TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

MÔN: CƠ SỞ TRÍ TUỆ NHÂN TẠO

|  |
| --- |
| BÁO CÁO ĐỒ ÁN 1  ROBOT TÌM ĐƯỜNG |

MỤC LỤC

[*1.Thông tin nhóm 2*](#_Toc22994663)

[*2.Đánh giá mức độ hoàn thành 2*](#_Toc22994664)

[*3.Cách tổ chức và thực hiện đồ án 2*](#_Toc22994665)

[*4.Các mức độ hoàn thành 4*](#_Toc22994666)

[***a. Mức 1: Cài đặt thuật toán thuật toán Greedy Search để tìm đường đi 4***](#_Toc22994667)

[***b. Mức 2: Cài đặt thêm 2 thuật toán khác 7***](#_Toc22994668)

[\*Thuật toán Dijkstra  Search: 7](#_Toc22994669)

[\*Thuật toán A\*: 10](#_Toc22994670)

[\*So sánh và nhận xét 3 thuật toán: 12](#_Toc22994671)

[***c. Mức 3: Đi qua các điểm đón: 14***](#_Toc22994672)

[***d. Mức 4: Các đa giác có thể di động: 20***](#_Toc22994673)

[*5.Tài liệu tham khảo 21*](#_Toc22994674)

# **1.Thông tin nhóm**

|  |  |  |
| --- | --- | --- |
| Thành viên | | Công việc |
| 1712442 | Trần Trung Hiếu | Cài đặt thuật toán A\*,Dijkstra,Greedy, code mức 3 và mức 4, thuật toán tạo đa giác |
| 1712447 | Lê Long Hồ | Thể hiện giao diện đồ họa, viết báo cáo, cài đặt thuật toán Greedy |
| 1712459 | Nguyễn Việt Hoàng | Thuật toán A\*, Greedy |

# **2.Đánh giá mức độ hoàn thành**

|  |  |
| --- | --- |
| 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, …) | 90% |
| Mức 3: Đi qua các điểm đón | 95% |
| Mức 4: các hình đa giác có thể di động được với tốc độ h tọa độ/s | 90% |
| Mức 5: thể hiện mô hình trên không gian 3 chiều | 0% |

# **3.Cách tổ chức và thực hiện đồ á****n**

Mã nguồn được viết bằng ngôn ngữ lập trình Python, tổ chức thành 3 class Robot, World, Polygan.

Các hàm chức năng chính:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Class | Tên thuộc tính | Mô tả thuộc tính | Tên phương thức | Chức năng của phương thức | Tham số truyền vào |
| Robot | start: (int,int) | Tọa độ của điểm bắt đầu | get\_start\_point():start | Lấy tọa độ điểm bắt đầu |  |
| end: (int,int) | Tọa độ điểm kết thúc | get\_end\_point():end | Lấy tọa độ điểm kết thúc |  |
|  |  | Phương thức khởi tạo **\_\_init\_\_(int,int,int,int):**  **robot** | Khởi tạo đối tượng Robot | 4 số nguyên lần lượt là tọa độ x,y của start và end |
| World | Leng:int | Chiều ngang của bản đồ | Phương thức khởi tạo **\_\_init\_\_():world** | Khởi tạo đối tượng world |  |
| Width:int | Chiều dọc của bản đồ | getWidth():width | Trả về độ lớn chiều dọc của bản đồ |  |
| amount\_polygan:int | Số lượng đa giác | getLeng():leng | Trả về độ lớn chiều ngang của bản đồ |  |
| Polygans:list | Danh sách tọa độ các điểm tạo thành đa giác | read\_input(string):void | Đọc dữ liệu đầu vào từ file | Chuỗi string là tên của file input |
| Area:list | Danh sách | eucliean\_distance(int,int,int,int):float | Tính khoảng cách giữa 2 điểm trên bản đồ (hàm heuristic) | 4 số nguyên x1,y1,x2,y2 lần lượt là tọa độ của 2 điểm |
| Robot: Robot | Thông tin của Robot(chứa tọa độ điểm start và end) | drawing\_polygan(list):  void | Từ tọa độ các đỉnh đa giác theo chiều kim đồng hồ, tiến hành vẽ đa giác, trả về tọa độ tất cả các điểm tạo thành đa giác | List tọa độ các điểm tạo thành đa giác theo chiều kim đồng hồ |
| amount\_stop:int | Số lượng điểm dừng | find\_permutation(int,list):void | Sinh ra hoán vị tốt nhất của các điểm dừng(thứ tự các điểm đón lần lượt đi đến) dựa trên hàm heuristic | Chiều dài mảng con để xét hoán vị và list chứa một hoán vị của tập hợp điểm dừng |
| Stops:list | Danh sách tọa độ các điểm dừng |  |  |  |
|  |  | greedy\_search(  GraphWin):float | Thực hiện thuật toán greedy search tìm kiếm đường đi từ điểm S đến điểm G (đi qua các điểm dừng nếu có) và thể hiện đường đi trên cửa sổ đồ họa, trả về chi phí đường đi, nếu không có đường đi return -1 | Cửa sổ đồ họa GrapWin nơi thuật toán hiển thị output đường đi |
|  |  | dijkstra\_search(GrapWin): float | Triển khai thuật toán Dijkstra search tìm kiếm đường đi ngắn nhất từ điểm S đến điểm G , vẽ đường đi trên cửa sổ đồ họa và trả về chi phí đường đi, nếu không có đường đi return -1 | Cửa sổ đồ họa GrapWin nơi thuật toán hiển thị output đường đi |
|  |  | astar\_search(GrapWin): float | Thực hiện thuật toán A\* search, tìm đường đi ngắn nhất từ điểm S đến điểm G và thể hiện đường đi đó bằng cửa sổ đồ họa, trả về chi phí đường đi, nếu không có đường đi return -1 | Cửa sổ đồ họa GrapWin nơi thuật toán hiển thị output đường đi |
|  |  | print\_area(GrapWin):void | In bản đồ gồm điểm bắt đầu và kết thúc, các điểm dừng ra màn hình đồ họa GrapWin | Cửa sổ đồ họa cần thể hiện ra |
|  |  | moving\_polygan(polygan : list, step : tuple): bool,list | Thực hiện di chuyển tọa độ các đa giác, nếu không di chuyển được thì trả về false và list rỗng, nếu di chuyển thành công trả về true và tọa độ polygan | Danh sách các đỉnh nối thành polygan |
|  |  | drawing\_dynamic\_polygan(polygans): list | Vẽ đa giác đã di chuyển ra màn hình đồ họa | Tọa độ polygan cần vẽ |
|  |  | greedy\_search\_with\_dynamic(polygan\_borders, color\_robot,win) | Triển khai thuật toán greedy search tìm đường đi đến điểm đích tránh các đa giác di chuyển | Polygan sau khi di chuyển, màu thể hiện đường đi và cửa sổ đồ họa cần vẽ |
| Polygan | Points:list | Danh sách chứa tọa độ của polygan |  |  |  |

Ngoài ra còn các hàm chính thể hiện đồ họa của biểu diễn đa giác và đường đi, có sử dụng thư viện đồ họa Graphics trong các hàm

1. drawGrid(width,height,win): vẽ bản đồ với chiều ngang width và chiều dọc height
2. drawText (x,y,height,win,text,size): điền văn bản text lên điểm có tọa độ (x,y) trên bản đồ
3. drawPath(list,color,win,height): vẽ đường đi của robot hoặc hình dạng của đa giác với danh sách tọa độ chứa trong list.

\*Cách thức chương trình hoạt động:

Đầu tiên chương trình sẽ đọc dữ liệu đầu vào từ file input bằng phương thức read\_input() của class world. Sau khi thực hiện phương thức này, chiều dài và chiều rộng của không gian được gán vào thuộc tính leng của class World. Tọa độ điểm bắt đầu, điểm kết thúc được truyền vào Robot(). Tọa độ các điểm dừng được thêm vào vào mảng stop[] trong World. Đồng thời tập hợp các tọa độ của đa giác cũng được thêm vào list polygan[] của lớp World. Tùy theo tham số do người dung nhập vào mà thực hiện thuật toán tương ứng và sử dụng các hàm vẽ để thể hiện output bằng đồ họa kèm với chi phí của đường đi.

# **4.Các mức độ hoàn thành**

## ***a. Mức 1: Cài đặt thuật toán thuật toán Greedy Search để tìm đường đi***

Ý tưởng thực hiện:

Tại mỗi bước đi duyệt qua 8 vị trí xung quanh của điểm hiện tại:

|  |  |  |
| --- | --- | --- |
| (x-1, y+1) | (x, y+1) | (x+1, y+1) |
| (x-1, y) | (x, y) | (x+1, y |
| (x-1, y-1) | (x, y-1) | (x+1, y-1) |

**Cho phép đi chéo:** Xét điều kiện nếu là bước đi chéo thì 1 trong 2 ô bên cạnh phải trống, nếu cả 2 ô bên là điểm thuộc đa giác thì không xét vì nếu xét sẽ bị đi vào trong đa giác

Đưa các điểm xung quanh vào hàng đợi nếu nó chưa duyệt qua. Tập mở và tập đóng (để xét điểm đó có duyệt trước đó hay chưa) được miêu tả bằng một không gian ma trận như không gian của bản đồ, giá trị 0 là chưa duyệt và 1 là đã duyệt rồi.Nếu tại đích =0 tức là không có đường đi đến

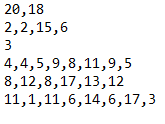
Nếu điểm nào thỏa điều kiện:

1. Không thuộc hay nằm trên đa giác
2. Không vượt giới hạn biên của bản đồ
3. Khoảng cách đến điểm đích (hàm heuristic: eucliean\_distance() ) nhỏ nhất

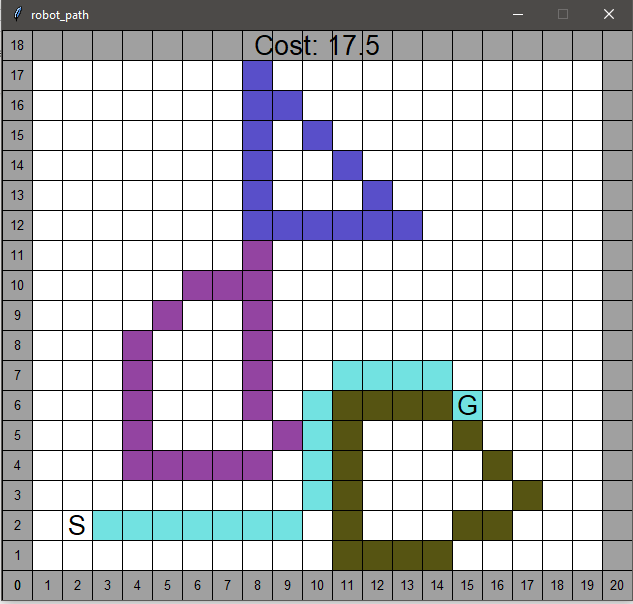
thì điểm đó là điểm tốt nhất và sẽ được đặt ở đầu hàng đợi để bước tiếp theo lấy nó ra và chi chuyển đến đó. Vòng lặp tiếp theo tiếp tục lấy điểm từ hàng đợi ra, tiếp tục xét 8 điểm xung quanh của điểm đó rồi đẩy vào hàng đợi, lặp lại cho đến khi gặp điểm đích là dừng, mỗi lần di chuyển thì cũng kèm theo tính chi phí đường đi.

Qúa trình chạy thử:

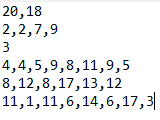
Với input:



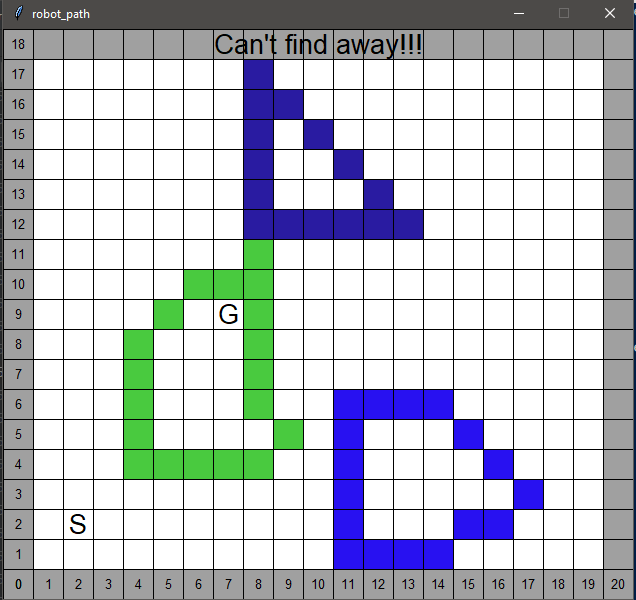
Output:



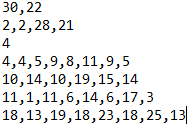
Input: (Trường hợp không tìm được đường đi)



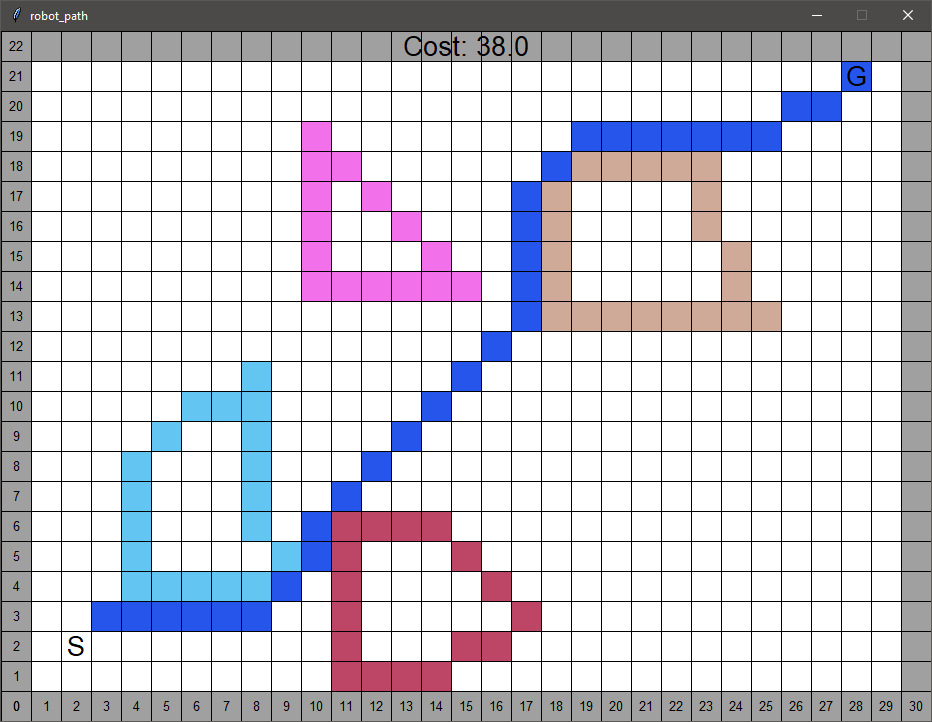
Output:



Input:



Output:



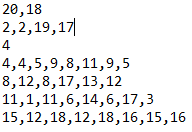
## ***b. Mức 2: Cài đặt thêm 2 thuật toán khác***

### *\*Thuật toán Dijkstra  Search:*

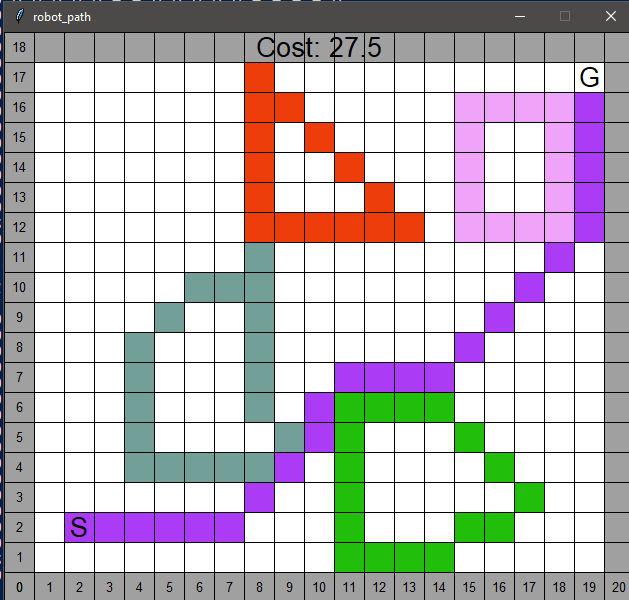
Ý tưởng thực hiện:

Tương tự như thuật toán Greedy Search, tại mỗi điểm đẩy 8 điểm xung quanh vào hàng đợi nếu nó chưa duyệt qua. Nhưng lúc này cập nhật lại khoảng cách tới 8 điểm xung quanh đó từ vị trí hiện tại nếu khoảng cách đó ngắn hơn và them vào hàng đợi nếu nó chưa được duyệt. Sau khi duyệt tới đích thì truy vét lại từ đích tới nơi bắt đầu để tìm đường đi ngắn nhất.

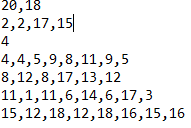
Input:



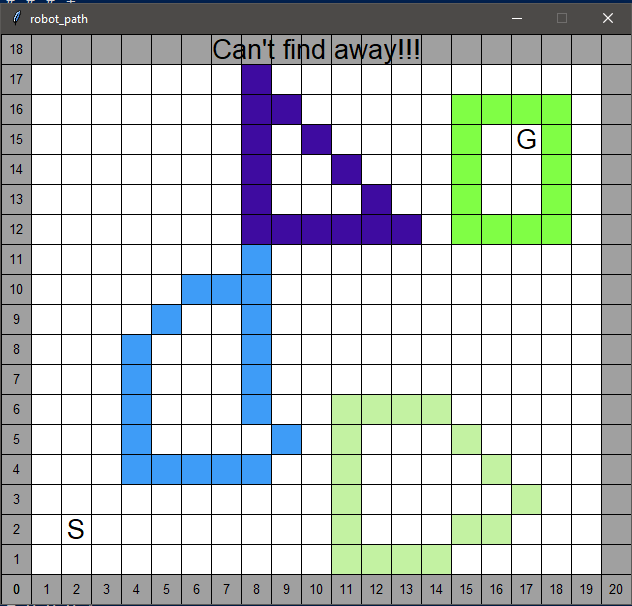
Output:



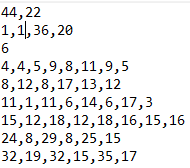
Input:



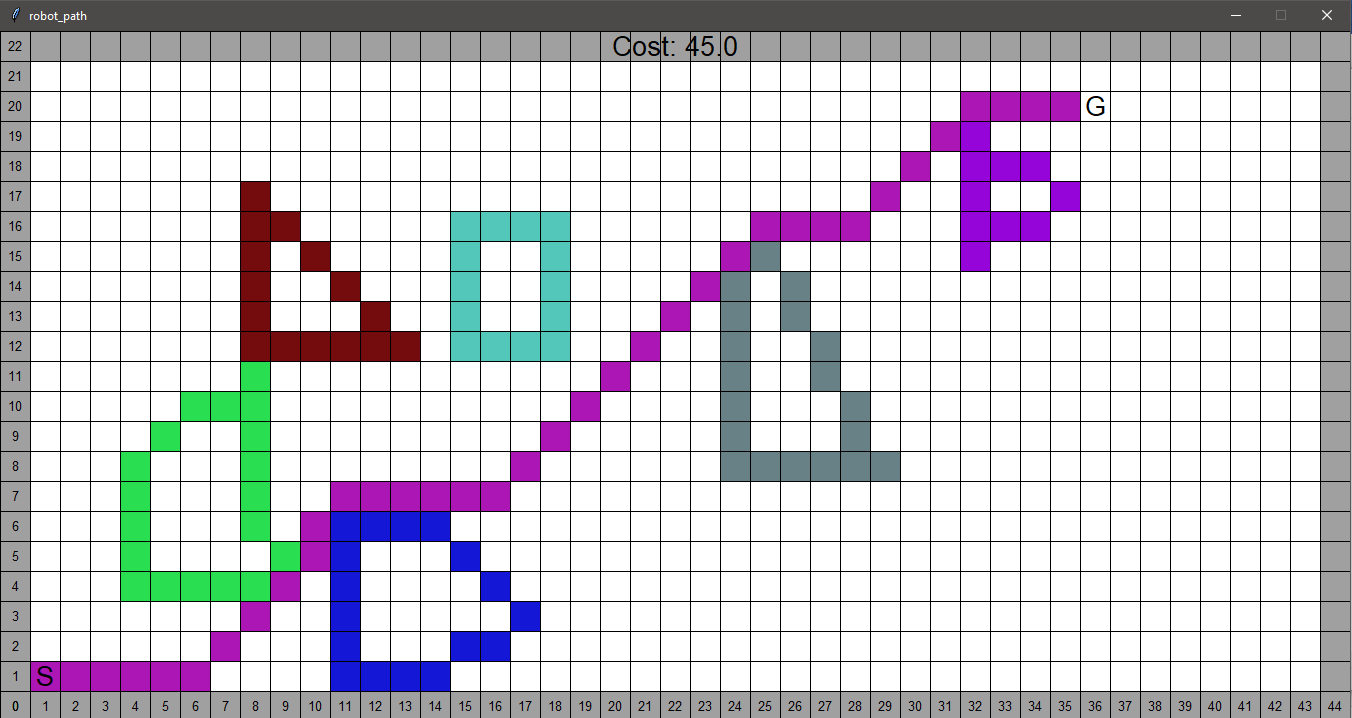
Output:



Input:



Output:

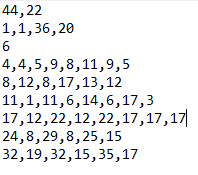


### *\*Thuật toán A\*:*

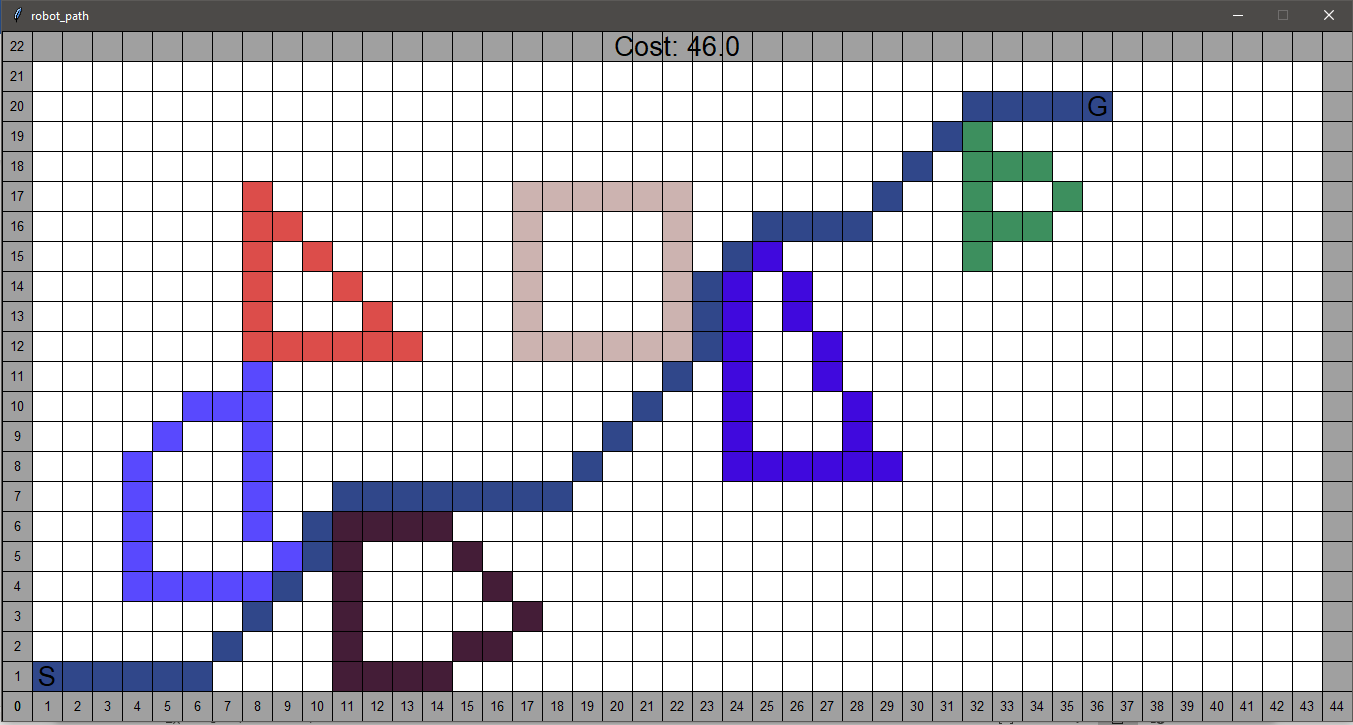
Ý tưởng thực hiện:

Tương tự greedy search nhưng điều kiện để xét điểm tốt nhất đẩy vô đầu hàng đợi là nếu tổng khoảng cách từ nút gốc (Dijkstra) với khoảng cách tới nút kết thúc (Heuristic) là nhỏ nhất.

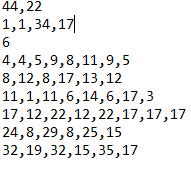
Input:



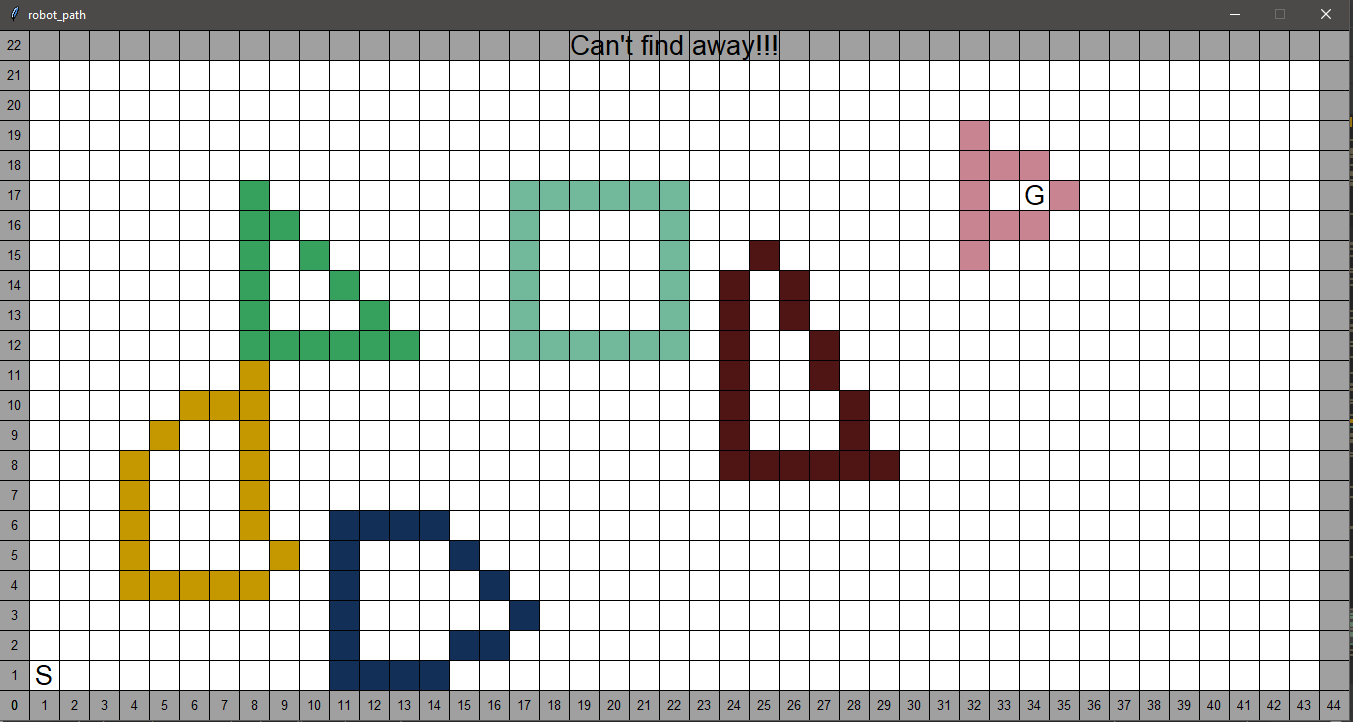
Output:



Input:

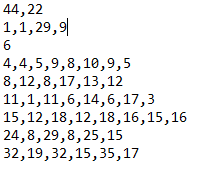


output:

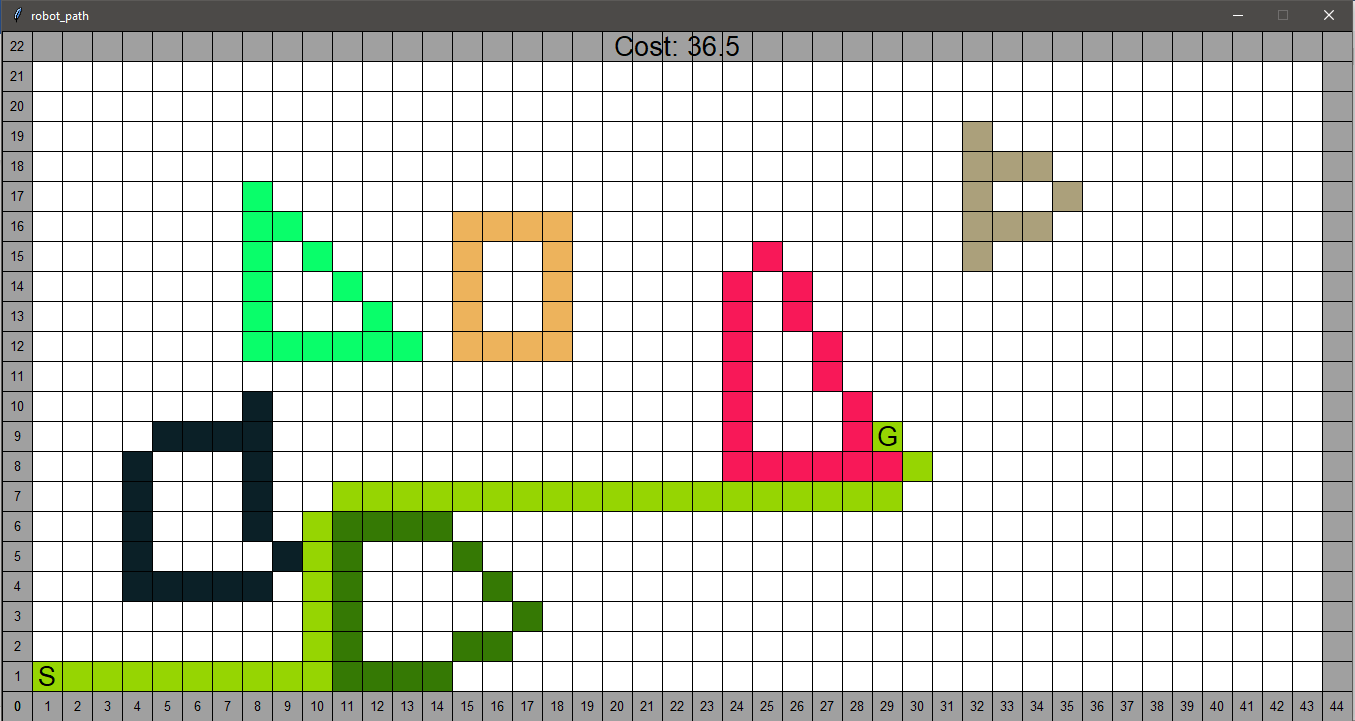


### **\*So sánh và nhận xét 3 thuật toán:**

Cùng một input:

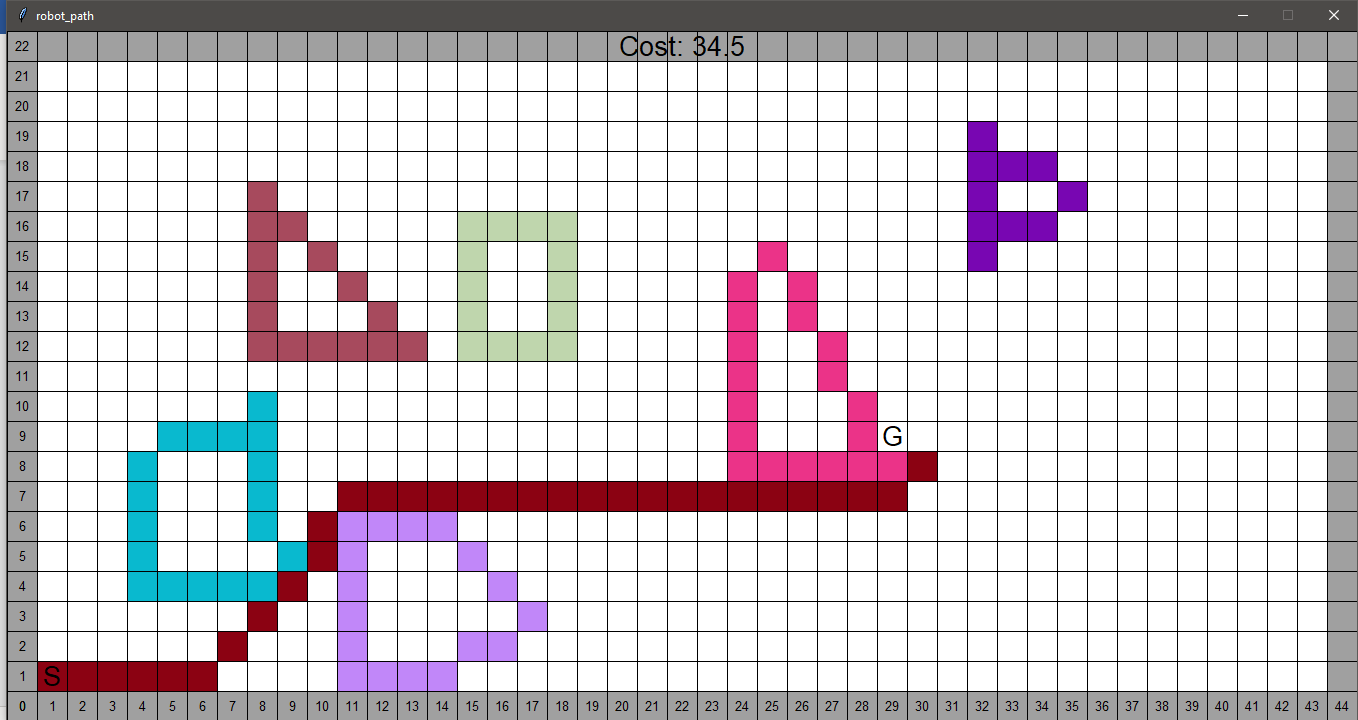


Output của Greedy Search:



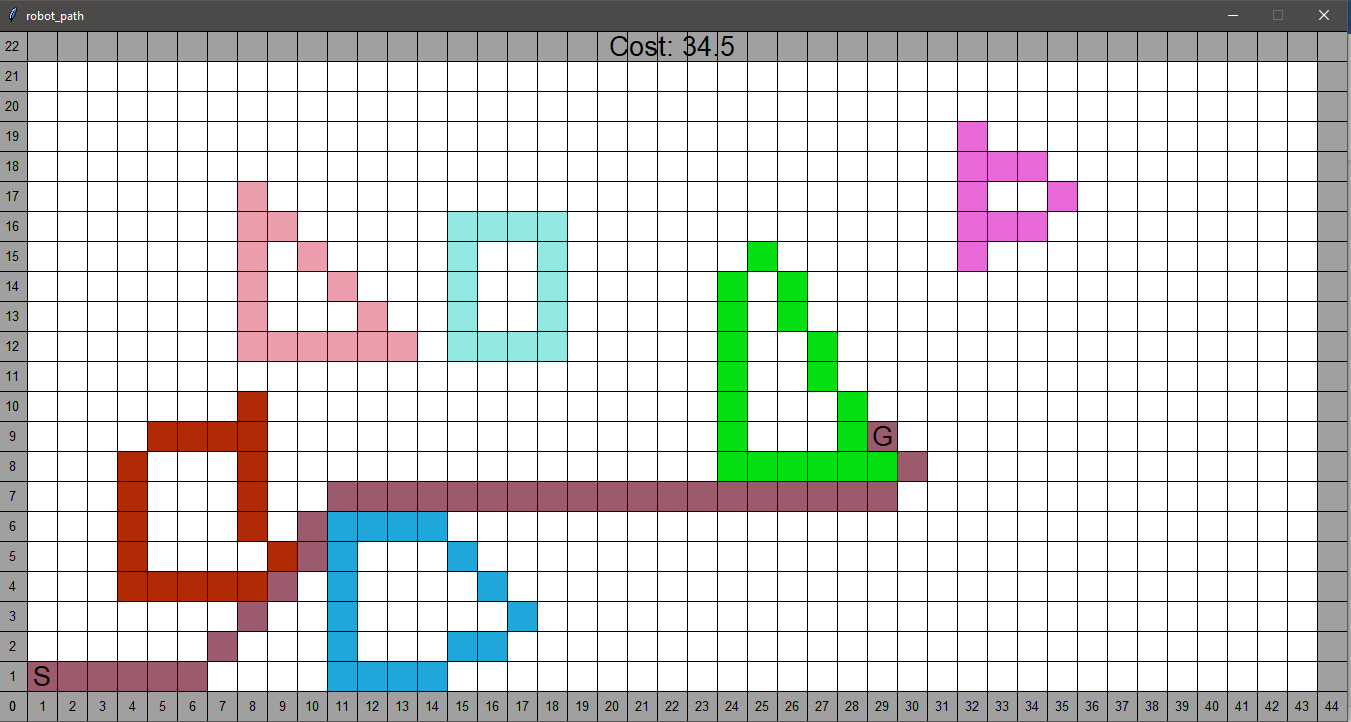
Nhận xét: -Greedy Search: thực hiện nhanh nhưng trường hợp bị chắn bởi đa giác thì sẽ tìm ra đường không tối ưu

Output của Dijkstra Search:



Nhận xét: dijkstra tìm ra đường đi tối ưu hơn, nhưng tốn thời gian hơn Greedy Search

Output của A\*:



Nhận xét: A\* tổng hợp ưu điểm của 2 thuật toán Dijkstra và Greedy Search, nhanh hơn thuật toán Dijkstra và tìm đường đi tối ưu hơn thuật toán Greedy Search nhưng vẫn không hoàn hảo

## ***c. Mức 3: Đi qua các điểm đón:***

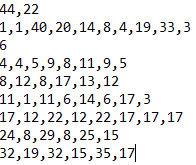
Thuật toán áp dụng:

-Với bài toán có điểm đón trên bản đồ thì trước tiên sinh hoán vị tốt nhất của các điểm đón( thứ tự các điểm đón sao cho đường đi đến lần lượt các điểm là ngắn nhất) bằng phương thức find\_permutation(), xem đích đến là điểm đón đầu tiên, bắt đầu tìm đường đi từ điểm S đến điểm đón đầu tiên bằng thuật toán greedy search hoặc A\*, rồi từ điểm đón đó đến điểm đón thứ hai (lúc này xem điểm đoán thứ nhất là vị trí bắt đầu và điểm đón thứ hai là vị trí kết thúc), lặp lại cho đến khi gặp điểm đón cuối cùng (cũng là đích đến cuối cùng).

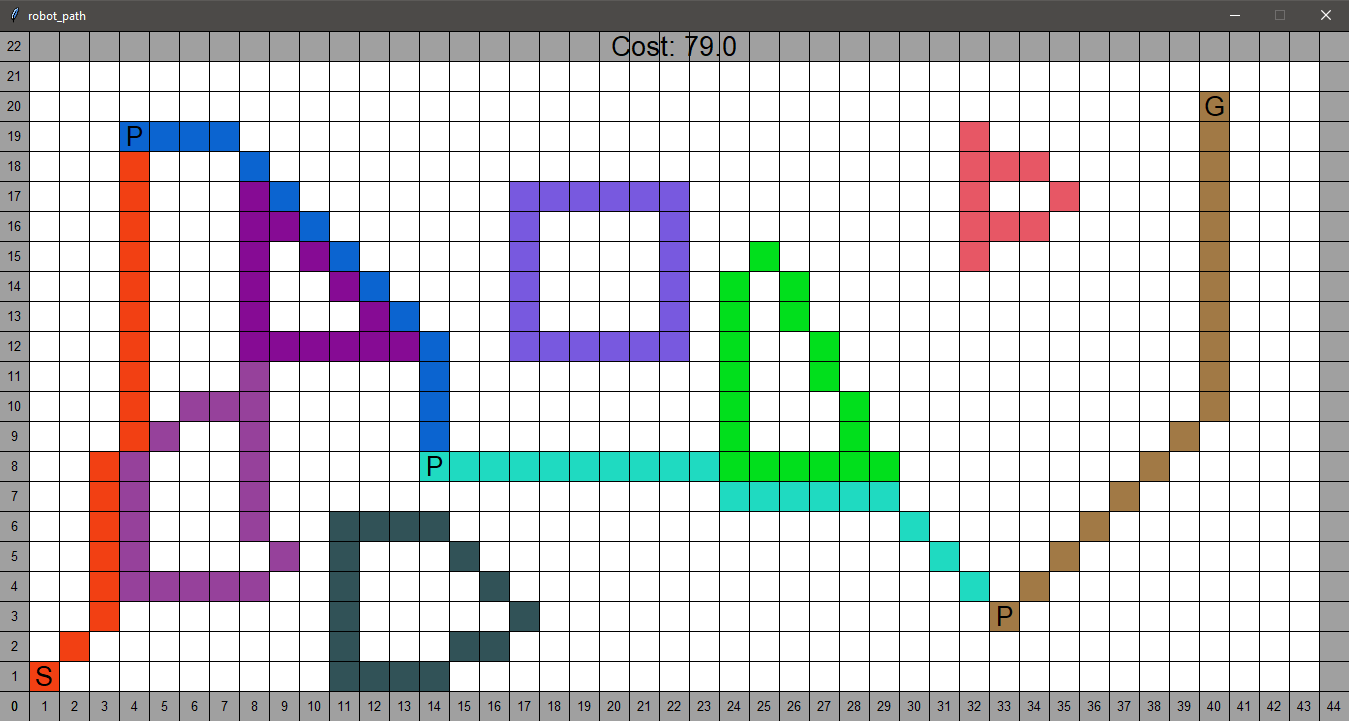
- Hàm find\_permutation() sẽ thực hiện chọn hoán vị tốt nhất dựa trên tổng khoảng cách heuristic (tổng khoảng cách từ điểm start đến các điểm đón rồi đến điểm goal), hoán vị nào tổng khoảng cách giữa các điểm đón ngắn nhất sẽ được chọn.

*\*Thuật toán A\**

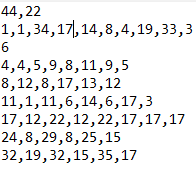
Input:



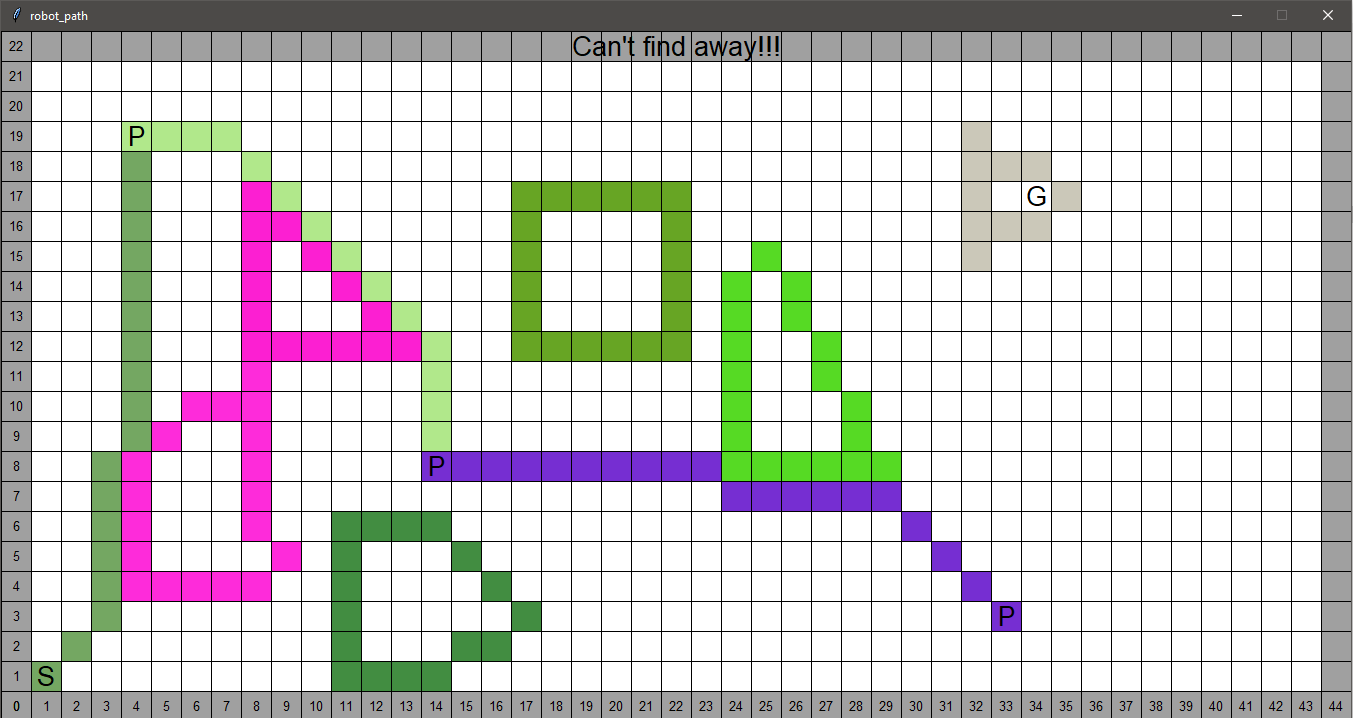
Output:



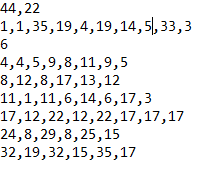
Input:



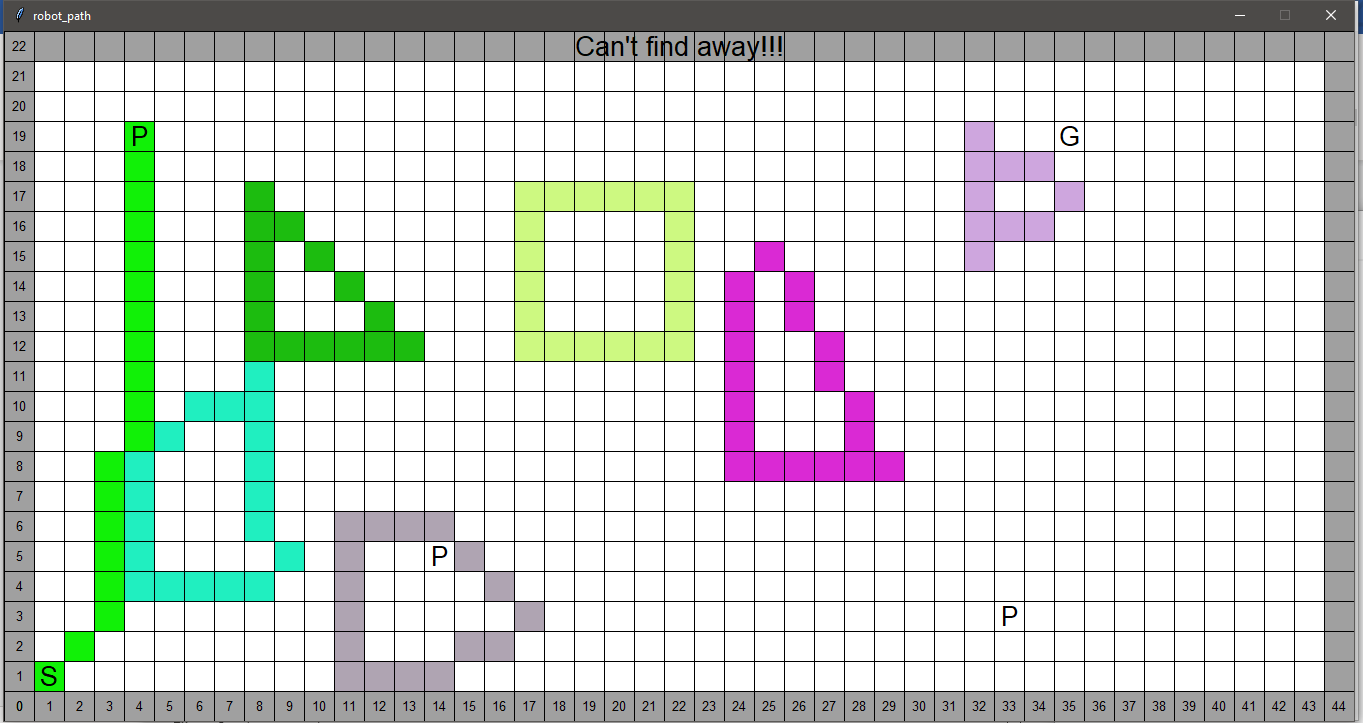
Output:



Input:

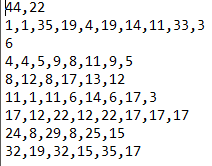


Output:

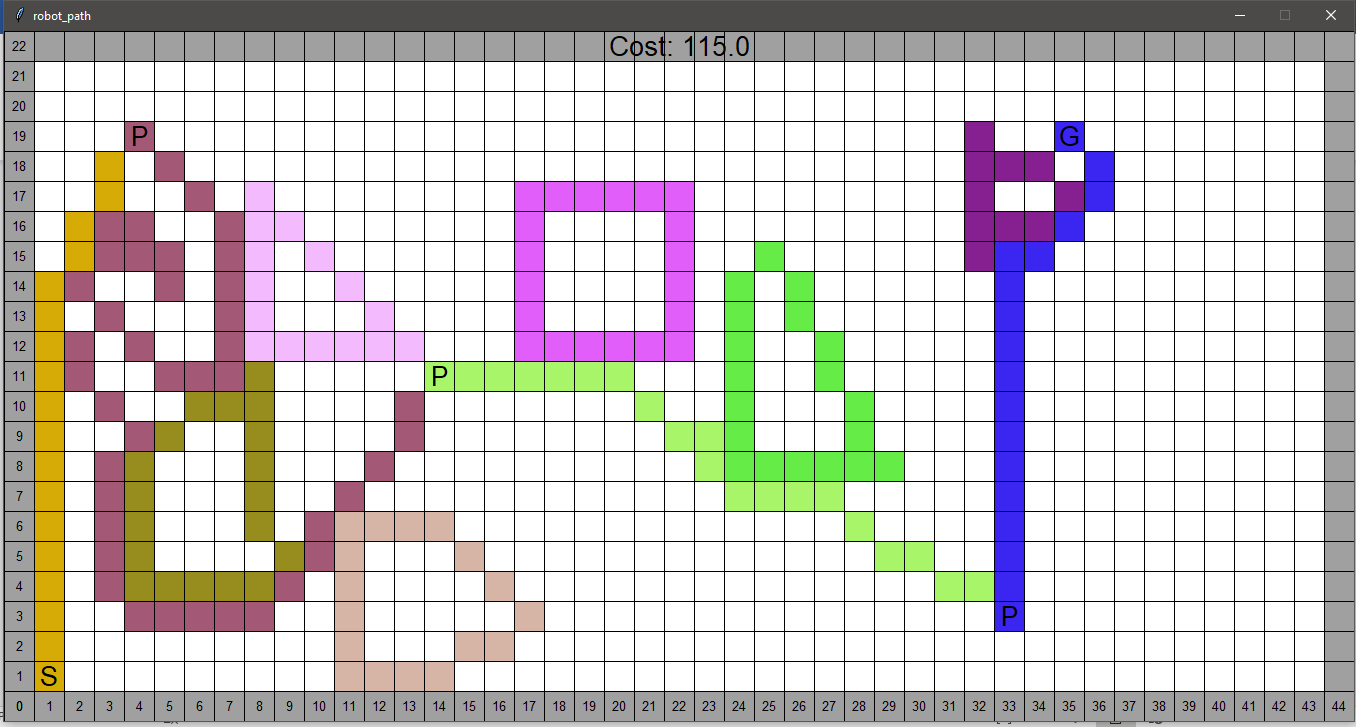


*Thuật toán greedy search:*

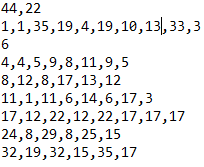
Input:



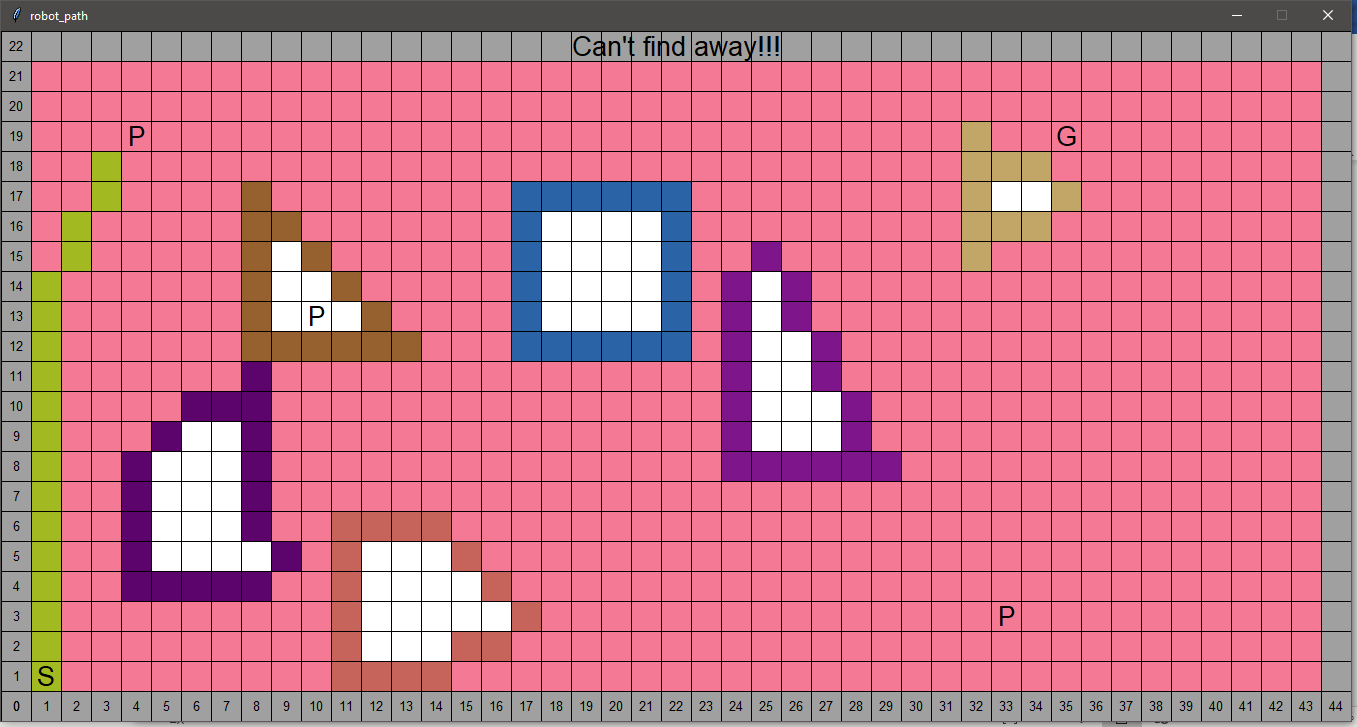
Output:



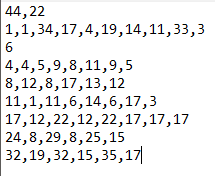
Input:



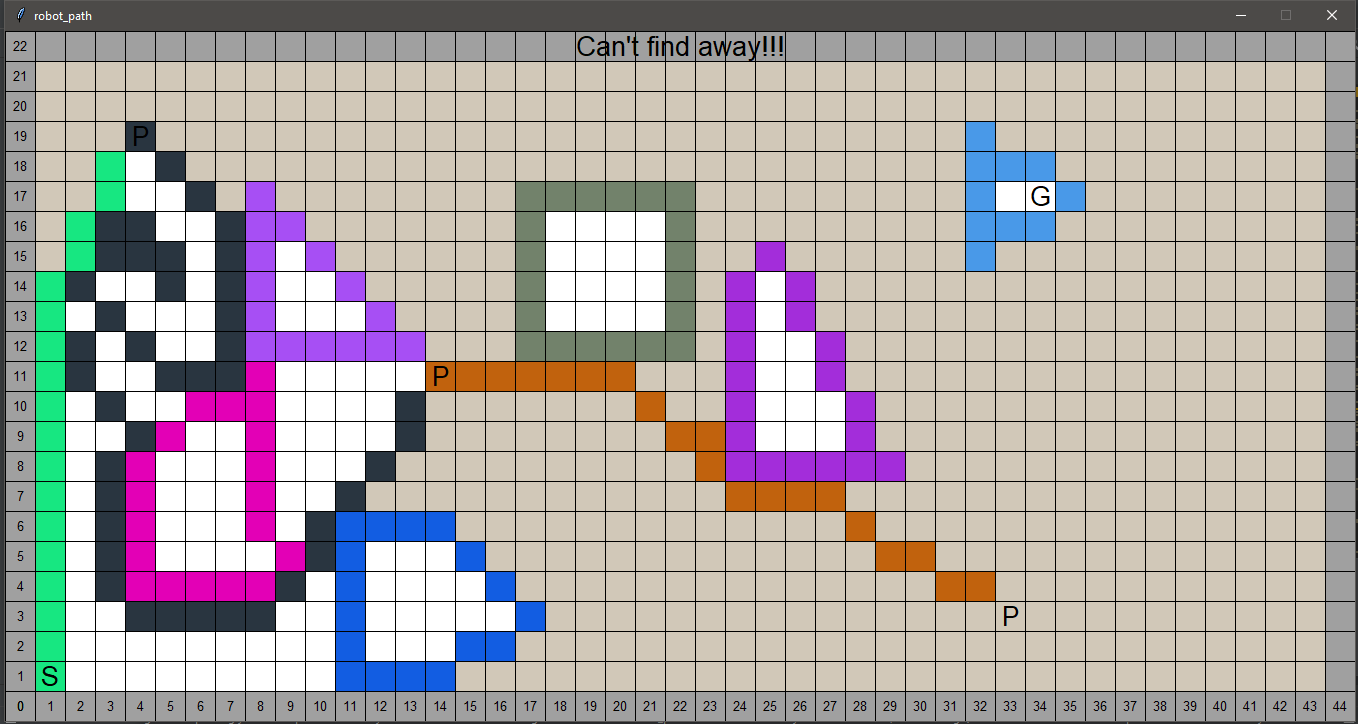
Output:



Input:



Output:



## ***d. Mức 4: Các đa giác có thể di động:***

Chi tiết thuật toán:

-Random các vị trí mà đa giác sẽ di chuyển

-Xét duyệt tọa độ vừa random qua hàm moving\_polygan(), nếu thỏa điều kiện:

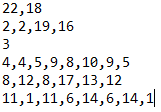
* 1. Các đa giác không đè lên nhau
  2. Đa giác không lấn lên hay vượt giới hạn của bản đồ

-Thực hiện vẽ các đa giác di chuyển bằng hàm drawing\_dynamic\_polygan()

-Mỗi lần đa giác di chuyển cập nhật lại bản đồ, sau đó dung thuật toán Heuristic xét đường đi cho robot ở bản đồ đó

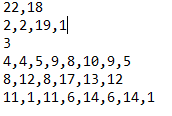
-Trường hợp đa giác đụng phải robot thì để xử lí robot không đi vô trong đa giác thì tô luôn cả diện tích của đa giác

Input:



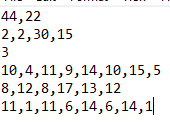
Video output: <https://www.youtube.com/watch?v=s4YqpJWJOvw&feature=youtu.be>

Input:



Video output: <https://www.youtube.com/watch?v=UDz2sokwWRA&feature=youtu.be>

Input:



Video output: <https://www.youtube.com/watch?v=dQQIkS89TtA&feature=youtu.be>

# **5.Tài liệu tham khảo**

Thư viện đồ họa *Graphics :* <http://anh.cs.luc.edu/handsonPythonTutorial/graphics.html>