ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

KHOA Toán - Tin Học

~~~~~~*~~~~~~

BÁO CÁO THỰC HÀNH NHẬP MÔN TRÍ TUỆ NHÂN TẠO

Sinh viên	**:**	**VÒNG VĨNH PHÚ**
MSSV	**:**	**19110413**
Môn Học	**:**	**NHẬP MÔN TRÍ TUỆ NHÂN TẠO**
Trường	**:**	**ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

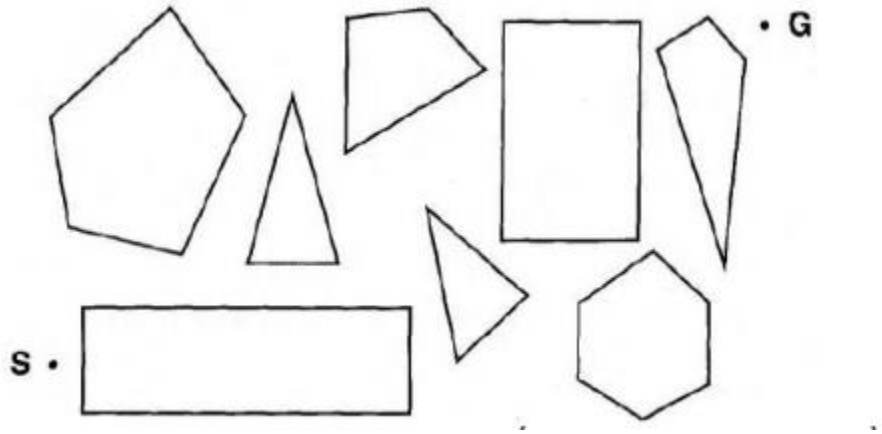
THÀNH PHỐ HỒ CHÍ MINH, THÁNG 11 NĂM 2021
~~~~~~

Mục lục

1. BÀI TOÁN

Đề bài :

- Xét bài toán tìm đường đi ngắn nhất từ điểm S tới điểm G trong một mặt phẳng có các chướng vật là những đa giác lồi



(a) Giả sử không gian trạng thái chứa tất cả các vị trí (x, y) nằm trong mặt phẳng. Có bao nhiêu trạng thái ở đây? Có bao nhiêu đường đi từ đỉnh xuất phát tới đỉnh đích?

(b) Giải thích ngắn gọn vì sao đường đi ngắn nhất từ một đỉnh của đa giác tới một đỉnh khác trong mặt phẳng nhất định phải bao gồm các đoạn thẳng nối một số đỉnh của các đa giác? Hãy định nghĩa lại không gian trạng thái. Không gian trạng thái này sẽ lớn bao nhiêu?

(c) Định nghĩa các hàm cần thiết để thực thi bài toán tìm kiếm, bao gồm hàm successor nhận một đỉnh làm đầu vào và trả về tập đỉnh có thể đi đến được từ đỉnh đó trong vòng 1 bước

(d) Áp dụng một thuật toán tìm kiếm để giải bài toán.

2. Giải Quyết Bài toán

(a) Giả sử không gian trạng thái chứa tất cả các vị trí (x, y) nằm trong mặt phẳng. Có bao nhiêu trạng thái ở đây? Có bao nhiêu đường đi từ đỉnh xuất phát tới đỉnh đích?

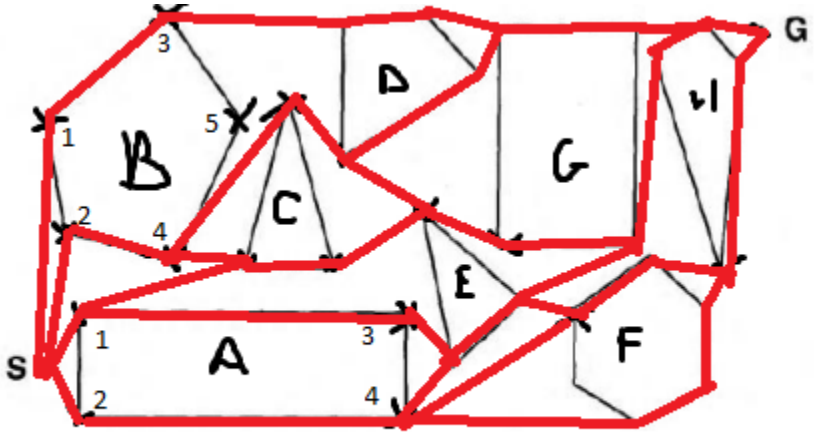
Trả lời : Giả sử không gian trạng thái chứa tất cả các vị trí (x,y) nằm trong mặt phẳng thì có vô số trạng thái và cũng vô số đường đi để đích

(b) Giải thích ngắn gọn vì sao đường đi ngắn nhất từ một đỉnh của đa giác tới một đỉnh khác trong mặt phẳng nhất định phải bao gồm các đoạn thẳng nối một số đỉnh của các đa giác? Hãy định nghĩa lại không gian trạng thái. Không gian trạng thái này sẽ lớn bao nhiêu?

Trả lời : Bằng cách đi từ đỉnh của đa giác trạng thái có thể đi được ít hơn và tối ưu hơn khi robot phải chọn nhiều điểm trên mặt phẳng và các điểm nối tới đỉnh là đường thẳng nên cũng ngắn hơn nhiều

Gọi lần lượt A,B,C,D,E,G,H là các đa giác và 2 điểm start/goal

Với A là 4 đỉnh, B 5 đỉnh, C 3 đỉnh, D 4 đỉnh, E 3 đỉnh, G 4 đỉnh, H 4 đỉnh, Start/Goal = 2 => 29 trạng thái chứa vị trí (x,y) trong mặt phẳng



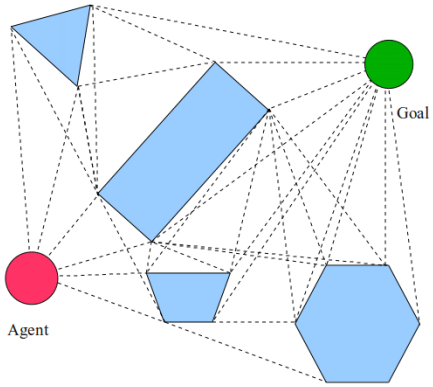
Sơ đồ minh hoạ không gian trạng thái của bài toán

3. Xây dựng hàm và ý tưởng bài toán

3.1 Ý tưởng :

- Do không gian trạng thái chưa biết (các đa giác, toạ độ đỉnh của đa giác trên mặt phẳng) nên ta sẽ tự ngầm định hoặc tự cho các con số bằng cách đưa thông tin để làm tối ưu bài toán nên trong bài này em đã giảm số lượng đa giác từ 9 đa giác thành 3 đa giác để dễ dàng xét trạng thái và các đường đi có thể đi được
- Để lưu trữ được thông tin có thể đi được từ start/goal ta sử dụng phương pháp Visibility graph (đồ thị khả năng tìm đường đi cho robot thông qua các đỉnh cho trước)

  Đọc thêm : https://en.wikipedia.org/wiki/Visibility_graph
- Đồ thị sẽ tìm đường đi bằng cách duyệt hết các điểm có thể đi được và thông qua đó xây dựng bản đồ đường đi cho agent vì có rất nhiều khả năng để đến đích nên sử dụng các bài toán đường đi có chi phí rất hữu dụng cho bài tập lần này



- Khi có đồ thị như trên ta sẽ lưu các vị trí đó vào các node (polygon) và thực hiện tính toán khoảng cách đây chính là chi phí đường đi cho các cạnh (edge) đến các đỉnh (vertex)

- Sử dụng thuật toán UCS để duyệt các đường đi có tổng trọng số là nhỏ nhất để tìm đường đi ngắn nhất

3.2 Xây dựng bài toán :

(a) Input/Output :

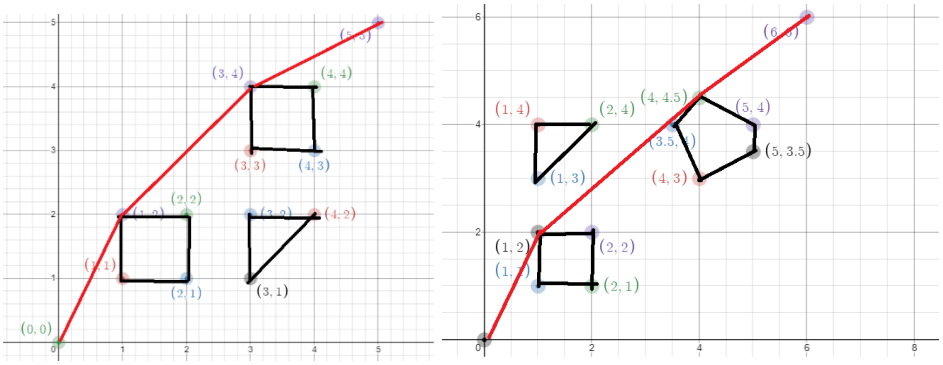
- Input : 1 list cho biết số lượng polygon vật cản với mỗi polygon là list trống để chuẩn bị lưu trữ các đỉnh, điểm S/G là điểm đầu và điểm cuối có dạng tuple, polygon[i] là list các vị trí (x,y) chứa đỉnh
- Output : Thông tin khoảng cách từ điểm đầu đến điểm cuối, và các đỉnh đã đi qua

(b) Class/Function :

- Class (Lớp) :
  - Xây dựng một lớp AdjList để tổ chức data input bao gồm đỉnh và cạnh thông tin các node mở rộng
  - Xây dựng lớp Vgraph là lớp cho biết các đường nối với các đỉnh và các polygon lưu vào trong list
  - Xây dựng một lớp BinaryHeap đây là lớp thuật toán để lấy phần tử (chi phí) nhỏ nhất trong đồ thị để ta có thể lấy thông tin đường đi ngắn nhất để xét
  - Xây dựng lớp ListNode để lưu trữ data bao gồm đỉnh, toạ độ, chi phí đường đi các đỉnh liền kề
- Function in Class : phần này ta sẽ tập trung đi vào AdjList và Vgraph vì BinaryHeap chỉ là lớp để tìm kiếm và lấy phần tử nhỏ nhất cũng như sắp xếp lại lớp list node đã tạo
  - Line_intersection (Vgraph) : Hàm nối 2 đoạn thẳng lại với nhau để tạo được polygon và các đường đi được khi cho các điểm cho trước

- FindEdges (Vgraph) : tìm các cạnh từ S-G để lưu thông tin đường đi cho graph
- Form/ComputeWeight (AdjList) : Tính toán và khởi tạo khoảng cách đường đi giữa các node đi được khi khởi tạo Vgraph hoàn tất
- UCS/GetPath : Tìm kiếm đường đi bằng cách duyệt tất cả khoảng cách giữa cách các đoạn đường có tổng chi phí nhỏ nhất và trả về đường đi ngắn nhất bằng cách truy vết từ điểm đầu đến đích (Thuật toán đã trình bày từ những bài trước)

- Function Path_Finding (main) : Đây là hàm khởi tạo Vgraph và khoảng cách các node bằng các method từ class xuất ra output cho bài toán bằng truyền vào list polygon, điểm S và điểm G để tiến hành tính toán khoảng cách và khởi tạo Vgraph cho mô hình



Demo cách bài toán chạy với số lượng polygon là 3