INTRODUCTION TO ARTIFICIAL INTELLIGENCE

Nhập môn Trí tuệ nhân tạo

Group: 4

TASK ASSIGNMENT TABLE

MEMBER	ID - Email	MISSION	COMPLETE
NGUYỄN ĐÌNH VIỆT HOÀNG	522H0120 - 522H0120@student.tdtu.edu.vn	PSEUDOCODE	100%
NGUYỄN NHẬT CHIÊU	522H0133 - 522H0133@student.tdtu.edu.vn	PSEUDOCODE; Powerpoint	100%
ĐẶNG CÔNG MINH	522H0095 - 522H0095@student.tdtu.edu.vn	Task 2 - PACMAN	100%
TRẦN THIÊN ÂN	522H0165 - 522H0165@student.tdtu.edu.vn	Task 1 - 8-Puzzle	100%
VÕ MINH TÀI	522H0168 - 522H0168@student.tdtu.edu.vn	Presentation; Powerpoint	100%

For more info:
GROUP | FOUR | WITH | LOVE

You can visit our members:
ÂN | HOÀNG | CHIÊU | MINH | TÀI

WAY TO SOLVE THE PROBLEM

01 BFS for Task 1

02 A* for Task 1

03 UCS for task 2

04 A* for Task 2

05 Pseudocode for illustrate

06 Try Hard

01

Task 1: 8-Puzzle

8-Puzzle

BFS

Tìm đường đi ngắn nhất
Tìm đường đi có chi phí thấp nhất
Không sử dụngđệ quy
Có thể duyệt đồ thị theo từng cấp độ

Tốn bộ nhớ
Chậm hơn
Không tối ưu cho đường đi có trọng số

A*

Giải quyết vấn đề phức tạp
Hoàn chỉnh, hiệu quả và tối ưu

Phục thuộc vào hàm heuristic
Giả quết vấn để phức tập

Ưu Điểm

Nhược Điểm

Pseudocode of BFS

```
function process_bfs(start, goal):
    start ← Node(start, 0, 0)
    open_bfs ← QUEUE WITH start
    closed_bfs ← EMPTY SET
    path_cost_bfs ← 0
    while open_bfs is not EMPTY:
        cur ← DEQUEUE open_bfs
        PRINT cur.data
        if h(cur.data, goal) == 0:
            PRINT "Actions taken:"
            actions ← get_actions(cur)
            PRINT actions
            PRINT "Step: ", cur.step_count
            APPEND path_cost_bfs to self.path_costs
            self.total_cost ← path_cost_bfs + cur.step_count
            PRINT "Path cost:", self.path_cost
            PRINT "Total cost:", self.total_cost
            break
        for each i in cur.generate_child():
            if i not in closed_bfs:
                ENQUEUE i to open_bfs
                ADD i to closed_bfs
        path_cost_bfs ← path_cost_bfs + 1
```

Pseudocode of A*

```
function process_astar(start, goal):
    start ← Node(start, 0, 0)
    start.fval ← f(start, goal)
    PUSH (start.fval, start) to self.open
    path_cost_astar ← 0
    while self.open is not EMPTY:
        cur ← POP MINIMUM ELEMENT from self.open
        PRINT cur.data
        if h(cur.data, goal) == 0:
            PRINT "Actions :"
            actions ← get_actions(cur)
            PRINT actions
            PRINT "Step: ", cur.step_count
            APPEND path_cost_astar to self.path_costs
            self.total_cost ← path_cost_astar + cur.step_count
            PRINT "Path cost:", self.path_cost
            PRINT "Total cost:", self.total_cost
            break
        for each i in cur.generate_child():
            if i not in self.closed:
                i.fval ← f(i, goal)
                PUSH (i.fval, i) to self.open
        APPEND cur to self.closed
        path_cost_astar ← path_cost_astar + 1
```

02

Task 2: PACMAN

Pseudocode of UCS

```
function ucs(graph, start, goal):
    frontier = PriorityQueue()
    frontier.push(start, 0)
    explored = set()

    while not frontier.empty():
        current_node = frontier.pop()
        print(current_node.data)

        if current_node.data == goal:
            actions = get_actions_list(current_node)
            print("Actions:", actions)
            print("Step:", current_node.step_count)
            return

        explored.add(current_node.data)

        for neighbor in current_node.generate_child():
            if neighbor not in explored:
                neighbor_cost = current_node.step_count + 1
                frontier.push(neighbor, neighbor_cost)
```

Pseudocode of A*

```
function a_star(graph, start, goal):
    frontier = PriorityQueue()
    frontier.push(start, 0)
    explored = set()

    while not frontier.empty():
        current_node = frontier.pop()
        print(current_node.data)

        if current_node.data == goal:
            # Goal reached, return the path
            actions = get_actions_list(current_node)
            print("Actions:", actions)
            print("Step:", current_node.step_count)
            return

        explored.add(current_node.data)

        for neighbor in current_node.generate_child():
            if neighbor not in explored:
                neighbor_cost = current_node.step_count + 1
                heuristic_value = h(neighbor.data, goal)
                total_cost = neighbor_cost + heuristic_value
                frontier.push(neighbor, total_cost)
```

DEMO

THANK YOU

Do you have any questions?

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**

THE END