

**Báo cáo**

Đề tài: Thuật toán A\*

Môn : Trí tuệ nhân tạo

GV: Trần Đình Toàn

Nhóm: 5

|  |  |  |
| --- | --- | --- |
| STT | Họ và Tên | MSSV |
| 1 | Nguyễn Thị Thùy Linh | 2001210927 |
| 2 | Nguyễn Ngọc Quân | 2001210779 |
| 3 | Phùng Huỳnh Thanh Ngân | 2001210235 |

**Mục Lục**

[1.Lời mở đầu 3](#_Toc144071625)

[2. Ý tưởng và Phương pháp 3](#_Toc144071626)

[3. Hàm heuristic 4](#_Toc144071627)

[4.Thuật toán A\* 4](#_Toc144071628)

[4.1 Thuật toán A\* là gì? 4](#_Toc144071629)

[4.2. Mô tả thuật toán: 5](#_Toc144071630)

[4.3.Cài đặt thuật toán: 5](#_Toc144071631)

[4. Công dụng của thuật toán: 11](#_Toc144071632)

[4.4.Ưu điểm 12](#_Toc144071633)

[4.5.Nhược điểm 12](#_Toc144071634)

[5.So sánh với thuật toán Dijkstra 12](#_Toc144071635)

[6. Mối quan hệ thuật toán a\* với các thuật toán khác 13](#_Toc144071636)

[7.Ứng dụng thực tế 14](#_Toc144071637)

[8.Đánh giá và kết luận 15](#_Toc144071638)

[9. Đánh giá và nhận xét thành viên: 16](#_Toc144071639)

[10.Nguồn tham khảo 17](#_Toc144071640)

# **1.Lời mở đầu**

Trí tuệ nhân tạo (AI) là lĩnh vực nghiên cứu về việc tạo ra các hệ thống máy tính có thể thực hiện các tác vụ đòi hỏi trí thông minh của con người, chẳng hạn như học hỏi, suy luận, giải quyết vấn đề và giao tiếp.

AI đã được ứng dụng trong nhiều lĩnh vực thực tế, bao gồm:

* Học máy: Học máy giúp máy tính học từ dữ liệu mà không cần được lập trình rõ ràng.
* Xử lý ngôn ngữ tự nhiên: Xử lý ngôn ngữ tự nhiên giúp máy tính xử lý ngôn ngữ của con người.
* Trí tuệ nhân tạo hình ảnh: Trí tuệ nhân tạo hình ảnh giúp máy tính xử lý hình ảnh.
* Robotics: Robotics giúp phát triển các robot.

AI là một lĩnh vực đang phát triển nhanh chóng và có tác động sâu sắc đến cuộc sống của chúng ta.

Trong khoa học máy tính, thuật toán tìm kiếm là một trong những vấn đề quan trọng. Có nhiều thuật toán tìm kiếm khác nhau, mỗi thuật toán có ưu nhược điểm riêng. Trong đó, thuật toán A\* là một trong những thuật toán tìm kiếm phổ biến nhất. Thuật toán A\* được mô tả lần đầu vào năm 1968 bởi Peter Hart, Nils Nilsson, và Bertram Raphael. Thuật toán này tìm một đường đi từ một nút khởi đầu tới một nút đích cho trước (hoặc tới một nút thỏa mãn một điều kiện đích).

# **2. Ý tưởng và Phương pháp**

Ý tưởng chính của thuật toán A\* là sử dụng một hàm heuristic để ước tính chi phí của đường đi từ một nút đến nút đích. Ý tưởng này giúp thuật toán A\* tìm ra đường đi tối ưu từ nút khởi đầu đến nút đích.

Thuật toán A\* có thể được mô tả như sau:

1. Khởi tạo một danh sách mở (open list) và danh sách đóng (closed list).
2. Thêm nút khởi đầu vào danh sách mở.
3. Lặp lại các bước sau cho đến khi tìm thấy nút đích hoặc danh sách mở trống:
   * Lấy nút có chi phí ước tính nhỏ nhất trong danh sách mở.
   * Kiểm tra xem nút này có phải là nút đích hay không. Nếu có, kết thúc thuật toán và trả về đường đi từ nút khởi đầu đến nút đích.
   * Thêm tất cả các nút lân cận của nút này vào danh sách mở.
   * Cập nhật chi phí ước tính của các nút lân cận này.
4. Nếu danh sách mở trống, thì không có đường đi nào từ nút khởi đầu đến nút đích.

# **3. Hàm heuristic**

Heuristic là phương pháp giải quyết vấn đề dựa trên phỏng đoán, ước chừng, kinh nghiệm, trực giác để tìm ra giải pháp gần như là tốt nhất và nhanh chóng.

Hàm Heuristic là hàm ứng với mỗi trạng thái hay mỗi sự lựa chọn một giá trị ý nghĩa đối với vấn đề dựa vào giá trị hàm này ta lựa chọn hành động.

Hàm heuristic là một hàm ước tính chi phí của đường đi từ một nút đến nút đích. Hàm heuristic có thể được tính toán dựa trên các thông tin về đồ thị và nút.

Một số hàm heuristic phổ biến bao gồm:

* **Hàm Manhattan**: Tính khoảng cách Manhattan giữa nút hiện tại và nút đích.
* **Hàm Euclidean**: Tính khoảng cách Euclidean giữa nút hiện tại và nút đích.
* **Hàm Djikstra:** Tính tổng chi phí di chuyển từ nút hiện tại đến nút đích.

# **4.Thuật toán A\***

4.1 Thuật toán A\* là gì?  
Thuật toán A\* (A-star): là một thuật toán tìm kiếm đường đi trong đồ thị hoặc không gian trạng thái. Nó được sử dụng phổ biến trong các bài toán tìm đường đi từ một điểm xuất phát đến một điểm đích trên bản đồ hoặc trong không gian trạng thái.

Thuật toán A\* kết hợp giữa hai phương pháp tìm kiếm: tìm kiếm theo chiều rộng (BFS - Breadth-First Search) và tìm kiếm theo chi phí nhỏ nhất (Best-First Search). Nó sử dụng một hàm ước lượng để đánh giá chi phí còn lại từ một nút đến đích dự đoán. Hàm ước lượng này thường được gọi là hàm heuristic.

## 4.2. Mô tả thuật toán:

Giả sử n là một trạng thái đã được đạt tới, có một đường đi từ trạng thái ban đầu n0 đến trạng thái n. Chúng ta sử dụng hàm đánh giá f(n) để định giá chi phí tổng thể ước lượng để đi từ trạng thái ban đầu đến trạng thái n.

F(n)=g(n)+h(n)

Trong đó:

* g(n) là chi phí tích lũy từ trạng thái ban đầu n0 đến trạng thái hiện tại n.
* h(n) là một ước lượng chi phí từ trạng thái hiện tại n đến trạng thái đích.
* f(n) là tổng chi phí ước lượng để đi từ trạng thái ban đầu đến trạng thái n thông qua đường đi hiện tại.

Một ước lượng heuristic h(n) được xem là chấp nhận được nếu với mọi trạng thái n, giá trị ước lượng h(n) nằm trong khoảng từ 0 đến h\*(n), trong đó h\*(n) là chi phí thực tế để đi từ trạng thái n đến trạng thái đích.

Điều này đảm bảo rằng ước lượng heuristic không vượt quá chi phí thực tế tối ưu nhất để đạt được đích. Điều kiện 0 ≤ h(n) ≤ h\*(n) đảm bảo tính chấp nhận được của ước lượng heuristic trong thuật toán A\*.

Thuật toán A\* sử dụng hàm đánh giá f(n) để tìm kiếm đường đi tối ưu từ trạng thái ban đầu đến trạng thái đích thông qua việc lựa chọn các trạng thái tiếp theo dựa trên giá trị f(n) nhỏ nhất. Kết hợp giữa chi phí tích lũy g(n) và ước lượng chi phí h(n) giúp thuật toán A\* tìm kiếm một đường đi có chi phí tổng thể là tối ưu hoặc rất gần tối ưu.

## 4.3.Cài đặt thuật toán:

Cách hoạt động của thuật toán A\* như sau:

1. Khởi tạo một hàng đợi ưu tiên (priority queue) và thêm điểm xuất phát vào hàng đợi. Gán chi phí g(n) của điểm xuất phát là 0.

2. Trong khi hàng đợi không rỗng, thực hiện các bước sau:

• Lấy nút có chi phí thấp nhất (f(n) = g(n) + h(n)) từ hàng đợi.

• Nếu nút này là điểm đích, thuật toán đã tìm thấy đường đi thành công. Kết thúc thuật toán.

• Nếu không, mở rộng nút này bằng cách tạo các nút con và tính toán chi phí g(n) và h(n) cho các nút con. Thêm các nút con vào hàng đợi.

• Nếu một nút con đã tồn tại trong hàng đợi với chi phí thấp hơn, cập nhật lại chi phí g(n) của nút con đó.

3. Nếu hàng đợi trở thành rỗng mà không tìm thấy đường đi, tức là không có đường đi từ điểm xuất phát đến điểm đích.

Hàm heuristic (h(n)) được sử dụng để ước lượng chi phí còn lại từ một nút đến đích. Hàm heuristic phải là một ước lượng hợp lý, không quá lớn so với chi phí thực tế. Một ví dụ thông thường của hàm heuristic là khoảng cách Euclid từ nút hiện tại đến điểm đích trên mặt phẳng.

Thuật toán A\* có khả năng tìm ra đường đi tối ưu với độ phức tạp thời gian tương đối hiệu quả. Tuy nhiên, nếu không được triển khai đúng cách hoặc trong một không gian trạng thái lớn, nó có thể tốn rất nhiều bộ nhớ và thời gian tính toán.

VD: Cho đồ thị tìm đường đi ngắn nhất từ A đến L hoặc N

6

A

1

3

2

3

4

5

5

D

C

B

nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn

4

2

4

2

4

1

4

2

6

3

5

2

1

3

2

0

4

4

0

6

4

I

O

N

L

M

K

K

H

G

F

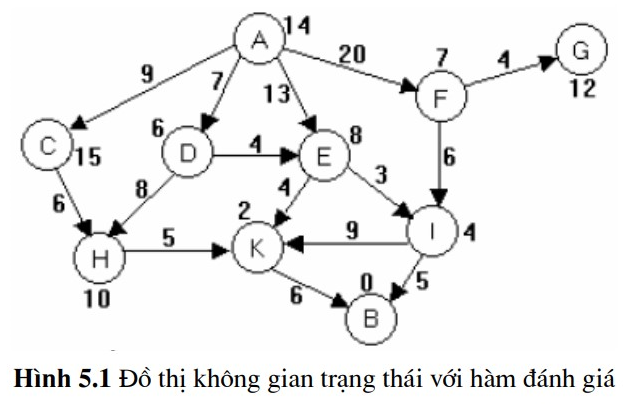
E

Giải:

|  |  |  |
| --- | --- | --- |
| STT | Nút mở rộng | Tập biên Opend |
| 0 |  | A(0+6) |
| 1 | A | B(2+3) ,**C**(1+4) ,D(3+5) |
| 2 | B | F(6+1), E(7+3) |
| 3 | C | **H**(4+2), G(7+6) |
| 4 | H | N(6+0) |
| 5 | N | Đích |

* N H C A
* A C H N

Chi Phí: cp=6



Thuật toán A\* là thuật toán sử dụng kỹ thuật tìm kiếm tốt nhất đầu tiên với hàm đánh giá f(u).

Để thấy được thuật toán A\* làm việc như thế nào, ta xét đồ thị không gian trạng thái trong hình 5.1. Trong đó, trạng thái ban đầu là trạng thái A, trạng thái đích là B, các số ghi cạnh các cung là độ dài đường đi, các số cạnh các đỉnh là giá trị của hàm h. Đầu tiên, phát triển đỉnh A sinh ra các đỉnh con C, D, E và F.

Tính giá trị của hàm f tại các đỉnh này ta có:

g(C) = 9, f(C) = 9+15=24, g(D) = 7, f(D) = 7+6 =13

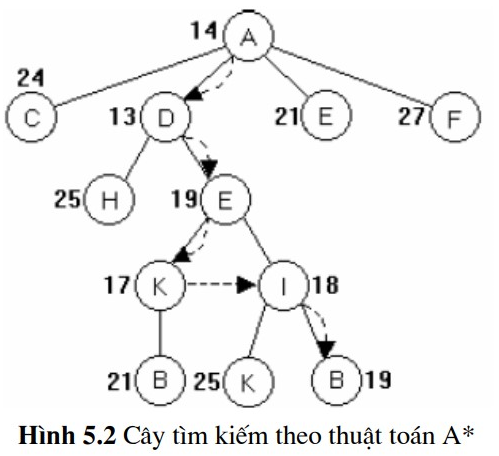
g(E) = 13, f(E) = 13+ 8 = 21, g(F) = 20, f(F) = 20+7 = 27

Như vậy đỉnh tốt nhất là D (vì f(D) = 13 là nhỏ nhất). Phát triển D, ta nhận được các đỉnh con H và E. Ta đánh giá H và E (mới):

g(H) = g(D) + Độ dài cung(D, H) = 7 + 8 = 15, f(H) = 15 + 10 = 25.

Đường đi tới E qua D có độ dài: g(E) = g(D) + Độ dài cung(D, E) = 7+4=11.

Vậy đỉnh E mới có đánh giá là f(E) = g(E) + h(E) = 11 + 8 = 19. Trong số các đỉnh cho phát triển, thì đỉnh E với đánh giá f(E) = 19 là đỉnh tốt nhất. Phát triển đỉnh này, ta nhận được các đỉnh con của nó là K và I. Chúng ta tiếp tục quá trình trên cho tới khi đỉnh được chọn để phát triển là đỉnh kết thúc B, độ dài đường đi ngắn nhất tới B là g(B)= 19. Quá trình tìm kiếm trên được mô tả bởi cây tìm kiếm trong hình 5.2, trong đó các số cạnh các đỉnh là các giá trị của hàm đánh giá f(u).



**procedure A\*;**

**begin**

1. Khởi tạo danh sách L chỉ chứa trạng thái ban đầu;

**2. loop do**

end;

2.1 **if** L rỗng **then**

{thông báo thất bại; stop };

2.2 Loại trạng thái u ở đầu danh sách Li

2.3 **if** u là trạng thái đích **then**

{thông báo thành công; **stop** }

2.4 **for** mỗi trạng thái v kề u **do**

{g(v)g(u)+k(u,v);

f(v) ←g(v) + h(v);

Đặt v vào danh sách L; }

2.5 Sắp xếp L theo thứ tự tăng dần của hàm f sao cho trạng thái có giá trị của

hàm f nhỏ nhất ở đầu danh sách;

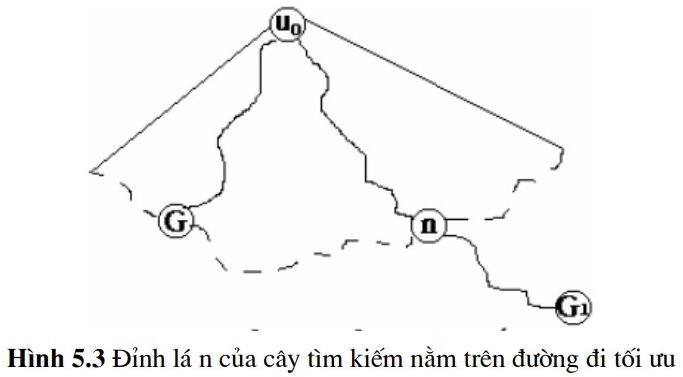
**end;**

Chúng ta đưa ra một số nhận xét về thuật toán A\*.

• Người ta chứng minh được rằng, nếu hàm đánh giá h(u) là đánh giá thấp nhất (trường hợp đặc biệt, h(u) = 0 với mọi trạng thái u) thì thuật toán A\* là thuật toán tối ưu, tức là nghiệm mà nó tìm ra là nghiệm tối ưu. Ngoài ra, nếu độ dài của các cung không nhỏ hơn một số dương δ nào đó thì thuật toán A\* là thuật toán đầy đủ theo nghĩa rằng, nó luôn dừng và tìm ra nghiệm.

Chúng ta chứng minh tính tối ưu của thuật toán A\*.

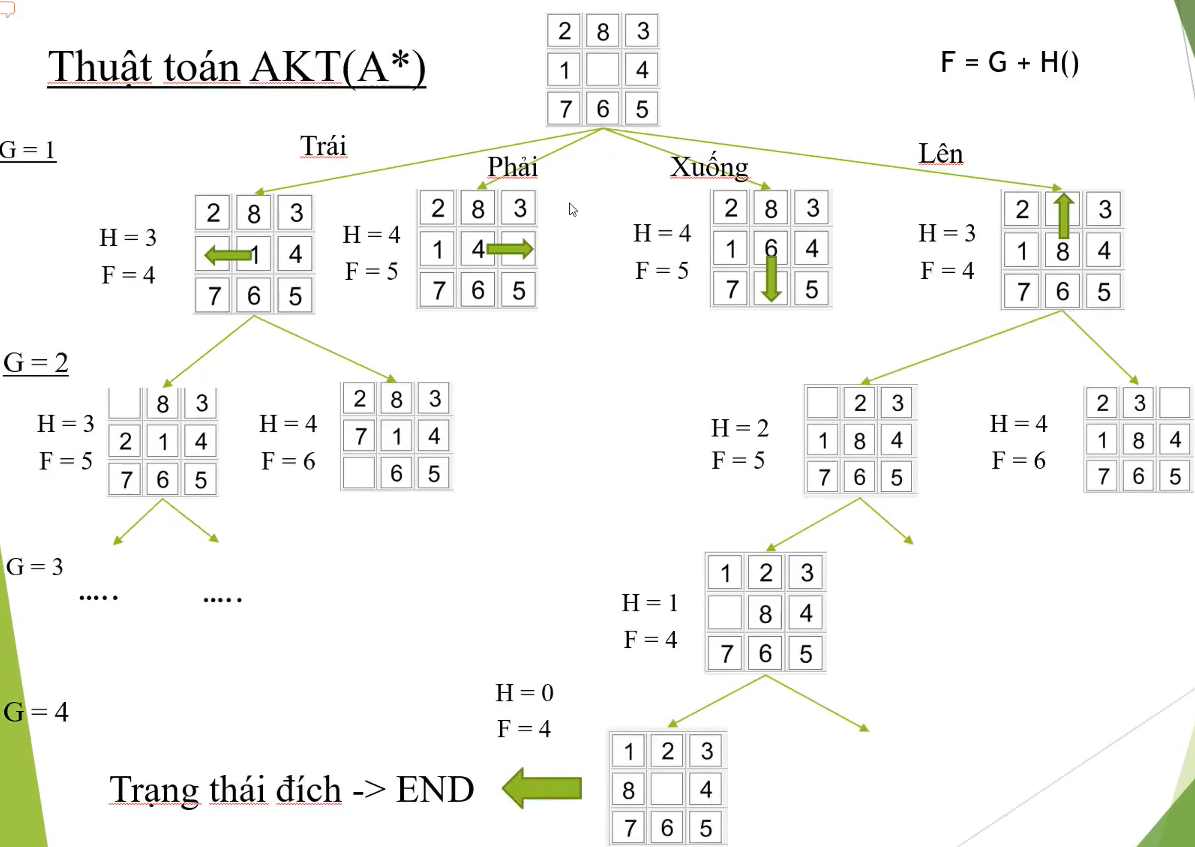
Giả sử thuật toán dừng lại ở đỉnh kết thúc G với độ dài đường đi từ trạng thái ban đầu u0 tới G là g(G). Vì G là đỉnh kết thúc, ta có h(G) = 0 và f(G) = g(G) + h(G) = g(G). Giả sử nghiệm tối ưu là đường đi từ u0 tới đỉnh kết thúc G1 với độ dài 1. Giả sử đường đi này “thoát ra” khỏi cây tìm kiếm tại đỉnh lá n (Xem hình 5.3). Có thể xảy ra hai khả năng: n trùng với G1, hoặc không. Nếu n là Gı thì vì G được chọn để phát triển trước Gı, nên f(G) ≤ f(Gı), do đó g(G) ≤g(Gı)=1. Nếu n ≠ G, thì do h(u) là hàm đánh giá thấp, nên f(n) = g(n) +h(n)≤1. Mặt khác, cũng do G được chọn để phát triển trước n nên f(G)<f(n), do đó, g(G) ≤1.



Như vậy, ta đã chứng minh được rằng độ dài của đường đi mà thuật toán tìm ra g(G) không dài hơn độ dài 1 của đường đi tối ưu. Vậy nó là độ dài đường đi tối ưu.

• Trong trường hợp hàm đánh giá g(u) = 0 với mọi u, thuật toán A\* chính là thuật toán tìm kiếm tốt nhất đầu tiên với hàm đánh giá g(u) mà ta đã nói đến.

•Thuật toán A\* đã được chứng tỏ là thuật toán hiệu quả nhất trong số các thuật toán đầy đủ và tối ưu cho vấn đề tìm kiếm đường đi ngắn nhất.



Quá trình tạo ra các bước chạy của thuật toán A\* để giải bài toán 8-Puzzle như sau:

1. Khởi tạo hàng đợi ưu tiên (priority queue) với trạng thái ban đầu, đặt giá trị ưu tiên ban đầu là 0 (f = g + h, trong đó g là chi phí đã di chuyển từ trạng thái ban đầu đến trạng thái hiện tại, h là giá trị heuristic từ trạng thái hiện tại đến trạng thái đích).

2. Lặp cho đến khi hàng đợi trống:

a. Lấy trạng thái có giá trị f nhỏ nhất từ hàng đợi.

b. Nếu trạng thái này là trạng thái đích, quá trình tìm kiếm kết thúc.

c. Nếu không, mở rộng trạng thái hiện tại bằng cách tạo các trạng thái con có thể đạt được thông qua các bước di chuyển hợp lệ.

d. Đối với mỗi trạng thái con:

i. Tính giá trị g (chi phí đã di chuyển) và h (ước tính khoảng cách) cho trạng thái con.

ii. Nếu trạng thái con chưa có trong hàng đợi hoặc có giá trị f nhỏ hơn trạng thái đã có, thêm trạng thái con vào hàng đợi với giá trị ưu tiên là f.

Quá trình này sẽ tiếp tục cho đến khi trạng thái đích được tìm thấy hoặc hàng đợi trở thành trống, tại đó chúng ta biết không có giải pháp.

Vì mỗi bước di chuyển trong 8-Puzzle tạo ra nhiều trạng thái con, việc tạo ra chuỗi bước cụ thể dựa trên kết quả của A\* là công việc phức tạp và không thể thể hiện trong một phần trả lời ngắn. Tuy nhiên, thuật toán A\* đảm bảo tìm ra giải pháp tối ưu hoặc gần tối ưu để đạt được trạng thái đích từ trạng thái ban đầu trong trò chơi 8-Puzzle.

## 4. Công dụng của thuật toán:

Thuật toán A\* (A-star) là một thuật toán tìm kiếm đường đi trong đồ thị hoặc lưới dựa trên độ ưu tiên. Nó được sử dụng rộng rãi trong lĩnh vực trí tuệ nhân tạo và trò chơi điện tử, đặc biệt trong các bài toán tìm đường đi từ điểm xuất phát đến điểm đích.

Dưới đây là một số công dụng của thuật toán A\*:

* 1. Tìm đường đi tối ưu: Thuật toán A\* được sử dụng để tìm đường đi tối ưu giữa hai điểm trong một đồ thị hoặc lưới. Nó sử dụng một hàm heuristics để ước lượng chi phí còn lại từ điểm hiện tại đến điểm đích và sử dụng thông tin này để tìm kiếm đường đi ngắn nhất.
  2. Trò chơi điện tử: A\* thường được sử dụng để tìm đường đi cho các nhân vật hoặc đối tượng trong trò chơi điện tử. Ví dụ, trong trò chơi chiến thuật, nhân vật cần tìm đường đi tốt nhất để đến được vị trí mục tiêu hoặc tránh các chướng ngại vật. A\* giúp tính toán đường đi thông qua các ô hoặc các điểm trong lưới trò chơi một cách hiệu quả và tối ưu.
  3. Robot di chuyển: A\* có thể được sử dụng để lập kế hoạch và điều khiển di chuyển của robot trong các ứng dụng thực tế như robot hút bụi, robot giao hàng hoặc robot tự động hóa công nghiệp. Thuật toán A\* giúp robot tìm kiếm đường đi an toàn và hiệu quả qua các vị trí trong môi trường.
  4. Lập kế hoạch đường đi trong hệ thống điều hướng: A\* có thể được sử dụng để lập kế hoạch đường đi trong hệ thống điều hướng tự động như hệ thống định vị toàn cầu (GPS) hoặc hệ thống điều hướng trên ô tô. Thuật toán giúp tính toán đường đi tối ưu và nhanh chóng từ điểm ban đầu đến đích, đồng thời xem xét các ràng buộc như tốc độ, hạn chế đường phố, và các yếu tố khác.
  5. Lập lịch và tối ưu hóa: A\* có thể được sử dụng trong các bài toán lập lịch và tối ưu hóa, ví dụ như lập lịch công việc, lập lịch sản xuất hoặc lập lịch giao hàng. Thuật toán giúp tìm kiếm lựa chọn tốt nhất và tối ưu trong không gian tìm kiếm lớn.

Những công dụng trên chỉ là một số ví dụ phổ biến của thuật toán A\*. Thuật toán này được áp dụng trong nhiều lĩnh vực khác nhau và có thể được điều chỉnh

## 4.4.Ưu điểm

Một thuật giải linh động, tổng quát, trong đó hàm chứa cả [tìm kiếm chiều sâu](https://www.iostream.vn/article/thuat-toan-depth-first-search-APnHi1), [tìm kiếm chiều rộng](https://www.iostream.vn/article/thuat-toan-breadth-first-search-sBPnH) và những nguyên lý Heuristic khác. Nhanh chóng tìm đến lời giải với sự định hướng của hàm Heuristic. Chính vì thế mà người ta thường nói A\* chính là thuật giải tiêu biểu cho Heuristic.

## 4.5.Nhược điểm

A\* rất linh động nhưng vẫn gặp một khuyết điểm cơ bản - giống như chiến lược tìm kiếm chiều rộng - đó là tốn khá nhiều bộ nhớ để lưu lại những trạng thái đã đi qua.

# **5.So sánh với thuật toán Dijkstra**

Thuật toán A\* và thuật toán Dijkstra đều là các thuật toán tìm kiếm đường đi ngắn nhất trong đồ thị. Cả hai thuật toán đều bắt đầu từ nút gốc và khám phá các nút kề của nút hiện tại.

Tuy nhiên, có một số điểm khác biệt chính giữa hai thuật toán này:

* *Thuật toán A sử dụng một hàm ước lượng để đánh giá độ tốt của mỗi nút.*\* Hàm ước lượng cung cấp một ước tính cho chi phí của đường đi từ nút hiện tại đến nút đích. Điều này giúp thuật toán A\* tìm ra đường đi ngắn nhất nhanh hơn thuật toán Dijkstra.
* *Thuật toán A chỉ xem xét các nút có độ tốt ước lượng thấp nhất.*\* Điều này giúp thuật toán A\* tránh khám phá các nút không cần thiết, dẫn đến hiệu quả cao hơn.

Dưới đây là một bảng tóm tắt các điểm khác biệt chính giữa hai thuật toán:

|  |  |  |
| --- | --- | --- |
| **Đặc điểm** | **Thuật toán A\*** | **Thuật toán Dijkstra** |
| Sử dụng hàm ước lượng | Có | Không |
| Xem xét các nút có độ tốt ước lượng thấp nhất | Có | Không |
| Hiệu quả | Cao | Không |
| Thời gian chạy | Tỷ lệ thuận với log của số lượng đỉnh | Tỷ lệ thuận với số lượng đỉnh |

Do sử dụng hàm ước lượng, thuật toán A\* có thể tìm ra đường đi ngắn nhất trong thời gian ngắn hơn thuật toán Dijkstra. Tuy nhiên, thuật toán A\* cũng yêu cầu nhiều bộ nhớ hơn để lưu trữ các nút trong hàng đợi mở.

Trong thực tế, thuật toán A\* thường được sử dụng để tìm đường đi trong các đồ thị lớn, nơi hiệu quả là yếu tố quan trọng. Thuật toán Dijkstra thường được sử dụng trong các đồ thị nhỏ, nơi bộ nhớ không phải là vấn đề đáng lo ngại.

# **Mối quan hệ thuật toán a\* với các thuật toán khác**

Dưới đây là mối quan hệ của thuật toán A\* với một số thuật toán khác:

* Tìm kiếm theo chiều rộng (Breadth-First Search, BFS):\*\* BFS tìm kiếm bằng cách mở rộng tất cả các trạng thái con trong lớp hiện tại trước khi di chuyển đến lớp kế tiếp. A\* thực hiện tìm kiếm tương tự, nhưng sử dụng hàm heuristic để ước tính khoảng cách còn lại từ mỗi trạng thái đến đích, giúp tìm kiếm hướng tới mục tiêu.
* Tìm kiếm theo chi phí thực tế (Uniform-Cost Search, UCS):\*\* UCS tìm kiếm đường đi ngắn nhất bằng cách ưu tiên mở rộng các trạng thái có chi phí thấp hơn. A\* cũng tìm kiếm đường đi ngắn nhất, nhưng nó sử dụng hàm heuristic để ước tính chi phí còn lại, giúp nó tập trung vào những trạng thái có khả năng tốt hơn để đạt đến đích.
* Tìm kiếm Greedy Best-First:\*\* Thuật toán này ưu tiên mở rộng các trạng thái có ước tính heuristic thấp nhất, không quan tâm đến chi phí thực tế để đến được trạng thái đó. A\* cũng sử dụng ước tính heuristic, nhưng nó kết hợp cả chi phí thực tế, từ đó có khả năng tìm ra đường đi tốt hơn.
* Thuật toán Dijkstra:\*\* Dijkstra tìm kiếm đường đi ngắn nhất bằng cách mở rộng các trạng thái theo thứ tự tăng dần của chi phí. A\* sử dụng hàm heuristic để ước tính chi phí còn lại, giúp nó tìm kiếm hiệu quả hơn trong nhiều trường hợp.
* Tìm kiếm động (Dynamic Programming):\*\* Dynamic Programming giải quyết các bài toán tối ưu hóa bằng cách phân chia chúng thành các bài toán con nhỏ hơn và lưu trữ các kết quả để tránh tính toán lại. A\* không phải là thuật toán Dynamic Programming, nhưng cũng sử dụng một loại quan hệ tương tự khi nó lưu trữ thông tin về các trạng thái đã được duyệt qua và tối ưu hóa việc tìm kiếm.

Như vậy, A\* là một thuật toán kết hợp giữa các yếu tố tìm kiếm thông tin và tìm kiếm theo chi phí, giúp nó đạt được hiệu suất tốt trong việc tìm kiếm đường đi ngắn nhất.

# **7.Ứng dụng thực tế**

**1. Trò chơi và đồ họa máy tính:**

- Trong trò chơi máy tính, A\* được sử dụng để tìm đường đi cho nhân vật, quái vật, hoặc các đối tượng di chuyển trong môi trường 2D hoặc 3D.

- Trong đồ họa máy tính, A\* có thể được sử dụng để tạo ra hiệu ứng đổ bóng, tạo đường đi cho các particle trong một môi trường phức tạp.

**2 .Robot và tự động hóa:**

- Trong robot di động, A\* có thể giúp xác định đường đi an toàn và hiệu quả cho robot trong môi trường đa dạng.

- Trong tự động hóa, A\* có thể được sử dụng để lập kế hoạch cho robot công nghiệp hoặc máy móc di chuyển.

**3. Tự định vị và hệ thống thông tin địa lý (GIS):**

- A\* có thể được sử dụng để tìm đường đi ngắn nhất trên bản đồ địa lý, hỗ trợ cho việc tự định vị GPS và lập kế hoạch tuyến đường.

**4.Trình tạo bản đồ và khám phá:**

- Trong trình tạo bản đồ, A\* có thể giúp tạo ra bản đồ chi tiết bằng cách xác định các đường đi qua các vùng không xác định.

- Trong robot khám phá, A\* có thể giúp robot tìm kiếm và khám phá môi trường một cách hiệu quả.

**5. Ứng dụng trong truy vết và tìm kiếm:**

- A\* có thể được sử dụng để truy vết các tín hiệu (ví dụ: sóng sót, tàu biển) để xác định vị trí và tìm kiếm.

- Trong tình huống cứu hộ, A\* có thể giúp lập kế hoạch tìm kiếm vùng lớn một cách hiệu quả.

**6.Lập lịch và lập kế hoạch trong robotica:**

- A\* có thể được sử dụng để lập kế hoạch di chuyển và đường đi cho các robot tự động thực hiện các nhiệm vụ như thu thập dữ liệu hoặc giao hàng.

**7. Giải quyết bài toán tối ưu:**

- Ngoài việc tìm kiếm đường đi, A\* còn có thể được áp dụng để giải quyết các bài toán tối ưu khác như lập lịch công việc, lập kế hoạch sản xuất, và quản lý tài nguyên.

Đây chỉ là một số ví dụ về các lĩnh vực ứng dụng của thuật toán A\*. Do tính linh hoạt và hiệu quả của nó, A\* đã trở thành một công cụ quan trọng trong nhiều lĩnh vực khác nhau trong khoa học máy tính và kỹ thuật ứng dụng.

# **8.Đánh giá và kết luận**

Thuật toán A\* là một thuật toán tìm kiếm phổ biến được sử dụng trong nhiều ứng dụng, bao gồm trò chơi điện tử, lập trình tự động và trí tuệ nhân tạo dựa trên chi phí, tìm kiếm một đường đi từ một trạng thái ban đầu đến một trạng thái mục tiêu với chi phí thấp nhất. Thuật toán này hoạt động bằng cách xây dựng một cây tìm kiếm, với các nút là các trạng thái và các cạnh là các bước có thể được thực hiện từ một trạng thái sang trạng thái khác. Đường đi được tìm thấy bằng cách bắt đầu từ nút gốc và khám phá các nút có chi phí thấp nhất.

Nhìn chung, thuật toán A\* là một thuật toán tìm kiếm hiệu quả có thể được sử dụng để giải quyết nhiều vấn đề khác nhau. Tuy nhiên, điều quan trọng là phải hiểu những ưu điểm và nhược điểm của thuật toán này để có thể sử dụng nó một cách hiệu quả.

Kết luận của thuật toán A\* là nó là một thuật toán tìm kiếm hiệu quả có thể tìm thấy đường đi tối ưu trong nhiều trường hợp. Tuy nhiên, thuật toán này không phải lúc nào cũng hiệu quả, đặc biệt là trong các vấn đề có kích thước lớn hoặc có nhiều trạng thái có chi phí cao.

# **9. Đánh giá và nhận xét thành viên:**

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Mã Số Sinh Viên | Họ và Tên | Đánh giá |
| 1 | 2001210927 | Ngô Thị Thùy Linh | 9/10 |
| 2 | 2001210779 | Nguyễn Ngọc Quân | 9/10 |
| 3 | 2001210235 | Phùng Huỳnh Thanh Ngân | 9/10 |

# **10.Nguồn tham khảo**

[https://vi.wikipedia.org/wiki/Giai\_thuat\_tim\_kiem\_A\*](https://vi.wikipedia.org/wiki/Giai_thuat_tim_kiem_A*)

<https://www.iostream.vn/article/thuat-giai-a-DVnHj>

<https://voer.edu.vn/m/giai-thuat-tim-kiem-a/d169b9dd>