

Bài 2: A* với Pacman

Ngày 19 tháng 10 năm 2025

1 Mô tả bài toán

- **Mục tiêu:** Pacman ăn hết thức ăn (.) và tới cổng E với tổng số bước tối thiểu.
- **Trạng thái đích:** $|\text{foods}| = 0$ và vị trí Pacman trùng vị trí Exit ở hệ hiện tại.
- **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 $\text{TTL} > 0$.
 - **Bánh 0:** Khi ăn, đặt $\text{TTL} = 5$. 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 \Rightarrow trạng thái vô hiệu.
 - **Xoay mê cung:** Mỗi 30 bước xoay 90° CW; toàn bộ tọa độ Pacman/food/pie/ghost/exit được nội suy tương ứng. Các ô tường đã bị “ăn” được bảo toàn (ghi theo tọa độ gốc).

2 Không gian trạng thái

Một trạng thái s gồm:

$$s = (\text{pacman}, \text{foods}, \text{pies}, \text{ghosts}, \text{ttl}, \text{steps_mod30}, \text{rot_idx}, \text{removed})$$

- **pacman:** tọa độ hiện tại (r, c) ở hệ hiện tại.
- **foods, pies:** tập tọa độ chưa ăn.
- **ghosts:** $(\text{pos}, \text{dir} \in \{-1, +1\})$.
- **ttl:** số bước còn lại được xuyên/ăn tường.
- **steps_mod30, rot_idx:** phục vụ xoay định kỳ.
- **removed:** tập các ô tường đã bị phá (lưu theo *tọa độ gốc*) để quy chiếu ổn định qua xoay.

Hành động $\mathcal{A}(s)$:

- Luôn có $\{N, S, W, E\}$; nếu Pacman ở neo góc \Rightarrow thêm $\{TUL, TUR, TBL, TBR\}$ (được ưu tiên cho A*).

Hàm chuyển $s \xrightarrow{a} s'$:

- 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 ./0; 0 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 $pacman = \text{Exit}$.

3 Thuật toán A*

3.1 Mô tả

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**.

3.2 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 \mathcal{A} là tập neo góc.

Khoảng cách cặp có xét teleport (lower bound).

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

Trong đó +1 là chi phí một bước teleport.

Heuristic tổng hợp bằng MST. Gọi $P = \{\text{Pacman}\} \cup \text{foods} \cup \{\text{Exit}_{\text{hiện tại}}\}$. 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.

3.3 Mã giả

Algorithm 1 A* cho Pacman

```
1: function ASTAR(problem, h)
2:    $s_0 \leftarrow \text{problem.initial\_state}()$ 
3:   OPEN  $\leftarrow$  priority queue by  $f = g + h$ ;  $\text{best\_g} \leftarrow$  map with  $+\infty$ 
4:   push ( $f = h(s_0), g = 0, s = s_0$ ) vào OPEN;  $\text{best\_g}[s_0] \leftarrow 0$ 
5:   while OPEN is not empty do
6:     ( $f, g, s$ )  $\leftarrow$  pop_min(OPEN)
7:     if problem.is_goal(s) then
8:       return RECONSTRUCTPATH(s)
9:     end if
10:    for  $a \in \text{problem.actions}(s)$  do
11:       $s' \leftarrow \text{problem.result}(s, a)$ 
12:      if  $s' = \text{None}$  then
13:        continue
14:      end if
15:       $g' \leftarrow g + \text{problem.step\_cost}(s, a, s')$ 
16:      if  $g' < \text{best\_g.get}(s', +\infty)$  then
17:         $\text{best\_g}[s'] \leftarrow g'$ 
18:         $f' \leftarrow g' + h(s')$ 
19:        push ( $f', g', s'$ ) vào OPEN
20:      end if
21:    end for
22:  end while
23:  return failure
24: end function
```

4 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`: heuristic *BFS + teleport-aware + MST (Prim)*; không dùng Manhattan/Euclid.
- `astar.py`: A* chuẩn, graph-search, ưu tiên teleport.
- `experiments.py`: script chạy A* và in thống kê; có thể ghép nhiều chặng (ăn từng food rồi về Exit).
- `gui/main.py`: (nếu sử dụng) điều khiển Manual/AUTO, hiển thị HUD, replan nền khi xoay, ghi kết quả.

5 Đầu ra & định dạng file

- **output.txt** (chỉ khi hoàn thành):

```
cost: <tong_so_buoc>
actions:
North
West
...
```

Teleport được ghi là **Stop**.

- **path.txt**: mỗi dòng là một toạ độ r, c của Pacman sau mỗi bước.

6 Ưu & 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.