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***

**Giải thuật tìm kiếm tối ưu (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 lược 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 đươc 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.

Các Bước Giải

1. Open: tập các trạng thái đã được sinh ra nhưng chưa được xét đến.
2. Close: tập các trạng thái đã được xét đến.
3. Cost(p, q): là khoảng cách giữa p, q.
4. g(p): khoảng cách từ trạng thái đầu đến trạng thái hiện tại p.
5. h(p): giá trị được lượng giá từ trạng thái hiện tại đến trạng thái đích.
6. f(p) = g(p) + h(p)
   * 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)

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 q khỏi Close

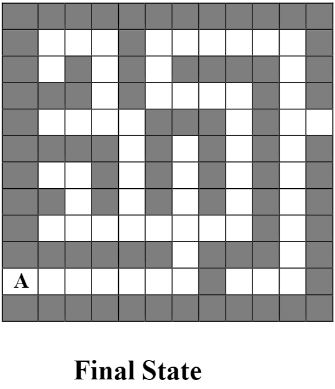
Thêm q vào Open

* + Bước 3: Không tìm được.

**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

Mô tả được tạo tự động**

* **Khai báo hàm heuristic**

****

* **Sử dụng phương pháp tìm kiếm tối ưu ( thuật toán A\* )**
  + Khai báo function “A\_Star” 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 open là 1 hàng đợi ưu tiên 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

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

* + Tính heuritic của khối sau đó tính f và đẩy khối vào hàng đợi ưu tiên

Ảnh có chứa văn bản

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 rộng ( DFS )**
  + Thực hiện phương pháp tìm kiếm theo chiều rộng với các bước thực hiện được nói tới trong bài kiểm tra trước
* **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 tối ưu (thuật toán A\*) -> 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 rộng -> thông báo quả tương ứng với kết quả được trả về

Ảnh có chứa văn bản, màn hình, chỉ

Mô tả được tạo tự động

***2. Đánh giá kết quả tìm được:***

* **Kết quả tìm được**

**Ảnh có chứa văn bản

Mô tả được tạo tự động**

=>so với phương pháp tìm kiếm theo chiều rộng ta thấy khi thực hiện giải thuật tìm kiếm tối ưu (tìm kiếm A\*) có số bước thực hiện ít hơn do vậy nó tối ưu hơn.

=> 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 tương đương so với phương pháp giải thuật tìm kiếm tối ưu (tìm kiếm A\*)

**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,…