极客大学算法训练营
第十三课
高级搜索

覃超

Sophon Tech 创始人，前 Facebook 工程师

目录

- 剪枝
- 双向 BFS
- 启发式搜索（A*）

初级搜索

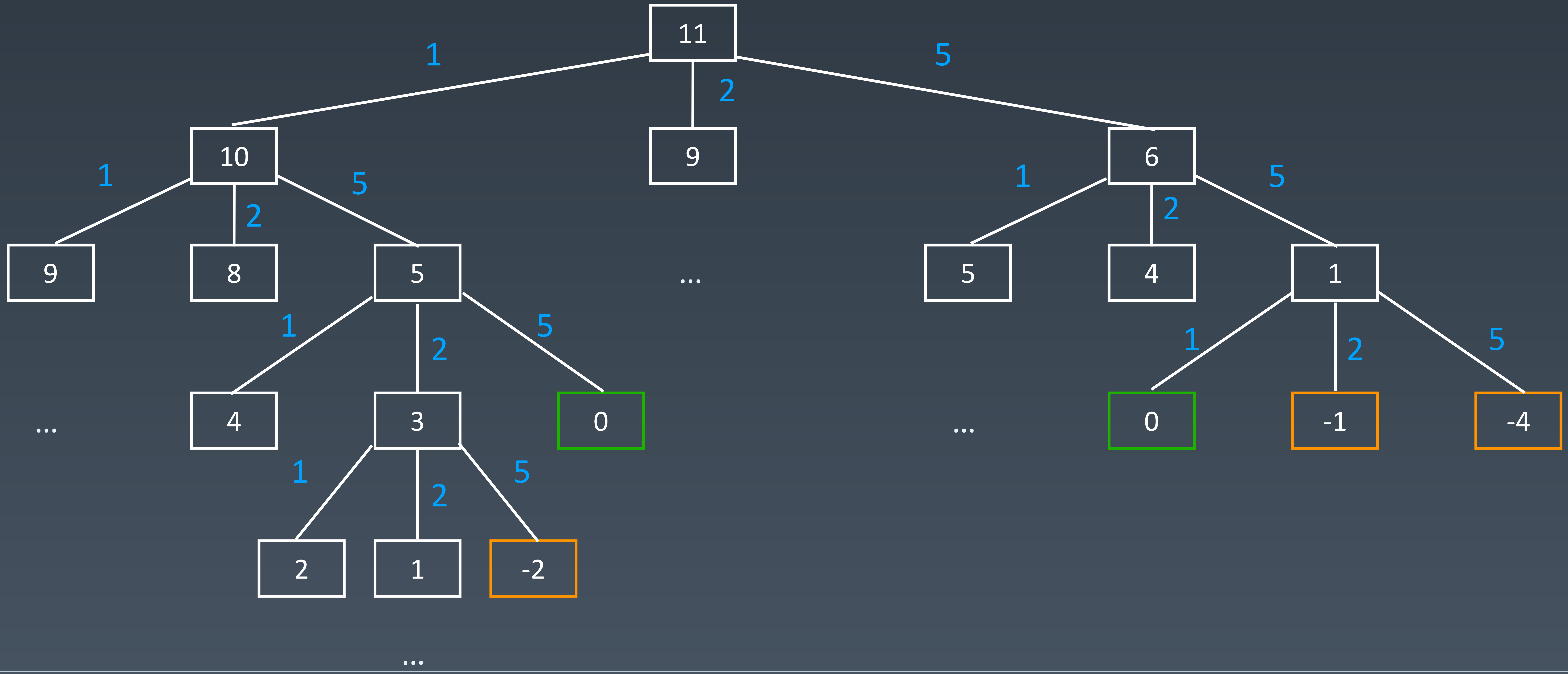
1. 朴素搜索

2. 优化方式：不重复（fibonacci）、剪枝（生成括号问题）

3. 搜索方向：
   DFS: depth first search 深度优先搜索
   BFS: breadth first search 广度优先搜索

   双向搜索、启发式搜索

Coin change（零钱置换）的状态树



DFS 代码 - 递归写法

```
visited = set()

def dfs(node, visited):
    if node in visited: # terminator
        # already visited
        return

    visited.add(node)

    # process current node here.
    ...
    for next_node in node.children():
        if not next_node in visited:
            dfs(next_node, visited)
```

DFS 代码 - 非递归写法

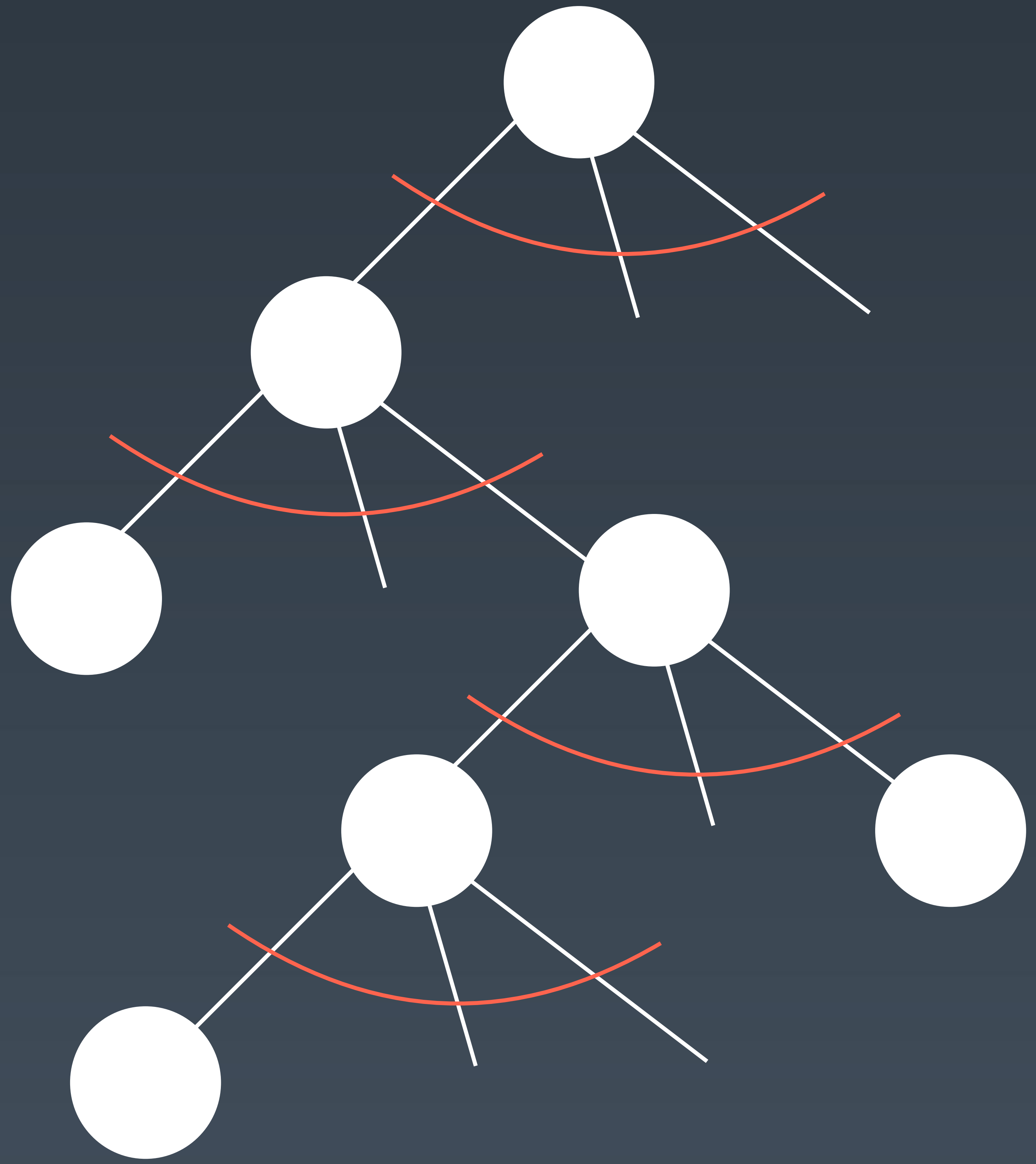
```
def DFS(self, tree):

    if tree.root is None:
        return []

    visited, stack = [], [tree.root]

    while stack:
        node = stack.pop()
        visited.add(node)

        process (node)
        nodes = generate_related_nodes(node)
        stack.push(nodes)

    # other processing work
    ...
```

BFS 代码

```
def BFS(graph, start, end):

    queue = []
    queue.append([start])
    visited.add(start)

    while queue:
        node = queue.pop()
        visited.add(node)

        process(node)
        nodes = generate_related_nodes(node)
        queue.push(nodes)
```

剪枝

剪枝



-> 50 个分支

-> 50 个分支

-> 50 个分支

-> 50 个分支

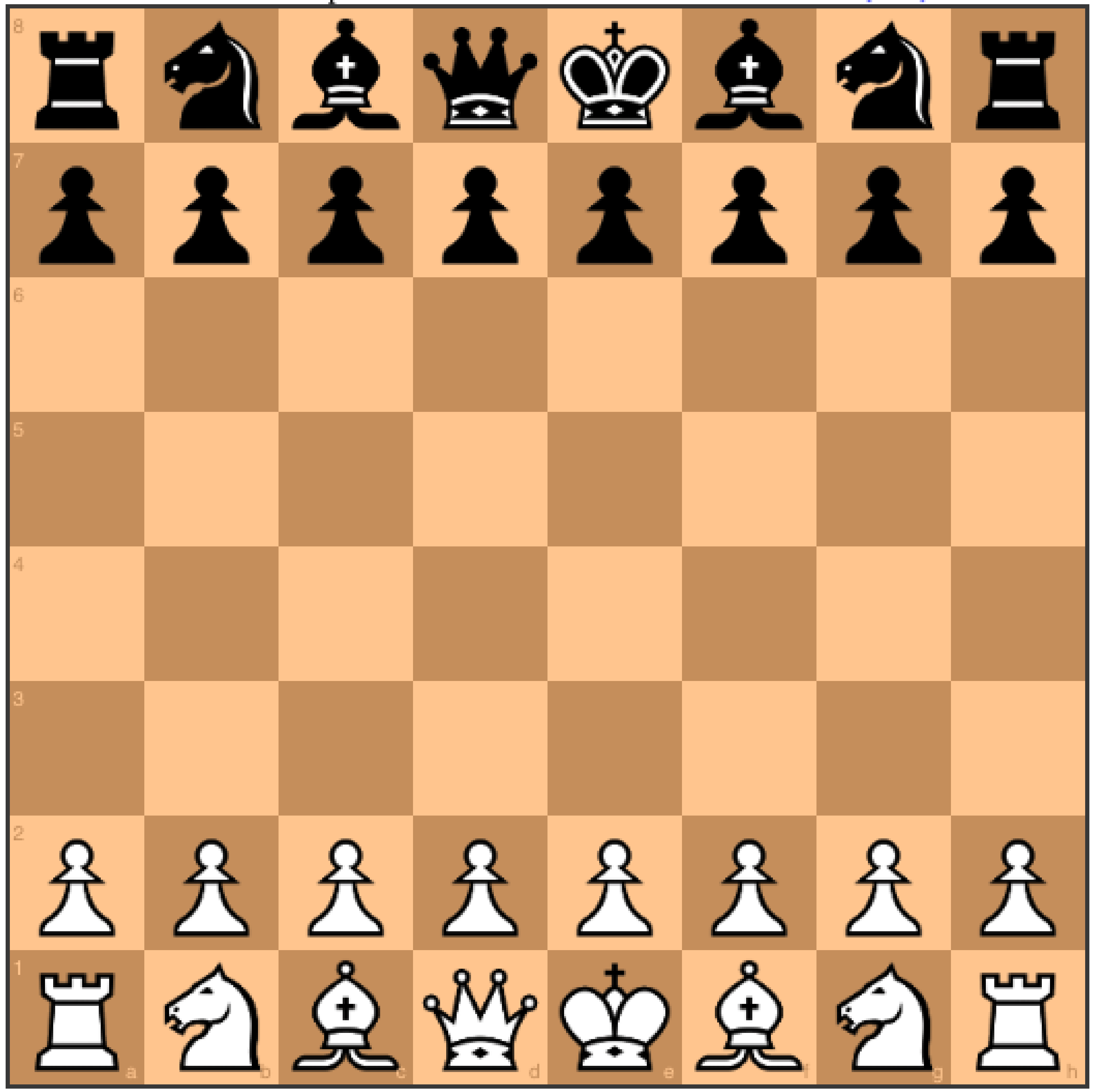
Garry Kasparov
VS
DEEP BLUE

Deep Blue (Computer) vs Garry Kasparov

"Sacre Blue!" (game of the day Jun-05-2008)

Match (1996), Philadelphia, PA USA, rd 1, Feb-10

Sicilian Defense: Alapin Variation. Barmen Defense Modern Line (B22) · 1-0

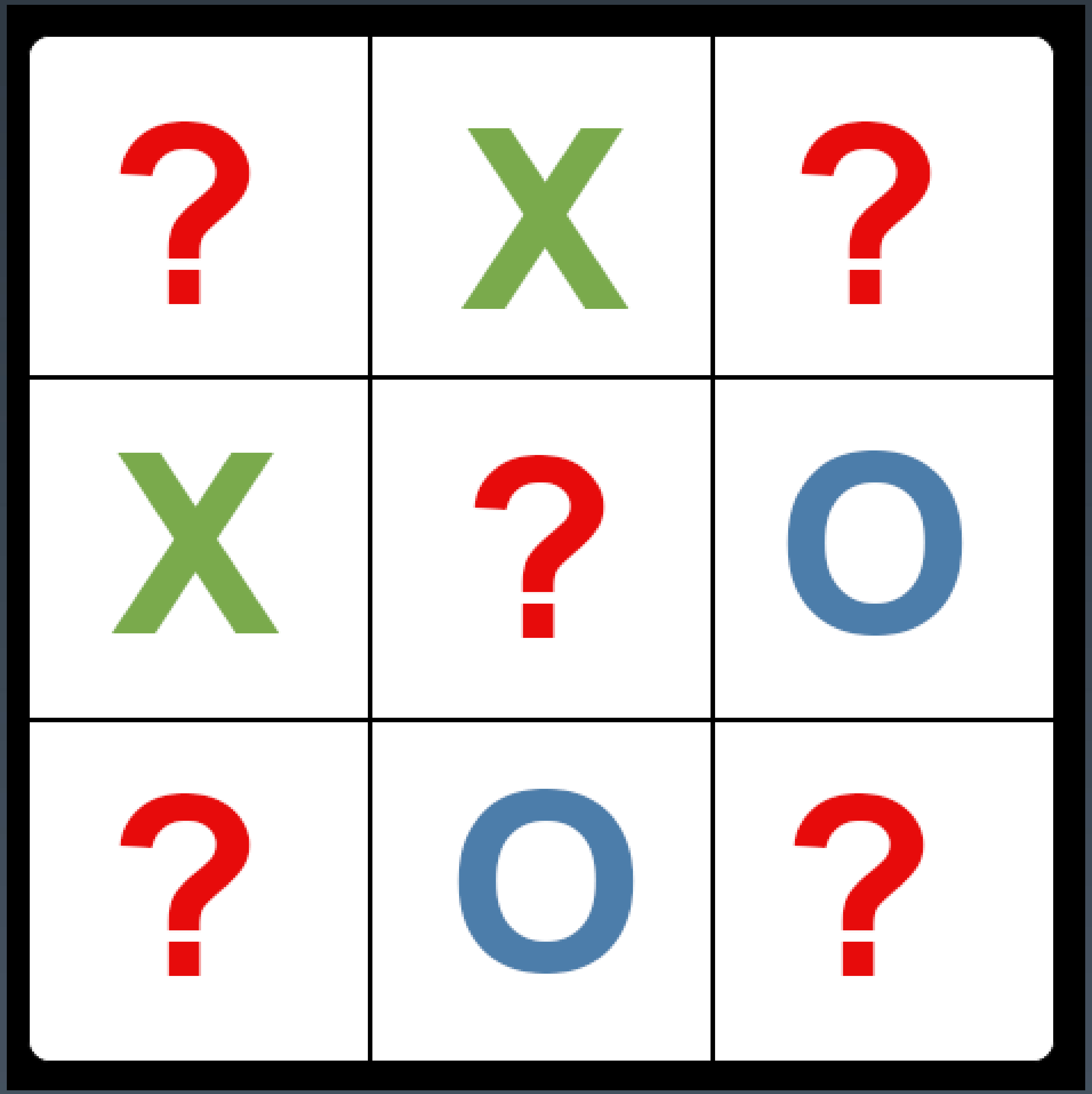


« < > » +

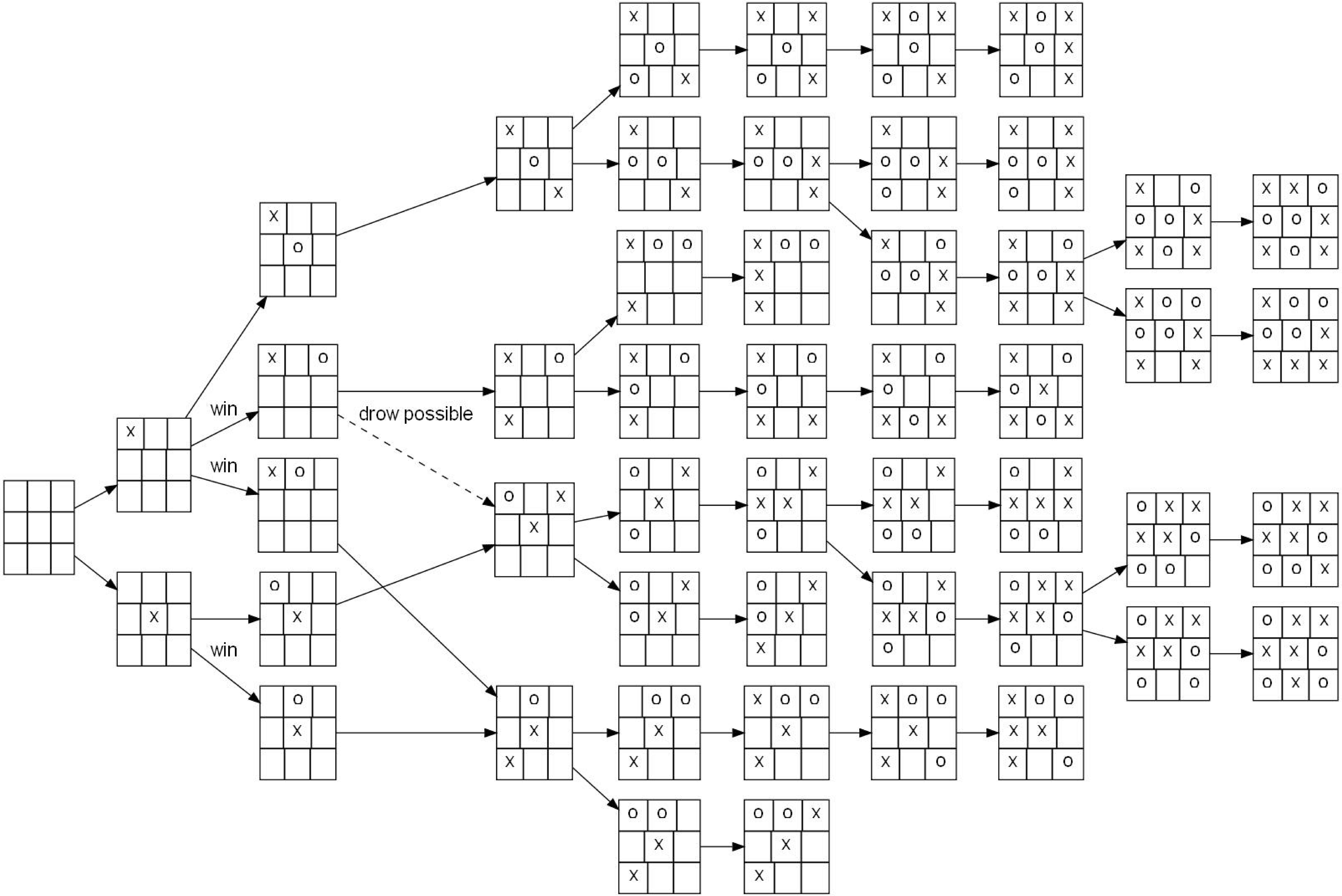
White to move.

1.e4 c5 2.c3 d5 3.exd5 ♕xd5 4.d4 ♘f6 5.♘f3 ♗g4 6.♗e2 e6 7.h3 ♗h5 8.O-O ♘c6 9.♗e3 cxd4 10.cxd4 ♗b4 11.a3 ♗a5 12.♘c3 ♕d6 13.♘b5 ♕e7 14.♘e5 ♗xe2 15.♕xe2 O-O 16.♖ac1 ♖ac8 17.♗g5 ♗b6 18.♗xf6 gxf6 19.♘c4 ♖fd8 20.♘xb6 axb6 21.♖fd1 f5 22.♕e3 ♕f6 23.d5 ♖xd5 24.♖xd5 exd5 25.b3 ♔h8 26.♕xb6 ♖g8 27.♕c5 d4 28.♘d6 f4 29.♘xb7 ♘e5 30.♕d5 f3 31.g3 ♘d3 32.♖c7 ♖e8 33.♘d6 ♖e1+ 34.♔h2 ♘xf2 35.♘xf7+ ♔g7 36.♘g5+ ♔h6 37.♖xh7+ 1-0

三子棋、五子棋等



?	X	?
X	?	O
?	O	?

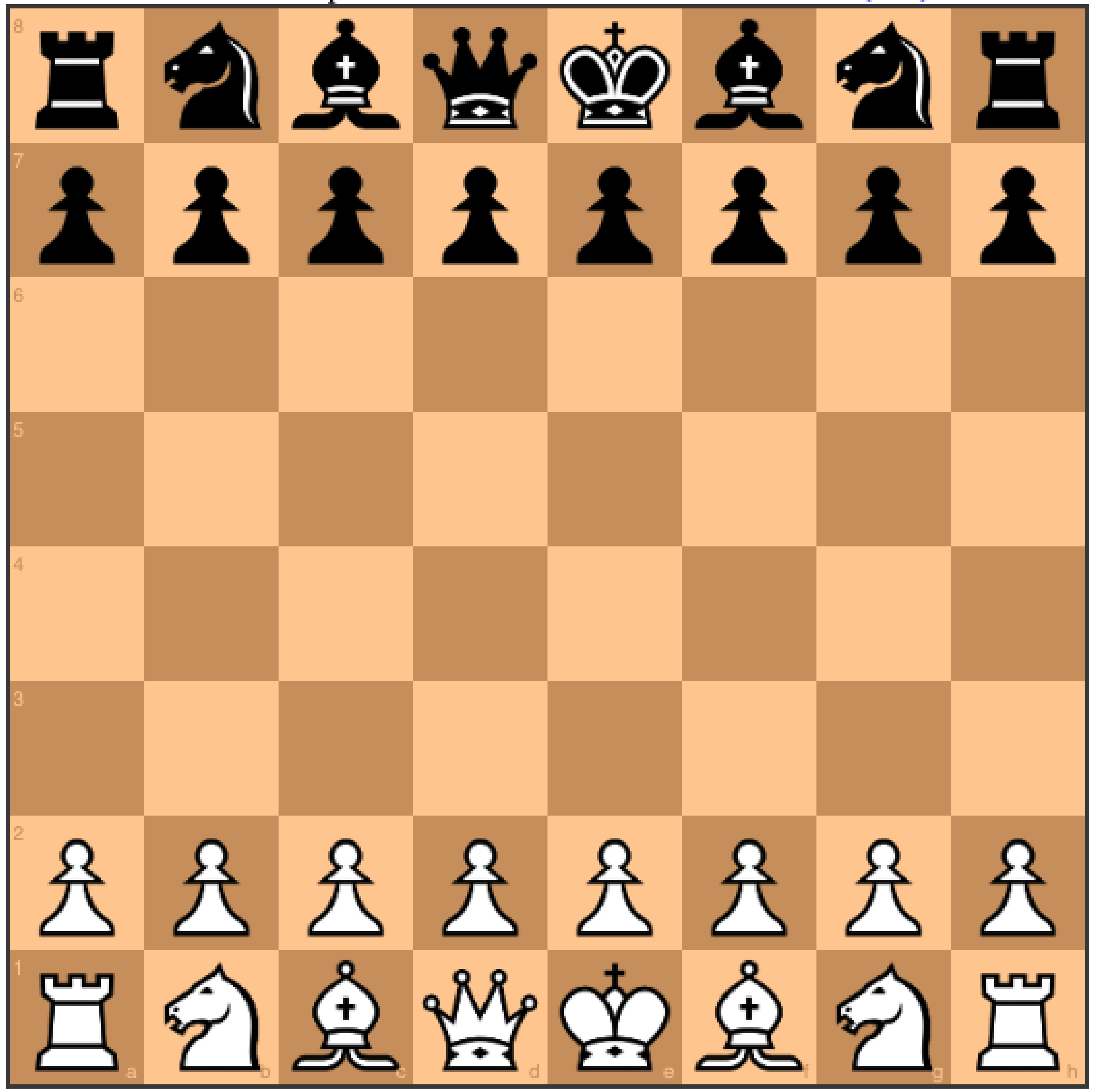

win
drow possible
win
win

Deep Blue (Computer) vs Garry Kasparov

"Sacre Blue!" (game of the day Jun-05-2008)

Match (1996), Philadelphia, PA USA, rd 1, Feb-10

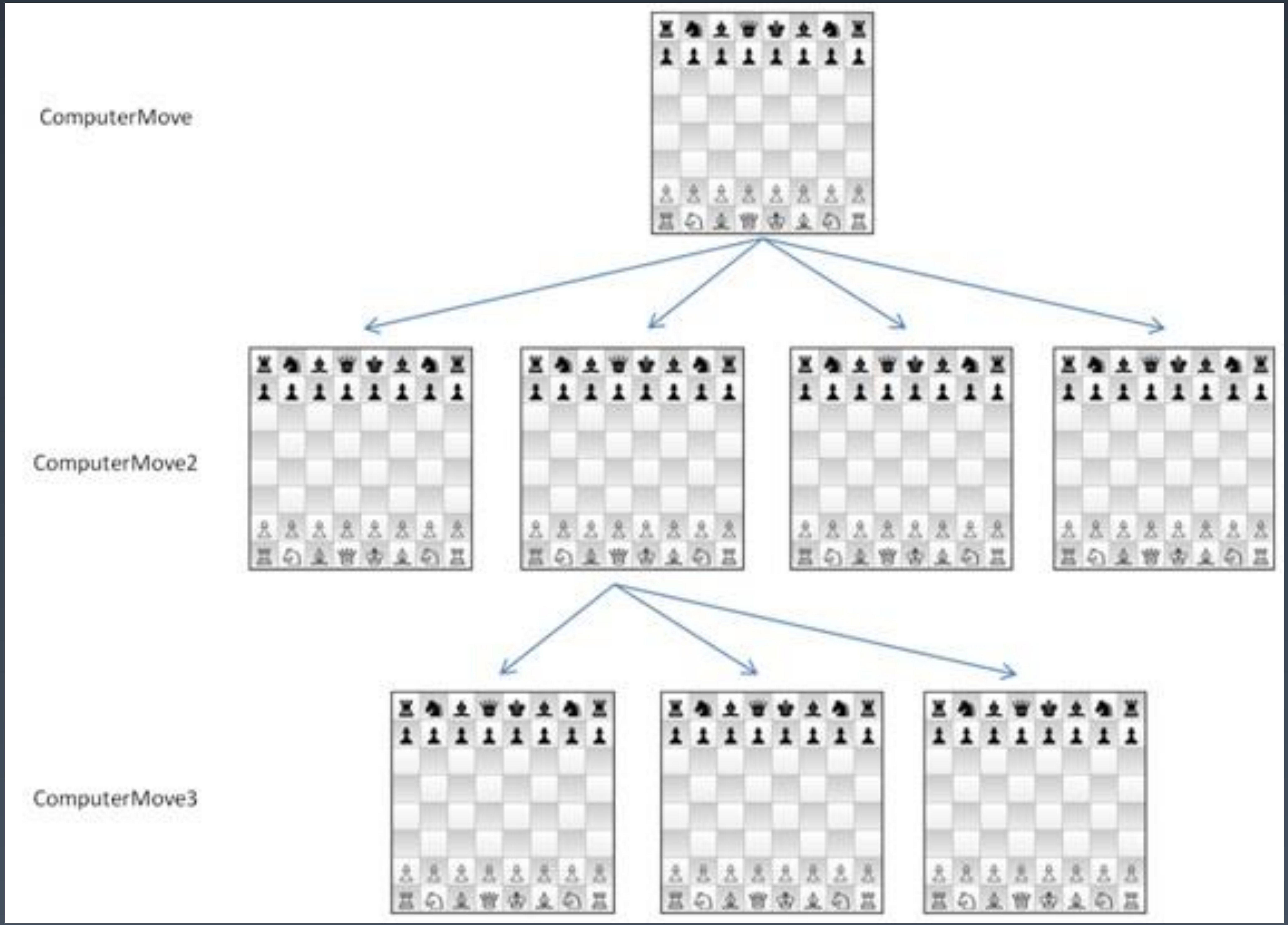
Sicilian Defense: Alapin Variation. Barmen Defense Modern Line (B22) · 1-0

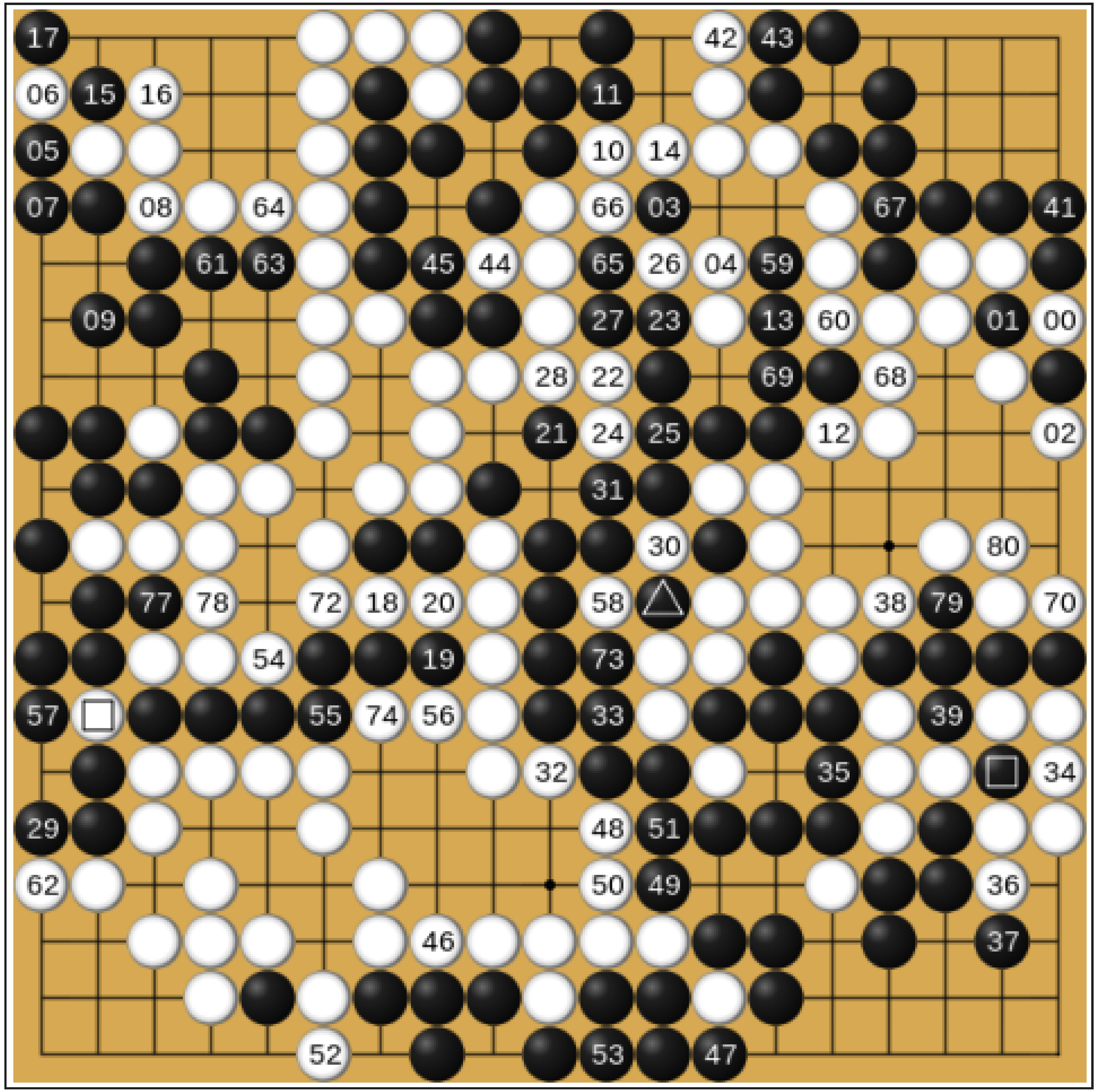


« < > » +

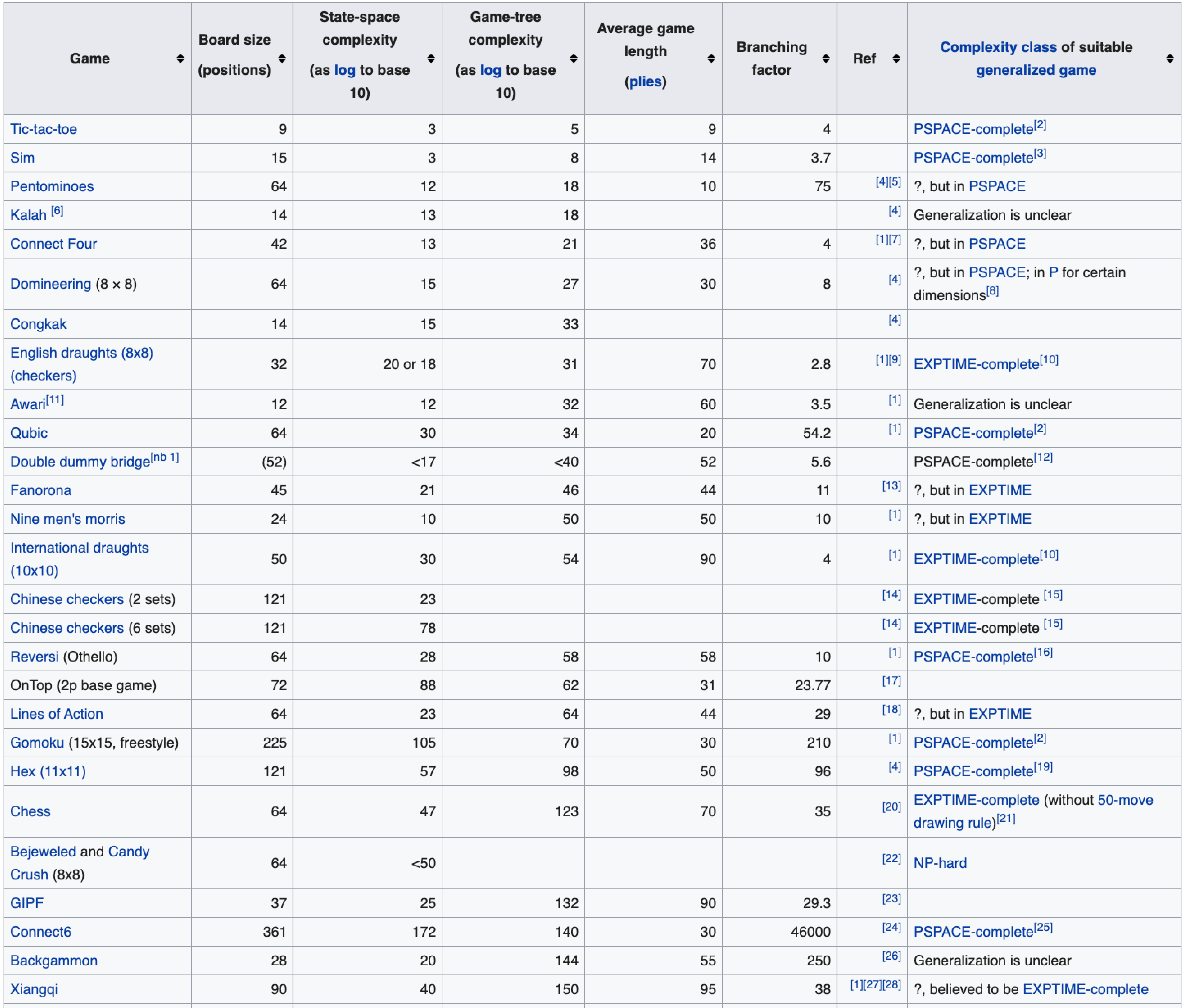
White to move.

1.e4 c5 2.c3 d5 3.exd5 ♕xd5 4.d4 ♘f6 5.♘f3 ♗g4 6.♗e2 e6 7.h3 ♗h5 8.O-O ♘c6 9.♗e3 cxd4 10.cxd4 ♗b4 11.a3 ♗a5 12.♘c3 ♕d6 13.♘b5 ♕e7 14.♘e5 ♗xe2 15.♕xe2 O-O 16.♖ac1 ♖ac8 17.♗g5 ♗b6 18.♗xf6 gxf6 19.♘c4 ♖fd8 20.♘xb6 axb6 21.♖fd1 f5 22.♕e3 ♕f6 23.d5 ♖xd5 24.♖xd5 exd5 25.b3 ♔h8 26.♕xb6 ♖g8 27.♕c5 d4 28.♘d6 f4 29.♘xb7 ♘e5 30.♕d5 f3 31.g3 ♘d3 32.♖c7 ♖e8 33.♘d6 ♖e1+ 34.♔h2 ♘xf2 35.♘xf7+ ♔g7 36.♘g5+ ♔h6 37.♖xh7+ 1-0


ComputerMove
ComputerMove2
ComputerMove3



Moves 200–280 (240 at 200, 271 at ■,
275 at □, 276 at △)



Game	Board size (positions)	State-space complexity (as log to base 10)	Game-tree complexity (as log to base 10)	Average game length (plies)	Branching factor	Ref	Complexity class of suitable generalized game
Tic-tac-toe	9	3	5	9	4		PSPACE-complete[2]
Sim	15	3	8	14	3.7		PSPACE-complete[3]
Pentominoes	64	12	18	10	75	[4][5]	?, but in PSPACE
Kalah [6]	14	13	18			[4]	Generalization is unclear
Connect Four	42	13	21	36	4	[1][7]	?, but in PSPACE
Domineering (8 × 8)	64	15	27	30	8	[4]	?, but in PSPACE; in P for certain dimensions[8]
Congkak	14	15	33			[4]	
English draughts (8x8) (checkers)	32	20 or 18	31	70	2.8	[1][9]	EXPTIME-complete[10]
Awari[11]	12	12	32	60	3.5	[1]	Generalization is unclear
Qubic	64	30	34	20	54.2	[1]	PSPACE-complete[2]
Double dummy bridge[nb 1]	(52)	<17	<40	52	5.6		PSPACE-complete[12]
Fanorona	45	21	46	44	11	[13]	?, but in EXPTIME
Nine men's morris	24	10	50	50	10	[1]	?, but in EXPTIME
International draughts (10x10)	50	30	54	90	4	[1]	EXPTIME-complete[10]
Chinese checkers (2 sets)	121	23				[14]	EXPTIME-complete [15]
Chinese checkers (6 sets)	121	78				[14]	EXPTIME-complete [15]
Reversi (Othello)	64	28	58	58	10	[1]	PSPACE-complete[16]
OnTop (2p base game)	72	88	62	31	23.77	[17]	
Lines of Action	64	23	64	44	29	[18]	?, but in EXPTIME
Gomoku (15x15, freestyle)	225	105	70	30	210	[1]	PSPACE-complete[2]
Hex (11x11)	121	57	98	50	96	[4]	PSPACE-complete[19]
Chess	64	47	123	70	35	[20]	EXPTIME-complete (without 50-move drawing rule)[21]
Bejeweled and Candy Crush (8x8)	64	<50				[22]	NP-hard
GIPF	37	25	132	90	29.3	[23]	
Connect6	361	172	140	30	46000	[24]	PSPACE-complete[25]
Backgammon	28	20	144	55	250	[26]	Generalization is unclear
Xiangqi	90	40	150	95	38	[1][27][28]	?, believed to be EXPTIME-complete

回溯法

回溯法采用试错的思想，它尝试分步的去解决一个问题。在分步解决问题的过程中，当它通过尝试发现现有的分步答案不能得到有效的正确的解答的时候，它将取消上一步甚至是上几步的计算，再通过其它的可能的分步解答再次尝试寻找问题的答案。

回溯法通常用最简单的递归方法来实现，在反复重复上述的步骤后可能出现两种情况：

- 找到一个可能存在的正确的答案
- 在尝试了所有可能的分步方法后宣告该问题没有答案

在最坏的情况下，回溯法会导致一次复杂度为指数时间的计算。

实战练习

- https://leetcode-cn.com/problems/climbing-stairs/
- https://leetcode-cn.com/problems/generate-parentheses/

实战练习

1. https://leetcode-cn.com/problems/n-queens/

2. https://leetcode-cn.com/problems/valid-sudoku/description/

3. https://leetcode-cn.com/problems/sudoku-solver/#/description

八皇后代码

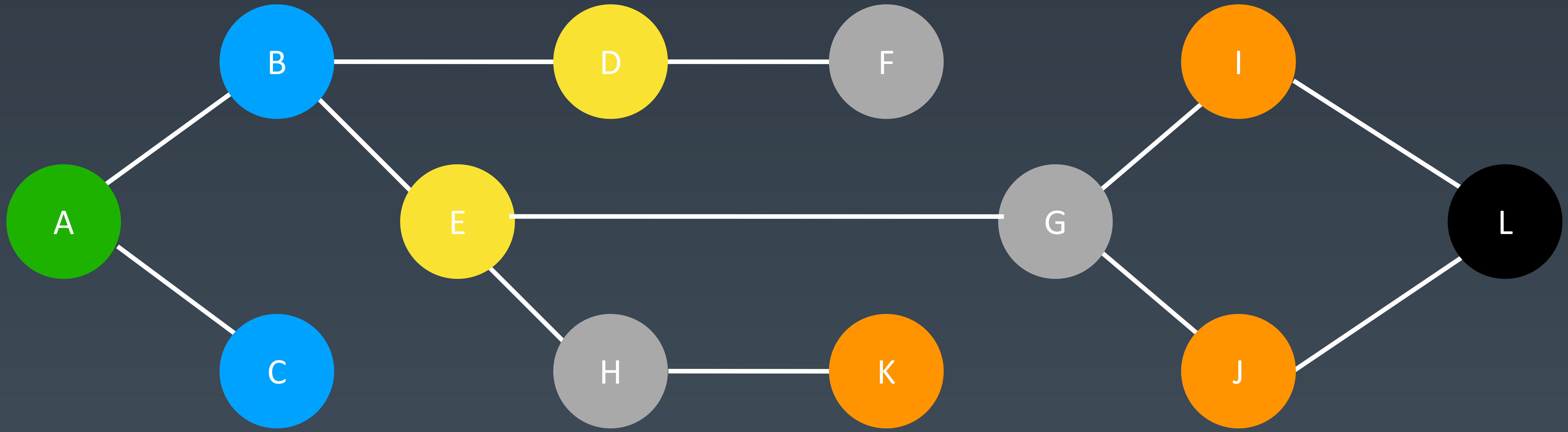
```
def solveNQueens(self, n):
   if n < 1: return []
   self.result = []
   self.cols = set(); self.pie = set(); self.na = set()
   self.DFS(n, 0, [])
   return self._generate_result(n)

def DFS(self, n, row, cur_state):
   # recursion terminator
   if row >= n:
      self.result.append(cur_state)
      return

   for col in range(n):
      if col in self.cols or row + col in self.pie or row - col in self.na:
        # go die!
        continue

      # update the flags
     self.cols.add(col)
     self.pie.add(row + col)
     self.na.add(row - col)

     self.DFS(n, row + 1, cur_state + [col])

     self.cols.remove(col)
     self.pie.remove(row + col)
     self.na.remove(row - col)
```

LeetCode 讨论区代码剖析

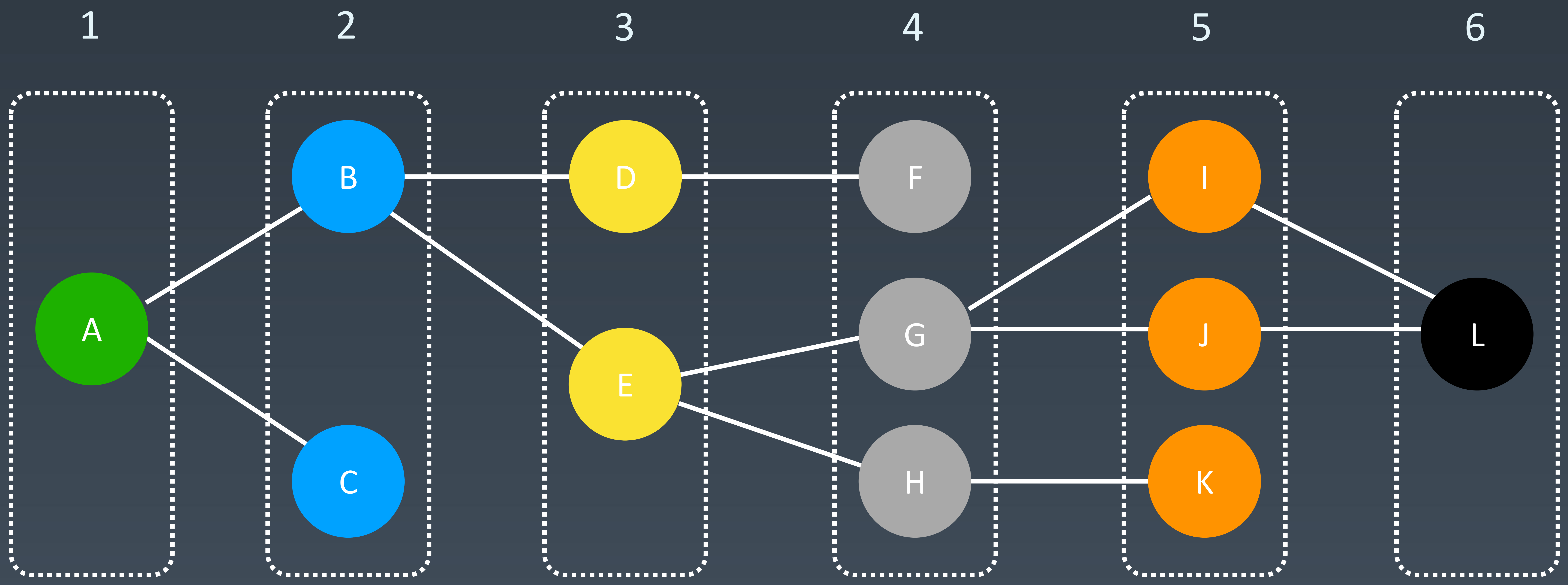
- https://leetcode.com/problems/n-queens/discuss/19808/Accepted-4ms-c%2B%2B-solution-use-backtracking-and-bitmask-easy-understand
- https://leetcode.com/problems/n-queens/discuss/19810/Fast-short-and-easy-to-understand-python-solution-11-lines-76ms

双向 BFS

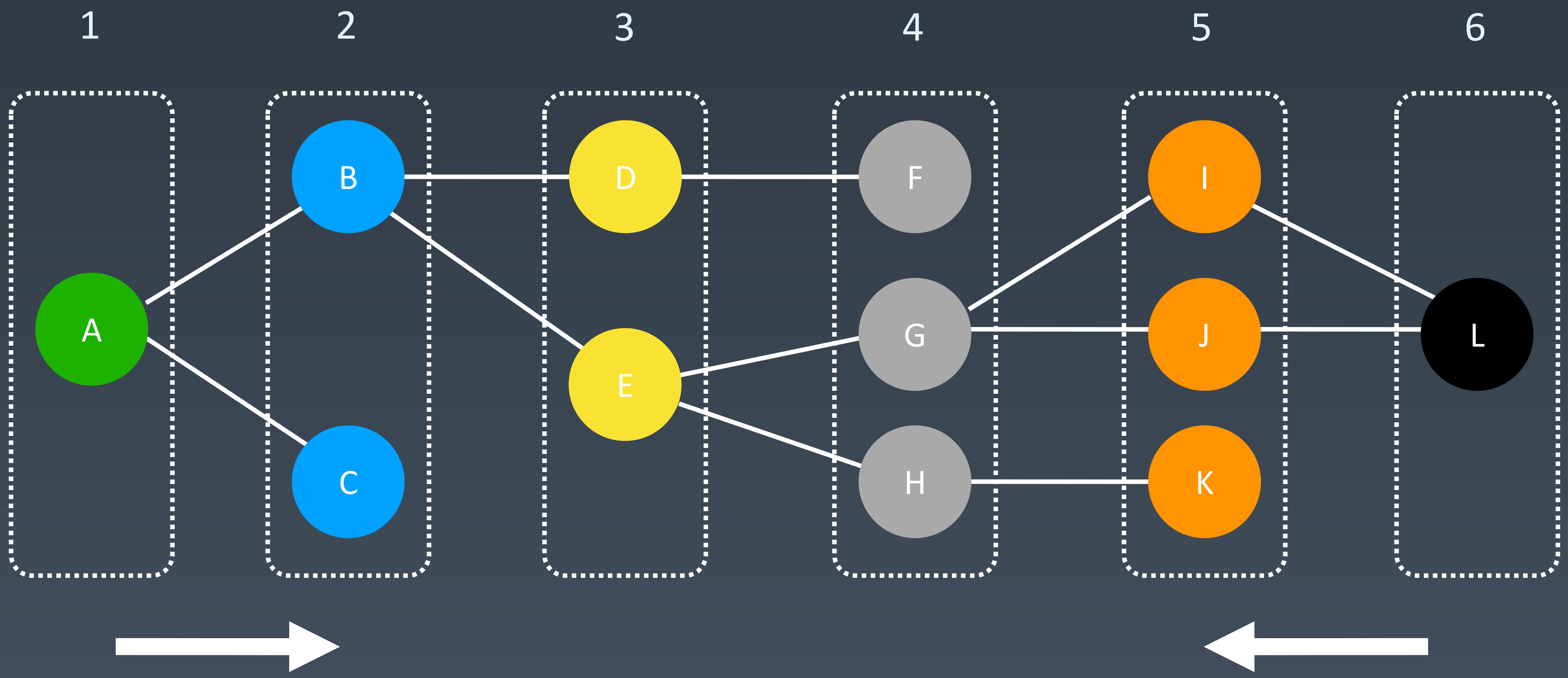
Breadth First Search (BFS)



Breadth-First Search Levels



Two-ended BFS 双向BFS



实战题目

1. https://leetcode-cn.com/problems/word-ladder/

2. https://leetcode-cn.com/problems/minimum-genetic-mutation/

启发式搜索

Heuristic Search (A*)

基于 BFS 代码

```
def BFS(graph, start, end):

    queue = []
    queue.append([start])
    visited.add(start)

    while queue:
        node = queue.pop() # can we add more intelligence here ?
        visited.add(node)

        process(node)
        nodes = generate_related_nodes(node)
        queue.push(nodes)
```

A* search

```
def AstarSearch(graph, start, end):

    pq = collections.priority_queue() # 优先级 —> 估价函数
    pq.append([start])
    visited.add(start)

    while pq:
        node = pq.pop() # can we add more intelligence here ?
        visited.add(node)

        process(node)
        nodes = generate_related_nodes(node)
      unvisited = [node for node in nodes if node not in visited]
        pq.push(unvisited)
```

估价函数

启发式函数： h(n)，它用来评价哪些结点最有希望的是一个我们要找的结点，h(n) 会返回一个非负实数,也可以认为是从结点n的目标结点路径的估计成本。

启发式函数是一种告知搜索方向的方法。它提供了一种明智的方法来猜测哪个邻居结点会导向一个目标。

实战题目

1. https://leetcode-cn.com/problems/shortest-path-in-binary-matrix/
2. https://leetcode-cn.com/problems/sliding-puzzle/
3. https://leetcode-cn.com/problems/sudoku-solver/

Shortest Path

1. BFS: 经典的BFS代码

2. A* search

   估价函数：
   h(current_point) = dist(current_point, destination_point)

   https://dataaspirant.com/2015/04/11/five-most-popular-similarity-measures-implementation-in-python/

3. https://leetcode.com/problems/shortest-path-in-binary-matrix/discuss/313347/A*-search-in-Python

Sliding Puzzle

1. BFS: 经典的BFS代码:
   https://leetcode-cn.com/problems/sliding-puzzle/submissions/

2. A* search

   估价函数:
   h(current_state) = distance(current_state, target_state)

3. https://zxi.mytechroad.com/blog/searching/8-puzzles-bidirectional-astar-vs-bidirectional-bfs/

THANKS!

极客大学