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:** |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 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 mê 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).

2 Không gian trạng thái

Một trạng thái s gồm:

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

- `pacman`: toạ độ hiện tại (r, c) ở hệ hiện tại.
- `foods`, `pies`: tập toạ độ chưa ăn.
- `ghosts`: (pos, 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 *toạ 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 ⇒ 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: $|\text{foods}| = 0$ và pacman = 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).

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

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

```
function AStar(problem, h)
    s_0 ← problem.initial_state()
    OPEN ← priority queue by f = g + h;   best_g ← map with +∞
    push (f = h(s_0), g = 0, s = s_0) vào OPEN;   best_g[s_0] ← 0
    while OPEN không rỗng do
        (f, g, s) ← pop_min(OPEN)
        if problem.is_goal(s) then
            return ReconstructPath(s)
        end if
        for a ∈ problem.actions(s) do
            s' ← problem.result(s, a)
            if s' = None then
                continue
            end if
            g' ← g + problem.step_cost(s, a, s')
            if g' < best_g.get(s', +∞) then
                best_g[s'] ← g'
                f' ← g' + h(s')
                push (f', g', s') vào OPEN
            end if
        end for
    end while
    return failure
end function
```

4 Chi tiết triển khai

- `pacman_problem.py`: định nghĩa state, `actions()`, `result()`, `is_goal()`, `step_cost()`, phép xoay 90°/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.