TRƯỜNG ĐẠI HỌC THỦY LỢI

**KHOA CÔNG NGHỆ THÔNG TIN**

A blue and white logo

Description automatically generated with low confidence

**BÀI TẬP LỚN**

HỌC PHẦN: TRÍ TUỆ NHÂN TẠO

**ĐỀ TÀI: TÌM ĐƯỜNG RA KHỎI MÊ CUNG**

Giáo viên hướng dẫn: Nguyễn Thị Kim Ngân

Sinh viên/nhóm sinh viên thực hiện:

1. Nguyễn Gia Tuấn - 2051063712

2. Bùi Xuân Hiếu - 1851061359

3. Hoàng Thị Phương Liên - 2051063679

**Hà Nội, năm 2021**

**Phần 1: Tổng quan**

1. ***Phương pháp***

**A** **Giải thuật tìm kiếm theo chiều rộng**

Giải thuật tìm kiếm theo chiều rộng (Breadth First Search – viết tắt là BFS) duyệt qua một đồ thị theo chiều rộng và sử dụng hàng đợi (queue) để ghi nhớ đỉnh liền kề để bắt đầu việc tìm kiếm khi không gặp được đỉnh liền kề trong bất kỳ vòng lặp nào.

Việc triển khai BFS tiêu chuẩn sẽ thực hiện chia mỗi đỉnh của biểu đồ thành một trong hai loại:

1. Đã được duyệt.
2. Không được duyệt.

Mục đích của thuật toán là đánh dấu mỗi đỉnh là đã duyệt trong khi tránh thực hiện quá trình duyệt theo vòng tròn.

Cách thuật toán hoạt động như sau:

1. Bắt đầu bằng cách đặt bất kỳ một trong các đỉnh của biểu đồ ở phía cuối của hàng đợi.
2. Lấy phần tử đầu tiên của hàng đợi và thêm nó vào danh sách đã truy cập.
3. Tạo danh sách các nút liền kề của đỉnh đó. Thêm những nút mà không có trong danh sách đã truy cập vào phía sau hàng đợi.
4. Tiếp tục lặp lại bước 2 và 3 cho đến khi hàng đợi trống rỗng.

**B** **Giải thuật tìm kiếm theo chiều sâu**

Giải thuật tìm kiếm theo chiều sâu (Depth First Search – viết tắt là DFS), còn được gọi là giải thuật tìm kiếm ưu tiên chiều sâu, là giải thuật duyệt hoặc tìm kiếm trên một cây hoặc một đồ thị và sử dụng stack (ngăn xếp) để ghi nhớ đỉnh liền kề để bắt đầu việc tìm kiếm khi không gặp được đỉnh liền kề trong bất kỳ vòng lặp nào.

Việc triển khai BFS tiêu chuẩn sẽ thực hiện chia mỗi đỉnh của biểu đồ thành một trong hai loại:

1. Đã được duyệt.
2. Không được duyệt.

Mục đích của thuật toán là đánh dấu mỗi đỉnh là đã duyệt trong khi tránh việc duyệt theo vòng tròn chu kỳ.

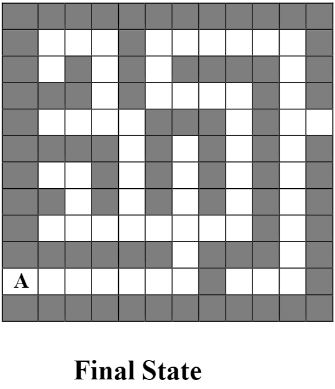
Thuật toán DFS hoạt động như sau:

1. Bắt đầu bằng cách đặt bất kỳ một trong các đỉnh của biểu đồ lên trên một ngăn xếp.
2. Lấy ra phần tử trên cùng của ngăn xếp và thêm nó vào danh sách đã truy cập.
3. Tạo danh sách các nút liền kề của đỉnh đó. Thêm những nút mà không có trong danh sách đã được duyệt vào đầu ngăn xếp.
4. Tiếp tục lặp lại các bước 2 và 3 cho đến khi ngăn xếp trống rỗng.

**Phần 2: Thực nghiệm (code)**

***1.Mô tả bài toán***

***Ảnh có chứa bàn

Mô tả được tạo tự động ***

* **Tên bài toán**: Sử dụng phương pháp giải thuật tìm kiếm theo chiều rộng và tìm kiếm theo chiều sâu áp dụng vào bài toán tìm kiếm đường ra khỏi mê cung.

Coi như chúng ta có một mê cung cần được giải quyết.

Mục tiêu là di chuyển xung quanh mê cung để tìm được đường ra ( nghĩa là bạn sẽ được cung cấp 1 vị trí bắt đầu tại lối vào mê cung và quá trình chỉ kết thúc khi bạn có mặt tại cánh cửa lối ra mê cung hay còn gọi là điểm đích)

**Input**:

* + Một mê cung dạng lưới kích thước 12x12
  + Điểm bắt đầu
  + Điểm kết thúc
* **Ouput:** 
  + Tổng quãng đường đã di chuyển
  + Quãng đường ngắn nhất để đi ra khỏi mê cung
  + Thời gian thực hiện ứng với từng phương pháp
* **Method (phương pháp thực hiện):**

**+ Mô tả cách xây dựng trạng thái.**

Hình thành bài toán bằng việc xác định 4 tham số:

* + Initial State
  + Successor
  + Goal Test
  + Path Cost

**+ Các quy luật chuyển trạng thái:**

Các hành động khả dụng là di chuyển trái, phải, lên xuống.

**+ Mô tả trạng thái đầu, trạng thái đích.**

Điểm bắt đầu được đặt tại vị trí 5x12 trong mê cung

Điểm kết thúc được đặt tại vị trí 11x1 trong mê cung

**+ Mô tả cách thực hiện để tìm ra đường đi từ trạng thái đầu đến trạng thái đích.**

* **Import các thư viện cần thiết**

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

* **Tạo lớp Grid\_Position xác định tọa độ từng khối trong mê cung**

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

* **Tạo lớp Node để định nghĩa thuộc tính vị trí riêng và chi phí các bước thực hiện để tới khối đó**

**Ảnh có chứa văn bản, đồng hồ đo, thiết bị

Mô tả được tạo tự động**

* **Sử dụng phương pháp tìm kiếm theo chiều rộng ( BFS )**
  + Khai báo function “bfs” với các biến đầu vào ứng với input



* + Tạo các mảng giúp xác định các khối lân cận có thể di chuyển tới

****

* + Tạo một mảng 2 chiều visited\_blocks để lưu các vị trí đã đi qua, sau đó đẩy vị trí xuất phát vào nó

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

* + Tạo queue sau đó đẩy điểm bắt đầu vào queue đó

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

* + Tạo vòng lặp while duyệt các nút có trong queue
  + Lấy ra nút ở đầu queue, sử dụng câu lệnh điều kiện kiểm tra nếu là nút cần tìm thì dừng lại và thông báo thành công, trả ra các kết quả về quãng đường đã đi và quãng đường ngắn nhất

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

* + Nếu chưa phải là nút cần tìm, tiếp tục thực hiện các thuật toán tiếp theo trong vòng lặp.
  + Xét nút đó chưa được đi qua -> lưu vị trí đó vào visited\_blocks, cập nhật giá trị chi phí di chuyển, lưu các giá trị của nút đang xét

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

* + Tạo vòng lặp for để tìm các nút lân cận với nút hiện tại sau đó đẩy vào cuối queue

**Ảnh có chứa văn bản, tường, màu bạc

Mô tả được tạo tự động**

* + Return -1 nếu không tìm được điểm đích
  + 
  + Tiếp tục thực hiện vòng lặp while duyệt queue
* **Sử dụng phương pháp tìm kiếm theo chiều sâu ( DFS )**
  + Việc thực hiện giải thuật theo phương pháp tìm kiếm theo chiều sâu về cách thực hiện cũng tương tự như khi tìm kiếm theo chiều sâu
  + Tuy nhiên, đối với việc giải thuật tìm kiếm theo chiều sâu chúng ta sẽ khởi tạo stack sau đó chúng ta sẽ đẩy các nút vào stack và thực hiện theo nguyên lý LIFO.
* **Tạo hàm main** 
  + Khai báo các input đầu vào cần thiết cho bài toán

Ảnh có chứa mũi tên

Mô tả được tạo tự động

* + Gọi hàm tìm kiếm theo chiều rộng -> thông báo quả tương ứng với kết quả được trả về

Ảnh có chứa văn bản

Mô tả được tạo tự động

* + Gọi hàm tìm kiếm theo chiều sâu -> thông báo quả tương ứng với kết quả được trả về

***2. Đánh giá kết quả tìm được:***

* **Kết quả tìm được**

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

=>Do input đầu vào khiến cho kết quả về quãng đường di chuyển và quãng đường ngắn nhất để tới đích là không có quá nhiều sự khác biệt. Tuy nhiên ta

vẫn có thể thấy phương pháp giải thuật tìm kiếm theo chiều sâu có chi phí di

chuyển thấp hơn so với giải thuật tìm kiếm theo chiều rộng.

=> khi tính thời gian thực hiện 2 phương pháp ta thấy thời gian thực hiện

phương pháp giải thuật tìm kiếm theo chiều rộng sẽ nhanh hơn so với phương pháp giải thuật tìm kiếm theo chiều sâu

**Kết luận**

Thời gian làm bài tập lớn là một dịp tốt để chúng em hệ thống lại kiến thức đã được học ở môn Trí Tuệ Nhân Tạo, giúp chúng em làm quen với công việc của một lập trình viên trong tương lai. Đồng thời còn giúp chúng em liên hệ giữa kiến thức lý thuyết để áp dụng vào thực tế.

Vì thời gian làm bài tập lớn có hạn, kiến thức của bản thân còn nhiều hạn chế, lần đầu tiên tiếp xúc với một vấn đề lớn mà bản thân chưa có kinh nghiệm thực tế. Do vậy, nội dung của bài tập chưa thật sự sâu sắc, không tránh khỏi những sai sót và hạn chế nhất định. Em kính mong được Cô chỉ bảo, góp ý thêm để bài tập lớn của chúng em được hoàn thiện hơn.

Cuối cùng chúng em xin chân thành cảm ơn Cô Nguyễn Thị Kim Ngân đã giúp chúng em tiếp nhận được thật nhiều kiến thức từ môn học này và đã giúp đỡ em hoàn thành bài tập này!

**Tài liệu tham khảo**

* Slide bài giảng môn Trí Tuệ Nhân Tạo
* Các tài liệu tham khảo online: google,…