TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TPHCM

KHOA CÔNG NGHỆ THÔNG TIN

**BÁO CÁO MÔN HỌC**

**CƠ SỞ TRÍ TUỆ NHÂN TẠO**

Đồ án 1

**TÌM KIẾM HEURISTIC VỚI A\***

Sinh viên thực hiện: Vũ Lê Thế Anh 1612838

Nguyễn Lê Hồng Hạnh 1612849

|  |  |
| --- | --- |
| **Công việc** | **Mức độ hoàn thành** |
| Yêu cầu 1 | **100%** |
| Yêu cầu 2 | **100%** |

# **Thuật toán A\***

A\* là thuật toán nhằm giải quyết bài toán tìm đường đi từ 1 nút khởi đầu đến 1 nút đích, thuộc lớp các thuật toán tìm kiếm với thông tin cho trước (informed search).

## **Bài toán:**

Cho một lưới ô vuông , nhiệm vụ của chúng ta là tìm một tập hợp có thứ tự các ô vuông có tọa độ nguyên ( là chỉ số dòng, là chỉ số cột) liên tiếp kề nhau, với ô đầu là tọa độ nút khởi đầu và ô cuối là tọa độ nút đích .

Hai ô vuông được gọi là kề nhau khi có chung cạnh hoặc chung đỉnh với nhau:

## **Ý tưởng thuật toán:**

Trong quá trình tìm kiếm đường đi, khác với tìm kiếm mù, A\* sử dụng thêm một hàm đánh giá heuristic để tính khoảng cách từ một điểm bất kì đến điểm đích. Hàm heuristic giúp định lượng mức độ gần đến đích của một ô, giúp thuật toán có thể đưa ra lựa chọn giữa các ô kề với vị trí hiện tại.

## **Mô tả thuật toán:**

Thuật toán A\* duy trì hai tập là tập mở (OPEN) và tập đóng (CLOSE). Tập mở có vai trò lưu trữ các ô có thể bước đến. Tập đóng (CLOSE) lưu trữ các ô đã bước đến. Do mỗi ô chỉ được bước đến 1 lần, các ô đã ở trong tập đóng sẽ không được xét lại nữa.

Ở mỗi bước, thuật toán A\* sẽ chọn ra một ô từ tập mở để bước tới. Việc lựa chọn ô được quyết định bởi hàm , với là chi phí của đường đi cho đến thời điểm hiện tại,  là hàm đánh giá heuristic về chi phí nhỏ nhất để đến đích. có giá trị càng thấp thì độ ưu tiên càng cao. Việc bước tới tương ứng với lấy ô đó ra khỏi tập mở và chuyển vào tập đóng. Sau đó, thuật toán đưa tất cả những ô kề với ô đó mà không có trong tập đóng vào trong tập mở.

Quá trình này lặp lại cho đến khi ô đích được lấy ra khỏi tập mở hoặc không còn ô nào ở trong tập mở. Nếu ô đích được lấy ra khỏi tập mở, một đường đi đã được tìm thấy. Ngược lại, nếu tập mở rỗng trước khi ô đích được lấy ra thì không tồn tại đường đi từ ô xuất phát đến ô đích.

Lưu ý:

* Khi cài đặt thuật toán, tập mở sẽ được cài đặt sử dụng cấu trúc dữ liệu vun đống (heap), hay còn gọi là hàng đợi ưu tiên (priority queue) để tối ưu quá trình tìm kiếm phần tử nhỏ nhất. Hàng đợi này luôn đảm bảo phần tử đầu tiên trong hàm đợi là phần tử có giá trị nhỏ nhất (ở đây là giá trị ).
* Để truy ngược đường đi tìm được, cần thêm một tập lưu thông tin ô cha (parent) của các ô, là ô liền trước trên đường đi từ ô xuất phát đến ô đích (nếu có thuộc đường đi).
* Do kích thước lưới ô vuông có thể là rất lớn, thay vì lưu giá trị của tất cả các ô từ đầu, ta sử dụng một từ điển và cập nhật hoặc tính lại chỉ khi ô đó được xét đến.

# **Lựa chọn Heuristic**

Theo tài liệu, một heuristic được gọi là chấp nhận được khi nó không bao giờ tính quá chi phí thật sự để đến được trạng thái đích từ trạng thái hiện tại. Nói cách khác, nếu ta có một heuristic hoàn hảo (tức luôn bằng chính xác chi phí tối ưu đi từ trạng thái hiện tại đến trạng thái đích) thì .

Một số heuristic đáng xem xét là các khoảng cách trong các không gian (với là trạng thái/ô hiện tại và là trạng thái/ô đích):

* Khoảng cách Euclid :
* Khoảng cách Manhattan (:
* Khoảng cách Chebyshev (:

Yêu cầu đồ án sử dụng khoảng cách Euclid, tuy nhiên, đây là một heuristic không chấp nhận được. Xét trường hợp đơn giản một lưới , trạng thái hiện tại là , đích là . Lúc này, theo :

Có do chúng ta được phép di chuyển theo đường chéo.

Đối với khoảng cách Manhattan, ta cũng có nhận xét tương tự. Xét trường hợp tương tự:

Heuristic hợp lý có thể sử dụng đó là hay khoảng cách Chebyshev. Có thể chứng minh như sau:

Giả sử bỏ qua các vật cản, chi phí tối ưu tại mọi ô là . Do việc thêm vật cản làm cho chi phí có thể tăng lên, ta có .

Cách di chuyển tối ưu để đến đích là đi đường chéo cho đến khi có cùng tung độ hoặc hoành độ với đích rồi đi đường dọc/ngang đến đích. Với chiến thuật tối ưu này, mỗi bước ta chỉ giảm (hoặc giảm ) là chênh lệch tung độ (hoành độ) so với đích, và do đó cần đi bước. Vậy .

Như vậy, chứng tỏ đây là heuristic chấp nhận được.

Một số ví dụ minh họa thuật toán A\* với heuristic khác nhau:



*Hình 1. Thuật toán A\* với heuristic khoảng cách Chebyshev (bên trái) chỉ cần 12 bước trong khi heuristic khoảng cách Euclid (bên phải) cần 13 bước*

# **Thuật toán ARA\***

Thuật toán Anytime Replanning (Repairing) A\* (hay ARA\*) là một biến thể cải tiến của thuật toán tìm kiếm A\*.

Cách thức hoạt động của ARA\* trước hết dựa trên việc đánh trọng số cho heuristic khi tính giá trị :

Giả sử ta có hai heuristic và . Nếu thì thuật toán A\* khi chạy với sẽ nhanh hơn khi chạy với (do giảm bớt khả năng có hai trạng thái có cùng giá trị heuristic). Như vậy, có thể xem việc đánh trọng số như là chạy thuật toán A\* với heuristic để tăng tốc độ tìm kiếm.

Tuy nhiên, việc nhân trọng số này sẽ làm cho heuristic có khả năng trở nên không chấp nhận được , dẫn đến kết quả tìm kiếm không phải là tối ưu. Để hạn chế vấn đề này, thay vì cố định giá trị , ta giảm dần nó từ một giá trị khởi đầu xuống đến 1. Việc giảm dần này diễn ra trong một khoảng thời gian giới hạn. Như vậy, thuật toán hướng tới một điểm cân bằng giữa độ chính xác và thời gian tìm kiếm.

Một vấn đề đáng chú ý là khi giảm giá trị , nếu thực hiện lại việc tìm kiếm từ đầu thì sẽ lãng phí những kết quả tìm kiếm trước đó. Thuật toán ARA\* giải quyết vấn đề này bằng cách thêm vào thuật toán A\* truyền thống một tập gọi là tập Inconsistent (không ổn định). Tập này lưu lại những trạng thái có giá trị giảm trong quá trình tìm kiếm. Những trạng thái này sẽ được xét đến trong những lần chạy tiếp theo (với giảm).

Khi thực hiện ARA\*, một điểm khác biệt nữa là điều kiện dừng. Đối với A\*, thuật toán tìm kiếm dừng khi trạng thái đích được lấy ra khỏi tập mở. Tuy nhiên, do ARA\* sử dụng 1 tập mở xuyên suốt (cập nhật lại mỗi khi giảm ) nên điều kiện này không còn áp dụng được nữa. Tuy nhiên, một điều kiện tương tự có thể áp dụng là dừng khi giá trị của trạng thái đích nhỏ hơn giá trị của tất cả phần tử trong tập mở.

Ngoài ra, có thể xem A\* là một trường hợp đặc biệt của thuật toán ARA\* với giá trị và không giới hạn thời gian chạy.

# **Cấu trúc dữ liệu**

Chương trình bao gồm các lớp (class) chính, được tổ chức như sau:

* PriorityQueue: đóng gói các phương thức của thư viện heapq do Python cung cấp, là cấu trúc dữ liệu heap (hay còn gọi là hàng đợi ưu tiên, priority queue). Mặc định đây là heap cực tiểu (tức phần tử đầu hàng đợi/có độ ưu tiên cao nhất mang giá trị nhỏ nhất.
  + Thuộc tính:
    - : đối tượng mảng lưu trữ các phần tử của hàng đợi ưu tiên. Trong đồ án này, hàng đợi chủ yếu lưu cặp với là giá trị của trạng thái (gồm tọa độ của ô đang đứng). là thành phần quyết định độ ưu tiên của phần tử.
  + Phương thức
    - : cho biết hàng đợi có đang rỗng
    - : đẩy phần tử vào hàng đợi
    - : lấy phần tử ở đầu hàng đợi ra
    - : xem thông tin của phần tử ở đầu hàng đợi
    - : in hàng đợi ra console (dùng để kiểm lỗi)
* SearchAlgorithm: quản lý các thuật toán tìm kiếm khác nhau, gồm các lớp kế thừa là AStar và ARAStar
* Path: quản lý các đường đi tìm được
  + Thuộc tính:
    - : tọa độ điểm bắt đầu
    - : tọa độ điểm đích
    - : đối tượng mảng lưu trữ các bước trên đường đi
  + Phương thức:
    - : cho biết độ dài đường đi
    - : cho biết đường đi rỗng (không tìm được đường đi)
    - : xuất đường đi với luồng đầu ra
    - : kiểm tra xem tọa độ có là một bước trên đường đi
    - : thể hiện đường đi trên giao diện người dùng, sử dụng khung vẽ và màu (mặc định xanh dương)
    - : in đường đi ra console
* View: quản lý việc mô phỏng quá trình giải quyết bài toán bằng giao diện người dùng
  + Thuộc tính
    - đối tượng khung vẽ, nơi để vẽ bản đồ và biểu diễn thuật toán A\*/ARA\*.
* : đối tượng bản đồ, nơi giữ các thông tin của trò chơi
* số nguyên dương cho biết kích thước từng ô vuông nhỏ
* : cỡ chữ của các số trong ô vuông
* từ điển các màu sắc của từng loại ô khác nhau (ô Start màu đỏ, ô Goal màu vàng, ô trống màu xanh lá, ô vật cản màu đen,…)
* : lưu các đối tượng ô vuông của tkinter (để dễ dàng chỉnh sửa về sau mà không cần tạo mới)
* : lưu các đối tượng chuỗi của tkinter (tương tự tiles)
* Phương thức
* : liên kết với đối tượng bản đồ để lấy thông tin và tương tác với người dùng
  + khởi tạo dựa trên thông tin bản đồ
  + vẽ từng ô nhỏ tại tọa độ (x,y) và kiểu . Từ đó có thể viết , và các trường hợp vẽ đặc biệt thông dụng.
  + : tạo ô vuông ứng với tọa độ trên bản đồ
  + *:* cập nhật ô tại tọa độ (x, y) với thông tin từ bản đồ
  + : tạo đối tượng chuỗi trong lòng ô vuông ứng với tọa độ
  + : cập nhật giá trị chuỗi trong lòng ô vuông ứng với tọa độ thành
* Board: quản lý bản đồ và các thông số của trò chơi
  + Thuộc tính:
    - : lưu trữ kích thước bản đồ
    - lưu trữ tọa độ điểm đầu và điểm đích của bản đồ
    - : đối tượng ma trận lưu giữ trạng thái trống/vật cản của các ô
    - : liên kết với đối tượng mô phỏng trên giao diện người dùng
  + Phương thức:
    - : thiết lập kích thước, điểm đầu, điểm cuối của bản đồ
    - : chuyển đổi giữa hai trạng thái trống/vật cản của ô tọa độ
    - : tạo bản đồ ngẫu nhiên
    - : đọc tập tin có đường dẫn để khởi tạo bản đồ
    - : lưu thông tin bản đồ hiện tại
    - : tìm kiếm đường đi (không biểu diễn trên GUI) với thuật toán (ví dụ: “AStar”, “ARAStar”), sử dụng và giá trị
    - : lưu kết quả vào tập tin với định dạng theo yêu cầu đồ án
    - : kiểm tra tọa độ có thuộc bản đồ hay không
    - : tìm 8 tọa độ kề với tọa độ
    - : mô phỏng quá trình tìm kiếm theo thuật toán với bộ tham số cho trước
* GUI: là lớp chính để chạy chương trình và chứa các thuộc tính để tương tác với người dùng. Lớp này sẽ gọi các hàm từ Board và View để thực hiện việc chạy và mô phỏng thuật toán.

# **Mô tả phần mềm**

## **Phiên bản Command Line:**

Chương trình có thể được chạy bằng Command Line với cú pháp:

*1612838\_1612849.exe <input\_dir> <output\_dir>*

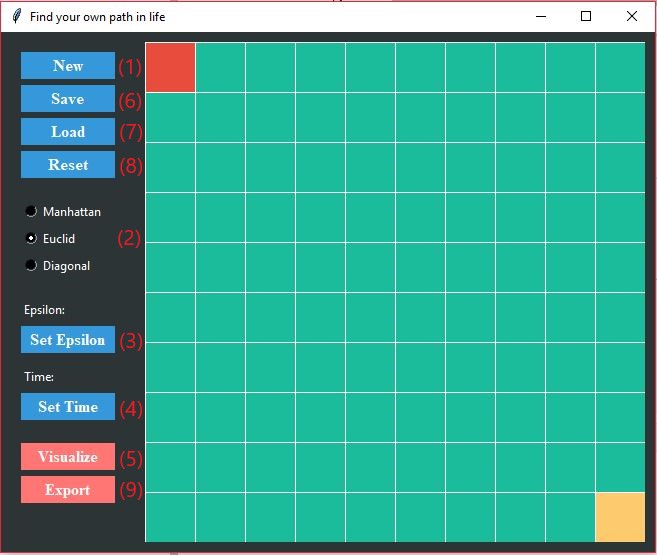
Trong đó, input\_dir là đường dẫn tới tập tin đầu vào, output\_dir là đường dẫn tới tập tin đầu ra.

Ví dụ: *1612838\_1612849.exe problem.txt solution.txt*

Nếu không có <output\_dir>, mặc định tên tập tin đầu ra là “solution.txt” (tự động tạo ở cùng thư mục file .exe).

Nếu không có cả <input\_dir> lẫn <output\_dir>, giao diện đồ họa người dùng sẽ hiện ra.

## **Phiên bản giao diện đồ họa người dùng:**



*Hình 2. Giao diện màn hình chính.*

(1) Nút *“New”* dùng để khởi tạo kích thước Board (, nơi biểu diễn thuật toán A\*.

(2) Các chọn lựa heuristic. Heuristic mặc định: Diagonal (khoảng cách Chebyshev).

(3) Nút *“Set Epsilon”* dùng để khởi tạo giá trị . Giá trị mặc định .

(4) Nút *“Set Time”* dùng để khởi tạo thời gian được phép để chạy thuật toán, hay nói cách khác, thuật toán chỉ được phép tìm kiếm trong khoảng thời gian đã được thiết lập đó. Nhập 0 nếu không muốn giới hạn thời gian. Giá trị mặc định là 0 ().

(5) Nút *“Visualize”* dùng để biểu diễn quá trình chạy thuật toán A\*, sau khi đã thiết lập đầy đủ các giá trị.

(6) Nút *“Save”* dùng để lưu Board thành file .txt (theo định dạng như yêu cầu đồ án).

(7) Nút *“Load”* dùng để vẽ Board từ file .txt (theo định dạng như yêu cầu đồ án).

(8) Nút *“Reset”* dùng để khởi tạo lại board.

(9) Nút *“Export”* dùng để lưu kết quả tìm kiếm đường đi từ điểm S đến điểm G.

(10) Ngoài ra, trong bảng trò chơi bên cạnh, có thể bấm vào bất kì ô vuông nào (trừ hai ô xuất phát và đích) để đổi trạng thái qua lại giữa trống và bị chặn.

# **Một số test mẫu:**

**Test 01:**

|  |  |
| --- | --- |
|  |  |
|  |  |

Với Epsilon khác nhau (từ Epsilon = 5):

Epsilon = 5.0

[55] (0, 0) (1, 0) (2, 1) (3, 0) (4, 0) (5, 1) (5, 2) (4, 3) (5, 4) (6, 5) (7, 6) (7, 7) (6, 8) (5, 7) (4, 8) (3, 7) (2, 8) (2, 9) (3, 10) (4, 10) (5, 10) (6, 11) (7, 11) (8, 11) (9, 11) (10, 11) (11, 12) (12, 13) (13, 13) (14, 12) (15, 13) (16, 13) (17, 13) (18, 12) (19, 13) (20, 13) (21, 13) (22, 13) (23, 13) (24, 14) (23, 15) (23, 16) (23, 17) (22, 18) (21, 19) (20, 20) (21, 21) (22, 22) (23, 23) (24, 24) (25, 25) (26, 26) (27, 27) (28, 28) (29, 29)

Epsilon = 1.5625

[49] (0, 0) (0, 1) (0, 2) (0, 3) (0, 4) (0, 5) (0, 6) (0, 7) (0, 8) (0, 9) (1, 10) (2, 9) (3, 10) (4, 10) (5, 10) (6, 11) (7, 11) (8, 11) (9, 11) (10, 11) (11, 12) (12, 13) (13, 13) (14, 12) (15, 13) (16, 13) (17, 13) (18, 12) (19, 13) (20, 13) (21, 13) (22, 13) (23, 13) (24, 14) (23, 15) (23, 16) (23, 17) (22, 18) (21, 19) (20, 20) (21, 21) (22, 22) (23, 23) (24, 24) (25, 25) (26, 26) (27, 27) (28, 28) (29, 29)

Epsilon = 1.53125

[41] (0, 0) (0, 1) (0, 2) (0, 3) (0, 4) (0, 5) (0, 6) (0, 7) (0, 8) (0, 9) (1, 10) (1, 11) (2, 12) (3, 13) (4, 14) (4, 15) (5, 16) (6, 16) (7, 15) (8, 16) (9, 16) (10, 16) (11, 17) (12, 16) (13, 16) (14, 17) (15, 16) (16, 17) (17, 18) (18, 19) (19, 20) (20, 20) (21, 21) (22, 22) (23, 23) (24, 24) (25, 25) (26, 26) (27, 27) (28, 28) (29, 29)

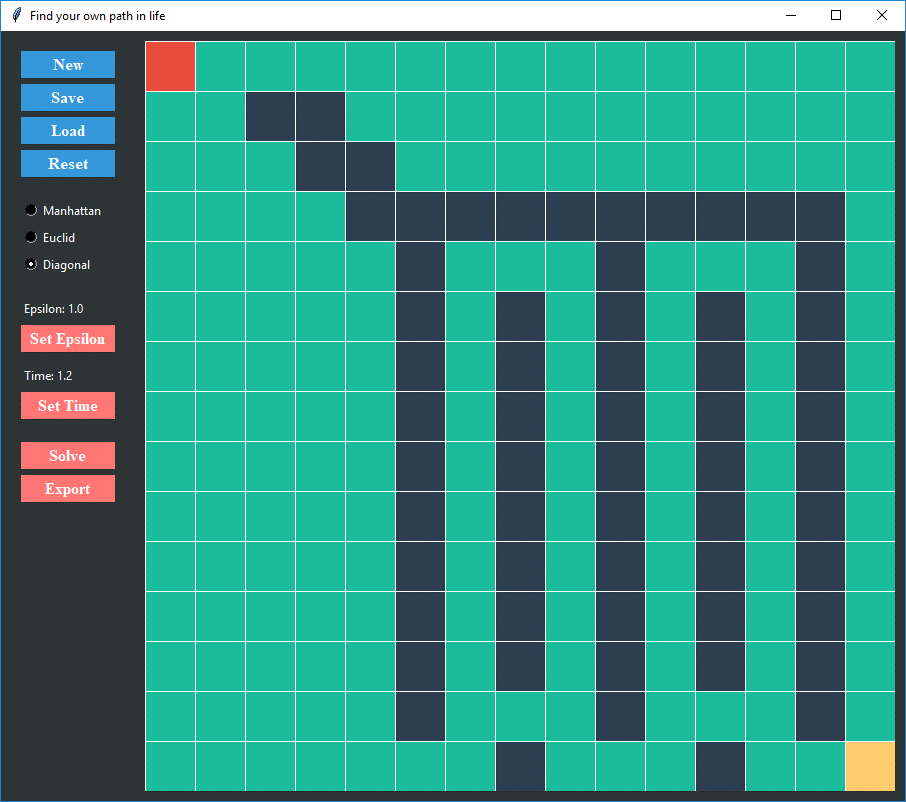
**Test 02:**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

**Test 03:**

|  |  |
| --- | --- |
|  | -1 |
|  |  |

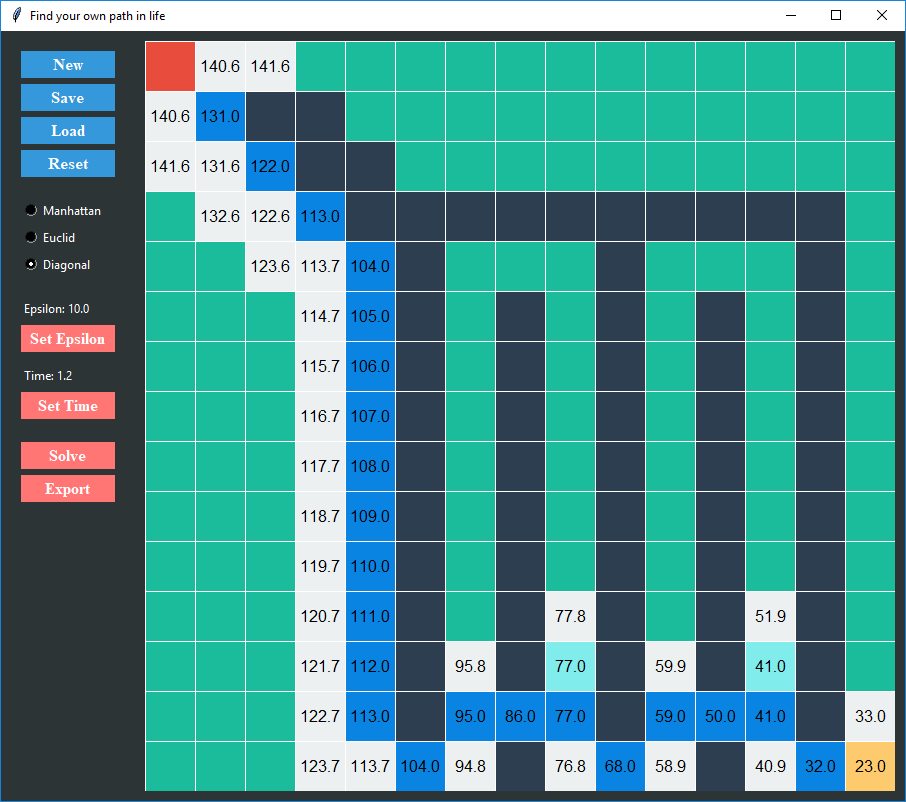
**Test 04:**



Epsilon = 1.0, Time = 1.2 (chạy A\* thuần túy với giới hạn thời gian)



Epsilon = 10.0, Time = 1.2 (chạy heuristic có trọng số)



(Lưu ý: thời gian chỉ mang tính tương đối – tức có trường hợp hai lần chạy với cùng thời gian với hai trọng số khác nhau cho kết quả khác nhau, không mang tính tuyệt đối, cần điều chỉnh với các máy có cấu hình khác nhau để có kết quả tương tự)

# **Phân công**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sinh viên** | **Công việc** | **Tỉ lệ hoàn thành so với công việc được giao** | **Phần trăm đóng góp** |
| Nguyễn Lê Hồng Hạnh | Yêu cầu 1  Front-end GUI A\* | **100%** | **40%** |
| Vũ Lê Thế Anh | Yêu cầu 2  Back-end GUI A\* | **100%** | **60%** |