CƠ SỞ TRÍ TUỆ NHÂN TẠO

ĐỒ ÁN 1

ROBOT TÌM ĐƯỜNG

Nhóm CQ2017/22 - 2

1712607 - Nguyễn Văn Hoài Nam

1712615 - Nguyễn Trọng Nghĩa

1712616 - Đinh Văn Ngọc

Khoa Công nghệ Thông tin

Đại học Khoa học Tự nhiên TP HCM

Tháng 10/2019

MỤC LỤC

1

Tổng quan

Thông tin nhóm

MSSV	Họ tên	Email	Tỷ lệ thực hiện
1712607	Nguyễn Văn Hoài Nam	1712607@student.hcmus.edu.vn	
1712615	Nguyễn Trọng Nghĩa	1712615@student.hcmus.edu.vn	
1712616	Đinh Văn Ngọc	1712616@student.hcmus.edu.vn	

Thông tin đồ án

Đồ án này viết bằng ngôn ngữ Python. Mục tiêu chọn và cài đặt ít nhất 3 thuật toán tìm đường đi ngắn nhất từ điểm bắt đầu S đến điểm đích G. Giải thuật thực hiện trên một bản đồ được minh họa bằng thư viện đồ họa. Dữ liệu đọc từ file input có quy định tọa độ điểm bắt đầu S đến điểm đích G, viền giới hạn bản đồ, các vật cản và các điểm đón trung gian. Tọa độ sử dụng là tọa độ nguyên, đường đi ngắn nhất theo chiều ngang - dọc, không có trường hợp đi chéo. Có 3 mức cài đặt, ở mỗi mức chạy thử ít nhất 3 bản đồ khác nhau.

Mức	Yêu cầu	Hoàn thành	Ghi chú
1 (40%)	Cài đặt thành công 1 thuật toán để tìm đường đi từ S tới G	100%	
2 (30%)	Cài đặt ít nhất 3 thuật toán khác nhau	100%	
3 (30%)	Trên bản đồ sẽ có những điểm đón. Từ S đi qua các điểm đón để đến G. Thứ tự các điểm đón không quan trọng	100%	

2

Chi tiết đồ án

2.1

Mức 1: Cài đặt thành công 1 thuật toán để tìm đường đi từ S tới G

Thuật toán breadth first search

Chi tiết thuật toán:

B1: Tạo 1 queue Q. Điểm bắt đầu được push vào Q đầu tiên

B2: Vòng lặp:

Tại mỗi điểm (x,y) khi pop ra từ Q ta xét các điểm kề của nó. Nếu các điểm kề đó hoặc là không thuộc Q, không là điểm đón, không là điểm kết thúc, không là vật cản, không là viền map thì thêm vào Q.

B3:

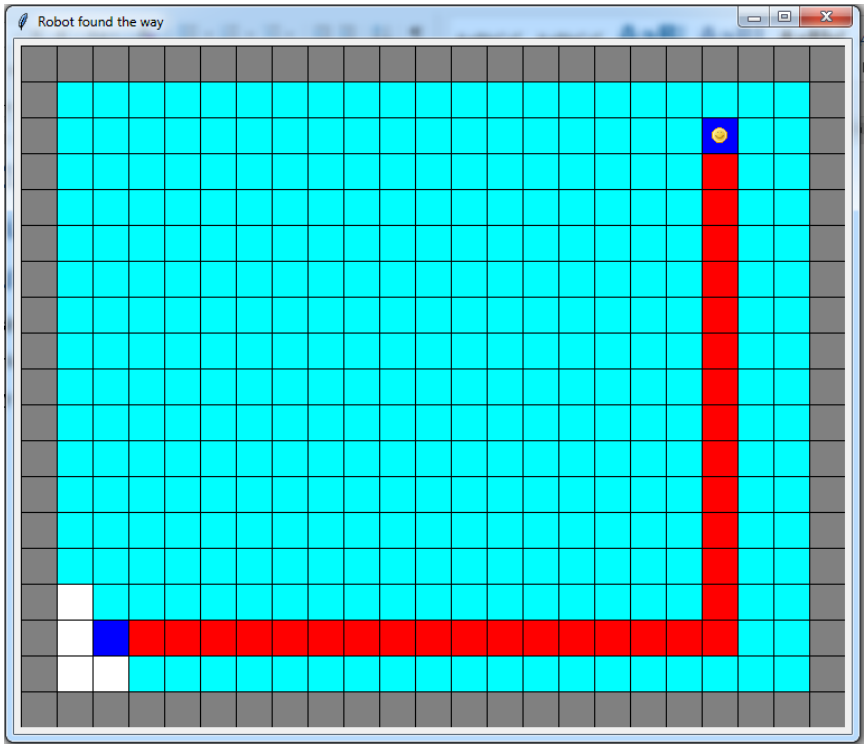
Nếu điểm (x,y) là điểm đón chưa đi qua thì ta trở lại B1 với điểm bắt đầu là (x,y).

Nếu điểm (x,y) là điểm kết thúc và đã đón hết điểm đón thì kết thúc thuật toán - Tìm được đường đi

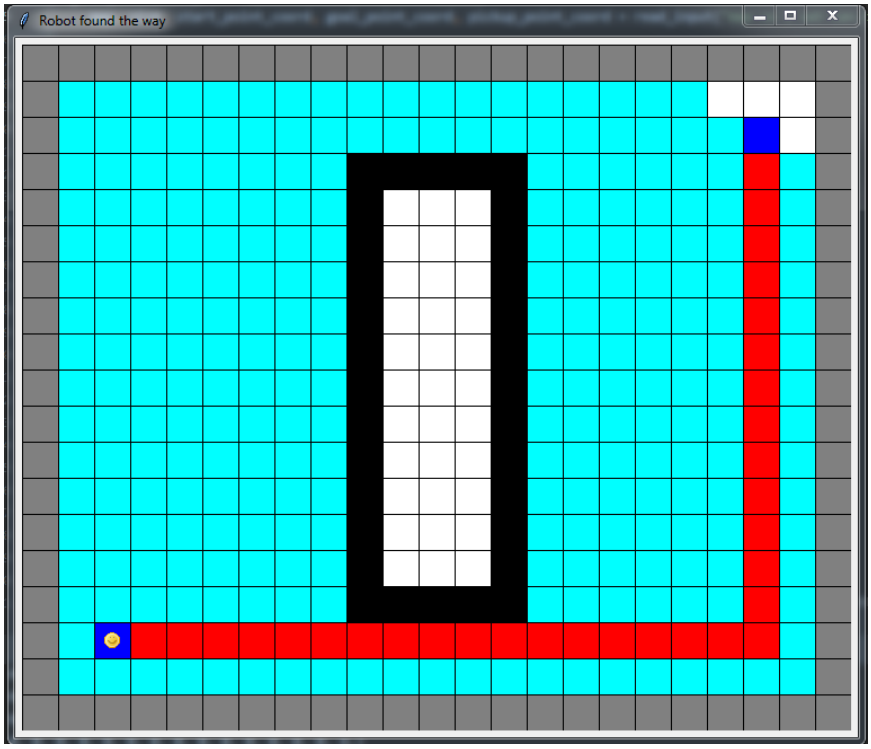
Nếu Q= empty thì kết thúc thuật toán - không có đường đi

Chạy ví dụ:

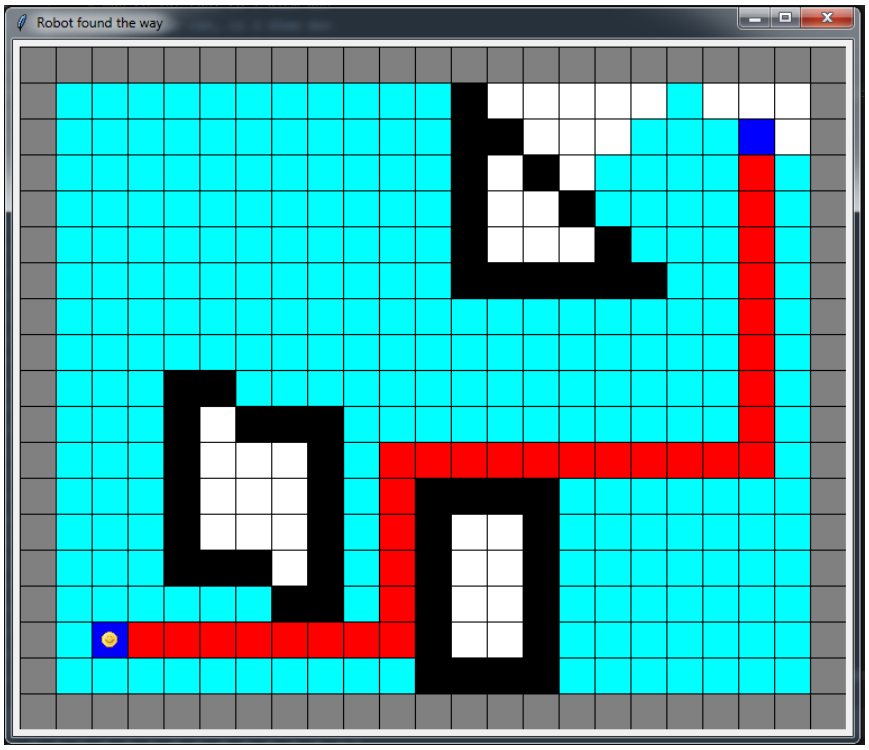
1. Trường hợp bản đồ không có vật cản



2. Trường hợp bản đồ có một vật cản

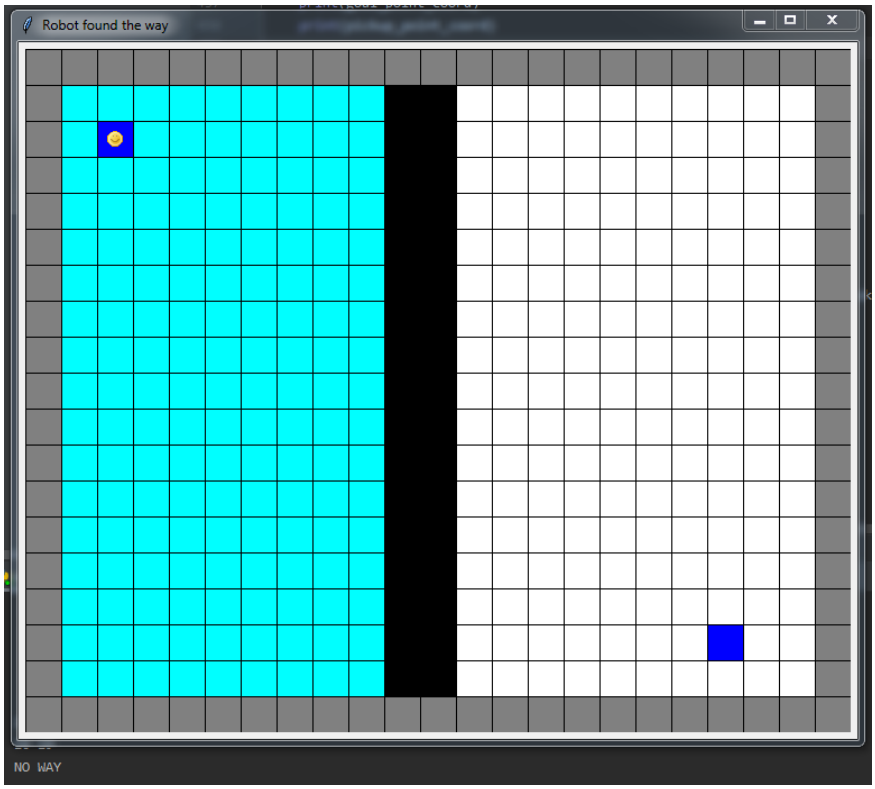


3. Trường hợp bản đồ có ba vật cản



Xét các trường hợp đặc biệt (nếu có):

Trường hợp không có đường đi



Sau khi duyệt hết và không có đường đi sẽ in ra "NO WAY"

Nhận xét:

- Chắc chắn tìm được đường đi nếu có.
- Thời gian: Lâu
- Không gian: Có thể lưu trữ tất cả các node vào queue

2.2
Mức 2: Cài đặt ít nhất 3 thuật toán khác nhau

Thuật toán breadth first search đã trình bày ở trên

Thuật toán greedy best first search

Chi tiết thuật toán:

Heuristic f(x): Chi phí ước tính từ điểm đang xét đến điểm kết thúc |x_end-x| + |y_end-y|

Trước khi vô thuật Greedy, ta dùng chỉnh hợp để sắp xếp các điểm đón sao cho tổng chi phí ước tính nhỏ nhất.

B1: Tạo 1 hàng đợi ưu tiên Q cho chi phí f(x). Điểm bắt đầu được push vào Q đầu tiên

B2: Loop: Tại mỗi điểm (x,y) khi pop ra từ Q ta xét các điểm kề của nó. Nếu các điểm kề đó hoặc là không thuộc Q, không là điểm đón, không là điểm kết thúc, không là vật cản, không là viền map thì thêm vào Q.

B3:

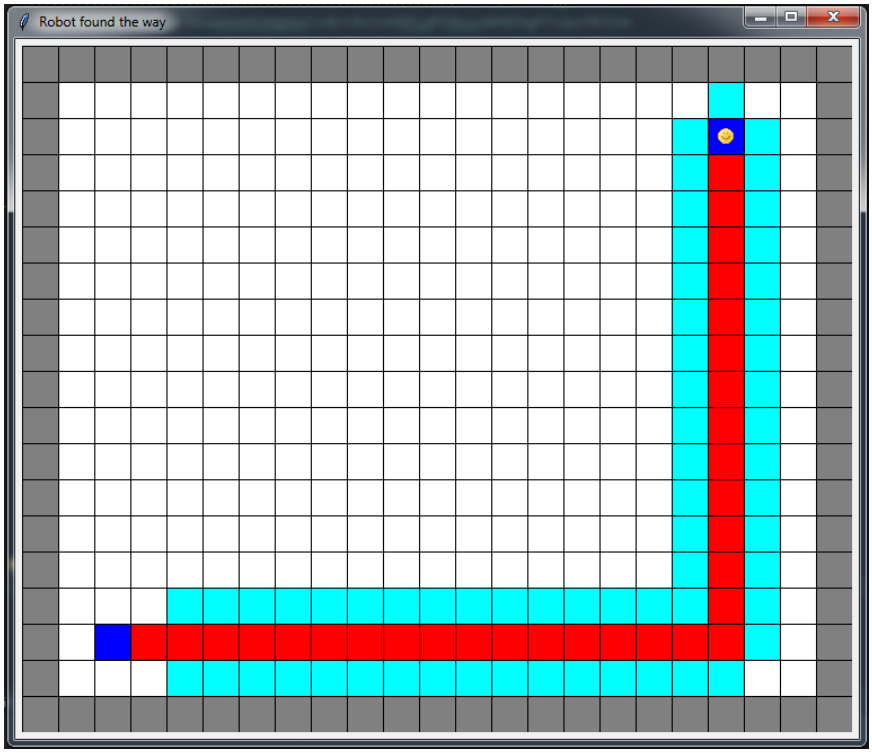
Nếu điểm (x,y) là điểm đón chưa đi qua, thì ta trở lại B1 với điểm bắt đầu là (x,y).

Nếu điểm (x,y) là điểm kết thúc và đã đón hết điểm đón thì kết thúc thuật toán - Tìm được đường đi

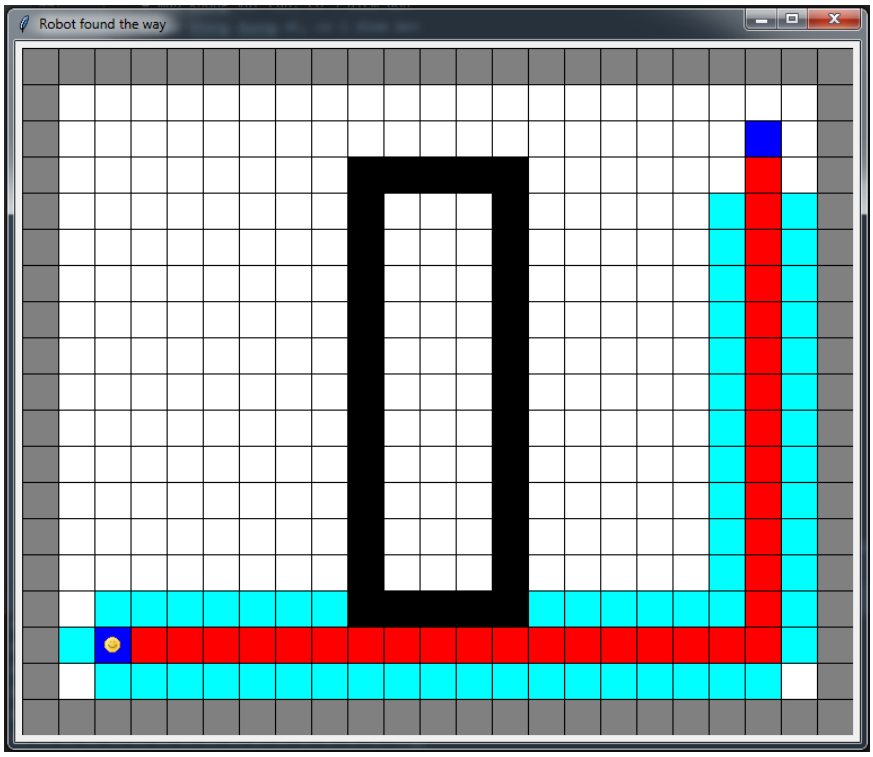
Nếu Q= empty thì kết thúc thuật toán - không có đường đi

Chạy ví dụ:

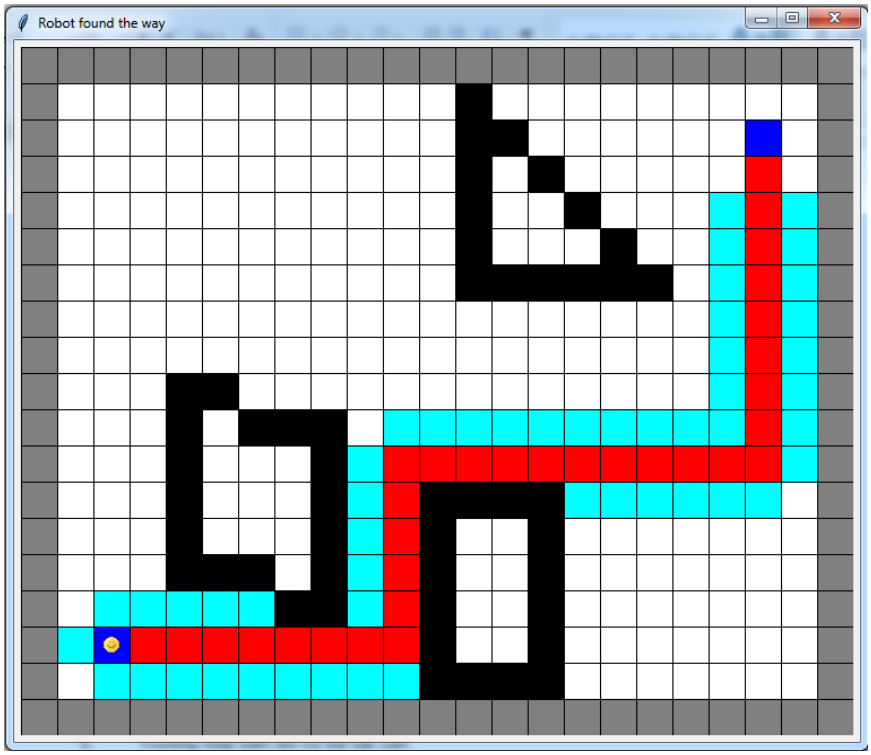
1. Trường hợp bản đồ không có vật cản



2. Trường hợp bản đồ có một vật cản

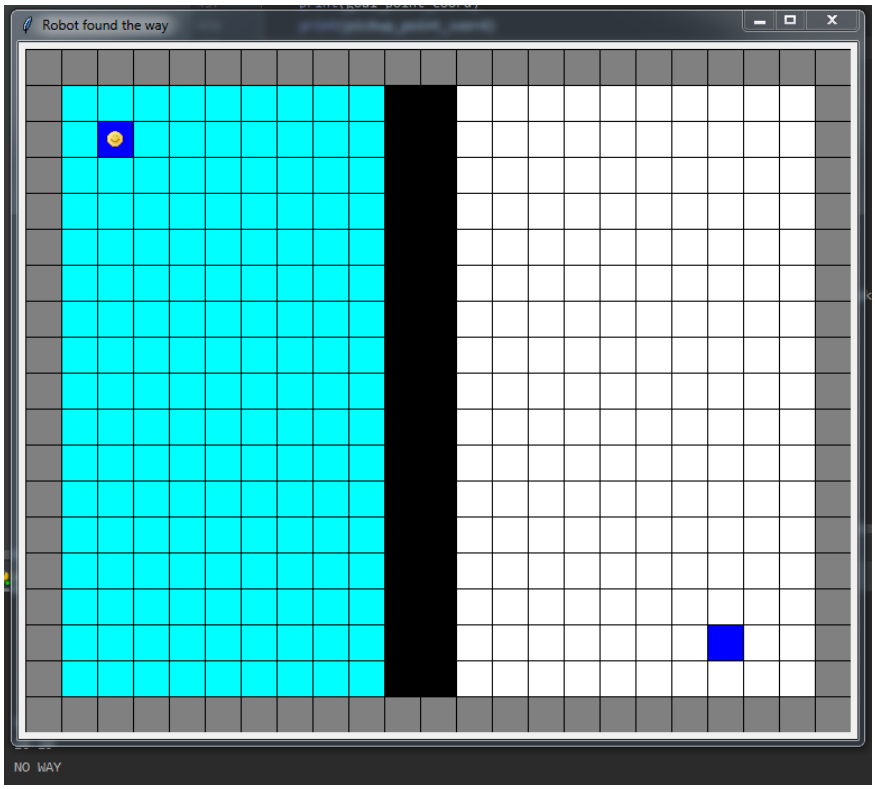


3. Trường hợp bản đồ có ba vật cản



Xét các trường hợp đặc biệt (nếu có):

Trường hợp không có đường đi



Sau khi duyệt hết và không có đường đi sẽ in ra "NO WAY".

Nhận xét:

- Không đảm bảo tìm kiếm được đường đi ngắn nhất (nếu có), nhưng tìm được 1 đường đi trong thời gian nhanh nhất
- Thời gian: Nhanh
- Không gian: có thể thêm tất cả các node vào queue.

Thuật toán A*

Chi tiết thuật toán:

f(x)=g(x) +h(x)

- g(x): Chi phí thực tế từ điểm bắt đầu đến điểm đang xét.
- h(x): Chi phí ước tính từ điểm đang xét đến điểm kết thúc |x_end-x| + |y_end-y|.

Trước khi vô thuật A*, ta dùng chỉnh hợp để sắp xếp các điểm đón sao cho tổng chi phí ước tính nhỏ nhất.

B1: Tạo 1 hàng đợi ưu tiên Q cho chi phí f(x). Điểm bắt đầu được push vào Q đầu tiên

B2: Vòng lặp:

Tại mỗi điểm (x,y) khi pop ra từ Q ta xét các điểm kề của nó. Nếu các điểm kề đó hoặc là không thuộc Q, không là điểm đón, không là điểm kết thúc, không là vật cản, không là viền map thì thêm vào Q.

B3:

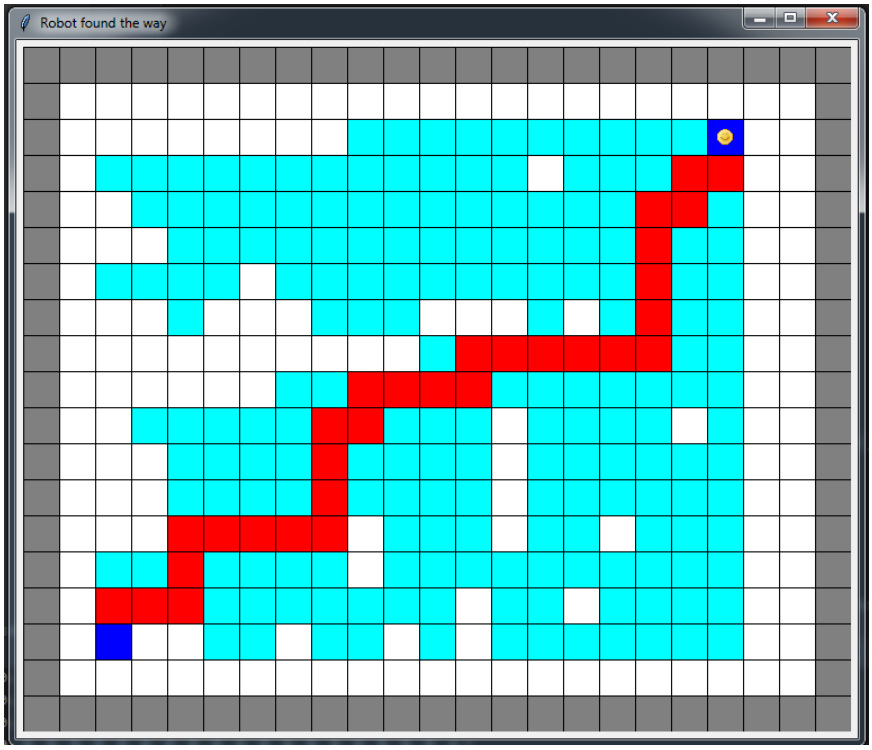
Nếu điểm (x,y) là điểm đón chưa đi qua, thì ta trở lại B1 với điểm bắt đầu là (x,y).

Nếu điểm (x,y) là điểm kết thúc và đã đón hết điểm đón thì kết thúc thuật toán - Tìm được đường đi

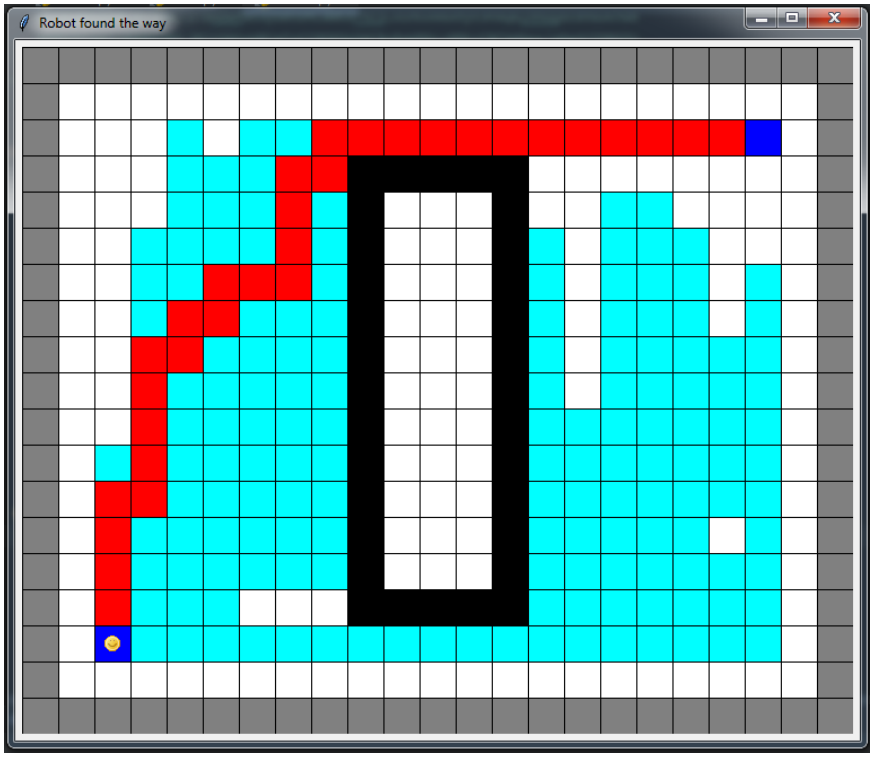
Nếu Q= empty thì kết thúc thuật toán - không có đường đi

Chạy ví dụ:

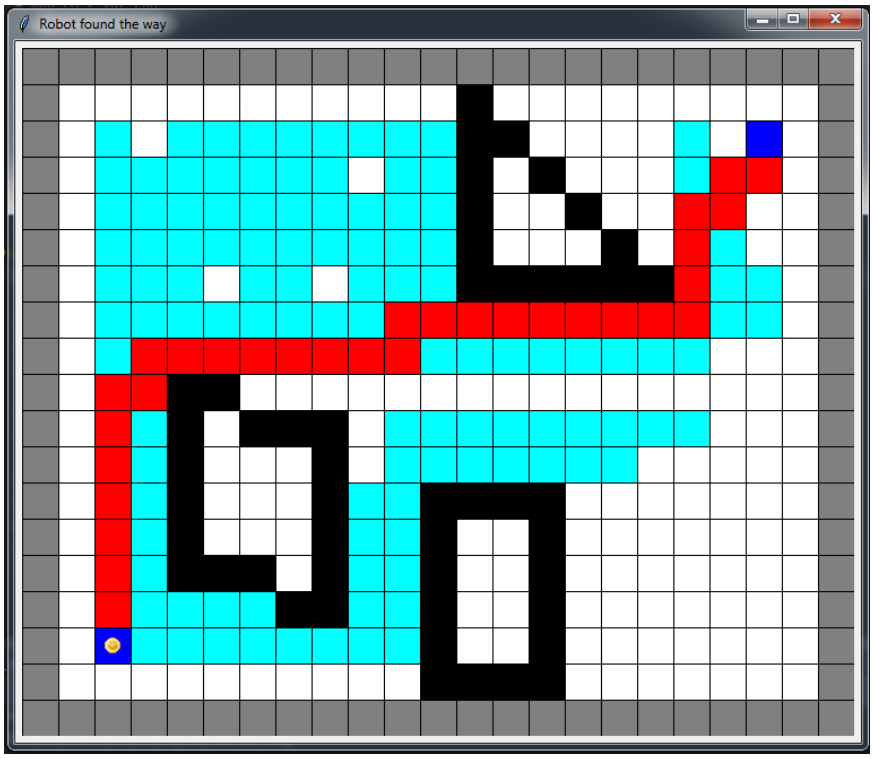
1. Trường hợp bản đồ không có vật cản



2. Trường hợp bản đồ có một vật cản

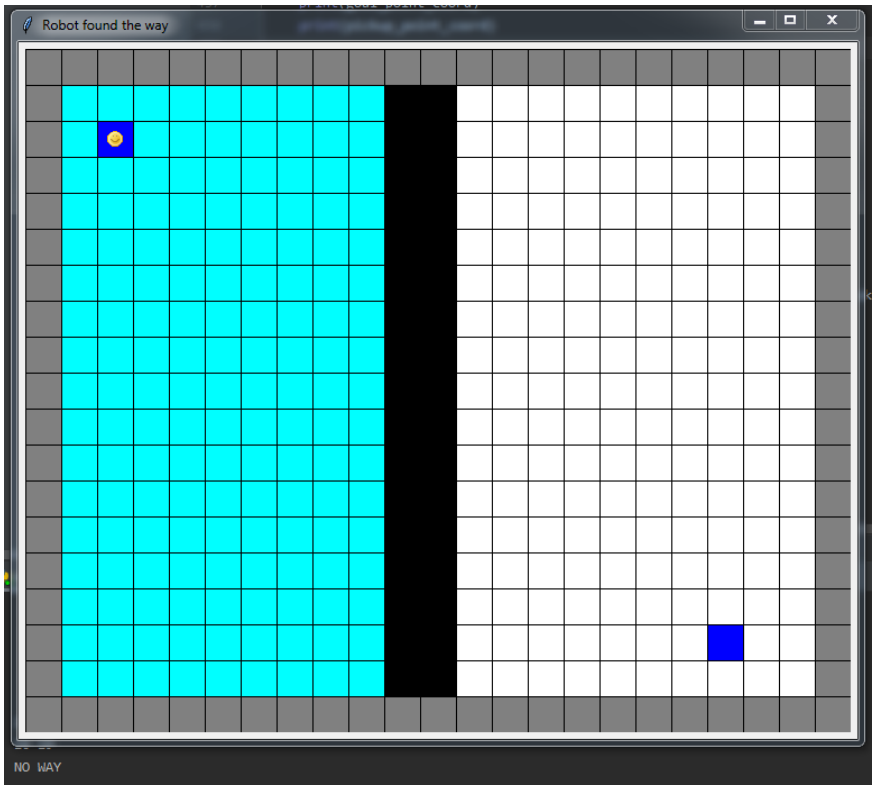


3. Trường hợp bản đồ có ba vật cản



Xét các trường hợp đặc biệt (nếu có):

Trường hợp không có đường đi



Sau khi duyệt hết và không có đường đi sẽ in ra "NO WAY".

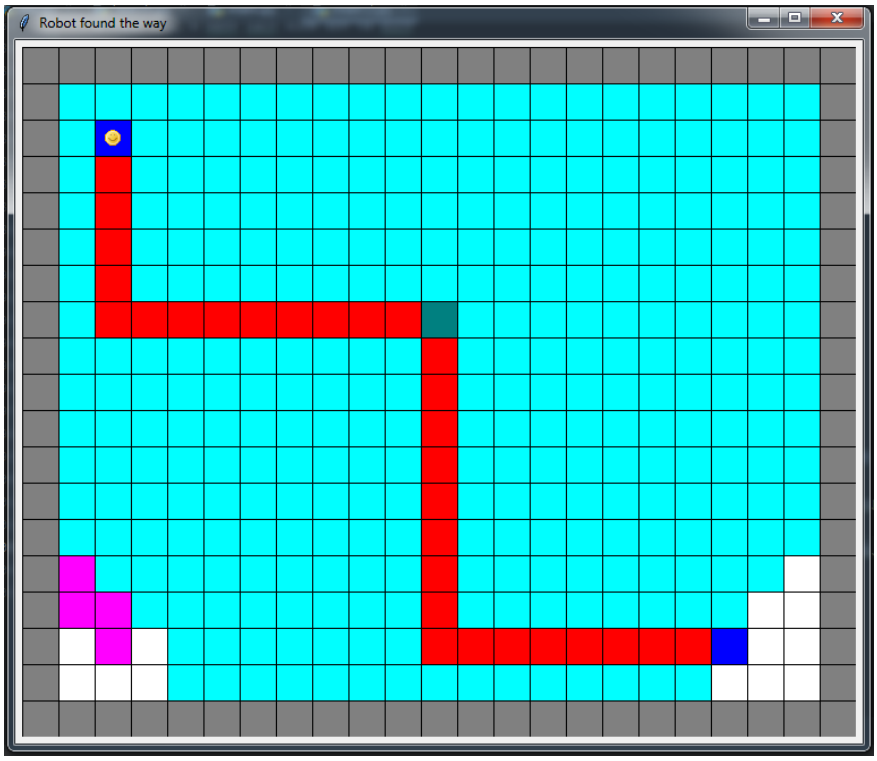
Nhận xét:

- Đảm bảo tìm kiếm được đường đi ngắn nhất (nếu có)
- Thời gian: Vì có heuristic nên tìm kiếm nhanh hơn các thuật tìm kiếm mù khác.
- Không gian: có thể thêm tất cả các node vào queue.

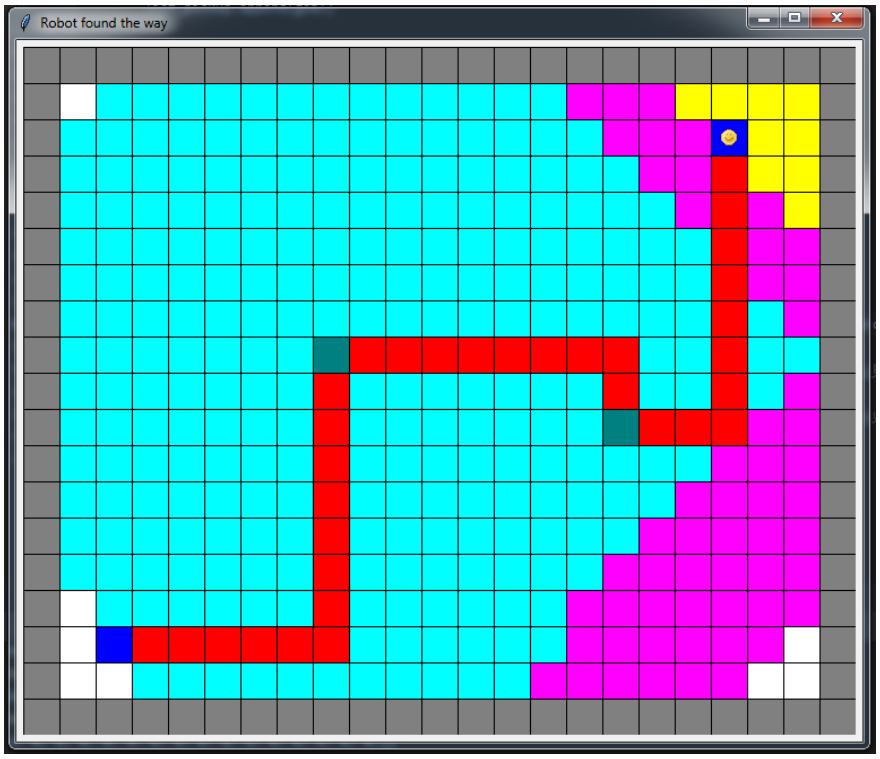
2.3
Mức 3: Trên bản đồ sẽ có những điểm đón. Từ S đi qua các điểm đón để đến G. Thứ tự các điểm đón không quan trọng

Chạy ví dụ:

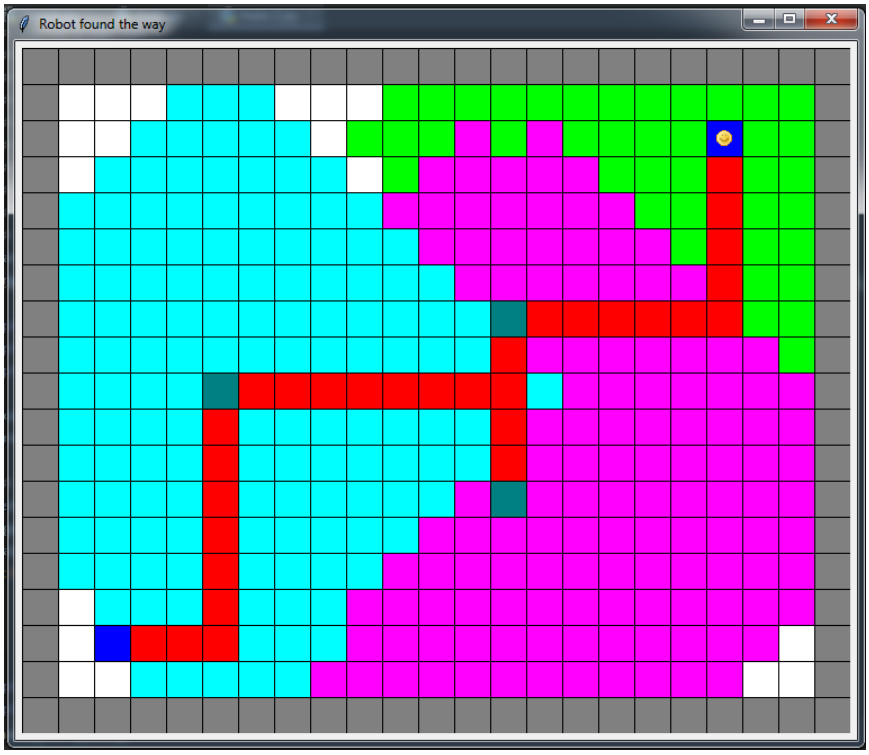
1. Giải thuật **breadth first search** với bản đồ không có vật cản, có 1 điểm đón



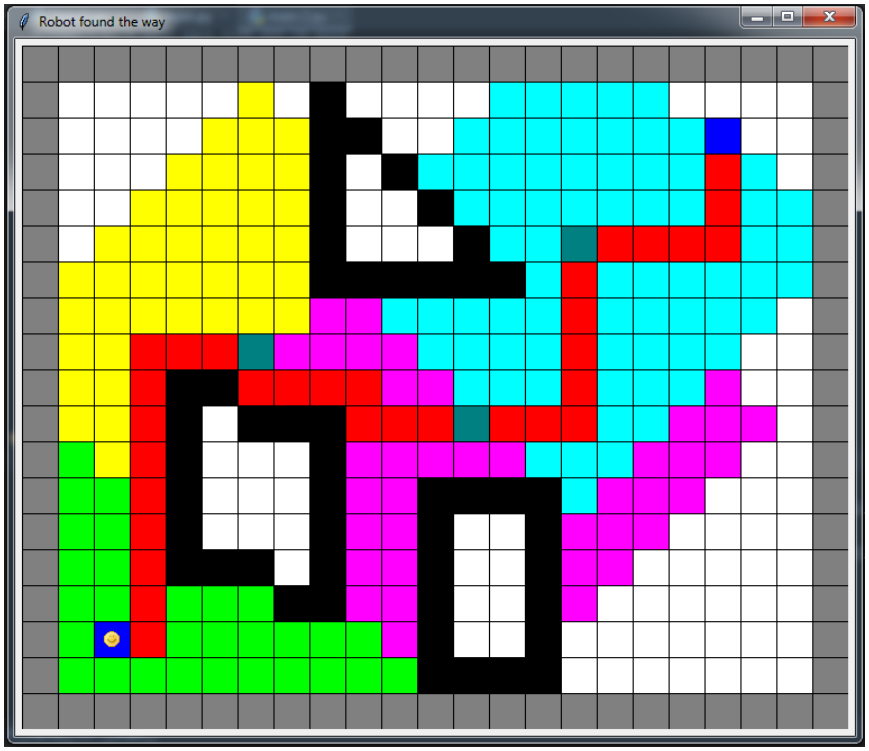
2. Giải thuật **breadth first search** với bản đồ không có vật cản, có 2 điểm đón



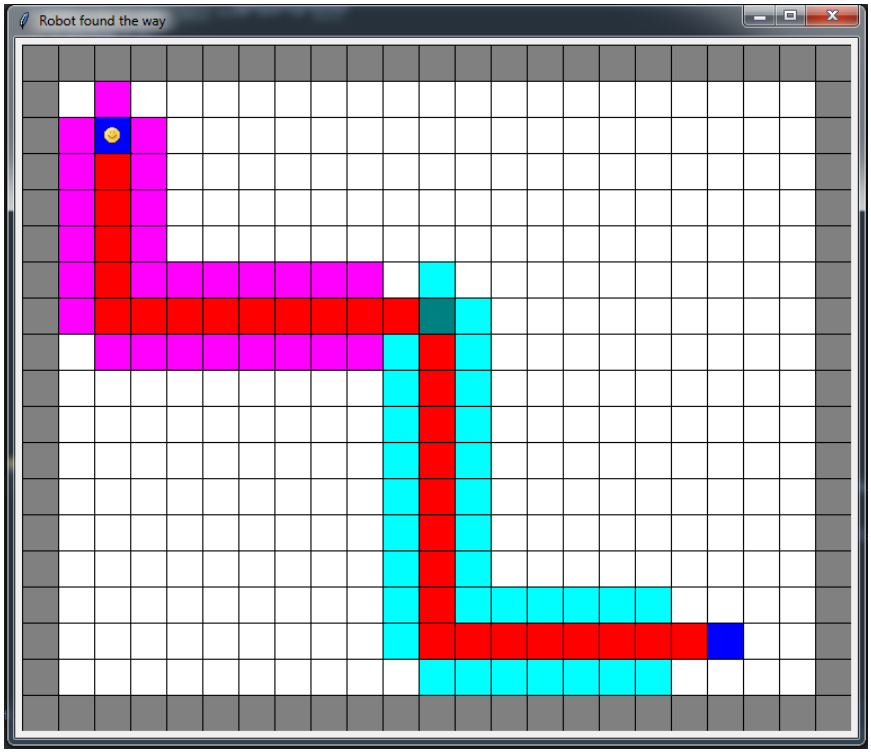
3. Giải thuật **breadth first search** với bản đồ không có vật cản, có 3 điểm đón



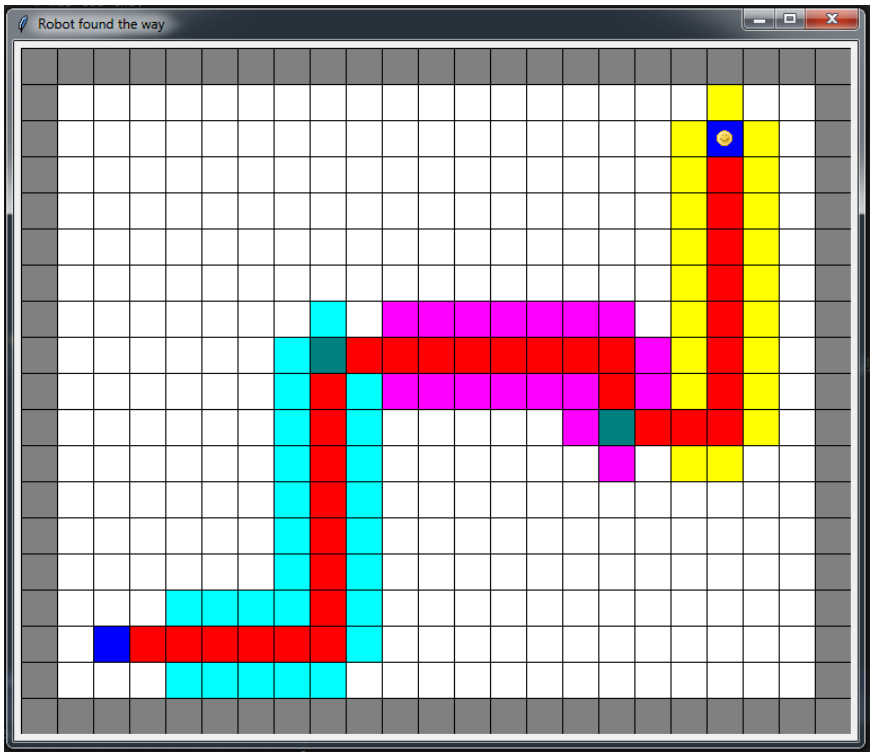
4. Giải thuật **breadth first search** với bản đồ có vật cản có 3 điểm đón



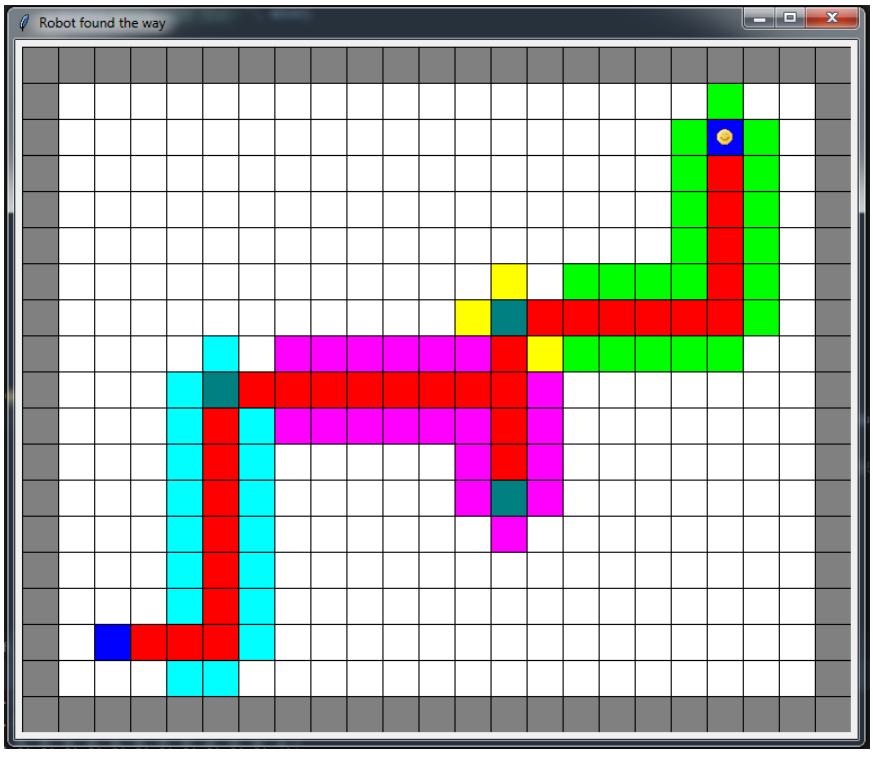
5. Giải thuật **greedy best first search** với bản đồ không có vật cản, có 1 điểm đón



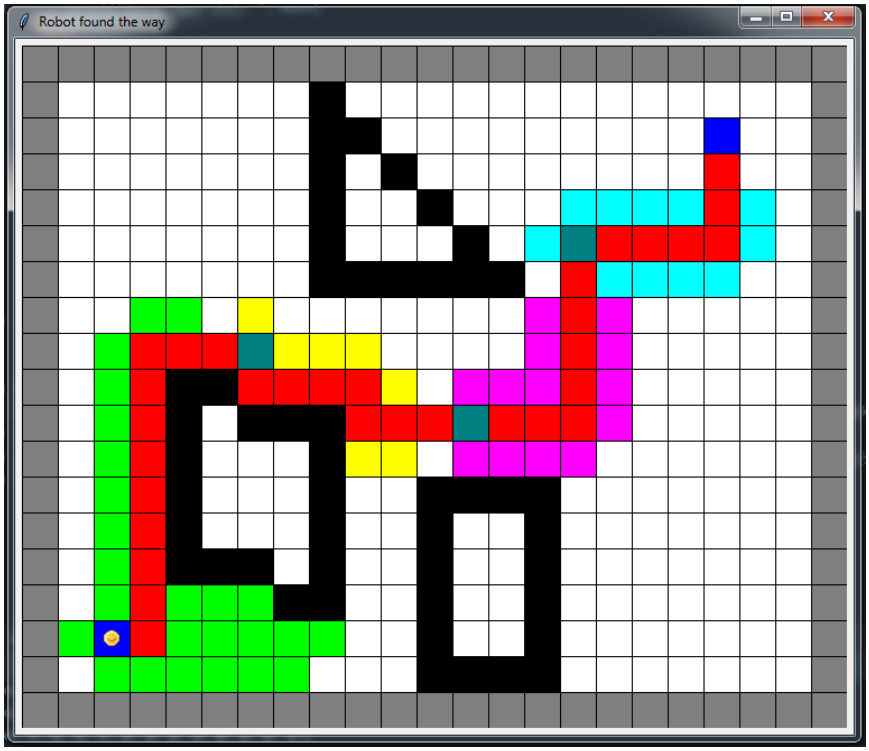
6. Giải thuật **greedy best first search** với bản đồ không có vật cản, có 2 điểm đón



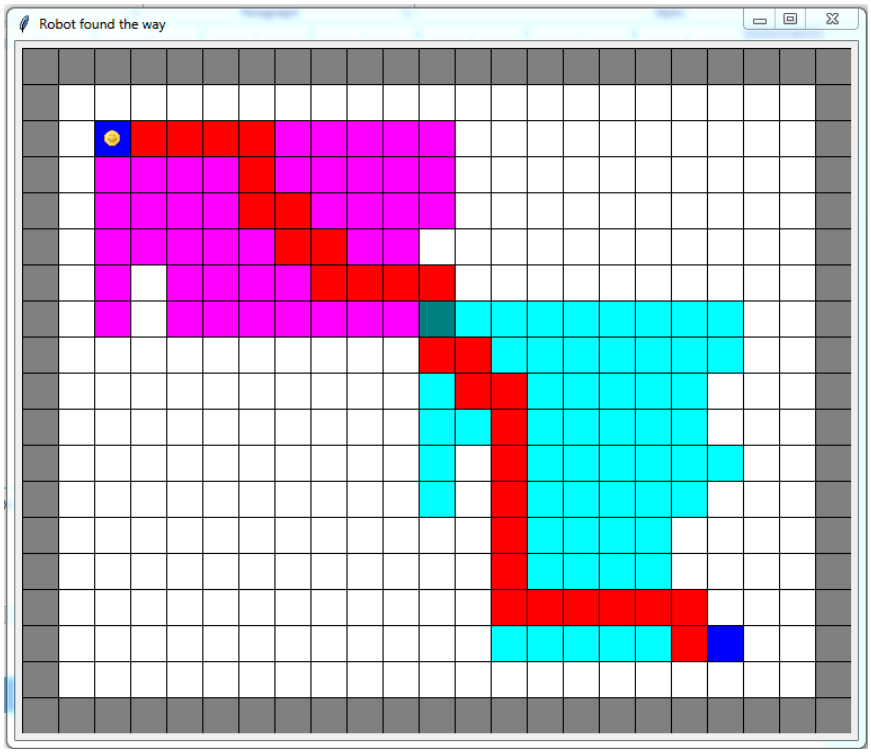
7. Giải thuật **greedy best first search** với bản đồ không có vật cản, có 3 điểm đón



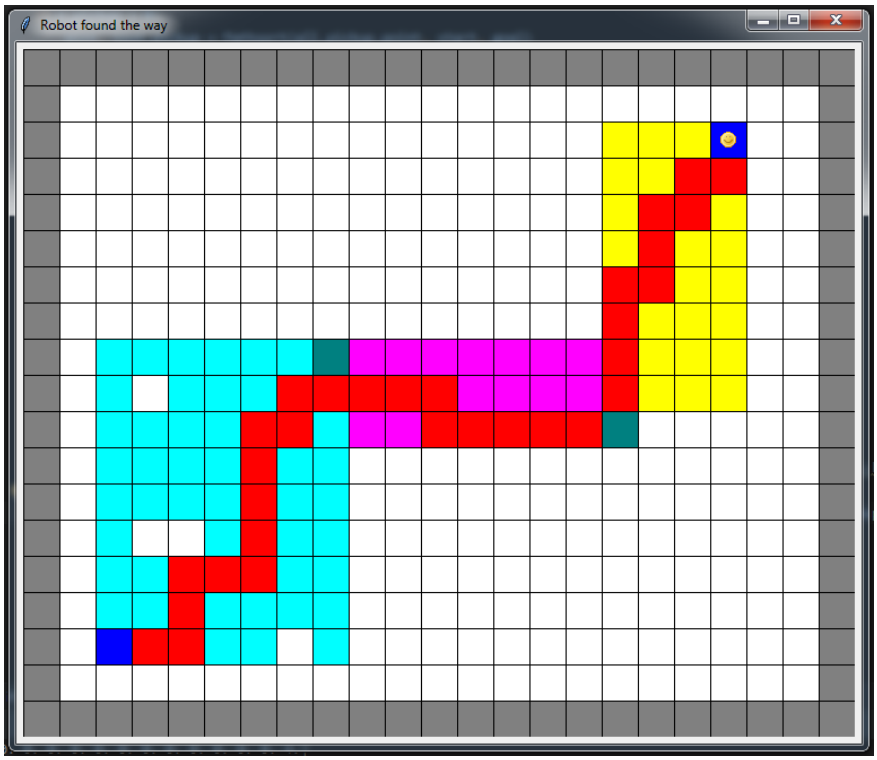
8. Giải thuật **greedy best first search** với bản đồ có vật cản, có 3 điểm đón



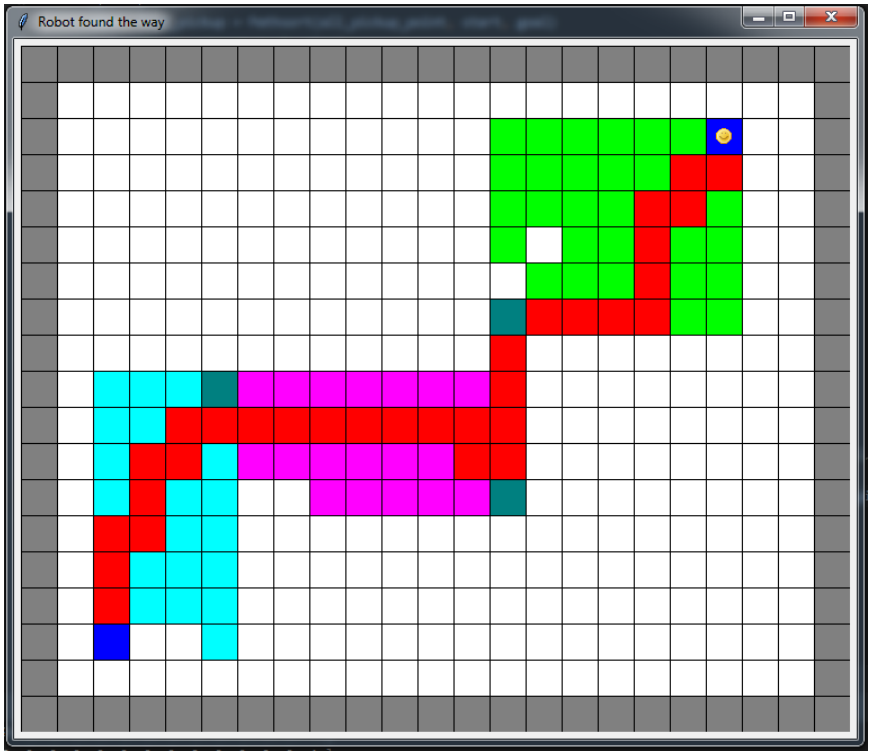
9. Giải thuật **A*** với bản đồ không có vật cản, có 1 điểm đón



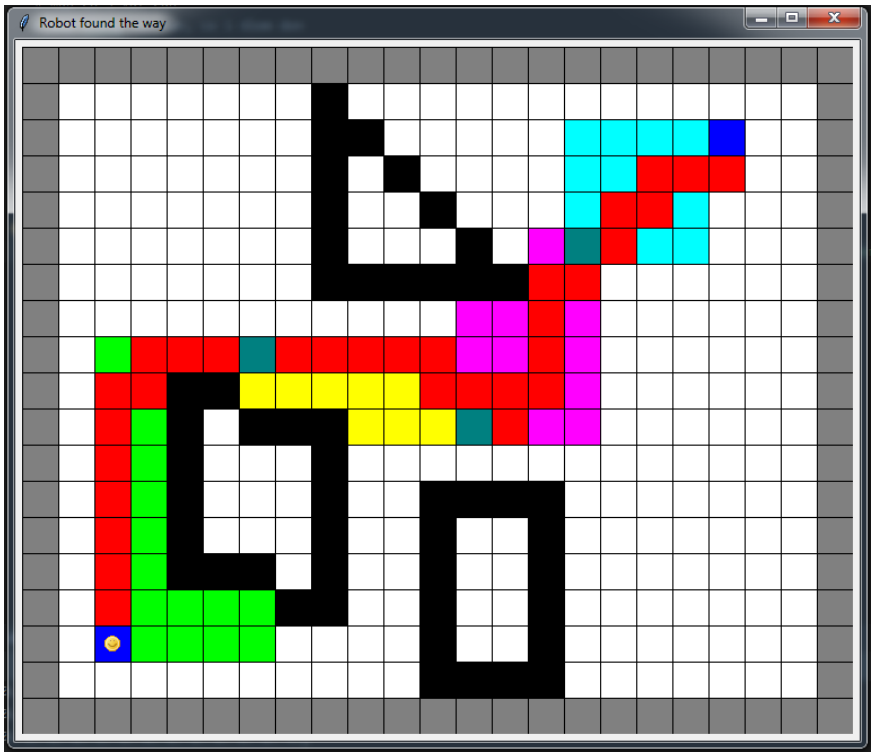
10. Giải thuật **A*** với bản đồ không có vật cản, có 2 điểm đón



11. Giải thuật **A*** với bản đồ không có vật cản, có 3 điểm đón

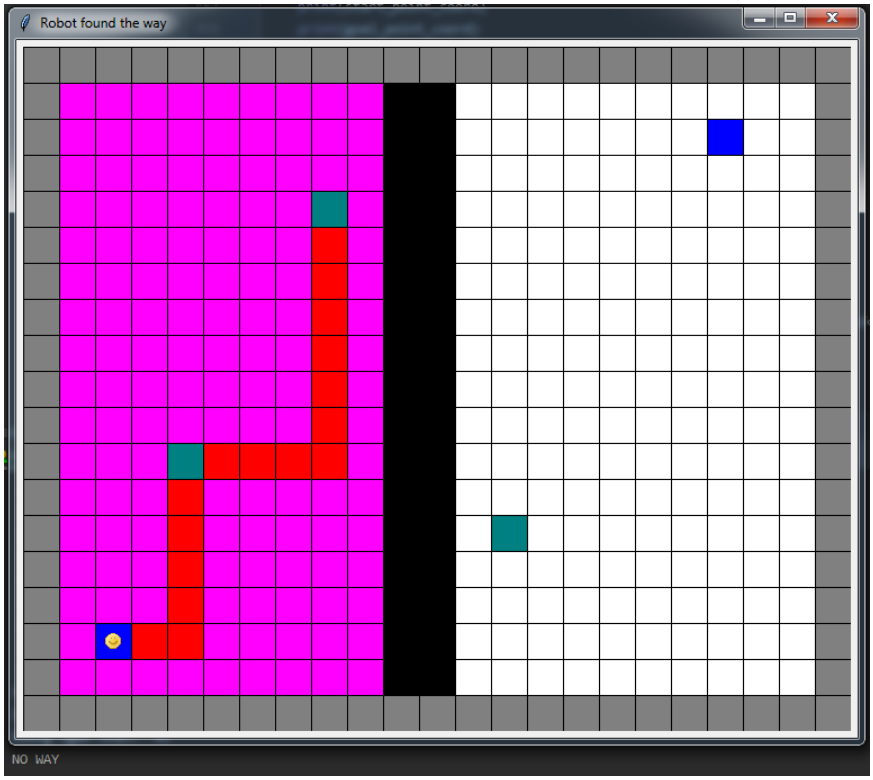


12. Giải thuật **A*** với bản đồ có vật cản, có 3 điểm đón



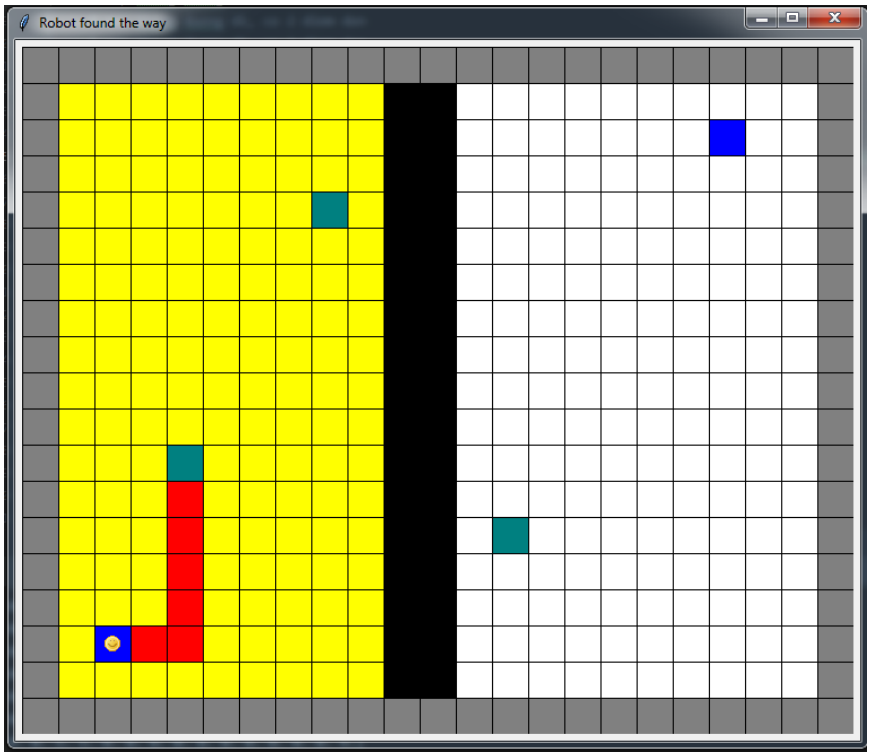
Xét các trường hợp đặc biệt (nếu có):

1. Giải thuật **breadth first search** với bản đồ không có đường đi, có 3 điểm đón



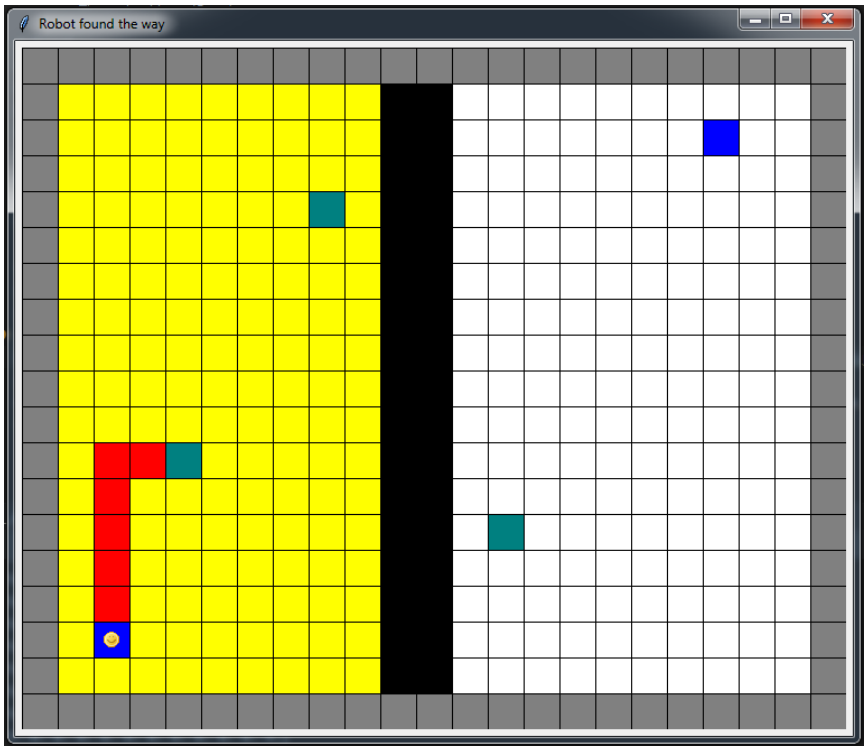
Sau khi duyệt hết và không có đường đi sẽ in ra "NO WAY".

2. Giải thuật **greedy best first search** với bản đồ không có đường đi, có 3 điểm đón



Sau khi duyệt hết và không có đường đi sẽ in ra "NO WAY".

3. Giải thuật **A*** với bản đồ không có đường đi, có 3 điểm đón



Sau khi duyệt hết và không có đường đi sẽ in ra "NO WAY".

Tài liệu tham khảo

matrix - How to define a two-dimensional array in Python

The N-dimensional array (ndarray) — NumPy v1.13 Manual

python - Generating a filled polygon inside a numpy array

Bresenham's line algorithm

Thuật toán Breshenham vẽ đoạn thẳng

A* search algorithm