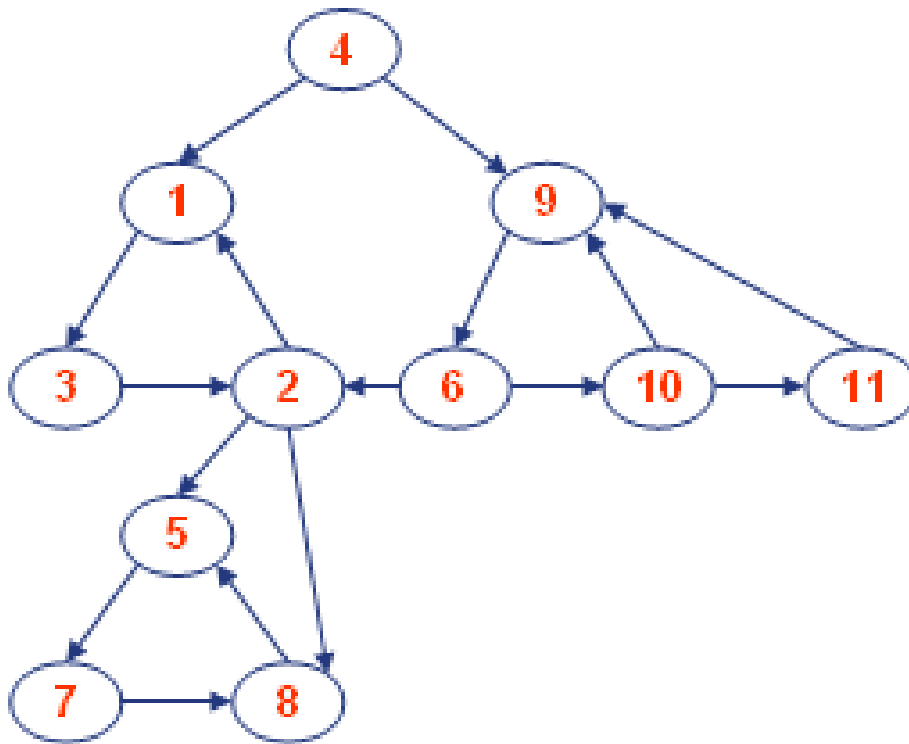




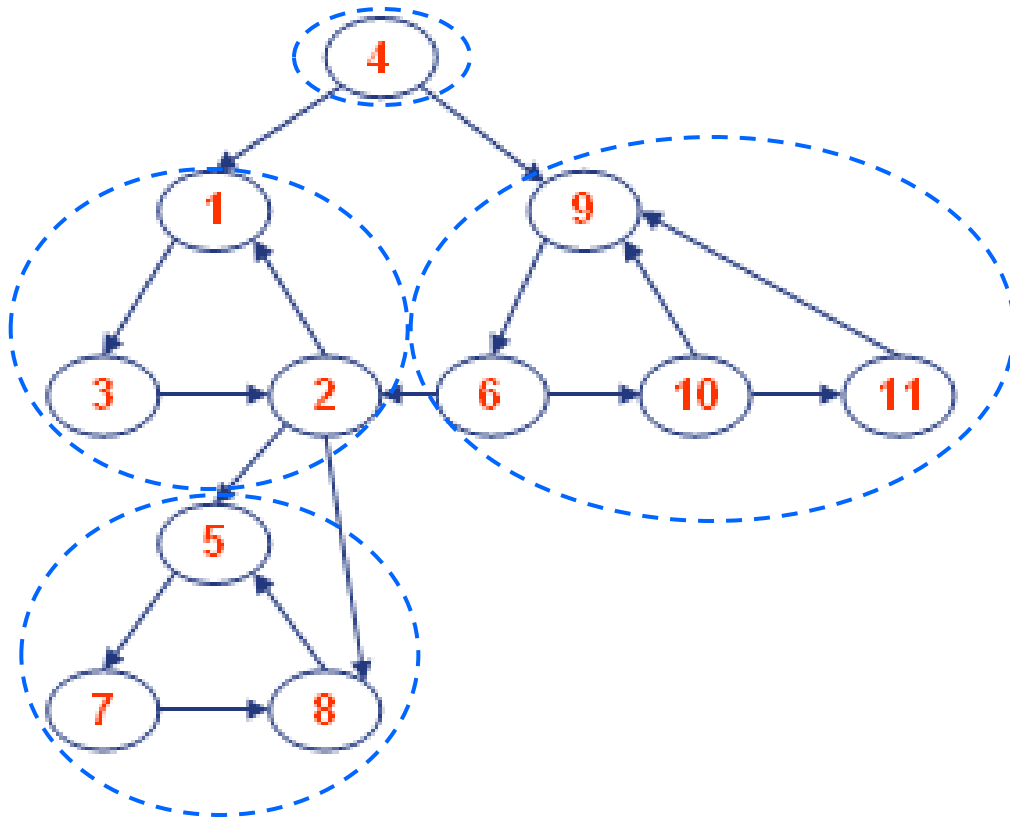
BÀI TOÁN LIỆT KÊ CÁC THÀNH PHẦN LIÊN THÔNG MẠNH TRÊN ĐỒ THỊ CÓ HƯỚNG

THÀNH PHẦN LIÊN THÔNG MẠNH



Hãy xác định các thành phần liên thông mạnh của đồ thị có hướng bên trái.

THÀNH PHẦN LIÊN THÔNG MẠNH



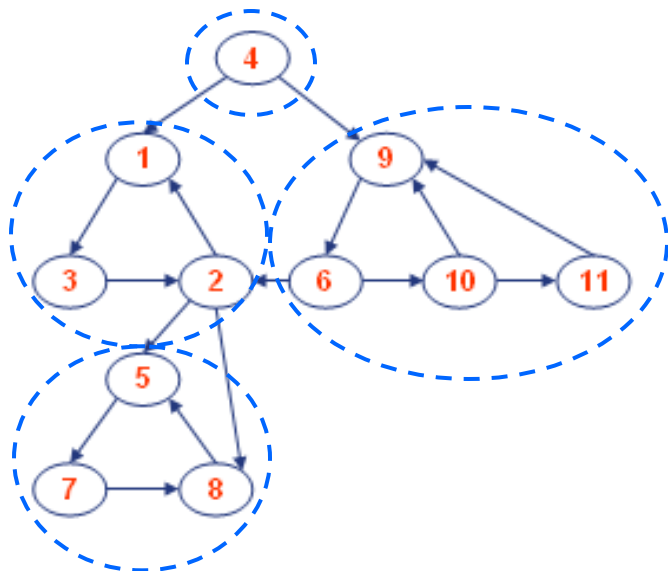
Hãy xác định các thành phần liên thông mạnh của đồ thị có hướng bên trái.

BÀI TOÁN LIỆT KÊ CÁC THÀNH PHẦN LIÊN THÔNG MẠNH

Input: file văn bản SCONNECT.INP:

- ❖ Dòng đầu: Ghi số đỉnh n (≤ 100) và số cung m của đồ thị cách nhau một dấu cách
- ❖ m dòng tiếp theo, mỗi dòng ghi hai số nguyên u, v cách nhau một dấu cách thể hiện có cung (u, v) trong đồ thị

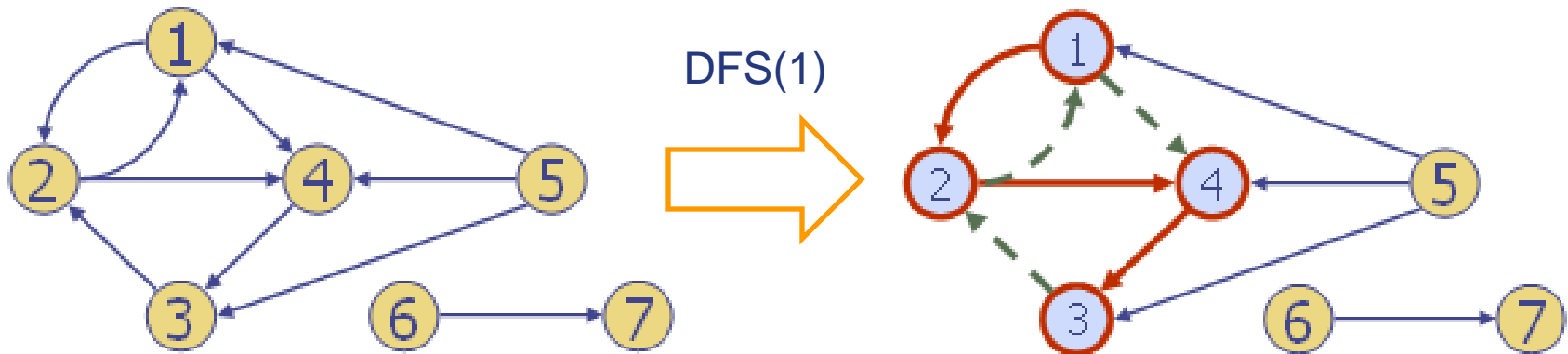
Output: file văn bản SCONNECT.OUT, liệt kê các thành phần liên thông mạnh



SCONNECT.INP	SCONNECT.OUT
11 16	Component 1:
4 1	8, 7, 5,
1 3	Component 2:
3 2	2, 3, 1,
2 1	Component 3:
2 5	11, 10, 6, 9,
2 8	Component 4:
5 7	4,
7 8	
8 5	
4 9	
9 6	
6 2	
6 10	
10 9	
10 11	
11 9	

CÂY DFS

- ❖ Quá trình tìm kiếm theo chiều sâu DFS(s) cho ta một cây DFS gốc s.
- ❖ Quan hệ *cha-con* trên cây được định nghĩa là: nếu từ đỉnh u tới thăm đỉnh v (DFS(u) gọi DFS(v)) thì u được gọi là nút cha của nút v.



CÂY DFS

```
Procedure DFS(u:integer) ;  
Var v:integer;  
Begin  
  //Đánh dấu u đã thăm  
  inc(count); number[u]:=count;  
  for v:=1 to n do //xét mỗi đỉnh v  
    if a[u,v] then //nếu v kề u  
      if number[v]=0 then //nếu v chưa thăm  
        DFS(v);
```

End;

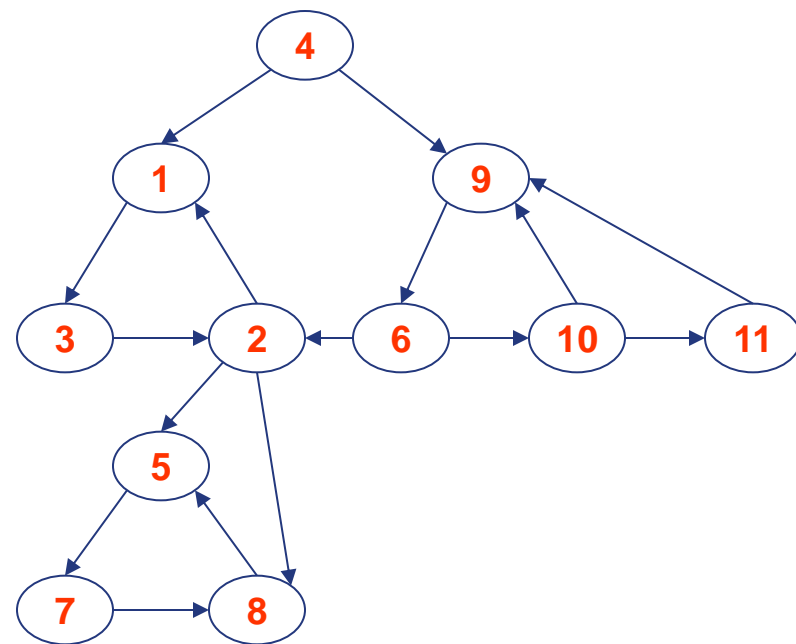
BEGIN

.....

```
  for v:=1 to n do number[v]:=0;  
  count:=0;  
  DFS(2);
```

.....

END.



Dựa theo quá trình duyệt DFS(2) thực hiện:

- Liệt kê các đỉnh và thứ tự thăm.
- Liệt kê các cung (u,v) nối đỉnh u đã thăm đến đỉnh v chưa thăm

CÂY DFS

```

Procedure DFS(u:integer) ;
Var v:integer;
Begin
    //Đánh dấu u đã thăm
    inc(count); number[u]:=count;
    for v:=1 to n do //xét mỗi đỉnh v
        if a[u,v] then //nếu v kề u
            if number[v]=0 then //nếu v chưa thăm
                DFS(v);

```

End;

BEGIN

.....

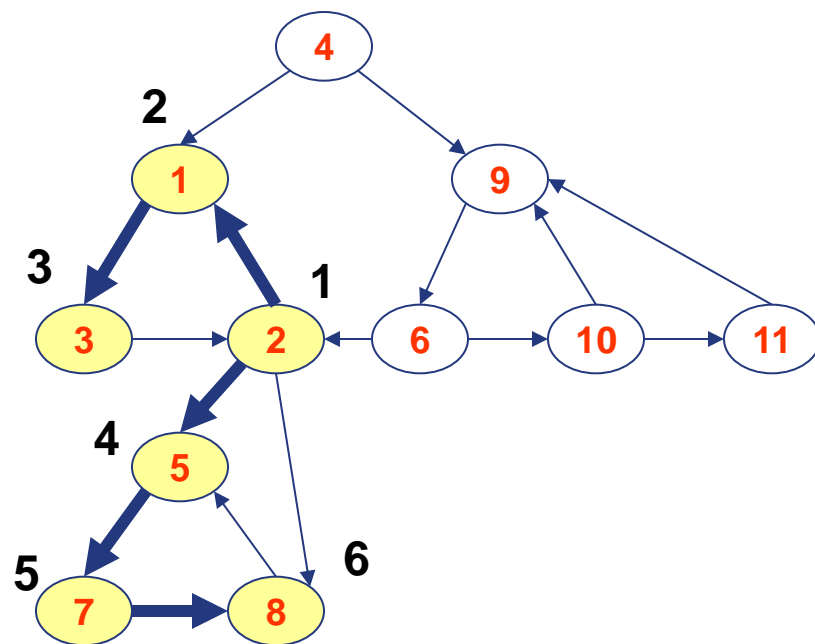
```

for v:=1 to n do number[v]:=0;
count:=0;
DFS(2);

```

.....

END.



Gọi **DFS(2)**,

Các đỉnh và thứ tự thăm:

2, 1, 3, 5, 7, 8

1 2 3 4 5 6

Các cung (u,v) nối đỉnh u đã thăm đến đỉnh v chưa thăm:

(2,1) (1,3) (2,5) (5,7) (7,8)

CÂY DFS

```
Procedure DFS(u:integer) ;  
Var v:integer;  
Begin  
  //Đánh dấu u đã thăm  
  inc(count); number[u]:=count;  
  for v:=1 to n do //xét mỗi đỉnh v  
    if a[u,v] then //nếu v kề u  
      if number[v]=0 then //nếu v chưa thăm  
        DFS(v);
```

End;

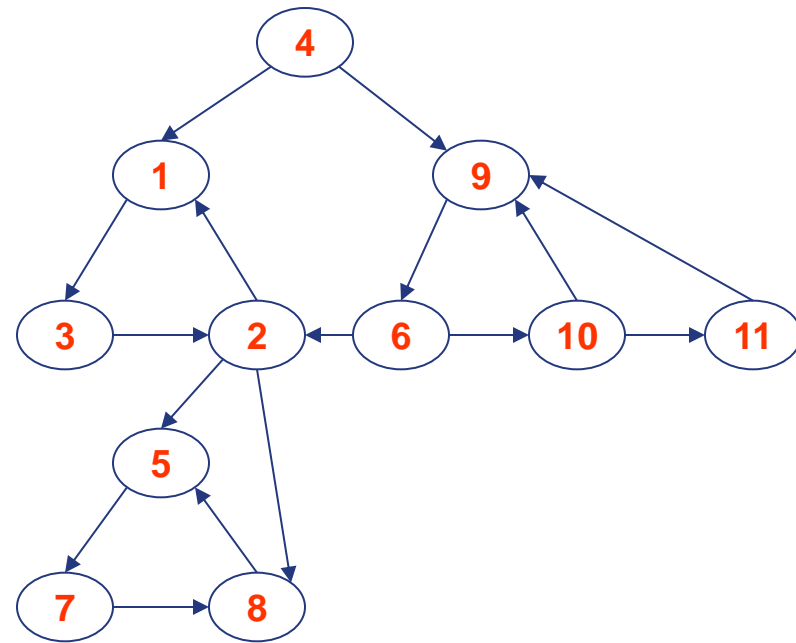
BEGIN

.....

```
  for v:=1 to n do number[v]:=0;  
  count:=0;  
  DFS(10);
```

.....

END.



Dựa theo quá trình duyệt DFS(10) thực hiện:

- Liệt kê các đỉnh và thứ tự thăm.
- Liệt kê các cung (u,v) nối đỉnh u đã thăm đến đỉnh v chưa thăm

CÂY DFS

Procedure DFS(u:integer) ;

Var v:integer;

Begin

//Đánh dấu u đã thăm

inc(count); number[u]:=count;

for v:=1 **to** n **do** *//xét mỗi đỉnh v*

if a[u,v] **then** *//nếu v kề u*

if number[v]=0 **then** *//nếu v chưa thăm*

DFS(v);

End;

BEGIN

.....

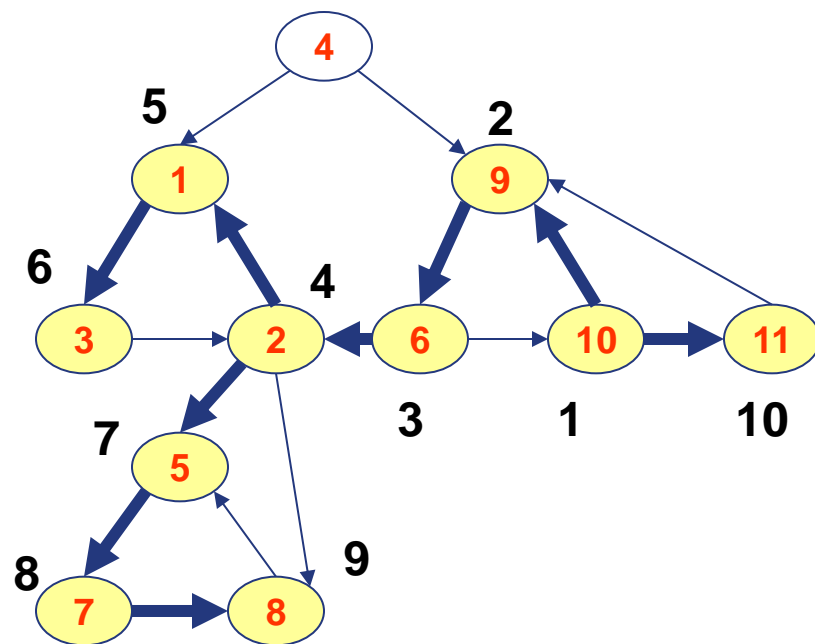
for v:=1 **to** n **do** number[v]:=0;

count:=0;

DFS(10);

.....

END.



Gọi DFS(10),

Các đỉnh và thứ tự thăm:

10, 9, 6, 2, 1, 3, 5, 7, 8, 11

1 2 3 4 5 6 7 8 9 10

Các cung (u,v) nối đỉnh u đã thăm đến đỉnh v chưa thăm:

**(10,9) (9,6) (6,2) (2,1) (1,3) (2,5)
(5,7) (7,8) (10,11)**

CÂY DFS

```
Procedure DFS(u:integer) ;  
Var v:integer;  
Begin  
  //Đánh dấu u đã thăm  
  inc(count); number[u]:=count;  
  for v:=1 to n do //xét mỗi đỉnh v  
    if a[u,v] then //nếu v kề u  
      if number[v]=0 then //nếu v chưa thăm  
        DFS(v);
```

End;

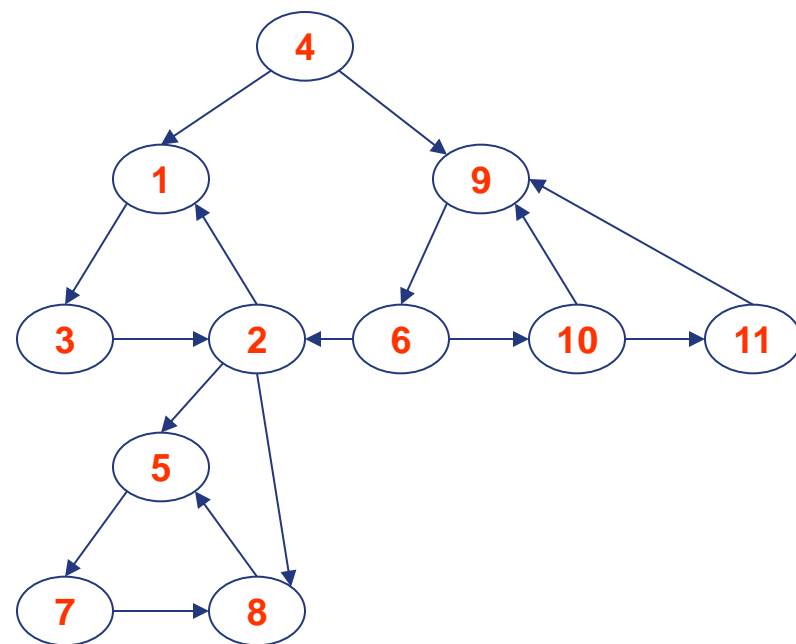
BEGIN

.....

```
  for v:=1 to n do number[v]:=0;  
  count:=0;  
  DFS(4);
```

.....

END.



Dựa theo quá trình duyệt DFS(4) thực hiện:

- Liệt kê các đỉnh và thứ tự thăm.
- Liệt kê các cung (u,v) nối đỉnh u đã thăm đến đỉnh v chưa thăm

CÂY DFS

```

Procedure DFS(u:integer) ;
Var v:integer;
Begin
    //Đánh dấu u đã thăm
    inc(count); number[u]:=count;
    for v:=1 to n do //xét mỗi đỉnh v
        if a[u,v] then //nếu v kề u
            if number[v]=0 then //nếu v chưa thăm
                DFS(v);

```

End;

BEGIN

.....

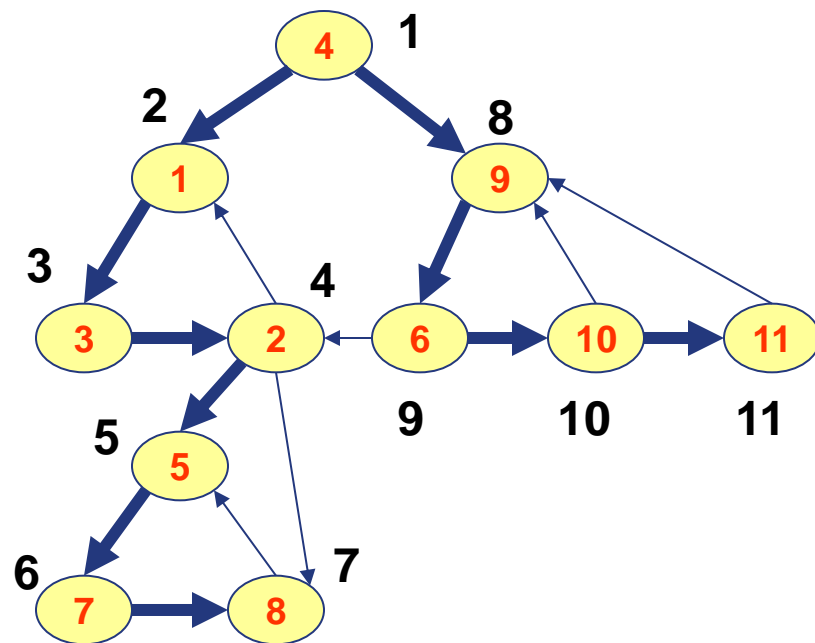
```

for v:=1 to n do number[v]:=0;
count:=0;
DFS(4);

```

.....

END.



Gọi **DFS(4)**,

Các đỉnh và thứ tự thăm:

4, 1, 3, 2, 5, 7, 8, 9, 6, 10, 11

1 2 3 4 5 6 7 8 9 10 11

Các cung (u,v) nối đỉnh u đã thăm đến đỉnh v chưa thăm:

(4,1) (1,3) (3,2) (2,5) (5,7) (7,8) (4,9)

(9,6) (6,10) (10,11)

BA LOẠI CUNG KHÔNG THUỘC CÂY DFS

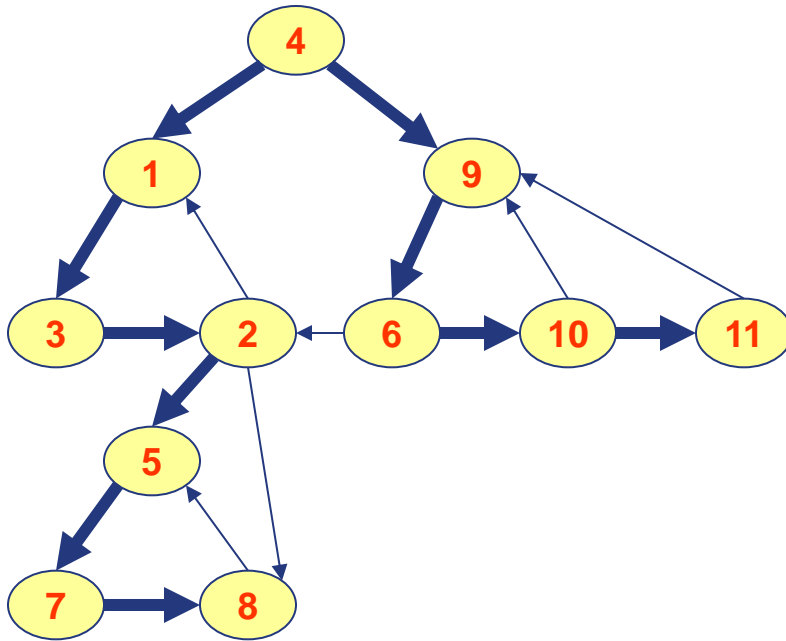
Xét thủ tục DFS(u), thủ tục này xét tất cả những đỉnh v nối từ u ,

- nếu v **chưa thăm** thì gọi DFS(v), cung (u, v) được gọi là **thuộc** cây DFS.
- nếu v **đã thăm** thì cung (u, v) được gọi là **không thuộc** cây DFS.

Các cung (u, v) **không thuộc** cây DFS được chia làm 3 loại:

- Đỉnh u thăm trước đỉnh v , do thủ tục DFS(u) khi thực hiện dây chuyền đệ quy đã gọi tới DFS(v) theo một nhánh khác, sau đó khi quay lui lại, DFS(u) tiếp tục xét tiếp **đỉnh v đã thăm** thì **cung (u, v) được gọi là cung xuôi**.
- Đỉnh u thăm trước đỉnh v , do thủ tục DFS(u) khi thực hiện dây chuyền đệ quy gọi tới DFS(v), sau đó thủ tục DFS(v) tiếp tục **xét tiếp đỉnh u đã thăm** thì **cung (u, v) được gọi là cung ngược**.
- Đỉnh u và đỉnh v thuộc hai nhánh DFS khác nhau, do thủ tục DFS(w) sau khi thực hiện dây chuyền đệ quy gọi DFS(v) theo một nhánh, sau đó quay lui lại, DFS(w) thực hiện dây chuyền đệ quy gọi DFS(u) theo một nhánh khác, và DFS(u) xét tiếp đỉnh v thì **cung (u, v) được gọi là cung chéo**.

BA LOẠI CUNG KHÔNG THUỘC CÂY DFS



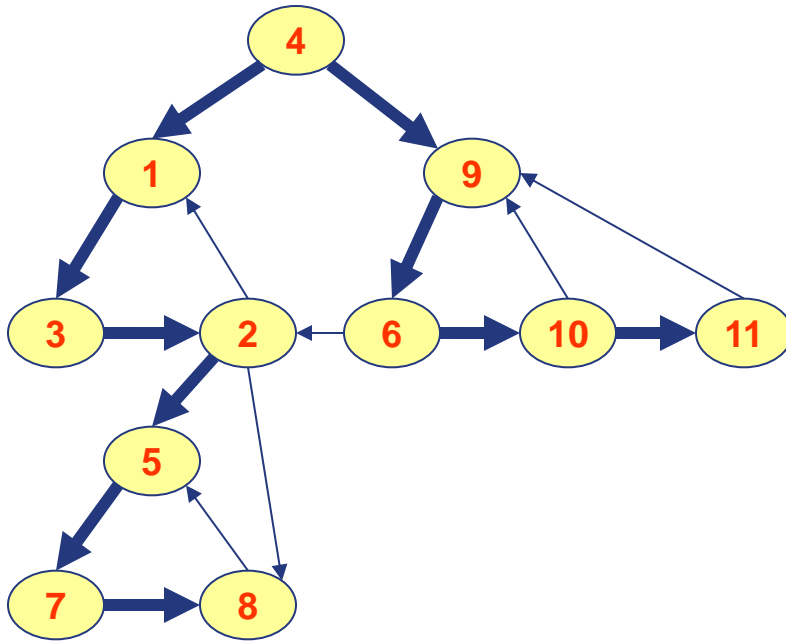
Cung không thuộc cây DFS chia làm 3 loại:

- **Cung xuôi** (u, v) nối từ u đến v với đỉnh u là **tiền bối** của đỉnh v trên cây DFS. (u và v thuộc cùng 1 nhánh DFS)
- **Cung ngược** (u, v) nối từ u đến v với đỉnh u là **hậu duệ** của đỉnh v trên cây DFS. (u và v thuộc cùng 1 nhánh DFS)
- **Cung chéo** (u, v) nối từ u đến v với đỉnh u và đỉnh v thuộc **2 nhánh khác nhau** trên cây DFS.

Khi gọi DFS(4), hãy cho biết:

- Các cung thuộc cây DFS gốc 4
- Các cung không thuộc cây DFS gốc 4 (cung xuôi, cung ngược, cung chéo)

BA LOẠI CUNG KHÔNG THUỘC CÂY DFS



Cung không thuộc cây DFS chia làm 3 loại:

- **Cung xuôi** (u,v) nối từ u đến v với đỉnh u là **tiền bối** của đỉnh v trên cây DFS. (u và v thuộc cùng 1 nhánh DFS)
- **Cung ngược** (u, v) nối từ u đến v với đỉnh u là **hậu duệ** của đỉnh v trên cây DFS. (u và v thuộc cùng 1 nhánh DFS)
- **Cung chéo** (u, v) nối từ u đến v với đỉnh u và đỉnh v thuộc **2 nhánh khác nhau** trên cây DFS.

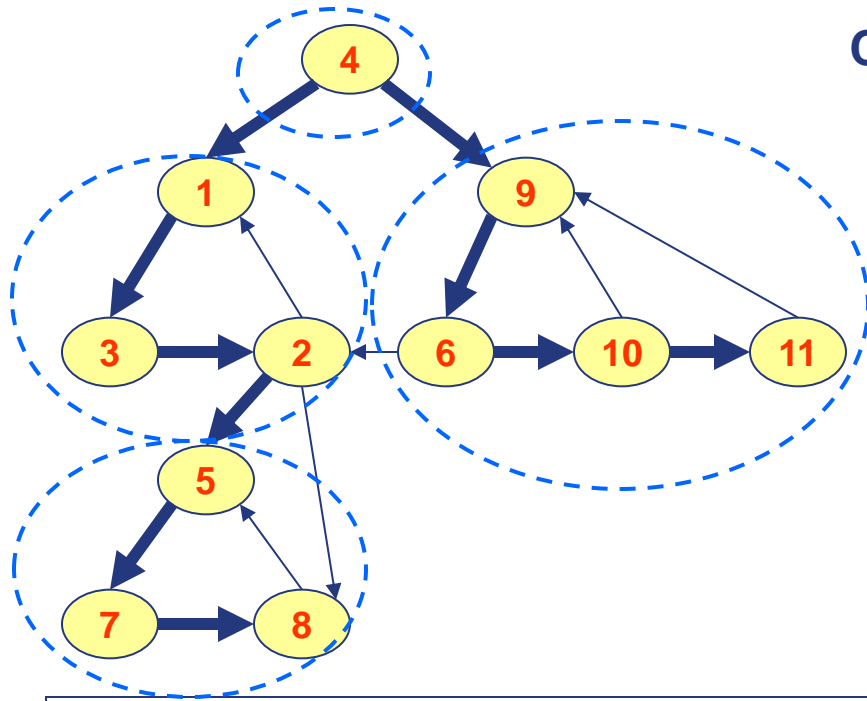
Khi gọi **DFS(4)**

Cung thuộc cây DFS gốc 4 là: $(4,1)$ $(1,3)$ $(3,2)$ $(2,5)$ $(5,7)$ $(7,8)$ $(4,9)$ $(9,6)$ $(6,10)$ $(10,11)$

Cung không thuộc cây DFS gốc 4 là:

- **Cung xuôi:** $(2,8)$
- **Cung ngược:** $(2,1)$ $(8,5)$ $(10,9)$ $(11,9)$
- **Cung chéo:** $(6,2)$

BA LOẠI CUNG KHÔNG THUỘC CÂY DFS



Cung không thuộc cây DFS chia làm 3 loại:

- **Cung xuôi** (u,v) nối từ u đến v với đỉnh u là **tiền bối** của đỉnh v trên cây DFS. (u và v thuộc cùng 1 nhánh DFS)
- **Cung ngược** (u, v) nối từ u đến v với đỉnh u là **hậu duệ** của đỉnh v trên cây DFS. (u và v thuộc cùng 1 nhánh DFS)
- **Cung chéo** (u, v) nối từ u đến v với đỉnh u và đỉnh v thuộc **2 nhánh khác nhau** trên cây DFS.

Khi gọi **DFS(4)**

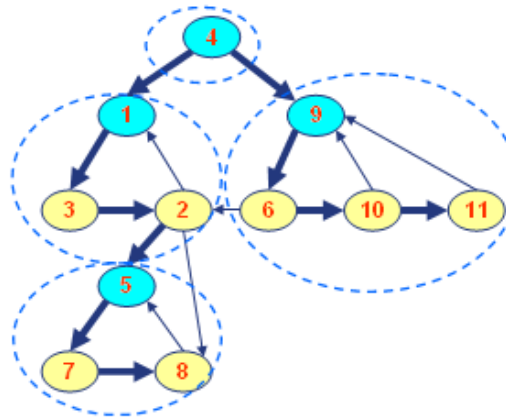
Cung thuộc cây DFS gốc 4 là: (4,1) (1,3) (3,2) (2,5) (5,7) (7,8) (4,9) (9,6) (6,10) (10,11)

Cung không thuộc cây DFS gốc 4 là:

- **Cung xuôi:** (2,8)
- **Cung ngược:** (2,1) (8,5) (10,9) (11,9)
- **Cung chéo:** (6,2)

TÍNH CHẤT

1. Với một thành phần liên thông mạnh C bất kì, luôn **tồn tại duy nhất một đỉnh $r \in C$** sao mọi đỉnh của C đều thuộc nhánh DFS gốc r . Đỉnh r được gọi là **chốt** của thành phần liên thông mạnh C .



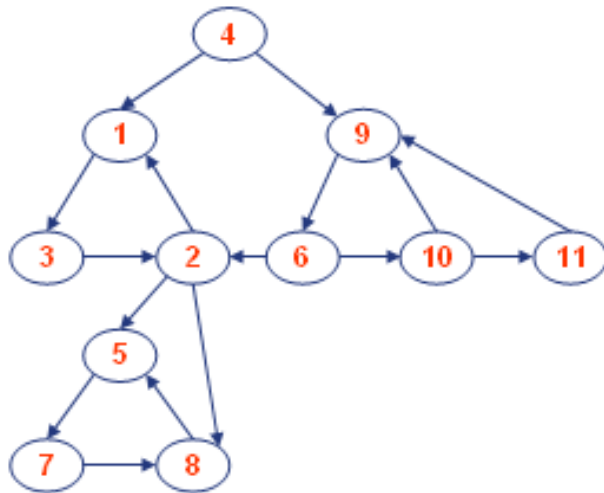
2. Đỉnh r là **chốt** nếu và chỉ nếu không tồn tại cung nối từ một đỉnh thuộc nhánh DFS gốc r tới một đỉnh thăm trước r .
3. Luôn tồn tại chốt r thỏa điều kiện, quá trình tìm kiếm theo chiều sâu bắt đầu từ r không thăm được bất kì một chốt nào khác. Khi đó các đỉnh thuộc nhánh DFS gốc r chính là các đỉnh thuộc thành phần liên thông mạnh chứa r .

BÀI TOÁN LIỆT KÊ CÁC THÀNH PHẦN LIÊN THÔNG MẠNH

Input: file văn bản SCONNECT.INP:

- ❖ Dòng đầu: Ghi số đỉnh n (≤ 100) và số cung m của đồ thị cách nhau một dấu cách
- ❖ m dòng tiếp theo, mỗi dòng ghi hai số nguyên u, v cách nhau một dấu cách thể hiện có cung (u, v) trong đồ thị

Output: file văn bản SCONNECT.OUT, liệt kê các thành phần liên thông mạnh

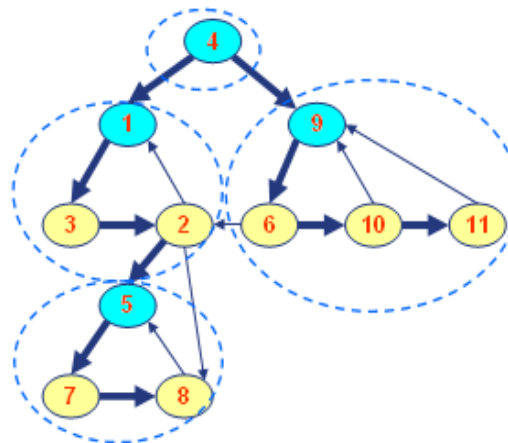


CONNECT.INP	CONNECT.OUT
11 16 4 1 1 3 3 2 2 1 2 5 2 8 5 7 7 8 8 5 4 9 9 6 6 2 6 10 10 9 10 11 11 9	Component 1: 8, 7, 5, Component 2: 2, 3, 1, Component 3: 11, 10, 6, 9, Component 4: 4,

THUẬT TOÁN TARJAN

Ý tưởng: Sử dụng ba tính chất 1, 2, 3.

Chọn u là chốt mà từ đó quá trình tìm kiếm theo chiều sâu không thăm thêm bất kỳ một chốt nào khác, chọn lấy thành phần liên thông mạnh thứ nhất là nhánh DFS gốc u . Sau đó loại bỏ nhánh DFS gốc u ra khỏi cây DFS, lại tìm thấy một đỉnh chốt v khác mà nhánh DFS gốc v không chứa chốt nào khác, lại chọn lấy thành phần liên thông mạnh thứ hai là nhánh DFS gốc ... Có thể hình dung thuật toán Tarjan “bẻ” cây DFS tại vị trí các chốt để được các nhánh rời rạc, mỗi nhánh là một thành phần liên thông mạnh



THUẬT TOÁN TARJAN

//Visit(u) tương tự DFS(u), trong đó có bổ sung thêm phần xác định chót và xác định thành phần liên thông mạnh

procedure Visit(u);

Begin

 «đánh dấu u đã thăm»

for với mọi đỉnh v kề u **do**

 if «v chưa thăm» then Visit(v);

if «u là chót» **then**

Begin

 «Liệt kê thành phần liên thông mạnh tương ứng với chót u»

 «Loại bỏ các đỉnh đã liệt kê khỏi đồ thị và cây DFS»

End;

End;

BEGIN

 «đánh dấu mọi đỉnh đều chưa thăm»

for mọi đỉnh v **do**

if «v chưa thăm» **then** Visit(v);

END.

THUẬT TOÁN TARJAN

CÀI ĐẶT

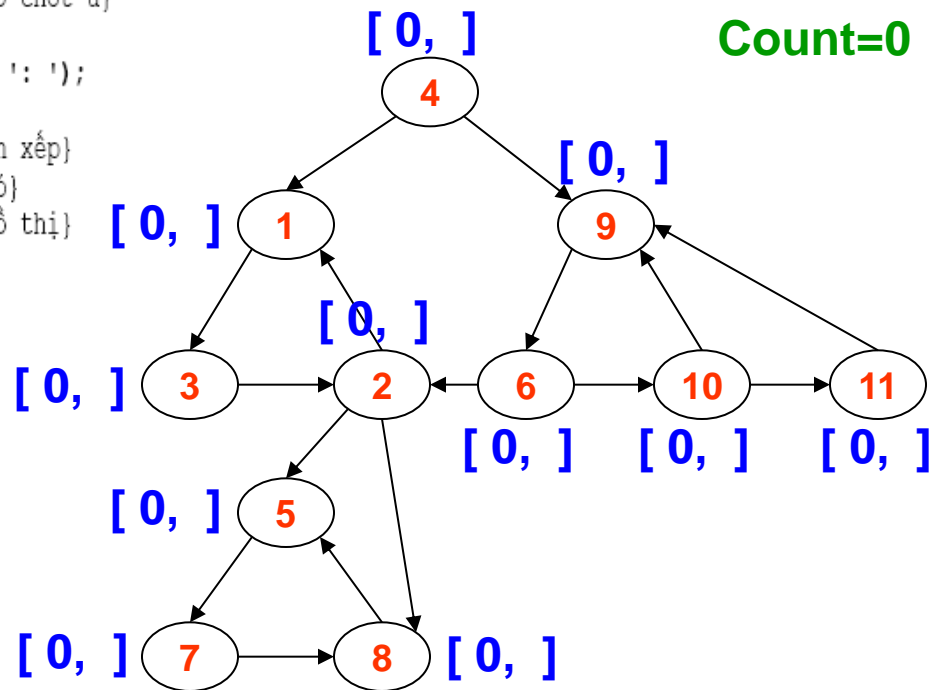
- ❖ Mảng Number đánh số thứ tự các đỉnh khi duyệt bằng DFS. Đồng thời cũng cho biết đỉnh u đã thăm hay chưa. $\text{Number}[u] \neq 0$ tương ứng đỉnh u đã thăm.
- ❖ Mảng Free cho biết một đỉnh đã bị loại ra khỏi đồ thị hay chưa. $\text{Free}[u] = \text{False}$ nghĩa là đỉnh u đã bị loại ra khỏi đồ thị.
- ❖ Mảng Low dùng để xác định chốt của thành phần liên thông mạnh. Xét thành phần liên thông mạnh C gồm có các đỉnh v_1, v_2, \dots, v_k và có chốt là t . Khi đó $\text{Low}[v_1] = \text{Low}[v_2] = \dots = \text{Low}[v_k] = \text{Number}[t]$
- ❖ Như vậy, nếu t là chốt thì ta có $\text{Low}[t] = \text{Number}[t]$

```

procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ', '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rồi loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;

procedure Solve;
var
  u: Integer;
begin
  for u := 1 to n do
    if Number[u] = 0 then Visit(u);
end;

```



```

procedure Visit(u: Integer); ← toán tìm kiếm theo chiều sâu bắt đầu từ u
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ', '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rồi loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;

```

```

end;
procedure Solve;

```

```

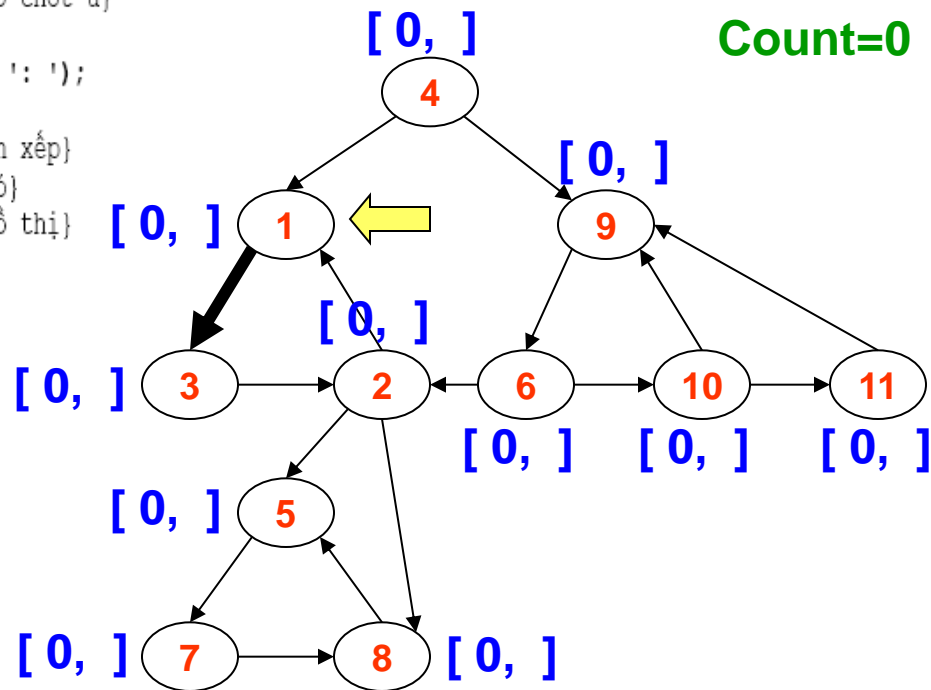
var
  u: Integer;
begin

```

```

  for u := 1 to n do
    if Number[u] = 0 then Visit(u);
  end;

```



```

procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cùng tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ', '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rồi loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;

```

```

end;
procedure Solve;

```

```

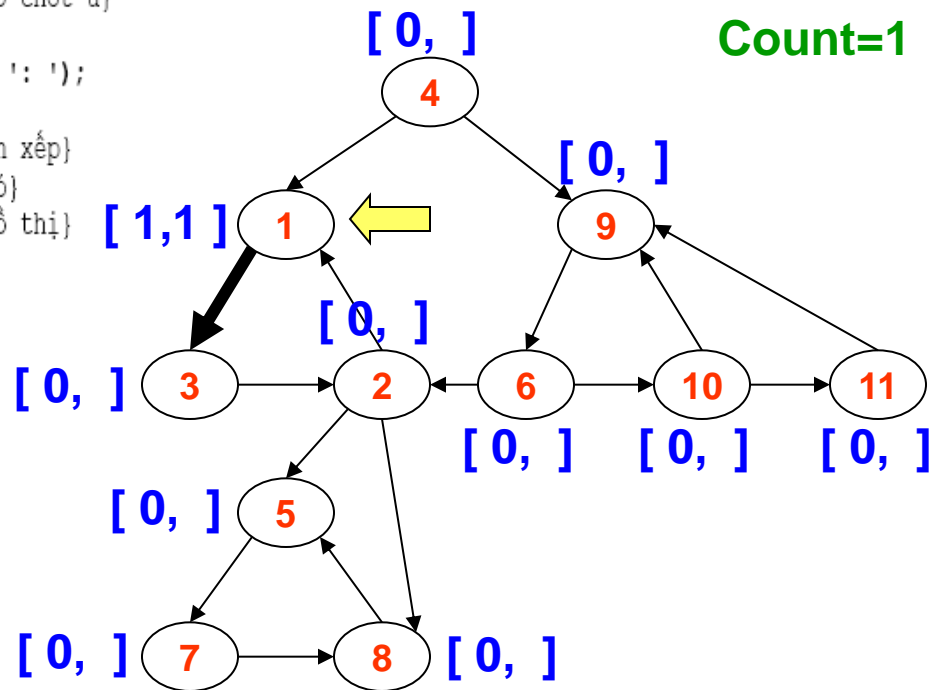
var
  u: Integer;
begin

```

```

  for u := 1 to n do
    if Number[u] = 0 then Visit(u);
  end;

```



Count=1

1
STACK

```

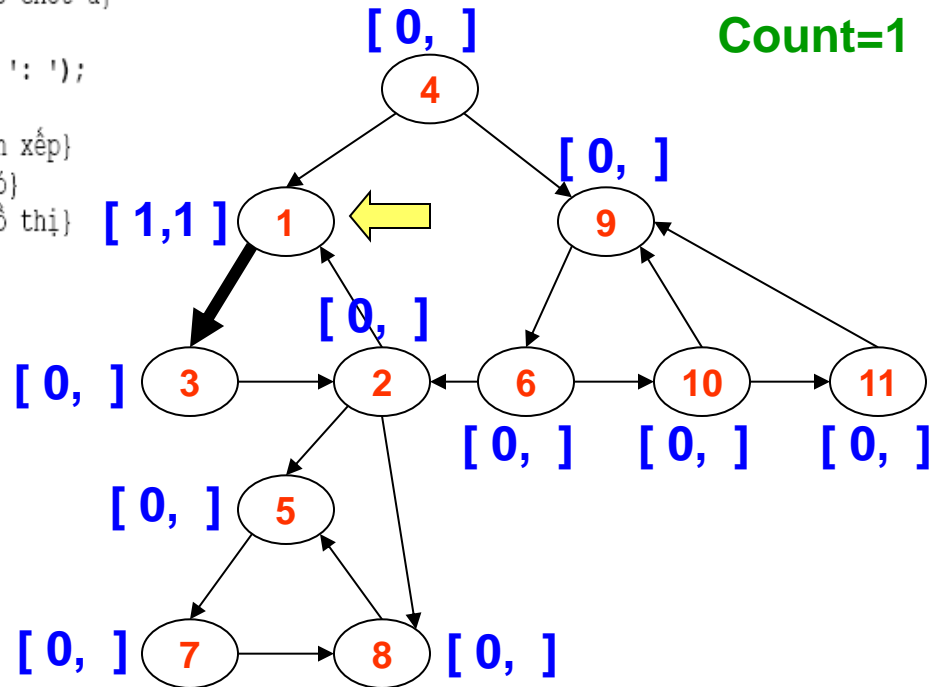
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ', '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rồi loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;

```

```


procedure Solve;
var
  u: Integer;
begin
  for u := 1 to n do
    if Number[u] = 0 then Visit(u);
  end;
end;

```



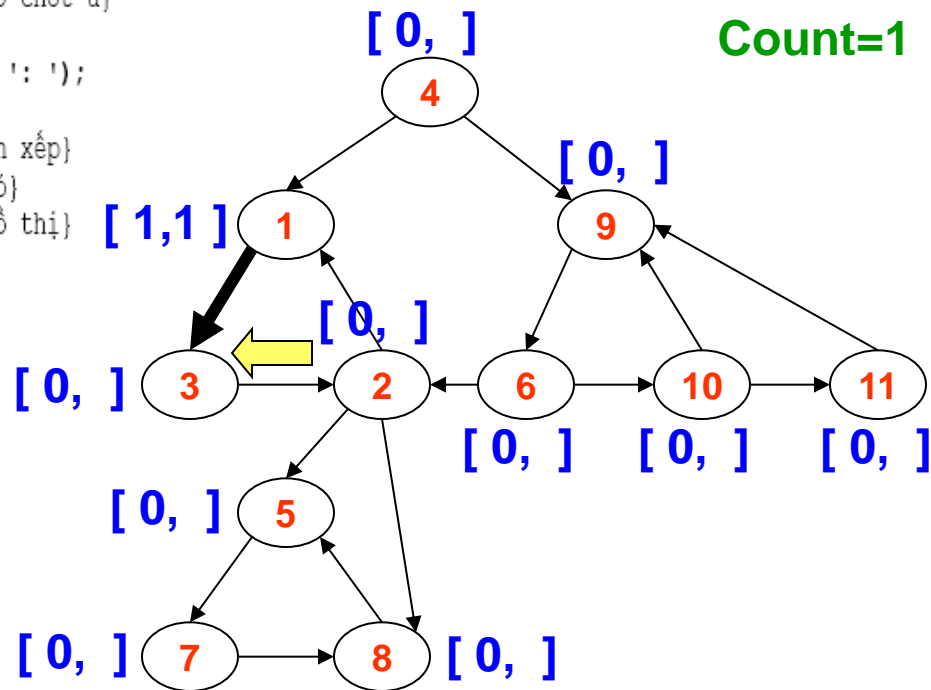
1
STACK


```

procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v);  {tục tìm kiếm theo chiều sâu bắt đầu từ v}
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ', '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rồi loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;

procedure Solve;
var
  u: Integer;
begin
  for u := 1 to n do
    if Number[u] = 0 then Visit(u);
end;

```



Count=1

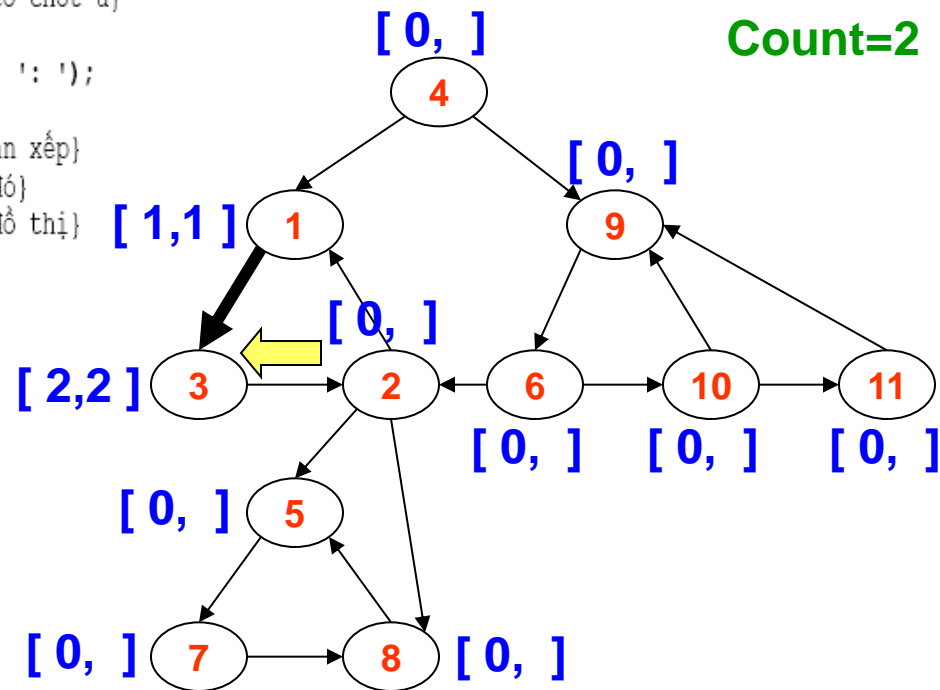
1
STACK

```

procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ', '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rồi loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;

procedure Solve;
var
  u: Integer;
begin
  for u := 1 to n do
    if Number[u] = 0 then Visit(u);
end;

```



Count=2

3
1

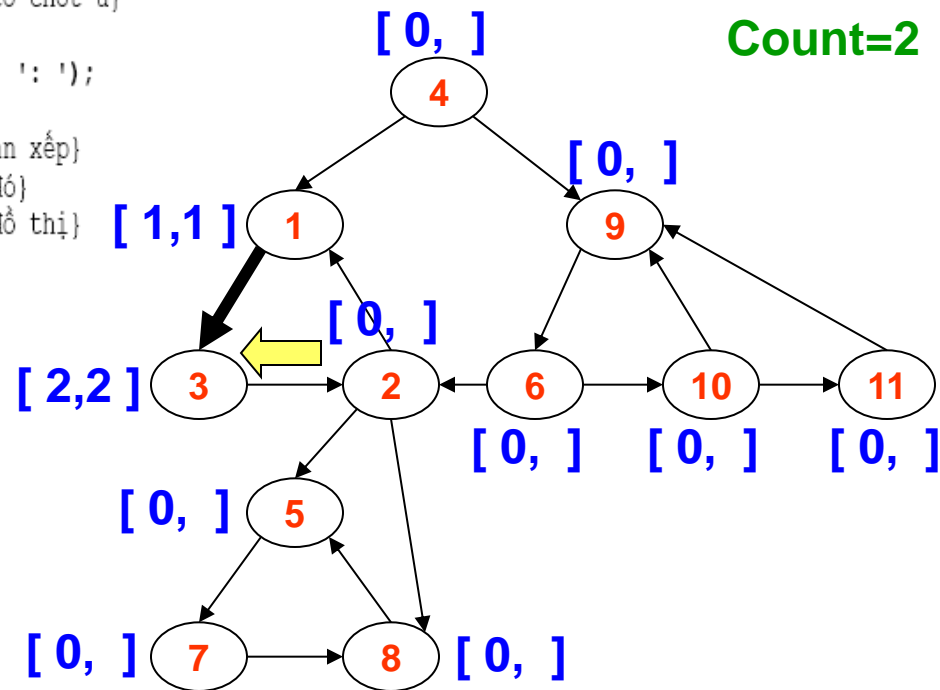
STACK

```

procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ' '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rồi loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;

procedure Solve;
var
  u: Integer;
begin
  for u := 1 to n do
    if Number[u] = 0 then Visit(u);
end;

```



Count=2

3
1

STACK

```

procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); ← tục tìm kiếm theo chiều sâu bắt đầu từ v
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ', '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rồi loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;

```

{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}

if Number[u] = Low[u] then {Nếu u là chốt}

begin {Liệt kê thành phần liên thông mạnh có chốt u}

Inc(ComponentCount);

WriteLn(fo, 'Component ', ComponentCount, ': ');

repeat

v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}

Write(fo, v, ', '); {Liệt kê các đỉnh đó}

Free[v] := False; {Rồi loại luôn khỏi đồ thị}

until v = u; {Cho tới khi lấy tới đỉnh u}

WriteLn(fo);

end;

end;
procedure Solve;

var

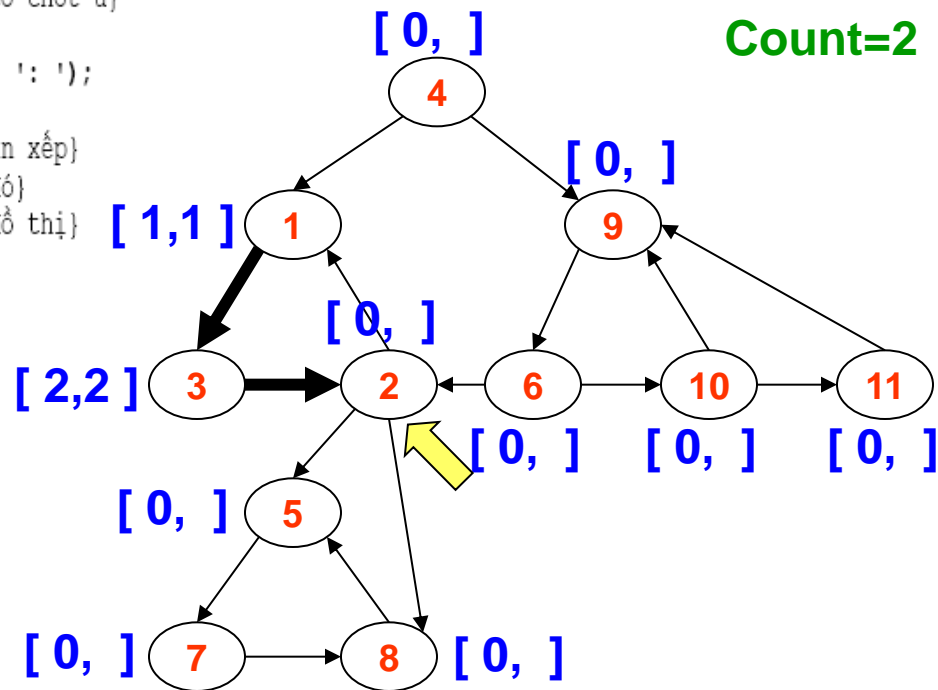
u: Integer;

begin

for u := 1 to n do

if Number[u] = 0 then Visit(u);

end;



Count=2

3
1

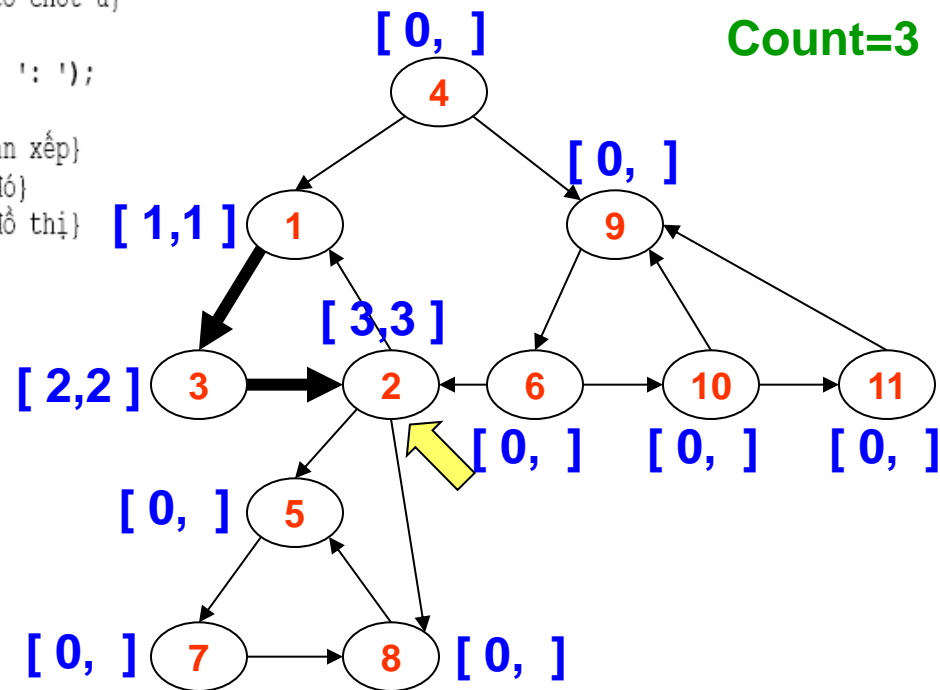
STACK

```

procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ' '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rồi loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;

procedure Solve;
var
  u: Integer;
begin
  for u := 1 to n do
    if Number[u] = 0 then Visit(u);
end;

```



2
3
1

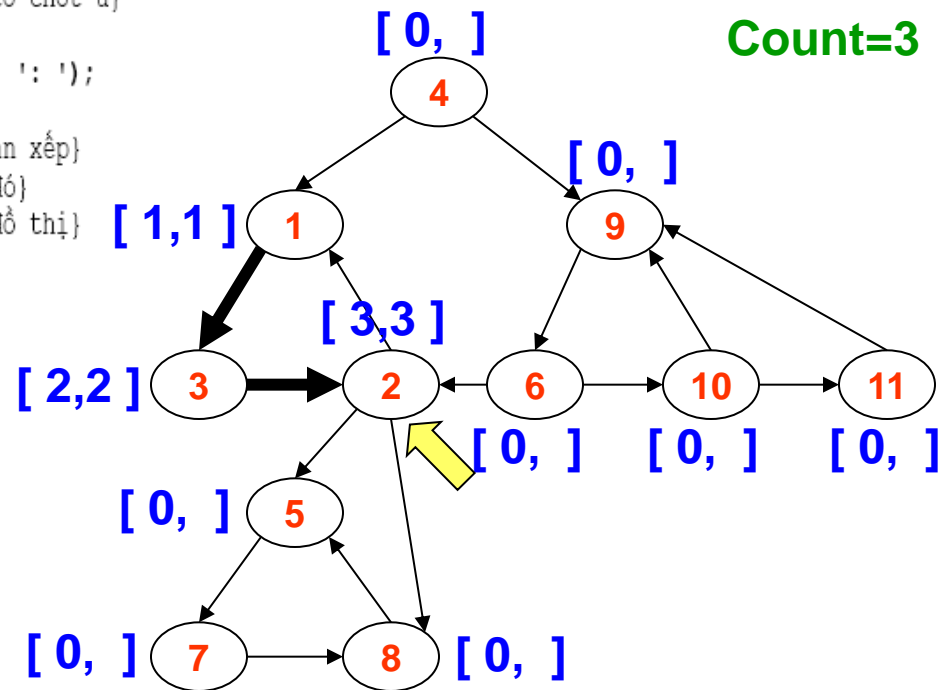
STACK

```

procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ', '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rồi loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;

procedure Solve;
var
  u: Integer;
begin
  for u := 1 to n do
    if Number[u] = 0 then Visit(u);
end;

```



2
3
1

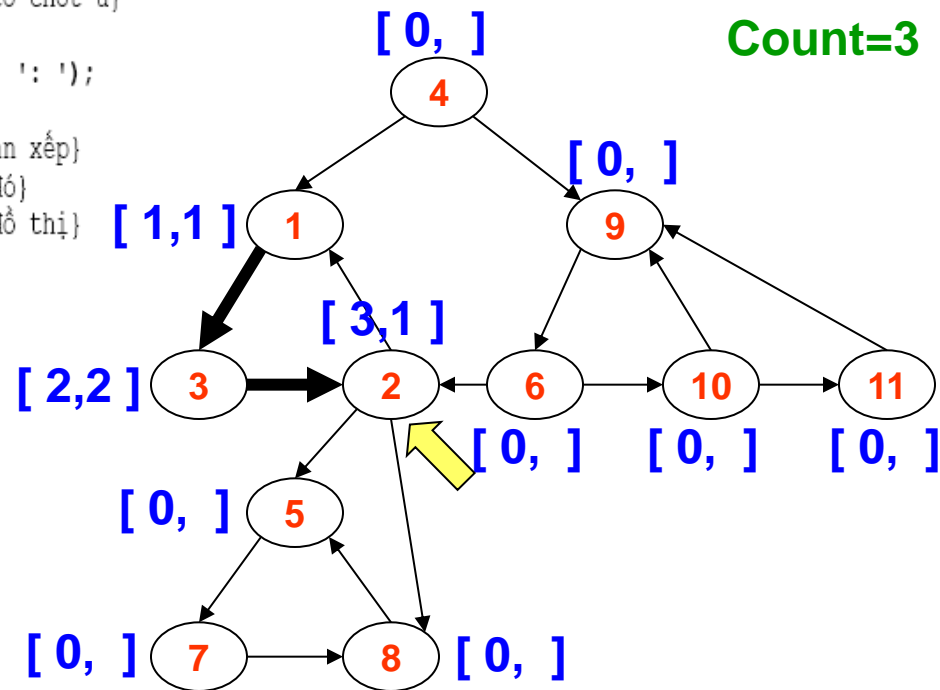
STACK

```

procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) ← tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ', '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rồi loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;

procedure Solve;
var
  u: Integer;
begin
  for u := 1 to n do
    if Number[u] = 0 then Visit(u);
end;

```



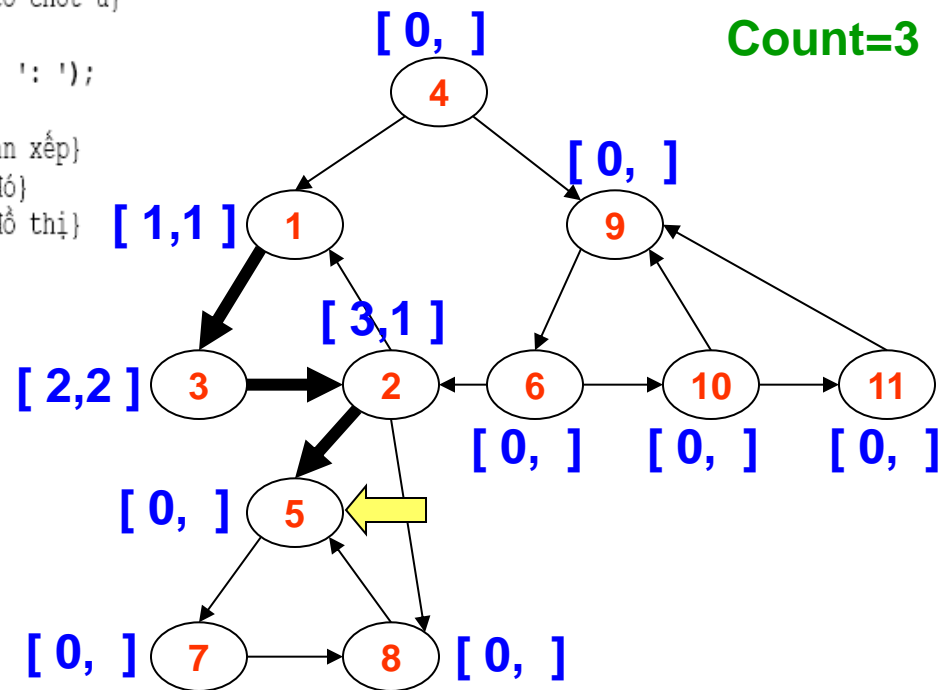
2
3
1

STACK


```

procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); ← tục tìm kiếm theo chiều sâu bắt đầu từ v
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ', '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rồi loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;
procedure Solve;
var
  u: Integer;
begin
  for u := 1 to n do
    if Number[u] = 0 then Visit(u);
end;

```



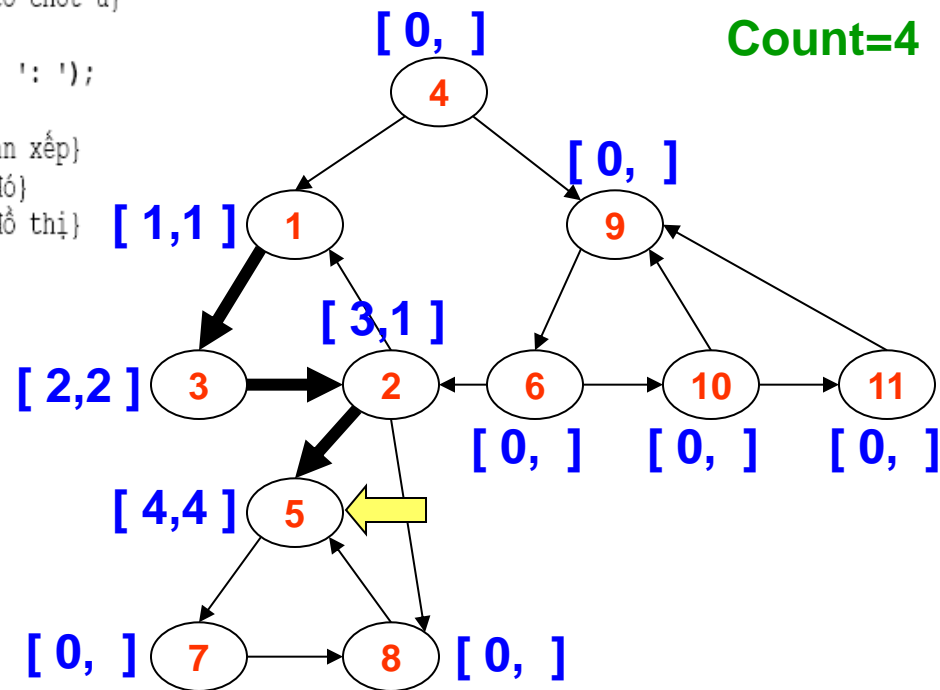
2
3
1

STACK


```

procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ', '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rồi loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;
procedure Solve;
var
  u: Integer;
begin
  for u := 1 to n do
    if Number[u] = 0 then Visit(u);
end;

```



5
2
3
1

STACK

```

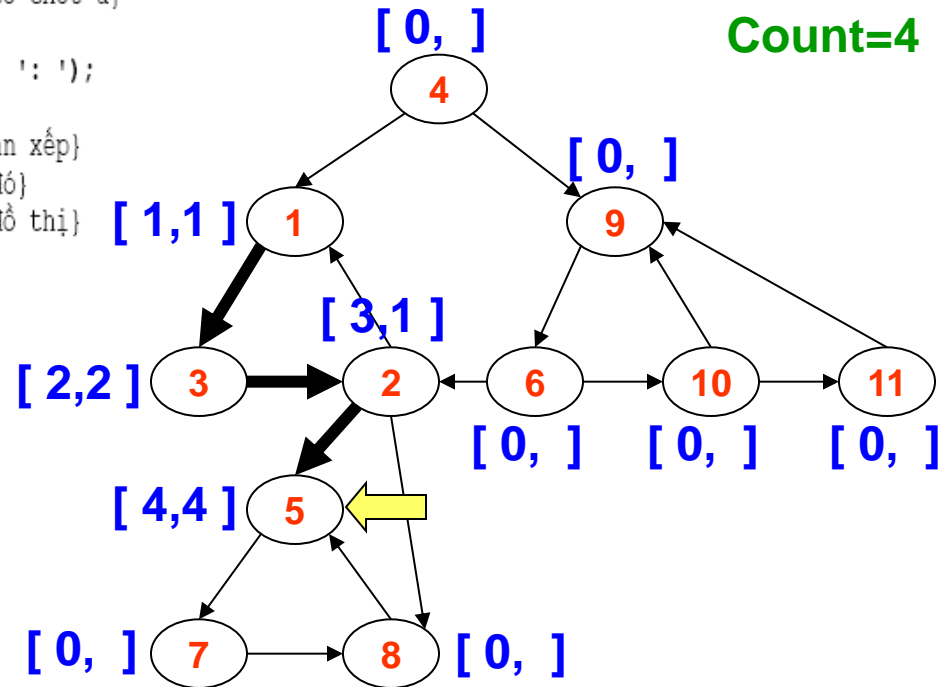
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ', '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rồi loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;

```

```

procedure Solve;
var
  u: Integer;
begin
  for u := 1 to n do
    if Number[u] = 0 then Visit(u);
  end;
end;

```



5
2
3
1

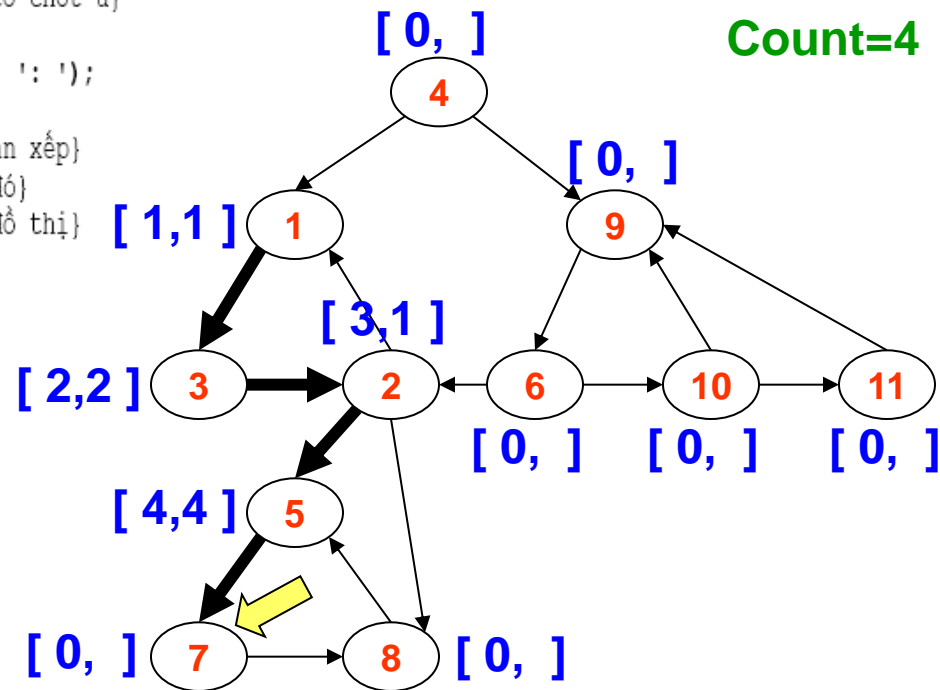
STACK

```

procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); ← tục tìm kiếm theo chiều sâu bắt đầu từ v
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ', '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rồi loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;

procedure Solve;
var
  u: Integer;
begin
  for u := 1 to n do
    if Number[u] = 0 then Visit(u);
end;

```



5
2
3
1

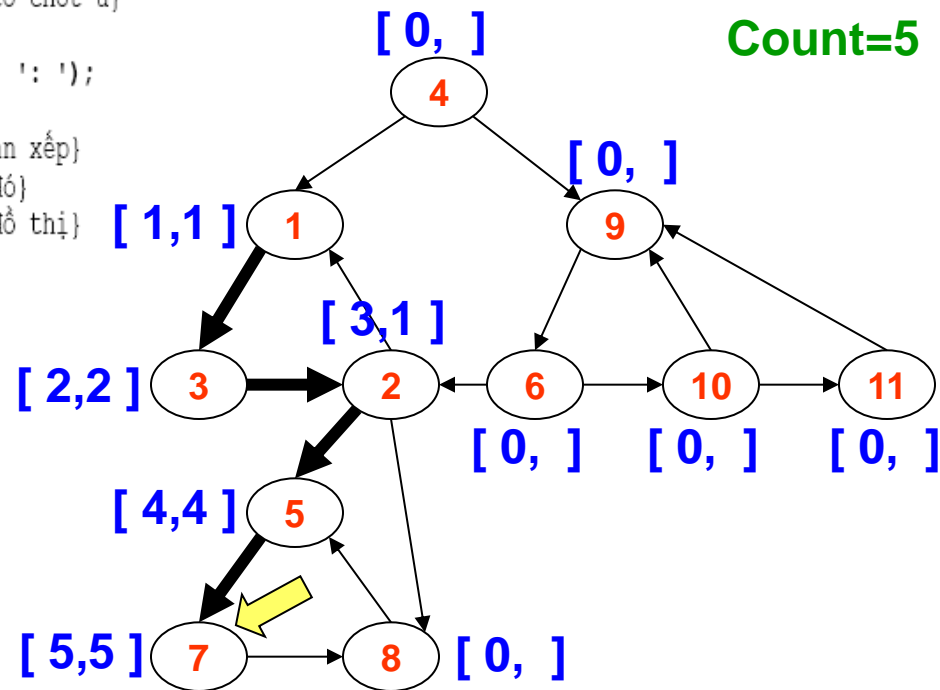
STACK

```

procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ', '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rồi loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;

procedure Solve;
var
  u: Integer;
begin
  for u := 1 to n do
    if Number[u] = 0 then Visit(u);
end;

```



7
5
2
3
1

STACK

```

procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ', '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rồi loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;

```

{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}

if Number[u] = Low[u] then {Nếu u là chốt}

begin {Liệt kê thành phần liên thông mạnh có chốt u}

Inc(ComponentCount);

WriteLn(fo, 'Component ', ComponentCount, ': ');

repeat

v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}

Write(fo, v, ', '); {Liệt kê các đỉnh đó}

Free[v] := False; {Rồi loại luôn khỏi đồ thị}

until v = u; {Cho tới khi lấy tới đỉnh u}

WriteLn(fo);

end;

```

end;
procedure Solve;
var

```

u: Integer;

```

begin

```

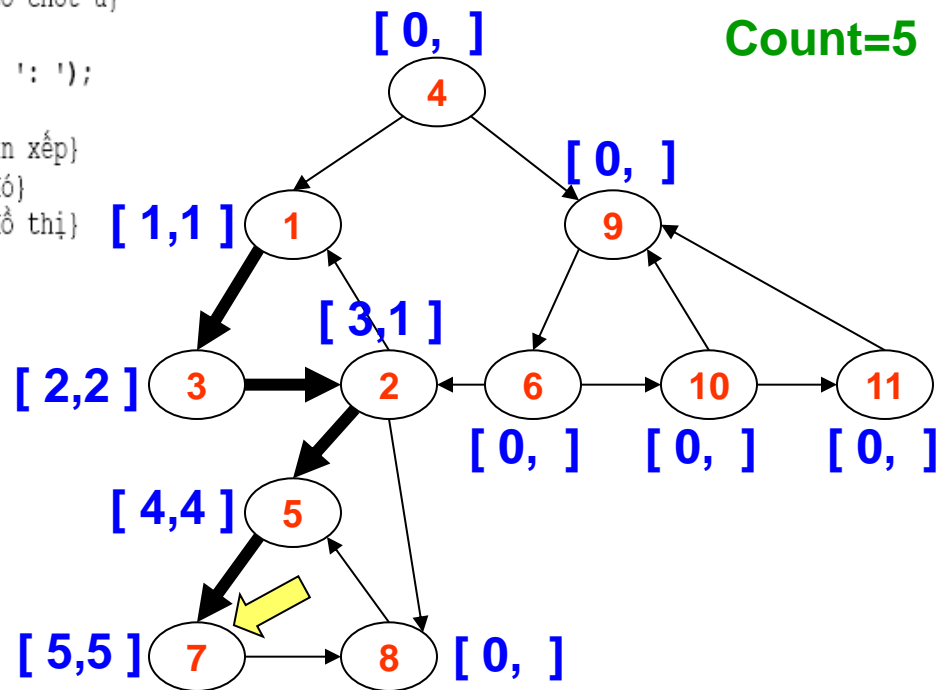
for u := 1 to n do

if Number[u] = 0 then Visit(u);

```

end;

```



7
5
2
3
1

STACK

```

procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); ← tục tìm kiếm theo chiều sâu bắt đầu từ v
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ', '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rồi loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;

```

{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}

if Number[u] = Low[u] then {Nếu u là chốt}

begin {Liệt kê thành phần liên thông mạnh có chốt u}

Inc(ComponentCount);

WriteLn(fo, 'Component ', ComponentCount, ': ');

repeat

v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}

Write(fo, v, ', '); {Liệt kê các đỉnh đó}

Free[v] := False; {Rồi loại luôn khỏi đồ thị}

until v = u; {Cho tới khi lấy tới đỉnh u}

WriteLn(fo);

end;

end;
procedure Solve;

var

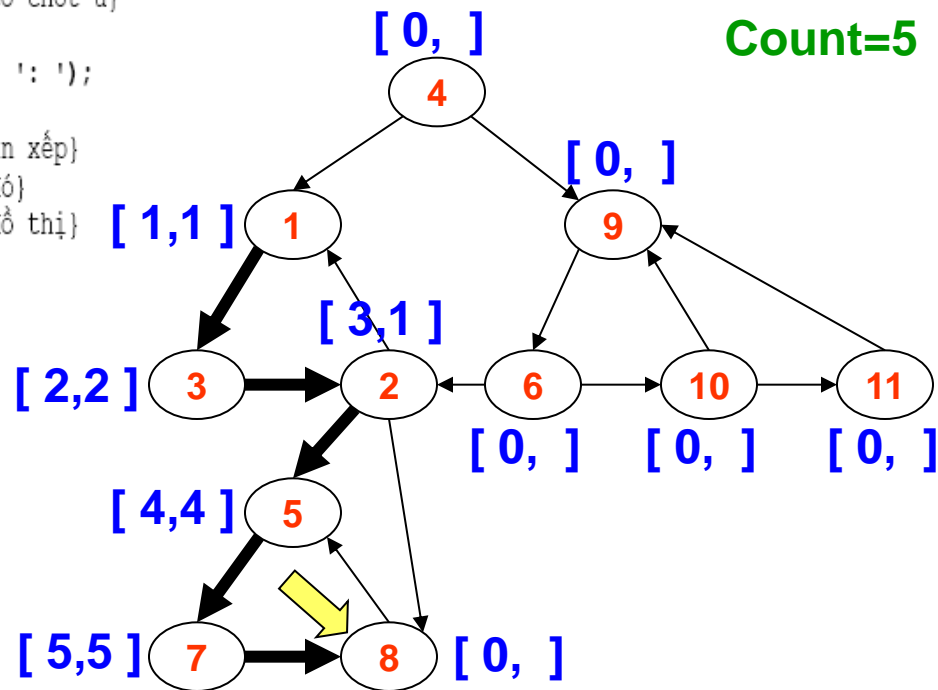
u: Integer;

begin

for u := 1 to n do

if Number[u] = 0 then Visit(u);


end;



7
5
2
3
1

STACK

```

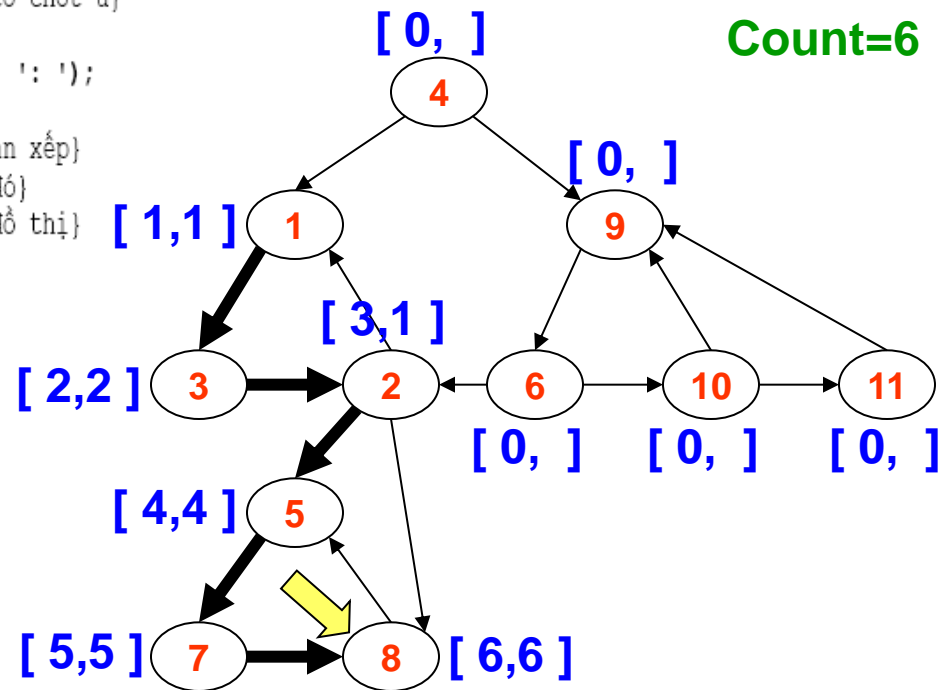
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u và  xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ', '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rồi loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;

```

```

procedure Solve;
var
  u: Integer;
begin
  for u := 1 to n do
    if Number[u] = 0 then Visit(u);
  end;
end;

```



8
7
5
2
3
1

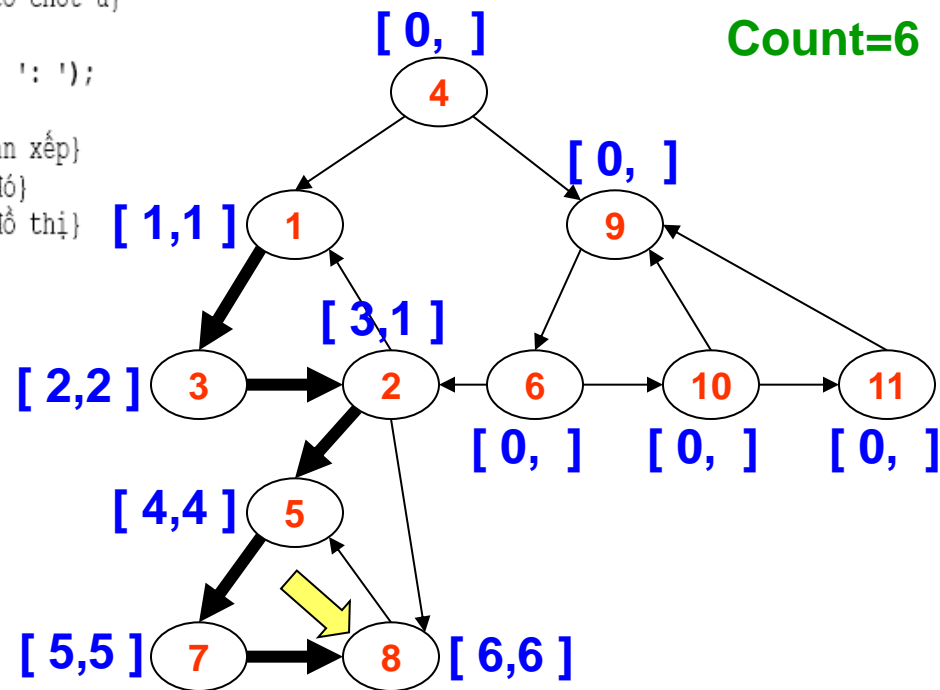
STACK


```

procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ', '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rồi loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;

procedure Solve;
var
  u: Integer;
begin
  for u := 1 to n do
    if Number[u] = 0 then Visit(u);
end;

```



8
7
5
2
3
1

STACK


```

procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) ← {liều hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ', '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rồi loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;

```

```

end;
procedure Solve;
var

```

```

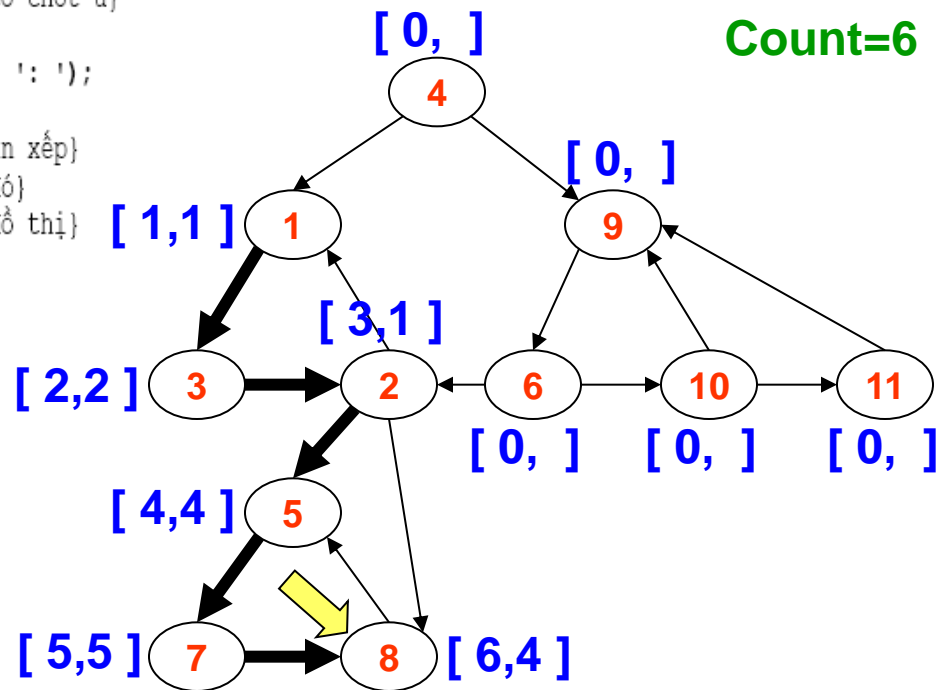
  u: Integer;
begin

```

```

  for u := 1 to n do
    if Number[u] = 0 then Visit(u);
  end;

```



8
7
5
2
3
1

STACK

```

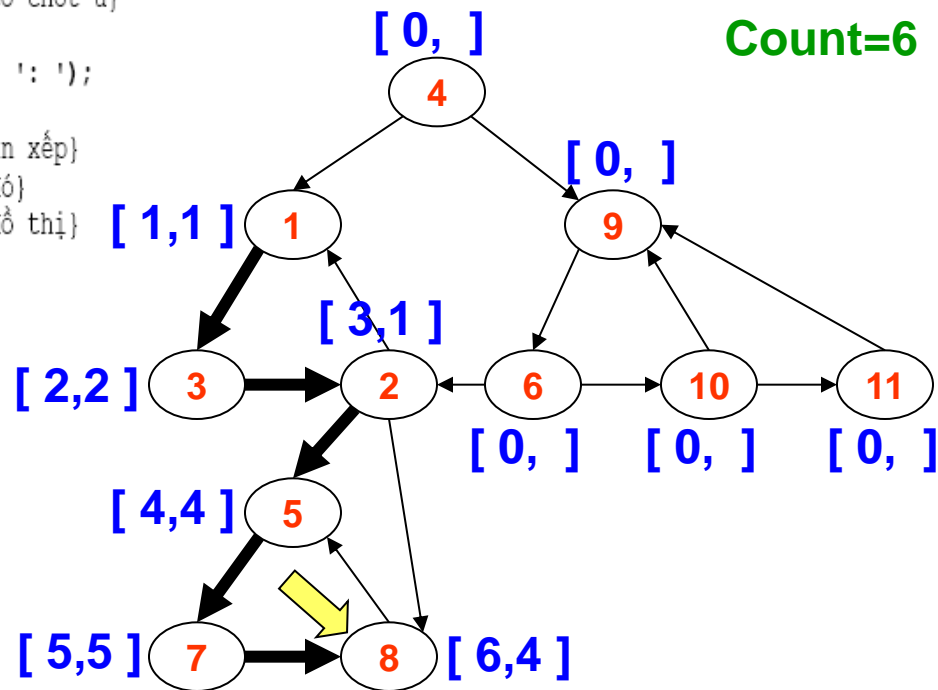
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then ← là chốt}
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
    Inc(ComponentCount);
    WriteLn(fo, 'Component ', ComponentCount, ': ');
    repeat
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
    until v = u; {Cho tới khi lấy tới đỉnh u}
    WriteLn(fo);
  end;
end;

```

```

procedure Solve;
var
  u: Integer;
begin
  for u := 1 to n do
    if Number[u] = 0 then Visit(u);
  end;
end;

```



8
7
5
2
3
1

STACK

```

procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
          Low[u] := Min(Low[u], Low[v]); ← cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ', '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rời loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;

```

{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}

if Number[u] = Low[u] then {Nếu u là chốt}

begin {Liệt kê thành phần liên thông mạnh có chốt u}

Inc(ComponentCount);

WriteLn(fo, 'Component ', ComponentCount, ': ');

repeat

v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}

Write(fo, v, ', '); {Liệt kê các đỉnh đó}

Free[v] := False; {Rời loại luôn khỏi đồ thị}

until v = u; {Cho tới khi lấy tới đỉnh u}

WriteLn(fo);

end;

```

end;
procedure Solve;
var

```

u: Integer;

```

begin

```

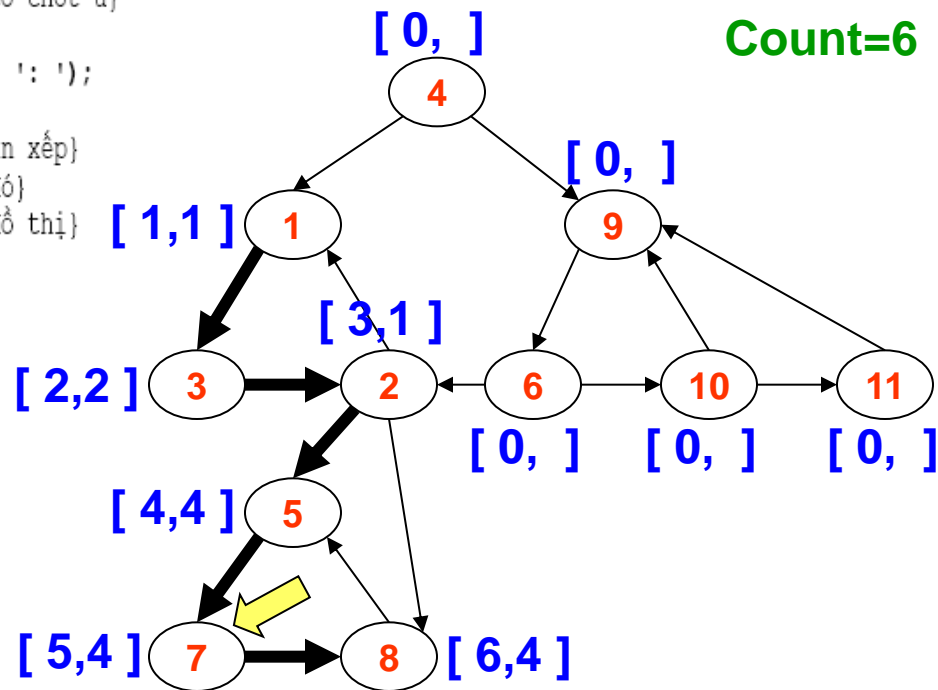
for u := 1 to n do

if Number[u] = 0 then Visit(u);

```

end;

```



Count=6

8
7
5
2
3
1

STACK

```

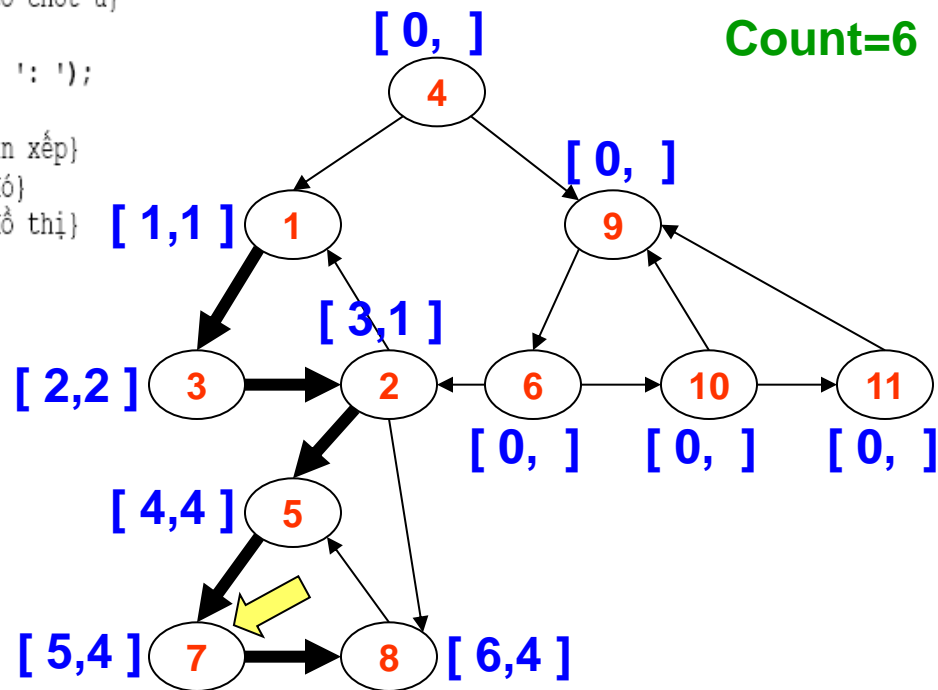
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ', '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rồi loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;

```

```

procedure Solve;
var
  u: Integer;
begin
  for u := 1 to n do
    if Number[u] = 0 then Visit(u);
  end;
end;

```



8
7
5
2
3
1

STACK

```

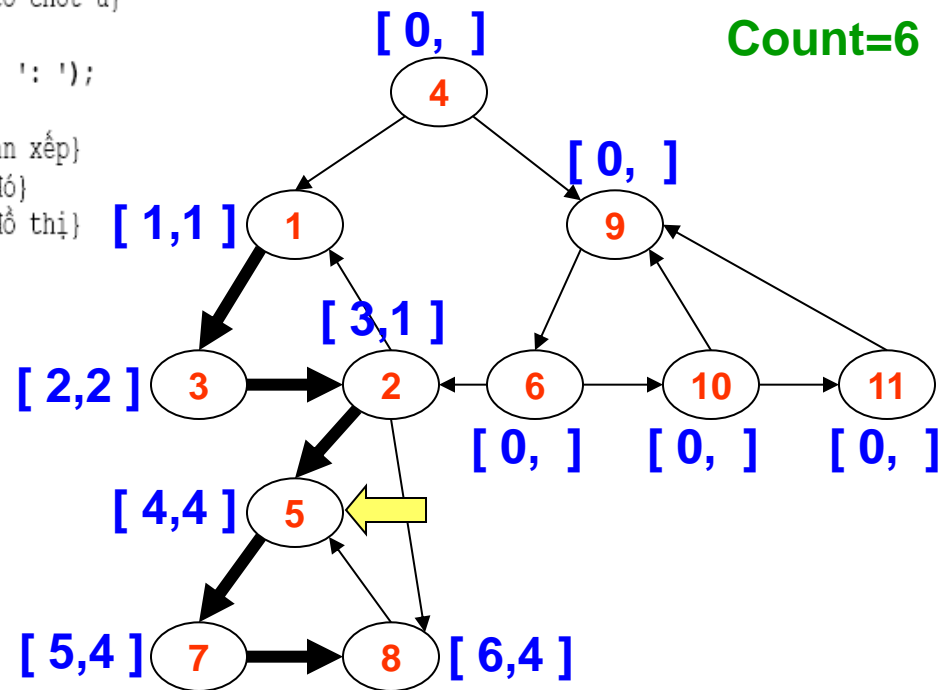
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
          Low[u] := Min(Low[u], Low[v]); ← cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ', '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rời loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;

```

```

procedure Solve;
var
  u: Integer;
begin
  for u := 1 to n do
    if Number[u] = 0 then Visit(u);
  end;
end;

```



8
7
5
2
3
1

STACK

```

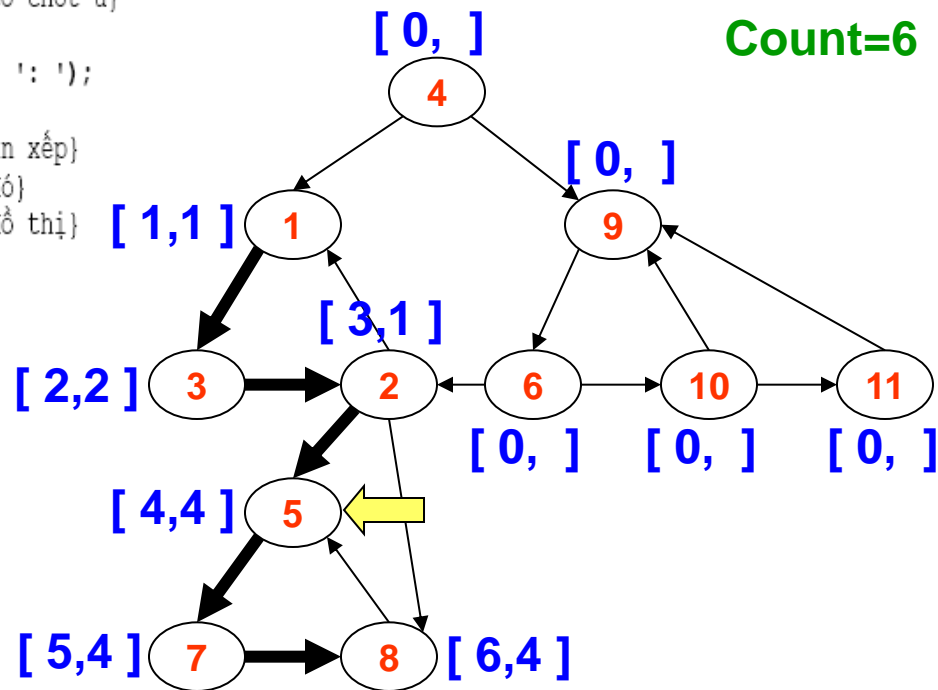
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ', '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rồi loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;

```

```

procedure Solve;
var
  u: Integer;
begin
  for u := 1 to n do
    if Number[u] = 0 then Visit(u);
  end;
end;

```



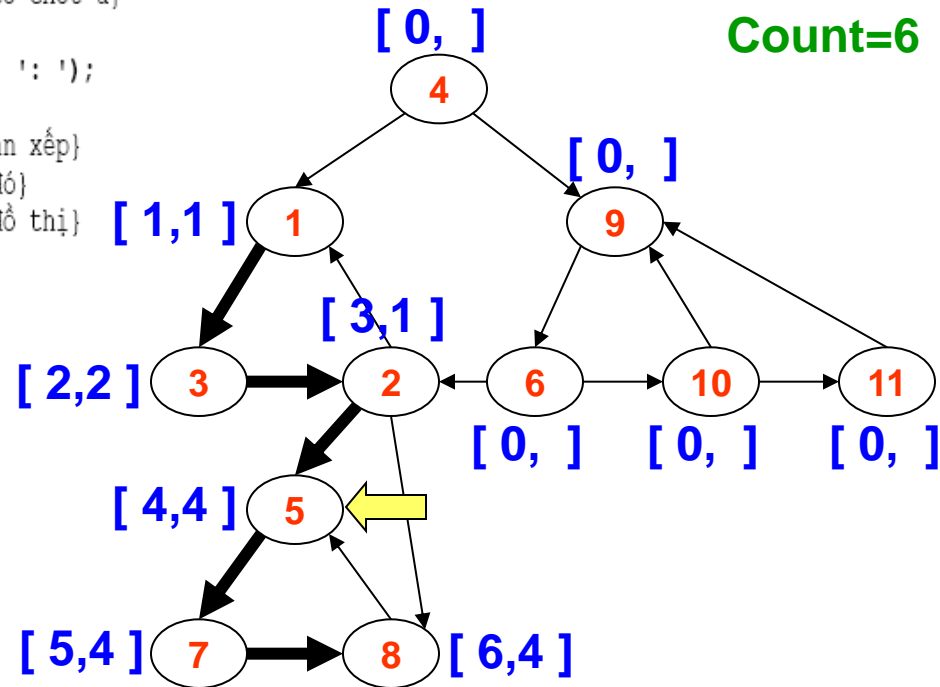
8
7
5
2
3
1

STACK

Component 1:

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ', '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rồi loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;
```

```
procedure Solve;
var
  u: Integer;
begin
  for u := 1 to n do
    if Number[u] = 0 then Visit(u);
  end;
```



Count=6

8
7
5
2
3
1

STACK


```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cùng tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat ←
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
procedure Solve;
```

```
var
```

```
  u: Integer;
```

```
begin
```

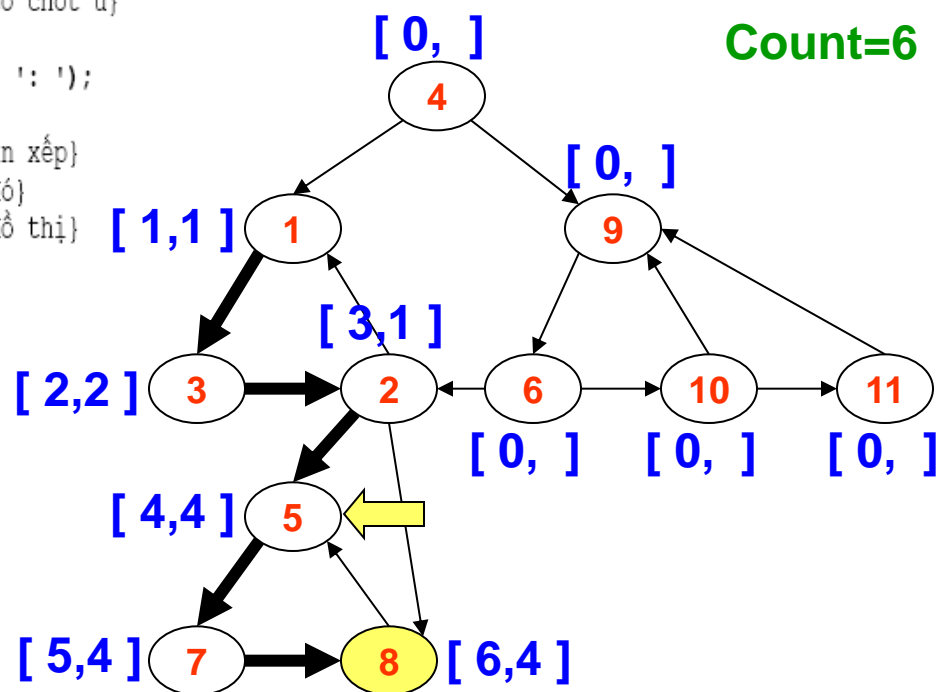
```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

```
end;
```

Component 1:

8



7
5
2
3
1

STACK


```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
procedure Solve;
```

```
var
```

```
  u: Integer;
```

```
begin
```

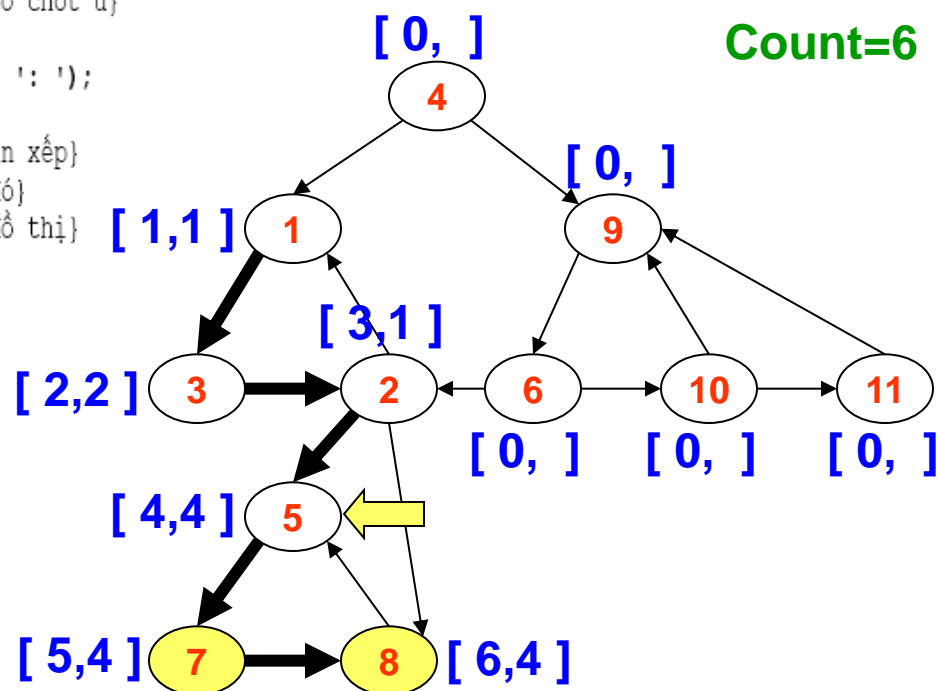
```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

```
end;
```

Component 1:

8 7



Count=6

5
2
3
1

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chót}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chót u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat ←
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;  
procedure Solve;
```

```
var
```

```
  u: Integer;
```

```
begin
```

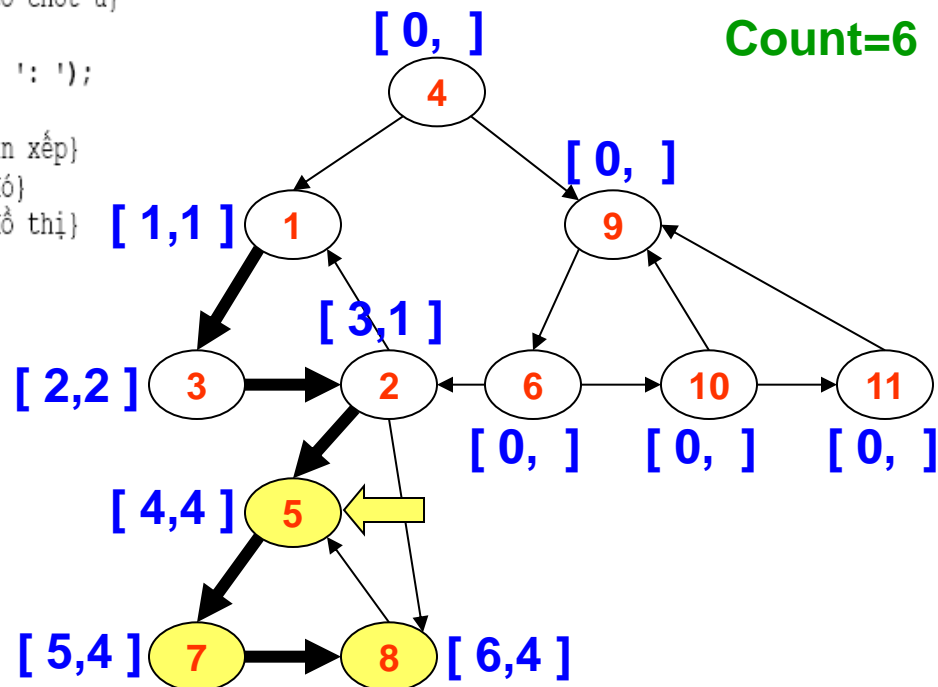
```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

```
end;
```

Component 1:

8 7 5



Count=6

2
3
1

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); ← cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rời loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

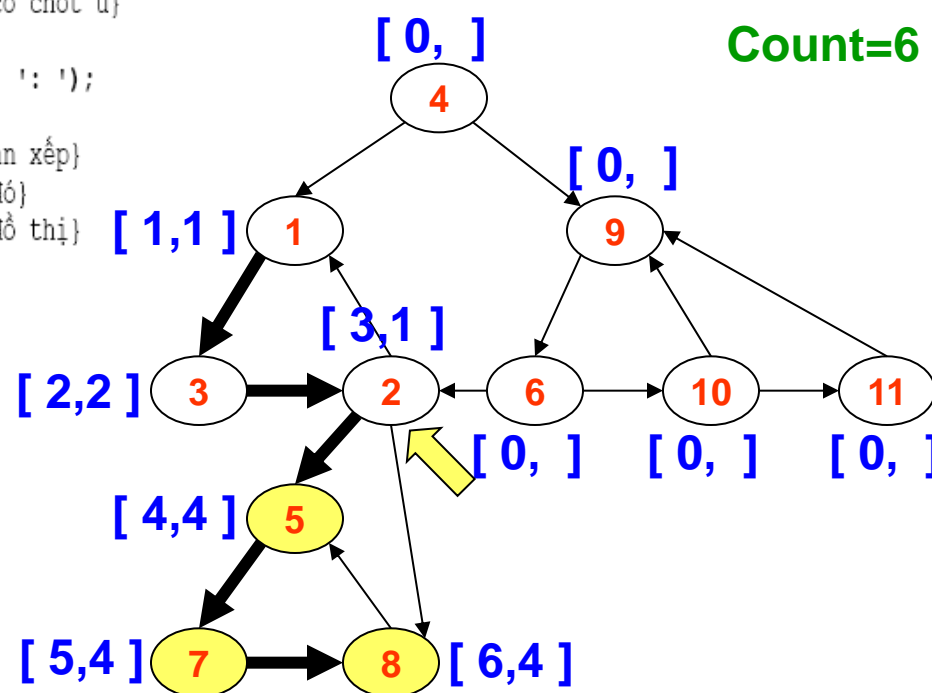
```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

```
end;
```

Component 1:

8 7 5



2
3
1

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt} ←
```

```
begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
  Inc(ComponentCount);
```

```
  WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
  repeat
```

```
    v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
    Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
    Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
  until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
  WriteLn(fo);
```

```
end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

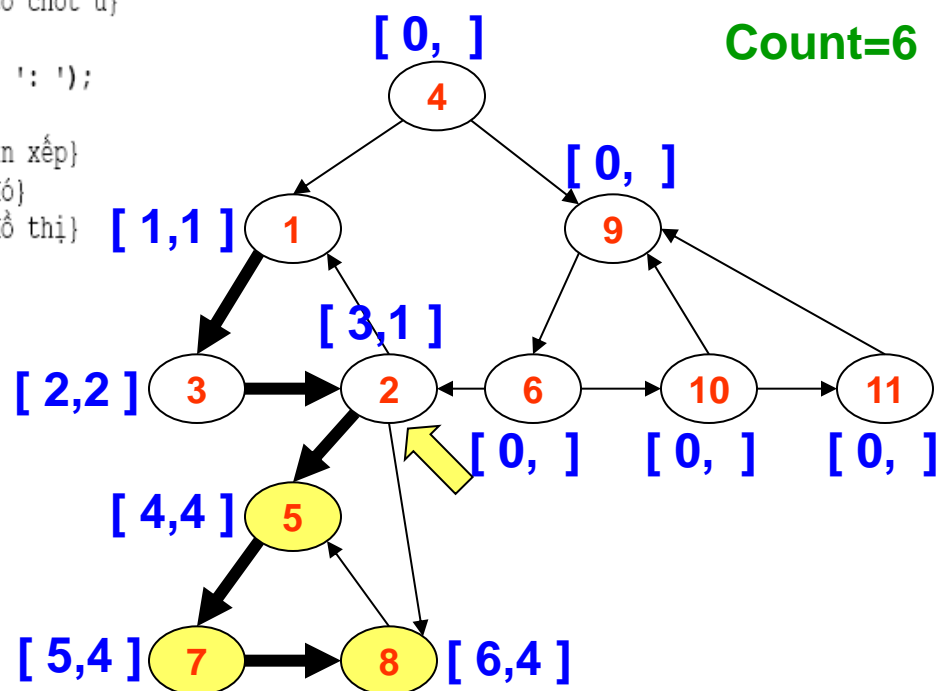
```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

```
end;
```

Component 1:

8 7 5



2
3
1

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); ← cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rời loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

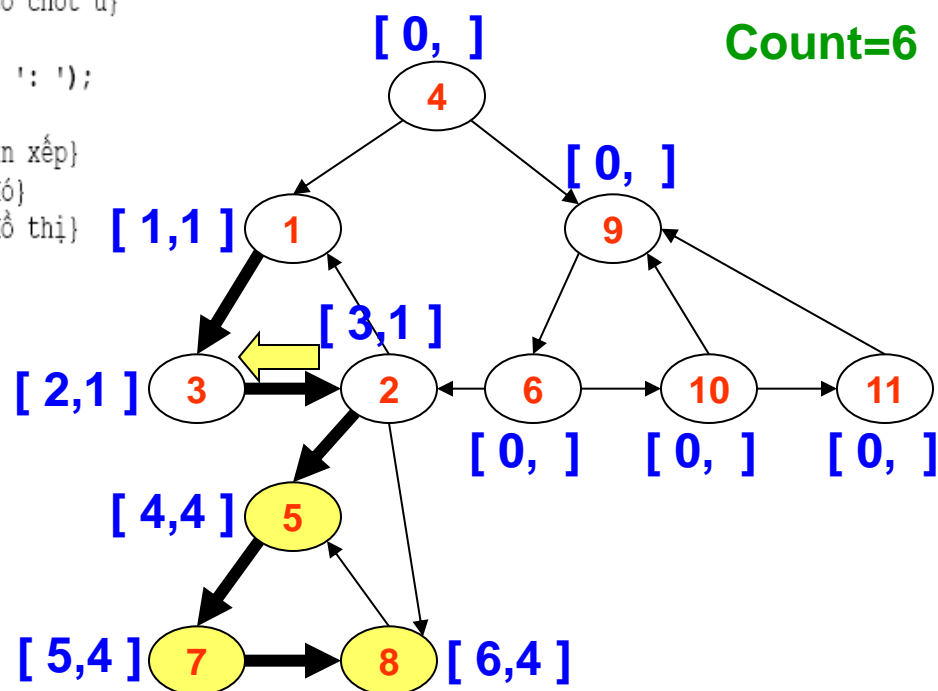
```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

```
end;
```

Component 1:

8 7 5



2
3
1

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt} ←
```

```
begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
  Inc(ComponentCount);
```

```
  WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
  repeat
```

```
    v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
    Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
    Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
  until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
  WriteLn(fo);
```

```
end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

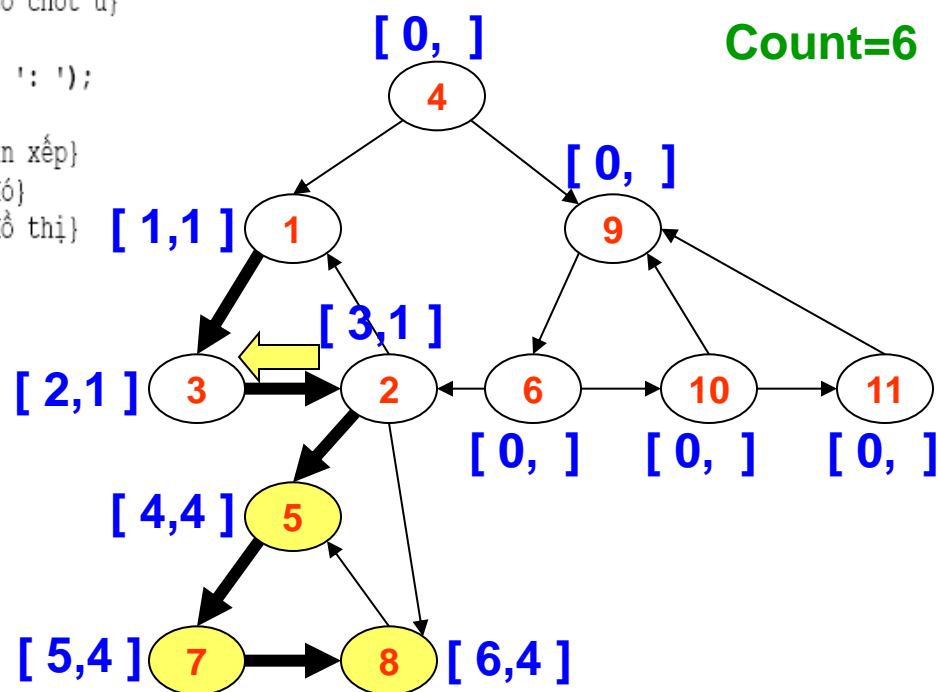
```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

```
end;
```

Component 1:

8 7 5



2
3
1

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); ← cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rời loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

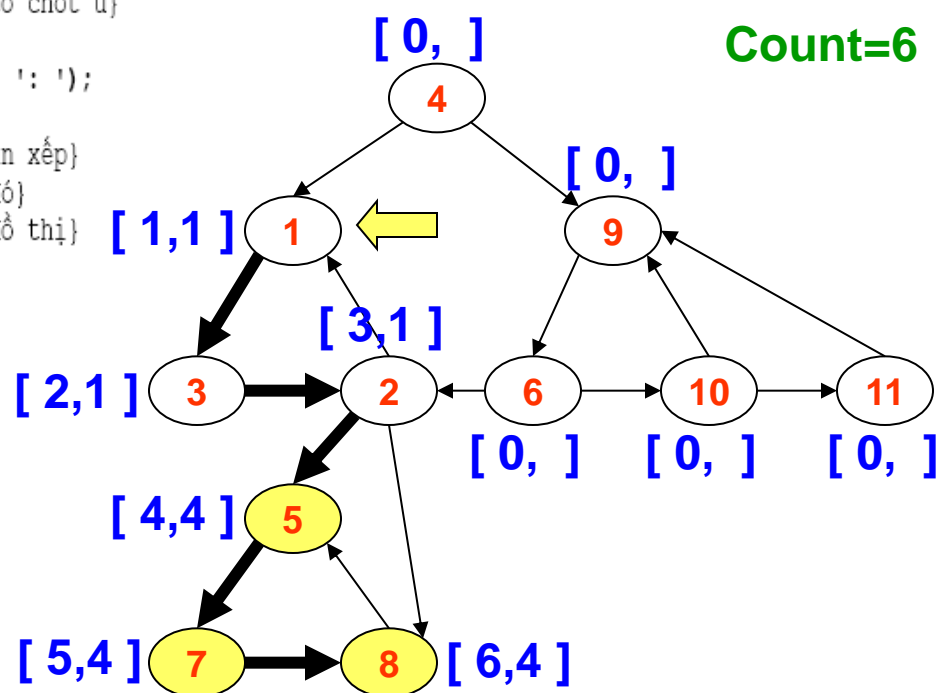
```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

```
end;
```

Component 1:

8 7 5



2
3
1

STACK


```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt} ←
```

```
begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
  Inc(ComponentCount);
```

```
  WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
  repeat
```

```
    v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
    Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
    Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
  until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
  WriteLn(fo);
```

```
end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

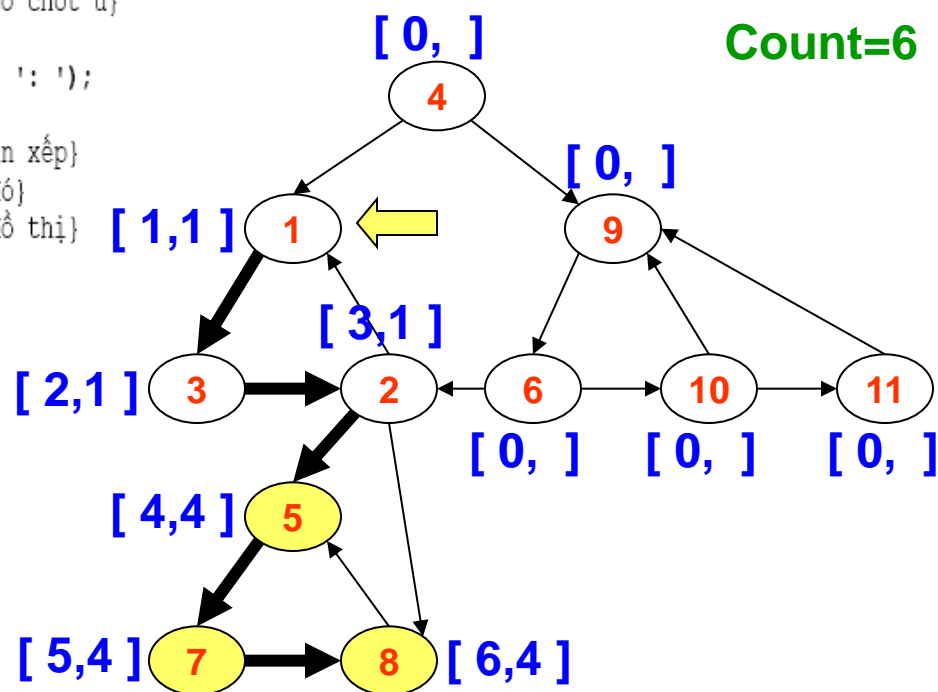
```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

```
end;
```

Component 1:

8 7 5



2
3
1

STACK


```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cùng tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

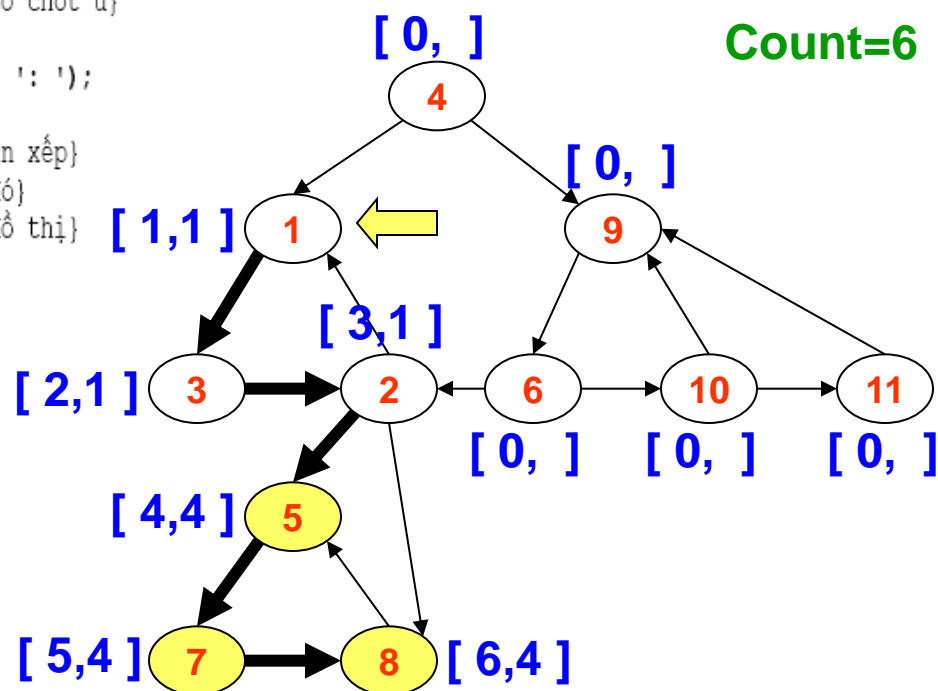
```
    if Number[u] = 0 then Visit(u);
```

```
end;
```

Component 1:

8 7 5

Component 2:



2
3
1

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cùng tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

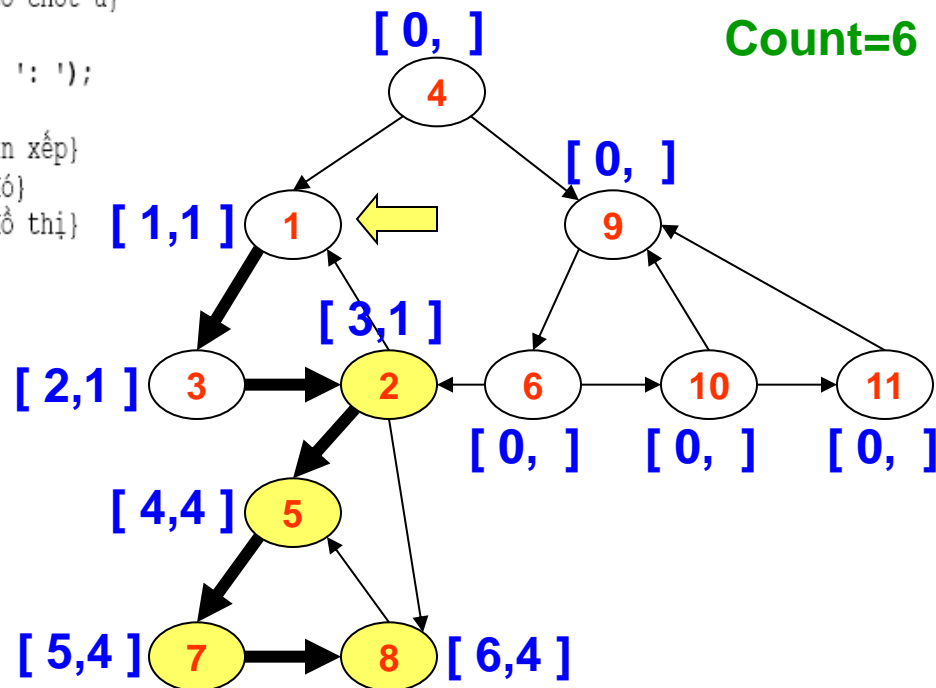
```
end;
```

Component 1:

8 7 5

Component 2:

2



3
1

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cùng tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

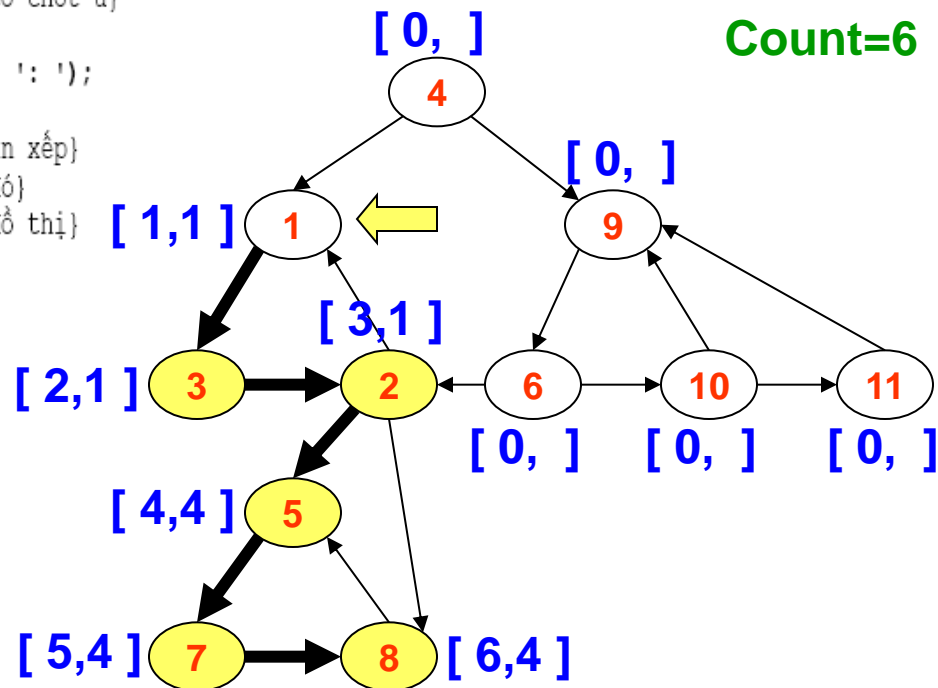
```
end;
```

Component 1:

8 7 5

Component 2:

2 3



1
STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

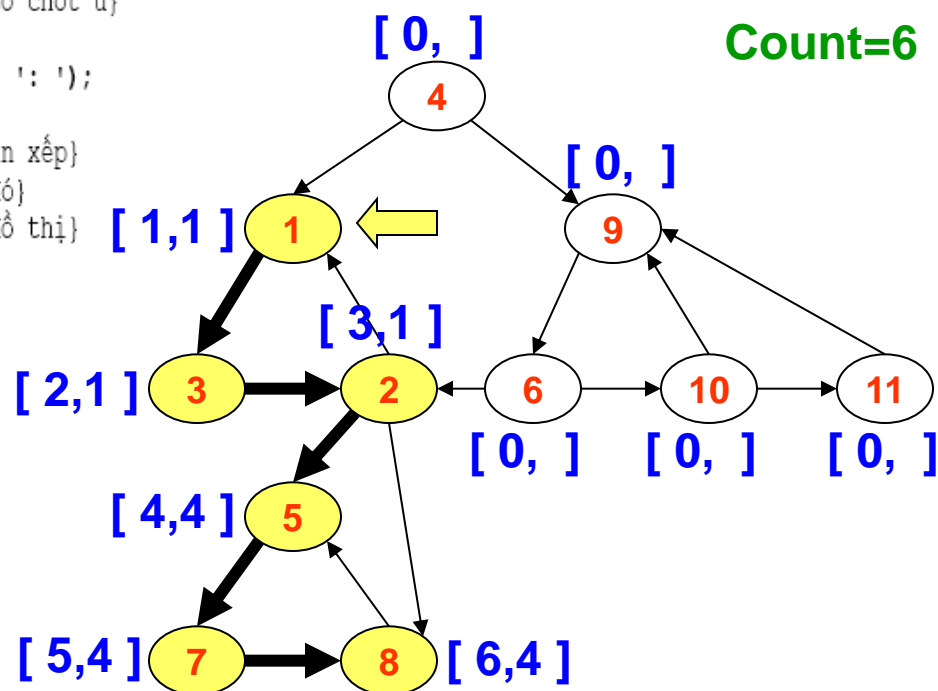
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1



STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cùng tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

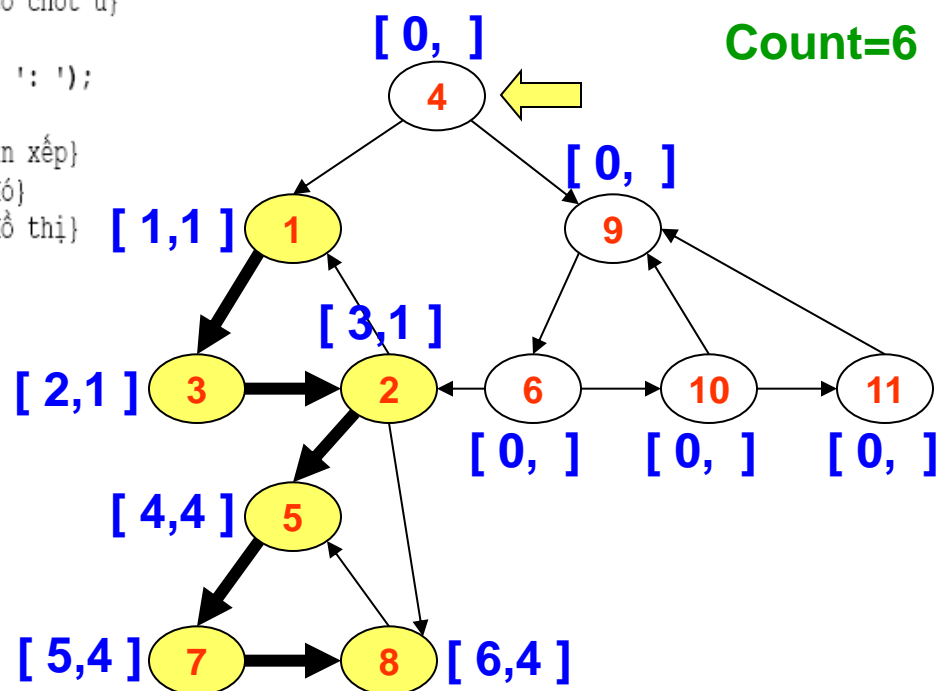
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1



```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp} ←
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

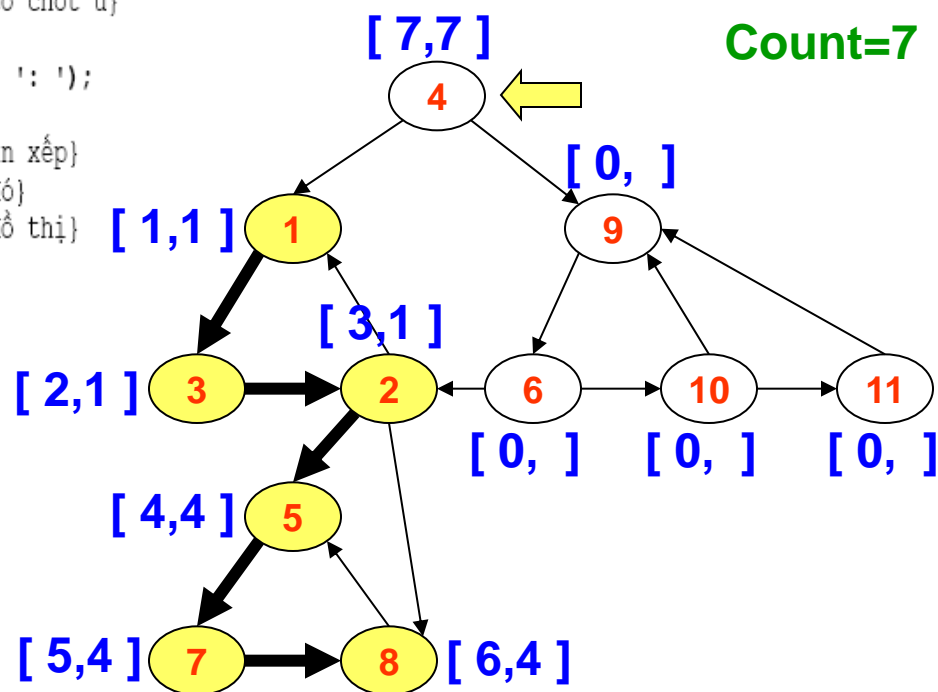
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1



```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chót}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chót u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;  
procedure Solve;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

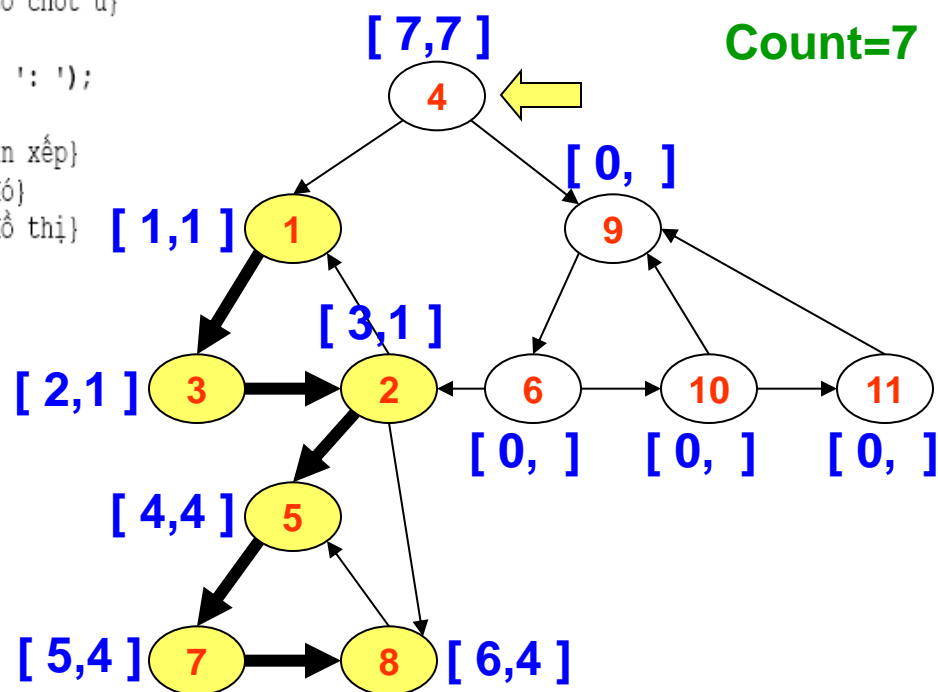
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1




```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```


```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v);  {tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
procedure Solve;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

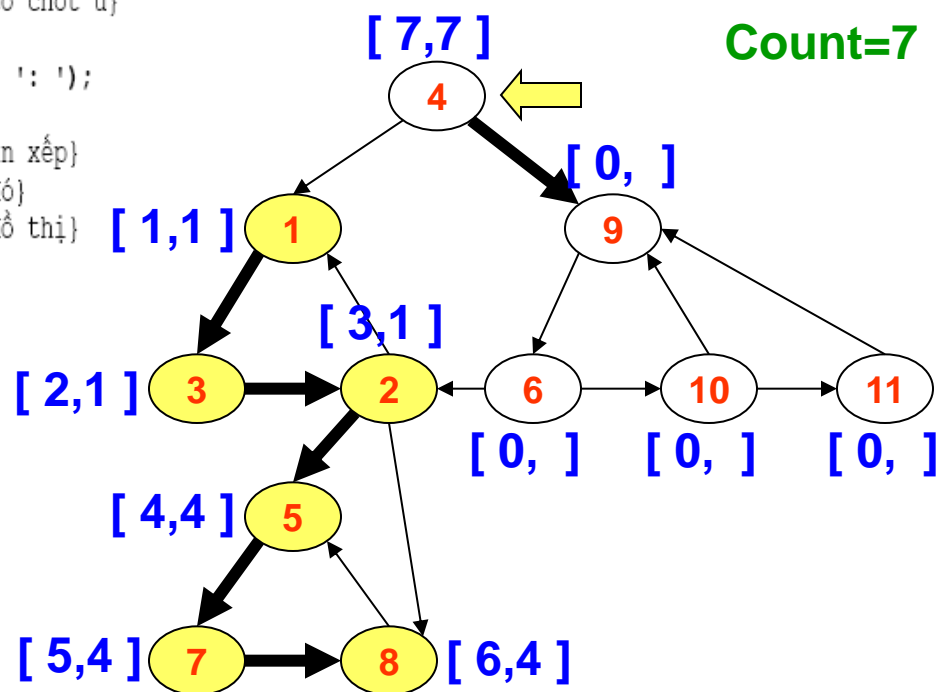
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1



4

STACK


```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```


```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v);  {tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

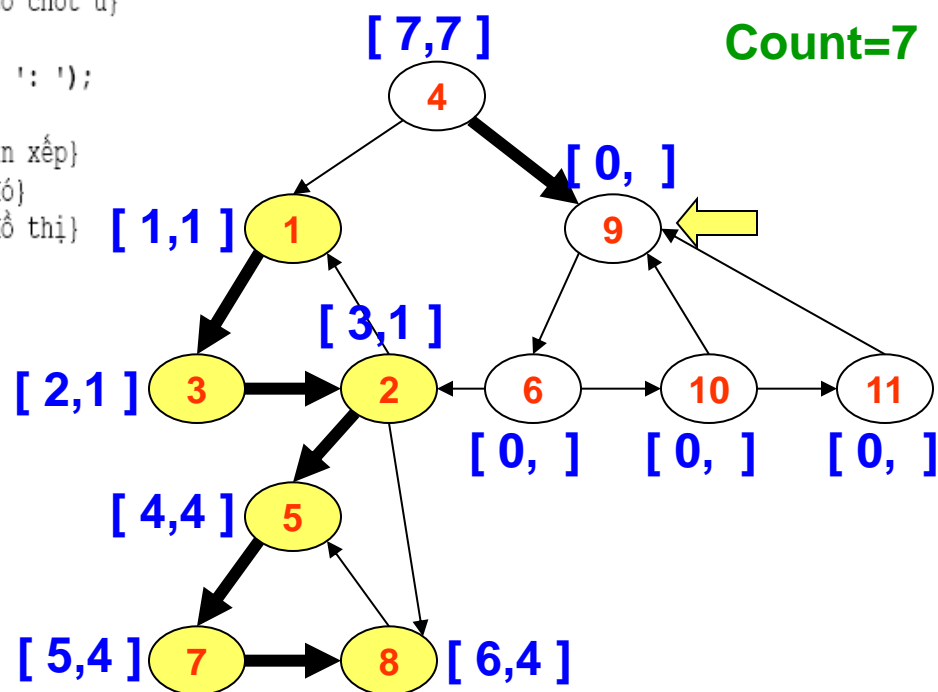
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1



4

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp} ←
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
procedure Solve;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

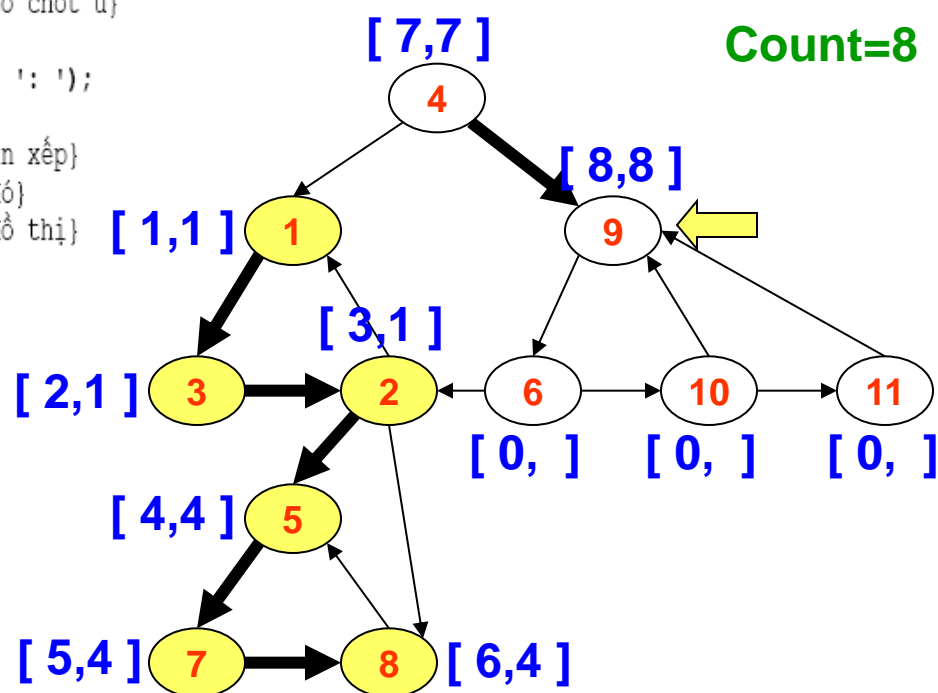
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1



9
4

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```



```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

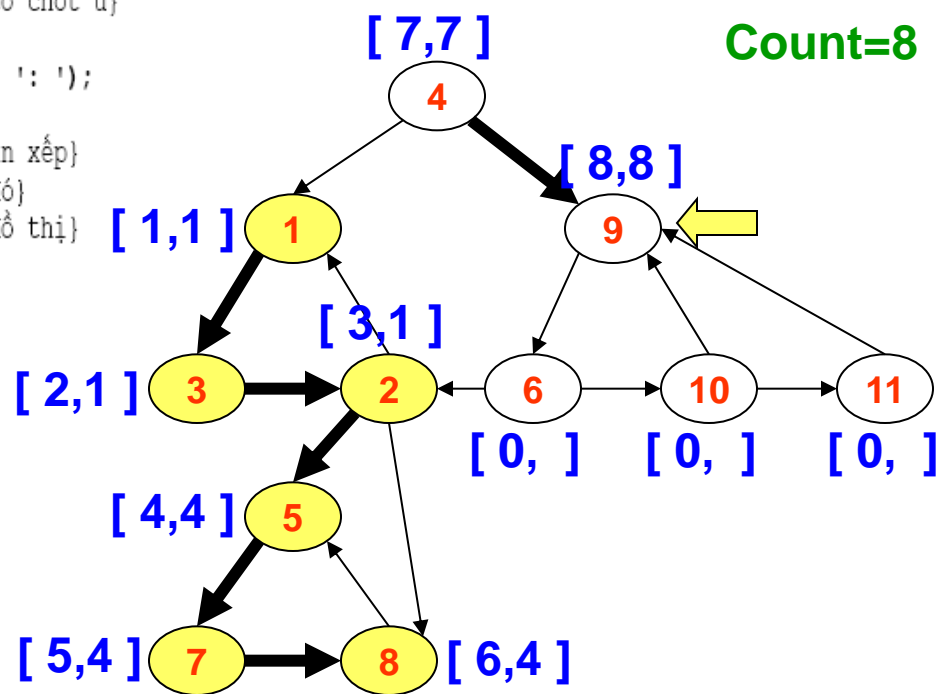
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1



Count=8

9
4

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```


```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v);  tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
procedure Solve;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

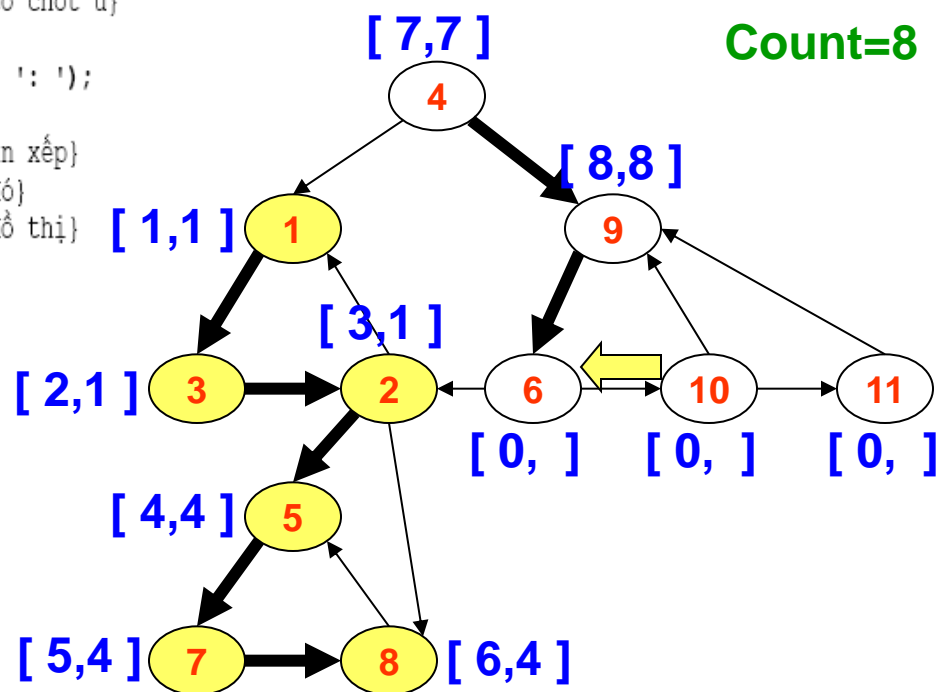
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1



9
4

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp} ←
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

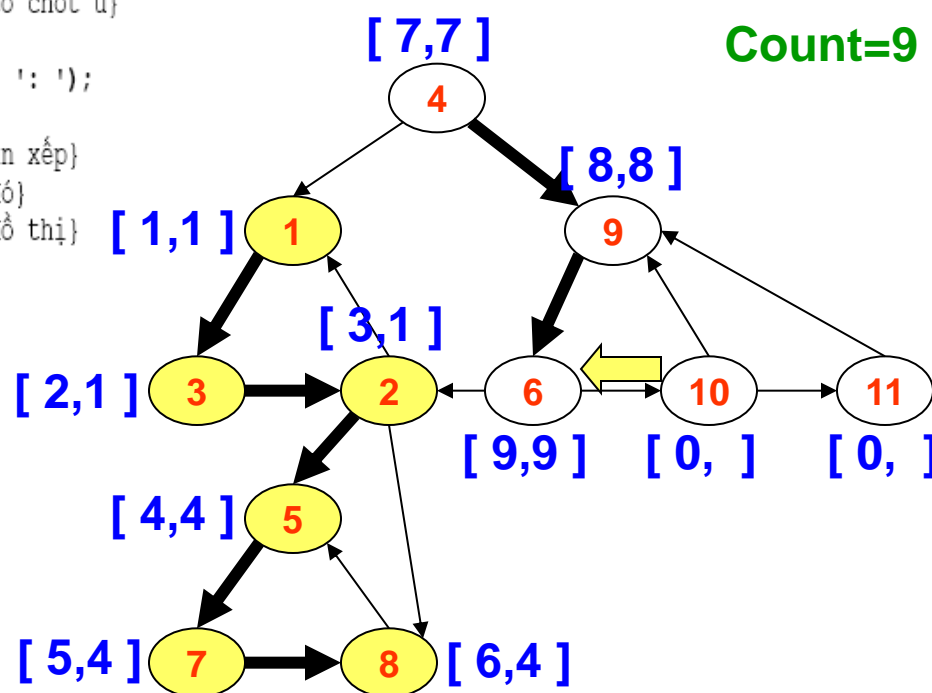
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1



6
9
4

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

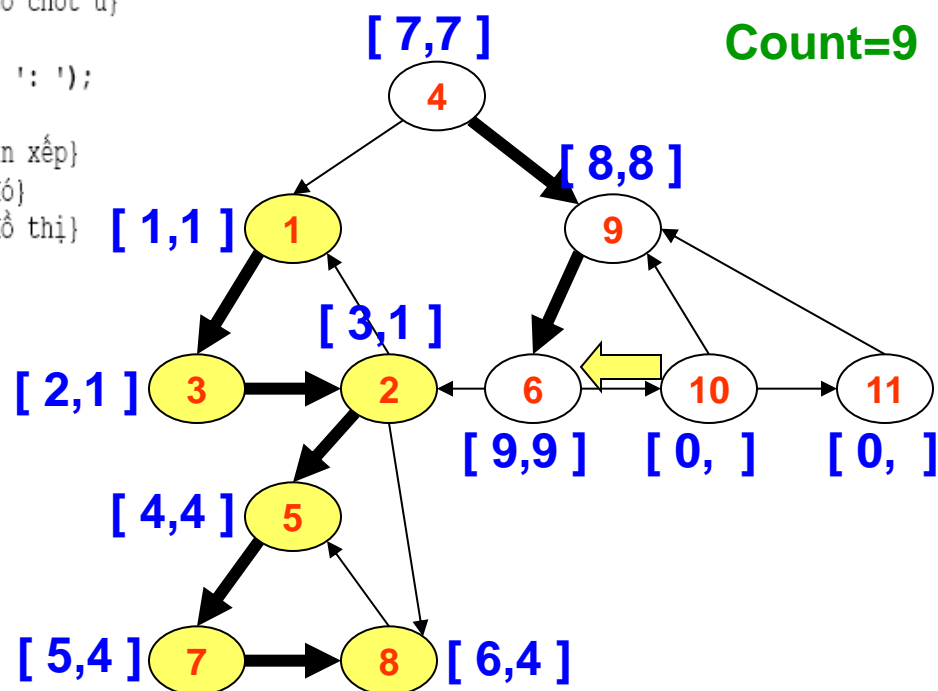
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1



6
9
4

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```


```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v);  {tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

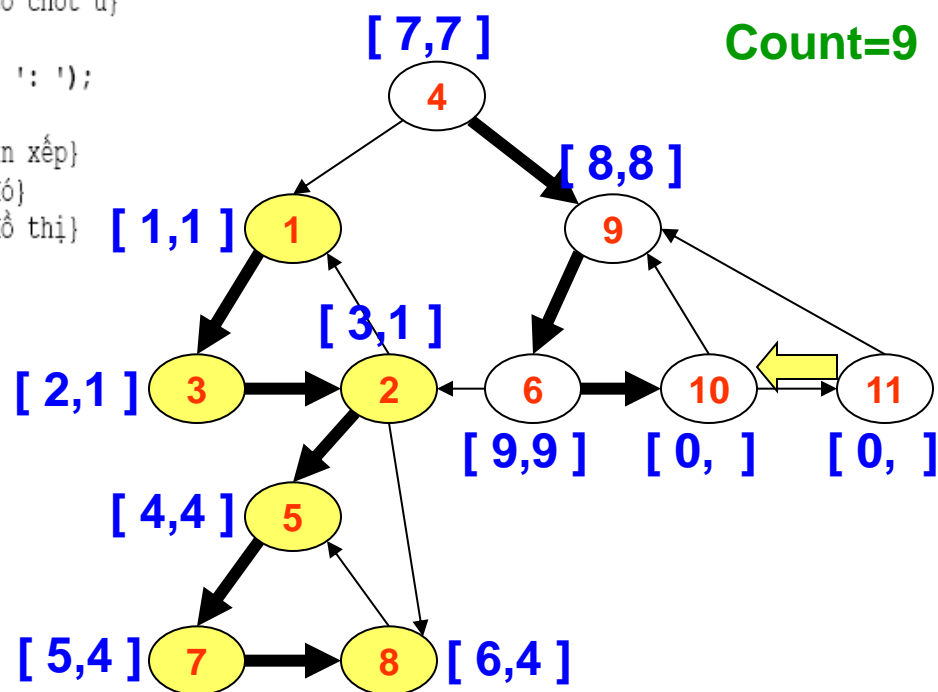
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1



Count=9

6
9
4

STACK


```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp} ←
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;  
procedure Solve;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

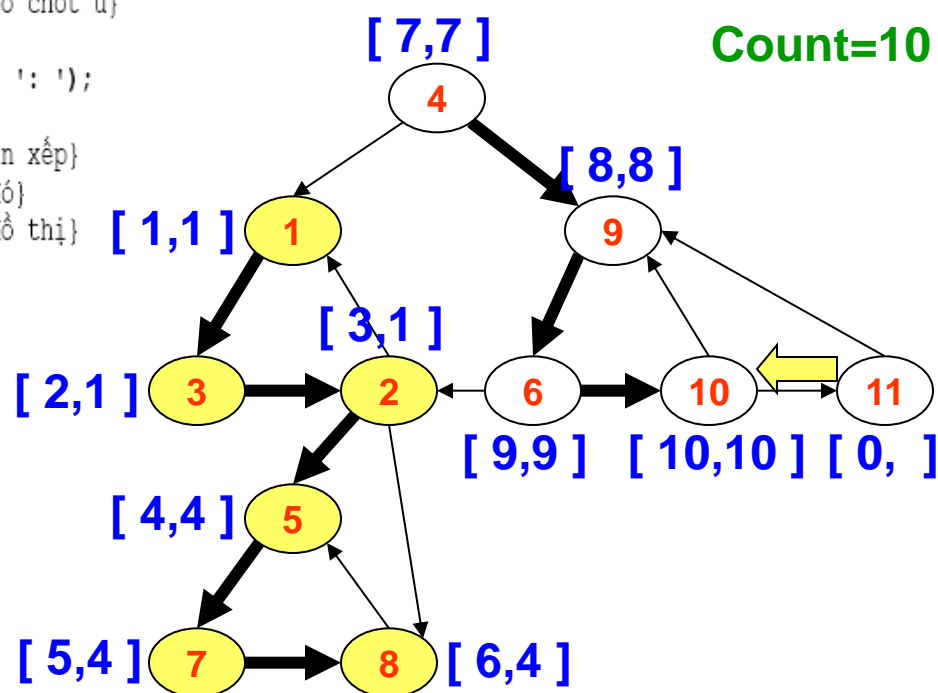
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1



10
6
9
4

STACK


```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

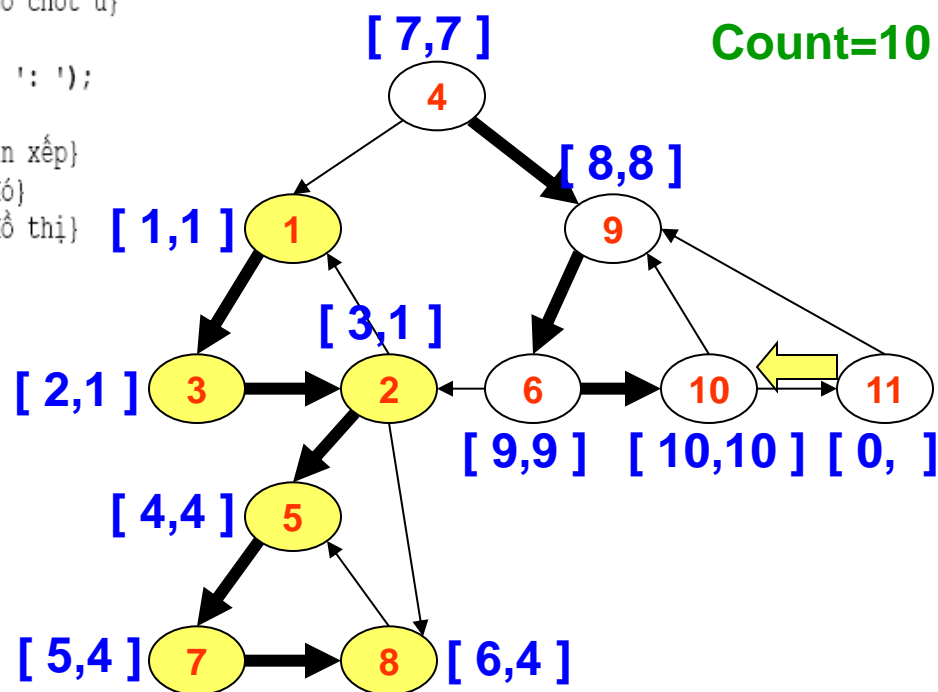
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1



10
6
9
4

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) ← {liều hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

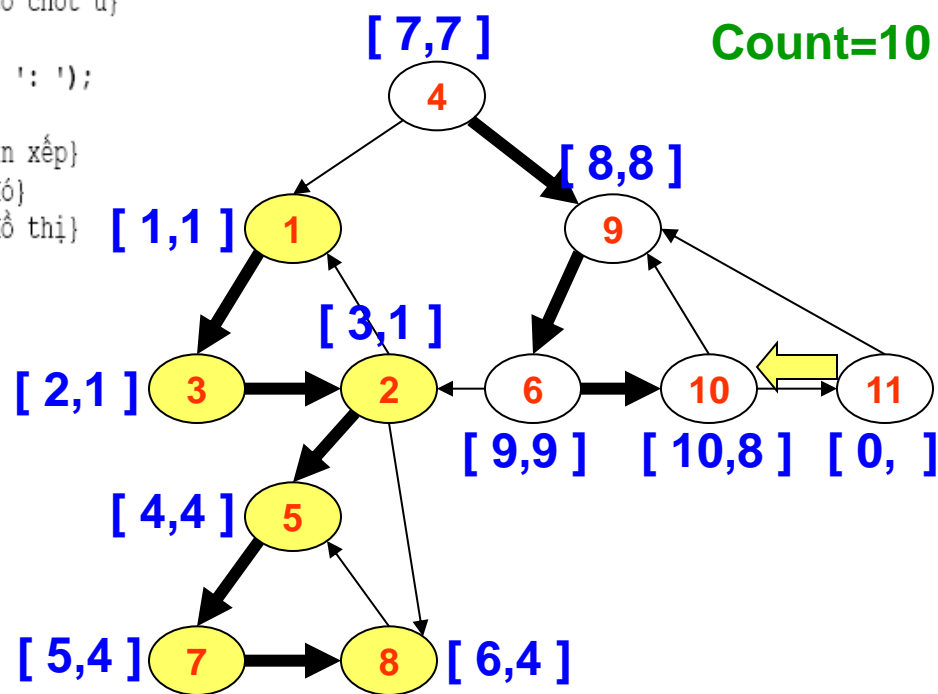
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1



Count=10

10
6
9
4

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```


```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v);  {tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ' : ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ' '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

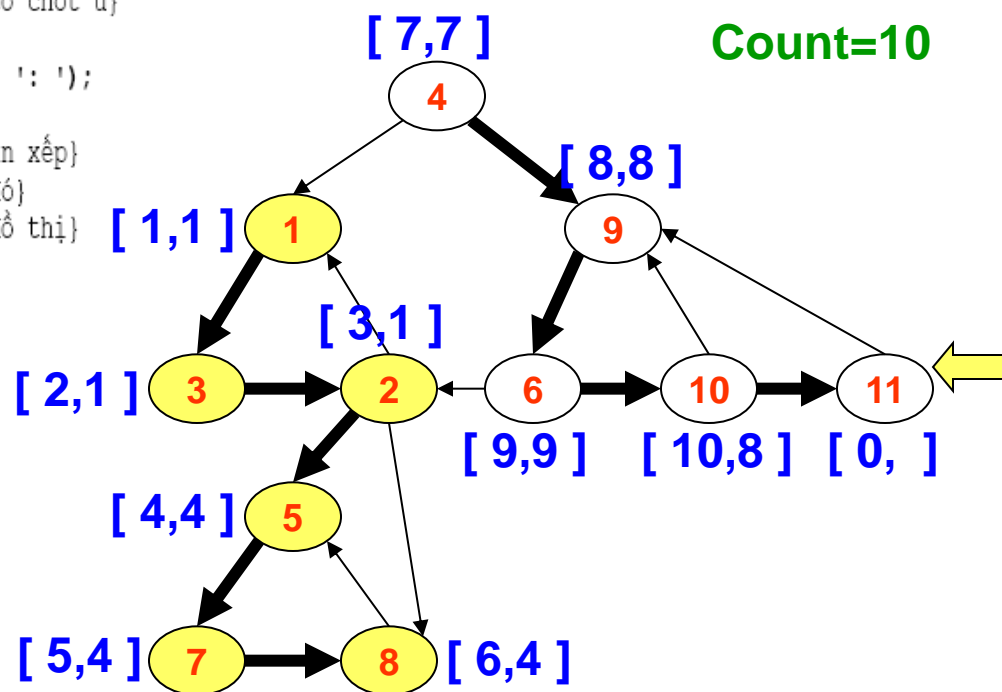
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1



10
6
9
4

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp} ←
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ' '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

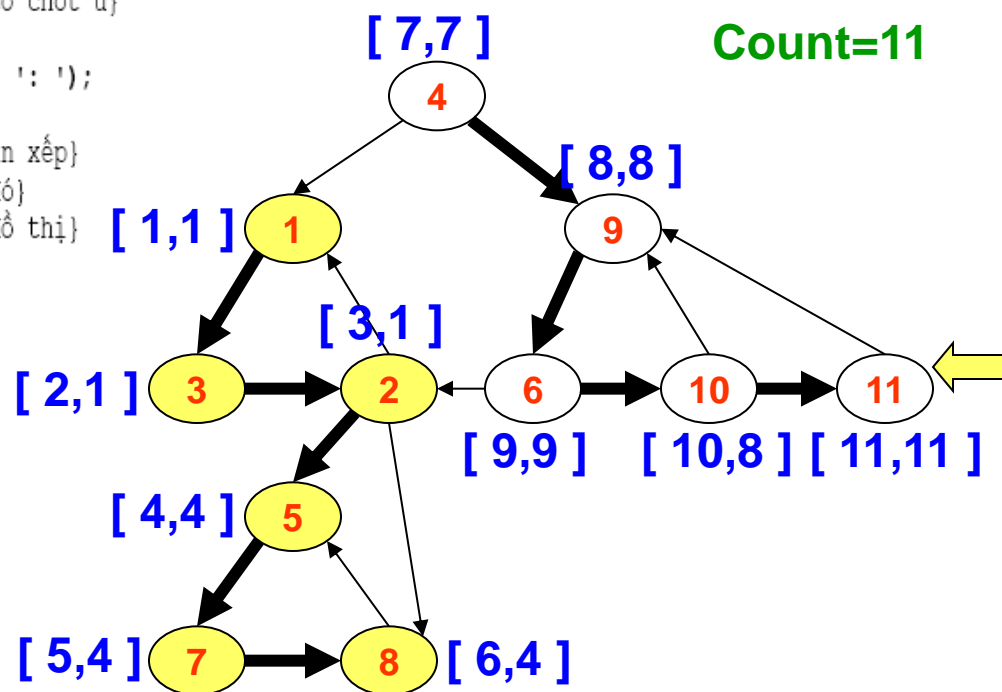
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1



11
10
6
9
4

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chót}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chót u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;  
procedure Solve;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

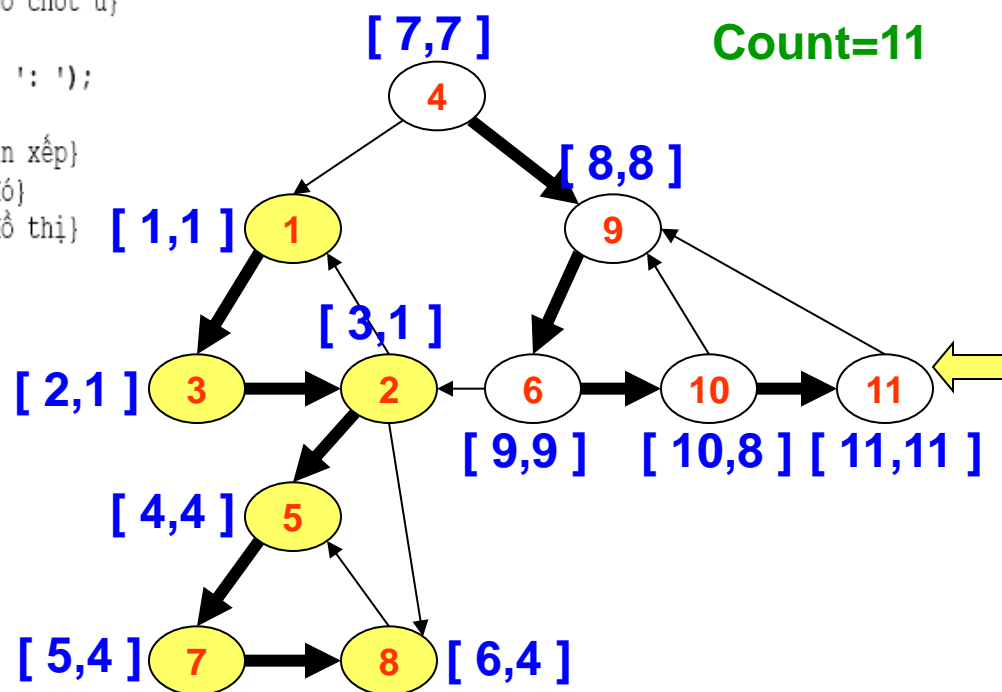
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1



11
10
6
9
4

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cùng tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) ← {liều hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

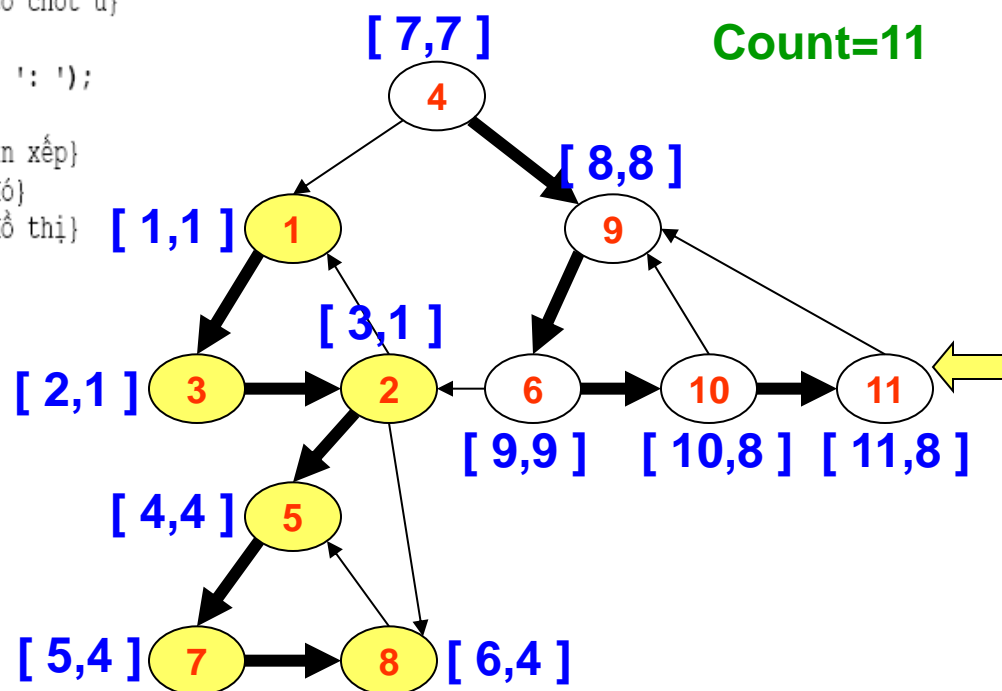
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1



11
10
6
9
4

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt} ←
```

```
begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
  Inc(ComponentCount);
```

```
  WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
  repeat
```

```
    v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
    Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
    Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
  until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
  WriteLn(fo);
```

```
end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

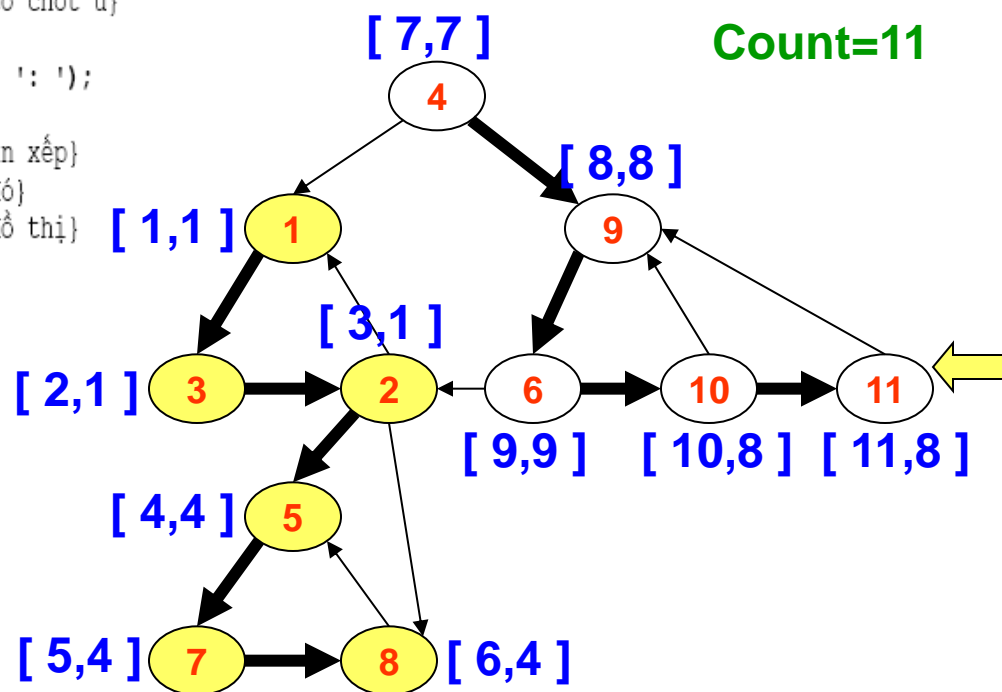
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1



11
10
6
9
4

STACK


```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); ← cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rời loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

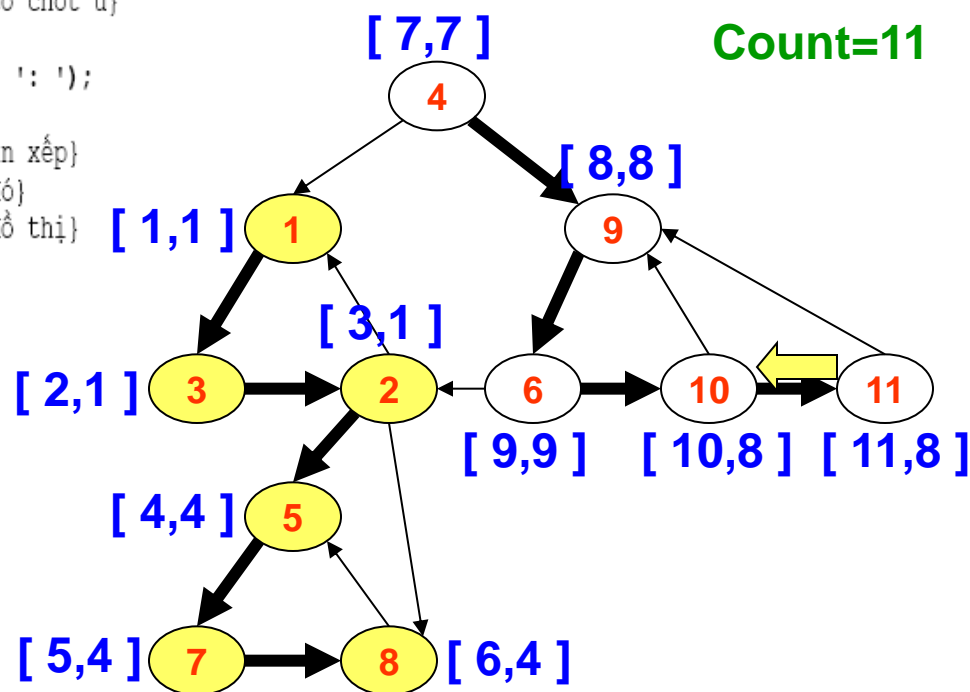
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1



Count=11

11
10
6
9
4

STACK


```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt} ←
```

```
begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
  Inc(ComponentCount);
```

```
  WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
  repeat
```

```
    v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
    Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
    Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
  until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
  WriteLn(fo);
```

```
end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

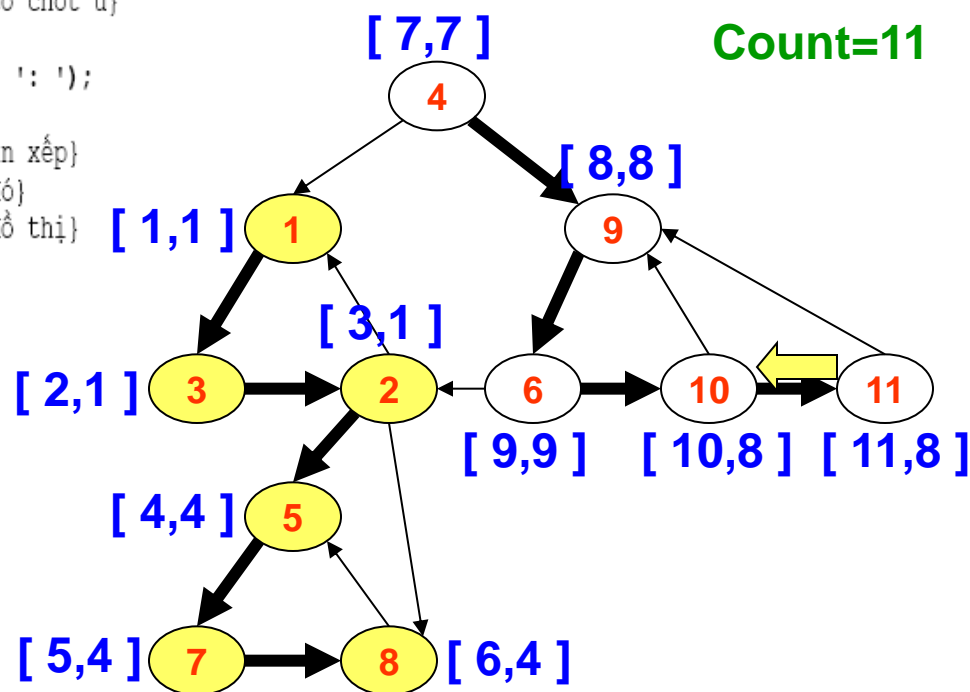
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1



Count=11

11
10
6
9
4

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); ← cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chót}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chót u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rời loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

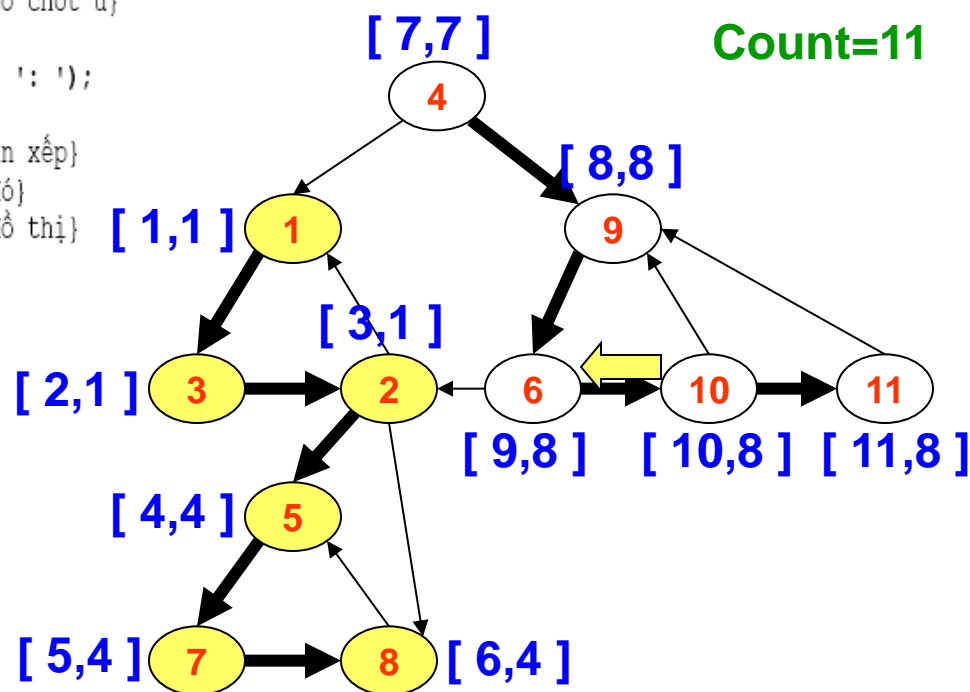
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1



11
10
6
9
4

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt} ←
```

```
begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
  Inc(ComponentCount);
```

```
  WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
  repeat
```

```
    v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
    Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
    Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
  until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
  WriteLn(fo);
```

```
end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

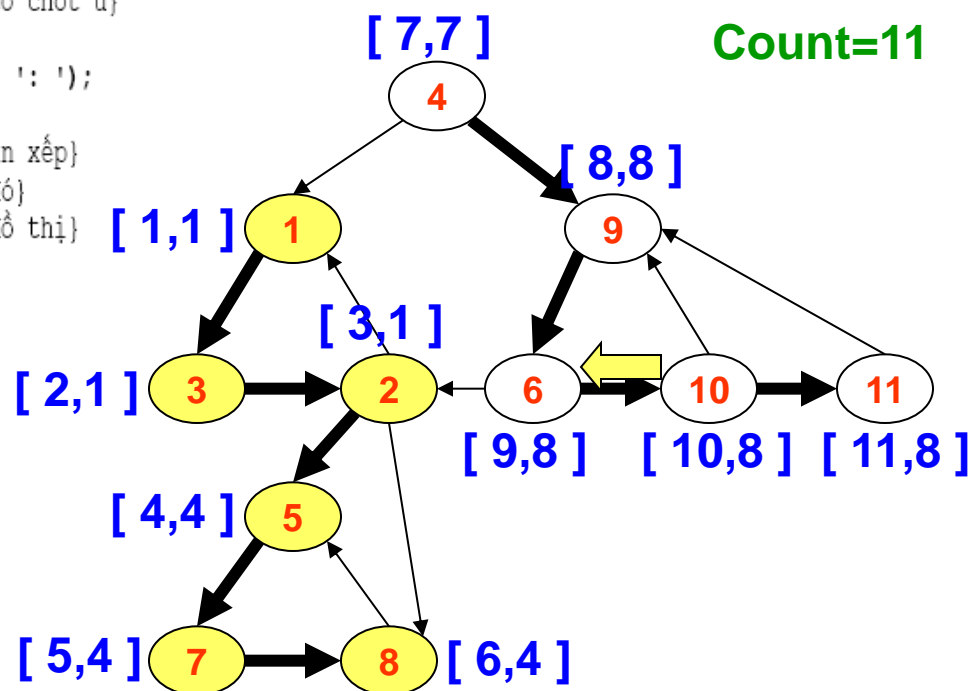
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1



11
10
6
9
4

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); ← cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rời loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

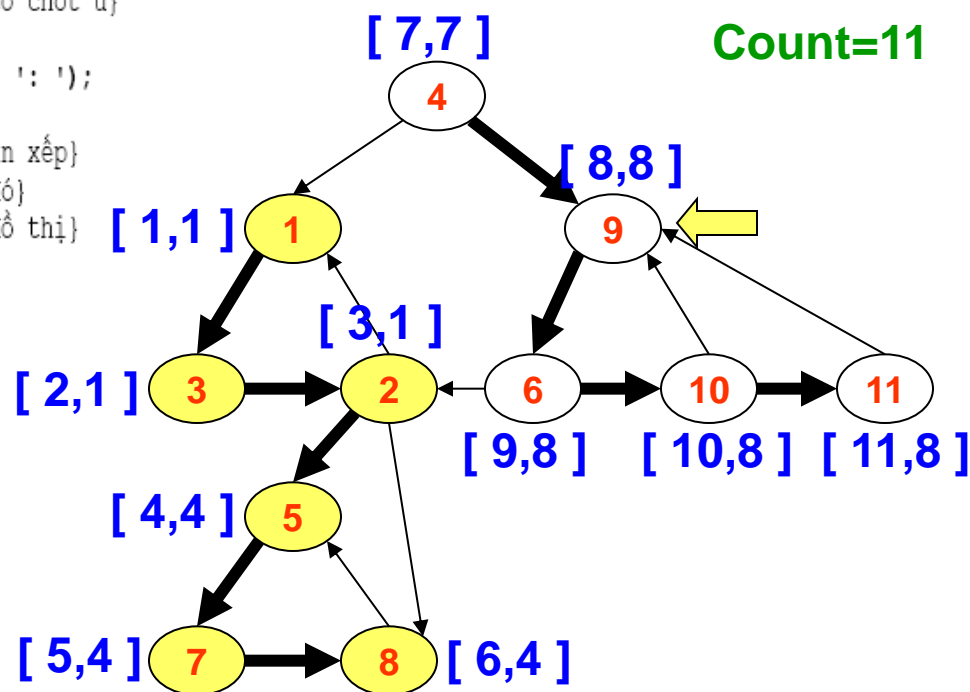
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1



11
10
6
9
4

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt} ←
```

```
begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
  Inc(ComponentCount);
```

```
  WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
  repeat
```

```
    v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
    Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
    Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
  until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
  WriteLn(fo);
```

```
end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

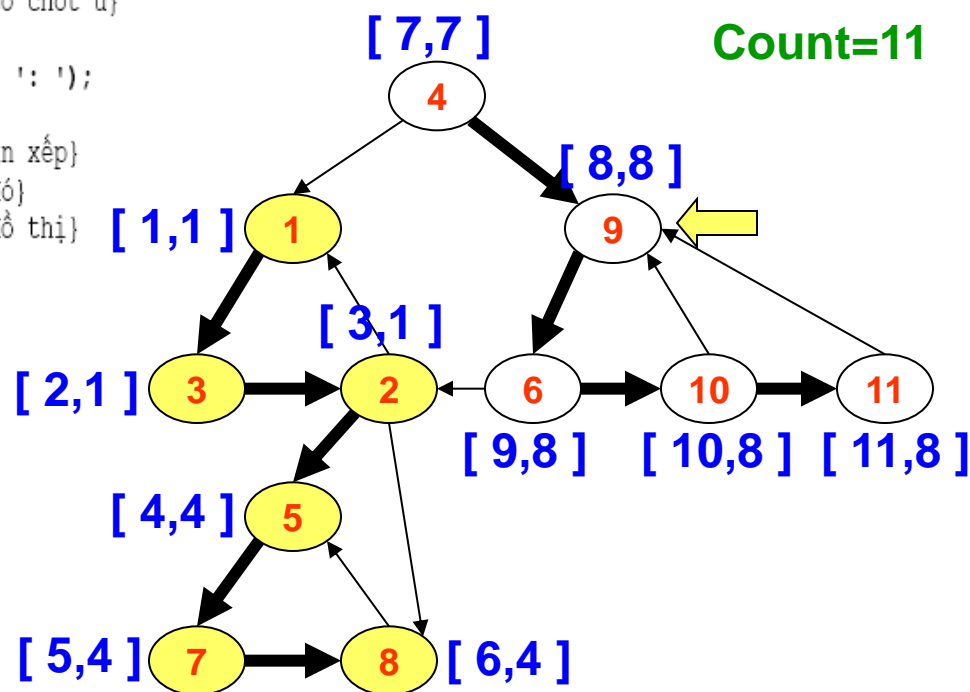
```
end;
```

Component 1:

8 7 5

Component 2:

2 3 1



11
10
6
9
4

STACK

```

procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
var
  v: Integer;
begin
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
  Push(u); {Đẩy u vào ngăn xếp}
  for v := 1 to n do
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
      if Number[v] <> 0 then {Nếu v đã thăm}
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
      else {Nếu v chưa thăm}
        begin
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
        end;
  {Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
  if Number[u] = Low[u] then {Nếu u là chốt}
    begin {Liệt kê thành phần liên thông mạnh có chốt u}
      Inc(ComponentCount);
      WriteLn(fo, 'Component ', ComponentCount, ': ');
      repeat
        v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
        Write(fo, v, ' '); {Liệt kê các đỉnh đó}
        Free[v] := False; {Rồi loại luôn khỏi đồ thị}
      until v = u; {Cho tới khi lấy tới đỉnh u}
      WriteLn(fo);
    end;
end;
procedure Solve;
var
  u: Integer;
begin
  for u := 1 to n do
    if Number[u] = 0 then Visit(u);
end;

```

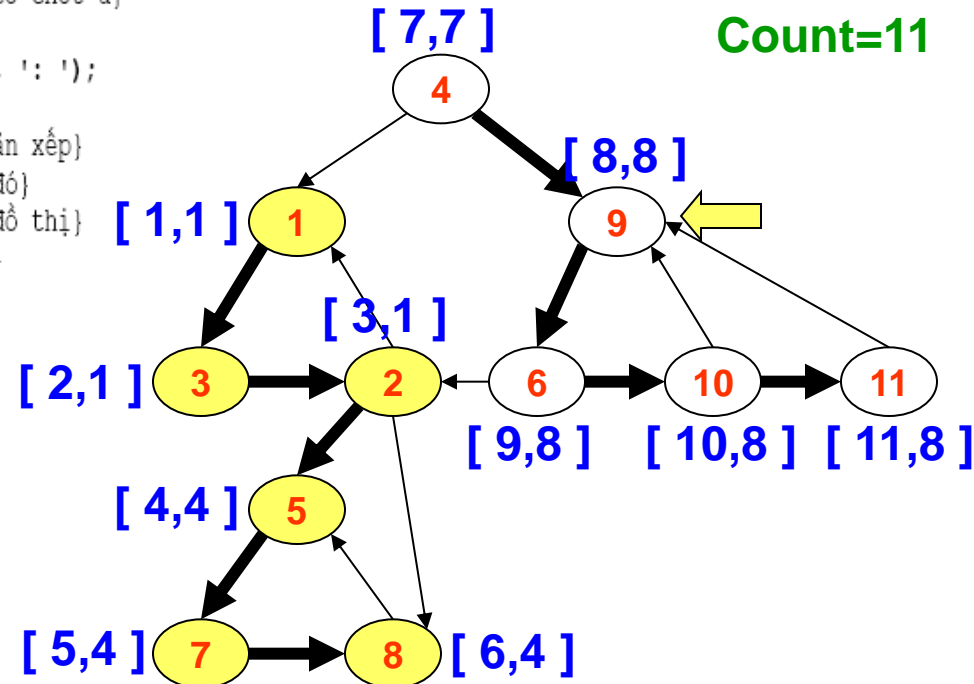
Component 1:

8 7 5

Component 2:

2 3 1

Component 3:



11
10
6
9
4

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

```
end;
```

Component 1:

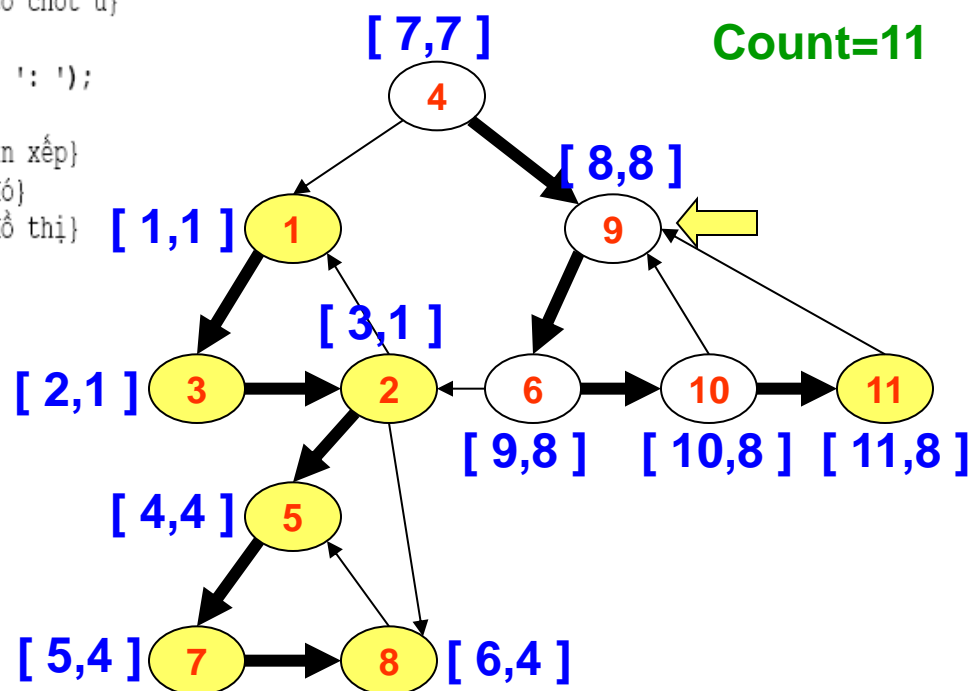
8 7 5

Component 2:

2 3 1

Component 3:

11



10
6
9
4

STACK


```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

```
end;
```

Component 1:

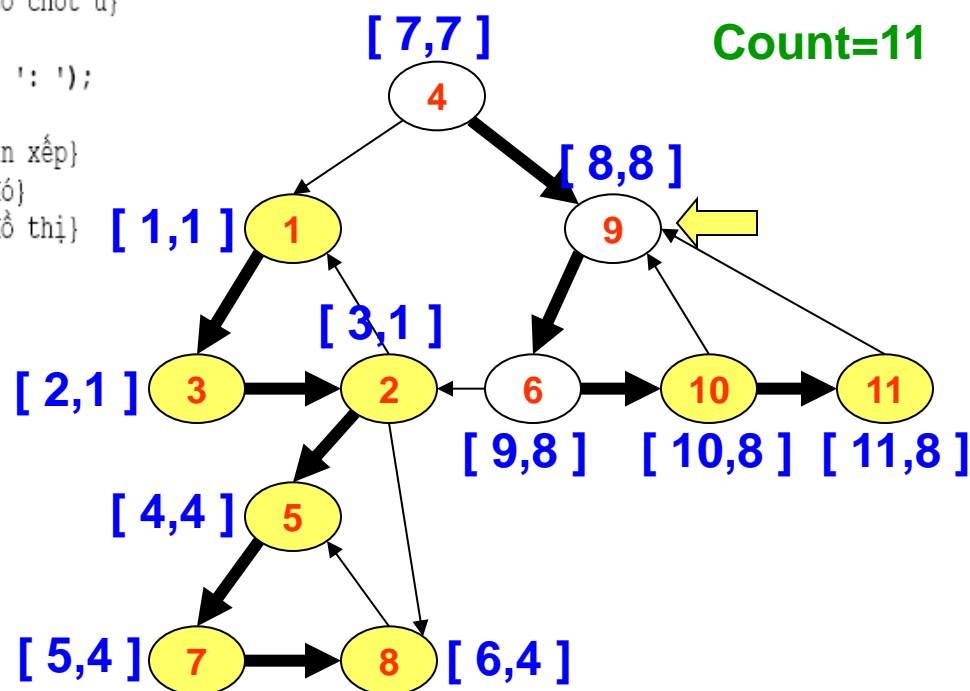
8 7 5

Component 2:

2 3 1

Component 3:

11 10



6
9
4

STACK


```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat ←
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
procedure Solve;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

```
end;
```

Component 1:

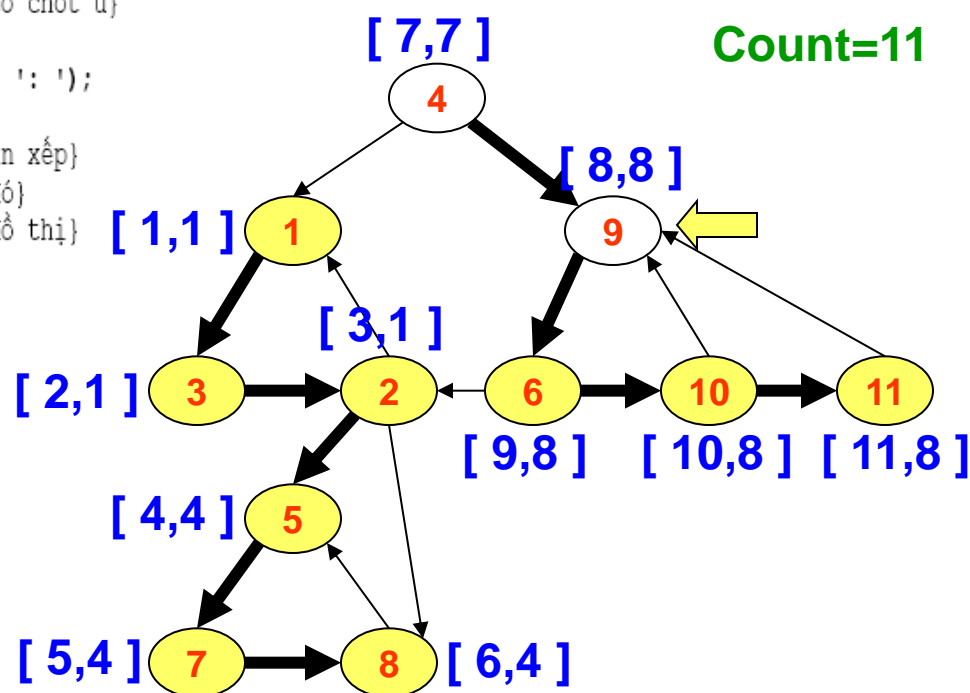
8 7 5

Component 2:

2 3 1

Component 3:

11 10 6



9
4

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cùng tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat ←
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
procedure Solve;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

```
end;
```

Component 1:

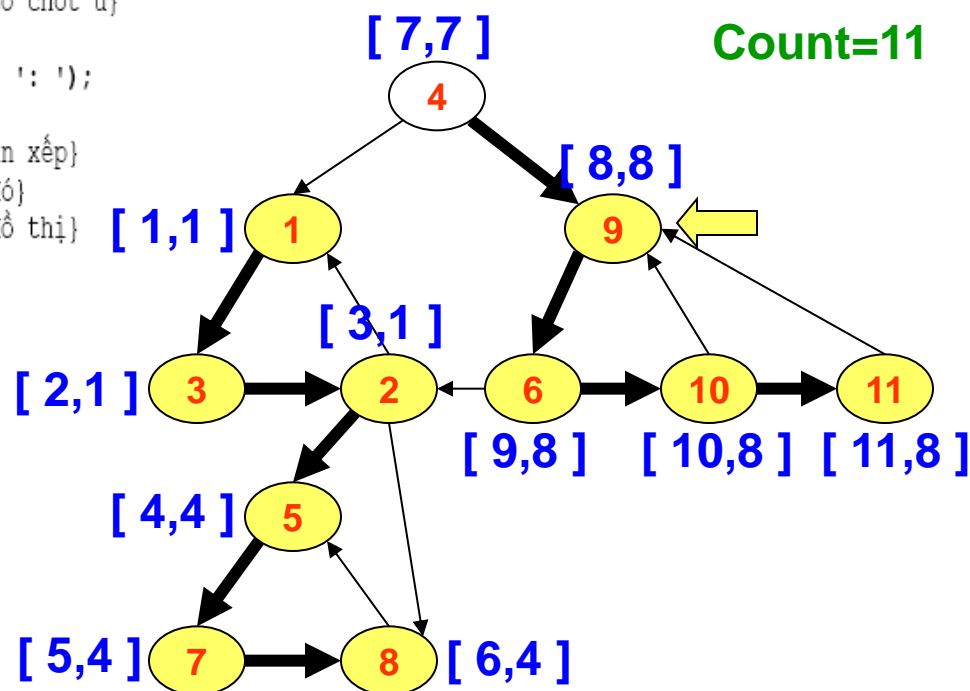
8 7 5

Component 2:

2 3 1

Component 3:

11 10 6 9



4

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); ← cực tiểu hoá Low[u] theo công thức này
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rời loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

```
end;
```

Component 1:

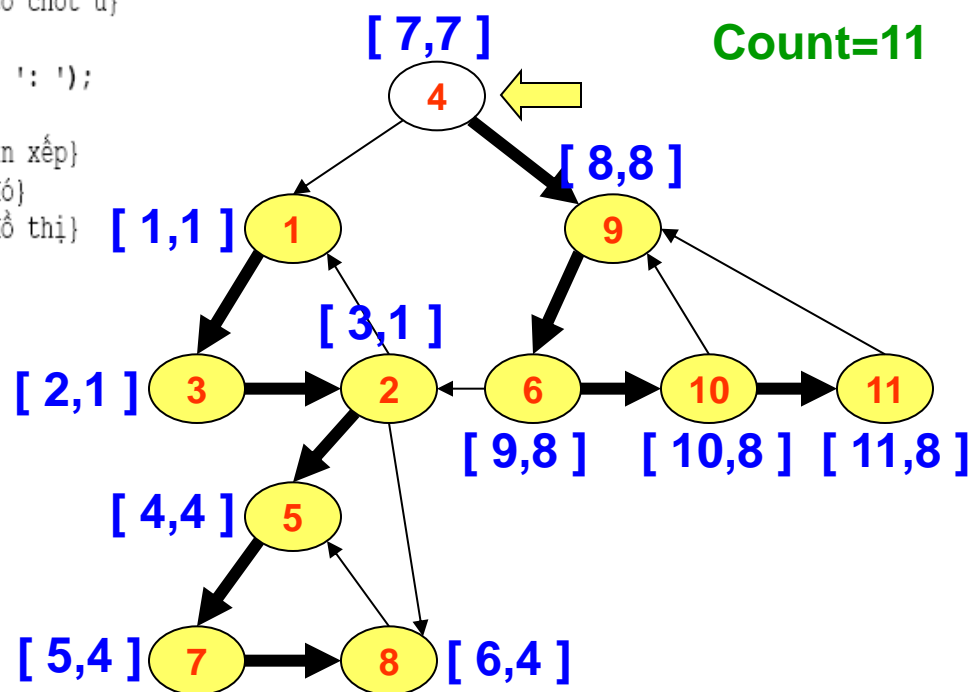
8 7 5

Component 2:

2 3 1

Component 3:

11 10 6 9



Count=11

4
STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cung tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt} ←
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
procedure Solve;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

```
end;
```

Component 1:

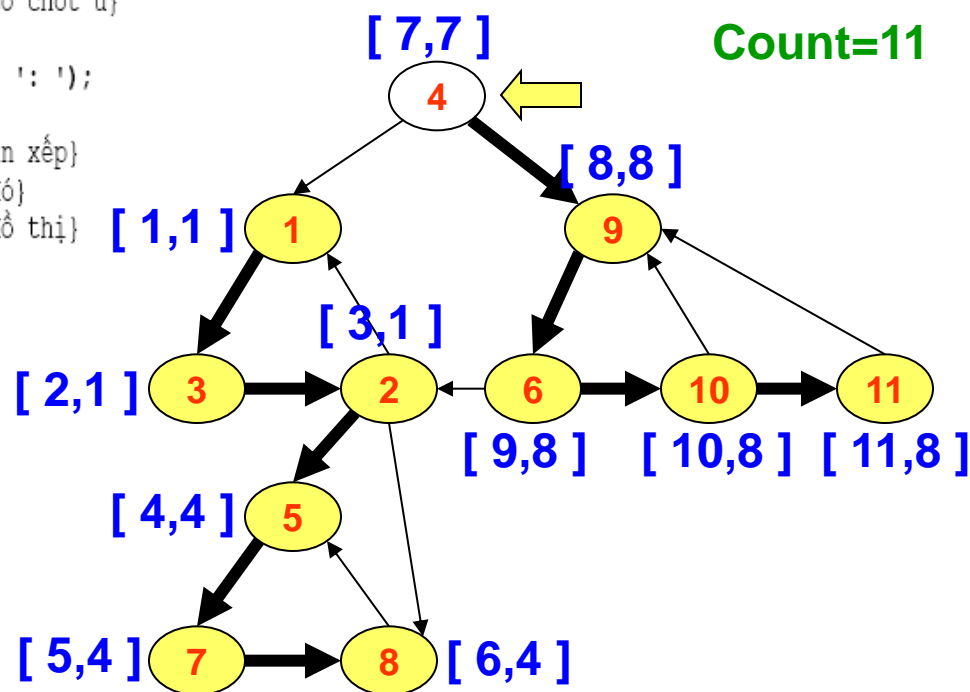
8 7 5

Component 2:

2 3 1

Component 3:

11 10 6 9



4

STACK

```
procedure Visit(u: Integer); {Thuật toán tìm kiếm theo chiều sâu bắt đầu từ u}
```

```
var
```

```
  v: Integer;
```

```
begin
```

```
  Inc(Count); Number[u] := Count; {Trước hết đánh số cho u}
```

```
  Low[u] := Number[u]; {Coi u có cùng tới u, nên có thể khởi gán Low[u] thế này rồi sau cực tiểu hoá dần}
```

```
  Push(u); {Đẩy u vào ngăn xếp}
```

```
  for v := 1 to n do
```

```
    if Free[v] and a[u, v] then {Xét những đỉnh v kề u}
```

```
      if Number[v] <> 0 then {Nếu v đã thăm}
```

```
        Low[u] := Min(Low[u], Number[v]) {Cực tiểu hoá Low[u] theo công thức này}
```

```
      else {Nếu v chưa thăm}
```

```
        begin
```

```
          Visit(v); {Tiếp tục tìm kiếm theo chiều sâu bắt đầu từ v}
```

```
          Low[u] := Min(Low[u], Low[v]); {Rồi cực tiểu hoá Low[u] theo công thức này}
```

```
        end;
```

```
{Đến đây thì đỉnh u được duyệt xong, tức là các đỉnh thuộc nhánh DFS gốc u đều đã thăm}
```

```
if Number[u] = Low[u] then {Nếu u là chốt}
```

```
  begin {Liệt kê thành phần liên thông mạnh có chốt u}
```

```
    Inc(ComponentCount);
```

```
    WriteLn(fo, 'Component ', ComponentCount, ': ');
```

```
    repeat
```

```
      v := Pop; {Lấy dần các đỉnh ra khỏi ngăn xếp}
```

```
      Write(fo, v, ', '); {Liệt kê các đỉnh đó}
```

```
      Free[v] := False; {Rồi loại luôn khỏi đồ thị}
```

```
    until v = u; {Cho tới khi lấy tới đỉnh u}
```

```
    WriteLn(fo);
```

```
  end;
```

```
end;
```

```
var
```

```
  u: Integer;
```

```
begin
```

```
  for u := 1 to n do
```

```
    if Number[u] = 0 then Visit(u);
```

```
end;
```

Component 1:

8 7 5

Component 2:

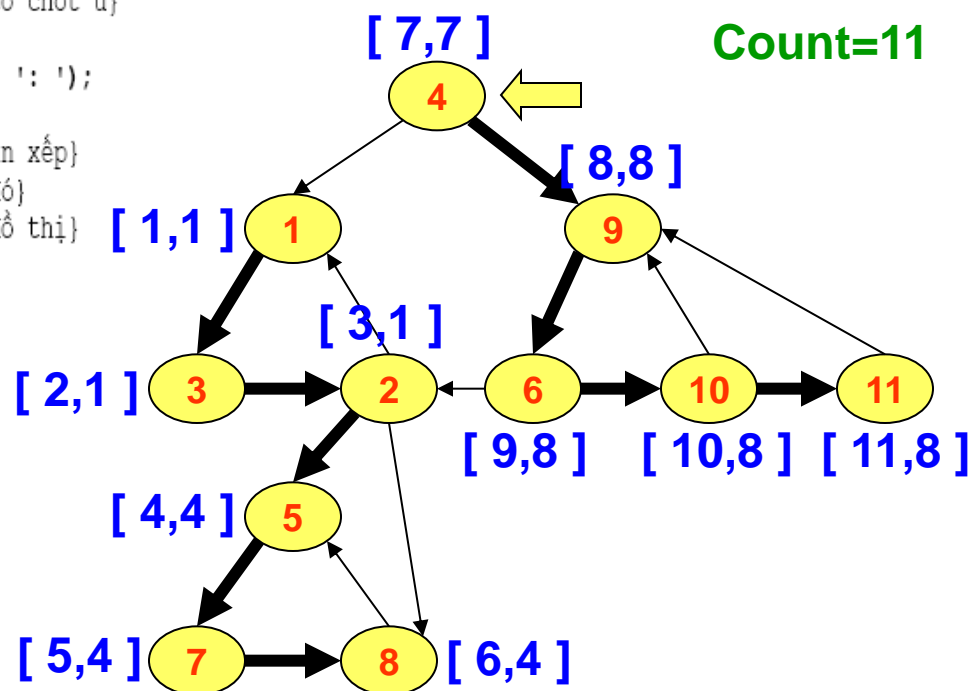
2 3 1

Component 3:

11 10 6 9

Component 4:

4



STACK



Thank You !