**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**LAB 1 - SEARCH**

**MÔN: CƠ SỞ TRÍ TUỆ NHÂN TẠO**

**| Giáo viên hướng dẫn |**

**Thầy: Hoàng Xuân Trường**

**Sinh viên thực hiện: CAO TẤT CƯỜNG - 18120296**

**Chuyên ngành: Khoa học máy tính**

Thành phố Hồ Chí Minh – 2020

# MỤC LỤC

[MỤC LỤC 2](#_Toc54730119)

[Phần I: Tự đánh giá 3](#_Toc54730120)

[Phần II: Lý thuyết cơ bản của các thuật toán 4](#_Toc54730121)

[2.1. Breadth First Search 4](#_Toc54730122)

[2.2. Depth First Search 5](#_Toc54730123)

[2.3. Uniform Cost Search 6](#_Toc54730124)

[2.4. A\* 7](#_Toc54730125)

[2.5. Greedy Best First Search 8](#_Toc54730126)

[Phần 3: Chi tiết và minh họa các thuật toán 9](#_Toc54730127)

[3.1 Breadth First Search 9](#_Toc54730128)

[3.2 Depth First Search 11](#_Toc54730129)

[3.3 Uniform Cost Search 13](#_Toc54730130)

[3.4 A\* 15](#_Toc54730131)

[3.5 Greedy Best First Search 17](#_Toc54730132)

[PHỤ LỤC 19](#_Toc54730133)

[TÀI LIỆU THAM KHẢO 20](#_Toc54730134)

# Phần I: Tự đánh giá

|  |  |  |
| --- | --- | --- |
| Thuật toán | Mức độ hoàn thành | Điểm |
| Breadth First Search | 100% | 10 |
| Depth First Search | 100% |
| Uniform Cost Search | 100% |
| A\* | 100% |
| Greedy Best First Search | 100% |

# Phần II: Lý thuyết cơ bản của các thuật toán

## Breadth First Search

#### Ý tưởng

Từ một đỉnh (nút) gốc ban đầu. Xác định và lần lượt duyệt các đỉnh kề xung quanh đỉnh gốc vừa xét. Tiếp tục quá trình duyệt qua các đỉnh kề đỉnh vừa xét cho đến khi đạt được kết quả cần tìm hoặc duyệt qua tất cả các đỉnh.

#### Thuật toán

Thuật toán sử dụng [hàng đợi](https://vi.wikipedia.org/wiki/H%C3%A0ng_%C4%91%E1%BB%A3i) để lưu trữ thông tin trung gian trong quá trình tìm kiếm:

Bước 1: Chèn đỉnh gốc vào hàng đợi

Bước 2: Lấy ra đỉnh đầu tiên trong hàng đợi và quan sát nó

* Nếu đỉnh này chính là đỉnh đích, dừng quá trình tìm kiếm và trả về kết quả.
* Nếu không phải thì chèn tất cả các đỉnh kề với đỉnh vừa thăm nhưng chưa được quan sát trước đó vào hàng đợi.

Bước 3: Nếu hàng đợi là rỗng, thì tất cả các đỉnh có thể đến được đều đã được quan sát – dừng việc tìm kiếm và trả về "không thấy".

Bước 4: Nếu hàng đợi không rỗng thì quay về bước 2.

#### Độ phức tạp

Với hệ số phân nhánh là b và độ sâu tối đa là d

* **Độ phức tạp thời gian:** 1 + b + b2 + ... + bd = O(bd)
* **Độ phức tạp không gian:** O(bd)

**Nhận xét:**

* **Ưu điểm**
* Đường đi tìm được đi qua ít đỉnh nhất.
* Nếu số đỉnh là hữu hạn, thuật toán chắc chắn tìm ra kết quả.
* BFS sẽ không bao giờ bị mắc kẹt trong con hẻm mù mịt, có nghĩa là các nút không mong muốn.
* Nếu có nhiều hơn một giải pháp thì nó sẽ tìm ra giải pháp với các bước tối thiểu.
* **Khuyết điểm**
* 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.
* Tìm kiếm lời giải theo thuật toán đã định trước, do vậy tìm kiếm một cách máy móc; khi không có thông tin hổ trợ cho quá trình tìm kiếm, không nhận ra ngay lời giải.
* Không hiệu quả nếu lời giải ở sâu. Phương pháp này không phù hợp cho trường hợp có nhiều đường dẫn đến kết quả nhưng đều sâu.

## Depth First Search

#### Ý tưởng

Từ đỉnh (nút) gốc ban đầu. Duyệt đi xa nhất theo từng nhánh. Khi nhánh đã duyệt hết, lùi về từng đỉnh để tìm và duyệt những nhánh tiếp theo. Quá trình duyệt chỉ dừng lại khi tìm thấy đỉnh cần tìm hoặc tất cả đỉnh đều đã được duyệt qua.

#### Thuật toán

#### 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 đầu tập Open. Quay lại bước 2.

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

#### Độ phức tạp

Với hệ số phân nhánh là b, độ sâu tối đa là d và chiều dài của con đường dài nhất là m

* **Độ phức tạp thời gian:** 1 + b + b2 + ... + bd = O(bd)
* **Độ phức tạp không gian:** O(bm)

**Nhận xét**

* **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.
* Yêu cầu bộ nhớ là Nút WRT tuyến tính.
* Ít phức tạp về thời gian và không gian hơn là BFS.
* **Khuyết điểm**
* Mang tính chất vét cạn, không nên áp dụng nếu duyệt số đỉnh quá lớn.
* Không đảm bảo rằng nó sẽ cung cấp cho bạn giải pháp.
* Độ sâu cắt nhỏ hơn nên thời gian phức tạp hơn.
* Xác định độ sâu cho đến khi tiến hành tìm kiếm.

## Uniform Cost Search

#### Ý tưởng

Tìm kiếm theo chi phí thống nhất là một thuật toán tìm kiếm được sử dụng để duyệt qua một cây hoặc đồ thị có trọng số. Thuật toán này phát huy tác dụng khi có một chi phí khác nhau cho mỗi cạnh. Mục tiêu chính của tìm kiếm chi phí thống nhất là tìm đường dẫn đến nút mục tiêu có chi phí tích lũy thấp nhất. Tìm kiếm chi phí thống nhất mở rộng các nút theo chi phí đường dẫn của chúng tạo thành nút gốc. Thuật toán tìm kiếm chi phí thống nhất được thực hiện bởi hàng đợi ưu tiên. Nó ưu tiên tối đa cho chi phí tích lũy thấp nhất.

#### Thuật toán

Khởi tạo: PQ rỗng, CLOSE rỗng.

Đưa trạng thái ban đầu START vào PQ, độ ưu tiên g(START) = 0

Lặp đến khi PQ rỗng

Lấy một trạng thái n (có g thấp nhất) ra khỏi PQ. Đưa n vào CLOSE.

Nếu n là trạng thái đích GOAL thì “đã tìm thấy”. Dừng thuật toán.

Nếu không, với mỗi trạng thái con n’ chưa xét (n’ không thuộc CLOSE) của n:

Tính độ ưu tiên: g(n’) = g(n) + cost(n, n’) và đưa (n’, g(n’)) vào PQ

Cuối lặp

Thông báo không có đường đi từ START đến GOAL.

#### Độ phức tạp

Với C là chi phí của giải pháp tối ưu

#### Độ phức tạp thời gian: O(b(1 + C / ε))

* **Độ phức tạp không gian:** O(b(1 + C / ε))

**Nhận xét**

* **Ưu điểm**
* Tìm kiếm chi phí thống nhất là tối ưu vì ở mọi trạng thái, con đường có chi phí ít nhất được chọn
* **Khuyết điểm**
* Không quan tâm đến số bước liên quan đến việc tìm kiếm và chỉ quan tâm đến chi phí đường dẫn. Do đó thuật toán này có thể bị mắc kẹt trong một vòng lặp vô hạn.

## A\*

#### Ý tưởng

Để biết những tuyến đường nào có khả năng sẽ dẫn tới đích, A\* sử dụng một "đánh giá heuristic" về khoảng cách từ điểm bất kỳ cho trước tới đích. Trong trường hợp tìm đường đi, đánh giá này có thể là khoảng cách đường chim bay - một đánh giá xấp xỉ thường dùng cho khoảng cách của đường giao thông.Từ trạng thái hiện tại A\* xây dựng tất cả các đường đi có thể đi dùng hàm ước lược khoảng cách Heuristic để đánh giá đường đi tốt nhất có thể đi. Tùy theo mỗi dạng bài khác nhau mà hàm Heuristic sẽ được đánh giá khác nhau. A\* luôn tìm được đường đi ngắn nhất nếu tồn tại đường đi như thế.Thứ tự ưu tiên cho một đường đi được quyết định bởi hàm Heuristic:

f(x) = g(x) + h(x)

* g(x) là chi chi phí của đường đi từ điểm xuất phát cho đến thời điểm hiện tại.
* h(x) là hàm ước lượng chi phí từ đỉnh hiện tại đến đỉnh đích.
* f(x) có giá trị càng thấp thì độ ưu tiên càng cao.

#### Thuật toán

#### Bước 1:

* + Open: = {s}
  + Close: = {}

Bước 2: while (Open !={})

* + Chọn trạng thái (đỉnh) tốt nhất p trong Open (xóa p khỏi Open).
  + Nếu p là trạng thái kết thúc thì thoát.
  + Chuyển p qua Close và tạo ra các trạng thái kế tiếp q sau p.
    - Nếu q đã có trong Open
      * Nếu g(q) > g(p) + Cost(p, q)
        + g(q) = g(p) + Cost(p, q)
        + f(q) = g(q) + h(q)
        + prev(q) = p (đỉnh cha của q là p)
    - Nếu q chưa có trong Open
      * g(q) = g(p) + cost(p, q)
      * f(q) = g(q) + h(q)
      * prev(q) = p
      * Thêm q vào Open
    - Nếu q có trong Close
      * Nếu g(q) > g(p) + Cost(p, q)
        + Bỏ q khỏi Close
        + Thêm q vào Open

Bước 3: Không tìm được.

**Nhận xét**

* **Ưu điểm**
* Đầy đủ và tối ưu.
* Một trong những kỹ thuật tốt nhất. Nó được sử dụng để giải quyết các vấn đề rất phức tạp.
* Nó hiệu quả tối ưu, tức là không có thuật toán tối ưu nào khác được đảm bảo để mở rộng ít nút hơn A \*.
* **Khuyết điểm**
* Đầy đủ và tối ưu.
* Tốc độ thực hiện tìm kiếm A \* phụ thuộc nhiều vào độ chính xác của thuật toán heuristic được sử dụng để tính h (n).
* Thuật toán này hoàn thành nếu hệ số phân nhánh là hữu hạn và mọi hành động đều có chi phí cố định.

**Điểm khác biệt giữa UCS và A\***

UCS thuộc nhóm hàm tìm kiếm mù, duyệt các trạng thái theo mọi hướng và không sử dụng thông tin của trạng thái đích.

Thuật giải A\* thuộc nhóm hàm tìm kiếm có hiểu biết, sử dụng hàm heuristic để ước lượng mức độ gần so với trạng thái đích. Thuật giải A\* là sự kết hợp của UCS và tìm kiếm tham lam.

## Greedy Best First Search

**Ý tưởng**

Thuật toán sẽ sử dụng 1 hàm đánh giá là hàm heuristic h(n). Hàm heuristic h(n) này sẽ đánh giá chi phí để đi từ nút hiện tại n đến nút đích (mục tiêu).

#### Thuật toán

Bước 1: Đặt nút bắt đầu vào danh sách MỞ.

Bước 2: Nếu danh sách MỞ trống, Dừng.

Bước 3: Loại bỏ nút n khỏi danh sách MỞ có giá trị thấp nhất của h(n) và đặt nó vào danh sách đã thăm

Bước 4: Mở rộng nút n và tạo ra các phần tử kế thừa của nút n.

Bước 6: Đối với mỗi nút kế thừa, thuật toán kiểm tra hàm đánh giá f (n) thêm nó vào danh sách MỞ.

Bước 7: Quay lại Bước 2.

#### Độ phức tạp

Với hệ số phân nhánh là b, độ sâu tối đa là d

#### Độ phức tạp thời gian: O(bd)

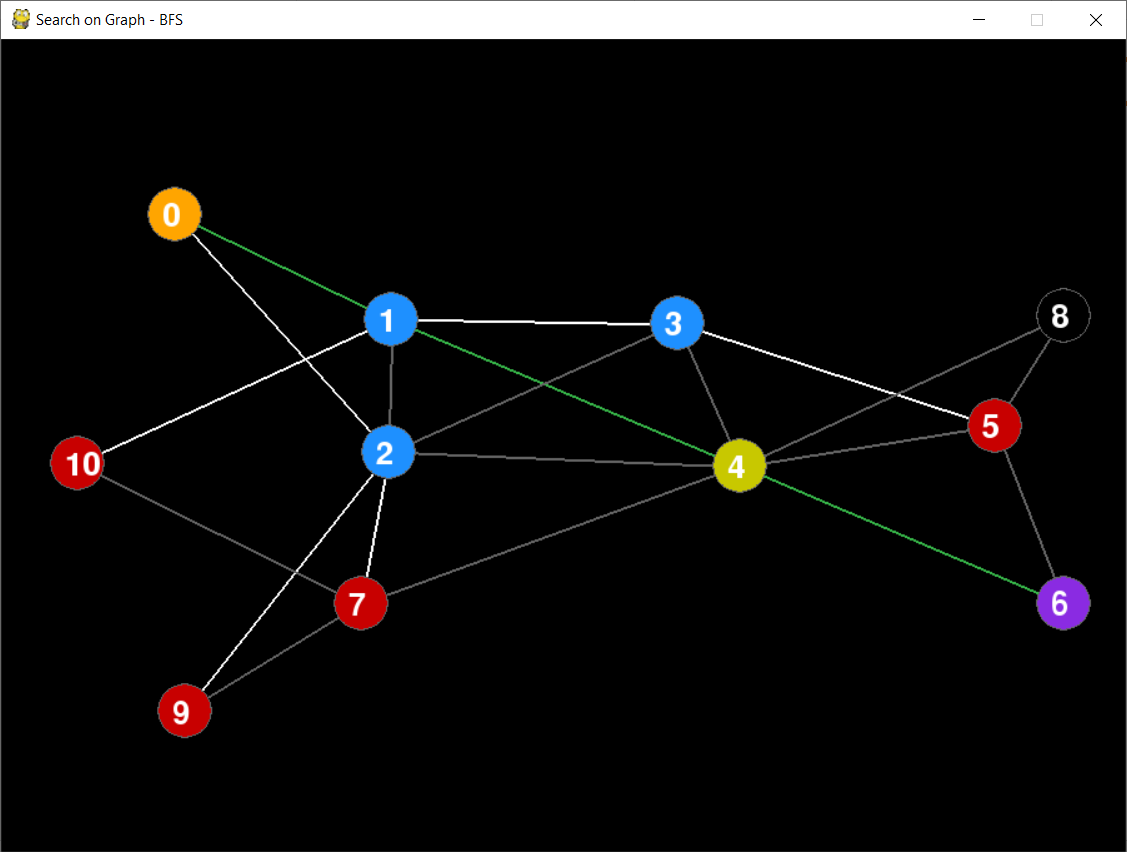
* **Độ phức tạp không gian:** O(bd)

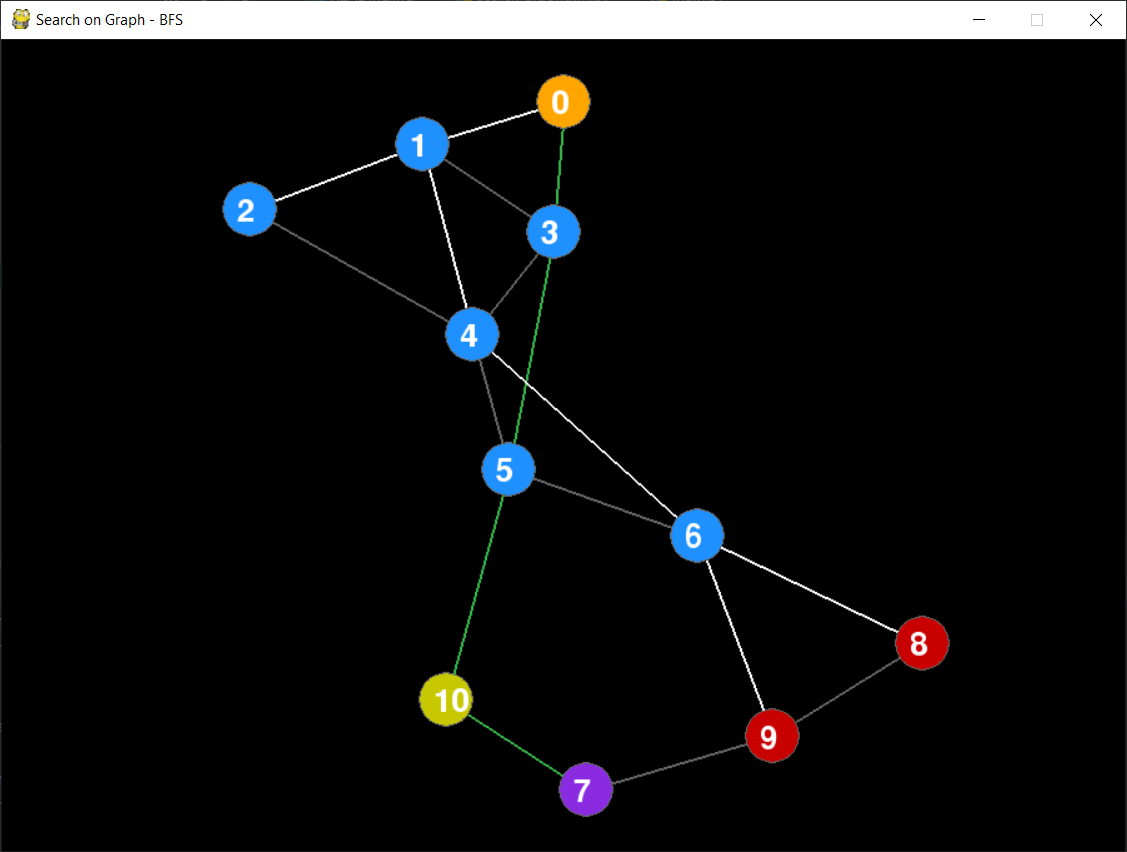
**Nhận xét**

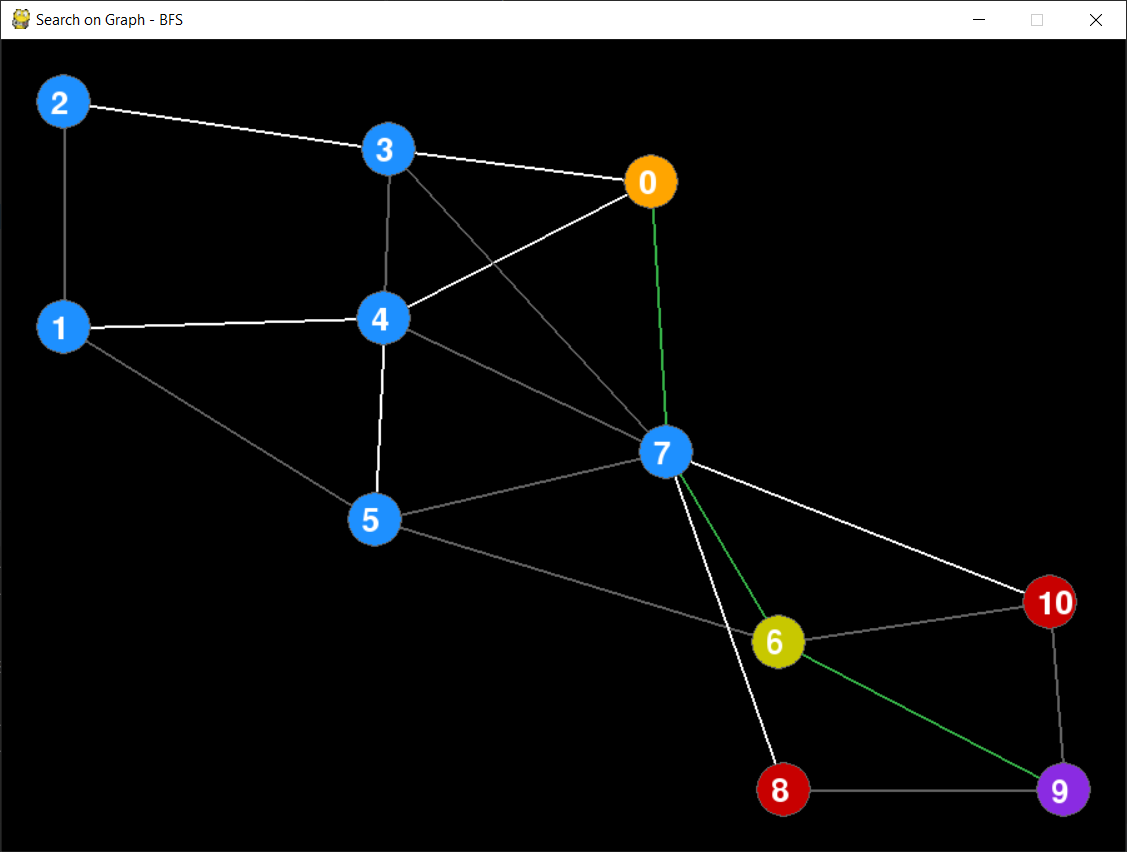
* **Ưu điểm**
* Có thể chuyển đổi giữa BFS và DFS bằng cách đạt được lợi thế của cả hai thuật toán.
* Thuật toán này hiệu quả hơn thuật toán BFS và DFS.
* **Khuyết điểm**
* Thuật toán này không tối ưu.
* Nó có thể bị mắc kẹt trong một vòng lặp như DFS.

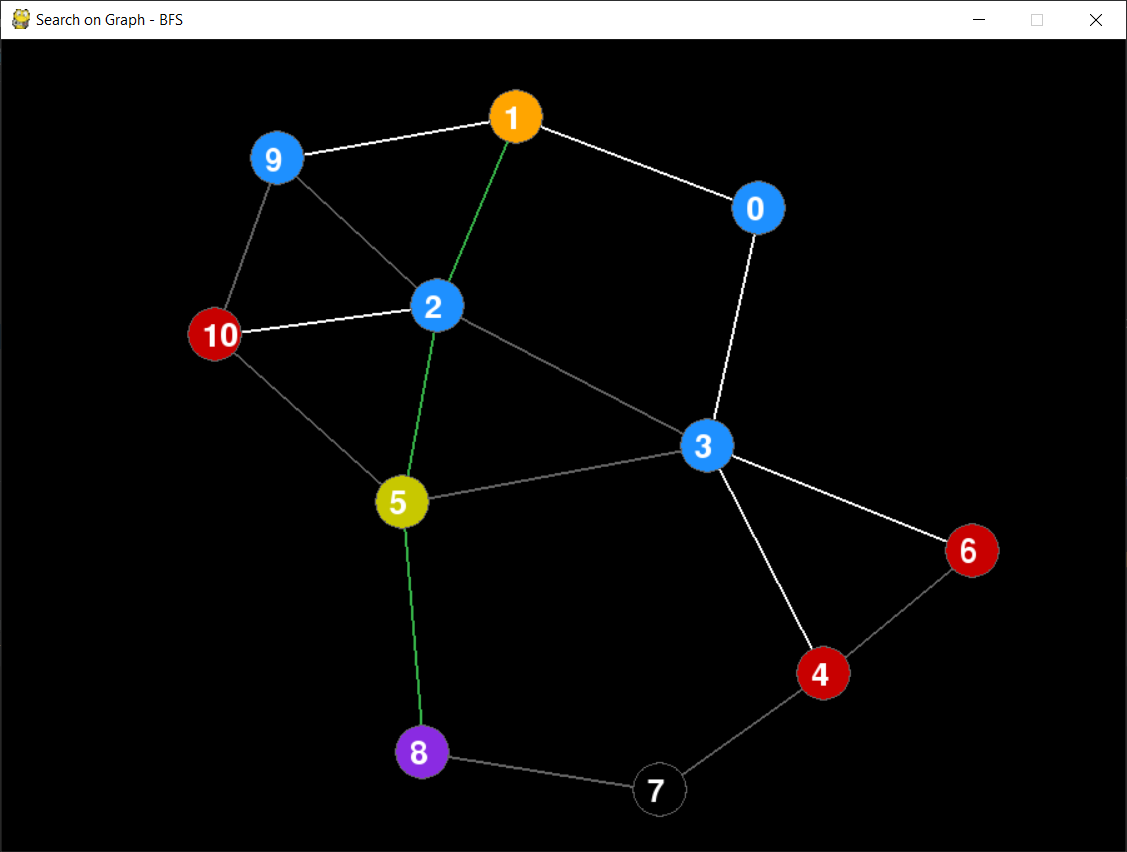
# Phần 3: Chi tiết và minh họa các thuật toán

## 3.1 Breadth First Search

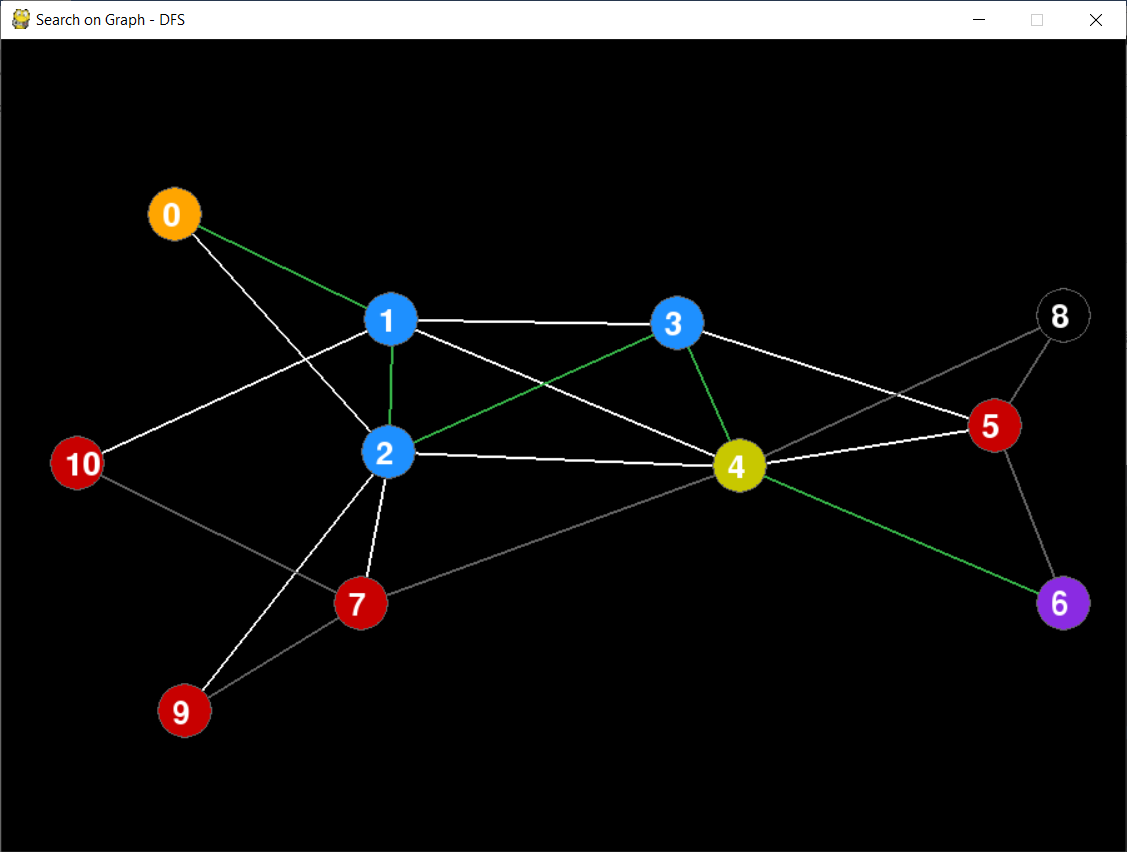
Test case 1

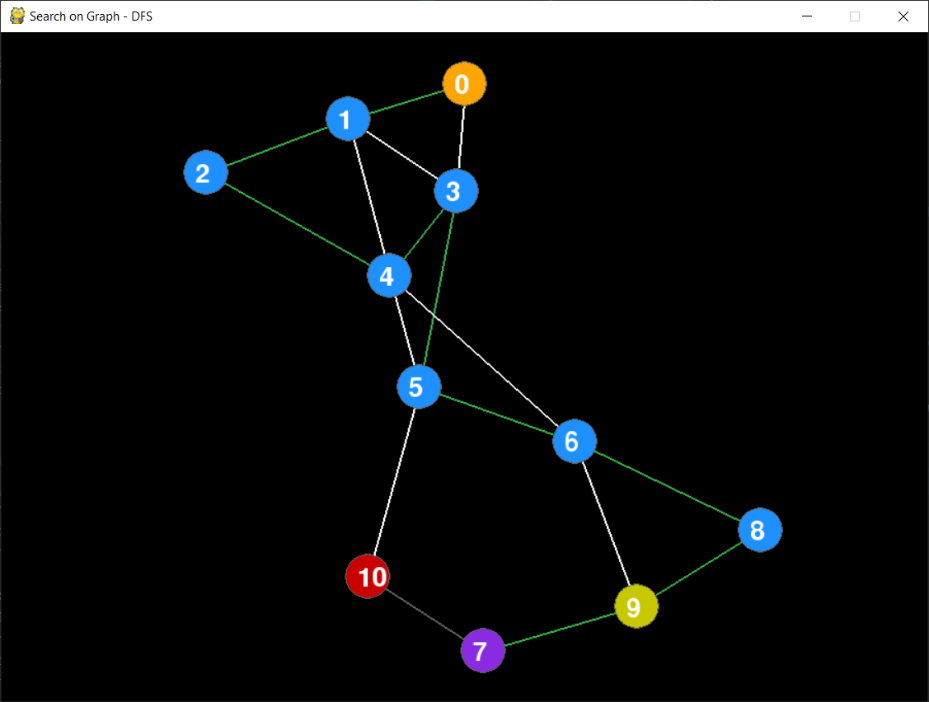
Test case 2:

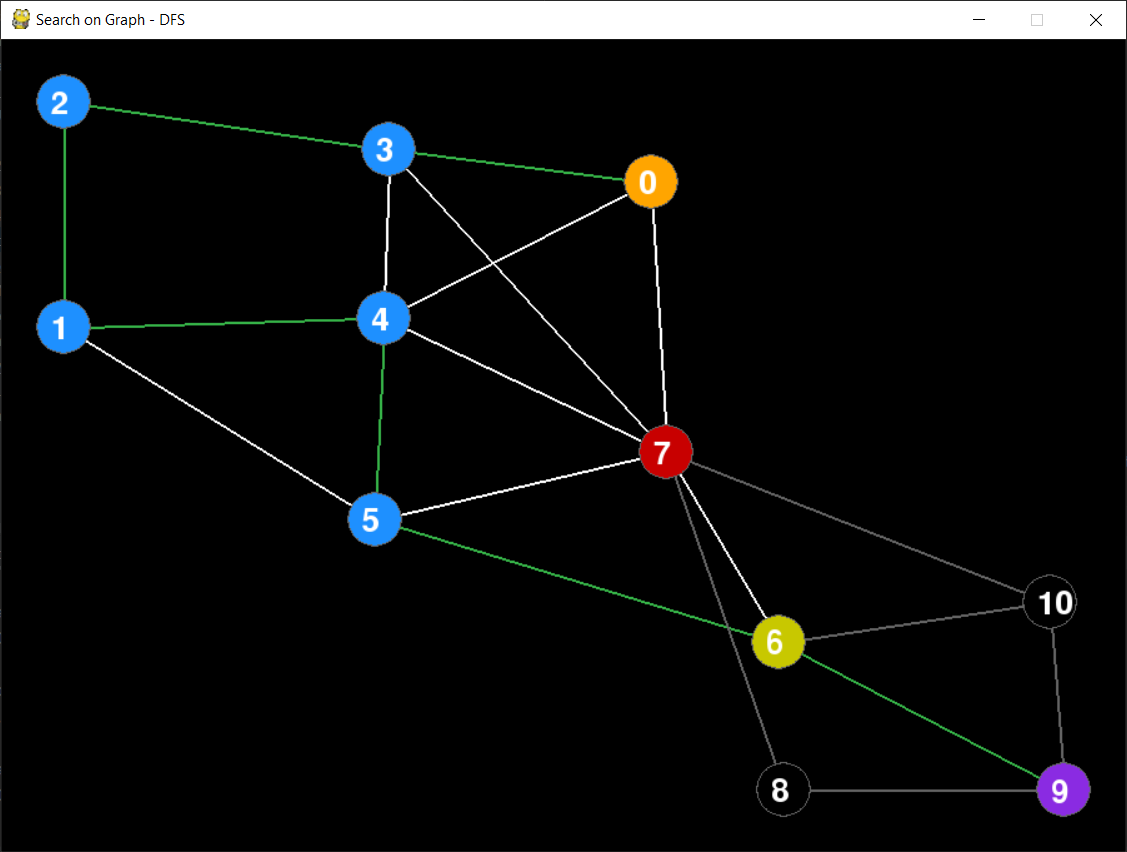
Test case 3

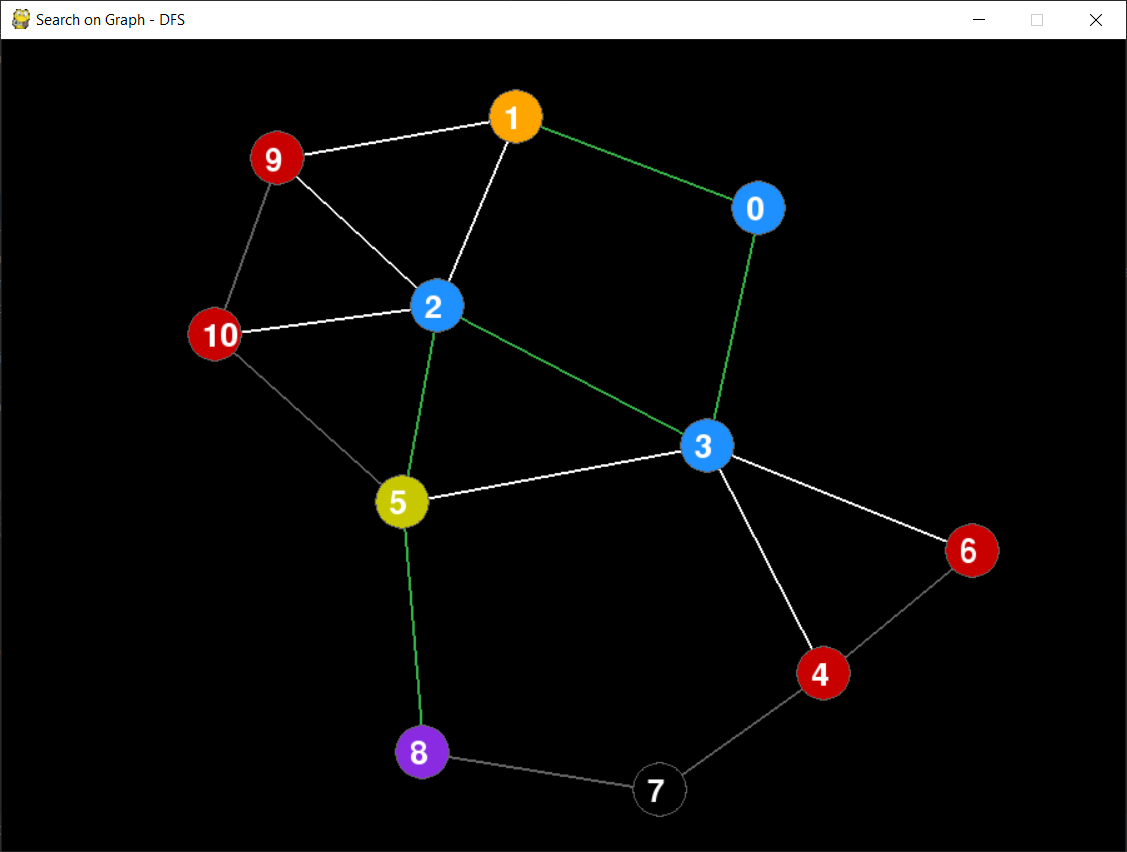
Test case 4

## 3.2 Depth First Search

Test case 1

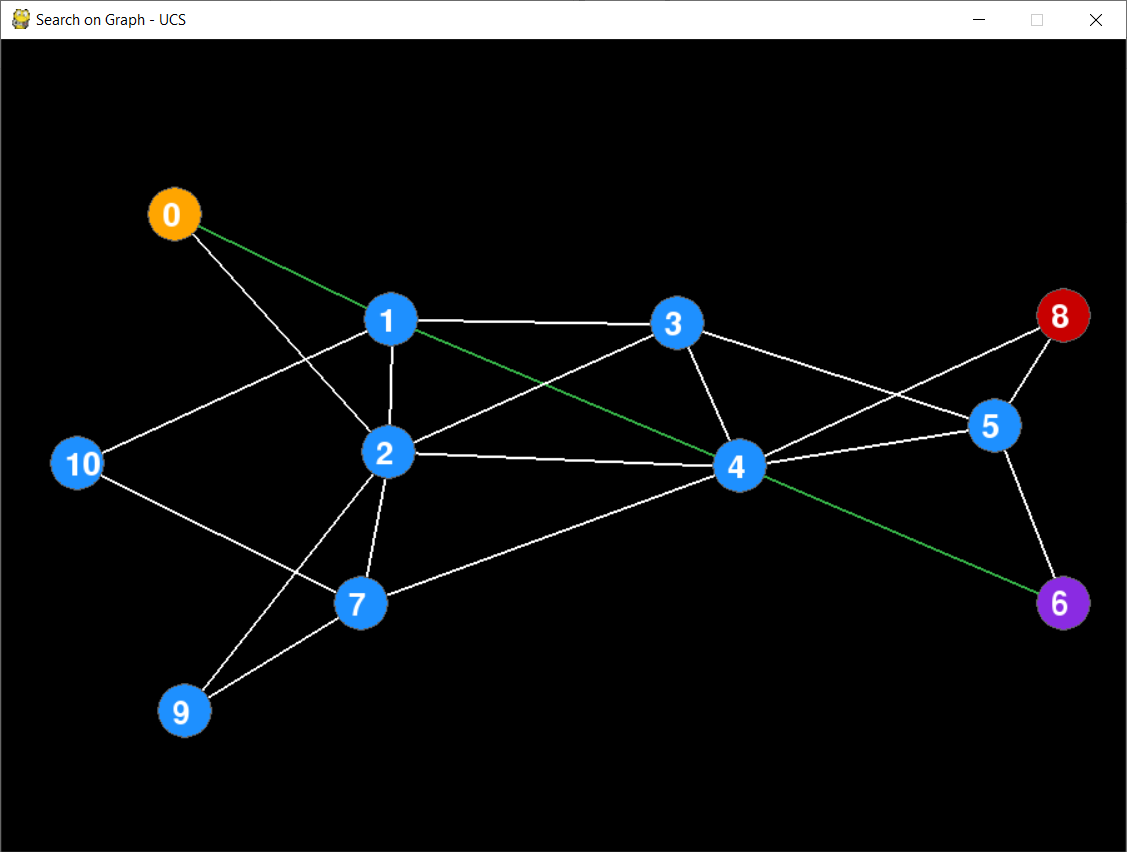
Test case 2

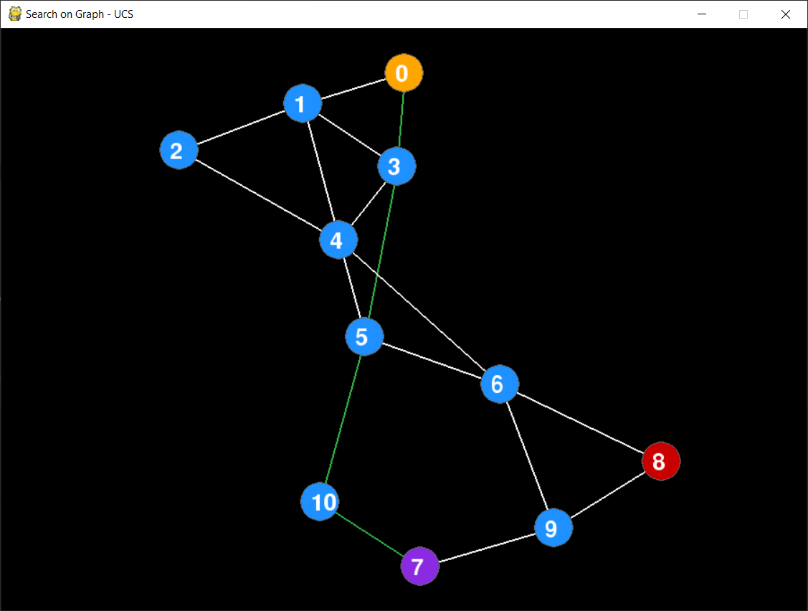
Test case 3

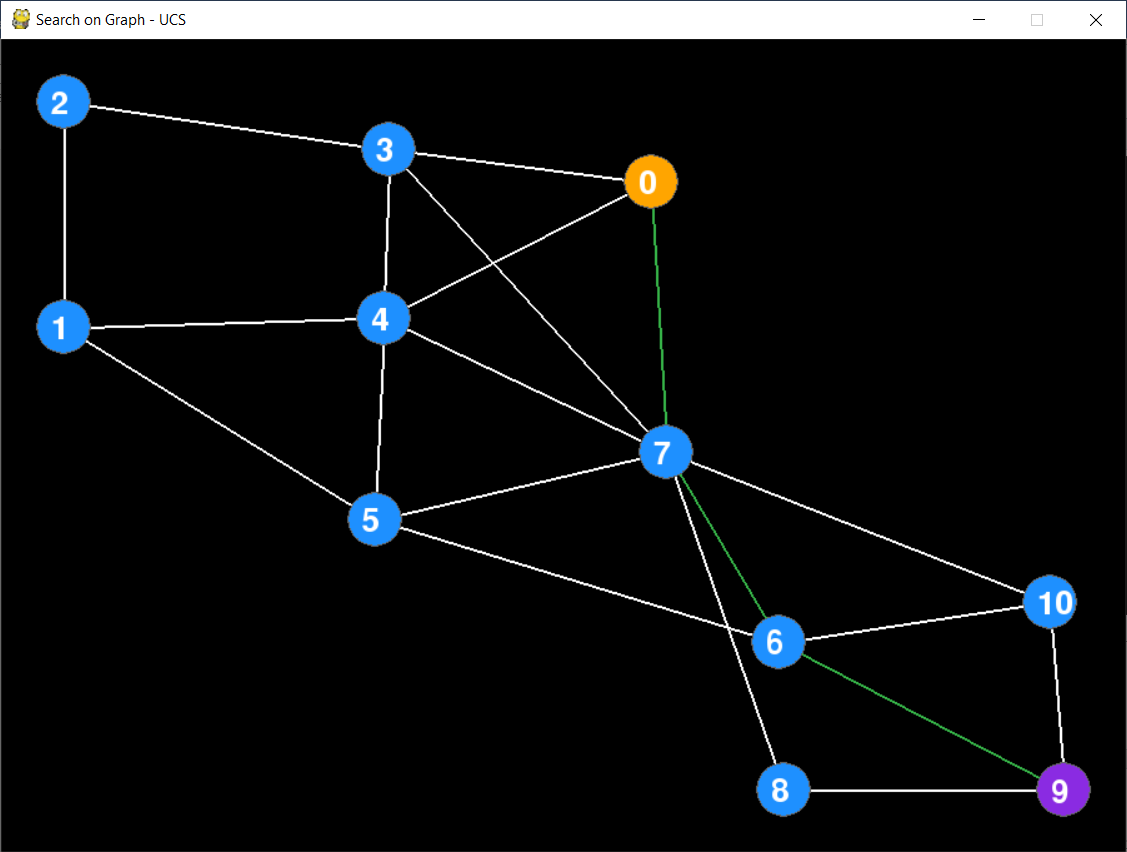
Test case 4

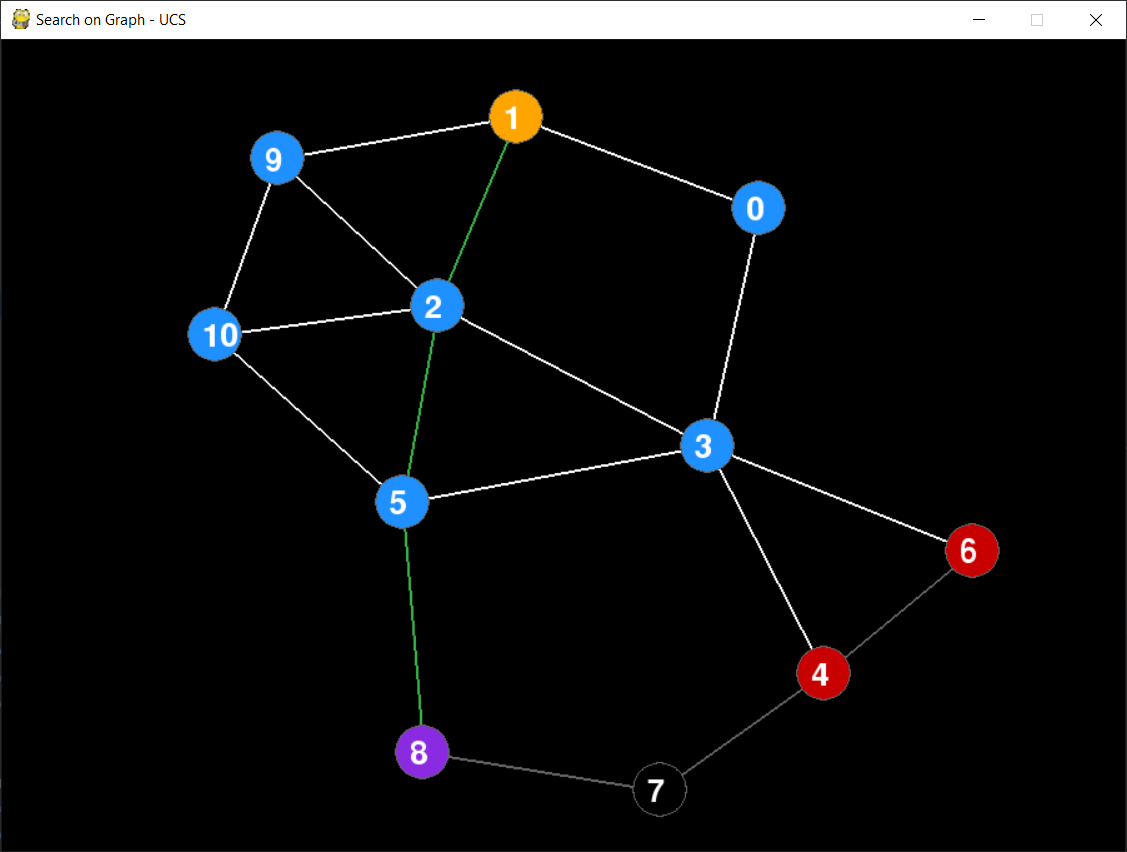
## 3.3 Uniform Cost Search

Chi phí giữa các đỉnh là khoảng cách Euclid giữa các đỉnh đó

Test case 1

Test case 2

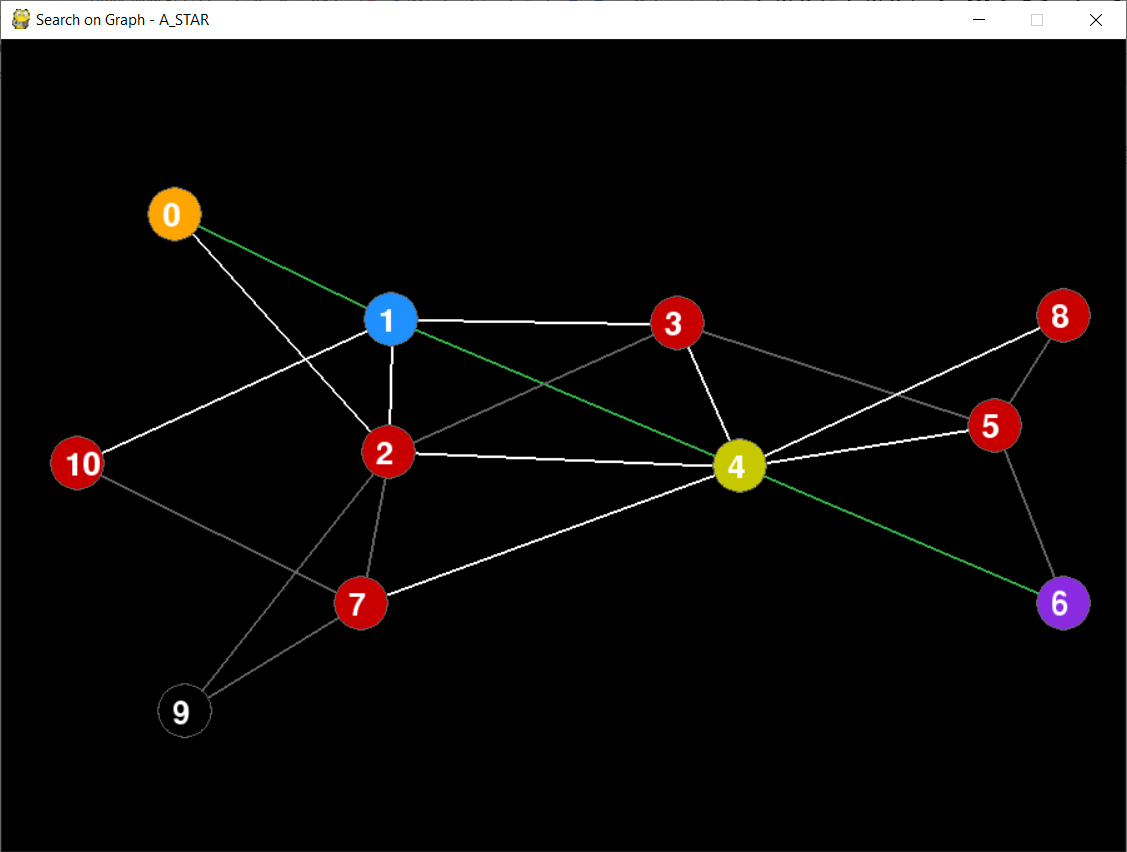
Test case 3

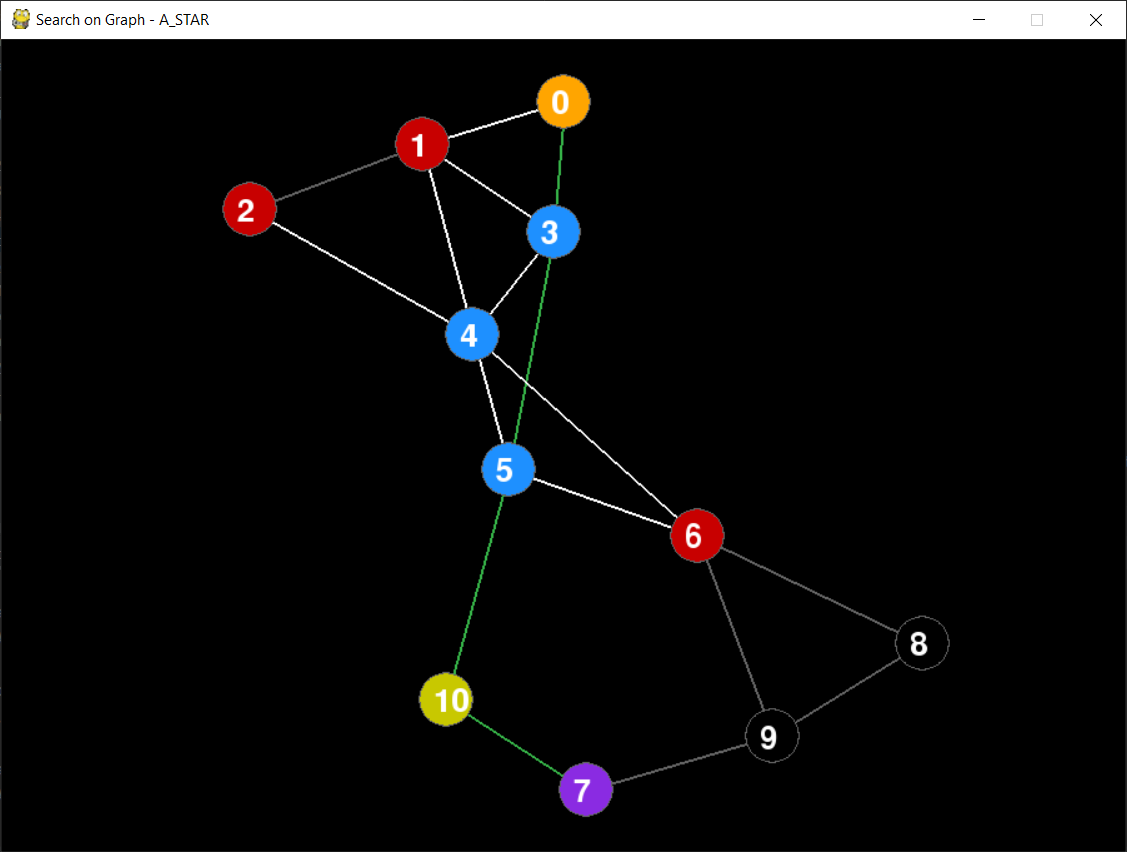
Test case 4

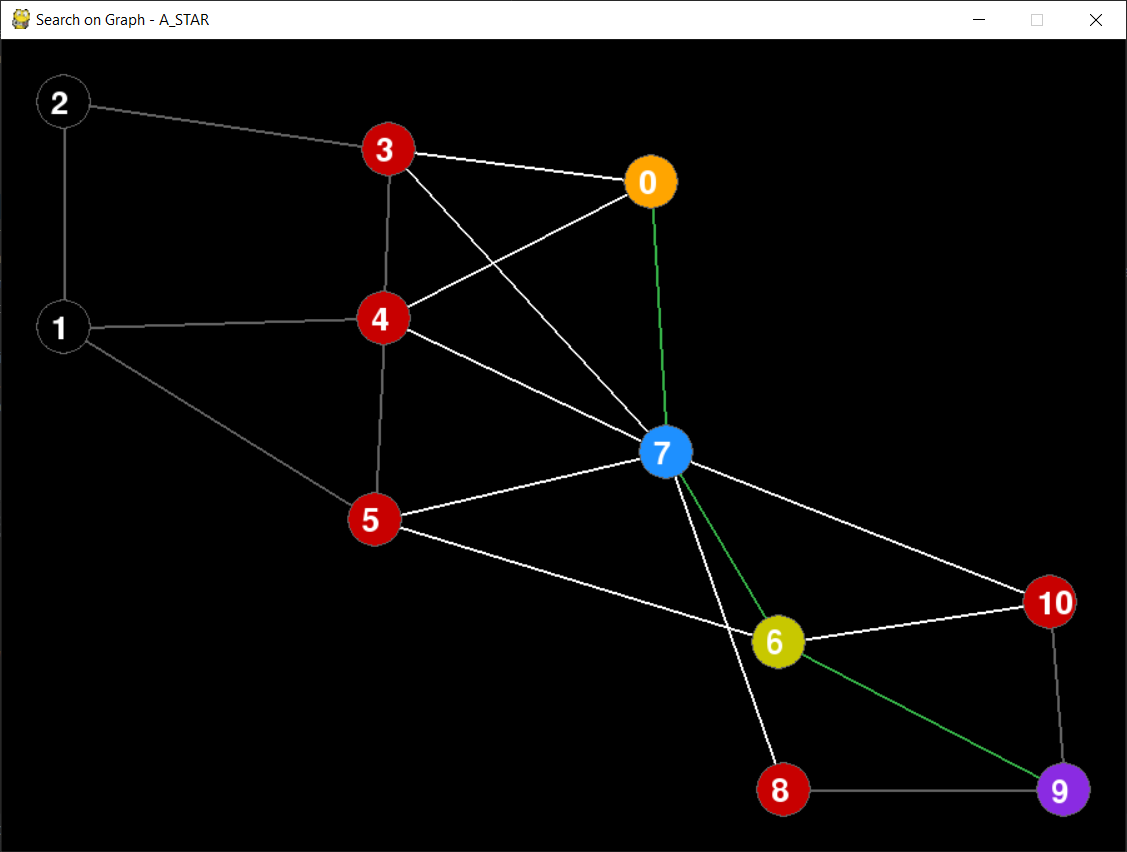
## 3.4 A\*

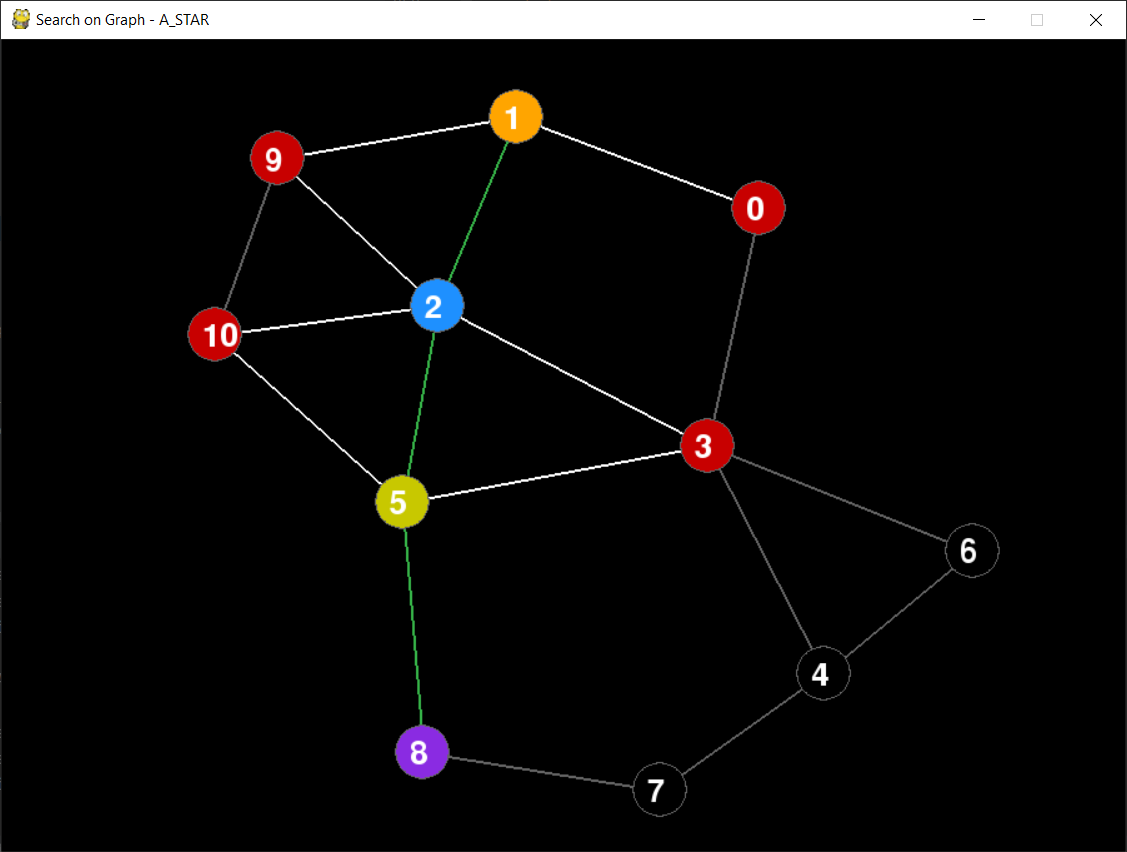
Chi phí giữa các đỉnh là khoảng cách Euclid giữa các đỉnh đó

Hàm heuristic là khoảng cách từ mỗi đỉnh đến đỉnh goal (đường chim bay). Em chọn hàm heuristic này vì nó là một đánh giá xấp xỉ thường dùng cho khoảng cách của đường giao thông.

Test cas 1

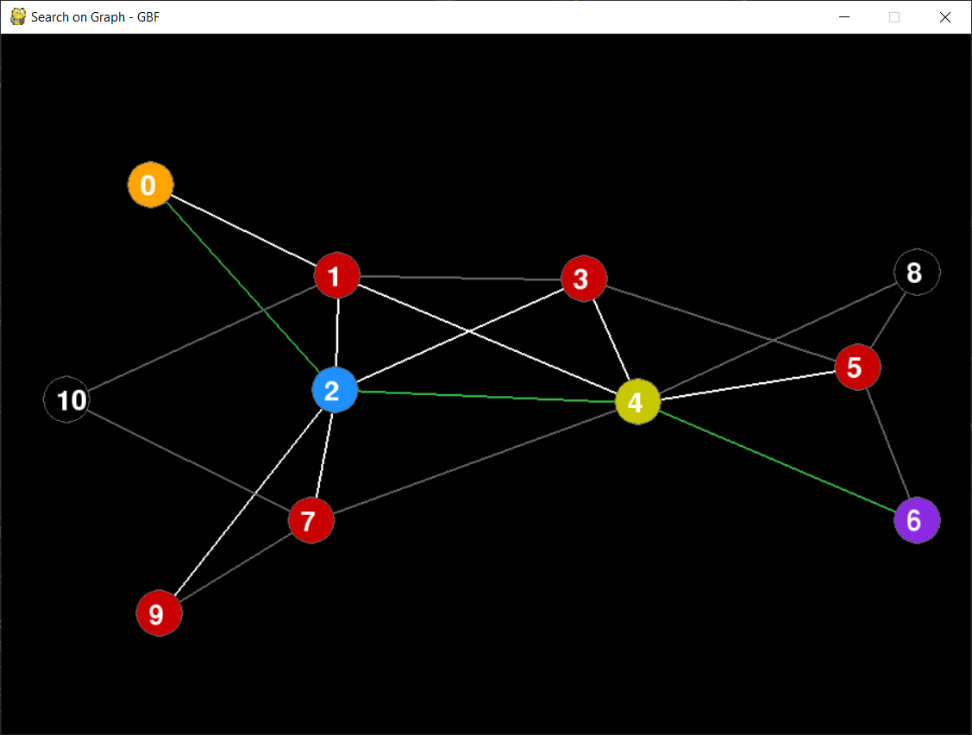
Test case 2

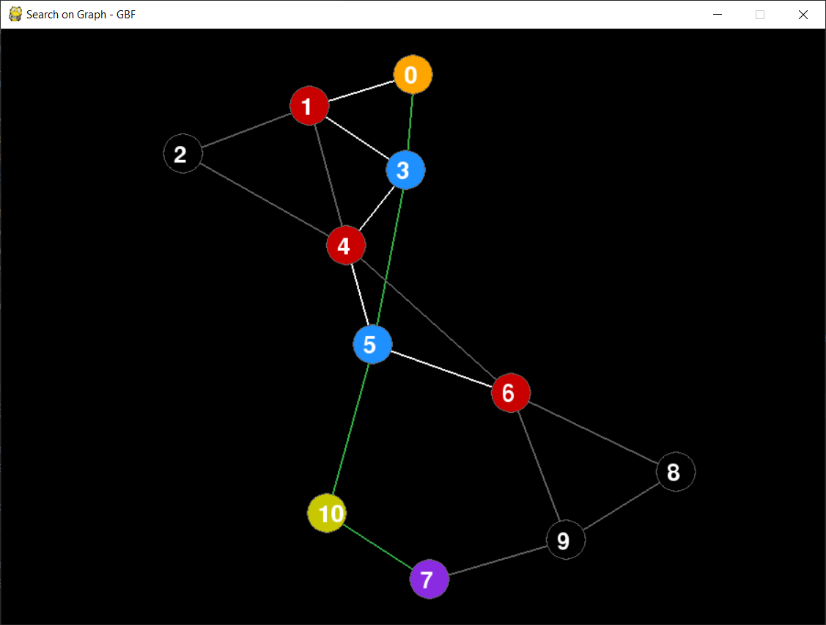
Test case 3

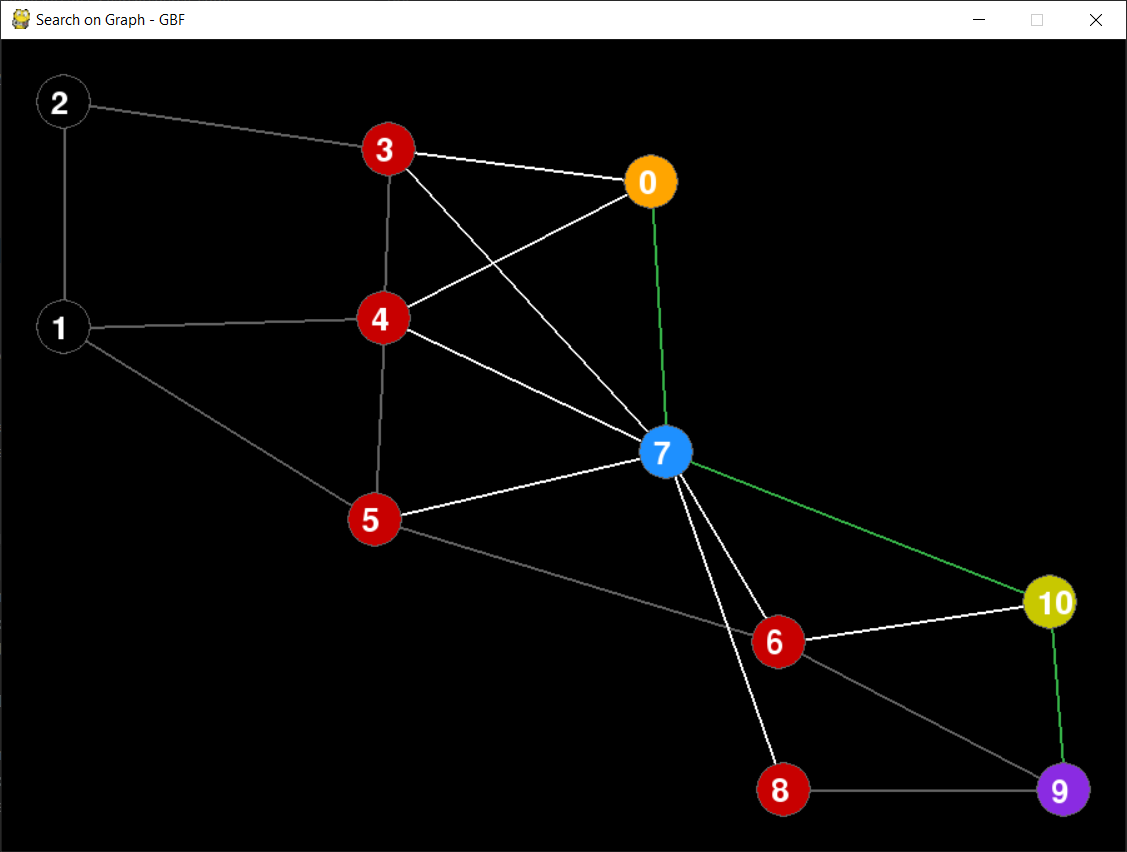
Test case 4

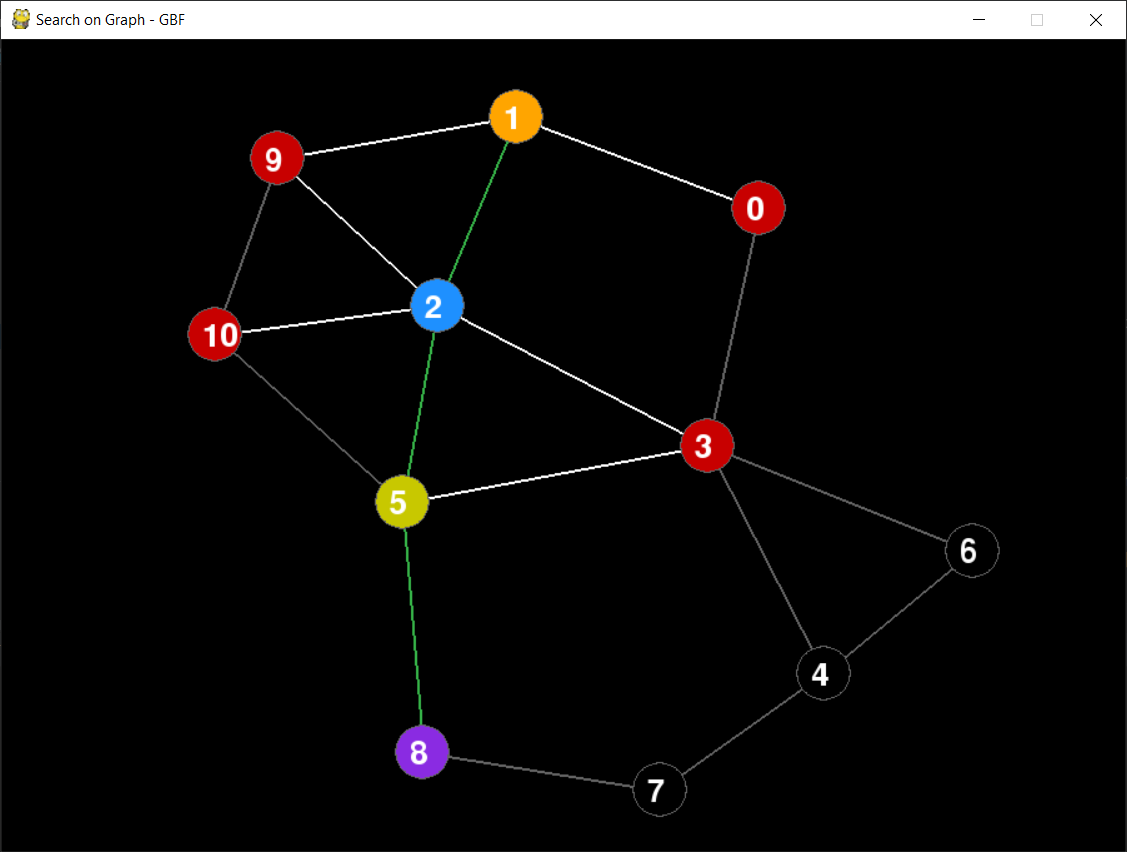
## 3.5 Greedy Best First Search

Hàm heuristic là khoảng cách từ mỗi đỉnh đến đỉnh goal (đường chim bay). Em chọn hàm heuristic này vì nó là một đánh giá xấp xỉ thường dùng cho khoảng cách của đường giao thông.

Test case 1:

Test case 2:

Test case 3

Test case 4

# PHỤ LỤC

\

# TÀI LIỆU THAM KHẢO

<https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/>

<https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/>

<https://www.geeksforgeeks.org/uniform-cost-search-dijkstra-for-large-graphs/>

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

<https://www.geeksforgeeks.org/best-first-search-informed-search/>