PROJECT 1

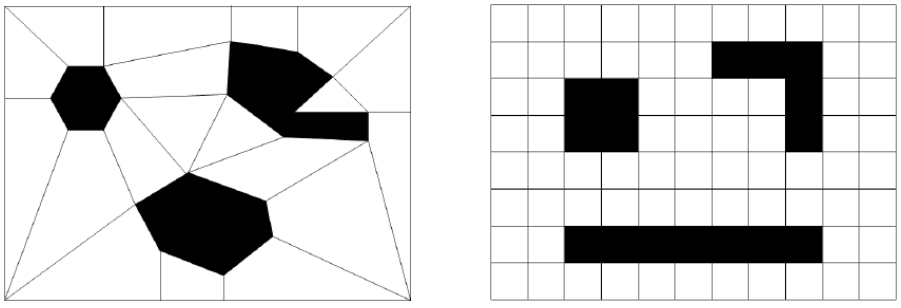
CHỦ ĐỀ: TÌM KIẾM HEURISTIC VỚI A*

1 Giới thiệu bài toán

Cho một vị trí bắt đầu (start) và một vị trí đích (goal), bài toán chỉ ra một đường nối hai vị trí này với nhau được gọi là bài toán tìm đường. Chúng ta thường gặp bài toán này trong lĩnh vực robot di động, trong đó mục tiêu là robot có khả năng xác định nó phải di chuyển như thế nào từ vị trí hiện tại tới vị trí đích mong muốn. Do vậy, chúng ta cần phải xem xét các vị trí bắt đầu và đích cũng như địa hình của môi trường xung quanh (ví dụ: vị trí của các chướng ngại vật, kích thước của chúng…)

Một cách giải quyết bài toán này thường thấy là chia không gian di chuyển thành nhiều ô và biểu diễn nó dưới dạng mạng lưới. Một mạng lưới đơn giản là một cấu trúc dữ liệu mã hoá cách bố trí của không gian di chuyển. Nói cách khác, nó là một bản đồ cho thấy các ô trống là các vị trí có thể tự do di chuyển, các ô bị tô đen là các vị trí có chướng ngại vật.

Hình 1(a) thể hiện một lưới của không gian di chuyển đã được phân rã ra thành các ô với kích thước khác nhau. Hình 1(b) thể hiện một lưới với các ô cố định kích thước. Từ các ví dụ này, chúng ta thấy rằng có thể biểu diễn bất kỳ lưới nào dưới dạng mảng 0, 1; trong đó 0 biểu diễn ô trống và 1 biểu diễn ô chướng ngại vật. Hơn nữa, cách biểu diễn không gian dạng lưới này cũng giúp chúng ta dễ dàng xác định đường đi ngắn nhất giữa hai ô bất kỳ bằng cách tìm chuỗi các ô trống ngắn nhất nối hai ô đó với nhau.



(a) (b)

Hình 1. Phân rã không gian di chuyển thành mạng lưới với các ô khác kích thước (a) và cùng kích thước (b)

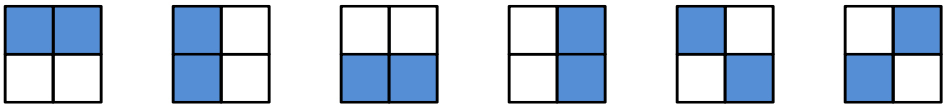
Để làm điều này, chúng ta xem bài toán tìm đường như một bài toán tìm kiếm trên đồ thị. Mỗi ô trống trên lưới là 1 đỉnh trên đồ thị. Tồn tại một cạnh nối hai đỉnh của đồ thị nếu hai

ô biểu diễn bởi 2 đỉnh đó nằm liền kề nhau và có thể đi được từ ô này đến ô kia. Sau khi biểu diễn bài toán dưới dạng đồ thị, chúng ta dễ dàng áp dụng nhiều thuật toán tìm kiếm trên đồ thị để giải bài toán tìm đường cho robot. Ví dụ: breadth-first-search, depth-first-search, Dijkstra's, A*…

Mục tiêu là cài đặt thuật toán A* để giải bài toán tìm đường đi ngắn nhất cho robot.

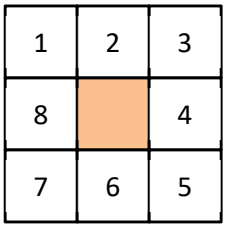
2 Yêu cầu 1

Cài đặt thuật toán A* giúp robot tìm đường đi ngắn nhất từ vị trí bắt đầu đến vị trí đích chỉ định trước trong một không gian cho sẵn (phải né chướng ngại vật). Đường đi phải đi qua các ô liền kề nhau (Xem hình 2).



Hình 2. Các trường hợp 2 ô liền kề nhau (tô màu xanh)

Sử dụng heuristic là khoảng cách Euclidean giữa ô hiện tại và ô đích. Các ô trống phải được mở theo thứ tự như hình 3.



Hình 3. Thứ tự mở các ô liền kề với ô chính giữa (màu cam) khi duyệt tìm đường đi

Thông tin cho trước bao gồm:

- Vị trí start và goal
- Số chiều của không gian tìm kiếm (số dòng x cột)
- Bản đồ không gian tìm kiếm (có chướng ngại vật)

Thông tin xuất bởi chương trình yêu cầu bao gồm:

- Đường đi ngắn nhất từ start đến goal thể hiện trên bản đồ không gian tìm kiếm.

Chương trình chạy dưới dạng tham số dòng lệnh như sau:

<tên chương trình> <input> <output>

Trong đó:

- ***<tên chương trình>***: lab02.exe
- ***<input>:*** đường dẫn đến tập tin đầu vào.
- ***<output>:*** đường dẫn đến tập tin kết quả.

2.1 Định dạng của tập tin đầu vào

Là 1 tập tin text (.txt) lưu trữ thông tin cho trước với định dạng như sau:

N //kích thước bản đồ (NxN) *S_x,S_y //toạ độ điểm bắt đầu* *G_x,G_y //toạ độ điểm kết thúc* *//N dòng tiếp theo: bản đồ di chuyển với 0: ô trống,* *//1: chướng ngại vật* *//mỗi ô cách nhau 1 khoảng trắng*	7 0 0 6 6 0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 1 0 0 0 0 1 0 1 1 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0

Lưu ý:

- N, M, S, G là các số nguyên dương
- Mỗi vị trí toạ độ gồm 2 chỉ số dòng, cột cách nhau bởi dấu phẩy (không có khoảng trắng). Chỉ số đánh từ 0.

2.2 Định dạng của tập tin đầu ra

Là một tập tin text (.txt) thể hiện bản đồ tìm kiếm và giải pháp đường đi:

-1 *//không tìm được đường đi*

Ngược lại, ghi ra file:

K *//số nguyên, cho biết tìm được đường đi ngắn nhất qua K bước (tính cả start và goal)* *(x_1,y_1) (x_2,y_2) … (x_k,y_k) //Chuỗi toạ độ đường đi, mỗi toạ độ cách nhau bởi 1 khoảng trắng* *// Vẽ bản đồ đường đi, trong đó:* • *S: start* • *G: goal* • *-: ô trống* • *o: obstacle (chướng ngại vật)* • *x: ô trên đường đi ngắn nhất*

Mỗi ô cách nhau bởi 1 khoảng trắng (xem ví dụ)

```
9
(0,0) (0,1) (1,2) (2,3) (3,4) (4,4) (5,4) (6,5) (6,6)
S - - - - - o
- x - - - o o
o o x - - - o
- o o x - - -
- o o - x o -
- o - - x o -
- - - - - x G
```

Ví dụ 1 tập tin đầu ra với chiều dài đường đi ngắn nhất là 9 (tính cả start và goal)

3 Yêu cầu 2

SV thực hiện các yêu cầu nâng cao sau đây:

a) SV tự đề xuất hàm Heuristic của mình. Lưu ý: phải chứng minh Heuristic đã lựa chọn là chấp nhận được. Chứng minh rõ trong báo cáo.

b) Tìm hiểu thuật toán ARA* [2], một biến thể của thuật toán A* với hằng số ∈ thêm vào. Nếu ∈= 1.5 nghĩa là đường đi tìm được sẽ 50% tối ưu so với đường đi tìm được bởi cùng một hàm heuristic h(x). Hằng số ∈ được thêm vào để tăng tốc quá trình tìm kiếm như sau:

$$f(x) = g(x) + \in \times h(x)$$

Giả sử người dùng muốn giới hạn thời gian tìm kiếm của A*, nghĩa là thuật toán A* phải trả về kết quả đường đi (không cần phải tối ưu) trong khoảng thời gian cho phép. Hãy chỉnh sửa thuật toán A* của bạn sao cho người dùng có thể định sẵn một khoảng thời gian tmax (tính theo msec) để có được kết quả đường đi. Nếu tìm được đường đi sau khoảng thời gian t < tmax, tiếp tục tìm kiếm với ∈ nhỏ hơn cho tới khi t=tmax (hết giờ) để tìm được đường đi tối ưu hơn. Báo cáo kết quả đường đi với mỗi ∈ khác nhau.

c) Minh hoạ từng bước chạy của thuật toán A* bằng giao diện đồ hoạ người dùng (graphic user interface)

4 Báo cáo

Trong báo cáo, sinh viên ghi rõ:

- Thông tin từng thành viên
- Công việc được giao cho mỗi thành viên và mức độ hoàn thành của mỗi thành viên so với yêu cầu đặt ra.

- Mức độ hoàn thành của đồ án.
- Những vấn đề chưa thực hiện được.
- Các bộ test đã chạy (ít nhất 3 bộ test có đặc trưng khác nhau)
- Các báo cáo liên quan đến yêu cầu 2
- Sơ đồ biểu diễn hệ thống phần mềm (các hàm chính, công dụng, …).
- Mô tả cấu trúc dữ liệu đã cài đặt.
- Mô tả thuật toán chính đã thực hiện, những cải tiến hay thay đổi (nếu có).
- Tài liệu tham khảo (nếu có). Lưu ý tham khảo không có nghĩa là đạo văn. Việc sao chép (không quan tâm tới giống bao nhiêu %) nếu bị phát hiện sẽ 0 điểm môn học.

Các bài làm không có báo cáo, hoặc báo cáo không đủ các mục, GV được từ chối chấm bài.

5 Qui định

- Dạng bài tập: ***nhóm 2 người.***
- Đồ án có thể được chấm vấn đáp trực tiếp.
- Ngôn ngữ lập trình bắt buộc: Python. Trong trường hợp không thể, SV có thể sử dụng ngôn ngữ khác nhưng điểm giới hạn tối đa 7 điểm.
- Hạn nộp: ***xem trên Moodle***
- Đặt tên chương trình là MSSV1_MSSV2_..._Labxx, với MSSV là mã số sinh viên.
- Bài nộp gồm 2 thư mục:
  - ***Source***: chứa source code chương trình
  - ***Release***: chứa file thực thi
  - ***Report***: chứa tập tin báo cáo (.doc, .docx hoặc pdf)
- Nén toàn bộ thư mục thành 1 file trước khi nộp.
- Trong quá trình thực hiện nếu có thắc mắc, gửi mail về lnthanh@fit.hcmus.edu.vn

**** Lưu ý: Các bài làm giống nhau sẽ bị 0 điểm môn học.***

Tài liệu tham khảo

[1] Ani Hsieh, "A* Search Algorithm", course E28, Mobile Robotics.
[2] Maxim Likhachev, Geoff Gordon and Sebastian Thrun, "ARA*: Anytime A* with Provable Bounds on Sub-Optimality," Advances in Neural Information Processing Systems 16 (NIPS), MIT Press, Cambridge, MA, 2004.