**BỘ MÔN CƠ SỞ TRÍ TUỆ NHÂN TẠO**

**BÁO CÁO ĐỒ ÁN**

**ROBOT TÌM ĐƯỜNG**

* Giáo viên lý thuyết:
  + Lê Hoài Bắc
* Giáo viên thực hành:
  + Lê Ngọc Thành
  + Nguyễn Ngọc Thảo
  + Nguyễn Ngọc Đức

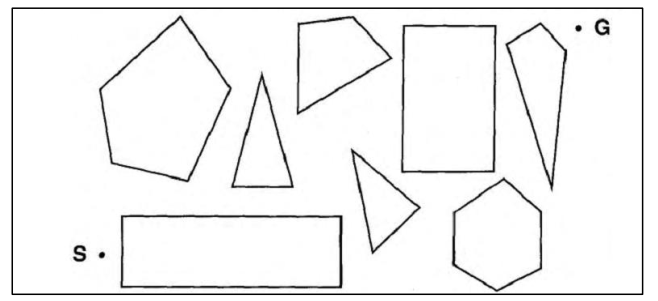
# Thông tin chung

## Thông tin nhóm

|  |  |  |
| --- | --- | --- |
| **MSSV** | **Họ Tên** | **Email** |
| 1712273 | Lâm Đức Anh | [1712273@student.hcmus.edu.vn](mailto:1712273@student.hcmus.edu.vn) |
| 1712838 | Trương Khắc Triệu | 1712838@student.hcmus.edu.vn |
| 1712304 | Hoàng Đức Công | 1712304@student.hcmus.edu.vn |

## Yêu cầu đồ án

* Nghiên cứu, cài đặt và trình bày thuật toán tìm kiếm đường đi.
* Kết hợp cài đặt thuật toán vào chương trình đồ họa.
* Yêu cầu chi tiết:



* + Cho một mặt phẳng xOy ( góc phần tư thứ nhất ), trên đó đặt một điểm bắt đầu S(xs, ys) và một điểm đích G(xG, yG). Đồng thời đặt các chướng ngại vật là các hình đa giác lồi, sao cho đa giác không chồng lên nhau hay có điểm chung. Không gian bản đồ được giới hạn trong một khung hình chữ nhật có góc trái dưới trùng với gốc tọa độ, đồ dày của khung là 1 đơn vị. Không có điểm nào trong bản đồ được vượt qua khung.
  + **Yêu cầu**: cài đặt thuật toán tìm đường đi ngắn nhất từ S đến G sao cho đường đi không được đi xuyên qua đa giác.

# Môi trường lập trình và nội dung thực hiện

## Cấu trúc thư mục chương trình

File cài đặt các hàm hỗ trợ thuật toán

File cài đặt hàm vẽ đồ họa

File chạy chính của chương trình

Lớp biểu diễn cho đa giác

Lớp biểu diễn cho điểm

File cài đặt các hàm hỗ trợ việc tạo các animation

Thư mục chứa file đầu vào

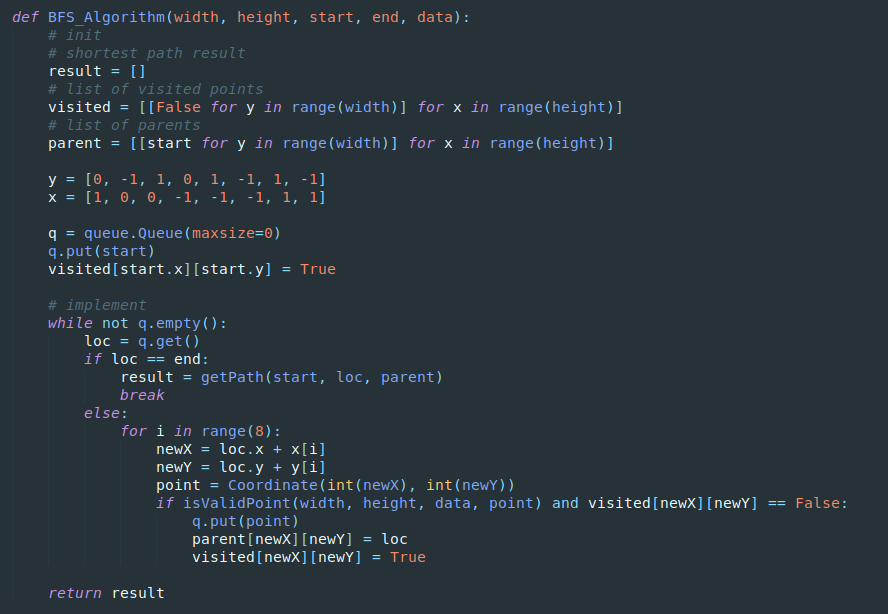
File cài đặt thuật toán tìm kiếm tham lam

File cài đặt thuật toán tìm kiếm A star

File cài đặt thuật toán tìm kiếm theo chiều rộng

## Thuật toán cài đặt trong đồ án

* + 1. **Tìm kiếm mù ( uninformed search )**
       - **Tìm kiếm theo chiều rộng ( Breath First Search)**
         * **Thuật toán**



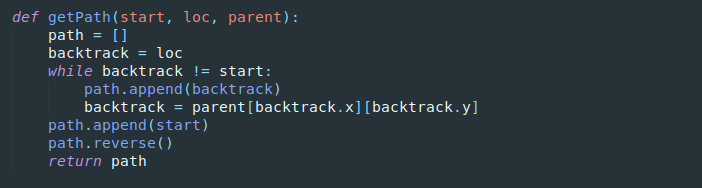
Tham số truyền vào hàm theo thứ tự từ trái sang phải là chiều rộng bản đồ, chiều cao bản đồ, điểm bắt đầu, điểm kết thúc và mảng dữ liệu.

Đầu tiên ta khởi tạo mảng result (mảng lưu đường đi kết quả), mảng visited (mảng lưu trạng thái của các điểm) và mảng parent (mảng lưu lại trạng thái trước đó của điểm hiện tại).

Khởi tạo một hàng đợi (queue ) rỗng. Đặt điểm bắt đầu vào hàng đợi và thay đổi trại thái của điểm bắt đầu thành đã thăm.

Ta lần lượt lấy các điểm trong hàng đợi ra. Kiểm tra điểm đó có phải là điểm cần tới hay không, nếu phải thì ta kết thúc quá trình tìm kiếm và truy vết để tìm đường đi từ điểm bắt đầu tới điểm kết thúc, còn nếu chưa là điểm kết thúc thì ta sẽ xét tất cả các điểm xung quanh điểm hiện tại mà có thể đi được và ta lần lượt đẩy các điểm này vào hàng đợi.

Ta lặp lần lượt quá trình này cho tới khi gặp được điểm kết thúc. Nếu hàng đợi đã trở nên rỗng mà vẫn chưa đạt được trạng thái đích thì đường đi trả về sẽ rỗng.



Đây là hàm truy vết đường đi khi đã đạt được vị trí cần đến.

Giải thích hàm số:

+start: là điểm bắt đầu đường đi.

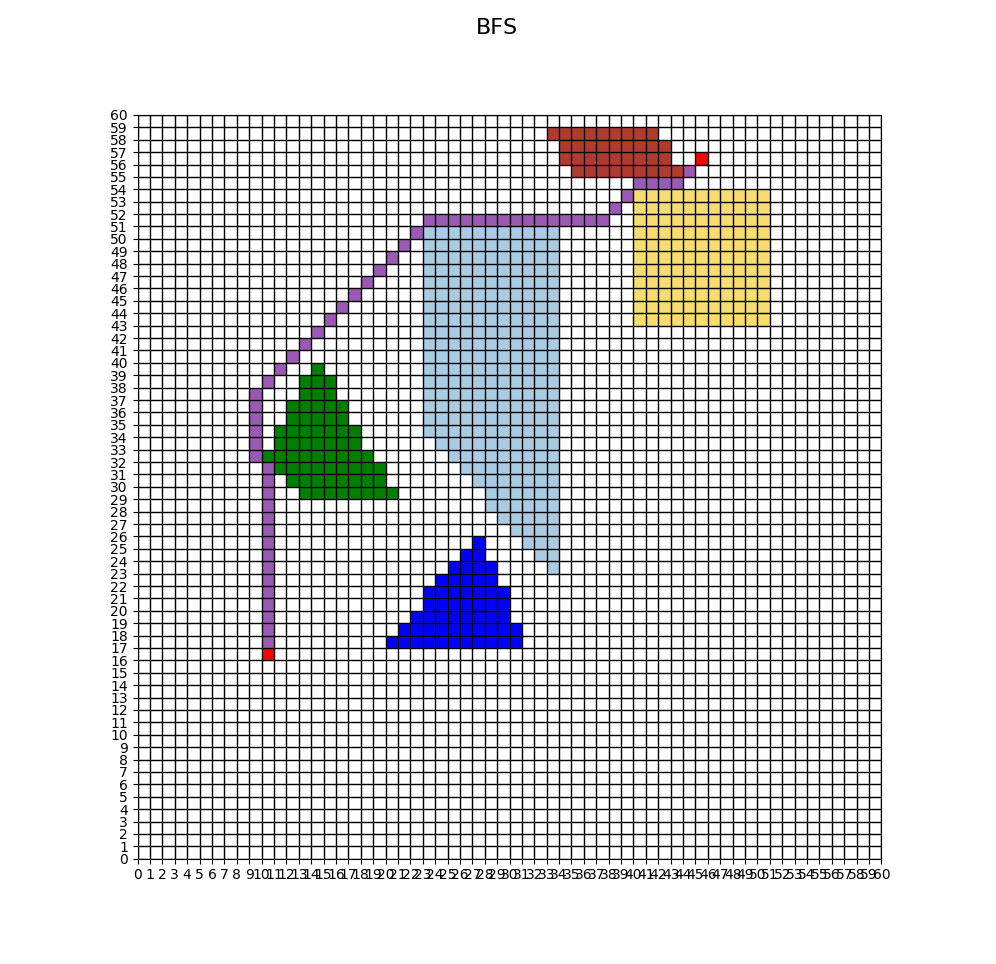
+ loc: là điểm kết thúc (điểm hiện tại)

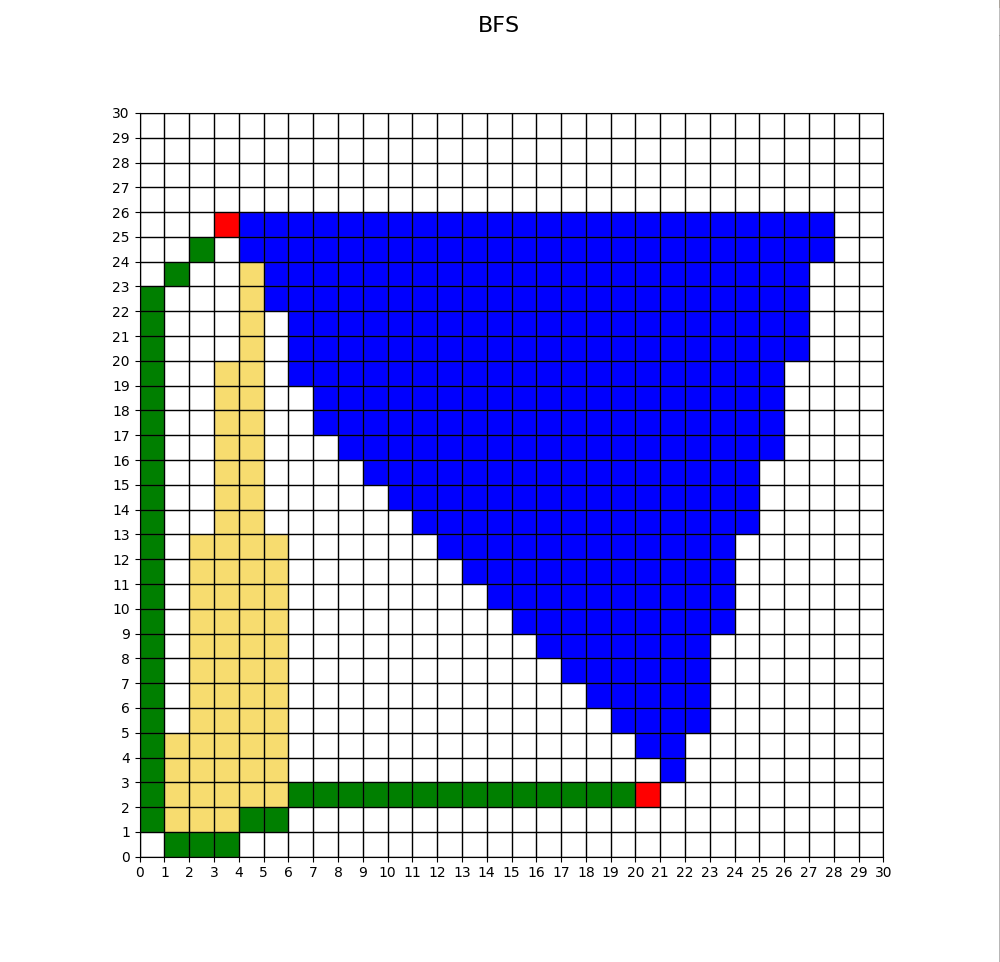
+ parent: là mảng chứa các trạng thái trước của điểm.

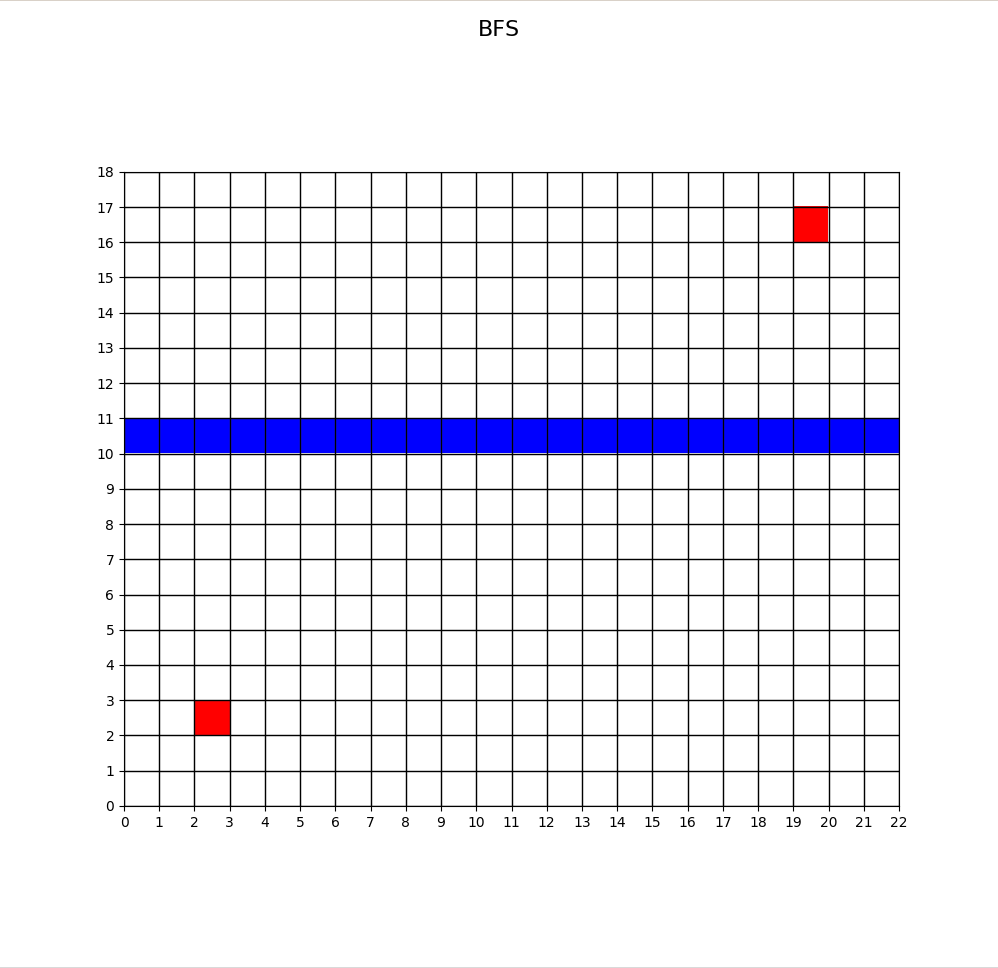
Ta sẽ gọi giá trị parent của điểm hiện tại cho backtrack cho và đẩy backtrack vào mảng path (mảng đường đi). Cho tới khi nào đạt được điểm start thì dừng và trả về mảng path. Đây là đường đi ngắn nhất khi sử dụng thuật toán BFS.

* + - * + **Thử nghiệm**

Bản đồ 1:



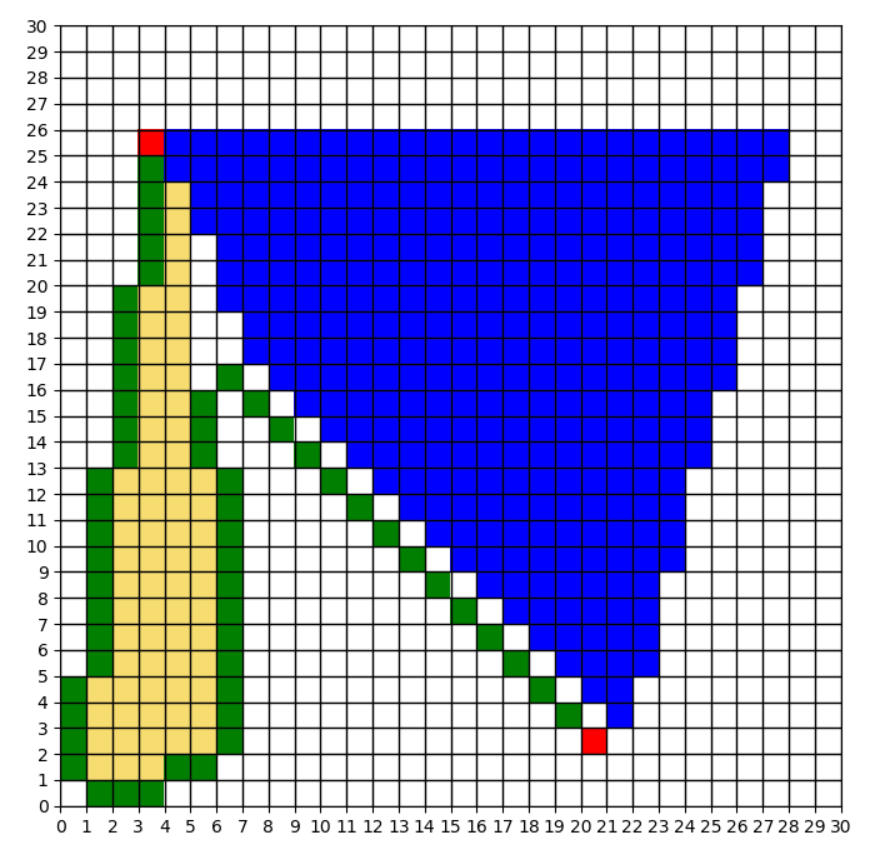
Bản đồ 2:

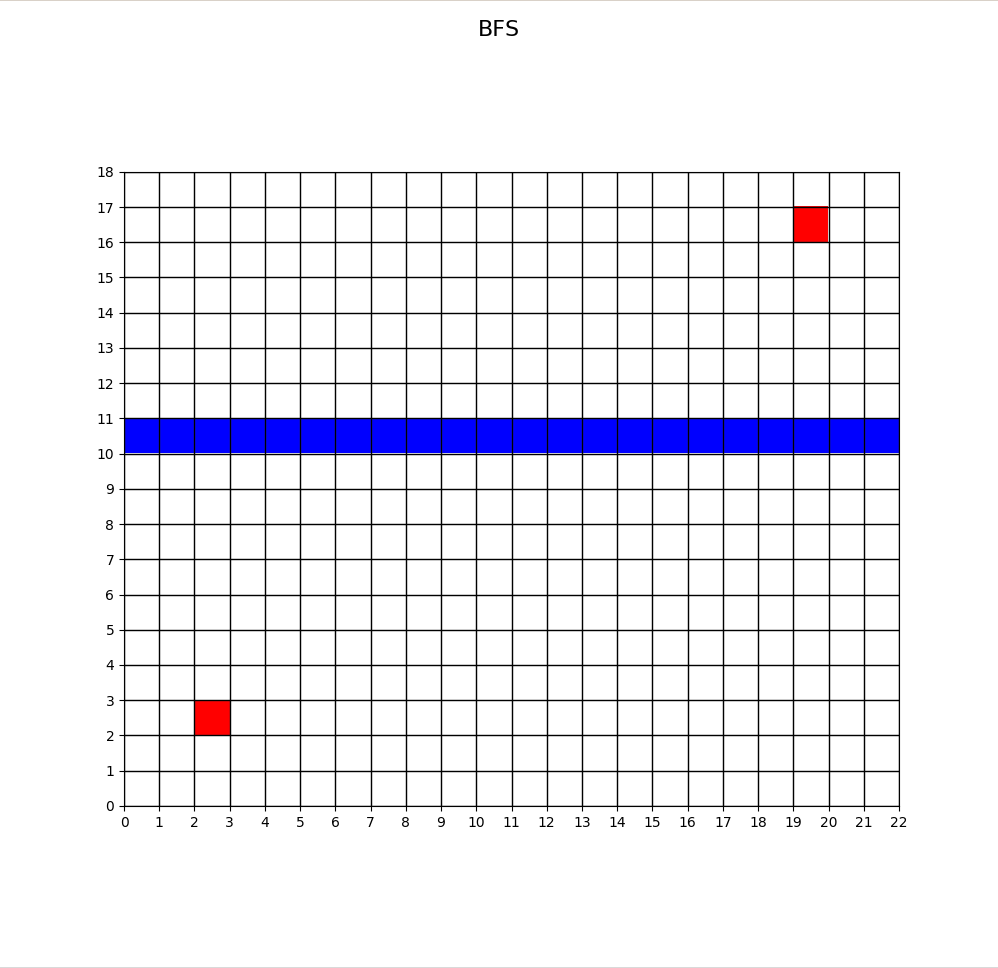
Bản đồ 3 (trường hợp không có đường đi) :

* + 1. **Tìm kiếm có thông tin ( informed search)**
       - **Tìm kiếm tham lam ( Greedy Best First Search )**
         * **Thuật toán**
* Chọn hàm Heuristic h((x,y)) là khoảng cách đường chim bay từ điểm (x,y) đến điểm đích.
* Cấu trúc dữ liệu sử dụng: hàng đợi ưu tiên.
* (\*) Với mỗi điểm đang được kích hoạt, ta mở ra các điểm có thể đi được ( các điểm không nằm ngoài khung, các điểm không bị vướng các đa giác).
* Ta tính chi phí f((x,y)) = h((x,y)), rồi đưa vào hàng đợi điểm được xét với độ ưu tiên là h((x,y)) càng nhỏ thì càng được ưu tiên.
* Ta chọn lấy điểm trên cùng của hàng đợi ưu tiên và lập lại quá trình từ (\*) cho đến khi đến điểm đích.
* Nếu thực hiện đi qua hết các ô mà không tới được đích thì không tồn tại được đi.
  + - * + **Thử nghiệm**

**A close up of a cage

Description automatically generated**Bản đồ 1:

Bản đồ 2:

Bản đồ 3 (trường hợp không có đường đi) :

* + - * **Tìm kiếm A\* ( A\* search )**
        + **Thuật toán**
    1. Đối tượng Successor:
* Thuộc tính Coordinate: Tọa độ của một trạng thái
* Các thuộc tính f, g, h: trong đó f = g+h. Và g là khoảng cách từ tọa độ đang xét đến điểm bắt đầu (đi ngang dọc là 1, đi chéo là 1.5), h là khoảng cách ước tính từ tọa độ hiện tại đến điểm đích (chọn hàm Heuristic là h= (x-xG)2+ (y-yG)2)
* Thuộc tính parrent: successor cha của nút hiện tại.
  + 1. Chi tiết thuật toán:
* Khởi tạo danh sách để chứa các trạng thái sinh ra mà chưa được xét (openList)
* Khởi tạo danh sách để chứa các trạng thái sinh ra đã được xét (closeList)
* Tạo Successor của tọa độ bắt đầu và kết thúc (startSuccessor, goalSuccessor)
* Thêm startSuccessor vào openList
* while openList != empty:
* Lấy trạng thái tốt nhất (q) trong openList ra, thêm vào closeList
* Nếu q == goalSuccessor thì tạo đường đi (path), chi phí (cost)

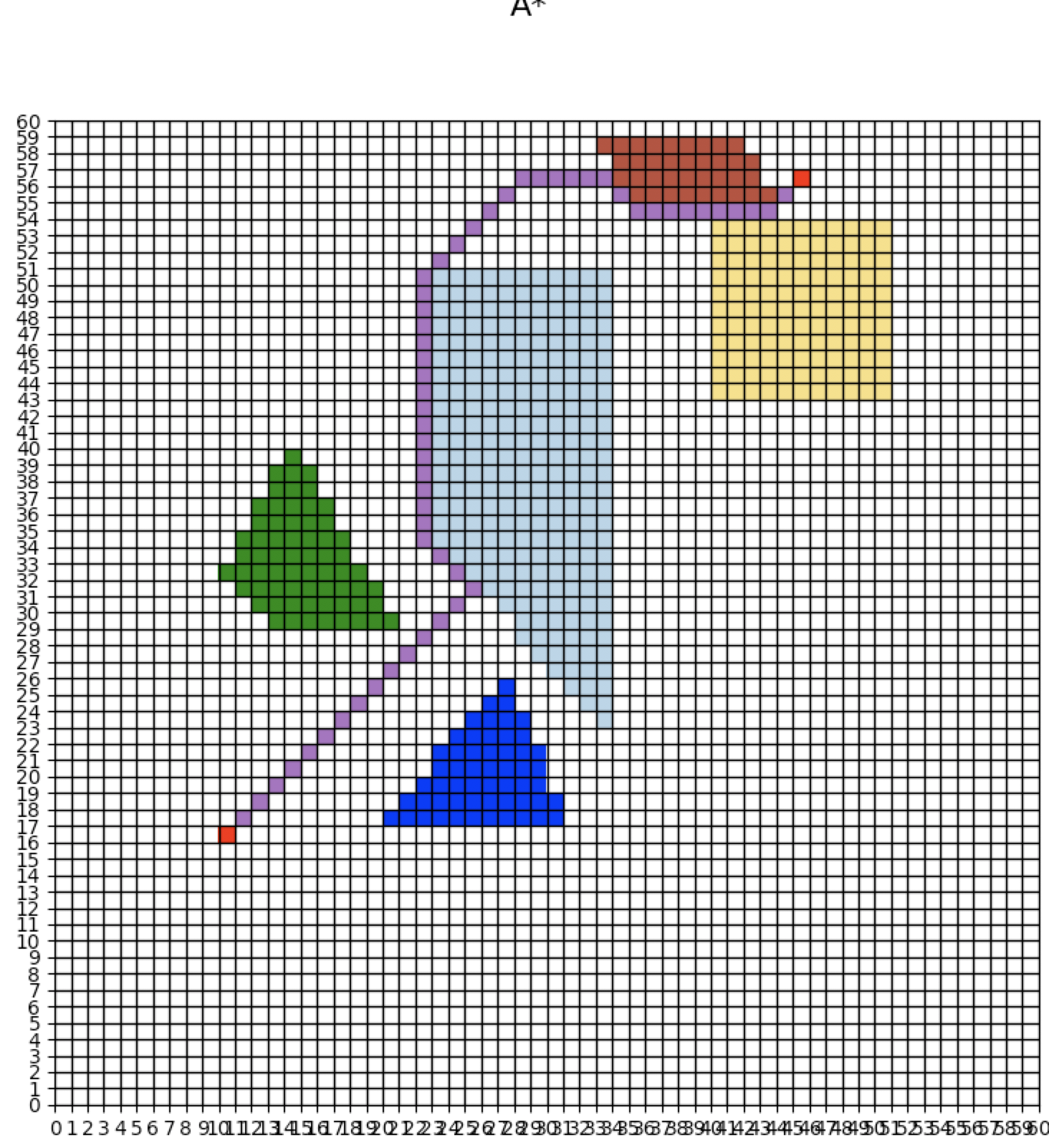
return path, cost

* Tạo các trạng thái con có thể có của q là p (p.parrent = q)
  + Nếu trạng thái p có cùng tọa độ với một trạng thái (x) trong closeList và p.g > x.g thì bỏ qua
  + Khởi tạo giá trị g, h, f cho trạng thái q
  + Nếu trạng thái p có cùng tọa độ với một trạng thái (y) trong openList và p.g > x.g thì bỏ qua
  + Thêm p vào openList
* Tìm kiếm thất bại
  + 1. Cách tạo đường đi:

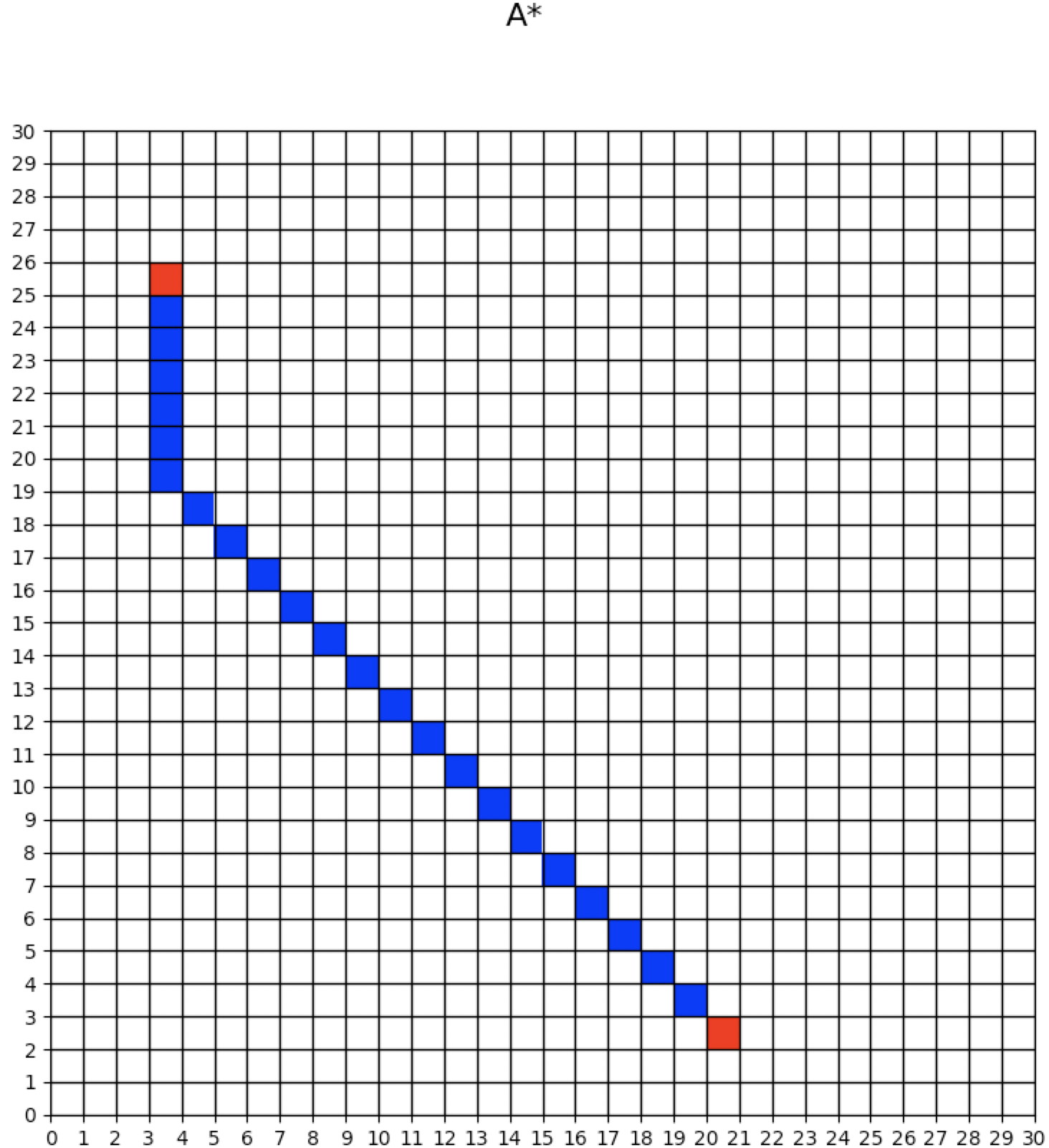
Backtracking từ goalSuccessor (gọi current = goalSuccessor)

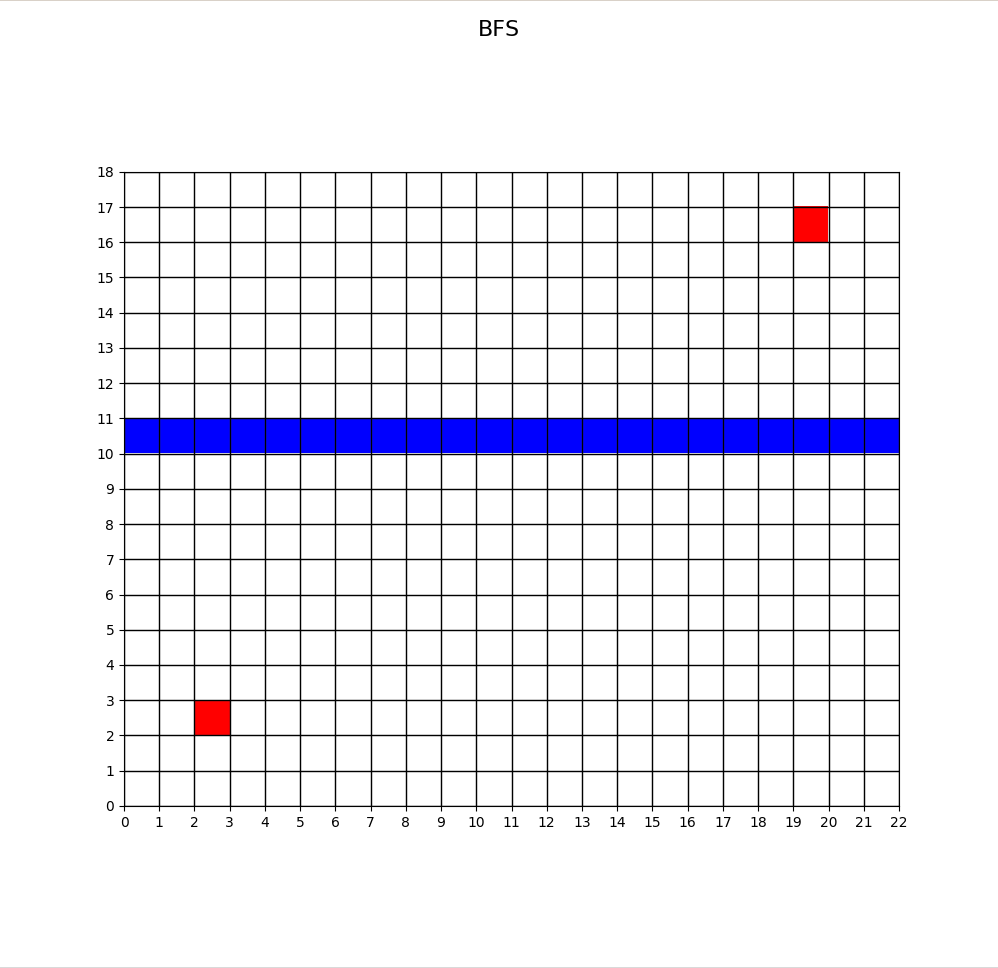
* while current.parrent != None:
  + Thêm current.coord vào path
  + current = current.parrent
    - * + **Thử nghiệm**

Bản đồ 1:

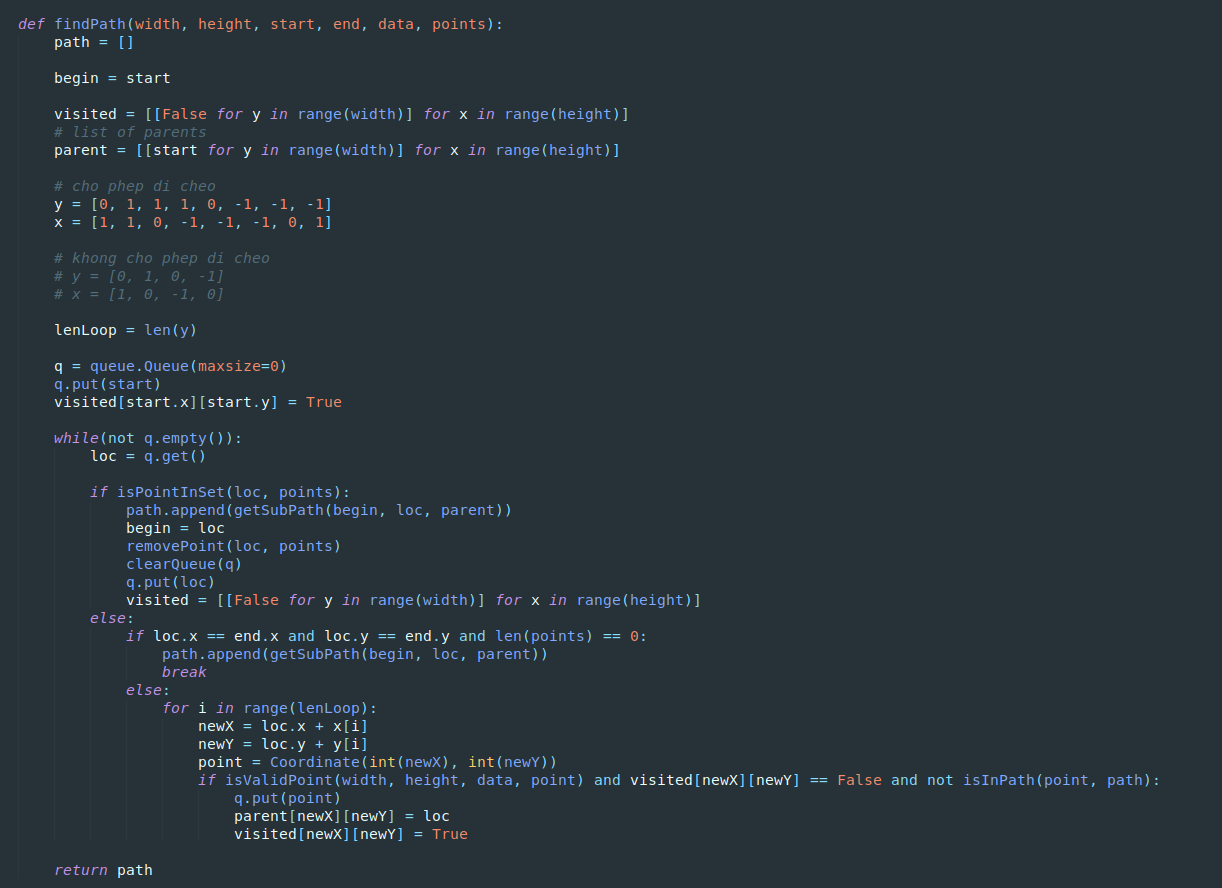
****

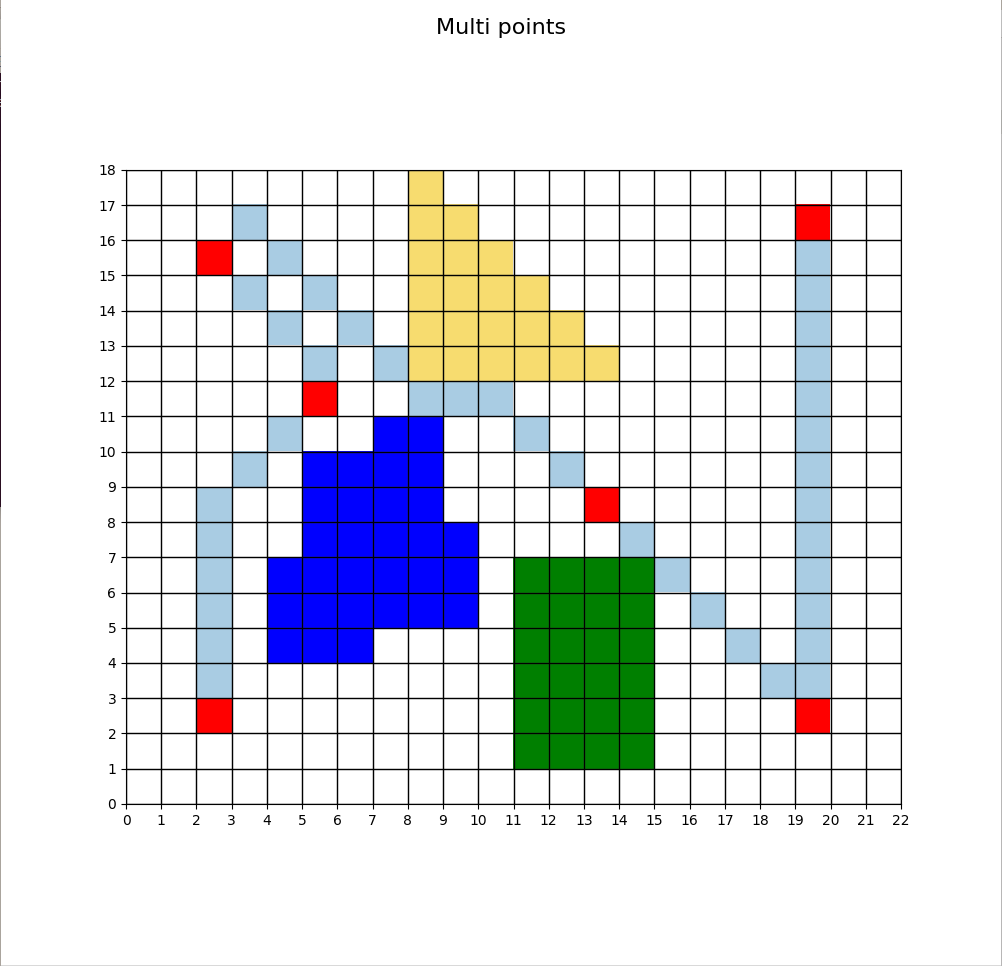
Bản đồ 2:

****

Bản đồ 3 (trường hợp không có đường đi) :

## Nội dung thực hiện mức 3 và mức 4

* + 1. **Tìm đường đi đển đích qua các điểm đón**
       - **Thuật toán**
         * Thuật toán sử dụng là một thuật giải Heuristic ứng dụng nguyên lý Greedy.
         * Mô tả: Từ điểm bắt đầu ta sẽ đi đến điểm gần nó nhất, và mới mọi điểm còn lại ta cũng làm vậy cho tới khi đạt được điểm đích.
         * Tham số truyền vào hàm lần lượt là chiều rộng bản đồ (width), chiều cao bản đồ (height), điểm bắt đầu (start), điểm kết thúc (end), danh sách các điểm cần đi qua (points).
         * Thuật toán này tương tự như BFS nhưng thay thì tìm đường đi đến điểm đích thì ta sẽ tìm đường đến điểm gần nó nhất.
         * Ta khởi tạo queue rỗng, đưa điểm start vào hàng đợi (queue) và đánh dấu điểm start đã được thăm.
         * Ta lấy dữ liệu từ hàng đợi (1 điểm), sau đó xét tất cả các điểm mà điểm có có thể tới và ta đẩy điểm mới vào trong hàng đợi.
         * Nếu điểm lấy ra từ hàng đợi là một điểm trong các điểm đến. Ta sẽ lưu đường đi từ điểm đến đã xét trước đó đến điểm đến hiện tại với hàm getSubPath(). Đồng thời ta xóa dữ liệu trong queue để đưa queue vể trạng thái rỗng và xóa điểm hiện tại khỏi mảng các điểm cần đến.
         * Nếu dữ liệu lấy trong queue ra là điểm kết thúc thì ta kết thúc thuật toán và trả về mảng path[] là mảng các đường đi.
       - **Thử nghiệm**



* + 1. **Tìm đường đi đến đích với các polygon có chuyển động**
       - **Thuật toán**
         * Trong phần này, ta sử dụng thuật toán GBFS ( Greedy Best First Search ) để tìm đường đi.
         * Ở mỗi một thời điểm, ta chọn ngẫu nhiên hướng cho từng mỗi một đa giác. Sau đó, đồng thời dịch chuyển các ô thuộc polygon đó theo hướng đã chọn một ô (Chọn cố định tốc độ của polygon là 1 ô / bước ).
         * Ta phát sinh mỗi các bước có thể di chuyển từ nút đang đứng hiện tại.
         * Chọn bước đi có chi phí ước lượng nhỏ nhất.
         * Lặp lại quá trình trên đến khi đến đích.
       - **Thử nghiệm**
         * File video được chứa trong thư mục Video ( PolygonAnimation.FLV )
       - **Nhận xét**
         * Trong một số trường hợp, robot bị mắc kẹt ở điểm cực trị địa phương ( tức các điểm sinh ra xung quanh có chi phí cao hơn hẳn điểm đang đứng, dù cho có đường đi ở hướng khác )

# Các tài liệu đã tham khảo

* Tài liệu đồ án bộ môn Cơ sở trí tuệ nhân tạo trên moodle
* Trang web tham khảo thuật toán geeksforgeeks: <https://www.geeksforgeeks.org>
* Thuật toán vẽ đường thẳng trên tọa độ: <https://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm?fbclid=IwAR0TiBG42hKE7Xi3IfoWshVTgltjWdE8utajs4fWc_sGcrZWrj7nO09uQtY>

# Đánh giá công việc

|  |  |  |
| --- | --- | --- |
| **Tên chức năng** | **Mức độ hoàn thành** | **Ghi chú** |
| 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 ( ví dụ tìm kiếm mù, tham lam, heuristic, ...) | 100% |  |
| Mức 3: trên bản đồ sẽ xuất hiện thêm một số điểm khác gọi là điểm đón. Xuất phát từ S, sau đó đi đón tất cả các điểm này rồi đến trạng thái G. Tìm ra cách để tổng đường đi là nhỏ nhất. | 100% |  |
| Mức 4: các 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. | 100% |  |
| Mức 5: thế hiện mô hình trên không gian 3 chiều (3D) | 0% | Mức điểm cộng 10% |