**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP.HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**NHÓM 6**

**TÊN ĐỀ TÀI: THUẬT TOÁN A\* ỨNG DỤNG TÌM ĐƯỜNG ĐI TRONG MÊ CUNG**

**TP. HỒ CHÍ MINH, THÁNG 11 NĂM 2022**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP.HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**TÊN ĐỀ TÀI: THUẬT TOÁN A\* ỨNG DỤNG TÌM ĐƯỜNG ĐI TRONG MÊ CUNG**

|  |  |
| --- | --- |
| **Nhóm: 6**  **Thành viên:**  **1. Lê Thành Thái Dương - 2001200690**  **2. Phạm Lê Xuân Phong – 2001200218**  **3. Lê Tất Đảm – 2001207022** | **Giảng viên hướng dẫn: Phùng Thế Bảo** |

**TP. HỒ CHÍ MINH, THÁNG 11 NĂM 2022**

1. LỜI CẢM ƠN

“Để hoàn thành tiểu luận này, em xin gửi lời cảm ơn chân thành đến:

Ban giám hiệu trường Đại Học Công Nghiệp Thực Phẩm Thành phố Hồ Chí Minh vì đã tạo điều kiện về cơ sở vật chất với hệ thống thư viện hiện đại, đa dạng các loại sách, tài liệu thuận lợi cho việc tìm kiếm, nghiên cứu thông tin.

Xin cảm ơn giảng viên bộ môn – Thầy Phùng Thế Bảo đã giảng dạy tận tình, chi tiết để em có đủ kiến thức và vận dụng chúng vào bài tiểu luận này.

Do chưa có nhiều kinh nghiệm làm để tài cũng như những hạn chế về kiến thức, trong bài tiểu luận chắc chắn sẽ không tránh khỏi những thiếu sót. Rất mong nhận được sự nhận xét, ý kiến đóng góp, phê bình từ phía Thầy để bài tiểu luận được hoàn thiện hơn.

Lời cuối cùng, em xin kính chúc thầy nhiều sức khỏe, thành công và hạnh phúc.”

Kí tên

Nhóm 6

1. LỜI MỞ ĐẦU

**Lời mở đầu**

Trong cuộc sống có nhiều vấn đề buộc ta phải lựa chọn hoặc tìm ra những phương án để giải quyết. Trong toán học cũng vậy để giải một bài toán đòi hỏi ta phải chọn được phương pháp giải được bài toán một cách tối ưu để thu được kết quả mong muốn. Trong lập trình phải tìm ra được giải thuật đúng để làm nền tảng xây dựng chương trình chạy đúng kết quả bài toán, hay đề tài của người yêu cầu đặt ra. Chẳng hạn như bài toán mê cung, đòi hỏi ta phải xây dựng thuật toán tìm được lối đi từ cửa vào đến được lối ra. Trong khi đó, có thể đứng trước nhiều ngã rẽ và phải tìm được lối đi cho đến khi thoát khỏi mê cung.

**Đặc tả đề tài**

**a. Yêu cầu**

- Chương trình phải đọc mê cung từ ma trận kề trên tập tin văn bản (rùa và trứng)

**b. Mục tiêu đề ra**

- Tìm được đường đi đến cửa ra

**c. Ngôn ngữ áp dụng**

- Ngôn ngữ Python

1. MỤC LỤC TỰ ĐỘNG

[LỜI CẢM ƠN 1](#_Toc120552843)

[LỜI MỞ ĐẦU 2](#_Toc120552844)

[MỤC LỤC TỰ ĐỘNG 3](#_Toc120552845)

[NỘI DUNG 5](#_Toc120552846)

[I. TÌM HIỂU SƠ LƯỢC THUẬT TOÁN TÌM KIẾM A\* 5](#_Toc120552847)

[1. Các khái niệm cơ bản thuật toán A\* 5](#_Toc120552848)

[2. Tại sao lại là thuật toán tìm kiếm A\*? 5](#_Toc120552849)

[3. Thuật toán: 5](#_Toc120552850)

[4. Ưu điểm của thuật toán A\* trong Python 6](#_Toc120552851)

[5. Nhược điểm của thuật toán A\* trong Python 6](#_Toc120552852)

[II. ÁP DỤNG THUẬT TOÁN A\* VÀO BÀI TOÁN TÌM ĐƯỜNG ĐI TRONG MÊ CUNG 6](#_Toc120552853)

[1. INPUT 6](#_Toc120552854)

[2. OUTPUT 7](#_Toc120552855)

[2.1) \*Chú thích: Các ký hiệu # cho biết chướng ngại vật, x là vị trí của rùa (điểm bắt đầu) và o là vị trí của quả trứng (điểm kết thúc) 7](#_Toc120552856)

[3. \* Cài đặt thuật toán bằng ngôn ngữ python: 7](#_Toc120552857)

[3.1) Các thư viện cần thiết: 7](#_Toc120552858)

[3.2) Các phương thức phải có: 7](#_Toc120552859)

[3.3) Mô tả cài đặt thuật toán 8](#_Toc120552860)

[3.4) \* Kết quả 14](#_Toc120552861)

[LỜI KẾT 16](#_Toc120552862)

[1. Hướng dẫn sử dụng source code: 16](#_Toc120552863)

[2. Tổng quát: 16](#_Toc120552864)

[TÀI LIỆU THAM KHẢO 17](#_Toc120552865)

2. NỘI DUNG
   1. TÌM HIỂU SƠ LƯỢC THUẬT TOÁN TÌM KIẾM A\*
      1. Các khái niệm cơ bản thuật toán A\*

Thuật toán tìm kiếm A\* là một trong những kỹ thuật tốt nhất và phổ biến được sử dụng trong tìm đường và duyệt đồ thị.

g(n): Đường dẫn chi phí thực tế từ nút bắt đầu đến nút hiện tại.

h(n): Đường dẫn chi phí thực tế từ nút hiện tại đến nút mục tiêu.

f(n): Đường dẫn chi phí thực tế từ nút bắt đầu đến nút mục tiêu.

Để thực hiện thuật toán A\*, chúng ta phải sử dụng hai mảng là Open và Close.

**Open:**

Một mảng chứa các nút đã được tạo nhưng chưa được kiểm tra cho đến nay.

**Close:**

Một mảng chứa các nút được kiểm tra.

* + 1. Tại sao lại là thuật toán tìm kiếm A\*?

Nói một cách không chính thức, thuật toán A\* Search, không giống như các kỹ thuật duyệt khác, nó có “bộ não”. Điều đó có nghĩa là nó thực sự là một thuật toán thông minh tách nó ra khỏi các thuật toán thông thường khác. Thực tế này được giải thích chi tiết trong các phần dưới đây.

Và cũng cần nhắc lại rằng nhiều trò chơi và bản đồ trên web sử dụng thuật toán này để tìm đường đi ngắn nhất rất hiệu quả (xấp xỉ).

* + 1. Thuật toán:

**1:**  Đầu tiên, Đặt nút bắt đầu vào OPEN và tìm giá trị f(n) của nó.

**2:**  Sau đó loại bỏ nút khỏi OPEN, có giá trị f(n) nhỏ nhất. Nếu đó là một nút mục tiêu, sau đó dừng lại và quay lại thành công.

**3:**  Ngược lại xóa nút khỏi Open và tìm tất cả các nút kế tiếp.

**4:**  Tìm giá trị f(n) của tất cả các nút kế tiếp, đặt chúng vào Open và đặt nút đã loại bỏ vào Close.

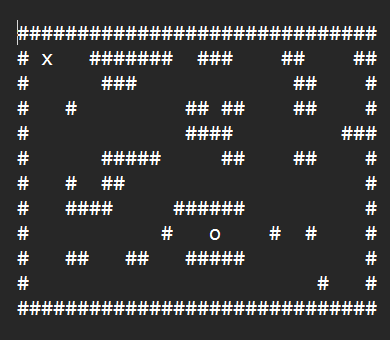
**5:**  Đi đến Bước 2.

**6:**  Thoát.

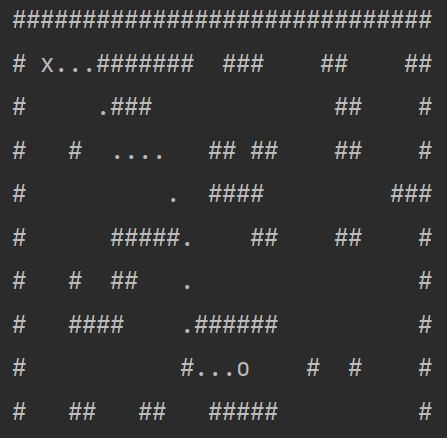
* + 1. Ưu điểm của thuật toán A\* trong Python
* Nó hoàn toàn đầy đủ và tối ưu.
* Đây là một trong những kỹ thuật tốt nhất trong tất cả các kỹ thuật khác. Chúng tôi sử dụng để giải quyết tất cả các vấn đề phức tạp thông qua thuật toán này.
* Thuật toán có hiệu quả tối ưu, nghĩa là không có thuật toán tối ưu nào khác được đảm bảo mở rộng ít nút hơn A\*.
  + 1. Nhược điểm của thuật toán A\* trong Python
* Thuật toán này hoàn thành nếu hệ số phân nhánh của thuật toán là hữu hạn và mọi hành động đều có chi phí cố định.
* Tốc độ thực hiện tìm kiếm A\* phụ thuộc nhiều vào độ chính xác của thuật toán heuristic được sử dụng để tính toán h(n) và chậm hơn một chút so với các thuật toán khác.
* Nó đang có những vấn đề phức tạp.
  1. ÁP DỤNG THUẬT TOÁN A\* VÀO BÀI TOÁN TÌM ĐƯỜNG ĐI TRONG MÊ CUNG

*Đề bài: Hãy giúp rùa tìm đường đi từ vị trí hiện tại của nó đến quả trứng, sử dụng thuật toán A\* được hổ trờ bởi tìm kiếm heuristic.*

* + 1. INPUT

******

* + 1. OUTPUT

******

* + - 1. \*Chú thích: Các ký hiệu # cho biết chướng ngại vật, x là vị trí của rùa (điểm bắt đầu) và o là vị trí của quả trứng (điểm kết thúc)
    1. \* Cài đặt thuật toán bằng ngôn ngữ python:
       1. Các thư viện cần thiết:

- Thư viện os: cho phép chúng ta làm việc với các tập tin và thư mục.

- Thư viện math: thực hiện các phép toán dễ dàng hơn bằng các hàm toán học có trong math.

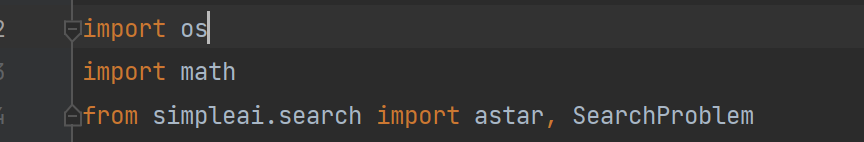
- Thư viện simpleai.search: triển khai nhiều thuật toán trí tuệ nhân tạo được mô tả trong cuốn sách “Trí tuệ nhân tạo, cách tiếp cận hiện đại”, của Stuart Russel và Peter Norvig.

Để giải quyết vấn đề tìm kiếm bằng simpleai, trước tiên bạn cần lập trình các chi tiết cụ thể của vấn đề. Để làm điều này, thư viện cung cấp cho bạn một lớp có tên SearchProblem mà bạn sẽ kế thừa và sau đó điền vào các chi tiết cụ thể của vấn đề đó.

Sau khi đã xác định vấn đề của mình, bạn có thể gọi bất kỳ thuật toán tìm kiếm nào có trong simpleai để tìm giải pháp cho vấn đề.

* + - 1. Các phương thức phải có:
* actions: phương thức này nhận một trạng thái và phải trả về danh sách các hành động có thể được thực hiện từ trạng thái cụ thể đó.
* result: phương thức này nhận một trạng thái và một hành động và phải trả về trạng thái kết quả của việc áp dụng hành động cụ thể đó từ trạng thái cụ thể đó.
* is\_goal: phương thức này nhận trạng thái và phải trả về True nếu trạng thái là trạng thái mục tiêu hoặc Sai nếu không.
  + - 1. Mô tả cài đặt thuật toán

- Import các thư viện

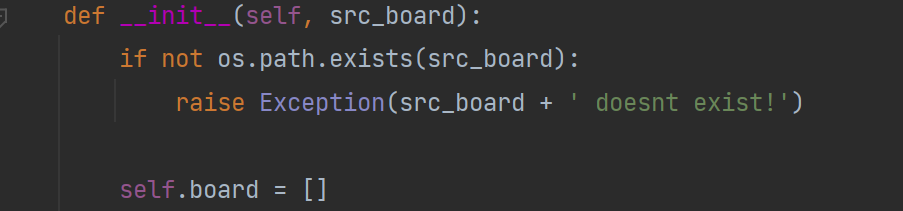


- Tạo class và kế thừa class SearchProblem

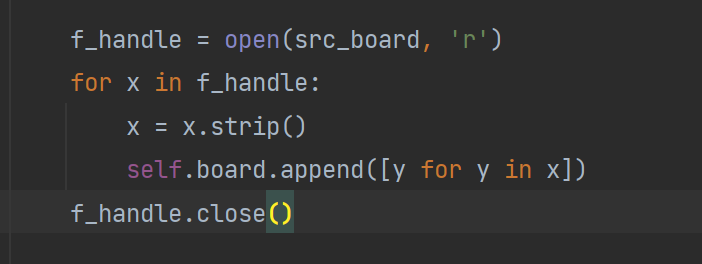


- Viết phương thức khởi tạo, tham số truyền vào là source file input

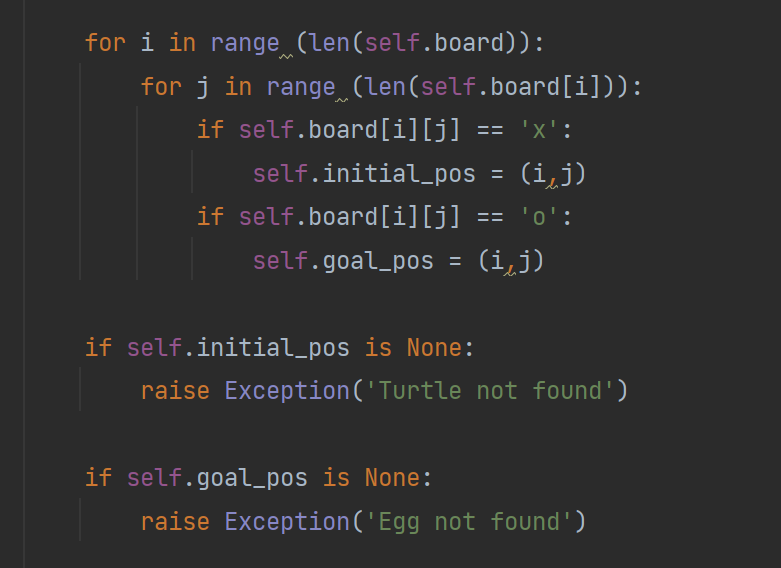
* Kiểm tra file có tồn tại hay không? Nếu không, thông báo lỗi 'file doesnt exist!',thoát chương trình
* Nếu có, tạo một mảng rỗng



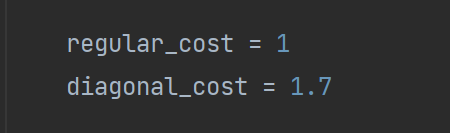
* Mở file để đọc
* Dùng foreach để thêm các phần tử trong file vào mảng vừa khởi tạo
* Đóng file



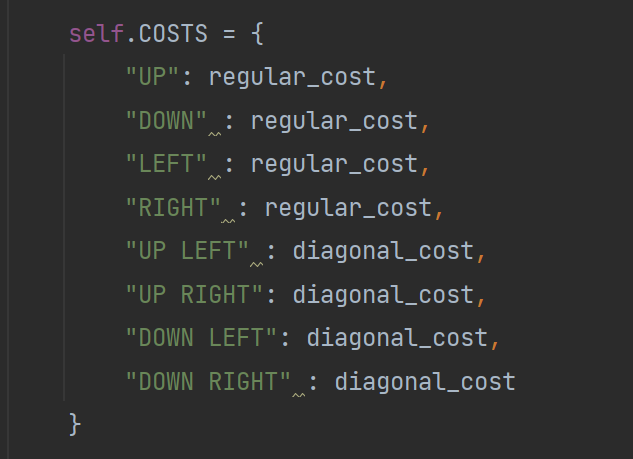
* Khởi tạo điểm khởi đầu và kết thúc bằng none
* Dùng vòng lập for(i,j) để tìm vị trí của x và o trong mảng sau đó gán tọa độ đó cho điểm khởi đầu (rùa) và điểm kết thúc (trứng)
* Nếu không tìm ra x và o thì báo lỗi và thoát chương trình



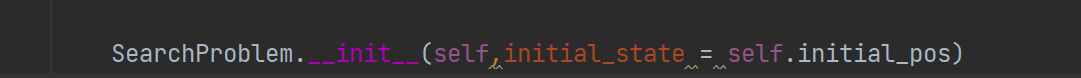
* Nếu tìm thấy x và o, khởi tạo biến chi phí regular\_cost = 1(trên || dưới || trái || phải), diagonal\_cost = 1.7 (các đường chéo)



* Tạo dictionary lưu các giá trị của các khóa 'up, down,...'

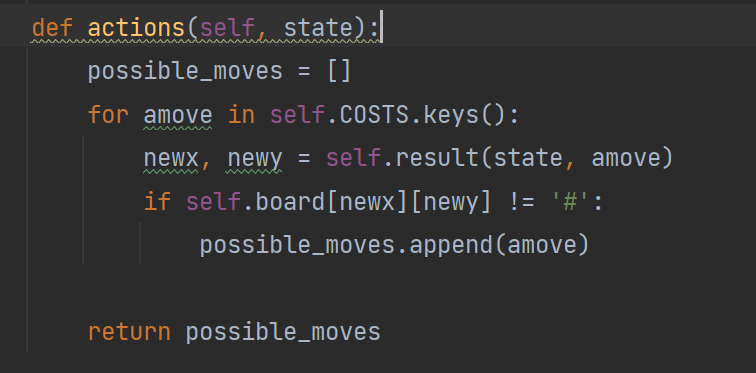


* Sau khi đã xác định được vấn đề của bài toán, truyền tham số khởi tạo initial\_state = self.initial\_pos vào lớp SearchProblem



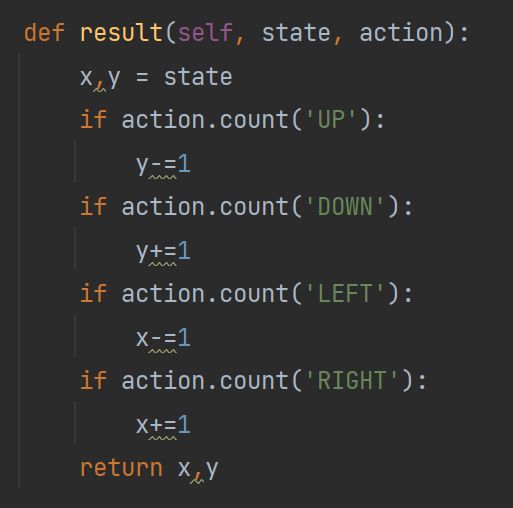
- Viết phương thức actions (phương thức này nhận một trạng thái và phải trả về một danh sách các hành động có thể được thực hiện)

* Tạo mảng chứa các hướng có thể di chuyển
* Dùng vòng for biến amove đi qua từng phần từ trong dictionary costs.keys()
* Gán newx, newy = selft.result (sẽ tạo sau)
* Nếu board[newx][newy] != '#'
* Thêm amove vào cuối mảng các hướng có thể di chuyển, return mảng

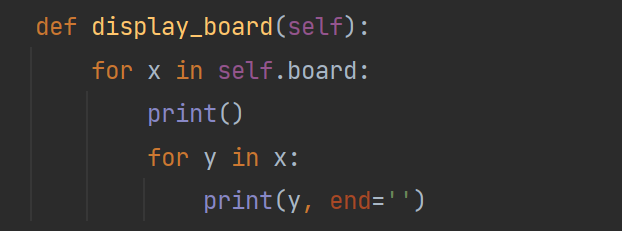


- Viết phương thức results (phương thức này nhận trạng thái hiện tại (tọa độ) và một hành động (di chuyển), nó phải trả về một trạng thái mới (tọa độ) là kết quả của việc áp dụng phương thức actions (di chuyển))

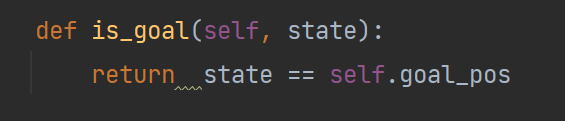
* Gán x, y = state (tọa độ hiện tại)
* Nếu action.count = 'up' thì tọa độ y=y-1
* Nếu action.count = 'down' thì tọa độ y=y+1
* Nếu action.count = 'left' thì tọa độ x=x-1
* Nếu action.count = 'right' thì tọa độ x=x+1, các trường hợp còn lại tọa độ không đổi
* Trả về tọa độ mới



- Viết phương thức hiển thị

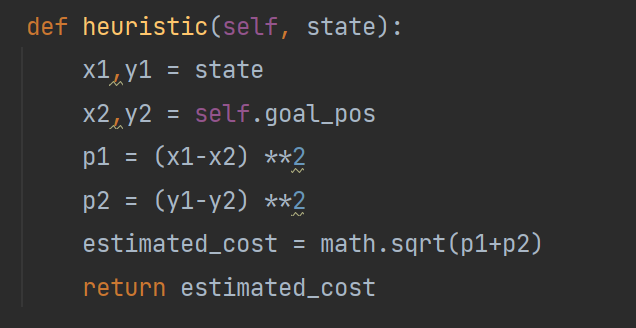


- Viết phương thức kiểm tra vị trí kết thúc

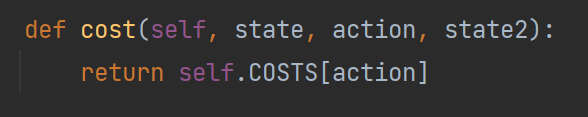


- Viết phương thức tính heuristic

* Gán x1, y1 bằng vị trí hiện tại
* Gán x2, y2 bằng vị trí kết thúc
* Tính p1, p2, sau đó ta tính được giá trị h = sqrt của p1 và p2
* Return h

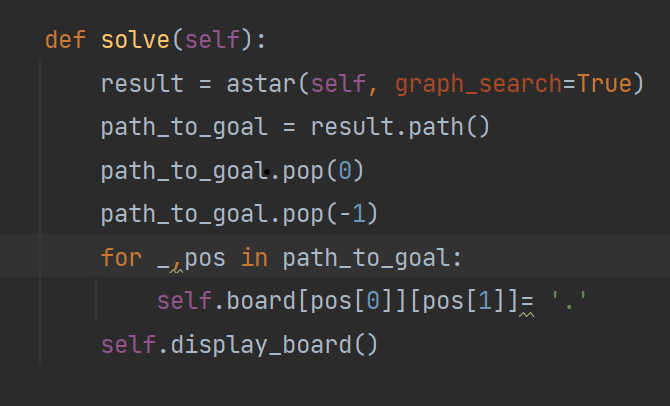


- Viết phương thức tính chi phí (phương thức này trả về chi phí khi di chuyển)

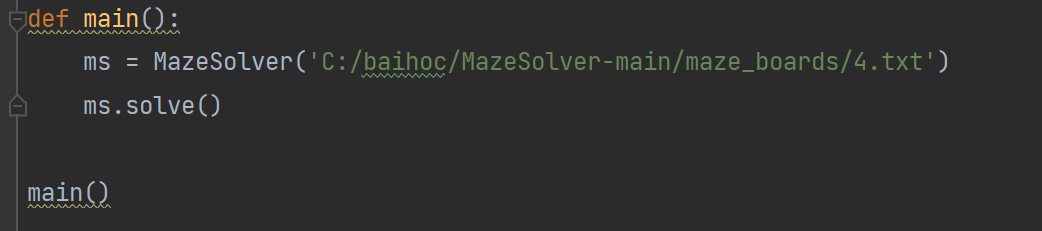


- Viết phương giải quyết bài toán

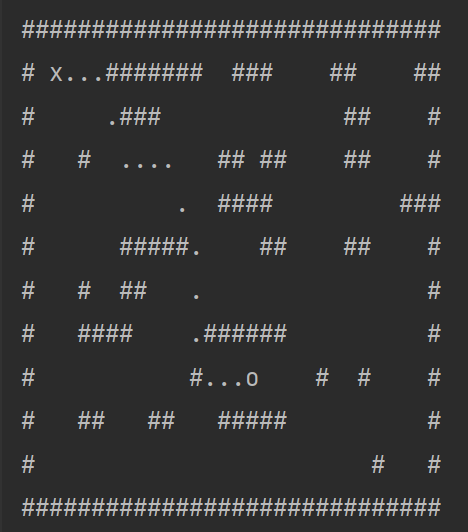
* Tạo biến result gán cho kết quả trả về của hàm astar với tham số truyền vào là grap\_search = true (tránh khám phá các trạng thái lặp đi lặp lại)
* Gán path\_to\_goal = result.path() (path() là hàm chứa đường đi từ trạng thái ban đầu đến trạng thái đích)
* Xóa phần từ đầu tiên trong path\_to\_goal (vị trí của rùa)
* Xóa phần từ cuối cùng trong path\_to\_goal (vị trí của quả trứng)
* Dùng vòng lập trong path\_to\_goal để vẽ đường đi và được biểu diễn bằng dấu chấm
* Gọi hàm hiển thị



- Viết và gọi hàm main()

**

* + - 1. \* Kết quả

**

1. LỜI KẾT
   * 1. Hướng dẫn sử dụng source code:

Cài đặt chương trình có hỗ trợ python (một số chương trình gợi ý như sau: Visual studio code, Pycharm,...)

Bước 1: Cài đặt phiên bản python mới nhất.

Bước 2: Cài đặt gói pip.

Bước 3: Cài đặt gói simpleai.

Bước 4: Chạy file.

* + 1. Tổng quát:

Vì thuật toán tìm đường đi trong mê cung của rùa đến trứng khá là rộng nên ta cần các công cụ, thư viện cũng như là các phương thức đặc trưng riêng để hỗ trợ cho việc thực hiện. Thời gian tìm kiếm cũng mất một khoảng thời gian dài và nhiều vấn khó khăn. Mong thầy và các bạn có thể châm chước bỏ qua một số lỗi hoặc sai sót của nhóm nếu có.s

Cảm ơn thầy và các bạn !!!

1. TÀI LIỆU THAM KHẢO

[1] <https://github.com/simpleai-team/simpleai> , fisadev Merge branch 'develop'.

[2] <https://www.geeksforgeeks.org/a-search-algorithm/?ref=lbp>, GeeksforGeeks.

[3] <https://pip.pypa.io/en/stable/installation/> , The pip developers made [Sphinx](https://www.sphinx-doc.org/).

[4] <https://github.com/duongl/MazeSolver-A-/blob/main/README.md> , Github.