TRƯỜNG ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN
CƠ SỞ TRÍ TUỆ NHÂN TẠO

ĐỀ TÀI

SEARCH ALGORITHMS

GV hướng dẫn: Bùi Duy Đăng - Trần Quốc Huy

Nhóm sinh viên thực hiện:

Họ và tên	*MSSV*
Nguyễn Ngọc Như Huyền	21120475
Nguyễn Lê Tuấn Khải	20120503
Đỗ Đăng Huy	20120492
Nguyễn Hữu Khôi	19120261

TP. Hồ Chí Minh, 2023

Mục lục

1 Giới thiệu chung

Pac-man World là trò chơi nổi tiếng được phát hành vào năm 1980 bởi *Namaco*. Trong trò chơi, có một nhân vật hình quả bóng màu vàng (yellow puck) tên là Pac-man, người chơi phải điều khiển nó trong mê cung để ăn tất cả các quả bóng (food) và phải tránh được những con ma.

Và trò chơi gồm 4 cấp độ:

- **Cấp độ thứ nhất (level 1):** Pac-Man biết vị trí của thức ăn trên bản đồ và không có con ma nào xuất hiện. Chỉ có một thức ăn trên bản đồ.
- **Cấp độ thứ hai (level 2):** Những con ma đứng yên và không di chuyển xung quanh. Nếu Pac-Man và một con ma va chạm với nhau, trò chơi kết thúc. Vẫn có một thức ăn trên bản đồ và Pac-Man biết vị trí của nó. *Những con ma đứng yên và không di chuyển xung quanh.*
- **Cấp độ thứ ba (level 3):** Tầm nhìn của Pac-Man bị giới hạn ở ba bước gần nhất. Thức ăn bên ngoài phạm vi này không hiển thị với Pac-Man. Pac-Man chỉ có thể quét các ô liền kề trong phạm vi 8 ô x 3 ô. Có nhiều thức ăn được rải khắp bản đồ. N*hững con ma có thể di chuyển một bước theo bất kỳ hướng hợp lệ nào xung quanh vị trí ban đầu của chúng khi bắt đầu trò chơi.* Cả Pac-Man và những con ma đều di chuyển một bước mỗi lượt.
- **Cấp độ thứ tư (level 4):** *Bao gồm một bản đồ kín nơi những con ma không ngừng truy đuổi Pac-Man.* Pac-Man phải thu thập càng nhiều thức ăn càng tốt trong khi tránh bị bất kỳ con ma nào bắt kịp. Những con ma có khả năng đi xuyên qua nhau. Cả Pac-Man và những con ma đều di chuyển một bước mỗi lượt và bản đồ chứa rất nhiều thức ăn.

Trong project này, chúng tôi tự động hóa cho nhân vật Pac-man bằng cách mô hình hóa bài toán cho từng level và áp dụng các thuật toán tìm kiếm để so sánh hiệu suất của các thuật toán.

2 Bảng phân chia công việc

STT	Công việc	Thành viên thực hiện
1	Triển khai các thuật toán cho lv1 & lv2	Tất cả
2	Triển khai các thuật toán cho lv3	Huyền, Huy
3	Triển khai các thuật toán cho lv4	Huyền, Khải
4	Thiết kế các Map	Khôi, Khải

3 Đánh giá mức độ hoàn thành của các yêu cầu

STT	Đánh giá	Mức độ hoàn thành
1	Hoàn thành level 1.	100%
2	Hoàn thành level 2.	100%
3	Hoàn thành level 3.	100%
4	Hoàn thành level 4.	100%
5	Hiển thị đồ họa cho từng bước chạy của quá trình.	100%
6	Thiết kế map với cấu trúc và số lượng tường, thức ăn và quái khác nhau.	100%
7	Báo cáo thuật toán và thử nghiệm.	100%

- Level 1 và 2: Hoàn thành và triển khai được 4 thuật toán khác nhau.
- Level 3: Hoàn thành và triển khai được 1 thuật toán A*.
- Level 4: Hoàn thành và triển khai được 1 thuật toán alpha-beta search.

4 Môi trường triển khai và cách thực thi

4.1 Môi trường triển khai

- Python 3.10.x trở lên
- Các thư viện gồm argparse, tk, pillow (thông tin chi tiết nằm trong file 'Pipfile')

4.2 Cách cài đặt môi trường thực thi

- **Windows (cmd)**
  Set-ExecutionPolicy RemoteSigned -Scope Process
  py -3 -m venv venv or py -3 -m virtualenv venv
  ./venv/Scripts/activate
  pip install pipenv
  pipenv install
- **Linux**
  python3 -m venv venv or python -m virtualenv venv (if python ~ python3)
  source venv/bin/activate
  pip install pipenv
  pipenv install

4.3 Cách thực thi

Chúng ta chạy hàm main.py với cấu trúc sau: *<python/py -3> main.py –lv <level?> –build_in_map <map?.txt> –map <map path> –algo <algo name>*

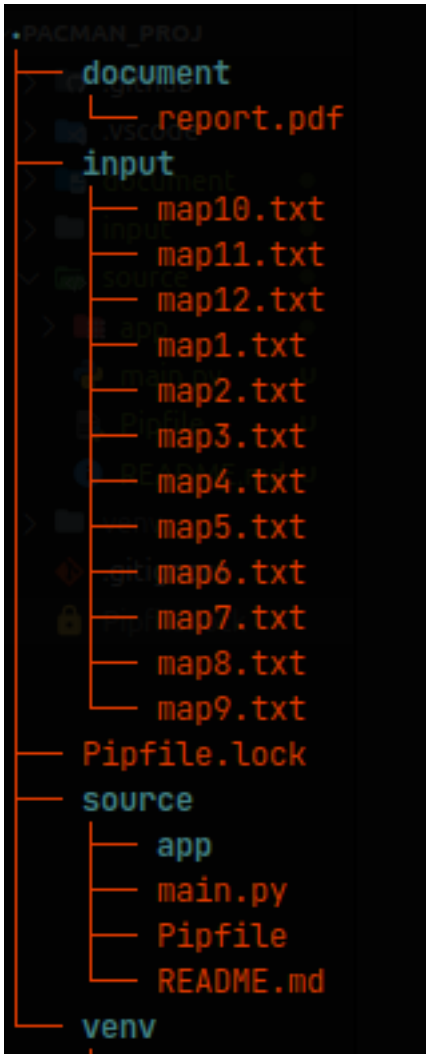
Lưu ý:

- Nếu dùng build_in_map thì không dùng map path vì (build in map) là các map có sẵn trong thư mục app/maps. Và map path là đường dẫn (tuyệt đối) đến map bất kỳ.
- Với algo name tùy thuộc vào level mà bạn muốn. Với level 1 và level 2 thì algo name sẽ là một trong (ucs, bfs, dfs, a_star), với level 3 thì algo name phải là ucs và với level 4 algo name phải là alpha_beta_search.
- Ví dụ 1: python main.py –lv 3 –map /home/harikngnguyen/.project/intro_AI/pacman_proj/input/map1.txt –algo a_star_search
- Ví dụ 2: python source/main.py –lv 3 –build_in_map map5.txt –algo a_star_search

4.4 Quản lý đồ án

Quản lý thông qua Github cụ thể như sau: Github Link

5 Cấu trúc thư mục của đồ án

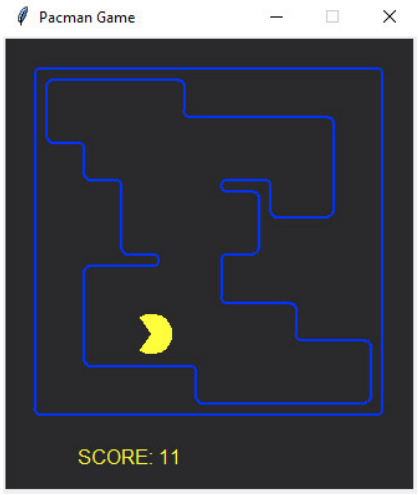


Hình 1: Cấu trúc thư mục đồ án

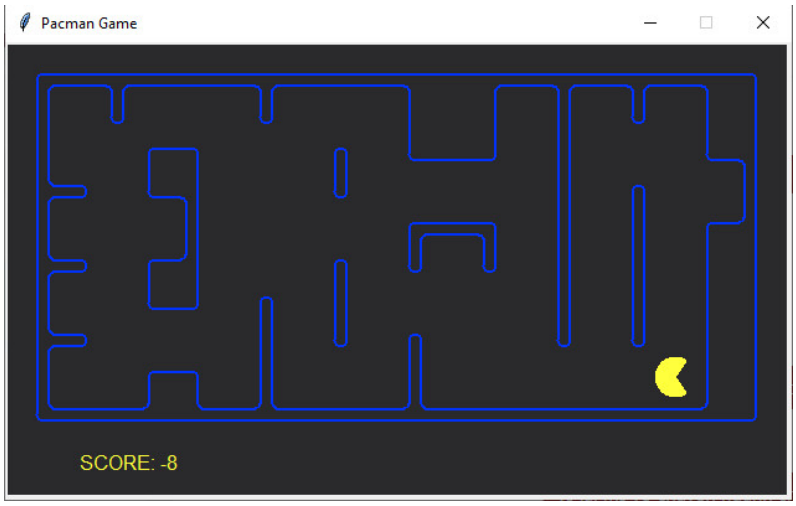
Trong đó thư mục *app* chứa các module để thực thi dự án (gồm constants chứa các hằng số, *graphic* là module để xử lý graphic, *maps* chứa các file map được thiết kế sẵn, *search_ algo* gồm 4 modules con gồm các thuật toán cho từng level, *settings* modules chứa các cài đặt của dự án, *utils* modules chứa các thư viện sử dụng chung) và file *main.py* là file thực thi các thuật toán.

6 Các bản đồ *map* được thiết kế sẵn

- Các map dùng cho level 1: map11, map12

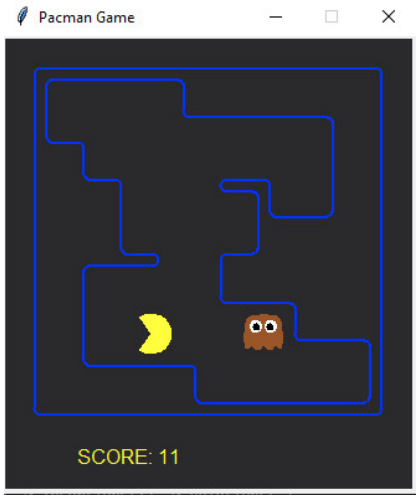


Hình 2: map11

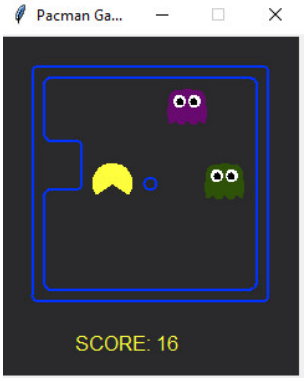


Hình 3: map12

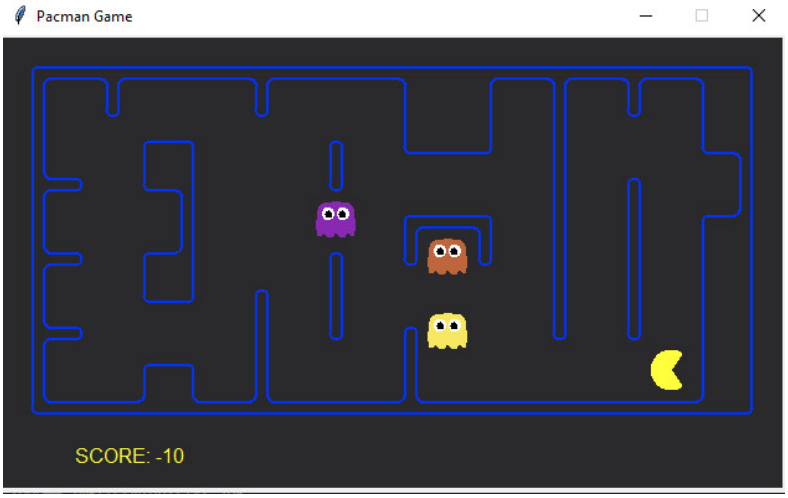
- Các map dùng cho level 2: map1, map2, map3



Hình 4: map1

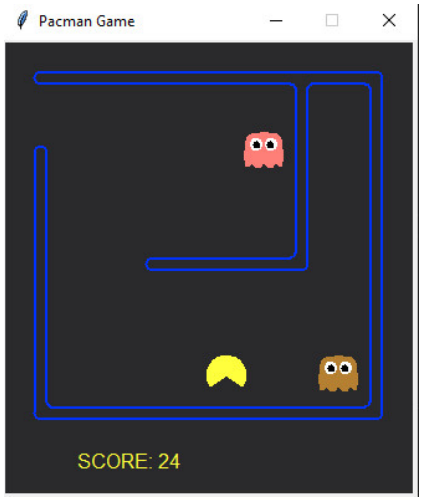


Hình 5: map2

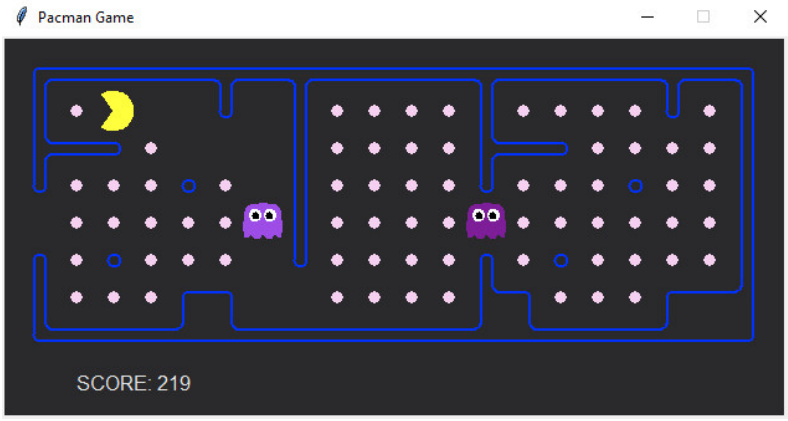


Hình 6: map3

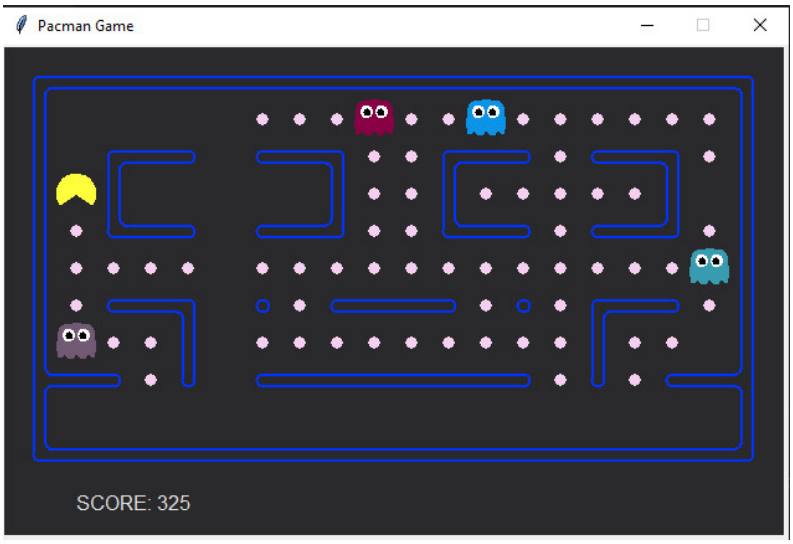
- Các map dùng cho level 3 và level 4: map4, map5, map6, map7, map8, map9, map10



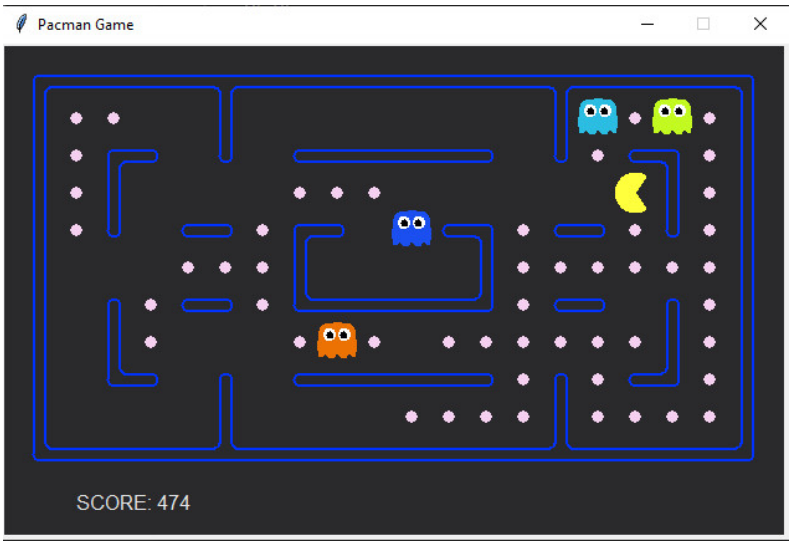
Hình 7: map4



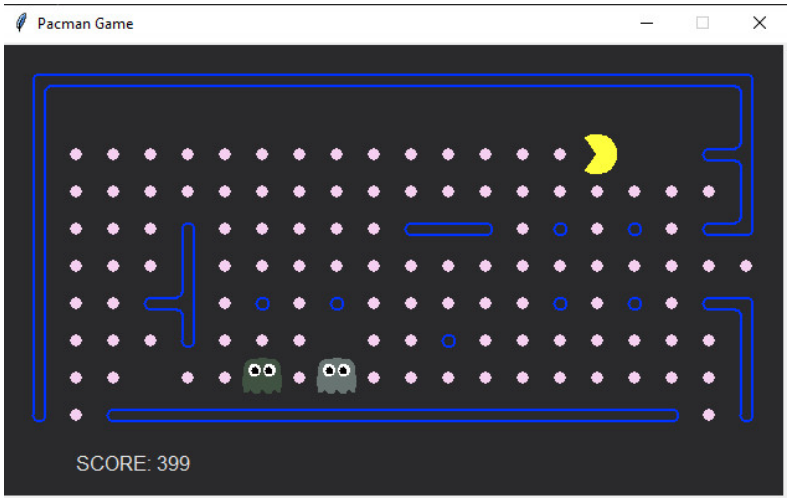
Hình 8: map5



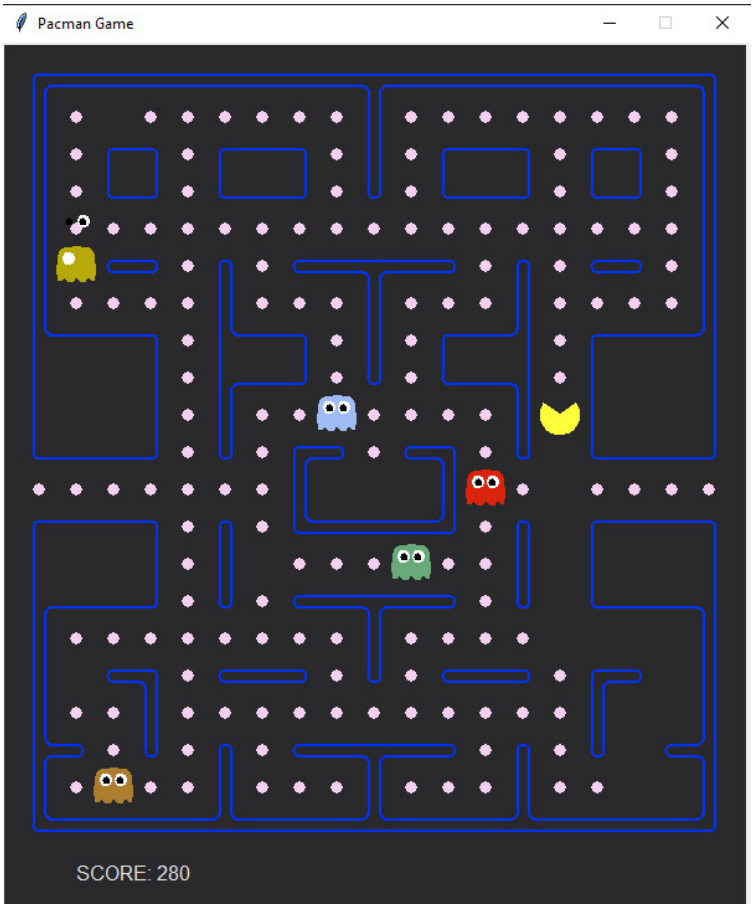
Hình 9: map6



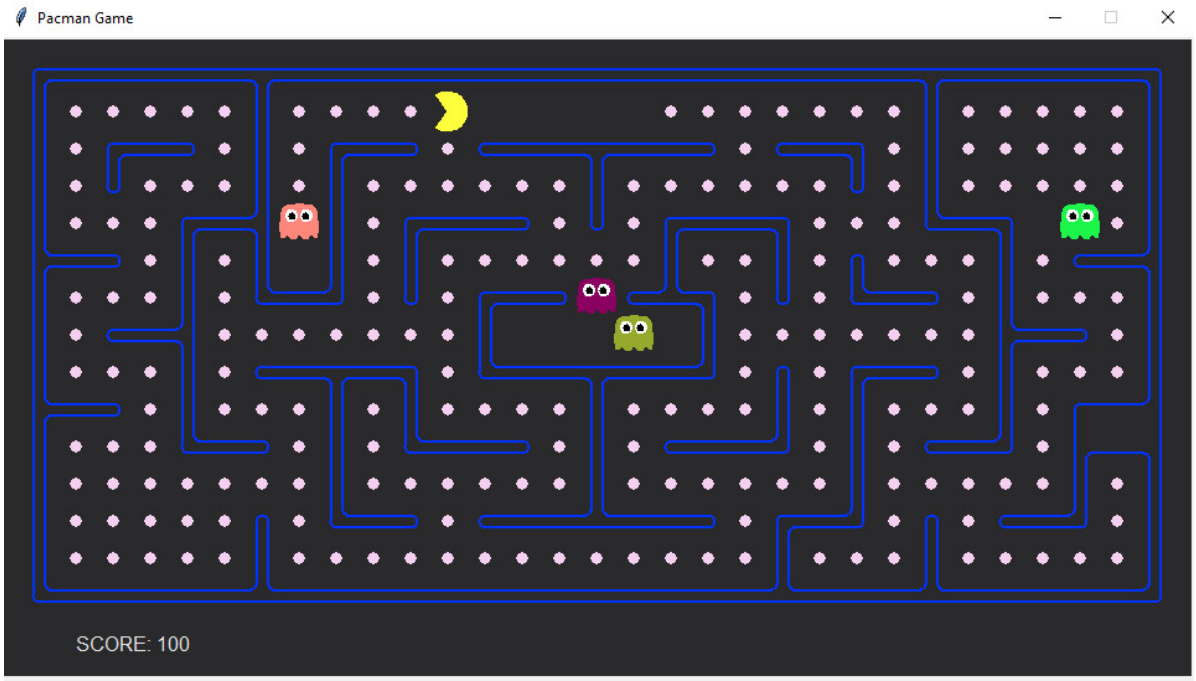
Hình 10: map7



Hình 11: map8



Hình 12: map9



Hình 13: map10

7 Các thuật toán cho level 1

7.1 Mô tả bài toán

Với level 1, chỉ có duy nhất một đồ ăn (food) trong map, map không có ma (ghosts), tác nhân (agent) có thể nhìn thấy toàn map để pacman có thể tìm thấy được food.

Với bài toán này ta có thể áp dụng các thuật toán tìm kiếm như BFS, DFS, UCS, A* để tìm ra đường đi ngắn nhất từ vị trí bắt đầu của pacman đến vị trí của food.

7.2 Các hướng giải quyết

7.2.1 BFS - Breadth First Search

- BFS là thuật toán tìm kiếm theo chiều rộng, bắt đầu từ vị trí hiện tại của pacman và duyệt tất cả các đỉnh lân cận. Nếu đỉnh lân cận là vị trí của food, thì thuật toán sẽ trả về đường đi từ vị trí hiện tại đến vị trí của food.
- Ưu điểm của BFS là đơn giản và dễ triển khai. Nhược điểm của BFS là có thể dẫn đến đường đi dài, đặc biệt là trong các map phức tạp.

7.2.2 DFS - Depth First Search

- DFS là thuật toán tìm kiếm theo chiều sâu, bắt đầu từ vị trí hiện tại của pacman và duyệt tất cả các đỉnh con của đỉnh hiện tại. Nếu một đỉnh con là vị trí của food, thì thuật toán sẽ trả về đường đi từ vị trí hiện tại đến vị trí của food.

- Ưu điểm của DFS là có thể tìm ra đường đi ngắn nhất trong các map phức tạp. Nhược điểm của DFS là có thể dẫn đến việc duyệt quá sâu vào các nhánh không cần thiết, dẫn đến tốn thời gian.

7.2.3 UCS - Uniform Cost Search

- UCS là thuật toán tìm kiếm theo chi phí, bắt đầu từ vị trí hiện tại của pacman và duyệt tất cả các đỉnh với chi phí thấp nhất. Nếu một đỉnh có chi phí thấp nhất là vị trí của food, thì thuật toán sẽ trả về đường đi từ vị trí hiện tại đến vị trí của food.

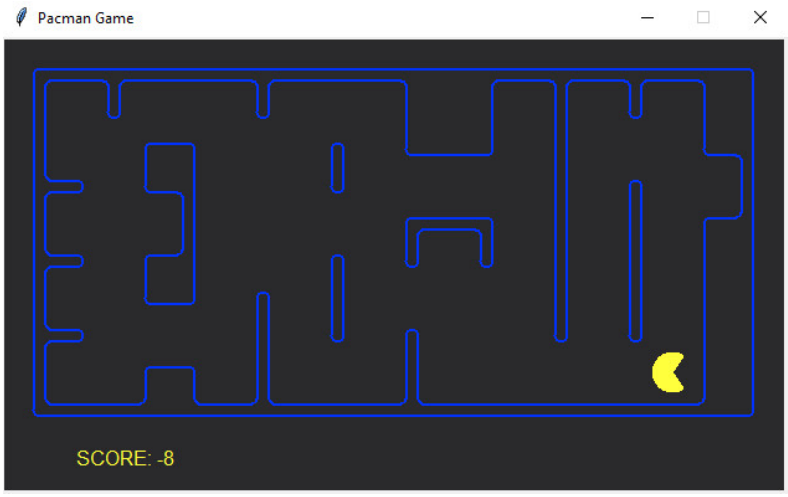
- Ưu điểm của UCS là có thể tìm ra đường đi ngắn nhất trong các map phức tạp. Nhược điểm của UCS là có thể dẫn đến việc duyệt quá sâu vào các nhánh không cần thiết, dẫn đến tốn thời gian.

7.2.4 A*

- A* là thuật toán tìm kiếm theo chi phí có trọng số, bắt đầu từ vị trí hiện tại của pacman và duyệt tất cả các đỉnh với chi phí thấp nhất. Chi phí của một đỉnh được tính bằng tổng của chi phí từ vị trí hiện tại đến đỉnh đó và chi phí từ đỉnh đó đến vị trí của food. Nếu một đỉnh có chi phí thấp nhất là vị trí của food, thì thuật toán sẽ trả về đường đi từ vị trí hiện tại đến vị trí của food.

- Ưu điểm của A* là có thể tìm ra đường đi ngắn nhất trong các map phức tạp. Nhược điểm của A* là có thể tốn nhiều thời gian để triển khai. Thuật toán A* phải cần một hàm heurictic đủ tốt để tìm kiếm hiệu quả.

7.3 Kết quả so sánh



Hình 14: Map level 1

```
(venv) PS D:\IntroAI\Projects\intro_AI_pacman_prj> python main.py --lv 1 --build_in_map map12.txt --algo bfs
Execution time:  0.0050013065338134766
(venv) PS D:\IntroAI\Projects\intro_AI_pacman_prj> python main.py --lv 1 --build_in_map map12.txt --algo dfs
Execution time:  0.006001949310302734
(venv) PS D:\IntroAI\Projects\intro_AI_pacman_prj> python main.py --lv 1 --build_in_map map12.txt --algo ucs
Execution time:  0.004998683929443359
(venv) PS D:\IntroAI\Projects\intro_AI_pacman_prj> python main.py --lv 1 --build_in_map map12.txt --algo a_star
Execution time:  0.006005048751831055
```

Hình 15: Kết quả thực thi các thuật toán level 1

Thuật toán	Thời gian	Điểm
BFS	0.005001	-8
DFS	0.0060019	-22
UCS	0.004998	-8
A*	0.006005	-8

Dựa vào thời gian thực thi trên, ta dễ dàng thấy BFS, UCS và A* có thời gian thực thi gần như là như nhau, riêng DFS lại tốn thời gian lâu hơn. Về đường đi thì thuật toán BFS, UCS và A* cho ra đường đi tối ưu hơn so với thuật toán DFS.

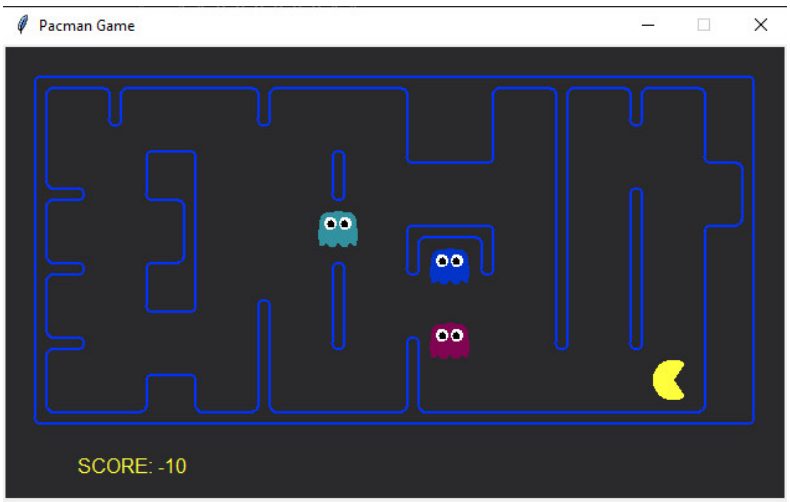
8 Các thuật toán cho level 2

8.1 Mô tả bài toán

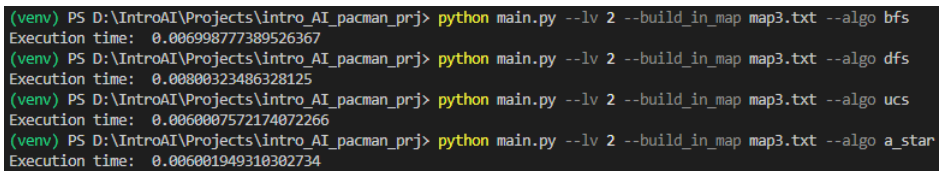
Tương tự như level 1 nhưng ở level 2 thì map xuất hiện những con ma (ghosts). Tuy nhiên, những con ma đứng yên. Ta có thể xem những con ma đó là những bức tường (wall).

Do đó, ta có thể áp dụng các thuật toán ở level 1 (BFS, DFS, UCS, A*) để tìm đường đi ngắn nhất từ vị trí bắt đầu của pacman đến vị trí của food.

8.2 Kết quả so sánh



Hình 16: Map level 2



Hình 17: Kết quả thực thi các thuật toán level 2

Thuật toán	Thời gian	Điểm
BFS	0.006998	-10
DFS	0.008003	-20
UCS	0.006000	-10
A*	0.0060019	-10

Dựa vào thời gian thực thi trên, ta dễ dàng thấy BFS, UCS và A* có thời gian thực thi gần như là như nhau, riêng DFS lại tốn thời gian lâu hơn. Về đường đi thì thuật toán BFS, UCS và A* cho ra đường đi tối ưu hơn so với thuật toán DFS.

9 Các thuật toán cho level 3

9.1 Mô tả bài toán

Ở level 3, Pacman sẽ bị giới hạn tầm nhìn trong hình vuông 7x7, với tâm hình vuông là vị trí hiện tại của Pacman. Thức ăn sẽ được rải tại nhiều nơi trong map và nhiệm vụ của Pacman là cố gắng ăn hết các thức ăn này và tránh đụng phải ma.

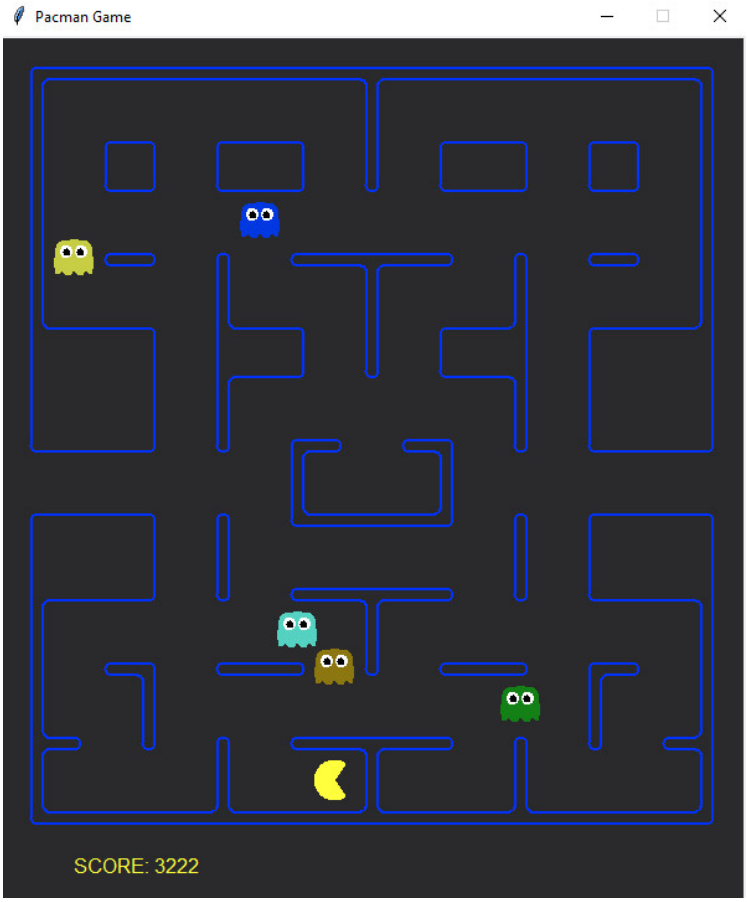
9.2 Hướng giải quyết

Sử dụng thuật toán tìm kiếm A* tìm thức ăn nằm trong phạm vi Pacman nhìn thấy và gần Pacman nhất. Trường hợp không có thức ăn trong phạm vi nhìn thấy, Pacman sẽ chọn goal là điểm mù gần nhất. Mỗi bước di chuyển của pacman sẽ cập nhật lại bản đồ của pacman, cập nhật lại những điểm mù. Với ghost đã được cập nhật trong bản đồ pacman thấy được mà xa pacman hơn 3 bước thì sẽ cập nhật lại là road. Khi lại gần ghost thì sẽ chuyển hướng đi khác.

Mỗi lần Pacman ăn thức ăn thành công ta sẽ cập nhật lại map (vị trí của thức ăn trở thành đường đi).

Để cập nhật đường đi cho từng ghost riêng biệt, ta sẽ khởi tạo danh sách các dictionary với key là vị trí bắt đầu của ghost và value là danh sách các vị trí mà ghost đã đi qua.

9.3 Kết quả



Hình 18: Kết quả với map 1

10 Các thuật toán cho level 4

10.1 Mô tả bài toán

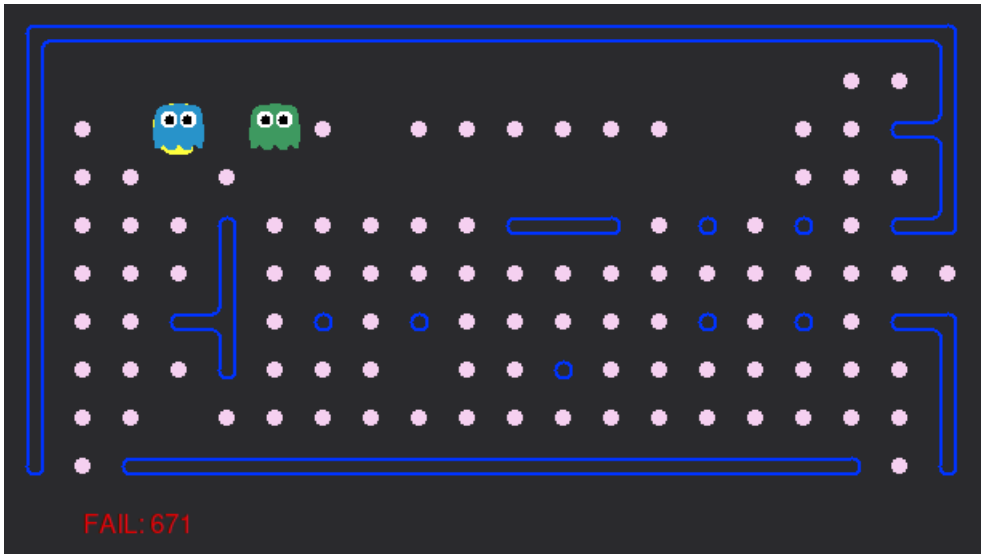
Ở level 4, trong bản đồ đóng nơi những con ma không ngừng truy đuổi Pacman, và trong bản đồ chứa nhiều thức ăn. Những con ma có thể đi xuyên qua nhau, và Pacman phải ăn được nhiều thức ăn nhất có thể và tránh được những con ma.

10.2 Hướng giải quyết

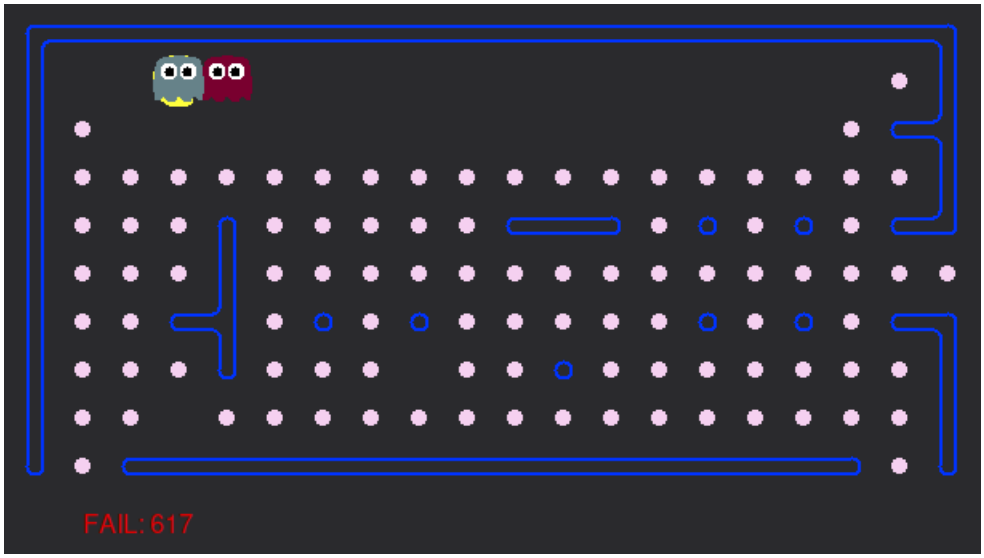
Sử dụng thuật toán tìm kiếm alpha - beta để dự đoán và tìm kiếm con đường tối ưu nhất (vì cây tìm kiếm quá sâu nên ở đây chúng tôi chỉ xét trong cây tìm kiếm với độ sâu bằng 3 - tức là Pacman dự đoán tối đa 2 nước sắp tới). Để mô phỏng bài toán cụ thể, những con ma (ghosts) sẽ tìm đường để bẫy pacman. Chúng tôi sử dụng thuật toán A để những con ma có thể tìm đường đến vị trí bẫy. Vị trí bẫy được tính toán là những vị trí mà trong vòng vài nước đi Pacman có thể đến di chuyển đến.

10.3 Kết quả

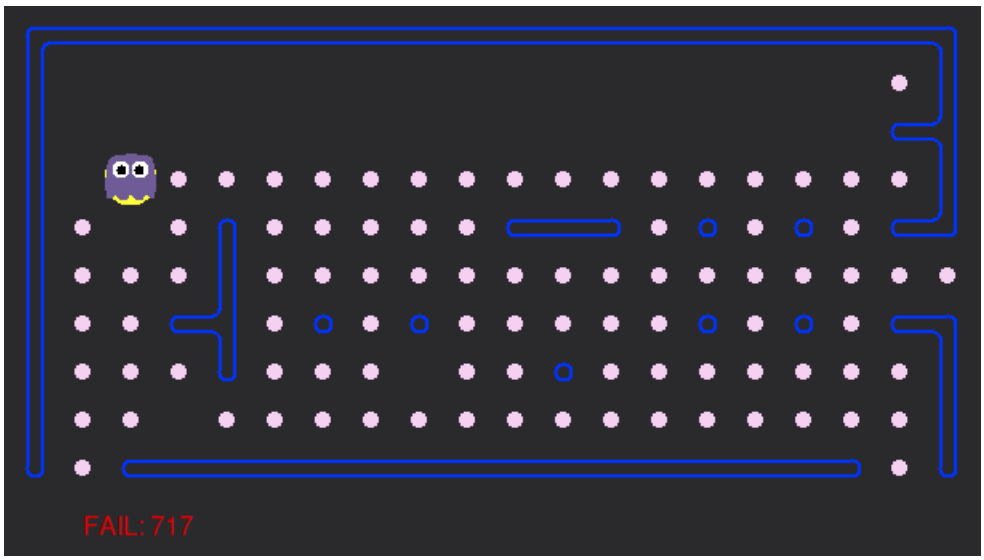
Do Pacman có thể dự đoán trong tối đa 2 nước, cho nên kết quả mỗi lần chạy sẽ khác nhau. Cụ thể như sau:



Hình 19: Kết quả với map 8 - lần 1



Hình 20: Kết quả với map 8 - lần 2



Hình 21: Kết quả với map 8 - lần 3

11 Tham khảo

- Slide bài giảng môn Cơ sở trí tuệ nhân tạo - GV Bùi Duy Đăng
- UC Berkeley AI Pacman Project