BÁO CÁO GIỮA KỲ

Môn: Nhập môn Trí tuệ nhân tạo
Mã môn học: 503043

Nhóm: 09

Thành viên

1. Lý Tuấn An - 52000620
2. Lý Tiểu Long - 52200168
3. Giản Hoàng Huy - 52200147
4. Huỳnh Hoài Nam - 52200151
5. Lê Hồng Quang - 52200156

Mục lục

- Phần 1: Mô hình hóa trò chơi 8-Puzzle
- Phần 2: Mô hình hóa trò chơi Pacman
- Phần 3: Pacman với đồ họa

Mô hình hóa trò chơi 8-puzzle

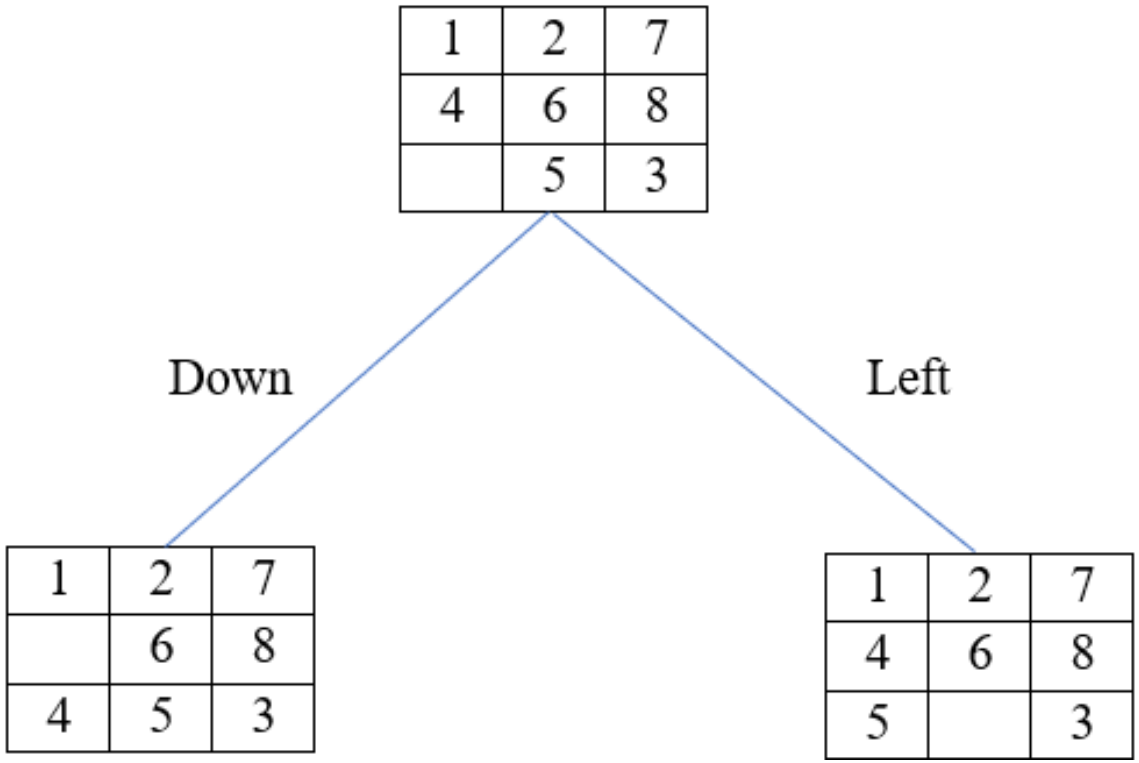
- **Trạng thái:** Vị trí của mỗi ô số từ 1-8 và ô trống.
- **Trạng thái bắt đầu:** Bất kỳ cách sắp xếp 8 ô số và ô trống.

2	5	7
1	8	3
6		4

- **Hành động:** Chuyển động của ô trống
  - Left, Right, Up, Down.
  - Có thể có các tập hợp hành động khác nhau tùy thuộc vào vị trí của ô trống.

Mô hình hóa trò chơi 8-puzzle

- **Mô hình chuyển tiếp:** Trả về kết quả cho một trạng thái và một hành động.



Mô hình hóa trò chơi 8-puzzle

- **Trạng thái đích:** Một trong hai trạng thái

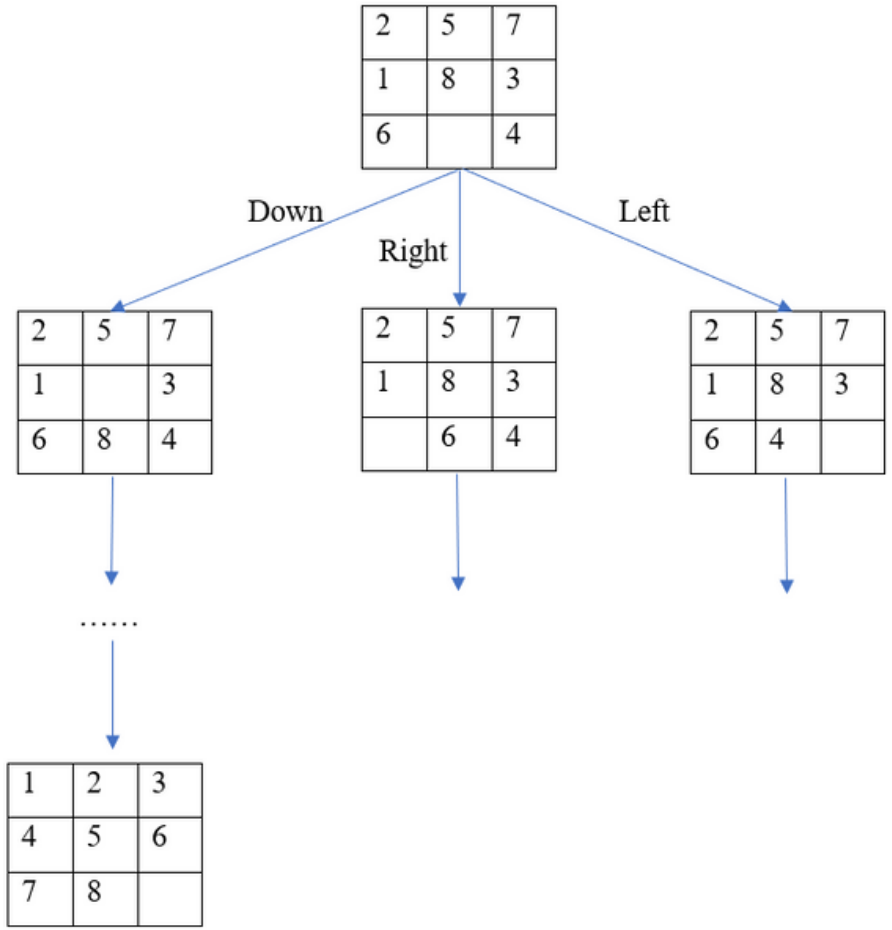
1	2	3
4	5	6
7	8	

	1	2
3	4	5
6	7	8

- **Chi phí:** Mỗi bước di chuyển tốn 1.

Mô hình hóa trò chơi 8-puzzle

- **Mục tiêu:** di chuyển các khối số từ trạng thái bắt đầu (sắp xếp theo thứ tự bất kì) để đạt trạng thái đích.



8-puzzle
Breadth-first Search

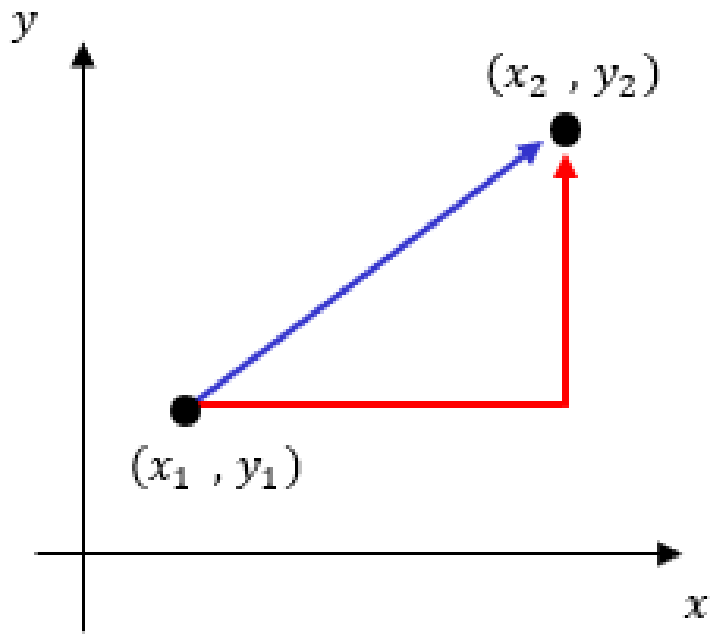
```
function BFS(initial_node) returns path
    frontier ← an empty FIFO Queue
    path ← an empty list
    frontier ← INSERT((initial_node, path))
    explored ← an empty set
    loop do
        current_node, current_path ← POP(frontier)
        if current_node.GOAL_TEST then
            return path
        add current_node.ID to explored
        for each successor in current_node.GET_SUCCESSORS()
            if successor.ID not in explored then
                add successor.ID to explored
                frontier ← INSERT((successor, current_path + [successor.ACTION])
```

8-puzzle
A_star

```
function Astar (initial_node, heuristics_func) returns path
    initial_node.SET_F_VALUE(0 + heuristic_func(initial_node))
    frontier ← an empy Priority Queue
    path ← an empty list
    frontier ← INSERT((initial_node, path))
    explored ← an empty set
    loop do
        current_node, current_path ← POP(frontier)
        if current_node.GOAL_TEST then
            return path
        add current_node.ID to explored
        for each successor in current_node.GET_SUCCESSORS()
            if successor.ID not in explored then
                add successor.ID to explored
                successor.SET_F_VALUE(1 + heuristic_func(successor))
                frontier ← INSERT((successor, current_path + [successor.ACTION])
```

8-puzzle
Heuristic Function

Hàm heuristic: Manhattan, Euclidean



Manhattan Distance L^1

Euclidean Distance L^2

$$L^1 = |x_2 - x_1| + |y_2 - y_1|$$

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

8-puzzle
A_star Manhattan

```
function manhattan_distance(node, goal) =
    dx = abs(node.x - goal.x)
    dy = abs(node.y - goal.y)
    return dx + dy
```

board

1	2	3	4	5	6	7	8
1	2	0	0	2	2	0	3

Manhattan = 10
(1 + 2 + 2 + 2 + 3)

goal

8-puzzle
A_star Euclidean

function euclidean_distance(node, goal) =
    Δx = (node.x - goal.x) ^ 2
    Δy = (node.y - goal.y) ^ 2
    return (Δx + Δy) ^ 0.5

board

1	2	3	4	5	6	7	8
1	1	0	0	1	1	0	√5

Euclidean = 6.2
(1 + 1 + 1 + 1 +√5)

goal

8-puzzle
Random 1000 trạng thái bắt đầu

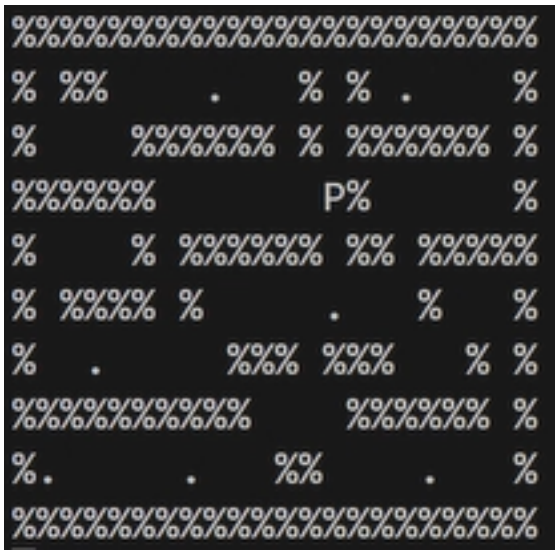
searcher ← search()
qualified_set1 ← goal_1
qualified_set2 ← goal_2
all_qualified_states ← save all possible states from qualified_set1 and qualified_set2
all_qualified_states.txt ← save to file

lines ← openfile ‘all_qualified_states.txt’
n_nandom = 1000
list_random ← random 1000 case from lines

loop do
 current_node ← Node()
 list_node1 ← BFS(current_node)
 list_node2 ← AStar_manhattan(current_node)
 list_node3 ← AStar_euclidean(current_node)
 dot, path, cost ← visualize(list_node[i])

Mô hình hóa trò chơi Pacman

- **Trạng thái:** Được xác định bởi cả vị trí của vật cản, vị trí các ô trống, vị trí của Pacman và vị trí của điểm thức ăn.
  - Vị trí của vật cản, ô trống và điểm thức ăn là cố định.
  - Vị trí của vật cản phải phủ được chu vi của bản đồ.
- **Trạng thái bắt đầu:** Bất kỳ trạng thái nào cũng có thể được chỉ định là trạng thái ban đầu.



“%” : vật cản

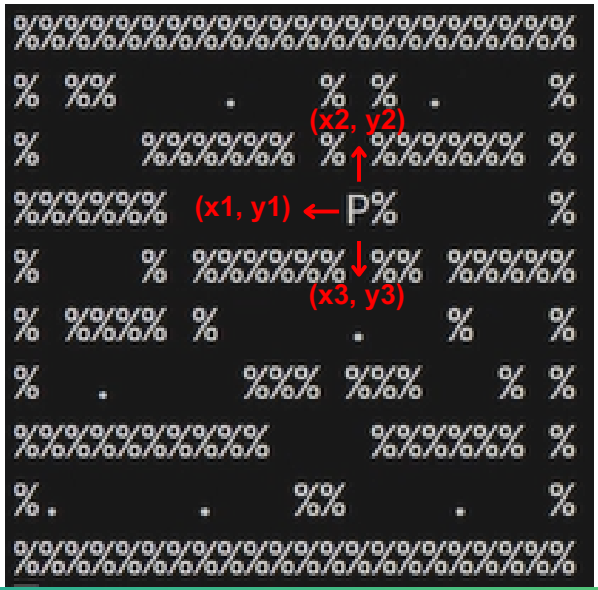
“P” : vị trí của Pacman

“ “ : vị trí mà Pacman có thể đi qua

“.” : điểm thức ăn (có vị trí và số lượng bất kỳ)

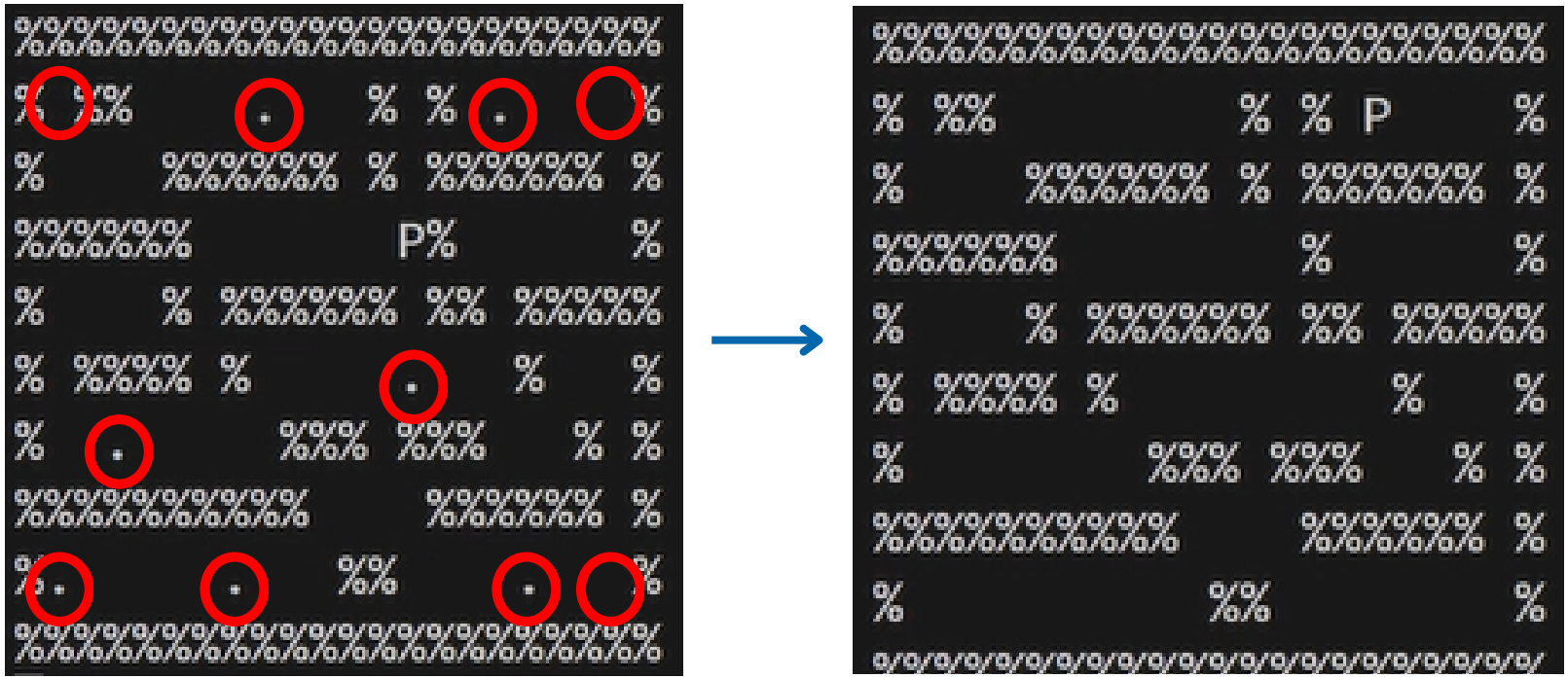
Mô hình hóa trò chơi Pacman

- **Hành động:** Chuyển động của Pacman
  - North, South, West, East, Stop
  - Có thể có các tập hợp hành động khác nhau tùy thuộc vào vị trí và nhiệm vụ của Pacman.
- **Mô hình chuyển tiếp:** Trả về kết quả cho vị trí của Pacman và một hành động.



Mô hình hóa trò chơi Pacman

- **Mục tiêu:** Pacman ăn hết các điểm thức ăn và đi qua 4 góc của bản đồ theo thứ tự bất kỳ.



- **Chi phí:** Mỗi bước hành động tốn 1.

Pacman
Uniform-cost Search

```
function UCS(problem) returns path
    frontier ← an empty Priority Queue ordered by PATH-COST
    initial_pos ← problem.PACMAN_POS
    path ← an empty list
    frontier ← INSERT((initial_pos, path))
    explored ← an empty set
    loop do
        current_pos, current_path ← POP(frontier)
        if problem.GOAL_TEST(current_pos) then
            problem.TARGET_POS.discard(current_pos)
            if not problem.TARGET_POS then
                return path + ”STOP”
            explored.CLEAR() & frontier.CLEAR()
            frontier ← INSERT((current_pos, current_path))
        add current_pos to explored
```

Pacman
Uniform-cost Search

```
function UCS(problem) returns path
    ...
    loop do
        ...
        add current_pos to explored
        for each action, next_pos in problem.GET_SUCCESSORS(current_pos)
            if next_pos not in explored then
                add next_pos to explored
                frontier ← INSERT((next_pos, current_path + action)
```

Pacman
A_star

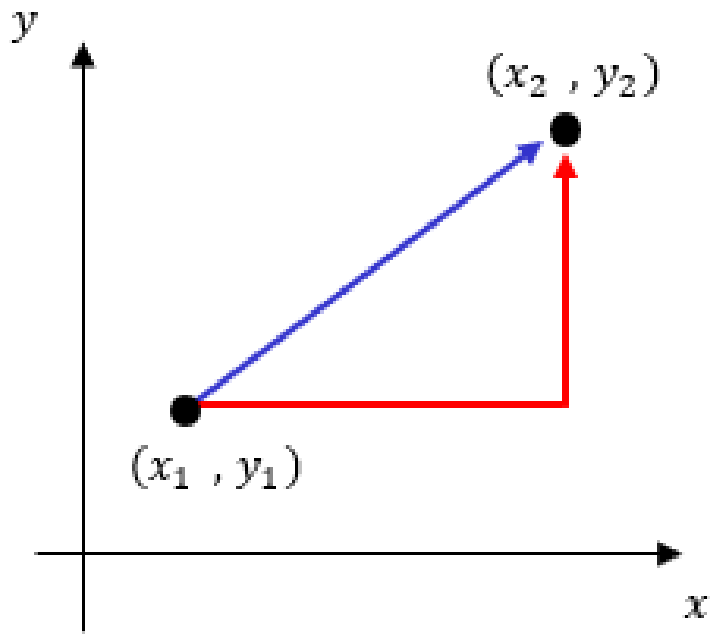
```
function Astar (problem, heuristics_func) returns path
    frontier ← an empy Priority Queue
    initial_pos ← problem.PACMAN_POS
    f_value ← 0 + heuristic_func(initial_pos)
    path ← an empty list
    frontier ← INSERT((f_value, initial_pos, path))
    explored ← an empty set
    loop do
        current_pos, current_path ← POP(frontier)
        if problem.GOAL_TEST(current_pos) then
            problem.TARGET_POS.discard(current_pos)
            if not problem.TARGET_POS then
                return path + ”STOP”
            explored.CLEAR() & frontier.CLEAR()
            f_value ← 0 + heuristic_func(current_pos)
            frontier ← INSERT((f_value, current_pos, current_path))
```

Pacman
A_star

```
function Astar (problem, heuristics_func) returns path
    ...
    loop do
        ...
        add current_pos to explored
        for each action, next_pos in problem.GET_SUCCESSORS(current_pos)
            if next_pos not in explored then
                add next_pos to explored
                f_value ← 1 + heuristic_func(next_pos)
                frontier ← INSERT((f_value, next_pos, current_path + action)
```

Pacman
Heuristic Function

Hàm heuristic: Manhattan, Euclidean



Manhattan Distance L^1

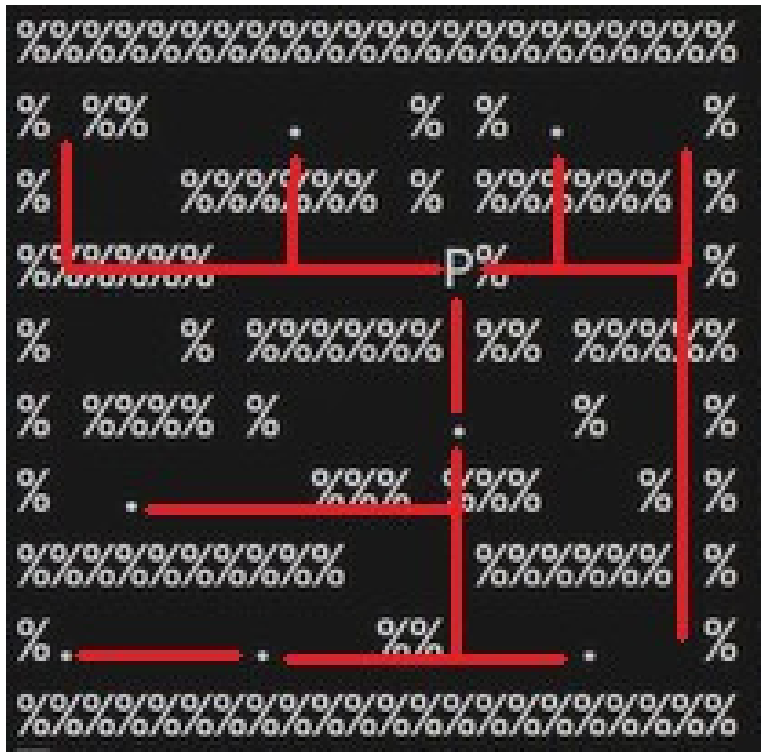
Euclidean Distance L^2

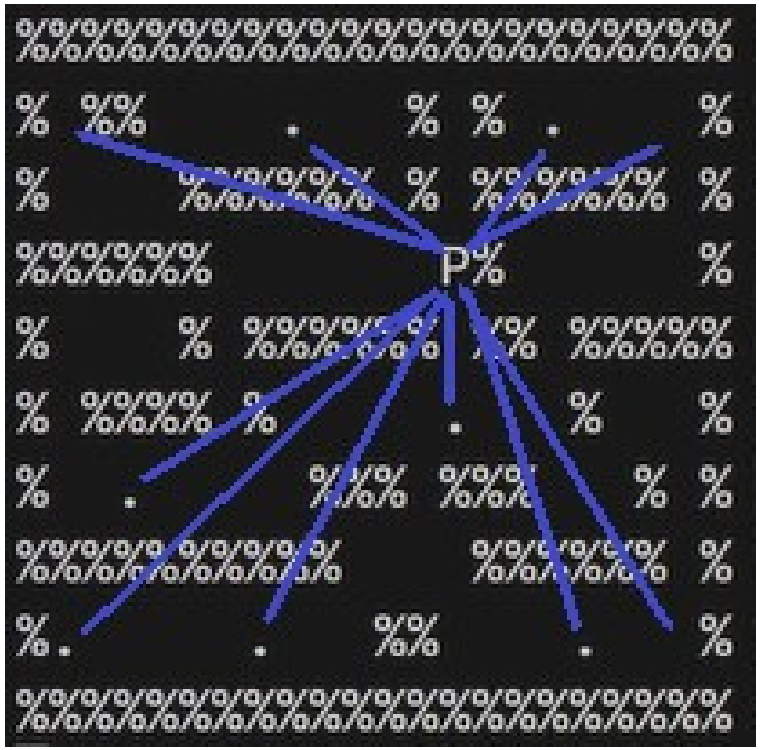
$$L^1 = |x_2 - x_1| + |y_2 - y_1|$$

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

Pacman
Heuristic Function

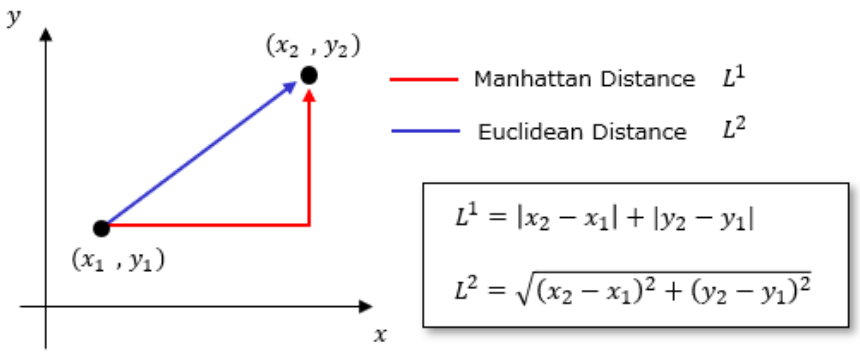
Hàm heuristic: Manhattan, Euclidean





Manhattan Heuristic

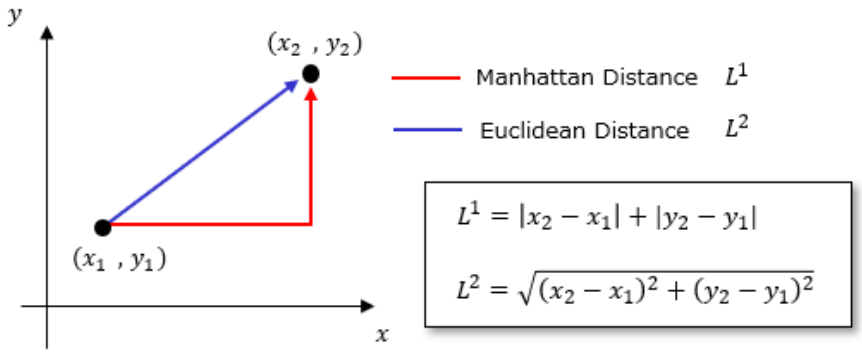
- **Tính admissibility:** Manhattan heuristic tính toán khoảng cách Manhattan giữa các ô trên trạng thái hiện tại và trạng thái mục tiêu. Vì khoảng cách Manhattan không bao giờ lớn hơn thực tế (số bước cần thiết để di chuyển), nên Manhattan heuristic là admissible.



- **Tính consistency:** Với Manhattan heuristic, giá trị heuristic của một ô bất kỳ không bao giờ lớn hơn giá trị heuristic của bất kỳ ô kế tiếp trên đường đi tốt nhất đến trạng thái mục tiêu. Do đó, Manhattan heuristic là consistent.

Euclidean heuristic

- **Tính admissibility:** Euclidean heuristic tính toán khoảng cách Euclidean giữa các ô trên trạng thái hiện tại và trạng thái mục tiêu. Tương tự như Manhattan heuristic, khoảng cách Euclidean không bao giờ lớn hơn thực tế, vì vậy Euclidean heuristic cũng là admissible.
- **Tính consistency:** Tương tự như Manhattan heuristic, Euclidean heuristic cũng đảm bảo tính consistency vì giá trị heuristic của một ô không bao giờ lớn hơn giá trị heuristic của ô kế tiếp trên đường đi tốt nhất đến trạng thái mục tiêu.



Phân chia công việc và mức đồ hoàn thành

1. Lý Tuấn An
   - Email: 52000602@student.tdtu.edu.vn
   - Công việc: Câu 2, chương trình random ở câu 1, làm slide thuyết trình.
   - Mức độ hoàn thành: Hoàn thành.
2. Lý Tiểu Long
   - Email: 52200168@student.tdtu.edu.vn
   - Công việc: Câu 1, làm slide thuyết trình.
   - Mức độ hoàn thành: Hoàn thành.
3. Lê Hồng Quang
   - Email: 52200156@student.tdtu.edu.vn
   - Công việc: Câu 1, làm slide thuyết trình.
   - Mức độ hoàn thành: Hoàn thành.

Phân chia công viên và mức đồ hoàn thành

4. Huỳnh Hoài Nam

- Email: 52200151@student.tdtu.edu.vn
- Công việc: Câu 1, làm slide thuyết trình.
- Mức độ hoàn thành: Hoàn thành.

5. Giản Hoàng Huy

- Email: 52200147@student.tdtu.edu.vn
- Công việc: Câu 2, câu 4 thuyết trình.
- Mức độ hoàn thành: Hoàn thành.

Thuận lợi và khó khăn

- **Thuận lợi:**

+ Có nhiều nguồn tài liệu tham khảo bài toán 8 puzzle và Pacman, những kiến thức đã được giảng dạy ở trên lớp, nhận được sự hỗ trợ từ thầy và những người bạn.

- **Khó khăn:**

+ Khá khó khăn để giải thích cách hoạt động thuật toán phức tạp như heuristic và A* đối với sinh viên lần đầu tiếp cận.

+ Việc áp dụng các thuật toán vào thực tế khá khó khăn khi đối với các bài toán có độ phức tạp cao và kích thước lớn.

Bảng đánh giá mức độ hoàn thành

Câu 1: 8-Puzzle	Đã hoàn thành
Câu 2: pacman	Đã hoàn thành
Yêu cầu nâng cao	Chưa hoàn thành
Câu 4: Thuyết trình	Đã hoàn thành

Tài liệu tham khảo

[1] Từ Minh Phương, Giải quyết vấn đề bằng tìm kiếm, *Giáo trình Nhập môn trí tuệ nhân tạo,* Hà Nội, 2014, 22.

[2] GS. TSKH Hoàng Kiếm, ThS. Đình Nguyễn Anh Dũng, *Giáo Trình Nhập Môn Trí Tuệ Nhân Tạo,* NXB Đại học Quốc gia 2005, Đại học Quốc gia TP.HCM, Trường Đại học Công nghệ thông tin.

[3] Scott Miner, "Solving the 8-Puzzle with A* and SimpleAI" Jun 17, 2023. [Trực tuyến]. Địa chỉ: "https://scottminer.rbind.io/project/solving-the-8-puzzle-with-informed-search-heuristics/" . [Truy cập ngày 5/3/2024].

THANKS FOR WATCHING AND LISTENING