

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN



---

# BÁO CÁO PHÂN HOẠCH ĐỒ THỊ

---

*Người thực hiện:*

22C15009 - Nguyễn Ngọc Minh Khánh

22C15026 - Nguyễn Khắc Duy

# Báo cáo phân hoạch đồ thị

## Thành viên

1. 22C15009 - Nguyễn Ngọc Minh Khánh
2. 22C15026- Nguyễn Khắc Duy

## Phân công công việc

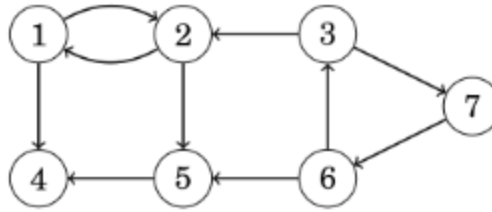
Công việc	Khánh	Duy
Tìm hiểu lý thuyết đồ thị liên thông	x	
Đọc hiểu thuật toán <b>Kosaraju</b>		x
Lập trình phân hoạch đồ thị	x	x
Viết báo phần lý thuyết	x	
Viết báo cáo phần mã nguồn		x

## Liên thông mạnh

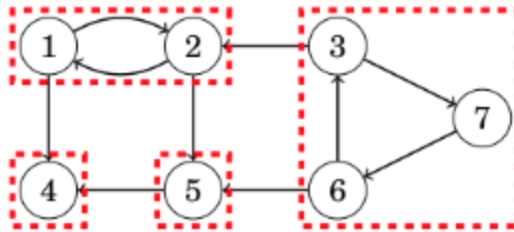
Một đồ thị được gọi là liên thông (connected) nếu có đường đi giữa mọi cặp đỉnh phân biệt của đồ thị. Ngược lại, đồ thị này được gọi là không liên thông.

*Liên thông mạnh* (strongly connected): Đồ thị có hướng gọi là liên thông mạnh nếu có đường đi từ  $a$  tới  $b$  và từ  $b$  tới  $a$  với mọi cặp đỉnh  $a$  và  $b$  của đồ thị. Một **thành phần liên thông mạnh** của một đồ thị có hướng là một đồ thị con tối đại liên thông mạnh. Nếu mỗi thành phần liên thông mạnh được co lại thành một đỉnh, thì đồ thị sẽ trở thành một đồ thị có hướng không chu trình.

Ví dụ cho đồ thị sau:



Có các thành phần liên thông mạnh như hình dưới, do mỗi cặp đỉnh trong đồ con bên dưới đều có thể kết nối tới nhau,  $A = \{1, 2\}$ ,  $B = \{3, 6, 7\}$ ,  $C = \{4\}$  và  $D = \{5\}$ .



## Mã giả

Gọi  $G = (V, E)$  là đồ thị cần phân hoạch với  $V$  là tập các đỉnh và  $E$  là số đỉnh của đồ thị.

$$E = \{(u, v) : u, v \in V\}$$

**Bước 1:** Tạo ngăn xếp  $S = \emptyset$  để lưu thứ tự duyệt DFS trong đồ thị  $G$ . Sau đó chạy DFS.

**Bước 2:** Tạo ma trận chuyển vị  $G^T$

$$G^T = (V, E^T)$$

Trong đó,

$$E^T = \{(v, u) : u, v \in V\}$$

**Bước 3:** Lần lượt lấy phần tử từ đỉnh ngăn xếp  $S$ . Nếu một đỉnh không thuộc về một thành phần, thuật toán tạo ra một thành phần mới và bắt đầu chạy DFS để thêm tất cả

các đỉnh mới được tìm thấy trong suốt quá trình tìm kiếm vào thành phần mới.

## Cách để thêm thuật toán vào

Tận dụng mã nguồn đã cài đặt ở **lab2**:

[https://github.com/KhanhNguyen4999/AlMath\\_Lab3](https://github.com/KhanhNguyen4999/AlMath_Lab3) sau đó cài đặt và sửa lại một số hàm cần thiết.

## Thêm **stack** vào thuộc tính của lớp **Graph**

```
class Graph:
    def __init__(self):
        self.vertList = {}
        self.numVertices = 0
        self.stack = []
```

## Viết thêm hàm để tạo các thành phần liên thông trên đồ thị

```
def get_SCC(self):
    # Step 1: Create a stack
    self.dfs()

    # Step 2: Create transpose graph
    g_tran = self.transpose_graph()

    # Step 3: DFS and print SCC
    new_stack = [g_tran.vertList[v.getId()] for v in self.stack]
    res = []
    while new_stack:
        s = new_stack.pop(-1)
        if s.getColor() == "white":
            scc = []
            g_tran.dfsvisit(s, scc)
            res.append(scc)
    return res
```

## Sửa lại hàm `dfsvisit`

```
def dfsvisit(self, startVertex, scc = None):
    if scc is not None:
        scc.append(startVertex.getId())

    startVertex.setColor('gray')
    self.time += 1
    startVertex.setDiscovery(self.time)

    for nextVertex in startVertex.getConnections():
        if nextVertex.getColor() == 'white':
            nextVertex.setPred(startVertex)
            self.dfsvisit(nextVertex, scc = scc)

    self.stack.append(startVertex)
    startVertex.setColor('black')
    self.time += 1
    startVertex.setFinish(self.time)
```

## Thêm hàm `transpose_graph`

```
def transpose_graph(self):
    g = Graph()
    for v in self.getVertices():
        g.addVertex(key=v)

    for v in self.vertList.values():
        for w in v.getConnections():
            src = v.getId()
            tgt = w.getId()
            g.addEdge(tgt, src)

    g.show_graph()

    return g
```

C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python39\_64\python.exe

BEFORE TRANSPORE:

0 , 2 )

0 , 3 )

1 , 0 )

2 , 1 )

3 , 4 )

AFTER TRANSPORE:

2 , 0 )

0 , 1 )

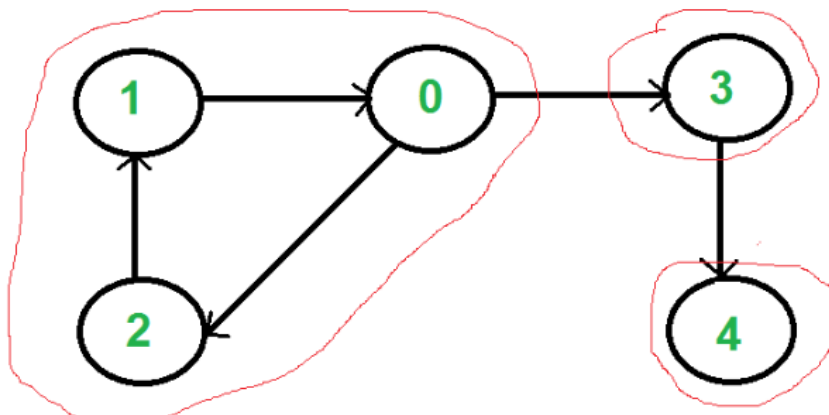
3 , 0 )

1 , 2 )

4 , 3 )

Press any key to continue . . . .

## Ví dụ minh họa



```
g = Graph()

# Create vertex
for i in range(5):
    g.addVertex(i)

# Create edge
g.addEdge(1, 0)
g.addEdge(0, 2)
g.addEdge(2, 1)
```

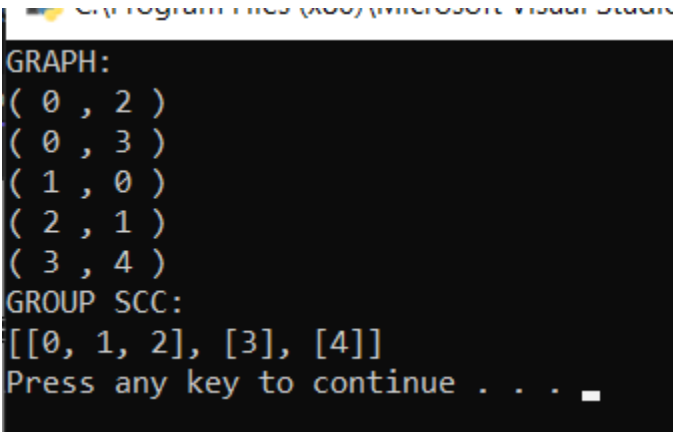
```

g.addEdge(0, 3)
g.addEdge(3, 4)

print("GRAPH: ")
for v in g:
    for w in v.getConnections():
        print(f"({v.getId()} , {w.getId()} )")

# Get group scc
print("GROUP SCC:")
group_scc = g.get_SCC()
print(group_scc)

```



```

GRAPH:
( 0 , 2 )
( 0 , 3 )
( 1 , 0 )
( 2 , 1 )
( 3 , 4 )
GROUP SCC:
[[0, 1, 2], [3], [4]]
Press any key to continue . . .

```

## Tham khảo

1. [Strongly Connected Components - GeeksforGeeks](#)
2. [Strongly Connected Components - Lecture by Rashid Bin Muhammad, PhD. \(kent.edu\)](#)
3. [Vallicon | Khai dân trí - Chấn dân khí - Hậu dân sinh](#)