**Hành động Không xác định: Giải quyết Tic-Tac-Toe bằng Tìm kiếm Cây AND-OR**

**Giới thiệu**

Trò chơi nhiều người chơi có thể được triển khai như sau:

1. **Hành động không xác định (Nondeterministic actions):** Đối thủ được xem như một phần của môi trường với các hành động không xác định. Tính không xác định là kết quả của các nước đi không rõ của đối thủ.
2. **Quyết định tối ưu (Optimal Decisions):** Tìm kiếm Minimax (tìm kiếm toàn bộ cây trò chơi) và cắt tỉa alpha-beta.
3. **Tìm kiếm Cây Alpha-Beta Heuristic (Heuristic Alpha-Beta Tree Search):** Cắt bớt cây tìm kiếm và sử dụng heuristic để ước tính giá trị trạng thái.
4. **Tìm kiếm Cây Monte Carlo (Monte Carlo Tree search):** Mô phỏng các lượt chơi để ước tính giá trị trạng thái.

Ở đây, chúng ta sẽ triển khai tìm kiếm cho Tic-Tac-Toe.

Chúng ta sẽ triển khai:

* **Tìm kiếm Cây AND-OR (AND-OR-Tree search).**

Mỗi hành động bao gồm nước đi của người chơi (các cấp OR trong cây) và tất cả các phản ứng có thể (tức là không xác định) của đối thủ (các cấp AND). Do đó, hành động dẫn đến một tập hợp các trạng thái có thể.

Chúng ta sẽ tìm kiếm một **kế hoạch có điều kiện (conditional plan)** bằng cách sử dụng tìm kiếm Cây AND-OR.

Mã cho các hàm cơ bản được sử dụng cho tìm kiếm được triển khai trong tictactoe.py.

**Mã nguồn:**

Python

from tictactoe import empty\_board, actions, result, terminal, utility, other, show\_board

from tictactoe import random\_player, play

**Giải quyết Tic-Tac-Toe bằng Tìm kiếm Cây AND-OR**

Đối thủ có thể được mô hình hóa như một phần của môi trường. Hành động của đối thủ như một phản ứng với hành động của tác nhân làm cho môi trường có vẻ không xác định (xem AIMA trang 125 hoặc các slide). Thuật toán Cây ADS-OR sử dụng các hàm sau:

* is\_goal(state): xác định xem bàn cờ/trạng thái hiện tại có người chiến thắng hay không.
* is\_cycle(path): xác định xem chúng ta có đang ở trong một vòng lặp hay không (vấn đề tic-tac-toe không có vòng lặp nên tôi không triển khai nó).
* results(state, action): trả về tất cả các trạng thái sau trạng thái + hành động.

**Chu trình (Cycles)**

Đường dẫn thường được sử dụng để kiểm tra vòng lặp. Chúng ta bỏ qua nó ở đây vì tic-tac-toe không có chu trình.

**Mục tiêu (Goals)**

Đạt được mục tiêu có nghĩa là tác nhân đã hoàn thành nhiệm vụ thành công.

Đối với các trò chơi, terminal(s) kiểm tra xem s có phải là trạng thái kết thúc hay không, nhưng trạng thái kết thúc cũng có thể chỉ ra rằng người chơi đã thua (điều này chỉ ra một thất bại). Điều này có thể được xác định bởi hàm utility(s). Tic-tac-toe cũng có các trận hòa và chúng ta cần quyết định xem một trận hòa có được coi là đạt được mục tiêu thành công hay là một thất bại.

**Kết quả Không xác định (Non-deterministic Results)**

Vì đối thủ và các nước đi của họ là một phần của môi trường, results() trả về một tập hợp các trạng thái mới cho một kết hợp trạng thái $\times$ hành động, phản ánh tất cả các phản ứng có thể của đối thủ.

**Mã nguồn:**

Python

def results(state, action, player = 'x'):

"""produce the set of states after the provided action for player.

It is the set of boards with the action and all possible reactions by the opponent."""

opponent = other(player)

state = state.copy()

# player's move

state[action] = player

# opponent reacts

r = list()

o\_actions = actions(state)

# board is full

if len(o\_actions) < 1 : return [state]

for o\_a in o\_actions:

s = state.copy()

s[o\_a] = opponent

r.append(s)

return r

show\_board(empty\_board())

print()

print("Set of possible state for placing an x at position 4 of an empty board and the opponent's move:")

results(empty\_board(), 4)

**Kết quả:**

[Hình ảnh: Bàn cờ Tic-Tac-Toe trống]

Set of possible state for placing an x at position 4 of an empty board and the opponent's move:

[['o', ' ', ' ', ' ', 'x', ' ', ' ', ' ', ' '],

[' ', 'o', ' ', ' ', 'x', ' ', ' ', ' ', ' '],

[' ', ' ', 'o', ' ', 'x', ' ', ' ', ' ', ' '],

[' ', ' ', ' ', 'o', 'x', ' ', ' ', ' ', ' '],

[' ', ' ', ' ', ' ', 'x', 'o', ' ', ' ', ' '],

[' ', ' ', ' ', ' ', 'x', ' ', 'o', ' ', ' '],

[' ', ' ', ' ', ' ', 'x', ' ', ' ', 'o', ' '],

[' ', ' ', ' ', ' ', 'x', ' ', ' ', ' ', 'o']]

**Tìm kiếm AND-OR Đệ quy (DFS)**

**Mã nguồn:**

Python

# define global variables for debugging

DEBUG = 1

COUNT = 0 # used to report the number of searched nodes

def and\_or\_search(state, player = 'x', draw\_is\_win = True):

"""start the search. Consider draw\_is\_win a goal state?"""

global DEBUG, COUNT

COUNT = 0

plan = or\_search(state, player, draw\_is\_win)

if DEBUG >= 1:

print(f"Number of nodes searched: {COUNT}")

return plan

def or\_search(state, player, draw\_is\_win):

"""Or step of the search: the player makes a move.

We try all possible action and return a conditional

plan for the first action that only has goal states as leaf nodes.

If none can be found, then failure (None) is returned.

Note: path is not used in this implementation since tic-tac-toe does not

need cycle checking.

"""

global DEBUG, COUNT

COUNT += 1

# goal check

u = utility(state, player)

if u == -1 or (not draw\_is\_win and u == 0): return None # failure

# cycles check: tic-tac-toe has no cycles

# check all possible actions

for action in actions(state):

plan = and\_search(results(state, action, player), player, draw\_is\_win)

if plan is not None:

return [action, plan]

# failure

return None

def and\_search(states, player, draw\_is\_win):

"""And step of the search: Represents all opponent's possible moves.

Follow all possible states (call the or step).

Return a conditional plan only if all paths lead to a goal state."""

global DEBUG, COUNT

COUNT += 1

# return plans only if no state fails

plans = []

for s in states:

# goal check

u = utility(s, player)

if u == 1: return "win" # success

if draw\_is\_win and u == 0: return "draw"

plan = or\_search(s, player, draw\_is\_win)

if plan is None:

return None # found a state that fails so we abandon this subtree!

plans.append(['if', s, 'then', plan])

return plans

Tìm kiếm And-or tìm kiếm một cây con (tức là một hành động) đảm bảo chiến thắng và trả về một kế hoạch có điều kiện cho cây con này. Nếu không tìm thấy cây con nào như vậy, không có kế hoạch nào được trả về. Thuật toán chỉ tìm kiếm một phần của cây trò chơi vì nó từ bỏ cây con hiện tại khi tìm thấy nút lá đầu tiên không phải là chiến thắng (xem hàm and\_search).

**Một số Thử nghiệm**

**# x sắp thắng**

**Mã nguồn:**

Python

board = empty\_board()

board[0] = 'x'

board[1] = 'o'

board[3] = 'o'

board[4] = 'x'

print("Board:")

show\_board(board)

print()

print("Win or draw:")

%time display(and\_or\_search(board, player = 'x', draw\_is\_win = True))

**Kết quả:**

Board:

[Hình ảnh: Bàn cờ với x ở [0, 4] và o ở [1, 3]]

Win or draw:

Number of nodes searched: 22

[2,

[['if', ['x', 'o', 'x', 'o', 'x', 'o', ' ', ' ', ' '], 'then', [6, 'win']],

['if',

['x', 'o', 'x', 'o', 'x', ' ', 'o', ' ', ' '],

'then',

[5,

[['if', ['x', 'o', 'x', 'o', 'x', 'x', 'o', 'o', ' '], 'then', [8, 'win']],

['if',

['x', 'o', 'x', 'o', 'x', 'x', 'o', ' ', 'o'],

'then',

[7, 'draw']]]]],

['if',

['x', 'o', 'x', 'o', 'x', ' ', ' ', 'o', ' '],

'then',

[5,

[['if', ['x', 'o', 'x', 'o', 'x', 'x', 'o', 'o', ' '], 'then', [8, 'win']],

['if',

['x', 'o', 'x', 'o', 'x', 'x', ' ', 'o', 'o'],

'then',

[6, 'win']]]]],

['if',

['x', 'o', 'x', 'o', 'x', ' ', ' ', ' ', 'o'],

'then',

[5,

[['if',

['x', 'o', 'x', 'o', 'x', 'x', 'o', ' ', 'o'],

'then',

[7, 'draw']],

['if',

['x', 'o', 'x', 'o', 'x', 'x', ' ', 'o', 'o'],

'then',

[6, 'win']]]]]]]

CPU times: total: 0 ns

Wall time: 5.55 ms

Tại sao nó chơi 2 mà không phải 8!!!

**Mã nguồn:**

Python

print("Win only:")

%time display(and\_or\_search(board, player = 'x', draw\_is\_win = False))

**Kết quả:**

Win only:

Number of nodes searched: 30

[2,

[['if', ['x', 'o', 'x', 'o', 'x', 'o', ' ', ' ', ' '], 'then', [6, 'win']],

['if', ['x', 'o', 'x', 'o', 'x', ' ', 'o', ' ', ' '], 'then', [8, 'win']],

['if',

['x', 'o', 'x', 'o', 'x', ' ', ' ', 'o', ' '],

'then',

[5,

[['if', ['x', 'o', 'x', 'o', 'x', 'x', 'o', 'o', ' '], 'then', [8, 'win']],

['if',

['x', 'o', 'x', 'o', 'x', 'x', ' ', 'o', 'o'],

'then',

[6, 'win']]]]],

['if', ['x', 'o', 'x', 'o', 'x', ' ', ' ', ' ', 'o'], 'then', [6, 'win']]]]

CPU times: total: 0 ns

Wall time: 5.65 ms

**x có thể hòa\_là\_thắng (draw\_is\_win) nếu chọn 7**

**Mã nguồn:**

Python

board = empty\_board()

board[0] = 'x'

board[1] = 'o'

board[2] = 'x'

#board[3] = 'o'

board[4] = 'o'

print("Board:")

show\_board(board)

print()

print("Win or draw:")

%time display(and\_or\_search(board, player = 'x', draw\_is\_win = True))

**Kết quả:**

Board:

[Hình ảnh: Bàn cờ với x ở [0, 2] và o ở [1, 4]]

Win or draw:

Number of nodes searched: 64

[7,

[['if',

['x', 'o', 'x', 'o', 'o', ' ', ' ', 'x', ' '],

'then',

[5,

[['if', ['x', 'o', 'x', 'o', 'o', 'x', 'o', 'x', ' '], 'then', [8, 'win']],

['if',

['x', 'o', 'x', 'o', 'o', 'x', ' ', 'x', 'o'],

'then',

[6, 'draw']]]]],

['if',

['x', 'o', 'x', ' ', 'o', 'o', ' ', 'x', ' '],

'then',

[3,

[['if',

['x', 'o', 'x', 'x', 'o', 'o', 'o', 'x', ' '],

'then',

[8, 'draw']],

['if',

['x', 'o', 'x', 'x', 'o', 'o', ' ', 'x', 'o'],

'then',

[6, 'win']]]]],

['if',

['x', 'o', 'x', ' ', 'o', ' ', 'o', 'x', ' '],

'then',

[3,

[['if',

['x', 'o', 'x', 'x', 'o', 'o', 'o', 'x', ' '],

'then',

[8, 'draw']],

['if',

['x', 'o', 'x', 'x', 'o', ' ', 'o', 'x', 'o'],

'then',

[5, 'draw']]]]],

['if',

['x', 'o', 'x', ' ', 'o', ' ', ' ', 'x', 'o'],

'then',

[3,

[['if', ['x', 'o', 'x', 'x', 'o', 'o', ' ', 'x', 'o'], 'then', [6, 'win']],

['if',

['x', 'o', 'x', 'x', 'o', ' ', 'o', 'x', 'o'],

'then',

[5, 'draw']]]]]]]

CPU times: total: 0 ns

Wall time: 6.63 ms

**Mã nguồn:**

Python

print("Win only:")

%time display(and\_or\_search(board, player = 'x', draw\_is\_win = False))

**Kết quả:**

Win only:

Number of nodes searched: 69

None

CPU times: total: 0 ns

Wall time: 6.15 ms

**o sắp thắng**

**Mã nguồn:**

Python

board = empty\_board()

board[0] = 'o'

board[1] = 'o'

board[3] = 'o'

board[4] = 'x'

board[8] = 'x'

print("Board:")

show\_board(board)

print()

print("Win or draw:")

%time display(and\_or\_search(board, player = 'x', draw\_is\_win = True))

**Kết quả:**

Board:

[Hình ảnh: Bàn cờ với o ở [0, 1, 3] và x ở [4, 8]]

Win or draw:

Number of nodes searched: 11

None

CPU times: total: 0 ns

Wall time: 1.52 ms

Kiểm tra cho người chơi o

**Mã nguồn:**

Python

print("Win only:")

%time display(and\_or\_search(board, player = 'o', draw\_is\_win = False))

**Kết quả:**

Win only:

Number of nodes searched: 2

[2, 'win']

CPU times: total: 0 ns

Wall time: 2.09 ms

**Bàn cờ trống: Chỉ có thể đảm bảo hòa\_là\_thắng (draw\_is\_win)**

**Mã nguồn:**

Python

board = empty\_board()

print("Board:")

show\_board(board)

print()

print("Win or draw:")

%time display(and\_or\_search(board, player = 'x', draw\_is\_win = True))

**Kết quả:**

Board:

[Hình ảnh: Bàn cờ Tic-Tac-Toe trống]

Win or draw:

Number of nodes searched: 966

[0,

[['if',

['x', 'o', ' ', ' ', ' ', ' ', ' ', ' ', ' '],

'then',

[2,

[['if',

['x', 'o', 'x', 'o', ' ', ' ', ' ', ' ', ' '],

'then',

[4,

[['if',

['x', 'o', 'x', 'o', 'x', 'o', ' ', ' ', ' '],

'then',

[6, 'win']],

['if',

['x', 'o', 'x', 'o', 'x', ' ', 'o', ' ', ' '],

'then',

[5,

[['if',

['x', 'o', 'x', 'o', 'x', 'x', 'o', 'o', ' '],

'then',

[8, 'win']],

['if',

['x', 'o', 'x', 'o', 'x', 'x', 'o', ' ', 'o'],

'then',

[7, 'draw']]]]],

['if',

['x', 'o', 'x', 'o', 'x', ' ', ' ', 'o', ' '],

'then',

[5,

[['if',

['x', 'o', 'x', 'o', 'x', 'x', 'o', 'o', ' '],

'then',

[8, 'win']],

['if',

['x', 'o', 'x', 'o', 'x', 'x', ' ', 'o', 'o'],

'then',

[6, 'win']]]]],

['if',

['x', 'o', 'x', 'o', 'x', ' ', ' ', ' ', 'o'],

'then',

[5,

[['if',

['x', 'o', 'x', 'o', 'x', 'x', 'o', ' ', 'o'],

'then',

[7, 'draw']],

['if',

['x', 'o', 'x', 'o', 'x', 'x', ' ', 'o', 'o'],

'then',

[6, 'win']]]]]]]],

['if',

['x', 'o', 'x', ' ', 'o', ' ', ' ', ' ', ' '],

'then',

[7,

[['if',

['x', 'o', 'x', 'o', 'o', ' ', ' ', 'x', ' '],

'then',

[5,

[['if',

['x', 'o', 'x', 'o', 'o', 'x', 'o', 'x', ' '],

'then',

[8, 'win']],

['if',

['x', 'o', 'x', 'o', 'o', 'x', ' ', 'x', 'o'],

'then',

[6, 'draw']]]]],

['if',

['x', 'o', 'x', ' ', 'o', 'o', ' ', 'x', ' '],

'then',

[3,

[['if',

['x', 'o', 'x', 'x', 'o', 'o', 'o', 'x', ' '],

'then',

[8, 'draw']],

['if',

['x', 'o', 'x', 'x', 'o', 'o', ' ', 'x', 'o'],

'then',

[6, 'win']]]]],

['if',

['x', 'o', 'x', ' ', 'o', ' ', 'o', 'x', ' '],

'then',

[3,

[['if',

['x', 'o', 'x', 'x', 'o', 'o', 'o', 'x', ' '],

'then',

[8, 'draw']],

['if',

['x', 'o', 'x', 'x', 'o', ' ', 'o', 'x', 'o'],

'then',

[5, 'draw']]]]],

['if',

['x', 'o', 'x', ' ', 'o', ' ', ' ', 'x', 'o'],

'then',

[3,

[['if',

['x', 'o', 'x', 'x', 'o', 'o', ' ', 'x', 'o'],

'then',

[6, 'win']],

['if',

['x', 'o', 'x', 'x', 'o', ' ', 'o', 'x', 'o'],

'then',

[5, 'draw']]]]]]]],

['if',

['x', 'o', 'x', ' ', ' ', 'o', ' ', ' ', ' '],

'then',

[3,

[['if',

['x', 'o', 'x', 'x', 'o', 'o', ' ', ' ', ' '],

'then',

[6, 'win']],

['if',

['x', 'o', 'x', 'x', ' ', 'o', 'o', ' ', ' '],

'then',

[4,

[['if',

['x', 'o', 'x', 'x', 'x', 'o', 'o', 'o', ' '],

'then',

[8, 'win']],

['if',

['x', 'o', 'x', 'x', 'x', 'o', 'o', ' ', 'o'],

'then',

[7, 'draw']]]]],

['if',

['x', 'o', 'x', 'x', ' ', 'o', ' ', 'o', ' '],

'then',

[4,

[['if',

['x', 'o', 'x', 'x', 'x', 'o', 'o', 'o', ' '],

'then',

[8, 'win']],

['if',

['x', 'o', 'x', 'x', 'x', 'o', ' ', 'o', 'o'],

'then',

[6, 'win']]]]],

['if',

['x', 'o', 'x', 'x', ' ', 'o', ' ', ' ', 'o'],

'then',

[4,

[['if',

['x', 'o', 'x', 'x', 'x', 'o', 'o', ' ', 'o'],

'then',

[7, 'draw']],

['if',

['x', 'o', 'x', 'x', 'x', 'o', ' ', 'o', 'o'],

'then',

[6, 'win']]]]]]]],

['if',

['x', 'o', 'x', ' ', ' ', ' ', 'o', ' ', ' '],

'then',

[4,

[['if',

['x', 'o', 'x', 'o', 'x', ' ', 'o', ' ', ' '],

'then',

[5,

[['if',

['x', 'o', 'x', 'o', 'x', 'x', 'o', 'o', ' '],

'then',

[8, 'win']],

['if',

['x', 'o', 'x', 'o', 'x', 'x', 'o', ' ', 'o'],

'then',

[7, 'draw']]]]],

['if',

['x', 'o', 'x', ' ', 'x', 'o', 'o', ' ', ' '],

'then',

[3,

[['if',

['x', 'o', 'x', 'x', 'x', 'o', 'o', 'o', ' '],

'then',

[8, 'win']],

['if',

['x', 'o', 'x', 'x', 'x', 'o', 'o', ' ', 'o'],

'then',

[7, 'draw']]]]],

['if',

['x', 'o', 'x', ' ', 'x', ' ', 'o', 'o', ' '],

'then',

[8, 'win']],

['if',

['x', 'o', 'x', ' ', 'x', ' ', 'o', ' ', 'o'],

'then',

[7,

[['if',

['x', 'o', 'x', 'o', 'x', ' ', 'o', 'x', 'o'],

'then',

[5, 'draw']],

['if',

['x', 'o', 'x', ' ', 'x', 'o', 'o', 'x', 'o'],

'then',

[3, 'draw']]]]]]]],

['if',

['x', 'o', 'x', ' ', ' ', ' ', ' ', 'o', ' '],

'then',

[4,

[['if',

['x', 'o', 'x', 'o', 'x', ' ', ' ', 'o', ' '],

'then',

[5,

[['if',

['x', 'o', 'x', 'o', 'x', 'x', 'o', 'o', ' '],

'then',

[8, 'win']],

['if',

['x', 'o', 'x', 'o', 'x', 'x', ' ', 'o', 'o'],

'then',

[6, 'win']]]]],

['if',

['x', 'o', 'x', ' ', 'x', 'o', ' ', 'o', ' '],

'then',

[3,

[['if',

['x', 'o', 'x', 'x', 'x', 'o', 'o', 'o', ' '],

'then',

[8, 'win']],

['if',

['x', 'o', 'x', 'x', 'x', 'o', ' ', 'o', 'o'],

'then',

[6, 'win']]]]],

['if',

['x', 'o', 'x', ' ', 'x', ' ', 'o', 'o', ' '],

'then',

[8, 'win']],

['if',

['x', 'o', 'x', ' ', 'x', ' ', ' ', 'o', 'o'],

'then',

[6, 'win']]]]],

['if',

['x', 'o', 'x', ' ', ' ', ' ', ' ', ' ', 'o'],

'then',

[3,

[['if',

['x', 'o', 'x', 'x', 'o', ' ', ' ', ' ', 'o'],

'then',

[6, 'win']],

['if',

['x', 'o', 'x', 'x', ' ', 'o', ' ', ' ', 'o'],

'then',

[4,

[['if',

['x', 'o', 'x', 'x', 'x', 'o', 'o', ' ', 'o'],

'then',

[7, 'draw']],

['if',

['x', 'o', 'x', 'x', 'x', 'o', ' ', 'o', 'o'],

'then',

[6, 'win']]]]],

['if',

['x', 'o', 'x', 'x', ' ', ' ', 'o', ' ', 'o'],

'then',

[7,

[['if',

['x', 'o', 'x', 'x', 'o', ' ', 'o', 'x', 'o'],

'then',

[5, 'draw']],

['if',

['x', 'o', 'x', 'x', ' ', 'o', 'o', 'x', 'o'],

'then',

[4, 'draw']]]]],

['if',

['x', 'o', 'x', 'x', ' ', ' ', ' ', 'o', 'o'],

'then',

[6, 'win']]]]]]]],

['if',

['x', ' ', 'o', ' ', ' ', ' ', ' ', ' ', ' '],

'then',

[3,

[['if',

['x', 'o', 'o', 'x', ' ', ' ', ' ', ' ', ' '],

'then',

[4,

[['if',

['x', 'o', 'o', 'x', 'x', 'o', ' ', ' ', ' '],

'then',

[6, 'win']],

['if',

['x', 'o', 'o', 'x', 'x', ' ', 'o', ' ', ' '],

'then',

[5, 'win']],

['if',

['x', 'o', 'o', 'x', 'x', ' ', ' ', 'o', ' '],

'then',

[5, 'win']],

['if',

['x', 'o', 'o', 'x', 'x', ' ', ' ', ' ', 'o'],

'then',

[5, 'win']]]]],

['if', ['x', ' ', 'o', 'x', 'o', ' ', ' ', ' ', ' '], 'then', [6, 'win']],

['if', ['x', ' ', 'o', 'x', ' ', 'o', ' ', ' ', ' '], 'then', [6, 'win']],

['if',

['x', ' ', 'o', 'x', ' ', ' ', 'o', ' ', ' '],

'then',

[4,

[['if',

['x', 'o', 'o', 'x', 'x', ' ', 'o', ' ', ' '],

'then',

[5, 'win']],

['if',

['x', ' ', 'o', 'x', 'x', 'o', 'o', ' ', ' '],

'then',

[8, 'win']],

['if',

['x', ' ', 'o', 'x', 'x', ' ', 'o', 'o', ' '],

'then',

[5, 'win']],

['if',

['x', ' ', 'o', 'x', 'x', ' ', 'o', ' ', 'o'],

'then',

[5, 'win']]]]],

['if',

['x', ' ', 'o', 'x', ' ', ' ', ' ', 'o', ' '],

'then',

[4,

[['if',

['x', 'o', 'o', 'x', 'x', ' ', ' ', 'o', ' '],

'then',

[5, 'win']],

['if',

['x', ' ', 'o', 'x', 'x', 'o', ' ', 'o', ' '],

'then',

[6, 'win']],

['if',

['x', ' ', 'o', 'x', 'x', ' ', 'o', 'o', ' '],

'then',

[5, 'win']],

['if',

['x', ' ', 'o', 'x', 'x', ' ', ' ', 'o', 'o'],

'then',

[5, 'win']]]]],

['if',

['x', ' ', 'o', 'x', ' ', ' ', ' ', ' ', 'o'],

'then',

[5,

[['if',

['x', 'o', 'o', 'x', ' ', 'x', ' ', ' ', 'o'],

'then',

[4, 'win']],

['if',

['x', ' ', 'o', 'x', 'o', 'x', ' ', ' ', 'o'],

'then',

[6, 'win']],

['if',

['x', ' ', 'o', 'x', ' ', 'x', 'o', ' ', 'o'],

'then',

[4, 'win']],

['if',

['x', ' ', 'o', 'x', ' ', 'x', ' ', 'o', 'o'],

'then',

[4, 'win']]]]]]]],

['if',

['x', ' ', ' ', 'o', ' ', ' ', ' ', ' ', ' '],

'then',

[1,

[['if',

['x', 'x', 'o', 'o', ' ', ' ', ' ', ' ', ' '],

'then',

[4,

[['if',

['x', 'x', 'o', 'o', 'x', 'o', ' ', ' ', ' '],

'then',

[7, 'win']],

['if',

['x', 'x', 'o', 'o', 'x', ' ', 'o', ' ', ' '],

'then',

[5,

[['if',

['x', 'x', 'o', 'o', 'x', 'x', 'o', 'o', ' '],

'then',

[8, 'win']],

['if',

['x', 'x', 'o', 'o', 'x', 'x', 'o', ' ', 'o'],

'then',

[7, 'win']]]]],

['if',

['x', 'x', 'o', 'o', 'x', ' ', ' ', 'o', ' '],

'then',

[5,

[['if',

['x', 'x', 'o', 'o', 'x', 'x', 'o', 'o', ' '],

'then',

[8, 'win']],

['if',

['x', 'x', 'o', 'o', 'x', 'x', ' ', 'o', 'o'],

'then',

[6, 'draw']]]]],

['if',

['x', 'x', 'o', 'o', 'x', ' ', ' ', ' ', 'o'],

'then',

[5,

[['if',

['x', 'x', 'o', 'o', 'x', 'x', 'o', ' ', 'o'],

'then',

[7, 'win']],

['if',

['x', 'x', 'o', 'o', 'x', 'x', ' ', 'o', 'o'],

'then',

[6, 'draw']]]]]]]],

['if', ['x', 'x', ' ', 'o', 'o', ' ', ' ', ' ', ' '], 'then', [2, 'win']],

['if', ['x', 'x', ' ', 'o', ' ', 'o', ' ', ' ', ' '], 'then', [2, 'win']],

['if', ['x', 'x', ' ', 'o', ' ', ' ', 'o', ' ', ' '], 'then', [2, 'win']],

['if', ['x', 'x', ' ', 'o', ' ', ' ', ' ', 'o', ' '], 'then', [2, 'win']],

['if',

['x', 'x', ' ', 'o', ' ', ' ', ' ', ' ', 'o'],

'then',

[2, 'win']]]]],

['if',

['x', ' ', ' ', ' ', 'o', ' ', ' ', ' ', ' '],

'then',

[1,

[['if',

['x', 'x', 'o', ' ', 'o', ' ', ' ', ' ', ' '],

'then',

[6,

[['if',

['x', 'x', 'o', 'o', 'o', ' ', 'x', ' ', ' '],

'then',

[5,

[['if',

['x', 'x', 'o', 'o', 'o', 'x', 'x', 'o', ' '],

'then',

[8, 'draw']],

['if',

['x', 'x', 'o', 'o', 'o', 'x', 'x', ' ', 'o'],

'then',

[7, 'draw']]]]],

['if',

['x', 'x', 'o', ' ', 'o', 'o', 'x', ' ', ' '],

'then',

[3, 'win']],

['if',

['x', 'x', 'o', ' ', 'o', ' ', 'x', 'o', ' '],

'then',

[3, 'win']],

['if',

['x', 'x', 'o', ' ', 'o', ' ', 'x', ' ', 'o'],

'then',

[3, 'win']]]]],

['if', ['x', 'x', ' ', 'o', 'o', ' ', ' ', ' ', ' '], 'then', [2, 'win']],

['if', ['x', 'x', ' ', ' ', 'o', 'o', ' ', ' ', ' '], 'then', [2, 'win']],

['if', ['x', 'x', ' ', ' ', 'o', ' ', 'o', ' ', ' '], 'then', [2, 'win']],

['if', ['x', 'x', ' ', ' ', 'o', ' ', ' ', 'o', ' '], 'then', [2, 'win']],

['if',

['x', 'x', ' ', ' ', 'o', ' ', ' ', ' ', 'o'],

'then',

[2, 'win']]]]],

['if',

['x', ' ', ' ', ' ', ' ', 'o', ' ', ' ', ' '],

'then',

[2,

[['if',

['x', 'o', 'x', ' ', ' ', 'o', ' ', ' ', ' '],

'then',

[3,

[['if',

['x', 'o', 'x', 'x', 'o', 'o', ' ', ' ', ' '],

'then',

[6, 'win']],

['if',

['x', 'o', 'x', 'x', ' ', 'o', 'o', ' ', ' '],

'then',

[4,

[['if',

['x', 'o', 'x', 'x', 'x', 'o', 'o', 'o', ' '],

'then',

[8, 'win']],

['if',

['x', 'o', 'x', 'x', 'x', 'o', 'o', ' ', 'o'],

'then',

[7, 'draw']]]]],

['if',

['x', 'o', 'x', 'x', ' ', 'o', ' ', 'o', ' '],

'then',

[4,

[['if',

['x', 'o', 'x', 'x', 'x', 'o', 'o', 'o', ' '],

'then',

[8, 'win']],

['if',

['x', 'o', 'x', 'x', 'x', 'o', ' ', 'o', 'o'],

'then',

[6, 'win']]]]],

['if',

['x', 'o', 'x', 'x', ' ', 'o', ' ', ' ', 'o'],

'then',

[4,

[['if',

['x', 'o', 'x', 'x', 'x', 'o', 'o', ' ', 'o'],

'then',

[7, 'draw']],

['if',

['x', 'o', 'x', 'x', 'x', 'o', ' ', 'o', 'o'],

'then',

[6, 'win']]]]]]]],

['if', ['x', ' ', 'x', 'o', ' ', 'o', ' ', ' ', ' '], 'then', [1, 'win']],

['if', ['x', ' ', 'x', ' ', 'o', 'o', ' ', ' ', ' '], 'then', [1, 'win']],

['if', ['x', ' ', 'x', ' ', ' ', 'o', 'o', ' ', ' '], 'then', [1, 'win']],

['if', ['x', ' ', 'x', ' ', ' ', 'o', ' ', 'o', ' '], 'then', [1, 'win']],

['if',

['x', ' ', 'x', ' ', ' ', 'o', ' ', ' ', 'o'],

'then',

[1, 'win']]]]],

['if',

['x', ' ', ' ', ' ', ' ', ' ', 'o', ' ', ' '],

'then',

[1,

[['if',

['x', 'x', 'o', ' ', ' ', ' ', 'o', ' ', ' '],

'then',

[4,

[['if',

['x', 'x', 'o', 'o', 'x', ' ', 'o', ' ', ' '],

'then',

[5,

[['if',

['x', 'x', 'o', 'o', 'x', 'x', 'o', 'o', ' '],

'then',

[8, 'win']],

['if',

['x', 'x', 'o', 'o', 'x', 'x', 'o', ' ', 'o'],

'then',

[7, 'win']]]]],

['if',

['x', 'x', 'o', ' ', 'x', 'o', 'o', ' ', ' '],

'then',

[7, 'win']],

['if',

['x', 'x', 'o', ' ', 'x', ' ', 'o', 'o', ' '],

'then',

[8, 'win']],

['if',

['x', 'x', 'o', ' ', 'x', ' ', 'o', ' ', 'o'],

'then',

[7, 'win']]]]],

['if', ['x', 'x', ' ', 'o', ' ', ' ', 'o', ' ', ' '], 'then', [2, 'win']],

['if', ['x', 'x', ' ', ' ', 'o', ' ', 'o', ' ', ' '], 'then', [2, 'win']],

['if', ['x', 'x', ' ', ' ', ' ', 'o', 'o', ' ', ' '], 'then', [2, 'win']],

['if', ['x', 'x', ' ', ' ', ' ', ' ', 'o', 'o', ' '], 'then', [2, 'win']],

['if',

['x', 'x', ' ', ' ', ' ', ' ', 'o', ' ', 'o'],

'then',

[2, 'win']]]]],

['if',

['x', ' ', ' ', ' ', ' ', ' ', ' ', 'o', ' '],

'then',

[1,

[['if',

['x', 'x', 'o', ' ', ' ', ' ', ' ', 'o', ' '],

'then',

[6,

[['if',

['x', 'x', 'o', 'o', ' ', ' ', 'x', 'o', ' '],

'then',

[4,

[['if',

['x', 'x', 'o', 'o', 'x', 'o', 'x', 'o', ' '],

'then',

[8, 'win']],

['if',

['x', 'x', 'o', 'o', 'x', ' ', 'x', 'o', 'o'],

'then',

[5, 'draw']]]]],

['if',

['x', 'x', 'o', ' ', 'o', ' ', 'x', 'o', ' '],

'then',

[3, 'win']],

['if',

['x', 'x', 'o', ' ', ' ', 'o', 'x', 'o', ' '],

'then',

[3, 'win']],

['if',

['x', 'x', 'o', ' ', ' ', ' ', 'x', 'o', 'o'],

'then',

[3, 'win']]]]],

['if', ['x', 'x', ' ', 'o', ' ', ' ', ' ', 'o', ' '], 'then', [2, 'win']],

['if', ['x', 'x', ' ', ' ', 'o', ' ', ' ', 'o', ' '], 'then', [2, 'win']],

['if', ['x', 'x', ' ', ' ', ' ', 'o', ' ', 'o', ' '], 'then', [2, 'win']],

['if', ['x', 'x', ' ', ' ', ' ', ' ', 'o', 'o', ' '], 'then', [2, 'win']],

['if',

['x', 'x', ' ', ' ', ' ', ' ', ' ', 'o', 'o'],

'then',

[2, 'win']]]]],

['if',

['x', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'o'],

'then',

[2,

[['if',

['x', 'o', 'x', ' ', ' ', ' ', ' ', ' ', 'o'],

'then',

[3,

[['if',

['x', 'o', 'x', 'x', 'o', ' ', ' ', ' ', 'o'],

'then',

[6, 'win']],

['if',

['x', 'o', 'x', 'x', ' ', 'o', ' ', ' ', 'o'],

'then',

[4,

[['if',

['x', 'o', 'x', 'x', 'x', 'o', 'o', ' ', 'o'],

'then',

[7, 'draw']],

['if',

['x', 'o', 'x', 'x', 'x', 'o', ' ', 'o', 'o'],

'then',

[6, 'win']]]]],

['if',

['x', 'o', 'x', 'x', ' ', ' ', 'o', ' ', 'o'],

'then',

[7,

[['if',

['x', 'o', 'x', 'x', 'o', ' ', 'o', 'x', 'o'],

'then',

[5, 'draw']],

['if',

['x', 'o', 'x', 'x', ' ', 'o', 'o', 'x', 'o'],

'then',

[4, 'draw']]]]],

['if',

['x', 'o', 'x', 'x', ' ', ' ', ' ', 'o', 'o'],

'then',

[6, 'win']]]]],

['if', ['x', ' ', 'x', 'o', ' ', ' ', ' ', ' ', 'o'], 'then', [1, 'win']],

['if', ['x', ' ', 'x', ' ', 'o', ' ', ' ', ' ', 'o'], 'then', [1, 'win']],

['if', ['x', ' ', 'x', ' ', ' ', 'o', ' ', ' ', 'o'], 'then', [1, 'win']],

['if', ['x', ' ', 'x', ' ', ' ', ' ', 'o', ' ', 'o'], 'then', [1, 'win']],

['if',

['x', ' ', 'x', ' ', ' ', ' ', ' ', 'o', 'o'],

'then',

[1, 'win']]]]]]]

CPU times: total: 46.9 ms

Wall time: 59.9 ms

**Mã nguồn:**

Python

print("Win only:")

%time display(and\_or\_search(board, player = 'x', draw\_is\_win = False))

**Kết quả:**

Win only:

Number of nodes searched: 16213

None

CPU times: total: 484 ms

Wall time: 469 ms

**Thử nghiệm**

**Tìm kiếm Cây And-Or vs. Ngẫu nhiên**

Đặt tìm kiếm AND-OR vào một hàm tác nhân bao bọc. Lưu ý rằng nếu tìm kiếm AND-OR không thể đảm bảo chiến thắng thì nó không trả về kế hoạch, mà là None. Trong trường hợp này, chúng ta không biết nước đi tốt nhất (tức là "ít tệ nhất") là gì, vì vậy tôi để tác nhân chơi ngẫu nhiên. Các phương pháp khác mà chúng ta sẽ tìm hiểu sau này có thể xác định tiện ích của một nước đi, ngay cả khi nó không đảm bảo chiến thắng.

**Mã nguồn:**

Python

DEBUG = 0

def and\_or\_player(board, player = 'x'):

plan = and\_or\_search(board, player)

# if there is no plan then we do a random action

if plan is None:

return np.random.choice(actions(board))

else:

return plan[0]

print("and-or-search vs. random:")

%time display(play(and\_or\_player, random\_player))

print()

print("random vs. and-or-search")

%time display(play(random\_player, and\_or\_player))

**Kết quả:**

and-or-search vs. random:

{'x': 90, 'o': 0, 'd': 10}

CPU times: total: 3.42 s

Wall time: 3.44 s

random vs. and-or-search

{'x': 0, 'o': 71, 'd': 29}

CPU times: total: 4.8 s

Wall time: 4.84 s

**Lưu ý rằng điều này chậm như thế nào!**

Việc triển khai lười biếng của tôi là một *tác nhân phản xạ đơn giản* (simple reflex agent) tái tạo một kế hoạch có điều kiện mỗi khi nó được yêu cầu một hành động. Nhưng vì tìm kiếm And-Or đã tạo ra một kế hoạch có điều kiện hoàn chỉnh cho bàn cờ trống, một triển khai tốt hơn sẽ sử dụng một *tác nhân phản xạ dựa trên mô hình* (model-based reflex agent) lưu trữ kế hoạch được tính toán trước này và chỉ cần tuân theo nó mà không cần lập kế hoạch thêm.

**Tìm kiếm Cây And-Or vs. Tìm kiếm Cây And-Or**

**Mã nguồn:**

Python

# No randomness -> run only once

%time display(play(and\_or\_player, and\_or\_player, N = 1))

**Kết quả:**

{'x': 0, 'o': 0, 'd': 1}

CPU times: total: 93.8 ms

Wall time: 96.2 ms