TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN KHOA HỌC MÁY TÍNH

BÁO CÁO ĐỒ ÁN SỐ 01

MÔN: CƠ SỞ TRÍ TUỆ NHÂN TẠO

Chủ đề: **Robot Tìm Đường**

Giảng viên lý thuyết: Thầy Lê Hoài Bắc
Giảng viên hướng dẫn thực hành: Thầy Lê Ngọc Thành

MỤC LỤC

- **THÔNG TIN NHÓM :**

  STT Nhóm: 22

STT	*Họ và tên*	*MSSV*
1.	Nguyễn Đỗ Chí Thảo	1712159
2.	Nguyễn Trọng Văn	1712202
3.	Lê Quang Vũ	1712209

- **THỰC HIỆN:**

  - **Mức 1:**

    Nhóm đã cài đặt thành công ít nhất 1 thuật toán: Uniform Cost Search để tìm đường đi chạy từ S đến G.

  - **Mức 2:**

    Nhóm đã cài đặt 3 thuật toán:

    + Uniform Cost Search (UCS)

    + Greedy Search

    + A Star

    Đánh giá các thuật toán:

    - Uniform Cost Search:
      + Chắc chắn tìm được đường đi
      + Đường đi tìm được sẽ là ngắn nhất
      + Hiệu quả với bản đồ nhiều vật cản như trong mê cung
      + Không hiệu quả với bản đồ trống (S -> G không có vật cản)
      + Tốn nhiều bộ nhớ và thời gian

- Greedy Search:

  + Phiên bản thuật toán được dùng trong chương trình được cải tiến thêm việc đánh dấu đường đã đi qua, nên thuật toán chắn chắn sẽ tìm được đường đi (nếu có)

  + Đường đi tìm được không chắc là ngắn nhất

  + Hiệu quả với bản đồ trống, không nhiều vật cản

  + Không hiệu quả với bản đồ nhiều vật cản và chật hẹp

  + Tốn nhiều thời gian nếu bản đồ phức tạp nhưng tốn ít bộ nhớ

  + Độ hiệu quả phụ thuộc lớn vào hàm Heuristic

- A Star:

  + Chắn chắn tìm được đường đi

  + Đường đi tìm được chưa chắc là ngắn nhất (Phụ thuộc vào hàm Heuristic)

  + Hiệu quả với bản đồ trống, không nhiều vật cản

  + Không hiệu quả với bản đồ nhiều vật cản và chật hẹp

  + Tốn thời gian nếu bản đồ phức tạp nhưng tốn ít bộ nhớ

  + Độ hiệu quả phụ thuộc lớn vào hàm Heuristic

- **Mức 3:**

  Để tìm đường từ S đến G mà phải đi qua hết tất cả điểm đoán. Xét tại điểm hiện tại, giả sử là điểm M, tìm điểm có khoảng cách "đường chim bay" gần M nhất mà chưa đi qua, giả sử là điểm N. Tìm đường đi ngắn nhất từ điểm M đến điểm N. Để tỉm đường đi ngắn nhất từ M đến N, ta có thể áp dụng cả ba thuật toán đã được áp dụng ở mức 2.

  Làm tương tự cho tất cả các điểm còn lại.

- **Mở rộng:**

  Nhóm cho phép thuật toán đi chéo. Chi phí đi chéo = 1.5*ngang

Để thực hiện được việc đi chéo, cần phải xác định được các ô ở " bên trong" thuộc đa giác lồi từ tọa độ các đỉnh. Việc giúp xác định được các ô ở bên trong đa giác lồi giúp xử lý được trường hợp đi xuyên qua đa giác.

Để thực hiện điều này, nhóm đã áp dụng giải thuật: "Convex Polygon Rasterization"

- **Giao diện:**

  Khi chạy chương trình sẽ xuất hiện hai cửa sổ:

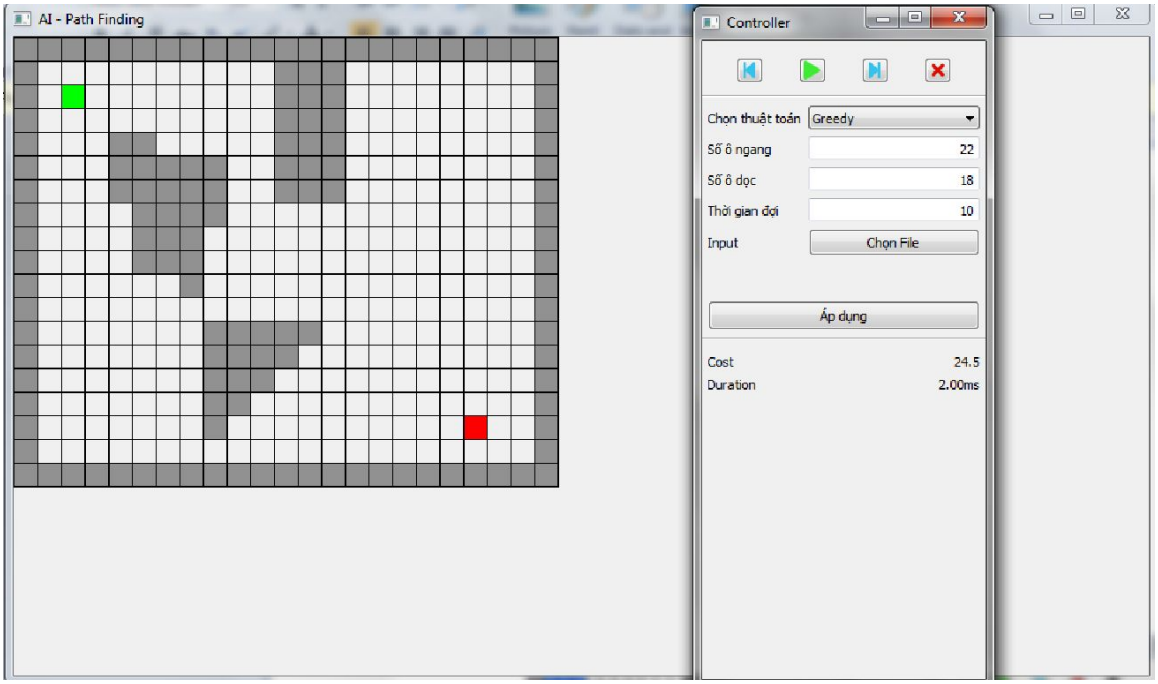
  + AI -Path Finding: In ra bản đồ từ file Input của người dùng và mô phỏng lại quá trình tìm kiếm đường đi của thuật toán (có thể zoom và di chuyển nếu bản đồ to quá màn hình)

  + Controller: Cho phép người dùng hiệu chỉnh các tham số, chức năng của chương trình:

    - Thay đổi thuật toán.
    - Thay đổi số dòng, số cột.
    - Thay đổi thời gian giữa hai bước thực hiện kế tiếp trong quá trình tìm kiếm đường đi của thuật toán
    - Chọn file input

  Trong quá trình tìm kiếm đường đi, người dùng người dùng có thể tạm dừng, và quay lùi trạng thái trước đó hay đi tới trạng thái kế tiếp của quá trình tìm kiếm (Chỉ áp dụng với bản đồ nhỏ hơn 50 x 50).

  Sau khi đã thiết lập bản đồ và các tham số cần thiết liên quan. Người dùng sẽ chọn Áp dụng -> Play để chương trình thực hiện.



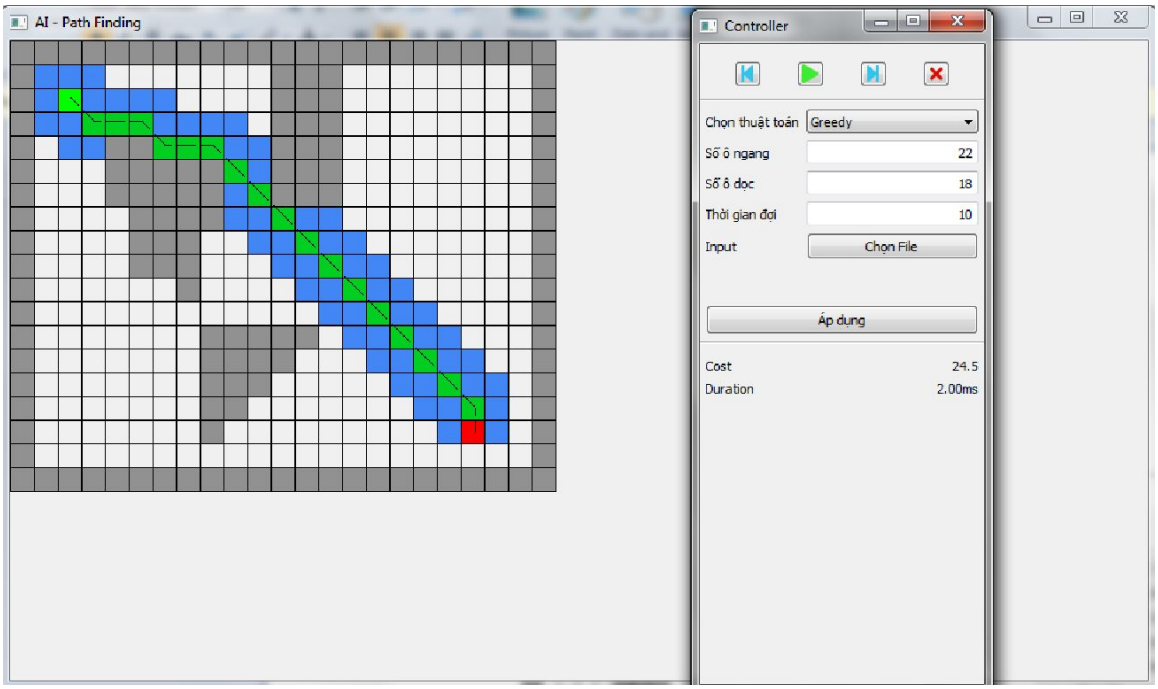
Khi kết thúc quá trình tìm kiếm, chương trình sẽ xuất ra cho người dùng các thông tin:

+ Chi phí của thuật toán. Với chi phí đi ngang =1, chi phí đi chéo =1.5

+ Thời gian chạy thuật toán (ms).

+ Các ô đã đi qua và đường đi ngắn nhất tìm được

Ví dụ:



- **ĐÁNH GIÁ THỰC HIỆN**

Yêu cầu	*Đánh giá*
Mức 1: Cài đặt thành công 1 thuật toán để tìm đường đi từ S tới G.	100%
Mức 2: Cài đặt ít nhất 3 thuật toán khác nhau	100%
Mức 3: Tìm đường đi ngắn nhất từ S đến G, có đi qua tất cả các điểm đón	100%

TÀI LIỆU THAM KHẢO:

https://www.learnpyqt.com/courses/start/basic-widgets/

https://stackoverflow.com/questions/15829782/how-to-restrict-user-input-in-qlineedit-in-pyqt

http://www.angelfire.com/linux/myp/ConvexPolRas/ConvexPolRas.html

https://github.com/OneLoneCoder/videos/blob/master/OneLoneCoder_PanAndZoom.cpp