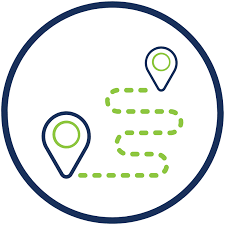
**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BÁO CÁO ĐỒ ÁN 1**

**CƠ SỞ TRÍ TUỆ NHÂN TẠO**



**ROBOT TÌM ĐƯỜNG**

Mục lục

[I. Thông tin thành viên 3](#_Toc20787599)

[II. Mức độ hoàn thành 3](#_Toc20787600)

[III. Tổ chức đồ án 3](#_Toc20787601)

[IV. Tài liệu tham khảo 4](#_Toc20787602)

# Thông tin thành viên

|  |  |
| --- | --- |
| **MSSV** | **HỌ VÀ TÊN** |
| 1712078 | Ngô Phan Nhật Lâm |
| 1712360 | Phạm Hoàng Đức |
| 1712475 | Cao Nhơn Hưng |

# Mức độ hoàn thành

|  |  |  |
| --- | --- | --- |
| **STT** | **YÊU CẦU** | **HOÀN THÀNH** |
| 1 | Cài đặt thành công 1 thuật toán để tìm đường đi từ S tới G. Báo  cáo lại thuật toán và quá trình chạy thử. Chạy thử trường hợp không có đường đi. | 100 % |
| 2 | Cài đặt ít nhất 3 thuật toán khác nhau (ví dụ tìm kiếm mù, tham  lam, heuristic, …). Báo cáo nhận xét sự khác nhau khi chạy thử 3 thuật toán. | 100 % |
| 3 | Trên bản đồ sẽ xuất hiện thêm một số điểm khác được gọi là điểm đón. Xuất phát từ S, sau đó đi đón tất cả các điểm này rồi đến trạng thái G. Thứ tự các điểm đón không quan trọng. Mục tiêu là tìm ra cách để tổng đường đi là nhỏ nhất. Báo cáo thuật toán đã áp dụng và quá trình chạy thử. | 100 % |
| 4 | Các hình đa giác có thể di động được với tốc độ h tọa độ/s.  Cách thức di động ở mức đơn giản nhất là tới lui một khoảng nhỏ để đảm bảo không đè lên đa giác khác. Chạy ít nhất 1 thuật toán trên đó. Quay video và đính kèm link vào báo cáo. | 100 % |
| 5 | Thể hiện mô hình trên không gian 3 chiều (3D) | 0 % |

# Tổ chức đồ án

Có 4 file code python tương ứng với 4 mức đồ án, mỗi mức có các file input, output riêng tuân theo chuẩn input, output củ yêu cầu đô án

## Mức độ 1: muc1.py

Abc

## Mức độ 2: muc2.py

Abc

## Mức độ 3: muc3.py 3.1 Vấn đề:

Trên bảng đồ sẽ xuất hiện thêm một số điểm khác nhau (được gọi là điểm đón). Tìm đường đi xuất phát từ S đến G sao cho lần lượt đi qua các điểm này và phải sắp xếp thứ tự ưu tiên đi qua chúng sao cho đoạn đường đi là ngắn nhất.

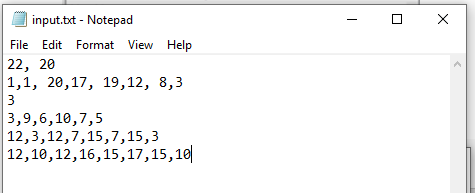
## 3.2 Cài đặt:

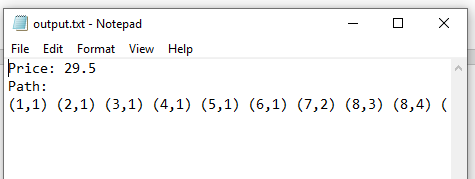
- Đầu tiên đọc các tham số cần thiết từ file input.txt bằng hàm readfile().

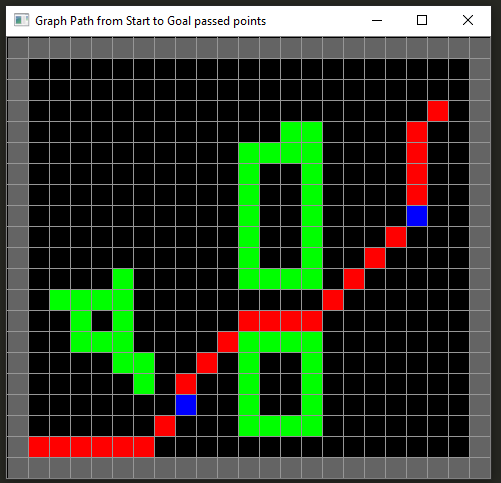
- Tiếp theo, ta sẽ đi tìm thứ tự ưu tiên đi qua các điểm đón nào trước bằng hàm FindPriority(). Quy ước chi phí được tính là quãng đường đi giữa 2 điểm, nếu đi dọc hay ngang thì chi phí mỗi bước là 1, nếu đi chéo thì chi phí là 1,5. Quá trình tìm này dựa vào ý tưởng thuật toán AStar với hàm ước lượng heristic f(x). Trước hết, ta sẽ tìm một điểm gần G nhất trong số các điểm đón rồi đưa điểm đó vào tập đóng stack, cứ tiếp tục tìm trong tập các điểm đón còn lại gần nhất với điểm trên top của stack, cho đến khi đi về điểm S.

- Sau khi đã có thứ tự ưu tiên đi qua mỗi điểm thì ta sẽ tìm đường đi chi tiết từ S đi qua các điểm đó và đến G sao cho tránh các vật thể đã cho bằng hàm FindPath(). Cuối cùng là xuất kết quả ra file output.txt và biểu diễn đồ họa đường đi.

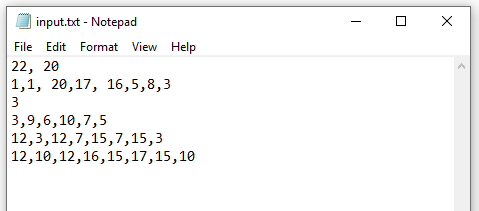
*Test1:*

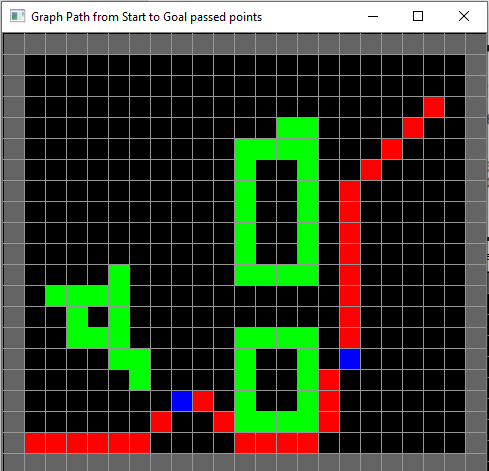
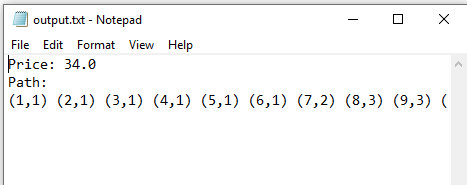




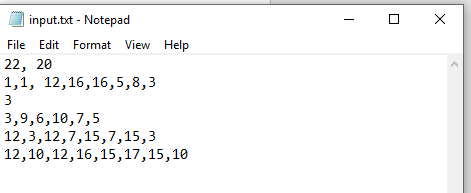


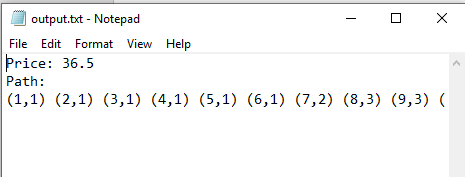
*Test2:*

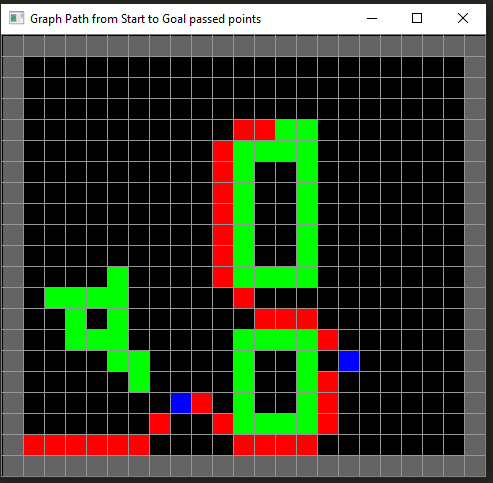




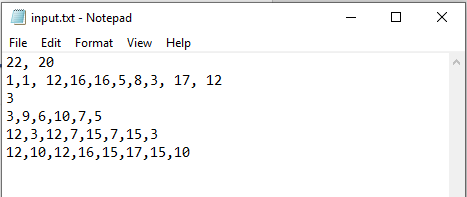
*Test 3:*

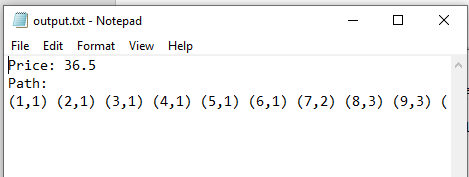


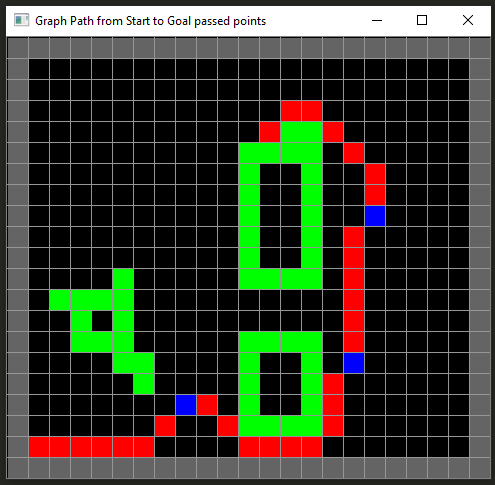




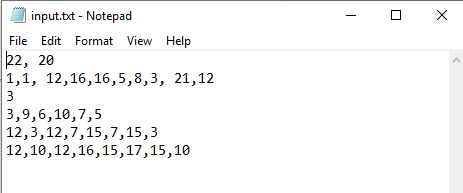
*Test 4:*

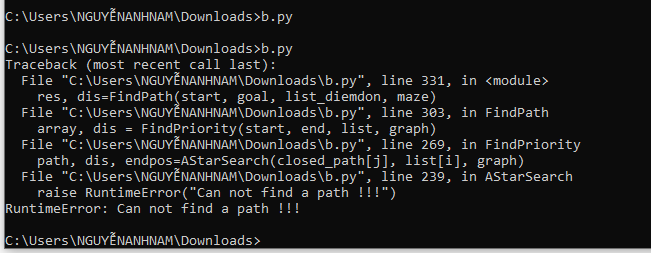






*Test 4:*

**



## Mức độ 4: muc4.py

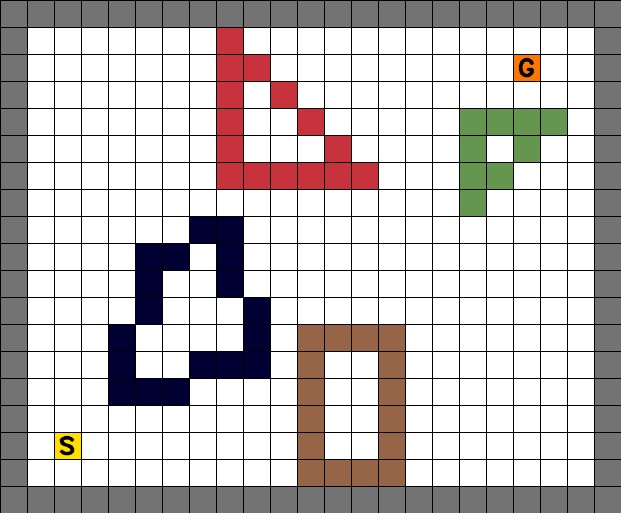
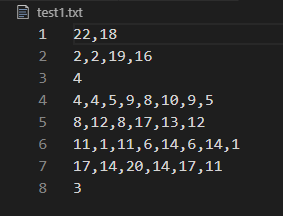
1. **Yêu cầu**:

Robot di chuyển đến đích trong không gian có các vật cản hình đa giác có thể di động được với tốc độ h tọa độ/s. Cách thức di động có thể ở mức đơn giản nhất là tới lui một khoảng nhỏ để đảm bảo không đè lên đa giác khác.

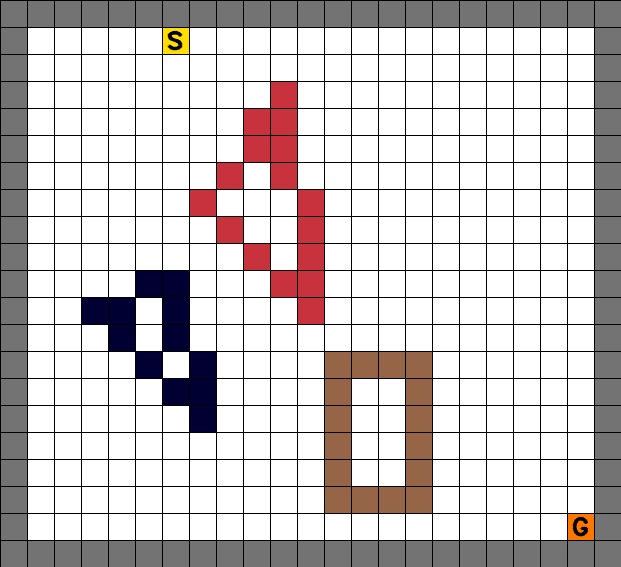
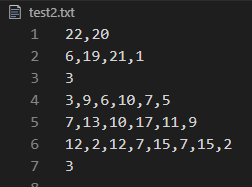
1. **Input**:

Như chuẩn đề bài và thêm 1 dòng để chứa tốc độ *h*

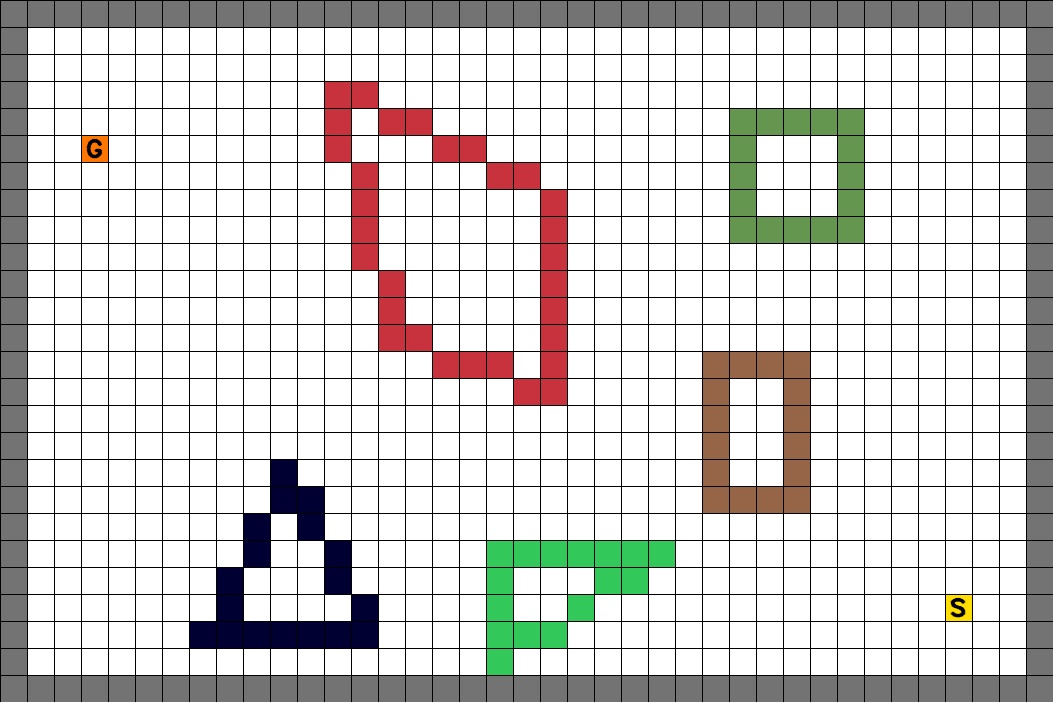
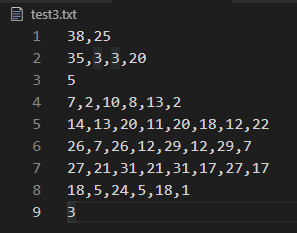
* test1.txt



* test2.txt



* test3.txt



1. **Tổng thể:**

Các vật cản sẽ di chuyển bất kỳ theo 8 hướng, robot tìm đường đi ban đầu và đi theo, nếu gặp vật cản không thể đi nữa thì tìm đường đi mới.

1. **Chi tiết cách thực hiện**:

* Chỉ xét các tọa độ nguyên; robot có thể đi chéo
* Đọc tất cả giá trị từ input và đánh dấu tương ứng vào mảng 2 chiều ***maze***, chỉ số 1 ô của mảng (i, j) tương ứng với tọa độ (j, i) trên đồ thị thực tế

|  |  |
| --- | --- |
| Giá trị | Nội dung biểu thị |
| -1 | border |
| 0 | empty spot |
| 1 | start point |
| 2 | goal point |
| 3 | pickup points (trong mức độ này ta không xét) |
| 4 | robot spot |
| >= 5 | polygons |

* Dùng giải thuật BFS để tìm kiếm đường đi ngắn nhất cho robot: hàm *bfs(maze, start)* trả về một list các cặp tọa độ là điểm mà robot có thể đi để đến được đích
* Ta cho biến route chứa đường đi của frame gốc ban đầu – đọc từ input
* Sau đó nhích robot lên 1 bước theo route, sau đó thực hiện xử lý các frame sau
* Tạo vòng lặp while để xử lý từng frame
  + Mỗi frame tương ứng với vị trí các vật thể tại một thời điểm
  + Thời gian chờ của mỗi frame là *1000(ms)/h* để tương ứng *h* tọa độ/s
* Tại mỗi frame:
  + Cho các đa giác di chuyển đến 1 hướng bất kỳ trong bán kính 1 ô lân cận
  + Xét điểm tiếp theo trong route, nếu có thể đi được (không dính vật cản, không dính lề, không bị đi vào trong vật cản) thì ta cho robot nhích đến ô đó; ngược lại ta qua frame để các vật cản di chuyển, nếu đồ thị mới vẫn không đi được thì ta thực hiện tính toán lại route mới cho robot, bắt đầu từ điểm đang xét
  + Đồng thời ta cũng xét đến việc nếu vật cản di chuyển liên tục qua một số lượng frame nhất định nhưng robot vẫn không tìm được đường đi thì xuất ra màn hình không có đường đi, kết thúc while.
  + Robot được xem là đến đích khi route chỉ còn chứa 1 điểm là điểm đích.
  + Với mỗi bước di chuyển được của robot, ta cộng vào tổng chi phí, các bước di chuyển ngang/dọc có chi phí 1, chéo có chi phí 1.5

1. **Output**:

* *Đồ họa biểu diễn*: video demo output cho file input: test3.txt

<link>

* File output.txt chứa một số là tổng chi phí để đến được đích của robot

1. **Hướng dẫn chạy code:**

File readme trong folder SourceCode

## Mức độ 5: Chưa thực hiện.

# Tài liệu tham khảo

1. [https://rosettacode.org/wiki/A\*\_search\_algorithm#Python](https://rosettacode.org/wiki/A*_search_algorithm%23Python)
2. <https://docs.opencv.org/2.4/modules/core/doc/drawing_functions.html>