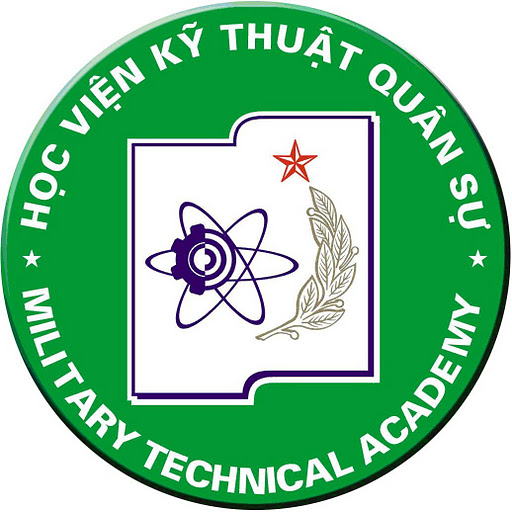
****

**BÁO CÁO**

**PHÂN TÍCH THIẾT KẾ GIẢI THUẬT**

**Đề tài 13:**

***Bài toán tìm thành phần liên thông mạnh trên đồ thị và thuật toán Kosaraju***

* **Giáo viên giảng dạy:** Hà Đại Dương
* **Họ Tên:** Phạm Huy Tùng
* **MSV:** 15150153
* **Lớp:** MMT14

MỤC LỤC

[MỤC LỤC 2](#_Toc501724918)

[PHẦN 1: Nêu Bài Toán 3](#_Toc501724919)

[*Ví Dụ* 4](#_Toc501724920)

[PHẦN 2: Mô Tả Chi Tiết Thuật Toán 5](#_Toc501724921)

[*1) Bước 1* 5](#_Toc501724922)

[*2) Bước 2* 6](#_Toc501724924)

[*Ví Dụ:* 8](#_Toc501724926)

[PHẦN 3: Đánh Giá Độ Phức Tạp Của Thuật Toán 10](#_Toc501724927)

[PHẦN 4: Chi Tiết Thuật Toán 11](#_Toc501724928)

[1) Xét Bộ 1 12](#_Toc501724929)

[2) Xét bộ 2 15](#_Toc501724930)

[Tài liệu tham khảo 17](#_Toc501724931)

# PHẦN 1: Nêu Bài Toán

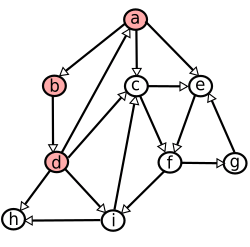
- Một đồ thị có hướng là liên thông mạnh nếu luôn tồn tại đường đi giữa hai đỉnh bất kỳ của đồ thị.

- 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 có chu trình.

- Thành phần liên thông mạnh của một đồ thị có hướng không thay đổi nếu ta đảo chiều tất cả các cung của đồ thị.

- Một đồ thị có hướng được gọi là liên thông mạnh (strongly connected) nếu tồn tại một đường đi có hướng từ **u** tới **v** với mọi cặp đỉnh **u**, **v** của đồ thị. Một đồ thị con **C** của đồ thị có hướng **G (V, )** được gọi là một thành phần liên thông mạnh (strongly connected component) nếu nó liên thông mạn và tối đại (maximal) không tồn tại một đồ thị con liên thông mạnh **D** của **G** mà **C** là đồ thị con của **D**.

## - *Ví Dụ:*

-Ba đỉnh {**a, b, d**} tạo nên một thành phần liên thông mạnh của đồ thị. Ba đỉnh {**e, f, g**} không tạo nên thành phần liên thông mạnh vì tồn tại một đồ thị con liên thông mạnh lớn hơn chứa 3 đỉnh này.

# PHẦN 2: Mô Tả Chi Tiết Thuật Toán

**Thuật toán Kosaraju gồm hai bước chính**:

## *1) Bước 1*:

## - Duyệt đồ thị G (V,) sử dụng duyệt theo chiều sâu DFS.

- Trong bước này, thuật toán còn duy trì một ngăn xếp **S**. - Bắt đầu duyệt **DFS** ở đỉnh **v** của đồ thị **G** (**v** chưa trong **S**).

- Khi mà thủ tục đệ quy trên mỗi đỉnh **v** xong thì mỗi đỉnh **v** sau khi đã duyệt xong thì sẽ được đẩy vào **S**, cứ như vậy cho đến khi duyệt hết tất cả các đỉnh. Một đỉnh được gọi là duyệt xong khi mà mọi hàng xóm được thăm sau nó đã được đánh dấu là duyệt (**visited**). Nói cách khác, một đỉnh được duyệt xong khi mà mọi đỉnh nằm trong cây con **DFS** của nó đã được duyệt.

- **DFS** là một dạng tìm kiếm thông tin không đầy đủ mà quá trình tìm kiếm được phát triển tới đỉnh con đầu tiên của nút đang tìm kiếm cho tới khi gặp được đỉnh cần tìm hoặc tới một nút không có con. Khi đó **giải thuật quay lui** về đỉnh vừa mới tìm kiếm ở bước trước. Trong dạng không đệ quy, tất cả các đỉnh chờ được phát triển được bổ sung vào một ngăn xếp **LIFO**.

## *2) Bước 2*:

## - Bước này ta in ra các thành liên thông mạnh qua thủ tục *PrintConnectedComponent* (H(V,),S).

- Chú ý ở đây ta sẽ thao tác trên đồ thị ngược của **G**. Ta sẽ bắt đầu từ đỉnh **v** nằm trên đầu của ngăn xếp **S** và tiến hành thủ tục **DFS** lần thứ **2** coi **v** là gốc.

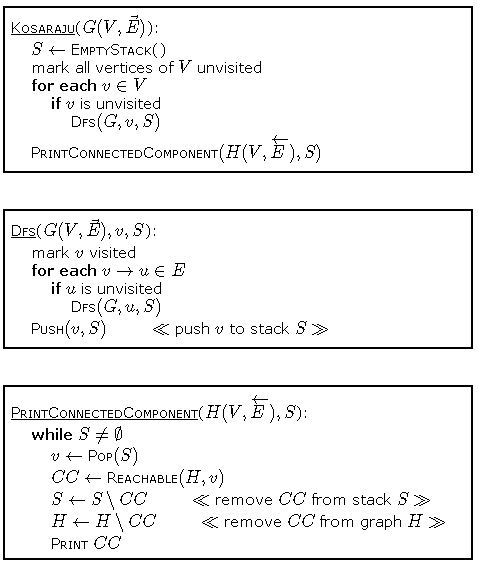
- Tập hợp gồm tất cả các đỉnh đã được thăm từ **v** sẽ cho ra thành phần liên thông mạnh bao gồm cả **v**, loại bỏ tập này ra khỏi **Stack** **S** và đồ thị **H** và lặp lại đến khi **Stack** rỗng. Ta tìm các đỉnh nằm trong tầm với (reachable) của **v** trong **H**.

- Một đỉnh **u** được gọi là nằm trong tầm với của **v** nếu như tồn tại một đường đi có hướng từ **v** tới **u** trong **H**. Tập các đỉnh này (bao gồm cả **v**) sẽ là một thành phần liên thông mạnh của **G** chứa **v**. Sau khi đã tìm được tập đỉnh này rồi , ta xóa khỏi ngăn xếp **S** và đồ thị **H**. Chúng ta cần phải cẩn thận trong việc thực thi phép xóa một tập đỉnh ra khỏi **S** và **H** để thuật toán cuối cùng vẫn là tuyến tính.

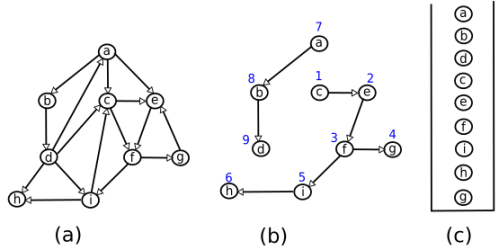
- Chúng ta cần đảo ngược đồ thị vì nếu chúng ta gọi **DFS** lần đầu tiên trên đỉnh **v**, ta sẽ tìm ta những đỉnh kề với **v** (**v** -> **n**). Để đảm bảo rằng duyệt qua tất cả các đỉnh kề với **v** (**n** - > **v**) và mặt khác xác định được sự liên thông mạnh ta cần phải đi qua tất cả các cạnh theo hướng ngược lại. Và cách dễ dàng tốn ít thời gian và tài nguyên nhất là đảo ngược hướng của đồ thị.

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

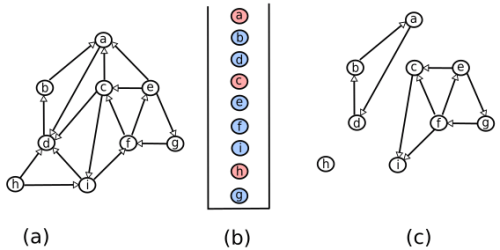
- Mỗi cạnh được duyệt qua **1** lần với đồ thị **có hướng**. Nếu là đồ thị **vô hướng** thì mỗi cạnh được duyệt qua **2** lần.



## - *Ví Dụ:*

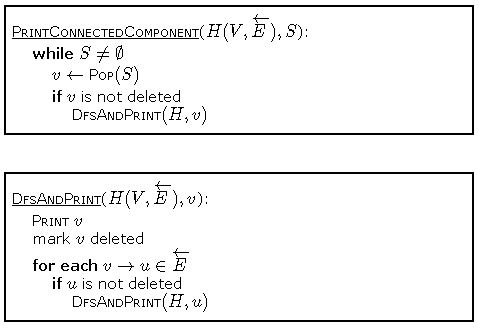
****

- ***Bước 1*** của thuật toán áp dụng cho đồ thị đầu vào ở hình (**a**). Hình (**b**) là rừng **DFS**, thu được bằng cách áp dụng thuật toán **DFS** bắt đầu từ đỉnh **c**; con số bên cạnh mỗi đỉnh là thứ tự đỉnh được thăm bởi **DFS**. Hình (**c**) là thứ tự các đỉnh được đẩy vào ngăn xếp khi thăm **DFS**.



- ***Bước 2*** của thuật toán áp dụng cho đồ thị ngược (**a**) của đồ thị **G**. Các đỉnh {**a**, **c**, **h**} lần lượt là các đỉnh được lấy ra khỏi ngắn xếp trong vòng lặp while của thử tục ***PrintConnectedComponent*** qua phương **Pop**. Các đỉnh còn lại sẽ bị xóa khỏi ngăn xếp sau khi thành phần liên thông mạnh của nó đã được xác định. Hình (**c**) là các thành phần liên thông mạnh tương ứng.

Để tìm các đỉnh trong tầm với của **v**, ta chỉ cần thực hiện duyệt đồ thị từ **H** từ đỉnh **v** (sử dụng **DFS**). Để xóa một tập ra khỏi ngăn xếp **S**, ta sẽ không duyệt qua ngăn xếp, vì như vậy sẽ mất thời gian **O(V)** , do đó, tổng thời gian thuật toán sẽ là **O(KV)** với **K** là số thành phần liên thông mạnh. Ta chỉ cần đánh dấu tập đỉnh đó bị xóa sử dụng một mảng đánh dấu. Do đó, xóa tập **C** chỉ mất thời gian **|C**|. Xóa tập đỉnh **C** ra khỏi đồ thị **H** ta làm tương tự. Như vậy, về cơ bản, thuật toán **Korasaju** chỉ duyệt qua đồ thị **2 lần**. Do đó, thủ tục ***PrintConnecteComponent*** có thể được viết lại như sau:



# PHẦN 3: Đánh Giá Độ Phức Tạp Của Thuật Toán

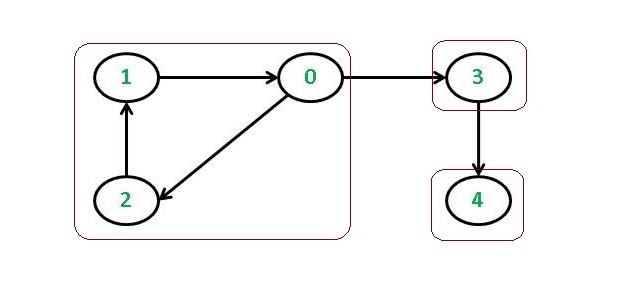
- Giả sử đồ thị được biểu diễn dưới dạng danh sách kề, thuật toán **Kosaraju** duyệt qua toàn bộ đồ thị hai lần, do đó có độ phức tạp tính toán tuyến tính **O(|V| + |E|).** Độ phức tạp này là tối ưu bởi mọi thuật toán đều phải xem xét tất cả các đỉnh và cung của đồ thị. Thuật toán này vô cùng đơn giản nhưng trong thực tế không hiệu quả bằng các thuật toán của Tarjan và Gabow, do chúng chỉ duyệt qua đồ thị đúng một lần.

- Nếu đồ thị được biểu diễn dưới dạng ma trận kề, thuật toán chạy trong thời gian **Ο().**

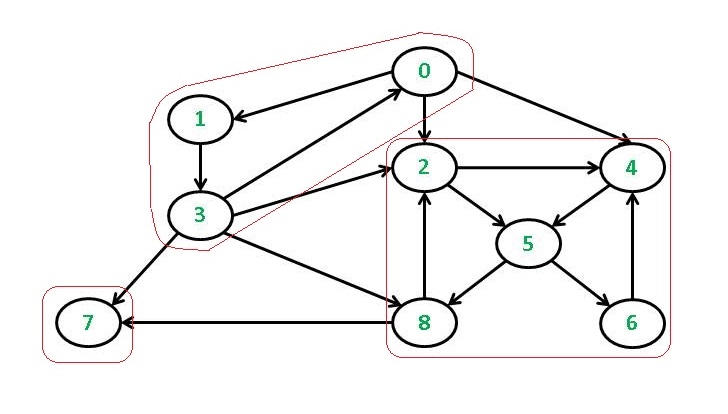
# PHẦN 4: Chi Tiết Thuật Toán

***Ta có hai bộ dữ liệu:***

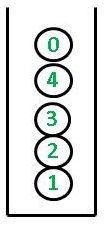
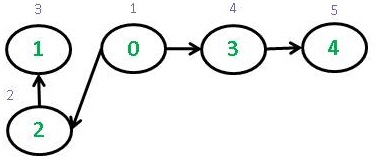
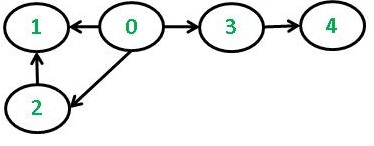
**1)** **Bộ 1: N = 5**



**2)** **Bộ 2: N = 9**



## 1) Xét Bộ 1:



- ***Bước 1***:

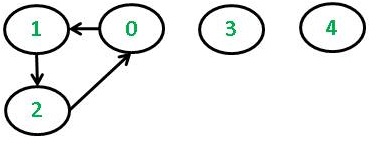
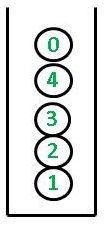
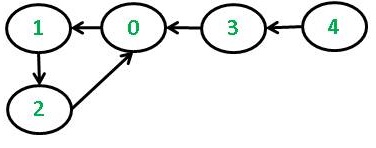
- Chọn đỉnh **v** bắt đầu từ đỉnh **0**. Ta thực hiện duyệt **DFS** lần thứ **1** theo thứ tự duyệt các đỉnh như hình (**b**) và lần lượt push vào **Stack** **S** theo quy tắc: Ta duyệt đỉnh **0** và thăm tất cả các đỉnh **u** kề với **0** cho đến khi không còn đỉnh nào kề với **u** nữa thì ta push **u** vào **Stack** **S**.

- Mỗi đỉnh được gọi là duyệt xong khi mà mọi hàng xóm được thăm sau nó đã được đánh dấu là đã duyệt (**visited**) hoặc nó không còn đỉnh kề nào nữa. Nói cách khác, một đỉnh được duyệt xong khi mà mọi đỉnh nằm trong cây con **DFS** của nó đã được duyệt.

- Ví dụ: Xét đỉnh **0**, đỉnh kề với **0** là **2**, đỉnh kề với **2** là **1**. Ta thấy **1** không còn đỉnh kề nữa. Như vậy **6** đã được duyệt (**visited**), push **1** vào **Stack** **S**. Quay lại xét đỉnh **2** cứ thế theo quy tắc cho đến hết.

- Ta được một **Stack** **S** như hình (**c**) là thứ tự các đỉnh được đẩy vào ngăn xếp khi thăm **DFS**.

***- Bước 2*:**



- Đây sẽ là bước in ra các thành phần liên thông mạnh qua thủ tục ***PrintConnectedComponent().***

- Ta sẽ thao tác trên đồ thị ngược (**a**) của **G** ta sẽ gọi là đồ thị **H**.

- Ta sẽ chọn đỉnh bắt đầu là **0** ở đỉnh của ngăn xếp. Ta sẽ thực hiện duyệt **DFS** lần thứ **2** để tìm các đỉnh trong tầm với của **0** trong **H**. Một đỉnh **u** được gọi là trong tầm với của **0** nếu như tồn tại một đường đi có hướng từ **0** tới **u** trong **H**.

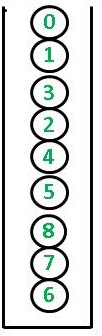
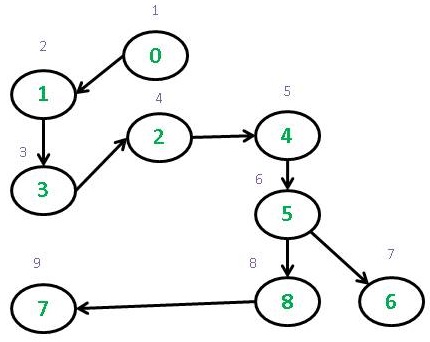
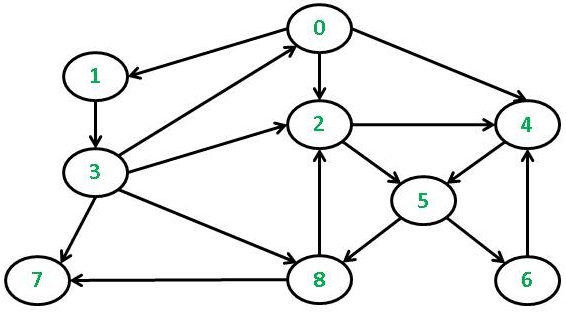
- Tập các đỉnh này (bao gồm cả đỉnh **0**) sẽ là một thành phần liên thông mạnh của **G** chứa đỉnh **0**. Sau khi đã tìm được tập đỉnh này rồi ta sẽ xóa nó khỏi **Stack S** và đồ thị **H**.

- Để xóa tập này ta sẽ không duyệt qua **Stack S** vì sẽ mất tổng thời gian **O(|K| . |V|).** **V** là số đỉnh trong **Stack**, **K** là số thành phần liên thông mạnh. Ta chỉ cần đánh dấu tập đỉnh đó bị xóa bằng một mảng đánh dấu **Dm** chỉ mất thời gian |**Dm**|. Xóa tập đỉnh khỏi đồ thị H cũng tương tự.

- Cứ thế lần lượt các đỉnh {**0**, **4**, **3**} lần lượt là các đỉnh được lấy ra từ ngăn xếp trong vòng lặp **while** của thủ tục ***PrintConnectedComponent()*** qua phương thức **pop**. Các đỉnh còn lại sẽ bị xóa khỏi ngăn xếp sau khi thành phần liên thông mạnh của nó đã được xác định. Hình (**c**) là các thành phần liên thông mạnh tương ứng.

- ***Ví dụ***: Ta duyệt đỉnh **Stack** là đỉnh **0**, xét trên đồ thị **H**, đỉnh trong tầm với của **0** là **1**, trong tầm với của **1** là **2**, trong tầm với của **2** là **0**. Như vậy ta xác định được thành phần liên thông mạnh đầu tiên là: **0**, **1**, **2**. Đánh dấu tập {**0**,**1**,**2**} vào tập đánh dấu **Dm** là đã xóa. Tập **Dm** ánh xạ tham trị đến **H** và **Stack**. Cứ thế cho đến khi **Stack** rỗng ta sẽ được kết quả như hình (**c**).

## 2) Xét bộ 2:

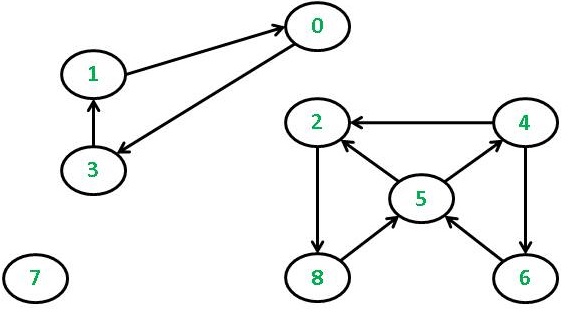
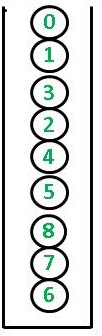
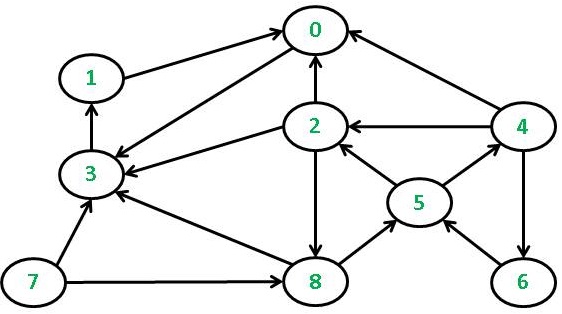


- ***Bước 1***: Chọn đỉnh **v** bắt đầu từ đỉnh **0**. Ta thực hiện duyệt **DFS** lần thứ **1** theo thứ tự duyệt các đỉnh như hình (**b**) và lần lượt push vào **Stack** **S** theo quy tắc: Ta duyệt đỉnh **0** và thăm tất cả các đỉnh **u** kề với **0** cho đến khi không còn đỉnh nào kề với **u** nữa thì ta push **u** vào **Stack** **S**.

- ***Ví dụ***: Xét đỉnh **0**, đỉnh kề với **0** là **1**, đỉnh kề với **1** là **3**, đỉnh kề **3** là **2**, kề với **2** là **4**, kề với **4** là **5**, kề với **5** là **6**. Ta thấy **6** kề với **4** nhưng **4** đã được đánh dấu (**visited**) như vậy **6** đã được duyệt, push **6** vào **Stack** **S**. Quay lại xét đỉnh **5** cứ thế theo quy tắc cho đến hết.

- Ta được một **Stack S** như hình (**c**)là thứ tự các đỉnh được đẩy vào ngăn xếp khi thăm **DFS**.

**- *Bước 2***:



- Ta sẽ thao tác trên đồ thị ngược (**a**) của **G** ta sẽ gọi là đồ thị **H**.

- Ta sẽ chọn đỉnh bắt đầu là **0** ở đỉnh của ngăn xếp. Ta sẽ thực hiện duyệt **DFS** lần thứ **2** để tìm các đỉnh trong tầm với của **0** trong **H**. Một đỉnh **u** được gọi là trong tầm với của **0** nếu như tồn tại một đường đi có hướng từ **0** tới **u** trong **H**.

- Cứ thế lần lượt các đỉnh {**0**, **2**, **7**} lần lượt là các đỉnh được lấy ra từ ngăn xếp trong vòng lặp while của thủ tục ***PrintConnectedComponent()*** qua phương thức **pop**. Các đỉnh còn lại sẽ bị xóa khỏi ngăn xếp sau khi thành phần liên thông mạnh của nó đã được xác định. Hình (**c**) là các thành phần liên thông mạnh tương ứng.

- ***Ví dụ***: Ta duyệt đỉnh **Stack** là đỉnh **0**, xét trên đồ thị **H**, đỉnh trong tầm với của **0** là **3**, trong tầm với của **3** là **0**. Như vậy ta xác định được thành phần liên thông mạnh đầu tiên là: **0**, **3**, **1**. Đánh dấu tập {**0**, **3**, **1**} vào tập đánh dấu **Dm** là đã xóa. Tập **Dm** ánh xạ tham trị đến **H** và **Stack**. Cứ thế cho đến khi **Stack** rỗng ta sẽ được kết quả như hình (**c**).

**~HẾT~**

# - Tài liệu tham khảo

*1) Slide bài giảng.*

*2) giaithuatlaptrinh.com*

*3) geeksforgeeks.org*