

MIDTERM PRESENTATION

Introduction to **ARTIFICIAL INTELLIGENCE**



PowerRanger

		StudentID	Email	Todo	Completed
	Truong Thai Dan Huy	52100222	52100222@student.tdtu.edu.vn	Ex 1,2	100%
	Nguyen Trong Dat	52100176	52100176@student.tdtu.edu.vn	Ex 1,2	100%
	Trinh Lam Nhu	52100916	52100916@student.tdtu.edu.vn	Ex 1, slide maker	100%
	La Quoc Bao	52100872	52100872@student.tdtu.edu.vn	Ex 2,3	100%
	Dinh Hoang Phuc	52100290	52100290@student.tdtu.edu.vn	Ex 1, present	100%

TABLE OF CONTENTS

PROBLEM 1

Giúp Pacman di chuyển từ vị trí xuất phát đến vị trí mỗi sử dụng Uninformed search.

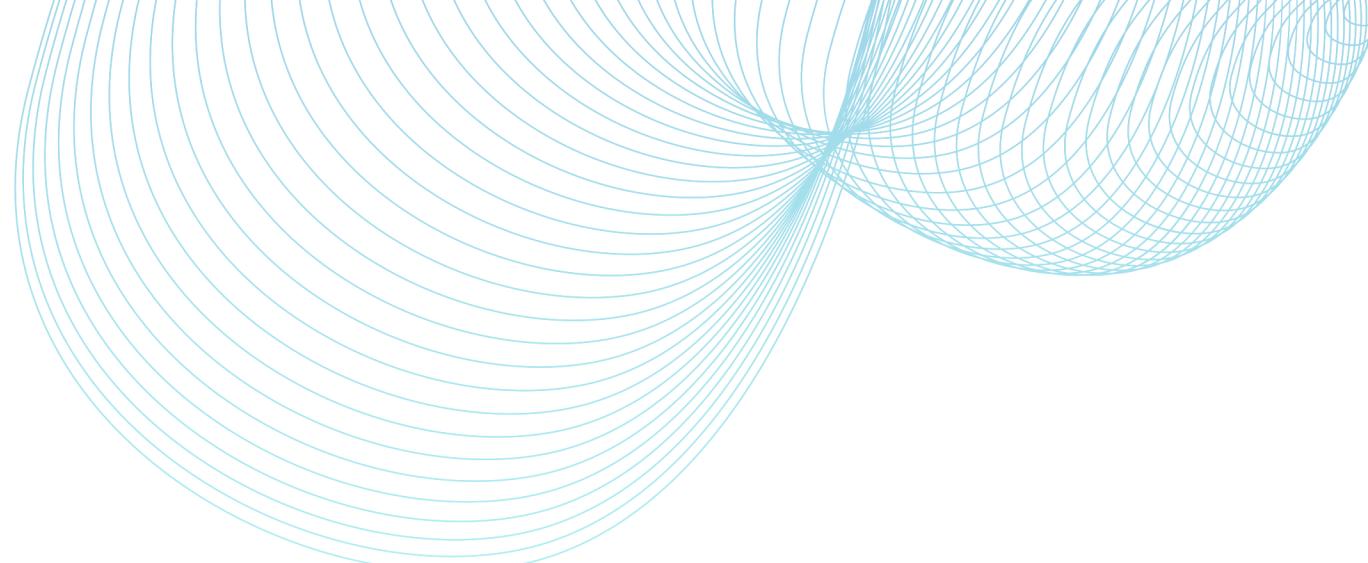
PROBLEM 2

Giúp Pacman ăn hết tất cả các điểm mỗi sử dụng Best-First Search.

PROBLEM 3

Sử dụng thuật toán Hill Climbing để sắp xếp các quân hậu trên bàn cờ 8x8 sao cho ít va chạm nhau nhất.

PROBLEM 1 AND SOLUTION



IDEA

- initial_state: Trạng thái ban đầu của Pacman
- goal_state: Vị trí điểm mồi
- matrix: Ma trận bản đồ
- Đọc file -> in mê cung ra màn hình.
- Lấy vị trí hiện tại của pacman và điểm mồi
- Định nghĩa hàm tìm vị trí vật cản xung quanh vị trí truyền vào

```
function successor(state):
    successors = empty list
    row, col = state

    if matrix[row + 1][col] is not a wall:
        add (row + 1, col) to successors

    if matrix[row - 1][col] is not a wall:
        add (row - 1, col) to successors

    if matrix[row][col + 1] is not a wall:
        add (row, col + 1) to successors

    if matrix[row][col - 1] is not a wall:
        add (row, col - 1) to successors

    return successors
```

IDEA

- Định nghĩa các data structure như stack, queue, priority queue.
- Implement thuật toán bfs, dfs, ucs dựa trên data structure trên.
- Di chuyển dựa trên kết quả trả về của thuật toán



IDEA

```
function animate(actions):
    print_maze_again()
    wait_for_input()
    current_state = get_initial_state()

    for action in actions:
        row, col = current_state

        # Erase current position
        set_matrix_value(row, col, ' ')

        # Move to new position
        if action == 'N':
            row -= 1
        elif action == 'S':
            row += 1
```

```
        elif action == 'W':
            col -= 1
        elif action == 'E':
            col += 1
        elif action == 'Stop':
            set_matrix_value(row, col, 'P')
            print_maze_again()
            return

    # Update matrix and print maze
    set_matrix_value(row, col, 'P')
    current_state = (row, col)
    print_maze_again()
    wait_for_input()
```

PROBLEM 2 AND SOLUTION



DISTANCE MEASURE

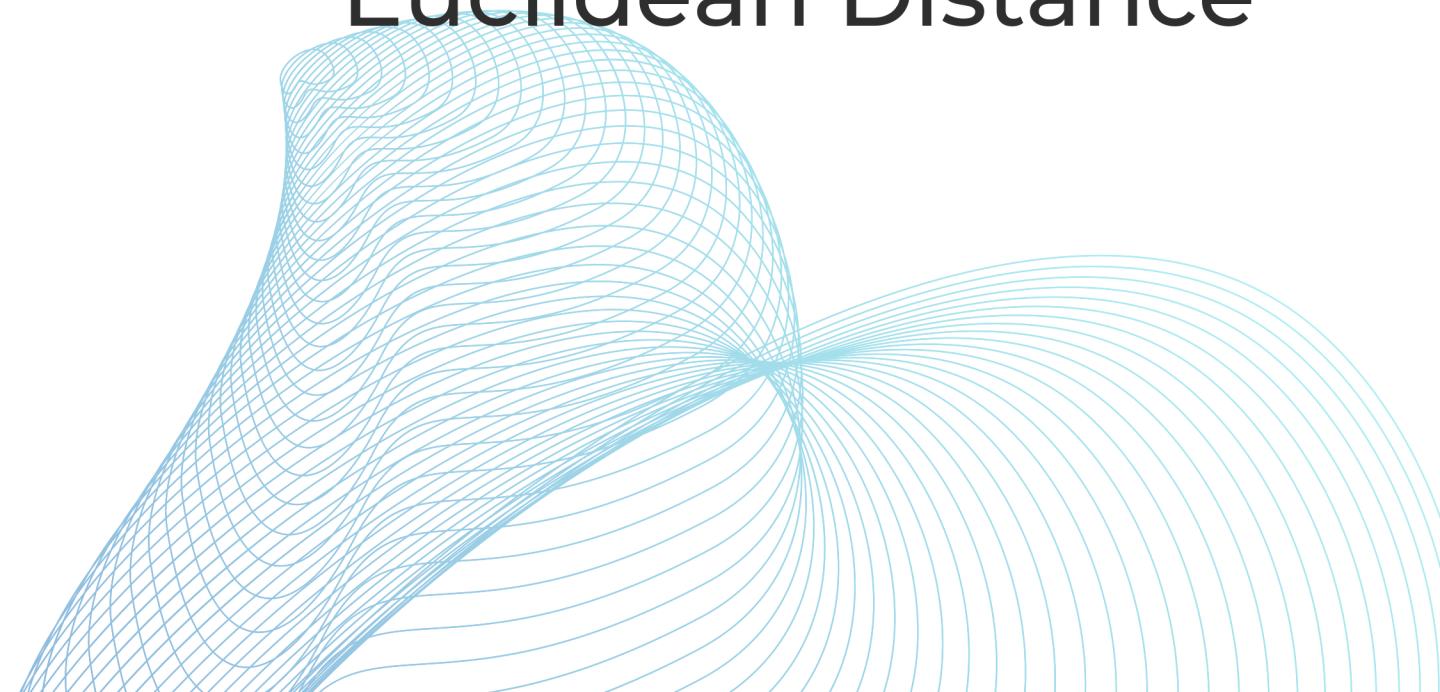
Manhattan Distance

(hay còn gọi là
khoảng cách taxicab)

$$|x_1 - x_2| + |y_1 - y_2|$$

Euclidean Distance

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$



HEURISTIC



```
FUNCTION h_manhattan(state):
    IF not isinstance(state,
problems.SingleFoodSearchProblem):
        RETURN False
    ENDIF
    row_goal, col_goal <- state.get_goal_state()
    row_state, col_state <- state.initial_state
    result <- abs(col_state - col_goal) + abs(row_state -
row_goal)
    RETURN result
```

HEURISTIC

```
● ● ●

FUNCTION h_euclidean(state):
    IF not isinstance(state,
problems.SingleFoodSearchProblem):
        RETURN False
    ENDIF
    row_goal, col_goal <- state.get_goal_state()
    row_state, col_state <- state.initial_state
    result <- math.sqrt((col_state - col_goal)2 + (row_state -
row_goal)2)
    RETURN result
```

EXPLAIN

Manhattan:

- Mang tính **admissible** vì nó không bao giờ ước tính chi phí lớn hơn thực tế để đi tới đích.
- Nó cũng là **consistent** vì khoảng cách Manhattan giữa hai điểm luôn nhỏ hơn hoặc bằng khoảng cách Euclid giữa hai điểm.
- Do đó, nếu chúng ta di chuyển từ trạng thái hiện tại đến trạng thái kế tiếp, chi phí tăng thêm ít nhất khoảng cách Manhattan giữa hai trạng thái.

EXPLAIN

Euclid:

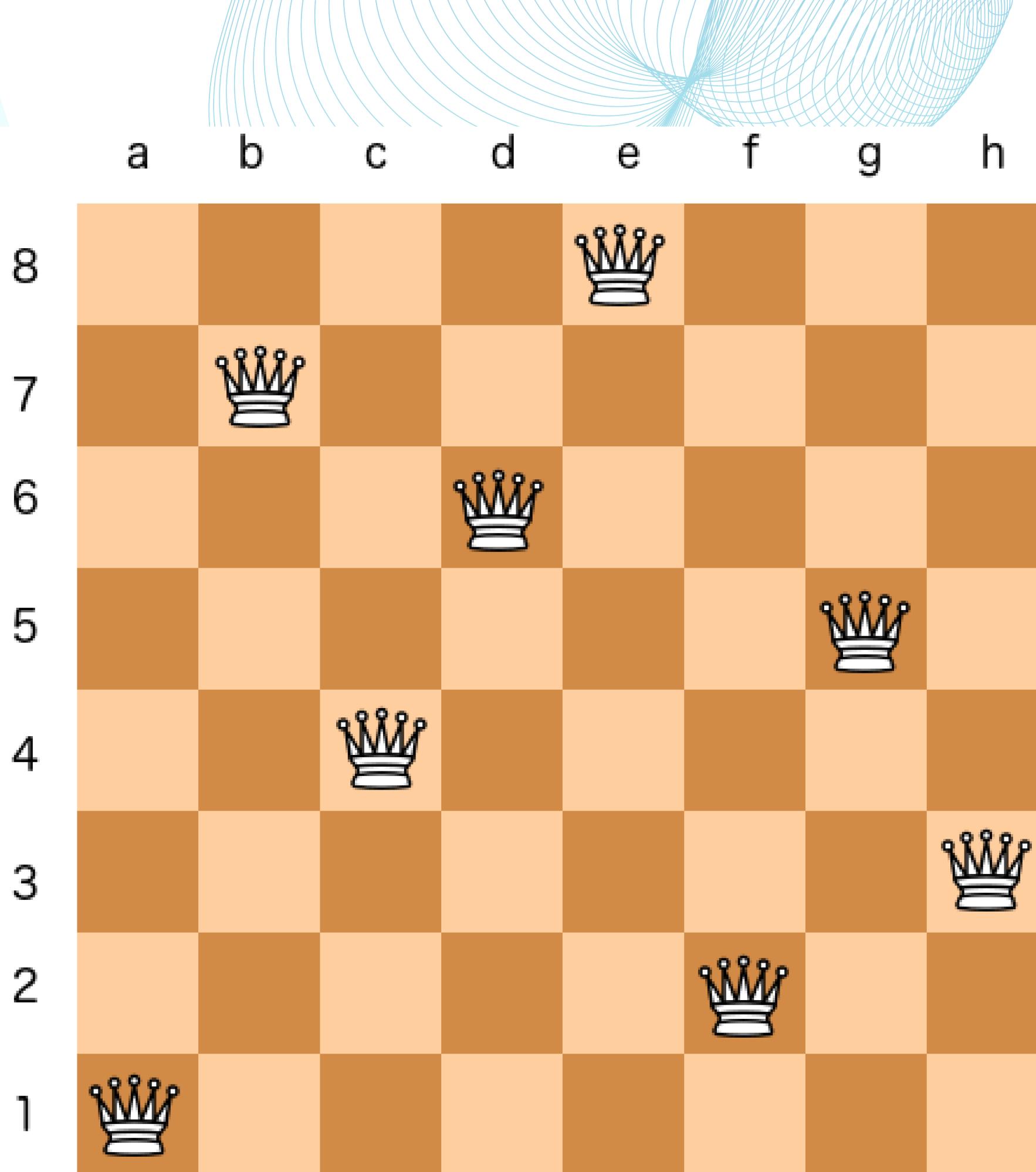
- Mang tính **admissible** vì nó không bao giờ ước tính chi phí nhỏ hơn thực tế để đi tới đích.
- Nó cũng là **consistent** vì khoảng cách Euclid giữa hai điểm là một ước tính chính xác về chi phí để di chuyển từ trạng thái hiện tại đến trạng thái kế tiếp.

HEURISTIC



```
FUNCTION h_multi(state):
    IF not isinstance(state, problems.MultiFoodSearchProblem):
        RETURN False
    ENDIF
    arr_goal_state <- state.get_goal_state()
    row_state, col_state <- state.initial_state
    result <- 0
    for row_goal,col_goal in arr_goal_state:
        result += abs(col_state - col_goal) + abs(row_state -
row_goal)
        row_state, col_state <- row_goal, col_goal
    ENDFOR
    RETURN result
```

PROBLEM 3 AND SOLUTION



IDEA

18	12	14	13	13	12	14	14
14	16	13	15	12	14	12	16
14	12	18	13	15	12	14	14
15	14	14	👑	13	16	13	16
👑	14	17	15	👑	14	16	16
17	👑	16	18	15	👑	15	👑
18	14	👑	15	15	14	👑	16
14	14	13	17	12	14	12	18

Hàm $h()$: trả về số cặp hậu có thể tấn công lẫn nhau, vừa trực tiếp lẫn gián tiếp.

- Duyệt theo từng cột và hàng, đến khi tìm được vị trí của quân hậu
- Kiểm tra tại vị trí đó có bao nhiêu quân hậu có thể tấn công nó

HILL CLIMBING SEARCH

```
FUNCTION hill_climbing_search(self):
    state <- cloneBoard( board, [] )
    FOR col IN range(len( board )):
        min_h <- 100000
        index <- 0
        FOR row IN range(len( board )):
            state[row][col] <- '0'
        ENDFOR
        board <- cloneBoard(state, board)
        FOR j IN range(len( board )):
            FOR row IN range(len( board )):
                board[row][col] <- '0'
            ENDFOR
            board[j][col] <- 'Q'
            IF min_h > h( board ):
                min_h <- h( board )
                index <- j
            ENDIF
            board[j][col] <- '0'
        ENDFOR
        state[index][col] <- 'Q'
    ENDFOR
    RETURN state
```

PROS AND CONS



PROS

- Nhờ bài tập về nhà của thầy giao và các mã giả của các thuật toán có trong slide học, các data structure đã được học.
- Có mô tả chi tiết về các thuật toán mà đề bài yêu cầu

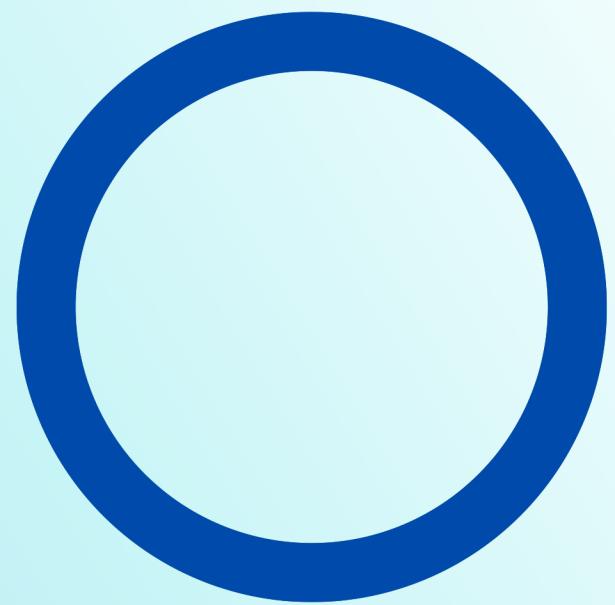
VS



CONS

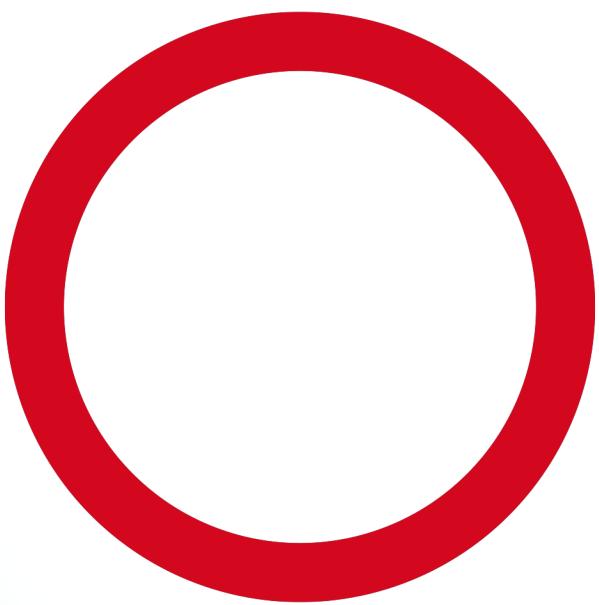
- Mất nhiều thời gian vào yc1-4, yc2
- Giải thuật dài và tính chính xác được chi tiết thuật toán cần phải chạy của một quân hậu trên một cột và một bàn cờ

Overall Rating



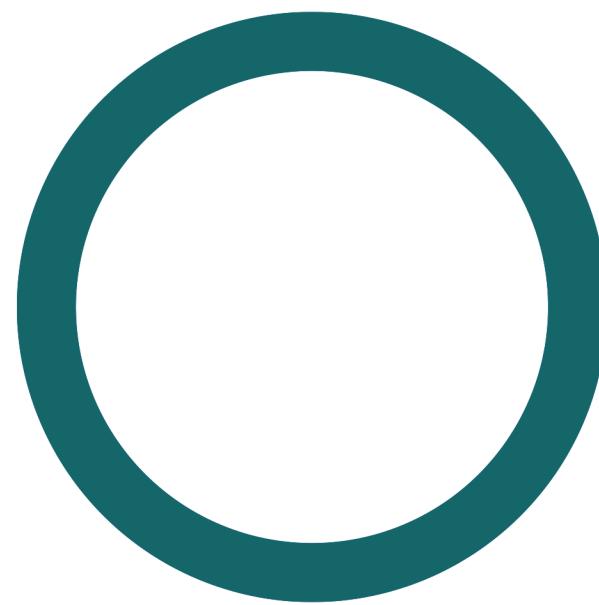
Problem 1

100%



Problem 2

100%



Problem 3

100%

THANK YOU

