

TÌM THÀNH PHẦN LIÊN THÔNG MẠNH VÀ BÀI TOÁN 2-SAT

PHÂN TÍCH THIẾT KẾ THUẬT TOÁN

GIẢNG VIÊN: TS ĐỖ PHAN THUẬN

SINH VIÊN: PHẠM MINH TÂM

OUTLINE

Tìm thành phần liên thông mạnh

- Thuật toán Kosaraju
- Thuật toán Tarjan

Bài toán 2-SAT

Bài toán 22E - codeforces

Bài toán 776D - codeforces

THÀNH PHẦN LIÊN THÔNG MẠNH

- Cho đồ thị có hướng $G(V,E)$. Đồ thị được gọi là liên thông mạnh nếu tồn tại một đường đi từ u tới v với mọi cặp đỉnh $u,v \in V$.
- Một đồ thị con C của G được gọi là một thành phần liên thông mạnh nếu nó liên thông mạnh và tối đại (nếu thêm bất kỳ đỉnh $w \in G$ nào khác vào C thì C không liên thông mạnh nữa).
- 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.
- Thuật toán tìm thành phần liên thông mạnh:
 - Kosaraju
 - Tarjan

THUẬT TOÁN KOSARAJU

- Là một thuật toán tìm thành phần liên thông mạnh trong đồ thị có hướng.
- Sử dụng tính chất : trong đồ thị chuyển vị (đồ thị trong đó mọi cung được đảo ngược so với đồ thị ban đầu), các thành phần liên thông mạnh là không đổi so với đồ thị ban đầu.

THUẬT TOÁN KOSARAJU

for đỉnh u chưa thăm : DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Hoàn thành DFS(u), order.push_back(u)

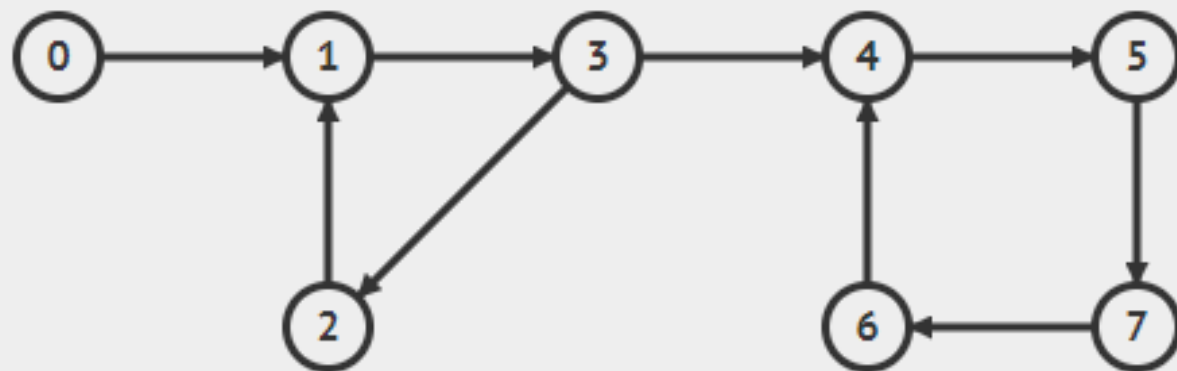
Đảo ngược cạnh

$u = \text{order.pop}()$: if u chưa thăm DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh

THUẬT TOÁN KOSARAJU



for đỉnh u chưa thăm : DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Hoàn thành DFS(u), order.push_back(u)

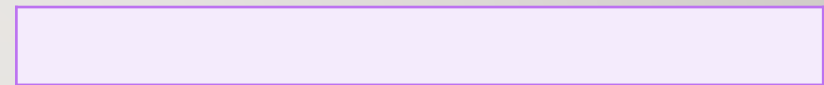
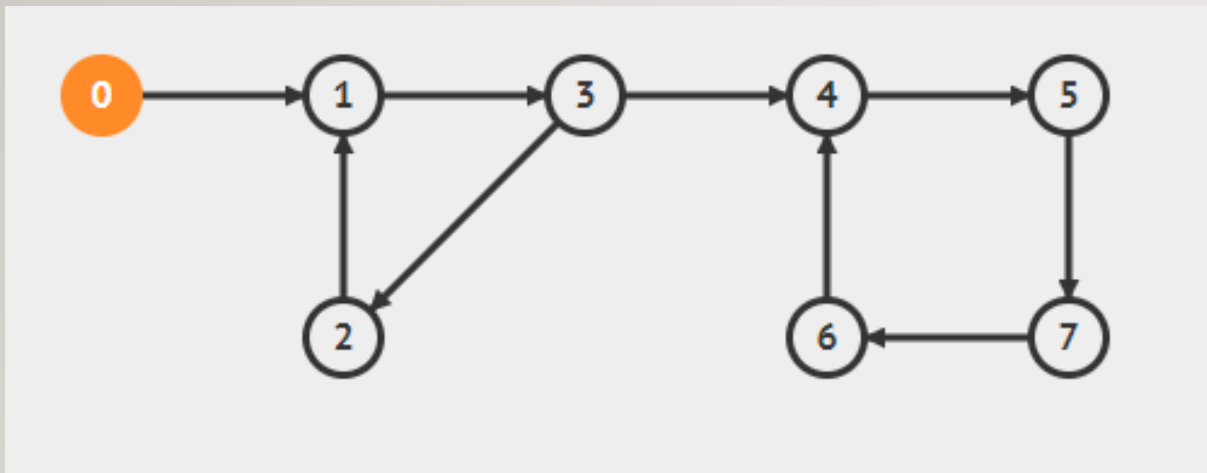
Đảo ngược cạnh

u= order.pop(): if u chưa thăm DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh

THUẬT TOÁN KOSARAJU



for đỉnh u chưa thăm : DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Hoàn thành DFS(u), order.push_back(u)

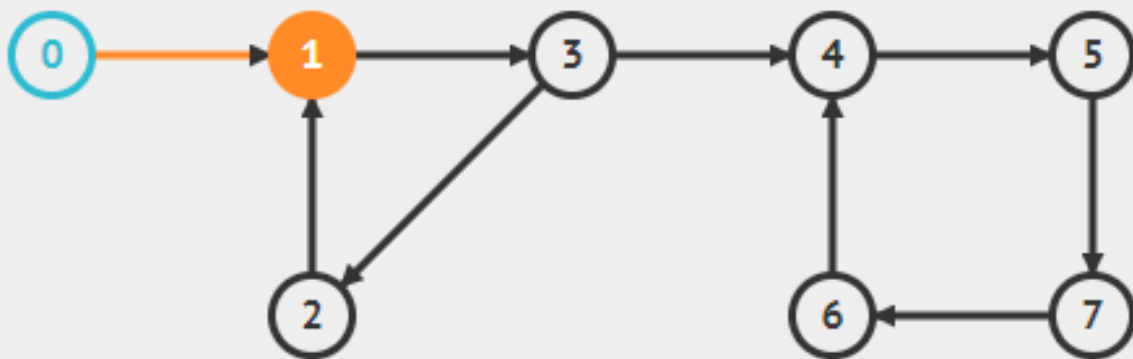
Đảo ngược cạnh

$u = \text{order.pop}()$: if u chưa thăm DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh

THUẬT TOÁN KOSARAJU



for đỉnh u chưa thăm : DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Hoàn thành DFS(u), order.push_back(u)

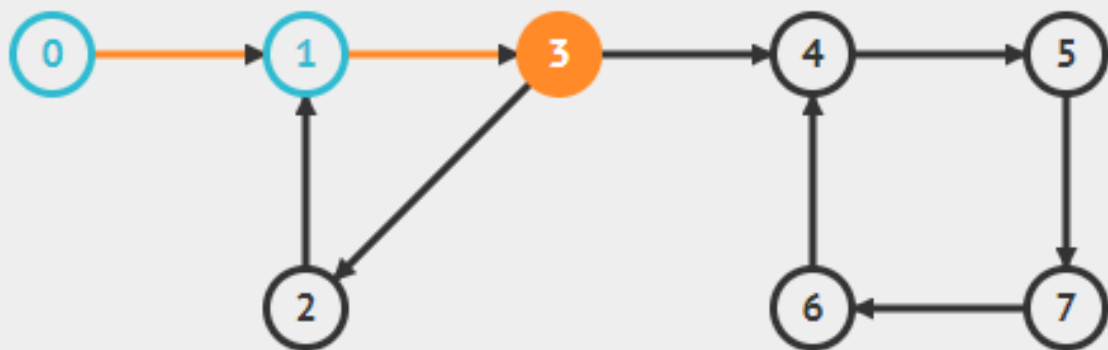
Đảo ngược cạnh

$u = \text{order.pop}()$: if u chưa thăm DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh

THUẬT TOÁN KOSARAJU



for đỉnh u chưa thăm : DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Hoàn thành DFS(u), order.push_back(u)

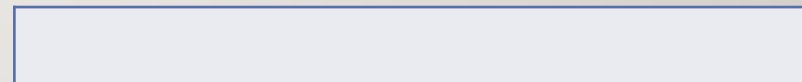
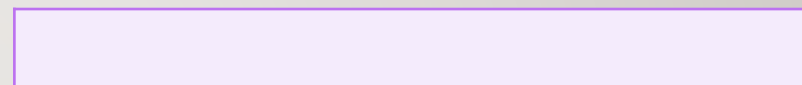
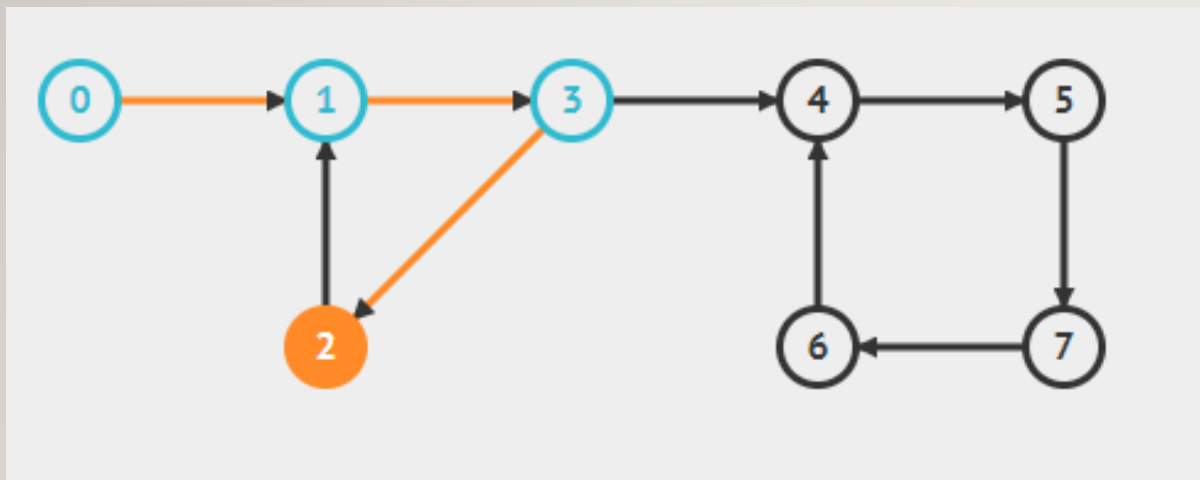
Đảo ngược cạnh

$u = \text{order.pop}()$: if u chưa thăm DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh

THUẬT TOÁN KOSARAJU



for đỉnh u chưa thăm : DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Hoàn thành DFS(u), order.push_back(u)

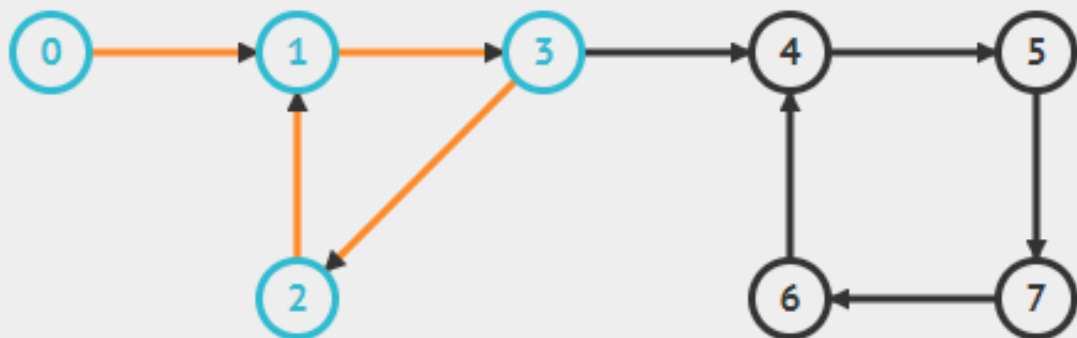
Đảo ngược cạnh

$u = \text{order.pop}()$: if u chưa thăm DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh

THUẬT TOÁN KOSARAJU



for đỉnh u chưa thăm : DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Hoàn thành DFS(u), order.push_back(u)

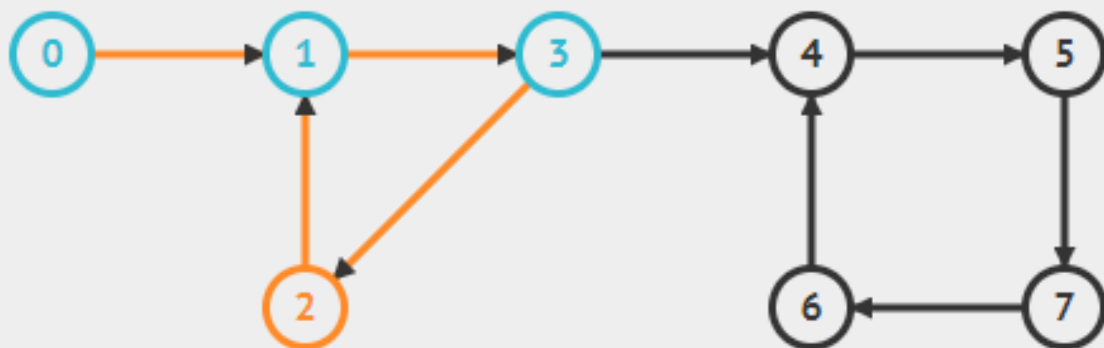
Đảo ngược cạnh

$u = \text{order.pop}()$: if u chưa thăm DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh

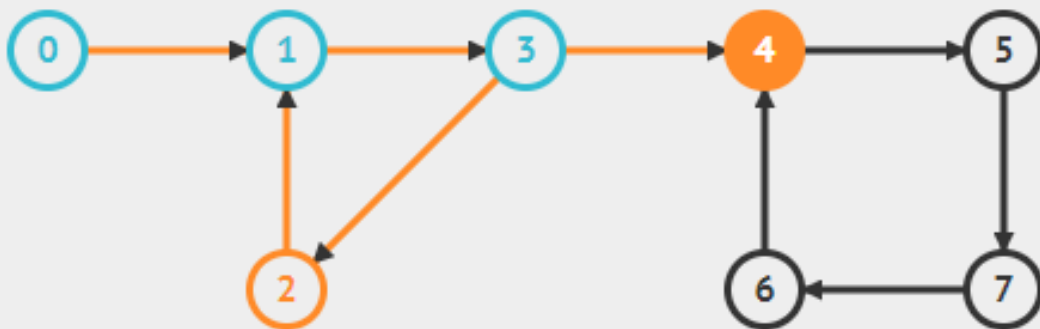
THUẬT TOÁN KOSARAJU



2

```
for đỉnh u chưa thăm : DFS(u)
  for v với (u,v) ∈ E && v chưa thăm: DFS(v)
  Hoàn thành DFS(u), order.push_back(u)
Đảo ngược cạnh
u= order.pop(): if u chưa thăm DFS(u)
  for v với (u,v) ∈ E && v chưa thăm: DFS(v)
Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh
```

THUẬT TOÁN KOSARAJU



2

for đỉnh u chưa thăm : DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Hoàn thành DFS(u), order.push_back(u)

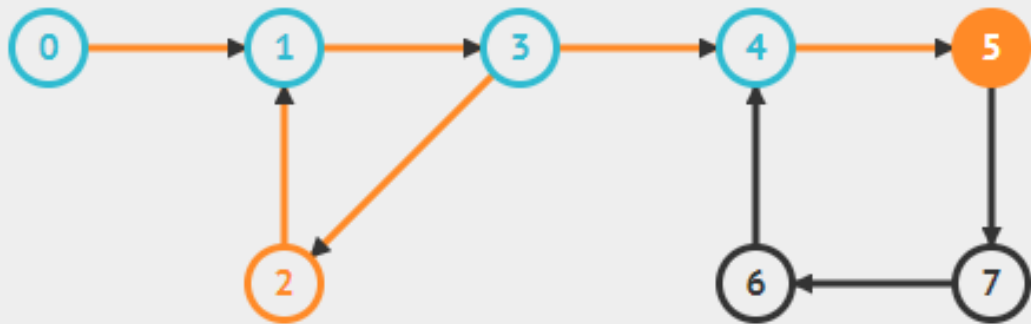
Đảo ngược cạnh

$u = \text{order.pop}()$: if u chưa thăm DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh

THUẬT TOÁN KOSARAJU



2

```
for đỉnh u chưa thăm : DFS(u)
    for v với (u,v) ∈ E && v chưa thăm: DFS(v)
    Hoàn thành DFS(u), order.push_back(u)
Đảo ngược cạnh
u= order.pop(): if u chưa thăm DFS(u)
    for v với (u,v) ∈ E && v chưa thăm: DFS(v)
Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh
```


THUẬT TOÁN KOSARAJU



2

for đỉnh u chưa thăm : DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Hoàn thành DFS(u), $order.push_back(u)$

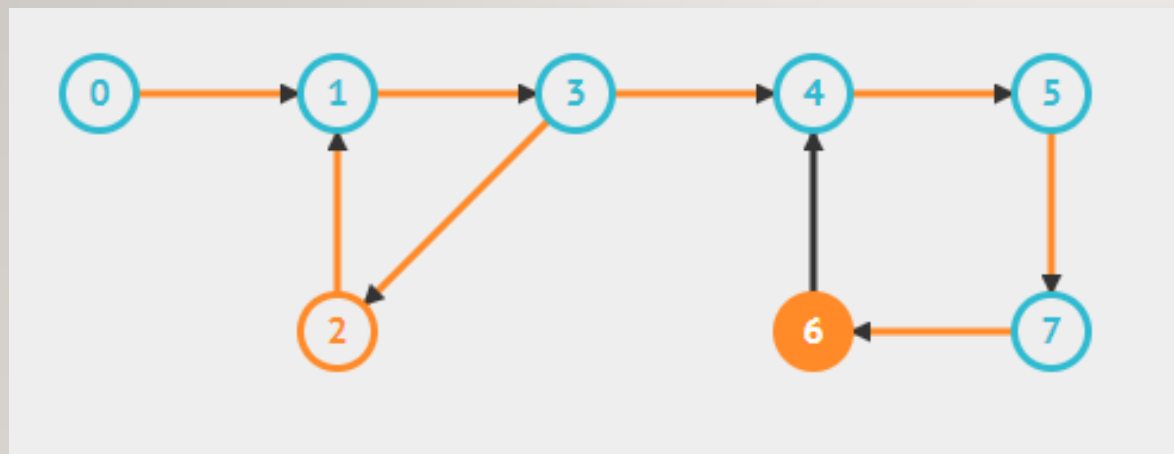
Đảo ngược cạnh

$u = order.pop()$: if u chưa thăm DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh

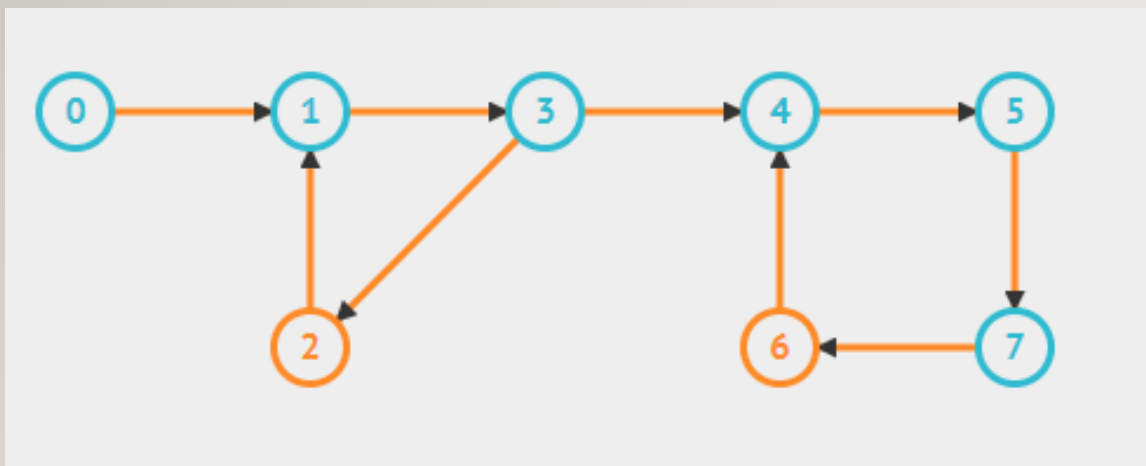
THUẬT TOÁN KOSARAJU



2

```
for đỉnh u chưa thăm : DFS(u)
    for v với  $(u,v) \in E$  && v chưa thăm: DFS(v)
    Hoàn thành DFS(u), order.push_back(u)
Đảo ngược cạnh
u= order.pop(): if u chưa thăm DFS(u)
    for v với  $(u,v) \in E$  && v chưa thăm: DFS(v)
Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh
```

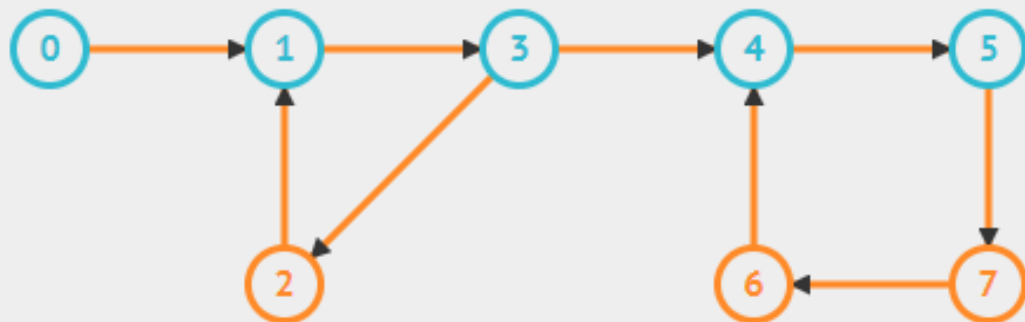
THUẬT TOÁN KOSARAJU



2,6

```
for đỉnh u chưa thăm : DFS(u)
  for v với (u,v) ∈ E && v chưa thăm: DFS(v)
  Hoàn thành DFS(u), order.push_back(u)
Đảo ngược cạnh
u= order.pop(): if u chưa thăm DFS(u)
  for v với (u,v) ∈ E && v chưa thăm: DFS(v)
Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh
```

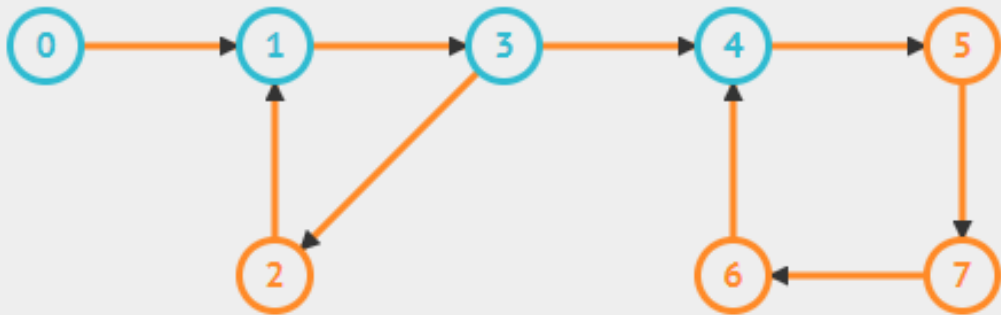
THUẬT TOÁN KOSARAJU



2,6,7

```
for đỉnh u chưa thăm : DFS(u)
    for v với (u,v) ∈ E && v chưa thăm: DFS(v)
    Hoàn thành DFS(u), order.push_back(u)
Đảo ngược cạnh
u= order.pop(): if u chưa thăm DFS(u)
    for v với (u,v) ∈ E && v chưa thăm: DFS(v)
Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh
```

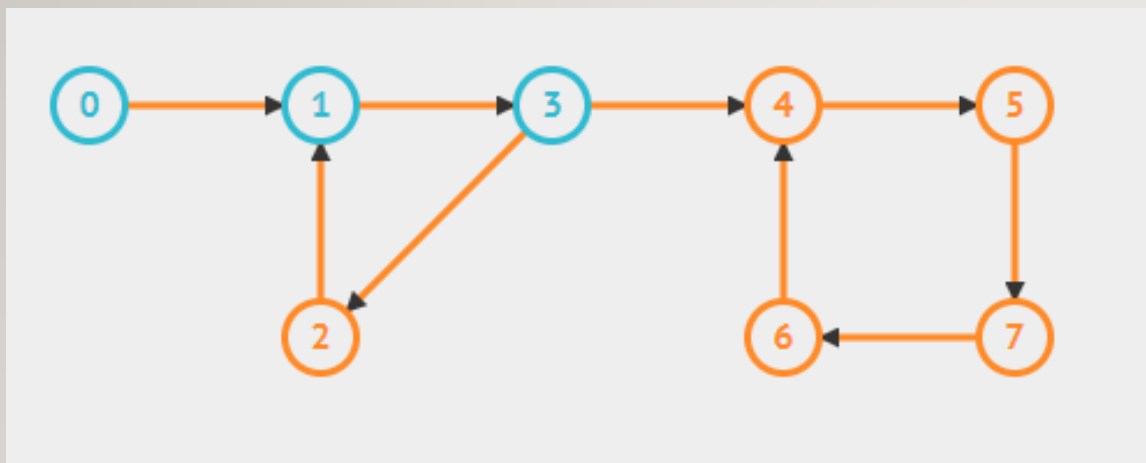
THUẬT TOÁN KOSARAJU



2,6,7,5

```
for đỉnh u chưa thăm : DFS(u)
    for v với (u,v) ∈ E && v chưa thăm: DFS(v)
    Hoàn thành DFS(u), order.push_back(u)
Đảo ngược cạnh
u= order.pop(): if u chưa thăm DFS(u)
    for v với (u,v) ∈ E && v chưa thăm: DFS(v)
Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh
```

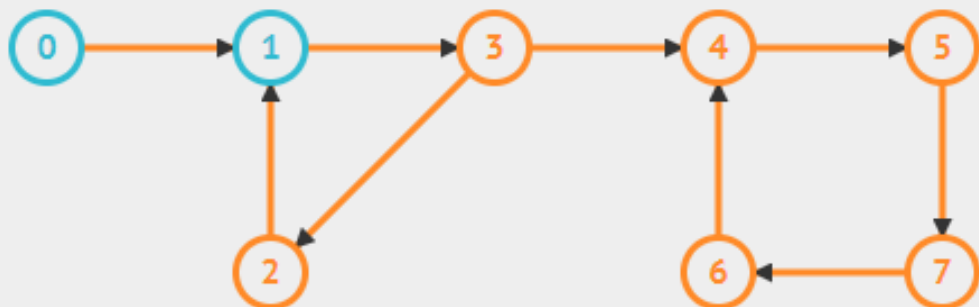

THUẬT TOÁN KOSARAJU



2,6,7,5,4

```
for đỉnh u chưa thăm : DFS(u)
    for v với (u,v) ∈ E && v chưa thăm: DFS(v)
    Hoàn thành DFS(u), order.push_back(u)
Đảo ngược cạnh
u= order.pop(): if u chưa thăm DFS(u)
    for v với (u,v) ∈ E && v chưa thăm: DFS(v)
Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh
```

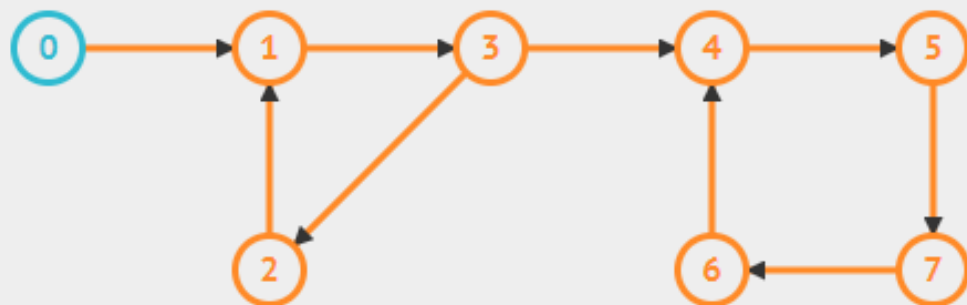

THUẬT TOÁN KOSARAJU



2,6,7,5,4,3

```
for đỉnh u chưa thăm : DFS(u)
    for v với (u,v) ∈ E && v chưa thăm: DFS(v)
    Hoàn thành DFS(u), order.push_back(u)
Đảo ngược cạnh
u= order.pop(): if u chưa thăm DFS(u)
    for v với (u,v) ∈ E && v chưa thăm: DFS(v)
Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh
```

THUẬT TOÁN KOSARAJU



2,6,7,5,4,3,1

```
for đỉnh u chưa thăm : DFS(u)
    for v với (u,v) ∈ E && v chưa thăm: DFS(v)
    Hoàn thành DFS(u), order.push_back(u)
Đảo ngược cạnh
u= order.pop(): if u chưa thăm DFS(u)
    for v với (u,v) ∈ E && v chưa thăm: DFS(v)
Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh
```

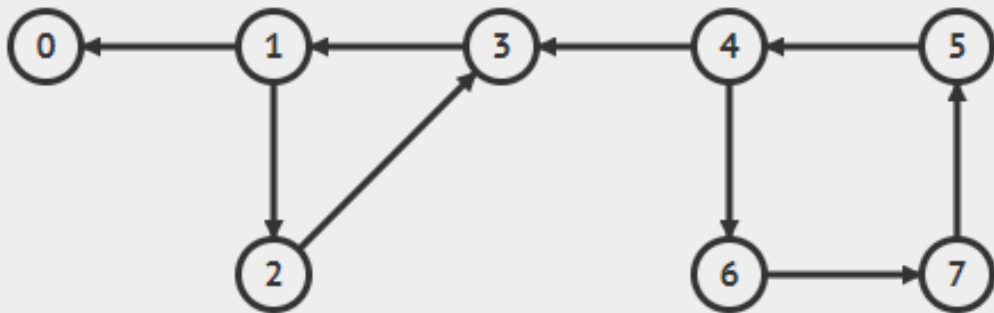
THUẬT TOÁN KOSARAJU



2,6,7,5,4,3,1,0

```
for đỉnh u chưa thăm : DFS(u)
  for v với (u,v) ∈ E && v chưa thăm: DFS(v)
  Hoàn thành DFS(u), order.push_back(u)
Đảo ngược cạnh
u= order.pop(): if u chưa thăm DFS(u)
  for v với (u,v) ∈ E && v chưa thăm: DFS(v)
Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh
```

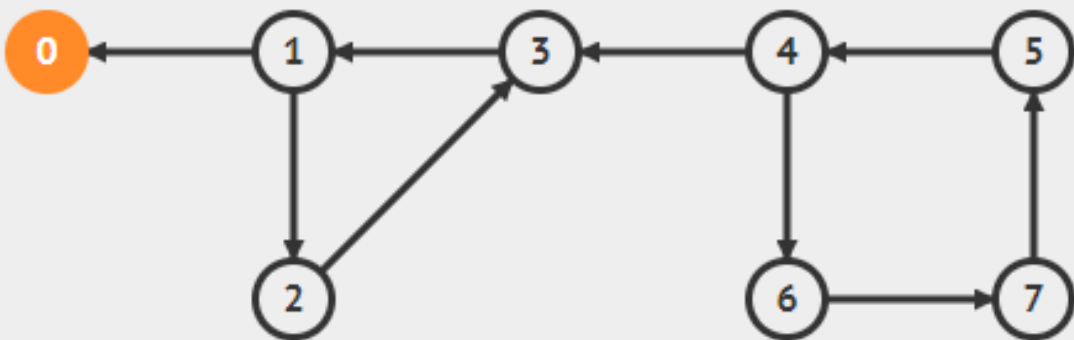
THUẬT TOÁN KOSARAJU



2,6,7,5,4,3,1,0

for đỉnh u chưa thăm : DFS(u)
 for v với $(u,v) \in E$ && v chưa thăm: DFS(v)
 Hoàn thành DFS(u), order.push_back(u)
Đảo ngược cạnh
 $u = \text{order.pop}()$: if u chưa thăm DFS(u)
 for v với $(u,v) \in E$ && v chưa thăm: DFS(v)
Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh

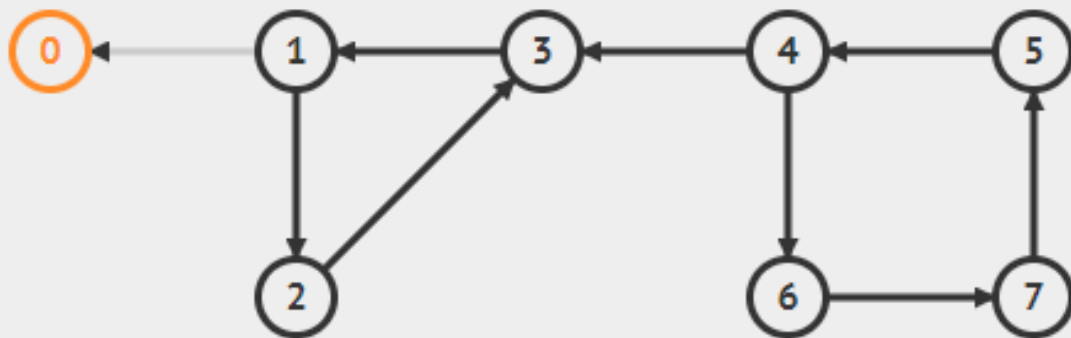
THUẬT TOÁN KOSARAJU



2,6,7,5,4,3,1

```
for đỉnh u chưa thăm : DFS(u)
    for v với (u,v) ∈ E && v chưa thăm: DFS(v)
    Hoàn thành DFS(u), order.push_back(u)
Đảo ngược cạnh
u= order.pop(): if u chưa thăm DFS(u)
    for v với (u,v) ∈ E && v chưa thăm: DFS(v)
Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh
```


THUẬT TOÁN KOSARAJU



2,6,7,5,4,3,1

(0),

for đỉnh u chưa thăm : DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Hoàn thành DFS(u), order.push_back(u)

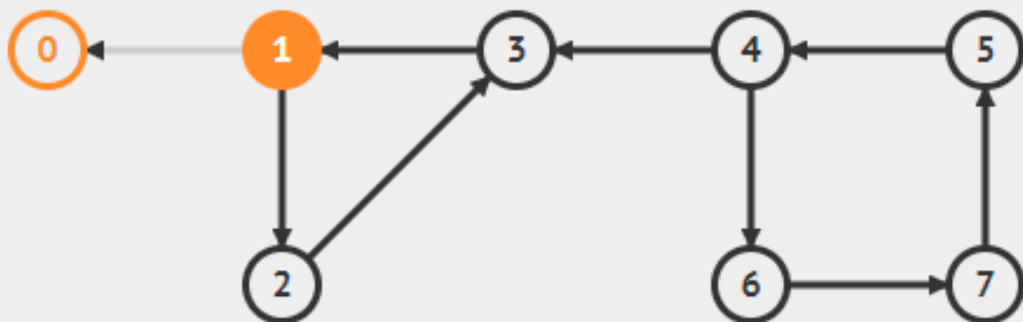
Đảo ngược cạnh

u= order.pop(): if u chưa thăm DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh

THUẬT TOÁN KOSARAJU

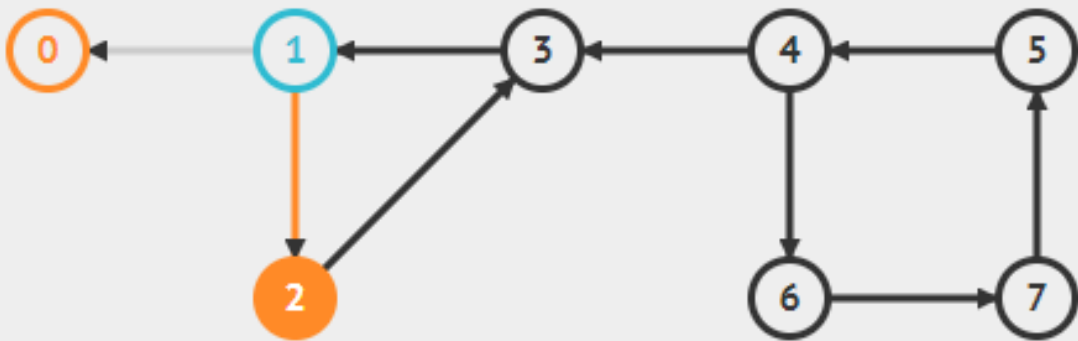


2,6,7,5,4,3

(0),

```
for đỉnh u chưa thăm : DFS(u)
    for v với (u,v) ∈ E && v chưa thăm: DFS(v)
    Hoàn thành DFS(u), order.push_back(u)
Đảo ngược cạnh
u= order.pop(): if u chưa thăm DFS(u)
    for v với (u,v) ∈ E && v chưa thăm: DFS(v)
Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh
```

THUẬT TOÁN KOSARAJU



2,6,7,5,4,3

(0),

for đỉnh u chưa thăm : DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Hoàn thành DFS(u), order.push_back(u)

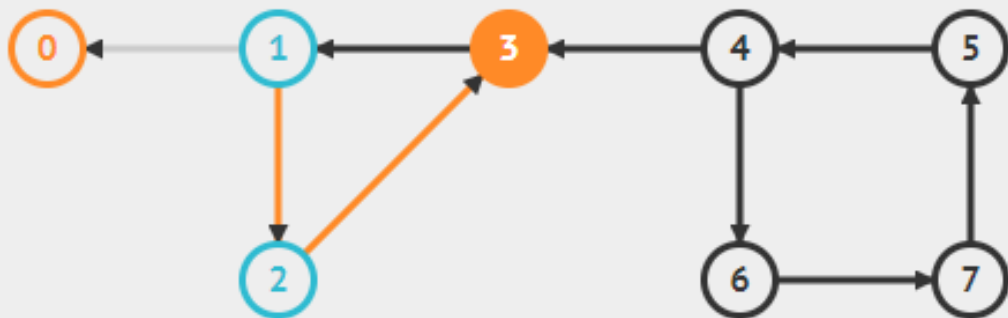
Đảo ngược cạnh

$u = \text{order.pop}()$: if u chưa thăm DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh

THUẬT TOÁN KOSARAJU



2,6,7,5,4,3

(0),

for đỉnh u chưa thăm : DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Hoàn thành DFS(u), order.push_back(u)

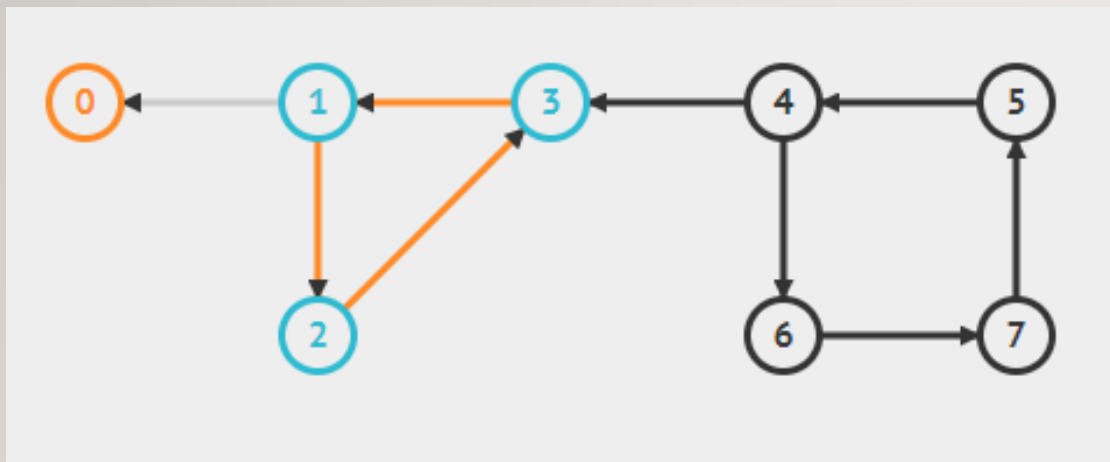
Đảo ngược cạnh

$u = \text{order.pop}()$: if u chưa thăm DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh

THUẬT TOÁN KOSARAJU



2,6,7,5,4,3

(0),

for đỉnh u chưa thăm : DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Hoàn thành DFS(u), order.push_back(u)

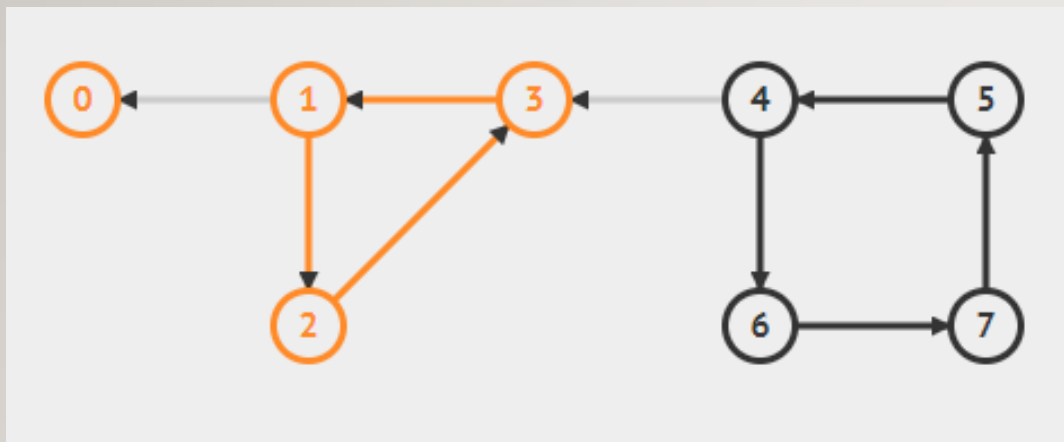
Đảo ngược cạnh

$u = \text{order.pop}()$: if u chưa thăm DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh

THUẬT TOÁN KOSARAJU



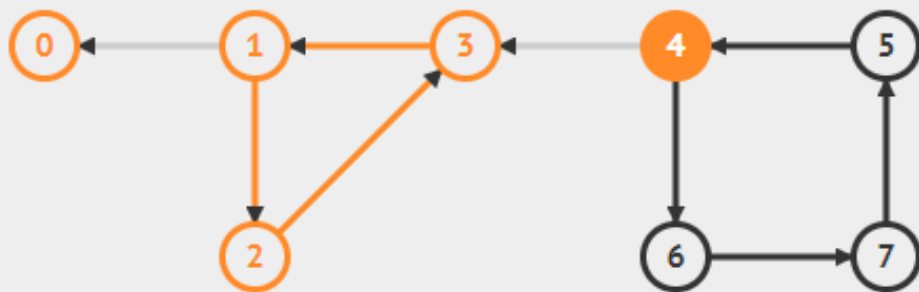
2,6,7,5,4,3

(0), (3,2,1),

```
for đỉnh u chưa thăm : DFS(u)
    for v với (u,v) ∈ E && v chưa thăm: DFS(v)
    Hoàn thành DFS(u), order.push_back(u)
Đảo ngược cạnh
u= order.pop(): if u chưa thăm DFS(u)
    for v với (u,v) ∈ E && v chưa thăm: DFS(v)
```

Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh

THUẬT TOÁN KOSARAJU



2,6,7,5

(0), (3,2,1),

for đỉnh u chưa thăm : DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Hoàn thành DFS(u), order.push_back(u)

Đảo ngược cạnh

$u = \text{order.pop}()$: if u chưa thăm DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh

THUẬT TOÁN KOSARAJU



2,6,7,5

(0), (3,2,1),

```
for đỉnh u chưa thăm : DFS(u)
    for v với (u,v) ∈ E && v chưa thăm: DFS(v)
    Hoàn thành DFS(u), order.push_back(u)
Đảo ngược cạnh
u= order.pop(): if u chưa thăm DFS(u)
    for v với (u,v) ∈ E && v chưa thăm: DFS(v)
Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh
```

THUẬT TOÁN KOSARAJU



2,6,7,5

(0), (3,2,1),

for đỉnh u chưa thăm : DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Hoàn thành DFS(u), order.push_back(u)

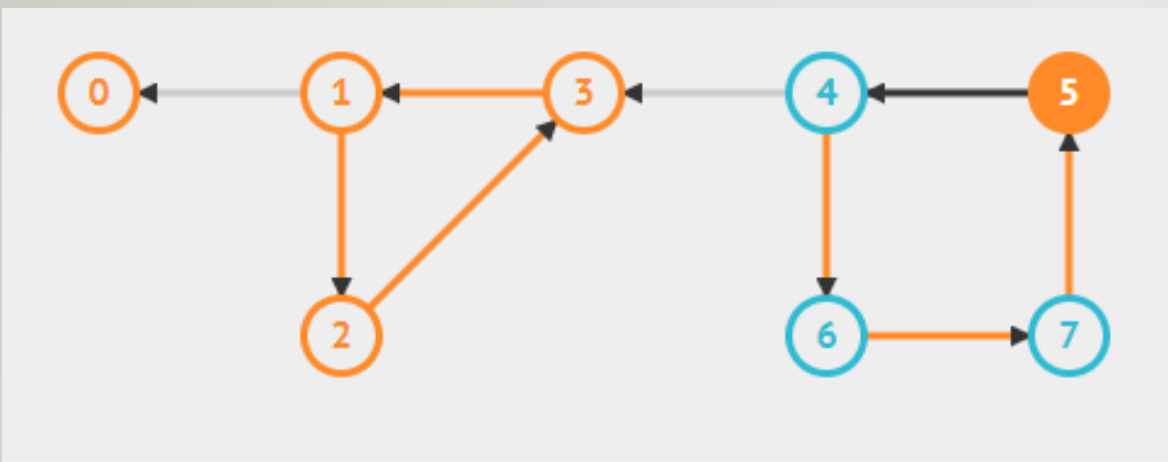
Đảo ngược cạnh

$u = \text{order.pop}()$: if u chưa thăm DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh

THUẬT TOÁN KOSARAJU

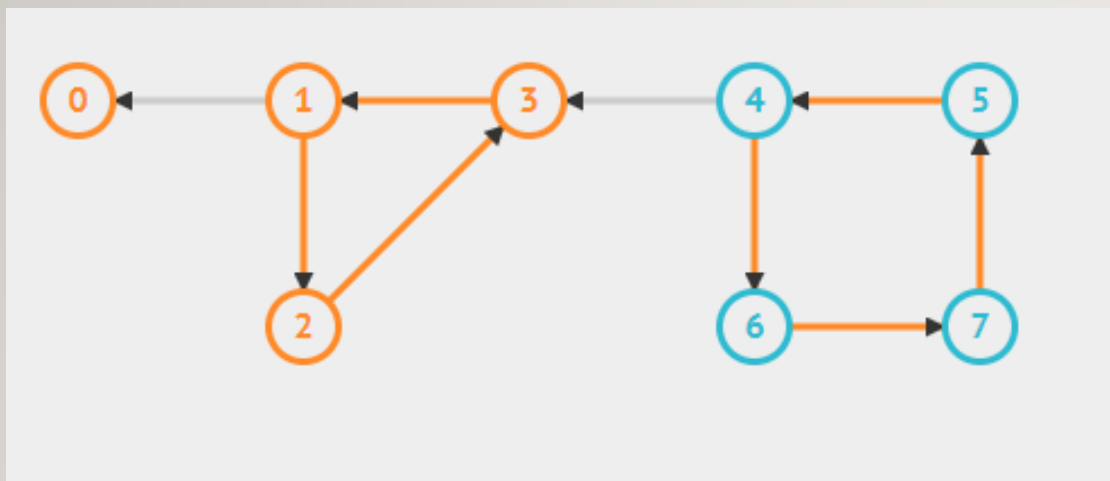


2,6,7,5

(0), (3,2,1),

```
for đỉnh u chưa thăm : DFS(u)
    for v với (u,v) ∈ E && v chưa thăm: DFS(v)
    Hoàn thành DFS(u), order.push_back(u)
Đảo ngược cạnh
u= order.pop(): if u chưa thăm DFS(u)
    for v với (u,v) ∈ E && v chưa thăm: DFS(v)
Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh
```

THUẬT TOÁN KOSARAJU

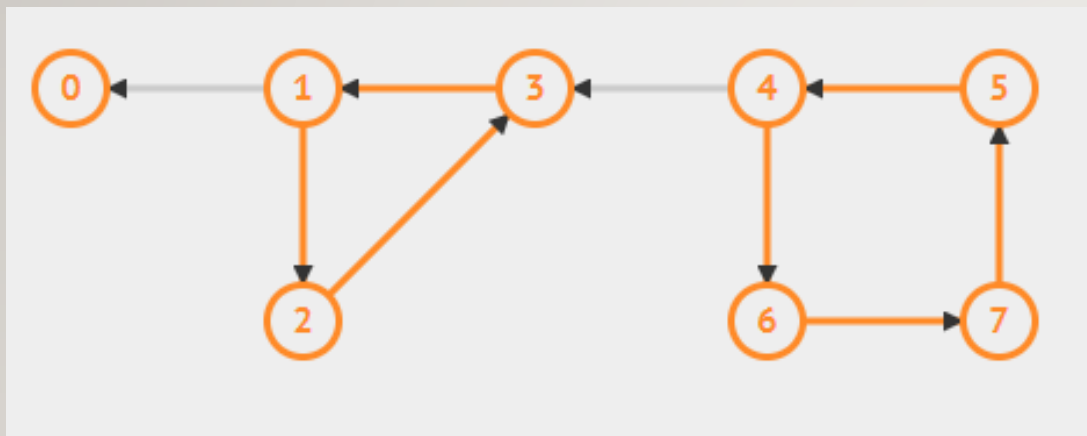


2,6,7,5

(0), (3,2,1),

```
for đỉnh u chưa thăm : DFS(u)
    for v với  $(u,v) \in E$  && v chưa thăm: DFS(v)
    Hoàn thành DFS(u), order.push_back(u)
Đảo ngược cạnh
u= order.pop(): if u chưa thăm DFS(u)
    for v với  $(u,v) \in E$  && v chưa thăm: DFS(v)
Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh
```

THUẬT TOÁN KOSARAJU



2,6,7,5

(0), (3,2,1),(5,7,6,4)

for đỉnh u chưa thăm : DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Hoàn thành DFS(u), order.push_back(u)

Đảo ngược cạnh

u= order.pop(): if u chưa thăm DFS(u)

for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

Mỗi lần hoàn thành DFS ta được một thành phần liên thông mạnh

THUẬT TOÁN TARJAN

- Ý tưởng cơ bản của thuật toán này là: tìm kiếm theo chiều sâu bắt đầu từ một đỉnh tùy ý (và sau đó tìm kiếm sâu dần trên bất kỳ các đỉnh nào chưa được xét). Việc tìm kiếm sẽ không xét đến bất kỳ đỉnh nào đã được xét trước đó. Các thành phần liên thông mạnh tạo nên các cây con của cây tìm kiếm, gốc của những cây con đó chính là gốc của các thành phần liên thông mạnh.
- Các đỉnh được đưa vào một ngăn xếp theo thứ tự mà chúng đã được xét. Khi việc tìm kiếm trả về một cây con, các đỉnh được lấy ra khỏi ngăn xếp và được xác định xem liệu mỗi đỉnh được lấy ra có phải là gốc của một thành phần liên thông mạnh hay không. Nếu một đỉnh đã được xác định là gốc của một thành phần liên thông mạnh thì nó và tất cả các đỉnh được lấy ra trước đó hình thành nên thành phần liên thông mạnh.

THUẬT TOÁN TARJAN

for đỉnh u chưa thăm :

DFS(u): s.push(u), num[u] = low[u] = count

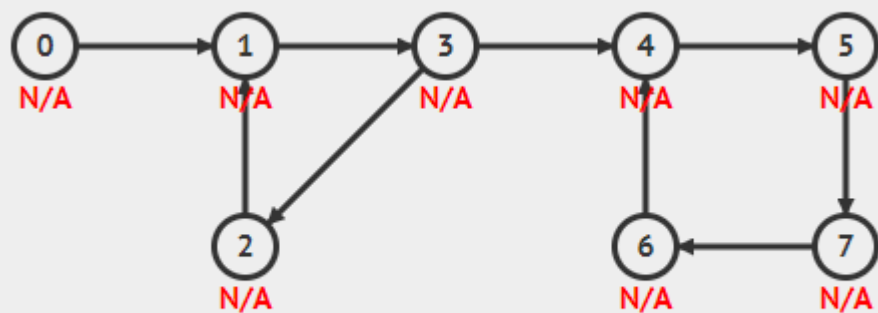
for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

low[u] = min(low[u], low[v])

if(low[u] == low[v]) //

pop stack cho đến khi nhận được cạnh u

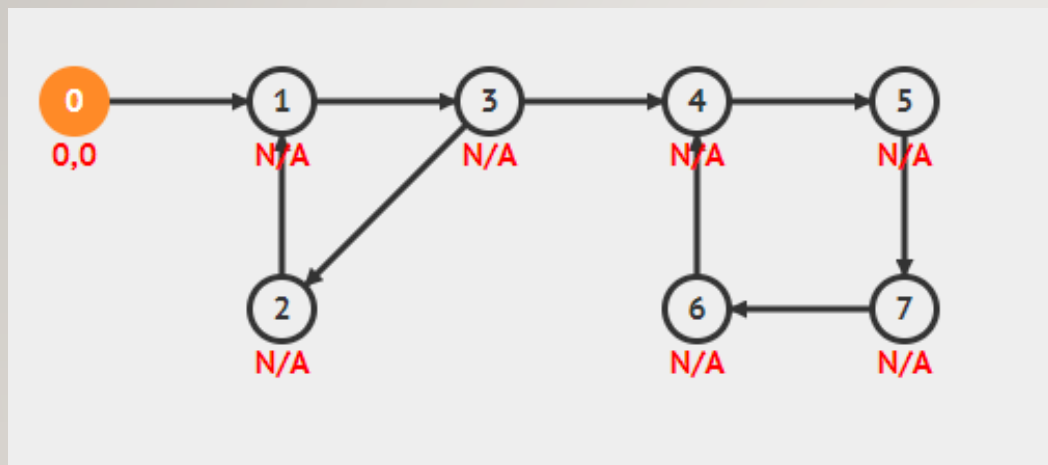
THUẬT TOÁN TARJAN



for đỉnh u chưa thăm :

```
DFS(u): s.push(u), num[u] = low[u] = count
  for v với (u,v) ∈ E && v chưa thăm: DFS(v)
  low[u] = min(low[u], low[v])
  if(low[u] == low[v]) //
    pop stack cho đến khi nhận được cạnh u
```

THUẬT TOÁN TARJAN



0

for đỉnh u chưa thăm :

DFS(u): **s.push(u), num[u] = low[u] = count**

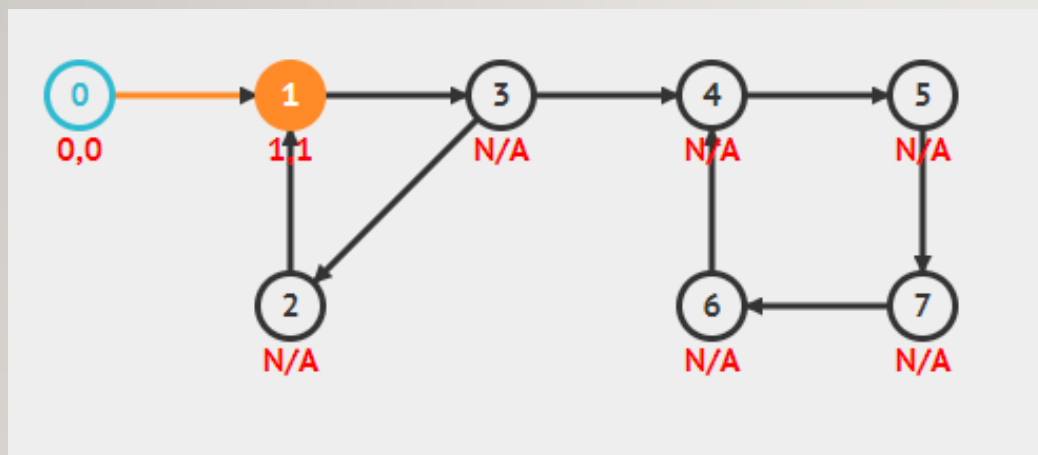
for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

$low[u] = \min(low[u], low[v])$

if($low[u] == low[v]$) //

pop stack cho đến khi nhận được cạnh u

THUẬT TOÁN TARJAN



0,1

for đỉnh u chưa thăm :

DFS(u): s.push(u), num[u] = low[u] = count

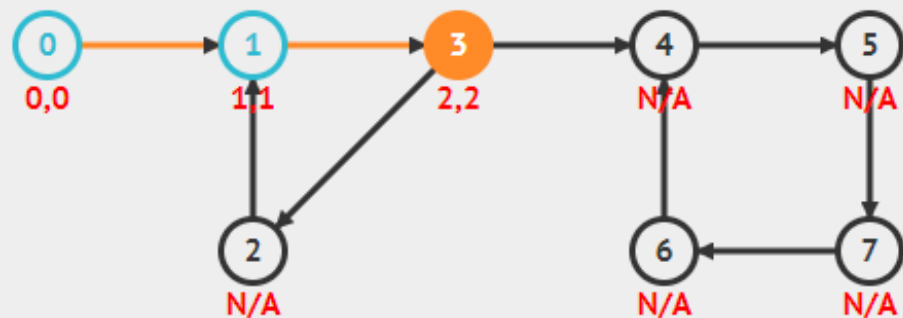
for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

low[u] = min(low[u], low[v])

if(low[u] == low[v]) //

pop stack cho đến khi nhận được cạnh u

THUẬT TOÁN TARJAN



0,1,3

for đỉnh u chưa thăm :

DFS(u): s.push(u), num[u] = low[u] = count

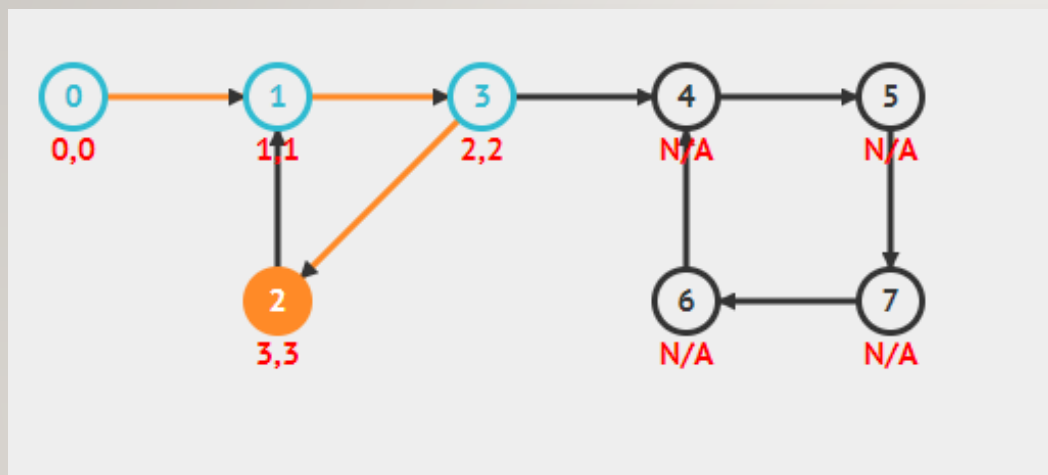
for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

low[u] = min(low[u], low[v])

if(low[u] == low[v]) //

pop stack cho đến khi nhận được cạnh u

THUẬT TOÁN TARJAN



0,1,3,2

for đỉnh u chưa thăm :

DFS(u): s.push(u), num[u] = low[u] = count

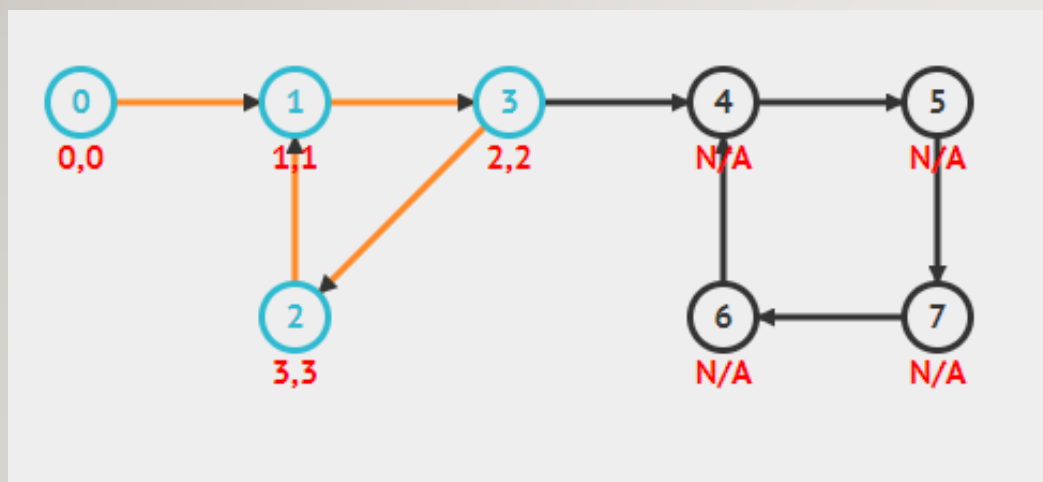
for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

low[u] = min(low[u], low[v])

if(low[u] == low[v]) //

pop stack cho đến khi nhận được cạnh u

THUẬT TOÁN TARJAN



0,1,3,2

for đỉnh u chưa thăm :

DFS(u): s.push(u), num[u] = low[u] = count

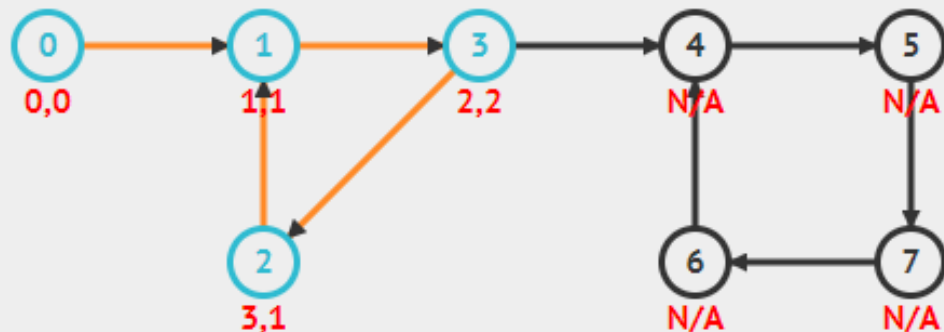
for v với (u,v) ∈ E && v chưa thăm: DFS(v)

low[u] = min(low[u], low[v])

if(low[u] == low[v]) //

pop stack cho đến khi nhận được cạnh u

THUẬT TOÁN TARJAN



0,1,3,2

for đỉnh u chưa thăm :

DFS(u): s.push(u), num[u] = low[u] = count

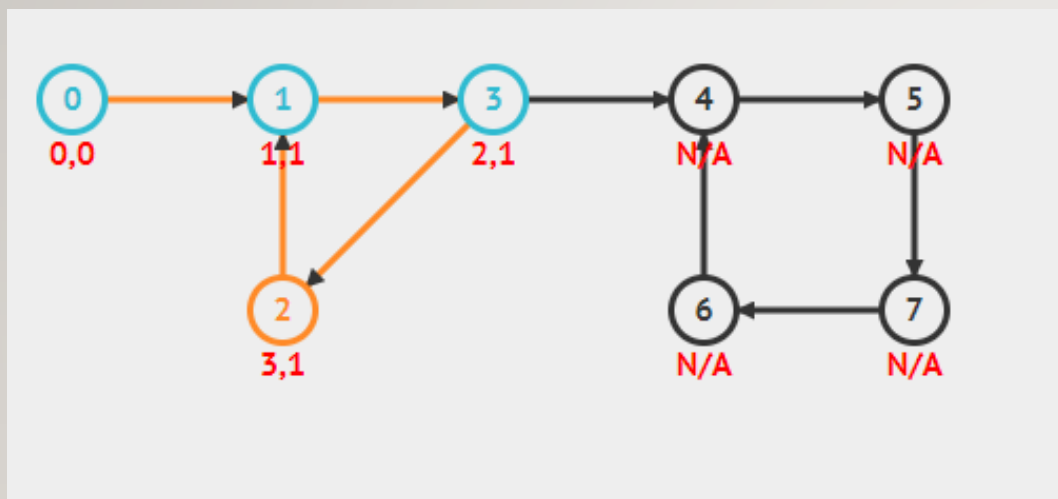
for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

low[u] = min(low[u], low[v])

if(low[u] == low[v]) //

pop stack cho đến khi nhận được cạnh u

THUẬT TOÁN TARJAN



0,1,3,2

for đỉnh u chưa thăm :

DFS(u): s.push(u), num[u] = low[u] = count

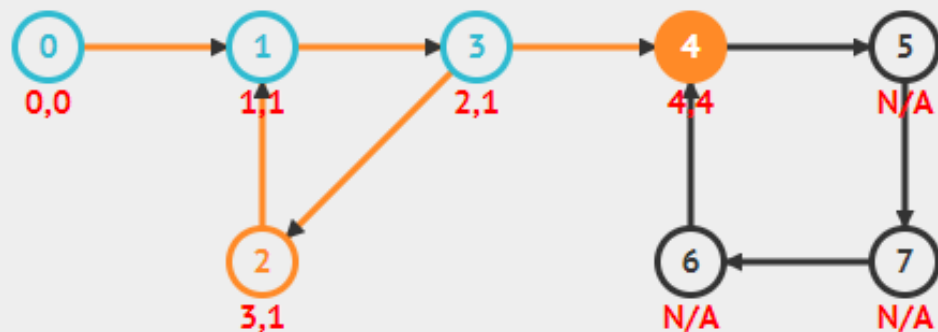
for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

low[u] = min(low[u], low[v])

if(low[u] == low[v]) //

pop stack cho đến khi nhận được cạnh u

THUẬT TOÁN TARJAN



0,1,3,2,4

for đỉnh u chưa thăm :

DFS(u): s.push(u), num[u] = low[u] = count

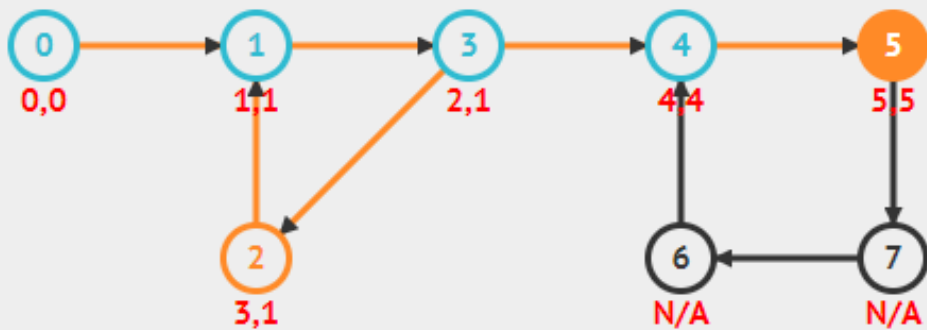
for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

low[u] = min(low[u], low[v])

if(low[u] == low[v]) //

pop stack cho đến khi nhận được cạnh u

THUẬT TOÁN TARJAN



0,1,3,2,4,5

for đỉnh u chưa thăm :

DFS(u): $s.push(u)$, $num[u] = low[u] = count$

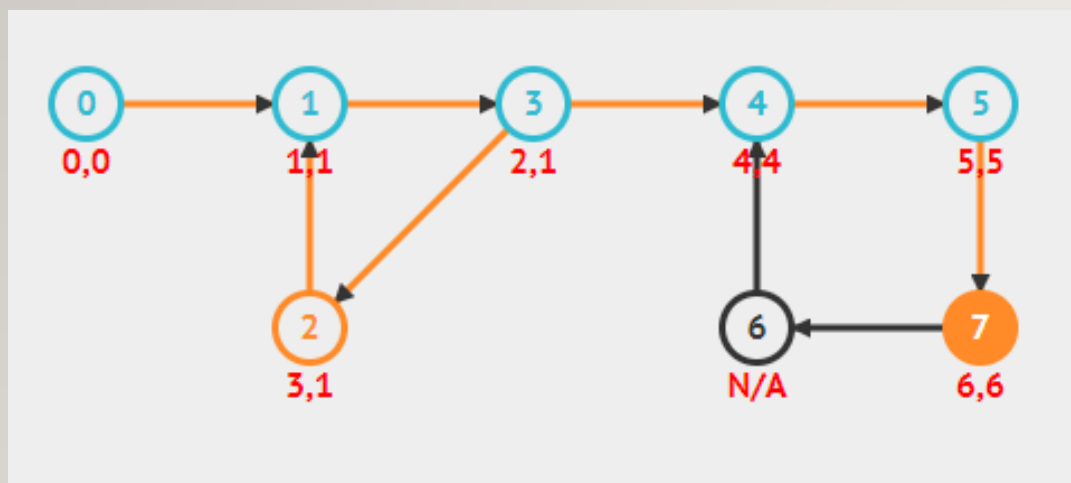
for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

$low[u] = \min(low[u], low[v])$

if($low[u] == low[v]$) //

pop stack cho đến khi nhận được cạnh u

THUẬT TOÁN TARJAN



0,1,3,2,4,5,7

for đỉnh u chưa thăm :

DFS(u): s.push(u), num[u] = low[u] = count

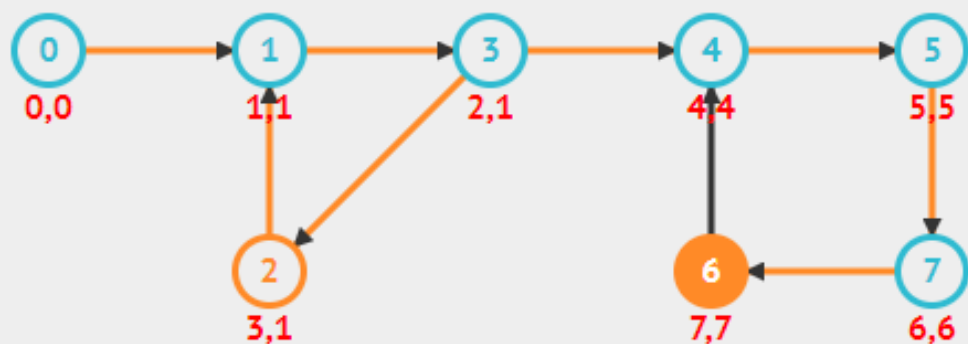
for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

low[u] = min(low[u], low[v])

if(low[u] == low[v]) //

pop stack cho đến khi nhận được cạnh u

THUẬT TOÁN TARJAN



0,1,3,2,4,5,7,6

for đỉnh u chưa thăm :

DFS(u): s.push(u), num[u] = low[u] = count

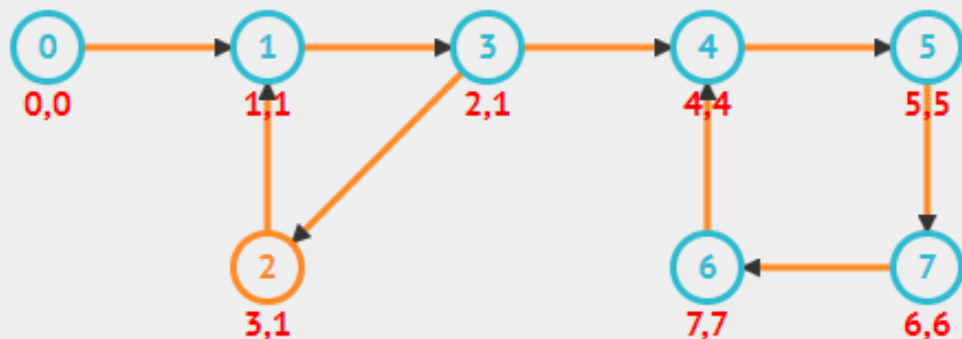
for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

low[u] = min(low[u], low[v])

if(low[u] == low[v]) //

pop stack cho đến khi nhận được cạnh u

THUẬT TOÁN TARJAN



0,1,3,2,4,5,7,6

for đỉnh u chưa thăm :

DFS(u): s.push(u), num[u] = low[u] = count

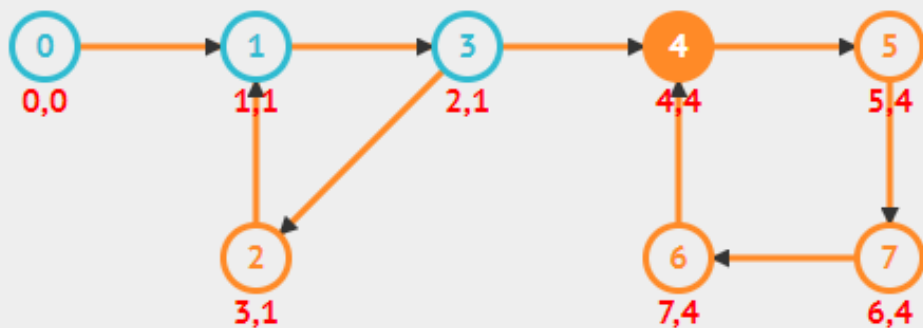
for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

low[u] = min(low[u], low[v])

if(low[u] == low[v]) //

pop stack cho đến khi nhận được cạnh u

THUẬT TOÁN TARJAN



0,1,3,2,4,5,7,6

for đỉnh u chưa thăm :

DFS(u): s.push(u), num[u] = low[u] = count

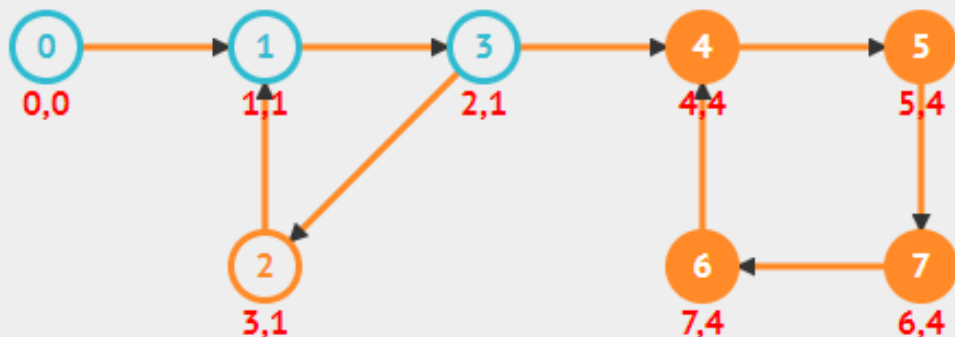
for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

low[u] = min(low[u], low[v])

if(low[u] == low[v]) //

pop stack cho đến khi nhận được cạnh u

THUẬT TOÁN TARJAN



0,1,3,2

(6,7,5,4)

for đỉnh u chưa thăm :

DFS(u): s.push(u), num[u] = low[u] = count

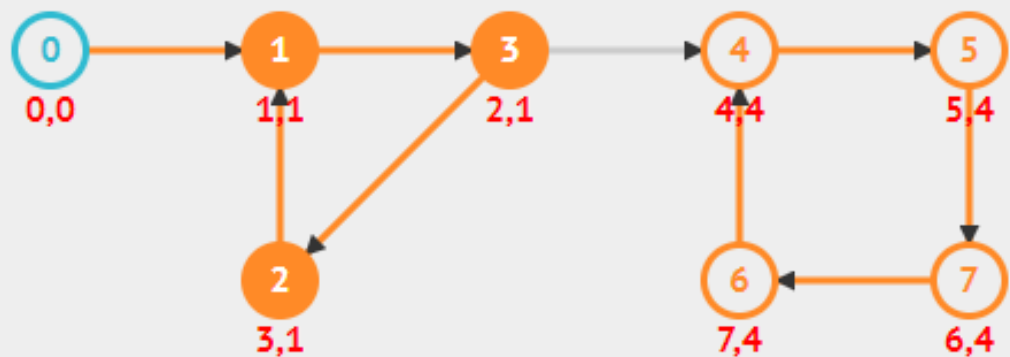
for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

low[u] = min(low[u], low[v])

if(low[u] == low[v]) //

pop stack cho đến khi nhận được cạnh u

THUẬT TOÁN TARJAN



0

(6,7,5,4),(2,3,1),

for đỉnh u chưa thăm :

DFS(u): s.push(u), num[u] = low[u] = count

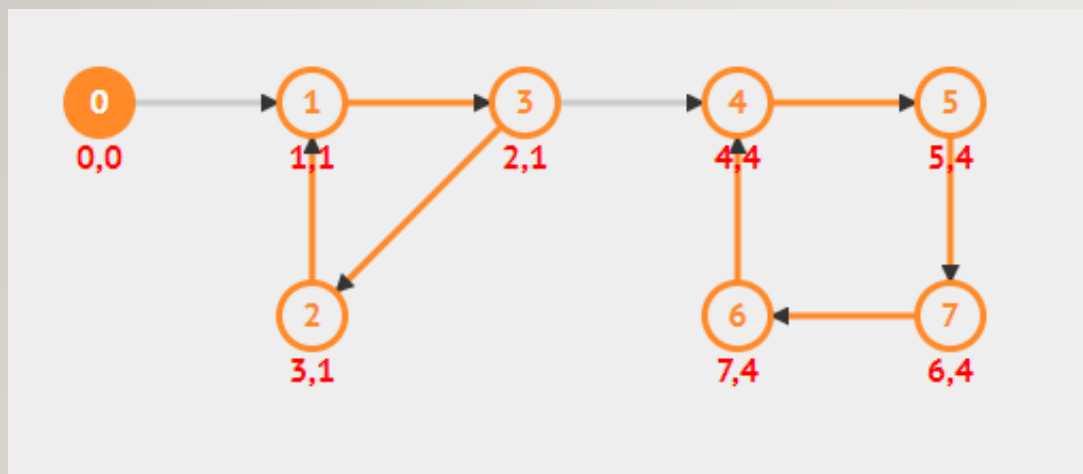
for v với $(u,v) \in E$ && v chưa thăm: DFS(v)

low[u] = min(low[u], low[v])

if(low[u] == low[v]) //

pop stack cho đến khi nhận được cạnh u

THUẬT TOÁN TARJAN



(6,7,5,4),(2,3,1),(0)

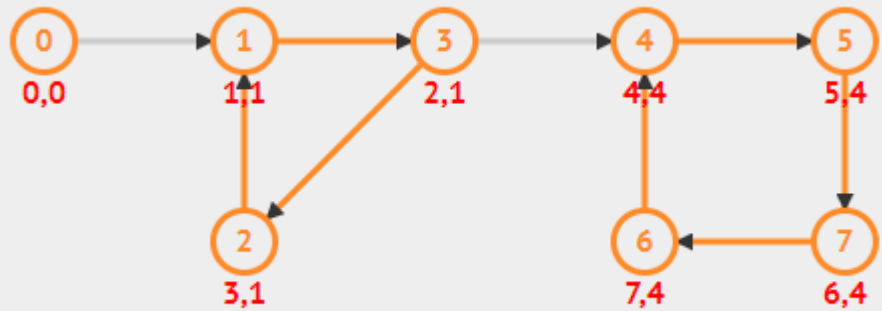
for đỉnh u chưa thăm :

DFS(u): s.push(u), num[u] = low[u] = count
for v với (u,v) $\in E$ && v chưa thăm: DFS(v)
low[u] = min(low[u], low[v])

if(low[u] == low[v]) //

pop stack cho đến khi nhận được cạnh u

THUẬT TOÁN TARJAN



(6,7,5,4),(2,3,1),(0)

OUTLINE

Tìm thành phần liên thông mạnh

- Thuật toán Kosaraju
- Thuật toán Tarjan

Bài toán 2-SAT

Bài toán 22E - codeforces

Bài toán 776D - codeforces

BÀI TOÁN 2-SAT

- Cho m biến logic: a_1, a_2, \dots, a_m và một biểu thức logic C có dạng:

$$C = (u_1 \vee v_1) \wedge (u_2 \vee v_2) \wedge \dots \wedge (u_n \vee v_n)$$

Trong đó u_i và v_i ($1 \leq i \leq n$) được thay bằng biến logic a_j hoặc $\neg a_j$ nào đó. ($1 \leq j \leq m$)

Hãy gán giá trị **TRUE/FALSE** cho các biến a_1, a_2, \dots, a_m sao cho biểu thức C nhận giá trị **TRUE**, hoặc thông báo không thể làm được. Bài toán được đưa về bài toán tìm thành phần liên thông.

- Bài toán được đưa về bài toán tìm thành phần liên thông.

BÀI TOÁN 2-SAT

- Xây dựng đồ thị có hướng $G = (V, E)$,

$$\begin{cases} V = \{a_1, \neg a_1, a_2, \neg a_2, \dots, a_m, \neg a_m\} \\ E = \{ (u \rightarrow v) \mid (\neg u, v) \in E' \} \end{cases}$$

- Như vậy đồ thị mới sẽ có $2m$ đỉnh và $2n$ cung có hướng.

BÀI TOÁN 2-SAT

1. Trên đồ thị $G = (V, E)$, tìm các thành phần liên thông mạnh C_1, C_2, \dots, C_k .
2. Nếu có hai đỉnh đối lập trong cùng 1 TPLTM thì thông báo bài toán vô nghiệm và kết thúc.
3. Gộp các đỉnh thuộc cùng một TPLTM, xây dựng đồ thị DAG $G_c V_c, E_c$:

$$\begin{cases} V_c = \{ C_i \mid 1 \leq i \leq k \} \\ E_c = \{ (C_i \rightarrow C_j) \mid \exists u, v \in V: u \in C_i, v \in C_j \text{ và } (u \rightarrow v) \in E \} \end{cases}$$

Ta định nghĩa 2 đỉnh đối lập trên G_c như sau

$$C_i = \neg C_j \Leftrightarrow \exists u \in V: u \in C_i \text{ và } \neg u \in C_j$$

4. Trên đồ thị G_c :

1. Khởi tạo tất cả các đỉnh đều chưa được gán giá trị.
2. Sắp xếp các đỉnh theo thứ tự Topo vào danh sách $L[1 \dots k]$.
3. Duyệt các đỉnh lần lượt theo thứ tự “Topo ngược” ($u = L[k] \rightarrow L[1]$):
 1. Nếu đỉnh $u \in V_c$ đang duyệt chưa được gán giá trị thì gán $u = \text{TRUE}$.
 2. Gán giá trị FALSE cho $\neg u$ và $\forall v \in V_c: v \in A(\neg u)$.

5. Từ cách gán giá trị cho các đỉnh trong G_c suy ra giá trị của các đỉnh trong G .

BÀI TOÁN 2-SAT

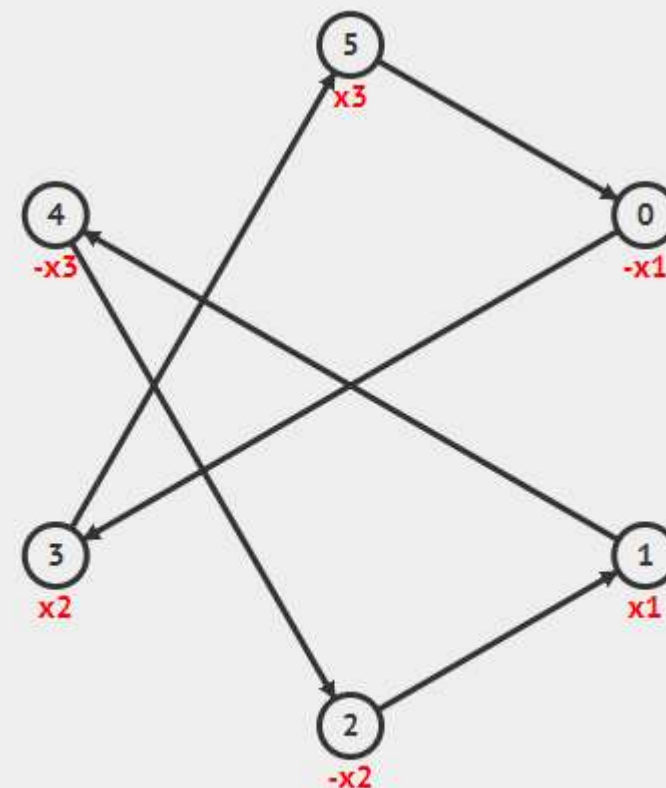
- Với bài toán 2-SAT có 3 biến:

$$(x_1 \vee x_2) \vee (\neg x_2 \vee x_3) \vee (\neg x_1 \vee \neg x_3)$$

$$(x_1 \vee x_2): (\neg x_1 \rightarrow x_2) \text{ và } (\neg x_2 \rightarrow x_1)$$

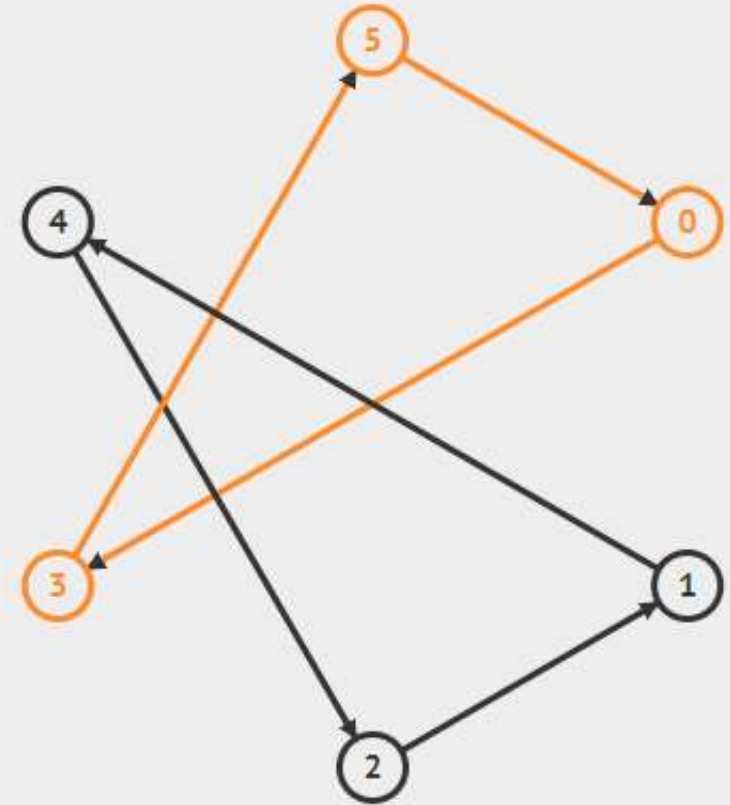
$$(\neg x_2 \vee x_3): (x_2 \rightarrow x_3) \text{ và } (\neg x_3 \rightarrow \neg x_2)$$

$$(\neg x_1 \vee \neg x_3): (x_1 \rightarrow \neg x_3) \text{ và } (x_3 \rightarrow \neg x_1)$$



BÀI TOÁN 2-SAT

- Kết quả : x_1 : false, x_2 :true, x_3 :true



OUTLINE

Tìm thành phần liên thông mạnh

- Thuật toán Kosaraju
- Thuật toán Tarjan

Bài toán 2-SAT

Bài toán 22E - codeforces

Bài toán 776D - codeforces

BÀI TOÁN 22E

- Đề bài: Để có thể nhận được tin tức về hệ điều hành mới sớm nhất có thể, cộng đồng BolgenOS từ Nizhni Tagil quyết định phát triển một kế hoạch. Theo kế hoạch đó, thành viên đầu tiên nhận được tin tức sẽ thông báo cho một thành viên khác, thành viên này lại thông báo cho một thành viên khác nữa biết tin tức, cứ như vậy cho đến hết,, vd thành viên thứ i khi nhận được tin tức sẽ thông báo cho thành viên thứ fi . Sau một thời gian, các thành viên nhận thấy kế hoạch của họ không có tác dụng, một số thành viên không nhận được tin tức. Họ muốn bổ sung cho kế hoạch bằng cách thêm vào một số cặp (x,y) với ý nghĩa người thứ x sẽ thông báo cho cả người thứ y biết về tin tức. Tìm số cặp tối thiểu mà họ phải thêm

BÀI TOÁN 22E

- Đầu vào:

Dòng đầu tiên là số n ($2 \leq n \leq 10^5$) tổng số thành viên trong cộng đồng BolgenOS.

Dòng thứ 2 chứa n số nguyên F_i ($1 \leq F_i \leq n$) là chỉ số của một người sẽ được gọi điện.

- Đầu ra:

Dòng đầu tiên là một số nguyên chỉ số lượng tối thiểu cặp (x,y) cần thêm vào.

Các dòng tiếp theo in ra các cặp (x,y) tương ứng.

- Giới hạn thời gian 2s, bộ nhớ 256MB

BÀI TOÁN 22E

- Mô hình hóa là một đồ thị G với cung (u,v) là người thứ u sẽ gọi cho người thứ v nếu nhận được tin. Tìm số cạnh nhỏ nhất cần thêm để đồ thị là liên thông mạnh.
- Ta thấy nổi đỉnh trong đồ thị có bán bậc ra đúng bằng 1 do một người khi nhận được tin chỉ gọi cho một người khác.

BÀI TOÁN 22E

- Đầu tiên, xét các đỉnh u có bán bậc vào bằng 0, duyệt đường đi bắt đầu tại u , đánh dấu u là đỉnh bắt đầu. Duyệt đến khi gặp đỉnh đã duyệt qua rồi, tức là đến khi gặp cạnh ngược, đánh dấu điểm đó là điểm kết thúc của đường đi.
- Tiến hành duyệt những đỉnh w chưa được đi qua trong đồ thị G tại bước 1. Những đỉnh này thuộc vào các thành phần liên thông khác nhau trong G . Điểm bắt đầu và kết thúc của đường đi này đều là w . Đánh dấu các điểm thuộc thành phần liên thông bắt đầu tại w là đã duyệt.
- Nối các đường đi lại với nhau bằng cách thêm các cung từ đỉnh kết thúc thành phần thứ i sang đỉnh bắt đầu thành phần thứ $(i+1)\%k$ với k là số đường đi của đồ thị G . Nếu $k=1$, và đỉnh đầu trùng đỉnh cuối G có duy nhất một đường đi, do đó G là đồ thị liên thông mạnh, không cần thêm bất cứ cung nào nữa. Với $k \geq 2$ cần thêm k cạnh để nối k thành phần lại với nhau.

BÀI TOÁN 22E

- Với bộ dữ liệu đầu vào

3

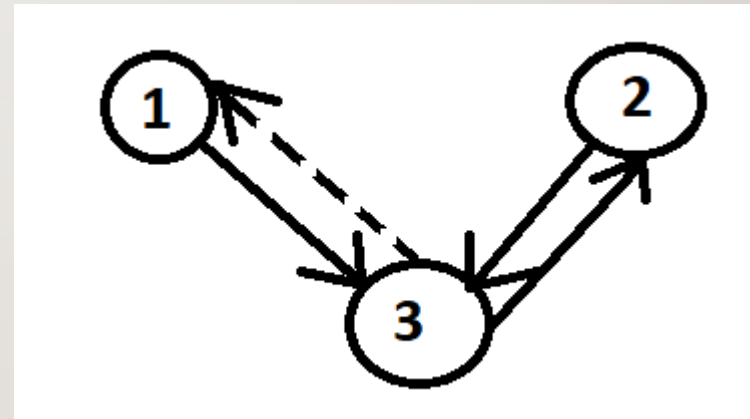
3 3 2

- Đồ thị tương ứng

- Kết quả

3


3 1






BÀI TOÁN 22E

- Bộ test
 - Test 1-9 <100
 - 10-20 1000-10000
 - 21-30 10000-100000
 - 31 đồ thị đã liên thông

- Submit code



pmihntam | Logout

[HOME](#)
[TOP](#)
[CONTESTS](#)
[GYM](#)
[PROBLEMSET](#)
[GROUPS](#)
[RATING](#)
[API](#)
[HELP](#)
[LYFT](#)
[MAILRU CUP](#)
[CALENDAR](#)

Please subscribe to the official Codeforces channel in Telegram via the link: https://t.me/codeforces_official

[PROBLEMS](#)
[SUBMIT CODE](#)
[MY SUBMISSIONS](#)
[STATUS](#)
[STANDINGS](#)
[CUSTOM INVOCATION](#)

My Submissions

#	When	Who	Problem	Lang	Verdict	Time	Memory
46427349	2018-12-01 05:08:18	pmihntam	E - Scheme	GNU C++14	Accepted	560 ms	2000 KB
46357343	2018-11-29 16:51:22	pmihntam	E - Scheme	GNU C++14	Accepted	530 ms	2000 KB
45565727	2018-11-11 17:20:34	pmihntam	E - Scheme	GNU C++14	Wrong answer on test 10	62 ms	1900 KB
45565682	2018-11-11 17:19:01	pmihntam	E - Scheme	GNU C++14	Wrong answer on test 11	62 ms	1900 KB

Codeforces Beta Round #22 (Div. 2 Only)

Finished

Practice

★

— Virtual participation

Virtual contest is a way to take part in past contest, as close as possible to participation on time. It is supported only ACM-ICPC mode for virtual contests. If you've seen these problems, a virtual contest is not for you - solve these problems in the archive. If you just want to solve some problem from a contest, a virtual contest is not for you - solve this problem in the archive. Never use someone else's code, read the tutorials or communicate with other person during a virtual contest.

Start virtual contest

— Practice

You are registered for practice. You can solve problems unofficially. Results can be found in the contest status and in the bottom of standings.

OUTLINE

Tìm thành phần liên thông mạnh

- Thuật toán Kosaraju
- Thuật toán Tarjan

Bài toán 2-SAT

Bài toán 22E - codeforces

Bài toán 776D - codeforces

BÀI TOÁN 776D

- Đề bài: Có n người bị kẹt trong n phòng khác nhau. Một số phòng bị khóa, một số khác thì không. Điều kiện để mọi người có thể thoát khỏi là tất cả các phòng phải được mở cùng một lúc. Có m công tắc. Mỗi công tắc điều khiển một số phòng, nhưng mỗi phòng chỉ bị điều khiển bởi đúng 2 công tắc. Bạn được đưa trạng thái ban đầu của các phòng. Nhấn bất kỳ công tắc nào nó sẽ đảo ngược trạng thái phòng mà công tắc đó điều khiển. Ví dụ công tắc 1 điều khiển 3 phòng 1, 2, 3 với trạng thái tương ứng là đóng, mở, mở. Khi nhấn công tắc 1 sẽ chuyển thành mở, đóng, đóng. Bạn cần nói cho Sherlock biết liệu có tồn tại một cách nào mở tất cả các phòng tại một thời điểm không.

BÀI TOÁN 776D

- Đầu vào:

Dòng đầu tiên là 2 số n, m ($2 \leq n \leq 10^5, 2 \leq m \leq 10^5$) là số phòng và số công tắc

Dòng tiếp theo là n số nguyên r_i ($r_i \in \{0,1\}$) là trạng thái các phòng với 0 là phòng bị khóa, 1 là phòng được mở.

m dòng tiếp theo là số nguyên x_i ($0 \leq x_i \leq n$) và x_i số nguyên ngăn cách nhau bởi khoảng trắng chỉ ra các phòng được điều khiển bởi khóa thứ i . Đảm bảo điều kiện phòng trong khoảng từ 1 đến n và mỗi phòng được điều khiển bởi duy nhất 2 công tắc.

BÀI TOÁN 776D

- Đầu ra:

In ra "YES" nếu có thể mở tất cả các phòng một lúc, ngược lại in ra "NO".

- Giới hạn thời gian 2s, bộ nhớ 256MB

BÀI TOÁN 776D

- Chuyển về bài toán đồ thị tô màu các đỉnh. Phòng là các cạnh và công tắc là các đỉnh. Phòng mở ứng với cạnh có giá trị 1, ngược lại nếu đóng thì cạnh có giá trị 0. Nếu phòng có giá trị 1 thì 2 đỉnh nút của nó phải có cùng màu, nếu là 0 thì 2 đỉnh phải có màu khác nhau. Tô màu đỉnh bằng 2 màu, gặp cạnh 0 sẽ đổi màu đỉnh sau đó, nếu gặp cạnh 1 sẽ giữ nguyên màu.

BÀI TOÁN 776D

- Bộ dữ liệu đầu vào

3 3

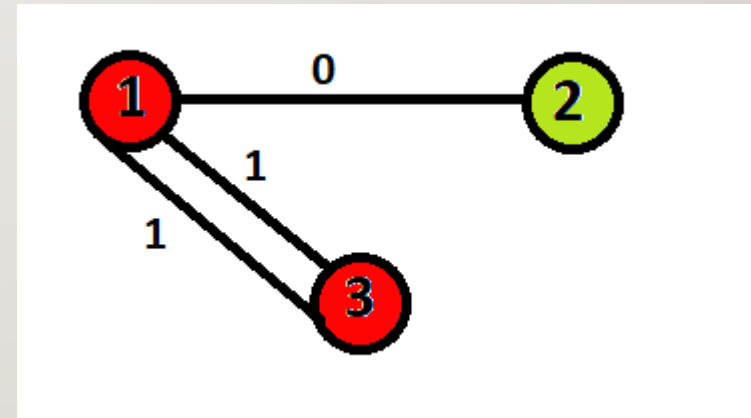
1 0 1

3 1 2 3

1 2

2 1 3

- Đồ thị
- Kết quả
YES

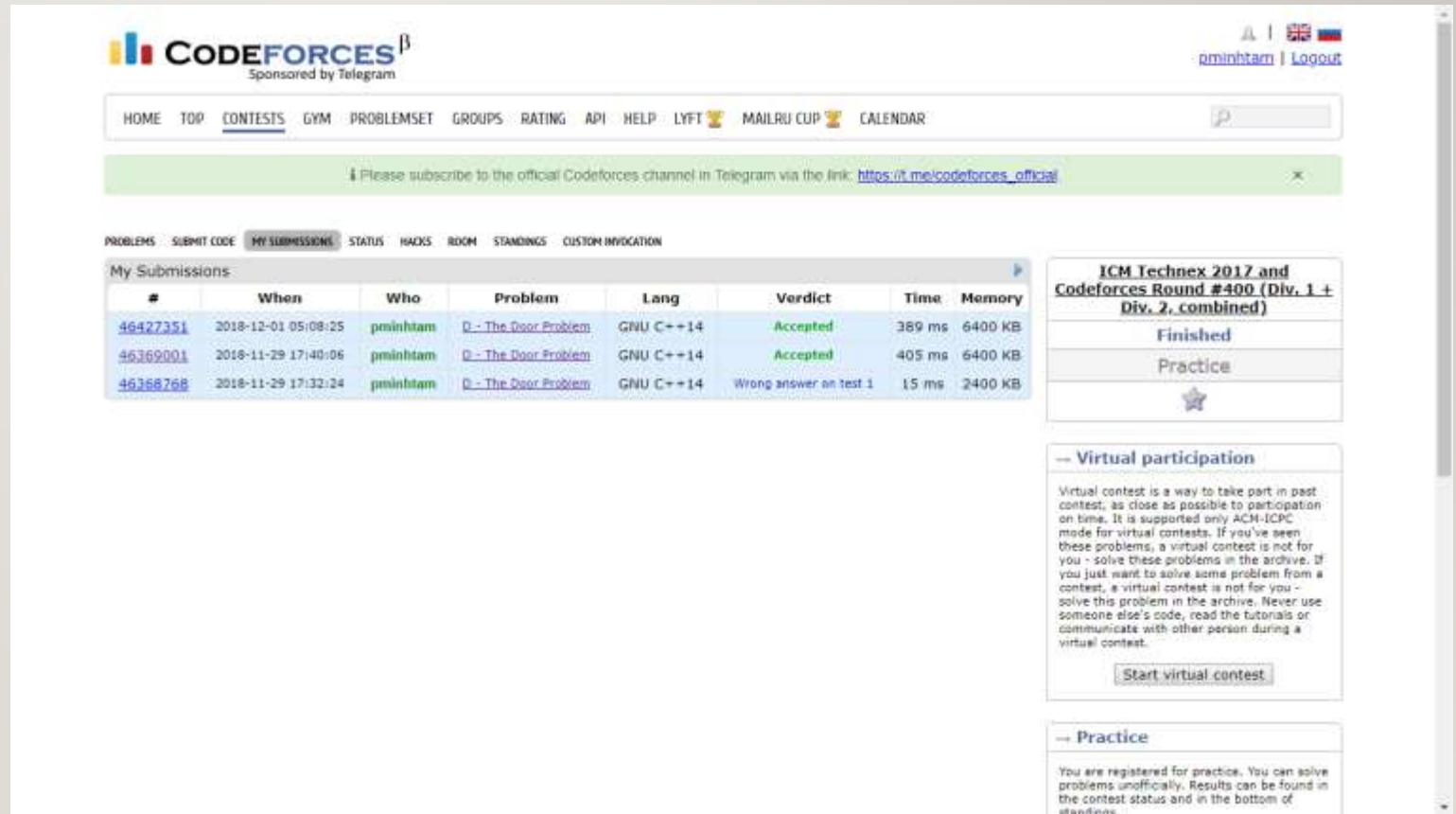


BÀI TOÁN 776D

- Bộ test
 - 1-9: max 100 phòng, 100 công tắc, mỗi công tắc max 10 phòng
 - 10-20: 1000 phòng, 1000 công tắc, max 5 phòng
 - 21-30 1000 phòng, 1000 công tắc, max 100 phòng
 - 31-40 10000 phòng, 10000 công tắc, max 20 phòng
 - 41-50 100000 phòng, 100000 công tắc, max 20 phòng
 - 51 tất cả các phòng đều đóng
 - 52-55 tất cả các phòng đều mở

BÀI TOÁN 776D

- Submit code



CODEFORCES^β
Sponsored by Telegram

HOME TOP CONTESTS GYM PROBLEMSET GROUPS RATING API HELP LYFT MAILRU CUP CALENDAR

Please subscribe to the official Codeforces channel in Telegram via the link: https://t.me/codeforces_official

PROBLEMS SUBMIT CODE MY SUBMISSIONS STATUS HACKS ROOM STANDINGS CUSTOM INVOCATION

My Submissions

#	When	Who	Problem	Lang	Verdict	Time	Memory
46427351	2018-12-01 05:08:25	pminhtam	D - The Door Problem	GNU C++14	Accepted	389 ms	6400 KB
46369001	2018-11-29 17:40:06	pminhtam	D - The Door Problem	GNU C++14	Accepted	405 ms	6400 KB
46368768	2018-11-29 17:32:24	pminhtam	D - The Door Problem	GNU C++14	Wrong answer on test 1	15 ms	2400 KB

ICM Technex 2017 and Codeforces Round #400 (Div. 1 + Div. 2, combined)

Finished

Practice

★

→ Virtual participation

Virtual contest is a way to take part in past contest, as close as possible to participation on time. It is supported only ACM-ICPC mode for virtual contests. If you've seen these problems, a virtual contest is not for you - solve these problems in the archive. If you just want to solve some problem from a contest, a virtual contest is not for you - solve this problem in the archive. Never use someone else's code, read the tutorials or communicate with other person during a virtual contest.

Start virtual contest

→ Practice

You are registered for practice. You can solve problems unofficially. Results can be found in the contest status and in the bottom of standings.

Chương trình quản lý học sinh

Kỳ thi Bài thi Thi sinh Chấm bài Hướng dẫn

Bảng điểm Hướng dẫn

Thi sinh	✓ 22E	✓ 776D	Tổng điểm
✓ 22E	✓ 31.00	3	31.00
✓ 776D	3	✓ 55.00	55.00

Sheet1 - Excel

File Home Insert Page Layout Formulas Data Review View Help Tell me

Q7

	A	B	C	E	F	G	H	I	J
1	Mã thi sinh	22E	776D	Tổng điểm					
2	22E	31.00	3 không nộp	31.00					
3	776D	3 không nộp	55.00	55.00					
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									

Tổng hợp điểm Chi tiết chấm

9:06 AM 12/1/2018

THANKS FOR
LISTENING

