**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**~~~~~\*~~~~~**

Logo

Description automatically generated

**BÁO CÁO TIỂU LUẬN CUỐI KỲ**

Học phần: Lập trình nâng cao

**Đề tài:**

Sử dụng thuật toán A\* xử lý bài toán “Tìm đường đi ngắn nhất”

***Nhóm thực hiện :*** Nhóm 02

***Các thành viên :*** 1. Dương Thành Nam (nhóm trưởng)

2. Đỗ Đức Thuận

3. Trần Duy Hiệp

**Hà Nội, ngày 08 tháng 12 năm 2022**

**MỤC LỤC**

**LỜI MỞ ĐẦU**

**I. Giới thiệu bài toán**

**II. Giới thiệu thuật toán A\* (A-star)**

1. Thuật toán A\* là gì?

2. Heuristic là gì?

**III. Mô tả và triển khai thuật toán A\*(A-star)**

1. Mã giả giải thuật A\*

2. Mô phỏng trên đồ thị

3. Triển khai thuật toán A\* trong lập trình Java

4. Đánh giá độ phức tạp thuật toán

**IV. Ưu, nhược điểm của thuật toán A\* (A-star)**

1. Ưu điểm

2. Nhược điểm

**V. So sánh thuật toán A\* (A-star) và thuật toán tìm đường khác (Dijkstra)**

1. Thuật toán Dijkstra

2. Thuật toán A-star

3. Dijkstra có là trường hợp đặc biệt của A\* không?

**LỜI MỞ ĐẦU**

Trong thực tế hiện nay, tối ưu hóa quãng đường di chuyển để tiết kiệm chi phí và thời gian là vấn đề luôn luôn được đặt lên hàng đầu trong nhiều lĩnh vực, đặt ra rất nhiều bài toán cần xử lý. Tuy nhiên các thuật toán tìm kiếm hiện tại như chúng ta được biết qua quá trình học các thuật toán như là Dijkstra chỉ xây dựng các tuyến đường có thể dẫn đến đích bằng cách xét tất cả những nút trong đồ thị, chưa tối ưu được quãng đường di chuyển. Chính vì thế, qua tìm hiểu và phân tích kỹ lưỡng, nhóm 02 lựa chọn thuật toán A\* (A sao) để giải quyết một phần vấn đề đó cũng như làm đề tài chính cho dự án cuối kỳ của nhóm.

Dưới đây, nhóm 02 xin trình bày nội dung bản báo cáo dự án cuối kỳ học phần Lập trình nâng cao: “Sử dụng thuật toán A\* xử lý bài toán Tìm đường đi ngắn nhất“.

**I. Giới thiệu bài toán**

Hiện nay, tại các thành phố lớn hay những khu đô thị - những nơi có dân cư sinh sống với mật độ lớn, cơ sở hạ tầng và giao thông ngày càng phát triển kéo theo nhiều vấn đề cần giải quyết như: tắc đường, cơ sở hạ tầng xuống cấp,… Trong những năm vừa qua, đối với người dân trên thế giới nói chung và Việt Nam nói riêng, những giải pháp để khắc phục vấn đề tắc đường luôn được ưu tiên đặt lên hàng đầu. Vấn đề ấy không còn quá lạ đối với mỗi chúng ta nhưng trong thời gian ngắn trở lại đây, tình hình thế giới biến động khiến cho giá xăng, dầu đột ngột tăng mạnh. Chi phí chính là nỗi lo lắng cho mỗi người dân về phương tiện di chuyển. Chính vì điều đó, việc tìm con đường đi ngắn nhất, tiết kiệm thời gian và chi phí đi lại giờ đây lại chính là ưu tiên số một và đặt ra cho chúng ta một bài toán cần giải quyết. Nhóm 02 sẽ dùng thuật toán A\* qua ngôn ngữ Python để giải quyết một phần của vấn đề này (có minh họa thuật toán)

**II. Giới thiệu thuật toán A\* (A-star)**

Năm 1964, Nils Nilsson phát minh ra một phương pháp tiếp cận dựa trên khám phá để tăng tốc độ của thuật toán Dijkstra. Thuật toán này được gọi là A1. Năm 1967, Bertram Raphael đã cải thiện đáng kể thuật toán này, nhưng không thể hiển thị tối ưu và đặt tên thuật toán này là A2.

Năm 1968, Peter E. Hart đã giới thiệu một đối số chứng minh thuật toán A2 của ông khi sử dụng thuật toán này với một đánh giá heuristic thích hợp sẽ thu được kết quả tối ưu. Chứng minh của ông về thuật toán cũng bao gồm một phần cho thấy rằng các thuật toán A2 mới là thuật toán tốt nhất có thể được đưa ra các điều kiện. Do đó, ông đặt tên cho thuật toán mới là A\* (A sao, A-star).

**1. Thuật toán A\* là gì?**

Trong khoa học máy tính, A\* (đọc là A sao) là thuật toán tìm kiếm trong đồ thị. Thuật toán này tìm đường đi từ một nút khởi đầu tới một nút đích cho trước. Thuật toán này sử dụng một đánh giá heuristic để đánh giá từng nút theo ước lượng về tuyến đường đi tốt nhất đi qua nút được xét. Thuật toán A\* duyệt các nút theo thứ tự đánh giá của heuristic.

**2. Heuristic là gì?**

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

Heuristic được tính toán bằng hàm Heuristic - hàm ứng với mỗi trạng thái hay mỗi sự lựa chọn một giá trị ý nghĩa của vấn đề. Dựa vào giá trị hàm này, ta thực hiện lựa chọn hành động.

Tổng chi phí của một nút (node) được thể hiện qua hàm f(x) được tính như sau:

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

***Trong đó:***

●      g(x) là hàm chi phí của đường đi cho đến thời điểm hiện tại, nghĩa là tổng trọng số của các cạnh đã đi qua.

●      h(x) là hàm đánh giá Heuristic về chi phí nhỏ nhất để đi đến đích từ đỉnh x (hiện tại).

●      f(x) thường có giá trị càng thấp thì độ ưu tiên càng cao.

**III.** **Mô tả và triển khai thuật toán A\*(A-star)**

**1. Mã giả giải thuật A\***

Tạo một danh sách C là tập hợp nodes đã được xét đến

Tạo một danh sách O là tập hợp nodes chưa  được xét đến và giá trị f tương ứng

**while** tập O không rỗng:

chọn 1 node n f từ O với giá trị f tốt nhất

**if** n là target node:

**return** tìm được đường đi

**for** toàn bộ m là neighbor của n:

**if** (m không thuộc tập C) **and** (m không thuộc tập O):

         thêm m vào O,  thiết lập n là cha của m

         tính g(m) và f(m) đồng thời lưu lại

**else :**

**if** f(m) từ lần lặp cuối cùng tốt hơn g(m) ở lần lặp hiện tại :

             thiết lập n là cha của m

             cập nhật g(m) và f(m)

**if** m thuộc tập C :

                 chuyển m vào tập O

chuyển n từ tập O đến tập C

**return** không tìm được đường đi

**2. Mô phỏng thuật toán trên đồ thị**

**Bài toán:**Tìm đường đi ngắn nhất từ A đến K. ***Biết:*** h(A) = 60 / h(B) = 53 / h(C) = 36 / h(D) = 35 / h(E) = 35 / h(F) = 19 / h(G) = 16 / h(H) = 38 / h(I) = 23 / h(J) = 0 / h(K) = 7

*(h(x) là hàm đánh giá Heuristic về chi phí nhỏ nhất để đi đến đích từ đỉnh x (hiện tại)).*

**Giải thuật:**

- Đỉnh bắt đầu: **A**

- Đỉnh kết thúc: **K**

Ước lượng chi phí từ đỉnh đầu cho đến đỉnh kết thúc khi qua đỉnh hiện tại, trong đó g là chi phí từ đỉnh đầu đến đỉnh hiện tại.

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

**Bước 0:** Đỉnh bắt đầu là đỉnh A, thêm A vào Open

Open = {**A**}

Close = {}

**Bước 1:** Đỉnh đang xét: **A**

f(A) = g(A) + h(A) = 0 + 60 =60

Đỉnh lân cận của đỉnh đang xét: B và H => Thêm B và H vào Open.

Open = {A, B, H}

f(B) = g(B) + h(B)= 11+53 = 64

f(H) = g(H)+ h(H) = 15+38 = 53

Loại A khỏi Open và thêm A vào Close.

Close = {A}

Có f(H) < f(B) => Đỉnh xét tiếp theo p = H

**Bước 2:**  Đỉnh đang xét: **H**

f(H) = 53

Đỉnh lân cận của đỉnh đang xét: G, I và A => Thêm G và I vào Open.

Open = {B, H, G, I}

f(G) = g(G) + h(G)= + 16= 34

f(I) = g(I)+ h(I) = + 23 = 45

Bỏ H khỏi Open và thêm H vào Close.

Close = {A, H}

Có f(G) < f(I) < f(B) => Đỉnh xét tiếp theo p = G

**Bước 3:** Đỉnh đang xét: **G**

f(G) = 34

Đỉnh lân cận của đỉnh đang xét: H, -K, F => Thêm K, F vào Open.

Open = {B, G, I, K, F}

f(K) = g(K) + h(K)= 25 + 7 = 32

f(F) = g(F)+ h(F) =34 + 19 = 53

Loại G khỏi Open và thêm G vào Close.

Close = {A, H, G}

Có f(K) < f(I) < f(F) < f(B) => Đỉnh xét tiếp theo p = K

**Bước 4:**  Đỉnh đang xét: **K**

=> Đến K (target) => **Dừng thuật toán.**

**3. Triển khai thuật toán A\* trong lập trình Python**

Cài đặt thư viện Pygame cho phép sử dụng các tính năng của nó để thực thi chương trình.

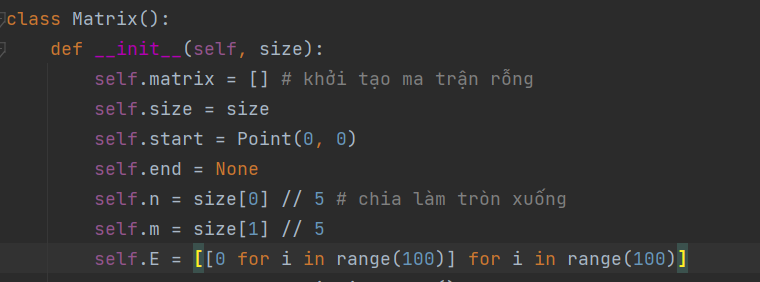
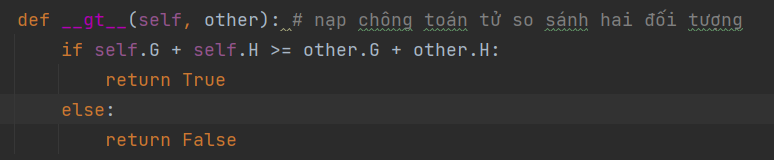
Khởi tạo 2 class Point và Window

class Point khởi tạo một constructor với các thuộc tính cho tọa độ và chi phí của đối tượng

Text

Description automatically generated

Và khởi tạo hàm so sánh chi phí

****Sau đó, class Matrix khởi tạo một ma trận rỗng với kích thước tùy ý nhập bởi người dùng, điểm xuất phát (start) khởi tạo với vị trí đầu tiên (0, 0), điểm kết thúc là rỗng. Tạo ra một mảng cha chứa các mảng con, mỗi mảng con đó chứa 100 các phần tử chứa chữ số 0.

Tạo một hàng đợi ưu tiên lưu trữ các tập đường đi, bắt đầu từ điểm xuất phát



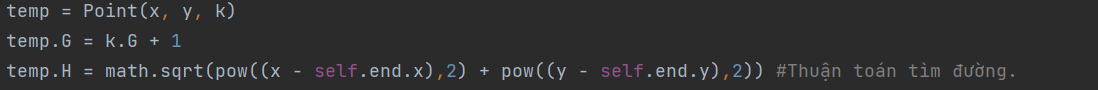
Phương thức **solve()** thực hiện thuật toán A\*. Mỗi ô trong ma trận có thể duyệt qua 8 ô xung quanh, quy định bởi pos1 và pos2.

Text

Description automatically generated

Sau đó, ta sẽ duyệt qua các ô thông qua hàng đợi ưu tiên và các ô được duyệt được quy định đổi sang màu xanh

Tính chi phí quãng đường đi với hàm f(x) = g(x) + h(x):



Quãng đường di chuyển sẽ được tính bằng căn bậc hai của tổng bình phương hiệu tọa độ giữa hai điểm. Sau đó các ô trong trạng thái chờ được duyệt sẽ chuyến sang màu vàng - “uncheck”

A picture containing outdoor, gate

Description automatically generatedSau đó là các phần khai báo giao diện và chạy chương trình. Khi chạy chương trình, màn hình hiển thị vùng xanh dương cho các chướng ngại vật trên đường đi, màu đỏ quy định điểm bắt đầu, màu tím là điểm kết thúc. Các ô xét duyệt trên đường đi sẽ chuyển màu xanh lá khi được xét, còn lại các ô màu vàng là các ô chờ xét duyệt (“uncheck”). Minh họa đồ thị:

**4. Đánh giá độ phức tạp của thuật toán**

Bỏ qua các trường hợp đặc biệt, độ phức tạp của A\* có thể được xấp xỉ dựa trên số lượng neighbor của mọi nút trên đường dẫn ngắn nhất. Giả sử rằng mọi nút đều có hầu hết các neighbor b và con đường ngắn nhất đi qua d node. Độ phức tạp của A\*:                     O(bd)

Trong trường hợp xấu nhất, nó sẽ thực hiện duyệt toàn bộ các nút, khi đó độ phức tạp của thuật toán sẽ là : O(N2)

**IV. Ưu, nhược điểm của thuật toán A\* (A-star)**

**1. Ưu điểm**

A\* là một thuật giải có thể biến đổi khéo léo tùy vào tình thế, tổng quát, trong đó hàm chứa cả tìm kiếm theo chiều sâu (DFS), tìm kiếm theo chiều rộng (BFS) và những nguyên lý Heuristic khác.

A\* nhanh chóng tìm được giải pháp tối ưu với sự định hướng của hàm Heuristic. Chính vì thế mà người ta thường nói A\* chính là một trong những thuật giải tối ưu ứng dụng phương pháp Heuristic.

**2. Nhược điểm**

Thuật toán A\* thường gây tốn bộ nhớ vì nó cần lưu lại những trạng thái mỗi bước khi mà nó đi qua, điều này chúng ta có thể thấy tương tự như với thuật toán tìm kiếm theo chiều rộng (BFS).

**V. So sánh thuật toán A-star và thuật toán tìm đường Dijkstra thuần túy**

**1. Thuật toán Dijkstra**

+      Có một hàm để tính chi phí, hàm này thể hiện giá trị chi phí thực từ điểm bắt đầu đến từng node:

**f(x) = g(x)**

+      Thuật toán Dijkstra tìm con đường ngắn nhất từ ​​điểm bắt đầu đến mọi node khác cần xét chỉ bằng cách xem xét chi phí thực khi thực hiện.

**2. Thuật toán A-star**

Thuật toán Astar dùng hai hàm g(x) và h(x) để thực hiện tính chi phí:

+      **g(x)**: hàm g(x) trong thuật toán A\* có chức năng giống như trong thuật toán Dijkstra. Hàm này là hàm tính chi phí thực để đạt được một node x.

+      **h(x)**: hàm h(x) trong thuật toán A\* là một hàm heuristic, hàm này tính chi phí ước lượng từ node x đến đỉnh kết thúc (target). Giá trị heuristic của hàm h(x) không bao giờ được tính cao hơn chi phí. Nghĩa là, chi phí thực để đến được đỉnh kết thúc (target) từ  một nút x nào đó phải lớn hơn hoặc bằng giá trị hàm h(x).

+      Tổng chi phí của mỗi node được tính bằng hàm **f(x)**:

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

+      Thuật toán A\* chỉ mở rộng một node nếu nó có khả năng tốt. Nó sẽ tập trung để tiếp cận điểm đích (target) từ nút (node) hiện tại, không tiếp cận các nút khác. Nếu như heuristic tốt, thì đó là phương pháp tối ưu để thực hiện giải bài toán.

**3. Dijkstra có là trường hợp đặc biệt của A\* không?**

+      Có thể nói Dijkstra là một trường hợp đặc biệt của A\* vì như đã được định nghĩa ở trên Dijkstra có thể hiểu là thuật toán A\* khi chi phí từ tất cả các đỉnh(hàm heuristic) đến đỉnh kết thúc đều bằng 0.

+      Trong trường hợp thuật toán A\* không thể tìm được đường đi tới điểm đích, nó sẽ thực hiện xét tất cả những nút trong đồ thị. Khi đó, thuật toán A\* thực hiện như thuật toán Dijkstra.

­