**ĐẠI HỌC QUỐC GIA HÀ NỘI**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

**~~~~000~~~~~**



**CHUYÊN ĐỀ**

**Informed Search Methods**

|  |  |  |
| --- | --- | --- |
| Sinh viên thực hiện | : | Nguyễn Minh Hiển ,  Dương Tự Trí Đức ,  Kiều Bá Đăng,  Đoàn Trần Quang Đạo. |
| Mã sinh viên | : | 21021299 , 21021289  21021282 , 21020583 |
| Môn học | : | Trí tuệ nhân tạo |
| Mã học phần | : | * [2324II\_INT3401\_20](https://courses.uet.vnu.edu.vn/course/view.php?id=10948) |
| Giáo viên hướng dẫn | : | Trần Hồng Việt |

**Hà Nội**

**MỤC LỤC**

[**I. Giới thiệu**](https://docs.google.com/document/d/1oJUr2-j7yyPdaefEA07W_otI6aC-gJeYECKCxMIFsmM/edit#heading=h.ftd6kqud8vec) **3**

[1. Định nghĩa Tìm kiếm heuristic](https://docs.google.com/document/d/1oJUr2-j7yyPdaefEA07W_otI6aC-gJeYECKCxMIFsmM/edit#heading=h.epa877lx0o2j) 3

[2. Vai trò của Tìm kiếm heuristic trong Trí tuệ nhân tạo](https://docs.google.com/document/d/1oJUr2-j7yyPdaefEA07W_otI6aC-gJeYECKCxMIFsmM/edit#heading=h.bqjql5b1pp19) 3

[3. Các ứng dụng thực tế của Tìm kiếm heuristic](https://docs.google.com/document/d/1oJUr2-j7yyPdaefEA07W_otI6aC-gJeYECKCxMIFsmM/edit#heading=h.gbicgns9sutg) 3

[**II. Hàm heuristic**](https://docs.google.com/document/d/1oJUr2-j7yyPdaefEA07W_otI6aC-gJeYECKCxMIFsmM/edit#heading=h.nb3sh1n55ojy) **4**

[1. Thế nào là hàm đánh giá Heuristic?](https://docs.google.com/document/d/1oJUr2-j7yyPdaefEA07W_otI6aC-gJeYECKCxMIFsmM/edit#heading=h.f83j472ltrak) 4

[2. Lấy ví dụ](https://docs.google.com/document/d/1oJUr2-j7yyPdaefEA07W_otI6aC-gJeYECKCxMIFsmM/edit#heading=h.pinju9jjesa) 4

[**III. Các giải thuật tìm kiếm heuristic**](https://docs.google.com/document/d/1oJUr2-j7yyPdaefEA07W_otI6aC-gJeYECKCxMIFsmM/edit#heading=h.m69t9ki35z11) **5**

[1. Giới thiệu về giải thuật tìm kiếm Heuristic](https://docs.google.com/document/d/1oJUr2-j7yyPdaefEA07W_otI6aC-gJeYECKCxMIFsmM/edit#heading=h.icsr7sf2y7bk) 5

[2. So sánh tìm kiếm kinh nghiệm và tìm kiếm mù](https://docs.google.com/document/d/1oJUr2-j7yyPdaefEA07W_otI6aC-gJeYECKCxMIFsmM/edit#heading=h.4yvynm4bgjyw) 5

[3. Các phương pháp tìm kiếm kinh nghiệm](https://docs.google.com/document/d/1oJUr2-j7yyPdaefEA07W_otI6aC-gJeYECKCxMIFsmM/edit#heading=h.lpscjjoxfqv1) 6

[a. Greedy Best-first search](https://docs.google.com/document/d/1oJUr2-j7yyPdaefEA07W_otI6aC-gJeYECKCxMIFsmM/edit#heading=h.ak1xga8w09c) 6

[b. Beam search](https://docs.google.com/document/d/1oJUr2-j7yyPdaefEA07W_otI6aC-gJeYECKCxMIFsmM/edit#heading=h.ptpfcnyclcv) 7

[c. Hill-climbing Search 8](https://docs.google.com/document/d/1oJUr2-j7yyPdaefEA07W_otI6aC-gJeYECKCxMIFsmM/edit#heading=h.ruum54vznugm)

[d. A\* search](https://docs.google.com/document/d/1oJUr2-j7yyPdaefEA07W_otI6aC-gJeYECKCxMIFsmM/edit#heading=h.svsmpz1kgphf) 9

[**IV. Áp dụng thuật toán A\* cho bài toán 8 puzzle**](https://docs.google.com/document/d/1oJUr2-j7yyPdaefEA07W_otI6aC-gJeYECKCxMIFsmM/edit#heading=h.s9m1bj9tj192) **11**

[1. Giới thiệu về bài toán 8 số.](https://docs.google.com/document/d/1oJUr2-j7yyPdaefEA07W_otI6aC-gJeYECKCxMIFsmM/edit#heading=h.dj74xagyxb3q) 11

[2. Điều kiện của trạng thái đầu.](https://docs.google.com/document/d/1oJUr2-j7yyPdaefEA07W_otI6aC-gJeYECKCxMIFsmM/edit#heading=h.ddtpcx9pe1wh) 11

[3. Áp dụng khoảng cách manhattan để tính hàm ước lượng heuristic h(n).](https://docs.google.com/document/d/1oJUr2-j7yyPdaefEA07W_otI6aC-gJeYECKCxMIFsmM/edit#heading=h.nam9vfy4jsle) 12

[4. Giải thuật của bài toán. 1](https://docs.google.com/document/d/1oJUr2-j7yyPdaefEA07W_otI6aC-gJeYECKCxMIFsmM/edit#heading=h.5p1e8n7gun4v)3

[5. Lập trình về bài toán 8 puzzle. 1](https://docs.google.com/document/d/1oJUr2-j7yyPdaefEA07W_otI6aC-gJeYECKCxMIFsmM/edit#heading=h.5t9x3gstz6ay)4

[6. Test case 1](https://docs.google.com/document/d/1oJUr2-j7yyPdaefEA07W_otI6aC-gJeYECKCxMIFsmM/edit#heading=h.zfbie8saqlns)6

[**V. Kết luận. 1**](https://docs.google.com/document/d/1oJUr2-j7yyPdaefEA07W_otI6aC-gJeYECKCxMIFsmM/edit#heading=h.hka4md5iaf2s)**7**

**VI. Tài liệu tham khảo 17**

# **Giới thiệu.**

## **Định nghĩa Tìm kiếm heuristic.**

Tìm kiếm Heuristic (hay còn gọi là Tìm kiếm kinh nghiệm) là một kỹ thuật giải quyết vấn đề dựa trên kinh nghiệm và quy tắc chung để đưa ra giải pháp mà không đảm bảo tối ưu. Nói cách khác, đây là phương pháp sử dụng các quy tắc đơn giản và trực quan để thu hẹp không gian tìm kiếm và nhanh chóng tìm đến giải pháp thỏa mãn yêu cầu.

## **Vai trò của Tìm kiếm heuristic trong Trí tuệ nhân tạo.**

Tìm kiếm Heuristic đóng vai trò quan trọng trong lĩnh vực trí tuệ nhân tạo bởi một số ưu điểm:

* Giúp giải quyết các vấn đề phức tạp bằng cách thu hẹp không gian tìm kiếm và nhanh chóng tìm đến giải pháp thỏa mãn yêu cầu.
* Tăng tốc độ tìm kiếm trong xử lý các bài toán thời gian thực.
* Khả năng thích ứng cao do có thể được điều chỉnh linh hoạt cho phù hợp với những bài toán cụ thể, tận dụng các kiến thức chuyên môn và kinh nghiệm trong các lĩnh vực đó.
* Heuristic còn có thể kết hợp với các phương pháp tìm kiếm tối ưu khác để tạo ra các thuật toán lai.

## **Các ứng dụng thực tế của Tìm kiếm heuristic.**

Tìm kiếm có thông tin được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau, bao gồm:

* **Trò chơi:** Tìm kiếm Heuristic được sử dụng để tìm kiếm các nước đi tốt nhất trong các trò chơi như cờ vua, cờ tướng, v.v.
* **Robot học:** Tìm kiếm Heuristic được sử dụng để lập kế hoạch cho robot di chuyển trong môi trường.
* **Lập kế hoạch:** Tìm kiếm Heuristic được sử dụng để lập kế hoạch cho các hoạt động như lập kế hoạch sản xuất, lập kế hoạch giao hàng, v.v.
* **Chẩn đoán y tế:** Tìm kiếm Heuristic được sử dụng để chẩn đoán bệnh dựa trên các triệu chứng của bệnh nhân.
* **Tìm kiếm thông tin:** Tìm kiếm Heuristic được sử dụng để cải thiện hiệu quả của các công cụ tìm kiếm web.

# **Hàm heuristic**

## **Thế nào là hàm đánh giá Heuristic?**

* Hàm đánh giá Heuristic là một phần quan trọng trong tìm kiếm Heuristic, đóng vai trò ước lượng mức độ gần gũi của một trạng thái nào đó với trạng thái đích. Có thể nói rằng hàm Heuristic cung cấp cho thuật toán tìm kiếm một giá trị để đánh giá về tiềm năng của mỗi trạng thái trong việc dẫn đến trạng thái đích. Dựa trên giá trị được đánh giá, thuật toán tìm kiếm sẽ ưu tiên khai thác các trạng thái có tiềm năng cao hơn, giúp thu hẹp không gian tìm kiếm và nhanh chóng tìm đến giải pháp thỏa mãn yêu cầu.

## **Lấy ví dụ**

Ví dụ trong trò chơi cờ vua, Hàm Heuristic được sử dụng để đánh giá mức độ tiềm năng của một vị trí cờ, giúp thuật toán tìm kiếm ưu tiên khám phá những vị trí có khả năng dẫn đến chiến thắng cao hơn.

* Số lượng quân cờ: Hàm heuristic đơn giản nhất là dựa vào số lượng quân cờ mỗi bên. Bên nào còn nhiều quân cờ hơn sẽ được đánh giá cao hơn. Ví dụ : gán giá trị 1 cho mỗi quân tốt, 3 cho Mã và tượng, 5 cho mỗi quân xe và Hậu là 10.
* Vị trí quân cờ:

Quân cờ ở trung tâm bàn cờ thường được đánh giá cao hơn so với quân ở biên

Quân cờ có khả năng di chuyển nhiều ô hơn cũng được đánh giá cao hơn

Quân cờ có thể tấn công nhiều quân cờ khác cũng được đánh giá cao hơn so với quân chỉ tấn công được 1 quân.

* Để thể hiện yếu tố vị trí, mỗi ô trên bàn cờ được gán cho một giá trị nhất định, cao hơn khi ở trung tâm và giảm khi gần biên.
* Mức độ an toàn của Vua:

Hàm Heuristic có thể sử dụng các yếu tố như số lượng quân cờ có thể bảo vệ vua, số lượng ô mà Vua có thể di chuyển đến một cách an toàn, và số lượng quân cờ đối phương có thể tấn công vua để đánh giá mức độ an toàn.

* Các ví dụ trên chỉ là một số hàm heuristic cơ bản được sử dụng trong cờ vua. Trên thực tế có rất nhiều hàm Heuristic khác nhau và mỗi hàm có ưu và nhược điểm riêng.

# **Các giải thuật tìm kiếm heuristic.**

## **Giới thiệu về giải thuật tìm kiếm Heuristic.**

* Phương pháp tìm kiếm mù có khả năng thám hiểm không gian trạng thái để tìm ra trạng thái đích. Tuy nhiên phương pháp này tốn nhiều thời gian và kém hiệu quả với hầu hết các bài toán.
* Từ đây khi áp dụng hàm đánh giá ta sẽ cung cấp tri thức và hướng dẫn tìm kiếm, giúp phương pháp tìm kiếm kinh nghiệm đạt hiệu quả cao và cải thiện tốc độ tìm kiếm.
* Các phương pháp tìm kiếm kinh nghiệm có thể kể tên như:
  + Greedy Best-first search
  + Beam Search
  + Hill-climbing Search
  + A\* search

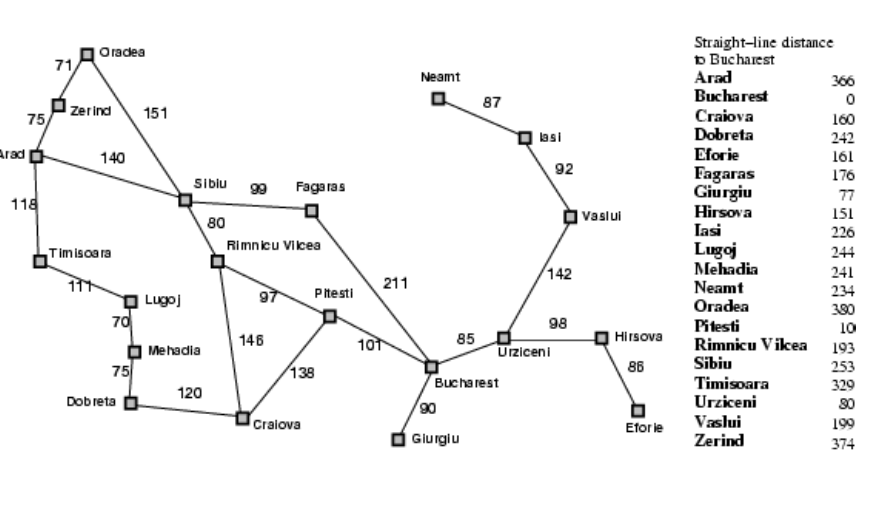
## **So sánh tìm kiếm kinh nghiệm và tìm kiếm mù.**

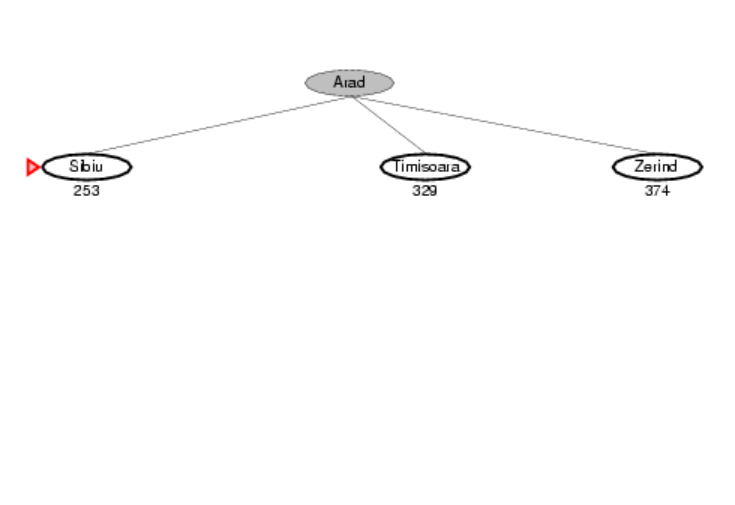
|  |  |
| --- | --- |
| **Tìm kiếm kinh nghiệm** | **Tìm kiếm mù** |
| Sử dụng hàm đánh giá làm tri thức, kinh nghiệm | Không sử dụng hàm đánh giá |
| Phương pháp tìm kiếm kinh nghiệm hiệu quả hơn do dựa vào hàm heuristic, tránh các đường xấu và chọn những con đường tốt hơn | Phương pháp tìm kiếm mù kém hiệu quả vì nó sẽ kiểm tra tất cả các nút con của một nút hoặc kiểm tra một đường sâu nhất có thể trước khi quay lại |
| Có đích đến. Phương pháp tìm kiếm kinh nghiệm được thiết kế để tìm ra giải pháp cho một vấn đề cụ thể. | Không xem xét chi phí để đạt được mục tiêu hoặc khả năng tìm ra giải pháp, dẫn đến quá trình tìm kiếm mù. |
| Phức tạp và đòi hỏi các yếu tố về hàm heuristic để đạt được kết quả tối ưu | Dễ hiểu và dễ triển khai |

## **Các phương pháp tìm kiếm kinh nghiệm.**

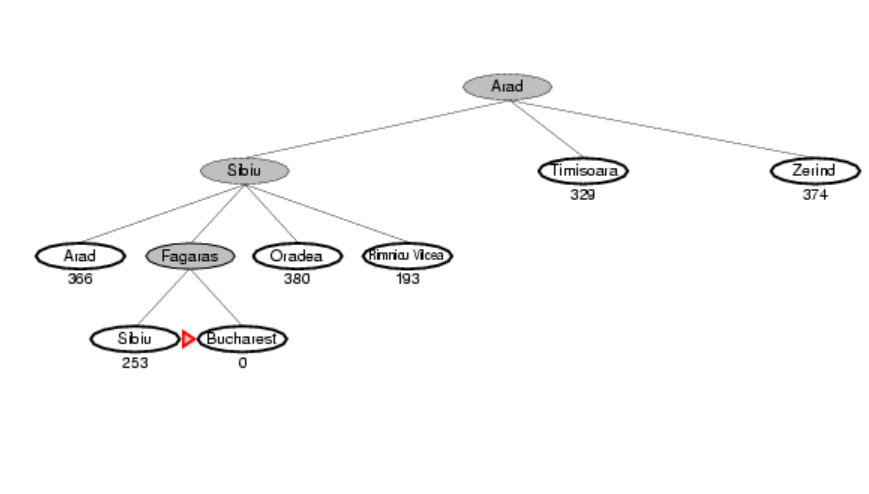
### **Greedy Best-first search.**

* Hàm đánh giá f(n) là hàm heuristic h(n).
* Hàm heuristic h(n) đánh giá chi phí để đi từ nút hiện tại n đến nút đích (mục tiêu).
* Ví dụ: Trong bài toán tìm đường đi từ Arad đến Bucharest, sử dụng: h(n) = Ước lượng khoảng cách đường thẳng từ thành phố hiện tại n đến Bucharest.
* Phương pháp tìm kiếm Greedy best-first search sẽ xét (phát triển) nút “có vẻ” gần với nút đích (mục tiêu) nhất.

*Hình 1: Bài toán tìm đường từ Arad đến Bucharest*



*Hình 2: Vòng lặp 1*

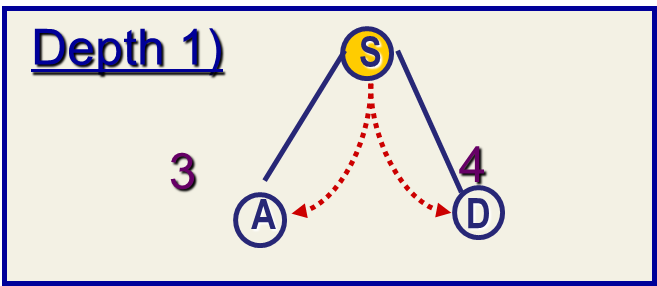
*Hình 3: Kết quả tìm kiếm*

* Phương pháp tìm kiếm có thể bị kẹt trong vòng Iasi → Neamt → Iasi → Neamt.
* Một hàm đánh giá tốt có thể giảm thời gian và không gian nhớ một cách đánh kể.

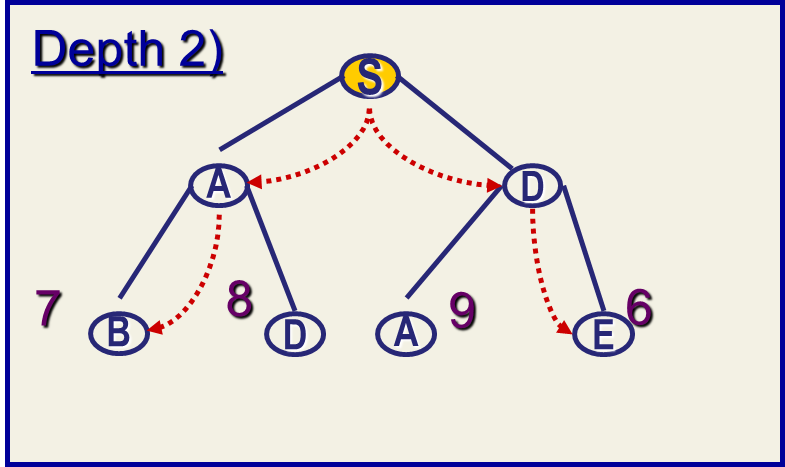
### **Beam search.**

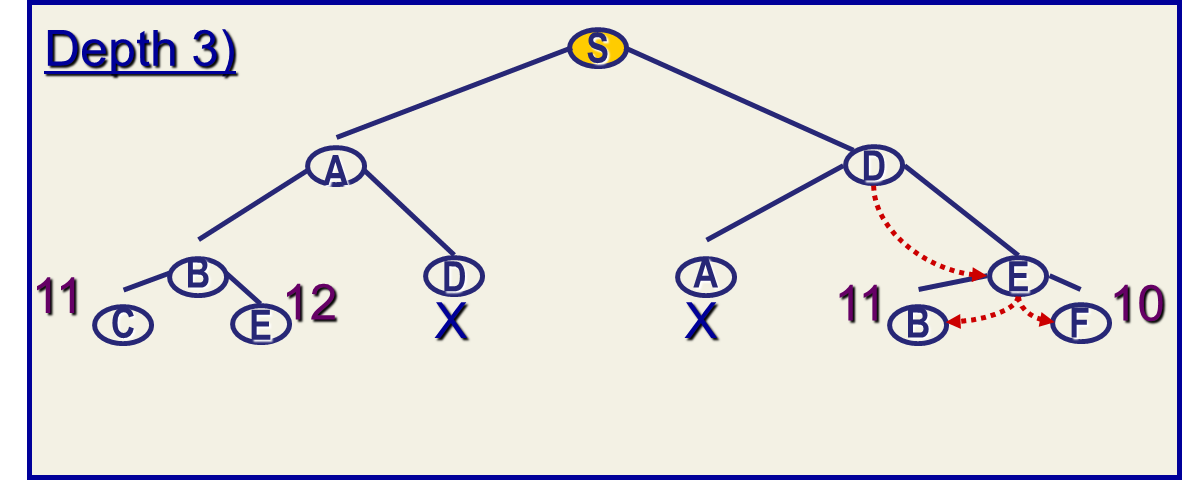
* Tương tự như tìm kiếm tốt nhất đầu tiên.
* Tuy nhiên chỉ phát triển k đỉnh ở mức tiếp theo chứ không phát triển toàn bộ.
* Ưu điểm: độ phức tạp tính toán tốt hơn.
* Nhược điểm: không tìm kiếm toàn bộ, nên có thể không tìm thấy đỉnh tối ưu nhất

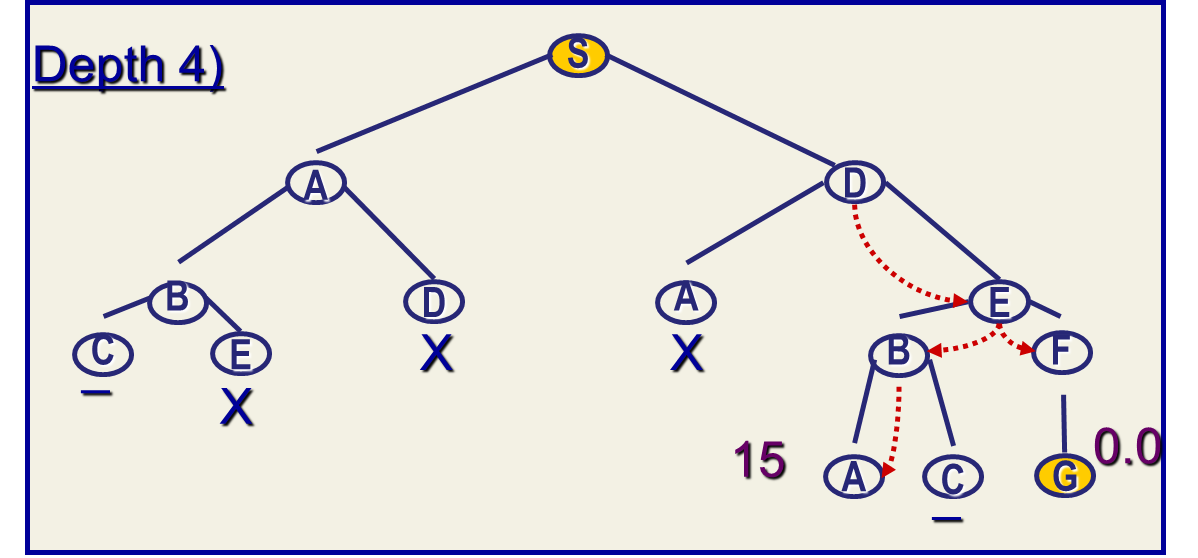
Ví dụ beam search với k = 2:



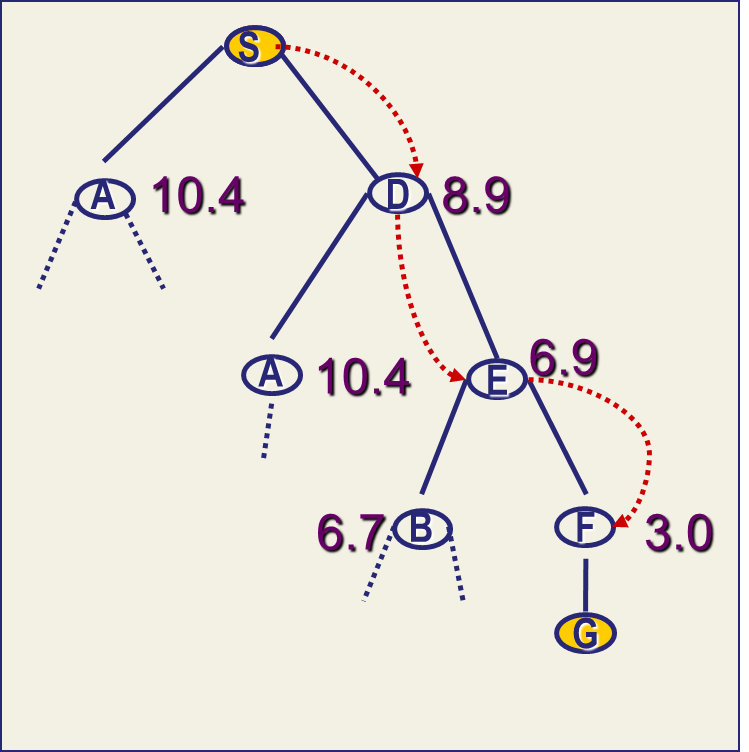
*Hình 4: Bước 1 beam search*

*Hình 5: Bước 2 beam search*

*Hình 6: Bước 3 beam search*

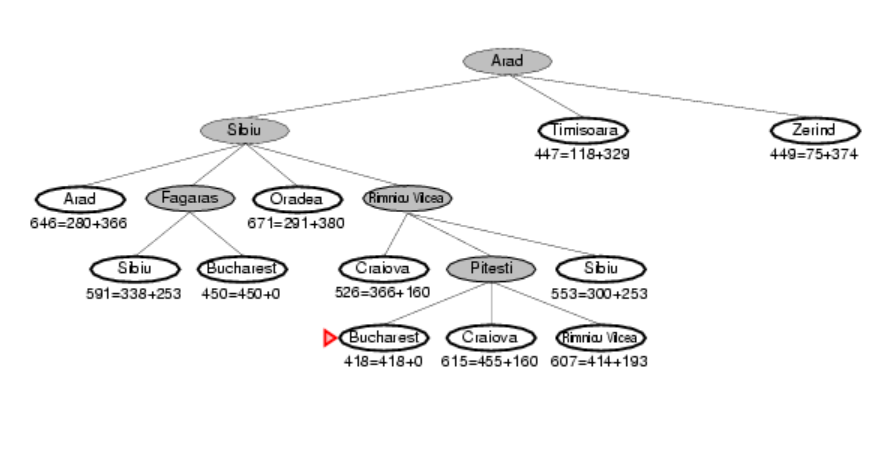
*Hình 7: Kết quả beam search*

### **Hill-climbing Search.**

* Tìm kiếm leo đồi = tìm kiếm theo độ sâu + hàm đánh giá.
* Khác với tìm kiếm theo độ sâu, ta phát triển một đỉnh u thì bước tiếp theo, ta chọn trong số các đỉnh con của u, đỉnh có nhiều hứa hẹn nhất để phát triển.
* Đỉnh hứa hẹn nhất là đỉnh có hàm đánh giá nhỏ nhất.

*Hình 8: Ví dụ về hill-climbing search*

### **A\* search**

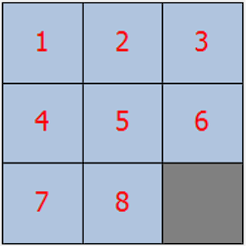
* Tương tự như greedy best-first search nhưng t sẽ thay đổi hàm heuristic để tránh các nhánh tìm kiếm đã xác định có chi phí cao.
* Sử dụng hàm đánh giá f(n) = g(n) + h(n).
* g(n) = chi phí từ nút gốc cho đến nút hiện tại n.
* h(n) = chi phí ước lượng từ nút hiện tại n tới đích.
* f(n) = chi phí tổng thể ước lượng của đường đi qua nút hiện tại n đến đích.
* 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, 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.
* Nếu không gian các trạng thái là vô hạn, thì giải thuật A\* là không hoàn chỉnh.

*Hình 9: Ví dụ về giải thuật A\**

# **Áp dụng thuật toán A\* cho bài toán 8 puzzle.**

## **Giới thiệu về bài toán 8 số.**

* Bài toán 8-puzzle (hay còn gọi là 8 số) là một bài toán trí tuệ cổ điển, quen thuộc với những người bắt đầu tiếp cận với môn Trí tuệ nhân tạo. Bài toán có nhiều phiên bản khác nhau dựa theo số ô, như 8-puzzle, 15-puzzle,…
* Bài toán gồm một bảng ô vuông kích thước 3x3, có tám ô được đánh số từ 1 tới 8 và một ô trống. Trạng thái ban đầu, các ô được sắp xếp một cách ngẫu nhiên, nhiệm vụ của người chơi là tìm cách đưa chúng về trạng thái mục tiêu đúng thứ tự như hình dưới:
* Trong quá trình giải bài toán, tại mỗi bước, ta giả định chỉ có ô trống là di chuyển. Như vậy, tối đa ô trống có thể có 4 khả năng di chuyển (lên trên, xuống dưới, sang trái, sang phải).

*Hình 10: Bài toán 8 số*

## **Điều kiện của trạng thái đầu.**

* Có những trạng thái của bảng số không thể chuyển về trạng thái đích. Người ta chứng minh được rằng, để có thể chuyển từ trạng thái đầu tới trạng thái đích, thì trạng thái đầu này phải thỏa mãn điều kiện được xác định như sau:
* Ta xét lần lượt từ trên xuống dưới, từ trái sang phải, với mỗi ô số đang xét (giả sử là ô thứ i), ta kiểm tra xem phía sau có bao nhiêu ô số có giá trị nhỏ hơn ô đó. Sau đó ta tính tổng N = n1 + n2 + … + n8.
* Ta có quy tắc chung sau cho bài toán n-puzzle:

Nếu số ô vuông lẻ:

N / 2 dư 0.

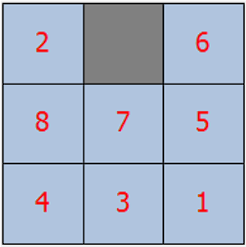
Nếu số ô vuông chẵn:

N / 2 dư 0 và ô trống phải nằm ở hàng chẵn xét từ trên xuống.

N / 2 dư 1 và ô trống phải nằm ở hàng lẻ xét từ trên xuống.

Cụ thể, ta đang xét bài toán 8-puzzle – có 9 ô vuông nên trạng thái đầu phải thỏa mãn điều kiện N / 2 dư 0.

Ví dụ: Cho trạng thái đầu sau 2-0-6-8-7-5-4-3-1:

*Hình 11: Trạng thái đầu của ô số*

Xét ô thứ nhất có giá trị 2: Phía sau có 1 ô nhỏ hơn (1) => n1 = 1

Xét ô thứ hai có giá trị 6: Phía sau có 4 ô nhỏ hơn (5,4,3,1) => n2 = 4

Xét ô thứ ba có giá trị 8: Phía sau có 5 ô nhỏ hơn (7,5,4,3,1) => n3 = 5

Xét ô thứ tư có giá trị 7: Phía sau có 4 ô nhỏ hơn (5,4,3,1) => n4 = 4

Xét ô thứ năm có giá trị 5: Phía sau có 3 ô nhỏ hơn (4,3,1) => n5 = 3

Xét ô thứ sáu có giá trị 4: Phía sau có 2 ô nhỏ hơn (3,1) => n6 = 2

Xét ô thứ bảy có giá trị 3: Phía sau có 1 ô nhỏ hơn (1) => n7 = 1

Xét ô thứ tám có giá trị 1: Phía sau không còn ô nào nhỏ hơn => n8 = 0

N = 1 + 4 + 5 + 4 + 3 + 2 + 1 + 0 = 20

Mà 20 / 2 dư 0 -> Thỏa mãn

* Những trạng thái của bảng số mà có thể chuyển về trạng thái đích gọi là cấu hình hợp lệ, ngược lại gọi là cấu hình không hợp lệ.
* Với bảng số kích thước mxm (m là cạnh) thì ta có không gian trạng thái là (mxm)!

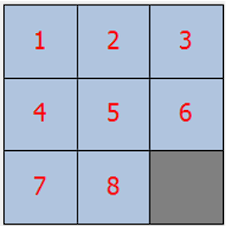
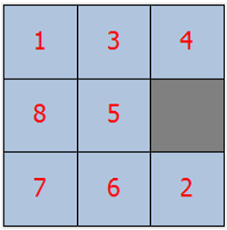
-> Với bài toán 8-puzzle, các trạng thái có thể có của bảng số là:

(3x3)! = 362880

## **Áp dụng khoảng cách manhattan để tính hàm ước lượng heuristic h(n).**

* Hàm heuristic được sử dụng trong thuật toán A\* ước tính chi phí đi từ trạng thái hiện tại đến trạng thái mục tiêu. Trong bài toán 8 Puzzle, một hàm heuristic phổ biến là khoảng cách Manhattan, được tính bằng tổng số bước di chuyển tối thiểu cần thiết để di chuyển mỗi ô số từ vị trí hiện tại đến vị trí mục tiêu.

Ví dụ: Có trạng thái bài toán sau:

*Hình 12: Trạng thái đầu* *Hình 13:Trạng thái đích*

Có 5 ô số nằm sai vị trí so với trạng thái đích

=> h1 = 5

Tính khoảng cách Manhattan, xác định tọa độ (dòng, cột) của ô khi ở sai vị trí và khi ở đúng vị trí:

Ô số 3 khi ở sai vị trí có tọa độ (0,1)

Ô số 3 khi ở đúng vị trí có tọa độ (0,2)

=> Khoảng cách Manhattan của ô số 3 bằng |0 – 0| + |2 – 1| = 1

Tương tự, ô số 4 có khoảng cách Manhattan là 3, ô số 8 có khoảng cách Manhattan là 2, ô số 6 có khoảng cách Manhattan là 2, ô số 2 có khoảng cách Manhattan là 3. Những ô số 1, 5, 7 đã ở đúng vị trí nên có khoảng cách Manhattan là 0.

= > h2 = 0 + 1 + 3 + 2 + 0 + 0 + 2 + 3 = 11

## **Giải thuật của bài toán.**



*Hình 15: Lưu đồ giải thuật A\**

* Để áp dụng thuật toán A\* vào bài toán 8 Puzzle, chúng ta cần biểu diễn bài toán dưới dạng đồ thị, với các nút là các trạng thái có thể xảy ra của bảng 8 Puzzle và các cạnh là các di chuyển hợp lệ giữa các trạng thái.
* Ban đầu ta có Open là tập chứa các trạng thái chưa được xét (sắp xếp theo f tăng dần), là một list chứa các trạng thái tiềm năng. Close là tập các trạng thái đã được xét. Ban đầu Open chỉ chứa trạng thái ban đầu, tập Close rỗng.
* Thuật toán:
* Lấy trạng thái có giá trị f(n) (tổng chi phí ước tính từ trạng thái ban đầu đến trạng thái n) thấp nhất từ danh sách mở.
* Di chuyển trạng thái này sang danh sách đóng.
* Mở rộng trạng thái này bằng cách xem xét tất cả các di chuyển hợp lệ.
* Đối với mỗi di chuyển, tính toán heuristic và chi phí di chuyển.
* Cập nhật giá trị f(n) và g(n) (chi phí di chuyển thực tế từ trạng thái ban đầu đến trạng thái n).
* Nếu đã tìm thấy trạng thái đích -> backtracking lại.

Begin

Open = {Start};

Close = {}

While do

Begin

X=Retrieve(Open)

Chọn X sao cho f(X) đạt là nhỏ nhất

If (X=Goal) then return True

Else

Begin

Sinh ra các trạng thái con của X;

For mỗi nút con Y của X do

If (Y  không thuộc Open) và (Y không thuộc Close)

Begin

Tính f(Y);

Open = Open + {Y};

End;

If (Y thuộc Open)

If (g(Y) < g(Y’)) -> Cập nhật giá trị của f(Y’), đặt parent của Y’ là X

If (Y thuộc Close)

If (g(Y) < g(Y’)) -> Cập nhật giá trị của f(Y’), đặt parent của Y’ là X, cập nhật lại giá trị f của tất cả các child của Y thuộc Open và Close

End;

Close = Close + {X};

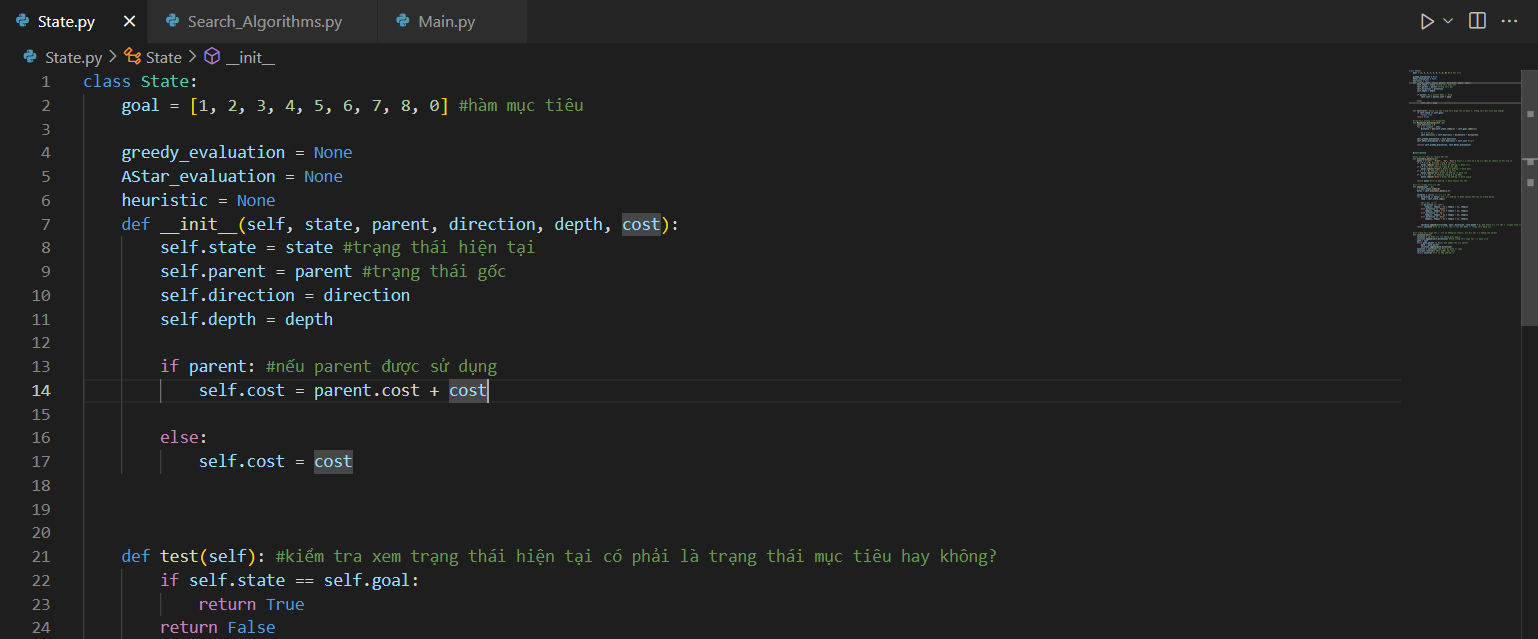
End;

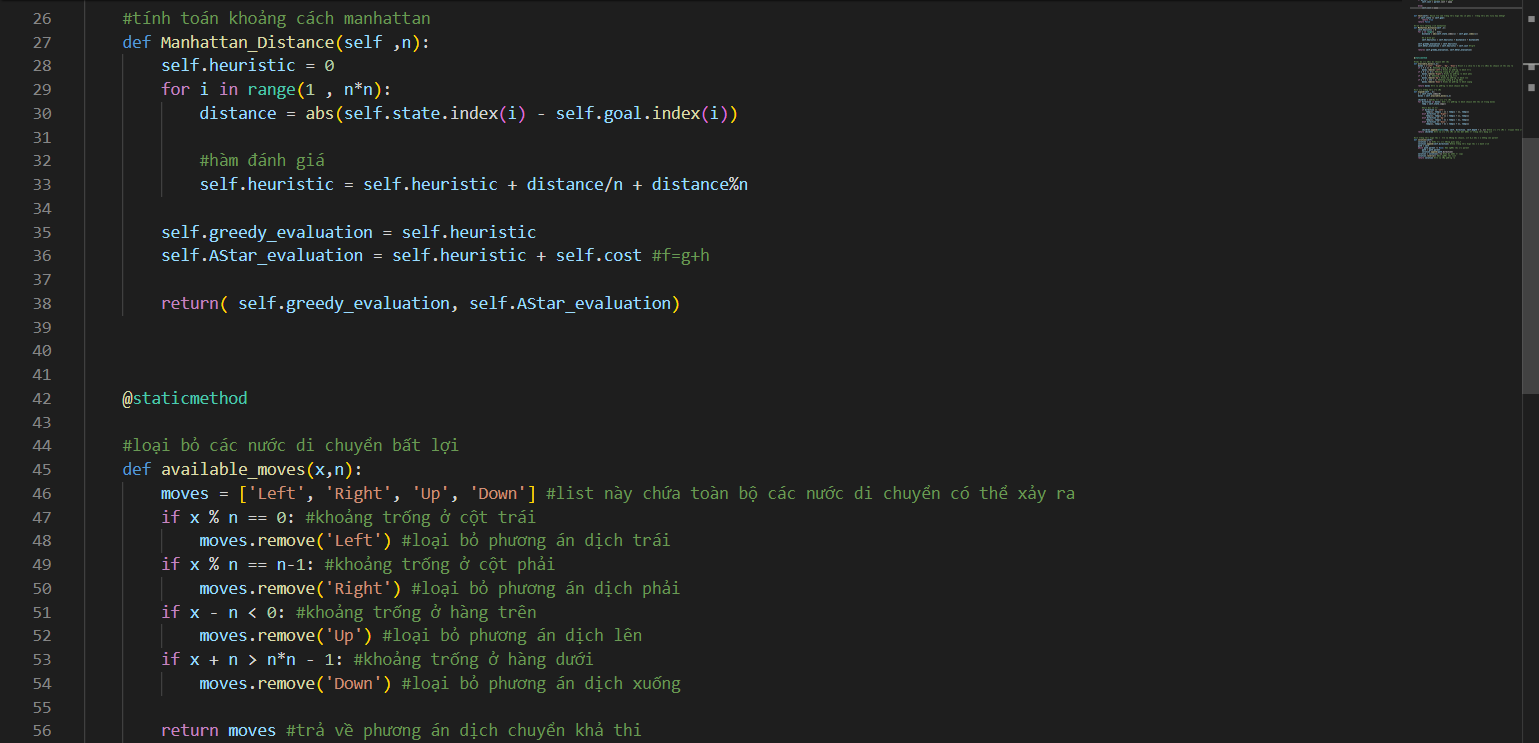
Return False;

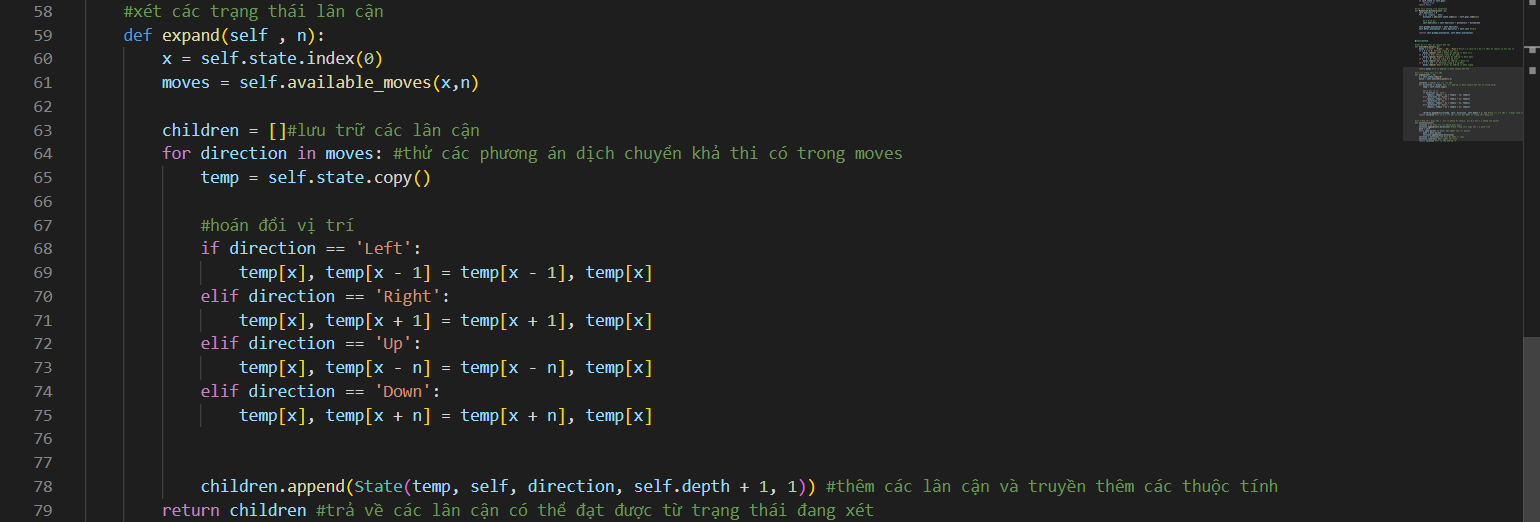
End.

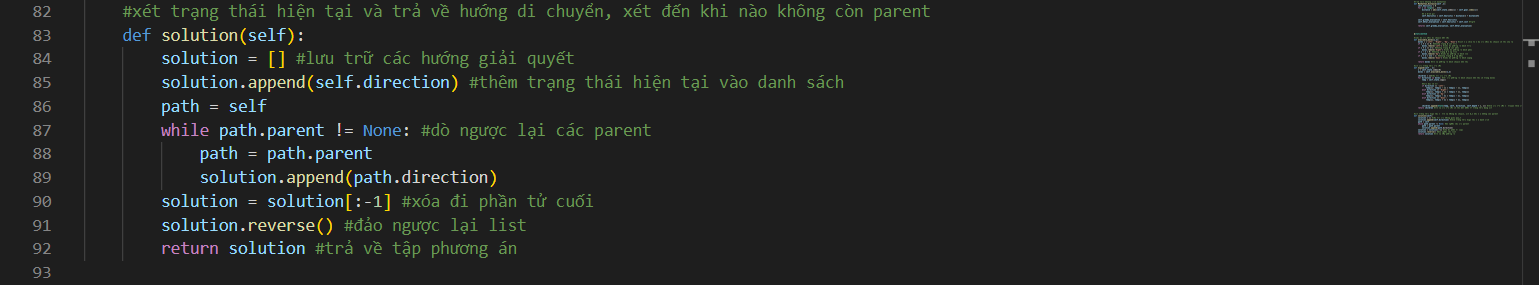
## **Lập trình về bài toán 8 puzzle.**

1. Trạng thái.

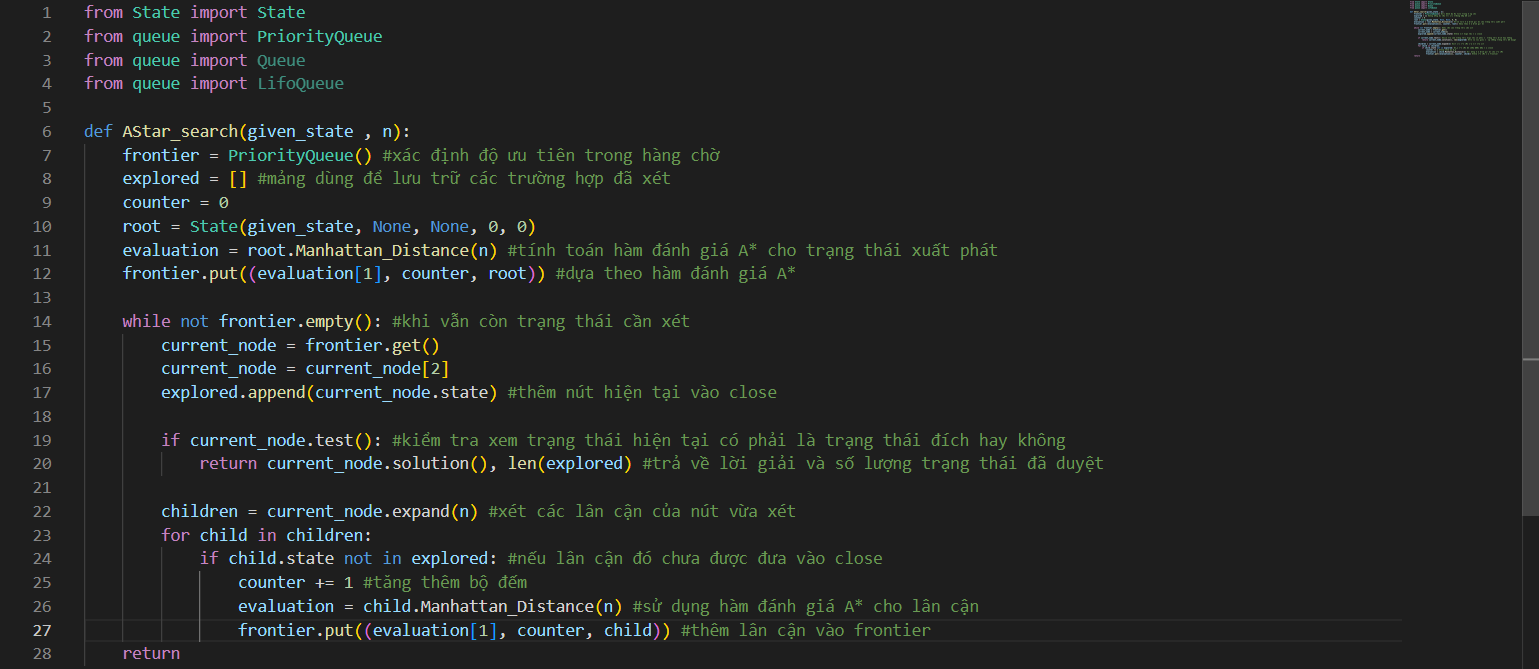




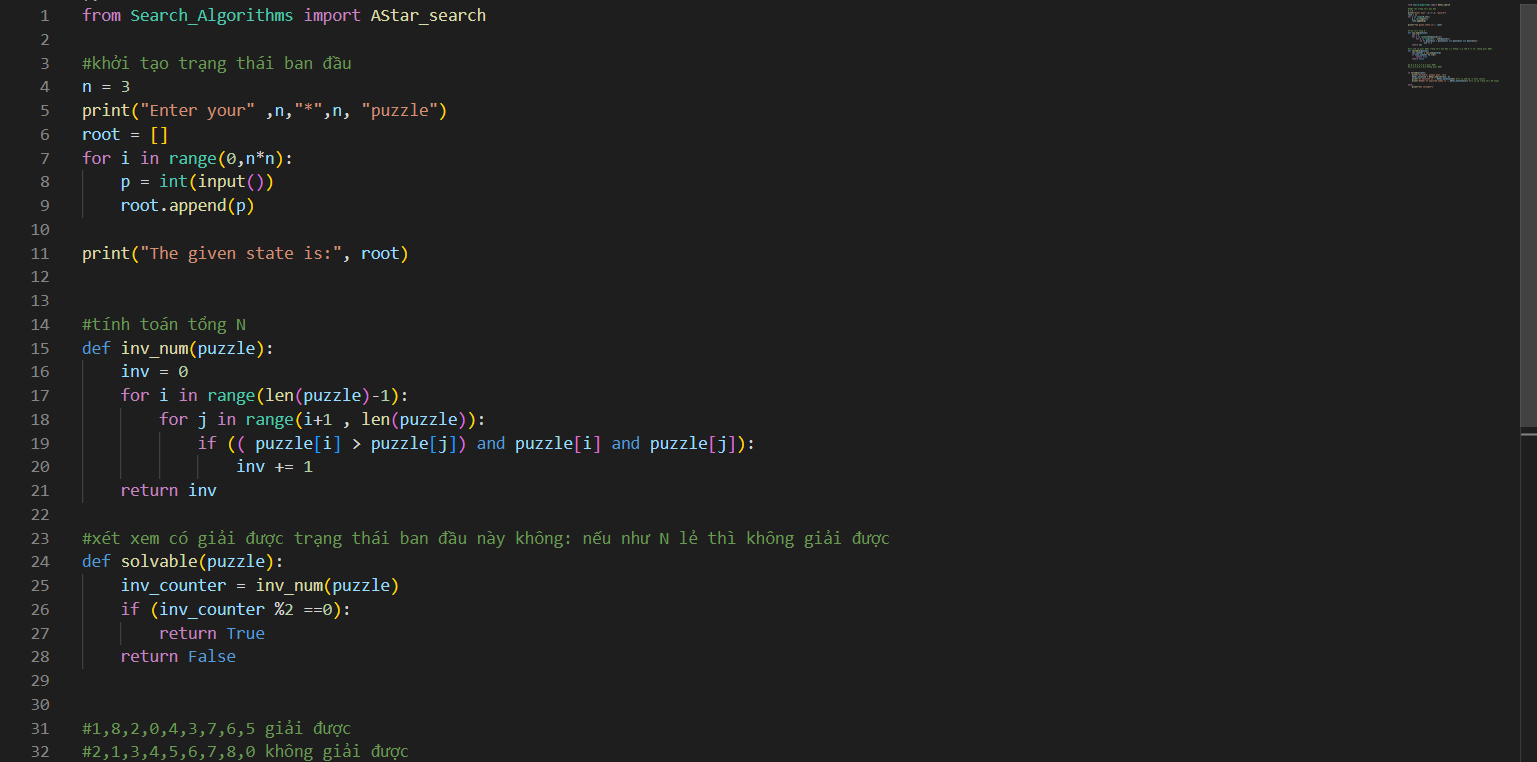


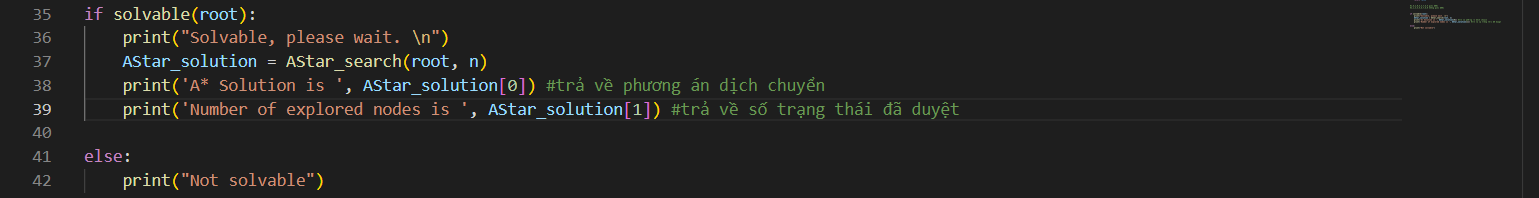


1. Áp dụng thuật toán A\*.



1. Hàm chạy chính.





## **Test case.**

1. Các trường hợp không có lời giải.

* [3, 7, 0, 4, 2, 5, 8, 1, 6]

N = 2+5+0+2+1+1+2+0+0 = 13 (số lẻ)

https://lh7-us.googleusercontent.com/-4T8d31YEc3I0I4LZekIv6SInx9l21YQjBegbGioXxJTHpPW3-Iw8Ff3dnrZkh8DYTLTwEsh6eVyhosiXBF68HzpcLULA1IitYHfk6-3_wl4_AnLJJiIthLUZDN3bCNV5zjvsXYkJ1CcSHI7

* [4, 6, 1, 0, 2, 5, 3, 8, 7]

N = 3+4+0+0+1+0+1+0 = 9 (số lẻ)

https://lh7-us.googleusercontent.com/6TmQruGOMVeVpGO-4zjUfxIOIRDdelGckzFcak5AK7MtZxSCxYDZ3Op5GGwCkpBGSxnK5Pj_6eGaD-Y4i-sMknMQTa-XoZn2GXTwRIsRRBqsmc7dcX12se3L_WiZPJW3gvmFEs0YBXH4vX54

* [7, 2, 1, 3, 4, 0, 6, 8, 5]

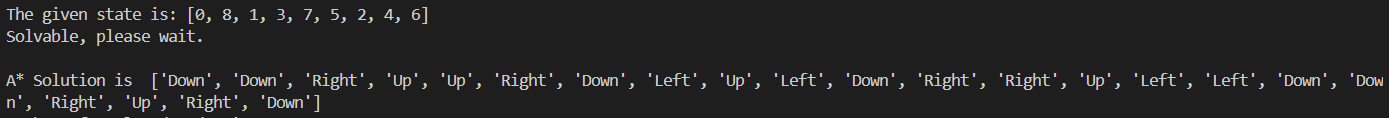
N = 6+1+0+0+0+1+1+0 = 9 (số lẻ)

https://lh7-us.googleusercontent.com/h4NRLbFOpLMauEtthS8FbsSU8OA1sqOONIJQ4fT9gBjbqG8jQA6qMEX9VA3WtC-yFY84ZUXeUHsC3Tez7P1oxhxSl5TUBuZB_VXnYhaxXLnb1wKW58jLTmWZiOMFEuFcpg77OlH3AIbArLy9

1. Các trường hợp có lời giải.

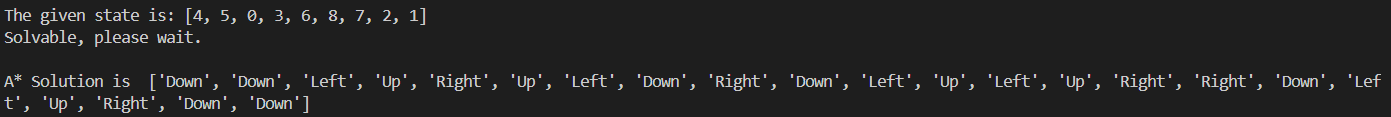
* [0, 8, 1, 3, 7, 5, 2, 4, 6]

N = 7+0+1+4+2+0+0+0 = 14 (số chẵn)



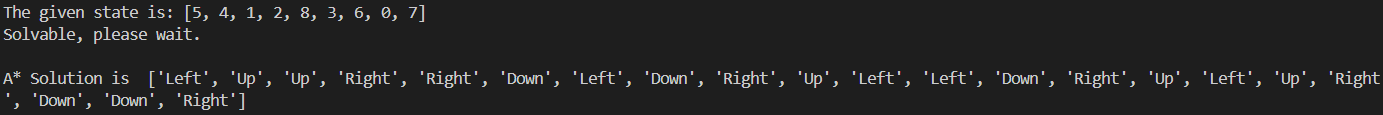
* [4, 5, 0, 3, 6, 8, 7, 2, 1]

N = 3+3+2+2+3+2+1+0 = 16 (số chẵn)



* [5, 4, 1, 2, 8, 3, 6, 0, 7]

N = 4+3+0+0+3+0+0+0 = 10 (số chẵn)



# **Kết luận.**

* Nếu không gian trạng thái là hữu hạn và có lời giải thì có thể nói giải thuật A\* là hoàn chỉnh, nhưng có thể chưa tối ưu. Muốn A\* tối ưu thì hàm h(n) phải có tính chấp nhận được – tức là nó không bao giờ đánh giá cao hơn chi phí nhỏ nhất thực sự của việc đi tới đích.
* Độ phức tạp của A\* phụ thuộc vào hàm đánh giá h(n).
* Thuật toán A\* khá hiệu quả khi giải bài toán 8 puzzle. Giải thuật A\* là thuật toán tìm kiếm trong đồ thị có thông tin phản hồi, sử dụng đánh giá heuristic để xếp loại từng nút và duyệt nút theo hàm đánh giá này.

-> Phương pháp tìm kiếm heuristic đã chứng minh được giá trị của mình trong việc giải quyết các bài toán tối ưu hóa và tìm kiếm. Với khả năng cung cấp giải pháp nhanh chóng thông qua việc ước lượng và đánh giá, các thuật toán heuristic không chỉ tiết kiệm thời gian mà còn đảm bảo hiệu quả trong nhiều tình huống phức tạp. Tuy nhiên, cần phải lưu ý rằng phương pháp này không phải lúc nào cũng đưa ra giải pháp chính xác nhất, mà đôi khi chỉ là giải pháp gần đúng. Do đó, việc lựa chọn và tinh chỉnh hàm heuristic sao cho phù hợp với từng bài toán cụ thể là yếu tố then chốt để tối ưu hóa hiệu suất của thuật toán. Trong tương lai, sự phát triển của các phương pháp heuristic hứa hẹn sẽ mở ra những khả năng mới trong việc giải quyết các thách thức mà thế giới hiện đại đang đối mặt. Hiện tại, tìm kiếm có thông tin được ứng dụng rộng rãi trong nhiều lĩnh vực như trò chơi, robot học, lập kế hoạch, chẩn đoán y tế, tra cứu thông tin,... .

# **Tài liệu tham khảo.**

https://vi.wikipedia.org/wiki/Gi%E1%BA%A3i\_thu%E1%BA%ADt\_t%C3%ACm\_ki%E1%BA%BFm\_A\*#:~:text=Trong%20khoa%20h%E1%BB%8Dc%20m%C3%A1y%20t%C3%ADnh,m%C3%A3n%20m%E1%BB%99t%20%C4%91i%E1%BB%81u%20ki%E1%BB%87n%20%C4%91%C3%ADch).

<https://aptech.vn/kien-thuc-tin-hoc/n-puzzle-tim-hieu-ve-cach-giai-bai-toan.html>

<https://www.geeksforgeeks.org/difference-between-informed-and-uninformed-search-in-ai/>