KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN MÔN HỌC
CƠ SỞ TRÍ TUỆ NHÂN TẠO

THUẬT TOÁN TÌM KIẾM

Trợ giảng:

Nguyễn Khánh Toàn
ktoan271199@gmail.com
Lê Minh Nhật
minhnhatvt2@gmail.com

THÔNG TIN NHÓM

MSSV	Họ và tên	Email
1712809	Nguyễn Gia Thụy	1712809@student.hcmus.edu.vn
1712292	Lý Quốc Bình	1712292@student.hcmus.edu.vn

MỤC TIÊU ĐỒ ÁN

- ***Nắm được:***
  - Các thuật toán tìm kiếm không có thông tin.
    - Thuật toán tìm kiếm DFS (Depth First Search).
    - Thuật toán tìm kiếm BFS (Breadth First Search).
  - Các thuật toán tìm kiếm có thông tin
    - Thuật toán tìm kiếm tham lam (Greedy Best First Search).
    - Thuật toán tìm kiếm A*.
    - Các hàm heuristic.
- ***Ứng dụng để tìm kiếm trên:***
  - Bản đồ không có điểm thưởng
  - Bản đồ có điểm thưởng

QUY ƯỚC TRONG ĐỒ ÁN

- Tác nhân luôn xét hướng đi theo thứ tự chiều kim đồng hồ. Tức là tác nhân sẽ luôn xét hướng đi với thứ tự lần lượt là: hướng lên, qua phải, hướng xuống, qua trái.

- Các hàm thuật giải sử dụng trọng số D bằng 1.

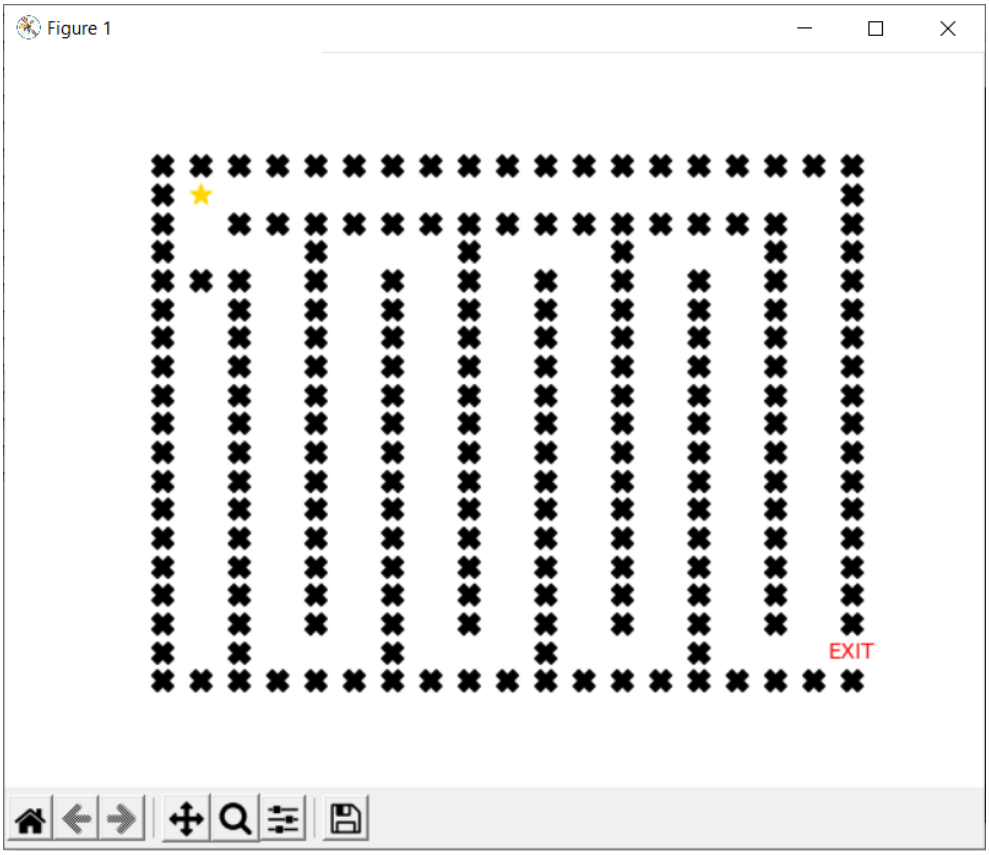
PHÂN TÍCH THUẬT TOÁN

I. Tổng quan các thuật toán

Depth First Search	Breadth First Search	Greedy Best First Search	A* Search
- Thuật toán luôn tìm kiếm đường đi sâu nhất có thể, dựa vào phương pháp LIFO. - Do thứ tự xét hướng là quan trọng, nên thuật toán DFS sẽ phát huy tốt khi đường đi được xét cuối cùng là ngắn nhất và ngược lại. - Việc tìm ra đường đi tối ưu là hoàn toàn ngẫu nhiên.	- Thuật toán luôn mở rộng không gian tìm kiếm theo chiều rộng (loang ra nhiều phía), dựa vào phương pháp FIFO. - Do thuật toán luôn mở rộng ra nhiều phía, cho nên thuật toán có chi phí thực thi thấp khi có ít hướng đi tại một thời điểm và ngược lại. - Luôn tìm ra đường đi ngắn nhất nếu các cạnh không có trọng số hoặc có trọng số như nhau.	- Thuật toán là phiên bản cải tiến của BFS với khả năng tối ưu đường đi theo một thuật giải nào đó. Thuật toán sẽ sử dụng thuật giải để sắp xếp lại các nốt trong hàng đợi sau mỗi lần mở rộng, từ đó làm thay đổi thứ tự duyệt các nốt. - Khả năng ước lượng của thuật toán phụ thuộc vào thuật giải được sử dụng. Tùy vào cấu trúc của không gian tìm kiếm mà mỗi thuật giải sẽ phát huy được thế mạnh riêng.	- Thuật toán là phiên bản cải tiến của Greedy Best First Search. Cụ thể là bên cạnh sử dụng thuật giải, thuật toán còn sử dụng thêm hàm để ước tính chi phí từ đầu đến nốt hiện tại. Từ đó sẽ đưa ra phương án chính xác hơn cho đường đi tối ưu. - Khả năng ước lượng của thuật toán phụ thuộc vào thuật giải được sử dụng. Tùy vào cấu trúc của không gian tìm kiếm mà mỗi thuật giải sẽ phát huy được thế mạnh riêng. Luôn tìm ra đường đi tối ưu.

II. Áp dụng tìm đường trong mê cung

1. Mê cung không có điểm thưởng



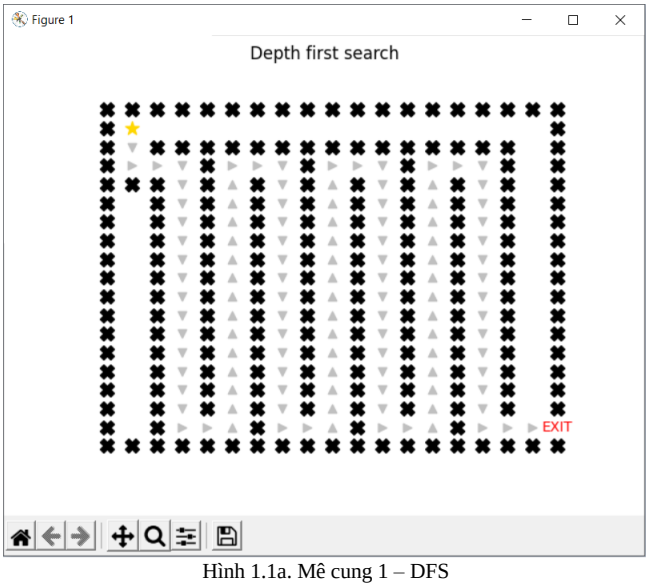
Hình 1.1. Mê cung 1

Mô tả mê cung: Từ điểm xuất phát ta có 2 hướng để đến lối thoát, với một hướng có đường đi không tối ưu, được tạo thành bởi nhiều lớp tường nên có quãng đường dài hơn đáng kể hướng còn lại.

Xét các thuật toán:

1.1a. Thuật toán DFS

- Nhận xét: Tác nhân lựa chọn hướng đi trong thuật toán này là hướng đi không tối ưu.
- Giải thích: Thuật toán xét qua phải rồi mới hướng xuống, nên theo phương pháp LIFO hướng đi quyết định sẽ là hướng xuống.

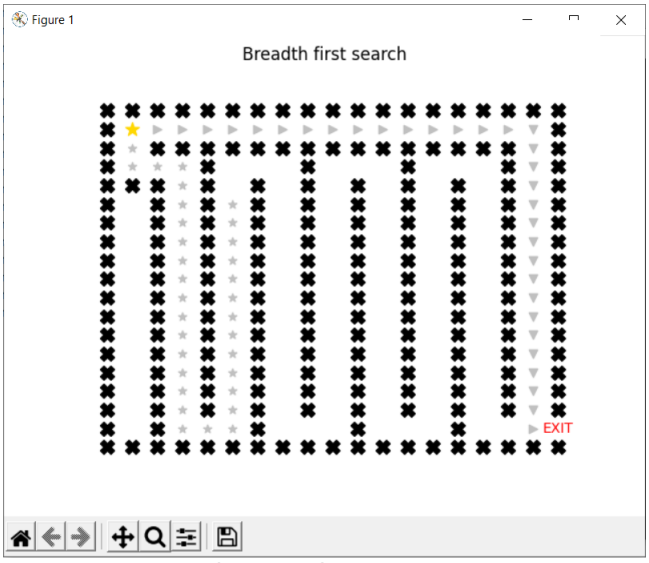


Hình 1.1a. Mê cung 1 – DFS

1.1b. Thuật toán BFS

❖ Nhận xét: Tác nhân lựa chọn hướng đi trong thuật toán này là hướng đi tối ưu.

❖ Giải thích: Thuật toán mở rộng các nốt đều nhau theo mọi phía cho tới khi tìm được lối thoát.

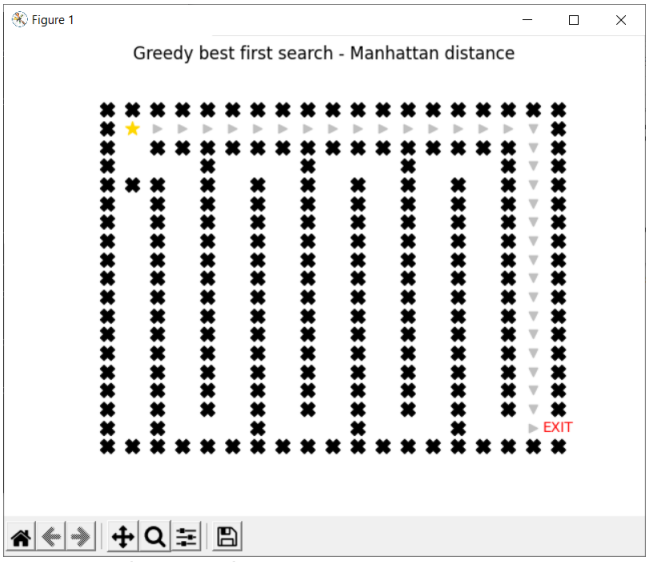


Hình 1.1b. Mê cung 1 – BFS

1.1c. Thuật toán Greedy Best First Search

❖ Nhận xét: Tác nhân lựa chọn hướng đi trong thuật toán này là hướng đi tối ưu.

❖ Giải thích: Thuật toán chỉ mở rộng về phía hướng đi ngắn hơn do sự ước lượng bởi thuật giải.

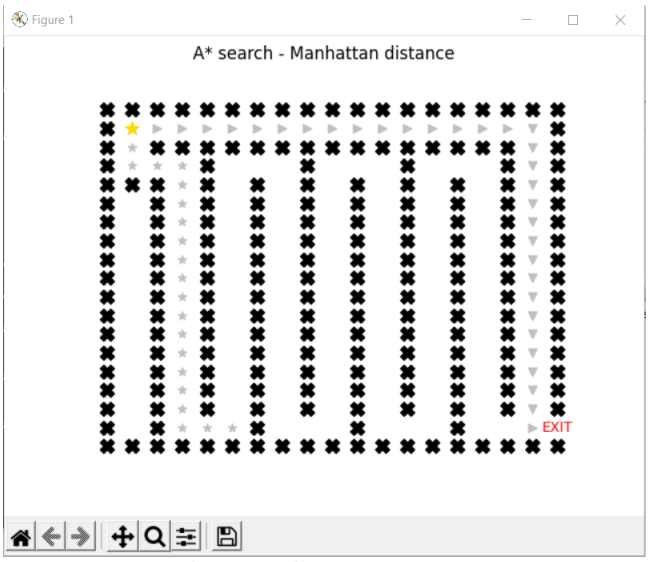


Hình 1.1c. Mê cung 1 – Greedy Best First Search

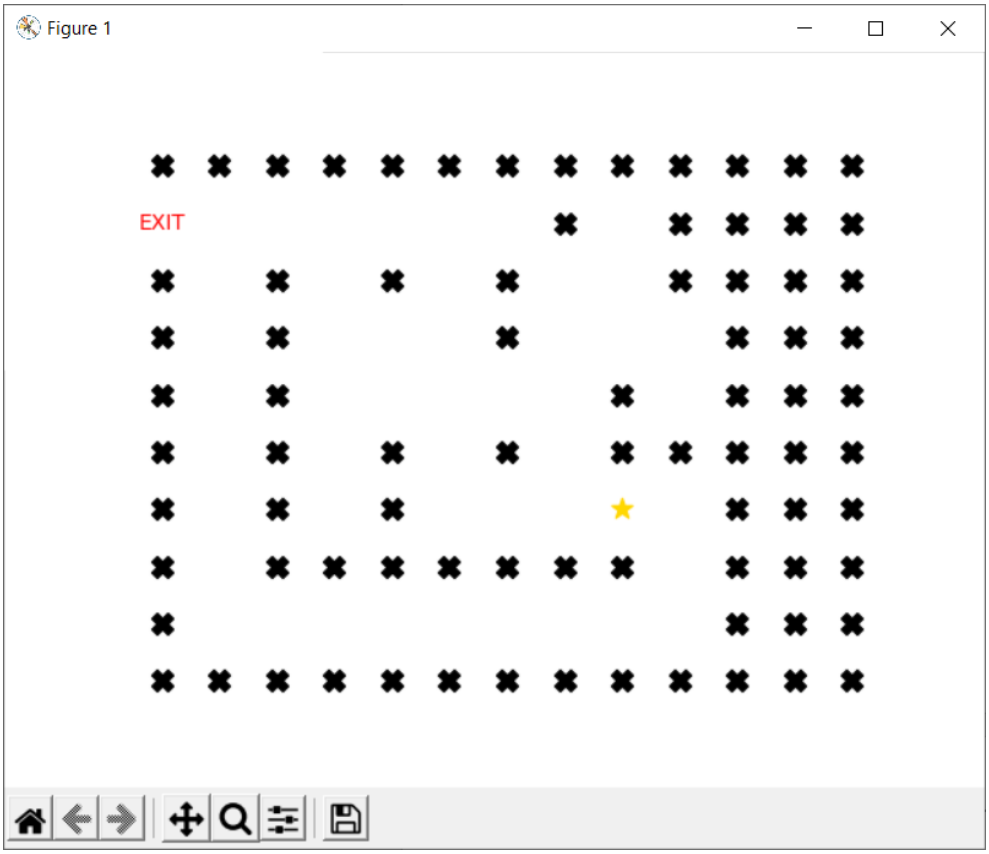
1.1d. Thuật toán A Search*

❖ Nhận xét: Tác nhân lựa chọn hướng đi trong thuật toán này là hướng đi tối ưu.

❖ Giải thích: Thuật toán chỉ mở rộng về phía hướng đi ngắn hơn do sự ước lượng bởi thuật giải và hàm chi phí. Mặc dù tìm ra hướng đi tối ưu nhưng chi phí về thời gian thực thi lại cao hơn thuật toán Greedy Best First Search.



Hình 1.1d. Mê cung 1 – A* Search



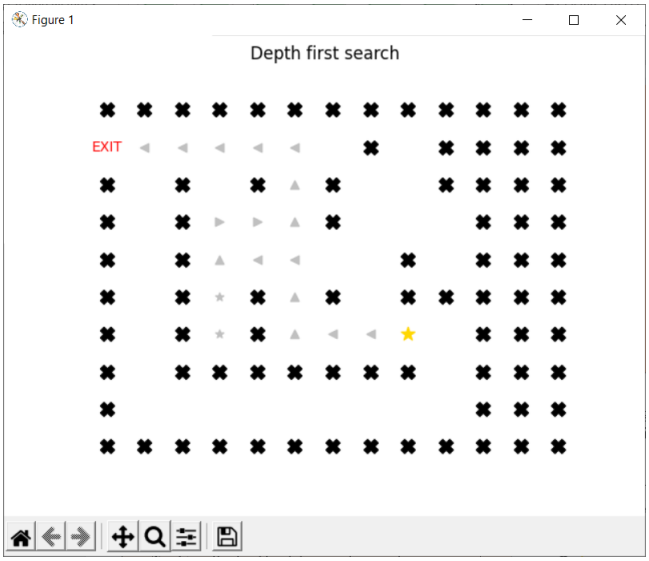
Hình 1.2. Mê cung 2

Mô tả mê cung: Từ điểm xuất phát ta có 2 hướng để đến lối thoát, với một hướng có đường đi không tối ưu và dài hơn hướng còn lại không đáng kể. Nhưng hướng đi này có tọa độ từng nốt gần lối thoát hơn hướng còn lại.

Xét các thuật toán:

1.2a. Thuật toán DFS

- Nhận xét: Tác nhân lựa chọn hướng đi trong thuật toán này là hướng đi không tối ưu.
- Giải thích: Như mê cung 1.1.

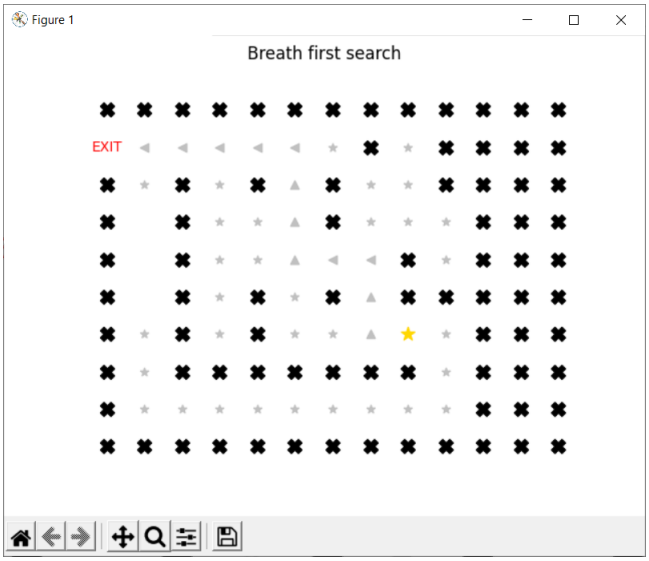


Hình 1.2a. Mê cung 2 – DFS

1.2b. Thuật toán BFS

❖ Nhận xét: Tác nhân lựa chọn hướng đi trong thuật toán này là hướng đi tối ưu.

❖ Giải thích: Thuật toán mở rộng các nốt đều nhau theo mọi phía cho tới khi tìm được lối thoát. Dù thuật toán luôn tìm được đường đi tối ưu trong mê cung nhưng số nốt duyệt để mở rộng của không gian tìm kiếm là rất lớn, dẫn đến không tối ưu về mặt thời gian thực thi.

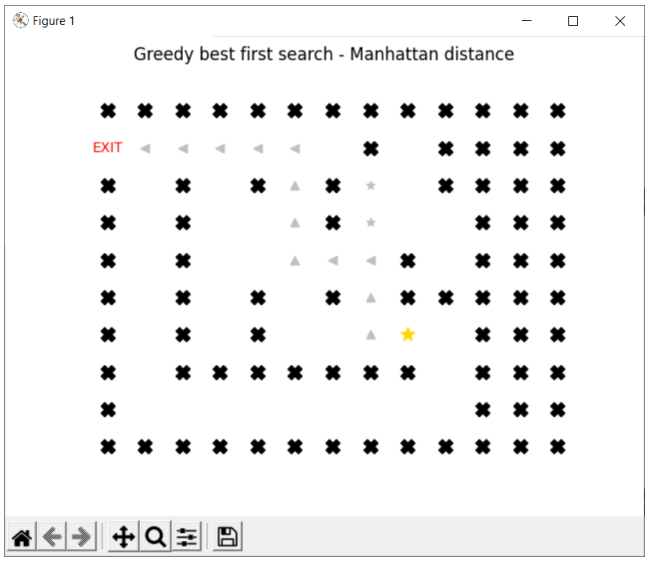


Hình 1.2b. Mê cung 2 – BFS

1.2c. Thuật toán Greedy Best First Search

❖ Nhận xét: Tác nhân lựa chọn hướng đi trong thuật toán này là hướng đi không tối ưu.

❖ Giải thích: Như mê cung 1.1.

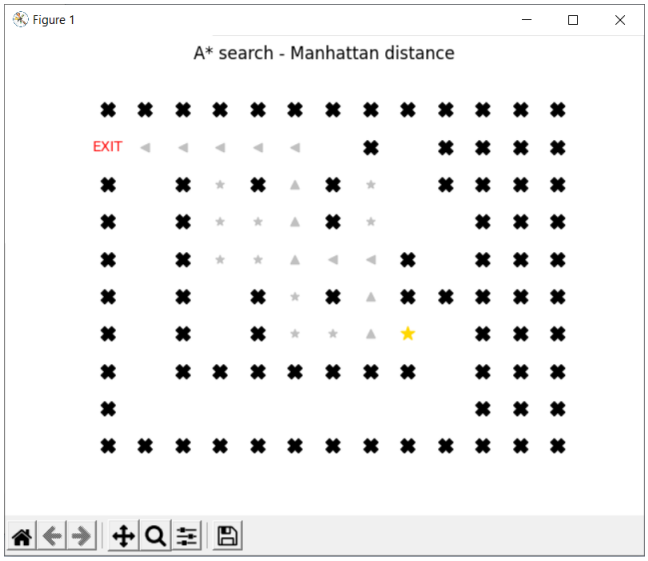


Hình 1.2c. Mê cung 2 – Greedy Best First Search

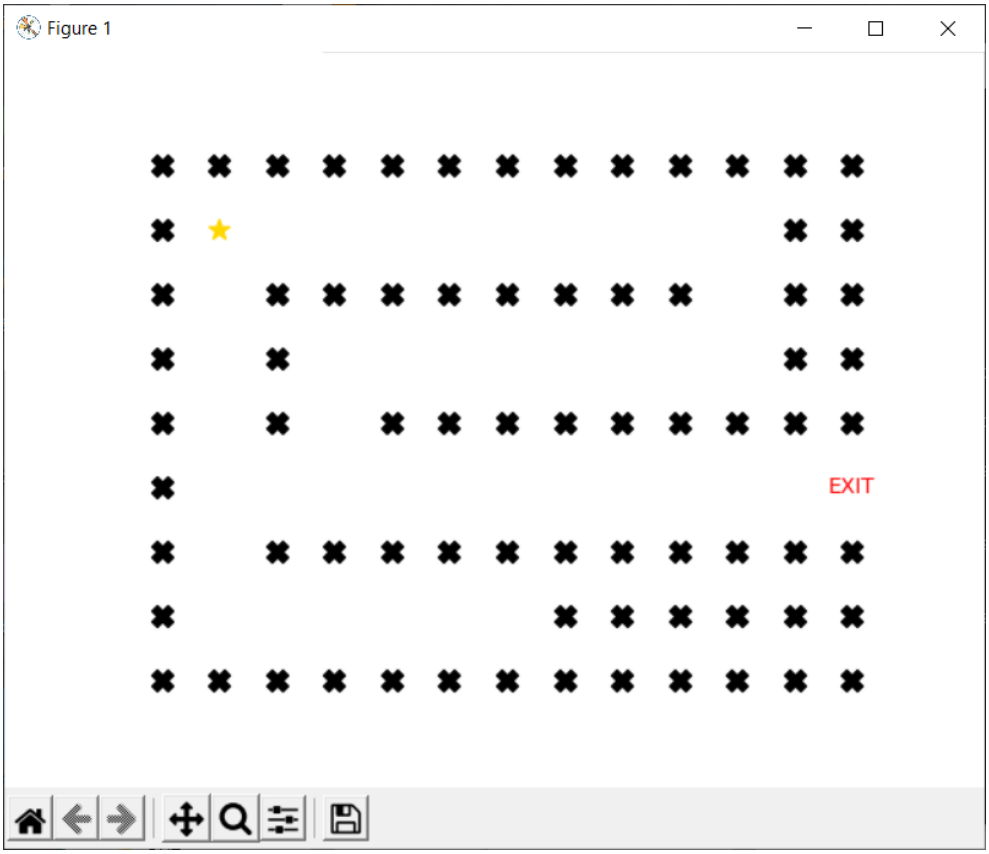
1.2d. Thuật toán A Search*

❖ Nhận xét: Tác nhân lựa chọn hướng đi trong thuật toán này là hướng đi tối ưu.

❖ Giải thích: Như mê cung 1.1.



Hình 1.2d. Mê cung 2 – A* Search



Hình 1.3. Mê cung 3

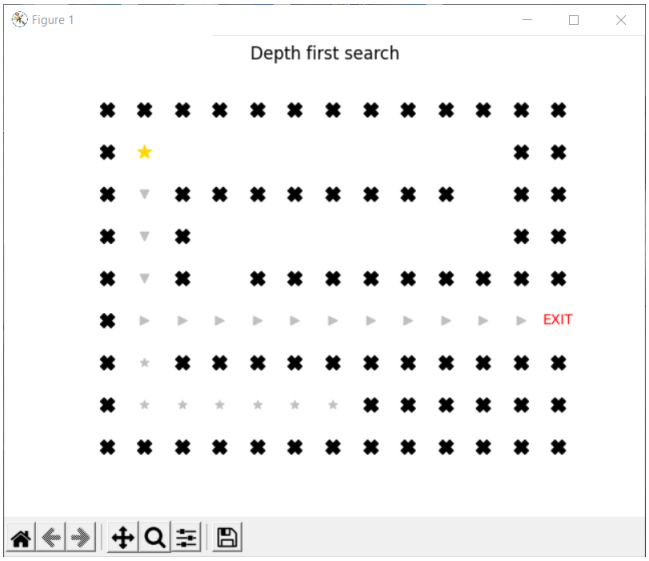
Mô tả mê cung: Từ điểm xuất phát ta có 2 hướng để đến lối thoát, với một hướng có đường đi không tối ưu và dài hơn hướng còn lại.

Xét các thuật toán:

1.2a. Thuật toán DFS

❖ Nhận xét: Tác nhân lựa chọn hướng đi trong thuật toán này là hướng đi tối ưu.

❖ Giải thích: Thuật toán xét qua phải rồi mới hướng xuống, nên theo phương pháp LIFO hướng đi quyết định sẽ là hướng xuống, từ đó giúp tác nhân tìm đến lối thoát theo hướng tối ưu một cách ngẫu nhiên.

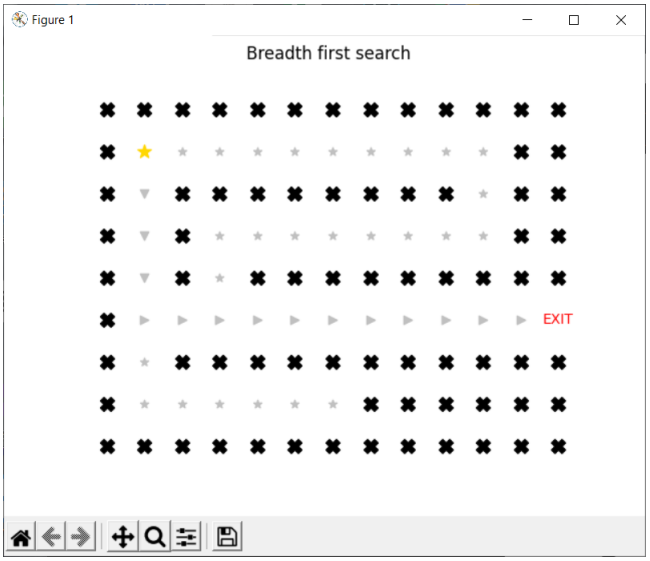


Hình 1.3a. Mê cung 3 – DFS

1.2b. Thuật toán BFS

❖ Nhận xét: Tác nhân lựa chọn hướng đi trong thuật toán này là hướng đi tối ưu.

❖ Giải thích: Như mê cung 1.2.

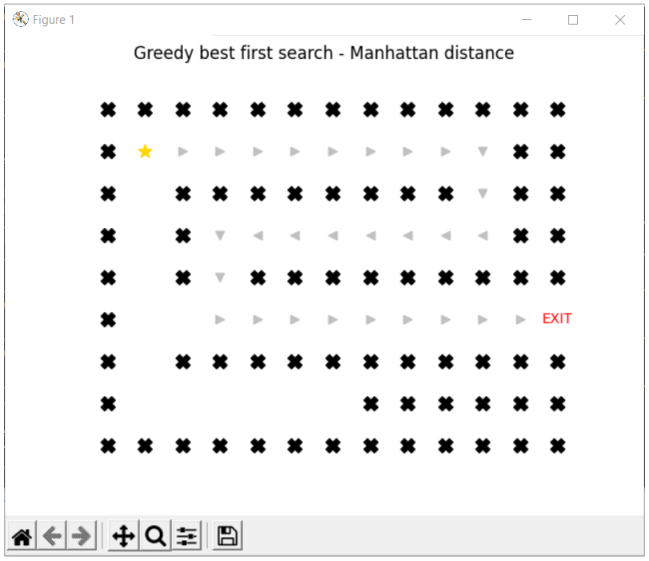


Hình 1.3b. Mê cung 3 – BFS

1.3c. Thuật toán Greedy Best First Search

❖ Nhận xét: Tác nhân lựa chọn hướng đi trong thuật toán này là hướng đi không tối ưu.

❖ Giải thích: Thuật toán chỉ mở rộng về phía hướng đi ngắn hơn do sự ước lượng bởi thuật giải. Trường hợp này, các nốt bên phải được ước lượng là gần lối thoát hơn so với các nốt bên dưới điểm bắt đầu. Do đó, tác nhân đã lựa chọn đi theo hướng bên phải dẫn đến đường đi không tối ưu.

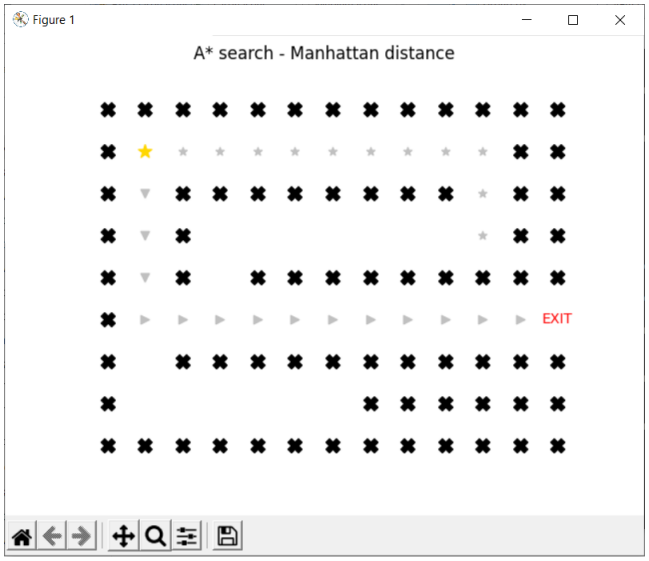


Hình 1.3c. Mê cung 3 – Greedy Best First Search

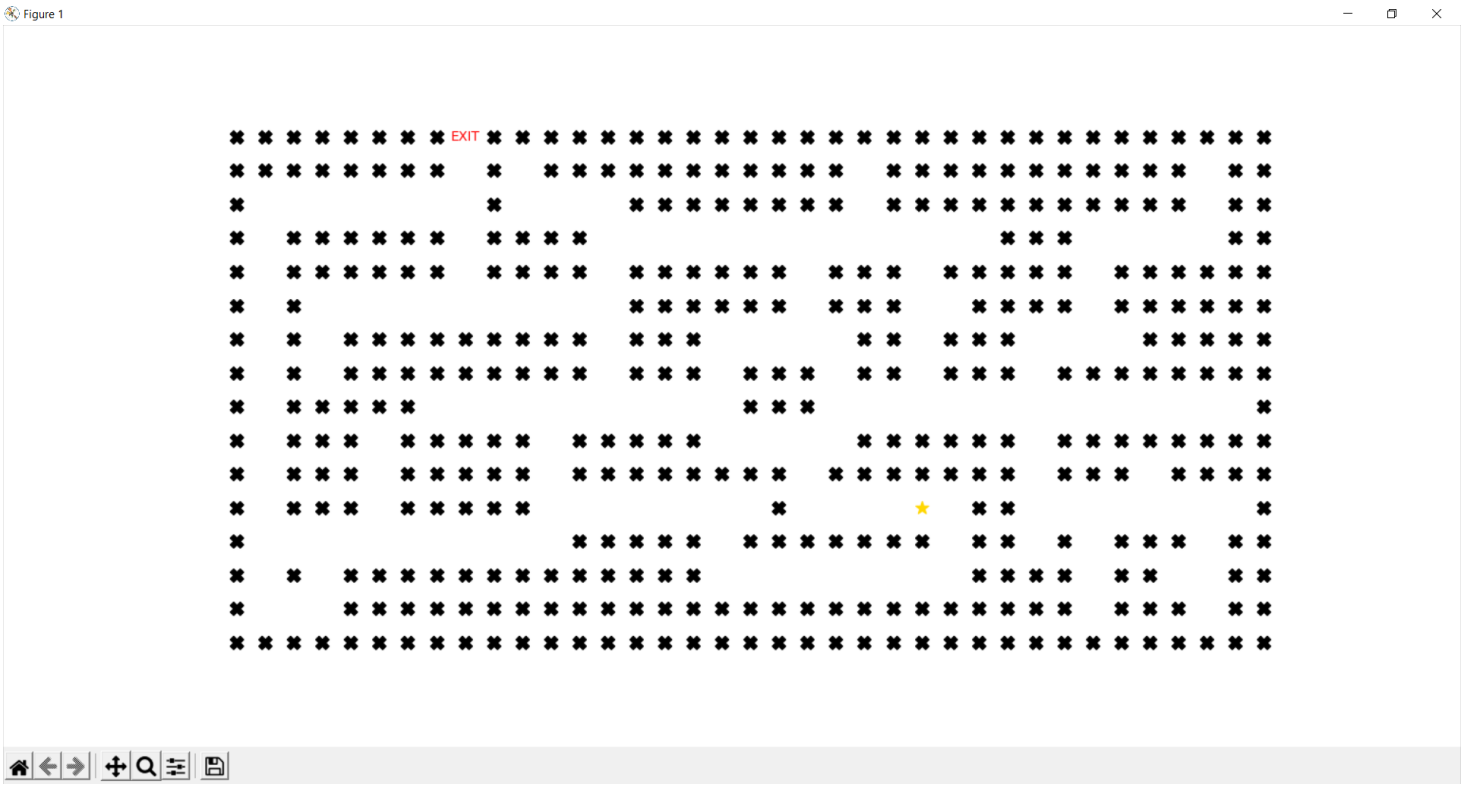
1.3d. Thuật toán A Search*

❖ Nhận xét: Tác nhân lựa chọn hướng đi trong thuật toán này là hướng đi tối ưu.

❖ Giải thích: Thuật toán chỉ mở rộng về phía hướng đi ngắn hơn do sự ước lượng bởi thuật giải và hàm chi phí. Trường hợp này, các nốt bên phải được ước lượng là gần lối thoát hơn so với các nốt bên dưới điểm bắt đầu. Dù vậy, khi xét các nốt thì hàm chi phí cho thấy nếu tác nhân lựa chọn hướng bên phải sẽ tìm ra đường đi dài hơn. Do đó, tác nhân đã lựa chọn hướng đi xuống làm đường đi tối ưu.



Hình 1.3d. Mê cung 3 – A* Search



Hình 1.4. Mê cung 4

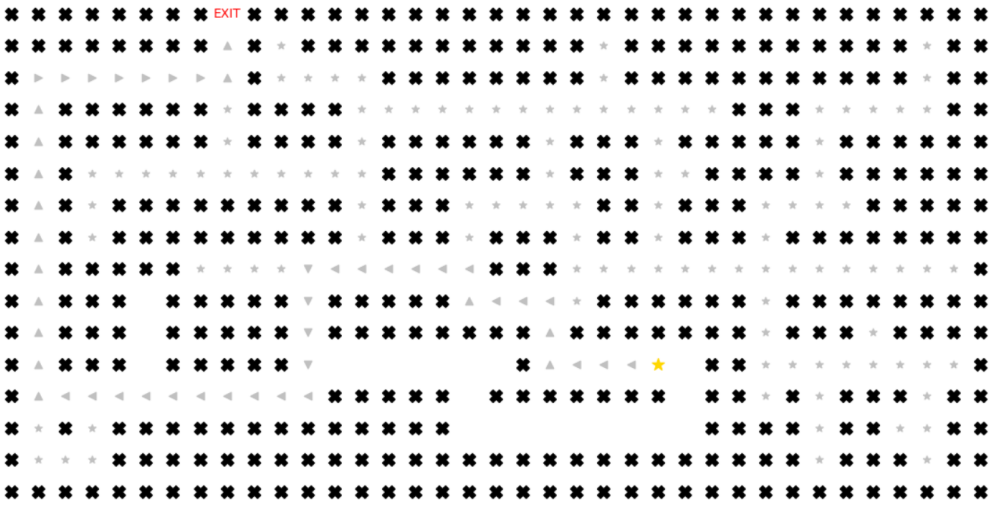
Mô tả mê cung: Mê cung lớn với nhiều chu trình.

Xét các thuật toán:

1.4a. Thuật toán DFS

❖ Nhận xét: Tác nhân lựa chọn hướng đi trong thuật toán này là hướng đi không tối ưu.

❖ Giải thích: Như mê cung 1.1.

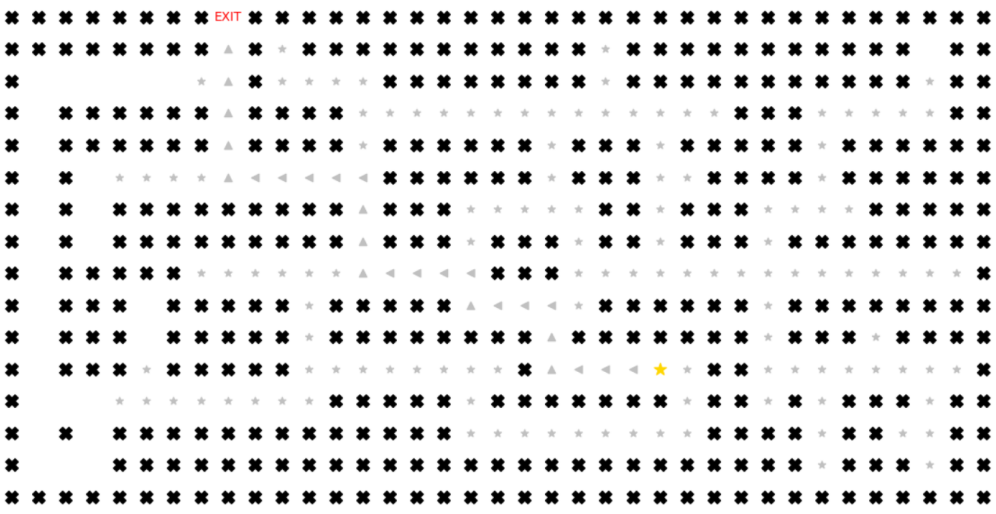


Hình 1.4a. Mê cung 4 – DFS

1.4b. Thuật toán BFS

❖ Nhận xét: Tác nhân lựa chọn hướng đi trong thuật toán này là hướng đi tối ưu.

❖ Giải thích: Như mê cung 1.2.

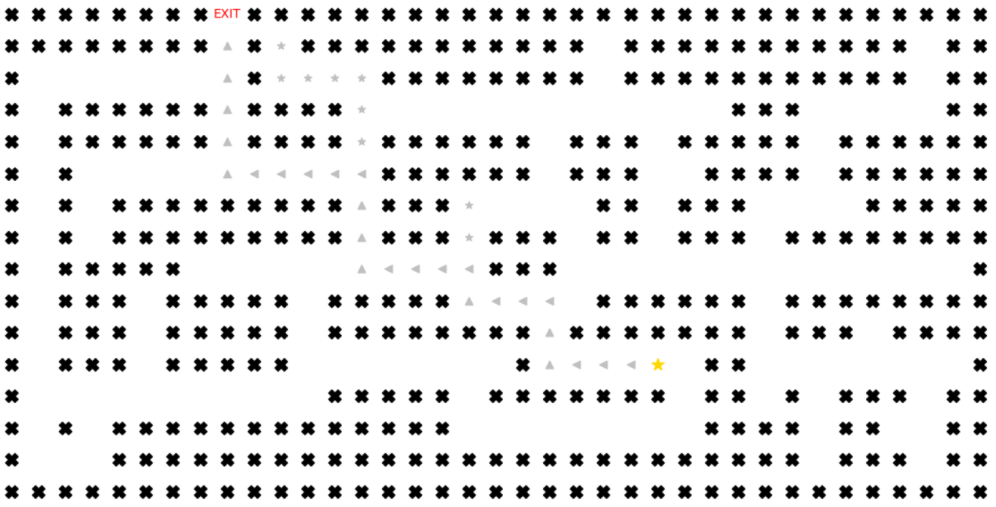


Hình 1.4b. Mê cung 4 – BFS

1.4c. Thuật toán Greedy Best First Search

❖ Nhận xét: Tác nhân lựa chọn hướng đi trong thuật toán này là hướng đi tối ưu.

❖ Giải thích: Như mê cung 1.1.

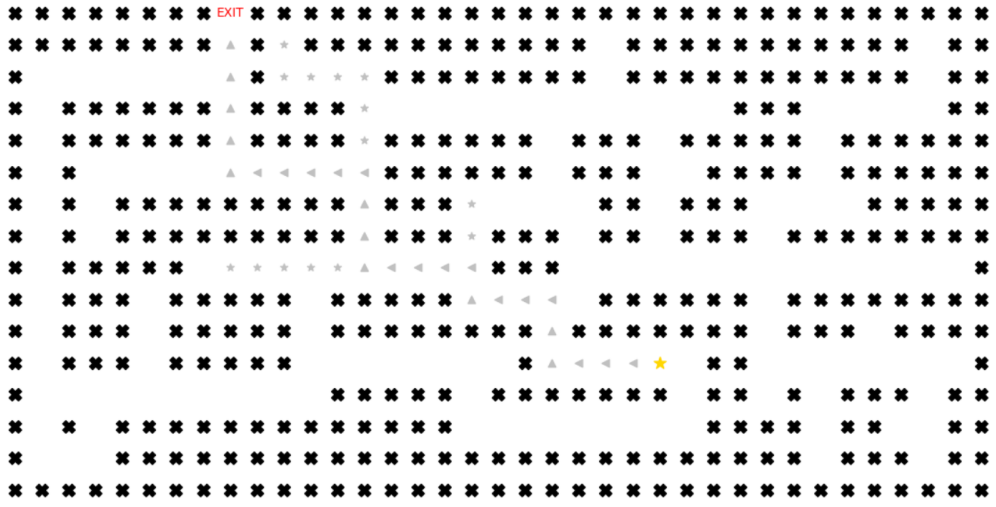


Hình 1.4c. Mê cung 4 – Greedy Best First Search

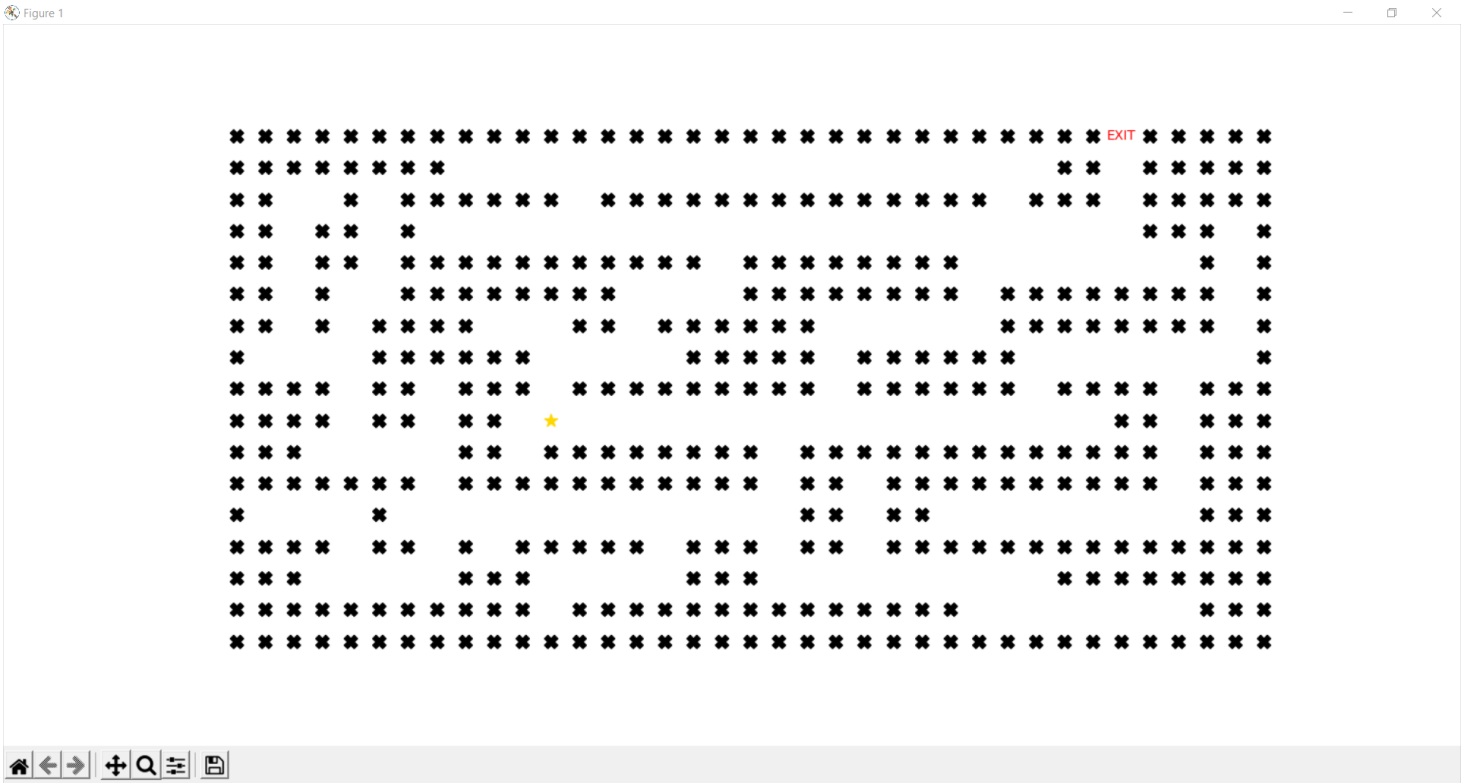
1.4d. Thuật toán A Search*

❖ Nhận xét: Tác nhân lựa chọn hướng đi trong thuật toán này là hướng đi tối ưu.

❖ Giải thích: Thuật toán chỉ mở rộng về phía hướng đi ngắn hơn do sự ước lượng bởi thuật giải và hàm chi phí. Trường hợp này, thuật toán tìm ra đường đi tối ưu với chi phí thấp nhất trong tất cả các thuật toán đang xét.



Hình 1.4d. Mê cung 4 – A* Search



Hình 1.5. Mê cung 5

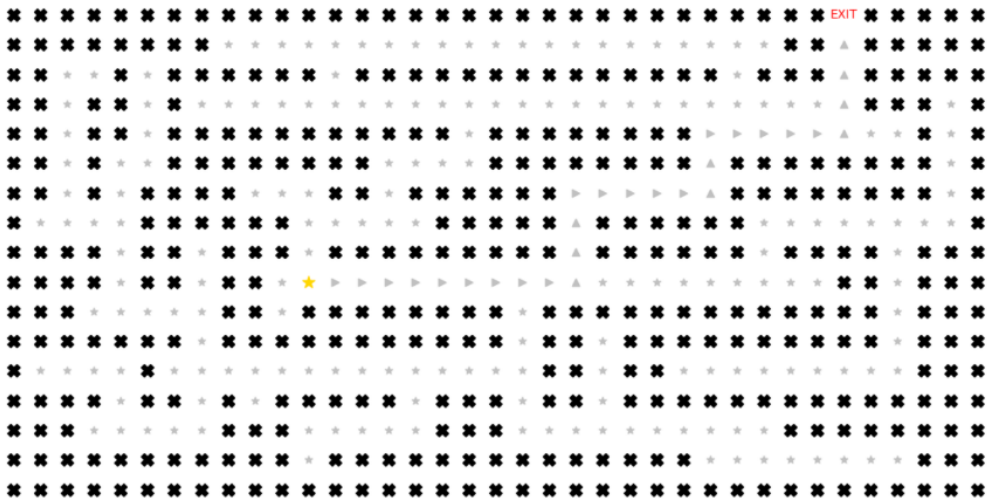
Mô tả mê cung: Mê cung lớn với nhiều hướng đi tối ưu.

Xét các thuật toán:

1.5a. Thuật toán DFS

❖ Nhận xét: Tác nhân lựa chọn hướng đi trong thuật toán này là hướng đi tối ưu.

❖ Giải thích: Dù cho mê cung chỉ có các hướng đi tối ưu, nhưng thuật toán phải mở rộng một không gian tìm kiếm tối đa. Do đó xét về thời gian thực thi thì thuật toán không phát huy được thế mạnh.

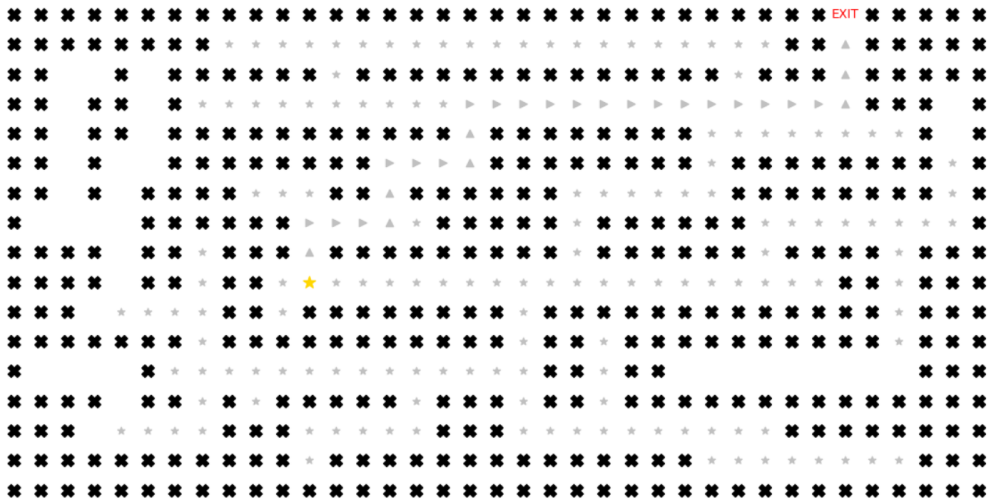


Hình 1.5a. Mê cung 5 – DFS

1.5b. Thuật toán BFS

❖ Nhận xét: Tác nhân lựa chọn hướng đi trong thuật toán này là hướng đi tối ưu.

❖ Giải thích: Thuật toán lựa chọn đường đi tối ưu khác với thuật toán DFS nhưng vẫn không hiệu quả về thời gian thực thi.



Hình 1.5b. Mê cung 5 – BFS

1.5c. Thuật toán Greedy Best First Search

❖ Nhận xét: Tác nhân lựa chọn hướng đi trong thuật toán này là hướng đi tối ưu.

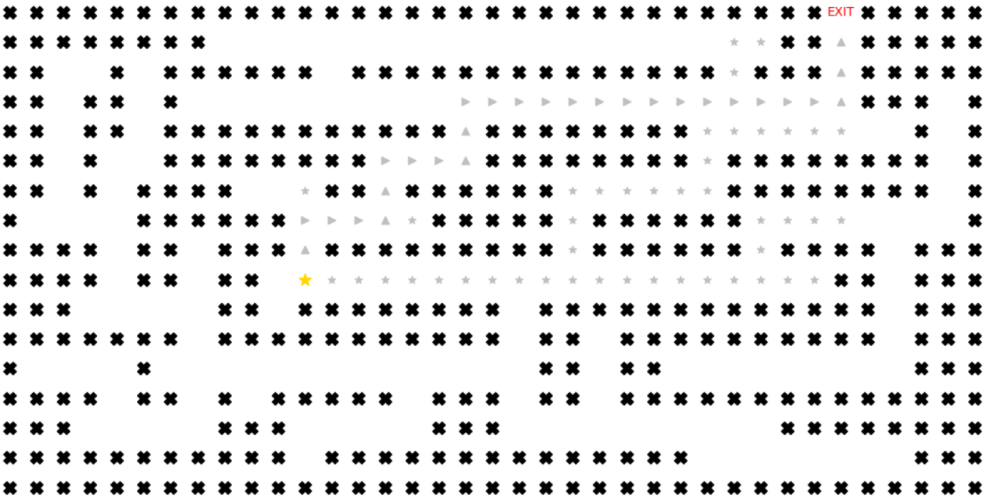
❖ Giải thích: Thuật toán lựa chọn đường đi tối ưu giống thuật toán BFS nhưng thể hiện tối đa tính hiệu quả về thời gian thực thi.

Hình 1.5c. Mê cung 5 – Greedy Best First Search

1.5d. Thuật toán A Search*

❖ Nhận xét: Tác nhân lựa chọn hướng đi trong thuật toán này là hướng đi tối ưu.

❖ Giải thích: Thuật toán lựa chọn đường đi tối ưu giống thuật toán BFS và thể hiện sự hiệu quả về thời gian thực thi. Song, trong trường hợp này thuật toán không thật sự hiệu quả bằng thuật toán Greedy Best First Search.

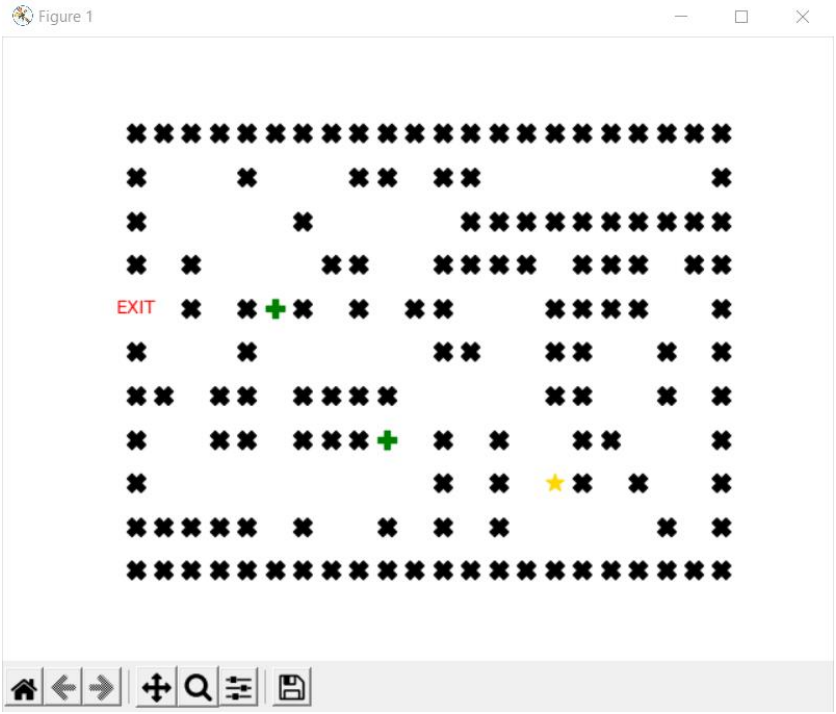


Hình 1.5d. Mê cung 5 – A* Search

2. Mê cung có điểm thưởng

Mô tả thuật toán: Sử dụng thuật toán BFS với hàm thuật giải là Dijkstra cải tiến – Có khả năng duyệt các nốt có trọng số âm nhờ vào việc áp dụng hàng đợi ưu tiên.

Xét các mê cung:

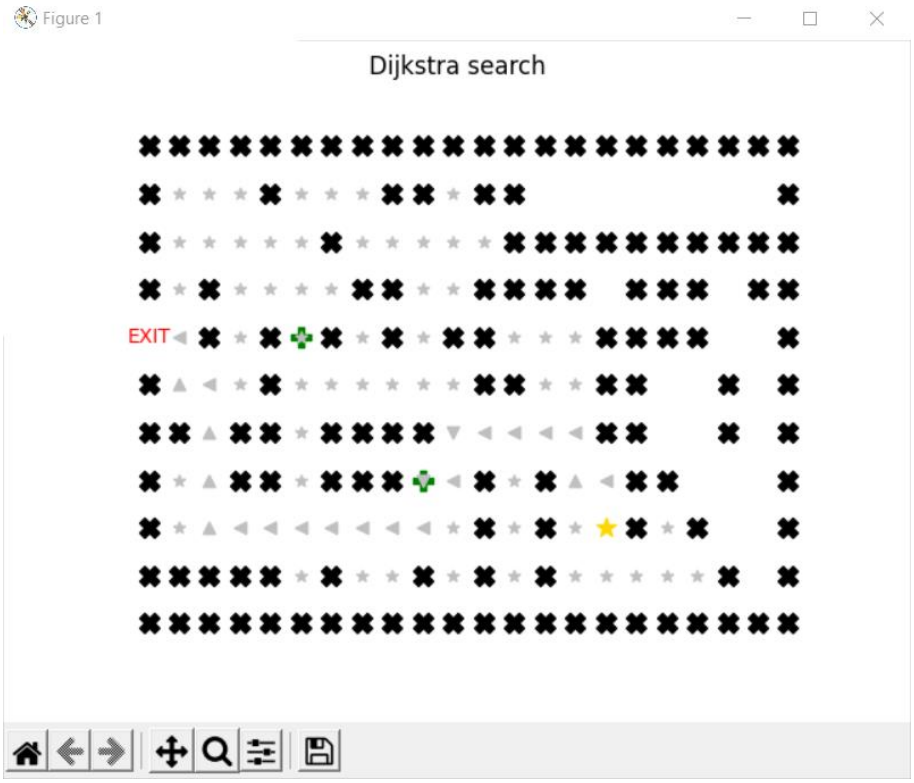


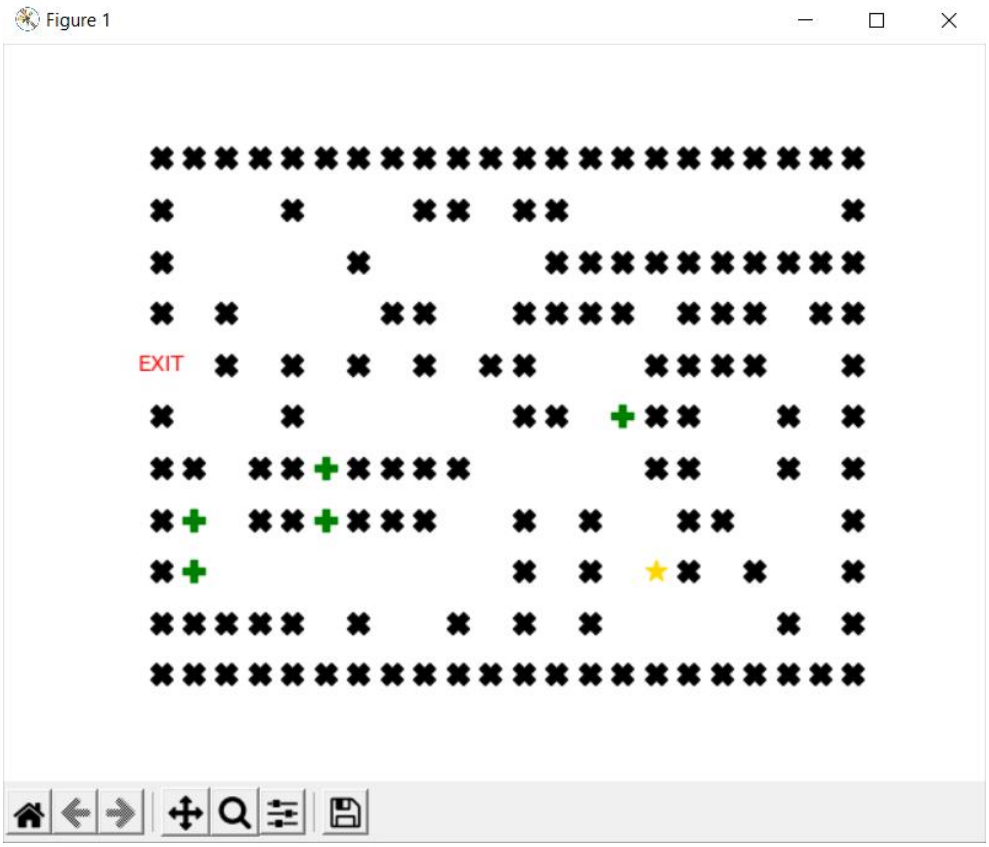
Hình 2.1. Mê cung 1

Mô tả mê cung:

Tọa độ X	Tọa độ Y	Điểm
4	5	-2
7	9	-5

Kết quả sau khi chạy thuật toán



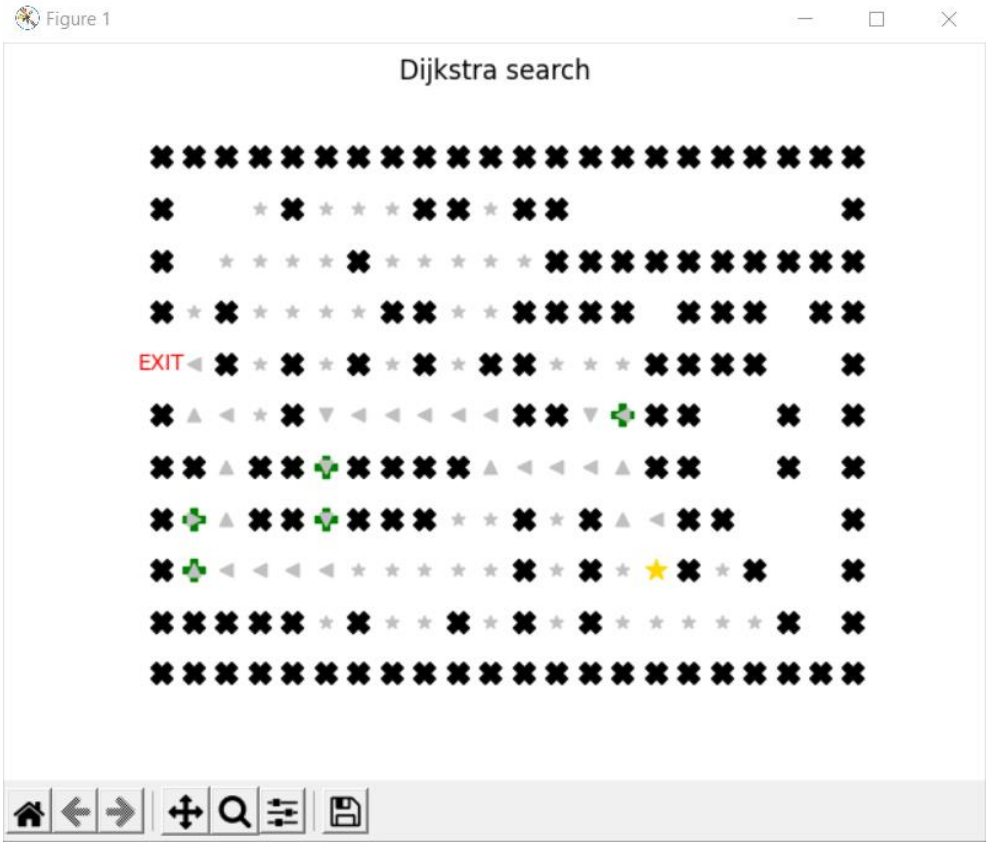


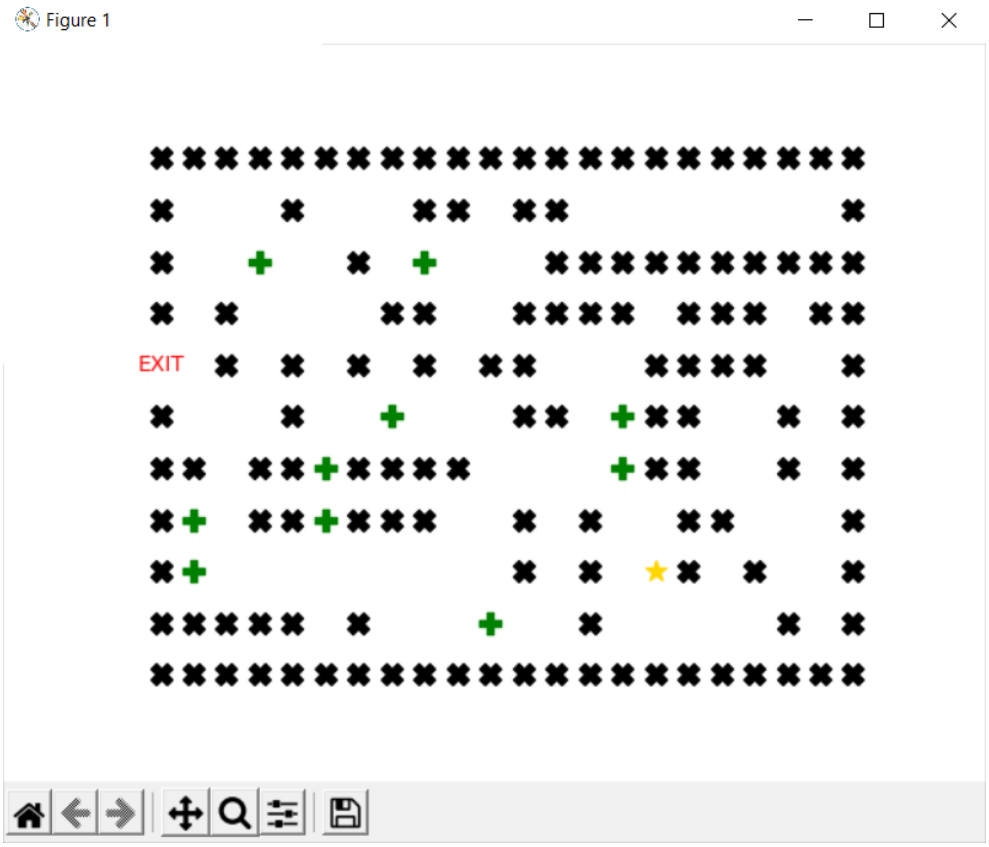
Hình 2.2. Mê cung 2

Mô tả mê cung:

Tọa độ X	Tọa độ Y	Điểm
5	14	-4
6	5	-1
7	5	-1
7	1	-1
8	1	-1

Kết quả sau khi chạy thuật toán



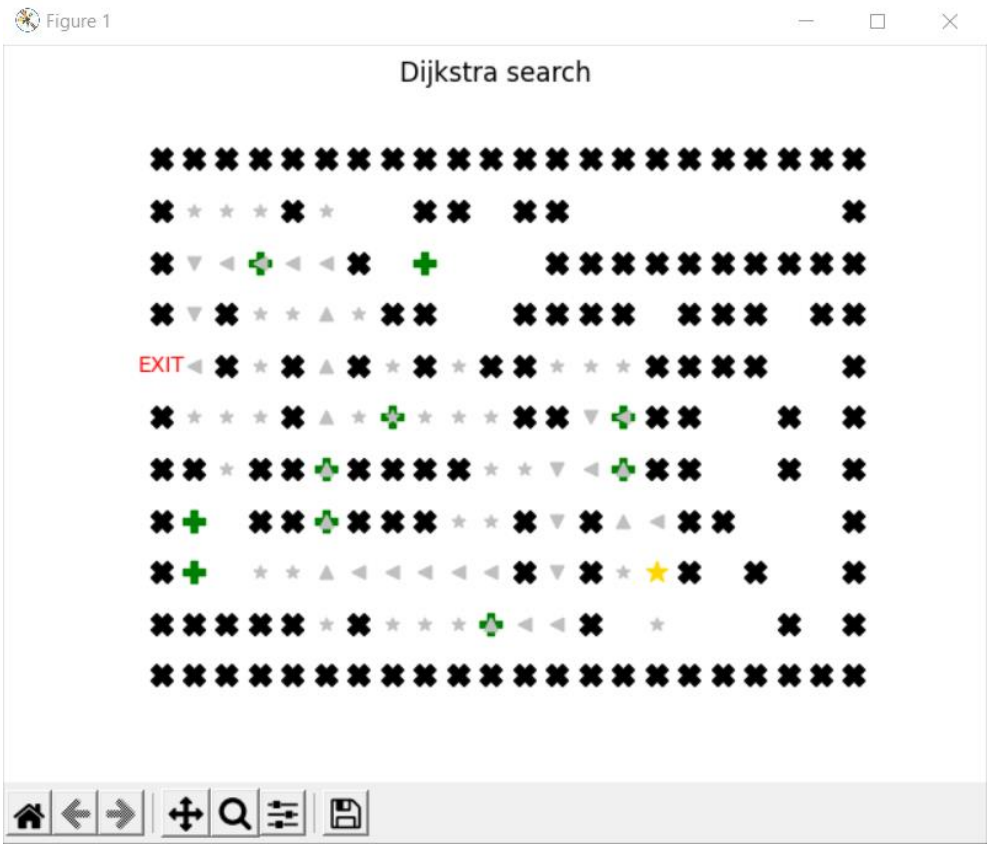


Hình 2.3. Mê cung 3

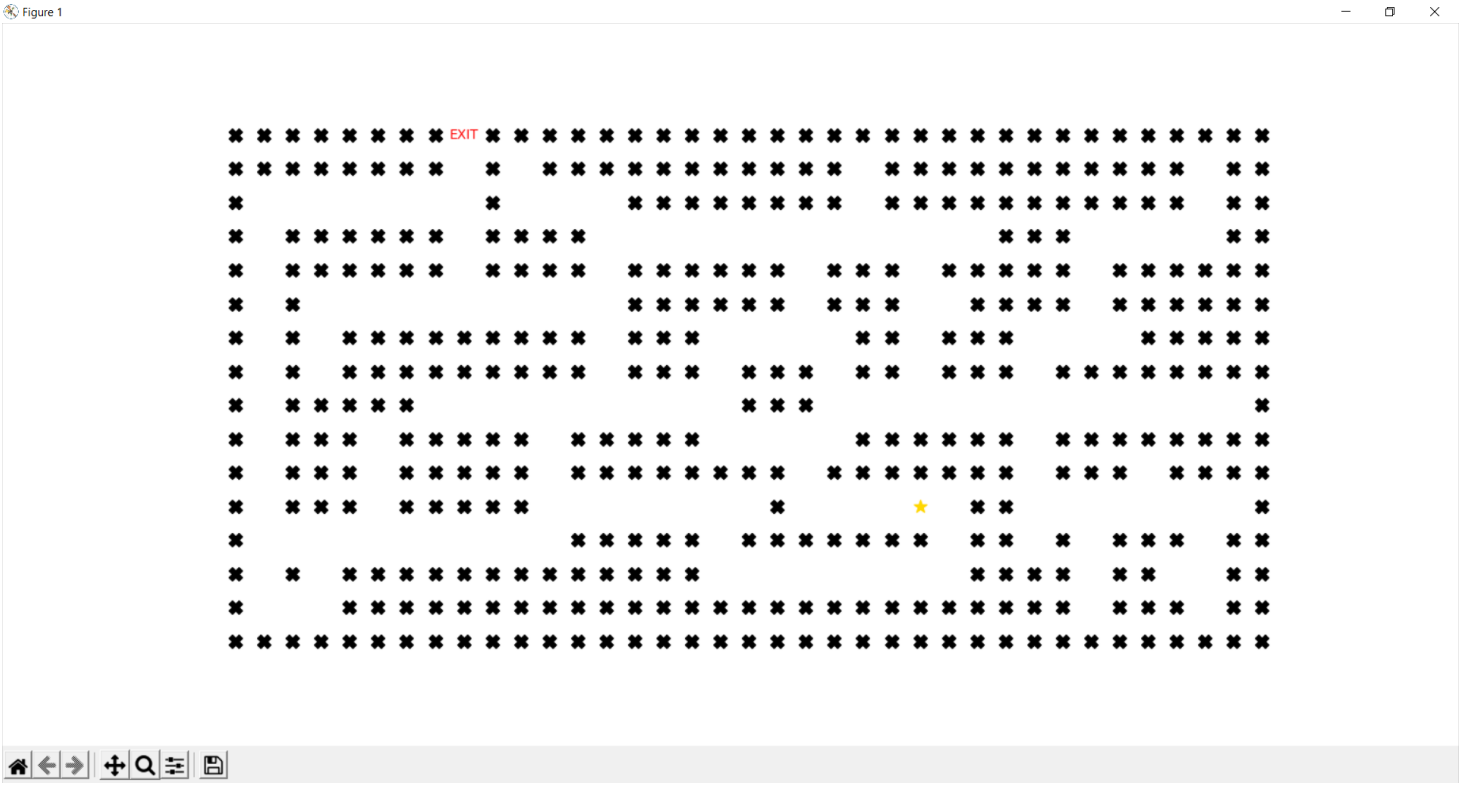
Mô tả mê cung:

Tọa độ X	Tọa độ Y	Điểm
2	3	-9
2	8	-7
5	7	-1
5	14	-6
6	14	-2
6	5	-2
7	5	-1
7	1	-3
8	1	-4
9	10	-10

Kết quả sau khi chạy thuật toán



3. So sánh các thuật giải trên thuật toán A* search



Hình 3. Mê cung lớn

Xét các thuật giải

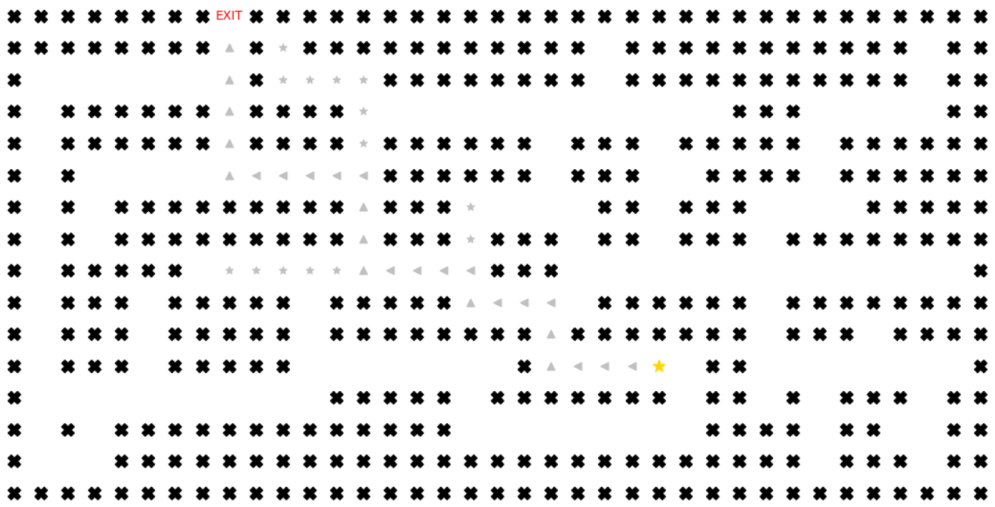
3.1. Thuật giải Manhattan

❖ Giải thích:

- Thuật giải sử dụng hàm thuật giải:

```
function heuristic(node) =
  dx = abs(node.x - goal.x)
  dy = abs(node.y - goal.y)
  return D * (dx + dy)
```

- Ứng dụng tốt khi tác nhân chỉ có 4 hướng di chuyển tại 1 thời điểm.



Hình 3.1. Mê cung lớn – Manhattan

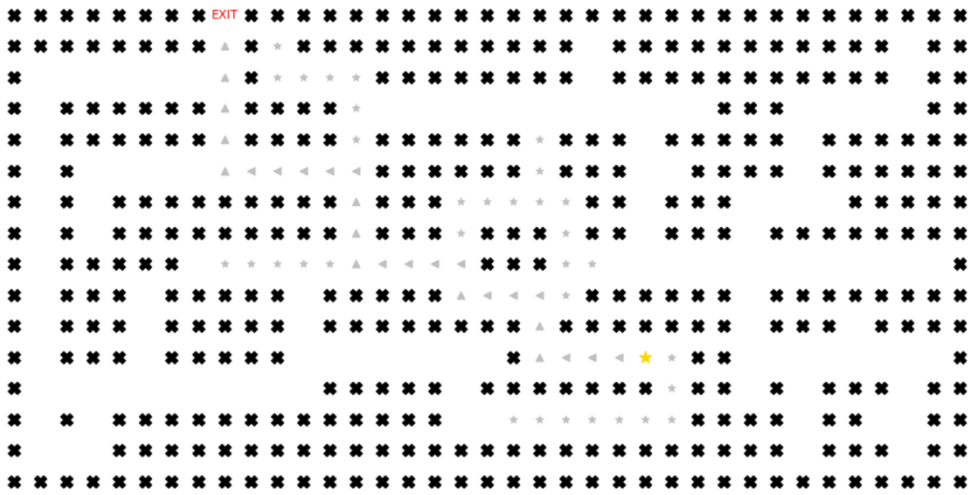
3.2. Thuật giải đường chéo

❖ Giải thích:

- Thuật giải sử dụng hàm thuật giải:

```
function heuristic(node) =
  dx = abs(node.x - goal.x)
  dy = abs(node.y - goal.y)
  return D * (dx + dy) + (D2 - 2 * D)
* min(dx, dy)
```

- Ứng dụng tốt khi tác nhân có 8 hướng di chuyển tại 1 thời điểm.



Hình 3.2. Mê cung lớn – Đường chéo

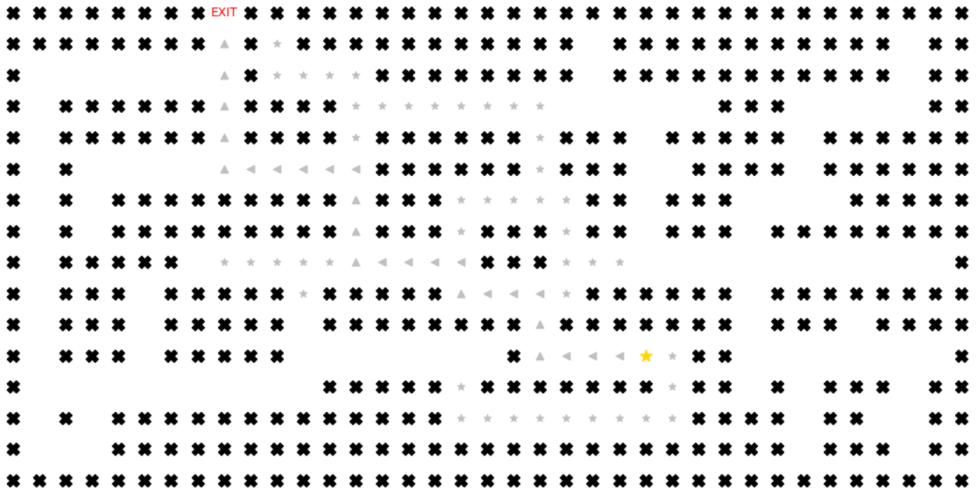
3.3. Thuật giải Euclid

❖ Giải thích:

- Thuật giải sử dụng hàm thuật giải:

```
function heuristic(node) =
  dx = abs(node.x - goal.x)
  dy = abs(node.y - goal.y)
  return D * sqrt(dx * dx + dy * dy)
```

- Ứng dụng tốt khi tác nhân có thể di chuyển mọi hướng tại 1 thời điểm.



Hình 3.3. Mê cung lớn – Euclid

REFERENCES:

[1] Heuristics From Amit's Thoughts on Pathfinding, Standford

http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html

[2] A* search algorithm, Wikipedia

https://en.wikipedia.org/wiki/A*_search_algorithm

[3] CS50's Introduction to Artificial Intelligence with Python 2020, Havard

shorturl.at/ezJWY