

# BÁO CÁO GK TRÍ TUỆ NHÂN TẠO

**Giảng viên: ThS. Nguyễn Thành An**

**Mã môn: 503043**





# 5 THÀNH VIÊN NHÓM

- 01 52300181 - Bùi Huy Báu
- 02 52300186 - Nguyễn Hải Đăng
- 03 52300201 - Nguyễn Thị Minh Hương
- 04 52300208 - Đăng Ngọc Kim Khánh
- 05 52300260 - Hàng Thị Anh Thư

# Bảng phân công

Họ và tên	Phân công	Tiến độ	Ghi chú
Bùi Huy Báu	<ul style="list-style-type: none"><li>Giao diện (GUI): Pygame/HUD/Auto-play,</li><li>Các lớp hỗ trợ chính và hoàn thiện thuật toán</li></ul>	100%	Chịu trách nhiệm chính; điều phối nhóm.
Nguyễn Hải Đăng	<ul style="list-style-type: none"><li>Định hướng chiến lược tổng thể, phân rã bài toán.</li><li>Thiết kế kiến trúc A*, phát triển heuristic và quy tắc trạng thái</li><li>Tổng hợp, rà soát chất lượng cuối.</li></ul>	100%	Đảm bảo trải nghiệm & tích hợp ổn định.
Nguyễn Thị Minh Hương	<ul style="list-style-type: none"><li>Xây dựng/hoàn thiện Task 1: A*, heuristic, vẽ cây tìm kiếm</li><li>Viết báo cáo ppt phần Task 1 (mô hình hóa, so sánh thuật toán, phân tích heuristic).</li></ul>	100%	Nội dung task 1 mạch lạc, đủ số liệu, slide báo cáo rõ ràng ,có biểu đồ minh họa.
Đặng Ngọc Kim Khánh	<ul style="list-style-type: none"><li>Hoàn thiện báo cáo pacman và đánh giá thuật toán</li><li>Cùng thiết kế/kiểm thử PacmanProblem.</li></ul>	100%	Nội dung chặt chẽ, bám sát yêu cầu đề. Kết quả ổn định.
Hàng Thị Anh Thư	<ul style="list-style-type: none"><li>Xây dựng/hoàn thiện thí nghiệm: triển khai <a href="#">experiments.py</a>.</li><li>Biên tập báo cáo phần kết quả &amp; biểu đồ; kiểm thử bộ xuất file</li></ul>	100%	Số liệu/đồ thị đầy đủ, slide trình bày rõ ràng.

# Bảng hoàn thành

Task	Mô tả ngắn	Trọng số (diagram)	Hoàn thành	Ghi chú
Task 1: A* with 8-Puzzle	Mô hình không gian trạng thái; A* với 02 heuristic, vẽ cây tìm kiếm; thí nghiệm hiệu quả 2 heuristic; OOP & class diagram; so sánh độ phức tạp với BFS.	4	100%	<ul style="list-style-type: none"> <li>Đã gộp heuristic vào 1 file</li> <li>Có hàm vẽ cây</li> <li>Thông kê, so sánh và thí nghiệm.</li> </ul>
Task 2: A* with Pacman	Mô hình Pacman (P, ., O, %, G, E); A* + heuristic (cầm Manhattan/Euclid); pygame GUI; chế độ tay (phím mũi tên) & tự động; teleport 4 góc, TTL ăn tường 5 bước, ma đi ngang, xoay 90° mỗi 30 bước; xuất actions + cost.	4	100%	<ul style="list-style-type: none"> <li>Heuristic MST Pacman</li> <li>Auto ghi output.txt &amp; path.txt</li> <li>Experiment tính toán A*</li> <li>Tránh va chạm ma</li> </ul>
Task 3: Presentation	Slide 4:3, nội dung: danh sách SV & tỉ lệ hoàn thành, mô tả tiếp cận (pseudo-code/sơ đồ), ưu/nhược điểm, tránh nhúng raw code.	2	100%	<ul style="list-style-type: none"> <li>Có bảng phân công, % hoàn thành</li> <li>Mô tả heuristic &amp; kết quả thí nghiệm.</li> </ul>

1

# **TASK 1: 8 PUZZLE**



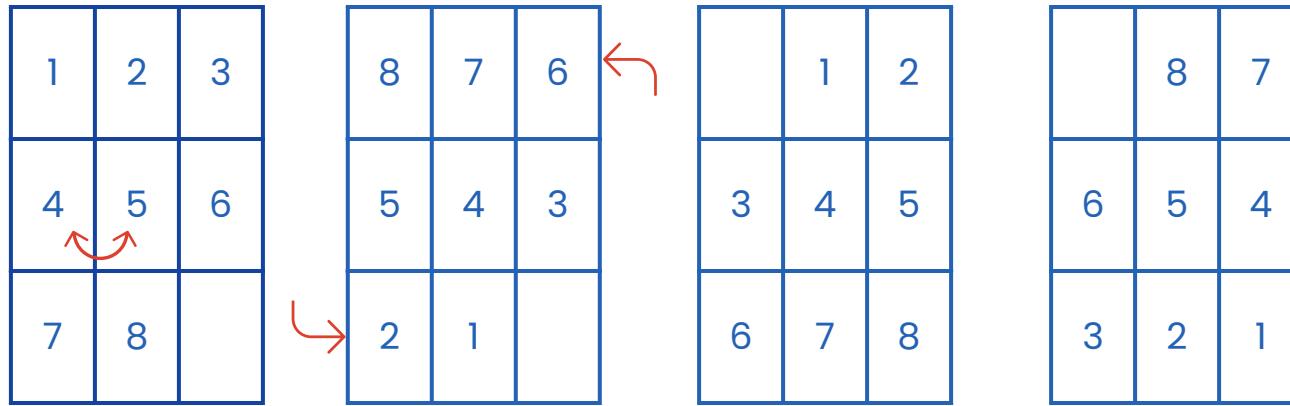


# LUẬT CHƠI

- **Sau một lần di chuyển**, nếu hai ô A và B thỏa mãn điều kiện  $A + B = 9$  và kề nhau theo hàng hoặc cột, người chơi được phép thực hiện thao tác hoán đổi (swap) hai ô đó.
- Ngoài ra, người chơi cũng có thể hoán đổi hai ô nằm ở các vị trí góc chéo nhau, cụ thể là giữa góc trên trái và góc dưới phải, hoặc giữa góc trên phải và góc dưới trái.
- Các hành động hợp lệ trong trò chơi bao gồm:
  - Di chuyển ô trống lên, xuống, trái hoặc phải (nếu hợp lệ).
  - Hoán đổi hai ô kề nhau theo hàng hoặc cột thỏa mãn  $A + B = 9$  (không hoán đổi ô trống).
  - Hoán đổi hai ô ở các vị trí góc chéo ( $TL \leftrightarrow BR$  hoặc  $TR \leftrightarrow BL$ ), ô trống không được phép swap.



# LUẬT CHƠI



Hình 1. 4 trạng thái đích và luật chơi

# CHI TIẾT CHƯƠNG TRÌNH

## 7 LỚP

- Action : Mô tả một hành động hợp lệ
- Problem : Lớp trừu tượng của bài toán tìm kiếm
- EightPuzzleProblem(Problem) : Cài đặt cụ thể cho 8-puzzle
- Node : Nút của cây tìm kiếm
- AStar : Bộ giải A\* theo heuristic
- BFS : Thuật toán bfs để so sánh
- RunStats : Lưu thống kê kết quả

## 2 HÀM HEURISTIC RIÊNG BIỆT

- h\_zero(state, problem)
- h\_pair(state, problem)



# 2 HÀM HEURISTIC ĐƯỢC SỬ DỤNG

## 1. Heuristic 1 – h\_zero(s)

- **Mục đích**

Đây là heuristic cơ bản nhất (được gọi là baseline heuristic).

Nó luôn trả về giá trị 0 cho mọi trạng thái.

Điều đó có nghĩa là A\* chỉ dựa vào chi phí thực tế  $g(n)$  để tìm đường → A\* tương đương với Uniform Cost Search (UCS).

- **Công thức**

$$h_{zero}(s) = 0$$

- **Tính chất**

- **Admissible:** luôn đúng vì  $h(n) = 0 \leq h^*(n)$  (chi phí thật đến đích)

- **Consistent:** vì  $h(n) = 0 \leq 1 + h(n')$  luôn thoả với mọi  $n'$

# 2 HÀM HEURISTIC ĐƯỢC SỬ DỤNG

## 1. Heuristic 1 – h\_zero(s)

- Công thức

$$h_{zero}(s) = 0$$

- Tính chất

- **Admissible:** luôn đúng vì  $h(n) = 0 \leq h^*(n)$  (chi phí thật đến đích)
- **Consistent:** vì  $h(n) = 0 \leq 1 + h(n')$  luôn thoả với mọi  $n'$

- Ý nghĩa

Khi chạy với  $h_{zero}$ , A\* sẽ duyệt nhiều trạng thái hơn, vì không có thông tin hướng dẫn.

- Mã giả

```
function h_zero(state, problem):  
    return 0
```

# 2 HÀM HEURISTIC ĐƯỢC SỬ DỤNG

## 2. Heuristic 2 – h\_pair(s)

- **Mục đích**

Heuristic này tận dụng đặc trưng riêng của đề bài:

Ngoài việc di chuyển ô trống (U/D/L/R), ta còn có thể:

- Hoán đổi (swap) hai ô kề nhau thoả mãn  $A + B = 9$
- Hoán đổi góc chéo (TL $\leftrightarrow$ BR, TR $\leftrightarrow$ BL)

→ Mỗi phép swap có thể sửa đúng đồng thời 2 ô sai vị trí

→ Số bước tối thiểu cần để đưa trạng thái về goal có thể ước lượng bằng  $\text{ceil}(\text{sai_vị_trí} / 2)$ .

- **Công thức**

$$h_{pair}(s) = \left[ \frac{\text{misplaced}(s)}{2} \right]$$

- **Trong đó**

$$\text{misplaced}(s) = \min_{\text{goal} \in \text{Goals}}$$

# 2 HÀM HEURISTIC ĐƯỢC SỬ DỤNG

## 2. Heuristic 2 – h\_pair(s)

- **Admissible:**

Vì heuristic không bao giờ đánh giá vượt quá chi phí thật.

$$h(s) \leq h^*(s)$$

- **Consistent:**

Vì với mỗi hành động có cost = 1, ta luôn có:

$$h(n) \leq 1 + h(n')$$

Cả hai điều kiện trên đều đã được kiểm chứng trong phần “Bảng A1, A2” khi chạy thực nghiệm

- **Ý nghĩa**

Giúp A\* thu hẹp không gian tìm kiếm, giảm số nút mở rộng và thời gian xử lý đáng kể.

# 2 HÀM HEURISTIC ĐƯỢC SỬ DỤNG

## 2. Heuristic 2 – h\_pair(s)

- Công thức

$$h_{pair}(s) = \left[ \frac{misplaced(s)}{2} \right]$$

- Trong đó

$$misplaced(s) = \min_{goal \in Goals}$$

- Tức là

- So sánh trạng thái s với tất cả 4 trạng thái đích.
- Tính số ô sai vị trí cho từng goal.
- Lấy giá trị nhỏ nhất trong 4 giá trị đó.

# 2 HÀM HEURISTIC ĐƯỢC SỬ DỤNG

## 2. Heuristic 2 – h\_pair(s)

- Mã giả

```
function misplaced_tiles_min_over_goals(state, goals):
    best ← +∞
    for each goal in goals:
        count ← 0
        for i = 0..2:
            for j = 0..2:
                if state[i][j] ≠ 0 and state[i][j] ≠ goal[i][j]:
                    count ← count + 1
                best ← min(best, count)
    return best

function h_pair(state, problem):
    mis ← misplaced_tiles_min_over_goals(state, problem.goals)
    return ceil(mis / 2)
```

# ƯU ĐIỂM VÀ NHƯỢC ĐIỂM

- **Ưu điểm**

- Cả hai heuristic ( $\text{ceil}(H/2)$  và misplaced tiles min over goals) đều admissible nên A\* đảm bảo tìm được đường đi ngắn nhất.
- Kết quả nhất quán giữa các lần chạy.
- Tính linh hoạt và mở rộng cao

- **Nhược điểm**

- Tốn bộ nhớ lớn khi không gian trạng thái mở rộng
- Không phù hợp với puzzle quá lớn (như 15-Puzzle)



# ĐÁNH GIÁ

## ĐỘ PHÚC TẠP THỜI GIAN VÀ KHÔNG GIAN GIỮA A\* VÀ BFS



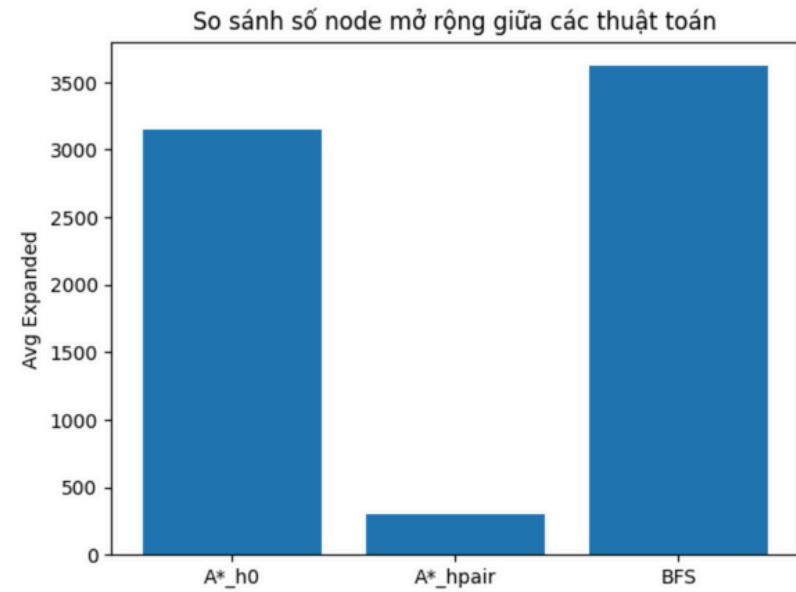
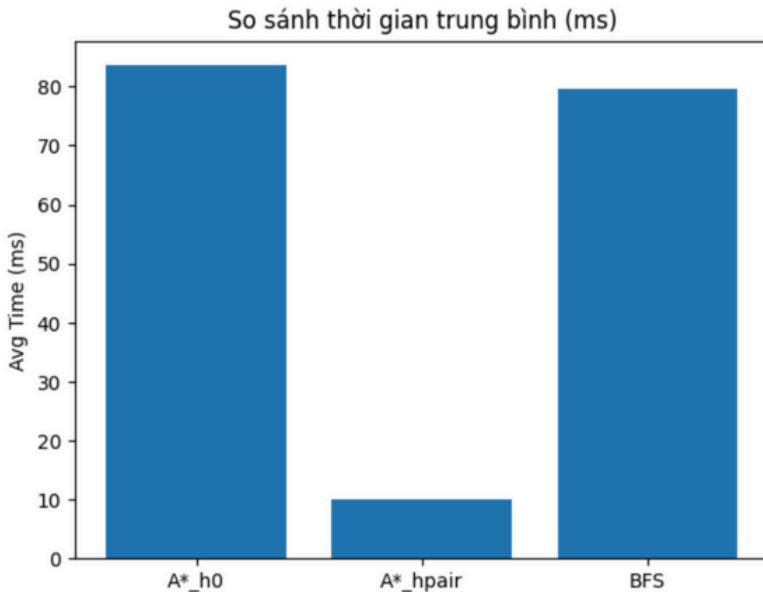
# A\*

- **Loại:** Informed Search (có heuristics)
- **Ưu điểm:**
  - Tìm đường đi tối ưu nhanh hơn
  - Ít mở rộng node khi sử dụng heuristic tốt (heuristic tốt khi đạt được admissibility và consistent)
  - Tiết kiệm thời gian và bộ nhớ hơn BFS
- **Nhược điểm:**
  - Cần thiết kế và tính toán heuristic phù hợp
  - Tốc độ phù thuộc vào chất lượng heuristic
- **Thực nghiệm:**
  - Với h\_pair số node mở rộng giảm rõ rệt
  - Thời gian trung bình thấp hơn BFS

# BFS

- **Loại:** Uninformed Search (không dùng heuristics)
- **Ưu điểm:**
  - Dễ cài đặt, đảm bảo tìm lời giải ngắn nhất
  - Không cần hàm heuristic
- **Nhược điểm:**
  - Tốn thời gian và bộ nhớ do phải mở rộng nhiều node
  - Không hiệu quả với bài toán với không gian lớn như 8-puzzle
- **Thực nghiệm:**
  - Mở rộng nhiều hơn A\*
  - Thời gian chạy lâu hơn

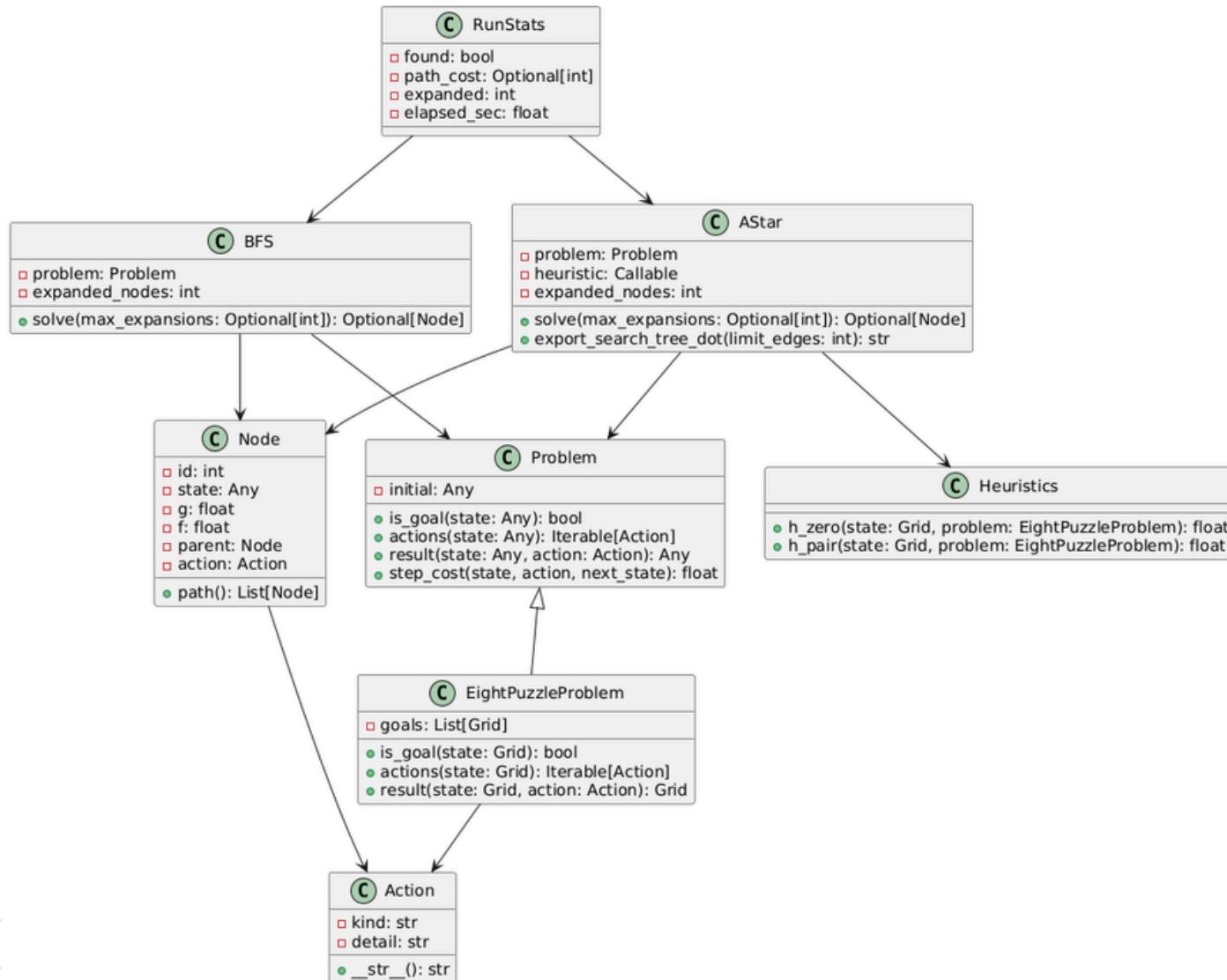
# BIỂU ĐỒ SO SÁNH



- Dựa trên kết quả thực nghiệm, thuật toán A\* thể hiện hiệu quả vượt trội về thời gian và không gian tìm kiếm so với BFS
- Việc sử dụng hàm heuristic giúp A\* định hướng mở rộng node tối ưu hơn, từ đó giảm chi phí tính toán mà vẫn đảm bảo tìm được đường đi tối ưu.

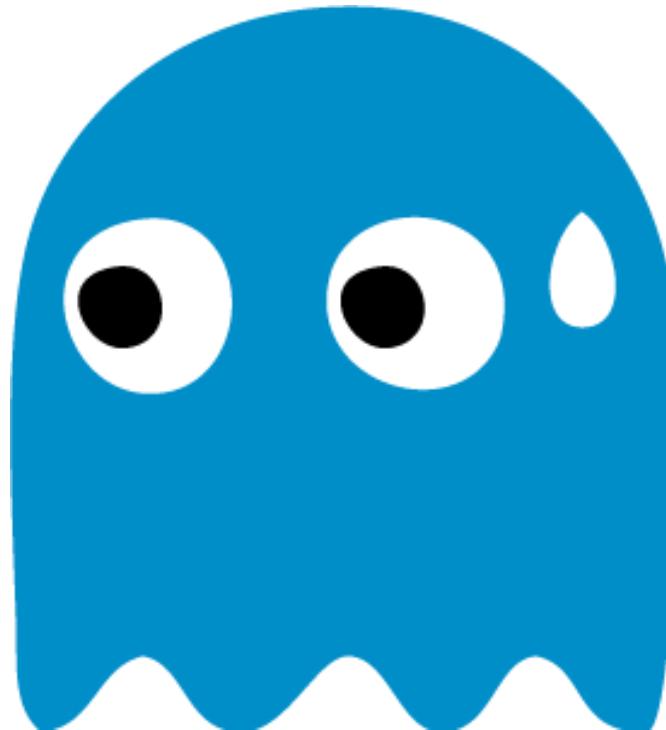
# CLASS DIAGRAM

Class Diagram - A\* Search for 8 Puzzle



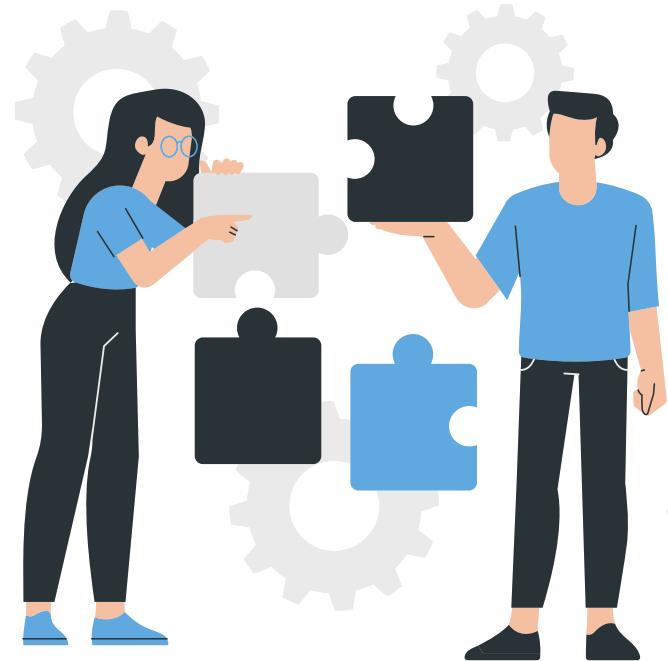
2

## **TASK 2: PACMAN**



# MÔ TẢ BÀI TOÁN

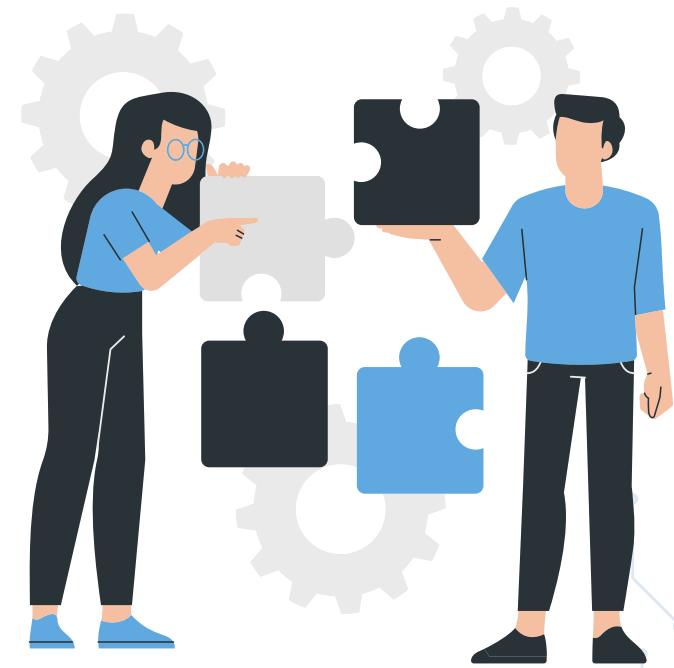
- **Mục tiêu:** Pacman ăn hết thức ăn (.) và tới cổng E với tổng số bước tối thiểu.
- **Trạng thái đích:**  $|foods|= 0$  và vị trí Pacman trùng với vị trí Exit ở hệ hiện tại.



# MÔ TẢ BÀI TOÁN

- **Quy luật:**

- *Hành động:* North, South, West, East. (Khi ghi file, hành động teleport được biểu diễn là Stop.)
- *Tường %:* Không đi xuyên, trừ khi đang có TTL > 0.
- *Bánh O:* Khi ăn, đặt TTL = 5. Trong TTL, Pacman đi xuyên và ăn tường (ô % biến thành ô trống vĩnh viễn).
- *Teleport 4 góc:* Khi đứng tại neo góc (ô đi được gần mỗi góc), có thể dịch chuyển tới bất kỳ neo góc khác (chi phí 1).
- *Ma G:* Di chuyển ngang mỗi bước, chạm tường đổi hướng; hàng kín đứng yên. Nếu Pacman đụng ma (trước/sau tick) hoặc swap cạnh trong cùng bước ⇒ trạng thái vô hiệu.
- *Xoay mè cung:* Mỗi 30 bước xoay 90 ° CW; toàn bộ toạ độ Pacman/food/pie/ghost/exit được nội suy tương ứng. Các ô tường đã bị “ăn” được bảo toàn (ghi theo toạ độ gốc)



# ĐỊNH HƯỚNG BÀI LÀM

## Không gian trạng thái

- Một trạng thái s gồm:

$s = \text{pacman}, \text{foods}, \text{pies}, \text{ghosts}, \text{ttl}, \text{steps\_mod30}, \text{rot\_idx}, \text{removed}$

- $\text{pacman}$ : tọa độ hiện tại ( $r, c$ ) ở hệ hiện tại.
- $\text{foods}, \text{pies}$ : tập tọa độ chưa ăn.
- $\text{ghosts}$ : ( $\text{pos}, \text{dir} \in \{-1, +1\}$ ).
- $\text{ttl}$ : số bước còn lại được xuyên/ăn tường.
- $\text{steps\_mod30}, \text{rot\_idx}$ : phục vụ xoay định kỳ.
- $\text{removed}$ : tập các ô tường đã bị phá (lưu theo tọa độ gốc) để quy chiếu ổn định qua xoay.

# ĐỊNH HƯỚNG BÀI LÀM

## Không gian trạng thái

- **Hành động A(s):**
  - Luôn có {N,S,W,E}; nếu Pacman ở neo góc  $\Rightarrow$  thêm {TUL, TUR, TBL, TBR} (được ưu tiên cho A\*).
- **Hàm chuyển s --(a)--> s':**
  - Đâm tường khi TTL = 0  $\Rightarrow$  invalid (không sinh trạng thái).
  - TTL > 0 đi vào %  $\Rightarrow$  hợp lệ & thêm ô tương ứng vào removed.
  - Ăn./O; O đặt TTL = 5.
  - Ma tick ngang; kiểm tra va chạm trước/sau/swap; mỗi bước tăng steps\_mod30 và xoay khi về 0.
- **Đích:** |foods|= 0 và pacman= Exit.



# THUẬT TOÁN A\*

- Dùng A\* với hàng đợi ưu tiên theo  $f = g + h$ , graph search với bảng best\_g[s]. Teleport được đặt lên đầu thứ tự hành động để A\* thử sớm. Các nhánh vi phạm va chạm ma bị loại do result() trả về None
- **Hàm heuristic:**
  - $h()$ : sử dụng BFS động để tính khoảng cách ngắn nhất giữa các điểm (có xét teleport và khả năng ăn tường)
- **Tính chất của heuristic:**
  - *Admissibility*: heuristic không bao giờ đánh giá quá thấp chi phí thực vì giá trị được tính dựa trên khoảng cách ngắn nhất hợp lệ.
  - *Consistency*: heuristic giảm đều theo chi phí thật giữa các trạng thái

# CHI TIẾT TRIỂN KHAI

- **pacman\_problem.py:** định nghĩa state, actions(), result(), is\_goal(), step\_cost(), phép xoay  $90^\circ / 30$  bước, tick ma, kiểm tra va chạm, tập removed (tường đã bị ăn).
- **heuristics.py:** PacmanMST heuristic; không dùng Manhattan/Euclid.
- **astar.py:** A\* chuẩn, graph-search, ưu tiên teleport.
- **experiments.py:** script chạy A\* và in thống kê.
- **\_main\_\_.py:** (nếu sử dụng) điều khiển Manual/AUTO, hiển thị HUD, replan nền khi xoay, ghi kết quả

# LỚP PacmanProblem

- **Mục đích:** Là lớp trung tâm, định nghĩa toàn bộ logic trò chơi và không gian trạng thái (state space).
- PacmanProblem mô tả toàn bộ logic bài toán Pacman động.
- Lớp này định nghĩa các hành động, trạng thái kế tiếp, kiểm tra đích và chi phí bước
- **Các nhóm phương thức chính:**

Nhóm	Tên phương thức	Chức năng
Khởi tạo	<code>__init__</code>	Đọc grid, khởi tạo state ban đầu
Trạng thái	<code>initial_state()</code>	Trả về state khởi đầu

# LỚP Pacman Problem

- Các nhóm phương thức chính:

Nhóm	Tên phương thức	Chức năng
Mục tiêu	is_goal(s)	Kiểm tra đã ăn hết food và tới Exit chưa
Hành động	actions(s)	Trả về các hành động hợp lệ (N, S, E, W, Teleport)
Kết quả	result(s,a)	Sinh state mới sau khi thực hiện hành động
Chi phí	step_cost(s,a,s2)	Trả về chi phí di chuyển
Tiện ích nội bộ	_rotate_world, _move_ghosts_dyn, _corner_anchor_positions	Xử lý logic động như xoay map, ma di chuyển, teleport, phá tường

# LỚP Heuristic Pacman MST

- Ký hiệu  $d_{\text{maze}}(u, v)$  là khoảng cách mê cung (BFS) trên lưới hiện tại (đã loại các tường bị phá). Gọi  $A$  là tập neo góc.
- **Khoảng cách cặp có xét teleport (lower bound).**

$$d(u, v) = \min \left( d_{\text{maze}}(u, v), \min_{a \in A} [d_{\text{maze}}(u, a) + 1 + \min_{b \in A} d_{\text{maze}}(b, v)] \right)$$

→ Trong đó  $+1$  là chi phí một bước teleport.

- Heuristic tổng hợp bằng MST. Gọi  $P = \{\text{Pacman}\} \cup \text{foods} \cup \{\text{Exit}\}$ . Heuristic là tổng trọng số cây khung nhỏ nhất (MST) của đồ thị đầy đủ trên  $P$  với trọng số cạnh  $d(\cdot, \cdot)$ . Vì  $d$  là lower bound khoảng cách thực, tổng MST là lower bound của chi phí tối thiểu còn lại  $\Rightarrow$  admissible. Thực nghiệm cho thấy heuristic ổn định/consistent.

# MÃ GIẢ A\*

```
function AStar(problem, h)
    s0 ← problem.initial_state()
    OPEN ← priority queue by  $f = g + h$ 
    best_g ← map with  $+\infty$ 

    push ( $f = h(s_0)$ ,  $g = 0$ ,  $s = s_0$ ) vào OPEN
    best_g[ $s_0$ ] ← 0

    while OPEN not empty do
        ( $f$ ,  $g$ ,  $s$ ) ← pop_min(OPEN)

        if problem.is_goal( $s$ ) then
            return ReconstructPath( $s$ )
        end if

        for  $a \in$  problem.actions( $s$ ) do
             $s' \leftarrow$  problem.result( $s$ ,  $a$ )
            if  $s' =$  None then
                continue
            end if

             $g' \leftarrow g +$  problem.step_cost( $s$ ,  $a$ ,  $s'$ )

            if  $g' <$  best_g.get( $s'$ ,  $+\infty$ ) then
                best_g[ $s'$ ] ←  $g'$ 
                 $f' \leftarrow g' + h(s')$ 
                push ( $f'$ ,  $g'$ ,  $s'$ ) vào OPEN
            end if
        end for
    end while

    return failure
end function
```

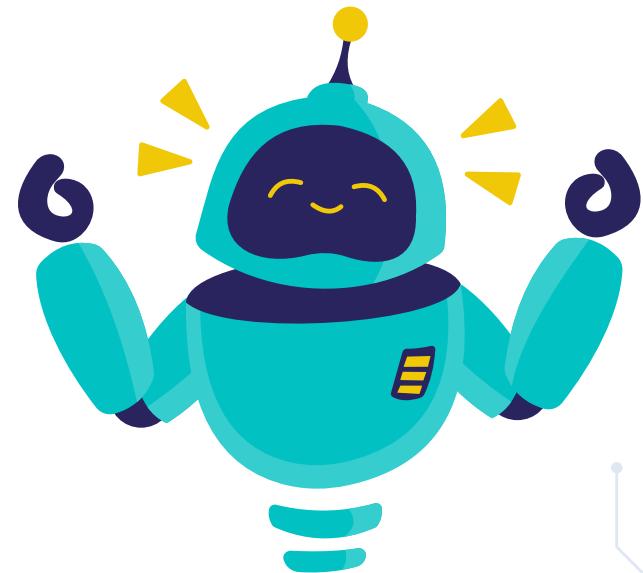
# ƯU ĐIỂM VÀ NHƯỢC ĐIỂM

- **Ưu điểm**

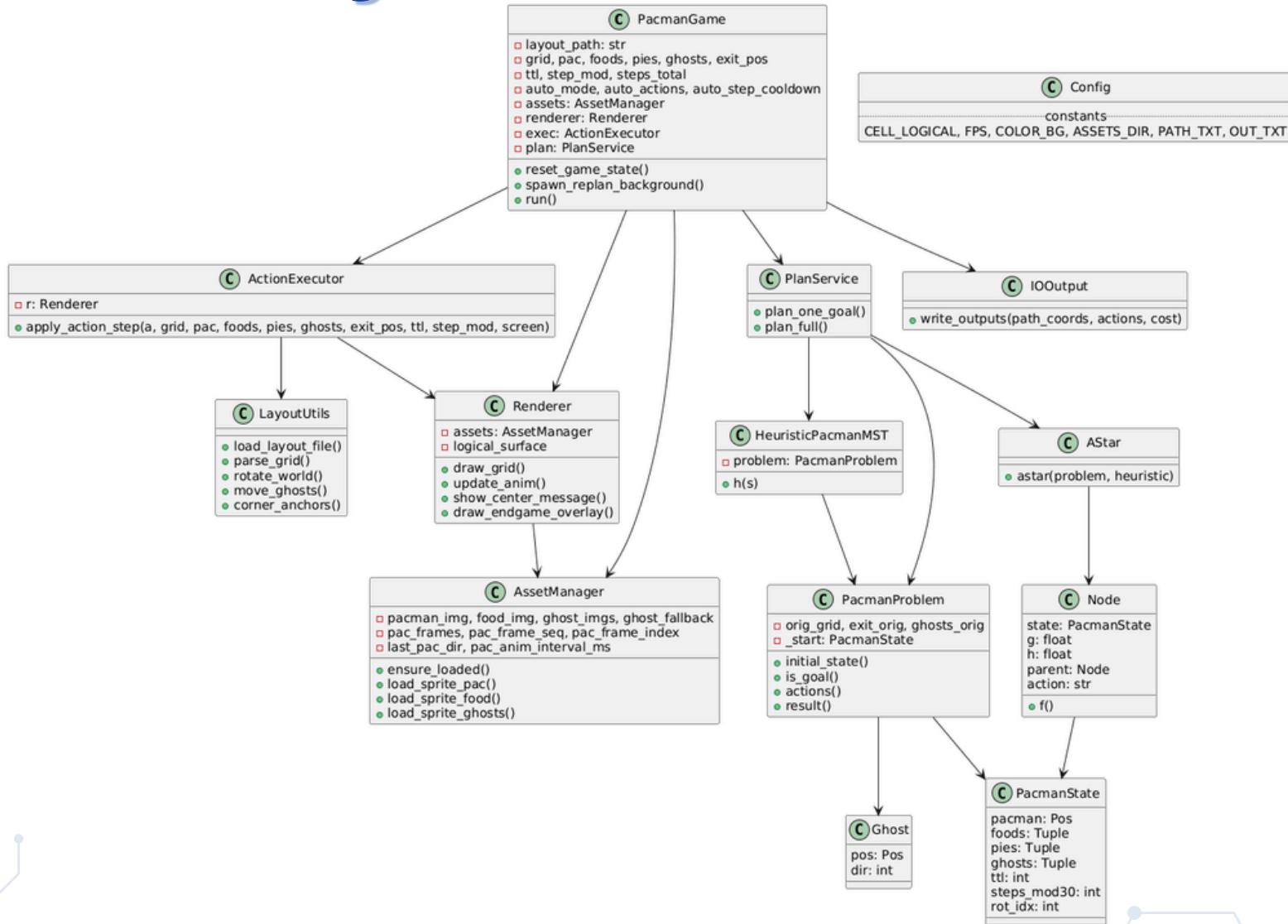
- Đáp ứng đủ động lực học phức tạp (teleport, TTL ăn tường, ma tick, xoay mê cung).
- Heuristic mạnh, admissible, không dùng Manhattan/Euclid; tận dụng teleport & tường đã phá.
- Tránh va chạm ma nhờ mã hoá quy tắc trong result().

- **Nhược điểm**

- Không gian trạng thái lớn (có removed, rot\_idx, ghost) khiến A\* tốn tài nguyên trên bản đồ rất lớn. Có thể dùng giới hạn max\_expanded, replan theo chặng, và cache BFS để giảm tải.



# Class diagram



# THANK YOU FOR LISTENING

