**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**======\*\*\*======**



**BÁO CÁO THỰC NGHIỆM MÔN HỌC**

**TRÍ TUỆ NHÂN TẠO**

Đề tài

**SỬ DỤNG THUẬT TOÁN HEURISTIC ỨNG DỤNG VÀO BÀI TOÁN TÌM ĐƯỜNG ĐI NGẮN NHẤT**

Giảng viên hướng dẫn: **Mai Thanh Hồng**

Nhóm : 15

Họ và tên sinh viên Mã SV:

Đặng Hải Nam 2023605947

Hà Quốc Hưng 2023603364

Đỗ Tuấn Khải 2024601092

Lớp : 20242IT6094010

Hà Nội, tháng 6 năm 2025

MỤC LỤC

[LỜI NÓI ĐẦU 2](#_Toc198158079)

[CHƯƠNG 1: KHÔNG GIAN TRẠNG THÁI VÀ CÁC THUẬT TOÁN TÌM KIẾM HEURISTIC 3](#_Toc198158080)

[I.Không gian trạng thái 3](#_Toc198158081)

[**1. Mô tả trạng thái.** 3](#_Toc198158082)

[**2. Toán tử chuyển trạng thái.** 3](#_Toc198158083)

[**3. Không gian trạng thái của bài toán.** 3](#_Toc198158084)

[II.Các thuật toán tìm kiếm Heuristic 4](#_Toc198158085)

[**1)Tổng quan về giải thuật tìm kiếm Heuristic.** 4](#_Toc198158086)

[**2 ) Tìm kiếm tối ưu**  (Best-First Search-BeFS) 6](#_Toc198158087)

[**3. Thuật toán 𝐴T** 8](#_Toc198158088)

[**4. Thuật toán 𝐴TK** 8](#_Toc198158089)

[**5. Thuật toán 𝐴\*** 9](#_Toc198158090)

[CHƯƠNG 2: ỨNG DỤNG THUẬT TOÁN HEURISTIC VÀO BÀI TOÁN TÌM ĐƯỜNG ĐI NGẮN NHẤT 12](#_Toc198158091)

[I.Không gian trạng thái của bài toán tìm đi ngắn nhất 12](#_Toc198158092)

[II. Phân tích các thành phần để cài đặt thuật toán giải quyết bài toán 13](#_Toc198158093)

[**Giải thuật sử dụng** 13](#_Toc198158094)

[**Đưa sơ đồ về dạng danh sách kề** 13](#_Toc198158095)

[**3.** **Xây dựng bài toán** 14](#_Toc198158096)

[**4.** **Cài đặt chi tiết** 17](#_Toc198158097)

[III. Kết quả bài toán 20](#_Toc198158098)

[IV. KẾT LUẬN 21](#_Toc198158099)

[Tài liệu tham khảo 22](#_Toc198158100)

**LỜI NÓI ĐẦU**

   George Polya định nghĩa heuristic là “sự nghiên cứu về các phương pháp và các quy tắc trong việc khám phá và phát minh” (Polya 1945). Nghĩa này có thể xuất phát từ gốc Hy Lạp của động từ eurisco nghĩa là “tôi phát hiện”. Khi Archimedes nhảy ra khỏi bồn tắm và chộp lấy chiếc vương miện bằng vàng, ông ta đã la lên “Eureka” có nghĩa là “Tôi đã tìm thấy nó!”. Trong tìm kiếm không gian trạng thái, heuristic là các luật dùng để chọn những nhánh nào có khả năng dẫn đến một giải pháp chấp nhận được.

   Trong lĩnh vực trí tuệ nhân tạo (AI), việc giải quyết bài toán bằng máy tính không chỉ đơn thuần là tìm ra lời giải, mà còn là tìm ra con đường tối ưu dẫn đến lời giải đó. Một trong những cách tiếp cận phổ biến và hiệu quả để xử lý các bài toán như vậy là biểu diễn bài toán dưới dạng **không gian trạng thái, kết hợp với các thuật toán tìm kiếm heuristic** – những phương pháp giúp định hướng quá trình tìm kiếm dựa trên kiến thức kinh nghiệm hoặc ước lượng.

   Tìm hiểu về không gian trạng thái và các thuật toán heuristic không chỉ giúp người học hiểu rõ cách AI mô phỏng quá trình giải quyết vấn đề như con người, mà còn là nền tảng quan trọng cho nhiều ứng dụng trong thực tế như lập kế hoạch, robot di chuyển, giải bài toán tổ hợp, và nhiều hệ thống thông minh khác.

   Với mục tiêu áp dụng lý thuyết vào thực tiễn, bài thực nghiệm này được lựa chọn nhằm giúp làm rõ hơn cách xây dựng không gian trạng thái cho một bài toán cụ thể và so sánh hiệu quả của các thuật toán heuristic trong quá trình tìm kiếm lời giải. Qua đó, người thực hiện không chỉ củng cố kiến thức nền tảng, mà còn phát triển tư duy phân tích và đánh giá thuật toán – những kỹ năng thiết yếu trong nghiên cứu và phát triển các hệ thống AI.

# **CHƯƠNG 1: KHÔNG GIAN TRẠNG THÁI VÀ CÁC THUẬT TOÁN TÌM KIẾM HEURISTIC**

## **I.Không gian trạng thái**

### **1. Mô tả trạng thái.**

   Giải bài toán trong không gian trạng thái, trước hết phải xác định dạng mô tả trạng thái bài toán sao cho bài toán trở nên đơn giản hơn, phù hợp bản chất vật lý của bài toán (Có thể sử dụng các xâu ký hiệu, véctơ, mảng hai chiều, cây, danh sách,...).

   Mỗi trạng thái chính là mỗi hình trạng của bài toán, các tình trạng ban đầu và tình trạng cuối của bài toán gọi là trạng thái đầu và trạng thái cuối.

### **2. Toán tử chuyển trạng thái.**

   Toán tử chuyển trạng thái thực chất là các phép biến đổi đưa từ trạng thái này sang trạng thái khác. Có hai cách dùng để biểu diễn các toán tử:

   - Biểu diễn như một hàm xác định trên tập các trạng thái và nhận giá trị cũng trong tập này.

   - Biểu diễn dưới dạng các quy tắc sản xuất S? A có nghĩa là nếu có trạng thái S thì có thể đưa đến trạng thái A.

### **3. Không gian trạng thái của bài toán.**

   - Không gian trạng thái là tập tất cả các trạng thái có thể có và tập các toán tử của bài toán.

         - Không gian trạng thái là một bộ bốn, Ký hiệu: K= (T, S, G, F).

Trong đó:

+ T: tập tất cả các trạng thái có thể có của bài toán.

+ S: trạng thái đầu.

+ G: tập các trạng thái đích.

+ F: tập các toán tử

Ví dụ : Không gian trạng thái của bài toán đong nước là bộ bốn T, S, G, F xác định như sau:

T ={(x,y) / 0 <= x <= m; 0 <=y<=n}

S =(0,0)

G={(x,k) hoặc (k,y) / 0 <= x <= m; 0 <= y <= n}

F = Tập các thao tác đong đầy, đổ ra hoặc đổ sang bình khác thực hiện trên một bình.

## **II.Các thuật toán tìm kiếm Heuristic**

### **1)Tổng quan về giải thuật tìm kiếm Heuristic.**

* 1. **Khái niệm.**
* Tìm kiếm heuristic là một phương pháp tìm kiếm có sử dụng thông tin bổ sung để đánh giá và lựa chọn các trạng thái tiềm năng trong không gian tìm kiếm. Không giống như tìm kiếm mù, vốn mở rộng trạng thái theo một quy tắc cố định (như BFS hoặc DFS), tìm kiếm heuristic dựa vào một hàm heuristic để ước lượng chi phí hoặc lợi ích của một bước đi nhất định, giúp tăng tốc độ tìm kiếm lời giải.

   - Heuristic chỉ là một phỏng đoán chứa các thông tin về bước tiếp theo sẽ được chọn dùng trong việc giải quyết một vấn đề.

   - Heuristic là những tri thức được rút ra từ những kinh nghiệm, “trực giác” của con người

   - Heuristic có thể là những tri thức đúng hoặc sai

    Vì các heuristic sử dụng những thông tin hạn chế nên chúng ít khi có khả năng đoán trước chính xác cách hành xử của không gian trạng thái ở những giai đoạn xa hơn

1. **Chức năng của Heuristic.**

   Các chương trình giải quyết những vấn đề trí tuệ nhân tạo sử dụng Heuristic cơ bản theo hai dạng:

     - Vấn đề có thể không có giải pháp chính xác vì những điều không rõ ràng trong diễn đạt vấn đề hoặc trong các dữ liệu có sẵn.

     - Vấn đề có thể có giải pháp chính xác, nhưng chi phí tính toán để tìm ra nó không cho phép.

**c.  Ưu điểm của Heuristic.**

   Thuật giải Heuristic thể hiện cách giải bài toán với các đặc tính sau:

   - Thường tìm được lời giải tốt (Nhưng không chắc là lời giải tốt nhất).

   - Giải bài toán theo thuật giải Heuristic thường dễ dàng và nhanh chóng đưa ra kết quả hơn so với giải thuật tối ưu, vì vậy chi phí thấp hơn.

   -Thuật giải Heuristic thường thể hiện khá tự nhiên, gần gũi với cách suy nghĩ và hành động con người.

**d. Thuật giải Heuristic:**

   - Heuristic là những tri thức được rút ra từ kinh nghiệm con người, nó có thể đúng hoặc sai.

   - Heuristic thường được dùng trong các trường hợp sau:

   + Bài toán có thể không có nghiệm chính xác do các mệnh đề không phát biểu chặt chẽ hay thiếu dữ liệu để khẳng định kết quả.

   + Bài toán có nghiệm chính xác nhưng phí tổn tính toán để đưa ra nghiệm là quá lớn(bùng nổ tổ hợp).

   - Thuật giải Heuristic là mở rộng khái niệm thuật toán và có đặc điểm:

  +Thường tìm lời giải tốt nhưng không tốt nhất.

  +Nhanh chóng tìm ra được giải thuật tối ưu, tiết kiệm chi phí hơn.

  +Thường thể hiện khá tự nhiên, gần gũi với cách hành động và suy nghĩ của con người.

**e. Thiết kế thuật giải heuristic:**

   Có nhiều phưong pháp để thiết kế thuật giải Heuristic, trong đó người ta sử dụng 2 nguyên lý cơ bản sau:

    -*Nguyên lý vét cạn thông minh*: Khi không gian tìm kiếm D lớn, ta thường tìm câch giới hạn lại không gian tìm kiếm này hoặc thực hiện một kiểu dò tìm đặc biệt dựa vào đặc thù của bài toán để nhanh chóng tìm ra mục tiêu.

    -*Nguyên lý tham lam*: Lấy tiêu chuẩn tối ưu nhất trên phạm vi toàn cục của bài toán để làm tiêu chuẩn chọn kựa hành động cho phạm vi cục bộ của từng bước trong quá trình tìm kiếm lời giải.

    -*Nguyên lý thứ tự*: Thực hiện hành động dựa trên một cấu trúc thứ tự hợp lý của không gian khảo sát nhằm nhanh chóng đặt được lời giải tốt.

   -*Tìm kiếm leo đồi*: Tìm kiếm leo đồi(HCS) thực chất là tìm kiếm theo chiều sâu nhưng không thể quay lui. Trong tìm kiếm leo đồi, việc lựa chọn trạng thái tiếp theo được quyết định dựa trên một hàm Heuristic:

Tư tưởng của thuật giải được thể hiện qua 2 bước:

   i, Nếu trạng thái bắt đầu cũng là trạng thái đích thì thoát và báo là tìm được lời giải.

   ii, Lặp lại cho đến khi trạng thái kết thúc goặc cho đến khi Ti không tồn tại 1 trạng thái nào tốt hơn trạng thái hiện tại.

     + Đặt S bằng tập tất cả các trạng thái có thể có của Ti và tốt hơn Ti.

     + Xác định Tmax là trạng thái tốt nhất trong tập S: Ti=Tmax

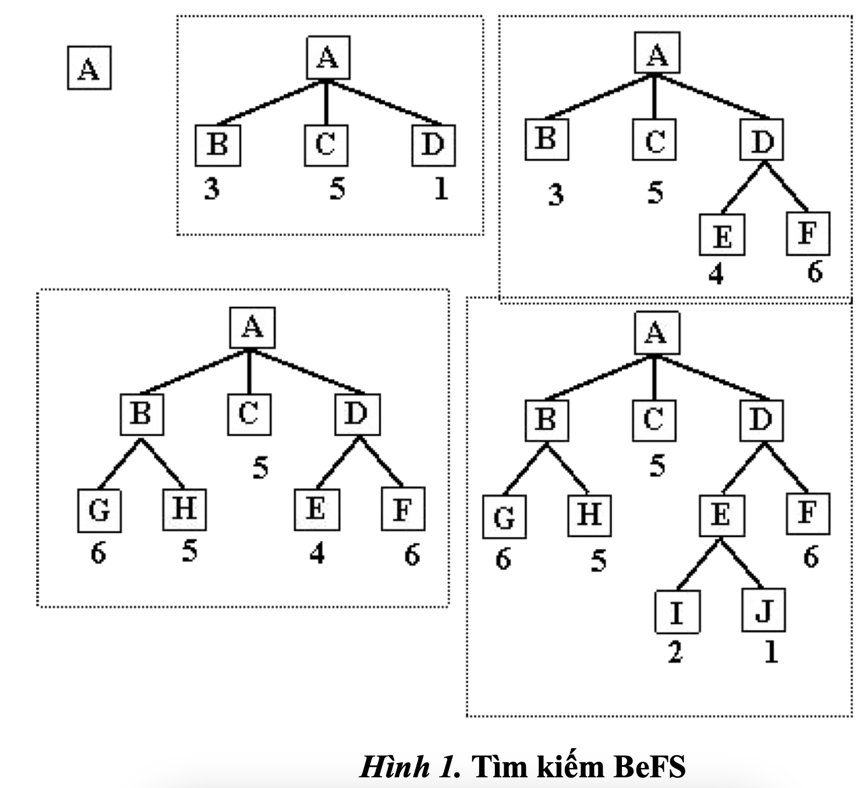
     Và Lặp lại cho đến khi báo dừng .

### **2 ) Tìm kiếm tối ưu**  (Best-First Search-BeFS)

   Ưu điểm của tìm kiếm theo chiều sâu là không phải quan tâm đến sự mở rộng của tất cả các nhánh. Ưu điểm của tìm kiếm chiều rộng là không bị sa vào các đường dẫn bế tắc (các nhánh cụt). Tìm kiếm tối ưu (Best-First Search-BeFS) sẽ kết hợp hai phương pháp trên cho phép ta đi theo một con đường duy nhất tại một thời điểm, nhưng đồng thời vẫn xét được những hướng khác. Nếu con đường đang đi không triển vọng bằng những con đường đang quan sát, ta sẽ chuyển sang đi theo một trong số các con đường này.

   Một cách cụ thể, tại mỗi bước của tìm kiếm BeFS, ta chọn đi theo trạng thái có khả năng cao nhất trong số các trạng thái đã được xét *cho đến thời điểm đó.* BeFS khác với tìm kiếm leo đồi là chỉ chọn trạng thái có khả năng cao nhất trong số các trạng thái kế tiếp có thể đến được thu trạng thái hiện tại. Như vậy, với tiếp cận này, ta sẽ ưu tiên đi vào những nhánh tìm kiếm có khả năng nhất (giống tìm kiếm leo đồi), nhưng ta sẽ không bị liên quan trong các nhánh này vì nếu càng đi sâu vào một hướng mà ta phát hiện ra rằng hướng này càng đi thì càng xấu, đến mức nó xấu hơn cả những hướng mà ta chưa đi, thì ta sẽ không đi tiếp hướng hiện tại nữa mà chọn đi theo một hướng tốt nhất trong số những hướng chưa đi. Đó là tư tưởng chủ đạo của tìm kiếm tối ưu.

VÍ DỤ:



Để làm rõ tư tưởng này ta xét đồ thị (hình 1). Ban đầu ta duyệt đỉnh , từ đỉnh này có thể đi đến các đỉnh B, , . họn đỉnh để đi tiếp vì có chi phí nhỏ nhất, tiếp đến ta phát triển đỉnh này được các đỉnh kề và F. Tuy nhiên các đỉnh này lại có chi pho lớn hơn đỉnh B do đó ta lại quay lại để duyệt đỉnh B. uá trình này cứ tiếp tục như vậy cho đến khi tìm được đỉnh đích.

**Thuật giải tìm kiếm tối ưu**

1. Đặt OPEN chứa trạng thái khởi đầu T0.

2. ho đến khi tìm được trạng thái đoch hoặc không còn nút nào trong OPEN, thực hiện :

a. họn trạng thái tốt nhất (Tmax) trong OPN (và xóa Tmax khỏi OPEN) b. Nếu Tmax là trạng thái kết thúc thì thoát.

c. Ngược lại, tạo ra các trạng thái kế tiếp Tk có thể có tổng trạng thái Tmax. Đối với mỗi trạng thái kế tiếp Tk thực hiện :

Tính f(Tk); Thêm Tk vào OPEN

BeFS khá đơn giản. Tuy vậy, trên thực tế, cũng như tìm kiếm chiều sâu và chiều rộng, hiếm khi ta dùng BeFS một cách trực tiếp. Thông thường, người ta thường dùng các phiên bản của BeFS là AT, AKT và A\*.

***Thông tin về quá khứ và tương lai:***

Thông thường, trong các phương án tìm kiếm theo kiểu BeFS, chi pho f của một trạng thái được tonh dựa theo hai giá trị mà ta gọi là là g và h. Trong đó h, như đã biết, đó là một ước lượng về chi phí từ trạng thái hiện hành cho đến trạng thái đích (thông tin tương lai), còn g là chiều dài quãng đường đã đi từ trạng thái ban đầu cho đến trạng thái hiện tại (thông tin quá khứ). Khi đó hàm ước lượng từng chi phí f(n) được tonh theo công thức:

f(n) = g(n) + h(n)

### **3. Thuật toán 𝐴T**

Thuật giải *AT* là một phương pháp tìm kiếm theo kiểu BeFS với chi phí của đỉnh là giá trị hàm g (tổng chiều dài thực sự của đường đi từ đỉnh bắt đầu đến đỉnh hiện tại).

**Giải thuật :**

1. Đặt OPEN chứa trạng thái khởi đầu.
2. Cho đến khi tìm được trạng thái đích hoặc không còn nút nào trong OPEN, thực hiện:
3. Chọn trạng thái (Tmax) có giá trị g nhỏ nhất trong OPEN (và xóa Tmax khỏi OPEN)
4. Nếu Tmax là trạng thái kết thúc thì thoát.
5. Ngược lại, tạo ra các trạng thái kế tiếp Tk có thể có từ trạng thái Tmax.

Đối với mỗi trạng thái kế tiếp Tk thực hiện:

g(Tk) = g(Tmax) + cost(Tmax, Tk)

Thêm Tk vào OPEN.

**\*Note:** Vì chỉ sử dụng hàm g (mà không dùng hàm ước lượng h’ để đánh giá độ tốt của một trạng thái nên ta cũng có thể xem AT chỉ là một thuật toán.

### **4. Thuật toán 𝐴TK**

Thuật giải 𝐴𝐾𝑇 trong quá trình tìm đường đi chỉ xét đến các đỉnh và giá của chúng. Nghĩa là việc tìm đỉnh triển vọng chỉ phụ thuộc hàm g(n) (thông tin quá khứ). Tuy nhiên thuật giải này không còn phù hợp khi gặp phải những bài toán phức tạp (độ phức tạp cấp hàm mũ) do ta phải tháo một lượng nút lớn. Để khắc phục nhược điểm này, người ta sử dụng thêm các thông tin bổ sung xuất phát từ bản thân bài toán để tìm ra các đỉnh có triển vọng, tức là đường đi tối ưu sẽ tập trung xung quanh đường đi tốt nhất nếu s dụng các thông tin đặc tả về bài toán (thông tin quá tương lai).

Theo thuật giải này, chi phí của đỉnh được xác định:

f(n) = g(n) + h(n)

Đỉnh n được chọn nếu f(n) min.

Việc xác định hàm ước lượng h(n) được thực hiện dựa theo:

- Chọn toán tử xây dựng cung sao cho có thể loại bớt các đỉnh không liên quan và tìm ra các đỉnh có triển vọng.

- Sử dụng thêm các thông tin bổ sung nhằm xây dựng tập OPEN và cách lấy các đỉnh trong tập OPEN.

Để làm được việc này người ta phải đưa ra độ đo, tiêu chun để tìm ra các đỉnh có triển vọng. ác hàm s dụng các kỹ thuật này gọi là hàm đánh giá. Sau đây là một số phương pháp xây dựng hàm đánh giá:

- Dựa vào xác suất của đỉnh trên đường đi tối ưu.

- Dựa vào khoảng cách, sự sai khác của trạng thái đang xét với trạng thái đích hoặc các thông tin liên quan đến trạng thái đích.

**Giải thuật :**

1. Đặt OPENchứa trạng thái khởi đầu.
2. Cho đến khi tìm được trạng thái đích hoặc không còn nút nào trong OPEN, thực hiện :
3. Chọn trạng thái (Tmax) có giá trị f nhỏ nhất trong OPEN (và xóa Tmax khỏi OPEN)
4. Nếu Tmaxlà trạng thái kết thúc thì thoát.
5. Ngược lại, tạo ra các trạng thái kế tiếp Tk có thể có từ trạng thái Tmax. Đối với mỗi trạng thái kế tiếp Tk thực hiện :

g(Tk) = g(Tmax) + cost(Tmax, Tk).

 Tính h’(Tk) .

f(Tk) = g(Tk) + h’(Tk) .

Thêm Tk vào OPEN.

### **5. Thuật toán 𝐴\***

A\* là giải thuật tìm kiếm trong đồ thị, tìm đường đi từ một đỉnh hiện tại đến đỉnh đích có sử dụng hàm để ước lượng khoảng cách hay còn gọi là hàm Heuristic.

Từ trạng thái hiện tại A\* xây dựng tất cả các đường đi có thể đi dùng hàm ước lượng khoảng cách (hàm Heuristic) để đánh giá đường đi tốt nhất có thể đi. Tùy theo mỗi dạng bài khác nhau mà hàm Heuristic sẽ được đánh giá khác nhau. A\* luôn tìm được đường đi ngắn nhất nếu tồn tại đường đi như thế.

A\* lưu giữ một tập các đường đi qua đồ thị, từ đỉnh bắt đầu đến đỉnh kết thúc, tập các đỉnh có thể đi tiếp được lưu trong tập OPEN.

Thứ tự ưu tiên cho một đường đi được quyết định bởi hàm Heuristic được đánh giá

f(x) = g(x) + h(x)

Trong đó:

- g(x) là chi chi phí của đường đi từ điểm xuất phát cho đến thời điểm hiện tại.

- h(x) là hàm ước lượng chi phí từ đỉnh hiện tại đến đỉnh đích f(x) thường có giá trị càng thấp thì độ ưu tiên càng cao.

- Open: tập các trạng thái đã được sinh ra nhưng chưa được xét đến.

**Giải thuật :**

* Close: tập các trạng thái đã được xét đến.
* Cost(p, q): là khoảng cách giữa p, q.
* g(p): khoảng cách từ trạng thái đầu đến trạng thái hiện tại p.
* h(p): giá trị được lượng giá từ trạng thái hiện tại đến trạng thái đích.
* f(p) = g(p) + h(p)

**Các bước thực hiện :**

* Bước 1:

Open: = {s}

Close: = {}

* Bước 2: while (Open !={})

+ Chọn trạng thái (đỉnh) tốt nhất p trong Open (xóa p khỏi Open).

+ Nếu p là trạng thái kết thúc thì thoát.

+ Chuyển p qua Close và tạo ra các trạng thái kế tiếp q sau p.

Nếu q đã có trong Open

* Nếu g(q) > g(p) + Cost(p, q) thì

g(q) = g(p) + Cost(p, q)

f(q) = g(q) + h(q)

prev(q) = p (đỉnh cha của q là p)

      Nếu q chưa có trong Open

g(q) = g(p) + cost(p, q)

f(q) = g(q) + h(q)

prev(q) = p

Thêm q vào Open

       Nếu q có trong Close

Nếu g(q) > g(p) + Cost(p, q)

Bỏ qkhỏi Close

Thêm q vào Open

* Bước 3: Không tìm được.

**Các tính chất :**

Cũng như tìm kiếm theo chiều rộng (breadth-first search), A\* là thuật toán đầy đủ (complete) theo nghĩa rằng nó sẽ luôn luôn tìm thấy một lời giải nếu bài toán có lời giải.

Nếu hàm heuristic h có tính chất thu nạp được (admissible), nghĩa là nó không bao giờ đánh giá cao hơn chi phí nhỏ nhất thực sự của việc đi tới đích, thì bản thân A\* có tính chất thu nạp được (hay tối ưu) nếu sử dụng một tập đóng. Nếu không sử dụng tập đóng thì hàm h phải có tính chất đơn điệu (hay nhất quán) thì A\* mới có tính chất tối ưu. Nghĩa là nó không bao giờ đánh giá chi phí đi từ một nút tới một nút kề nó cao hơn chi phí thực. Phát biểu một cách hình thức, với mọi nút x,y trong đó y là nút tiếp theo của x:

A\* còn có tính chất hiệu quả một cách tối ưu (optimally efficient) với mọi hàm heuristic h, có nghĩa là không có thuật toán nào cũng sử dụng hàm heuristic đó mà chỉ phải mở rộng ít nút hơn A\*, trừ khi có một số lời giải chưa đầy đủ mà tại đó h dự đoán chính xác chi phí của đường đi tối ưu.

Quan hệ với tìm kiếm chi phí đều (uniform-cost search)

Thuật toán Dijkstra là một trường hợp đặc biệt của A\* trong đó đánh giá heuristic là một hàm hằng h(x) = 0 với mọi x.

**-Mức độ phức tạp**

Vấn đề sử dụng bộ nhớ của A\* còn rắc rối hơn độ phức tạp thời gian. Trong trường hợp xấu nhất, A\* phải ghi nhớ số lượng nút tăng theo hàm mũ. Một số biến thể của A\* đã được phát triển để đối phó với hiện tượng này, một trong số đó là A\* lặp sâu dần (iterative deepening A\*), A\* bộ nhớ giới hạn (memory-bounded A\*- MA\*) và A\* bộ nhớ giới hạn đơn giản (simplified memory bounded A\*). Một thuật toán tìm kiếm có thông tin khác cũng có tính chất tối ưu và đầy đủ nếu đánh giá heuristic của nó là thu nạp được (admissible). Đó là tìm kiếm đệ quy theo lựa chọn tốt nhất (recursive best-first search- RBFS).

**-Ưu điểm**

Một thuật giải linh động, tổng quát, trong đó hàm chứa cả tìm kiếm chiều sâu, tìm kiếm chiều rộng 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.

**-Nhược điểm**

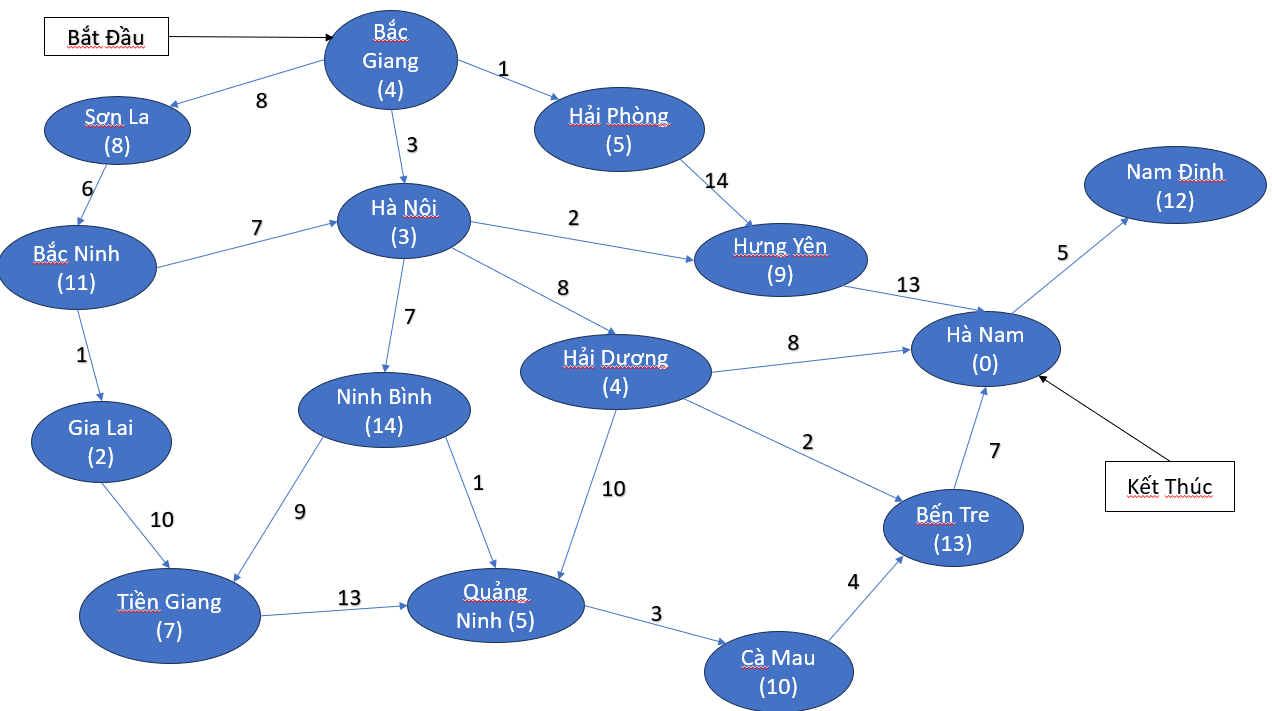
         A\* rất linh động nhưng vẫn gặp một khuyết điểm cơ bản- giống như chiến lược tìm kiếm chiều rộng- đó là tốn khá nhiều bộ nhớ để lưu lại những trạng thái đã đi qua.

# **CHƯƠNG 2: ỨNG DỤNG THUẬT TOÁN HEURISTIC VÀO BÀI TOÁN TÌM ĐƯỜNG ĐI NGẮN NHẤT**

## **I.Không gian trạng thái của bài toán tìm đi ngắn nhất**

**1. Giới thiệu bài toán**

Một người có trong ta bản đồ mạng lưới giao thông nối các tỉnh của nước Việt Nam. Du khách đang ở Bắc Giang và anh ta muốn tìm đường đi tới Hà Nam.

****

Hình 2 : Tìm đường đi ngắn nhất từ Bắc Giang đến Hà Nam

Trong bài toán này, các tỉnh thành có trong bản đồ là các trạng thái tỉnh Bắc Giang là trạng thái đầu còn Hà Nam là trạng thái kết thúc. Với bài toán nà ta có thể sử dụng cách biểu diễn của đồ thị (ma trận kề, ma trận trọng số, danh sách cạnh, danh sách kề).

## **II. Phân tích các thành phần để cài đặt thuật toán giải quyết bài toán**

### **Giải thuật sử dụng**

A\* là giải thuật tìm kiếm trong đồ thị, tìm đường đi từ một đỉnh hiện tại đến đỉnh đích có sử dụng hàm để ước lượng khoảng cách hay còn gọi là hàm Heuristic.

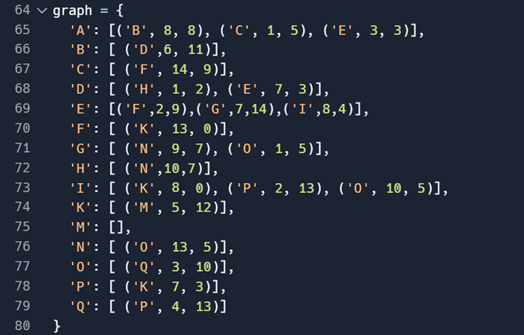
Từ trạng thái hiện tại A\* xây dựng tất cả các đường đi có thể đi dùng hàm ước lượng khoảng cách (hàm Heuristic) để đánh giá đường đi tốt nhất có thể đi. Tùy theo mỗi dạng bài khác nhau mà hàm Heuristic sẽ được đánh giá khác nhau. A\* luôn tìm được đường đi ngắn nhất nếu tồn tại đường đi như thế.

A\* lưu giữ một tập các đường đi qua đồ thị, từ đỉnh bắt đầu đến đỉnh kết thúc, tập các đỉnh có thể đi tiếp được lưu trong tập Open.

Thứ tự ưu tiên cho một đường đi được quyết định bởi hàm Heuristic được đánh giá f(x) = g(x) + h(x)

* g(x) là chi chi phí của đường đi từ điểm xuất phát cho đến thời điểm hiện tại.
* h(x) là hàm ước lượng chi phí từ đỉnh hiện tại đến đỉnh đích f(x) thường có giá trị càng thấp thì độ ưu tiên càng cao.

### **Đưa sơ đồ về dạng danh sách kề**



Hình 3: Dạng danh sách kề của đồ thị

Trong đó:

A: Bắc Giang B: Sơn La

C: Hải Phòng D: Bắc Ninh

E: Hà Nội F: Hưng Yên

G: Ninh Bình H: Gia Lai

I: Hải Dương K: Hà Nam

M: Nam Định N: Tiền Giang

O: Quảng Ninh P: Bến Tre

Q: Cà Mau

1. **Xây dựng bài toán**

-Đỉnh xuất phát từ A

-Goal = {K}

Ban đầu OPEN = {A, g(A) = 0, f() = 4}

Phát triển đỉnh A sinh ra các đỉnh con B, C và E. Tính giá trị của hàm f tại   các đỉnh này ta có:

OPEN = {g(B) = 8, f(B) = 8 + 8 = 16 , cha(B) = A ,

g(C) =1 , f(C) = 1 + 5 = 6, cha(C) = A,

g(E) = 3, f(E) = 3 + 3 = 6, cha(E) =A

}

CLOSE = {A, g(A) = 0, f(A) = 4 }

Do f(C) = f(E) nhỏ nhất nên chọn ngẫu nhiên 1 trong 2 đỉnh này. Giả sử ta chọn đỉnh C để phát triển. Phát triển C, ta nhận được đỉnh kế tiếp là đỉnh F.

g(F) = 1 + 14 = 15, f(F) = 15 + 9 = 24.

Bây giờ ta bổ sung đỉnh F này vào tập OPEN.

OPEN = {g(B) = 8, f(B) = 8 + 8 = 16 , cha(B) = A ,

g(E) = 3, f(E) = 3 + 3 = 6, cha(E) =A,

g(F) = 15, f(F) = 15 + 9 = 24, cha(F) = C

}

CLOSE = {A, g(A) = 0, f(A) = 4,

C, g(C) =1 , f(C) = 1 + 5 = 6, cha(C) = A

}

Với lập luận tương tự như trên ta chọn đỉnh E để phát triển. Các đỉnh kế tiếp của E là G, I và F.

OPEN = {g(B) = 8, f(B) = 8 + 8 = 16 , cha(B) = A ,

g(F) = 5, f(F) = 5 + 9 = 14, cha(F) = E,

g(G) = 10, f(G) = 10 + 14 = 24, cha(G) = E,

g(I) = 11, f(I) = 11 + 4 = 15, cha(I) = E

}

CLOSE = {A, g(A) = 0, f(A) = 4,

C, g(C) =1 , f(C) = 1 + 5 = 6, cha(C) = A,

E, g(E) = 3, f(E) = 3 + 3 = 6, cha(E) =A

}

Chọn đỉnh F để phát triển. Các đỉnh tiếp của F là K.

OPEN = {g(B) = 8, f(B) = 8 + 8 = 16 , cha(B) = A ,

g(G) = 10, f(G) = 10 + 14 = 24, cha(G) = E,

g(I) = 11, f(I) = 11 + 4 = 15, cha(I) = E,

g(K) = 18, f(K) =18 + 0 = 18, cha(K) = F

}

CLOSE = {A, g(A) = 0, f(A) = 4,

C, g(C) =1 , f(C) = 1 + 5 = 6, cha(C) = A,

E, g(E) = 3, f(E) = 3 + 3 = 6, cha(E) =A,

F, g(F) = 5, f(F) = 5 + 9 = 14, cha(F) = E

}

Chọn đỉnh I để phát triển. Các đỉnh tiếp của I là O, P và K.

OPEN = {g(B) = 8, f(B) = 8 + 8 = 16 , cha(B) = A ,

g(G) = 10, f(G) = 10 + 14 = 24, cha(G) = E,

g(K) = 18, f(K) =18 + 0 = 18, cha(K) = F,

g(O) = 21, f(O) = 21 + 5 = 26, cha(O) = I,

g(P) = 13, f(P) = 13+ 13 = 26, cha(P) = I

}

CLOSE = {A, g(A) = 0, f(A) = 4,

C, g(C) =1 , f(C) = 1 + 5 = 6, cha(C) = A,

E, g(E) = 3, f(E) = 3 + 3 = 6, cha(E) =A,

F, g(F) = 5, f(F) = 5 + 9 = 14, cha(F) = E,

I, g(I) = 11, f(I) = 11 + 4 = 15, cha(I) = E

}

          Chọn đỉnh B để phát triển. Các đỉnh tiếp của B là D.

OPEN = {

g(G) = 10, f(G) = 10 + 14 = 24, cha(G) = E,

g(K) = 18, f(K) =18 + 0 = 18, cha(K) = F,

g(O) = 21, f(O) = 21 + 5 = 26, cha(O) = I,

g(P) = 13, f(P) = 13+ 13 = 26, cha(P) = I

g(D) = 14, f(B) = 14 +11 = 25 , cha(D) = B ,

}

CLOSE = {A, g(A) = 0, f(A) = 4,

C, g(C) =1 , f(C) = 1 + 5 = 6, cha(C) = A,

E, g(E) = 3, f(E) = 3 + 3 = 6, cha(E) =A,

F, g(F) = 5, f(F) = 5 + 9 = 14, cha(F) = E,

I, g(I) = 11, f(I) = 11 + 4 = 15, cha(I) = E

                               B, g(B) =8, f(B) = 8 + 8 = 16, cha(B) = A

}

Chọn đỉnh K để phát triển.

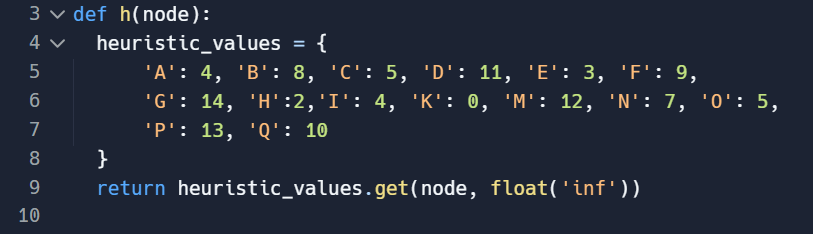
Do K ∈ Goal nên quá trình tìm kiếm kết thúc. Để đưa ra đường đi ta truy ngược lại trong tập CLOSE.

Khi đó đường đi tìm được có chi phí c(p) = 18 và trình tự các đỉnh là:

p: A→ E→ F → K

1. **Cài đặt chi tiết**

**a, Hàm h(node)**

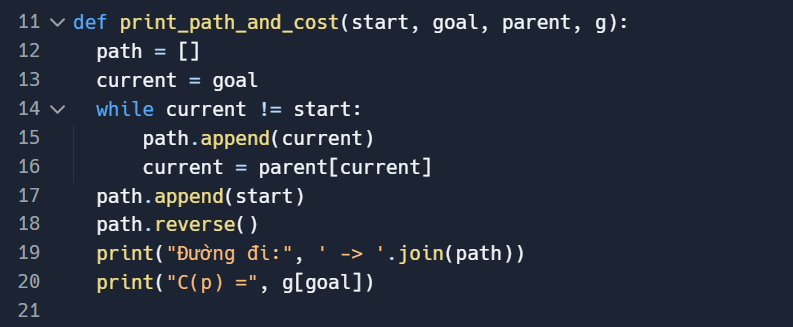


Hình 4: Hàm trả về giá trị heuristic cho nút

+ Hàm h nhận vào một nút (node) và trả về giá trị heuristic (ước lượng chi phí từ nút đó đến nút đích).

+ Dữ liệu heuristic được lưu trong từ điển heuristic\_values. Nếu node có trong từ điển, hàm trả về giá trị heuristic tương ứng. Nếu không, nó trả về float('inf'), một giá trị vô hạn, biểu thị rằng không có thông tin heuristic cho nút đó.

**b, Hàm in ra đường đi ngắn nhất**



Hình 5: hàm in ra đường đi

  + Hàm in ra **đường đi** từ nút start đến nút goal và **chi phí** tương ứng (g[goal], chi phí từ start đến goal).

          + Để in ra đường đi, hàm lần lượt truy ngược các nút từ goal về start thông qua từ điển parent, giúp xác định "cha" của mỗi nút.

+ Sau khi xây dựng xong đường đi (theo hướng từ goal ngược về start), hàm đảo ngược danh sách path để có đúng thứ tự từ start đến goal.

+ Cuối cùng, hàm in ra đường đi và chi phí C(p) (chi phí từ start đến goal).

**c, Xây dựng hàm tìm đường đi ngắn nhất**



Hình 6 : Hàm xây dựng đường đi

Hàm A\_star là thuật toán A\* để tìm đường đi ngắn nhất từ một đỉnh start đến một hoặc nhiều đỉnh đích trong đồ thị. Đoạn mã thực hiện các bước sau:

* **Khởi tạo**:
* **MO**: Tập các đỉnh mở, chứa các đỉnh mà thuật toán đã phát hiện nhưng chưa xét đến. Ban đầu, chỉ chứa đỉnh start.
* **DONG**: Tập các đỉnh đã xét xong.
* **g**: Từ điển lưu chi phí thực tế từ start đến mỗi đỉnh. Ban đầu g[start] = 0 (vì đi từ start đến chính nó không tốn chi phí).
* **f**: Từ điển lưu giá trị tổng hợp f(n) = g(n) + h(n), trong đó:
* g(n): chi phí thực tế từ start đến n.
* h(n): ước lượng chi phí từ n đến mục tiêu ban đầu, f[start] = h(start).
* **parent**: Từ điển lưu trữ "cha" của mỗi đỉnh, giúp truy vết lại đường đi sau này.
* **Quy trình thuật toán**:

+ Chọn đỉnh n có giá trị f(n) nhỏ nhất trong tập MO (ưu tiên đỉnh có ước lượng chi phí thấp nhất để xét trước).

+ Nếu n là một trong các đỉnh đích (có trong goals), thuật toán gọi hàm print\_path\_and\_cost để in ra đường đi từ start đến n và chi phí tổng.

+ Nếu n không phải là đích, thuật toán tiếp tục mở rộng các đỉnh lân cận m của n:

* Cập nhật chi phí g[m] từ start đến m.
* Tính giá trị f(m) là tổng chi phí thực tế g(m) và giá trị heuristic h(m).
* Nếu m chưa có trong MO hoặc DONG, thêm m vào MO và cập nhật thông tin của m.
* Nếu m đã có trong MO và có chi phí mới tốt hơn, cập nhật chi phí và cha của m.

     + Lặp lại quá trình cho đến khi tìm được đích hoặc không còn đỉnh nào trong MO.

## **III. Kết quả bài toán**

Sau khi thực hiện cài đặt và chạy thuật toán A\* trên đồ thị mô phỏng bản đồ các tỉnh thành, kết quả thu được là:

* Đường đi ngắn nhất từ Bắc Giang (A) đến Hà Nam (K) là:

→ A → E → F → K

* Chi phí (tổng quãng đường) tương ứng của đường đi này là:

→ C(p) = 18

Trong quá trình tìm kiếm, thuật toán A\* đã phát triển các đỉnh dựa trên giá trị hàm f(n) = g(n) + h(n), nơi:

* g(n) là chi phí thực tế từ điểm xuất phát đến đỉnh n.
* h(n) là ước lượng khoảng cách từ n đến đích (dựa trên hàm heuristic).

Thuật toán đã tối ưu việc mở rộng đỉnh bằng cách chỉ lựa chọn những đỉnh có khả năng dẫn đến lời giải tối ưu nhất theo từng bước. Kết quả chứng minh rằng A\* là một thuật toán hiệu quả trong bài toán tìm đường đi ngắn nhất, cả về chi phí lẫn độ chính xác.

## **IV. KẾT LUẬN**

Mỗi phương pháp giải đều có những ưu và nhược điểm riêng, tuy nhiên trong bài toán tìm đường đi ngắn nhất, giải thuật A\* tỏ ra vượt trội nhờ khả năng kết hợp hiệu quả giữa tìm kiếm có hướng dẫn và chi phí đường đi thực tế. A\* là một thuật toán tìm kiếm trên đồ thị có thông tin phản hồi, sử dụng hàm đánh giá heuristic để sắp xếp và lựa chọn nút mở rộng, từ đó tối ưu hóa quá trình tìm kiếm.

Trong các bài toán tìm kiếm trên không gian trạng thái, heuristic đóng vai trò quan trọng trong hai trường hợp điển hình:

* Khi bài toán không thể tìm được nghiệm chính xác do thiếu dữ liệu hoặc mô hình chưa đầy đủ.
* Khi có thể tìm được nghiệm chính xác nhưng chi phí tính toán quá cao, dẫn đến bùng nổ tổ hợp.

* Nhờ vào heuristic, giải thuật A\* cho phép tiếp cận lời giải một cách hiệu quả hơn, giảm thiểu chi phí tính toán mà vẫn đảm bảo độ chính xác cao trong nhiều trường hợp thực tiễn.

# **Tài liệu tham khảo**

<https://websitehcm.com/cac-thuat-toan-informed-search-algorithms/>

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

<https://voer.edu.vn/m/thuat-toan-tim-kiem-heuristic/90488a73>

<https://cuuduongthancong.com/atc/1153/cac-phuong-phap-tim-kiem-heuristic>