

**BÀI TẬP LAB 3**  
**Môn: Cơ sở trí tuệ nhân tạo (HK2 2024-2025) – Tuần 8**

**I. Thông tin sinh viên**

Họ và tên: Phan Ngọc Hà

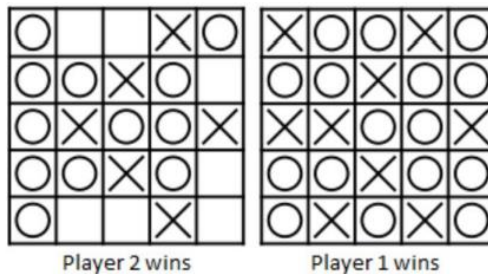
MSSV: 23880020

**II. Bài làm**

**Bài 1:**

**Đề bài:**

- Đọc hiểu và trình bày lại theo cách hiểu của mình, diễn giải kỹ hơn ở mỗi bước và ánh xạ đến kiến thức lý thuyết đã được học.
- Dựa trên code có sẵn để cài đặt chạy trên máy của mình.
- Điều chỉnh code để chấp nhận input bất kỳ (trạng thái ban đầu bất kỳ). Ví dụ, đã có sẵn một số bước đã thực hiện.
- Thực hiện trên input có giới hạn lớn hơn như 10x10 hay 20 x 20 với luật chơi có thể thay đổi như 5 ô liên tiếp, ...



**Bài làm:**

**Bài 1:**

**Ý 1:**

Khai báo các biến và hàm cần thiết:

```
from math import inf as infinity
from random import choice
import platform
import time
from os import system
```

```
HUMAN = -1
COMP = +1
board = [
    [0, 0, 0],
    [0, 0, 0],
    [0, 0, 0],
]
```

```
def evaluate(state):
    if wins(state, COMP):
        score = +1
    elif wins(state, HUMAN):
        score = -1
    else:
        score = 0

    return score
```

Khai báo biến:

- \*HUMAN là 1: đại diện cho con người
- \*COMP là +1: đại diện cho máy tính
- \*Ma trận dạng 3x3: thể hiện bảng chơi caro

Hàm tính điểm ứng với trạng thái hiện tại:

Hàm evaluate truyền vào param state là trạng thái hiện tại của bàn cờ.

Gọi tới hàm wins và kiểm tra để:

- \*gán score = + 1 : nếu máy tính thắng
- \*gán score = -1: nếu con người thắng

\*gán score = 0: nếu hòa

Trả về score

```

25 def wins(state, player):
26     win_state = [
27         [state[0][0], state[0][1], state[0][2]],
28         [state[1][0], state[1][1], state[1][2]],
29         [state[2][0], state[2][1], state[2][2]],
30         [state[0][0], state[1][0], state[2][0]],
31         [state[0][1], state[1][1], state[2][1]],
32         [state[0][2], state[1][2], state[2][2]],
33         [state[0][0], state[1][1], state[2][2]],
34         [state[2][0], state[1][1], state[0][2]],
35     ]
36     if [player, player, player] in win_state:
37         return True
38     else:
39         return False

```

Hàm kiểm tra thắng trò chơi:  
Hàm wins truyền vào 2 param  
\*state: trạng thái của bàn cờ hiện tại  
\*player: người chơi hoặc máy tính  
Ứng với 3 khả năng thắng:  
\* Ba hàng [X X X] hoặc [O O O]  
\* Ba cột [X X X] hoặc [O O O]  
\* Hai đường chéo [X X X] hoặc [O O O]  
Đưa tất cả các trường hợp thắng này vào mảng win\_state  
Nếu [player,player,player] thuộc mảng thì return true, nghĩa là đã có người thắng

```

41
42 def game_over(state):
43     return wins(state, HUMAN) or wins(state, COMP)
44

```

Hàm kiểm tra trò chơi kết thúc:  
Nếu người hoặc máy thắng thì trò chơi kết thúc, nghĩa là hàm wins trả về True thì hàm game\_over trả về true

```

45
46 def empty_cells(state):
47     cells = []
48
49     for x, row in enumerate(state):
50         for y, cell in enumerate(row):
51             if cell == 0:
52                 cells.append([x, y])
53
54     return cells

```

Hàm empty\_cells(state) có nhiệm vụ tìm và trả về danh sách tọa độ của các ô trống (ô có giá trị bằng 0) trong một ma trận state.  
\* enumerate(state) giúp duyệt qua từng hàng (row) của ma trận state, đồng thời lấy chỉ số hàng x cột y.  
\* Nếu giá trị của ô bằng 0, Thêm vị trí [x, y] vào danh sách cells. nghĩa là nó trống.

```

55
56
57 def valid_move(x, y):
58     if [x, y] in empty_cells(board):
59         return True
60     else:
61         return False
62

```

Hàm valid\_move truyền vào một tọa độ (x,y) sau đó kiểm tra có trống không (nằm trong danh sách mà hàm empty\_cell trả về). Nếu trống thì trả về true

```

63
64 def set_move(x, y, player):
65     if valid_move(x, y):
66         board[x][y] = player
67         return True
68     else:
69         return False
70

```

Hàm set\_move truyền vào một tọa độ (x,y) và player sau đó kiểm tra nếu bước đi là valid nghĩa là đi trong ô trống thì gán tọa độ trong board ứng với player, rồi trả về true. Ngược lại trả về false.

Hàm sử dụng thuật toán minimax:

```
72 def minimax(state, depth, player):
73     if player == COMP:
74         best = [-1, -1, -infinity]
75     else:
76         best = [-1, -1, +infinity]
77
78     if depth == 0 or game_over(state):
79         score = evaluate(state)
80         return [-1, -1, score]
81
82     for cell in empty_cells(state):
83         x, y = cell[0], cell[1]
84         state[x][y] = player
85         score = minimax(state, depth - 1, -player)
86         state[x][y] = 0
87         score[0], score[1] = x, y
88
89         if player == COMP:
90             if score[2] > best[2]:
91                 best = score # max value
92             else:
93                 if score[2] < best[2]:
94                     best = score # min value
95
96     return best
```

Hàm minimax truyền vào state, depth, player:  
\*Nếu player hiện tại là Computer thì tìm MAX nên khai báo -infinity,  
\*Nếu player hiện tại là Human thì tìm MIN nên khai báo +infinity

Điều kiện dừng đệ quy:  
nếu depth == 0 (đạt đến độ sâu tối đa) **hoặc** trò chơi đã kết thúc (game\_over(state) == True), ta đánh giá trạng thái bàn cờ bằng evaluate(state).

\*Duyệt tất cả các ô trống và thực hiện nước đi  
\*Gọi đệ quy để tìm giá trị tốt nhất cho trạng thái tiếp theo, với:  
depth - 1: Giảm độ sâu tìm kiếm. Thử từng vị trí  
-player: Đổi lượt chơi (ví dụ, nếu player = COMP = +1, lượt sau sẽ là HUMAN = -1).  
\*Trả lại trạng thái ban đầu (state[x][y] = 0) sau khi thử nước đi.  
\*Cập nhật vị trí của nước đi: Lưu lại vị trí (x, y) của nước đi hiện tại.

\*Chọn nước đi tối ưu dựa trên Minimax  
\*Nếu **lượt đi của máy (COMP)**, ta **chọn giá trị lớn nhất** (nước đi có điểm số cao nhất).  
\*Nếu **lượt đi của người chơi (HUMAN)**, ta **chọn giá trị nhỏ nhất** (nước đi có điểm số thấp nhất).  
\*Thuật toán Minimax:  
+ Máy chơi tối ưu (tối đa điểm số).  
+ Người chơi giả định cũng chơi tối ưu (tối thiểu điểm số của máy).  
\*Trả về nước đi tốt nhất

In ra màn hình và hàm main():

```
99 def clean():
100     os_name = platform.system().lower()
101     if 'windows' in os_name:
102         system('cls')
103     else:
104         system('clear')
105
106
107 def render(state, c_choice, h_choice):
108     chars = {
109         -1: h_choice,
110         +1: c_choice,
111         0: ' '
112     }
113     str_line = '-----'
114
115     print('\n' + str_line)
116     for row in state:
117         for cell in row:
118             symbol = chars[cell]
119             print(f'| {symbol} |', end='')
120     print('\n' + str_line)
```

Xóa console

In bảng trò chơi

```

123 def ai_turn(c_choice, h_choice):
124
125     depth = len(empty_cells(board))
126     if depth == 0 or game_over(board):
127         return
128
129     clean()
130     print(f'Computer turn [{c_choice}]')
131     render(board, c_choice, h_choice)
132
133     if depth == 9:
134         x = choice([0, 1, 2])
135         y = choice([0, 1, 2])
136     else:
137         move = minimax(board, depth, COMP)
138         x, y = move[0], move[1]
139
140     set_move(x, y, COMP)
141     time.sleep(1)

```

- \*Tính độ sâu và kiểm tra kết thúc trò chơi
- \*Xóa màn hình và hiển thị thông tin
- \*Máy tính chọn nước đi
- Nếu depth == 9, nghĩa là đây là **nước đi đầu tiên của ván chơi** (bàn cờ hoàn toàn trống).
- Máy tính chọn một ô ngẫu nhiên (x, y) trong các vị trí [0, 1, 2] để đi trước. Choice là hàm random.
- \*Nếu **không phải nước đi đầu tiên**, máy tính dùng **thuật toán Minimax** để tìm nước đi tối ưu nhất.

```

14 def human_turn(c_choice, h_choice):
15     depth = len(empty_cells(board))
16     if depth == 0 or game_over(board):
17         return
18
19     # Dictionary of valid moves
20     move = -1
21     moves = {
22         1: [0, 0], 2: [0, 1], 3: [0, 2],
23         4: [1, 0], 5: [1, 1], 6: [1, 2],
24         7: [2, 0], 8: [2, 1], 9: [2, 2],
25     }
26
27     clean()
28     print(f'Human turn [{h_choice}]')
29     render(board, c_choice, h_choice)
30
31     while move < 1 or move > 9:
32         try:
33             move = int(input('Use numpad (1..9): '))
34             coord = moves[move]
35             can_move = set_move(coord[0], coord[1], HUMAN)
36
37             if not can_move:
38                 print('Bad move')
39                 move = -1
40         except (EOFError, KeyboardInterrupt):
41             print('Bye')
42             exit()
43         except (KeyError, ValueError):
44             print('Bad choice')

```

- \*Tính độ sâu và kiểm tra kết thúc trò chơi
- \*Tạo danh sách nước đi hợp lệ
- \*Xóa màn hình và hiển thị thông tin
- \*Vòng lặp nhập nước đi.
- \*Kiểm tra và thực hiện nước đi
- \*Xử lý lỗi khi nhập

```
def main():
    clean()
    h_choice = '' # X or O
    c_choice = '' # X or O
    first = '' # if human is the first

    # Human chooses X or O to play
    while h_choice != 'O' and h_choice != 'X':
        try:
            print('')
            h_choice = input('Choose X or O\nChosen: ').upper()
        except (EOFError, KeyboardInterrupt):
            print('Bye')
            exit()
        except (KeyError, ValueError):
            print('Bad choice')

    # Setting computer's choice
    if h_choice == 'X':
        c_choice = 'O'
    else:
        c_choice = 'X'
```

- \*Xóa màn hình và khởi tạo biến
- \*Người chơi chọn X hoặc O
- \*Gán ký hiệu cho máy tính

```
# Human may starts first
clean()
while first != 'Y' and first != 'N':
    try:
        first = input('First to start?[y/n]: ').upper()
    except (EOFError, KeyboardInterrupt):
        print('Bye')
        exit()
    except (KeyError, ValueError):
        print('Bad choice')

# Main loop of this game
while len(empty_cells(board)) > 0 and not game_over(board):
    if first == 'N':
        ai_turn(c_choice, h_choice)
        first = ''

    human_turn(c_choice, h_choice)
    ai_turn(c_choice, h_choice)
```

- \*Người chơi quyết định ai đi trước
- \*Chạy vòng lặp chính của trò chơi

```
# Game over message
if wins(board, HUMAN):
    clean()
    print(f'Human turn [{h_choice}]')
    render(board, c_choice, h_choice)
    print('YOU WIN!')
elif wins(board, COMP):
    clean()
    print(f'Computer turn [{c_choice}]')
    render(board, c_choice, h_choice)
    print('YOU LOSE!')
else:
    clean()
    render(board, c_choice, h_choice)
    print('DRAW!')

exit()

if __name__ == '__main__':
    main()
```

- \*Kiểm tra kết quả trò chơi
- \*Thoát chương trình
- \*Chạy chương trình nếu file được thực thi trực tiếp

## Ý 2: Dựa trên code có sẵn để cài đặt chạy trên máy:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS F:\CODE\HCMUS\mon-hoc\17.co-so-tri-tue-nhan-tao\Drive\Code\Lab3-minimax\23880020_py_version> python minimax.py

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Choose X or O
Chosen: X

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
First to start?[y/n]: y

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Human turn [X]

-----
|  |  |  |
|  |  |  |
|  |  |  |
-----
Use numpad (1..9): 1

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Human turn [X]

-----
| X |  |  |
|  | O |  |
|  |  |  |
-----
Use numpad (1..9): 

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Human turn [X]

-----
| X |  |  |
| X | O |  |
| O |  |  |
-----
Use numpad (1..9): 6

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Human turn [X]

-----
| X | O |  |
| X | O | X |
| O |  |  |
-----
Use numpad (1..9): 8

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Computer turn [O]

-----
| X | O | O |
| X | O | X |
| O | X |  |
-----
YOU LOSE!
```

**Ý 3:** Điều chỉnh code để chấp nhận input bất kỳ (trạng thái ban đầu bất kỳ). Ví dụ, đã có sẵn một số bước đã thực hiện:

-Thêm hàm khai báo trạng thái ban đầu:

```
176 def get_initial_board():
177     """
178     Nhập trạng thái ban đầu của bảng từ người dùng.
179     Người dùng nhập 9 số (0: trống, 1: máy, -1: người) theo từng hàng.
180     """
181     print("Nhập trạng thái ban đầu của bảng (9 số, cách nhau bởi dấu cách):")
182     print("0: Ô trống, 1: Máy (O/X), -1: Người (X/O)")
183
184     while True:
185         try:
186             # Nhập 9 số nguyên
187             values = list(map(int, input().split()))
188
189             # Kiểm tra số lượng đúng 9 số
190             if len(values) != 9 or not all(v in [-1, 0, 1] for v in values):
191                 raise ValueError
192
193             # Chuyển danh sách thành ma trận 3x3
194             return [values[:3], values[3:6], values[6:]]
195
196         except ValueError:
197             print("Dữ liệu không hợp lệ. Vui lòng nhập lại!")
```

Chỉnh lại hàm main:

```
199 def main():
200     clean()
201
202     # Nhập trạng thái ban đầu của bảng từ người dùng
203     global board
204     board = get_initial_board()
205
206     # Xác định số lượt đi còn lại
207     depth = len(empty_cells(board))
208
209     # Kiểm tra nếu trò chơi đã kết thúc
210     if game_over(board) or depth == 0:
211         render(board, "O", "X") # Hiển thị trạng thái bảng
212         if wins(board, HUMAN):
213             print("Trò chơi đã kết thúc: Người chơi THẮNG!")
214         elif wins(board, COMP):
215             print("Trò chơi đã kết thúc: Máy TÍNH THẮNG!")
216         else:
217             print("Trò chơi đã kết thúc: HÒA!")
218         exit()
219
220     h_choice = '' # X or O
221     c_choice = '' # X or O
222     first = '' # if human is the first
223
224     # Human chooses X or O to play
225     while h_choice != 'O' and h_choice != 'X':
226         try:
227             print('')
228             h_choice = input('Choose X or O\nChosen: ').upper()
229         except (EOFError, KeyboardInterrupt):
230             print('Bye')
231             exit()
232         except (KeyError, ValueError):
233             print('Bad choice')
234
235     # Setting computer's choice
236     if h_choice == 'X':
237         c_choice = 'O'
238     else:
239         c_choice = 'X'
```

```

241     # Human may starts first
242     clean()
243     while first != 'Y' and first != 'N':
244         try:
245             first = input('First to start?[y/n]: ').upper()
246         except (EOFError, KeyboardInterrupt):
247             print('Bye')
248             exit()
249         except (KeyError, ValueError):
250             print('Bad choice')
251
252     # Main loop of this game
253     while len(empty_cells(board)) > 0 and not game_over(board):
254         if first == 'N':
255             ai_turn(c_choice, h_choice)
256             first = ''
257
258         human_turn(c_choice, h_choice)
259         ai_turn(c_choice, h_choice)

```

```

261     # Game over message
262     if wins(board, HUMAN):
263         clean()
264         print(f'Human turn [{h_choice}]')
265         render(board, c_choice, h_choice)
266         print('YOU WIN!')
267     elif wins(board, COMP):
268         clean()
269         print(f'Computer turn [{c_choice}]')
270         render(board, c_choice, h_choice)
271         print('YOU LOSE!')
272     else:
273         clean()
274         render(board, c_choice, h_choice)
275         print('DRAW!')
276
277     exit()
278
279
280 if __name__ == '__main__':
281     main()

```





**Ý 4:** Thực hiện trên input có giới hạn lớn hơn như 10x10 hay 20 x 20 với luật chơi có thể thay đổi như 5 ô liên tiếp.

```
1 from math import inf as infinity
2 from random import choice
3 import platform
4 import time
5 from os import system
6
7 def get_game_settings():
8     """ Nhập kích thước bảng và số lượng ô liên tiếp cần để thắng """
9     while True:
10         try:
11             # Nhập kích thước bảng và điều kiện thắng
12             size = int(input("Nhập kích thước bảng (ví dụ: 10 cho 10x10): "))
13             win_condition = int(input(f"Nhập số ô liên tiếp cần để thắng (tối đa {size}): "))
14
15             # Kiểm tra thông tin nhập vào
16             if size < 3 or win_condition > size or win_condition < 3:
17                 print("Thông tin không hợp lệ! Hãy nhập lại.")
18                 continue
19
20             return size, win_condition
21
22         except ValueError:
23             print("Vui lòng nhập số nguyên hợp lệ!")
24
25 # Lấy thông tin từ người chơi
26 BOARD_SIZE, WIN_CONDITION = get_game_settings()
27 HUMAN = -1
28 COMP = +1
29
30 # Cập nhật trạng thái ban đầu theo bảng dynamic
31 def get_initial_board(size):
32     """
33     Nhập trạng thái ban đầu của bảng từ người dùng.
34     Người dùng nhập `size * size` số (0: trống, 1: máy, -1: người) theo từng hàng.
35     """
36     print(f"Nhập trạng thái ban đầu của bảng ({size * size} số, cách nhau bởi dấu cách:")
37     print("0: Ô trống, 1: Máy (O/X), -1: Người (X/O)")
38
39     while True:
40         try:
41             # Nhập toàn bộ trạng thái bàn cờ
42             values = list(map(int, input().split()))
43
44             # Kiểm tra số lượng ô nhập vào có khớp với kích thước bảng không
45             if len(values) != size * size or not all(v in [-1, 0, 1] for v in values):
46                 raise ValueError
47
48             # Chuyển danh sách thành ma trận `size x size`
49             return [values[i * size:(i + 1) * size] for i in range(size)]
50
51         except ValueError:
52             print(f"Dữ liệu không hợp lệ! Vui lòng nhập đúng {size * size} số (0, 1, -1) cách nhau bởi dấu cách.")
53
54 # Khai báo board game
55 board = get_initial_board(BOARD_SIZE)
56
```

```

57 def evaluate(state):
58     """ Đánh giá trạng thái trò chơi """
59     if wins(state, COMP):
60         score = +1 # Máy thắng
61     elif wins(state, HUMAN):
62         score = -1 # Người thắng
63     else:
64         score = 0 # Hòa
65
66     return score
67
68 # Cập nhật luật thắng cho bảng lớn
69 def wins(state, player):
70     """ Kiểm tra người chơi có thắng không với điều kiện WIN_CONDITION """
71     size = len(state)
72
73     # Kiểm tra theo hàng ngang
74     for row in state:
75         for i in range(size - WIN_CONDITION + 1):
76             if all(cell == player for cell in row[i:i + WIN_CONDITION]):
77                 return True
78
79     # Kiểm tra theo cột dọc
80     for col in range(size):
81         for i in range(size - WIN_CONDITION + 1):
82             if all(state[i + j][col] == player for j in range(WIN_CONDITION)):
83                 return True
84
85     # Kiểm tra đường chéo chính \
86     for i in range(size - WIN_CONDITION + 1):
87         for j in range(size - WIN_CONDITION + 1):
88             if all(state[i + k][j + k] == player for k in range(WIN_CONDITION)):
89                 return True
90
91     # Kiểm tra đường chéo phụ /
92     for i in range(size - WIN_CONDITION + 1):
93         for j in range(WIN_CONDITION - 1, size):
94             if all(state[i + k][j - k] == player for k in range(WIN_CONDITION)):
95                 return True
96
97     return False

```

```

99 def game_over(state):
100     """ Kiểm tra trò chơi đã kết thúc chưa """
101     return wins(state, HUMAN) or wins(state, COMP)
102
103
104 def empty_cells(state):
105     """ Tìm các ô trống trên bàn cờ """
106     cells = []
107     for x, row in enumerate(state):
108         for y, cell in enumerate(row):
109             if cell == 0:
110                 cells.append([x, y])
111
112     return cells
113
114
115 def valid_move(x, y):
116     """ Kiểm tra nước đi hợp lệ """
117     if [x, y] in empty_cells(board):
118         return True
119     else:
120         return False
121
122
123 def set_move(x, y, player):
124     """ Thực hiện nước đi của người chơi hoặc máy """
125     if valid_move(x, y):
126         board[x][y] = player
127         return True
128     else:
129         return False

```

```

131 # Tối ưu Minimax với tia nhánh Alpha-Beta
132 def minimax(state, depth, player, alpha, beta):
133     """ Thuật toán Minimax tối ưu với tia nhánh Alpha-Beta """
134     if depth == 0 or game_over(state):
135         return [-1, -1, evaluate(state)] # Trả về điểm đánh giá nếu đã đến độ sâu tối thiểu hoặc kết thúc trò chơi
136
137     best = [-1, -1, -infinity] if player == COMP else [-1, -1, +infinity]
138
139     for x, y in empty_cells(state):
140         state[x][y] = player
141         score = minimax(state, depth - 1, -player, alpha, beta)
142         state[x][y] = 0
143         score[0], score[1] = x, y # Lưu lại vị trí nước đi
144
145         if player == COMP:
146             if score[2] > best[2]:
147                 best = score
148             alpha = max(alpha, score[2]) # Cập nhật alpha
149         else:
150             if score[2] < best[2]:
151                 best = score
152             beta = min(beta, score[2]) # Cập nhật beta
153
154         if beta <= alpha:
155             break # Cắt nhánh nếu alpha >= beta
156
157     return best
158
159
160 def clean():
161     """ Clear console """
162     os_name = platform.system().lower()
163     if 'windows' in os_name:
164         system('cls')
165     else:
166         system('clear')
167

```

```

168 # Cập nhật console để hiển thị bảng lớn hơn
169 ~ def render(state):
170     """ Hiển thị bàn cờ với kích thước linh hoạt """
171     size = len(state)
172     print("\n" + " " * size + ".join([str(i) for i in range(size)])")
173
174 ~ for i, row in enumerate(state):
175     row_str = f"{i} " + " | ".join(["X" if cell == HUMAN else "O" if cell == COMP else " " for cell in row])
176     print(row_str)
177     print(" " + "-" * (size * 4 - 1)) # In dấu "-" để phân cách các hàng
178
179
180 ~ def ai_turn():
181     """ Lượt đi của máy """
182     depth = len(empty_cells(board))
183 ~ if depth == 0 or game_over(board):
184     return
185
186     clean()
187     print("Lượt của Máy!")
188     render(board)
189
190     # Nếu bàn cờ toàn trống, máy chọn ngẫu nhiên
191 ~ if depth == BOARD_SIZE * BOARD_SIZE:
192     x, y = choice(range(BOARD_SIZE)), choice(range(BOARD_SIZE))
193 ~ else:
194     move = minimax(board, depth, COMP, -infinity, infinity)
195     x, y = move[0], move[1]
196
197     set_move(x, y, COMP)
198     time.sleep(1)
199

```

```
201 def human_turn():
202     """ Lượt đi của người chơi """
203     depth = len(empty_cells(board))
204     if depth == 0 or game_over(board):
205         return
206
207     clean()
208     print("Lượt của Bạn!")
209     render(board)
210
211     # Người chơi nhập tọa độ để đánh
212     while True:
213         try:
214             x, y = map(int, input("Nhập tọa độ (dạng: dòng cột): ").split())
215             if set_move(x, y, HUMAN):
216                 break
217             else:
218                 print("Nước đi không hợp lệ, thử lại.")
219         except ValueError:
220             print("Vui lòng nhập số hợp lệ!")
```

```

222 def main():
223     """ Chạy trò chơi """
224     clean()
225
226     # Nhập trạng thái ban đầu của bảng từ người dùng
227     global board
228     board = get_initial_board(BOARD_SIZE)
229
230     # Xác định số lượt đi còn lại
231     depth = len(empty_cells(board))
232
233     # Kiểm tra nếu trò chơi đã kết thúc
234     if game_over(board) or depth == 0:
235         render(board) # Hiển thị trạng thái bảng
236         if wins(board, HUMAN):
237             print("Trò chơi đã kết thúc: Người chơi THẮNG!")
238         elif wins(board, COMP):
239             print("Trò chơi đã kết thúc: Máy TÍNH THẮNG!")
240         else:
241             print("Trò chơi đã kết thúc: HÒA!")
242         exit()
243
244     h_choice = '' # X or O
245     c_choice = '' # X or O
246     first = '' # if human is the first
247
248     # Người chơi chọn X hoặc O
249     while h_choice != 'O' and h_choice != 'X':
250         try:
251             print('')
252             h_choice = input('Choose X or O\nChosen: ').upper()
253         except (EOFError, KeyboardInterrupt):
254             print('Bye')
255             exit()
256         except (KeyError, ValueError):
257             print('Bad choice')
258

```

```

259     # Chọn lựa của máy
260     if h_choice == 'X':
261         c_choice = 'O'
262     else:
263         c_choice = 'X'
264
265     # Người chơi có thể bắt đầu trước
266     clean()
267     while first != 'Y' and first != 'N':
268         try:
269             first = input('First to start?[y/n]: ').upper()
270         except (EOFError, KeyboardInterrupt):
271             print('Bye')
272             exit()
273         except (KeyError, ValueError):
274             print('Bad choice')
275
276     # Main loop của trò chơi
277     while len(empty_cells(board)) > 0 and not game_over(board):
278         if first == 'N':
279             ai_turn()
280             first = ''
281
282         human_turn()
283         ai_turn()

```

```

285     # Thông báo kết quả khi trò chơi kết thúc
286     if wins(board, HUMAN):
287         clean()
288         print(f'Human turn [{h_choice}]')
289         render(board)
290         print('YOU WIN!')
291     elif wins(board, COMP):
292         clean()
293         print(f'Computer turn [{c_choice}]')
294         render(board)
295         print('YOU LOSE!')
296     else:
297         clean()
298         render(board)
299         print('DRAW!')
300
301     exit()
302
303
304 if __name__ == '__main__':
305     main()
306

```

# -Kết quả

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS F:\CODE\HCMUS\mon-hoc\17.co-so-tri-tue-nhan-tao\Drive\Code\Lab3-minimax\23880020_py_version> python minimax-custom.py
Nhập kích thước bảng (ví dụ: 10 cho 10x10): 3
Nhập số ô liên tiếp cần để thắng (tối đa 3): 3
Nhập trạng thái ban đầu của bảng (9 số, cách nhau bởi dấu cách):
0: 0 trống, 1: Máy (O/X), -1: Người (X/O)
0 0 0 0 0 0 0 0 0
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Nhập trạng thái ban đầu của bảng (9 số, cách nhau bởi dấu cách):
0: 0 trống, 1: Máy (O/X), -1: Người (X/O)
0 0 0 0 0 0 0 0 0

Choose X or O
Chosen: X
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
First to start?[y/n]: y
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Lượt của Bạn!

  0 1 2
0  |  |
-----
1  |  |
-----
2  |  |
-----
Nhập tọa độ (dạng: dòng cột): 0 0
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Lượt của Bạn!

  0 1 2
0 X |  |
-----
1  | 0 |
-----
2  |  |
-----
Nhập tọa độ (dạng: dòng cột):
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Lượt của Bạn!

  0 1 2
0 X |  |
-----
1  | 0 |
-----
2  |  |
-----
Nhập tọa độ (dạng: dòng cột): 0 2
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Lượt của Bạn!

  0 1 2
0 X | 0 | X
-----
1  | 0 |
-----
2  |  |
-----
Nhập tọa độ (dạng: dòng cột): 2 0
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Lượt của Bạn!

  0 1 2
0 X | 0 | X
-----
1 0 | 0 |
-----
2 X |  |
-----
Nhập tọa độ (dạng: dòng cột): 2 2
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Computer turn [O]

  0 1 2
0 X | 0 | X
-----
1 0 | 0 |
-----
2 X |  | X
-----
YOU LOSE!
PS F:\CODE\HCMUS\mon-hoc\17.co-so-tri-tue-nhan-tao\Drive\Code\Lab3-minimax\23880020_py_version>
```

