

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG – HCM**  
**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN**  
**CÁC THUẬT TOÁN TÌM KIẾM**

Môn học: Cơ Sở Trí Tuệ Nhân Tạo

**Lớp: 20\_21**

**Thành viên nhóm:**

Nguyễn Ngọc Khánh Vy - 20120238

Thái Mai Khánh Vy – 20120239

Nguyễn Hoàng Anh – 20120246

**TPHCM, ngày 12 tháng 10 năm 2022**

## Mục lục

<b>I. THÔNG TIN CHUNG:</b>	3
1. Thành viên nhóm	3
2. Bảng phân công công việc	3
3. Mức độ hoàn thành	3
4. Mục tiêu đồ án	3
<b>II. MÔ TẢ CÀI ĐẶT THUẬT TOÁN:</b>	4
1. Thuật toán Depth First Search (DFS)	4
2. Thuật toán tìm kiếm Depth First Search (BFS)	5
3. Thuật toán tìm kiếm Uniform-cost Search (UCS)	7
4. Thuật toán tìm kiếm tham lam (GBFS)	9
4.1. Euclidean Distance Heuristic	10
4.2. Mahattan Distance Heuristic	11
5. Thuật toán tìm kiếm A*	12
5.1. Mahattan Distance	13
5.2. Diagonal Distance	14
6. Nhận xét các thuật toán:	15
<b>III. Bản đồ có điểm thưởng:</b>	15
1. Chiến lược đề xuất:	15
a. Phân tích bài toán:	15
b. Ý tưởng xử lý bài toán	15
2. Mô tả các bước chạy bài toán:	16
3. Kết quả chạy chương trình:	16
a. Bản đồ 2 điểm thưởng:	16
b. Bản đồ 5 điểm thưởng	17
c. Bản đồ 10 điểm thưởng	17
<b>IV. Kịch bản nâng cấp:</b>	18
1. Mô tả kịch bản	18
2. Đề xuất thuật toán	18
3. Kết quả	19
<b>IV. Video minh họa:</b>	20
1. Cài đặt môi trường:	20
2. Chú thích video minh họa:	26

## I. THÔNG TIN CHUNG:

### 1. Thành viên nhóm

- Nguyễn Ngọc Khánh Vy -20120239
- Thái Mai Khánh Vy – 20120239
- Nguyễn Hoàng Anh – 20120246

### 2. Bảng phân công công việc

STT	Thành viên	Công việc
1	Nguyễn Ngọc Khánh Vy	Thuật toán BFS,A*,thuật toán trên bản đồ có điểm thưởng
2	Thái Mai Khánh Vy	Thuật toán DFS, thiết kế 8 bản đồ, xuất file đồ họa và viết báo cáo
3	Nguyễn Hoàng Anh	Thuật toán GBFS,UCS,thuật toán trên bản đồ có điểm thưởng

### 3. Mức độ hoàn thành

Mức	Mức độ hoàn thành	Chú thích
1a	100%	
1b	100%	
2a	100%	
2b	100%	
Kịch bản nâng cấp	100%	
Video minh họa	100%	Cần được cài đặt môi trường

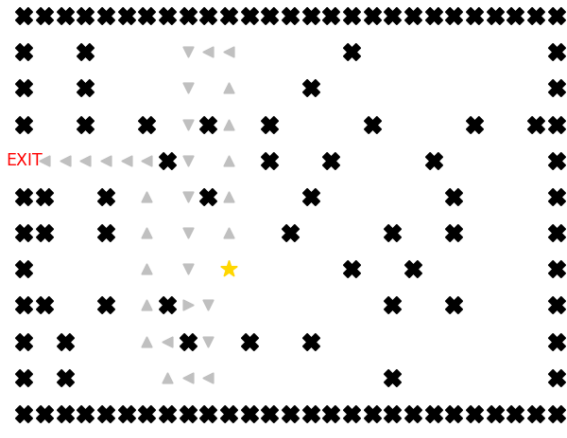
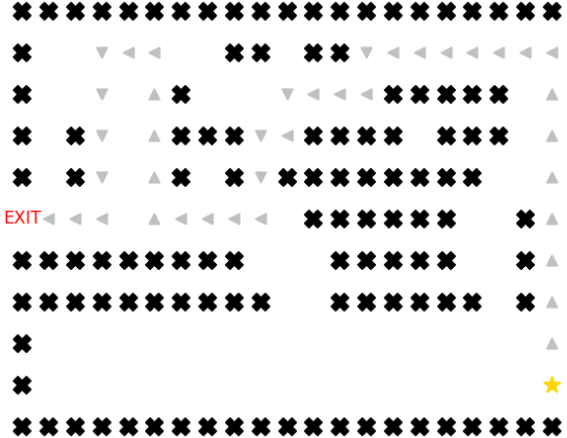
### 4. Mục tiêu đồ án

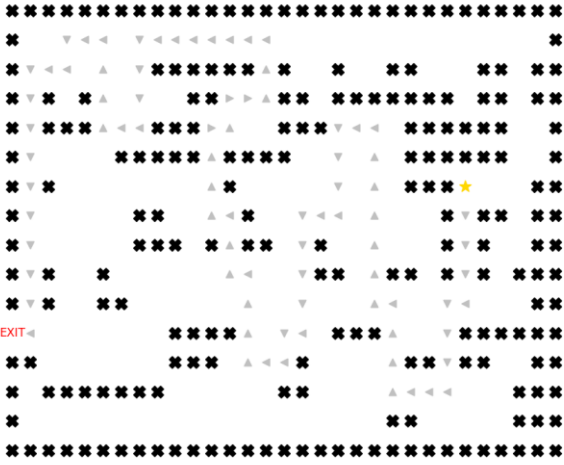
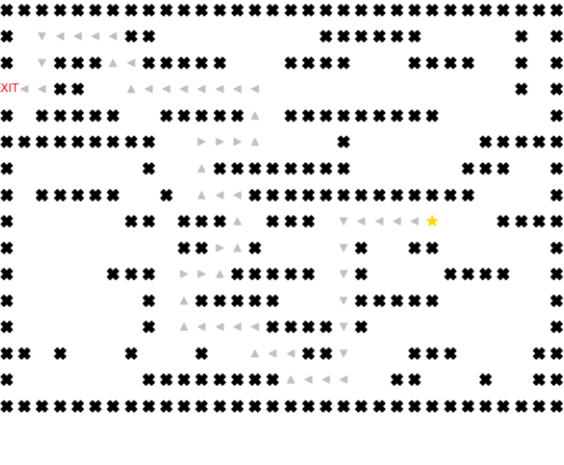
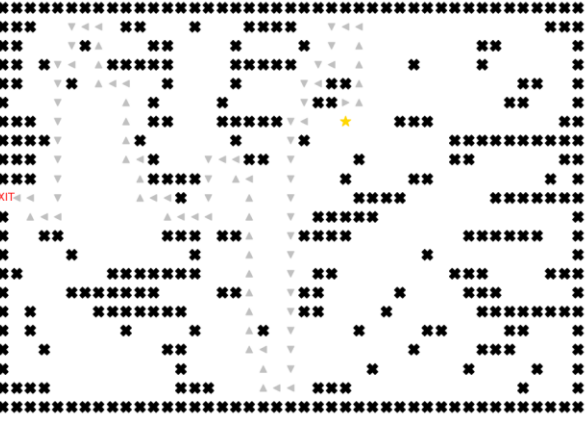
Nghiên cứu, cài đặt và so sánh các thuật toán tìm kiếm đường đi.

## II. MÔ TẢ CÀI ĐẶT THUẬT TOÁN:

### 1. Thuật toán Depth First Search (DFS)

Cài đặt	Độ phức tạp	Tính chất
Sử dụng cấu trúc <b>Stack</b> – <b>ngăn xếp</b> để đẩy vào các đỉnh ghé thăm. Ô đầu tiên được đẩy vào Stack là điểm bắt đầu “S”. Xét 4 ô liên kề với ô hiện tại được <b>Pop()</b> ra từ Stack, nếu 4 ô đó có ô thể đi được, ta tiến hành <b>Push</b> ô đó vào Stack và đánh dấu vào mảng <b>Trace</b> để truy vết tới ô trước đó của ô vừa được Push vào. Sau đó ta tiến hành Pop() ô cuối trong Stack ra (với cấu trúc <i>FILO</i> của Stack sẽ ứng với cấu trúc tìm kiếm theo chiều sâu <i>DFS</i> ) cho đến khi <b>tới được đích thì thoát ra</b> rồi tiến hành <b>truy vết mảng Trace</b> để có được đi đường đi cần tìm.	<b>Thời gian:</b> $O(B^{LMAX})$ <b>Không gian:</b> $O(LMAX)$	Vì có kỹ thuật đánh dấu, nên thủ tục DFS sẽ được gọi $\leq n$ lần (n là số đỉnh) Đường đi từ S tới F có thể có nhiều, DFS chỉ chỉ ra một trong số các đường đi. Cụ thể là đường đi có thứ tự từ điển nhỏ nhất.  Đây không phải là một thuật toán tối ưu để kiểm đường đi nhỏ nhất. Vì thuật toán chỉ kiểm ra đường đi và đồng thời tốn rất nhiều thời gian do mang tính chất mù quáng, duyệt tất cả đỉnh.

	
Map 1 (12x27) – Cost = 33	Map 2 (11x22) – Cost = 40

	
Map 3 (16x31) – Cost = 82	Map 4 (16x32) – Cost = 57
	
Map 5 (22x43) – Cost = 87	

## 2. Thuật toán tìm kiếm Depth First Search (BFS)

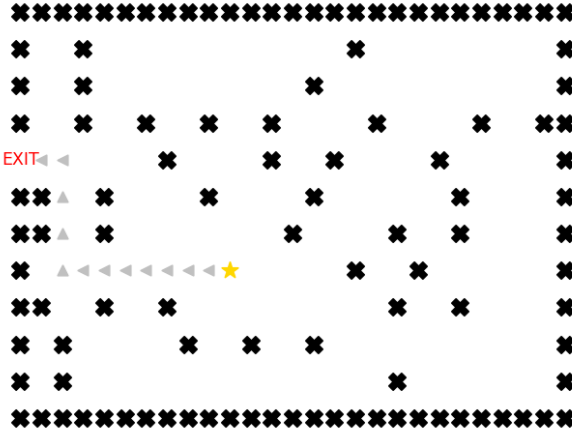
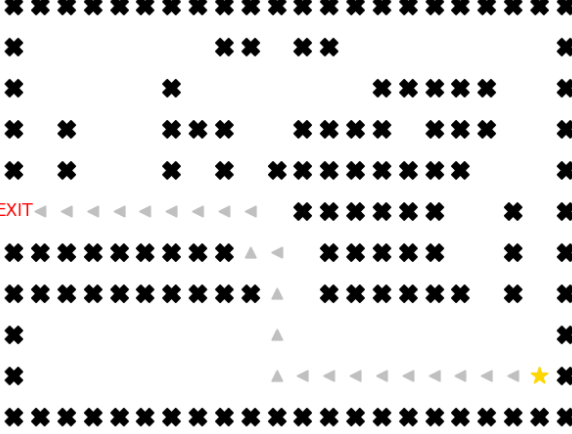
Cài đặt	Độ phức tạp	Tính chất
Sử dụng cấu trúc <b>Queue</b> - hàng đợi để đẩy các đỉnh sẽ ghé thăm. Ô đầu tiên được đẩy vào Queue là điểm bắt đầu "S".	<b>Thời gian:</b> $O(B^{m^2})$ <b>Không gian:</b> $O(B^{m^2})$	Vì có thủ tục đánh dấu, nên BFS sẽ được gọi $\leq n$ lần. Đường đi từ S đến G có thể có nhiều nhưng BFS chỉ tìm ra

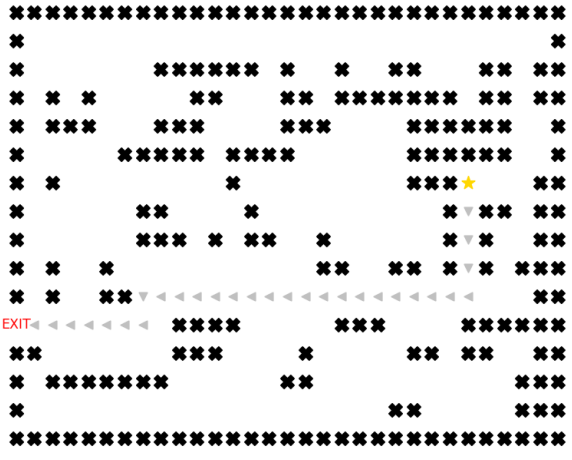

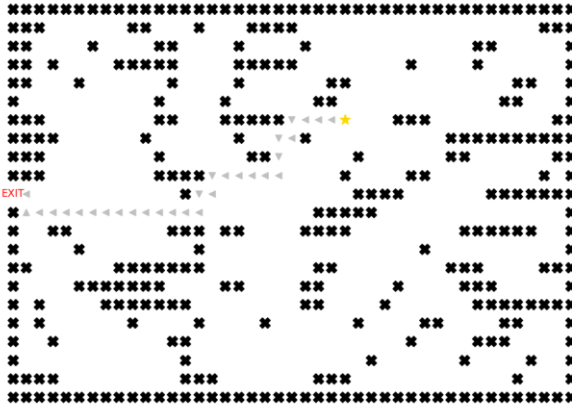
**Xét 4 ô liên kề:** trên, dưới, trái, phải của ô đang xét, nếu ô nào đi được ta sẽ Push ô đó vào Queue và đánh dấu vào mảng Trace để lưu truy vết và chi phí đường đi từ ô trước đó đến ô đang xét.

Với mỗi lần xét, ta tiến hành **Pop ô đầu tiên** trong Queue ra (*theo cấu trúc FIFO của Queue*) cho đến khi ô Pop ra là đích đến thì thoát ra, tiến hành truy vết mảng Trace để có được đường đi cần tìm.

đường đi có số bước ít nhất mà không phải là đường đi có chi phí nhỏ nhất. Nhưng với bản đồ này, vì chi phí bằng nhau nên BFS sẽ trả về đường đi ngắn nhất từ S đến G.

Tuy nhiên, do tính chất tìm đường đi mù quáng, mà thuật toán tốn nhiều thời gian và không gian cho việc tìm kiếm.

	
Map 1 (12x27) – Cost = 13	Map 2 (11x22) – Cost = 24

	
Map 3 (16x31) – Cost = 30	Map 4 (16x32) – Cost = 57
	
Map 5 (22x43) – Cost = 31	

### 3. Thuật toán tìm kiếm Uniform-cost Search (UCS)

Cài đặt	Độ phức tạp	Tính chất
<ul style="list-style-type: none"> <li>- Thuật toán có chiến lược tương tự như thuật toán Dijkstra, tại mỗi lần thực hiện sẽ mở đỉnh có chi phí thấp nhất. Để thực hiện ta cần cấu trúc dữ liệu hàng đợi ưu tiên để lưu chi phí các đỉnh và trả về đỉnh có chi phí thấp nhất.</li> <li>- Đầu tiên, khởi tạo hàng đợi ưu tiên chỉ gồm đỉnh start với độ ưu tiên là 0. Sau đó lặp lại quá trình lấy đỉnh u có độ ưu tiên cao nhất</li> </ul>	<p>Với <math>\epsilon</math> là chi phí di chuyển thấp nhất ở mỗi bước, <math>C^*</math> là chi phí lời giải tối ưu thì</p> <p><b>Thời gian:</b></p> <p><math>O(b^{1+\lceil \frac{C^*}{\epsilon} \rceil})</math>. (Khi tất cả chi phí di chuyển đều bằng nhau thì thời gian thực hiện là <math>O(b^{d+1})</math>).</p>	<ul style="list-style-type: none"> <li>- Tính đầy đủ: Có nếu không gian tìm kiếm có giới hạn và không có cạnh mang trọng số âm.</li> <li>- Tính tối ưu: Có</li> <li>- Thuật toán hoạt động hiệu quả hơn so với BFS trên đồ thị có trọng số không âm. Tuy nhiên, trong</li> </ul>

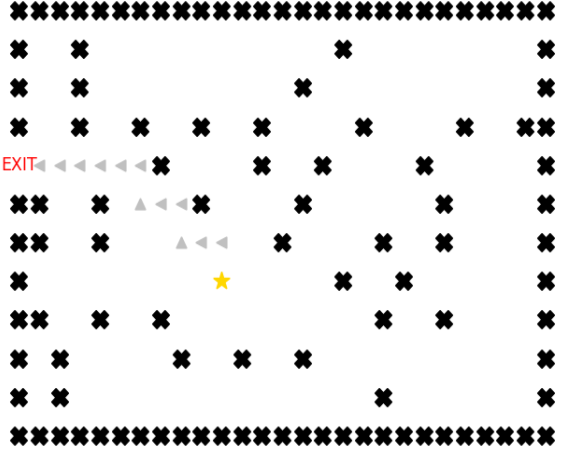
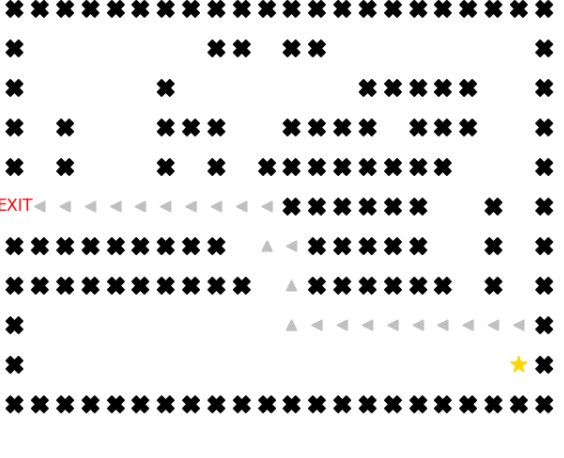
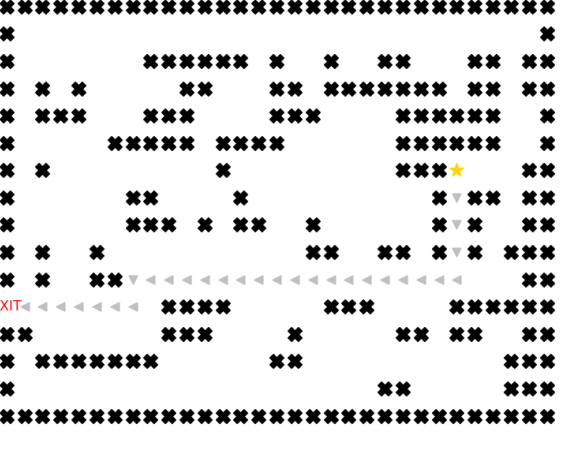

ra khỏi hàng đợi, xét các đỉnh  $v$  có thể đến được từ  $u$ . Nếu đường đi từ  $u$  đến  $v$  nhỏ hơn thì ta cập nhật lại hàng đợi ưu tiên.

- Để truy vết đường đi tìm được, ta dùng mảng Trace với  $\text{Trace}[u] = v$  là đỉnh  $u$  được thăm từ đỉnh  $v$ .

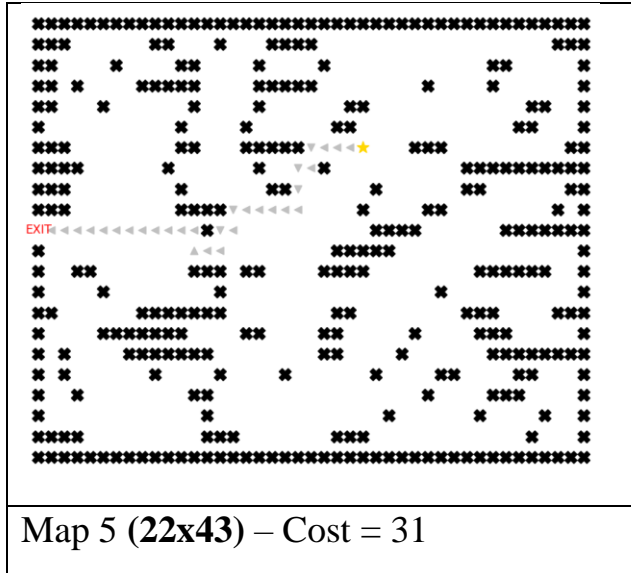
**Không gian:**

$$O(b^{1+\lceil \frac{C^*}{\epsilon} \rceil})$$

phạm vi bài toán, vì các chi phí đều bằng nhau nên UCS sẽ tương tự BFS như gặp vấn đề về bộ nhớ và thời gian thực hiện.

	
Map 1 (12x27) – Cost = 13	Map 2 (11x22) – Cost = 24
	
Map 3 (16x31) – Cost = 30	Map 4 (16x32) – Cost = 57

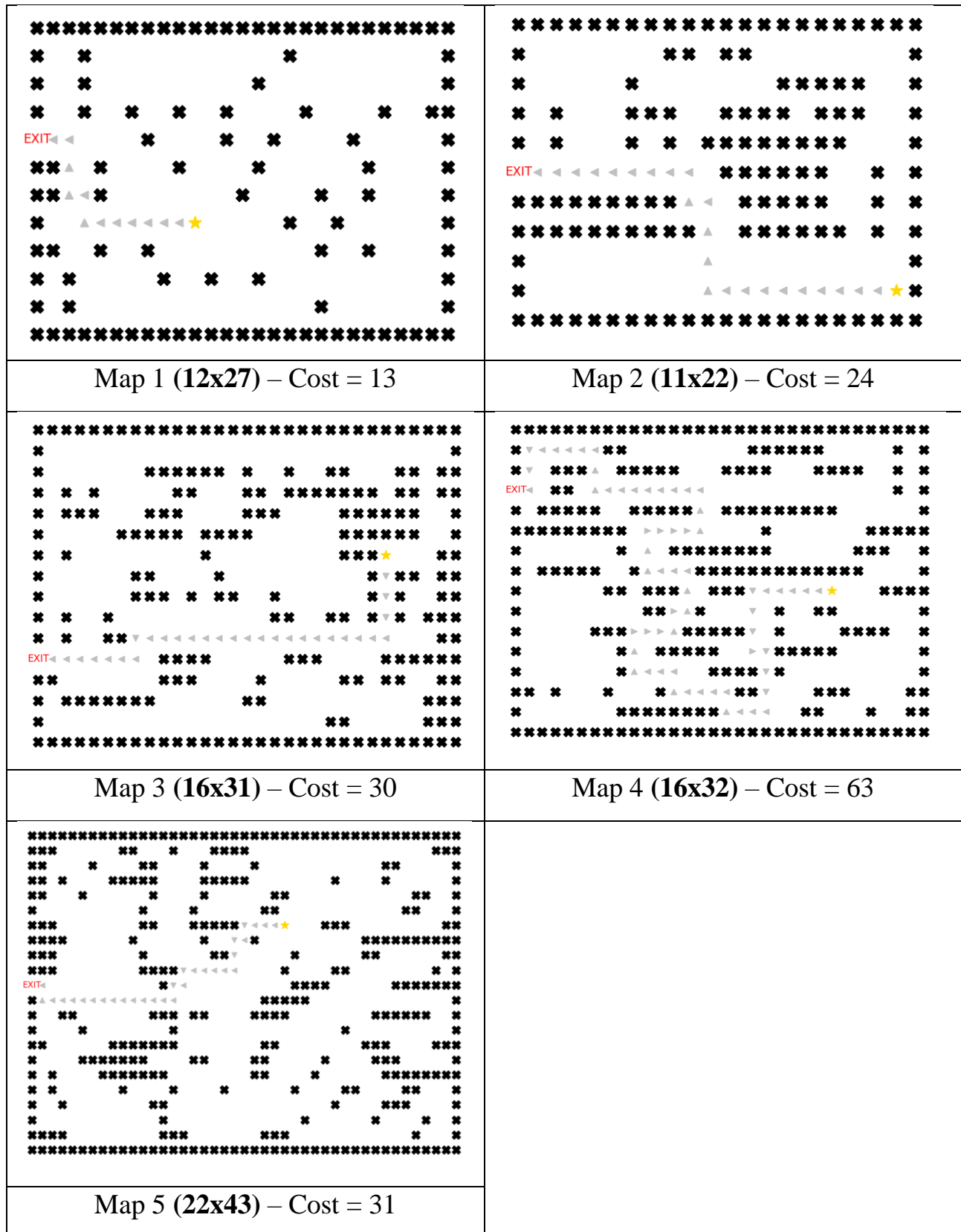




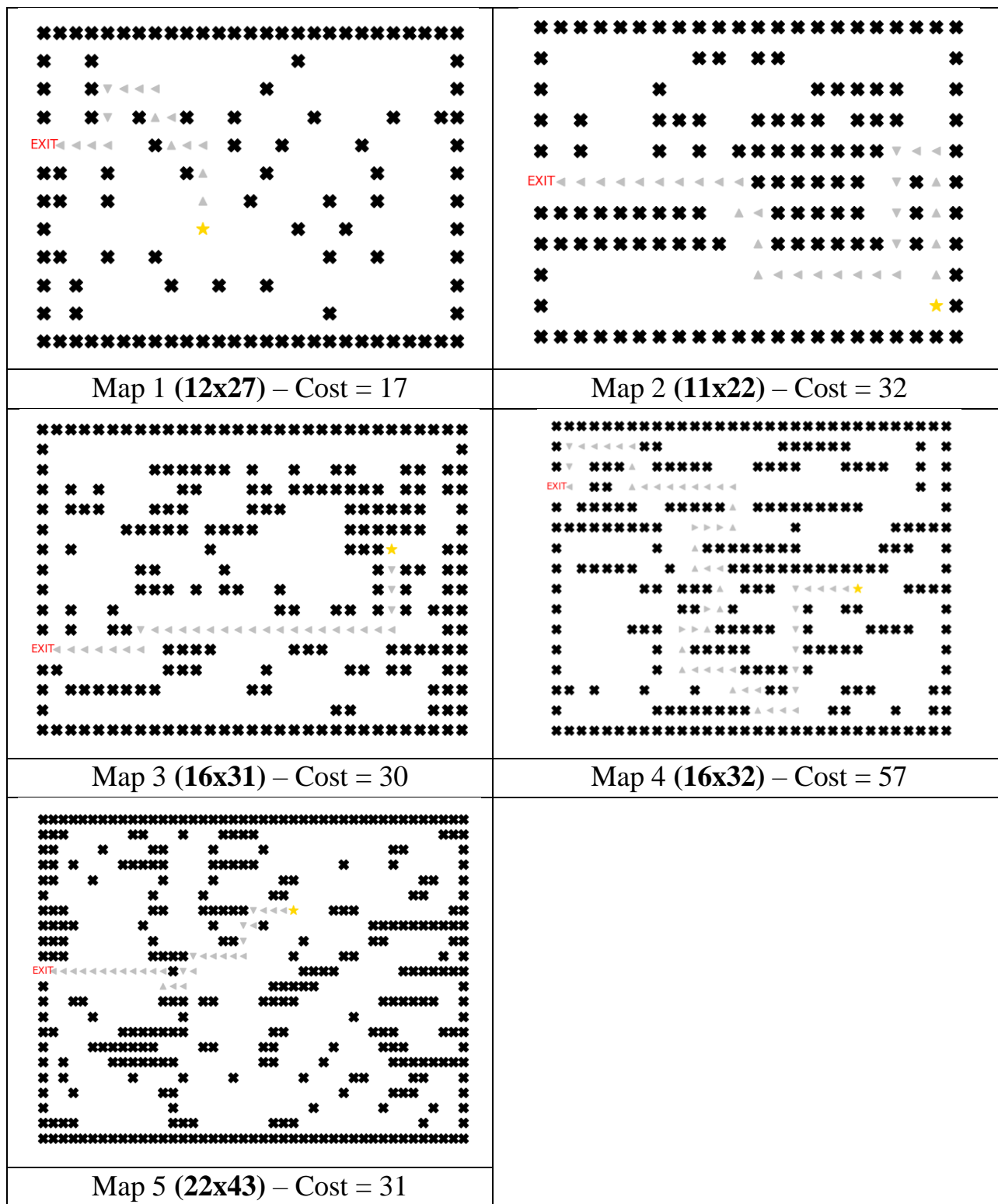
#### 4. Thuật toán tìm kiếm tham lam (GBFS)

Cài đặt	Độ phức tạp	Tính chất
<ul style="list-style-type: none"> <li>Cài đặt tương tự như UCS, nhưng sẽ mở đỉnh được ước lượng là gần với đích nhất. Đánh giá trạng thái hiện tại chỉ dựa trên heuristics.</li> <li>Các chiến lược heuristic được đề xuất: <ul style="list-style-type: none"> <li><b>Khoảng cách Euclide trong không gian 2 chiều:</b> <math display="block">h(n) = h(x, y)</math> <math display="block">= \sqrt{(x - Goal_x)^2 + (y - Goal_y)^2}</math> </li> <li><b>Khoảng cách Mahattan:</b> <math display="block">h(n) = h(x, y)</math> <math display="block">=  x - Goal_x  +  y - Goal_y </math> </li> </ul> </li> </ul>	<p><b>Thời gian:</b>  <math>O(b^m)</math>.  (Với heuristic đủ tốt, GBFS sẽ thực thi nhanh hơn BFS.)</p> <p><b>Không gian:</b>  <math>O(b^m)</math></p>	<ul style="list-style-type: none"> <li>Tính đầy đủ: Có nếu không gian tìm kiếm có giới hạn.</li> <li>Tính tối ưu: Không vì GBFS dựa hoàn toàn vào heuristics. Thuật toán sẽ không đảm bảo tìm được đường đi có chi phí tốt nhất.</li> </ul>

## 4.1. Euclidean Distance Heuristic



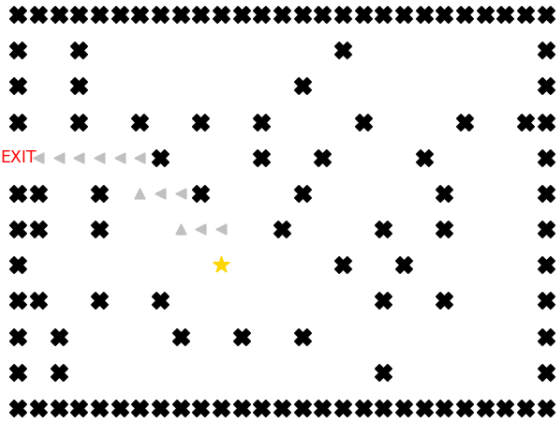
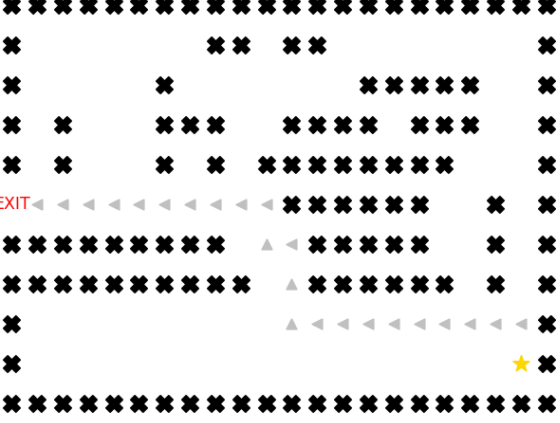
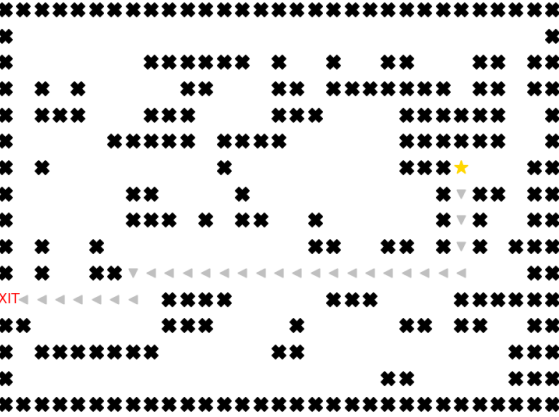

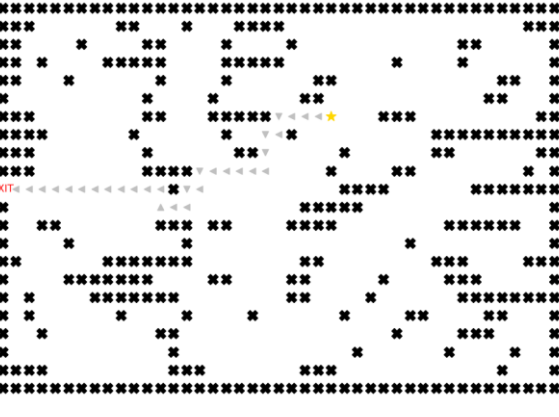
## 4.2. Mahattan Distance Heuristic



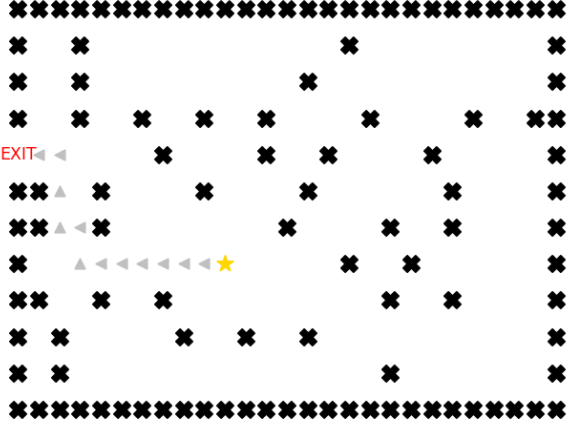
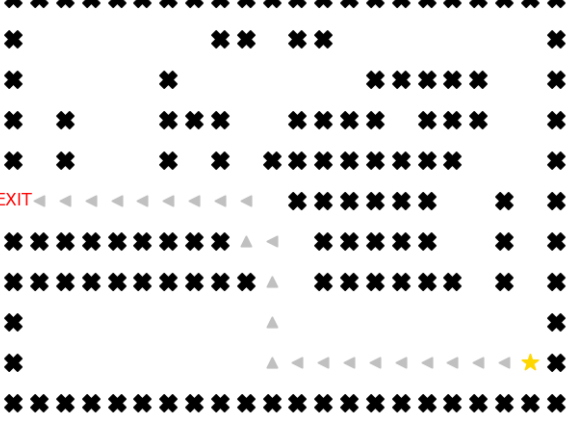
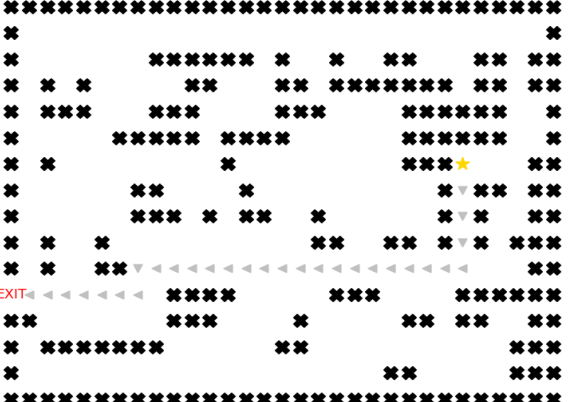
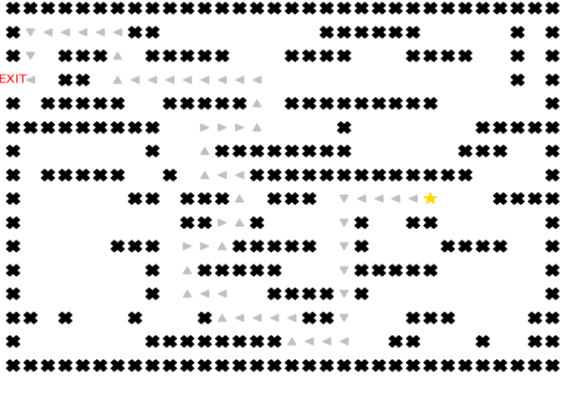
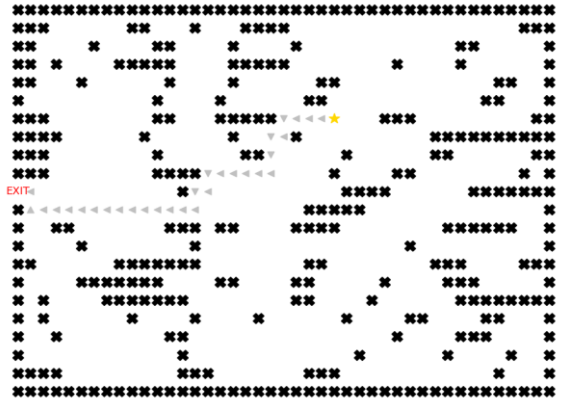
## 5. Thuật toán tìm kiếm A\*

Cài đặt	Độ phức tạp	Tính chất
<p>Sử dụng <b>hàng đợi ưu tiên</b> để lưu các ô đã mở xét với so sánh là <math>f(n) = g(n) + h(n)</math>.</p> <p>Hai heuristic được sử dụng để so sánh là <b>khoảng cách mahattan và diagonal</b>.</p> <p>Khởi tạo hàng đợi ưu tiên với phần tử đầu tiên là điểm S với chi phí là 0.</p> <p>Trong khi hàng đợi khác rỗng, Pop phần tử đầu của hàng đợi, kiểm tra có phải đích đến; kiểm tra tính hợp lệ của 4 ô liền kề, nếu đỉnh hợp lệ và có <math>f(n) &lt; f(\text{đỉnh đang xét})</math> thì Push đỉnh đó vào hàng đợi ưu tiên, lưu truy vết vào mảng Trace, cập nhật <math>f(n)</math>.</p> <p>Sau khi kiểm tra đủ 4 ô liền kề của ô đang xét thì đánh dấu ô đó là đã ghé để không đi trùng.</p> <p>Tiến hành truy vết tìm đường đi từ mảng Trace. .</p>	<p>(Độ phức tạp của A* dựa vào tính admissibility của heuristic)</p> <p><b>Thời gian:</b> Trường hợp tệ nhất là <math>O( E ) = O(b^d)</math> hay tốt nhất với <math>O(1)</math>.</p> <p><b>Không gian:</b> Tệ nhất là <math>O( V ) = O(bd)</math></p>	<p>A* là một giải thuật kết hợp giữa UCS (sử dụng <math>g(n)</math>) và GBFS (sử dụng <math>h(n)</math>), nên có thời gian thực hiện tốt hơn UCS và có tính hiệu quả, đảm bảo chi phí hơn so với GBFS.</p> <p>Tuy nhiên, vì đây là một giải thuật, nếu heuristic được sử dụng không hiệu quả, không có độ chính xác cao thì giải thuật A* sẽ trở nên không hiệu quả và đường đi tìm được có chi phí không tối ưu.</p> <p>A* là một giải thuật tốt để ứng dụng <b>tìm ra đường đi tốt, chứ không phải đường đi tối ưu</b>. Vì vậy, chi phí do A* tìm ra có tính chấp nhận được với thời gian chấp nhận được.</p>

## 5.1. Mahattan Distance

	
Map 1 (12x27) – Cost = 13	Map 2 (11x22) – Cost = 24
	
Map 3 (16x31) – Cost = 30	Map 4 (16x32) – Cost = 57
	
Map 5 (22x43) – Cost = 31	

## 5.2. Diagonal Distance

	
Map 1 (12x27) – Cost = 13	Map 2 (11x22) – Cost = 24
	
Map 3 (16x31) – Cost = 30	Map 4 (16x32) – Cost = 57
	
Map 5 (22x43) – Cost = 31	

## 6. Nhận xét các thuật toán:

Thuật toán BFS và A* cho ra 2 kết quả đường đi giống như nhau, là 2 thuật toán cho đường đi tối ưu nhất. Tuy nhiên BFS sẽ tốn chi phí thời gian hơn A*
UCS trên các bản đồ trên thì chi phí không khác gì BFS do chi phí đường đi là 1.
GBFS tuy cho hình dạng đường đi có khác biệt nhưng vẫn là đường đi có chi phí nhỏ nhất giống BFS và A* nếu ở trường hợp tốt nhất (Map 1,2,3,5). Nhưng nếu ở trường hợp như Map 4 thì thuật toán lộ rõ sự không tối ưu của mình.
DFS chỉ ra được 1 đường đi ra khỏi mê cung, nhưng đường đi này tốn chi phí hơn khá nhiều so với 4 thuật toán còn lại

### III. Bản đồ có điểm thưởng:

#### 1. Chiến lược đề xuất:

##### a. Phân tích bài toán:

- Với bản đồ không có điểm thưởng, thuật toán BFS cho ra đường đi có chi phí nhỏ nhất từ S  $\rightarrow$  G nhưng lại có độ phức tạp quá lớn, trong khi sử dụng A\* lại cho độ phức tạp nhỏ hơn nhưng có kết quả có chi phí đường đi tương đối nhỏ, thậm chí có thể nhỏ nhất. Việc xuất hiện các điểm thưởng (+) làm cho đường đi ngắn nhất không phải đường đi tối ưu nhất nếu vẫn theo heuristic cũ là khoảng cách từ điểm đang xét tới G. Đường đi ngắn nhất lúc này có thể là các đường đi qua các ô điểm thưởng S  $\rightarrow$  Bj  $\rightarrow$  ...  $\rightarrow$  Bj  $\rightarrow$  G và có thể lệch nhiều ra khỏi đường đi không có điểm thưởng cùng bản đồ.

##### b. Ý tưởng xử lý bài toán

- Thuật toán A\* có cách tìm kiếm đường đi tốt trong các thuật toán tìm kiếm hiện nay, và nguyên lý của nó có thể tái sử dụng cho cách giải quyết bài toán này. Xem bài toán là cách đi qua các cung từ điểm đang xét tới G. Đường đi nhỏ nhất là đường đi qua các điểm thưởng gần nhất với nó và gần với điểm G. Vậy bài toán từ việc tìm đường đi từ S  $\rightarrow$  G thành bài toán tìm B gần nhất và đi qua nó S  $\rightarrow$  B  $\rightarrow$  G.
- S  $\rightarrow$  G > S  $\rightarrow$  B  $\rightarrow$  G
- Nếu việc tìm kiếm điểm thưởng gần nhất là việc định hướng đường đi qua điểm thưởng thì xây dựng một hàm heuristic để định hướng và giúp thuật toán chọn được hướng đi là điều quan trọng. Và tất nhiên, đường đi sẽ được biến thành S  $\rightarrow$  Bi  $\rightarrow$  Bj  $\rightarrow$  G từ S  $\rightarrow$  Bi  $\rightarrow$  G nếu đường đi từ Bi  $\rightarrow$  G lớn hơn đường đi từ Bi  $\rightarrow$  Bj + Bj  $\rightarrow$  G. (đỉnh G ở đây là một đỉnh bất kỳ đang trong quá trình xét, chứ không phải luôn luôn là đỉnh Goal).
- Heuristic được sử dụng ở đây để định hướng thuật toán là khoảng cách từ điểm đang xét tới điểm thưởng gần nhất, vì thuật toán theo nguyên lý A\* sẽ tìm đi điểm có độ ưu tiên nhỏ nhất và đi càng gần điểm thưởng ta sẽ có điểm có độ ưu tiên càng nhỏ

hơn. Vì thế, đường đi sẽ dịch gần về điểm thưởng và đạt tới điểm thưởng trong khi vẫn đang tìm kiếm lối ra của mê cung.

- Chiến lược này tương tự với thuật toán A\*, nhưng khác ở chỗ chúng ta không hạn chế sự quay lại đỉnh đã đến trước đó, dẫn tới đường đi có thể chồng chéo lên nhau, nhưng sẽ có một chi phí tốt. Nhưng biến truy vết và hàng đợi chỉ thay đổi khi B  $\rightarrow$  G có sự thay đổi thì mới cần xét lại để cập nhật chứ không cần cập nhật liên tục, vì vậy sẽ làm giảm chi phí duyệt và tìm kiếm đường đi của bản đồ.

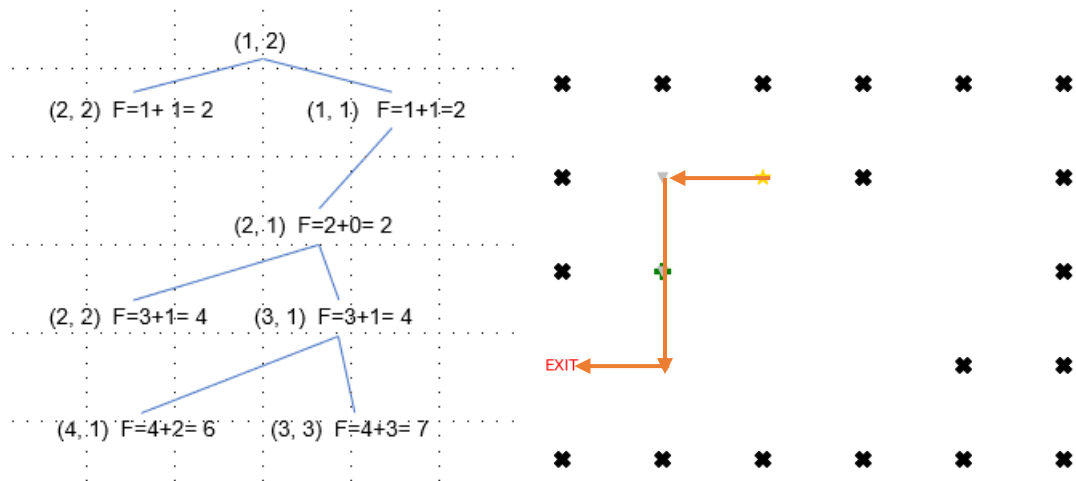
## 2. Mô tả các bước chạy bài toán:

- Bản đồ: 5 x 6
- Điểm thưởng: (2,1) = -4
- Chi phí: 0 so với 4 của bfs thông thường

```

xxxxxx
x 5x  x
x+   x
|   xx
xxxxxx

```

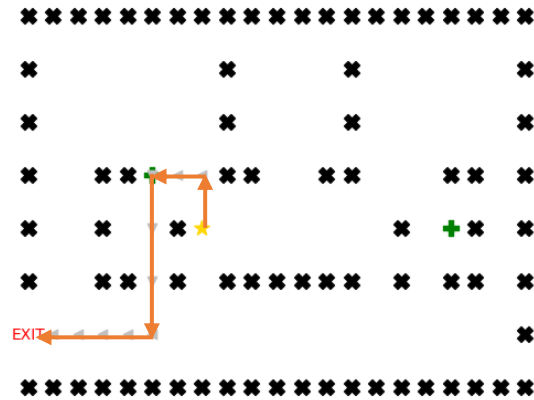


## 3. Kết quả chạy chương trình:

### a. Bản đồ 2 điểm thưởng:

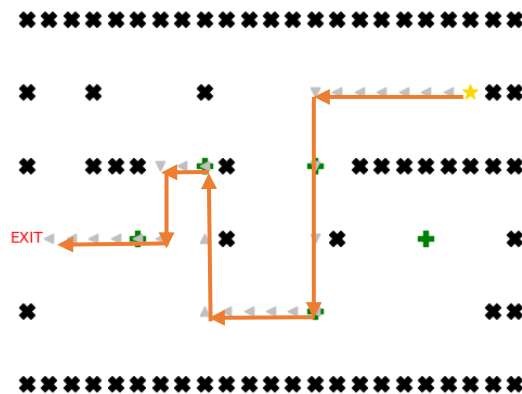
- Bản đồ: 8 x 21
- Điểm thưởng: (3,5) = -1; (4,17) = -5
- Chi phí: 10 so với 8 của bfs
- Thuật toán không hiệu quả lắm nếu giá trị điểm thưởng nhỏ và ít điểm thưởng





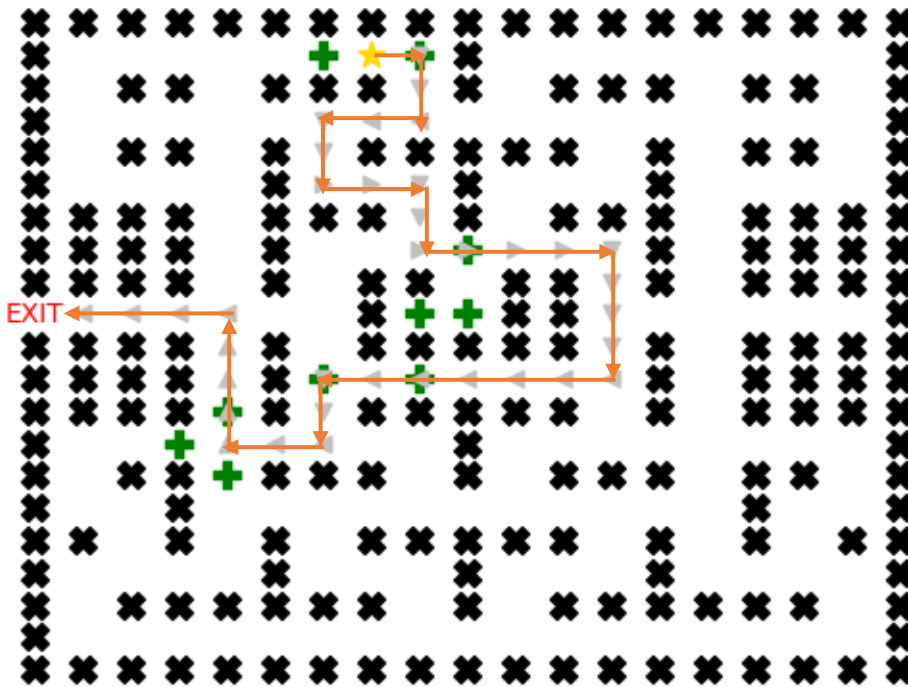
*b. Bản đồ 5 điểm thưởng*

- Bản đồ: 6 x 23
- Điểm thưởng:  $(2,8) = -1$ ;  $(2,13) = -5$ ;  $(3,5) = -2$ ;  $(3,18) = -7$ ;  $(4,13) = -3$
- Chi phí: 12 so với 23 của bfs
- Thuật toán hiệu quả hơn khi có nhiều điểm thưởng và giá trị điểm thưởng cao



*c. Bản đồ 10 điểm thưởng*

- Bản đồ: 21 x 18
- Điểm thưởng:  $(1,6) = -1$ ;  $(1,8) = -2$ ;  $(7,9) = -9$ ;  $(9,8) = -5$ ;  $(9,9) = -4$ ;  $(11,6) = -3$ ;  $(11,8) = -6$ ;  $(12,4) = -8$ ;  $(13,3) = -10$ ;  $(14,4) = -7$
- Chi phí: 5 so với 14 của bfs
- Thuật toán hiệu quả hơn khi có nhiều điểm thưởng và giá trị điểm thưởng cao



#### IV. Kịch bản nâng cấp:

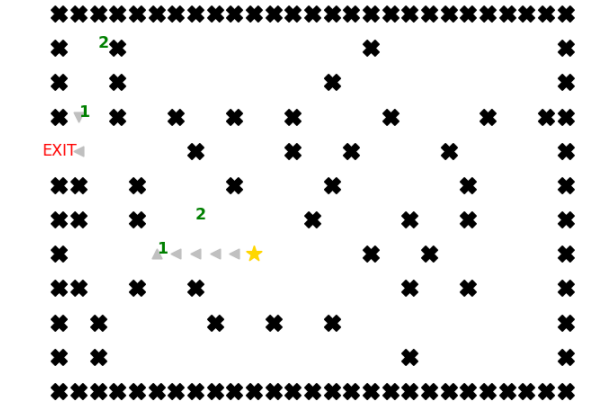
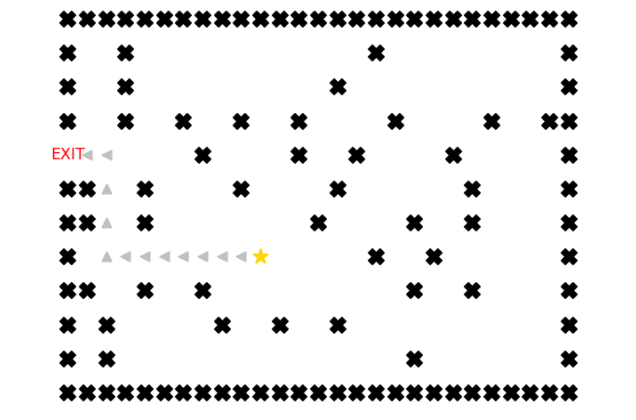
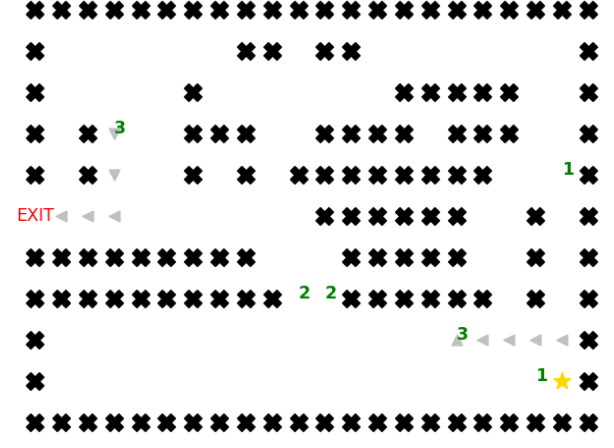
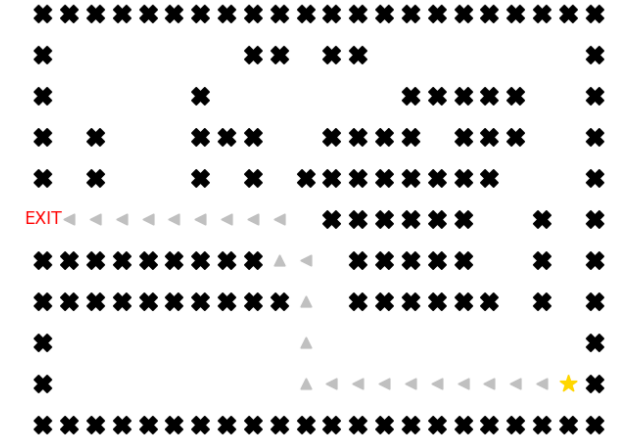
##### 1. Mô tả kịch bản



- Bản đồ sẽ có các ô có điểm dịch chuyển mà khi đi vào tác nhân sẽ được dịch chuyển tức thời sang ô có liên kết với nó với chi phí 1 tương tự như chi phí di chuyển thông thường (các điểm dịch chuyển là 2 chiều: nếu tác nhân dịch chuyển từ  $(x, y) \rightarrow (x', y')$  thì cũng có thể dịch chuyển từ  $(x', y') \rightarrow (x, y)$ ). Hai ô có liên kết dịch chuyển với nhau sẽ được đánh số thứ tự giống nhau tương ứng trên bản đồ.
- Tác nhân đi vào ô có điểm dịch chuyển sẽ lập tức bị dịch chuyển sang ô liên kết tương ứng. Tác nhân không được lựa chọn dịch chuyển hay không.

##### 2. Đề xuất thuật toán

Điểm khác của kịch bản này so với kịch bản chưa nâng cấp là đồ thị sẽ có thêm các cạnh  $(x, y) - (x', y')$ . Vì mục tiêu là tìm ra đường đi có chi phí ít nhất, tương đương với đường đi qua ít cạnh nhất nên chúng em đề xuất thuật toán BFS vì có khả năng tìm được đường đi ngắn nhất và tương đối dễ cài đặt.

### 3. Kết quả

	
Cost – Teleport = 6	Cost – BFS = 13
<p>Ở bản đồ đầu tiên này được thiết kế đơn giản để thử xem dịch chuyển tức thời có được thực hiện hay không, nên tác nhân vẫn đi theo đường cũ đã đi ở thuật toán tìm kiếm đường đi ngắn nhất.</p>	
	
Cost – Teleport = 9	Cost – BFS = 13
<p>Ở bản đồ này được thiết kế phức tạp hơn để thử xem thuật toán có thể tính toán được đi đến vùng dịch chuyển nào sẽ có lợi hơn.          Tác nhân đã đi lên phía trên để đến cửa dịch chuyển thứ 3 thay vì đi qua trái đến cửa 1 sẽ khiến tác nhân tốn chi phí đường đi hơn như ở bản đồ cũ.</p>	

	
Cost – Teleport = 22	Cost – BFS = 57
<p>Ở bản đồ đầu tiên này được thiết kế phức tạp hơn để thử xem thuật toán có thể tính toán được đi đến vùng dịch chuyển nào sẽ có lợi hơn.</p> <p>Tác nhân đã đi về phía bên phải để tìm đến cổng dịch chuyển 2 thì sẽ có một đường đi ngắn hơn. Vì nếu rẽ trái thì sẽ bị đẩy đến cổng 1 ở xa hơn rất nhiều với EXIT.</p>	

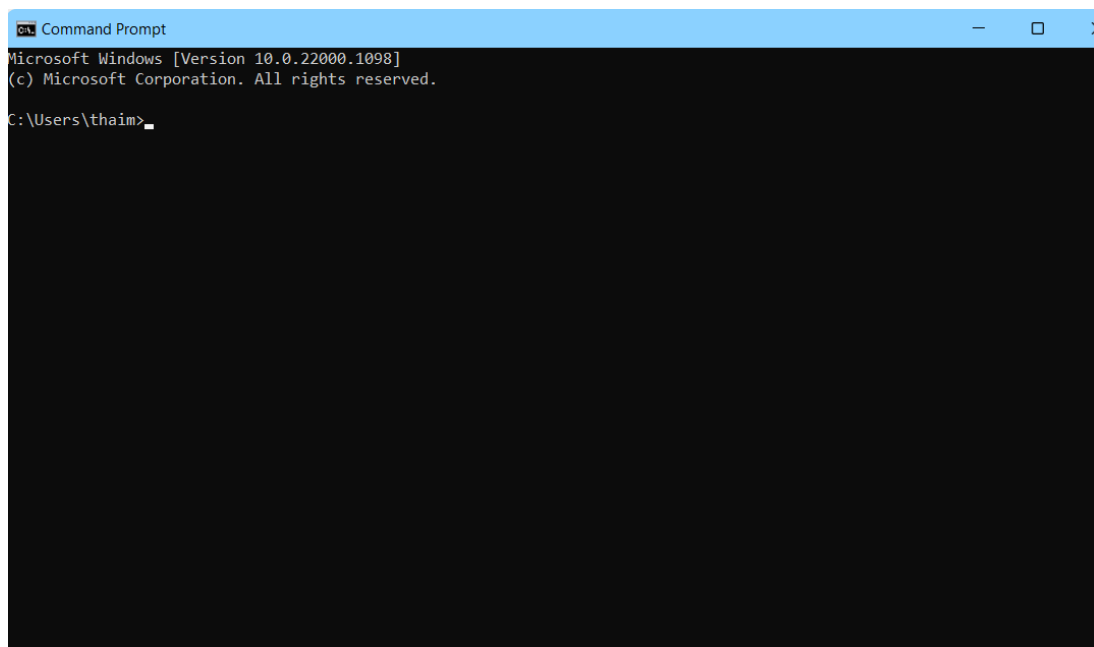
#### IV. Video minh họa:

##### 1. Cài đặt môi trường:

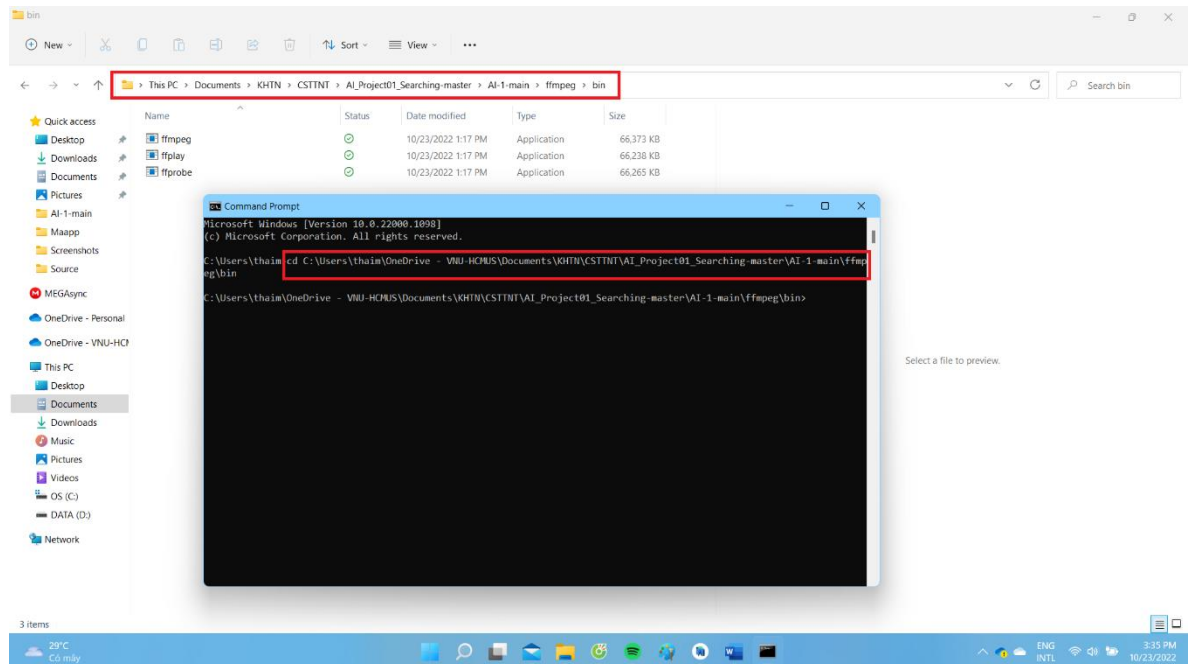
( Nếu không cài đặt được như bên dưới, có thể dùng tới:

[https://gist.github.com/tikitikipoo/2766152?fbclid=IwAR3w3Tal\\_fJj5adjV2H1huFOBp\\_wa5gZPVccF7KulRU2uOi3TrXVI52Ngtc#file-how\\_to\\_install\\_ffmpeg\\_souce](https://gist.github.com/tikitikipoo/2766152?fbclid=IwAR3w3Tal_fJj5adjV2H1huFOBp_wa5gZPVccF7KulRU2uOi3TrXVI52Ngtc#file-how_to_install_ffmpeg_souce))

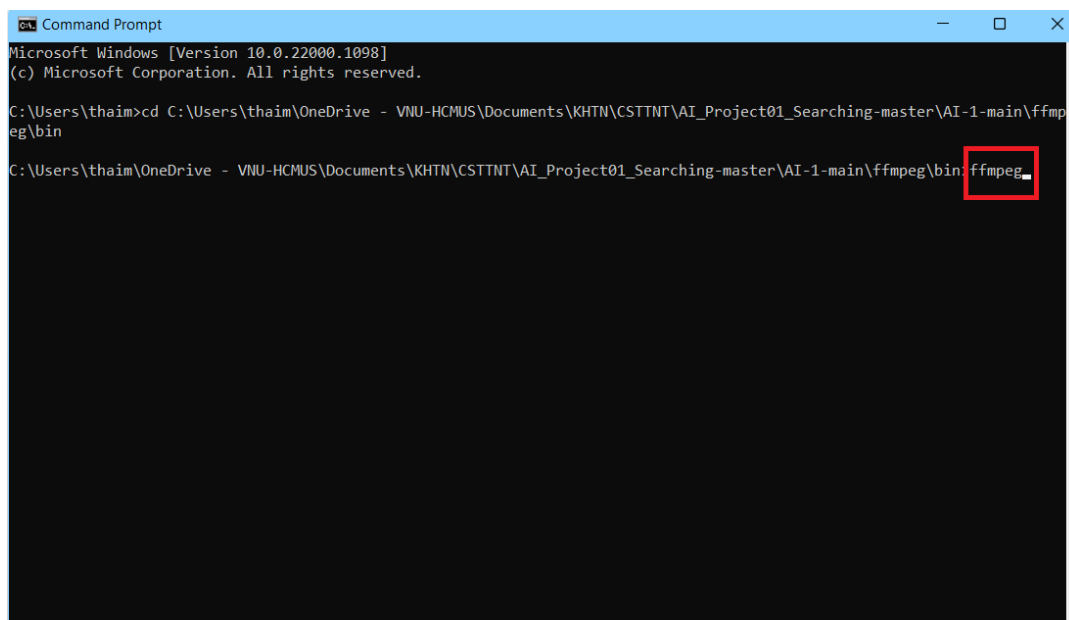
- Đầu tiên ta bật command prompt



- Trong các folder cùng cấp với folder input, có 1 folder tên **ffmpeg**. Chọn vào folder đó và chọn tiếp vào folder **bin** nằm bên trong. Tiến hành cd tới thư mục đó trong command prompt



- Chạy dòng lệnh ffmpeg và đợi chương trình được cài đặt



```

Command Prompt
(c) Microsoft Corporation. All rights reserved.

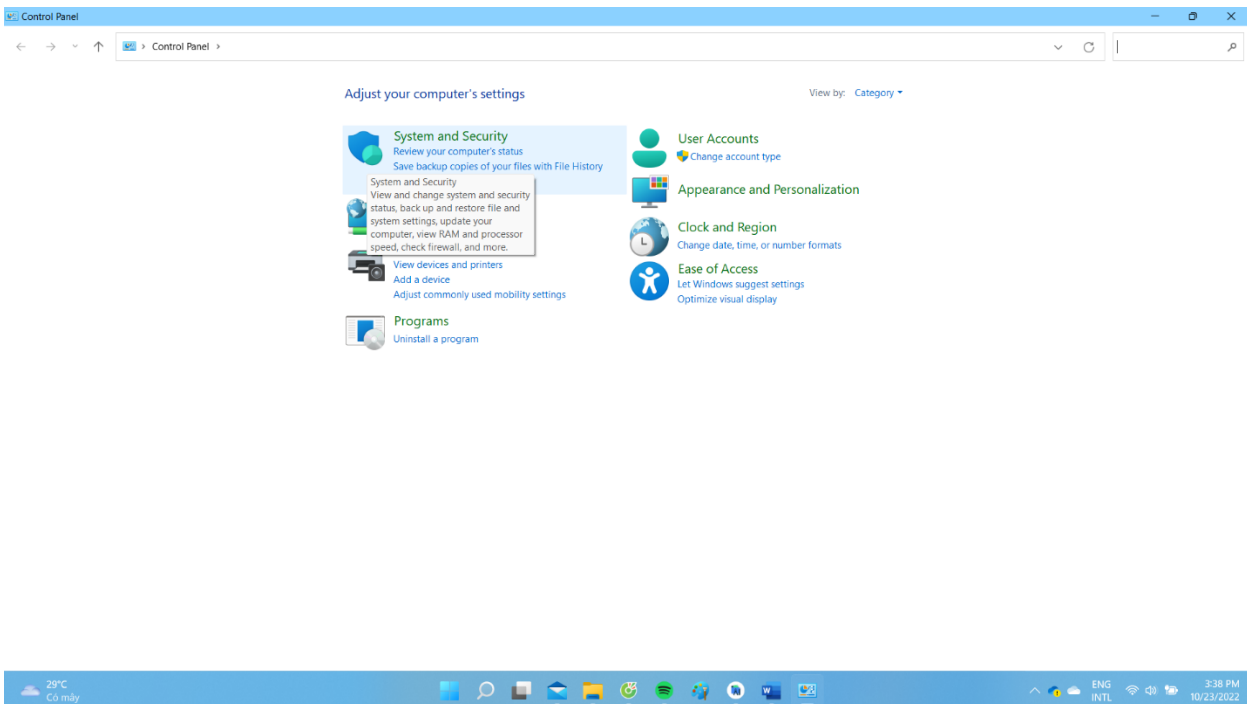
C:\Users\thaim>cd C:\Users\thaim\OneDrive - VNU-HCMUS\Documents\KHTN\CSTTNT\AI_Project01_Searching-master\AI-1-main\ffmpeg\bin

C:\Users\thaim\OneDrive - VNU-HCMUS\Documents\KHTN\CSTTNT\AI_Project01_Searching-master\AI-1-main\ffmpeg\bin>ffmpeg
ffmpeg version git-2020-04-26-1128aa8 Copyright (c) 2000-2020 the FFmpeg developers
  built with gcc 9.3.1 (GCC) 20200328
  configuration: --enable-gpl --enable-version3 --enable-sdl2 --enable-fontconfig --enable-gnutls --enable-iconv --enable-libass --enable-libdav1d --enable-libbluray --enable-libfreetype --enable-libmp3lame --enable-libopencore-amrnb --enable-libopencore-amrwb --enable-libopenjpeg --enable-libopus --enable-libshine --enable-lsnpappy --enable-libsoxr --enable-libsrtp --enable-libtheora --enable-libtwolame --enable-libvpx --enable-libwavpack --enable-libwebp --enable-libx264 --enable-libx265 --enable-libxml2 --enable-libzimg --enable-lzma --enable-zlib --enable-gmp --enable-libvidstab --enable-libvmaf --enable-libvorbis --enable-libvo-amrwbenc --enable-libmysofa --enable-lispeex --enable-libxvid --enable-libaom --enable-disable-w32threads --enable-libmfx --enable-ffnvcodec --enable-cuda-llvm --enable-cuvid --enable-d3d11va --enable-nvdec --enable-nvdec --enable-dxva2 --enable-avisynth --enable-libopenmpt --enable-amf
  libavutil      56. 43.100 / 56. 43.100
  libavcodec     58. 82.100 / 58. 82.100
  libavformat    58. 42.101 / 58. 42.101
  libavdevice    58.  9.103 / 58.  9.103
  libavfilter    7. 79.100 / 7. 79.100
  libswscale     5.  6.101 / 5.  6.101
  libswresample  3.  6.100 / 3.  6.100
  libpostproc   55.  6.100 / 55.  6.100
  hyper fast Audio and Video encoder
  Usage: ffmpeg [options] [[infile options] -i infile]... {[outfile options] outfile}...
  Use -h to get full help or, even better, run 'man ffmpeg'

C:\Users\thaim\OneDrive - VNU-HCMUS\Documents\KHTN\CSTTNT\AI_Project01_Searching-master\AI-1-main\ffmpeg\bin>

```

- Vào Control Panel -> System and Security -> System -> Advanced system settings -> Enviroment Variables



The image shows two screenshots from a Windows 11 desktop. The top screenshot is the 'System and Security' Control Panel window. The bottom screenshot is the 'System > About' settings window.

**Top Screenshot: System and Security Control Panel**

- Control Panel Home**
- System and Security**
  - Network and Internet
  - Hardware and Sound Programs
  - User Accounts
  - Appearance and Personalization
  - Clock and Region
  - Ease of Access
- Security and Maintenance**
  - Review your computer's status and resolve issues
  - Change User Account Control settings
  - Troubleshoot common computer problems
- Windows Defender Firewall**
  - Check firewall status
  - Allow an app through Windows Firewall
- System**
  - View System
  - View information about your computer, and change settings for hardware, performance, and remote connections
  - Allow remote access
  - Launch remote assistance
- Power**
  - Change what the power buttons do
  - Change when the computer sleeps
- File History**
  - Save backup copies of your files with File History
  - Restore your files with File History
- Backup and Restore (Windows 7)**
  - Backup and Restore (Windows 7)
  - Restore files from backup
- Storage Spaces**
  - Manage Storage Spaces
- Work Folders**
  - Manage Work Folders
- Windows Tools**
  - Free up disk space
  - Defragment and optimize your drives
  - Create and format hard disk partitions
  - View event logs
  - Schedule tasks
- SupportAssist OS Recovery**

**Bottom Screenshot: System > About Settings**

**System > About**

**Vy Foxy**  
Inspiron 7591

**Device specifications**

Device name	Vy Foxy
Processor	Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz 2.40 GHz
Installed RAM	8.00 GB (7.79 GB usable)
Device ID	DBE42D66-527B-4634-A47B-BBC4338A0433
Product ID	00327-35860-34058-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

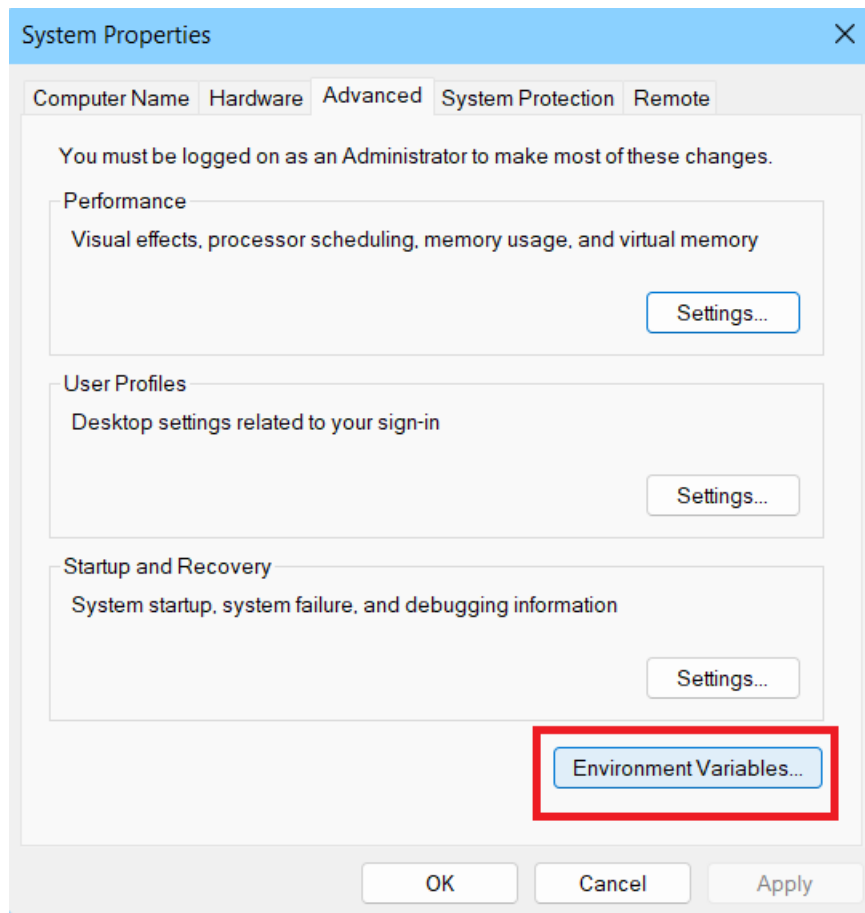
**Related links** Domain or workgroup System protection **Advanced system settings**

**Windows specifications**

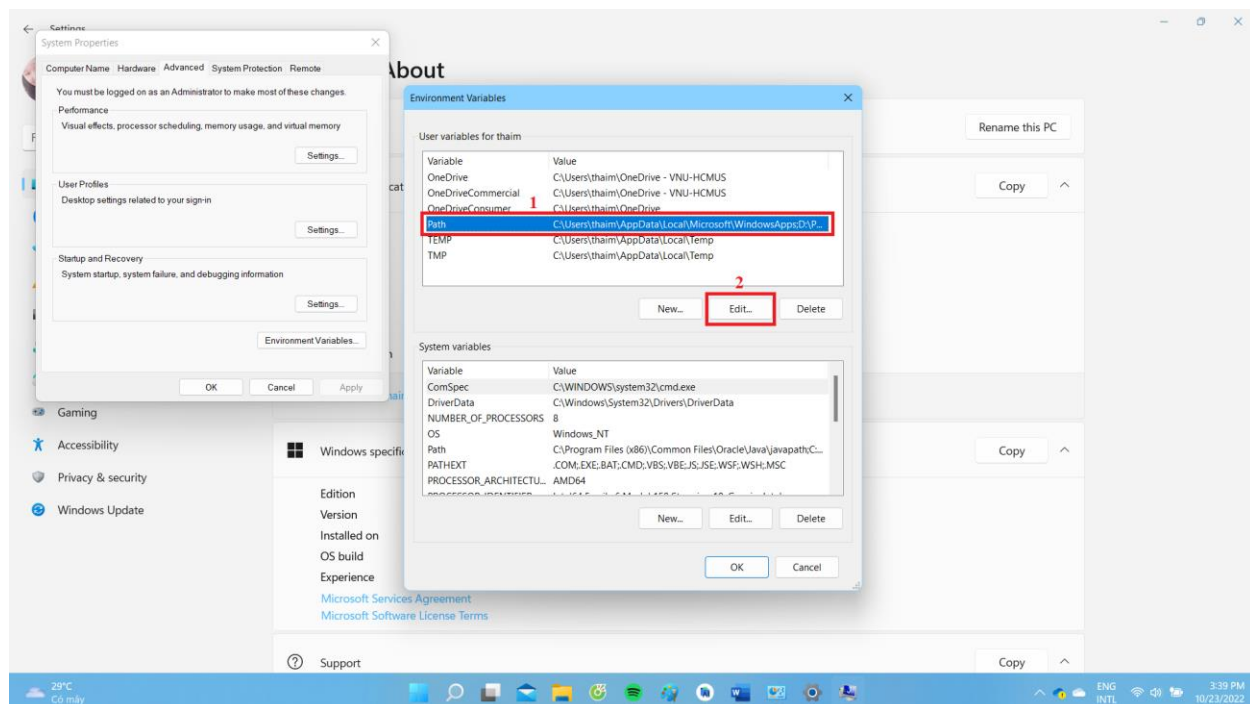
Edition	Windows 11 Home Single Language
Version	21H2
Installed on	11/15/2021
OS build	22000.1098
Experience	Windows Feature Experience Pack 1000.22000.1098.0

[Microsoft Services Agreement](#)  
[Microsoft Software License Terms](#)

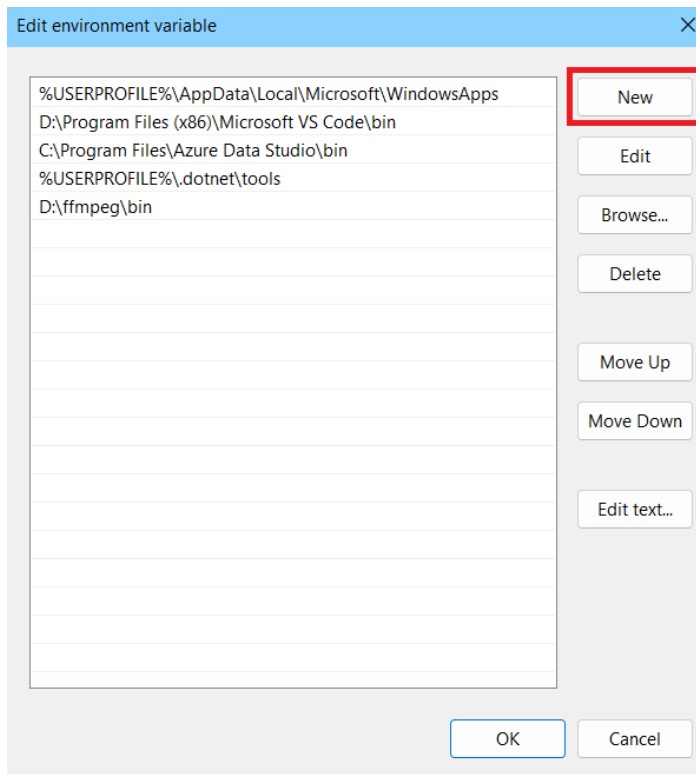
**Support**



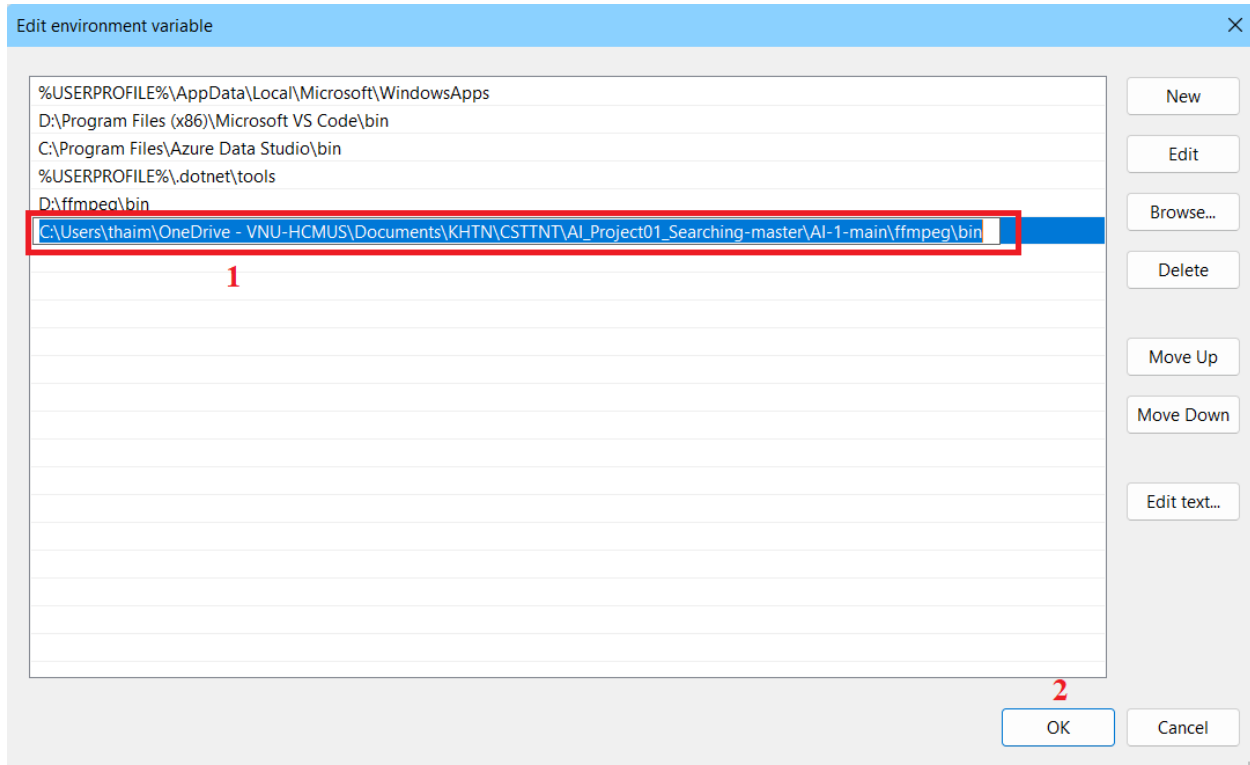
– Chọn Path -> Edit





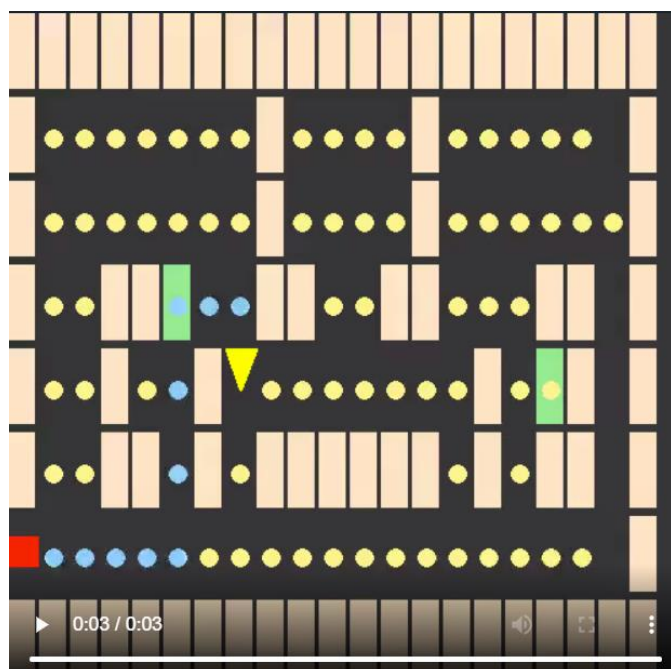








– Chọn new, sau đó Paste thư mục ffmpeg/bin vào.

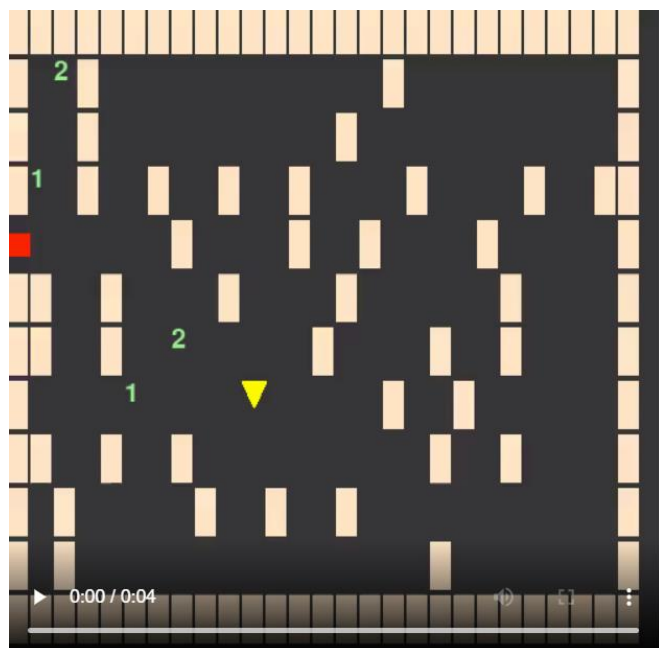


## 2. Chú thích video minh họa:

**VIDEO** chỉ mở được trên một số phần mềm khả dụng. Có thể mở bằng **Media Player Classic, Visual Studio Code**.



-  : Các cạnh của bản đồ. Tương ứng với dấu “x”
-  : Các đỉnh đã được mở trong quá trình tìm kiếm
-  : Đường đi cuối cùng tìm kiếm ra.
-  : Điểm bắt đầu. Tương ứng với “S”
-  : Điểm kết thúc. Tương ứng với “EXIT”
-  : Các ô chứa điểm thưởng. Tương ứng với dấu “+”



Các cặp số **1.1, 2..2** trên bản đồ thể hiện công dịch chuyển 2 chiều. Hai ô có số thứ tự giống nhau sẽ dịch chuyển được tới nhau,