**TRƯỜNG ĐẠI HỌC THỦ DẦU MỘT**

**VIỆN KỸ THUẬT CÔNG NGHỆ**

****

**BÀI TẬP GIỮA KỲ**

**MÔN NHẬP MÔN TRÍ TUỆ NHÂN TẠO**

*Người thực hiện*: **NGUYỄN ĐỨC LONG – MSSV1824801040033**

Lớp **: D18HT01**

Khoá  **: D18**

**BÌNH DƯƠNG, NĂM 2021**

**TRƯỜNG ĐẠI HỌC THỦ DẦU MỘT**

**VIỆN KỸ THUẬT CÔNG NGHỆ**

****

**BÀI TẬP GIỮA KỲ**

**MÔN NHẬP MÔN TRÍ TUỆ NHÂN TẠO**

*Người thực hiện*: **NGUYỄN ĐỨC LONG – MSSV1824801040033**

Lớp **: D18HT01**

Khoá  **: D18**

**BÌNH DƯƠNG, NĂM 2021**

PHẦN ĐÁNH GIÁ CỦA GIẢNG VIÊN

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

Bình Dương, ngày tháng năm

(kí và ghi họ tên)

**MỤC LỤC**

[BÀI 1- TRAVELLING IN ROMANIA 2](#_Toc66388291)

[1.1 Mô tả cấu trúc dữ liệu 2](#_Toc66388292)

[1.2 Giải thuật Greedy Best First Search (Heuristic) 2](#_Toc66388293)

[1.2.1 Sơ đồ giải thuật 2](#_Toc66388294)

[1.2.2 Hiện thực 4](#_Toc66388295)

[1.2.3Kết quả và thảo luận 8](#_Toc66388296)

[1.3 Giải thuật A\* 8](#_Toc66388297)

[1.3.1 Sơ đồ giải thuật 8](#_Toc66388298)

[1.3.3 Kết quả và thảo luận 10](#_Toc66388299)

[BÀI 2- ISLANDS 11](#_Toc66388300)

[2.1 Mô tả cấu trúc dữ liệu 11](#_Toc66388301)

[2.2 Giải thuật BFS – Breadth First Search 11](#_Toc66388302)

[2.2.1 Sơ đồ giải thuật 11](#_Toc66388303)

[2.2.2 Hiện thực 11](#_Toc66388304)

[2.2.3Kết quả và thảo luận 13](#_Toc66388305)

[2.3 Giải thuật UCS- Uniform Cost Search 14](#_Toc66388308)

[2.3.1 Sơ đồ giải thuật 14](#_Toc66388309)

[2.3.2 Hiện thực 15](#_Toc66388312)

[2.3.3 Kết quả và thảo luận 15](#_Toc66388313)

[TÀI LIỆU THAM KHẢO 17](#_Toc66388314)

# BÀI 1- TRAVELLING IN ROMANIA

## 1.1 Mô tả cấu trúc dữ liệu

Cấu trúc dữ liệu của bài này bao gồm địa điểm, khoảng cách giữa các địa điểm, đường đi giữa các địa điểm.

## 1.2 Giải thuật Greedy Best First Search (Heuristic)

### 1.2.1 Sơ đồ giải thuật

Từ điển ánh xạ các từ theo định nghĩa của chúng. Vì vậy, một từ điển có thể được thực hiện bằng cách ánh xạ hai tập A và B trong đó thành viên đầu vào A được ánh xạ với thành viên đầu ra B. Việc ánh xạ này có thể được thực hiện bằng một thuật toán tìm kiếm. Cây AVL được sử dụng làm thuật toán tìm kiếm trong dự án này. Thuật toán tìm kiếm so khớp thành viên đầu ra B với thành viên đầu vào đã cho A. Cây AVL là cây tìm kiếm nhị phân tự cân bằng. Cây AVL mất O (log n) thời gian cho các hoạt động chèn, xóa và tìm kiếm. Cây phải được cân bằng lại sau các thao tác chèn và xóa. Dự án này hỗ trợ các thao tác chèn, xóa và tìm kiếm cây AVL. Trong thao tác chèn, giá trị 'A' được chèn cho khóa 'K' đã cho. Trong thao tác tìm kiếm, giá trị 'A' được truy xuất cho khóa 'K' đã cho. Trong thao tác xóa, giá trị 'A' đã cho sẽ bị xóa khỏi từ điển.

Trong quá trình triển khai Từ điển bằng cách sử dụng cây Tìm kiếm nhị phân, một trong những vấn đề là chúng trở nên mất cân bằng một cách tồi tệ. Vì vậy, một cây Tìm kiếm Nhị phân cân bằng như cây AVL có thể được sử dụng để triển khai Từ điển. Cây AVL luôn cân đối. Trong cây AVL, chiều cao của hai cây con của bất kỳ nút nào chỉ khác nhau nhiều nhất là một. Cây AVL có thể mất cân bằng sau khi thực hiện thao tác chèn và xóa. Luân chuyển là cơ chế cơ bản tái cân bằng cây không cân bằng. Vòng quay là một sự điều chỉnh đối với cây, xung quanh một vật phẩm, duy trì thứ tự yêu cầu của các mặt hàng.

Dự án này hỗ trợ ba hoạt động cơ bản của cây AVL. 1. Chèn 2. Tìm kiếm 3. Xóa 1. Chèn hoạt động Một nút mới có thể được chèn nếu giá trị nút chưa có trong cây. Sau khi chèn mỗi nút, cây sẽ được kiểm tra xem có cân bằng không. Các nút duy nhất cần được kiểm tra là các nút nằm dọc theo đường chèn của nút mới được chèn. Một khi cây được phát hiện là không cân bằng thì hãy cân bằng lại bằng thao tác luân chuyển thích hợp. 2. Hoạt động Tìm kiếm Thao tác Tìm kiếm có thể được thực hiện với giá trị khóa cần được tìm kiếm trong cây AVL đã cho. Thao tác tìm kiếm trả về một nút từ cây nếu giá trị nút khớp với giá trị khóa. Nếu giá trị khóa không khớp với bất kỳ giá trị nút nào trong cây thì không có giá trị nào được trả về.

3. Thao tác Xóa Xóa phức tạp hơn chèn. Việc xóa một nút khỏi cây có thể làm giảm chiều cao của cây, dẫn đến cấu trúc cây không cân đối. Vì vậy hoạt động luân chuyển thích hợp được thực hiện để cân bằng lại cây. Cung cấp cho người dùng một menu (tức là tương tác) cho phép họ tìm kiếm một từ được chỉ định trong danh sách, chèn một từ mới, xóa một từ hiện có hoặc in toàn bộ danh sách theo thứ tự tăng dần của các từ. Kiểm tra chương trình của bạn với ít nhất 10 từ. Ví dụ về menu tương tác:

Nhập 1 - để chèn một từ mới

Nhập 2 - để tìm kiếm một từ

Nhập 3 - để xóa một từ

Nhập 4 - để hiển thị toàn bộ danh sách theo thứ tự tăng dần

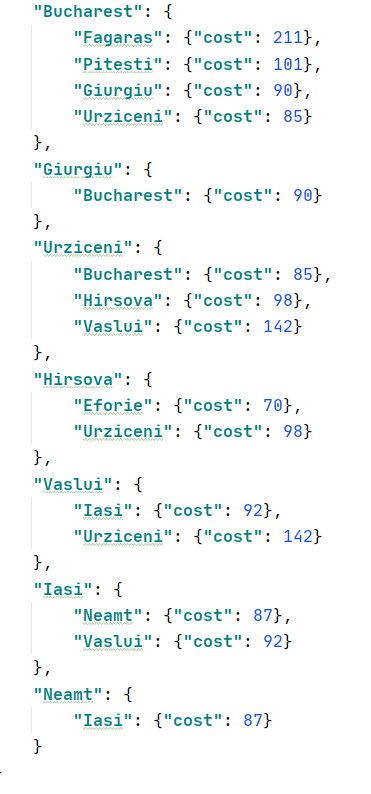
Nhập 0 - để thoát

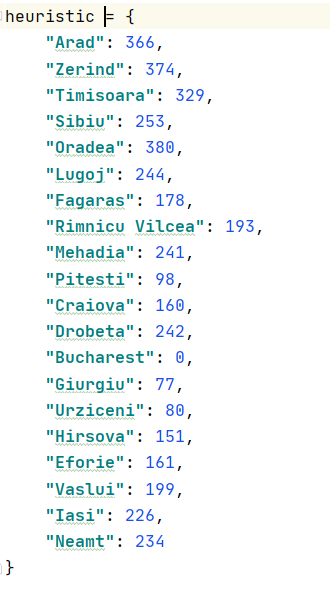
### 1.2.2 Hiện thực

Bây giờ chúng ta sẽ thực hiện bằng Python

Đầu tiên, ta sẽ import thư viện PriorityQueue, Sau đó ghi ra khoảng cách giữa các điểm







Sau đó khởi tạo hàm Greedy Best First, bắt đầu từ Arad và điểm đếm là Bucharest.



### 1.2.3 Kết quả và thảo luận

Kết quả của giải thuật trên

Có thể thấy với giải thuật trên nó đã tìm ra con đường ngắn nhất ở mỗi nút để đến đích

## 1.3 Giải thuật A\*

### 1.3.1 Sơ đồ giải thuật

B1. Khởi tạo danh sách OPEN = {trạng thái ban đầu}

B2. While true do

B2.1 if (OPEN rỗng) then

{thông báo tìm kiếm thất bại; stop};

B2.2 Loại trạng thái u ở đầu danh sách OPEN;

B2.3 if u là trạng thái kết thúc then

{thông báo tìm kiếm thành công; stop};

B2.4 for mỗi v kề u

g(v) = g(u) +k(u,v); //k(u,v) là chi phí thực tế để đi từ u tới v, được cho trước

f(v) = g(v) + h(v); //ưu tiên g nhỏ, cho phép duyệt lại v nếu f mới tốt hơn

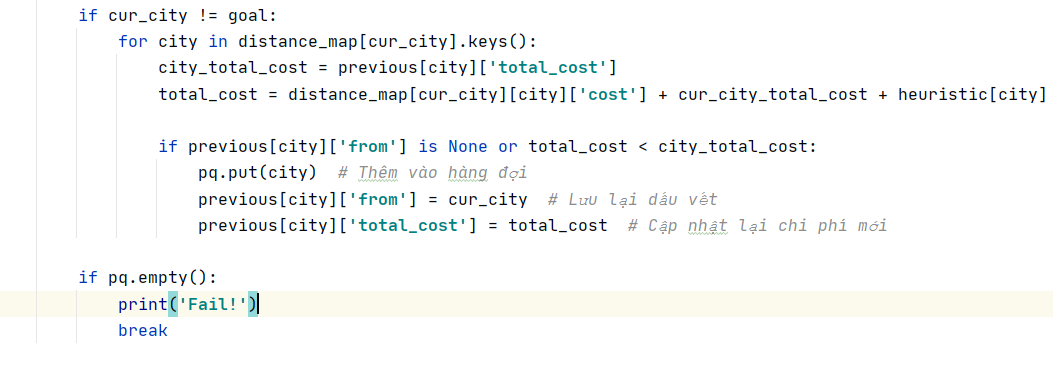
Chèn v vào OPEN sao cho OPEN được sắp xếp theo thứ tự tăng dần của f

END

1.3.2 Hiện thực

Ta lập trình cho giải thuật trên với đường đi bắt đầu từ Arad đến Bucharest





### 1.3.3 Kết quả và thảo luận

Kết quả đạt được



Với giải thuật A\*, dù trong bài này nó không thể hiện điều đó nhưng trên thực tế nó được sử dụng nhiều hơn bởi sự hiệu quả về thời gian và cách xử lý của nó.

# BÀI 2- ISLANDS

## 2.1 Mô tả cấu trúc dữ liệu

Cấu trúc dữ liệu của bài này gồm

Biển : là các ô màu xanh

Đất liền : là các ô màu trắng

Đất liền được đánh số thứ tự theo các khối : 1,2,3,4,5

## 2.2 Giải thuật BFS – Breadth First Search

### 2.2.1 Sơ đồ giải thuật

Bước 1: Tập Open chứa đỉnh gốc s chờ được xét.

Bước 2: Kiểm tra tập Open có rỗng không.

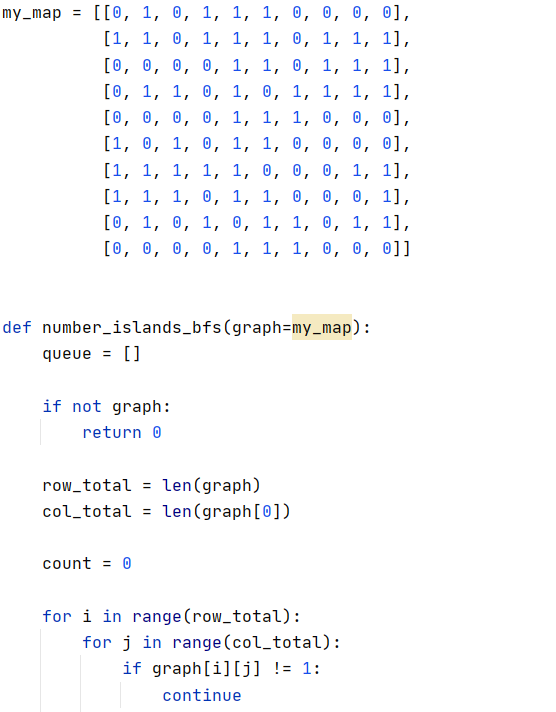
* + Nếu tập Open không rỗng, lấy một đỉnh ra khỏi tập Open làm đỉnh đang xét p. Nếu p là đỉnh g cần tìm, kết thúc tìm kiếm.
  + Nếu tập Open rỗng, tiến đến bước 4.

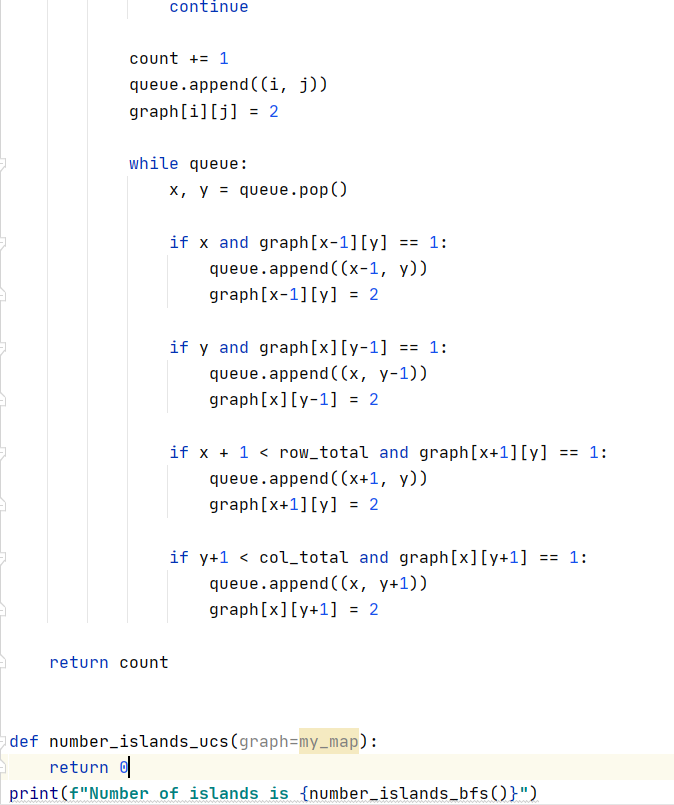
Bước 3: Đưa đỉnh p vào tập Close, sau đó xác định các đỉnh kề với đỉnh p vừa xét. Nếu các đỉnh kề không thuộc tập Close, đưa chúng vào cuối tập Open. Quay lại bước 2.

Bước 4: Kết luận không tìm ra đỉnh đích cần tìm

### 2.2.2 Hiện thực

Ta thực hiện thuật toán bằng python





### 2.2.3Kết quả và thảo luận

Kết quả đạt được



**Ưu điểm**

* Xét duyệt tất cả các đỉnh để trả về kết quả.
* Nếu số đỉnh là hữu hạn, thuật toán chắc chắn tìm ra kết quả.

**Khuyết điểm**

* Mang tính chất vét cạn, không nên áp dụng nếu duyệt số đỉnh quá lớn.
* Mang tính chất mù quáng, duyệt tất cả đỉnh, không chú ý đến thông tin trong các đỉnh để duyệt hiệu quả, dẫn đến duyệt qua các đỉnh không cần thiết.

## 2.3 Giải thuật UCS- Uniform Cost Search

### 2.3.1 Sơ đồ giải thuật

### tìm kiếm chi phí đều (hay còn gọi là tìm kiếm chi phí cực tiểu hoặc tìm kiếm theo giá thành thống nhất, viết tắt [tiếng Anh](https://vi.wikipedia.org/wiki/Ti%E1%BA%BFng_Anh) là [UCS](https://vi.wikipedia.org/wiki/UCS)) là một cách [duyệt cây](https://vi.wikipedia.org/wiki/Duy%E1%BB%87t_c%C3%A2y) dùng cho việc duyệt hay tìm kiếm một [cây](https://vi.wikipedia.org/wiki/C%C3%A2y_(c%E1%BA%A5u_tr%C3%BAc_d%E1%BB%AF_li%E1%BB%87u)), [cấu trúc cây](https://vi.wikipedia.org/w/index.php?title=C%E1%BA%A5u_tr%C3%BAc_c%C3%A2y&action=edit&redlink=1), hoặc [đồ thị](https://vi.wikipedia.org/w/index.php?title=%C4%90%E1%BB%93_th%E1%BB%8B_(c%E1%BA%A5u_tr%C3%BAc_d%E1%BB%AF_li%E1%BB%87u)&action=edit&redlink=1) có trọng lượng (chi phí). Việc tìm kiếm bắt đầu tại [nút](https://vi.wikipedia.org/wiki/N%C3%BAt_(khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh)) gốc.[[1]](https://vi.wikipedia.org/wiki/T%C3%ACm_ki%E1%BA%BFm_chi_ph%C3%AD_%C4%91%E1%BB%81u#cite_note-1) Việc tìm kiếm tiếp tục bằng cách duyệt các nút tiếp theo với trọng lượng hay chi phí thấp nhất tính từ nút gốc. Các nút được duyệt tiếp tục cho đến khi đến được nút đích cần đến.

### Thuật toán

**procedure** *UniformCostSearch*(Graph, root, goal)

node:= root, cost = 0

frontier:= priority queue containing node only

explored:= empty set

**do**

**if** frontier is empty

**return** failure

node:= frontier.pop()

**if** node is goal

**return** solution

explored.add(node)

**for each** of node's neighbors n

**if** n is not in explored

**if** n is not in frontier

frontier.add(n)

**else if** n is in frontier with higher cost

replace existing node with n

### 2.3.2 Hiện thực

### 2.3.3 Kết quả và thảo luận

**TỰ ĐÁNH GIÁ**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bài tập** | **Nội dung** | **Điểm** | **Ghi chú** |
| **Bài 1**  **(5 điểm)** | **Mô tả cấu trúc dữ liệu (1 đ)** | **0,25** |  |
| **Giải thuật Greedy Best First Search (Heuristic) (2 đ)** |  |  |
| Sơ đồ giải thuật (0.5 đ) | **0** |  |
| Hiện thực (1 đ) | **1** |  |
| Kết quả và thảo luận (0.5 đ) | **0,5** |  |
| **Giải thuật A\* (2 đ)** |  |  |
| Sơ đồ giải thuật (0.5 đ) | **0** |  |
| 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)** | **0,25** |  |
| **Giải thuật BFS (2 đ)** |  |  |
| Sơ đồ giải thuật (0.5 đ) | **0** |  |
| Hiện thực (1 đ) | **1** |  |
| Kết quả và thảo luận (0.5 đ) | **0,5** |  |
| **Giải thuật UCS (2 đ)** |  |  |
| Sơ đồ giải thuật (0.5 đ) | **0,5** |  |
| Hiện thực (1 đ) | **0** |  |
| Kết quả và thảo luận (0.5 đ) | **0** |  |
| **Tổng điểm** |  | **5,5** |  |

# TÀI LIỆU THAM KHẢO

1. Quách Ngọc Ân (1992), “Nhìn lại hai năm phát triển lúa lai”, *Di tuyền học ứng dụng*, 98(1), tr. 10-16.
2. Bộ nông nghiệp & PTNT (1996), *Báo cáo tổng kết 5 năm (1992-1996) phát triển lúa lai,* Hà Nội.
3. Nguyễn Hữu Đống, Đào Thanh Bằng, Lâm Quang Dụ, Phan Đức Trực (1997), *Đột biến –* *Cơ sở lý luận và ứng dụng,* Nhà xuất bản nông nghiệp, Viện khoa học kỹ thuật nông nghiệp Việt Nam, Hà Nội.
4. Nguyễn Thị Gấm (1996), *Phát hiện và đánh giá một số dòng bất dục đực cảm ứng nhiệt* *độ,* Luận văn thạc sĩ khoa học nông nghiệp, Viện khoa học kỹ thuật nông nghiệp Việt Nam, Hà Nội.

……….

1. Anderson J.E. (1985), The Relative Inefficiency of Quota, The Cheese Case, *American* *Economic Review*, 75(1), pp. 178-90.
2. Borkakati R. P., Virmani S. S. (1997), Genetics of thermosensitive genic male sterility in Rice, *Euphytica* 88, pp. 1-7.