KHOA CÔNG NGHỆ THÔNG TIN
ĐỒ ÁN MÔN HỌC CƠ SỞ TRÍ TUỆ NHÂN TẠO

THUẬT TOÁN TÌM KIẾM

Thành viên:
Hồ Công Lượng – 19120572
Nguyễn Thanh Minh - 19120587

I. Đánh giá mức độ hoàn thành:

Trường hợp bản đồ không có điểm thưởng (80%):

Yêu cầu	Mức độ hoàn thành
Thuật toán tìm kiếm DFS (Depth First Search).	100%
Thuật toán tìm kiếm BFS (Breadth First Search).	100%
Thuật toán tìm kiếm tham lam (Greedy Best First Search).	100%
Thuật toán tìm kiếm A*.	100%

Trường hợp bản đồ có điểm thưởng (20%):

Yêu cầu	Mức độ hoàn thành
Giải quyết và đề xuất chiến lược để tác nhân di chuyển sao cho chi phí đường đi từ điểm bắt đầu đến điểm thoát khỏi mê cung là nhỏ nhất có thể.	100%

II. Kết quả thực thi:

1) Trường hợp bản đồ không có điểm thưởng:

*Sự khác nhau giữa các chiến lược tìm kiếm:

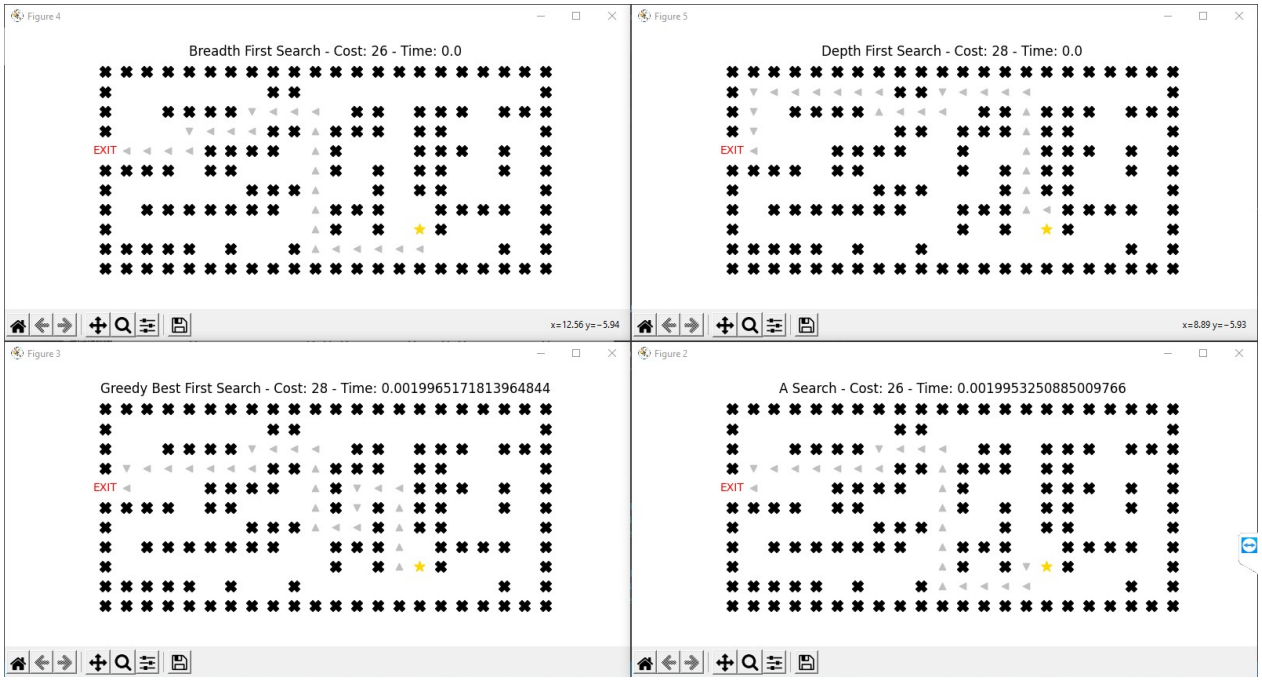
+ Breadth First Search: Thuật toán này luôn mở rộng nút gốc đầu tiên, sau đó tất cả những nút con có từ nút gốc sẽ được mở rộng, tiếp theo và sau đó là nút con của chúng và cứ thế tiếp tục cho tới khi gặp đích, tiết kiệm về mặt chi phí hơn so với DFS.

+ Depth First Search: Thuật toán này luôn mở rộng nút sâu nhất mãi theo một hướng, chỉ khi nào không mở rộng được nữa thì mới chuyển sang hướng khác (trong cài đặt là ưu tiên theo hướng theo thứ tự trên, trái, phải, dưới), cho đến khi gặp điểm đích thì sẽ dừng lại, tiết kiệm về mặt không gian.

+ Greedy Best First Search: Thuật toán mở rộng nút gần đích nhất với hi vọng cách làm này sẽ dẫn đến một lời giải nhanh nhất. Do đó chi phí thuật toán này đánh giá chi phí của các nút chỉ dựa trên hàm heuristic.

+ A*: Thuật toán này đánh giá một nút dựa trên chi phí đi từ nút gốc đến nút đó, cộng với chi phí từ nút đó đến đích. Để đảm bảo A* tìm được lời giải tối ưu thì hàm heuristic phải chấp nhận được, có nghĩa là nó không bao giờ ước lượng quá chi phí đến đích thực sự.

a) Bản đồ 1 (11x22):



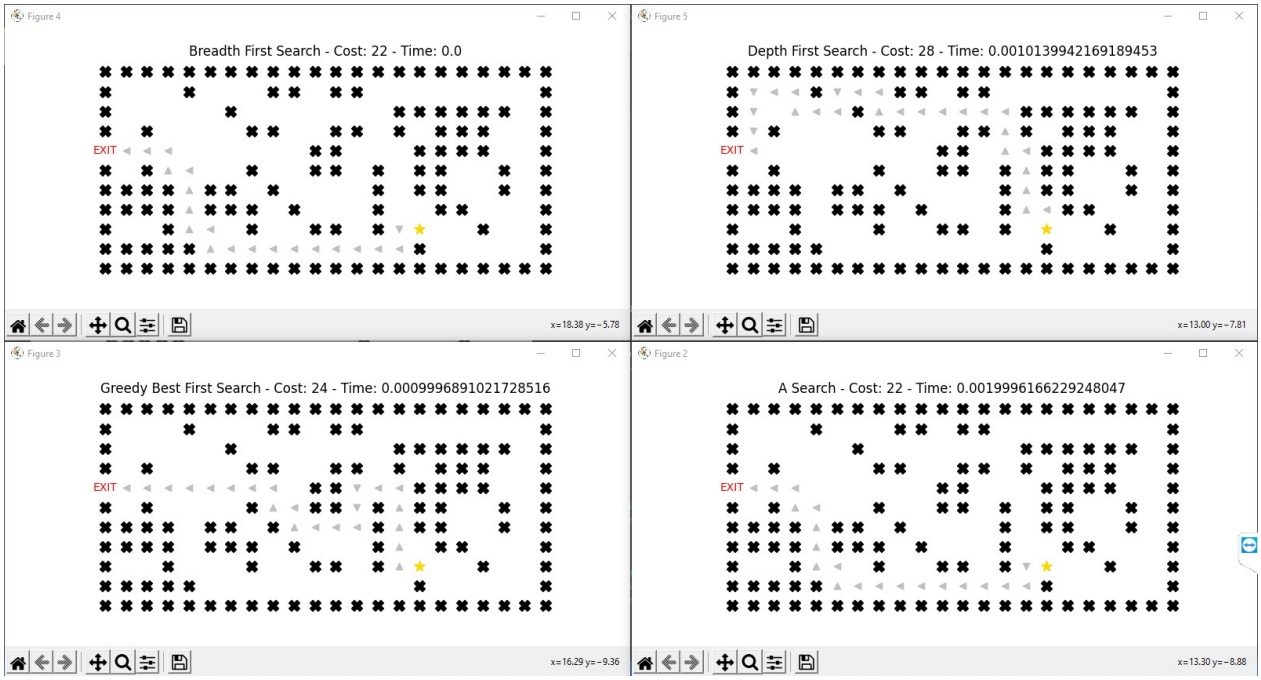
+ Breadth First Search: BFS sẽ đi theo mọi hướng để tới được đích.

+ Depth First Search: DFS sẽ ưu tiên hướng lên và duyệt tới đích.

+ Greedy Best First Search: GBFS sẽ duyệt ưu tiên về hướng các điểm lận cận EXIT theo hàm heuristic.

+ A*: Do có cộng thêm chi phí nên sẽ duyệt tới các điểm một cách có chủ đích.

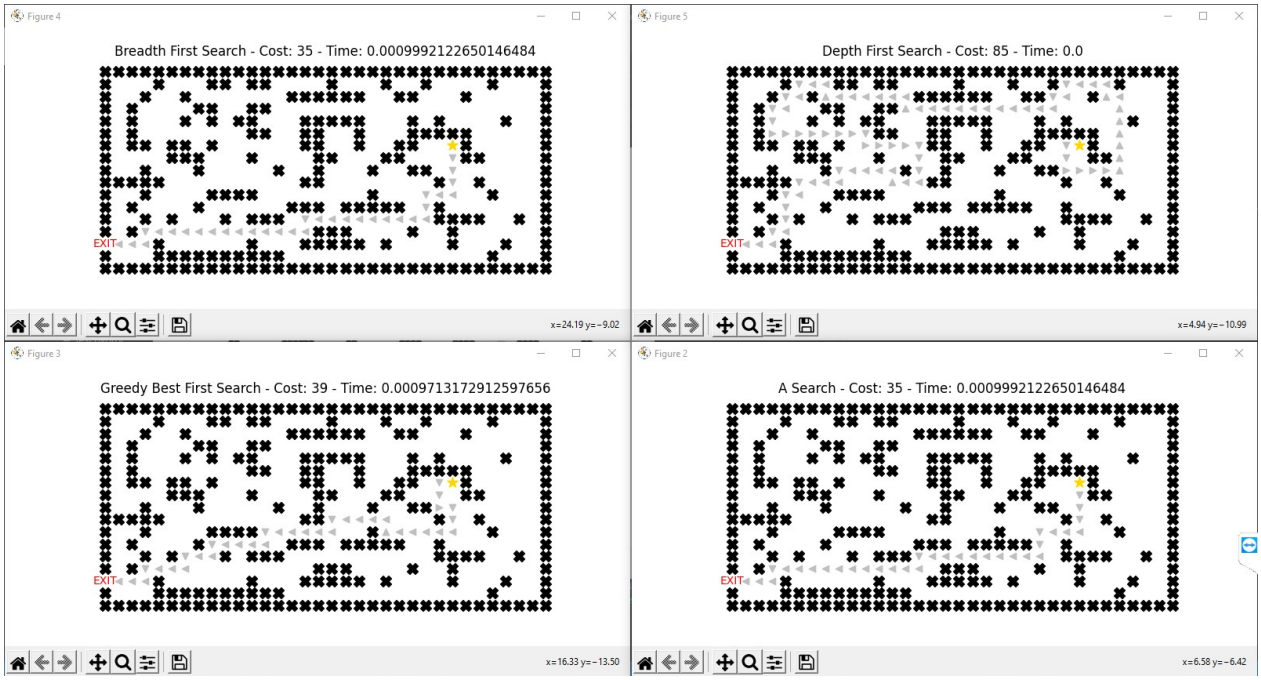
b) Bản đồ 2 (11x22):



+ BFS thực thi tốt trong tình huống này là do mọi chi phí là ngang bằng nhau và bằng 1, nếu thay thế bằng các đồ thị khác sẽ cho ra kết quả tốn chi phí hơn so với GBFS và A* Search.

+ A* nhờ có cộng thêm chi phí nên đã tính được đường đi có chi phí thấp hơn GBFS.

c) Bản đồ 3 (17x34):

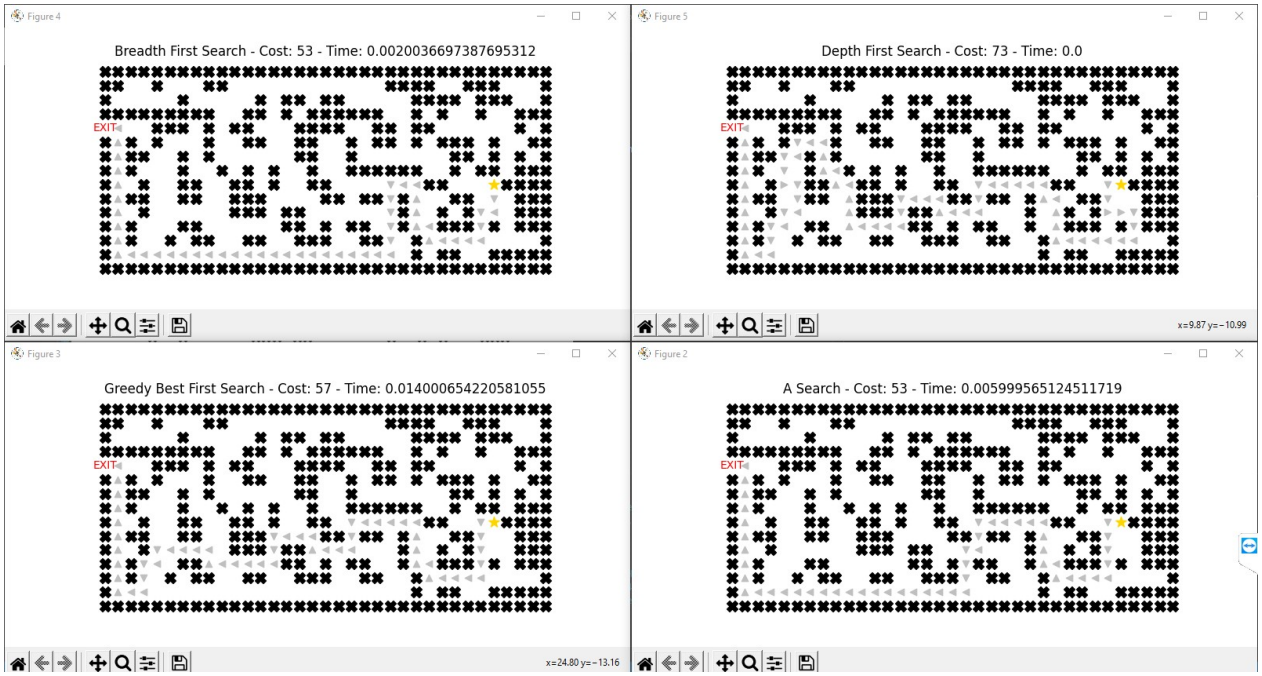


+ Depth First Search: DFS lúc này duyệt sang trái trước theo đúng thứ tự đã nói.

+ Greedy Best First Search: do không có chênh lệch giữa các chi phí nên kết quả trong trường hợp này không quá tối ưu.

+ A*: Đạt được trường hợp tối ưu nhờ các thông tin đưa vào.

d) Bản đồ 4 (15x35):

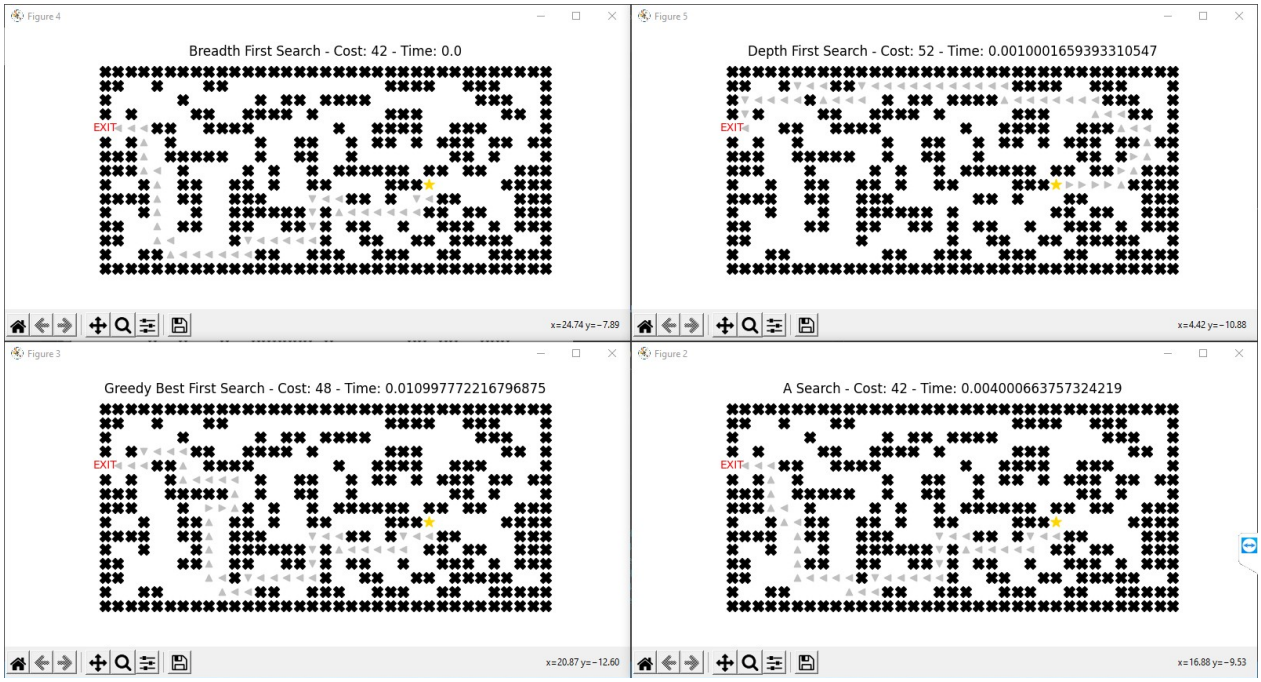


+ Depth First Search: Duyệt theo hướng trái vì trên đã bị tường ngăn chặn.

+ GBFS: Các điểm có heuristic nhỏ tạo nên một đường đi không tối ưu, GBFS không giải quyết tốt trong trường hợp này.

+ A*: có sự khác biệt giữa BFS là do luôn mở rộng ra các điểm có f(n) bé nhất trong các điểm lận cận.

e) Bản đồ 5 (15x35):



+ Các chiến lược khác nhau được thể hiện rõ qua đường đi của từng thuật toán.

+ Trong trường hợp này các điểm có khoảng cách heuristic gần hơn(Trong ảnh là khoảng cách Euclid) tập trung với nhau thành một cụm sẽ khiến cho chi phí của GBFS tăng lên rất nhiều

+ Với thời gian thực thi lâu hơn A* Search đã tính toán và không duyệt theo các điểm có heuristic gần hơn trong một cụm mà duyệt các điểm có tổng chi phí ít hơn.

Kết luận:

+ Nhìn chung qua 5 bản đồ ta thấy được các thuật toán đều có ưu và nhược ở từng trường hợp khác nhau nhưng đa số A* Search và BFS vẫn làm tốt hơn 2 thuật toán còn lại cho đặc thù của bản đồ, GBFS và DFS tạo nên những đường đi không tiết kiệm về chi phí, ảnh hưởng đến kết quả bài toán.

+ Không có thuật toán nào là tối ưu 100% cho mọi trường hợp.

*Sự khác nhau giữa 2 hàm heuristic:

```
def space(vertex, goal):
    return sqrt((vertex[0] - goal[0])**2 + (vertex[1] - goal[1])**2)

def space2(vertex, goal):
    return abs(goal[0] - vertex[0]) + abs(goal[1] - vertex[1])
```

- Trong đó:

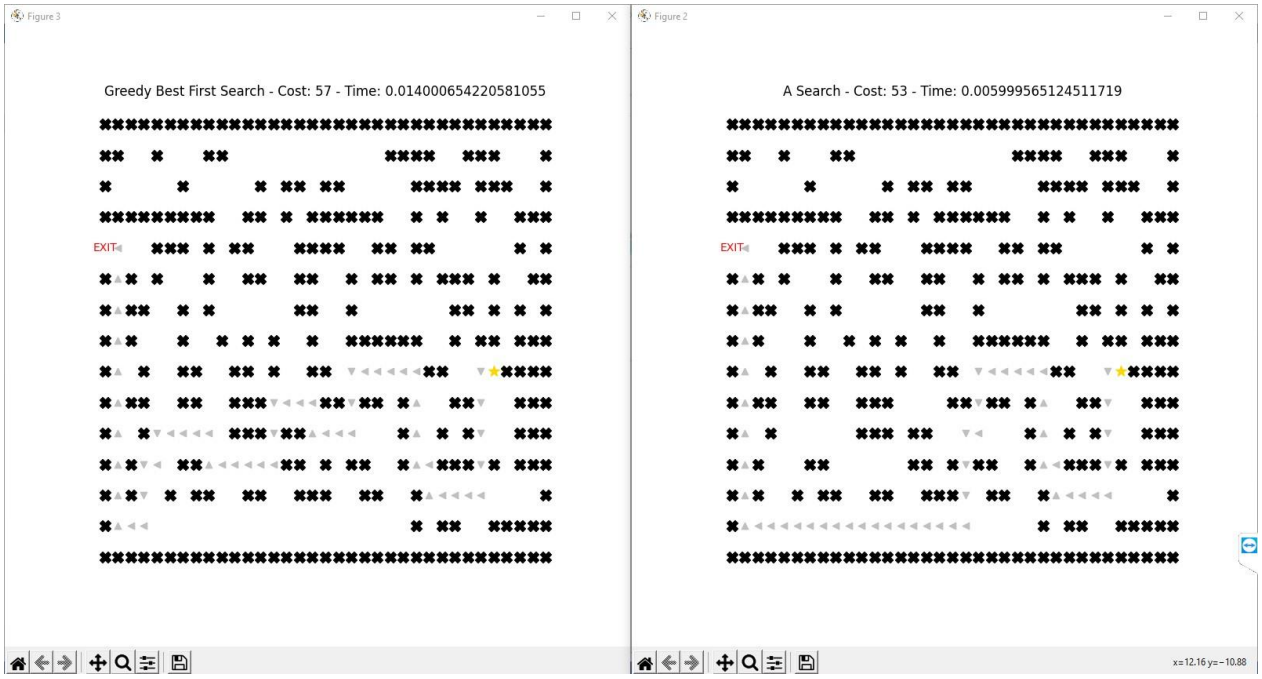
+ Hàm space: hàm tính khoảng cách Euclid từ 1 điểm đến đích.

+ Hàm space1: hàm tính chi phí ít nhất có thể để đi đến đích.

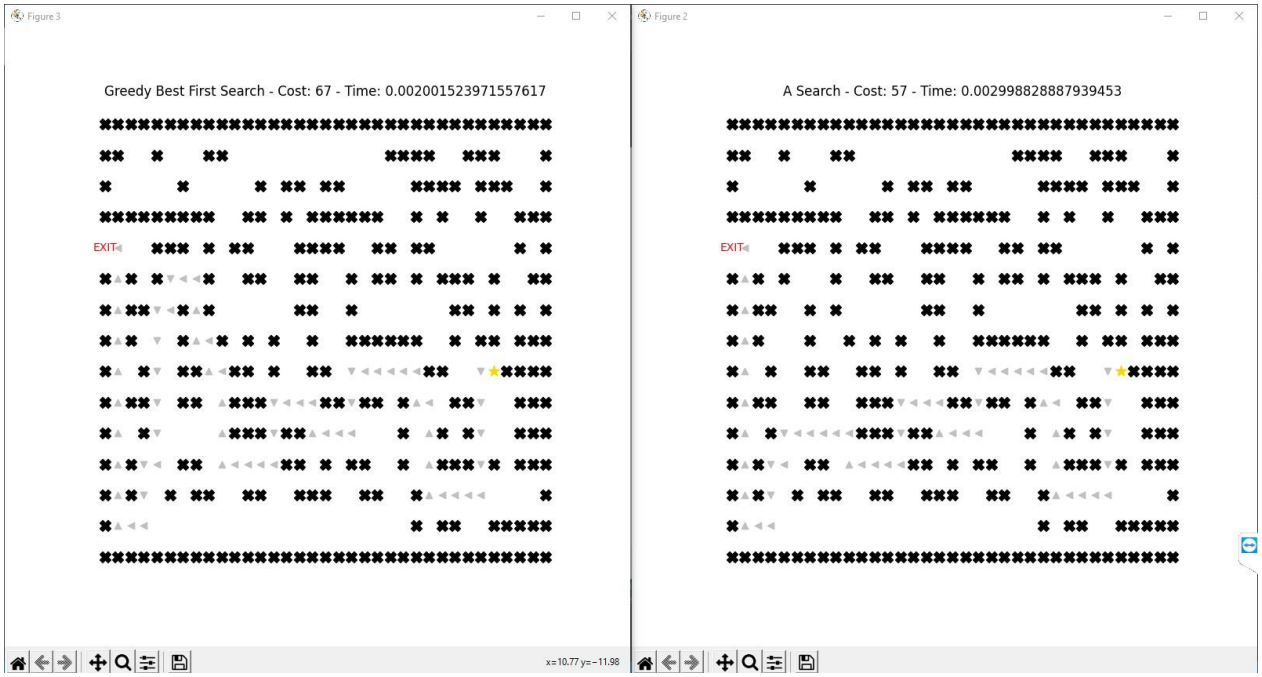
(Trong code đã sử dụng 2 hàm heuristic là space và space2) .

- Bản đồ 4:

+Space:

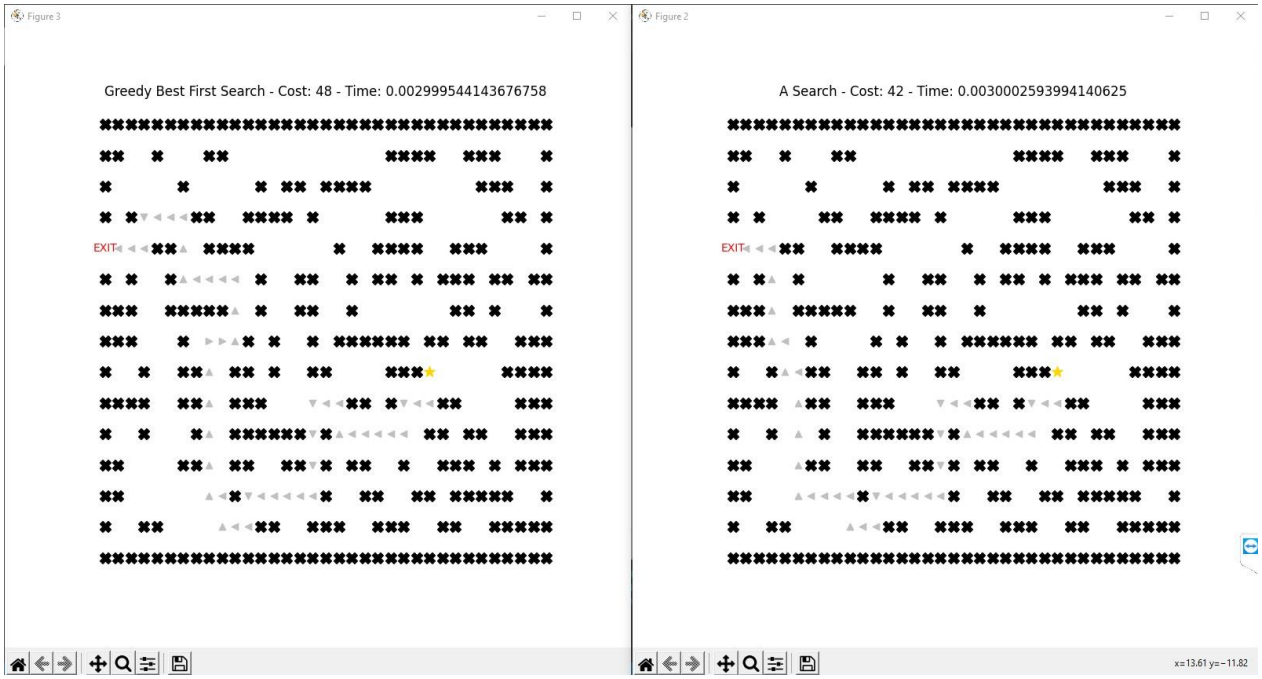


+Space2:

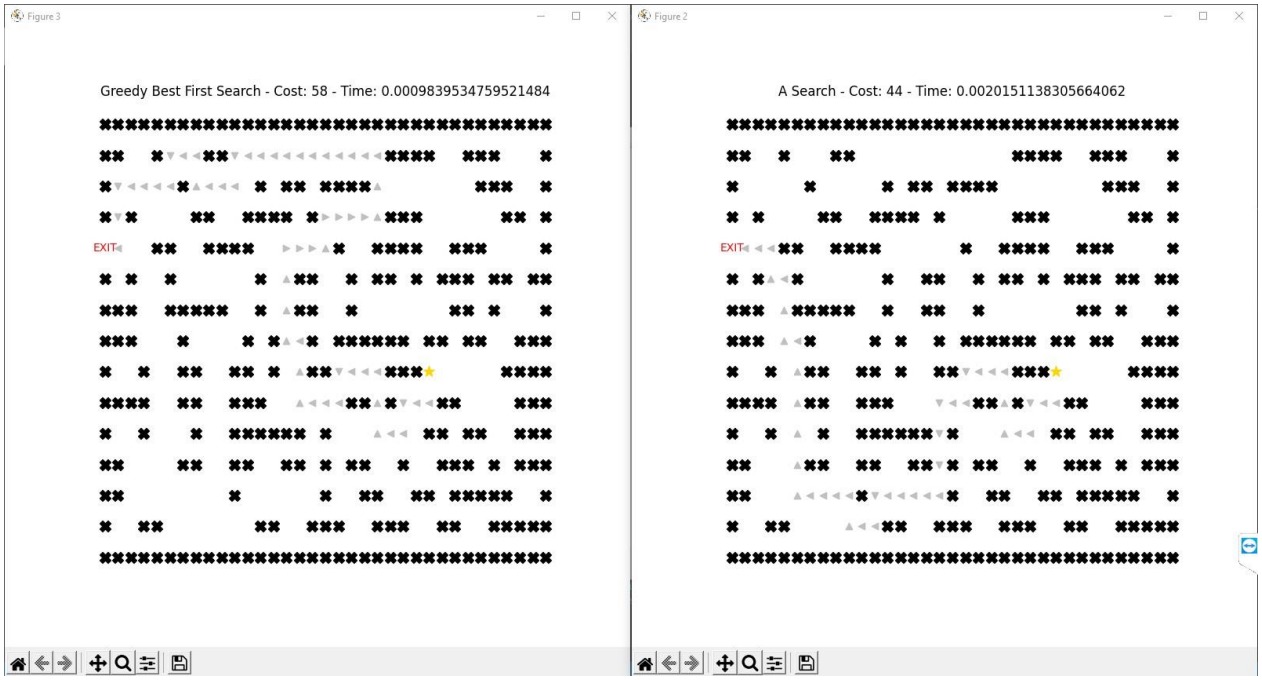


- Bản đồ 5:

+Space:



+Space2:



➔ Kết luận: Heuristic càng tốt thì ra kết quả càng tốt => Heuristic có vai trò rất lớn trong thuật toán tìm kiếm thông tin.

2) Trường hợp bản đồ có điểm thưởng:

*Tổng quát thuật toán:

- Kế thừa A* search ta sẽ dùng f(n) = g(n) + h(n)

  + g(n) là chi phí đi tới điểm.

  + h(n) là chi phí ước lượng.

- Khi duyệt qua một điểm sẽ đánh dấu là visited và cập nhật chi phí, nếu duyệt qua được một điểm thưởng chương trình tự động set điểm thưởng là start và tiếp tục quá trình duyệt đồng thời lưu trữ quãng đường đã đi để copy cho những lần cập nhật khi gặp một điểm thưởng mới.

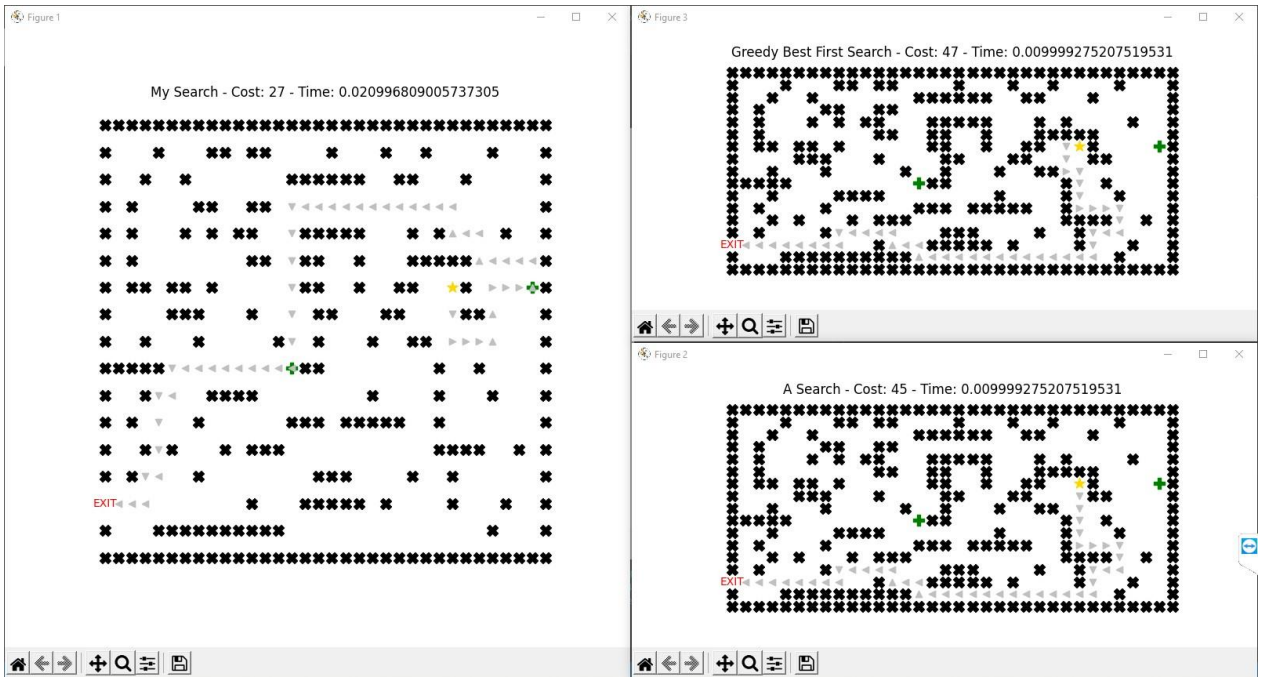
- Xử lí một vài trường hợp đặc biệt và chuẩn hóa dữ liệu cho thuật toán thêm thông minh và tránh trường hợp vòng lặp âm, tránh trường hợp điểm thưởng không lời để di chuyển tới.

- Hạn chế: Thuật toán chưa tối ưu 100% nhưng có thể xử lí nhìn chung hoàn hảo các trường hợp thường thấy, không đánh đố người chơi.

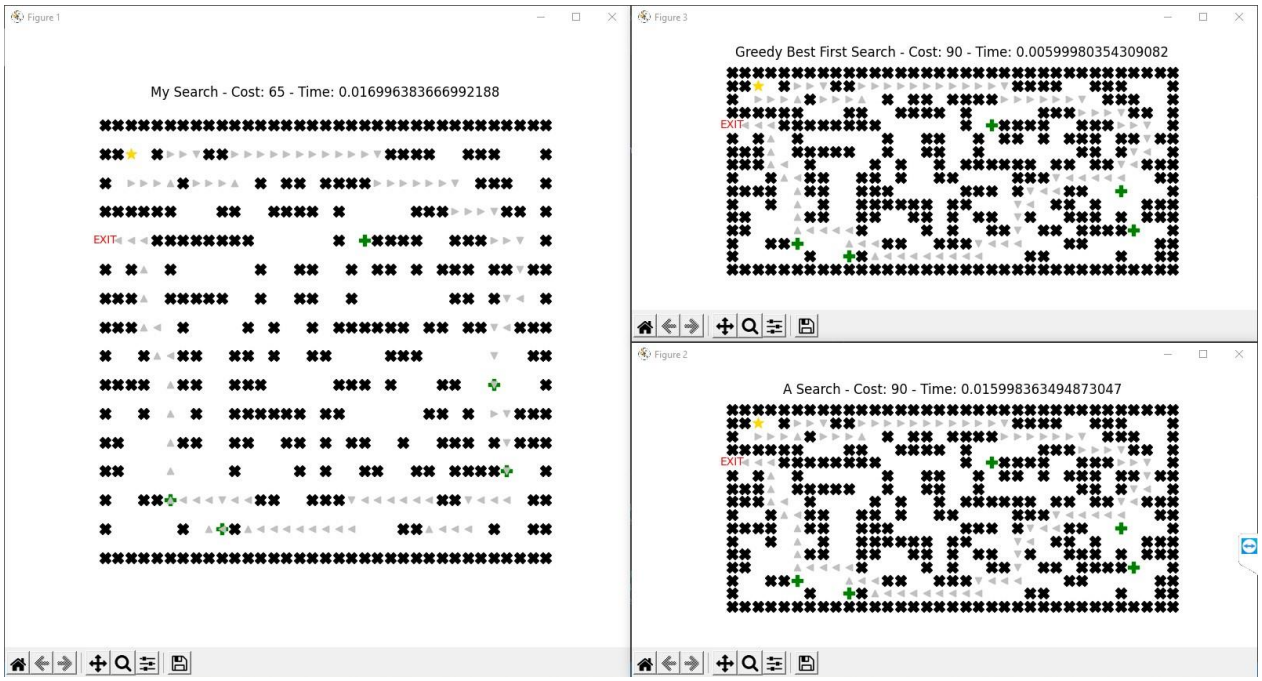
- Kết quả: Chi phí đi đến đích được giảm hơn rất nhiều so với A* search và GBFS.

Kết quả: (so sánh với A và GBFS)

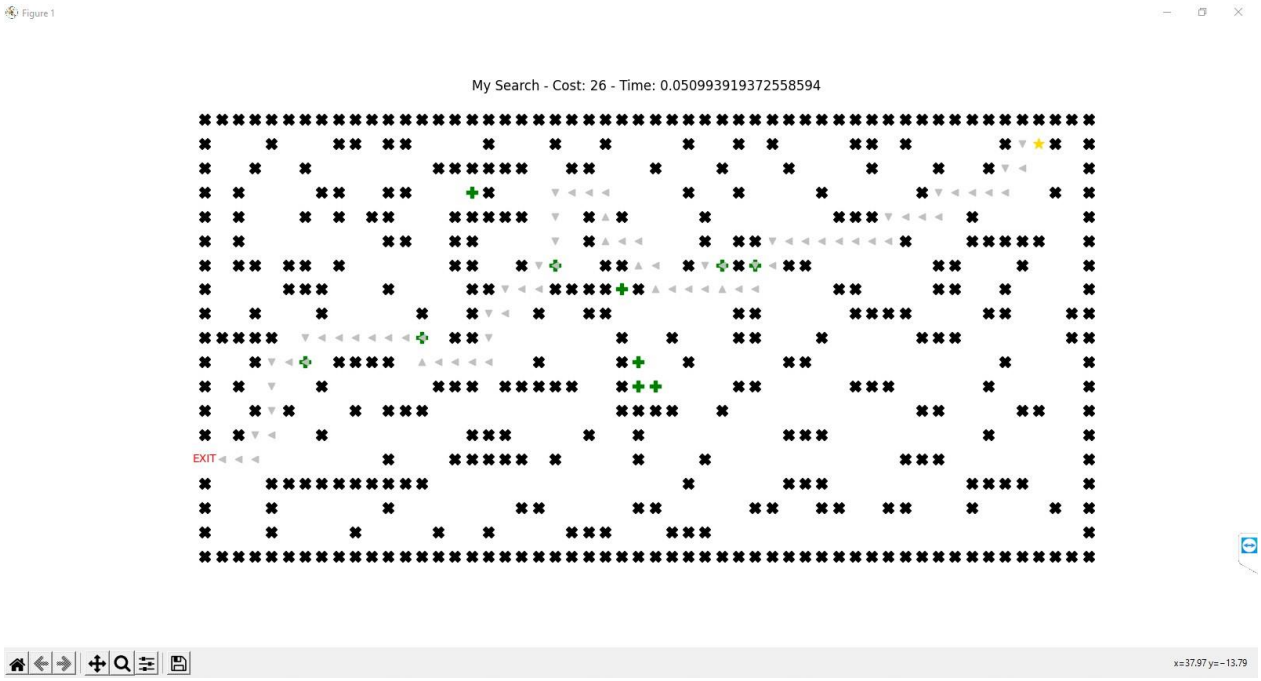
a) Bản đồ có 2 điểm thưởng:

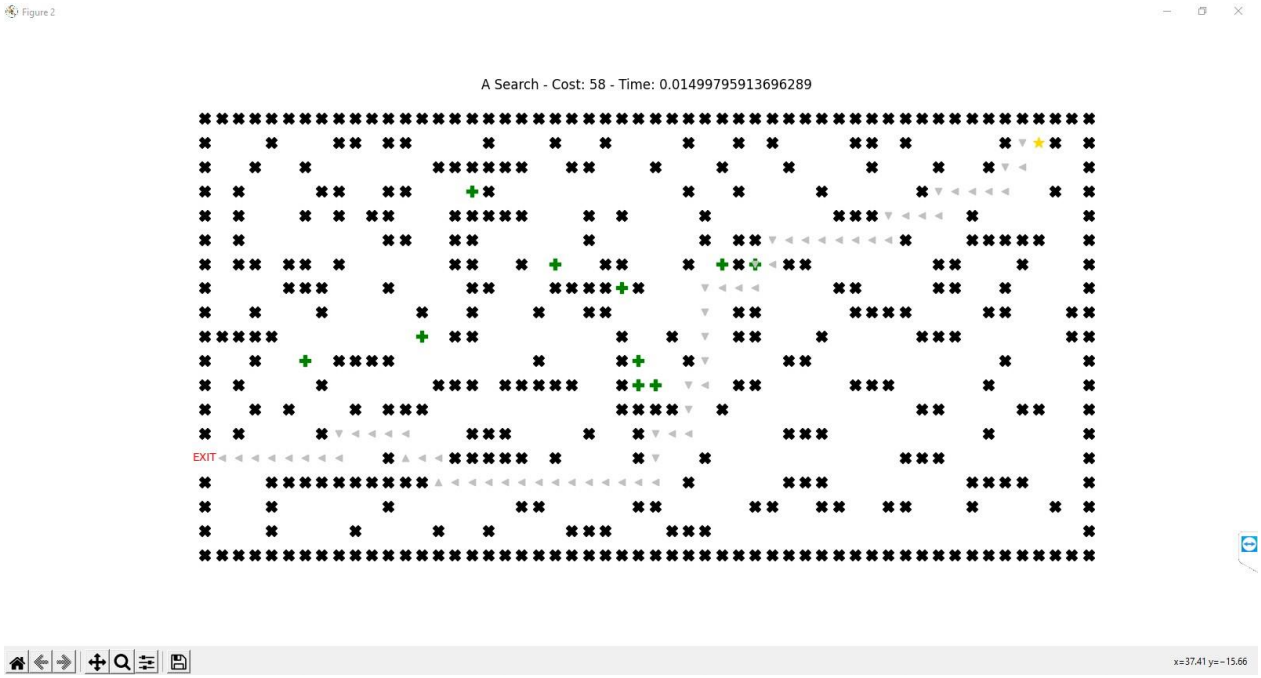


b) Bản đồ có 5 điểm thưởng:



c) Bản đồ có 10 điểm thưởng:





III. Tài liệu tham khảo:

- *Giáo trình Cơ Sở Trí Tuệ nhân tạo của trường đại học Khoa Học Tự Nhiên*

- ***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/breadth-first-search-or-bfs-for-a-graph/***

----------*Hết*----------