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

**KHOA CÔNG NGHỆ THÔNG TIN**

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

****

**BÁO CÁO THÍ NGHIỆM / THỰC NGHIỆM**

**HỌC PHẦN: TRÍ TUỆ NHÂN TẠO**

**ĐỀ TÀI**

**ÁP DỤNG THUẬT TOÁN BeFS VÀO TÌM ĐƯỜNG ĐI TỪ MỘT ĐIỂM ĐẾN MỘT ĐIỂM KHÁC TRONG BẢN ĐỒ**

|  |  |
| --- | --- |
| **Giáo viên hướng dẫn** | **: Mai Thanh Hồng** |
| **Lớp** | **: 20242IT6084010** |
| **Nhóm sinh viên thực hiện** | **: Nhóm 9** |
| 1. **Đặng Tuấn Linh** | **Mã SV: 2023601070** |
| 1. **Nguyễn Đăng Khôi** | **Mã SV: 2023600817** |
| 1. **Trịnh Xuân Huấn** | **Mã SV: 2023606415** |
| 1. **Trần Đức Huy** | **Mã SV: 2023602954** |

**Hà Nội, năm 2025**

PHIẾU HỌC TẬP CÁ NHÂN/NHÓM

**I. Thông tin chung**

1. Tên lớp: 20242IT6094003 Khóa:18

2. Họ và tên sinh viên: Đặng Tuấn Linh

3. Tên nhóm: Nhóm 9  
Họ và tên thành viên trong nhóm:

|  |  |
| --- | --- |
| **Họ và tên** | **Mã sinh viên** |
| Đặng Tuấn Linh | 2023601070 |
| Nguyễn Đăng Khôi | 2023600817 |
| Trịnh Xuân Huấn | 2023606415 |
| Trần Đức Huy | 2023602954 |

**II. Nội dung học tập**

1. Tên chủ đề Áp dụng thuật toán BeFS vào tìm đường đi từ một điểm đến một điểm khác trên bản đồ

2. Hoạt động của sinh viên

*¨  Hoạt động/Nội dung 1:* Tìm hiểu lý thuyết về thuật toán BeFS

Mục tiêu/chuẩn đầu ra: Sinh viên nắm vững kiến thức cơ bản về thuật toán Best First Search (BeFS), bao gồm: nguyên lý hoạt động, ưu nhược điểm, và cách thức ứng dụng BeFS trong bài toán tìm kiếm đường đi ngắn nhất.

*¨  Hoạt động/Nội dung 2:* Xây dựng mô hình bản đồ và dữ liệu đường đi

Mục tiêu/chuẩn đầu ra: Sinh viên có khả năng thiết kế một bản đồ (dạng đồ thị), mô tả các địa điểm và khoảng cách giữa chúng. Từ đó xây dựng dữ liệu đầu vào cho thuật toán BeFS hoạt động.

*¨  Hoạt động/Nội dung 3:* Triển khai thuật toán BeFS tìm đường đi và đánh giá kết quả

Mục tiêu/chuẩn đầu ra: Sinh viên có khả năng cài đặt thuật toán BeFS bằng ngôn ngữ lập trình phù hợp (ví dụ: Python, C++). Sau đó, chạy thuật toán trên tập dữ liệu bản đồ, phân tích kết quả tìm được và đánh giá hiệu quả của thuật toán.

3. Sản phẩm nghiên cứu

Một hệ thống tìm kiếm đường đi trên bản đồ sử dụng thuật toán BeFS, bao gồm:

-  Chương trình thực thi thuật toán BeFS tìm đường đi ngắn nhất.

-  Bộ dữ liệu bản đồ đã chuẩn hóa (vị trí các tỉnh và khu vực có thể chọn ).

-  Báo cáo mô tả quy trình thực hiện, phân tích kết quả, và những khó khăn gặp phải trong quá trình phát triển hệ thống.

**III. Nhiệm vụ học tập**

1. Hoàn thành Tiểu luận, Bài tập lớn, Đồ án/Dự án theo đúng thời gian quy định (từ ngày…./…../20….đến ngày……/……../20…….)

2. Báo cáo sản phẩm nghiên cứu theo chủ đề được giao trước giảng viên và những sinh viên khác

**IV. Học liệu thực hiện Tiểu luận, Bài tập lớn, Đồ án/Dự án**

1. Tài liệu học tập:………………………………………………………

2. Phương tiện, nguyên liệu thực hiện Tiểu luận, Bài tập lớn, Đồ án/Dự án (nếu có):……………………………………………………………………….

MỤC LỤC

[PHIẾU HỌC TẬP CÁ NHÂN/NHÓM 2](#_Toc200562394)

[MỤC LỤC 4](#_Toc200562395)

[DANH SÁCH HÌNH ẢNH 6](#_Toc200562396)

[LỜI NÓI ĐẦU 7](#_Toc200562397)

[MỞ ĐẦU 8](#_Toc200562398)

[CHƯƠNG I : CƠ SỞ LÍ THUYẾT 10](#_Toc200562399)

[1.1. Bài toán tìm đường đi trong Trí tuệ nhân tạo 10](#_Toc200562400)

[1.1.1. Tổng quan về bài toán 10](#_Toc200562401)

[1.1.2. Một số kỹ thuật giải thường dùng 10](#_Toc200562402)

[1.1.3. Hạn chế của các phương pháp truyền thống 10](#_Toc200562403)

[1.2. Giới thiệu thuật toán tìm kiếm tổng quát 11](#_Toc200562404)

[1.2.1. Khái niệm của thuật toán 11](#_Toc200562405)

[1.2.2. Một số bước cơ bản 11](#_Toc200562406)

[1.2.3. Một số thuật toán tìm kiếm cơ bản 11](#_Toc200562407)

[1.3. Nguyên lý thuật toán Best First Search (BeFS) 12](#_Toc200562408)

[1.3.1. Khái niệm BeFS 12](#_Toc200562409)

[1.3.2. Nguyên lý hoạt động 13](#_Toc200562410)

[1.4. Hàm Heuristic và vai trò trong Best First Search. 13](#_Toc200562411)

[1.4.1. Khái niệm hàm heuristic 13](#_Toc200562412)

[1.4.2. Đặc điểm của hàm heuristic 14](#_Toc200562413)

[1.4.3. Vai trò then chốt của hàm heuristic trong BeFS 14](#_Toc200562414)

[1.5. So sánh BeFS với BFS, DFS, A\* 15](#_Toc200562415)

[CHƯƠNG II : THỰC HIỆN BÀI TOÁN 18](#_Toc200562416)

[2.1. Mô tả bài toán thực nghiệm 18](#_Toc200562417)

[2.2. Cài đặt mô hình 18](#_Toc200562418)

[2.2.1. Công cụ và thư viện sử dụng 18](#_Toc200562419)

[2.2.2. Biểu diễn bản đồ dưới dạng đồ thị 19](#_Toc200562420)

[2.3. Cấu trúc dữ liệu sử dụng 20](#_Toc200562421)

[2.4. Triển khai thuật toán Best First Search 21](#_Toc200562422)

[2.4.1. Công thức Haversine 21](#_Toc200562423)

[2.4.2. Hàm heuristic 22](#_Toc200562424)

[2.4.3. Thuật toán Best-First Search 22](#_Toc200562425)

[CHƯƠNG III : KẾT QUẢ BÀI TOÁN 26](#_Toc200562426)

[3.1. Dữ liệu đầu vào/ra 26](#_Toc200562427)

[3.1.1. Cách sử dụng 26](#_Toc200562428)

[3.1.2. Ví dụ minh họa cụ thể 26](#_Toc200562429)

[3.2. Phân tích đường đi chi tiết 26](#_Toc200562430)

[3.3. Những ưu điểm và hạn chế 28](#_Toc200562431)

[KẾT LUẬN 29](#_Toc200562432)

[TÀI LIỆU THAM KHẢO 31](#_Toc200562433)

DANH SÁCH HÌNH ẢNH

[Hình 1.1: Ảnh minh họa thuật toán Best First Search 13](#_Toc200552565)

[Hình 1.2: Phương pháp suy nghiệm - heuristic function 14](#_Toc200552566)

[Hình 1.3: So sánh các thuật toán tìm đường đi 17](#_Toc200552567)

[Hình 2.1. Ảnh minh họa xây dựng dictionary PROVINCES 19](#_Toc200552568)

[Hình 2.2. Ảnh minh họa xây dựng dictionary COORDINATES 19](#_Toc200552569)

[Hình 2.3: Hình ảnh file SVG bản đồ Việt Nam 20](#_Toc200552570)

[Hình 2.4: Công thức Haversine 21](#_Toc200552571)

[Hình 2.5. Ảnh minh họa hàm tính khoảng cách 2 điểm 22](#_Toc200552572)

[Hình 2.6. Ảnh minh họa hàm dùng công thức heuristic 22](#_Toc200552573)

[Hình 2.7. Áp dụng thuật toán Best First Search 25](#_Toc200552574)

[Hình 3.1. Ảnh minh họa giao diện người dùng khi chọn tỉnh đi và đến 26](#_Toc200552575)

[Hình 3.2: Bảng phân tích đường đi 27](#_Toc200552576)

LỜI NÓI ĐẦU

Trong bối cảnh chuyển đổi số đang diễn ra sâu rộng trên toàn cầu, các giải pháp công nghệ thông minh ngày càng được ứng dụng mạnh mẽ vào đời sống và sản xuất. Một trong những lĩnh vực được quan tâm đặc biệt là các hệ thống tìm đường tự động – đóng vai trò then chốt trong giao thông thông minh, robot tự hành, trò chơi điện tử, và các ứng dụng bản đồ trực tuyến. Việc tìm kiếm lộ trình tối ưu từ một điểm xuất phát đến đích không chỉ giúp tiết kiệm thời gian, chi phí mà còn nâng cao hiệu quả vận hành trong nhiều hệ thống phức tạp. Để giải quyết bài toán này, các thuật toán tìm kiếm đóng vai trò trung tâm, trong đó thuật toán Best-First Search (BeFS) nổi bật bởi khả năng định hướng tìm kiếm nhanh chóng và hiệu quả dựa trên hàm đánh giá heuristic.

Xuất phát từ tầm quan trọng đó, chúng em quyết định chọn đề tài "Áp dụng thuật toán BeFS vào tìm đường đi từ một điểm đến một điểm khác trong bản đồ" với mong muốn nghiên cứu và phát triển một mô hình mô phỏng quá trình tìm kiếm đường đi tối ưu trong môi trường bản đồ hai chiều. BeFS là một thuật toán tìm kiếm theo hướng tốt nhất, hoạt động dựa trên tiêu chí đánh giá trước để hướng thuật toán tới đích nhanh hơn so với các phương pháp tìm kiếm mù như BFS hay DFS. Với khả năng khai thác thông tin heuristic, BeFS cho thấy hiệu quả vượt trội trong việc rút ngắn thời gian tìm kiếm và giảm thiểu tài nguyên tính toán trong các hệ thống thực tế.

Mục tiêu của đề tài là xây dựng một hệ thống tìm đường tự động giữa hai điểm bất kỳ trên bản đồ, với độ chính xác cao và tốc độ xử lý tối ưu. Ứng dụng tiềm năng của hệ thống có thể mở rộng trong các lĩnh vực như lập kế hoạch lộ trình cho robot, hỗ trợ điều hướng trong đô thị, mô phỏng hành vi di chuyển trong trò chơi, và phát triển bản đồ số thông minh.

Với đề tài này, chúng em mong muốn góp phần vào việc nghiên cứu và phát triển các hệ thống điều hướng thông minh, từ đó mở ra hướng tiếp cận mới trong các ứng dụng tối ưu hóa đường đi. Hy vọng kết quả nghiên cứu không chỉ mang lại giá trị học thuật mà còn có thể triển khai thực tiễn trong các sản phẩm và dịch vụ công nghệ hiện đại.

Trong quá trình làm bài tập lớn này, chúng em xin chân thành cảm ơn Th.s Mai Thanh Hồng- giảng viên môn học Trí tuệ nhân tạo đã tận tình hướng dẫn và chỉ bảo cho chúng em trong suốt quá trình làm đề tài này, giúp cho đề tài của chúng em được hoàn thiện hơn, tốt hơn.

MỞ ĐẦU

**1. Lý do chọn đề tài**

Trong thời đại công nghệ số phát triển mạnh mẽ, việc tìm kiếm và định hướng đường đi nhanh chóng, chính xác đang trở thành nhu cầu thiết yếu trong nhiều lĩnh vực như giao thông thông minh, robot tự hành, các trò chơi điện tử và các hệ thống bản đồ số. Các giải pháp tìm đường tự động không chỉ góp phần tiết kiệm thời gian, chi phí mà còn nâng cao hiệu quả hoạt động của các hệ thống phức tạp. Trong số các thuật toán tìm kiếm, Best-First Search (BeFS) là một phương pháp nổi bật nhờ khả năng tận dụng thông tin heuristic để định hướng tìm kiếm một cách thông minh và hiệu quả. Nhận thấy được vai trò thực tiễn và tiềm năng của thuật toán BeFS, nhóm chúng em đã quyết định chọn đề tài “Áp dụng thuật toán BeFS vào tìm đường đi từ một điểm đến một điểm khác trong bản đồ” để nghiên cứu và thực nghiệm.

**2. Mục đích nghiên cứu**

* Nghiên cứu lý thuyết về thuật toán Best-First Search (BeFS), bao gồm nguyên lý hoạt động, ưu điểm, nhược điểm và ứng dụng thực tiễn.
* Xây dựng mô hình dữ liệu bản đồ dạng đồ thị để phục vụ cho việc tìm đường đi.
* Triển khai thuật toán BeFS trên dữ liệu bản đồ Việt Nam và đánh giá kết quả tìm kiếm.
* Mở rộng ứng dụng tiềm năng của thuật toán trong các lĩnh vực thực tế.

**3. Đối tượng nghiên cứu**

* Thuật toán tìm kiếm Best-First Search (BeFS).
* Dữ liệu bản đồ Việt Nam được biểu diễn dưới dạng đồ thị.
* Các công cụ lập trình và công nghệ hỗ trợ cài đặt và hiển thị kết quả (Python, Flask, HTML/CSS, SVG…).

**4. Phạm vi nghiên cứu**

* Nghiên cứu và triển khai trên dữ liệu bản đồ Việt Nam (các tỉnh/thành phố).
* Tập trung vào việc tìm kiếm đường đi giữa hai điểm bất kỳ dựa trên khoảng cách địa lý (không xét các yếu tố như tốc độ, thời gian thực tế).
* Đánh giá chủ yếu dựa trên tốc độ tìm kiếm và tính khả thi của mô hình.

**5. Kết quả mong muốn đạt được của đề tài**

* Hiểu rõ cơ chế hoạt động của thuật toán BeFS.
* Xây dựng một hệ thống tìm đường đi dựa trên thuật toán BeFS, hoạt động chính xác và nhanh chóng trên dữ liệu bản đồ Việt Nam.
* Đưa ra đánh giá khách quan về hiệu quả và hạn chế của BeFS trong thực tế.
* Đề xuất hướng phát triển, ứng dụng hoặc cải tiến thuật toán trong tương lai.

**6. Cấu trúc của báo cáo**

Báo cáo này được chia thành các phần chính:

* **Chương I**: Cơ sở lý thuyết – Giới thiệu các thuật toán tìm kiếm, nguyên lý hoạt động và vai trò của heuristic trong BeFS.
* **Chương II**: Thực hiện bài toán – Mô tả mô hình hóa dữ liệu, các bước triển khai thuật toán BeFS trên bản đồ Việt Nam.
* **Chương III**: Kết quả bài toán – Trình bày kết quả tìm kiếm và phân tích, ưu điểm và hạn chế của thuật toán.
* **Kết luận**: Tóm tắt kết quả đạt được, đưa ra nhận xét và định hướng phát triển trong tương lai.

CHƯƠNG I : CƠ SỞ LÍ THUYẾT

1.1. Bài toán tìm đường đi trong Trí tuệ nhân tạo

1.1.1. Tổng quan về bài toán

Trong những năm gần đây, trí tuệ nhân tạo (AI) đã và đang thay đổi mạnh mẽ cách con người sống và làm việc. Bài toán tìm đường đi (Pathfinding) là một trong những ứng dụng quan trọng của Trí tuệ nhân tạo, đặc biệt trong các lĩnh vực như robot tự hành, lập lịch giao thông, trò chơi điện tử và bản đồ số. Mục tiêu của bài toán là tìm ra lộ trình tối ưu (ngắn nhất, nhanh nhất, rẻ nhất...) từ điểm xuất phát đến điểm đích trong một không gian trạng thái (thường được mô hình hóa dưới dạng đồ thị).

1.1.2. Một số kỹ thuật giải thường dùng

Trong Trí tuệ nhân tạo , chúng ta có thể đã phát triển và áp dụng một vài các kỹ thuật và thuật toán vào đời sống thực tiễn , mỗi phương pháp cũng như kỹ thuật đều cho ta hướng giải quyết phù hợp cho các bài toán khác nhau nhằm tối ưu mục đích sử dụng.  Dưới đây là một số kỹ thuật giải tiêu biểu thường được sử dụng trong lĩnh vực Trí tuệ nhân tạo :

* Breadth-First Search (BFS): khám phá theo từng tầng, đảm bảo tìm đường ngắn nhất nếu đồ thị không có trọng số.
* Depth-First Search (DFS): khám phá sâu một nhánh trước, hiệu quả bộ nhớ nhưng không tối ưu.
* Uniform Cost Search (UCS): tìm đường tối ưu dựa trên chi phí thực tế.
* Best-First Search (BeFS): hướng tìm kiếm dựa trên hàm heuristic.
* A\*: kết hợp heuristic và chi phí thực, đảm bảo tối ưu nếu heuristic chấp nhận được.

1.1.3. Hạn chế của các phương pháp truyền thống

Các thuật toán hay phương pháp truyền thông tuy có thể sử dụng được và áp dụng dụng khá là rộng rãi và đã được tin dùng bởi nhiều tập đoàn , công ty lớn , tuy vậy trong nhiều trường hợp thì chúng vẫn có những mặt hạn chết nhất định khi phải giải quyết các bài toàn phức tạp hoặc không tối ưu khi sử dụng các thuật toán trên khiến cho bài toán có thời gian xử lý hay không gian trạng thái lớn hơn.

Các thuật toán trên chủ yếu có những hạn chế cụ thể sau :

* + Không sử dụng thông tin định hướng , các thuật toán như BFS hay DFS đi vào chiều sâu và không quan tâm đến vị trí đích nên thường tốn thời gian và tài nguyên.
  + Tốc độ giảm đáng kể khi gặp các bài toán lớn , đường đi hoặc trọng số của phần tử cần xử lý càng nhiều thì các thuật toán truyền thông thường trở nên chậm chạp do phải kiểm tra hầu hết các trạng thái.
  + Không tối ưu bộ nhớ , tốn tài nguyên đối với các đồ thị có phân nhánh lớn.
  + Thiếu sự linh hoạt trong môi trường thực tế , các thuật toán truyền thông không tận dụng được các thông tin bổ sung cũng như khoảng cách ước lượng (heuristic) , vốn rất hữu ích trong bài toán yêu cầu thời gian phản hồi nhanh và tiết kiệm tài nguyên

Vậy nên các thuật toán tìm kiếm có định hướng như BeFS hay A\* thường được áp dụng cho các hệ thống hiện đại nhờ khả năng linh hoạt , rút ngắn quá trình tìm kiếm đường đi cũng như hiệu quả nó mang lại.

1.2. Giới thiệu thuật toán tìm kiếm tổng quát

1.2.1. Khái niệm của thuật toán

Thuật toán tìm kiếm tổng quát là cách tiếp cận để tìm kiếm từ trạng thái ban đầu đến mục tiêu.

1.2.2. Một số bước cơ bản

B1. Đặt trạng thái ban đầu vào danh sách cần xét.

B2. Lặp:

* Lấy ra một trạng thái từ danh sách.
* Nếu đó là mục tiêu ➔ Thành công.
* Nếu không ➔ Mở rộng các trạng thái kế tiếp và thêm vào danh sách.

B3. Nếu hết danh sách mà không đạt mục tiêu ➔ Thất bại.

1.2.3. Một số thuật toán tìm kiếm cơ bản

1.2.3.1. Tìm kiếm theo chiều sâu (DFS)

Tìm kiếm theo chiều sâu (Depth-First Search - DFS) là một thuật toán cơ bản trong duyệt cây hoặc đồ thị. Nó hoạt động theo cách: đi càng sâu càng tốt trước khi quay lại (backtrack). DFS bắt đầu từ nút gốc (hoặc một nút bất kỳ), khám phá một nhánh sâu nhất trước, sau đó quay lại các nhánh khác.

1.2.3.2. Tìm kiếm theo chiều rộng (BFS)

Tìm kiếm theo chiều rộng (Breadth-First Search - BFS) là một thuật toán cơ bản trong duyệt cây hoặc đồ thị. Thuật toán này duyệt theo "lớp", nghĩa là thăm các nút gần trước, rồi mới đến các nút xa hơn.

* Bắt đầu từ một nút gốc (hoặc nút bất kỳ).
* Lần lượt thăm các nút kề (cấp độ 1), rồi đến cấp độ 2, cấp độ 3, ...
* Sử dụng hàng đợi (queue) để quản lý thứ tự thăm.

1.2.3.3. Tìm kiếm theo A\*

A\* là một thuật toán tìm kiếm đường đi rất mạnh và thông minh, thường được sử dụng trong game, robot tự hành, tìm đường GPS… Nó kết hợp giữa BFS (tìm theo chiều rộng) và tìm kiếm theo chi phí/heuristic để tìm đường đi ngắn nhất.

A\* tìm kiếm dựa trên công thức đánh giá:

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

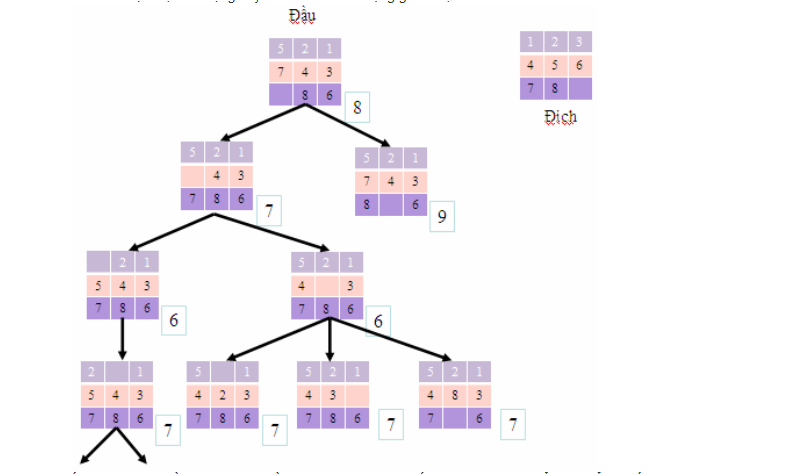
* g(n): Chi phí thực từ điểm bắt đầu đến điểm hiện tại.
* h(n): Heuristic - ước lượng khoảng cách từ điểm hiện tại đến đích (dự đoán).
* f(n): Tổng chi phí dự đoán để đến đích.

1.3. Nguyên lý thuật toán Best First Search (BeFS)

1.3.1. Khái niệm BeFS

Thuật toán Best First Search (BeFS) là một phương pháp tìm kiếm có định hướng, sử dụng thông tin heuristic để quyết định bước đi tiếp theo sao cho tiếp cận đích một cách “tốt nhất” tại mỗi thời điểm. Khác với các thuật toán tìm kiếm không định hướng như Breadth-First Search (BFS) hay Depth-First Search (DFS), BeFS không đơn thuần dựa vào độ sâu hay mức thứ tự duyệt mà ưu tiên những nút có khả năng dẫn đến đích sớm hơn, được đánh giá thông qua một hàm heuristic.

Nguyên lý cốt lõi của BeFS nằm ở việc sử dụng một hàm đánh giá (evaluation function), thường được ký hiệu là f(n), để xác định mức độ “hứa hẹn” của một nút n. Trong phiên bản đơn giản của BeFS (không xét đến chi phí đã đi), hàm này thường chính là h(n) – hàm heuristic ước lượng chi phí từ n đến đích. BeFS duyệt theo thứ tự các nút có giá trị h(n) thấp nhất trước, tức là càng gần đích theo đánh giá heuristic thì càng được ưu tiên.



Hình 1.1: Ảnh minh họa thuật toán Best First Search

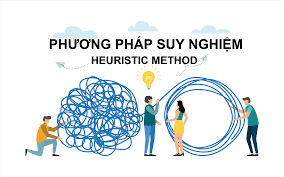
1.3.2. Nguyên lý hoạt động

1. Khởi tạo một hàng đợi ưu tiên (priority queue) chứa nút xuất phát.
2. Trong mỗi bước lặp:
   * Lấy nút n có giá trị h(n) nhỏ nhất ra khỏi hàng đợi.
   * Nếu n là nút đích → trả về lời giải.
   * Nếu không, mở rộng các nút con của n, tính toán h(n') cho từng nút con n', và đưa chúng vào hàng đợi.
3. Lặp lại cho đến khi hàng đợi rỗng hoặc tìm được nút đích.

1.4. Hàm Heuristic và vai trò trong Best First Search.

1.4.1. Khái niệm hàm heuristic

Hàm heuristic (ký hiệu: h(n)) là một thành phần cốt lõi của thuật toán BeFS. Đây là một hàm ước lượng chi phí từ một nút hiện tại n đến đích, mà không xét đến chi phí đã đi. Hàm này không đảm bảo chính xác tuyệt đối, nhưng nếu được thiết kế tốt, nó sẽ giúp BeFS dẫn đến lời giải một cách hiệu quả hơn.



Hình 1.2: Phương pháp suy nghiệm - heuristic function

1.4.2. Đặc điểm của hàm heuristic

* Không âm: Giá trị của h(n) ≥ 0 với mọi n.
* Nhanh: Tính toán h(n) cần nhanh và đơn giản.
* Xấp xỉ hợp lý: Càng gần giá trị thật của chi phí từ n đến đích thì càng tốt.

1.4.3. Vai trò then chốt của hàm heuristic trong BeFS

* Hướng dẫn tìm kiếm: Hàm heuristic cung cấp thông tin về "độ hứa hẹn" của mỗi nút trong việc dẫn đến mục tiêu. Bằng cách ưu tiên các nút có giá trị (h(n)) thấp hơn, BeFS cố gắng khám phá các con đường có khả năng dẫn đến đích nhanh hơn.
* Quyết định nút mở rộng: Tại mỗi bước, BeFS sử dụng giá trị heuristic để quyết định nút nào sẽ được lấy ra khỏi hàng đợi ưu tiên và mở rộng. Nút có giá trị (h(n)) nhỏ nhất được coi là "tốt nhất" và được chọn.
* Ảnh hưởng đến hiệu suất: Chất lượng của hàm heuristic có ảnh hưởng rất lớn đến hiệu suất của thuật toán BeFS:
  + Heuristic tốt: Một hàm heuristic chính xác và gần với chi phí thực tế có thể giúp BeFS tìm ra lời giải tối ưu (hoặc gần tối ưu) một cách nhanh chóng.
  + Heuristic kém: Một hàm heuristic không chính xác hoặc không cung cấp thông tin hữu ích có thể khiến BeFS đi lạc hướng, khám phá nhiều nút không cần thiết và thậm chí không tìm thấy lời giải (nếu không có cơ chế ngăn chặn vòng lặp).
* Tính chấp nhận được (Admissibility): Một hàm heuristic được gọi là chấp nhận được nếu nó không bao giờ đánh giá quá cao chi phí thực tế để đến mục tiêu từ bất kỳ nút nào. Tức là, h(n) <= h\*(n) với mọi nút (n), trong đó h\*(n) là chi phí tối ưu thực tế từ (n) đến mục tiêu. Mặc dù tính chấp nhận được không phải là yêu cầu bắt buộc đối với BeFS cơ bản, nhưng nó là một thuộc tính quan trọng khi so sánh với các thuật toán tìm kiếm khác như A\*.

1.5. So sánh BeFS với BFS, DFS, A\*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tiêu chí | Best First Search | Breadth-First Search (BFS) | Depth-First Search (DFS) | A\* |
| Mô tả cơ bản | Thuật toán tìm kiếm heuristic, ưu tiên mở rộng nút có giá trị heuristic thấp nhất (gần mục tiêu nhất). | Tìm kiếm theo chiều rộng, khám phá tất cả các nút ở cùng mức độ sâu trước khi đi sâu hơn. | Tìm kiếm theo chiều sâu, khám phá một nhánh đến độ sâu tối đa trước khi quay lại. | Kết hợp BFS và heuristic, sử dụng hàm đánh giá f(n) = g(n) + h(n) để tìm đường đi tối ưu. |
| Cơ chế hoạt động | - Sử dụng hàng đợi ưu tiên (priority queue). - Chọn nút có h(n) (heuristic) nhỏ nhất. - Không xem xét chi phí đường đi (g(n)). | - Sử dụng hàng đợi (queue). - Khám phá các nút theo thứ tự khoảng cách từ gốc (tăng dần). | - Sử dụng ngăn xếp (stack) hoặc đệ quy. - Khám phá nhánh sâu nhất trước. | - Sử dụng hàng đợi ưu tiên. - Chọn nút có f(n) = g(n) + h(n) nhỏ nhất (g(n): chi phí từ gốc, h(n): heuristic). |
| Độ phức tạp thời gian | O(b^m) trong trường hợp xấu nhất (b: độ phân nhánh, m: độ sâu tối đa). Tùy thuộc vào heuristic. | O(b^d) (d: độ sâu của giải pháp nông nhất). | O(b^m) trong trường hợp xấu nhất. | O(b^d) nếu heuristic tốt; có thể tệ hơn nếu heuristic không hiệu quả. |
| Độ phức tạp không gian | O(b^m) để lưu trữ hàng đợi ưu tiên. | O(b^d) để lưu trữ hàng đợi (các nút ở mức sâu nhất). | O(bm) để lưu trữ ngăn xếp (đường đi hiện tại). | O(b^d) để lưu trữ hàng đợi ưu tiên. |
| Tính tối ưu | Không đảm bảo (chỉ dựa vào heuristic, có thể bỏ qua đường đi tối ưu). | Đảm bảo tìm đường đi ngắn nhất trong đồ thị không trọng số. | Không đảm bảo (có thể tìm đường đi dài hơn). | Đảm bảo nếu h(n) là admissible (tìm đường đi có chi phí thấp nhất). |
| Heuristic | Yêu cầu hàm heuristic h(n) để ước lượng chi phí từ nút hiện tại đến mục tiêu. | Không sử dụng heuristic. | Không sử dụng heuristic. | Yêu cầu hàm heuristic h(n) admissible và lý tưởng là monotonic (consistent). |
| Ưu điểm | - Hiệu quả trong không gian trạng thái lớn nếu heuristic tốt. - Tìm giải pháp nhanh nếu heuristic chính xác. | - Đảm bảo tìm đường đi ngắn nhất trong đồ thị không trọng số. - Hoàn chỉnh trong không gian hữu hạn. | - Sử dụng ít bộ nhớ hơn BFS. - Hữu ích cho không gian trạng thái sâu. | - Tìm đường đi tối ưu với chi phí thấp nhất. - Hiệu quả hơn BFS nếu heuristic tốt. |
| Nhược điểm | - Không tối ưu, có thể bỏ qua đường đi tốt hơn. - Phụ thuộc nhiều vào chất lượng heuristic. | - Tiêu tốn nhiều bộ nhớ khi độ phân nhánh lớn. - Chậm trong không gian sâu. | - Không tối ưu, có thể bị kẹt trong nhánh vô hạn. - Không hoàn chỉnh nếu không kiểm soát vòng lặp. | - Phụ thuộc vào chất lượng heuristic. - Có thể tốn nhiều bộ nhớ hơn Best First Search. |
| Ứng dụng thực tế | - Tìm kiếm đường đi trong bản đồ (nếu không cần tối ưu). - Hệ thống AI cần tốc độ hơn tính chính xác. | - Tìm đường đi ngắn nhất trong mê cung. - Kiểm tra tính liên thông của đồ thị. - GPS không trọng số. | - Giải câu đố (như Rubik). - Tìm kiếm trong không gian trạng thái sâu. - Phát hiện chu trình trong đồ thị. | - Tìm đường đi tối ưu trong GPS. - Trò chơi AI (như cờ vua). - Quy hoạch đường đi robot. |
| Ví dụ minh họa | Tìm đường đi từ điểm A đến B dựa trên khoảng cách Euclidean (heuristic), nhưng có thể không chọn đường ngắn nhất. | Tìm đường đi ngắn nhất trong mê cung đơn giản, đảm bảo đường đi có ít bước nhất. | Khám phá một nhánh sâu trong cây trò chơi (như cờ vua) để tìm nước đi. | Tìm đường đi ngắn nhất từ thành phố A đến B, tính cả chi phí thực tế (khoảng cách, thời gian). |

Hình 1.3: So sánh các thuật toán tìm đường đi

=>Những lý do chọn thuật toán:

* Ưu tiên tốc độ tìm kiếm: Nếu mục tiêu chính của bạn là tìm ra một giải pháp nhanh chóng, ngay cả khi nó không đảm bảo là giải pháp tối ưu nhất, BeFS có thể là một lựa chọn tốt. Với một hàm heuristic tốt, BeFS có xu hướng đi thẳng đến đích hơn so với BFS hoặc DFS, giúp giảm đáng kể không gian tìm kiếm và thời gian thực hiện.
* Có thông tin heuristic hữu ích: BeFS đặc biệt hiệu quả khi bạn có một hàm heuristic có thể ước lượng khá chính xác chi phí còn lại để đạt được mục tiêu. Thông tin này giúp thuật toán đưa ra các quyết định "thông minh" về việc nút nào nên được khám phá tiếp theo.
* Bài toán không yêu cầu nghiệm tối ưu: Trong nhiều ứng dụng thực tế, việc tìm ra một giải pháp "đủ tốt" trong một khoảng thời gian chấp nhận được quan trọng hơn là tìm ra giải pháp tối ưu tuyệt đối. Ví dụ, trong các hệ thống gợi ý, tìm kiếm gần đúng có thể chấp nhận được.
* Không gian trạng thái lớn: Đối với các bài toán có không gian trạng thái quá lớn, BFS có thể tiêu tốn quá nhiều bộ nhớ để lưu trữ tất cả các nút ở mỗi cấp độ, và DFS có thể đi vào các nhánh vô hạn. BeFS, với sự hướng dẫn của heuristic, có thể tập trung vào các vùng có khả năng chứa nghiệm cao hơn, giúp giảm bớt gánh nặng về bộ nhớ và thời gian.

CHƯƠNG II : THỰC HIỆN BÀI TOÁN

2.1. Mô tả bài toán thực nghiệm

Bài toán đặt ra là: “Tìm đường đi ngắn nhất từ một tỉnh này đến một tỉnh khác trên bản đồ Việt Nam”, sử dụng thuật toán Best First Search (BeFS). Dữ liệu bản đồ được trừu tượng hóa thành một đồ thị, trong đó:

* Các đỉnh là các tỉnh/thành phố Việt Nam.
* Các cạnh là quan hệ kề nhau giữa các tỉnh (dựa trên địa lý thực).
* Chi phí heuristic là khoảng cách ước lượng theo công thức Haversine giữa hai tỉnh (tọa độ thật).

Trang web triển khai giải thuật này cho phép người dùng chọn tỉnh xuất phát và tỉnh đích, và hiển thị đường đi trực quan trên bản đồ SVG.

2.2. Cài đặt mô hình

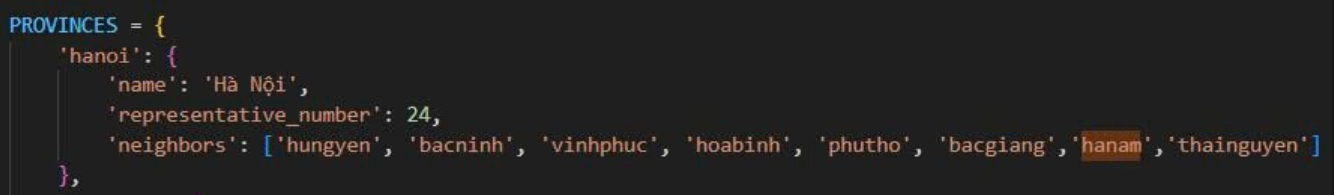
2.2.1. Công cụ và thư viện sử dụng

* **Ngôn ngữ lập trình chính: Python**
* Python có cú pháp ngắn gọn, dễ hiểu, phù hợp cho sinh viên và người mới bắt đầu phát triển các bài toán mô phỏng. Điều này giúp nhóm nhanh chóng nắm bắt, phát triển và mở rộng ứng dụng.
* Python xử lý tốt các tính toán số học (như công thức Haversine trong math), đồng thời hỗ trợ quản lý hàng đợi ưu tiên (PriorityQueue) hiệu quả – phù hợp để cài đặt thuật toán BeFS.
* **Môi trường phát triển**:
* Nhóm sử dụng Visual Studio Code để code và phát triển chương trình vì phần mềm này có thể biên dịch nhiều ngôn ngữ và có tích hợp git giúp quản lý dự án dễ hơn.
* Nhóm cũng sử dụng thêm Github và Render để xử lý và tạo trang web cho dự án
* **Các thư viện chính:**
* Flask: Framework web nhẹ, dùng để xây dựng các API và render template.
* os: Hỗ trợ thao tác với hệ điều hành, thư mục.
* math: Tính toán các giá trị toán học (công thức Haversine để tính khoảng cách).
* queue.PriorityQueue: Dùng để quản lý hàng đợi ưu tiên trong thuật toán BeFS.
* jsonify: Chuyển đổi dữ liệu Python sang JSON trả về cho client.

2.2.2. Biểu diễn bản đồ dưới dạng đồ thị

Bản đồ được biểu diễn bằng:

* Một dict tên là PROVINCES: mô tả danh sách tỉnh, mã hiệu, tên hiển thị, tỉnh kề.

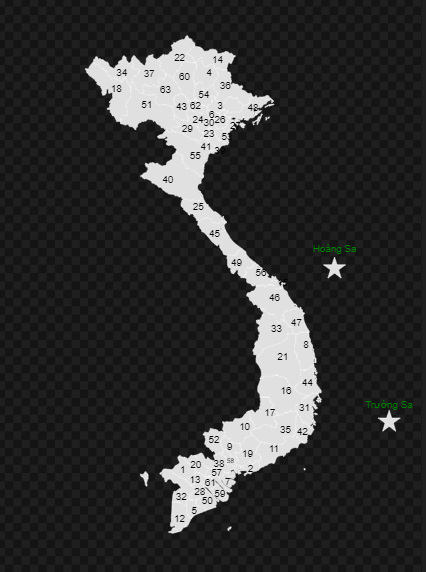


Hình 2.1. Ảnh minh họa xây dựng dictionary PROVINCES

* Một dict tên là COORDINATES: lưu tọa độ lat, lng thực tế của mỗi tỉnh.



Hình 2.2. Ảnh minh họa xây dựng dictionary COORDINATES

* File SVG bản đồ Việt Nam: Chứa hình ảnh bản đồ Việt Nam có thể tương tác được, mỗi số trên bản đồ (representative\_number) đại diện cho đỉnh của một tỉnh
* 

Hình 2.3: Hình ảnh file SVG bản đồ Việt Nam

Đây là dạng đồ thị có trọng số ngầm định, vì trọng số được tính bằng khoảng cách heuristic khi tìm kiếm.

2.3. Cấu trúc dữ liệu sử dụng

* dict: dùng để lưu danh sách tỉnh (PROVINCES) và tọa độ (COORDINATES).
* PriorityQueue: dùng trong thuật toán BeFS để ưu tiên nút có khoảng cách heuristic nhỏ nhất.
* set: lưu các tỉnh đã duyệt để tránh lặp.
* list: lưu đường đi hiện tại trong quá trình tìm kiếm.

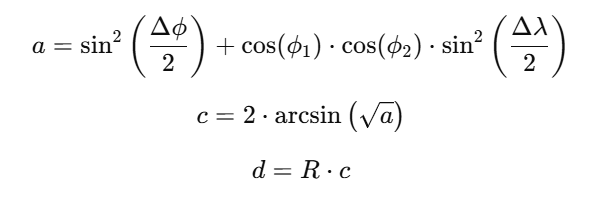
Ngoài ra, website sử dụng Flask để xử lý backend API, và frontend để hiển thị kết quả.

2.4. Triển khai thuật toán Best First Search

2.4.1. Công thức Haversine

Công thức Haversine được sử dụng để tính khoảng cách ngắn nhất giữa hai điểm trên bề mặt hình cầu

* + Giúp tính khoảng cách thực tế (theo mặt cong của Trái Đất) thay vì khoảng cách phẳng Euclidean.
  + Được sử dụng trong bản đồ số, GPS, ứng dụng tìm đường đi…
* Các ký hiệu sử dụng :
  + R = Bán kính trái đất (R = 6371 km)
  + ϕ1​,ϕ2: Vĩ độ (lat) của điểm xuất phát và kết thúc, tính bằng radian
  + λ1​,λ1: Kinh độ (lng) của điểm xuất phát và kết thúc, tính bằng radian
  + d: Khoảng cách giữa 2 điểm
* Công thức:

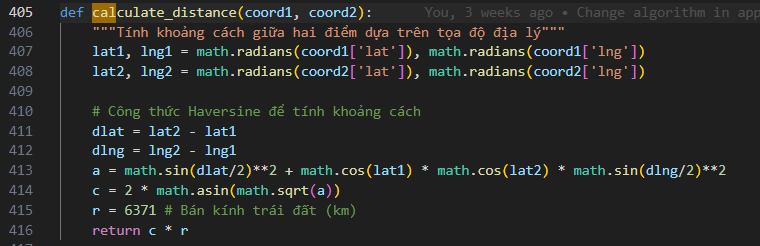


Hình 2.4: Công thức Haversine

Trong đó:

Δϕ=ϕ2​−ϕ1​

Δλ=λ2−λ1

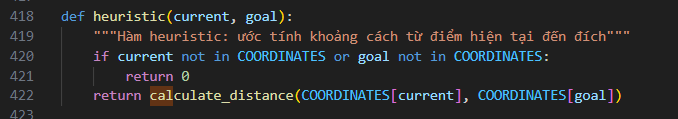


Hình 2.5. Ảnh minh họa hàm tính khoảng cách 2 điểm

2.4.2. Hàm heuristic

Heuristic là một hàm đánh giá độ "gần" của một tỉnh với đích đến. Trong thuật toán này, heuristic được tính bằng khoảng cách giữa tỉnh hiện tại và tỉnh đích (dựa vào tọa độ).

Nếu một trong hai tỉnh không có tọa độ, hàm sẽ trả về 0.



Hình 2.6. Ảnh minh họa hàm dùng công thức heuristic

2.4.3. Thuật toán Best-First Search

2.4.3.1. Ý tưởng của thuật toán:

* Bước 1: Khởi tạo một hàng đợi ưu tiên (Priority Queue), nơi các đường đi sẽ được xếp theo mức độ ưu tiên, tức là tỉnh nào có khoảng cách ước lượng đến đích nhỏ hơn sẽ được ưu tiên hơn.
* Bước 2: Đưa điểm bắt đầu (start) vào hàng đợi, đánh dấu là đã ghé thăm.
* Bước 3: Trong khi hàng đợi chưa rỗng:
  + Lấy ra đường đi có độ ưu tiên cao nhất (tức là gần đích nhất theo heuristic).
  + Nếu điểm cuối của đường đi này chính là điểm đến (end), thì trả về đường đi đó.
  + Nếu không, xét tất cả các tỉnh láng giềng của điểm hiện tại:
    - Nếu tỉnh đó chưa được thăm, tạo một đường đi mới bằng cách nối thêm tỉnh đó vào.
    - Tính khoảng cách heuristic từ tỉnh này đến đích.
    - Đưa đường đi mới này vào hàng đợi.
  + Nếu hàng đợi hết mà chưa đến được điểm đích, thì trả về không tìm thấy đường đi.

2.4.3.2. Giới thiệu các biến và vai trò

Để hiểu rõ hoạt động của thuật toán Best First Search, ta cần nắm được vai trò của các biến và cấu trúc dữ liệu chính trong chương trình:

* start, end (str): Là hai đầu vào đại diện cho tỉnh xuất phát và tỉnh đích mà người dùng yêu cầu tìm đường đi.
* queue (PriorityQueue): Hàng đợi ưu tiên dùng để lưu các đường đi đang được xét, với độ ưu tiên được xác định bởi chi phí ước lượng đến đích (heuristic).
* visited (set): Tập hợp lưu lại các tỉnh đã duyệt, nhằm tránh duyệt lại và ngăn thuật toán đi vòng.
* current\_cost (float): Chi phí ước lượng (độ ưu tiên) của đường đi hiện tại khi được lấy ra từ hàng đợi.
* path, new\_path (list): Danh sách tuyến đường đi qua các tỉnh tính đến hiện tại. new\_path là phiên bản cập nhật khi mở rộng thêm tỉnh kề.
* current (str): Tỉnh hiện tại đang được xét, là phần tử cuối của danh sách path.
* neighbor (str): Một tỉnh lân cận của current, được thuật toán kiểm tra để mở rộng đường đi.
* priority (float): Giá trị đánh giá mức độ gần đến đích, được tính bởi hàm heuristic(neighbor, end). Dùng để sắp xếp trong hàng đợi ưu tiên.

2.4.3.3. Phân tích thuật toán Best First Search

Sau khi hiểu các thành phần, ta xét chi tiết quá trình hoạt động của thuật toán:

**1. Khởi tạo ban đầu**

* Tạo một hàng đợi ưu tiên để lưu các đường đi, mỗi phần tử là cặp gồm độ ưu tiên và danh sách các tỉnh đã đi qua.
* Tạo một tập visited để lưu các tỉnh đã xét.
* Đưa tỉnh xuất phát vào hàng đợi với độ ưu tiên ban đầu là 0.

**2. Vòng lặp tìm kiếm**

* Thuật toán lặp lại cho đến khi hàng đợi rỗng:
  + Lấy ra đường đi có độ ưu tiên thấp nhất.
  + Xác định current là tỉnh cuối cùng trong đường đi hiện tại.
  + Nếu current trùng với end, thuật toán kết thúc và trả về kết quả.

**3. Mở rộng tỉnh lân cận**

* Nếu chưa đến đích:
  + Duyệt qua các tỉnh lân cận (neighbors) của current.
  + Với mỗi tỉnh chưa được duyệt:
    - Đánh dấu là đã duyệt.
    - Tạo đường đi mới bằng cách thêm tỉnh đó vào đường đi hiện tại.
    - Tính giá trị ưu tiên mới bằng hàm heuristic.
    - Đưa đường đi mới vào hàng đợi để xét tiếp.

**4. Trả về kết quả**

* Nếu tìm thấy đường đi, kết quả trả về ở dạng JSON gồm:
  + success: True
  + path: danh sách mã các tỉnh.
  + path\_names: danh sách tên tương ứng các tỉnh.
  + representative\_numbers: số đại diện của từng tỉnh.
* Nếu không có đường đi, trả về thông báo:
  + success: False
  + message: "Không tìm thấy đường đi"



Hình 2.7. Áp dụng thuật toán Best First Search

CHƯƠNG III : KẾT QUẢ BÀI TOÁN

3.1. Dữ liệu đầu vào/ra

3.1.1. Cách sử dụng

Đầu vào: người dùng chọn tỉnh bắt đầu và tỉnh đích từ danh sách dropdown.

Đầu ra: hiển thị tuyến đường được tìm bằng thuật toán BeFS trên bản đồ SVG Việt Nam. Ngoài ra, API còn trả về:

* + Danh sách các tỉnh đi qua (tên và số thứ tự).
  + Số tỉnh cần phải đi qua
  + Danh sách representative\_number để làm nổi bật trên bản đồ.

3.1.2. Ví dụ minh họa cụ thể

Giả sử cần tìm đường từ tỉnh "Hà Nội" đến "Hải Phòng".

Dữ liệu JSON trả về từ API sẽ như sau:

* Danh sách tên tỉnh: Hà Nội, Bắc Ninh, Hải Dương, Hải Phòng
* Số tỉnh cần phải đi qua: 3
* Danh sách mã đại diện: 24, 6, 26, 27



Hình 3.1. Ảnh minh họa giao diện người dùng khi chọn tỉnh đi và đến

3.2. Phân tích đường đi chi tiết

* Hệ thống sẽ lần lượt xét các tỉnh láng giềng của Hà Nội như Hưng Yên, Bắc Ninh, Hòa Bình, Vĩnh Phúc, v.v.
* Dựa vào hàm heuristic (khoảng cách địa lý), hệ thống thấy rằng Bắc Ninh gần Hải Phòng hơn nên chọn hướng đi qua Bắc Ninh.
* Tiếp tục từ Bắc Ninh, hệ thống xét các tỉnh láng giềng và chọn tiếp Hải Dương.
* Từ Hải Dương đến Hải Phòng là đường đi gần nhất, và vì thế sẽ được chọn tiếp.
* Kết quả cuối cùng là đường đi từ Hà Nội → Bắc Ninh → Hải Dương → Hải Phòng.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Bước** | **Đỉnh hiện tại** | **Đỉnh kề** | **Heuristic** | **OPEN** | **Đỉnh được chọn** | **CLOSED** |
| 0 | - | - | - | hanoi | - | - |
| 1 | Hà Nội | Bắc Ninh  Hưng Yên  Bắc Giang  Vĩnh Phúc  Hà Nam  Hòa Bình  Phú Thọ  Thái Nguyên | ~37.67 km  ~35.21 km  ~53.12 km  ~66.38 km  ~90.92 km  ~94.76 km  ~112.36 km  ~124.7 km | bacninh,  hungyen,  bacgiang, vinhphuc,  hanam,  hoabinh, phutho,  thainguyen | bacninh | hanoi |
| 2 | Bắc Ninh | Hải Dương | ~35.57 km | haiduong,  hungyen,  bacgiang,  vinhphuc,  hanam,  hoabinh, phutho,  thainguyen | haiduong | hanoi, bacninh |
| 3 | Hải Dương | Hải Phòng | ~0 km (đích) | haiphong, hungyen,  bacgiang, vinhphuc,  hanam  hoabinh,  phutho,  thainguyen | haiphong | hanoi, bacninh,  haiduong |
| 4 | Hải Phòng  (đích) | Không | - | - | - | hanoi, bacninh, haiduong, haiphong |

Hình 3.2: Bảng phân tích đường đi

3.3. Những ưu điểm và hạn chế

Ưu điểm:

* BeFS dùng hàm heuristic để hướng đến đích nhanh hơn, bỏ qua nhiều nhánh không tiềm năng.
* Hiệu quả khi heuristic chính xác.
* Chi phí tính toán thấp do chỉ dùng heuristic và không cần lưu nhiều trạng thái như A\*.

Hạn chế:

* Không đảm bảo tìm được đường đi ngắn nhất về tổng thể.
* Nếu heuristic sai lệch nhiều, thuật toán có thể chọn sai hướng và mất thời gian.
* Vì chỉ dựa vào hàm heuristic nên sẽ không tốt nếu muốn xét các yếu tố khác như tốc độ, thời gian, chi phí.

KẾT LUẬN

Trong nghiên cứu này, chúng em đã tìm hiểu chi tiết về thuật toán Best-First Search (BeFS), bao gồm nguyên lý hoạt động, cách đánh giá hàm heuristic và khả năng ứng dụng của thuật toán trong việc giải quyết bài toán tìm đường đi trên bản đồ.

Thuật toán BeFS đã được triển khai và áp dụng vào mô hình bản đồ để tìm đường đi ngắn nhất từ một điểm xuất phát đến một điểm đích, với các đặc điểm như tọa độ, kết nối giữa các nút, và chi phí di chuyển.

Thông qua quá trình xây dựng, cài đặt và kiểm thử, mô hình cho thấy:

* Khả năng tìm kiếm đường đi nhanh chóng và hiệu quả nhờ vào việc ưu tiên các nút có ước lượng chi phí thấp nhất.
* Tiết kiệm bộ nhớ hơn so với các thuật toán như A\* trong những trường hợp không cần đánh giá tổng chi phí chính xác.
* Dễ dàng mở rộng để áp dụng cho các hệ thống bản đồ số, định vị GPS, hoặc game.

Tuy nhiên, mô hình cũng có một số hạn chế nhất định:

* Chất lượng kết quả phụ thuộc vào hàm heuristic được chọn, nếu không hợp lý có thể dẫn đến kết quả không tối ưu.
* Không đảm bảo tìm ra đường đi tối ưu tuyệt đối nếu heuristic không thỏa mãn điều kiện đánh giá chặt.

Tổng thể, nghiên cứu đã đạt được mục tiêu đề ra ban đầu và khẳng định được tiềm năng ứng dụng của thuật toán BeFS trong các hệ thống dẫn đường và tìm đường tự động.

1. **Hướng phát triển tương lai của dự án**
   * Kết hợp thuật toán BeFS với các thuật toán khác như A\*, Dijkstra để cải thiện độ chính xác và tối ưu hóa chi phí tìm kiếm trong các bản đồ có cấu trúc phức tạp.
   * Thử nghiệm thuật toán trên các mô hình bản đồ thực tế với kích thước lớn và dữ liệu động (có thay đổi thời gian thực).
   * Xử lí được vấn đề người dùng muốn tìm đường đi qua nhiều tỉnh cùng một lúc
2. **Biện pháp ứng dụng vào thực tiễn:**
   * Phát triển ứng dụng dẫn đường sử dụng BeFS cho các hệ thống robot tự hành, xe không người lái hoặc bản đồ trong game.
   * Tích hợp vào hệ thống bản đồ số trong các ứng dụng du lịch, cứu hộ, giao thông nhằm cung cấp giải pháp tìm đường nhanh và hợp lý.
3. **Kiến nghị về cơ chế, chính sách:**
   * Đầu tư vào việc xây dựng các cơ sở dữ liệu bản đồ số chi tiết, mở và có tính cập nhật cao.
   * Khuyến khích và tài trợ các nghiên cứu ứng dụng trí tuệ nhân tạo và thuật toán tìm kiếm trong lĩnh vực giao thông thông minh, logistics và điều phối di chuyển.

Với kết quả đạt được và các đề xuất cụ thể, nghiên cứu này hy vọng sẽ đóng góp tích cực vào việc nâng cao hiệu quả của các hệ thống tìm đường thông minh, góp phần ứng dụng công nghệ vào đời sống hiện đại.

TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | *Trí tuệ nhân tạo*  Trường Đại học Công Nghiệp Hà Nội - NXB Giáo dục Việt Nam, 2012 |
| [2] | *Nguyễn Tấn Trần Minh Khang, Trí tuệ nhân tạo*  Các phương pháp và ứng dụng - NXB Đại học Quốc gia TP.HCM, 2016 |
| [3] | https://simplemaps.com/svg/country/vn |
| [4] | <https://labs.flinters.vn/algorithm/algorithm-cac-thuat-toan-tim-kiem-trong-ai/> |