**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

---------------- o 0 o ----------------



**BÁO CÁO TRÍ TUỆ NHÂN TẠO**

**ĐỀ TÀI: BÀI TOÁN TÌM ĐƯỜNG ĐI**

|  |  |
| --- | --- |
| Giáo viên hướng dẫn: | ThS. Ngô Văn Linh |
| Sinh viên thực hiện: | Nhóm 8 |
| 20160054 | Đinh Hoàng Anh |
| 20161935 | Hồ Xuân Hùng |
| 20161813 | Lê Quang Huy |
| 20164654 | Nguyễn Bá Việt |

**HÀ NỘI, THÁNG 12 NĂM 2018**

**MỤC LỤC**

[LỜI MỞ ĐẦU 2](#_Toc532113639)

[PHẦN 1: BÀI TOÁN TÌM LỘ TRÌNH ĐƯỜNG ĐI 3](#_Toc532113640)

[I. 1.Giới thiệu 3](#_Toc532113641)

[II. 2.Các thuật toán tìm đường đi 4](#_Toc532113642)

[PHẦN 2: THUẬT TOÁN 4](#_Toc532113643)

[I. Thuật toán UCS(Chi phí cực tiểu) 4](#_Toc532113644)

[1.Mô tả thuật toán 4](#_Toc532113645)

[2.Cài đặt thuật toán 4](#_Toc532113646)

[3.Các tính chất 5](#_Toc532113647)

[II. Thuật toán A\* 5](#_Toc532113648)

[1.Mô tả thuật toán 5](#_Toc532113649)

[2.Cài đặt thuật toán 6](#_Toc532113650)

[3.Các tính chất 6](#_Toc532113651)

[4.Hàm ước lượng heuristic 7](#_Toc532113652)

[a.các khái niệm 7](#_Toc532113653)

[b.các hàm heuristic 8](#_Toc532113654)

[III. Thuật toán Greedy best first 9](#_Toc532113655)

[1.Mô tả thuật toán 9](#_Toc532113656)

[2.Cài đặt thuật toán 10](#_Toc532113657)

[3.Các tính chất 10](#_Toc532113658)

[PHẦN 3: ĐÁNH GIÁ KẾT QUẢ 11](#_Toc532113659)

[I. 1.Giao diện người dùng 11](#_Toc532113660)

[II. 2.Lựa chọn hàm heuristic 12](#_Toc532113661)

[KẾT LUẬN 16](#_Toc532113662)

[TÀI LIỆU THAM KHẢO 16](#_Toc532113663)

# LỜI MỞ ĐẦU

Báo cáo Trí tuệ Nhân tạo, đề tài: Bài toán tìm lộ trình đường đi (A\* Search) là tài liệu mô tả tổng quan thuật toán A\* (đọc là A sao) và quy trình cài đặt phần mềm giúp giải quyết bài toán nêu trên. Tài liệu giúp người dùng hiểu được các chức năng của phần mềm cũng như tìm hiểu về định nghĩa, tính chất và các khái niệm liên quan của thuật toán A\*.

Để hoàn thành được bài tập lớn này, nhóm chúng em xin gửi lời cảm ơn tới thầy giáo hướng dẫn đề tài- ThS. Ngô Văn Linh, giảng viên Viện Công nghệ Thông tin và truyền thông, Trường Đại học Bách Khoa Hà Nội - đã hết lòng giúp đỡ, hướng dẫn, chỉ dạy tận tình để nhóm em hoàn thành được đề tài này. Đề tài này là sự nỗ lực của nhóm, tuy nhiên trong quá trình thực hiện không thể tránh khỏi sai sót. Rất mong được thầy cho ý kiến đóng góp để chúng em có thể hoàn thiện bài tập lớn này.

# PHẦN 1: BÀI TOÁN TÌM LỘ TRÌNH ĐƯỜNG ĐI

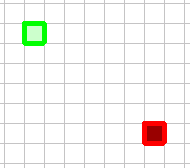
## 1.Giới thiệu

Bài toán tìm lộ trình đường đi là một bài toán khá phổ biến và có nhiều ứng dụng trong:

* tìm lộ trình giao thông trên bản đồ
* tìm đường đi cho robot trên các dạng địa hình
* tìm đường đi cho các đối tượng trong các ứng dụng mô phỏng thực tế, các nhân vật trong trò chơi

Việc giải quyết bài toán sẽ giúp tìm ra một lộ trình hợp lí từ điểm xuất phát của đối tượng đến điểm đích tùy chọn, giúp né tránh các chướng ngại vật, giảm thiểu chi phí.

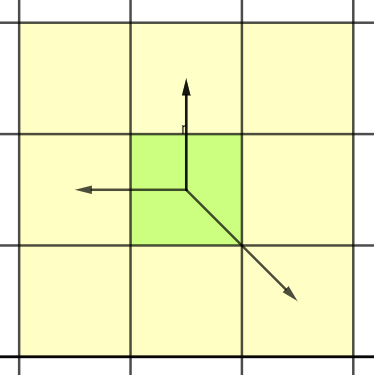
Phần mềm Pathfinder là một ứng dụng được viết bằng ngôn ngữ Java giúp giải quyết bài toán trên.



bản đồ dạng lưới (grid)

Ứng dụng mô phỏng một bản đồ dạng lưới hai chiều (2d-grid). Các ô lưới đại diện cho các đối tượng trên bản đồ, gồm 2 dạng là đặc (không cho phép đi qua) và rỗng (đi qua được). Trên ma trận này sẽ có 1 điểm đầu và 1 điểm cuối. Yêu cầu của bài toán là phải tìm đường đi từ điểm đầu đến điểm cuối sao cho chi phí di chuyển là tối ưu nhất.

Đối tượng có thể di chuyển giữa các ô lưới theo 8 hướng:



di chuyển 8 hướng

## 2.Các thuật toán tìm đường đi

Có rất nhiều thuật toán giúp giải quyết bài toán tìm đường đi, trong đó có:

* Dijkstra’s Algorithm (còn gọi là Uniform Cost Search)
* Greedy Best First Search (tìm kiếm tham lam)
* A\* (đọc là A sao)

# PHẦN 2: THUẬT TOÁN

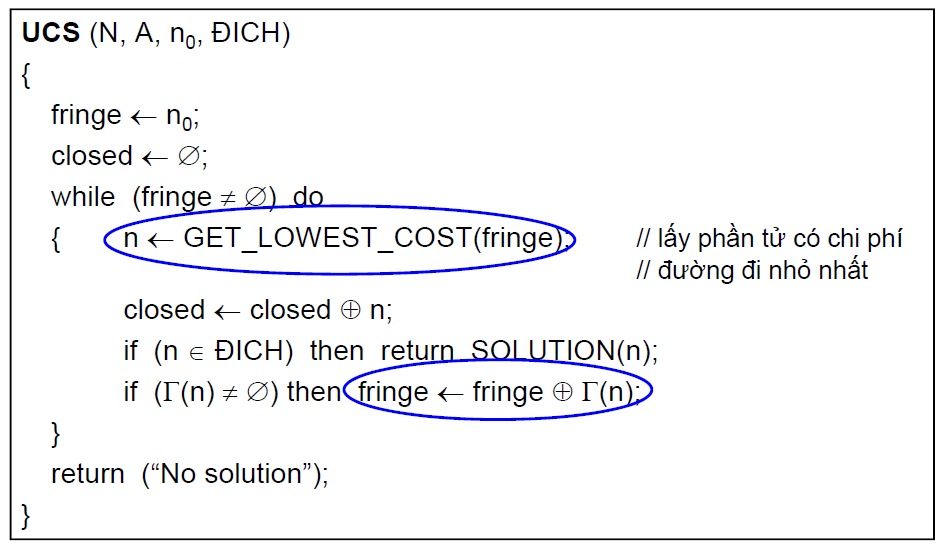
## Thuật toán UCS(Chi phí cực tiểu)

### 1.Mô tả thuật toán

Trong các ngành khoa học máy tính hay trí tuệ nhân tao, thuật toán tìm kiếm với chi phí cực tiểu (UCS) là một cách duyệt cây dùng cho việc duyệt hay tìm kiếm trên cây có trọng lượng chi phí. Thuật toán sẽ phát triển các nút chưa xét có chi phí thấp nhất – các nút được xét theo thứ tự chi phí tăng dần.Khi đồ thi có chi phí ở mỗi bước là như nhau thì thuật toán trở thành phương pháp tìm kiếm theo chiều rông.

### 2.Cài đặt thuật toán

* G(N,A): Cây biểu diễn không gian trạng thái bài toán
* n0 : Trạng thái ban đầu của bài toán
* ĐICH: tập hợp các trạng thái đích
* Closed: tập hợp các điểm đã duyệt
* Fringe: là một cấu trúc hàng đợi sắp xếp theo chi phí đường đi. Thứ tự ưu tiên được xác định theo hàm **f(n) = g(n)**. Trong đó:
  + **g(n)** là chi phí từ nút gốc n0 tới nút hiện tại n
  + **f(n)** chi phí tổng thể ước lượng của đường đi qua nút hiện tại n đến đích



### 3.Các tính chất

* Tính hoàn chỉnh?
  + Có (nếu chi phí ở mỗi bước ≥ ε)
* Độ phức tạp về thời gian?
  + Phụ thuộc vào tổng số các nút có chi phí ≤ chi phí của lời giải tối

ưu: *O*(*b*⎡*C\**/ *ε*⎤), trong đó *C*\* là chi phí của lời giải tối ưu

* Độ phức tạp về bộ nhớ?
  + Phụ thuộc vào tổng số các nút có chi phí ≤ chi phí của lời giải tối

ưu: *O*(*b*⎡*C\**/ *ε*⎤)

* Tính tối ưu?
  + Có (nếu các nút được xét theo thứ tự tăng dần về chi phí *g(n)*)

## Thuật toán A\*

### 1.Mô tả thuật toán

Trong khoa học máy tính, A\* là một thuật toán tìm kiếm trong đồ thị. Thuật toán này tìm một đường đi đơn từ nút bắt đầu tới một nút mục tiêu cho trước (hoặc tới một nút thỏa mãn điều kiện đích), sử dụng một đánh giá heuristic để xếp loại từng nút theo ước lượng về tuyến đường tốt nhất đi qua nút đó và duyệt các nút theo thứ tự của đánh giá heuristic đó. Do vậy, thuật toán A\* là một ví dụ của tìm kiếm theo lựa chọn tốt nhất (best-first search).

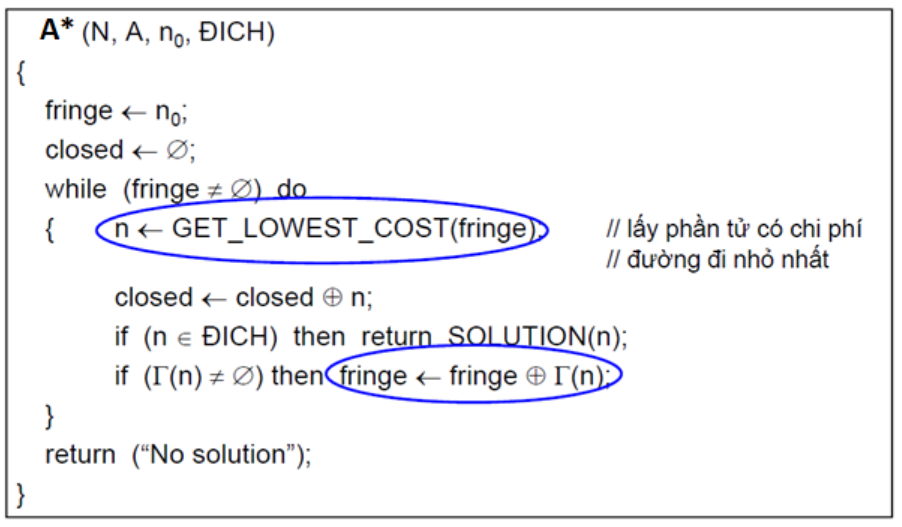
Thuật toán A\* được mô tả lần đầu vào năm 1968 bởi Peter Hart, Nils Nilsson, và Bertram Raphael. Trong bài báo của họ, thuật toán được gọi là thuật toán A; khi sử dụng thuật toán này với một đánh giá heuristic thích hợp sẽ thu được hoạt động tối ưu, do đó mà có tên A\*.

Trong bài toán tìm đường, A\* xây dựng tăng dần tất cả các tuyến đường từ điểm xuất phát cho tới khi nó tìm thấy một đường đi chạm tới đích. Để biết những tuyến đường nào có khả năng sẽ dẫn tới đích, A\* sử dụng một "đánh giá heuristic" về khoảng cách từ điểm bất kỳ cho trước tới đích.

Điểm khác biệt của A\* so với Greedy BFS là nó còn tính đến khoảng cách đã đi qua. Điều đó làm cho A\* "đầy đủ" và "tối ưu", nghĩa là A\* sẽ luôn luôn tìm thấy đường đi ngắn nhất nếu tồn tại một đường đi như vậy.

### 2.Cài đặt thuật toán

* G(N,A): Cây biểu diễn không gian trạng thái bài toán
* n0 : Trạng thái ban đầu của bài toán
* ĐICH: tập hợp các trạng thái đích
* Closed: tập hợp các điểm đã duyệt
* Fringe: là một cấu trúc hàng đợi sắp xếp theo chi phí đường đi. Thứ tự ưu tiên được xác định theo hàm **f(n) = g(n)+h(n)**. Trong đó:
  + **g(n)** là chi phí từ nút gốc n0 tới nút hiện tại n
  + **h(n)** là ước lượng heuristic
  + **f(n)** chi phí tổng thể ước lượng của đường đi qua nút hiện tại n đến đích



### 3.Các tính chất

* Nếu không gian các trạng thái là hữu hạn và có giải pháp để tránh việc xét (lặp) lại các trạng thái, thì giải thuật A\* là hoàn chỉnh (tìm được lời giải) – nhưng không đảm bảo là tối ưu
* Nếu không gian các trạng thái là hữu hạn và không có giải pháp để tránh việc xét (lặp) lại các trạng thái, thì giải thuật A\* là không hoàn chỉnh (không đảm bảo tìm được lời giải)
* Nếu không gian các trạng thái là vô hạn, thì giải thuật A\* là không hoàn chỉnh (không đảm bảo tìm được lời giải
* Tổng quát, A\* có tính hoàn chỉnh, tính tối ưu và độ phức tạp thời gian là bậc của hàm mũ
* Tính hoàn chỉnh?
  + Có (trừ khi có rất nhiều các nút có chi phí f *≤ f(G)* )
* Độ phức tạp về thời gian?
  + Bậc của hàm mũ – Số lượng các nút được xét là hàm mũ của độ

dài đường đi của lời giải

* Độ phức tạp về bộ nhớ?
  + Lưu giữ tất cả các nút trong bộ nhớ
* Tính tối ưu?
  + Có

### 4.Hàm ước lượng heuristic

#### a.Các khái niệm

* Một ước lượng (heuristic) ***h(n)*** được xem là chấp nhận được nếu đối với mọi nút n:

**0 ≤ *h(n)* ≤ h\*(n),**

trong đó ***h\*(n)*** là chi phí thật (thực tế) để đi từ nút n đến đích

* Một ước lượng chấp nhận được không bao giờ đánh giá quá cao (overestimate) đối với chi phí để đi tới đích.
* Định lý: Nếu ***h(n)*** là đánh giá chấp nhận được, thì phương pháp tìm kiếm A\* sử dụng giải thuật TREESEARCH là tối ưu.
* Ước lượng ***h2*** được gọi là ưu thế hơn / trội hơn (dominate) ước lượng ***h1*** nếu
* ***h1(n)*** và ***h2(n)*** đều là các ước lượng chấp nhận được, và
* ***h2(n) ≥ h1(n)*** đối với tất cả các nút n
* Nếu ước lượng ***h2*** ưu thế hơn ước lượng ***h1***, thì ***h2*** tốt hơn (nên được sử dụng hơn) cho quá trình tìm kiếm.
* Một ước lượng ***h*** được xem là kiên định (consistent), nếu với mọi nút n và với mọi nút tiếp theo n' của n (được sinh ra bởi hành động a)

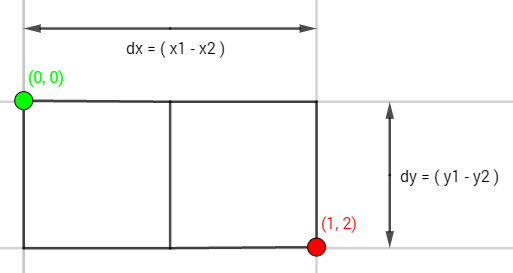
***h(n) ≤ c(n,a,n') + h(n')***

* Định lý: Nếu ***h(n)*** là kiên định, thì phương pháp tìm kiếm A\* sử dụng giải thuật GRAPH-SEARCH là tối ưu

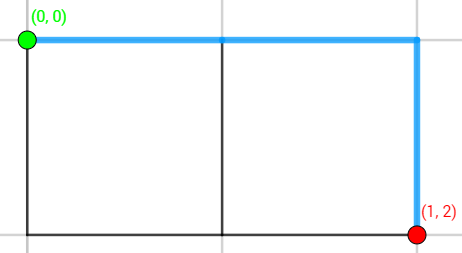
#### b.Các hàm heuristic

Trong bản đồ lưới, có các hàm heuristic được sử dụng như sau:

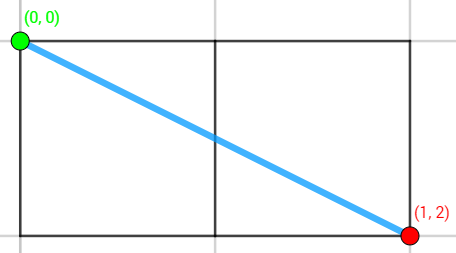
* Trong ô lưới vuông chỉ cho phép 4 hướng di chuyển, sử dụng Manhattan distance (L1).
* Trong ô lưới vuông cho phép 8 hướng di chuyển, sử dụng Diagonal distance (L∞).
* Trong ô lưới mà cho phép di chuyển đến bất kì hướng nào, sử dụng Euclidean distance (L2).



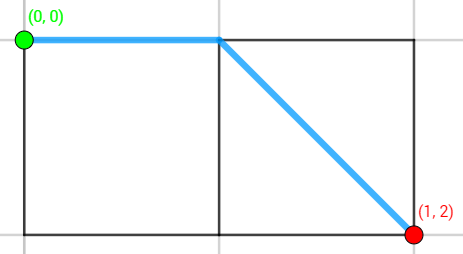
Manhattan distance



Euclide distance



Diagonal distance



#### c. Breaking Ties

Để đảm bảo “tính chấp nhận được” (admissibility), A\* sẽ phải duyệt qua các nút có giá trị đánh giá f bằng nhau. Nhiều con đường có chi phí bằng nhau sẽ bắt A\* phải mở rộng dần theo tất cả các con đường đó.

Để nới lỏng “tính chấp nhận được” thuật toán chỉ cần tìm thấy đường đi “tương đối” ngắn nhất điều này sẽ giúp tăng tốc đáng kể thời gian tính toán

Steven van Dijk gợi ý rằng cách để đi nhanh hơn về phá đích là giảm bớt sự so sánh h: khi f bằng nhau thì cần tránh trường hợp này bằng cách thay đổi h. Có 1 mẹo nhỏ cho vấn đề này là điều chỉnh g hoặc h: 1 lượng nhỏ tie breaker được thêm vào để làm cho f trở nên khác nhau mà ko làm mất đi tính chất ban đầu của => 1 cách thường dùng là điều chỉnh h 1 lượng rất nhỏ: nếu làm có h giảm nhẹ đi thì càng tiến về đích f sẽ càng tăng, đièu này làm cho A\* có xu hướng ưu tiên mở rộng hướng về điểm start thay vì điểm đích end. Vì vậy ta sẽ làm cho h tăng nhẹ: thậm chí là 0.1% thì A\* sẽ ưu tiên hướng về đích:

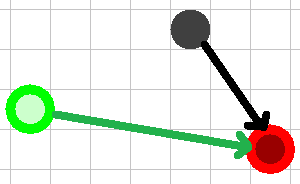
heuristic \*= (1.0 + p)

trọng số p < (Chi phí để di chuyển 1 bước) / (Chi phí tối đa để đi đến đích)

Breaking ties làm cho A\* tiến nhanh hơn!

Khi gặp tường chắn A\* vẫn sẽ tìm kiếm như bình thường nhưng thì thoát ra khỏi tường thì A\* sẽ ngay lậ tức tìm thấy đường đi.

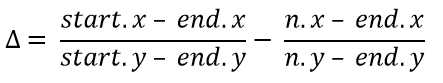
Trong Project này ta sử dụng phương pháp: Ưu tiên các điểm nằm trên đường chim bay từ Điểm xuất phát tới đích gần nhất:



Để làm được điều này ta có: vector (start, end): (start.x – end.x, start.y – end.y)

vector (n, end): (n.x – end.x, n.y – end.y)

Khi đó:



Ta có: delta càng nhỏ thì Điểm đang xét càng gần với đường chim bay từ Start đến End và ta lấy luôn mẫu số của delta (start.y – end.y)(n.y – end.y) = 1000 (Ta quy ước Đường đi thực tế tối đa của không gian đang xét không quá 1000)

Khi đó ta luôn tìm GTNN của h+delta hay h+cross\*0.001 chỉ thay đổi 1 lượng rất nhỏ nhưng có hiệu quả cao.

## Thuật toán Greedy best first

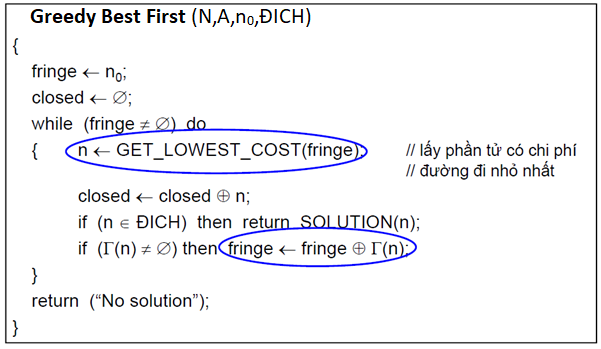
### 1.Mô tả thuật toán

Trong trí tuệ nhân tạo, thuật toán **greedy best first search** là một chiến lược tìm kiếm với tri thức bổ sung từ việc sử dụng các tri thức cụ thể của bài toán.

Thuật toán sẽ sử dụng 1 hàm đánh giá là hàm heuristic h(n), hàm heuristic h(n) này sẽ đánh giá chi phí để đi từ nút hiện tại n đến nút đích (mục tiêu). (***Các hàm heuristic trong phần nghiên cứu giống như trong thuật toán A\****)

### 2.Cài đặt thuật toán

* G(N,A): Cây biểu diễn không gian trạng thái bài toán
* n0 : Trạng thái ban đầu của bài toán
* ĐICH: tập hợp các trạng thái đích
* Closed: tập hợp các điểm đã duyệt
* Fringe: là một cấu trúc hàng đợi sắp xếp theo chi phí đường đi. Thứ tự ưu tiên được xác định theo hàm **f(n) = h(n)**. Trong đó:
  + **h(n)** là ước lượng heuristic
  + **f(n)** chi phí tổng thể ước lượng của đường đi qua nút hiện tại n đến đích



### 3.Các tính chất

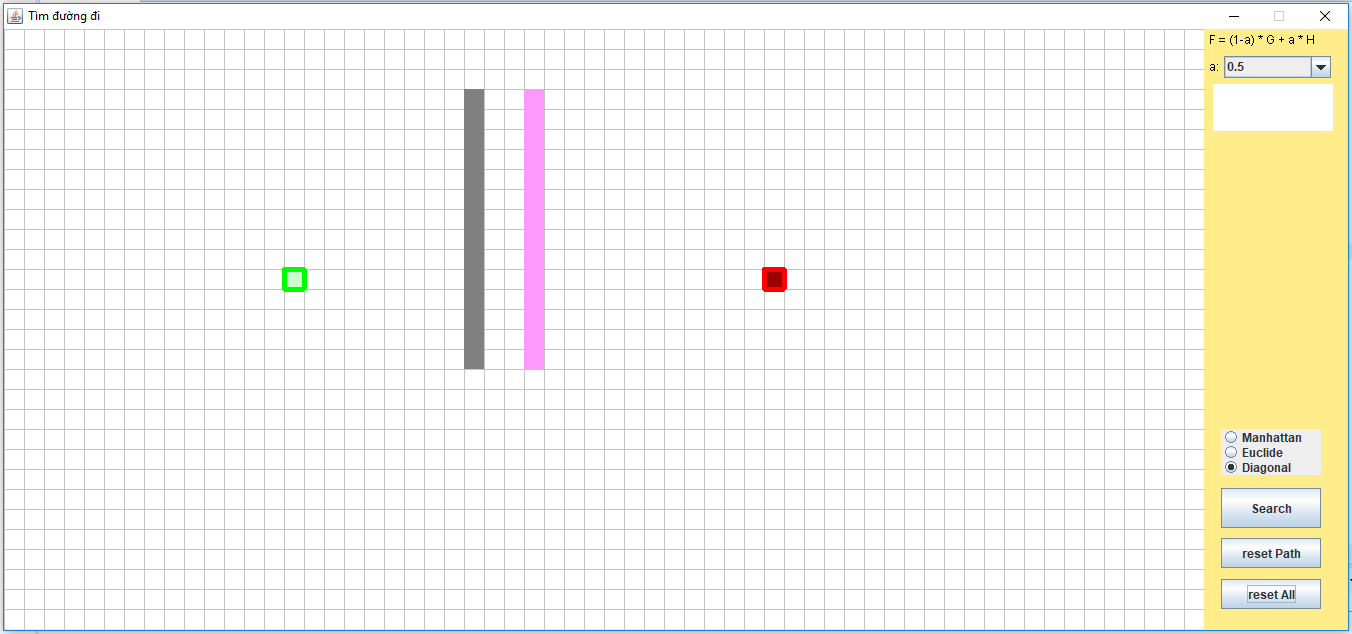
* Tính hoàn chỉnh?
  + Không Vì có thể vướng (chết tắc) trong các vòng lặp kiểu như:

Iasi -> Neamt -> Iasi -> Neamt ->…

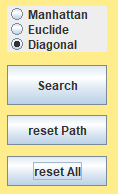
* Độ phức tạp về thời gian?
  + *O(bm)*
  + Một hàm heuristic tốt có thể mang lại cải thiện lớn
* Độ phức tạp về bộ nhớ?
  + *O(bm)* – Lưu giữ tất cả các nút trong bộ nhớ
* Tính tối ưu?
  + Không

# PHẦN 3: ĐÁNH GIÁ KẾT QUẢ

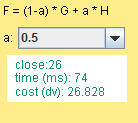
## 1.Giao diện người dùng



Trong đó, khu vực điều khiển gồm:



* Cột 1: lựa chọn hàm heuristic
* Phím “search”: bắt đầu tìm kiếm
* Phím “reset Path” xóa đường đi đã tìm thấy
* Phím “reset All” xóa hết tất cả các tường và đường đi đã tìm thấy



* Vùng khoảng trắng: hiển thị thông tin đường đi
* a: lựa chọn giữa 0,0 (giải thuật USC) hoặc 0,5 (giải thuật A\*) và 1,0 (giải thuật Greedy Best First)

Khu vực bản đồ lưới (grid):

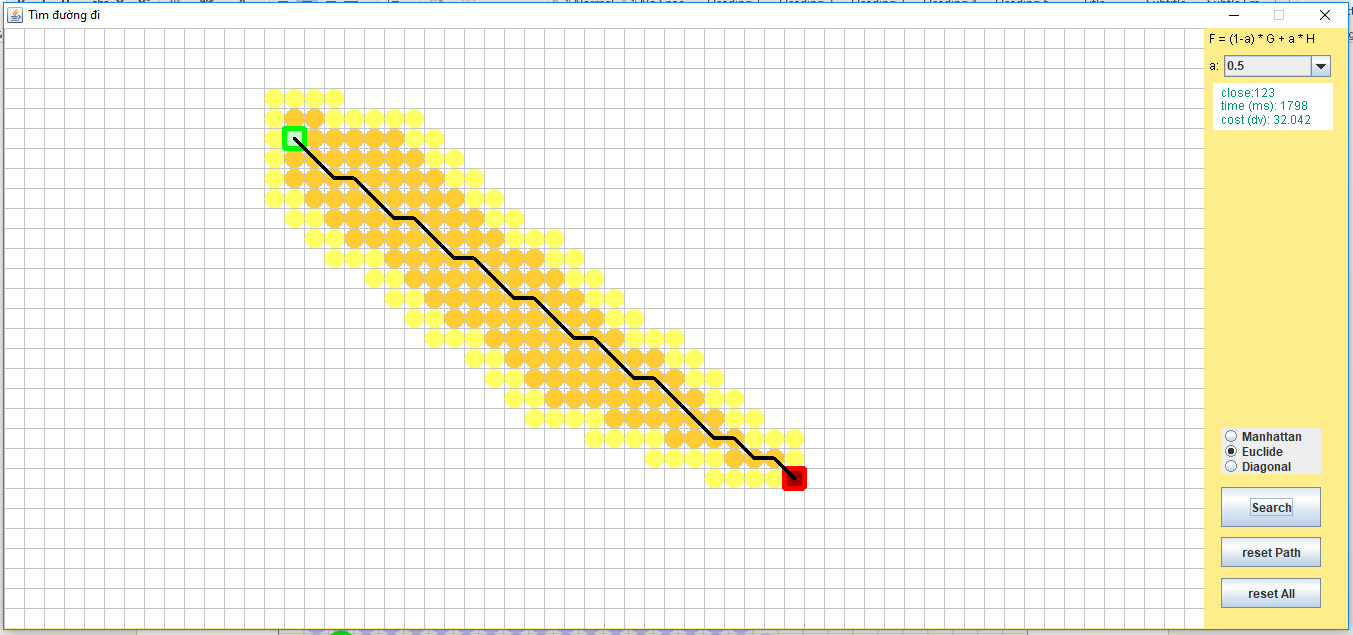
* Một điểm đầu (màu xanh)
* Một điểm đích (màu đỏ)
* Các ô lưới có màu:
  + Trắng: cho phép đối tượng đi qua
  + Xám (tường): không cho phép đối tượng đi qua
  + Hồng: cho phép đối tượng đi qua nhưng với chi phí cao hơn (gấp 2 lần bình thường – vùng Trắng)

## 2.Lựa chọn hàm heuristic

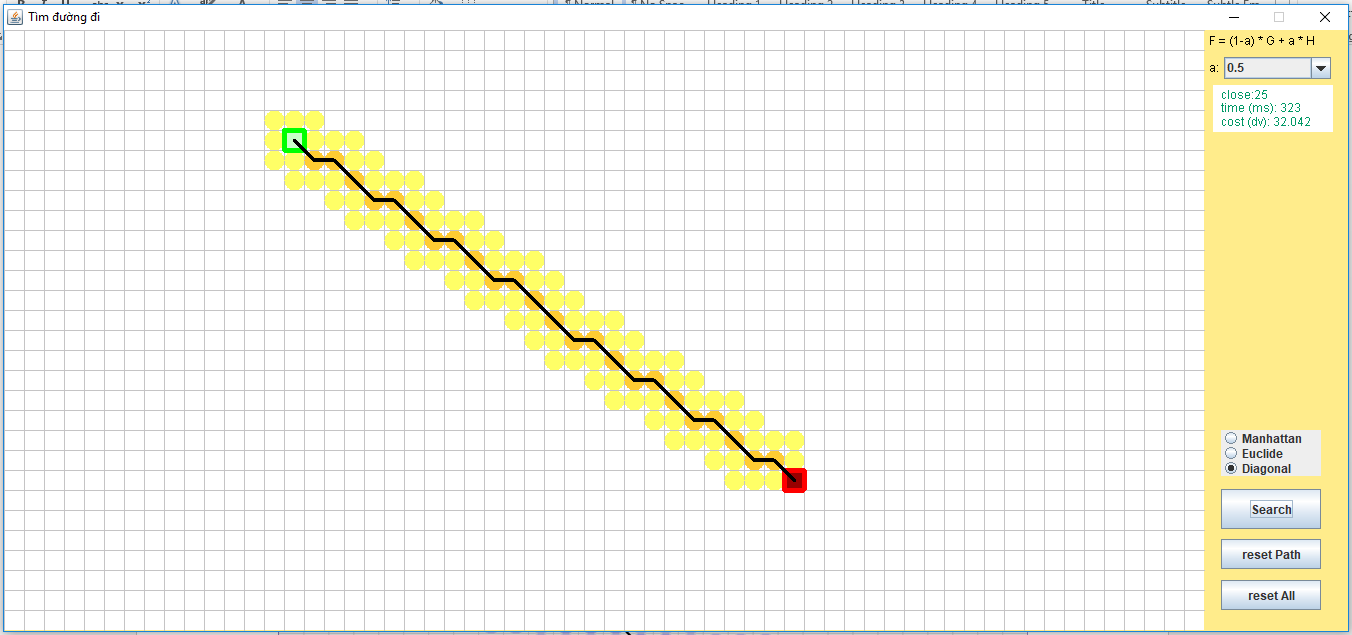
Một số thí nghiệm để so sánh giữa các hàm heuristic sau:

* (Euc) Euclide distance
* (Dia) Diagonal distance
* (Man) Manhattan distance

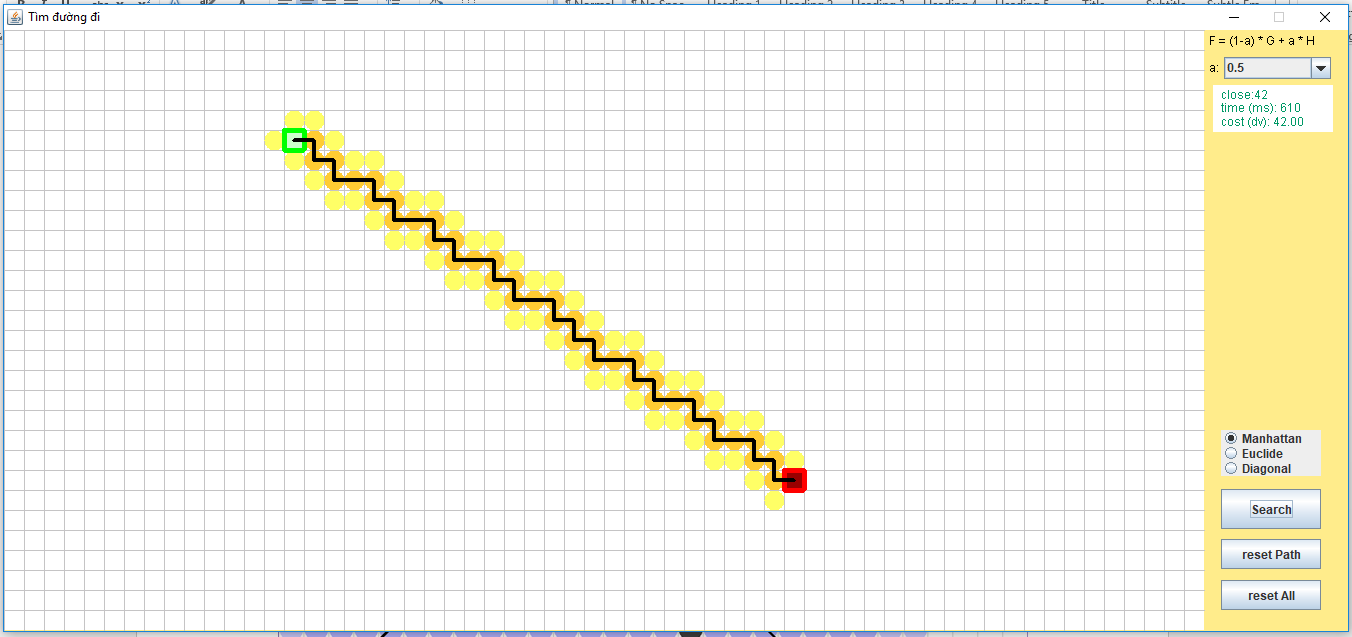
**TRƯỜNG HỢP 1:**



Euc trong TH1

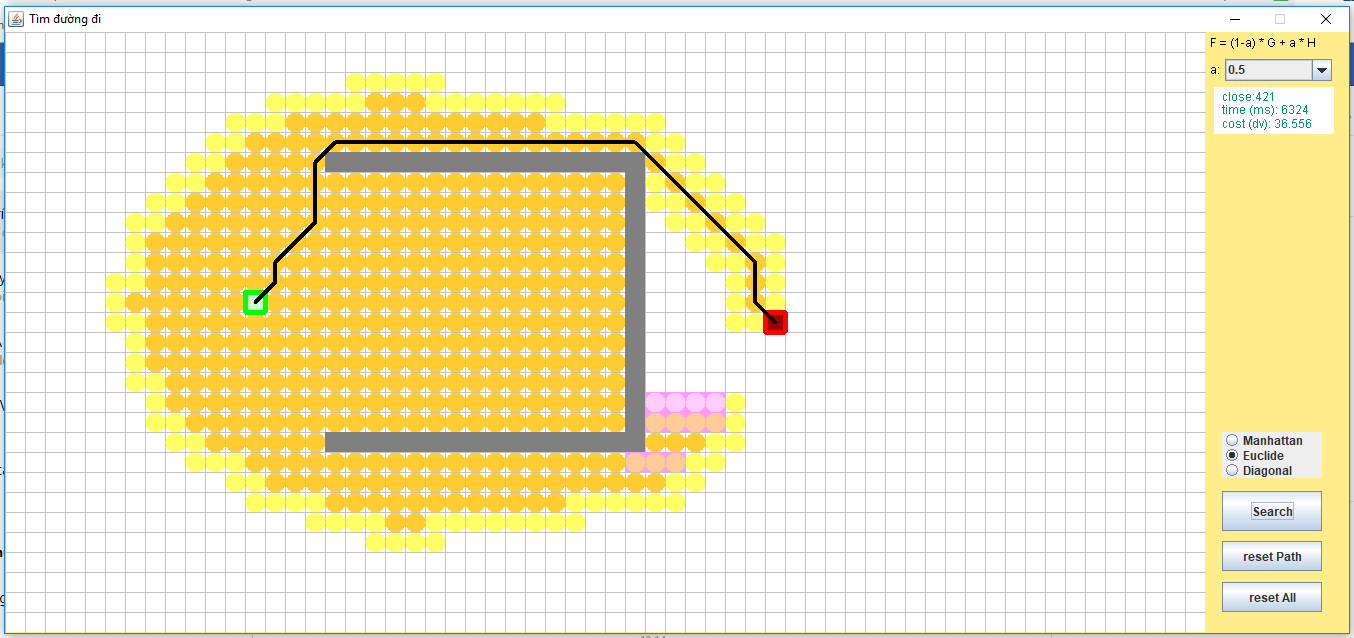


Dia trong TH1

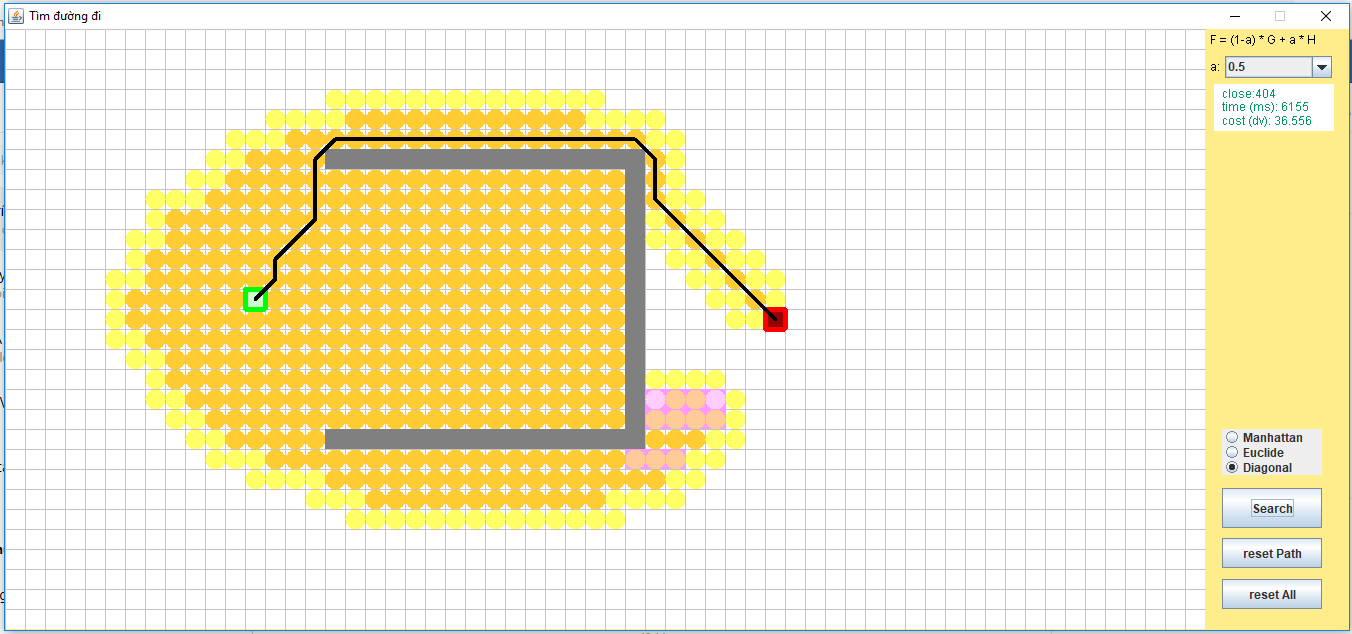


Man trong TH1

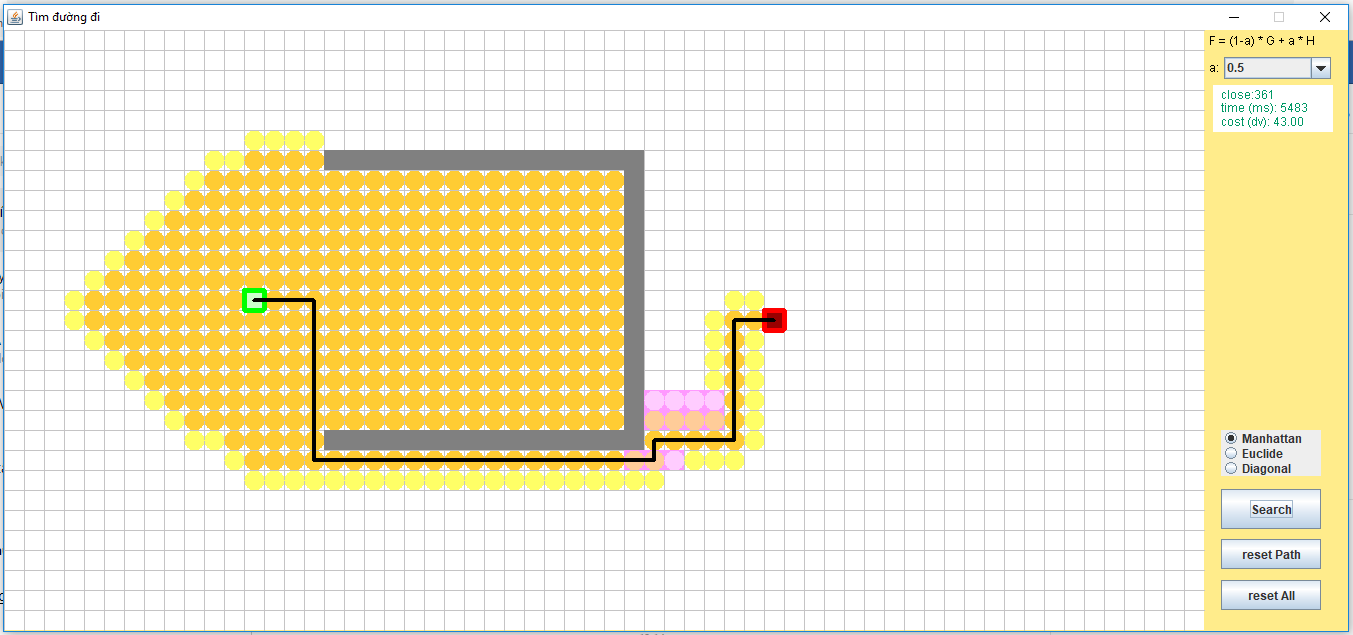
**TRƯỜNG HỢP 2:**



Euc trong TH2



Dia trong TH2



Man trong TH2

Thống kê lại 1 số kết quả sau các thí nghiệm chúng em đã thực hiện :

Bảng so sánh kích thước Closed Set

Bảng so sánh thời gian tính toán

***NHẬN XÉT: Trong tất cả các trường hợp được đánh giá, hàm sử dụng Diagonal distance luôn tìm ra kết quả tối ưu trong thời gian ngắn hơn so với 2 hàm ước lượng còn lại.***

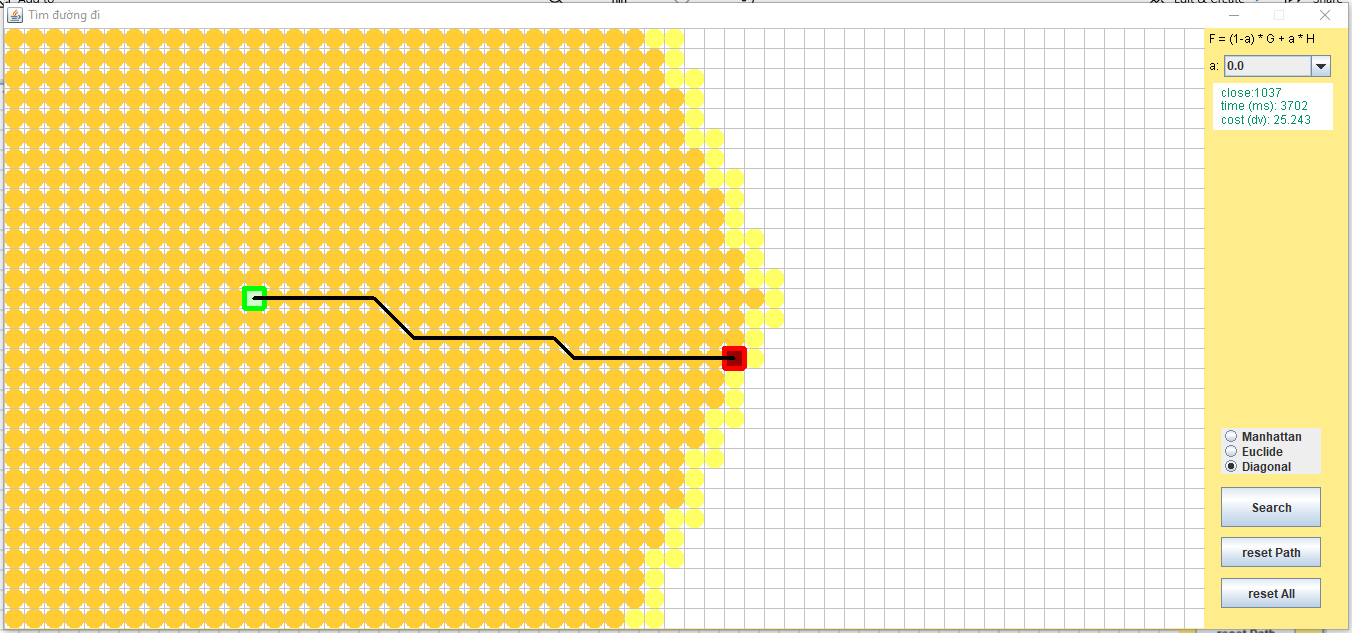
***🡺Hàm sử dụng Diagonal distance là lựa chọn tốt nhất***

## 3. Lựa chọn giải thuật

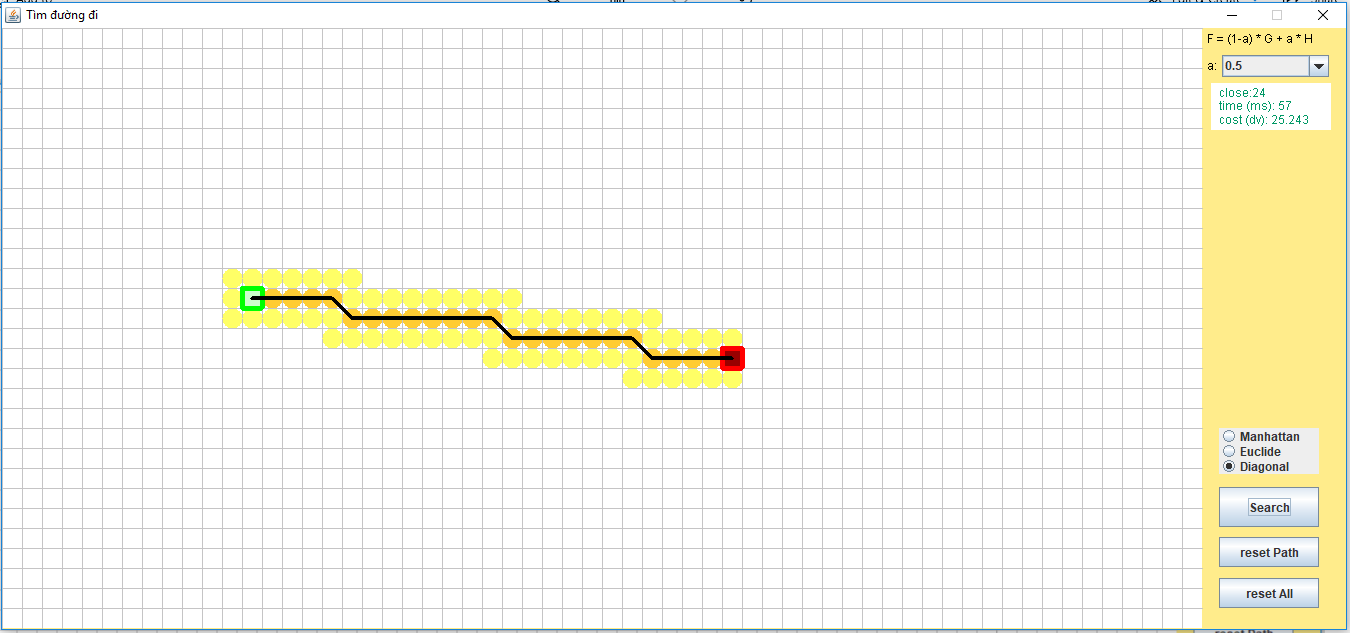
Một số thí nghiệm để so sánh giữa các giải thuật sau (Dùng hàm Dìgonal):

* Thuật toán UCS (chi phí cực tiểu)
* Thuật toán A\*
* Thuật toán Greedy Best First

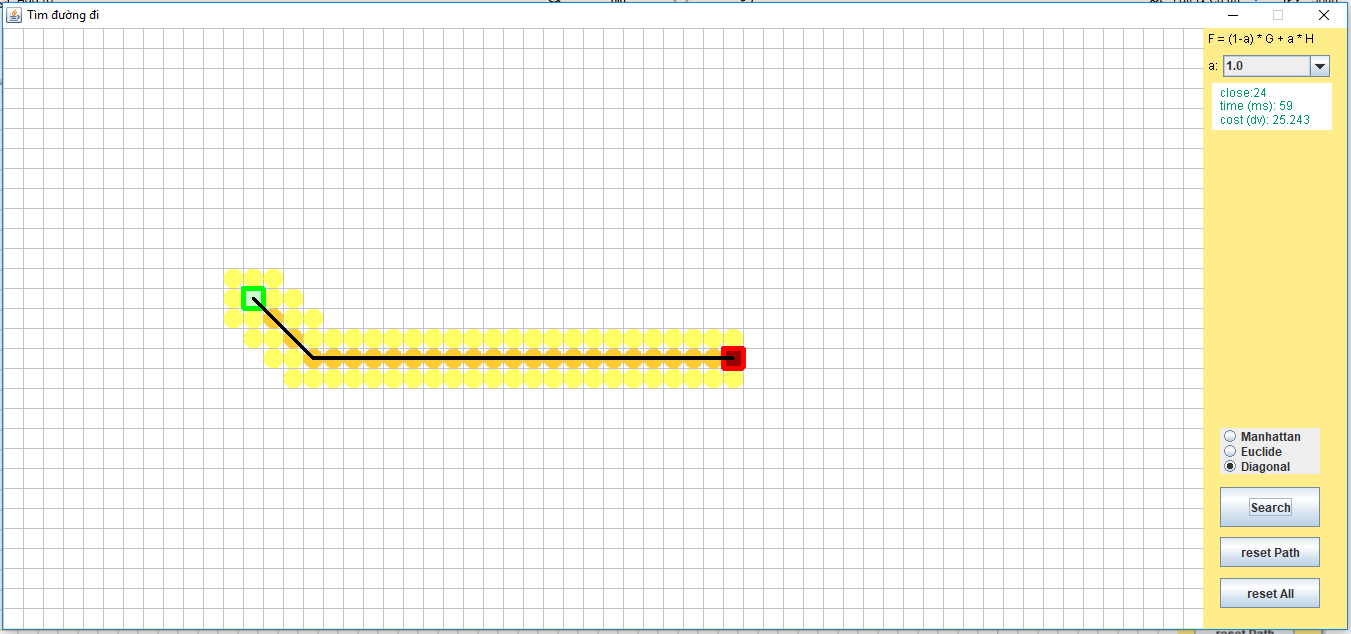
**TRƯỜNG HỢP 1:**

****

USC trong TH1

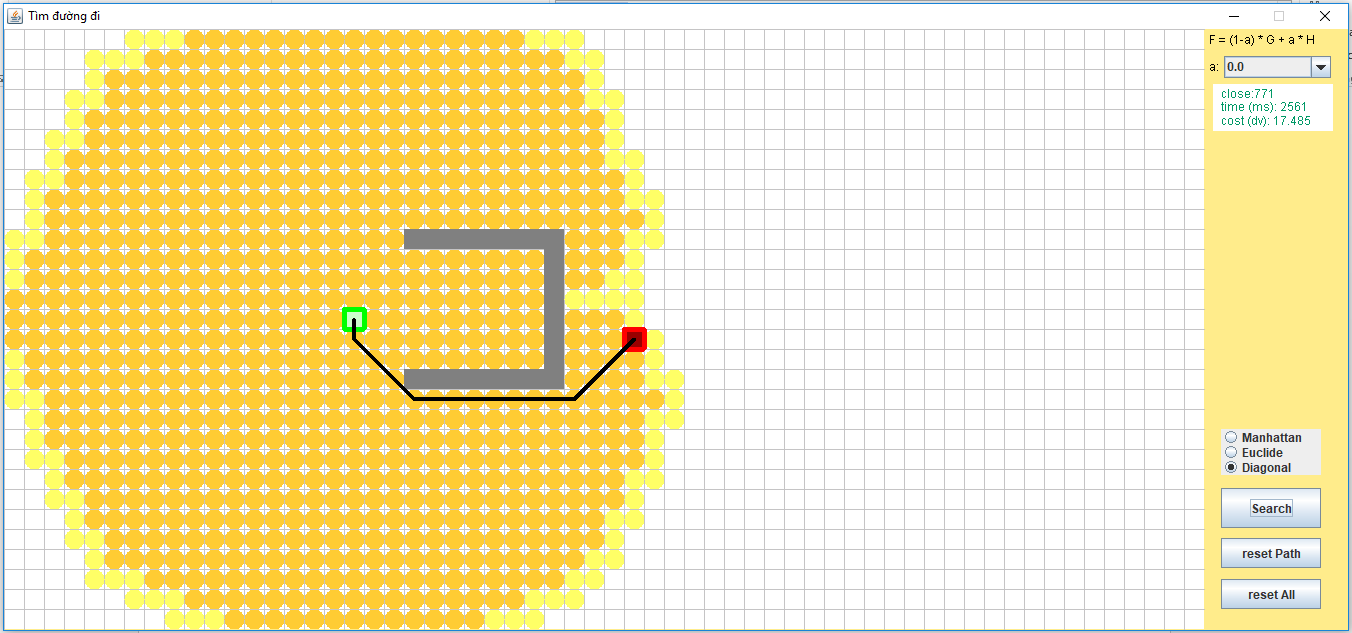


A\* trong TH1

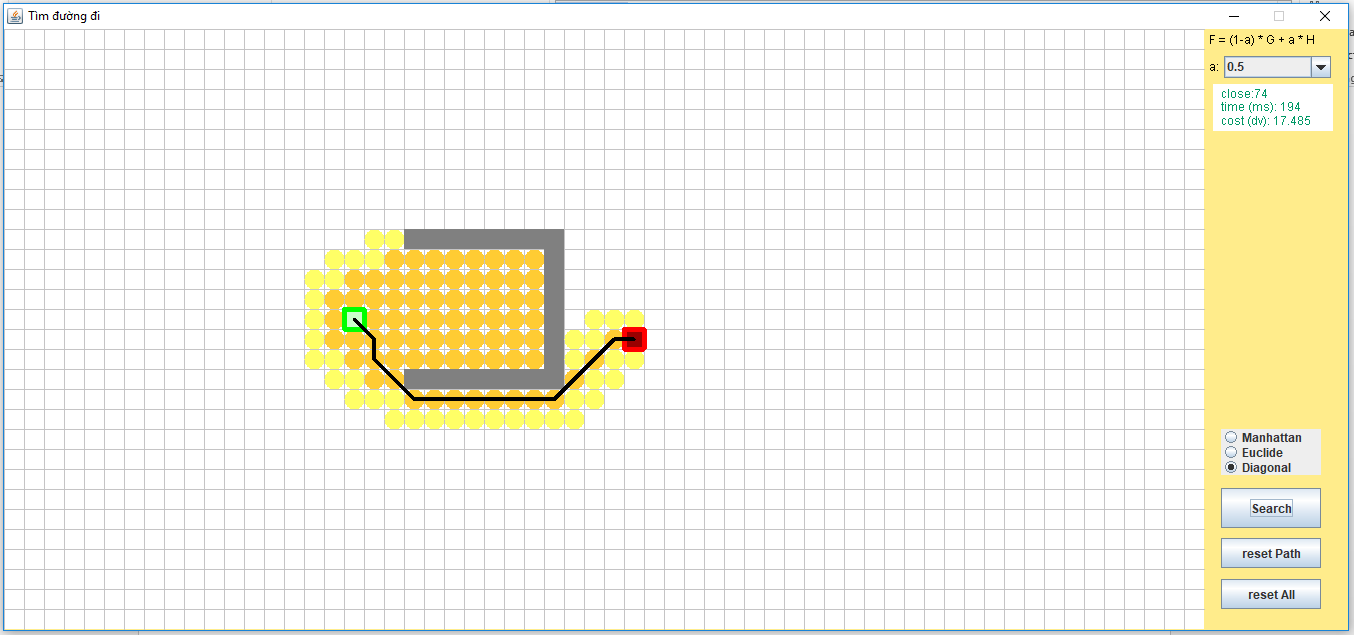


GBF trong TH1

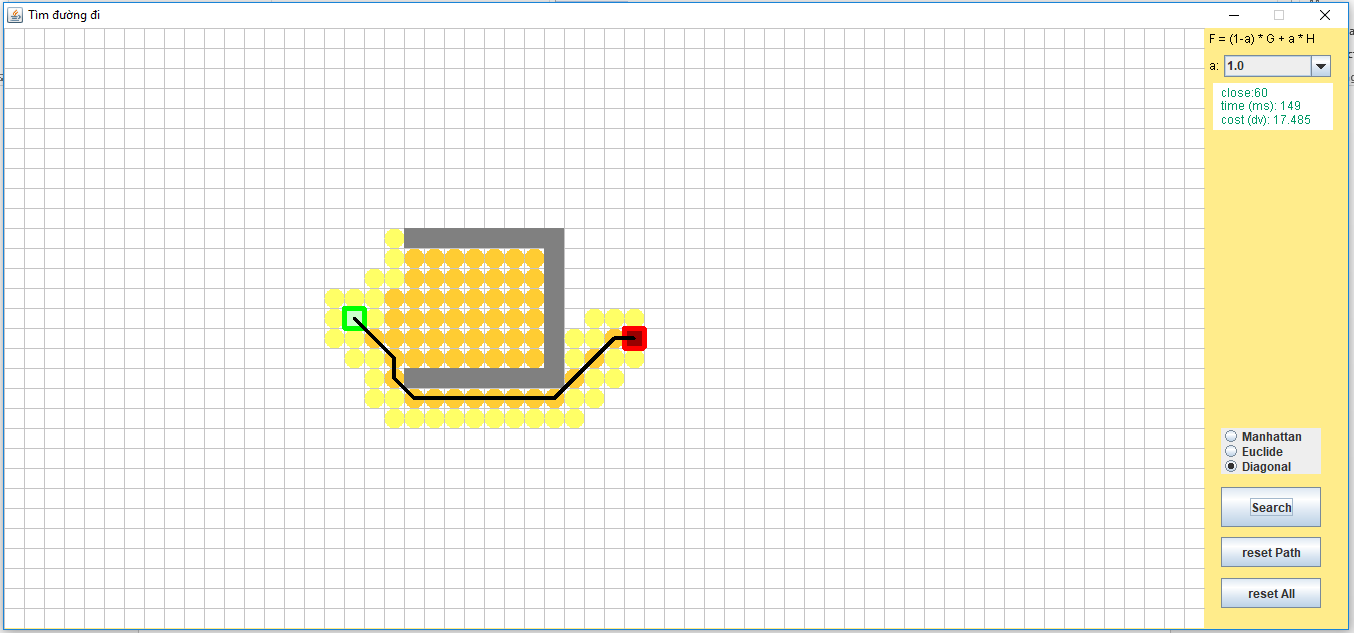
**TRƯỜNG HỢP 2**

****

USC trong TH2



A\* trong TH2



GBF trong TH2

Thống kê lại 1 số kết quả sau các thí nghiệm chúng em đã thực hiện:

Bảng so sánh kích thước Closed Set

Bảng so sánh thời gian tính toán

Bảng so sánh chi phí quãng đường

***NHẬN XÉT: Qua kết quả, có thể thấy rằng giải thuật Greedy best first tuy tìm ra đường đi với thời gian nhanh nhất, kích thước closed set ít nhất trong một số trường hợp nhưng giải thuật A\* vẫn vượt trội hơn hẳn bởi nó tìm ra đường đi với chi phí thấp nhất, bên cạnh đó vẫn đảm bảo yếu tố thời gian và kích thước closed set (lớn hơn GBF không đáng kể và nhỏ hơn UCS rất nhiều), còn về giải thuật UCS tuy cũng tìm ra đường đi với chi phí thấp tương đương A\* nhưng thời gian tìm kiếm và kích thước closed set lại quá lớn.***

***🡺 A\* là giải thuật tốt nhất***

# KẾT LUẬN

Thông qua việc tìm hiểu và nghiên cứu đề tài này giúp em có cái nhìn toàn diện hơn về trí tuệ nhân tạo và ứng dụng của nó vào giải quyết các bài toán thực tiễn. Tìm đường đi ngắn nhất là một bài toán điển hình trong trí tuệ nhân tạo cho các thuật toán mô hình hóa liên quan đến tìm kiếm có tri thức bổ sung. Đề tài này đã được nhiều người nghiên cứu giải quyết, nhưng cho đến nay vẫn chưa có cách giải quyết tối ưu cho tất cả không gian trạng thái trò chơi vì kích thước tăng không gian trạng thái là hàm mũ sẽ tăng lên rất nhanh. Hy vọng những nghiên cứu đánh giá của em sẽ góp phần bổ sung thêm một hướng giải quyết cho bài toán. Do thời gian có hạn nên đề tài không tránh khỏi những sai sót, mong thầy góp ý, đánh giá giúp em hoàn thiện đề tài.

# TÀI LIỆU THAM KHẢO

* wikipedia:

[https://en.wikipedia.org/wiki/a\*\_search\_algorithm](https://en.wikipedia.org/wiki/A*_search_algorithm)

* amit’s thought on path finding:
* <https://www.redblobgames.com/pathfinding/a-star/introduction.html>
* Slide môn Trí tuệ nhân tạo(thầy Nguyễn Nhật Quang)