TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP GIỮA KỲ**

**MÔN NHẬP MÔN TRÍ TUỆ NHÂN TẠO**

*Người thực hiện*: **HUỲNH THÀNH TRUNG – 51800145**

Lớp **: 18050203**

Khoá  **: 13**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP GIỮA KỲ**

**MÔN NHẬP MÔN TRÍ TUỆ NHÂN TẠO**

*Người thực hiện*: **HUỲNH THÀNH TRUNG – 51800145**

Lớp **: 18050203**

Khoá  **: 13**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021**

PHẦN ĐÁNH GIÁ CỦA GIẢNG VIÊN

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**MỤC LỤC**

***Trang***

**Bài 1- Travelling in Romania**

1.1 Mô tả cấu trúc dữ liệu

1.2 Giải thuật Greedy Best First Search (Heuristic)

1.2.1 Sơ đồ giải thuật

1.2.2 Hiện thực

* + 1. Kết quả và thảo luận

1.3 Giải thuật A\*

1.3.1 Sơ đồ giải thuật

1.3.2 Hiện thực

1.3.3 Kết quả và thảo luận

**Bài 2- Islands**

* 1. Mô tả cấu trúc dữ liệu

2.2 Giải thuật BFS – Breadth First Search

2.2.1 Sơ đồ giải thuật

2.2.2 Hiện thực

* + 1. Kết quả và thảo luận

2.3 Giải thuật UCS- Uniform Cost Search

2.3.1 Sơ đồ giải thuật

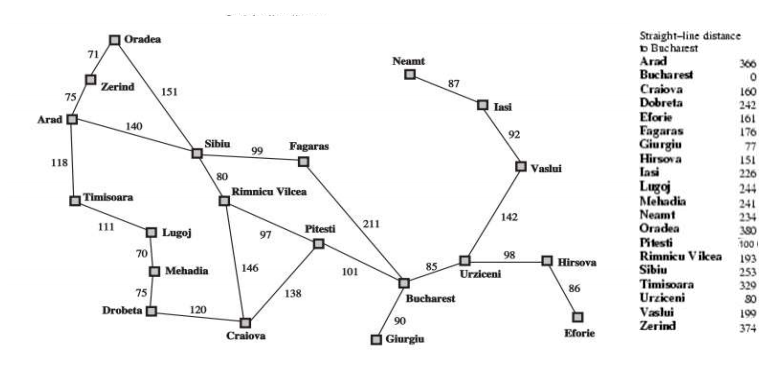
2.3.2 Hiện thực

2.3.3 Kết quả và thảo luận

**Tự đánh giá**

**BÀI 1- TRAVELLING IN ROMANIA**

* 1. Mô tả cấu trúc dữ liệu
     1. Bản đồ Romania và bảng giá trị trọng số Straight-line to Bucharest



* Bản đồ Romani được thiết kế theo dạng Graph với cấu trúc dữ liệu là Dictionary. Bên trong chứa các phần tử con với key là tên thành phố, value là các một List chứa đường đi đến các thành phố khác và trọng số Straight-line to Bucharest,
* Giá trị của các List được cấu tạo: phần tử cuối cùng của mảng là trong số Straight-line to Bucharest, mỗi phần tử còn lại là một List với phần tử 0 là tên của thành phố đến, phần tử 1 là khoảng cách đến thành phố đó.
* Ví dụ:

Romania = {“Oradea” : [[“Zerind”,71] , [“Sibiu”,151] , 380],

“Zerind” : [[“Oradea”,71] , [“Arad”,75] , 374],

“Sibiu” : [[“Oradea”,151] , [“Arad”,140] , [“Fagaras”,99] , [“Rimnicu Vilcea”,80] , 253],

“Arad” : [[“Zerind”,75] , [“Timisoara”,118] , [“Sibiu”,140] , 366],

…..}

Xét Romania:

+ Phần tử con “Oradea” có đường đi đến “Zerind” là 71, đến “Sibiu” là 151 và giá trị Straight-line to Bucharest là 380.

+ Phần tử con “Zerind” có đường đi đến “Oradea” là 71, đến “Arad” là 75 và giá trị Straight-line to Bucharest là 371.

+ Phần tử con “Sibiu” có đường đi đến “Oradea” là 151, đến “Arad” là 140, đến “Fagaras” là 99, đến “Rimnicu Vilcea” là 80 và giá trị Straight-line to Bucharest là 253

+ Phần tử con “Arad” có đường đi đến “Zerind” là 75, đến “Timisoara” là 118, đến “Sibiu” là 140 và giá trị Straight-line to Bucharest là 366

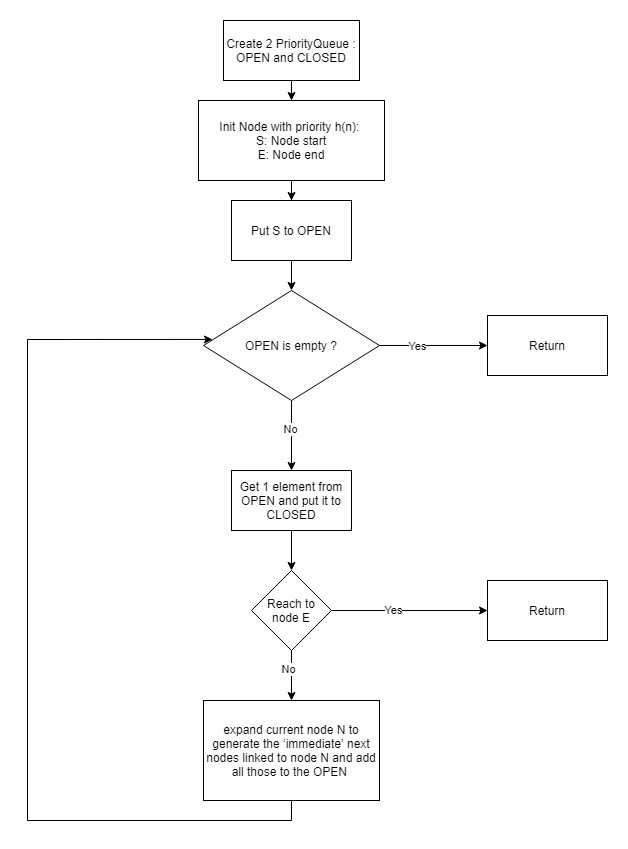
* + 1. Giải thuật Greedy Best First Search (Heuristic) và A\*
* Cả 2 giải thuật đều sữ dụng PriorityQueue để giải quyết vấn đề, cụ thể là 2 PriorityQueue OPEN và CLOSED để chứa các node con trong quá trình duyệt graph.
* Priority đối với giải thuật Greedy Best First Search (Heuristic) là hàm đánh giá h(n) đã được cung cấp sẵn đó là giá trị Straight-line to Bucharest của node bắt đầu đến node đích.
* Priority đối với giải thuật A\* là hàm đánh giá f(n) = h(h) + g(n) trong đó h() là giá trị Straight-line to Bucharest, g(n) là khoảng cách từ node bắt đầu đến node hiện tại.
* Node sữ dụng trong 2 giải thuật trên được xây dựng như sau :

+ Greedy Best First Search (Heuristic) node có 3 thuộc tính là node : tên của đỉnh hiện tại, par : tên của đỉnh cha của đỉnh hiện tại, h: giá trị Straight-line to Bucharest

+ A\* node có 4 thuộc tính trong đó 3 thuộc tính đầu giống với giải thuật Heuristic, thuộc tính còn lại là g chứa khoảng cách từ node bắt đầu đến node hiện tại.

1.2 Giải thuật Greedy Best First Search (Heuristic)

1.2.1 Sơ đồ giải thuật



* + 1. Hiện thực

-Chương trình tìm đường đi từ Arad đến Bucharest

+ Lần lặp 1 :

|  |  |
| --- | --- |
| **OPEN** | |
| Node | h(n) |
| Arad | 366 |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **CLOED** | |
| Node | Parent |
|  |  |
|  |  |
|  |  |

+ Lần lặp 2:

|  |  |
| --- | --- |
| **OPEN** | |
| Node | h(n) |
| Sibiu | 253 |
| Timisoara | 329 |
| Zerind | 374 |

|  |  |
| --- | --- |
| **CLOED** | |
| Node | Parent |
| Arad |  |
|  |  |
|  |  |
|  |  |

+ Lần lặp 3:

|  |  |
| --- | --- |
| **OPEN** | |
| Node | h(n) |
| Timisoara | 329 |
| Zerind | 374 |
| Fagaras | 176 |
| Oradea | 380 |
| Rimnicu vilcea | 193 |

|  |  |
| --- | --- |
| **CLOED** | |
| Node | Parent |
| Arad |  |
| Sibiu | Arad |
|  |  |
|  |  |

+ Lần lặp 4:

|  |  |
| --- | --- |
| **OPEN** | |
| Node | h(n) |
| Timisoara | 329 |
| Zerind | 374 |
| Oradea | 380 |
| Rimnicu vilcea | 193 |
| Bucharest | 0 |

|  |  |
| --- | --- |
| **CLOED** | |
| Node | Parent |
| Arad |  |
| Sibiu | Arad |
| Fagaras | Sibiu |
|  |  |

+ Lần lặp 5: Tìm được node kết thúc Bucharest.

* + 1. Kết quả và thảo luận

- Ưu điểm :

+ Thuật toán có thể chuyển đổi giữa BFS và DFS và đạt được những lợi thế của cả 2 thuật toán đó.

+ Thuật toán này hiệu quả hơn BFS và DFS

- Nhược điểm :

+ Nó có thể hoạt động như một tìm kiếm theo chiều sâu không có hướng dẫn trong trường hợp xấu nhất.

+ Nó có thể bị mắc kẹt trong một vòng lặp như DFS.

+ Thuật toán này không tối ưu.

-Đánh giá thuật toán:

+Time Complexity: Độ phức tạp về thời gian trong trường hợp xấu nhất là O(b^m)

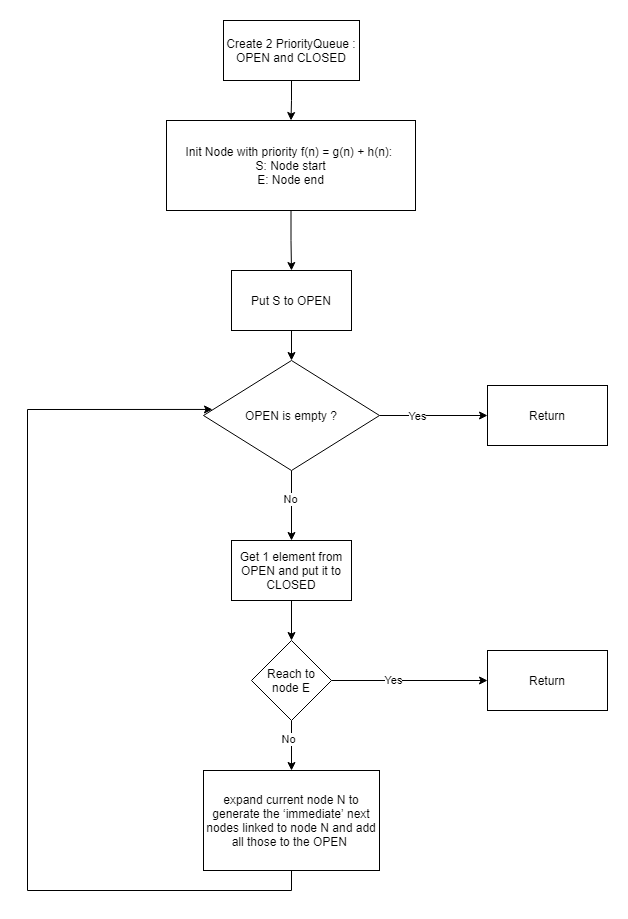
+Space Complexity: Độ phức tạp không gian trong trường hợp xấu nhất là O(b^m) với m là là độ sâu tối đa của không gian tìm kiếm

+ Complete: Greedy best-first-search cũng không đầy đủ, ngay cả khi không gian trạng thái đã cho là hữu hạn.

+ Optimal: Thuật toán này không tối ưu.

1.3 Giải thuật A\*

1.3.1 Sơ đồ giải thuật



1.3.2 Hiện thực

-Chương trình tìm đường đi từ Arad đến Bucharest

+ Lần lặp 1 :

|  |  |  |  |
| --- | --- | --- | --- |
| **OPEN** | | | |
| Node | g(n) | h(n) | f(n) |
| Arad | 0 | 366 | 366 |
|  |  |  |  |
|  |  |  |  |

|  |  |
| --- | --- |
| **CLOSED** | |
| Node | Parent |
|  |  |
|  |  |

+ Lần lặp 2:

|  |  |  |  |
| --- | --- | --- | --- |
| **OPEN** | | | |
| Node | g(n) | h(n) | f(n) |
| Sibiu | 140 | 253 | 393 |
| Tmnisoara | 118 | 329 | 447 |
| Zerind | 75 | 374 | 449 |

|  |  |
| --- | --- |
| **CLOSED** | |
| Node | Parent |
| Arad |  |
|  |  |

+ Lần lặp 3:

|  |  |  |  |
| --- | --- | --- | --- |
| **OPEN** | | | |
| Node | g(n) | h(n) | f(n) |
| Tmnisoara | 118 | 329 | 447 |
| Zerind | 75 | 374 | 449 |
| Fagaras | 239 | 176 | 415 |
| Oradea | 291 | 380 | 671 |
| Rimnicu Vilcea | 220 | 193 | 413 |

|  |  |
| --- | --- |
| **CLOSED** | |
| Node | Parent |
| Arad |  |
| Sibiu | Arad |

+Lần lặp 4:

|  |  |  |  |
| --- | --- | --- | --- |
| **OPEN** | | | |
| Node | g(n) | h(n) | f(n) |
| Tmnisoara | 118 | 329 | 447 |
| Zerind | 75 | 374 | 449 |
| Fagaras | 239 | 176 | 415 |
| Oradea | 291 | 380 | 671 |
| Craiova | 366 | 160 | 526 |
| Pitesti | 317 | 100 | 417 |
|  |  |  |  |

|  |  |
| --- | --- |
| **CLOSED** | |
| Node | Parent |
| Arad |  |
| Sibiu | Arad |
| Rimnicu Vilcea | Sibiu |
|  |  |

+Lần lặp 5:

|  |  |  |  |
| --- | --- | --- | --- |
| **OPEN** | | | |
| Node | g(n) | h(n) | f(n) |
| Tmnisoara | 118 | 329 | 447 |
| Zerind | 75 | 374 | 449 |
| Fagaras | 239 | 176 | 415 |
| Oradea | 291 | 380 | 671 |
| Craiova | 366 | 160 | 526 |
| Bucharest | 0 | 418 | 418 |
|  |  |  |  |

|  |  |
| --- | --- |
| **CLOSED** | |
| Node | Parent |
| Arad |  |
| Sibiu | Arad |
| Rimnicu Vilcea | Sibiu |
| Pitesti | Rimnicu Vilcea |

+Lần lặp 6 Tìm được Bucharest

1.3.3 Kết quả và thảo luận

- Ưu điểm:

+ Thuật toán A\* là thuật toán tốt nhất trong tất cả các thuật toán tìm kiếm.

+ Thuật toán A\* là tối ưu và đầy đủ.

+ Thuật toán A\* có thể giải được các vấn đề phức tạp.

- Nhược điểm:

+ Nó không phải lúc nào cũng tìm ra đường đi ngắn nhất vì nó chủ yếu dựa trên phương pháp phỏng đoán và tính gần đúng.

+ Nhược điểm chính của A \* là yêu cầu bộ nhớ vì nó giữ tất cả các nút được tạo trong bộ nhớ, vì vậy nó không thực tế cho các vấn đề quy mô lớn khác nhau

- Đánh giá thuật toán:

+Time Complexity: Độ phức tạp về thời gian phụ thuộc vào hàm đánh giá và số node mở rộng là cấp số nhân với độ sâu của nghiệm d. Vì vậy độ phức tạp về thời gian là O(b^d) với b là là hệ số phân nhánh.

+ Space Cpmplexity: O(b^d)

+ Complete: Thuật toán là đầy đủ nếu: Hệ số phân nhánh là hữu hạn và Chi phí cho mọi hành động là cố định.

+ Optimal: Thuật toán là tối ưu nếu đủ 2 điều kiện sau:

Admissible: điều kiện đầu tiên yêu cầu để tối ưu là h (n) phải là một heuristic có thể chấp nhận được cho tìm kiếm cây A \*.

Consistency: Điều kiện bắt buộc thứ hai là tính nhất quán chỉ dành cho thuật toán A \*

Nếu hàm heuristic được chấp nhận, thì thuật toán A \* sẽ luôn tìm ra đường dẫn có chi phí thấp nhất.

-Application: A \* thường được sử dụng cho vấn đề tìm đường phổ biến trong các ứng dụng như trò chơi điện tử, nhưng ban đầu được thiết kế như một thuật toán duyệt đồ thị chung, tìm kiếm thông tin trực tuyến

**BÀI 2- ISLANDS**

* 1. Mô tả cấu trúc dữ liệu

- Island được khởi tạo bằng mảng 2 chiều với giá trị 0 là biển, 1 là đất liền.

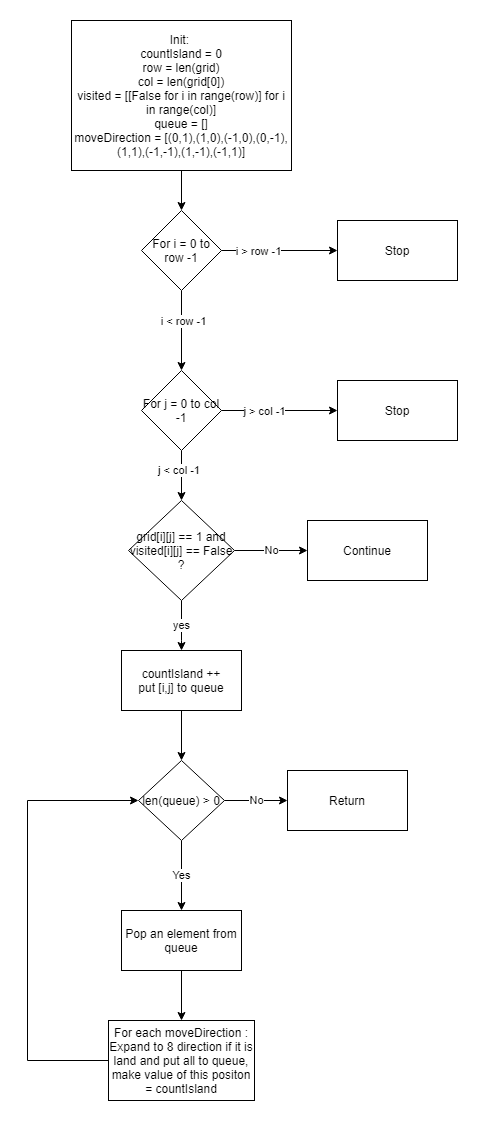
- Mỗi 1 vị trí trên island là 1 node với 2 thuộc tính: thuộc tính thứ nhất node là List chứa vị trí từng đơn vị trên island. Thuộc tính thứ 2 theo thuật toán : đối với BFS thì thuộc tính thứ 2 không cần thiết, đối với UCS là g chứa giá trị priority

- BFS sữ dụng hàng đợi queue để giải quyết vấn đề

- UCS sữ dụng hàng đợi ưu tiên PriorityQueue để giải quyết vấn đề

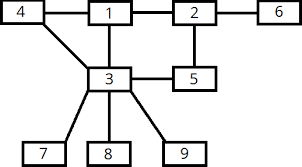
2.2 Giải thuật BFS – Breadth First Search

2.2.1 Sơ đồ giải thuật



* + 1. Hiện thực

- Ta có đồ thị :



- Duyệt cây từ nút 1 :

|  |  |  |  |
| --- | --- | --- | --- |
| Lần lặp | Nút được chọn | Nút đã duyệt | Queue |
| 0 |  | 1 | 1 |
| 1 | 1 | 1 | 2 3 4 |
| 2 | 2 | 1 2 | 3 4 5 6 |
| 3 | 3 | 1 2 3 | 4 5 6 7 8 9 |
| 4 | 4 | 1 2 3 4 | 5 6 7 8 9 |
| 5 | 5 | 1 2 3 4 5 | 6 7 8 9 |
| 6 | 6 | 1 2 3 4 5 6 | 7 8 9 |
| 7 | 7 | 1 2 3 4 5 6 7 | 8 9 |
| 8 | 8 | 1 2 3 4 5 6 7 8 | 9 |
| 9 | 9 | 1 2 3 4 5 6 7 8 9 |  |
|  | Kết thúc |  |  |

* + 1. Kết quả và thảo luận

- Ưu điểm:

+ BFS chắc chắn tìm ra được kết quả nếu như vấn đề đó có thể giải được.

+ BFS sẽ không bao giờ bị mắc kẹt bởi các node không mong muốn.

+ Nếu có nhiều hơn một giải pháp thì nó sẽ tìm ra một giải pháp với các bước tối thiểu.

- Nhược điểm:

+ Hạn chế về bộ nhớ Vì nó lưu trữ tất cả các nút của cấp độ hiện tại để chuyển sang cấp độ tiếp theo.

+ Nếu một giải pháp ở xa thì nó tiêu tốn thời gian.

- Time complexity: Nếu V và E là tập hợp các đỉnh và cung của đồ thị, thì thời gian thực thi thuật toán là O(|E| + |V|) = O(b^d) vì trong trường hợp xấu nhất. mỗi đỉnh và cung của đồ thị được thăm đúng một lần. Ghi chú O(|E| + |V|) nằm trong khoảng từ O(|V|) đến O(|V^2|), tùy theo số cung của đồ thị.

- Space complexity: nếu V là tập hợp đỉnh của đồ thị và |V| là số đỉnh thì không gian cần dùng của thuật toán là O(|V|) = O(b^d).

- Optimal: Tối ưu với đồ thị không trọng số.

- Completeness: BFS là đầy đủ có nghĩa là nếu node nằm ở độ sâu hữu hạn nào đó thì BFS sẽ tìm ra giải pháp.

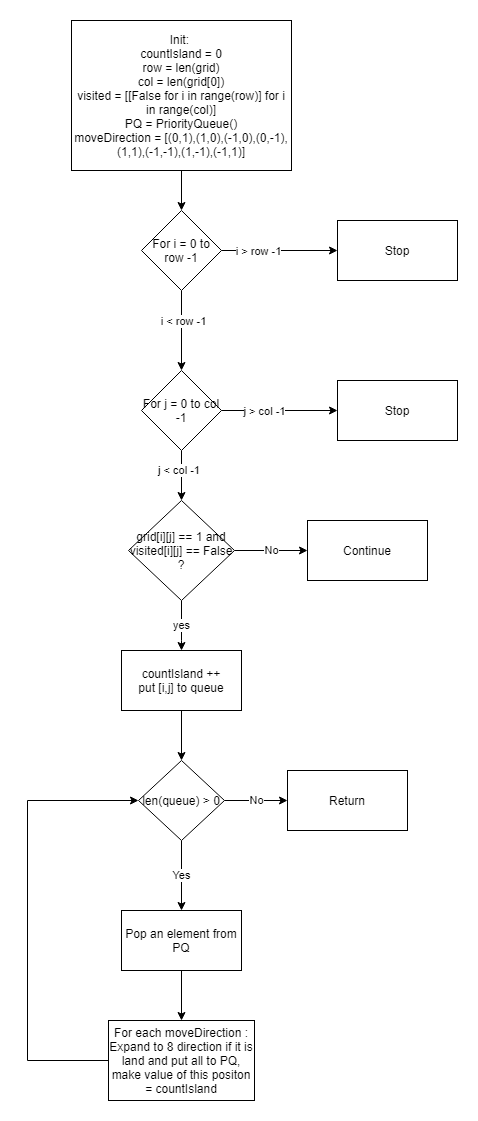
-Application:

+ Tìm kiếm theo chiều rộng có thể được sử dụng để giải quyết các trò chơi trong đó một loạt các lựa chọn dẫn đến trạng thái thắng hoặc thua. Ví dụ: BFS có thể giúp người chơi xác định chuỗi thắng của các nước đi để giải khối Rubik.

+ BFS cũng được sử dụng trong thuật toán Dijkstra nổi tiếng để tính toán đường đi ngắn nhất trong biểu đồ và thuật toán Ford-Fulkerson để tính toán lưu lượng tối đa trong mạng luồng.

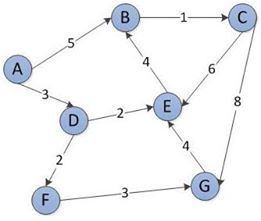
2.3 Giải thuật UCS- Uniform Cost Search

2.3.1 Sơ đồ giải thuật



2.3.2 Hiện thực

- Ta có đồ thị sau:



-Duyệt cây tử đỉnh A đến G với priority là chi phí từ đỉnh A đến đỉnh hiện tại đang được xét.

|  |  |  |  |
| --- | --- | --- | --- |
| Lần lặp | Nút được chọn | Nút đã duyệt | PriorityQueue |
| 0 |  | A | [A,0] |
| 1 | [A,0] | A | [B,5] [D,3] |
| 2 | [D,3] | A D | [B,5] [E,5] [F,5] |
| 3 | [B,5] | A D B | [E,5] [F,5] [C,6] |
| 4 | [E,5] | A D B E | [F,5] [C,6] |
| 5 | [F,5] | A D B E F | [C,6] [G,8] |
| 6 | [C,6] | A D B E F C | [G,8] |
| 7 | [G,8] | A D B E F C G |  |
|  | Kết thúc |  |  |
|  |  |  |  |
|  |  |  |  |

- Ta được đường đi A -> D -> F ->G với chi phí là 8

2.3.3 Kết quả và thảo luận

-Ưu điểm: UCS tối ưu vì mỗi bước chọn thuật toán này đều chọn ở chi phí thấp nhất

-Nhược điểm:

+Thuật toán UCS có thể bị mắc kẹt ở vòng lặp vô hạn vì nó không quan tâm đến số bước tìm kiếm mà nó chỉ quan tâm đến chi phí thấp nhất.

+ Bộ nhớ được yêu cầu lớn theo cấp số nhân

-Time Complexity: ta có C\* là chi phí của một giải pháp tối ưu và ɛ là mỗi bước đi để tiến gần đến node đích. Ở đây ta lấy ɛ +1 khi node bắt dầu từ trạng thái 0 và kết thúc ở C\*/ɛ. Do đó, độ phức tạp về thời gian trong trường hợp xấu nhất là

-Space Complexity: độ phức tạp về không gian trong trường hợp xấu nhất là

-Optimal : UCS tối ưu vì nó chỉ chọn đường đi với chi phí thấp nhất.

**-** Completeness: UCS là đầy đủ, nếu vấn đề có giải pháp thì UCS sẽ tìm ra giải pháp đó.

-Application: dùng nhiều trong tính toán để thiết lập các đường của đường dây điện hoặc đường ống dẫn dầu. Thuật toán cũng đã được sử dụng để tính toán các lối đi bộ đường dài tối ưu ở Ethiopia và đối chiếu chúng với tình hình trên mặt đất

**TỰ ĐÁNH GIÁ**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bài tập** | **Nội dung** | **Điểm** | **Ghi chú** |
| **Bài 1**  **(5 điểm)** | **Mô tả cấu trúc dữ liệu (1 đ)** | **1** |  |
| **Giải thuật Greedy Best First Search (Heuristic) (2 đ)** | **2** |  |
| Sơ đồ giải thuật (0.5 đ) | **1** |  |
| Hiện thực (1 đ) | **1** |  |
| Kết quả và thảo luận (0.5 đ) | **0.5** |  |
| **Giải thuật A\* (2 đ)** | **2** |  |
| Sơ đồ giải thuật (0.5 đ) | **0.75** |  |
| Hiện thực (1 đ) | **1** |  |
| Kết quả và thảo luận (0.5 đ) | **0.5** |  |
| **Bài 2**  **(5 điểm)** | **Mô tả cấu trúc dữ liệu (1 đ)** | **0.75** |  |
| **Giải thuật BFS (2 đ)** | **1.5** |  |
| Sơ đồ giải thuật (0.5 đ) | **0.25** |  |
| Hiện thực (1 đ) | **0.75** |  |
| Kết quả và thảo luận (0.5 đ) | **0.5** |  |
| **Giải thuật UCS (2 đ)** | **1.5** |  |
| Sơ đồ giải thuật (0.5 đ) | **0.25** |  |
| Hiện thực (1 đ) | **0.75** |  |
| Kết quả và thảo luận (0.5 đ) | **0.5** |  |
| **Tổng điểm** |  | **8.75** |  |

**TÀI LIỆU THAM KHẢO**

1. Wikipeida, Best-first search (2021):

<https://en.wikipedia.org/wiki/Best-first_search>

1. Wikipedia, A\* search algorithm (2021):

[https://en.wikipedia.org/wiki/A\*\_search\_algorithm#Complexity](https://en.wikipedia.org/wiki/A*_search_algorithm#Complexity)

1. Wikipedia, Dijkstra’s algorithm (2021):

<http://sciencewise.info/resource/Uniform-cost_search/Uniform-cost_search_by_Wikipedia>

1. Wikipedia, Tìm kiếm chi phí đều (2020):

<https://vi.wikipedia.org/wiki/T%C3%ACm_ki%E1%BA%BFm_chi_ph%C3%AD_%C4%91%E1%BB%81u>

1. Wikipedia, Breadth-first search (2021):

<https://en.wikipedia.org/wiki/Breadth-first_search>

1. Karlelgh Moore, Abhishek Gujrati, Breadth – First Search (BFS) :

<https://brilliant.org/wiki/breadth-first-search-bfs/#:~:text=%5B%5D%20Q%3D%5B%5D-,Complexity%20of%20Breadth%20First%20Search,edge%20will%20be%20checked%20once>.

1. JAVATPOINT, Uninformed Search Algoriths :

<https://www.javatpoint.com/ai-uninformed-search-algorithms#:~:text=The%20same%20logic%20is%20for,with%20the%20lowest%20path%20cost>.

1. Vatsal Unadkat, Advantages and Disadvantages of Search Algorithms:

<https://medium.com/@vatsalunadkat/advantages-and-disadvantages-of-ai-algorithms-d8fb137f4df2>

1. Great Learning Team, Best First Search Algorithm in AI | Concept, Implementation, Advantages, Disadvantages:

<https://www.mygreatlearning.com/blog/best-first-search-bfs/>

1. JAVATPOINT, Informed Search Algorithms:

[https://www.javatpoint.com/ai-informed-search-algorithms#:~:text=A\*%20Search%20Algorithm%3A,it%20solve%20the%20problem%20efficiently](https://www.javatpoint.com/ai-informed-search-algorithms#:~:text=A*%20Search%20Algorithm%3A,it%20solve%20the%20problem%20efficiently).