TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**Báo cáo giữa kỳ**

**Nhập môn AI**

**BÁO CÁO GIỮA KỲ**

**MÔN NHẬP MÔN AI**

*Người hướng dẫn:* **GV. BÙI THANH HÙNG**

*Người thực hiện:* **NGUYỄN TRUNG TÍN - 51900640**

Lớp : **19050302**

Khoá  **: 23**

**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**



**Báo cáo giữa kỳ**

**Nhập môn AI**

**BÁO CÁO GIỮA KỲ**

**MÔN NHẬP MÔN AI**

*Người hướng dẫn:* **GV. BÙI THANH HÙNG**

*Người thực hiện:* **NGUYỄN TRUNG TÍN - 51900640**

Lớp : **19050302**

Khóa  **: 23**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021**

# LỜI CẢM ƠN

Trước tiên, tôi xin gửi lời cảm ơn chân thành và lòng biết ơn sâu sắc đến GV. Bùi Thanh Hùng. Thầy là người đã luôn hỗ trợ và hướng dẫn tận tình cho chúng tôi trong suốt quá trình nghiên cứu và hoàn thành bài báo cáo này.

Tiếp theo, tôi xin gửi lời cảm ơn đến khoa Công Nghệ Thông Tin trường Đại học Tôn Đức Thắng vì đã tạo điều kiện cho chúng tôi được học tập và nghiên cứu môn học này. Khoa đã luôn sẵn sàng chia sẻ các kiến thức bổ ích cũng như chia sẻ các kinh nghiệm tham khảo tài liệu, giúp ích không chỉ cho việc thực hiện và hoàn thành đề tài nghiên cứu mà còn giúp ích cho việc học tập và rèn luyện trong quá trình thực hành tại trường Đại học Tôn Đức Thắng nói chung.

Cuối cùng, sau khoảng thời gian học tập trên lớp tôi đã hoàn tất đề tài cuối kì nhờ vào sự hướng dẫn, giúp đỡ và những kiến thức học hỏi. Do giới hạn về mặt kiến thức và khả năng lý luận nên nhóm vẫn còn nhiều thiếu sót và hạn chế, kính mong sự chỉ dẫn và đóng góp của Quý cô cô giáo để bài Nghiên cứu của tôi được hoàn thiện hơn. Hơn nữa, nhờ những góp ý từ cô cô và các bạn hữu, tôi sẽ hoàn thành tốt hơn ở những bài nghiên cứu trong tương lai. Tôi mong Quý cô cô và các bạn bè – những người luôn quan tâm và hỗ trợ chúng tôi – luôn tràn đầy sức khỏe và vạn sự bình an.

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi và được sự hướng dẫn của GV. Bùi Thanh Hùng ; Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Nguyễn Trung Tín*

**Phần đánh giá của GV chấm bài**

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

Tp. Hồ Chí Minh, ngày tháng năm

(ký và ghi họ tên)

MỤC LỤC

[LỜI CẢM ƠN](#_30j0zll) **1**

[**Bài 1- Số đảo**](#_xfb7hha7oe3h) **2**

[Mô tả cấu trúc dữ liệu](#_3mho2jxc99gm) 2

[1.2 Giải thuật BFS – Breadth First Search](#_tddhmtq9int7) 3

[1.2.1 Sơ đồ giải thuật:](#_962oe4a62o6g) 3

[Hiện thực](#_bncr2z8rg3c2) 5

[1.2.3 Kết quả và thảo luận](#_1uatb9h7skg4) 7

[Câu 1b](#_c07jf6tb9bhq) 7

[**Câu 2**](#_uzx6628mj5xm) **11**

[Câu 2a](#_4d34og8) 11

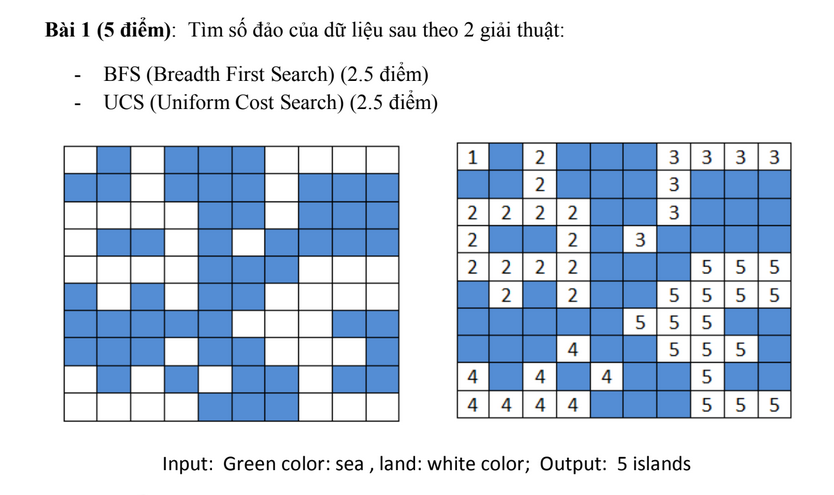
[Câu 2b](#_dxigrhnv6nwy) 17

[**Test**](#_rsoqxetdvsmw) **22**

# Bài 1- Số đảo

## Mô tả cấu trúc dữ liệu

Đề bài:

****

Tiếp cận vấn đề: theo đề có thể hiểu theo cách là:

* Những ô màu trắng tượng trưng cho đất liền
* Những ô màu xanh tượng trưng cho biển
* Ta xét trong 8 ô liền kề, những ô màu trắng được liền nhau thì là một đảo
* Output sẽ là tổng số đảo.

Từ đó ta sẽ cho rằng:

* đất liền sẽ là 1
* biển sẽ là 0
* Tập dữ liệu là file txt có dạng như sau:

1 1 0 0 0 0

1 0 0 1 0 1

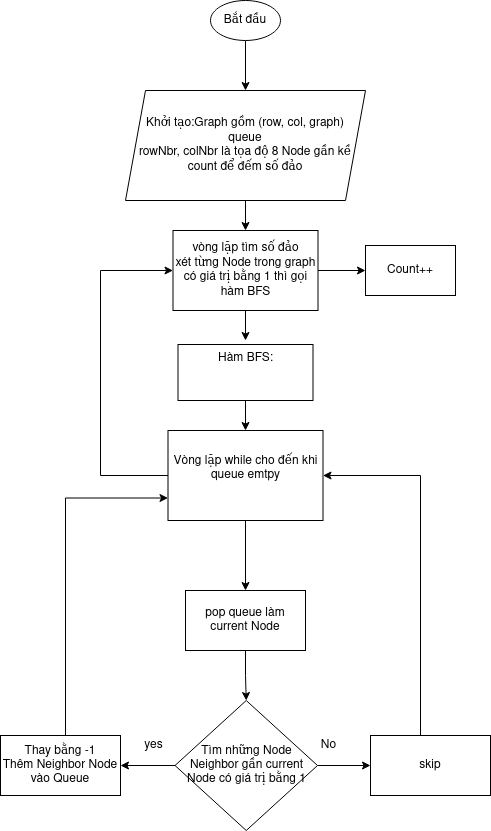
0 1 0 1 0 0

0 0 0 0 0 0

1 1 0 0 1 1

## 1.2 Giải thuật BFS – Breadth First Search

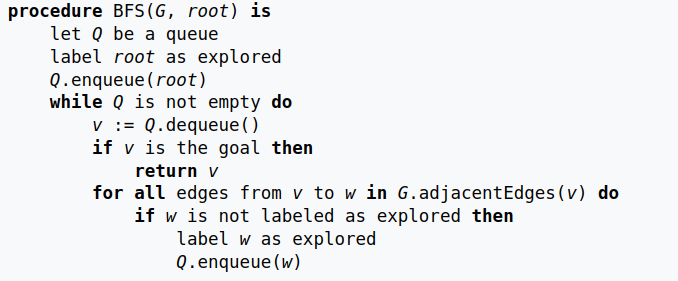
### 1.2.1 Sơ đồ giải thuật:



### Hiện thực

Sau khi một thời gian nghiên cứu thì tôi đã tìm thấy một bài viết trên wikipedia:

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



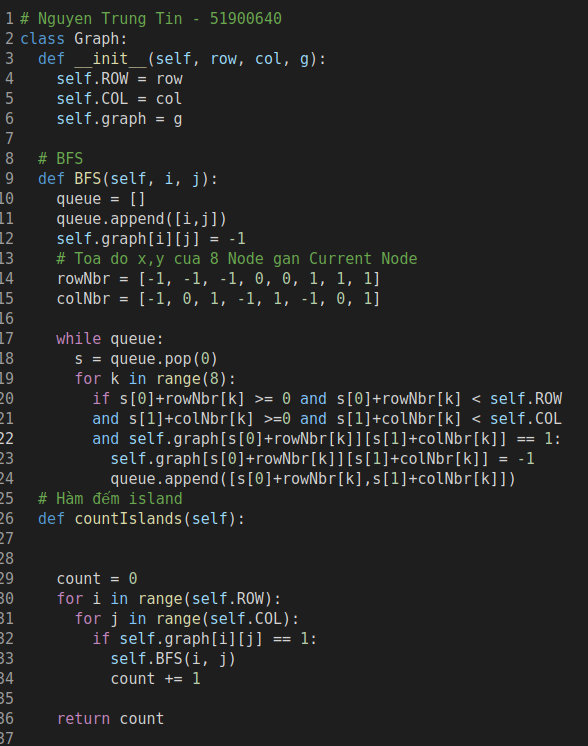
Từ mã giả này ta có thể viết lại cho phù hợp với yêu cầu đề bài:

Code khi viết:

* Quan trọng ở đây là vòng lặp, vì thuật toán sẽ liên tục lặp lại để đến các Node tiếp theo.
* Vì vậy ta sẽ bắt đầu với vòng lặp trước.
* Ta sẽ tìm 8 Node Neighbor xung quanh Node current và nếu Node đó hợp lệ và có giá trị bằng 1 (đất liền) thì ta cho nó = -1 (đánh dấu là đã duyệt).
* Rồi thêm Node vừa duyệt vào queue.
* Pop queue ra làm Node current.
* Cứ như vậy ta duyệt hết các phần tử trong 1 đảo

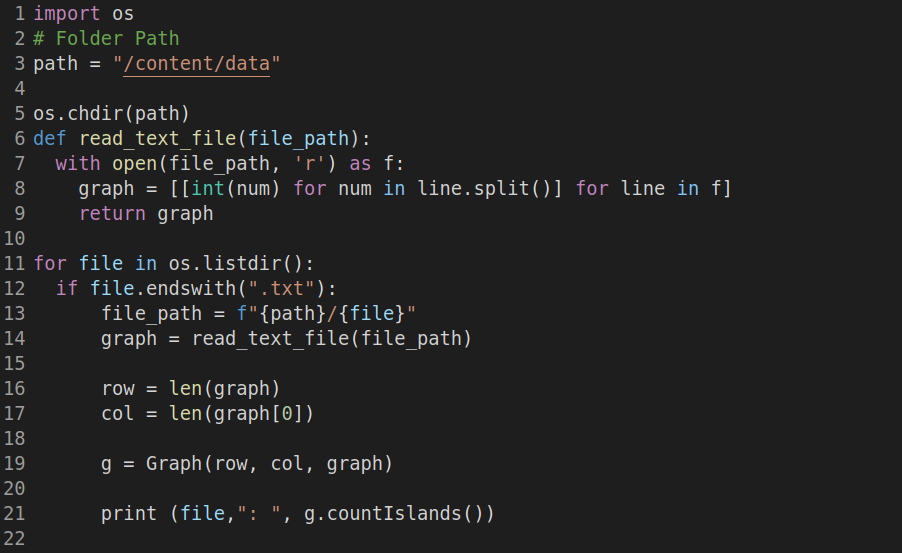
Bây giờ ta đã biết cách để duyệt các phần tử trong 1 đảo. Vậy để đếm tất cả các đảo thì phải làm như thế nào?

* Trước hết ta duyệt từng phần tử trong graph, nếu phần tử đó = 1 thì ta gọi hàm BFS để đánh dấu đảo đó vì giờ đây hàm BFS đúng hơn với tên gọi đánh dấu đảo vì khi gặp giá trị hợp lệ và = 1 nó đếu đánh dấu lại -1.
* Khi hàm BFS duyệt hết đảo thì ta cộng 1 cho biến Count (đã tìm thấy 1 đảo).
* Tiếp tục duyệt các phần tử trong đảo xem có còn giá trị 1 nào không rồi gọi hàm BFS.
* Cứ như vậy sau khi duyệt hết graph ta được biến count là giá trị tổng số đảo trong graph.

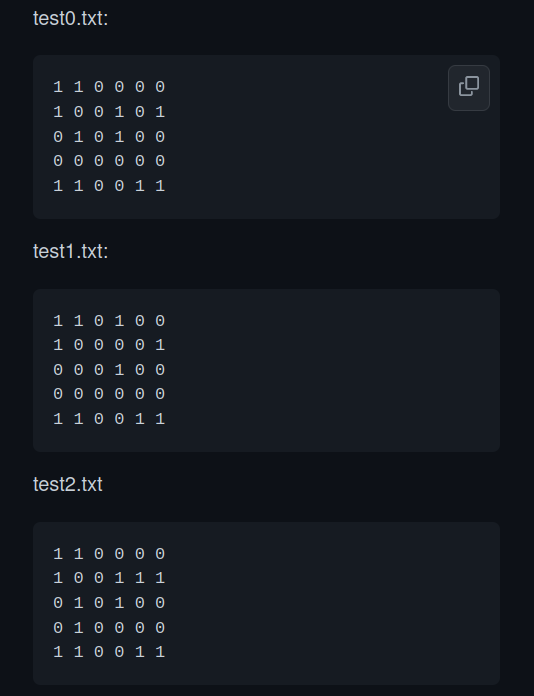


Ta cần phải viết một hàm để chuẩn bị giá trị input vào để có thể đưa giá trị cho hàm tìm đảo cũng như BFS.

* Ta sẽ viết một hàm mà trong đó giá trị đầu vào là một path của folder
* Trong folder đó sẽ có các file txt là các test case
* Duyệt từng test case và đếm đảo để in kết quả.



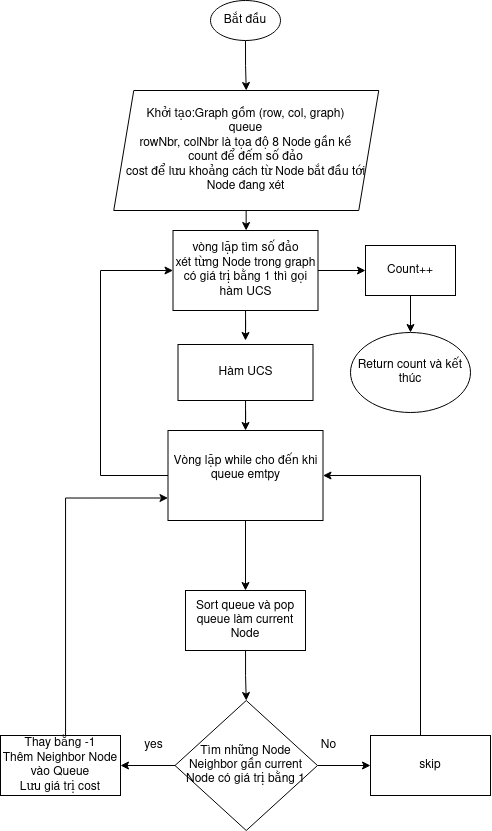
### 1.2.3 Kết quả và thảo luận





## 1.3 Giải thuật UCS- Uniform Cost Search

### 1.3.1 Sơ đồ giải thuật



### Hiện thực

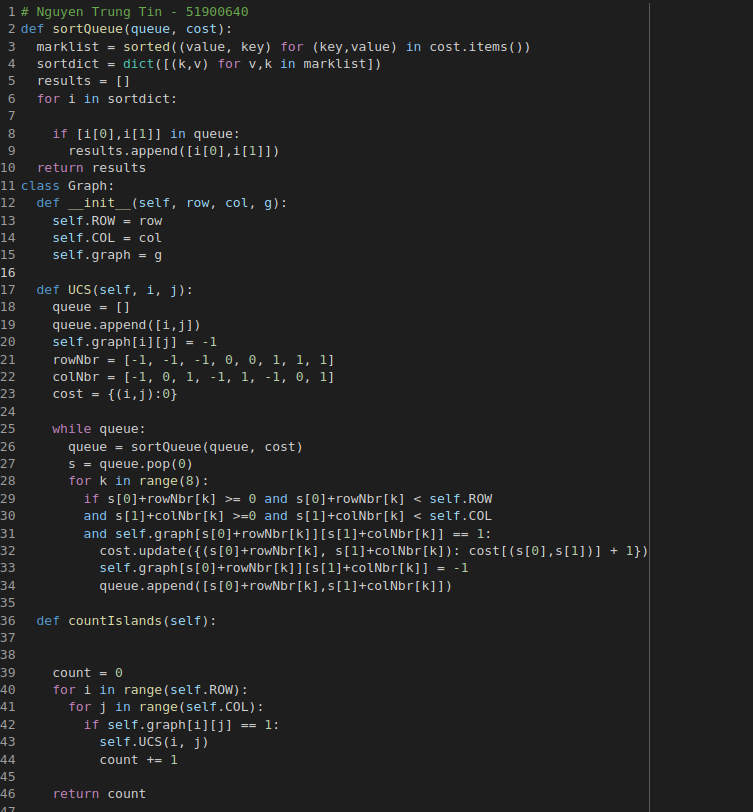
Ta tham khảo tài liệu về UCS ở wikipedia thì biết được là UCS là một biến thể của Dijkstra's Search chỉ khác một điều là một bên thì chỉ cần tìm thấy Node đích sẽ dừng thuật toán, một bên khi tìm được Node đích sẽ quay lại tìm đường đi tới Node đích khác cho đến khi duyệt hết các Node trong cây thì đưa ra đường đi ngắn nhất.

source: <https://en.wikipedia.org/wiki/Dijkstra's_algorithm>

Ta cần hiểu về thuật toán UCS:

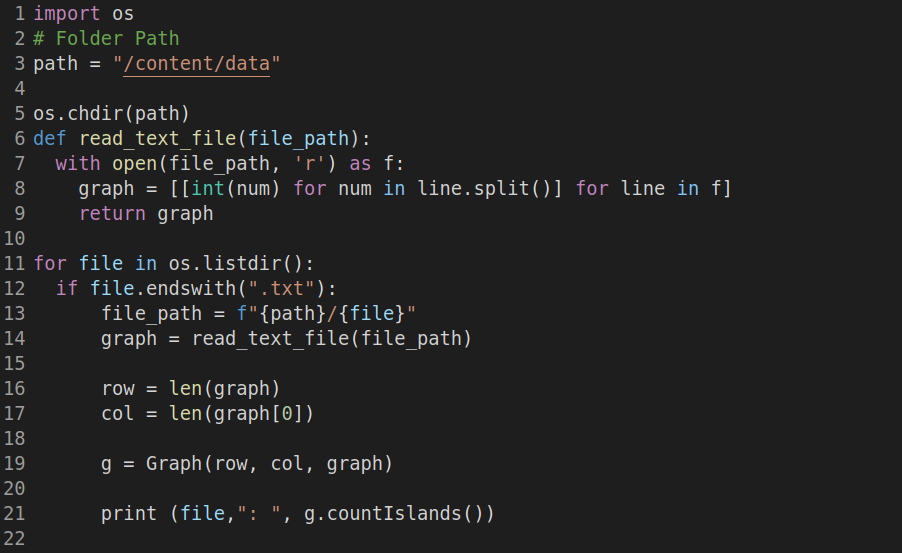
* Thuật toán UCS sẽ tìm đường đi ngắn nhất bằng cách kiểm tra xem giá trị của Node Neighbor đang xét có là ngắn nhất không.
* Nếu có thì ta đưa vào
* Ở BFS bài trước nếu Node ok thì ta đưa vào queue rồi pop
* Nhưng ở bài này ta phải pop phần tử có đường đi tới nó ngắn nhất nghĩa là ta phải sort lại queue rồi pop phần tử đầu tiên (phần tử có đường đi ngắn nhất) làm current Node.
* Rồi các bước tiếp theo ta làm như BFS.

***Node****: nhưng nếu để ý ta sẽ thấy có thể dùng y nguyên hàm BFS vẫn có thể duyệt cây theo con concept của UCS vì UCS sẽ quan trọng khoảng cách giữa các Node nhưng theo đề bài thì tất cả các Node cạnh nhau đều có khoảng cách bằng nhau. Hơn thế nữa là ta cũng không có Node đích. Còn hàm sort queue đã nêu trên thì cũng không cần vì queue ở bài BFS đã sort sẵn luôn rồi vì những phần tử vào queue sẽ theo thứ tự từ gần Node gốc (Có khoảng cách bằng 1) trước sau đó là các Node có khoảng cách xa hơn, cứ như vậy. Nhưng trong bài này ta vẫn nên làm cho đúng thuật toán.*

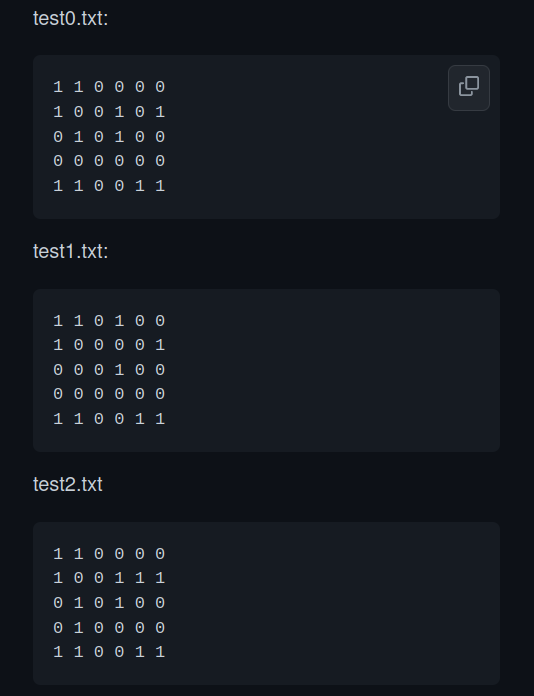


Ta cần phải viết một hàm để chuẩn bị giá trị input vào để có thể đưa giá trị cho hàm tìm đảo cũng như UCS.

* Ta sẽ viết một hàm mà trong đó giá trị đầu vào là một path của folder
* Trong folder đó sẽ có các file txt là các test case
* Duyệt từng test case và đếm đảo để in kết quả.



### 1.2.3 Kết quả và thảo luận



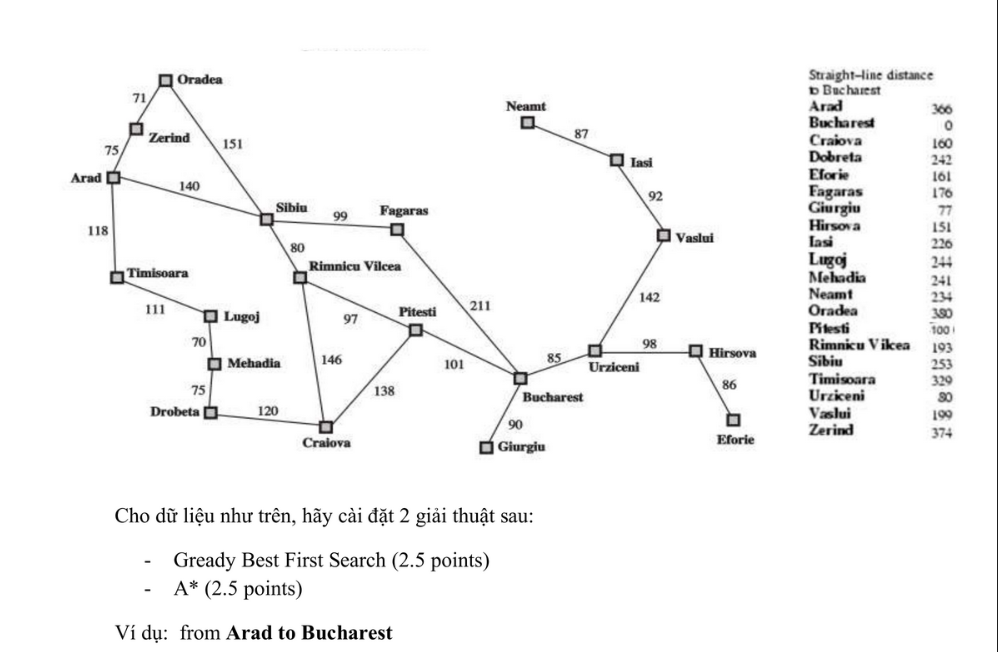


**Bài 2- Du lịch ở Romania**

# Câu 2a

## 2.1 Mô tả cấu trúc dữ liệu

Đề bài:



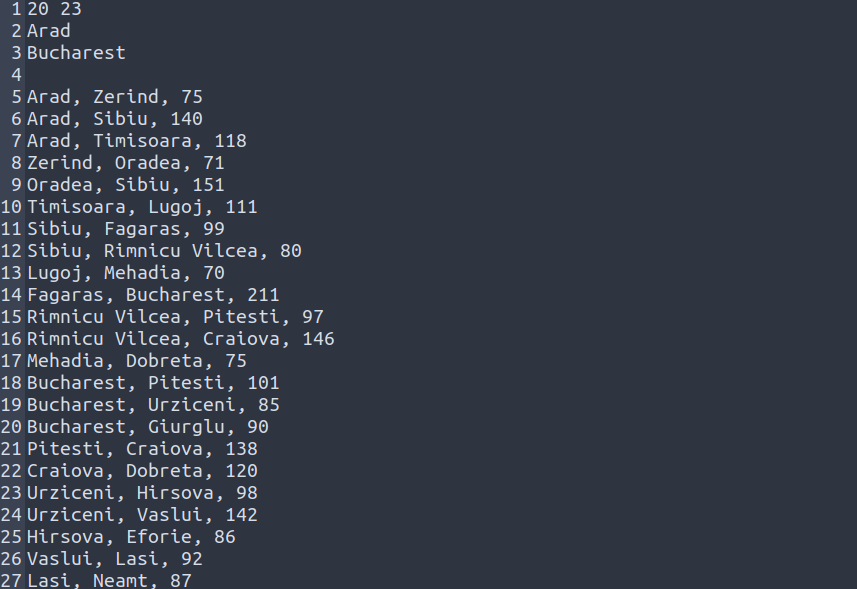
Tiếp cận vấn đề:

Theo đề bài ta có thể hiểu như sau:

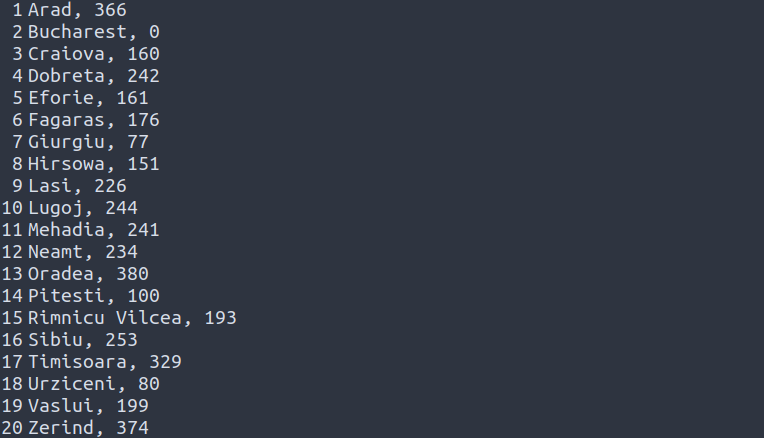
* Ta có một graph gồm các điểm là một thành phố
* Giữa các thành phố là khoảng cách
* Ta có một bảng dữ liệu Heuristic, là khoảng cách chim bay từ các thành phố đến thành phố đích, trong trường hợp này thành phố đích là Bucharest.
* Nhiệm vụ của ta là ta cần tìm đường đi ngắn nhất từ một thành phố bất kì đến thành phố Bucharest. Bằng thuật toán Gready Best First Search và A\*

Bước đầu tiên ta cần chuẩn bị data:

* Data bao gồm 2 file txt:
  + Graph.txt
    - dòng đầu là số thành phố và số đường nối giữa 2 thành phố với nhau.
    - 2 dòng kế tiếp là điểm bắt đầu(start) và điểm đích(destination)

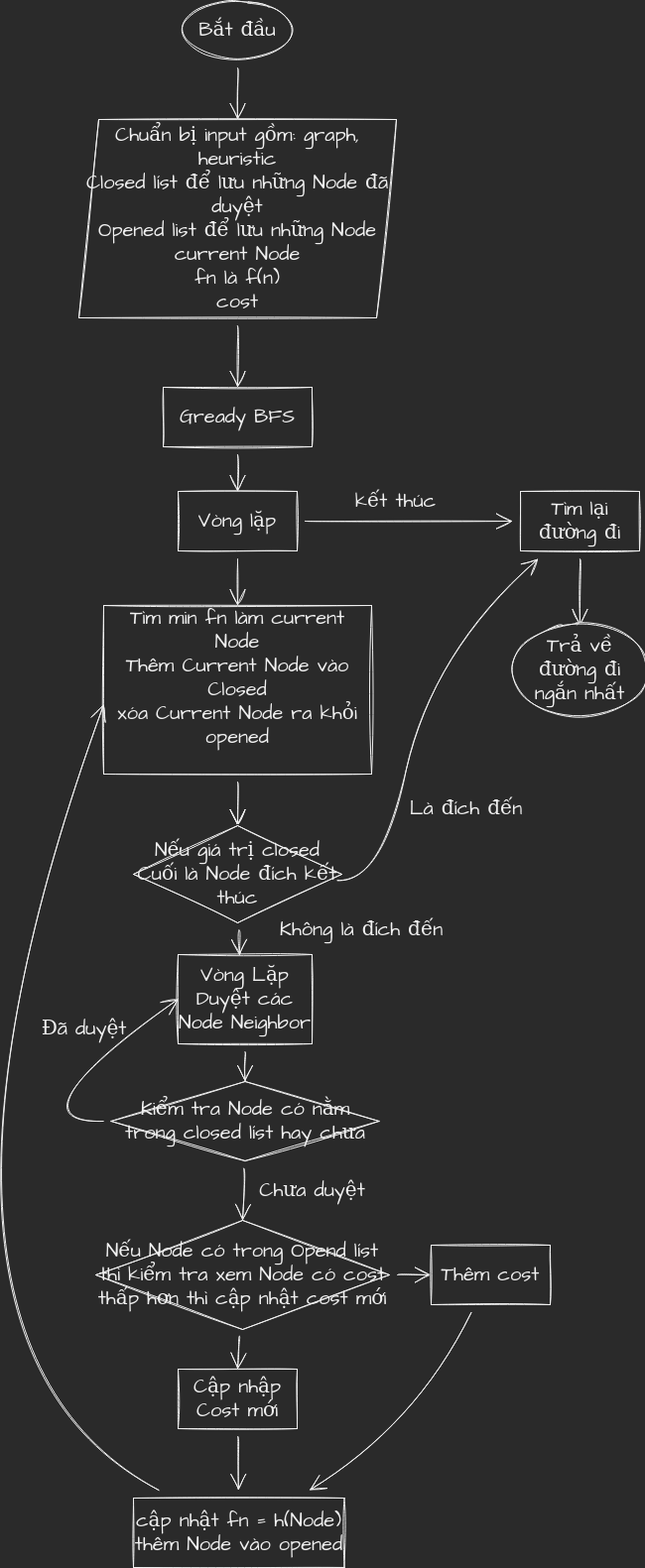


* + Heuristic.txt
    - Gồm độ dài đường chim bay đến điểm đích (Bucharest)



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

### 2.2.1 Sơ đồ giải thuật

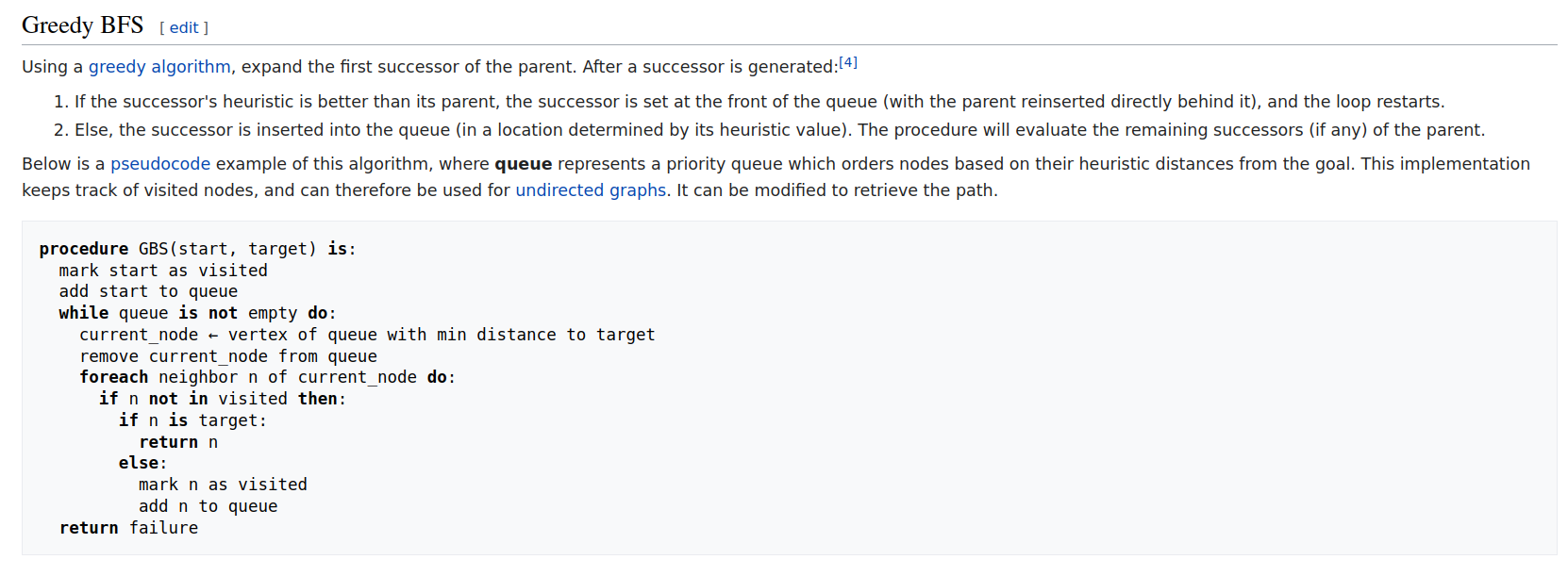


### Hiện thực

Bây giờ hãy bắt đầu xây dựng thuật toán.

Wikipedia đã cung cấp cho ta hướng dẫn cũng như là mã giả để ta có thể xây dựng thuật toán. Từ đó thì ta có thể chỉnh sửa lại để phù hợp với yêu cầu đề bài.

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



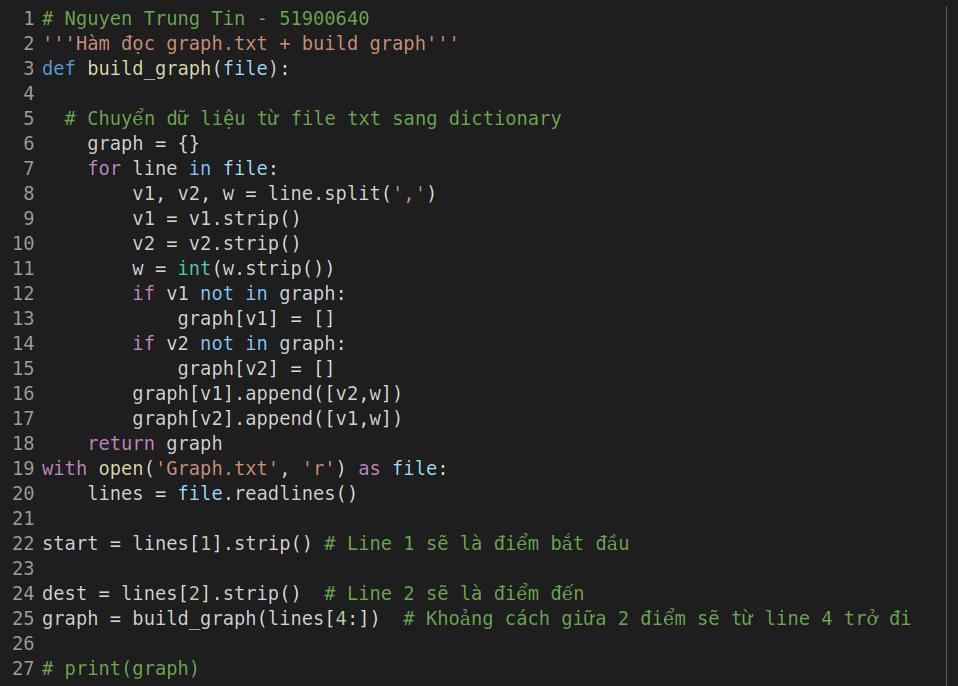
Quan trọng ở đây là vòng lặp. Vì ta phải hiểu là thuật toán sẽ như một chu trình lặp lại để kiểm tra điều kiện và tìm đường đi ngắn nhất đến đích.

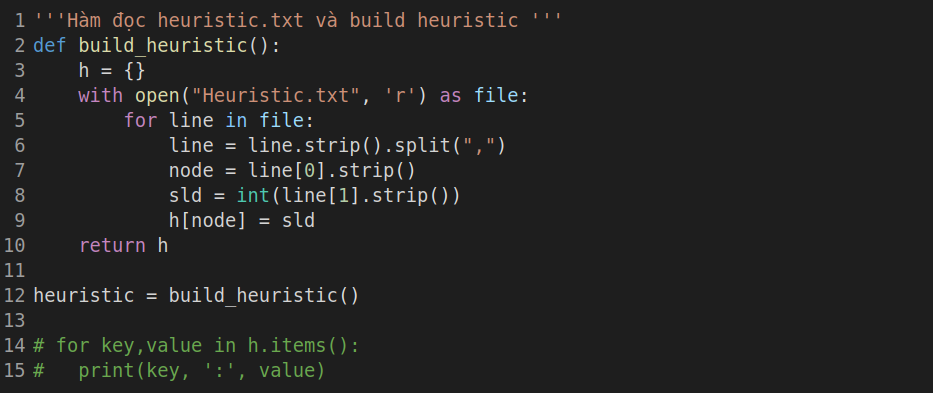
Điều thứ 2 ta cần phải quan tâm đó chính là điều kiện duyệt Node của thuật toán Gready BFS dựa trên công thức: f(n) = h(n). h(h) chính là Heuristic, khoảng cách đường chim bay giữa Node đang duyệt tới Node đích . f(n) là chỉ số mà ta cần phải xem xét giá trị nhỏ nhất để làm current Node.

Điều thứ 3 cần qua tâm đó chính là ta cần tạo một closed\_set và opend\_set để biết được Node nào đã duyệt Node nào đang trong hàng đợi cần phải duyệt.

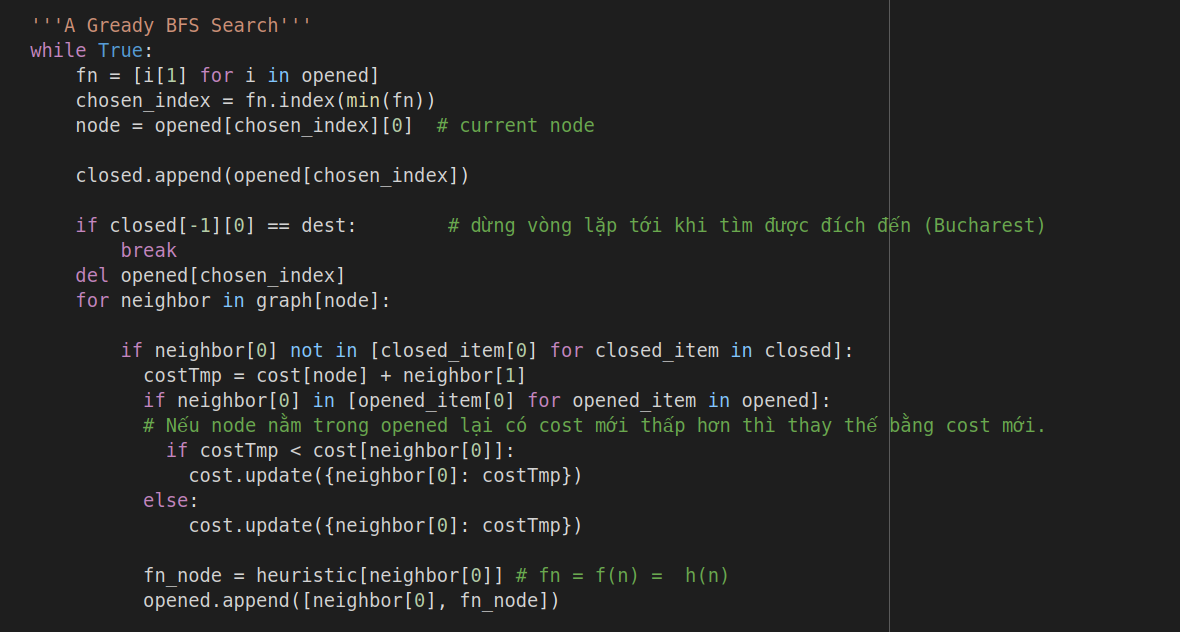
Và điều cuối cùng ta cần phải quan tâm đó chính là tìm lại đường đi sau khi thuật toán đã tìm đến đích.

Thuật toán để chuẩn bị input từ các file data:

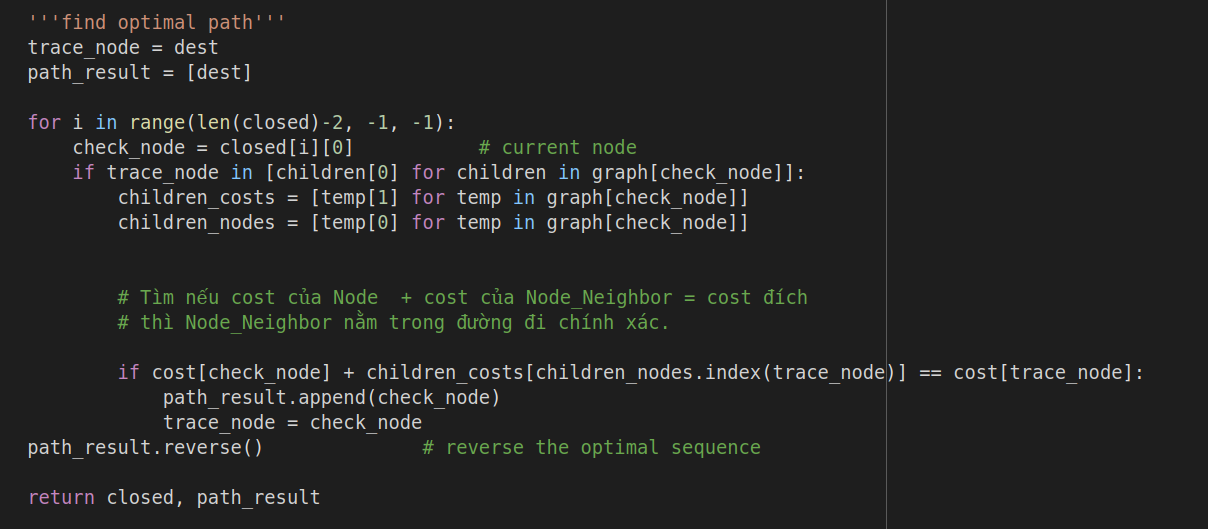




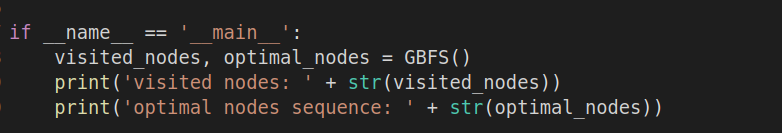
Thuật toán Gready BFS khi đã viết như sau:



Còn đây là thuật toán để tìm lại đường đi và tìm những Node đã duyệt.

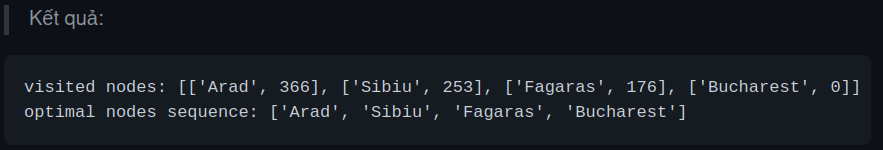


Cuối cùng ta chạy chương trình như sau:

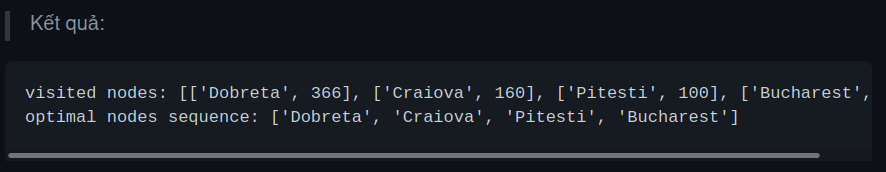


### 2.2.3 Kết quả và thảo luận

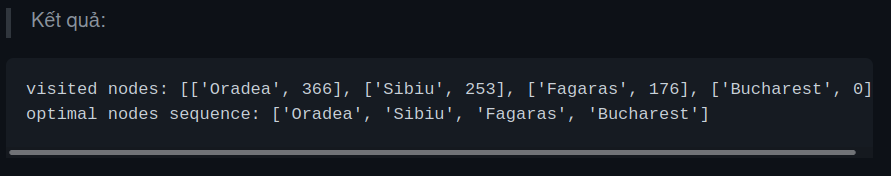
* Test case 1: Từ Arad đến Bucharest



* Test case 2: Từ Dobreta đến Bucharest

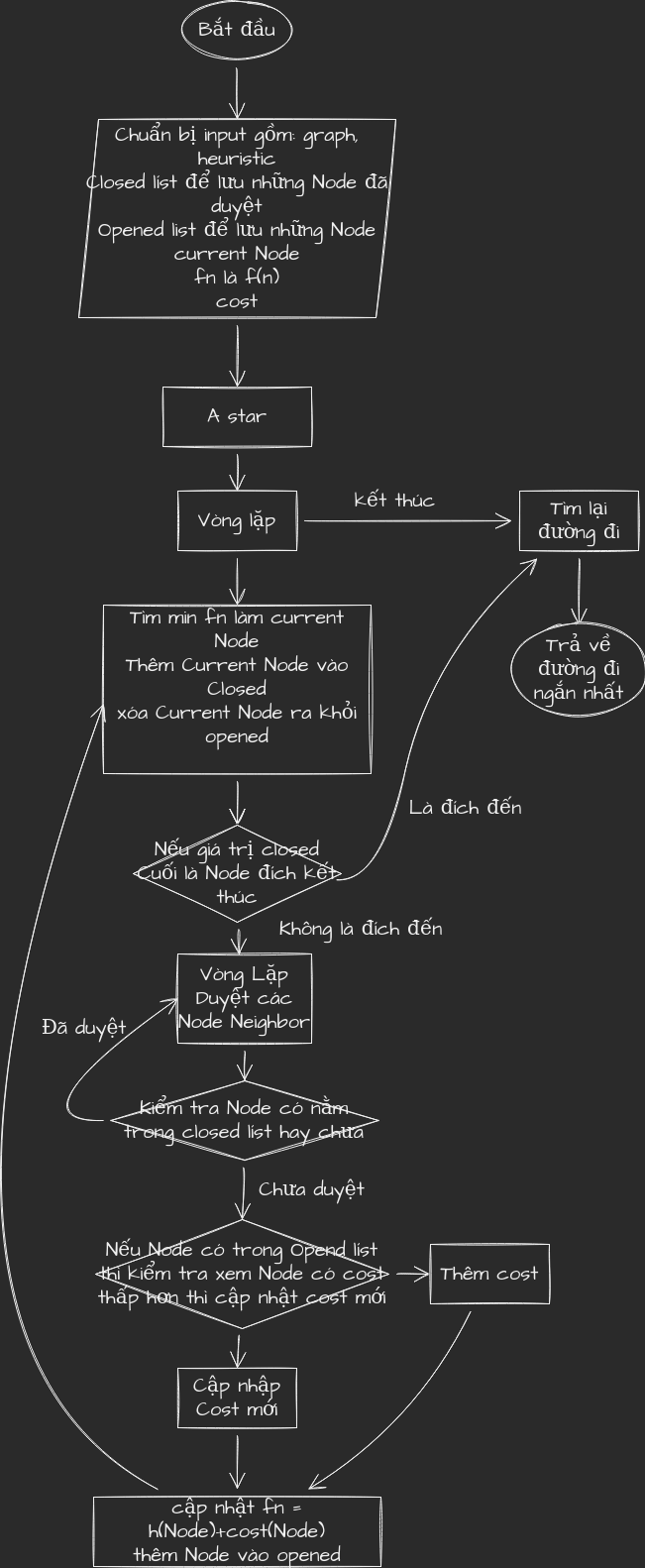


* Test case 3: Từ Oredea đến Bucharest



## 2.3 Giải thuật A\*

### 2.3.1 Sơ đồ giải thuật

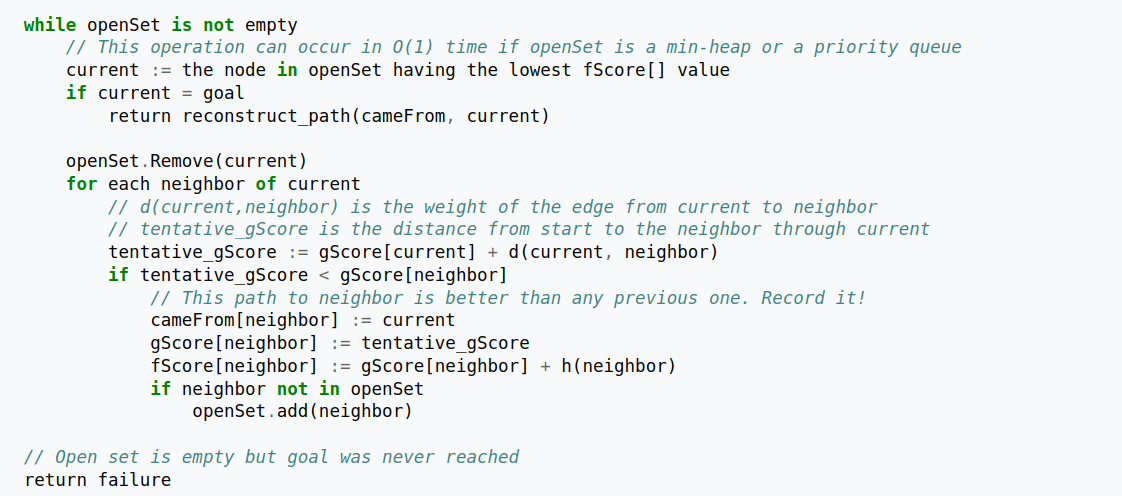


### 2.3.2 Hiện thực

Bây giờ hãy bắt đầu xây dựng thuật toán.

Wikipedia đã cung cấp cho ta hướng dẫn cũng như là mã giả để ta có thể xây dựng thuật toán. Từ đó thì ta có thể chỉnh sửa lại để phù hợp với yêu cầu đề bài.

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



Quan trọng ở đây là vòng lặp. Vì ta phải hiểu là thuật toán sẽ như một chu trình lặp lại để kiểm tra điều kiện và tìm đường đi ngắn nhất đến đích.

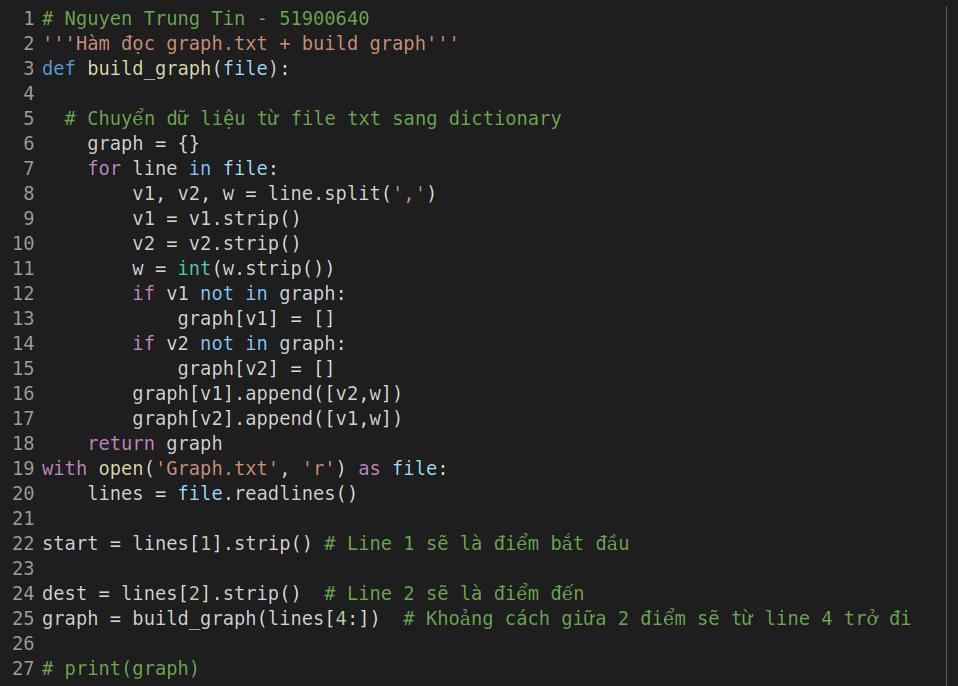
Điều thứ 2 ta cần phải quan tâm đó chính là điều kiện duyệt Node của thuật toán A\* dựa trên công thức: f(n) = h(n)+g(n).

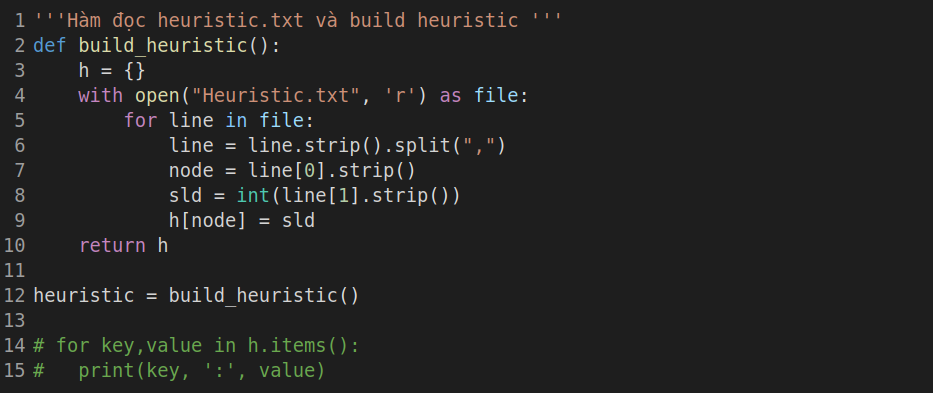
* h(h) chính là Heuristic, khoảng cách đường chim bay giữa Node đang duyệt tới Node đích.
* g(n) có nghĩa là cost (Khoảng cách từ điểm bắt đầu đến Node đang duyệt).
* f(n) là chỉ số mà ta cần phải xem xét giá trị nhỏ nhất để làm current Node.

Điều thứ 3 cần qua tâm đó chính là ta cần tạo một closed\_set và opend\_set để biết được Node nào đã duyệt Node nào đang trong hàng đợi cần phải duyệt.

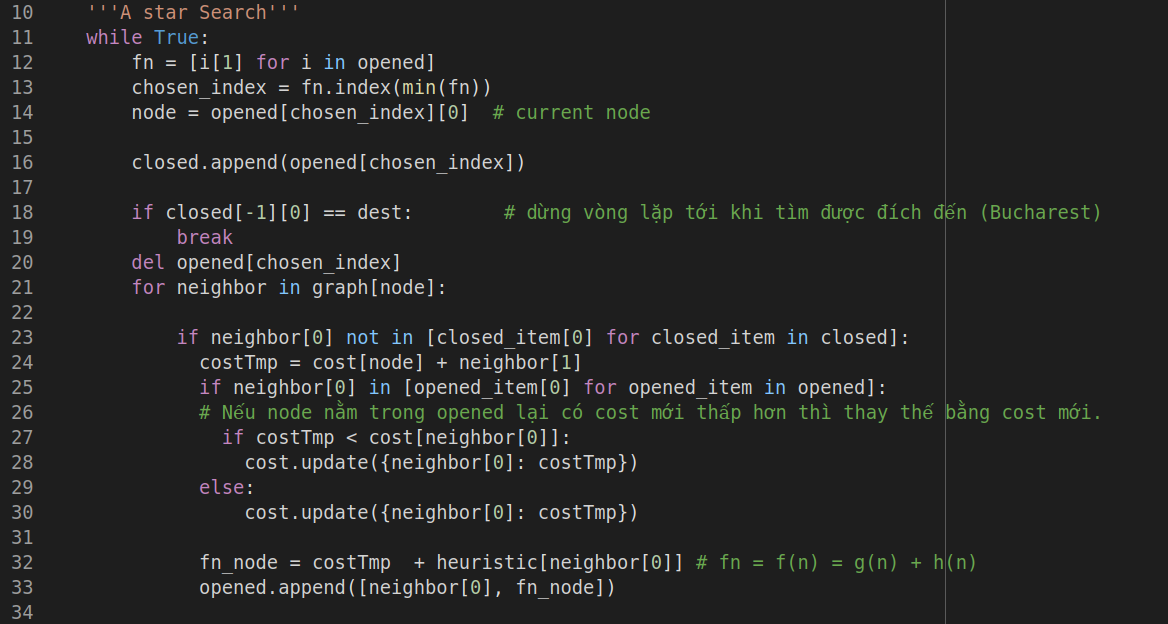
Và điều cuối cùng ta cần phải quan tâm đó chính là tìm lại đường đi sau khi thuật toán đã tìm đến đích.

Thuật toán để chuẩn bị input từ các file data:

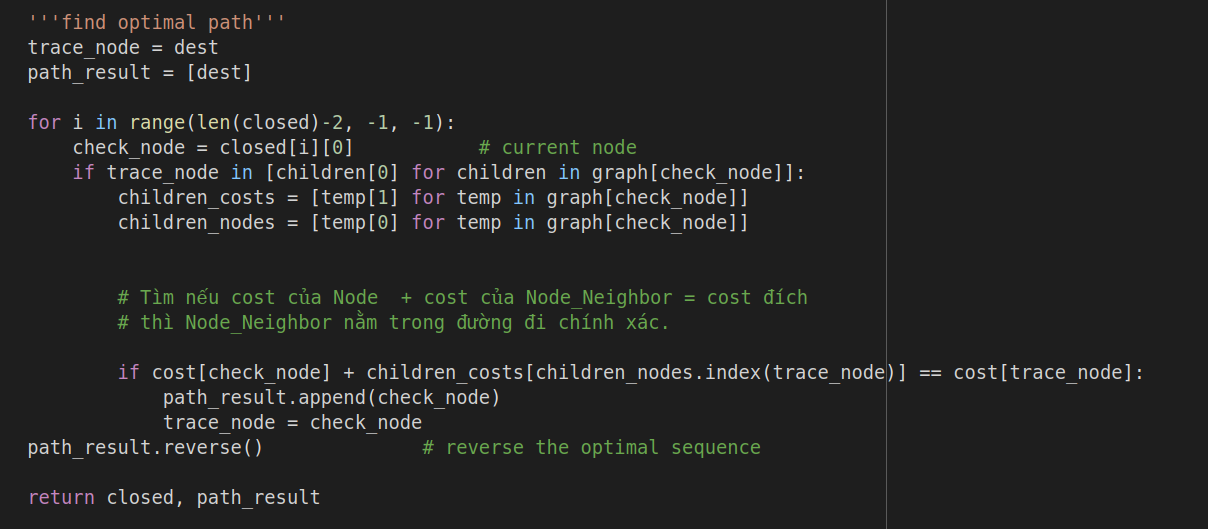




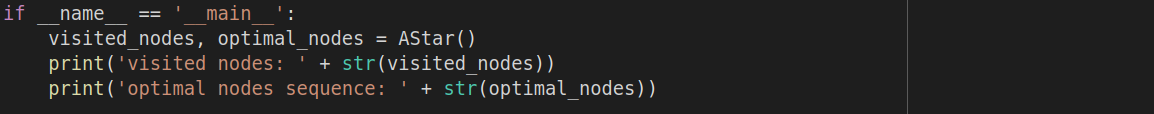
Thuật toán A\* khi đã viết như sau:



Còn đây là thuật toán để tìm lại đường đi và tìm những Node đã duyệt.



Cuối cùng ta chạy chương trình như sau:



### 2.3.3 Kết quả và thảo luận

* Test case 1: từ Arad đến Bucharest

**visited nodes: [['Arad', 366], ['Sibiu', 393], ['Rimnicu Vilcea', 413], ['Fagaras', 415], ['Pitesti', 417], ['Bucharest', 418]]**

**optimal nodes sequence: ['Arad', 'Sibiu', 'Rimnicu Vilcea', 'Pitesti', 'Bucharest']**

* Test case 2: Từ Dobreta đến Bucharest

**visited nodes: [['Dobreta', 366], ['Craiova', 280], ['Mehadia', 316], ['Pitesti', 358], ['Bucharest', 359]]**

**optimal nodes sequence: ['Dobreta', 'Craiova', 'Pitesti', 'Bucharest']**

* Test case 3: Từ Oradea đến Bucharest

**visited nodes: [['Oradea', 366], ['Sibiu', 404], ['Rimnicu Vilcea', 424], ['Fagaras', 426], ['Pitesti', 428], ['Bucharest', 429]]**

**optimal nodes sequence: ['Oradea', 'Sibiu', 'Rimnicu Vilcea', 'Pitesti', 'Bucharest']**

# Tự đánh giá.

| Số câu | Điểm tối đa | Điểm đạt được | Test case | Ghi chú |
| --- | --- | --- | --- | --- |
| Câu 1a: BFS | 2.5đ | 2.25đ | 3/3 | Thuật toán chưa được tối ưu |
| Câu 1b: UCS | 2.5đ | 2.0đ | 3/3 | Thuật toán chưa được tối ưu |
| Câu 2a: Gready BFS | 2.5đ | 2.0đ | 3/3 | Thuật toán chưa được tối ưu |
| Câu 2b: A\* | 2.5đ | 2.25đ | 3/3 | Thuật toán chưa được tối ưu |