KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
227 Nguyễn Văn Cừ, Phường 4, Quận 5, TP.HCM
Điện Thoại: (08) 38.354.266 - Fax:(08) 38.350.096

BÁO CÁO ĐỒ ÁN

PROJECT 1.

ROBOT TÌM ĐƯỜNG

MÔN: CƠ SỞ TRÍ TUỆ NHÂN TẠO

GVHD: LÊ NGỌC THÀNH

Nhóm thực hiện:

1. Nguyễn Phục Dược 1712372
2. Nguyễn Nhật Duy 1712387
3. Nguyễn Quí Em 1712399

MỤC LỤC

NỘI DUNG

1. Thông tin tổng quan

- Ngôn ngữ sử dụng: Python 3.7.3
- Python Package: numpy 1.16.4, matplotlib 3.1.0, Shaply 1.6.4.post2
- Text Editor: Visual Studio Code
- Hệ điều hành: Windows 10

2. Ý tưởng thực hiện

Tổ chức đồ án thành các module bao gồm 4 modules bfs.py, dfs.py, ucs.py, astar.py tương ứng với các thuật toán tìm kiếm BFS, DFS, UCS, A* và 1 module mapio.py dùng để đọc ghi file tạo bản đồ, mở rộng bản đồ,..

3. Phân công

STT	Họ và tên	MSSV	Công việc
1	*Nguyễn Phục Dược*	*1712372*	Viết module dfs, báo cáo
2	*Nguyễn Quí Em*	*1712399*	Viết module bfs, mapio
3	*Nguyễn Nhật Duy*	*1712387*	Viết module astart, mapio

4. Chi tiết các modules của đồ án

4.1. Module mapio

Module này có các hàm chính:

- loadmap: đọc bản đồ từ file 'input.txt'.
- fill_polygon: hàm này sẽ tô mà các vùng nằm bên trong một đa giác.
- create_graph: phục vụ cho việc tìm đường đi ngắn nhất qua các điểm đón, tạo ra ma trận kề với chi phí đường đi và đường đi.
- find_best_path: tìm đường đường đi ngắn nhất qua các điểm đón (sử dụng kỹ thuật brute force nên khi số điểm đón tăng thì thời gian tìm kiếm tăng rất lớn).
- fill_color: dùng để tô màu đường đi.
- find_path: hiện thực hóa việc tìm đường đi ngắn nhất qua các điểm đón.
- draw: vẽ đồ bản đồ.

4.2. Module bfs

Module này dùng để cài đặt thuật toán tìm kiếm BFS (Breath Fisrt Search)

Mô tả thuật toán:

- Dùng 1 hàng đợi queue để lưu các điểm hiện tại có thể lấy ra để mở rộng
- Dùng mảng visited để lưu các điểm được mở.
- Trong khi hàng đợi còn phần tử:
    - Lấy 1 điểm ra
        - Nếu điểm đó là goal thì dừng thuật toán
        - Ngược lại mở rộng điểm hiện tại ra các hướng có thể
        - Nếu điểm được mở rộng chưa nằm trong visited và queue thì thêm vào queue, đánh dấu và cập nhật đường đi cùng chi phí.

4.3. Module dfs

Dùng để cài đặt thuật toán tìm kiếm DFS (Depth First Search):

Mô tả thuật toán: giống với bfs nhưng dùng cơ chế stack thay cho queue.

4.4. Module astart

Dùng để cài đặt thuật toán A*:

Mô tả về thuật toán:

- Dùng hàm hueuristic (khoảng cách manhattan) làm hàm ước lượng
- Hàm tìm điểm tối ưu nhất sẽ tim điểm mà tại đó tổng đường đi từ start đến điểm hiện tại và heuristic từ điểm hiện tại đến goal là nhỏ nhất
- Dùng 1 list để lưu các đỉnh đã được mở, thuật toán sẽ dừng khi đã tìm được đích hoặc không còn đỉnh nào mở được nữa (không có đường đi):
    - Lấy 1 điểm được cho là tối ưu nhất có thể đến được đích.
    - Kiểm tra nếu là đích thì trả về kết quả.
    - Nếu không phải là đích: với điểm lấy được, ta mở rộng ra các hướng có thể
        - Với mỗi điểm mở được:
            - Nếu điểm chưa được duyệt hoặc chi phí đường đi mới tốt hơn thì cập nhật lại đường đi mới và các chi phí.

4.5. Module ucs

Dùng để cài đặt thuật toán UCS:

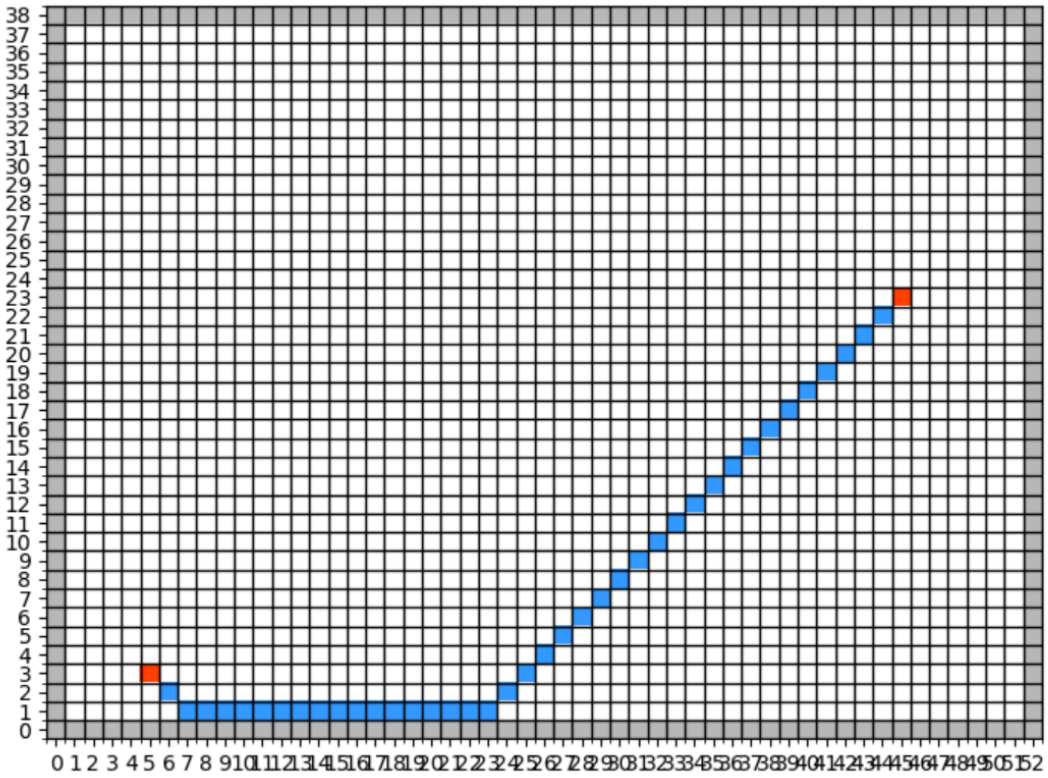
Thuật toán tương tự như A* chỉ khác ở hàm tìm điểm tối ưu ta sẽ chỉ dựa vào đường đi ngắn nhất từ start đến điểm hiện tại.

5. Hình ảnh minh họa chương trình

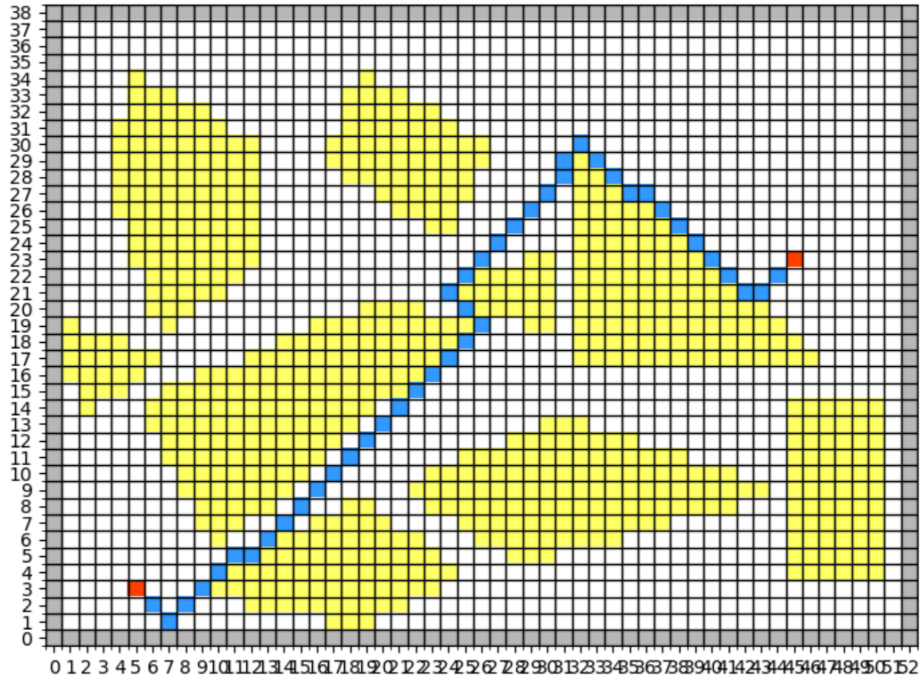
5.1. Breath First Search (BFS)

- Xét chạy thử với bản đồ kích thước 52 x 38, điểm S(5, 3) và điểm G(45, 23).

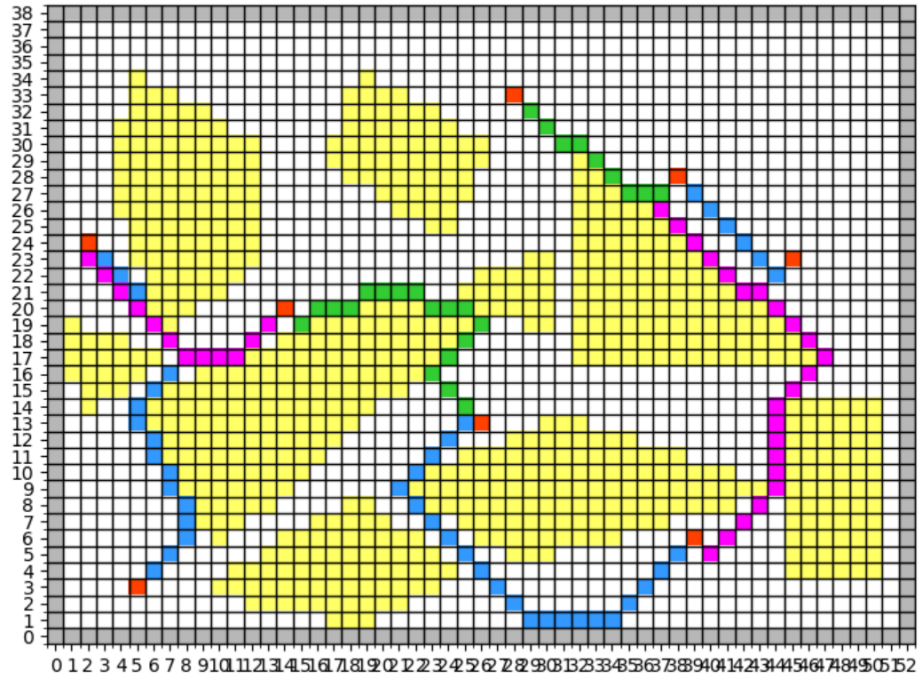
+ Trường hợp 1: không có vật cản



+ Trường hợp 2: có vật cản



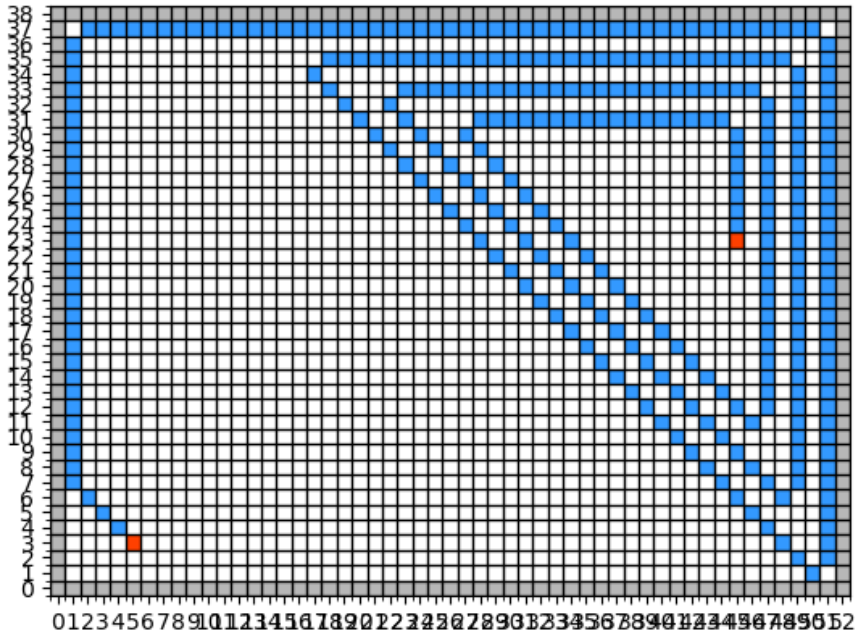
+ Trường hợp 3: có vật cản và điểm đón



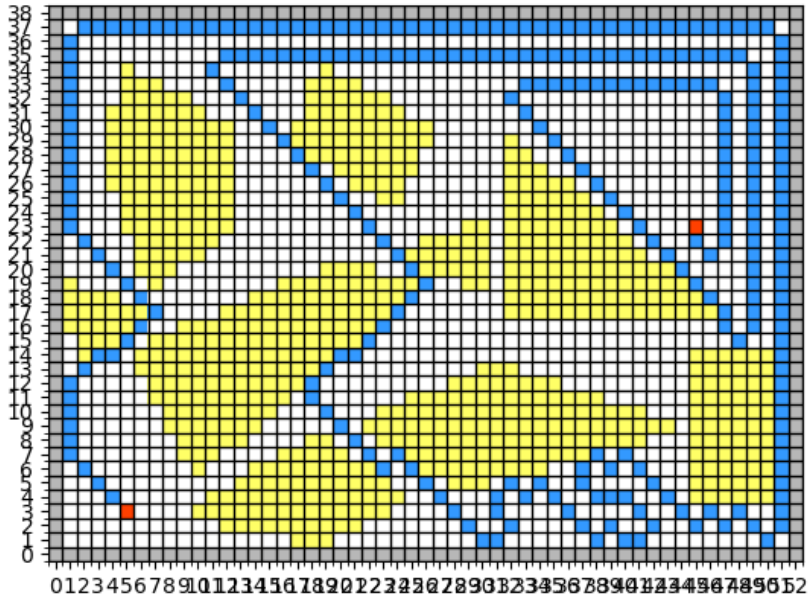
5.2. Depth First Search (DFS)

- Xét chạy thử với bản đồ kích thước 52 x 38, điểm S(5, 3) và điểm G(45, 23).

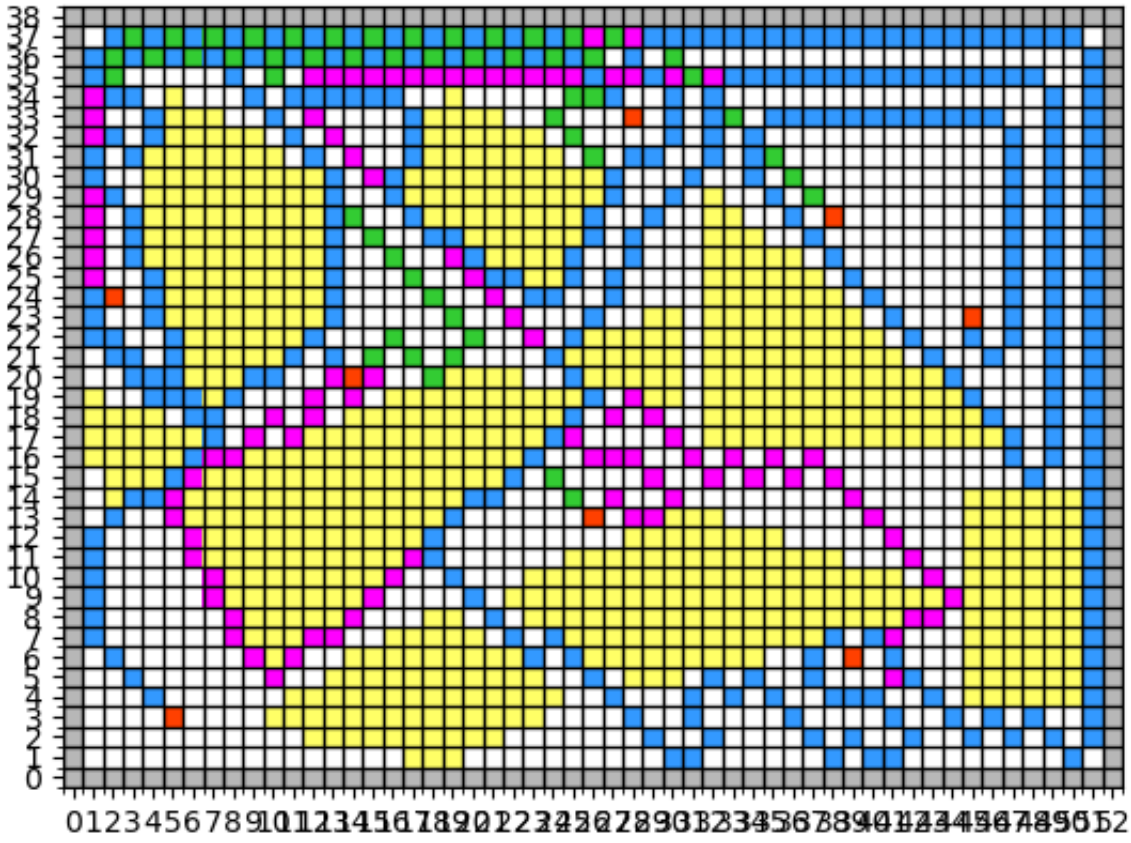
+ Trường hợp 1: không có vật cản



+ Trường hợp 2: có vật cản



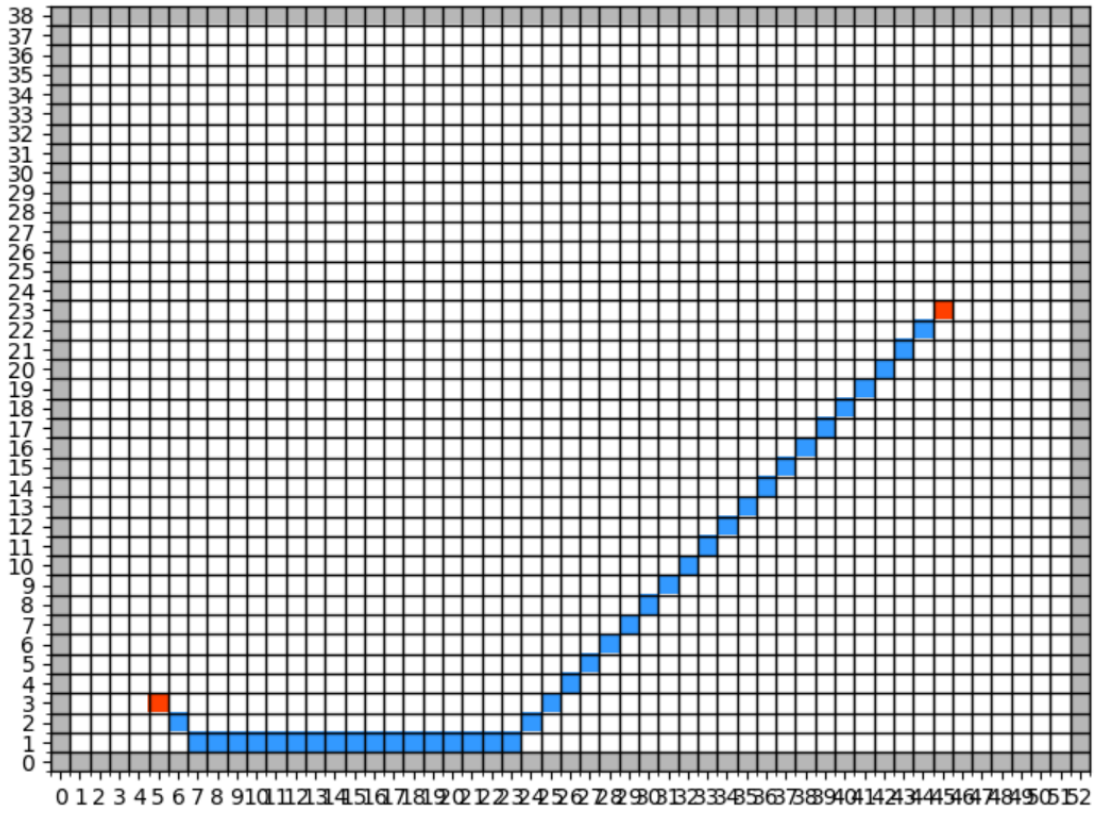
+ Trường hợp 3: có vật cản và điểm đón



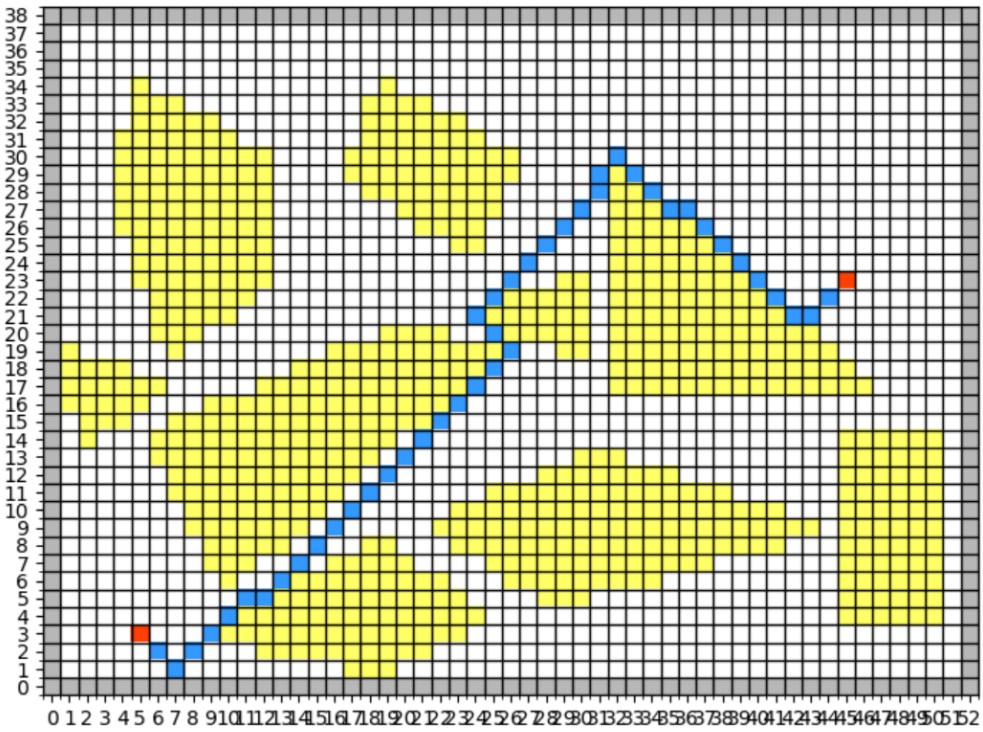
5.3. Uniform Cost Search (UCS)

- Xét chạy thử với bản đồ kích thước 52 x 38, điểm S(5, 3) và điểm G(45, 23).

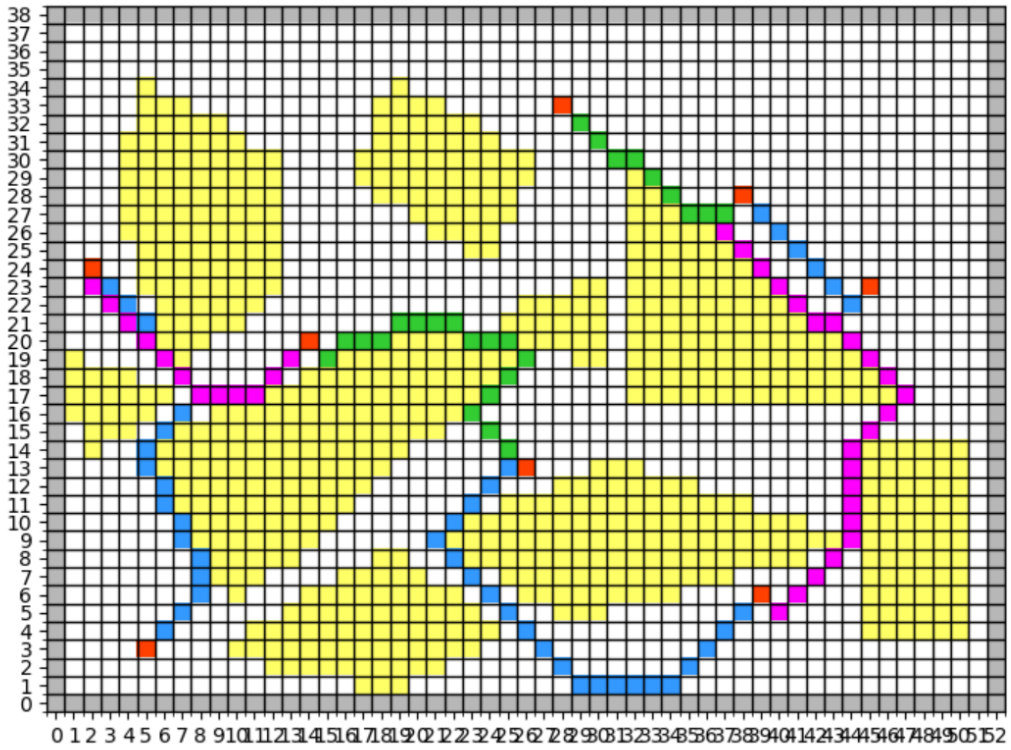
+ **Trường hợp 1: không có vật cản**



+ **Trường hợp 2: có vật cản**



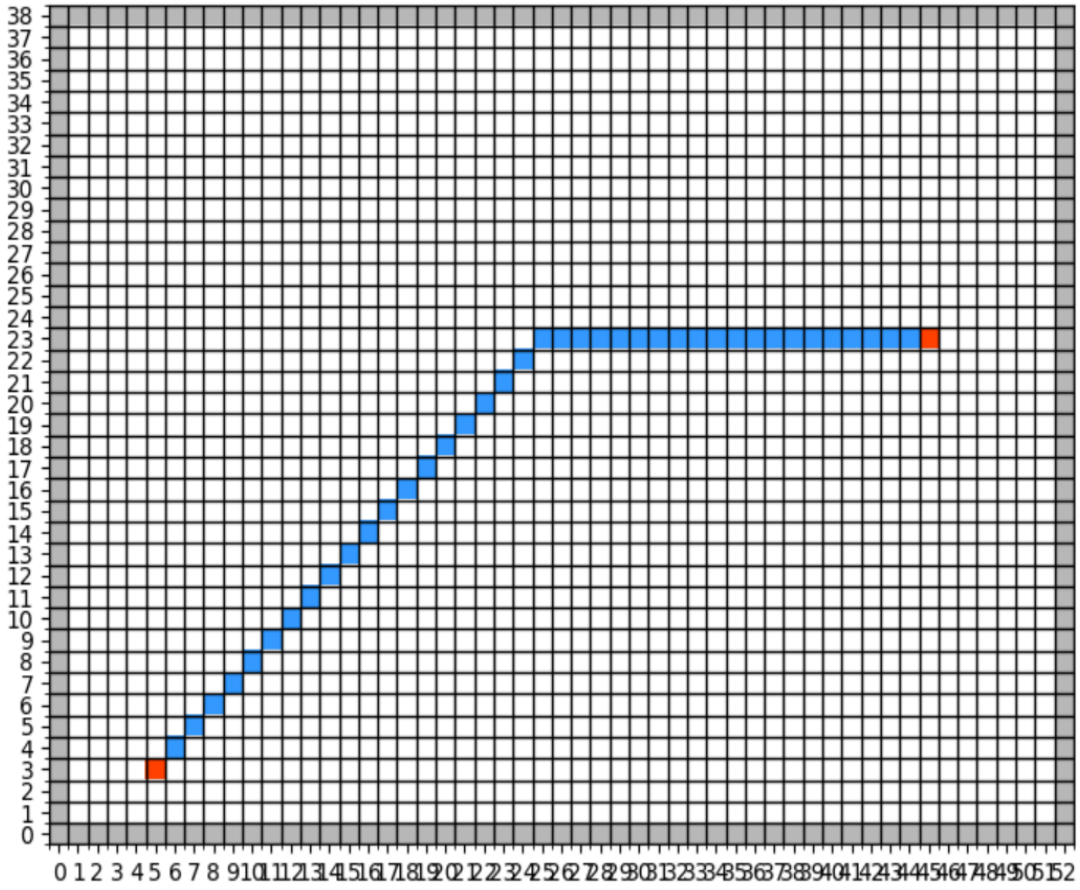
+ **Trường hợp 3: có vật cản và điểm đón**



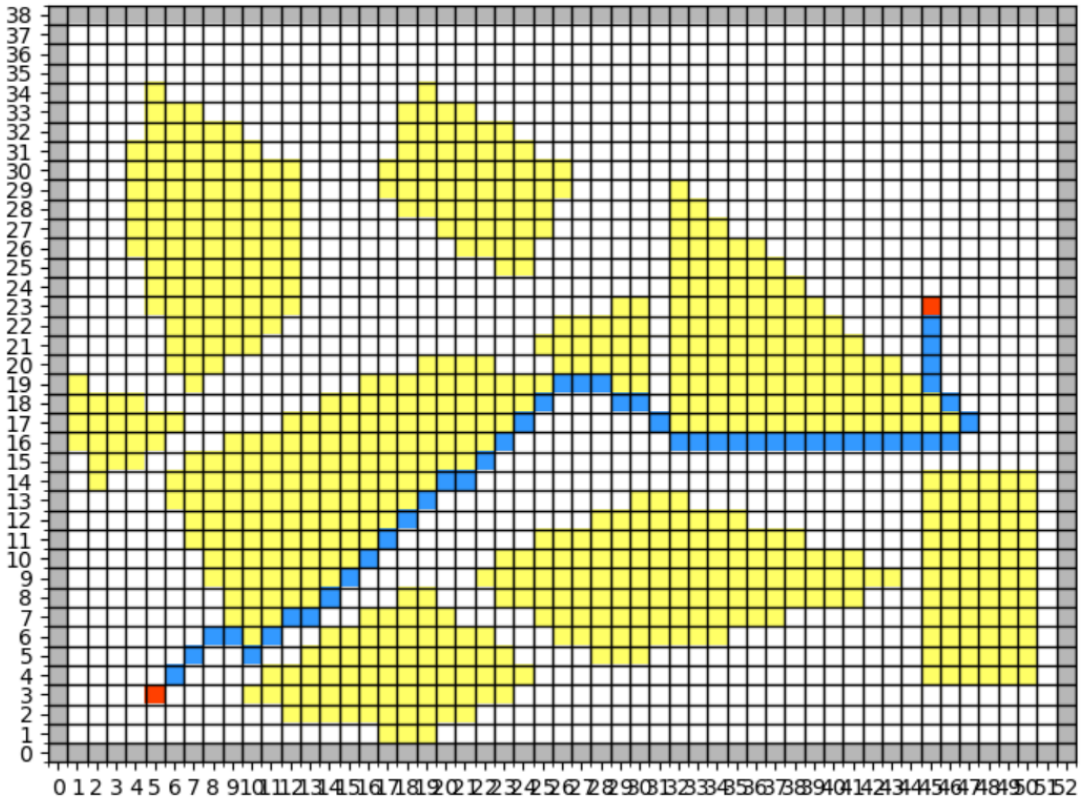
5.4. A Star Search (A*)

- Xét chạy thử với bản đồ kích thước 52 x 38, điểm S(5, 3) và điểm G(45, 23).

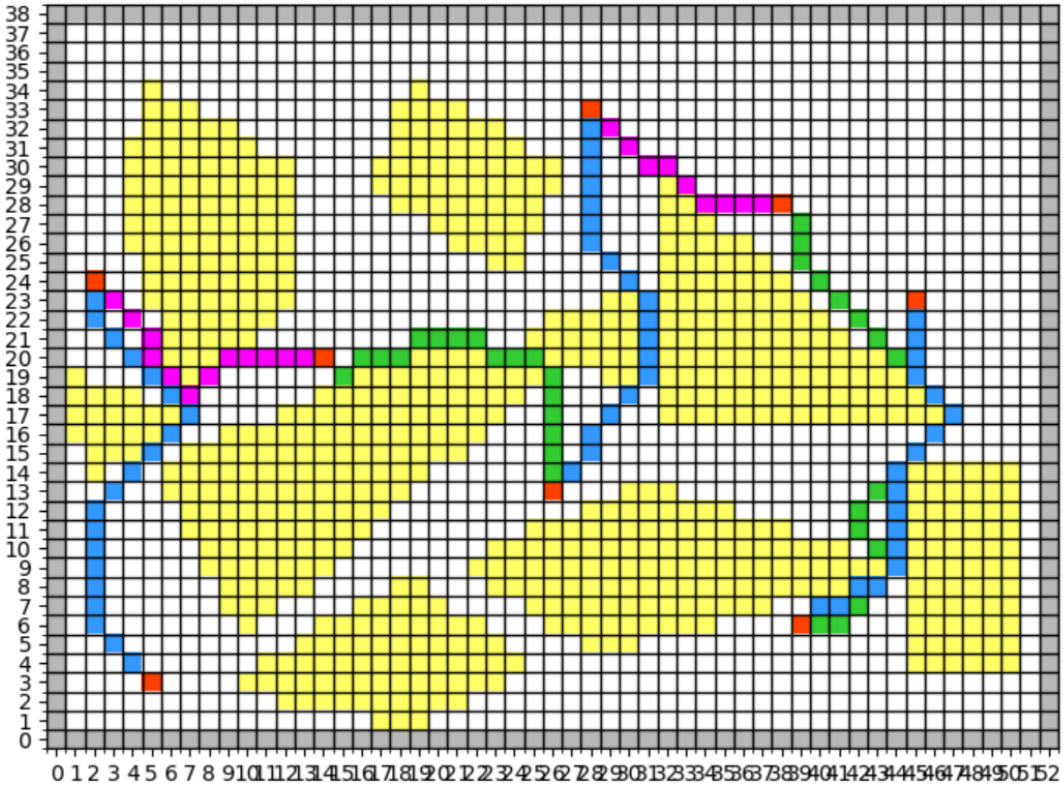
+ **Trường hợp 1: không có vật cản**



+ Trường hợp 2: có vật cản



+ Trường hợp 3: có vật cản và điểm đón



➢ Bảng thống kê **thời gian trung bình**(s hoặc ms) xử lí của các thuật toán được chạy **3 lần**:

STT	*Thuật toán*	*Size map*	*TH không có vật cản*	*TH có vật cản*	*TH có vật cản và điểm đón*
1	A* Search	**22 x 18**	0.0084s	0.015s	0.12s
	UCS		0.13s	0.10s	0.92s
	BFS		0.20s	0.096s	0.89s
	DFS		0.11s	0.063s	0.51s
2	A* Search	**52 x 38**	0.03s	0.145s	1.39s
	UCS		1.05s	0.94s	10.72s
	BFS		1.82s	0.85s	13.51s
	DFS		0.76s	0.29s	10.66s
3	A* Search	**100 x 100**	0.087s	0.22s	2.81s
	UCS		4.87s	5.35s	68.59s
	BFS		26.10s	26.61s	305.77s
	DFS		2.12s	14.46s	260.39s

Nhận xét. Dựa vào bảng thông kê trên ta rút ra được:
- A* Search cho kết quả tìm kiếm nhanh và tối ưu nhất.
- BFS và DFS lộ rõ khuyết điểm của việc đi qua tất cả các vị trí để tìm đường đi đến đích dẫn đến thời gian tìm kiếm lớn so với A* Search.

6. Đánh giá đồ án

Đánh giá dựa theo mức độ hoàn thành theo yêu cầu đề bài

Mức độ yêu cầu	Nội dung yêu cầu	Mức độ hoàn thành
Mức 1 (40%)	Cài đặt thành công 1 thuật toán đi từ S tới G	100% Điển hình: A*
Mức 2 (30%)	Cài đặt ít nhất 3 thuật toán khác nhau	100% Đã cài đặt 4 thuật toán: BFS, DFS, UCS, A*
Mức 3 (30%)	Đi qua các điểm đón	100%
Mức 4 (điểm cộng 10%)	Các đa giác có thể di động	0%
Mức 5 (điểm cộng 10%)	Thể hiện mô hình 3D	0%

7. Tài liệu tham khảo

- https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/
- https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/
- https://www.geeksforgeeks.org/uniform-cost-search-dijkstra-for-large-graphs/
- https://www.geeksforgeeks.org/a-search-algorithm/
- https://www.redblobgames.com/pathfinding/a-star/introduction.html
- Sách “Cơ trở trí tuệ nhân tạo” – Lê Hoài Bắc, Tô Hoài Việt – NXB Khoa học và Kỹ thuật
- Slide bài giảng – thầy Lê Hoài Bắc, Đại học Khoa Học Tự Nhiên TPHCM