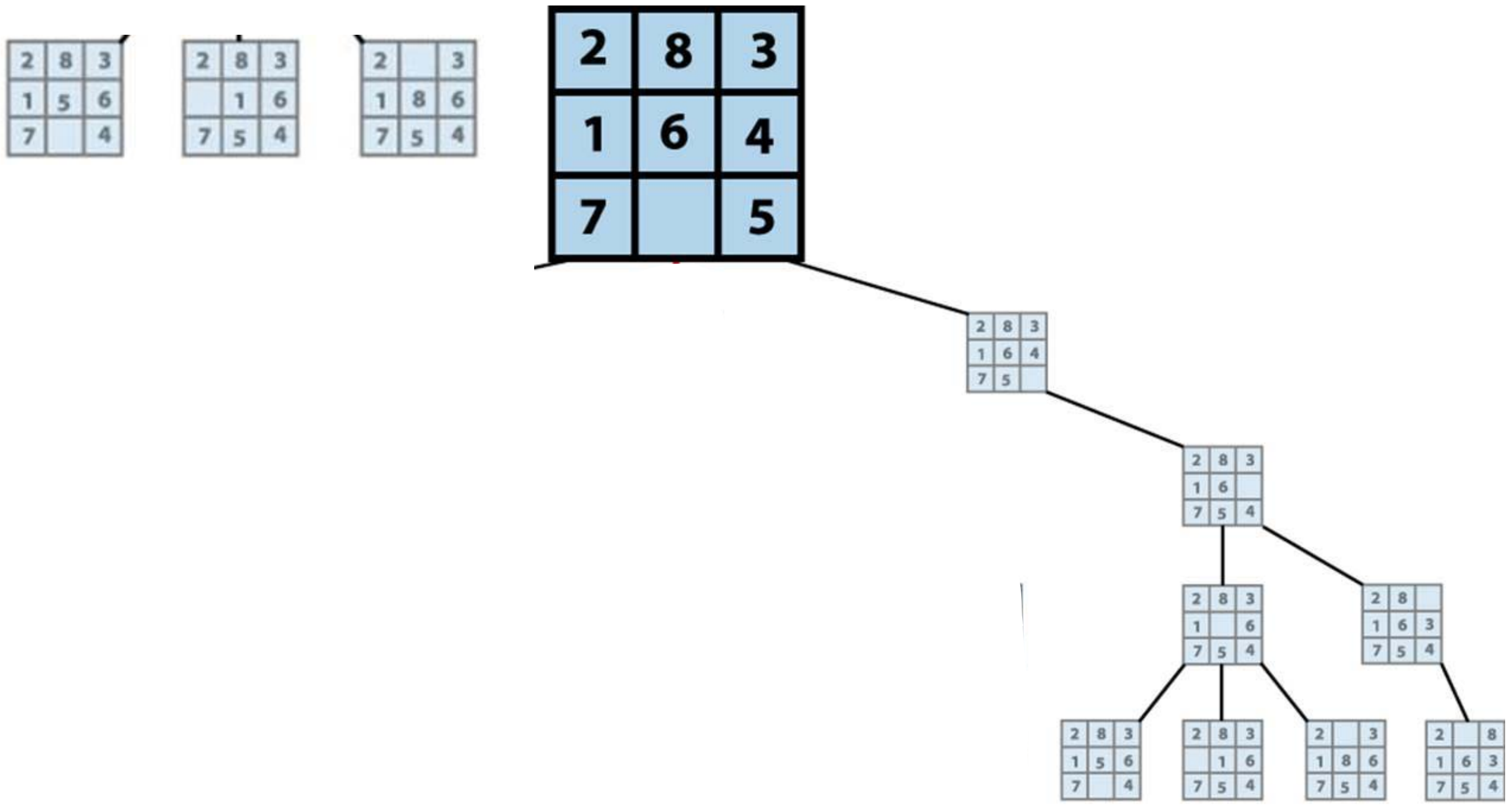
If we move 6, what are the states you can move to?





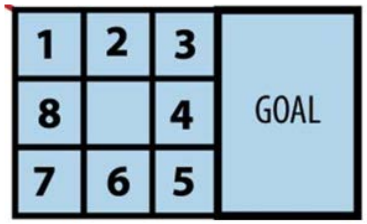
Example 1: 8-Puzzle game

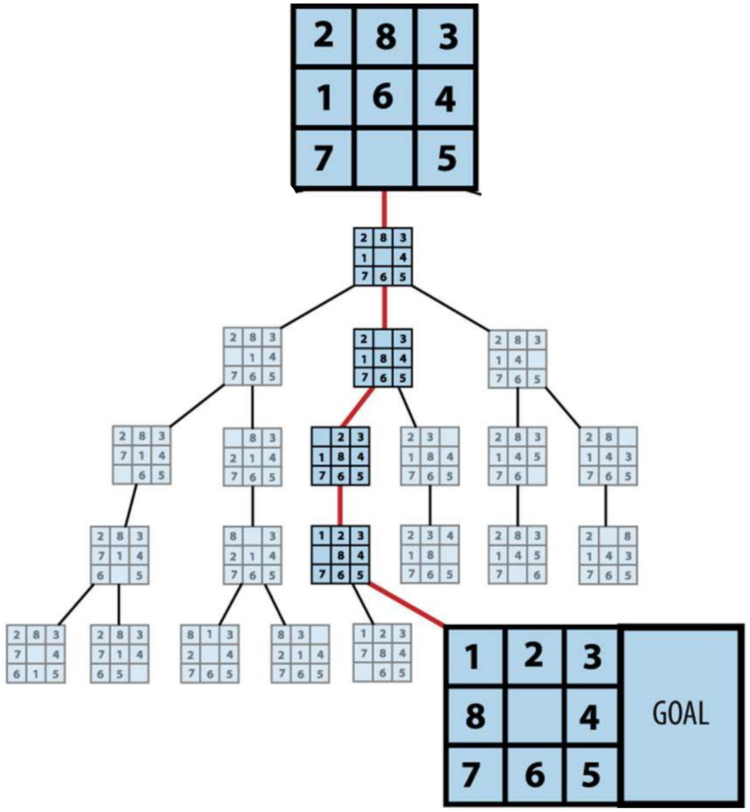
If we move 5, what are the states you can move to?



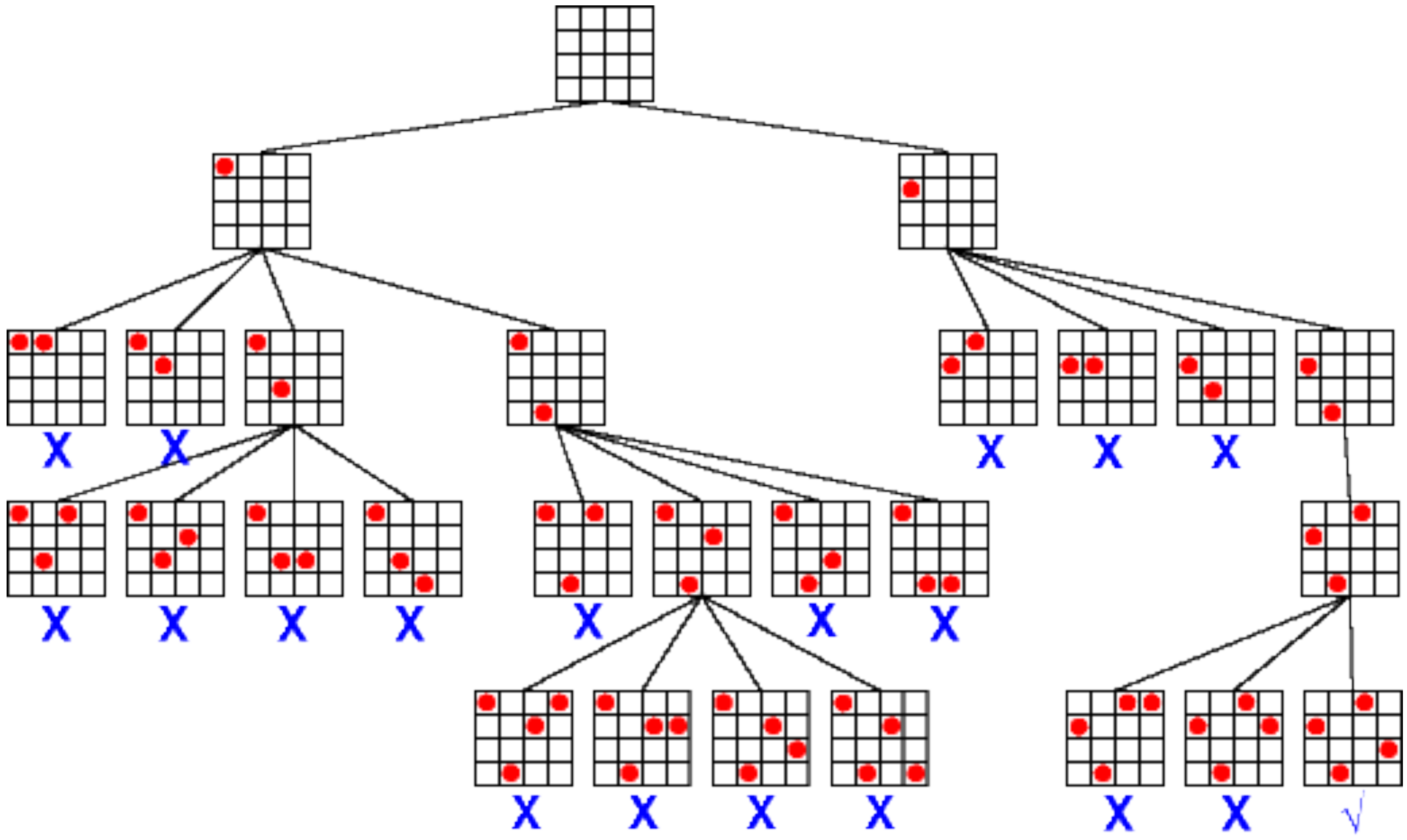
Example 1: 8-Puzzle game

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.