KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

# BÁO CÁO ĐỒ ÁN MÔN CƠ SỞ TRÍ TUỆ NHÂN TẠO

**Giảng viên: Hoàng Xuân Trường**

**Họ tên sinh viên: Lê Thanh Viễn**

**MSSV: 18120647**

Nội dung

## I) Mức độ hoàn thành:

| Tên thuật toán | Mức độ hoàn thành |
|---|---|
| **Breadth-first Search (BFS)** | **100%** |
| **Depth-first Search (DFS)** | **100%** |
| **Uniform-cost Search (UCS)** | **100%** |
| **AStar** | **100%** |



## II) Điểm tự đánh giá:

* 9/10

## III) Lý thuyết cơ bản:

### 1) Breadth-first Search (BFS):

* Ý tưởng thuật toán:

- Thuật toán sử dụng một cấu trúc dữ liệu hàng đợi để lưu trữ thông tin trung gian thu được trong quá trình tìm kiếm:

+ B1: Chèn đỉnh gốc vào hàng đợi (đang hướng tới)

+ B2: 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.

+ B3: 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".

+ B4: Nếu hàng đợi không rỗng thì quay về bước 2.

* Độ phức tạp thuật toán:

- Không gian:

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|)

- Thời gian:

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

## 2) Depth-first Search (DFS):

* Ý tưởng thuật toán:

- DFS trên đồ thị vô hướng cũng giống như khám phá mê cung với một cuộn chỉ và một thùng sơn đỏ để đánh dấu, tránh bị lạc. Trong đó mỗi đỉnh s trong đồ thị tượng trưng cho một cửa trong mê cung.



- Ta bắt đầu từ đỉnh s, buộc đầu cuộn chỉ vào s và đánh đấu đỉnh này "đã thăm". Sau đó ta đánh dấu s là đỉnh hiện hành u.

- Bây giờ, nếu ta đi theo cạnh (u,v) bất kỳ.

- Nếu cạnh (u,v) dẫn chúng ta đến đỉnh "đã thăm" v, ta quay trở về u.

- Nếu đỉnh v là đỉnh mới, ta di chuyển đến v và lăn cuộn chỉ theo. Đánh dấu v là "đã thăm". Đặt v thành đỉnh hiện hành và lặp lại các bước.

- Cuối cùng, ta có thể đi đến một đỉnh mà tại đó tất cả các cạnh kề với nó đều dẫn chúng ta đến các đỉnh "đã thăm". Khi đó, ta sẽ quay lui bằng cách cuộn ngược cuộn chỉ và quay lại cho đến khi trở lại một đỉnh kề với một cạnh còn chưa được khám phá. Lại tiếp tục quy trình khám phá như trên.

- Khi chúng ta trở về s và không còn cạnh nào kề với nó chưa bị khám phá là lúc DFS dừng.

* Độ phức tạp thuật toán:

- DFS được gọi đúng 1 lần ứng với mỗi đỉnh.

- Mỗi cạnh được xem xét đúng 2 lần, mỗi lần từ một đỉnh kề với nó.

- Với $n_s$ đỉnh và $m_s$ cạnh thuộc thành phần liên thông chứa s, một phép DFS bắt đầu tại s sẽ chạy với thời gian $O(n_s + m_s)$ nếu:

+ Đồ thị được biểu diễn bằng cấu trúc dữ liệu dạng danh sách kề.

+ Đặt nhãn cho một đỉnh là "đã thăm" và kiểm tra xem một đỉnh "đã thăm" chưa tốn chi phí O(degree).

+ Bằng cách đặt nhãn cho các đỉnh là "đã thăm", ta có thể xem xét một cách hệ thống các cạnh kề với đỉnh hiện hành nên ta sẽ không xem xét một cạnh quá 1 lần.

### 3) Uniform-cost Search (UCS):



* Ý tưởng thuật toán:

- Việc tìm kiếm bắt đầu tại nút gốc.

- 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.

* Độ phức tạp thuật toán:

- Thời gian:

$O(\log(Q) * \min(N, B^L))$

- Không gian:

$O(\min(N, b^L))$

## 4) Astar:

* Ý tưởng thuật toán:

- Xét bài toán tìm đường - bài toán mà A* thường được dùng để giải. A* xây dựng tăng dần tất cả các tuyến đường từ điểm xuất phát cho tới khi nó tìm thấy một đường đi chạm tới đích. Tuy nhiên, cũng như tất cả các thuật toán tìm kiếm có thông tin (informed tìm kiếm thuật toán), nó chỉ xây dựng các tuyến đường "có vẻ" dẫn về phía đích.

- 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.



- Điểm khác biệt của A* đối với tìm kiếm theo lựa chọn tốt nhất là nó còn tính đến khoảng cách đã đi qua. Điều đó làm cho A* "đầy đủ" và "tối ưu", nghĩa là, A* sẽ luôn luôn tìm thấy đường đi ngắn nhất nếu tồn tại một đường đi như vậy. A* không đảm bảo sẽ chạy nhanh hơn các thuật toán tìm kiếm đơn giản hơn. Trong một môi trường dạng mê cung, cách duy nhất để đến đích có thể là trước hết phải đi về phía xa đích và cuối cùng mới quay lại. Trong trường hợp đó, việc thử các nút theo thứ tự "gần đích hơn thì được thử trước" có thể gây tốn thời gian.

* Độ phức tạp thuật toán:

Độ phức tạp thời gian của A* phụ thuộc vào đánh giá heuristic. Trong trường hợp xấu nhất, số nút được mở rộng theo hàm mũ của độ dài lời giải, nhưng nó sẽ là hàm đa thức khi hàm heuristic h thỏa mãn điều kiện sau:

|h(x)-h*(x)| <= O(log h*(x))

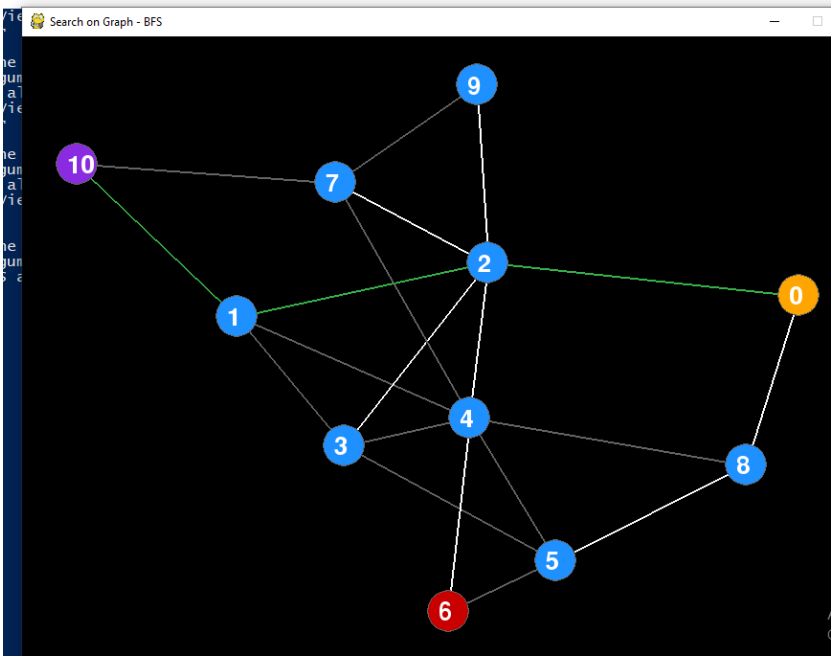
## IV) Điểm khác biệt UCS và A*:

| UCS | A* |
|---|---|
| **Không sử dụng Heuristic – Tìm kiếm mù** | **Sử dụng Heuristic – Tim kiếm có thông tin** |
| **Thời gian tìm kiếm chậm hơn** | **Thời gian tìm kiếm nhanh hơn (với heuristic chấp nhận được)** |
| | |

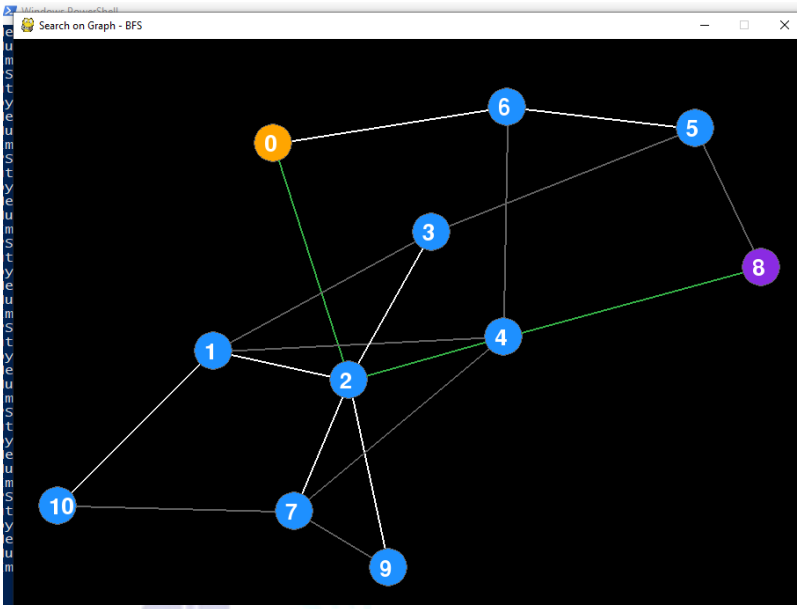
## V) Chi tiết:

### 1) Breadth-first Search (BFS):

Testcase 1:

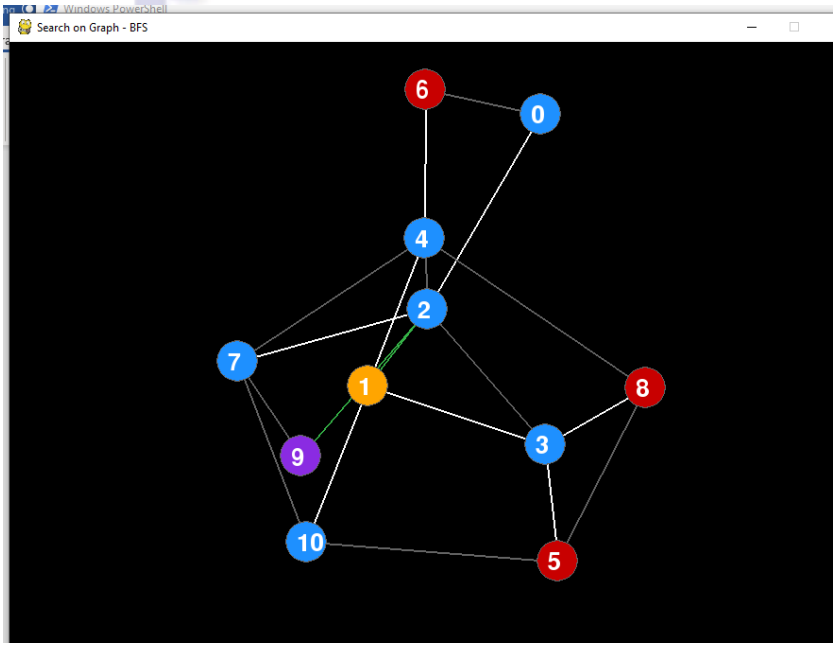


Testcase 2:






Testcase 3:



* 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.

- Khuyết điểm:

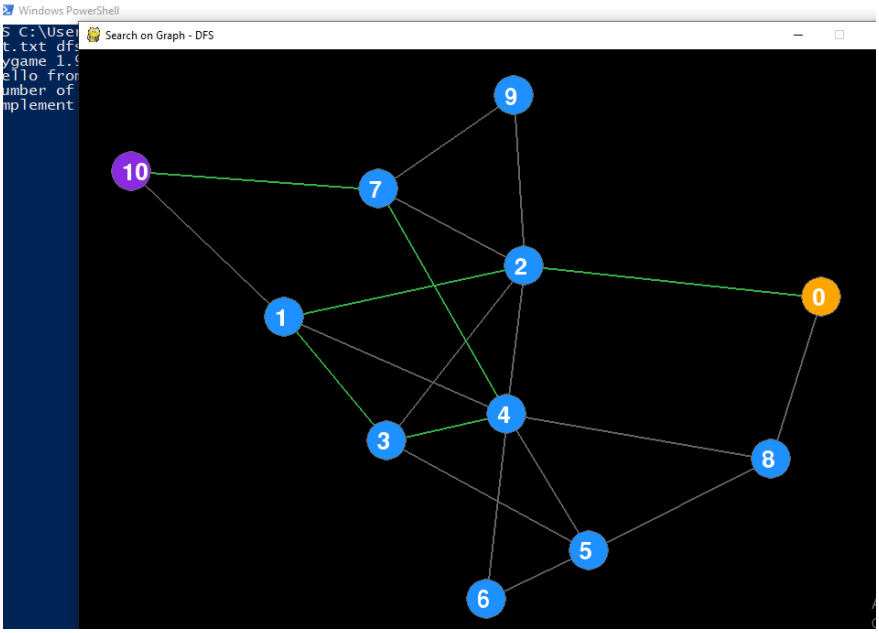
+ Mang tính chất vét cạn, không nên áp dụng nếu duyệt số đỉnh quá lớn.

+ 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.

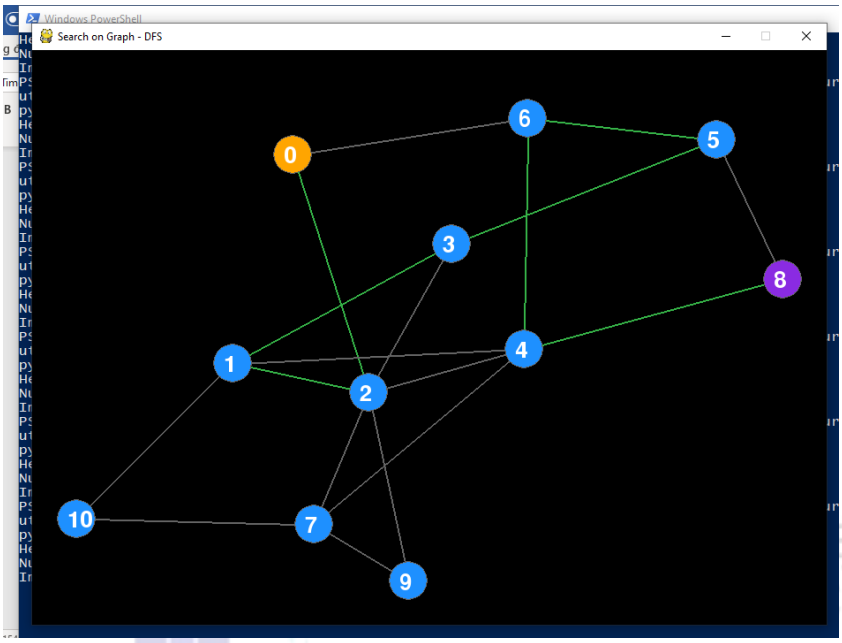


## 2) Depth-first Search (DFS):

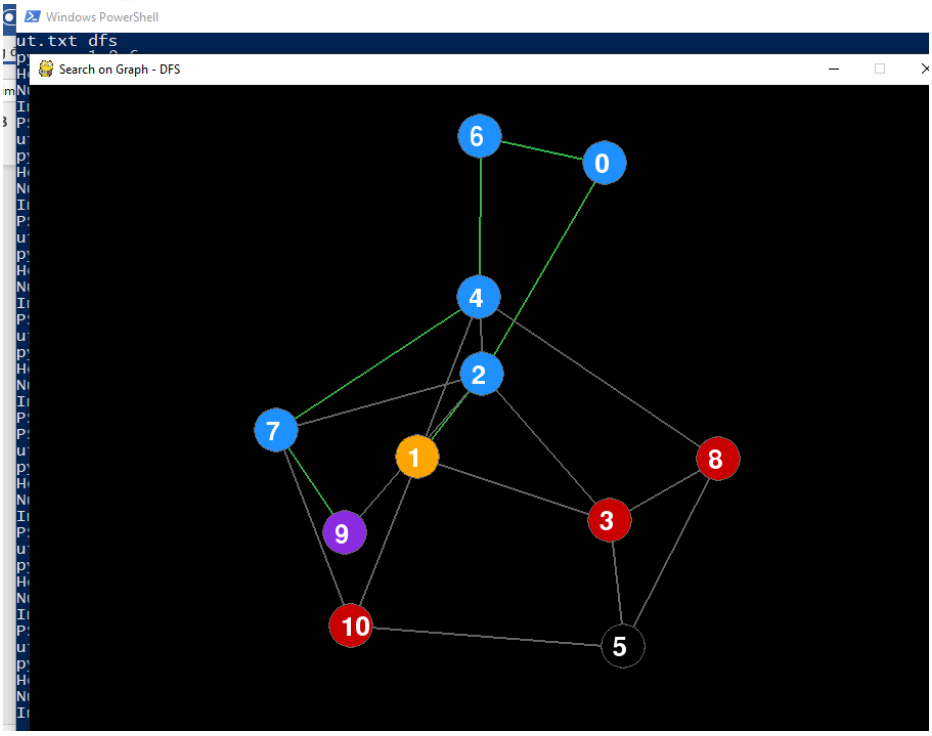
Testcase 1:



Testcase 2:




Testcase 3:



* 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.

- Khuyết điểm:

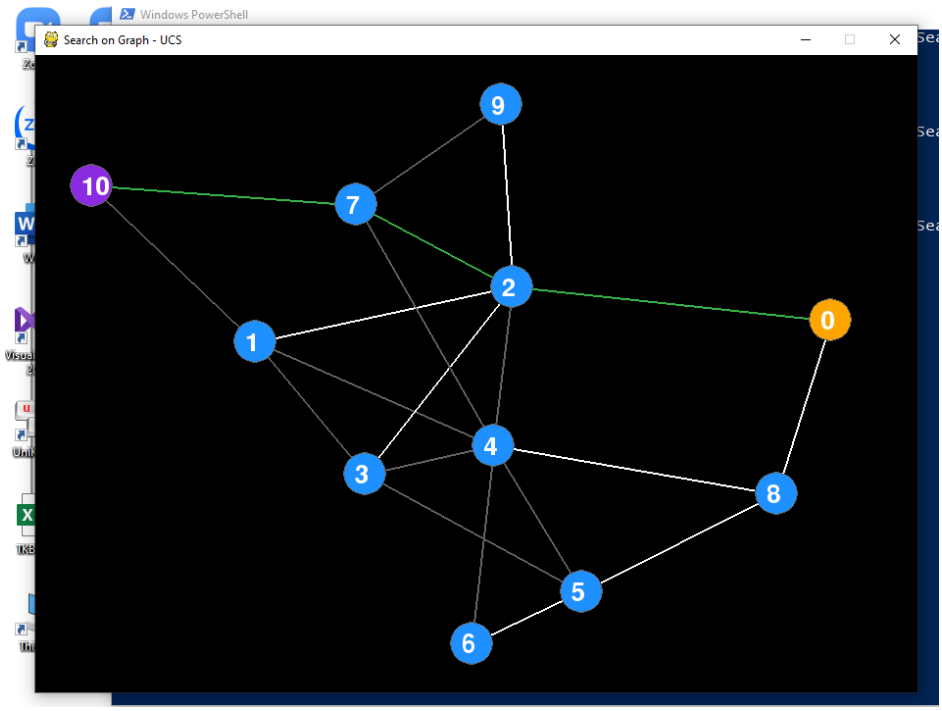
+ Mang tính chất vét cạn, không nên áp dụng nếu duyệt số đỉnh quá lớn.

+ 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.

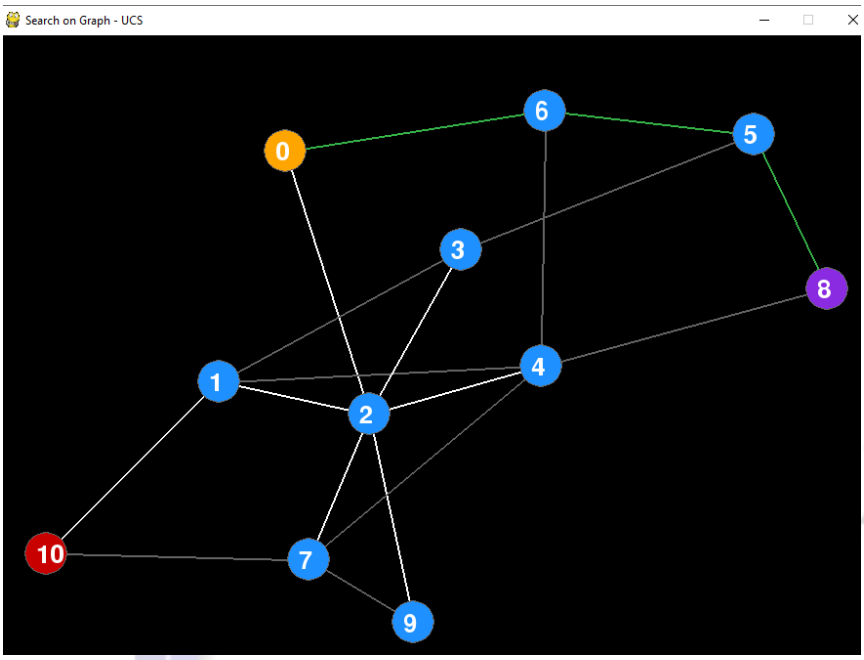


### 3) Uniform-cost Search (UCS):

Testcase 1:

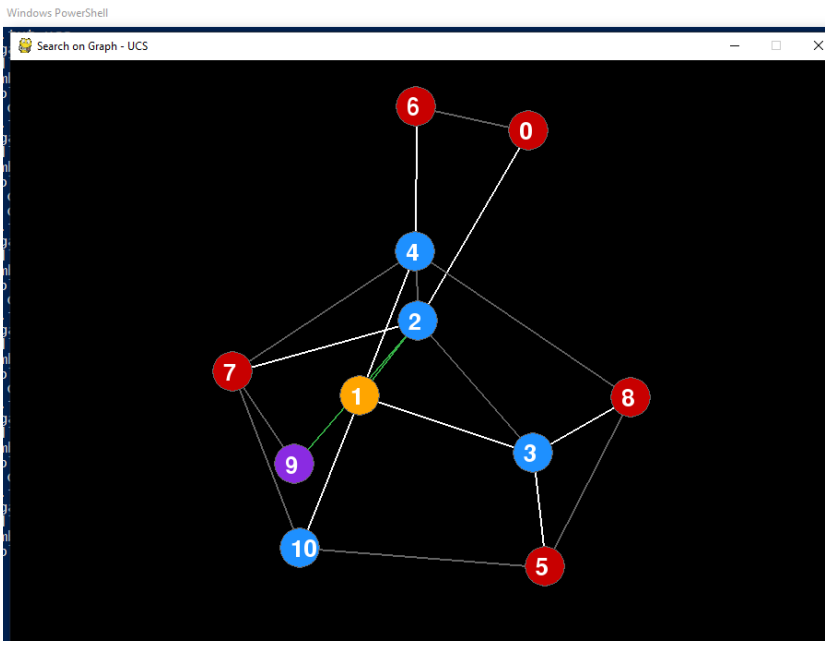


Testcase 2:






Testcase 3:



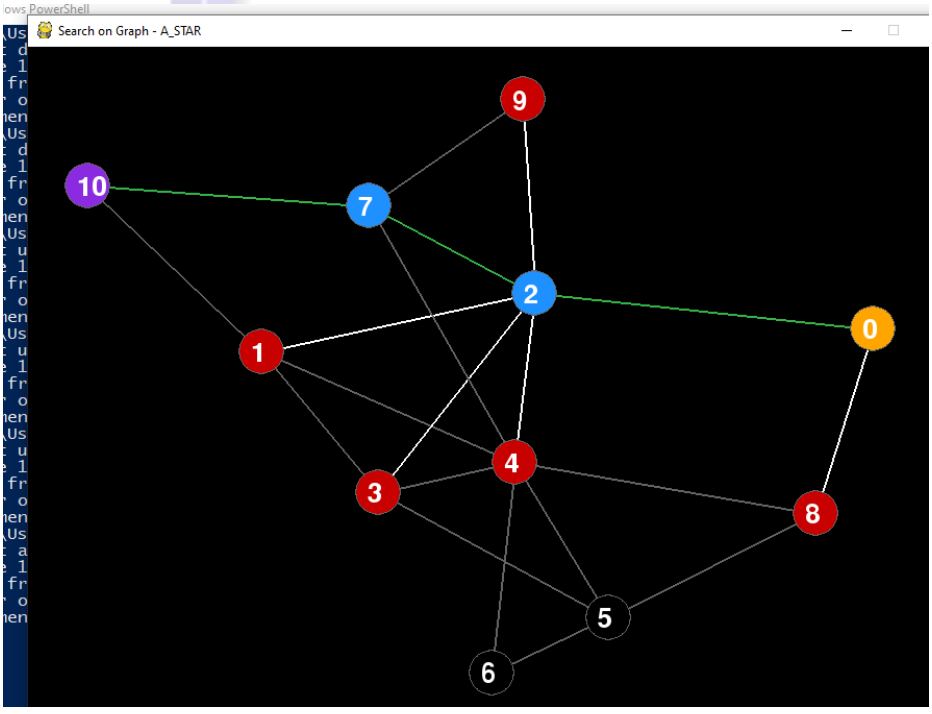
* Nhận xét:

- Tương tự như 2 thuật toán phía trên, UCS vẫn là thuật toán tìm kiếm mù. Ưu điểm của thuật toán này so với 2 thuật toán trên là tìm kiếm đường đi có chi phí thấp nhất.

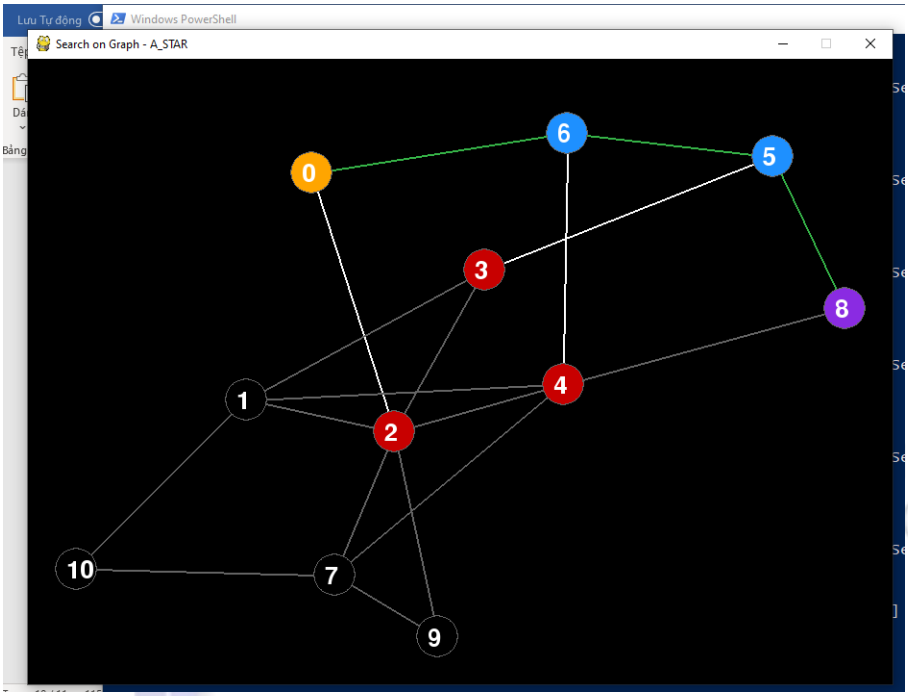


## 4) Astar:

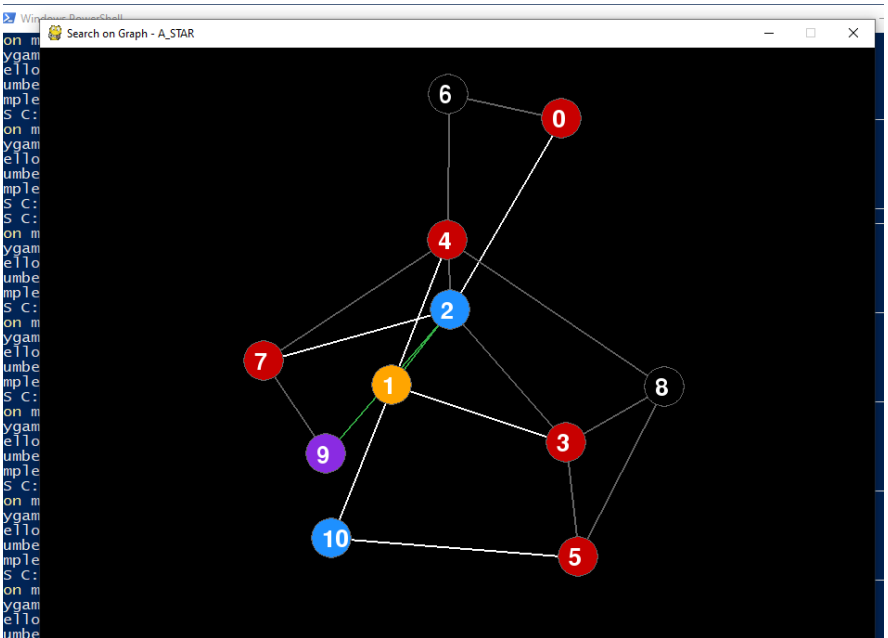
Testcase 1:



Testcase 2:




Testcase 3:



* Nhận xét:

- Ưu điểm:

+ Một thuật giải linh động, tổng quát, trong đó hàm chứa cả tìm kiếm chiều sâu, tìm kiếm chiều rộng và những nguyên lý Heuristic khác. Nhanh chóng tìm đến lời giải với sự định hướng của hàm Heuristic. Chính vì thế mà người ta thường nói A* chính là thuật giải tiêu biểu cho Heuristic.

- Nhược điểm:

+ A* rất linh động nhưng vẫn gặp một khuyết điểm cơ bản - giống như chiến lược tìm kiếm chiều rộng - đó là tốn khá nhiều bộ nhớ để lưu lại những trạng thái đã đi qua.



* Heuristic được dùng là khoảng cách từ điểm bắt đầu đến điểm kết thúc là ngắn nhất (đường chim bay).

* Dùng heuristic này vì hàm heuristic này chấp nhận được (đường chim bay trong thực tế luôn luôn là đường ngắn nhất) nên chi phí của nó luôn nhỏ hơn chi phí đến đích.

# THE END