



ĐỒ ÁN MÔN HỌC  
CƠ SỞ TRÍ TUỆ NHÂN TẠO

---

**THUẬT TOÁN TÌM KIẾM**

---



*Sinh viên thực hiện:*

**Võ Thành Nam – 19120301**

**Lương Ánh Nguyệt – 19120315**

# MỤC LỤC

<b>I.</b>	<b>Thông tin nhóm và phân công công việc.....</b>	<b>2</b>
<b>II.</b>	<b>Đánh giá mức độ hoàn thành .....</b>	<b>2</b>
<b>III.</b>	<b>Nội dung báo cáo .....</b>	<b>3</b>
1.	Bản đồ không có điểm thưởng .....	3
**	Nhận xét chung và phân tích về 4 thuật toán .....	8
2.	Bản đồ có điểm thưởng.....	9
a)	Chiến lược đề xuất.....	9
b)	Kết quả chạy chương trình.....	10

## I. Thông tin nhóm và phân công công việc

Họ và tên	MSSV	Công việc
Võ Thành Nam	19120301	Thuật toán tìm kiếm DFS, BFS, viết báo cáo, thiết kế bản đồ
Lương Ánh Nguyệt	19120315	Thuật toán tìm kiếm Greedy Best First Search, A*, thuật toán cho bản đồ có điểm thưởng, viết báo cáo

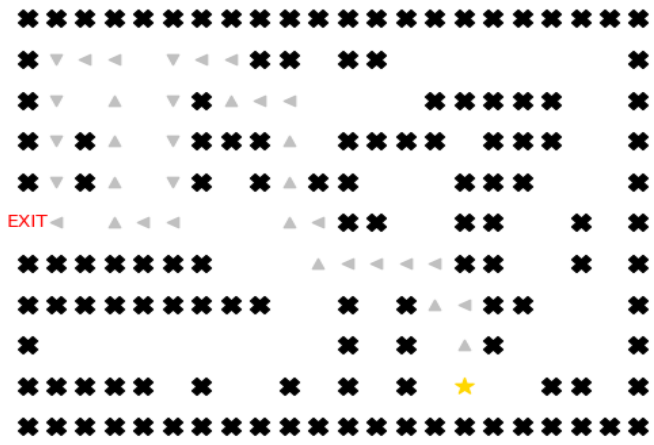
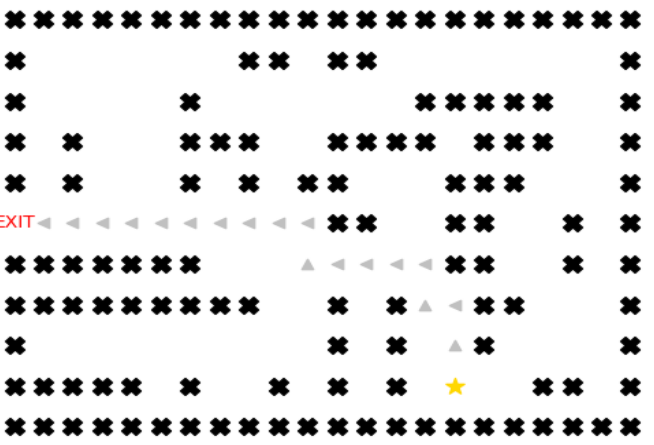
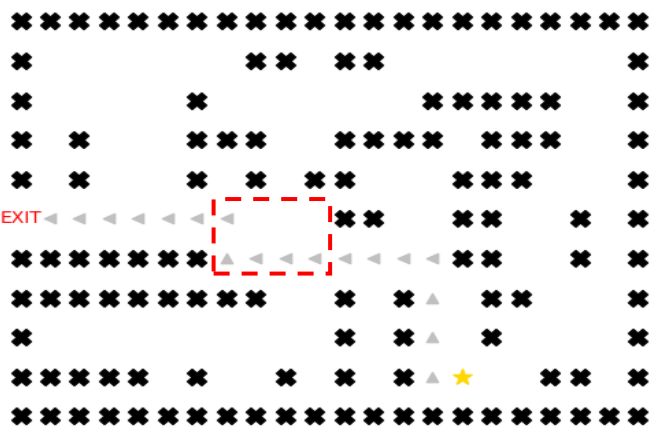
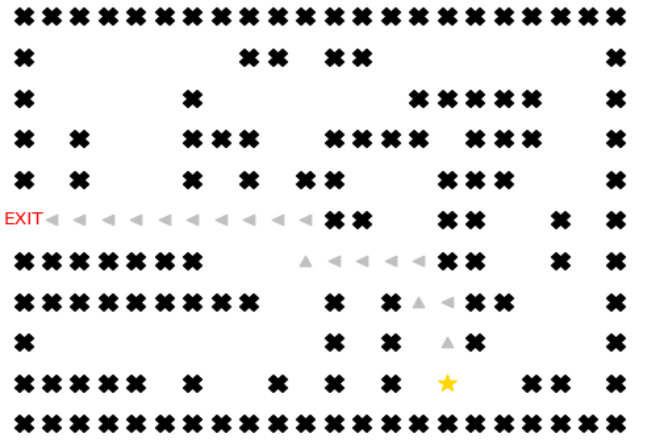
## II. Đánh giá mức độ hoàn thành

- Đối với bài toán bản đồ không có điểm thưởng: hoàn thành bốn giải thuật tìm kiếm không có thông tin và có thông tin.
- Đối với bản đồ có điểm thưởng: đề xuất được chiến lược tối ưu
- Đối với các kịch bản bản đồ khó hơn: chưa hoàn thành.

### III. Nội dung báo cáo

#### 1. Bản đồ không có điểm thưởng

##### a) Bản đồ 1

	
DFS – Chi phí: 36	BFS – Chi phí: 20
	
Greedy Best First Search – Chi phí: 20	A* - Chi phí: 20

- Kích thước bản đồ: 22x11
- Nhận xét:
  - ✓ Thuật toán BFS và A\* cho ra 2 kết quả đường đi giống như nhau
  - ✓ GBFS tuy cho hình dạng đường đi có khác biệt (tại vùng đã được khoanh đỏ) nhưng vẫn là đường đi có chi phí nhỏ nhất (20 bước) giống BFS và A\*

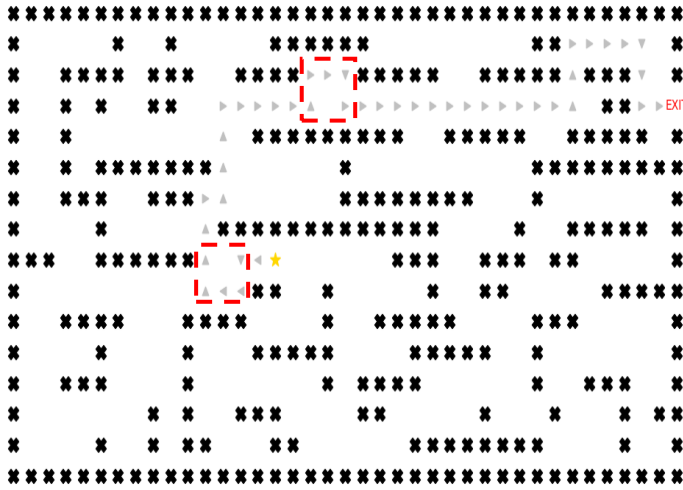
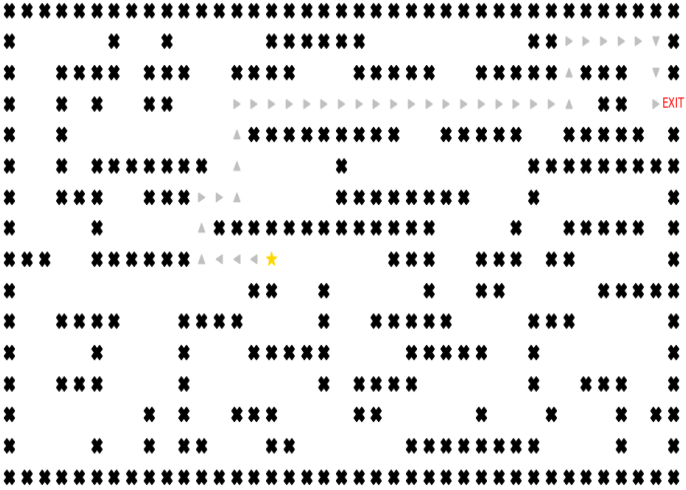
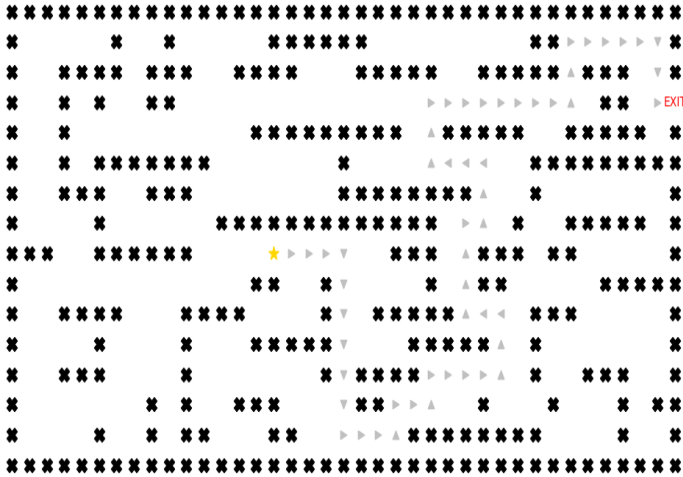
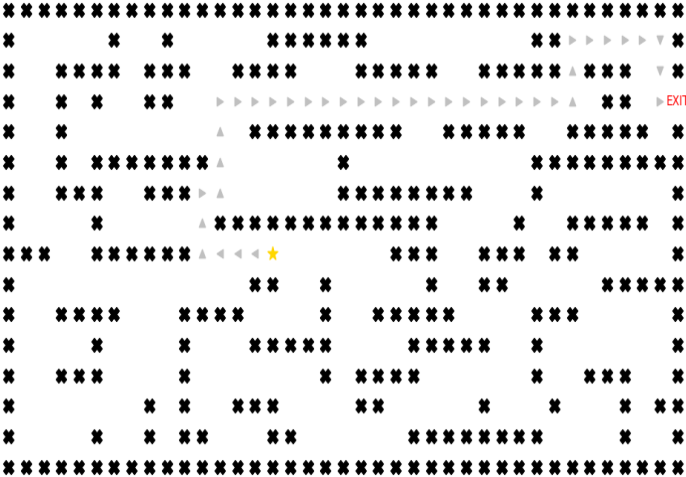
- ✓ DFS chỉ ra được 1 đường đi ra khỏi mê cung, nhưng đường đi này tốn chi phí hơn khá nhiều so với 3 thuật toán còn lại (36 bước)

## b) Bản đồ 2

DFS – Chi phí: 76	BFS – Chi phí: 60
Greedy Best First Search – Chi phí: 72	A* – Chi phí: 60

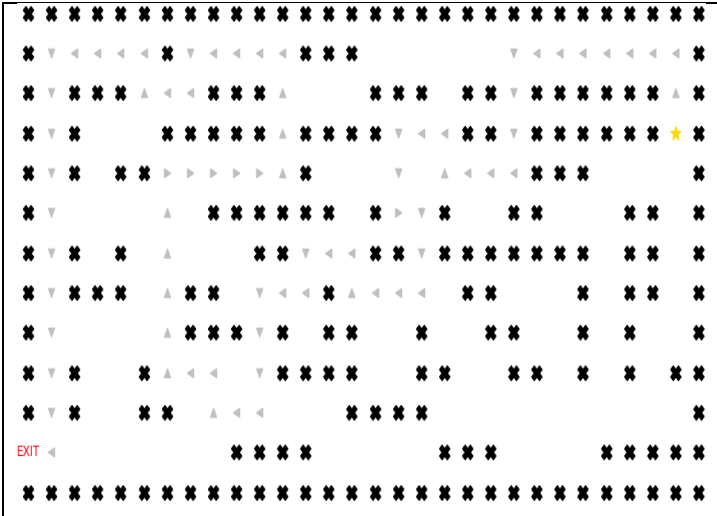
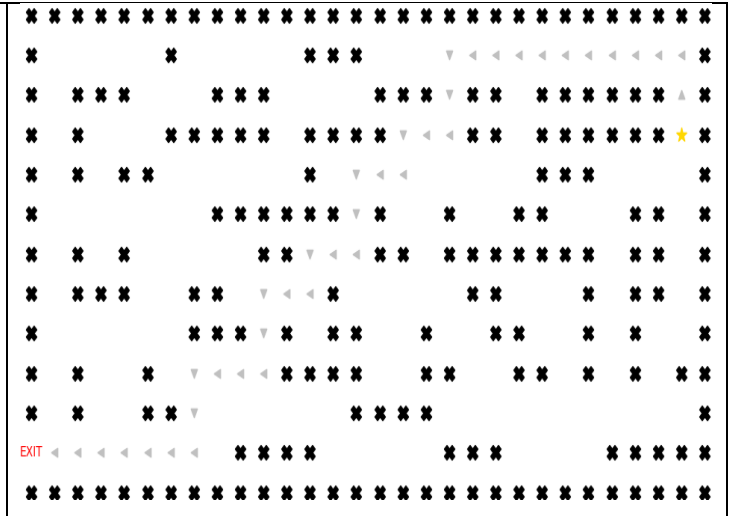
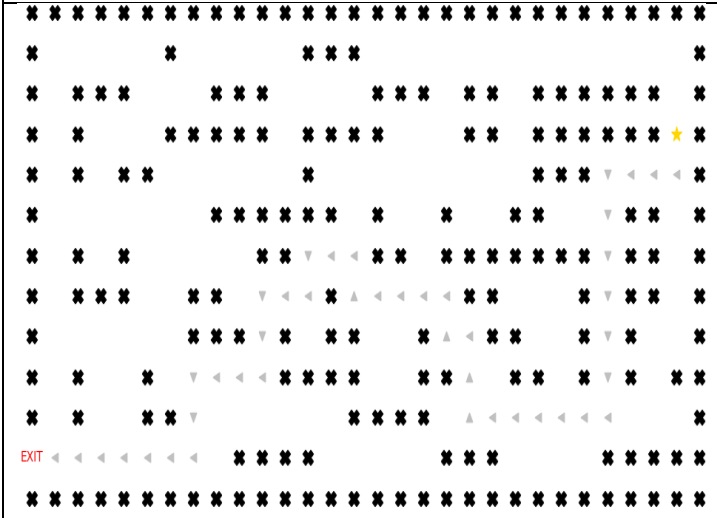
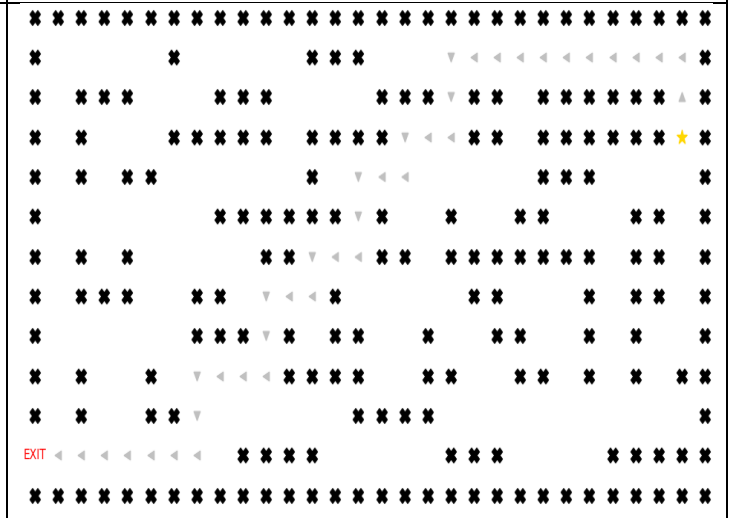
- Kích thước bản đồ: 34x14
- Nhận xét:
  - ✓ BFS và A\* vẫn cho ra đường đi tốn ít chi phí nhất giống nhau (60 bước)
  - ✓ GBFS và DFS tìm được đường đi có sự chênh lệch nhau ít (4 bước), nhưng cả 2 kết quả đều lớn hơn khá nhiều so với 2 thuật toán kia (hơn 12 và 16 bước)
  - ✓ Trong đó, đường đi của DFS có chi phí lớn nhất (76 bước)

### c) Bản đồ 3

	
DFS – Chi phí: 44	BFS – Chi phí: 40
	
Greedy Best First Search – Chi phí: 54	A* - Chi phí: 40

- Kích thước bản đồ: 39x16
- Nhận xét:
  - ✓ Kết quả đường đi của BFS và A\* là giống nhau và tốn ít chi phí nhất (40 bước)
  - ✓ DFS có hình dạng đường đi gần giống 2 thuật toán trên, tuy nhiên vẫn có sự khác biệt (tại 2 vị trí khoanh đỏ) dẫn đến chênh lệch 4 bước so với 2 thuật toán trên
  - ✓ GBFS tìm ra đường đi khác hoàn toàn so với 3 thuật toán còn lại, và cũng là đường đi tốn nhiều chi phí nhất (54 bước)

d) Bản đồ 4

	
DFS – Chi phí: 76	BFS – Chi phí: 40
	
Greedy Best First Search – Chi phí: 44	A* - Chi phí: 40

- Kích thước bản đồ: 30x13
- Nhận xét:
  - ✓ BFS và A\* tiếp tục cho ra đường đi ngắn nhất giống nhau (40 bước)
  - ✓ GBFS tìm được đường đi khác và có sự chênh lệch nhỏ so với 2 thuật toán trên (4 bước)
  - ✓ Trong khi đó, DFS lại có đường đi rất dài, với chi phí gần gấp đôi đường đi ngắn nhất (76 bước)

### e) Bản đồ 5

DFS – Chi phí: 50	BFS – Chi phí: 50
Greedy Best First Search – Chi phí: 50	A* - Chi phí: 50

- Kích thước bản đồ: 30x17
- Nhận xét:
  - ✓ BFS có đường đi có khác biệt nhỏ (tại vị trí khoanh đỏ) so với đường đi của các thuật toán khác
  - ✓ Dù vậy, cả 4 thuật toán đều có kết quả chi phí giống như nhau (50 bước), là chi phí nhỏ nhất



#### f) Nhận xét chung và phân tích về 4 thuật toán

✍ Qua các bản đồ trên, thuật toán BFS và A\* luôn cho kết quả đường đi là tối ưu nhất.

✍ Tuy nhiên, xét về độ phức tạp thời gian, BFS sẽ chạy chậm hơn so với A\* khi bản đồ lớn, vì BFS tìm kiếm mù theo chiều ngang dẫn đến việc có thể phải duyệt nhiều nút, nhiều nhánh mà không có khả năng nằm trong đường đi tối ưu. Trường hợp xấu, thuật toán sẽ duyệt toàn bộ bản đồ.

✍ Thuật toán DFS đảm bảo luôn tìm đường đi ra khỏi mê cung, nhưng lại không phải là lời giải tối ưu. Trong hầu hết các trường hợp (4/5 bản đồ được xét), DFS đều cho ra đường đi dài hơn đường đi ngắn nhất, thậm chí có bản đồ cho kết quả rất lớn (bản đồ 4). Trong trường hợp tốt, thuật toán này có thể đưa ra đường đi ngắn hơn đường đi của GBFS (bản đồ 3). Có rất ít trường hợp tốt để DFS có thể cho ra kết quả tối ưu. Một ví dụ với bản đồ 5, vì chỉ có 1 đường đi ra khỏi mê cung, nên đường đi DFS tìm được cũng là đường đi tối ưu.

✍ Thuật toán GBFS không có tính tối ưu (3/5 bản đồ không phải đường đi ngắn nhất), nhưng thời gian tìm kiếm kết quả tốt hơn thuật toán tìm kiếm không có thông tin. Có một số trường hợp tốt, thuật toán này sẽ tìm được đường đi ngắn nhất (bản đồ 1). Với trường hợp chỉ có 1 đường đi như bản đồ 5, GBFS chạy nhanh hơn BFS và DFS.

✍ Nhìn vào cả 5 bản đồ, kết quả nhận được khi sử dụng thuật toán A\* đều là đường đi ngắn nhất. Có thể thấy A\* là thuật toán tốt nhất trong 4 thuật toán khi vừa đảm bảo về thời gian tìm kiếm nhanh (giống GBFS) vừa đảm bảo về tính tối ưu (giống UCS).

## 2. Bản đồ có điểm thưởng

### a) Chiến lược đề xuất

#### ☆ *Phân tích bài toán*

Với bản đồ không có điểm thưởng, sử dụng thuật toán  $A^*$  sẽ cho ta đường đi ngắn nhất (cũng là đường đi tốn ít chi phí nhất) từ Start (ký hiệu S) đến End (ký hiệu E). Việc xuất hiện các điểm thưởng (ký hiệu  $B_i$ ) trên bản đồ làm cho đường đi ngắn nhất  $S \rightarrow E$  không phải là đường đi tốn ít chi phí nhất. Lúc này, đường đi cần tìm có thể phải là  $S \rightarrow B_i \rightarrow \dots \rightarrow B_j \rightarrow E$ .

#### ☆ *Ý tưởng giải quyết bài toán*

Dựa trên nguyên lý Bellman. Ta xem S,  $B_i$ , E là các đỉnh trên 1 đồ thị và các cung giữa các đỉnh sẽ là đường đi ngắn nhất giữa 2 đỉnh đó (tìm bằng thuật toán  $A^*$ , xem như bản đồ không có điểm thưởng). Ban đầu ta có được đường đi từ S đến các đỉnh còn lại. Với mỗi đỉnh  $B_i$ , ta tìm được đường đi  $B_i \rightarrow B_j$ , sau đó kiểm tra:

$$S \rightarrow B_j > S \rightarrow B_i + B_i \rightarrow B_j$$

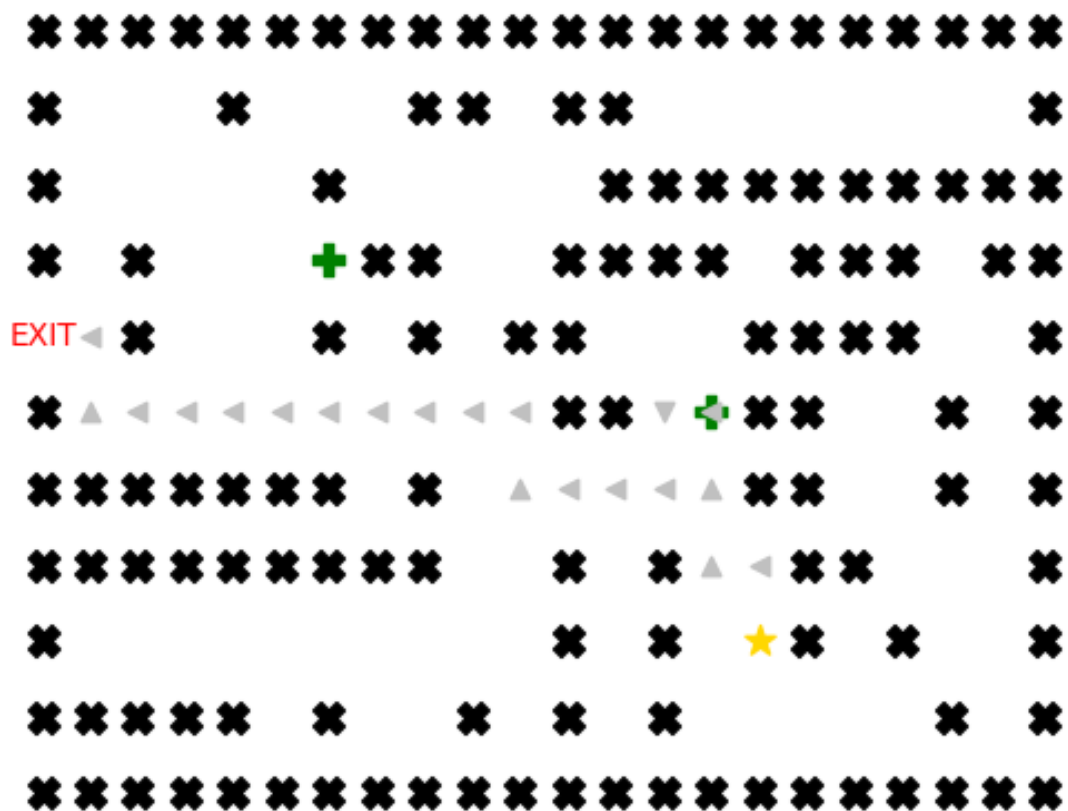
Nếu đường đi mới đến  $B_j$  thông qua  $B_i$  tốn ít chi phí hơn đường đi cũ  $S \rightarrow B_j$ , ta cập nhật đường đi mới  $S \rightarrow B_j = S \rightarrow B_i + B_i \rightarrow B_j$  (ở đây đỉnh E cũng được xem như 1 đỉnh  $B_j$  để xét).

Chiến lược này tương tự với thuật toán Bellman-Ford. Để giảm bớt thời gian chạy chương trình khi phải duyệt 1 đỉnh  $B_i$  nhiều lần, nếu  $S \rightarrow B_i$  có sự thay đổi, ta mới xét lại  $B_i$  để kiểm tra  $S \rightarrow B_i \rightarrow B_j$  theo giá trị  $S \rightarrow B_i$  mới cập nhật. Còn nếu không thì không cần xét lại  $B_i$  làm gì.

Kết quả cuối cùng  $S \rightarrow E$  sẽ là đường đi tốn ít chi phí nhất.

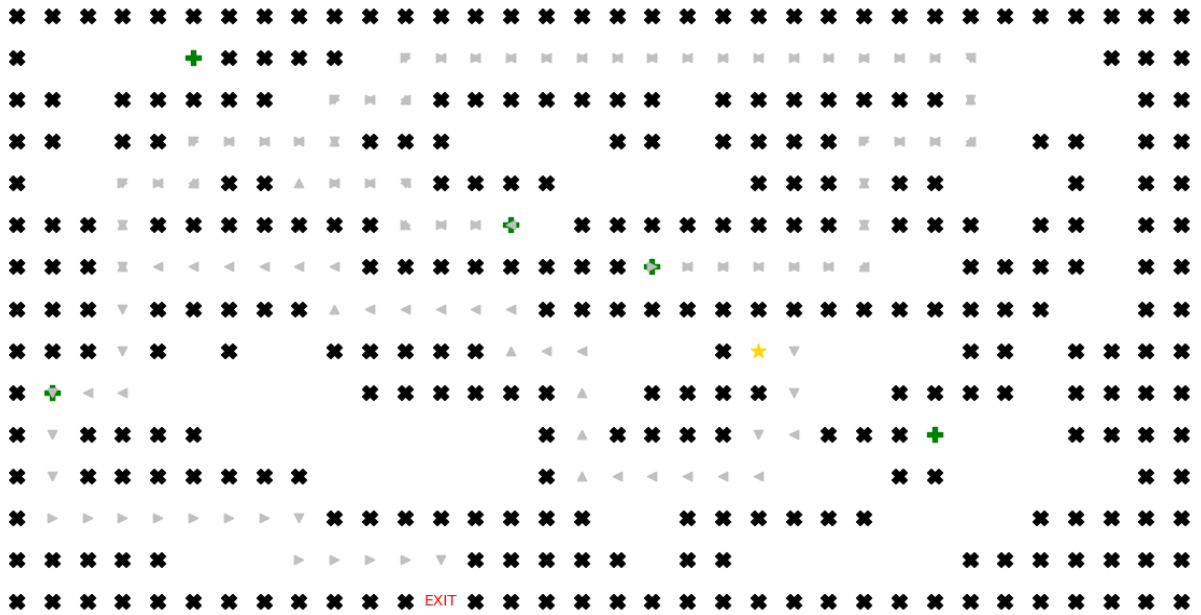
### b) Kết quả chạy chương trình

i. Bản đồ với hai điểm thưởng



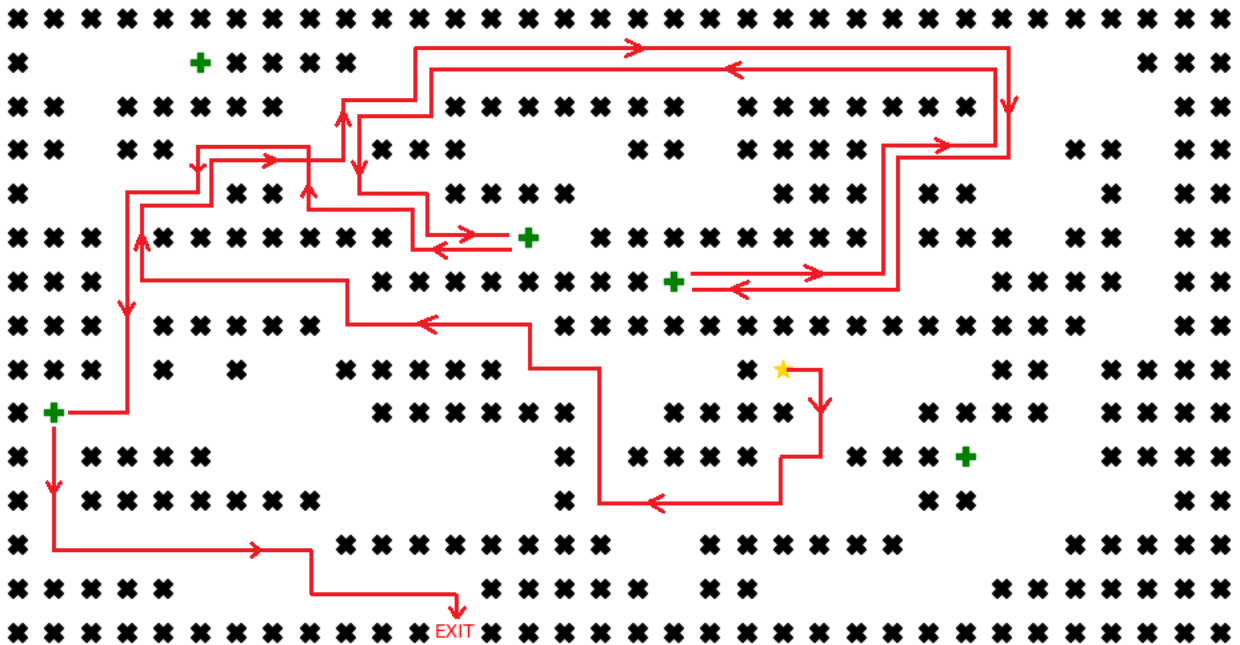
- Kích thước bản đồ: 22x11
- Tổng chi phí: 17

ii. Bản đồ với năm điểm thưởng

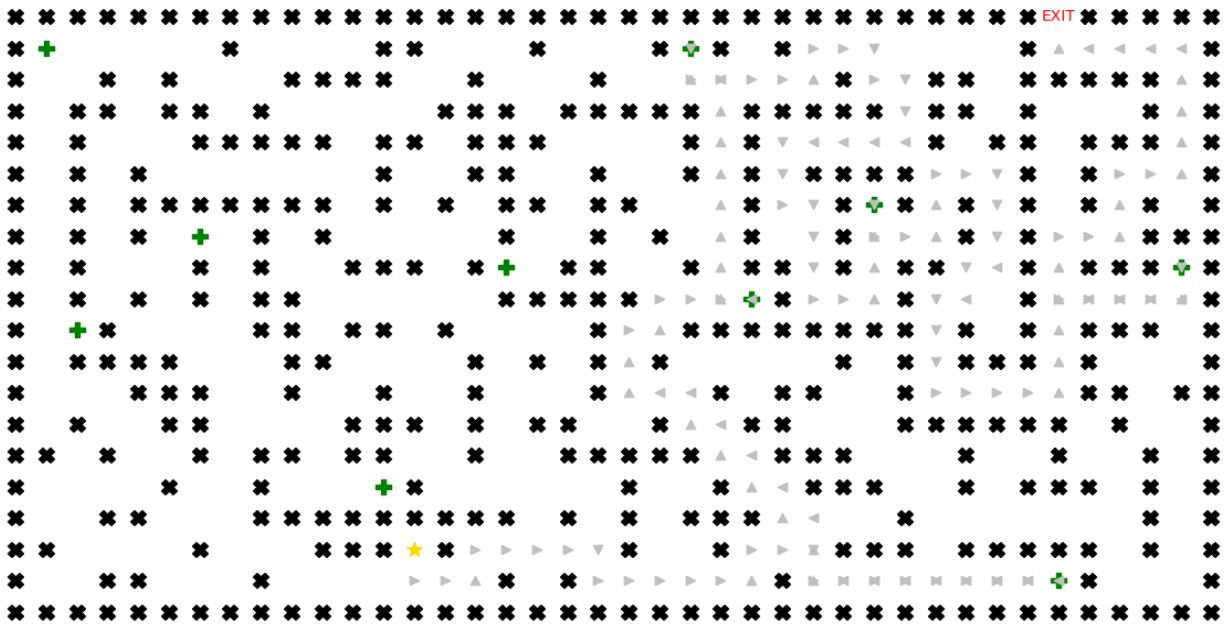


- Kích thước bản đồ: 34x15
- Tổng chi phí: 23

**\*\*** *Mô tả lại đường đi*



iii. Bản đồ với mười điểm thưởng



- Kích thước bản đồ: 40x20
- Tổng chi phí: 30

**\*\* Mô tả lại đường đi**

