**ĐẠI HỌC ĐÀ NẴNG**

**ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN CẤU TRÚC DỮ LIỆU & THUẬT TOÁN**

**Đề tài: Tìm đường đi ngắn nhất bằng thuật toán A\***

**SINH VIÊN THỰC HIỆN:**

**Hồ Phúc Hỷ LỚP: 20PFIEV3 NHÓM: 20.89**

**Đoàn Trọng Minh Hoàng LỚP: 20PFIEV3 NHÓM: 20.89**

**GIẢNG VIÊN HƯỚNG DẪN: TS. Đặng Thiên Bình**

**Đà Nẵng, 2022-2023**

# LỜI MỞ ĐẦU

Đầu tiên, chúng em xin gửi lời cảm ơn chân thành tới TS Đặng Thiên Bình đã giúp đỡ chúng em hoàn thành tốt đề tài thuật toán A\*. Trong quá trình hướng dẫn, thầy đã cung cấp cho chúng em những lý thuyết quan trọng và hỗ trợ tận tình trong việc phát triển kỹ năng lập trình, cũng như giải quyết các vấn đề trong đề tài.

Dù đã cố gắng hoàn thành đồ án trong phạm vi và khả năng của chúng em, chắc chắn sẽ không tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự thông cảm, góp ý và tận tình chỉ bảo từ quý Thầy Cô để chúng em có thể tiếp tục hoàn thiện và nâng cao chất lượng công trình nghiên cứu của mình.

**Hồ Phúc Hỷ**

**Đoàn Trọng Minh Hoàng**

**MỤC LỤC**

[1](#_Toc137154783)

[LỜI MỞ ĐẦU 3](#_Toc137154784)

[DANH MỤC HÌNH VẼ 6](#_Toc137154785)

[1. GIỚI THIỆU ĐỀ TÀI 7](#_Toc137154786)

[2. PHÂN TÍCH CHỨC NĂNG HỆ THỐNG 7](#_Toc137154787)

[2.1. Sơ đồ Use Case 7](#_Toc137154788)

[2.2. Biểu đồ hành động 8](#_Toc137154789)

[3. THIẾT KẾ CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN 8](#_Toc137154790)

[3.1. Phát biểu bài toán 8](#_Toc137154791)

[3.2. Phân tích và ứng dụng cấu trúc dữ liệu trong hệ thống 9](#_Toc137154792)

[3.2.1. Khoảng cách Manhattan 9](#_Toc137154793)

[3.2.2. Tìm kiếm A\* 10](#_Toc137154794)

[4. KẾT QUẢ ĐẠT ĐƯỢC 12](#_Toc137154795)

[4.1. Giao diện chính của chương trình 12](#_Toc137154796)

[4.2. Kết quả thực thi của chương trình 13](#_Toc137154797)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 14](#_Toc137154798)

[a. Kết luận 14](#_Toc137154799)

[b. Hướng phát triển 15](#_Toc137154800)

[TÀI LIỆU THAM KHẢO 16](#_Toc137154801)

**DANH MỤC HÌNH VẼ**

[HÌNH 1. Sơ đồ Use Case 6](#_Toc137157599)

[HÌNH 2. Biểu đồ hành động 7](#_Toc137157600)

[HÌNH 3. Cấu trúc dữ liệu 8](#_Toc137157601)

[HÌNH 4. Sơ đồ khối thuật toán 10](#_Toc137157602)

[HÌNH 5. Giao diện ban đầu 11](#_Toc137157603)

[HÌNH 6. Giao diện độ khó thứ nhất 11](#_Toc137157604)

[HÌNH 7. Giao diện độ khó thứ hai 12](#_Toc137157605)

[HÌNH 8. Giao diện độ khó thứ ba 12](#_Toc137157606)

[HÌNH 9. Giao diện độ khó khi đã hoàn thành xong 13](#_Toc137157607)

1. **GIỚI THIỆU ĐỀ TÀI**

Bài toán tìm đường đi ngắn nhất là một bài toán quan trọng trong lĩnh vực khoa học máy tính và trí tuệ nhân tạo. Thuật toán A\* là một thuật toán tìm kiếm đường đi trong đồ thị từ một điểm xuất phát đến một điểm đích. Nó kết hợp giữa thuật toán tìm kiếm theo chiều rộng (BFS) và thuật toán tìm kiếm theo đường thẳng (Dijkstra) bằng cách sử dụng một hàm ước tính chi phí để định hướng tìm kiếm.

Nhóm của chúng em sẽ thực hiện bài toán tìm đường đi ngắn nhất giữa hai điểm bằng cách vận dụng thuật toán A\*

Đồ án này tập trung vào cài đặt thuật toán A\* và áp dụng nó để giải quyết bài toán tìm đường đi ngắn nhất.

1. **PHÂN TÍCH CHỨC NĂNG HỆ THỐNG**
   1. **Sơ đồ Use Case**

**A picture containing diagram, circle, text, line

Description automatically generated**

HÌNH 1. Sơ đồ Use Case

* 1. **Biểu đồ hành động**

**A picture containing text, screenshot, diagram, number

Description automatically generated**

HÌNH 2. Biểu đồ hành động

1. **THIẾT KẾ CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN**
   1. **Phát biểu bài toán**

Vì đây là chương trình tìm kiếm đường đi ngắn nhất, nên ban đầu, đầu vào chương trình sẽ là:

Một ma trận n x n

Điểm bắt đầu và điểm kết thúc

Các chướng ngại

A screenshot of a crossword puzzle

Description automatically generated

HÌNH 3. Cấu trúc dữ liệu

Từ các dữ liệu đầu vào như vậy, chương trình sẽ tìm ra đường đi ngắn nhất giữa hai điểm bắt đầu và kết thúc.

* 1. **Phân tích và ứng dụng cấu trúc dữ liệu trong hệ thống**
     1. **Khoảng cách Manhattan**

Vì chúng ta chỉ cho phép di chuyển theo bốn hướng duy nhất là phải, trái, dưới, lên nên chúng ta sẽ sử dụng hàm heuristic bằng khoảng cách Manhattan. [2]

    x1, y1 = p1

    x2, y2 = p2

    return abs(x1 - x2) + abs(y1 - y2)

* + 1. **Tìm kiếm A\***

Giải thuật:

Các bước tiến hành như sau:

1. Cho đỉnh xuất phát vào open

2. Nếu open rỗng thì tìm kiếm thất bại, kết thúc việc tìm kiếm

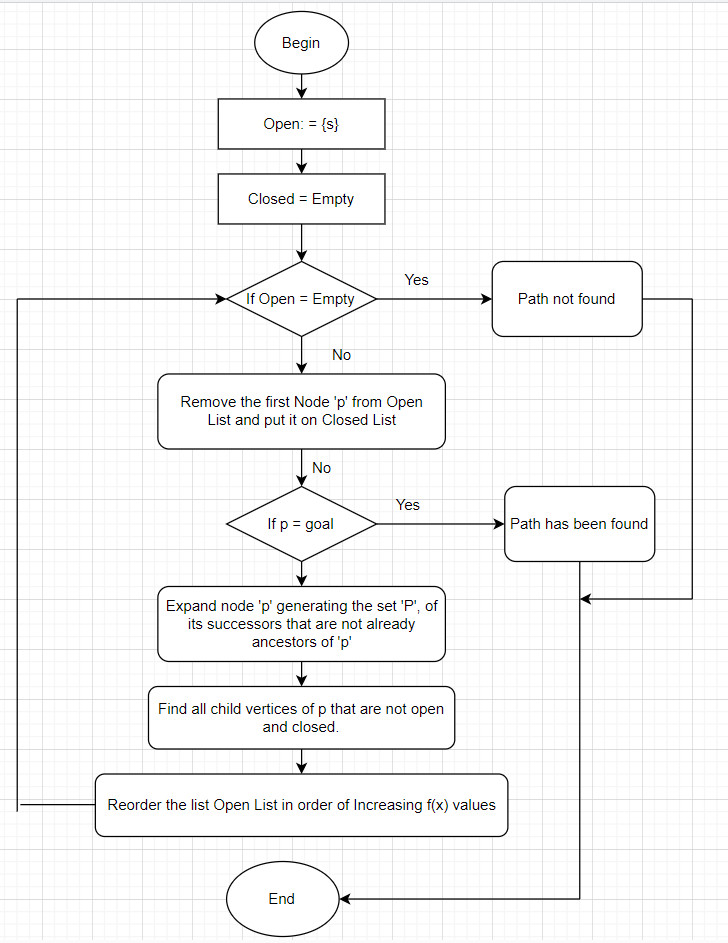
3. Lấy đỉnh đầu trong open ra và gọi đó là p. Cho p vào Closed

4. Nếu p là đỉnh thì tìm kiếm thành công, kết thúc việc tìm kiếm

5. Tìm kiếm tất cả đỉnh con của p không thuộc open và closed cho vào open theo thứ tự tăng dần đối với hàm f(x) = g(x) + h(x)

6. Trở lại bước 2.

Sơ đồ khối [3]



HÌNH 4. Sơ đồ khối thuật toán

Đặt giá trị ban đầu:

Count = 0

open\_set = PriorityQueue()

    open\_set.put((0, count, start))

    came\_from = {} # Lưu trữ các nút liền trước của mỗi nút

    g\_score = {spot: float("inf") for row in grid for spot in row}  # Lưu trữ giá trị g-score của mỗi nút

    g\_score[start] = 0 # G-score của điểm bắt đầu được đặt là 0

    f\_score = {spot: float("inf") for row in grid for spot in row} # Lưu trữ giá trị f-score của mỗi nút

    f\_score[start] = h(start.get\_pos(), end.get\_pos()) # F-score của điểm bắt đầu được tính toán

Thực hiện thuật toán: [1]

open\_set\_hash = {start} # Lưu trữ các nút đã được thêm vào hàng đợi ưu tiên

    while not open\_set.empty():

        current = open\_set.get()[2]  # Lấy nút có ưu tiên cao nhất từ hàng đợi ưu tiên

        open\_set\_hash.remove(current) # Xóa nút hiện tại khỏi tập hợp các nút đã thêm vào hàng đợi ưu tiên

        if current == end: # Nếu nút hiện tại là điểm đích

            reconstruct\_path(came\_from, end, draw) # Tái tạo đường đi từ điểm đích về điểm bắt đầu

            end.make\_end() # Đánh dấu điểm đích

            return True # Trả về True để cho biết đã tìm thấy đường đi

        for neighbor in current.neighbors: # Duyệt qua các nút láng giềng của nút hiện tại

            temp\_g\_score = g\_score[current] + 1 # Tính toán g-score tạm thời

            if temp\_g\_score < g\_score[neighbor]: # Nếu g-score tạm thời nhỏ hơn g-score của nút láng giềng

                came\_from[neighbor] = current # Cập nhật nút liền trước của nút láng giềng

                g\_score[neighbor] = temp\_g\_score # Cập nhật g-score của nút láng giềng

                f\_score[neighbor] = temp\_g\_score + h(neighbor.get\_pos(), end.get\_pos()) # Cập nhật f-score của nút láng giềng

                if neighbor not in open\_set\_hash: # Nếu nút láng giềng chưa có trong hàng đợi ưu tiên

                    count += 1

                    open\_set.put((f\_score[neighbor], count, neighbor))  # Thêm nút láng giềng vào hàng đợi ưu tiên

                    open\_set\_hash.add(neighbor) # Thêm nút láng giềng vào tập hợp các nút đã thêm vào hàng đợi ưu tiên

                    neighbor.make\_open() # Đánh dấu nút láng giềng là nút mở

        if current != start:

            current.make\_closed() # Đánh dấu nút hiện tại là nút đã xét

    return False # Trả về False nếu không tìm thấy đường đi

1. **KẾT QUẢ ĐẠT ĐƯỢC**
   1. **Giao diện chính của chương trình**

A screenshot of a computer

Description automatically generated

HÌNH 5. Giao diện ban đầu

A screenshot of a computer

Description automatically generated

HÌNH 6. Giao diện độ khó thứ nhất

A screenshot of a computer

Description automatically generated

HÌNH 7. Giao diện độ khó thứ hai

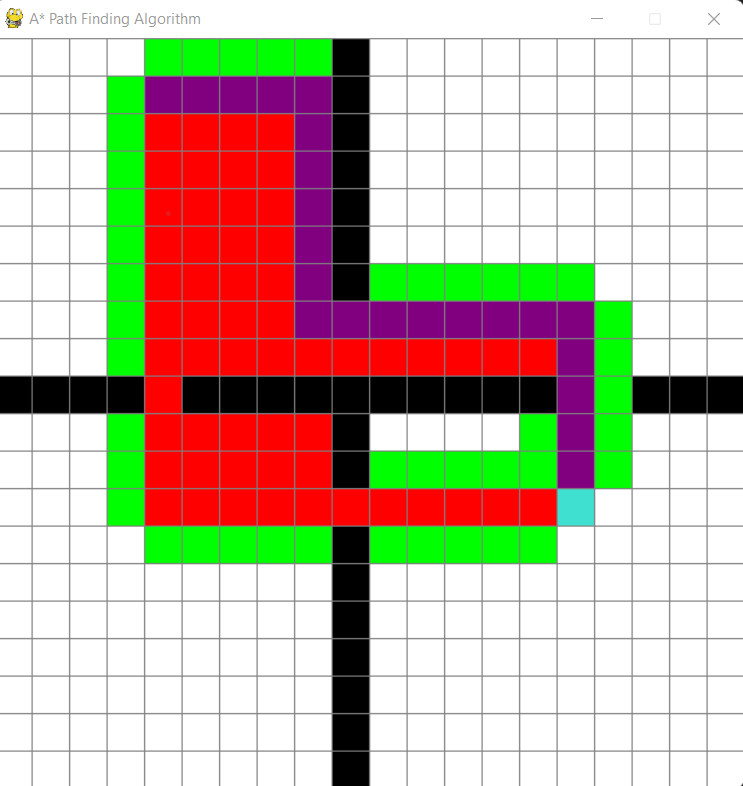
A screenshot of a computer

Description automatically generated

HÌNH 8. Giao diện độ khó thứ ba

* 1. **Kết quả thực thi của chương trình**

Khi chạy chương trình, chúng ta chọn một điểm bất kỳ là điểm bắt đầu với màu vàng và một điểm bất kỳ là điểm kết thúc với màu xanh. Sau đó, chúng ta bấm nút space để bắt đầu tìm đường đi ngắn nhất từ điểm bắt đầu đến điểm kết thúc. Đường màu đỏ là các đường mà có thể đi ở điểm bắt đầu để tới điểm kết thúc. Từ tất cả đường màu đỏ, chúng sẽ xét đường nào là ngắn nhất để đến đích và chuyển sang màu tím. Từ đó, chúng ta có thể tìm ra đường đi ngắn nhất.



HÌNH 9. Giao diện độ khó khi đã hoàn thành xong

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. **Kết luận**

Sau nhiều nỗ lực, cố gắng và sự hướng dẫn tận tâm của giảng viên, chúng em đã thành công trong việc hoàn thành chương trình với một mức độ hoàn thiện khá cao so với mục tiêu ban đầu. Tuy nhiên, sau khi xem xét kỹ lưỡng và so sánh với thực tế, chúng tôi đã rút ra được một số bài học quý giá và nhận thức được những ưu điểm và hạn chế của chương trình.

Chương trình của chúng em có một số ưu điểm đáng kể:

Hoàn thiện đáng kể: Chương trình đã được phát triển với mức độ hoàn thiện khá cao. Nó có khả năng tìm kiếm đường đi từ điểm bắt đầu đến điểm kết thúc bằng thuật toán A\* và hiển thị kết quả trực quan trên giao diện người dùng.

Tương tác người dùng: Chương trình cho phép người dùng tương tác trực tiếp thông qua giao diện người dùng. Người dùng có thể chọn điểm bắt đầu, điểm kết thúc và vẽ các rào cản trên lưới để kiểm tra thuật toán A\*.

Dễ dàng sử dụng: Giao diện người dùng thân thiện và trực quan, cho phép người dùng dễ dàng tương tác và điều chỉnh các tham số. Chương trình cung cấp các chức năng cơ bản và dễ hiểu, giúp người dùng dễ dàng sử dụng mà không cần có kiến thức chuyên sâu về thuật toán.

Tuy nhiên, chúng em cũng nhận ra rằng chương trình của chúng tôi vẫn còn một số hạn chế. Ví dụ, trong quá trình tìm hiểu và triển khai thuật toán A\*, chúng em chưa thể tận dụng tối đa các phương pháp cải tiến như pruning, caching và parallelization để tăng hiệu suất và tốc độ tìm kiếm.

1. **Hướng phát triển**

Tuy nhiên vẫn còn một ba điểm cần khắc phục ở trong chương trình.

Một là, cần phải thêm Sự kiện “dừng” để có thể dừng bài toàn tìm kiếm lúc nào.

Hai là, các kỹ thuật như pruning, caching và parallelization có thể được áp dụng để tăng tốc độ tìm kiếm và giảm thời gian xử lý.

Ba là, loại bỏ những vật cản khi cần chứ không nhất thiết phải thoát chương trình để khởi động lại.

# TÀI LIỆU THAM KHẢO

[1] <https://www.youtube.com/watch?v=JtiK0DOeI4A&t=4613s>: A\* Pathfinding Visualization Tutorial - Python A\* Path Finding Tutorial

[2] <https://www.geeksforgeeks.org/a-search-algorithm/>

[2] 5645

Trajectory Planning of a Mobile Robot using

Enhanced A-Star Algorithm

Priyanka Sudhakara and Velappa Ganapathy\*

[3] Trajectory Planning of a Mobile Robot Using Enhanced A-Star Algorithms by Priyanka Sudhakara and Velappa Ganapathy