ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN MÔN HỌC

CƠ SỞ TRÍ TUỆ NHÂN TẠO

CHỦ ĐỀ: THUẬT TOÁN TÌM KIẾM

Người thực hiện:

Họ và tên: Trần Bảo Tín

MSSV: 19120684

Thành phố Hồ Chí Minh, tháng 11 năm 2021

MỤC LỤC

I. Không có điểm thưởng.

i. Mê cung thứ nhất.

1. Thuật toán DFS.

Đường đi: (8, 15), (9, 15), (9, 14), (8, 14), (7, 14), (7, 15), (6, 15), (6, 14), (6, 13), (6, 12), (6, 11), (6, 10), (7, 10), (8, 10), (8, 9), (7, 9), (6, 9), (5, 9), (5, 8), (5, 7), (5, 6), (5, 5), (5, 4), (5, 3), (5, 2), (5, 1), (4, 1), (3, 1), (2, 1), (2, 2), (2, 3), (1, 3), (1, 2), (1, 1), (1 , 0)

Sở dĩ đường đi lòng vòng như thế là do trong lúc tìm phần tử kề cận em đã xét theo thứ tự RIGHT -> DOWN -> LEFT -> UP nên khi có đường đi ở dưới thì thuật toán sẽ xét xuống dưới trước nếu có đường trở ra thì nó sẽ đi tiếp, nếu không thì sẽ không đi được, tùy theo bản đồ và cách dùng thuật toán thì đường đi này không phải đường ngắn nhất.

2. Thuật toán BFS

Đường đi: (8, 15), (8, 14), (7, 14), (6, 14), (6, 13), (6, 12), (6, 11), (6, 10), (6, 9), (5, 9), (5, 8), (5, 7), (5, 6), (5, 5), (5, 4), (5, 3), (5, 2), (5, 1), (4, 1), (3, 1), (2, 1), (1, 1) , (1, 0)

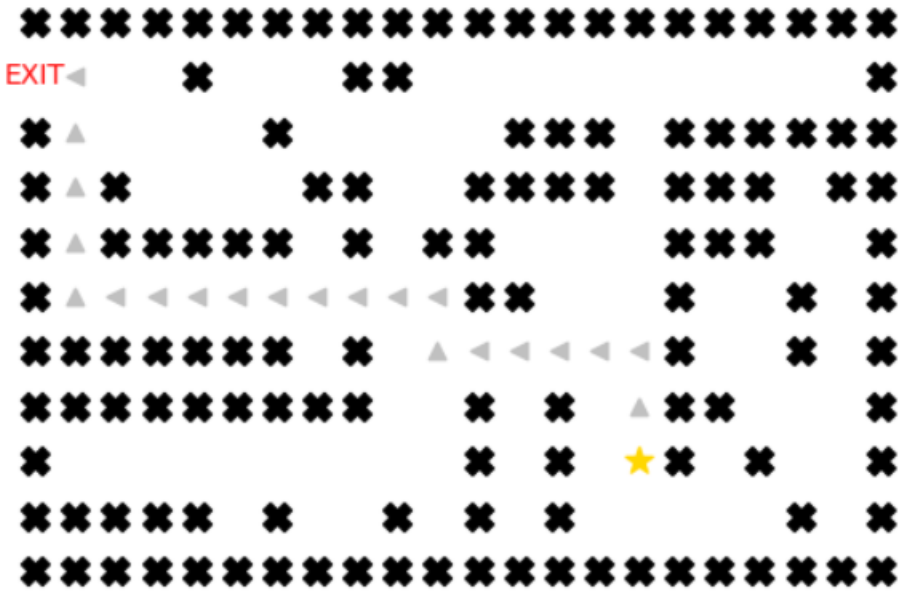
Còn BFS, đường đi đã rút ngắn hơn nhiều vì thuật toán này duyệt theo chiều rộng nên chỉ cần có đường đi thì thuật toán này sẽ cho ra đường đi ngắn nhất tới đích

3. Thuật toán GBFS.

Đường đi: (8, 15), (7, 15), (6, 15), (5, 15), (4, 15), (3, 15), (2, 15), (1, 15), (1, 14), (1, 13), (1, 12), (1, 11), (1, 10), (2, 10), (2, 9), (2, 8), (2, 7), (1, 7), (1, 6), (1, 5), (2, 5), (2, 4), (2, 3), (1, 3), (1, 2), (1, 1), (1, 0)

Ở thuật toán này, hàm heuristic là tổng số bước cần phải đi để về đích mà không có vật cản, nhưng vì nó chỉ xét theo hàm heuristic nên nó không quan tâm chi phí con đường đã đi nên sẽ tồn tại những bước đi dư thừa.

4. Thuật toán A*.



Đường đi: (8, 15), (7, 15), (6, 15), (6, 14), (6, 13), (6, 12), (6, 11), (6, 10), (5, 10), (5, 9), (5, 8), (5, 7), (5, 6), (5, 5), (5, 4), (5, 3), (5, 2), (5, 1), (4, 1), (3, 1), (2, 1), (1, 1), (1 , 0)

Ở thuật toán này, đường đi đã đỡ lòng vòng hơn thuật toán GBFS vì có sự cải thiện ở thuật toán khi xét thêm cả tổng số bước đã đi chứ không chỉ xét mỗi hàm heuristic. Chính vì vậy thuật toán này sẽ cho ra kết quả tốt hơn thuật toán GBFS.

ii. Mê cung thứ 2.

1. Thuật toán DFS.

Đường đi: (8, 21), (8, 22), (8, 23), (7, 23), (7, 24), (6, 24), (5, 24), (4, 24), (3, 24), (2, 24), (2, 23), (1, 23), (1, 22), (1, 21), (1, 20), (1, 19), (1, 18), (1, 17), (1, 16), (1, 15), (2 , 15), (3, 15), (4, 15), (5, 15), (6, 15), (7, 15), (7, 14), (6, 14), (6, 13), (6, 12), (6, 11), (

6, 10), (6, 9), (5, 9), (4, 9), (3, 9), (3, 10), (2, 10), (2, 9), (2, 8), (2, 7), (1, 7), (1, 6), (1, 5), (2, 5), (2, 4), (2, 3), (2, 2), (2, 1), (1, 1), (1, 0)

Sở dĩ đường đi lòng vòng như thế là do trong lúc tìm phần tử kề cận em đã xét theo thứ tự RIGHT -> DOWN -> LEFT -> UP nên khi có đường đi ở dưới thì thuật toán sẽ xét xuống dưới trước nếu có đường trở ra thì nó sẽ đi tiếp, nếu không thì sẽ không đi được, tùy theo bản đồ và cách dùng thuật toán thì đường đi này không phải đường ngắn nhất.

2. Thuật toán BFS

Đường đi: (8, 21), (8, 22), (8, 23), (7, 23), (7, 24), (6, 24), (5, 24), (4, 24), (3, 24), (2, 24), (2, 23), (1, 23), (1, 22), (1, 21), (1, 20), (1, 19), (1, 18), (1, 17), (1, 16), (1, 15), (1 , 14), (1, 13), (1, 12), (1, 11), (2, 11), (2, 10), (2, 9), (2, 8), (2, 7), (1, 7), (1, 6), (1, 5), (2, 5), (2, 4), (2, 3), (2, 2), (2, 1), (1, 1), (1, 0)
Còn BFS, đường đi đã rút ngắn hơn nhiều vì thuật toán này duyệt theo chiều rộng nên chỉ cần có đường đi thì thuật toán này sẽ cho ra đường đi ngắn nhất tới đích

3. Thuật toán GBFS.

Đường đi: (8, 21), (8, 20), (9, 20), (9, 19), (9, 18), (9, 17), (9, 16), (9, 15), (9, 14), (9, 13), (9, 12), (9, 11), (9, 10), (8, 10), (8, 9), (8, 8), (8, 7), (8, 6), (8, 5), (8, 4), (7, 4), (6, 4), (5, 4), (5, 5), (5, 6), (5, 7), (5, 8), (5, 9), (4, 9), (3, 9), (2, 9), (2, 8), (2, 7), (1, 7), (1 , 6), (1, 5), (2, 5), (2, 4), (2, 3), (1, 3), (1, 2), (1, 1), (1, 0)

Ở thuật toán này, hàm heuristic là tổng số bước cần phải đi để về đích mà không có vật cản, nhưng vì nó chỉ xét theo hàm heuristic nên nó không quan tâm chi phí con đường đã đi nên sẽ tồn tại những bước đi dư thừa.

4. Thuật toán A*.

Đường đi: (8, 21), (8, 22), (8, 23), (7, 23), (7, 24), (6, 24), (5, 24), (4, 24), (3, 24), (2, 24), (2, 23), (1, 23), (1, 22), (1, 21), (1, 20), (1, 19), (1, 18), (1, 17), (1, 16), (1, 15), (1 , 14), (1, 13), (1, 12), (1, 11), (1, 10), (2, 10), (2, 9), (2, 8), (2, 7), (1, 7), (1, 6), (1, 5), (2, 5), (2, 4), (2, 3), (1, 3), (1, 2), (1, 1), (1, 0)

Ở thuật toán này, đường đi đã đỡ lòng vòng hơn thuật toán GBFS vì có sự cải thiện ở thuật toán khi xét thêm cả tổng số bước đã đi chứ không chỉ xét mỗi hàm heuristic. Chính vì vậy thuật toán này sẽ cho ra kết quả tốt hơn thuật toán GBFS.

iii. Mê cung thứ 3.

1. Thuật toán DFS.

Đường đi: (13, 24), (13, 23), (13, 22), (12, 22), (12, 23), (11, 23), (10, 23), (10, 22), (9, 22), (9, 23), (8, 23), (8, 22), (8, 21), (8, 20), (7, 20), (7, 19), (7, 18), (6, 18), (6, 17), (5, 17), (5, 18), (4, 18), (4, 17), (4, 16), (4, 15), (5, 15), (6, 15), (6, 14), (7, 14), (7, 13) , (6, 13), (6, 12), (6, 11), (7, 11), (8, 11), (8, 10), (7, 10), (6, 10), (6, 9), (5, 9), (5, 8), (

5, 7), (5, 6), (5, 5), (5, 4), (5, 3), (5, 2), (4, 2), (3, 2), (3, 1), (2, 1), (2, 2), (2, 3), (1, 3), (1, 2), (1, 1), (1, 0)

Sở dĩ đường đi lòng vòng như thế là do trong lúc tìm phần tử kề cận em đã xét theo thứ tự RIGHT -> DOWN -> LEFT -> UP nên khi có đường đi ở dưới thì thuật toán sẽ xét xuống dưới trước nếu có đường trở ra thì nó sẽ đi tiếp, nếu không thì sẽ không đi được, tùy theo bản đồ và cách dùng thuật toán thì đường đi này không phải đường ngắn nhất.

2. Thuật toán BFS

Đường đi: (13, 24), (13, 23), (13, 22), (12, 22), (12, 21), (12, 20), (13, 20), (13, 19), (13, 18), (13, 17), (13, 16), (13, 15), (13, 14), (13, 13), (13, 12), (13, 11), (13, 10), (12, 10), (12, 9), (11, 9), (10, 9), (9, 9), (9, 10), (8, 10), (7, 10), (6, 10), (6, 9), (5, 9), (5, 8) , (5, 7), (5, 6), (5, 5), (5, 4), (5, 3), (5, 2), (4, 2), (3, 2), (3, 1), (2, 1), (1, 1), (1, 0)

Còn BFS, đường đi đã rút ngắn hơn nhiều vì thuật toán này duyệt theo chiều rộng nên chỉ cần có đường đi thì thuật toán này sẽ cho ra đường đi ngắn nhất tới đích

3. Thuật toán GBFS.

Đường đi: (13, 24), (12, 24), (12, 23), (11, 23), (10, 23), (9, 23), (8, 23), (8, 22), (8, 21), (8, 20), (7, 20), (6, 20), (5, 20), (4, 20), (4, 19), (4, 18), (3, 18), (3, 17), (2, 17), (2, 16), (2, 15), (1, 15), (1, 14), (1, 13), (1, 12), (1, 11), (2, 11), (2, 10), (2, 9), (2, 8), (2, 7), (1, 7), (1, 6), (1, 5), (2, 5), (2, 4), (2, 3), (1, 3), (1, 2), (1, 1), (1, 0)

Ở thuật toán này, hàm heuristic là tổng số bước cần phải đi để về đích mà không có vật cản, nhưng vì nó chỉ xét theo hàm heuristic nên nó không quan tâm chi phí con đường đã đi nên sẽ tồn tại những bước đi dư thừa. Tuy nhiên, bản đồ này vô tình được sắp sao cho nó không đi những bước dư thừa

4. Thuật toán A*.

Đường đi: (13, 24), (12, 24), (12, 23), (11, 23), (10, 23), (9, 23), (8, 23), (8, 22), (8, 21), (8, 20), (7, 20), (6, 20), (5, 20), (4, 20), (4, 19), (4, 18), (3, 18), (3, 17), (2, 17), (2, 16), (2, 15), (1, 15), (1, 14), (1, 13), (1, 12), (1, 11), (2, 11), (2, 10), (2, 9), (2, 8), (2, 7), (1, 7), (1, 6), (1, 5), (2, 5), (2, 4), (2, 3), (1, 3), (1, 2), (1, 1), (1, 0)

Ở thuật toán này, đường đi đã đỡ lòng vòng hơn thuật toán GBFS vì có sự cải thiện ở thuật toán khi xét thêm cả tổng số bước đã đi chứ không chỉ xét mỗi hàm heuristic. Chính vì vậy thuật toán này sẽ cho ra kết quả tốt hơn thuật toán GBFS.

iv. Mê cung thứ 4.

1. Thuật toán DFS.

Đường đi: (16, 24), (16, 23), (16, 22), (16, 21), (16, 20), (16, 19), (16, 18), (16, 17), (16, 16), (15, 16), (15, 15), (14, 15), (14, 16), (13, 16), (12, 16), (11, 16), (11, 17), (11, 18), (10, 18), (10, 19), (10, 20), (11, 20), (11, 21), (12, 21), (12, 22), (13, 22), (13, 23) , (14, 23), (15, 23), (15, 24), (15, 25), (14, 25), (14, 26), (14, 27), (13, 27), (12, 27), (1

2, 26), (12, 25), (11, 25), (11, 24), (10, 24), (9, 24), (8, 24), (7, 24), (6, 24), (5, 24), (5 , 23), (4, 23), (4, 22), (4, 21), (4, 20), (4, 19), (4, 18), (4, 17), (5, 17), (5, 16), (5, 15), (6, 15), (7, 15), (8, 15), (9, 15), (9, 14), (10, 14), (10, 13), (9, 13), (9, 12), (9, 11), (9, 1 0), (9, 9), (8, 9), (8, 10), (7, 10), (6, 10), (6, 9), (5, 9), (5, 10), (5, 11), (4, 11), (4, 12), (4, 13), (4, 14), (4, 15), (4, 16), (3, 16), (3, 15), (3, 14), (2, 14), (2, 13), (2, 12), (2, 11) , (1, 11), (1, 10), (1, 9), (1, 8), (2, 8), (2, 7), (2, 6), (2, 5), (2, 4), (2, 3), (1, 3), (1, 2), (1 , 1), (1, 0)

Sở dĩ đường đi lòng vòng như thế là do trong lúc tìm phần tử kề cận em đã xét theo thứ tự RIGHT -> DOWN -> LEFT -> UP nên khi có đường đi ở dưới thì thuật toán sẽ xét xuống dưới trước nếu có đường trở ra thì nó sẽ đi tiếp, nếu không thì sẽ không đi được, tùy theo bản đồ và cách dùng thuật toán thì đường đi này không phải đường ngắn nhất.

2. Thuật toán BFS

Đường đi: (16, 24), (16, 23), (16, 22), (16, 21), (16, 20), (16, 19), (16, 18), (16, 17), (16, 16), (15, 16), (14, 16), (13, 16), (12, 16), (11, 16), (11, 15), (11, 14), (10, 14), (10, 13), (9, 13), (9, 12), (9, 11), (9, 10), (8, 10), (7, 10), (6, 10), (5, 10), (5, 11), (4, 11), (3 , 11), (2, 11), (1, 11), (1, 10), (1, 9), (1, 8), (2, 8), (2, 7), (2, 6), (2, 5), (2, 4), (2, 3), (1, 3), (1, 2), (1, 1), (1, 0)

Còn BFS, đường đi đã rút ngắn hơn nhiều vì thuật toán này duyệt theo chiều rộng nên chỉ cần có đường đi thì thuật toán này sẽ cho ra đường đi ngắn nhất tới đích

3. Thuật toán GBFS.

Đường đi: (16, 24), (15, 24), (15, 23), (14, 23), (13, 23), (13, 22), (12, 22), (12, 21), (11, 21), (11, 20), (10, 20), (10, 19), (10, 18), (11, 18), (11, 17), (11, 16), (11, 15), (11, 14), (10, 14), (9, 14), (9, 13), (9, 12), (9, 11), (9, 10), (8, 10), (7, 10), (6, 10), (5, 10), (4, 10), (4, 11), (3, 11), (2, 11), (1, 11), (1, 10), (1, 9), (1, 8), (2, 8), (2, 7), (2, 6), (2, 5) , (2, 4), (2, 3), (1, 3), (1, 2), (1, 1), (1, 0)

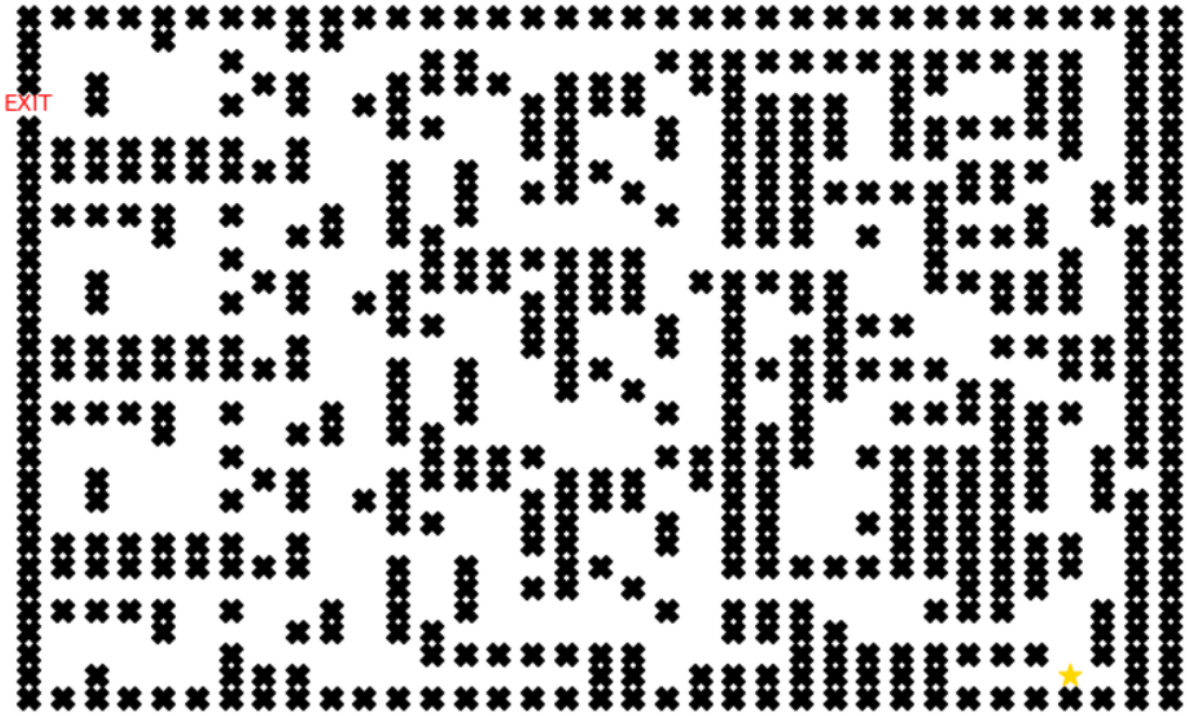
Ở thuật toán này, hàm heuristic là tổng số bước cần phải đi để về đích mà không có vật cản, nhưng vì nó chỉ xét theo hàm heuristic nên nó không quan tâm chi phí con đường đã đi nên sẽ tồn tại những bước đi dư thừa.

4. Thuật toán A*.

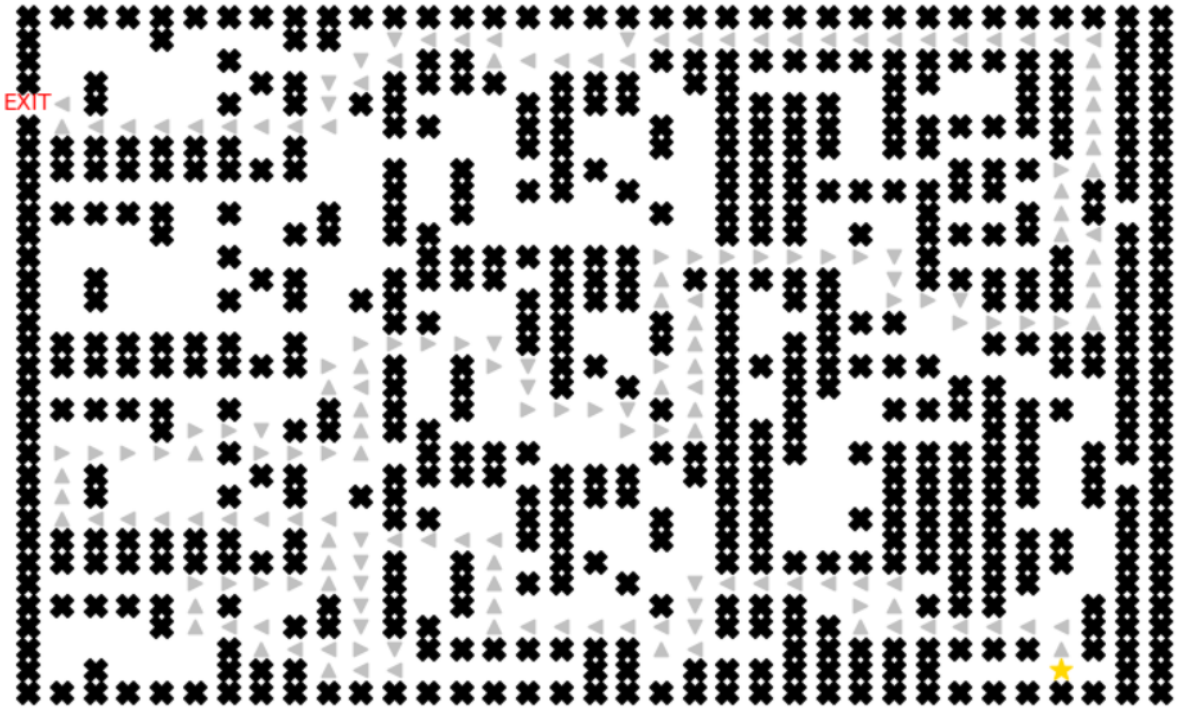
Đường đi: (16, 24), (16, 23), (16, 22), (16, 21), (16, 20), (16, 19), (16, 18), (16, 17), (16, 16), (15, 16), (14, 16), (13, 16), (12, 16), (11, 16), (11, 15), (11, 14), (10, 14), (9, 14), (9, 13), (9, 12), (9, 11), (9, 10), (8, 10), (7, 10), (6, 10), (5, 10), (4, 10), (4, 11), (3 , 11), (2, 11), (1, 11), (1, 10), (1, 9), (1, 8), (2, 8), (2, 7), (2, 6), (2, 5), (2, 4), (2, 3), (1, 3), (1, 2), (1, 1), (1, 0)

Ở thuật toán này, đường đi đã đỡ lòng vòng hơn thuật toán GBFS vì có sự cải thiện ở thuật toán khi xét thêm cả tổng số bước đã đi chứ không chỉ xét mỗi hàm heuristic. Chính vì vậy thuật toán này sẽ cho ra kết quả tốt hơn thuật toán GBFS.

v. Mê cung thứ 5.



1. Thuật toán DFS.

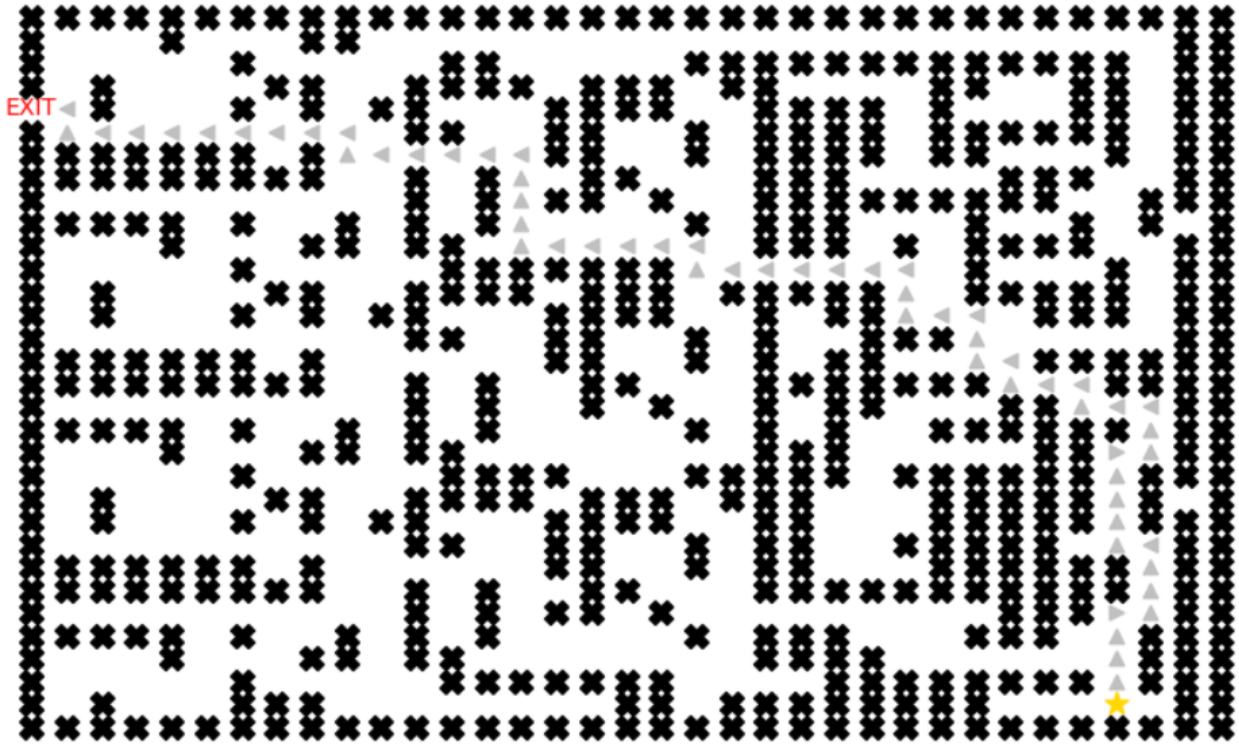


Đường đi: (30, 31), (29, 31), (28, 31), (28, 30), (28, 29), (28, 28), (28, 27), (28, 26), (28, 25), (27, 25), (27, 26), (26, 26), (26, 25), (26, 24), (26, 23), (26, 22), (26, 21), (26, 20), (27, 20), (28, 20), (29, 20), (29, 19), (28, 19), (28, 18), (28, 17), (28, 16), (28, 15) , (28, 14), (27, 14), (26, 14), (25, 14), (24, 14), (24, 13), (24, 12), (24, 11), (24, 10), (2 5, 10), (26, 10), (27, 10), (28, 10), (29, 10), (29, 11), (30, 11), (30, 10), (30, 9), (29, 9) , (29, 8), (29, 7), (28, 7), (28, 6), (28, 5), (27, 5), (26, 5), (26, 6), (26, 7), (26, 8), (26, 9), (25, 9), (24, 9), (23, 9), (23, 8), (23, 7), (23, 6), (23, 5), (23, 4), (23, 3), (23, 2), (23 , 1), (22, 1), (21, 1), (20, 1), (20, 2), (20, 3), (20, 4), (20, 5), (19, 5), (19, 6), (19, 7), (2 0, 7), (20, 8), (20, 9), (20, 10), (19, 10), (18, 10), (17, 10), (17, 9), (16, 9), (16, 10), (1 5, 10), (15, 11), (15, 12), (15, 13), (15, 14), (16, 14), (16, 15), (17, 15), (18, 15), (18, 16), (18, 17), (18, 18), (19, 18), (19, 19), (19, 20), (18, 20), (17, 20), (17, 19), (16, 19) , (16, 20), (15, 20), (14, 20), (13, 20), (13, 19), (12, 19), (11, 19), (11, 20), (11, 21), (1 1, 22), (11, 23), (11, 24), (11, 25), (11, 26), (12, 26), (13, 26), (13, 27), (13, 28), (14, 28), (14, 29), (14, 30), (14, 31), (14, 32), (13, 32), (12, 32), (11, 32), (10, 32), (10, 31) , (9, 31), (8, 31), (7, 31), (7, 32), (6, 32), (5, 32), (4, 32), (3, 32), (2, 32), (1, 32), (1, 3 1), (1, 30), (1, 29), (1, 28), (1, 27), (1, 26), (1, 25), (1, 24), (1, 23), (1, 22), (1, 21), (1, 20), (1, 19), (1, 18), (2, 18), (2, 17), (2, 16), (2, 15), (2, 14), (1, 14), (1, 13), (1, 12), (1

, 11), (2, 11), (2, 10), (3, 10), (3, 9), (4, 9), (5, 9), (5, 8), (5, 7), (5, 6), (5, 5), (5, 4), (5, 3), (5, 2), (5, 1), (4, 1), (4, 0)

Sở dĩ đường đi lòng vòng như thế là do trong lúc tìm phần tử kề cận em đã xét theo thứ tự RIGHT -> DOWN -> LEFT -> UP nên khi có đường đi ở dưới thì thuật toán sẽ xét xuống dưới trước nếu có đường trở ra thì nó sẽ đi tiếp, nếu không thì sẽ không đi được, tùy theo bản đồ và cách dùng thuật toán thì đường đi này không phải đường ngắn nhất.

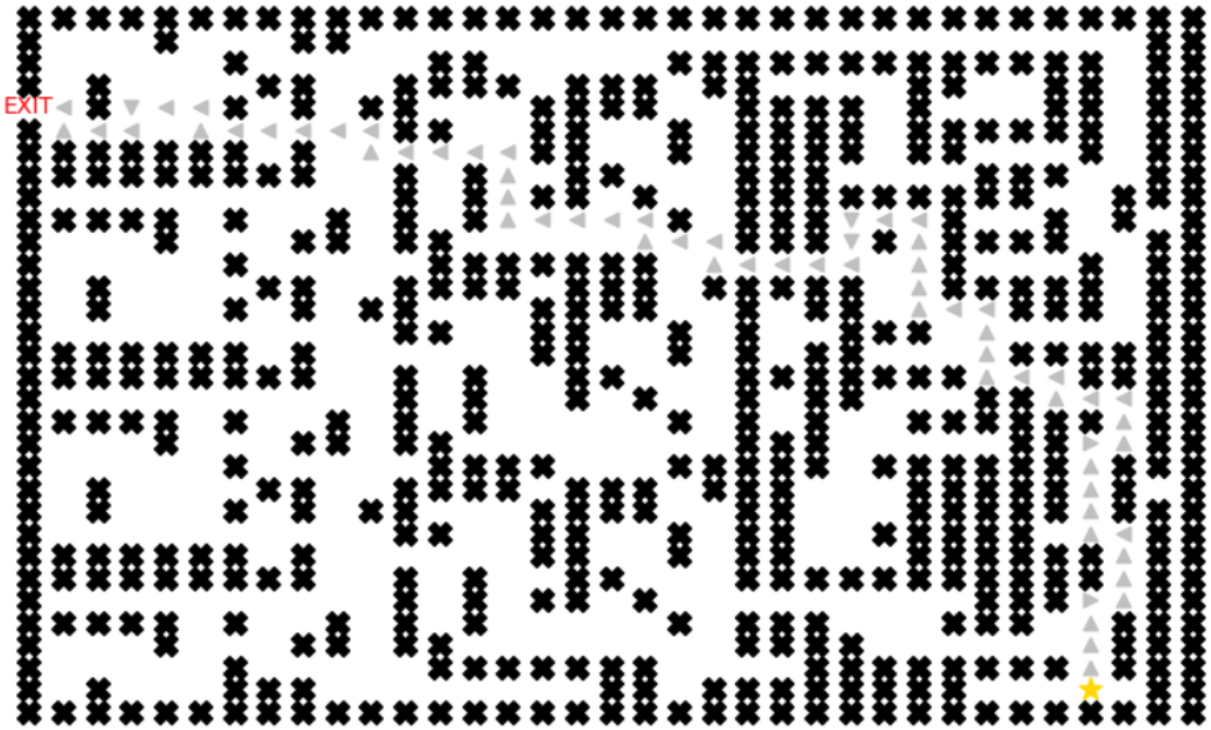
2. Thuật toán BFS



Đường đi: (30, 31), (29, 31), (28, 31), (27, 31), (26, 31), (26, 32), (25, 32), (24, 32), (23, 32), (23, 31), (22, 31), (21, 31), (20, 31), (19, 31), (19, 32), (18, 32), (17, 32), (17, 31), (17, 30), (16, 30), (16, 29), (16, 28), (15, 28), (15, 27), (14, 27), (13, 27), (13, 26), (13, 25), (12, 25), (11, 25), (11, 24), (11, 23), (11, 22), (11, 21), (11, 20), (11, 19), (10, 19), (10, 18), (10, 17), (10, 16), (10, 15), (10, 14), (9, 14), (8, 14), (7, 14), (6, 14), (6, 13), (6, 12), (6, 11), (6, 10), (6, 9), (5, 9), (5, 8), (5, 7), (5, 6), (5, 5), (5, 4), (5, 3), (5, 2), (5, 1), (4, 1), (4, 0)

Còn BFS, đường đi đã rút ngắn hơn nhiều vì thuật toán này duyệt theo chiều rộng nên chỉ cần có đường đi thì thuật toán này sẽ cho ra đường đi ngắn nhất tới đích

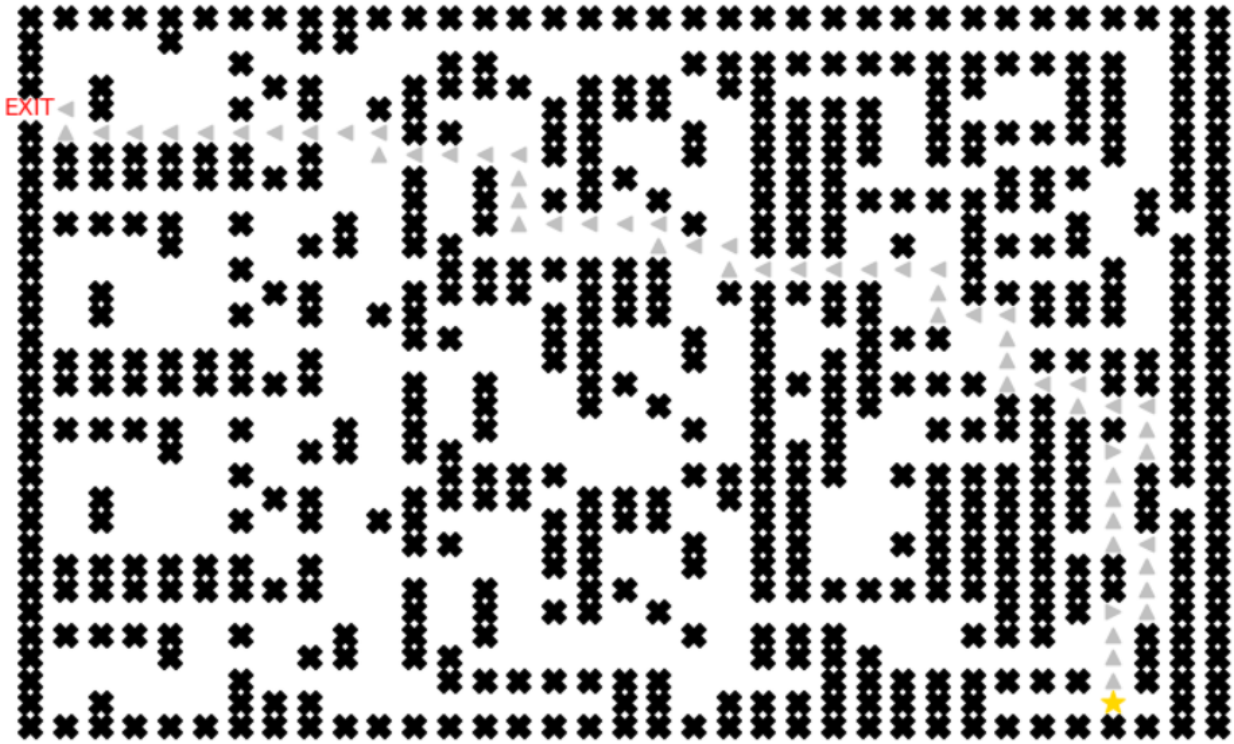
3. Thuật toán GBFS.



Đường đi: (30, 31), (29, 31), (28, 31), (27, 31), (26, 31), (26, 32), (25, 32), (24, 32), (23, 32), (23, 31), (22, 31), (21, 31), (20, 31), (19, 31), (19, 32), (18, 32), (17, 32), (17, 31), (17, 30), (16, 30), (16, 29), (16, 28), (15, 28), (14, 28), (13, 28), (13, 27), (13, 26) , (12, 26), (11, 26), (10, 26), (9, 26), (9, 25), (9, 24), (10, 24), (11, 24), (11, 23), (11, 2 2), (11, 21), (11, 20), (10, 20), (10, 19), (10, 18), (9, 18), (9, 17), (9, 16), (9, 15), (9, 1 4), (8, 14), (7, 14), (6, 14), (6, 13), (6, 12), (6, 11), (6, 10), (5, 10), (5, 9), (5, 8), (5, 7) , (5, 6), (5, 5), (4, 5), (4, 4), (4, 3), (5, 3), (5, 2), (5, 1), (4, 1), (4, 0)

Ở thuật toán này, hàm heuristic là tổng số bước cần phải đi để về đích mà không có vật cản, nhưng vì nó chỉ xét theo hàm heuristic nên nó không quan tâm chi phí con đường đã đi nên sẽ tồn tại những bước đi dư thừa.

4. Thuật toán A*.



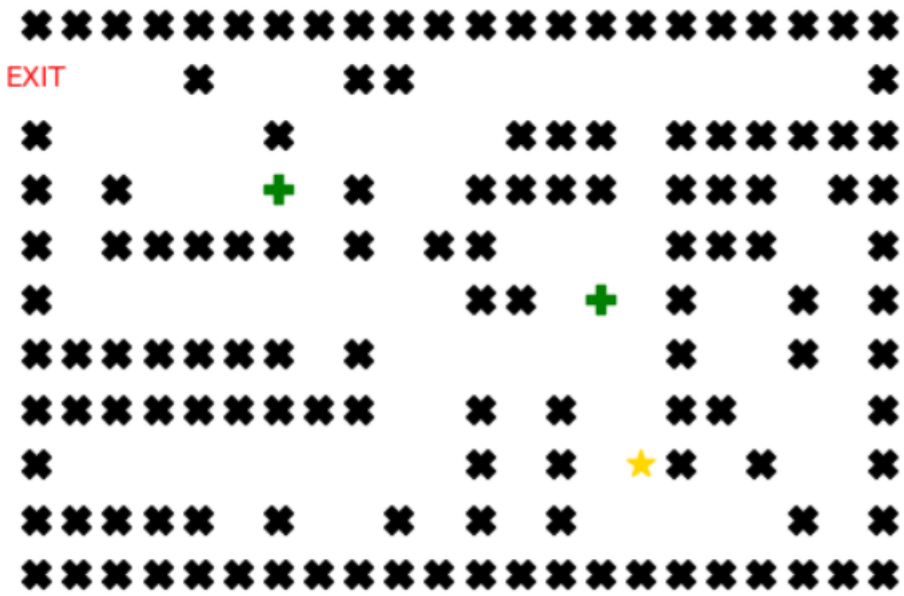
Đường đi: (30, 31), (29, 31), (28, 31), (27, 31), (26, 31), (26, 32), (25, 32), (24, 32), (23, 32), (23, 31), (22, 31), (21, 31), (20, 31), (19, 31), (19, 32), (18, 32), (17, 32), (17, 31), (17, 30), (16, 30), (16, 29), (16, 28), (15, 28), (14, 28), (13, 28), (13, 27), (13, 26) , (12, 26), (11, 26), (11, 25), (11, 24), (11, 23), (11, 22), (11, 21), (11, 20), (10, 20), (1 0, 19), (10, 18), (9, 18), (9, 17), (9, 16), (9, 15), (9, 14), (8, 14), (7, 14), (6, 14), (6, 13), (6, 12), (6, 11), (6, 10), (5, 10), (5, 9), (5, 8), (5, 7), (5, 6), (5, 5), (5, 4), (5, 3), (5, 2) , (5, 1), (4, 1), (4, 0)

Ở thuật toán này, đường đi đã đỡ lòng vòng hơn thuật toán GBFS vì có sự cải thiện ở thuật toán khi xét thêm cả tổng số bước đã đi chứ không chỉ xét mỗi hàm heuristic. Chính vì vậy thuật toán này sẽ cho ra kết quả tốt hơn thuật toán GBFS.

II. Điểm thưởng

Thuật toán tìm kiếm đường đi trong bản đồ điểm thưởng này là từ ban đầu sẽ xét ước tính đi tới điểm thưởng mà với giá trị điểm thưởng tại đó thì chi phí đường đi sẽ là ngắn nhất bằng A* sau đó sẽ bfs tới đó. Lặp liên tục từ điểm thưởng đó tới các điểm thưởng còn lại và xét giá trị ước tính tối ưu. Ta sẽ được đường đi tối ưu nhất.

i. Mê cung thứ nhất



2 điểm thưởng là:

- (3,6): -3,
- (5,14): -1.

1. Giải mê cung:

Đường đi: (8, 15), (8, 14), (7, 14), (6, 14), (6, 13), (6, 12), (6, 11), (6, 10), (6, 9), (5, 9), (5, 8), (5, 7), (4, 7), (3, 7), (3, 6), (3, 5), (3, 4), (3, 3), (2, 3), (2, 2), (2, 1), (1, 1), (1, 0)

Nhìn trực quan chúng ta thấy được điểm thưởng tại (5,14) quá nhỏ (-1) chính vì thế mà con đường ngắn nhất không thông qua nó vì như thế sẽ tốn chi phí về đường đi hơn.

ii. Mê cung thứ 2

5 điểm thưởng là:

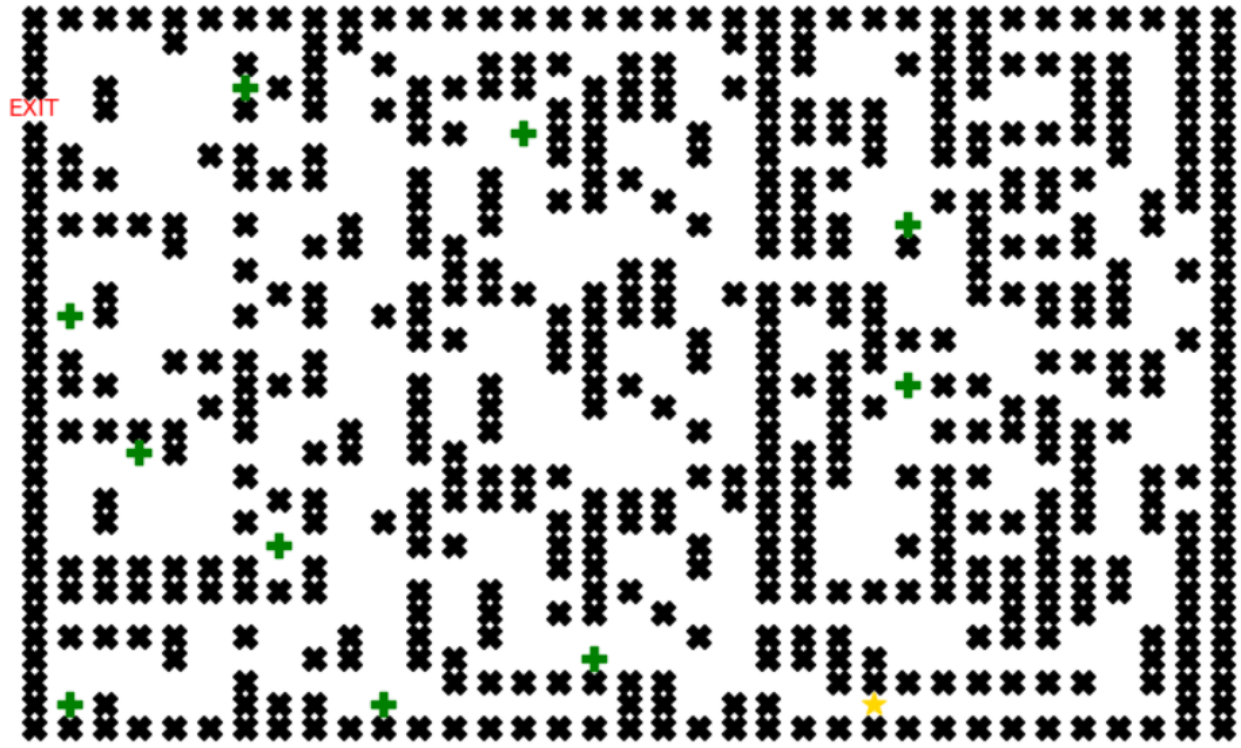
- (3,6): -3,
- (5,14): -5,
- (10,14): -4,
- (12,21): -2,
- (13,10): -2.

1. Giải mê cung:

Đường đi: (13, 19), (13, 18), (13, 17), (12, 17), (11, 17), (11, 16), (11, 15), (10, 15), (10, 14), (9, 14), (8, 14), (7, 14), (6, 14), (5, 14), (6, 14), (6, 13), (6, 12), (6, 11), (6, 10), (6, 9), (5, 9), (5, 8), (5, 7), (5, 6), (5, 5), (5, 4), (5, 3), (5, 2), (4, 2), (3, 2), (3, 1), (2, 1), (1, 1), (1, 0)
Chỗ bị lặp lại: (6, 14), (5, 14), (6, 14)

Chúng ta có thể thấy bản đồ không chọn điểm (13,10) và (12,21) vì giá trị điểm thưởng thấp, có đạt được thì nó cũng vẫn tốn chi phí hơn không đi qua nó. Nhưng ở vị trí (5,14) vì giá trị điểm thưởng này xứng đáng (Chi phí khi ghé điểm thưởng đó nhỏ hơn giá trị điểm thưởng nó mang lại) nên đã đi qua điểm đó rồi quay lại.

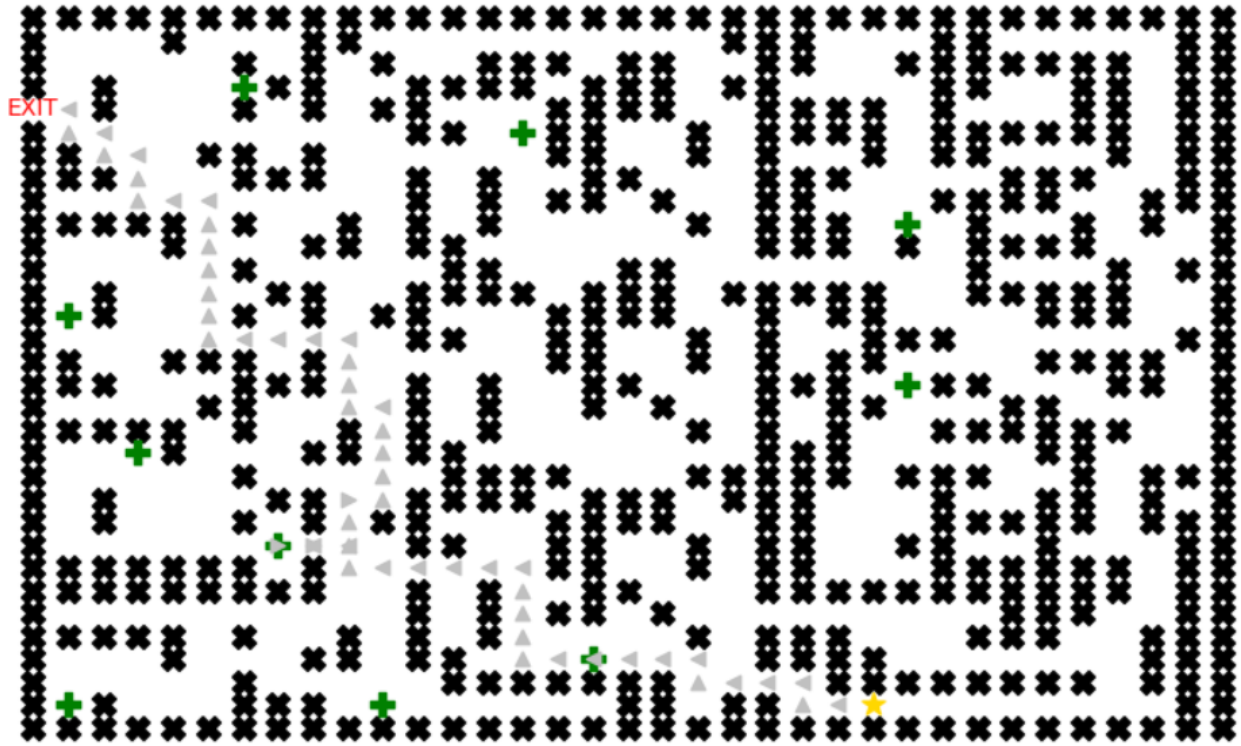
iii. Mê cung thứ 3



10 điểm thưởng:

- (3,6): -3,
- (5,14): -1,
- (9,25): -2,
- (13,1): -1,
- (16,25): -4,
- (19,3): -5,
- (23,7): -8,
- (28,16): -2,
- (30,1): -1,
- (30,10): -2

1. Giải mê cung:



Đường đi: (30, 24), (30, 23), (30, 22), (29, 22), (29, 21), (29, 20), (29, 19), (28, 19), (28, 18), (28, 17), (28, 16), (28, 15), (28, 14), (27, 14), (26, 14), (25, 14), (24, 14), (24, 13), (24, 12), (24, 11), (24, 10), (24, 9), (23, 9), (23, 8), (23, 7), (23, 8), (23, 9), (22, 9), (21, 9), (21, 10), (20, 10), (19, 10), (18, 10), (17, 10), (17, 9), (16, 9), (15, 9), (14, 9), (14, 8), (14, 7), (14, 6), (14, 5), (13, 5), (12, 5), (11, 5), (10, 5), (9, 5), (8, 5), (8, 4), (8, 3), (7, 3), (6, 3), (6, 2), (5, 2), (5, 1), (4, 1), (4, 0)

Chỗ bị lặp lại: (23, 9), (23, 8), (23, 7), (23, 8), (23, 9)

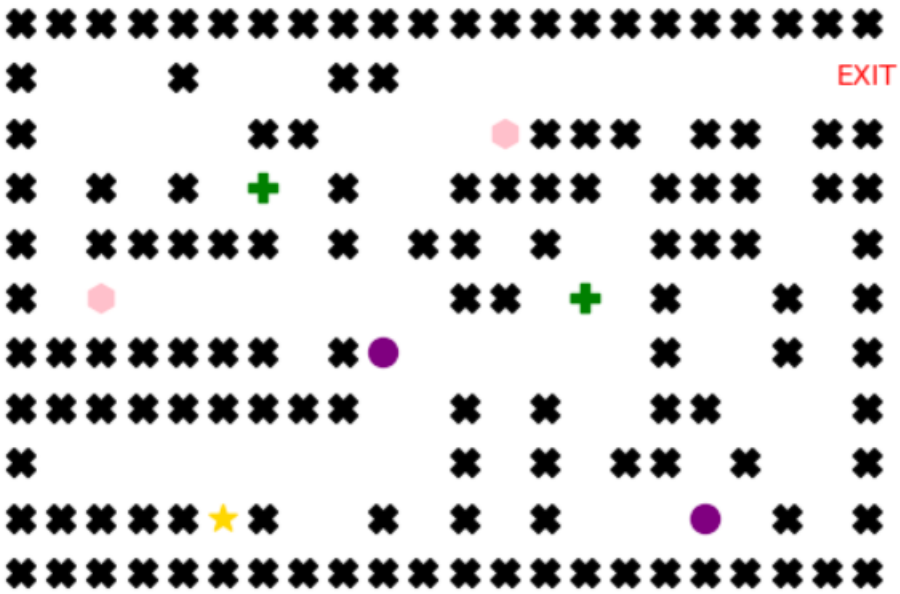
Chúng ta có thể thấy bản đồ hầu hết không chọn đi qua điểm thưởng vì giá trị điểm thưởng thấp hơn so với chi phí đi qua điểm đó. Tuy nhiên với những điểm thưởng mang lại chi phí ít hơn (23,7) nên đã đi qua điểm thưởng đó rồi quay lại.

III. Cổng dịch chuyển.

Thuật toán cài đặt cổng dịch chuyển:

Ta vẫn tìm các điểm thưởng nhưng lần này là kiểm tra chi phí khi đi bằng bfs vì bfs đưa ra đường đi ngắn nhất tới đích. Sau đó lặp lại việc đó cho tới khi ta tìm tới đích là sẽ được đường đi tối ưu nhất.

i. Mê cung thứ 1:



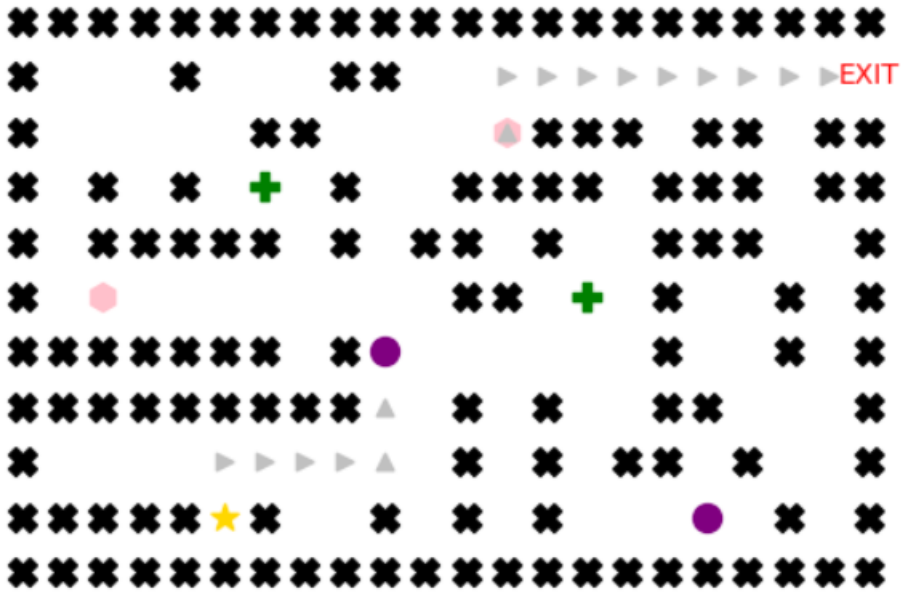
Điểm thưởng:

- (3,6): -3,
- (5,14): -1.

Cổng:

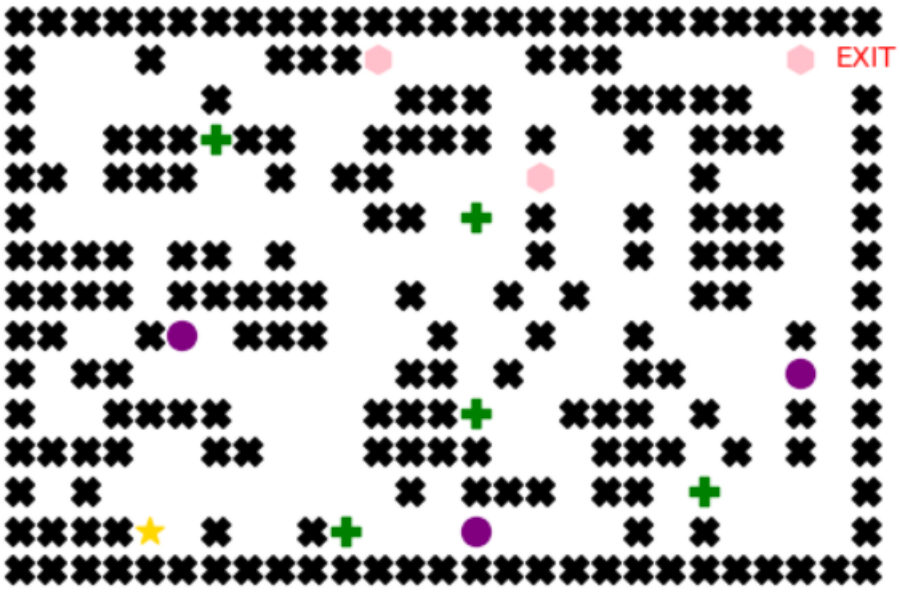
- (6,9): (2,12),
- (9,17): (5,2).

1. Giải mê cung



Đường đi: (9, 5), (8, 5), (8, 6), (8, 7), (8, 8), (8, 9), (7, 9), (6, 9), (1, 12), (1, 13), (1, 14), (1, 15), (1, 16), (1, 17), (1, 18), (1, 19), (1, 20), (1, 21)

ii. Mê cung thứ 2:



Điểm thưởng:

- (3,6): -3,
- (5,14): -5,
- (10:14) -4,
- (12:21) -2,
- (13,10): -2.

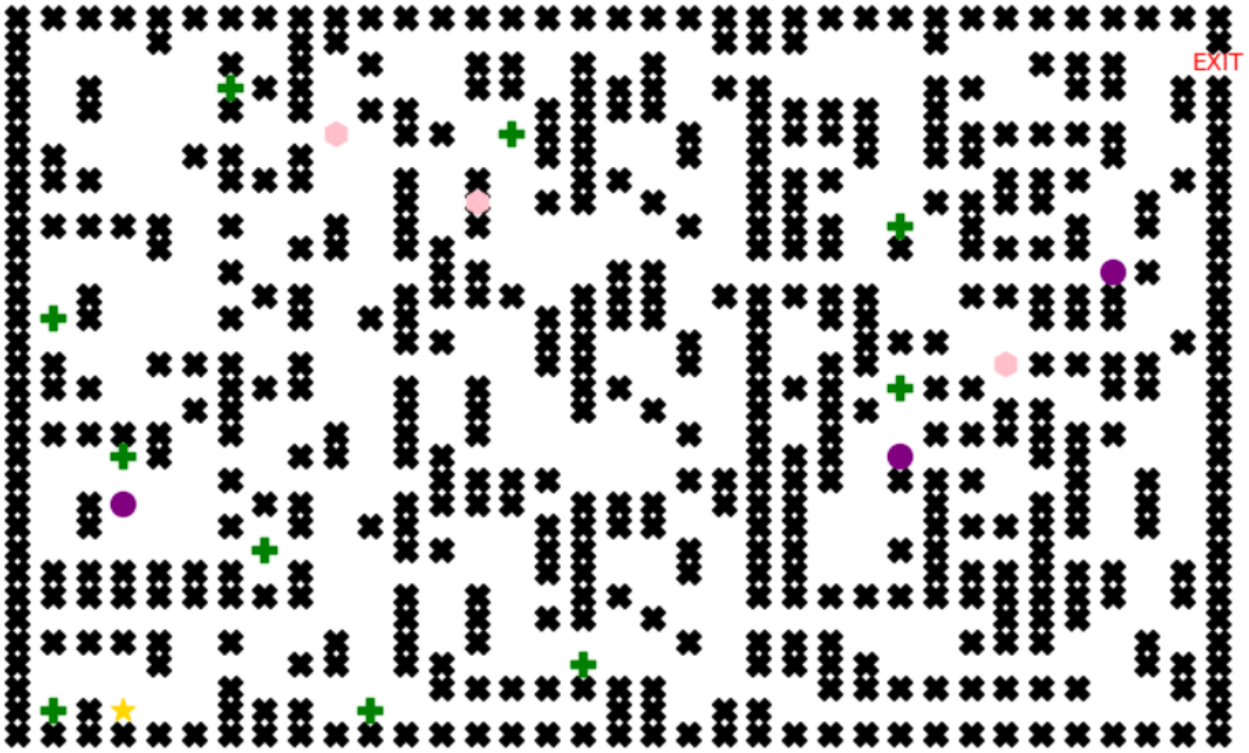
Cổng:

- (8,5): (1,11),
- (9,24): (1,24),
- (13,14): (4,16).

1. Giải mê cung

Đường đi: (13, 4), (13, 5), (12, 5), (12, 6), (12, 7), (12, 8), (12, 9), (12, 10), (13, 10), (13, 11), (13, 12), (13, 13), (13, 14), (4, 17), (4, 18), (4, 19), (4, 20), (5, 20), (6, 20), (7 , 20), (8, 20), (8, 21), (8, 22), (8, 23), (9, 23), (9, 24), (1, 25), (1, 26)

iii. Mê cung thứ 3:



Điểm thưởng:

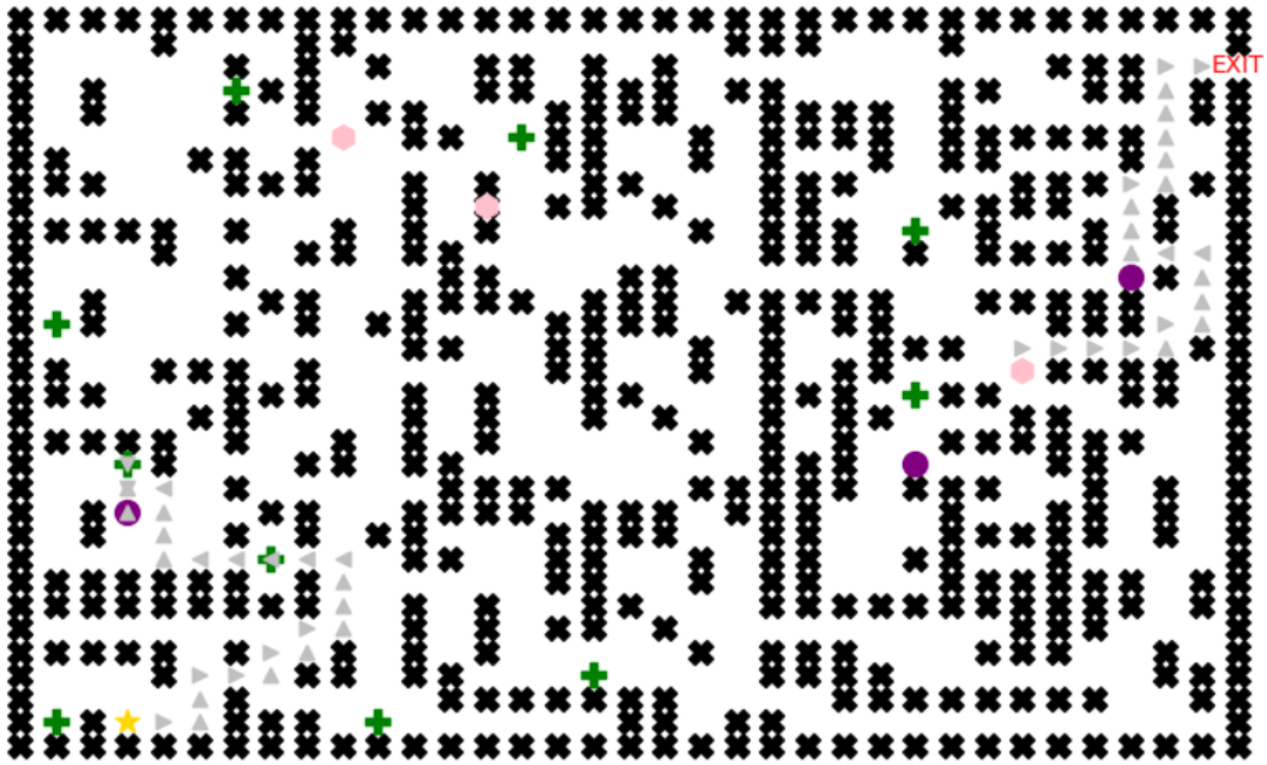
- (3,6): -3,
- (5,14): -1,
- (9,25): -2,
- (13,1): -1,
- (16,25): -4,
- (19,3): -5,
- (23,7): -8,
- (28,16): -2,
- (30,1): -1,
- (30,10): -2.

Cổng:

- (19,25): (8,13),
- (21,3): (15,28),

- (11,31): (5,9).

1. Giải mê cung



Đường đi: (30, 3), (30, 4), (30, 5), (29, 5), (28, 5), (28, 6), (28, 7), (27, 7), (27, 8), (26, 8), (26, 9), (25, 9), (24, 9), (23, 9), (23, 8), (23, 7), (23, 6), (23, 5), (23, 4), (22, 4), (21, 4), (20, 4), (20, 3), (19, 3), (20, 3), (21, 3), (14, 28), (14, 29), (14, 30), (14, 31), (14, 32), (13, 32), (13, 33), (12, 33), (11, 33), (10, 33), (10, 32), (10, 31), (9, 31), (8, 31), (7, 31), (7, 32), (6, 32), (5, 32), (4, 32), (3, 32), (2, 32), (2, 33), (2, 34)

Chỗ bị lặp lại: (20, 3), (19, 3), (20, 3)

Theo hình trên, ta có thể thấy nó đã bỏ qua các điểm thưởng mà tại đó làm tăng thêm chi phí đường đi của nó, lấy được điểm thưởng ở vị trí (19,3) dù phải đi ngược lại để tìm được con đường tối ưu nhất để tới đích.

IV. Tài liệu tham khảo:

A-Star A* Search in Python [Python Maze World- pyamaze] - YouTube

Vietnamese LiveCoding #3: Giải 1 bộ đề Google Onsite Interview - YouTube