

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN MÔN HỌC

CƠ SỞ TRÍ TUỆ NHÂN TẠO

|  |
| --- |
| THUẬT TOÁN TÌM KIẾM |

***Trợ giảng:***

Nguyễn Duy Khánh duykhanhnguyen360@gmail.com

Nguyễn Ngọc Băng Tâm

bangtamnguyenn@gmail.com

**Thông tin nhóm**

|  |  |  |
| --- | --- | --- |
| MSSV | Họ và tên | Email |
| 1712809 | Nguyễn Gia Thụy | 1712809@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).

Thuật toán tìm kiếm UCS (Uniform-Cost 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.

- Chi phí di chuyển từ 1 ô tới 1 ô lân cận trong mê cung là 1.

- Các ô đã được mở sẽ không được mở lại.

- Mỗi ô chỉ di chuyển qua 1 lần.

Mục lục

[**I. Tổng quan các thuật toán và thuật giải** 4](#_Toc117446563)

[1. Thuật toán 4](#_Toc117446564)

[2. Thuật giải 4](#_Toc117446565)

[**II. Áp dụng tìm đường trong mê cung** 5](#_Toc117446566)

[**1. Mê cung không có điểm thưởng** 5](#_Toc117446567)

[1.1. Mê cung 1 (9 x 12) 5](#_Toc117446568)

[1.2. Mê cung 2 (9x12) 7](#_Toc117446569)

[1.3. Mê cung 3 (9 x 12) 9](#_Toc117446570)

[1.4. Mê cung 4 (9 x 50) 11](#_Toc117446571)

[1.5. Mê cung 5 (17 x 14) 13](#_Toc117446572)

[**2. Mê cung có điểm thưởng** 16](#_Toc117446573)

[2.1. Mê cung 1 (5 x 22) 16](#_Toc117446574)

[2.2. Mê cung 2 (11 x 22) 16](#_Toc117446575)

[2.3. Mê cung 3 (11 x 22) 17](#_Toc117446576)

[**3. Mê cung có cổng dịch chuyển** 18](#_Toc117446577)

[3.1. Mê cung 1 (5 x 22) 18](#_Toc117446578)

[3.2. Mê cung 2 (5 x 22) 20](#_Toc117446579)

**PHÂN TÍCH THUẬT TOÁN VÀ THUẬT GIẢI**

# **I. Tổng quan các thuật toán và thuật giải**

## 1. Thuật toán

|  |  |  |
| --- | --- | --- |
| **Depth First Search** | **Breadth First Search** | **Uniform-Cost 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 do sử dụng thêm hàm tính chi phí tính đếm thời điểm hiện tại, luôn mở rộng không gian tìm kiếm theo chiều rộng (loang ra nhiều phía), dựa trên chi phí tính toán được.  - 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 do tính được chi phí đường đi. |

## 2. Thuật giải

### 2.1. Các thuật giải sử dụng

|  |  |  |
| --- | --- | --- |
| **Greedy Best First Search** | **A\* Search** | **Algo1 – Bonus-Oriented Search** |
| - Thuật giải là phiên bản cải tiến của DFS với khả năng tối ưu đường đi theo một hàm ước lượng nào đó. Thuật giải sẽ sử dụng thuật giải để sắp xếp lại các ô trong hàng đợi sau mỗi lần mở rộng, từ đó làm thay đổi thứ tự duyệt các ô.  - Khả năng ước lượng của thuật giải phụ thuộc vào hàm ước lượng được sử dụng. Tùy vào cấu trúc của không gian tìm kiếm mà mỗi hàm ước lượng sẽ phát huy được thế mạnh riêng. Đối với đồ án này, việc tìm được đường đi ngắn nhất là hoàn toàn ngẫu nhiên. | - Thuật giải là phiên bản kết hợp giữa Greedy Best First Search và UCS. Cụ thể là bên cạnh sử dụng hàm ước lượng, thuật giải còn sử dụng thêm hàm để tính chi phí từ đầu đến ô hiện tại. Từ đó sẽ đưa ra phương án chính xác hơn cho đường đi tối ưu.  - Khả năng ước lượng của thuật giải phụ thuộc vào hàm ước lượng được sử dụng. Tùy vào cấu trúc của không gian tìm kiếm mà mỗi hàm ước lượng sẽ phát huy được thế mạnh riêng. Luôn tìm ra đường đi tối ưu do mang thế mạnh của UCS. | - Thuật giải là sự kết hợp giữa A\* và nguyên lý luôn hướng tới những cụm điểm có giá trị cao nhất trong quá trình di chuyển.  - Thuật giải đầu tiên sẽ tính vector hướng từ vị trí hiện tại đến cụm điểm có giá trị cao nhất. Tiếp theo thuật giải sẽ tính vector hướng từ vị trí hiện tại đến vị trí đích. Cuối cùng, thuật giải sẽ thực hiện tính vector tổng của 2 vectors trên và lấy nửa độ dài của vector tổng đó, kết hợp thêm hàm tính chi phí tính tới thời điểm hiện tại để làm giá trị ước lượng.  - Thuật giải không đảm bảo đường đi ngắn nhất và cũng không đảm bảo số điểm cao nhất. Thuật giải cố gắng cân bằng giữa điểm số và đường đi ở mức chấp nhận được. |

### 2.2. Các hàm ước lượng

|  |
| --- |
| *3.1. 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.  \* Lưu ý: Để chạy cần thực hiện lệnh python main.py <heuristic> manhattan |
| *3.3. 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.  \* Lưu ý: Để chạy cần thực hiện lệnh python main.py <heuristic> euclid |
| *3.3. Hàm ước lượng tệ*   * + - * Giải thích:   - Thuật giải sử dụng hàm thuật giải:  function heuristic(node) =    topleft = sqrt(node.x \* node.x + node.y \* node.y)    topright = sqrt(node.x \* node.x + (matrix.size – node.y) \* (matrix.size – node.y))  bottomleft = sqrt((matrix.size – node.x – 1) \* (matrix.size – node.x – 1) + node.y \* node.y)  bottomright = sqrt((matrix.size – node.x – 1) \* (matrix.size – node.x – 1)  + (matrix.size – node.y – 1) \* (matrix.size – node.y – 1))    return max(topleft, topright, bottomleft, bottomright)  - Một ví dụ cho hàm ước lượng tệ.  \* Lưu ý: Để chạy cần thực hiện lệnh python main.py <heuristic> bad |

# **II. Áp dụng tìm đường trong mê cung**

## **1. Mê cung không có điểm thưởng**

### 1.1. Mê cung 1 (9 x 12)

**Xét các thuật toán:**

|  |  |
| --- | --- |
| *1.1a. Thuật toán DFS*   * Chi phí: 16 * Số ô mở: 16 | Hình 1.1a. Mê cung 1 – DFS |
| *1.1b. Thuật toán BFS*   * Chi phí: 16 * Số ô mở: 70 | Hình 1.1b. Mê cung 1 – BFS |
| *1.1c. Thuật toán Greedy Best First Search*   * Chi phí: 16 * Số ô mở: 16 | Hình 1.1c. Mê cung 1 – Greedy Best First Search |
| *1.1d. Thuật toán A\* Search*   * Chi phí: 16 * Số ô mở: 70 | Hình 1.1d. Mê cung 1 – A\* Search |

### 1.2. Mê cung 2 (9x12)

**Xét các thuật toán:**

|  |  |
| --- | --- |
| *1.2a. Thuật toán DFS*   * Chi phí: 65 * Số ô mở: 70 | Hình 1.2a. Mê cung 2 – DFS |
| *1.2b. Thuật toán BFS*   * Chi phí: 7 * Số ô mở: 35 | Hình 1.2b. Mê cung 2 – BFS |
| *1.2c. Thuật toán Greedy Best First Search*   * Chi phí: 7 * Số ô mở: 7 | Hình 1.2c. Mê cung 2 – Greedy Best First Search |
| *1.2d. Thuật toán A\* Search*   * Chi phí: 7 * Số ô mở: 7 | Hình 1.2d. Mê cung 2 – A\* Search |

### 1.3. Mê cung 3 (9 x 12)

**Xét các thuật toán:**

|  |  |
| --- | --- |
| *1.3a. Thuật toán DFS*   * Chi phí: “no” * Số ô mở: 69 | Hình 1.3a. Mê cung 3 – DFS |
| *1.3b. Thuật toán BFS*  Chi phí: “no”  Số ô mở: 69 | Hình 1.3b. Mê cung 3 – BFS |
| *1.3c. Thuật toán Greedy Best First Search*  Chi phí: “no”  Số ô mở: 0 | Hình 1.3c. Mê cung 3 – Greedy Best First Search |
| *1.3d. Thuật toán A\* Search*  *Chi phí: “no”*  *Số ô mở: 0* | Hình 1.3d. Mê cung 3 – A\* Search |

### 1.4. Mê cung 4 (9 x 50)

**Xét các thuật toán:**

|  |
| --- |
| *1.4a. Thuật toán DFS*  Chi phí: 123  Số ô mở: 178    Hình 1.4a. Mê cung 4 – DFS |
| *1.4b. Thuật toán BFS*  Chi phí: 77  Số ô mở: 239    Hình 1.4b. Mê cung 4 – BFS |
| *1.4c. Thuật toán Greedy Best First Search*  Chi phí: 81  Số ô mở: 127    Hình 1.4c. Mê cung 4 – Greedy Best First Search |
| *1.4d. Thuật toán A\* Search*  *Chi phí: 77*  *Số ô mở: 163*    Hình 1.4d. Mê cung 4 – A\* Search |

### 1.5. Mê cung 5 (17 x 14)

**Xét các thuật toán:**

|  |  |
| --- | --- |
| *1.5a. Thuật toán DFS*  Chi phí: 23  Số ô mở: 24 | Hình 1.5a. Mê cung 5 – DFS |
| *1.5b. Thuật toán BFS*  Chi phí: 23  Số ô mở: 49 | Hình 1.5b. Mê cung 5 – BFS |
| *1.5c. Thuật toán Greedy Best First Search*  Chi phí: 23  *Số ô mở: 27* | Hình 1.5c. Mê cung 5 – Greedy Best First Search |
| *1.5d. Thuật toán A\* Search*  Chi phí: 23  *Số ô mở: 29* | Hình 1.5d. Mê cung 5 – A\* Search |

## **2. Mê cung có điểm thưởng**

- Quy ước:

+ Các điểm thưởng có ký hiệu là dấu “+”

+ Giá trị theo màu R, G, B lần lượt là 10, 5, 2 (Giá trị khi tính toán là -10, -5, -2)

### 2.1. Mê cung 1 (5 x 22)

**Xét các thuật toán:**

|  |
| --- |
| *1.1a. Thuật toán Algo1*   * Chi phí: 22 * Số ô mở: 60 * Số điểm: 12     Hình 1.1a. Mê cung 1 – Algo1 |

### 2.2. Mê cung 2 (11 x 22)

**Xét các thuật toán:**

|  |
| --- |
| *1.1a. Thuật toán Algo1*   * Chi phí: 29 * Số ô mở: 79 * Số điểm: 10     Hình 1.1a. Mê cung 1 – Algo1 |

### 2.3. Mê cung 3 (11 x 22)

**Xét các thuật toán:**

|  |
| --- |
| *1.1a. Thuật toán Algo1*   * Chi phí: 31 * Số ô mở: 129 * Số điểm: 20     Hình 1.1a. Mê cung 1 – Algo1 |

## **3. Mê cung có cổng dịch chuyển**

- Quy ước:

+ Các cổng dịch chuyển có kí hiệu là wn với n = 1, 2, … (cổng ra và cổng vào đều chung 1 kí hiệu)

+ Các cổng dịch chuyển cho phép dịch chuyển 2 chiều, tức là (x, y) sang (x’, y’) và ngược lại

### 3.1. Mê cung 1 (5 x 22)

**Xét các thuật toán:**

|  |
| --- |
| *3.1a. Thuật toán DFS*   * Chi phí: 22 * Số ô mở: 22     Hình 1.1a. Mê cung 1 – DFS |
| *3.1a. Thuật toán BFS*   * Chi phí: 13 * Số ô mở: 58     Hình 1.1a. Mê cung 1 – BFS |
| *3.1a. Thuật toán UCS*   * Chi phí: 13 * Số ô mở: 58     Hình 1.1a. Mê cung 1 – UCS |
| *3.1a. Thuật toán Greedy Best First Search*   * Chi phí: 13 * Số ô mở: 13     Hình 1.1a. Mê cung 1 – DFS |
| *3.1a. Thuật toán A\**   * Chi phí: 13 * Số ô mở: 14     Hình 1.1a. Mê cung 1 – DFS |

### 3.2. Mê cung 2 (5 x 22)

**Xét các thuật toán:**

|  |
| --- |
| *3.2a. Thuật toán DFS*   * Chi phí: 56 * Số ô mở: 83     Hình 1.1a. Mê cung 2 – DFS |
| *3.2a. Thuật toán BFS*   * Chi phí: 8 * Số ô mở: 21     Hình 1.1a. Mê cung 2 – DFS |
| *3.2a. Thuật toán UCS*   * Chi phí: 8 * Số ô mở: 21     Hình 1.1a. Mê cung 2 – UCS |
| *3.2a. Thuật toán Greedy Best First Search*   * Chi phí: 25 * Số ô mở: 28     Hình 1.1a. Mê cung 2 – DFS |
| *3.2a. Thuật toán A\**   * Chi phí: 8 * Số ô mở: 11     Hình 1.1a. Mê cung 2 – DFS |

**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](https://en.wikipedia.org/wiki/A*_search_algorithm)

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

[shorturl.at/ezJWY](https://www.youtube.com/watch?v=D5aJNFWsWew&list=PLBw9d_OueVJS_084gYQexJ38LC2LEhpR4&index=1&ab_channel=CS50)