**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BÁO CÁO PROJECT 1**

**Robot Tìm Đường**

**MÔN CƠ SỞ TRÍ TUỆ NHÂN TẠO**



* **Thành viên nhóm:**

**Đào Thanh Thiện 1712782**

***Thuật toán A\*:***

***Thông tin chung***

-Thực hiện thao tác trên số tự nhiên.

- Toàn bộ thông tin của input sẽ được nạp vào class AstarGraph, trong đó có một ma trận 2 chiều Maze đóng vai trò như bản đồ.

- Các ô ở ngoài cùng (biên cạnh) của bản đồ sẽ được đánh dấu là không thể đi được.

- Rào chắn được vẽ tương đối khi di chuyển từ 1 đỉnh đến đỉnh khác, dựa trên cơ sở đường chim bay ngắn nhất. Rào chắn ưa tiên không vẽ vào các điểm đặc biệt như điểm Start, End hay các điểm Stop nên việc cùng các điểm nối của các rào chắn nhưng cho ra các hình dạng khác nhau có thể xãy ra.

- Các rào chắn được đánh dấu trực tiếp trên Maze, ngoài ra các điểm rào chắn được lưu vào list Barriers của class AstarGraph.

- Việc tìm đường đi không làm ảnh hưởng đến Maze, việc tìm đường đi chỉ nhận thông tin từ Maze chứ không trực tiếp thay đổi các giá trị trong Maze, kể cả việc đánh dấu đường đi tối ưa.

- Việc tìm kiếm cho phép đi chéo. Với chéo = 1.5 dọc/ngang.

***Thông tin của thuật toán tìm kiếm A\****

-Thuật toán này đánh giá một nút dựa trên chi phí đi từ nút góc đến nút đó – G(n), cộng với chi phí ước lượng ngắn nhất từ nút đó đến nút đích H(n).

F(n) = G(n) + H(n)

H(n) được tính bằng khoảng cách chim bay (direct distance) giữa điểm đó đến điểm đích. Luôn nhỏ hơn hoặc bằng chi phí thực sự đi từ điểm đó đến điểm đích nên đây là hàm heuristic chấp nhận được.

Việc tìm được đường đi mà không dùng đánh dấu trên bản đồ bằng cách để một List cameFrom lưu lại vị trí của điểm trước đó. Khi thuật toán tìm được đường điểm đích thì chỉ cần tìm lại trong list cameFrom để tìm đường đi.

***Phương pháp phát hiện việc c ó đi xuyên đa giác hay không***

Xem xét 2 vị trí ngang, dọc cạnh bên cùng hướng với điểm cần đến:

+ nếu cả 2 điểm cạnh bên cùng không thể đi được (vì có tường chắn, rào chắn) thì không thể đi chéo được

+ nếu cả 2 điểm cạnh bên có 1 hoặc cả 2 điểm có thể đi được (vì có tường chắn, rào chắn) thì có thể đi chéo được

|  |  |
| --- | --- |
| **Hình minh họa** | **Kết quả** |
| |  |  |  | | --- | --- | --- | |  | X |  | |  |  | X | |  |  |  | | Không thể đến |
| |  |  |  | | --- | --- | --- | |  | X |  | |  |  | X | |  |  |  | | Có thể đến |

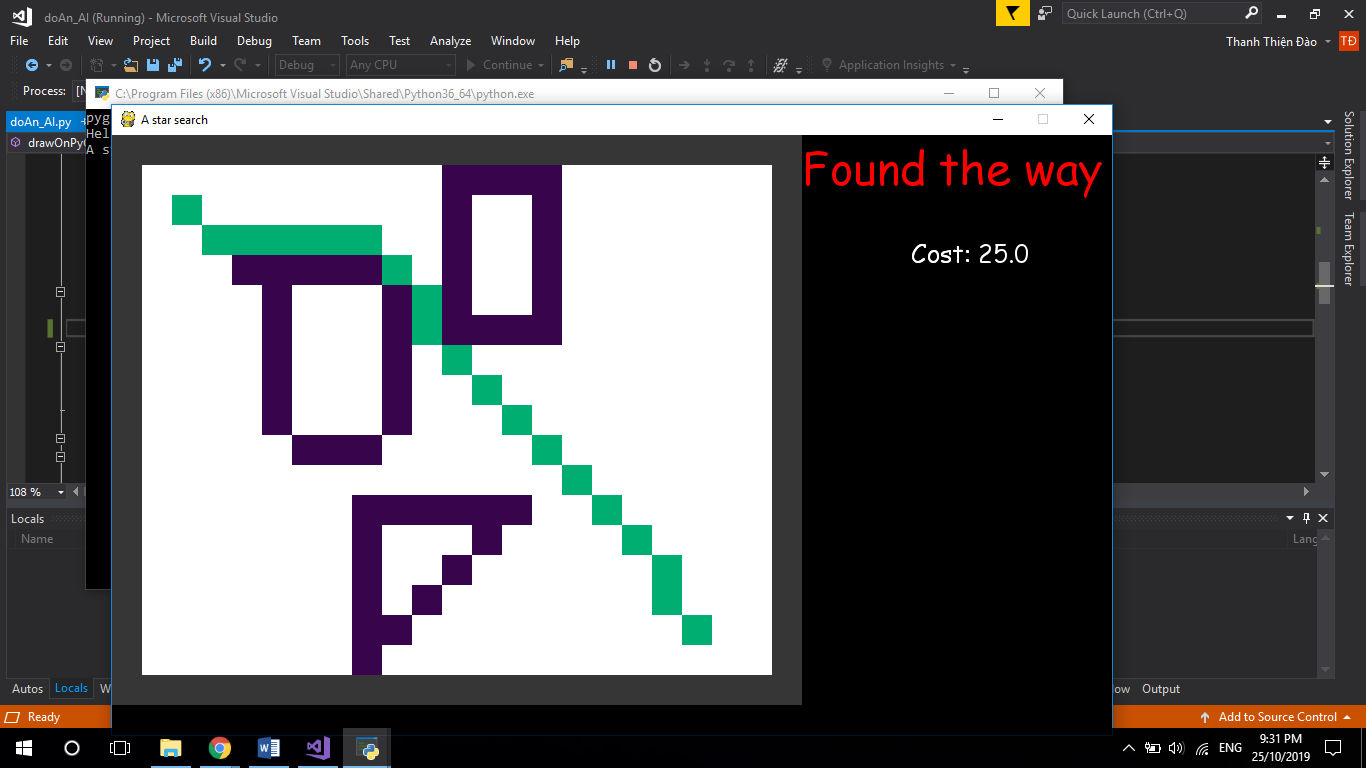
Chú Thích :

* Đen : Vị trí rào chắn , tường
* Đỏ : Vị trí hiện tại
* Xanh : Vị trí cần đến
* Trắng : Vị trí trống

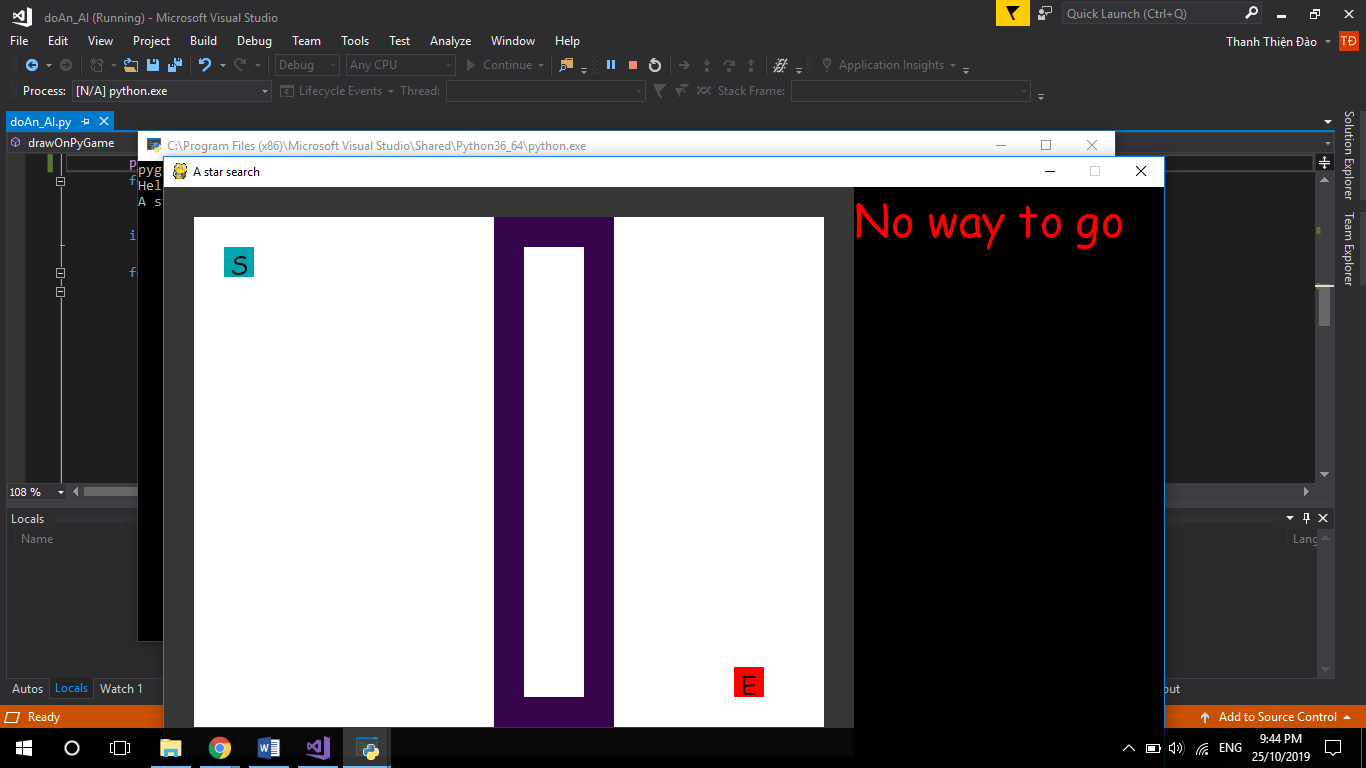
***Yêu cầu tìm đường đi***

1.Không điểm đón:

Vd minh họa:

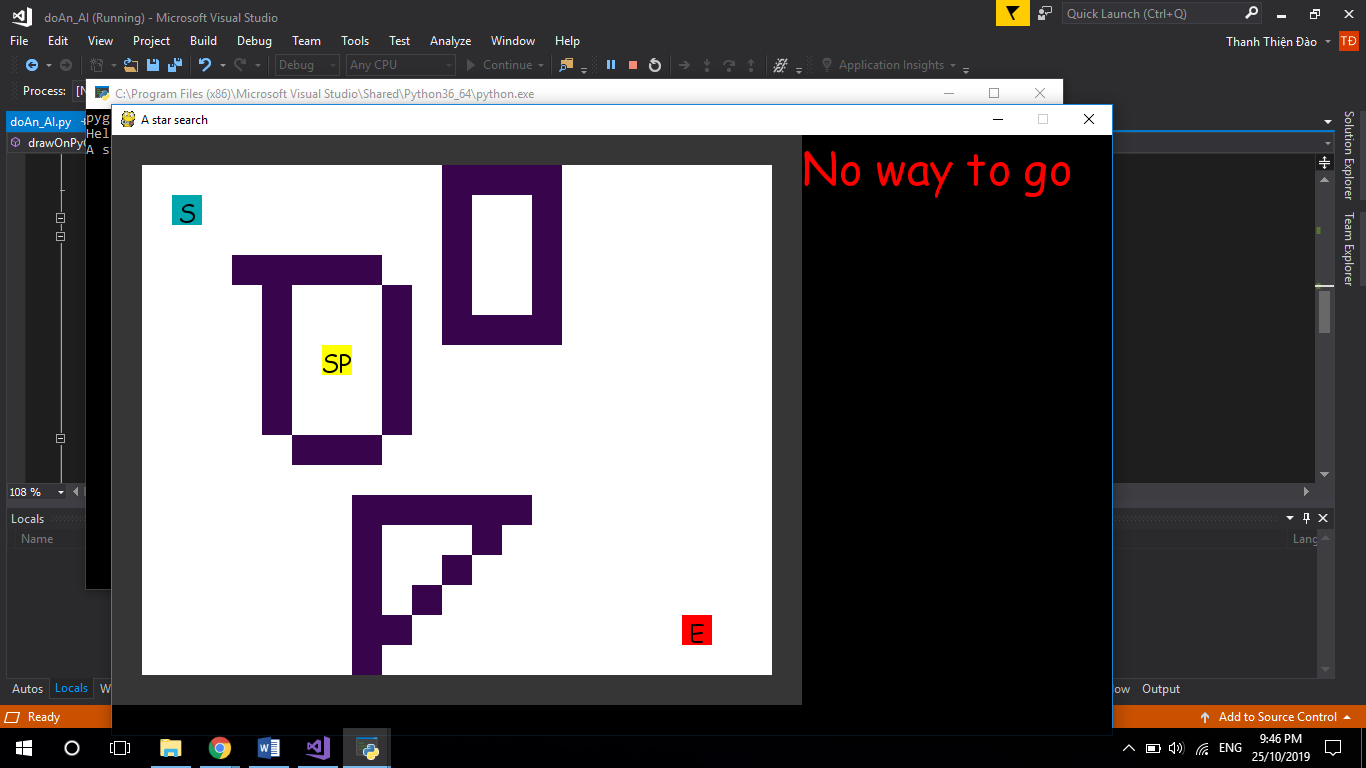


Vd minh họa:



2. Có 1 có điểm đón

Vd minh họa:



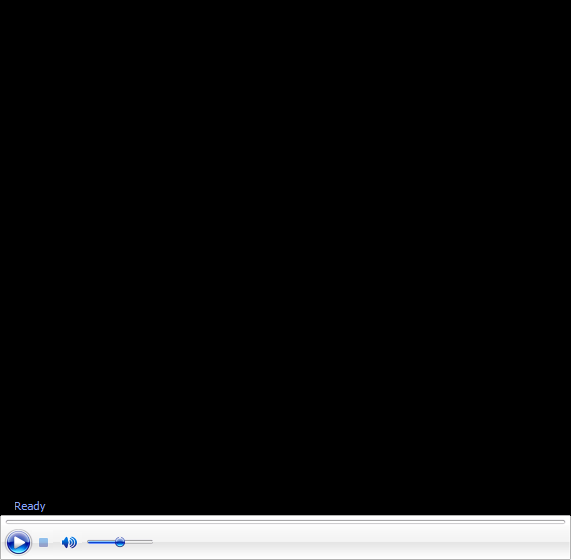
3.Có nhiều điểm đón:

-Ý tưởng: Vét cạn toàn bộ đường đi từ các biến thể của các hoán vị các điểm dừng và tìm ra đường đi ngắn nhất.

-Ưa điểm: Chắc chắn tìm ra được đường đi ngắn nhất.

-Nhược điểm:Do phải xét toàn bộ các trường hợp có thể xãy ra nên độ phức tạp là O(n2) nên sẽ tiêu tốn nhiều tài nguyên không gian và thời gian chạy.

Vd minh họa:



Đánh giá mức độ hoàn thành:

Mức 1 : 100%

Mức 2 : 100%

Mức 3 : 100%

Mức 4 : 0%

Mức 5 : 0%

Nguồn tài liệu tham khảo:

<https://medium.com/@nicholas.w.swift/easy-a-star-pathfinding-7e6689c7f7b2>

<http://code.activestate.com/recipes/578919-python-a-pathfinding-with-binary-heap/>

<https://www.youtube.com/watch?v=PzG-fnci8uE>