# TÌM KIẾM TRÊN ĐỒ THỊ

*Toán rời rạc 2*

# Nội dung

- Thuật toán tìm kiếm theo chiều sâu trên đồ thị.
- Thuật toán tìm kiếm theo chiều rộng trên đồ thị.
- Ứng dụng của thuật toán tìm kiếm theo chiều sâu.
- Ứng dụng của thuật toán tìm kiếm theo chiều rộng.

# Thuật toán tìm kiếm theo chiều sâu (Depth First Search)
# DFS

# Tư tưởng

- Trong quá trình tìm kiếm, ưu tiên “chiều sâu” hơn “chiều rộng”
  - Đi xuống sâu nhất có thể trước khi quay lại
- Bắt đầu tại một đỉnh v0 nào đó, chọn một đỉnh u bất kỳ kề với v0 và lấy nó làm đỉnh duyệt tiếp theo.
  - Cách duyệt tiếp theo được thực hiện tương tự như đối với đỉnh v0 với đỉnh bắt đầu là u.
- Để kiểm tra việc duyệt mỗi đỉnh đúng một lần, chúng ta sử dụng một mảng chuaxet[] gồm n phần tử (tương ứng với n đỉnh):
  - Nếu đỉnh thứ u đã được duyệt, phần tử tương ứng trong mảng chuaxet[u] có giá trị FALSE.
  - Ngược lại, nếu đỉnh chưa được duyệt, phần tử tương ứng trong mảng có giá trị TRUE.

# Biểu diễn thuật toán DFS

- DFS(u) có thể được mô tả bằng thủ tục đệ qui như sau:

**Thuật toán DFS (u):** *//u là đỉnh bắt đầu duyệt*

*Begin*

*<Thăm đỉnh u>; //Duyệt đỉnh u*

*chuaxet[u] := FALSE; //Xác nhận đỉnh u đã duyệt*

*for each v $\in$ Ke(u) do //Lấy mỗi đỉnh v $\in$ Ke(u).*

*If (chuaxet[v] ) then //Nếu đỉnh v chưa duyệt*

*DFS(v); //Duyệt theo chiều sâu bắt từ đỉnh v*

*EndIf;*

*EndFor;*

*End.*

# Thuật toán DFS(u) dùng ngăn xếp (khử đệ qui)

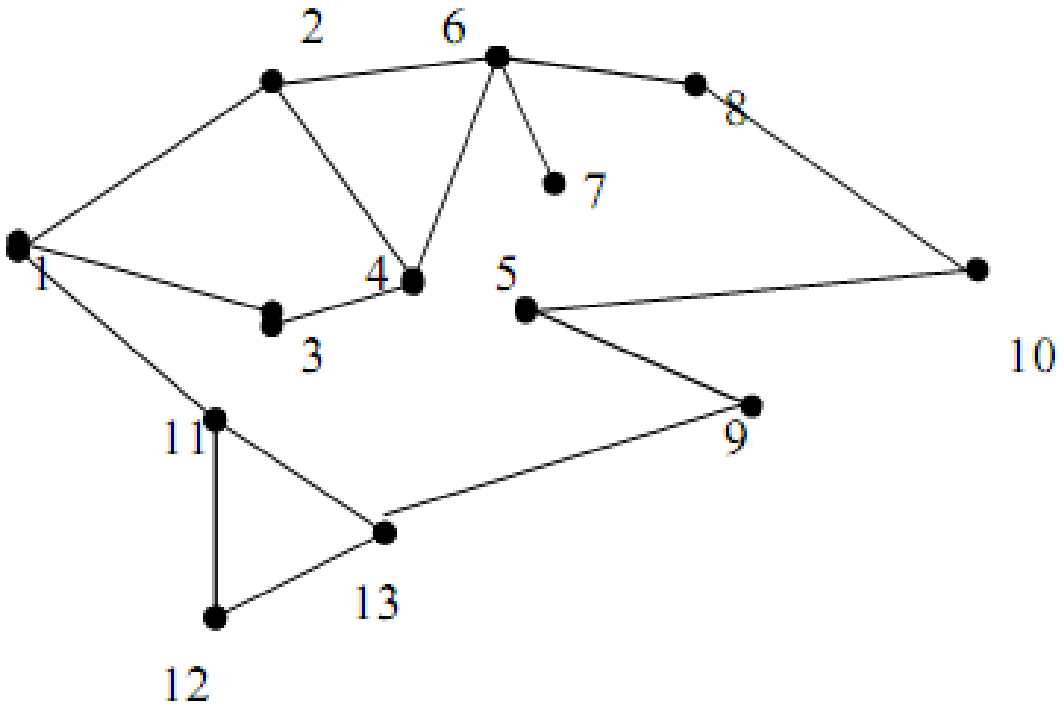
**Thuật toán DFS(u):**
**Begin**
**Bước 1 (Khởi tạo):**
stack = ∅; //*Khởi tạo stack là ∅*
Push(stack, u); //*Đưa đỉnh u vào ngăn xếp*
<Thăm đỉnh u>; //*Duyệt đỉnh u*
chuaxet[u] = False; //*Xác nhận đỉnh u đã duyệt*
**Bước 2 (Lặp) :**
while ( stack ≠ ∅ ) do
s = Pop(stack); //*Loại đỉnh ở đầu ngăn xếp*
for each t∈ Ke(s) do //*Lấy mỗi đỉnh t∈Ke(s)*
if ( chuaxet[t] ) then //*Nếu t đúng là chưa duyệt*
<Thăm đỉnh t>; // *Duyệt đỉnh t*
chuaxet[t] = False; // *Xác nhận đỉnh t đã duyệt*
Push(stack, s);//*Đưa s vào stack*
Push(stack, t); //*Đưa t vào stack*
break; //*Chỉ lấy một đỉnh t*
EndIf;
EndFor;
EndWhile;
**Bước 3 (Trả lại kết quả):**
Return(<Tập đỉnh đã duyệt>);
End.

# Độ phức tạp thuật toán DFS

- Độ phức tạp thuật toán DFS(u) phụ thuộc vào phương pháp biểu diễn đồ thị
  - Độ phức tạp thuật toán là $O(n^2)$ trong trường hợp đồ thị biểu diễn dưới dạng ma trận kề, với n là số đỉnh của đồ thị.
  - Độ phức tạp thuật toán là O(n.m) trong trường hợp đồ thị biểu diễn dưới dạng danh sách cạnh, với n là số đỉnh của đồ thị, m là số cạnh của đồ thị.
  - Độ phức tạp thuật toán là O(max(n, m)) trong trường hợp đồ thị biểu diễn dưới dạng danh sách kề, với n là số đỉnh của đồ thị, m là số cạnh của đồ thị.

# Kiểm nghiệm thuật toán DFS (1/3)

- Ví dụ 1: Cho đồ thị gồm 13 đỉnh như hình vẽ. Hãy kiểm nghiệm thuật toán DFS(1).



# Kiểm nghiệm thuật toán DFS (2/3)

| Đỉnh bắt đầu duyệt | Các đỉnh đã duyệt: chuaxet[u]=False | Các đỉnh chưa duyệt chuaxet[u]=True |
|---|---|---|
| DFS(1) | 1 | 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 |
| DFS(2) | 1, 2 | 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 |
| DFS(4) | 1, 2, 4 | 3, 5, 6, 7, 8, 9, 10, 11, 12, 13 |
| DFS(3) | 1,2,4, 3 | 5, 6, 7, 8, 9, 10, 11, 12, 13 |
| DFS(6) | 1,2,4,3, 6 | 5, 7, 8, 9, 10, 11, 12, 13 |
| DFS(7) | 1,2,4,3, 6,7 | 5, 8, 9, 10, 11, 12, 13 |
| DFS(8) | 1,2,4,3, 6,7,8 | 5, 9, 10, 11, 12, 13 |
| DFS(10) | 1,2,4,3, 6,7,8,10 | 5, 9, 11, 12, 13 |
| DFS(5) | 1,2,4,3, 6,7,8,10,5 | 9, 11, 12, 13 |
| DFS(9) | 1,2,4,3, 6,7,8,10,5,9 | 11, 12, 13 |
| DFS(13) | 1,2,4,3, 6,7,8,10,5,9,13 | 11, 12 |
| DFS(11) | 1,2,4,3, 6,7,8,10,5,9,13,11 | 12 |
| DFS(11) | 1,2,4,3, 6,7,8,10,5,9,13,11,12 | $\phi$ |

***Kết quả duyệt:*** ***1, 2, 4, 3, 6, 7, 8, 10, 5, 9, 13, 11, 12***

# Kiểm nghiệm thuật toán DFS (3/3)

| STT | Trạng thái ngăn xếp | Danh sách đỉnh được duyệt |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 1, 2 | 1, 2 |
| 3 | 1, 2, 4 | 1, 2, 4 |
| 4 | 1, 2, 4, 3 | 1, 2, 4, 3 |
| 5 | 1, 2, 4 | 1, 2, 4, 3 |
| 6 | 1, 2, 4, 6 | 1, 2, 4, 3, 6 |
| 7 | 1, 2, 4, 6, 7 | 1, 2, 4, 3, 6, 7 |
| 8 | 1, 2, 4, 6 | 1, 2, 4, 3, 6, 7 |
| 9 | 1, 2, 4, 6, 8 | 1, 2, 4, 3, 6, 7, 8 |
| 10 | 1, 2, 4, 6, 8, 10 | 1, 2, 4, 3, 6, 7, 8, 10 |
| 11 | 1, 2, 4, 6, 8, 10, 5 | 1, 2, 4, 3, 6, 7, 8, 10, 5 |
| 12 | 1, 2, 4, 6, 8, 10, 5, 9 | 1, 2, 4, 3, 6, 7, 8, 10, 5, 9 |
| 13 | 1, 2, 4, 6, 8, 10, 5, 9, 13 | 1, 2, 4, 3, 6, 7, 8, 10, 5, 9, 13 |
| 14 | 1, 2, 4, 6, 8, 10, 5, 9, 13, 11 | 1, 2, 4, 3, 6, 7, 8, 10, 5, 9, 13, 11 |
| 15 | 1, 2, 4, 6, 8, 10, 5, 9, 13, 11, 12 | 1, 2, 4, 3, 6, 7, 8, 10, 5, 9, 13, 11, 12 |
| 16- | Lần lượt bỏ các đỉnh ra khỏi ngăn xếp | |

Kết quả duyệt: 1, 2, 4, 3, 6, 7, 8, 10, 5, 9, 13, 11, 12

# Ví dụ 2

- Cho đồ thị gồm 13 đỉnh được biểu diễn dưới dạng ma trận kề như hình bên phải. Hãy cho biết kết quả thực hiện thuật toán trong DFS bắt đầu tại đỉnh u=1? Chỉ rõ trạng thái của ngăn xếp và tập đỉnh được duyệt theo mỗi bước thực hiện của thuật toán?

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

# Ví dụ 2: Kiểm nghiệm thuật toán DFS(1)

| STT | Trạng thái stack | Các đỉnh được duyệt |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 1, 2 | 1, 2 |
| 3 | 1, 2, 3 | 1, 2, 3 |
| 4 | 1, 2, 3, 4 | 1, 2, 3, 4 |
| 5 | 1, 2, 3, 4, 7 | 1, 2, 3, 4, 7 |
| 6 | 1, 2, 3, 4, 7, 5 | 1, 2, 3, 4, 7, 5 |
| 7 | 1, 2, 3, 4, 7, 5, 6 | 1, 2, 3, 4, 7, 5, 6 |
| 8 | 1, 2, 3, 4, 7, 5, 6, 12 | 1, 2, 3, 4, 7, 5, 6, 12 |
| 9 | 1, 2, 3, 4, 7, 5, 6, 12, 8 | 1, 2, 3, 4, 7, 5, 6, 12, 8 |
| 10 | **1, 2, 3, 4, 7, 5, 6, 12, 10** | 1, 2, 3, 4, 7, 5, 6, 12, 8, 10 |
| 11 | 1, 2, 3, 4, 7, 5, 6, 12, 10, 9 | 1, 2, 3, 4, 7, 5, 6, 12, 8, 10, 9 |
| 12 | 1, 2, 3, 4, 7, 5, 6, 12, 10, 9, 11 | 1, 2, 3, 4, 7, 5, 6, 12, 8, 10, 9, 11 |
| 13 | 1, 2, 3, 4, 7, 5, 6, 12, 10, 9, 11, 13 | 1, 2, 3, 4, 7, 5, 6, 12, 8, 10, 9, 11, 13 |
| 14 | ∅ | |

Kết quả duyệt DFS(1) = { 1, 2, 3, 4, 7, 5, 6, 12, 8, 10, 9, 11, 13}.

# Cài đặt thuật toán

- Hàm Init(): đọc dữ liệu theo khuôn dạng từ file dothi.in và thiết lập mảng chuaxet[u] =True (u=1, 2,..,n).
- Hàm DFS_Dequi: Cài đặt thuật toán DFS(u) bằng đệ qui.
- Hàm DFS_Stack: Cài đặt thuật toán DFS(u) dựa vào stack.

**dothi.in**

```
10
0  1  1  1  0  0  0  0  0  0
1  0  0  1  1  0  0  0  0  0
1  0  0  1  0  1  0  0  0  0
1  1  1  0  1  1  0  0  1  0
0  1  0  1  0  0  0  1  0  0
0  0  1  1  0  0  1  0  0  0
0  0  0  0  0  1  0  0  1  1
0  0  0  0  1  0  0  0  1  1
0  0  0  1  0  0  1  1  0  1
0  0  0  0  0  0  1  1  1  0
```

DFS(3) = 3, 1, 2, 4, 5, 8, 9, 7, 6, 10.

Xem code minh họa

# Thuật toán tìm kiếm theo chiều rộng (Breadth First Search)

# Tư tưởng

- Trong quá trình tìm kiếm, ưu tiên “chiều rộng” hơn “chiều sâu”
- Tìm kiếm xung quanh trước khi đi xuống sâu hơn
- Đỉnh được nạp vào hàng đợi đầu tiên là u
  - Các đỉnh kề với u là ( v1, v2, . . ., vk) được nạp vào hàng đợi nếu như nó chưa được xét đến.
- Quá trình duyệt tiếp theo được bắt đầu từ các đỉnh còn có mặt trong hàng đợi.
- Thuật toán dừng khi  hàng đợi rỗng.

# Thuật toán BFS

**Thuật toán BFS(u):**

**Bước 1(Khởi tạo):**

Queue = ∅; Push(Queue,u); chuaxet[u] = False;

**Bước 2 (Lặp):**

while (Queue ≠ ∅ ) do

s = Pop(Queue); <Thăm đỉnh s>;

for each t∈Ke(s) do

if ( chuaxet[t] ) then

Push(Queue, t); chuaxet[t] = False;

EndIf ;

EndFor ;

EndWhile ;

**Bước 3 (Trả lại kết quả) :**

Return(<Tập đỉnh được duyệt>) ;

**End.**

# Độ phức tạp thuật toán BFS

- Độ phức tạp thuật toán BFS(u) phụ thuộc vào phương pháp biểu diễn đồ thị
  - Độ phức tạp thuật toán là $O(n^2)$ trong trường hợp đồ thị biểu diễn dưới dạng ma trận kề, với n là số đỉnh của đồ thị.
  - Độ phức tạp thuật toán là O(n.m) trong trường hợp đồ thị biểu diễn dưới dạng danh sách cạnh, với n là số đỉnh của đồ thị, m là số cạnh của đồ thị.
  - Độ phức tạp thuật toán là O(max(n, m)) trong trường hợp đồ thị biểu diễn dưới dạng danh sách kề, với n là số đỉnh của đồ thị, m là số cạnh của đồ thị.

# Kiểm nghiệm thuật toán BFS (1/2)

- **Ví dụ 3.** Cho đồ thị gồm 13 đỉnh được biểu diễn dưới dạng ma trận kề như hình bên phải. Hãy cho biết kết quả thực hiện thuật toán BFS bắt đầu tại đỉnh u=1? Chỉ rõ trạng thái của hàng đợi và tập đỉnh được duyệt theo mỗi bước thực hiện của thuật toán?

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

# Kiểm nghiệm thuật toán BFS (2/2)

| STT | Trạng thái hàng đợi | Danh sách đỉnh được duyệt |
|---|---|---|
| 1 | 1 | Ø |
| 2 | 2, 3, 4 | 1 |
| 3 | 3, 4, 6 | 1, 2 |
| 4 | 4, 6, 5 | 1, 2, 3 |
| 5 | 6, 5, 7 | 1, 2, 3, 4 |
| 6 | 5, 7, 12 | 1, 2, 3, 4, 6 |
| 7 | 7, 12, 8 | 1, 2, 3, 4, 6, 5 |
| 8 | 12, 8 | 1, 2, 3, 4, 6, 5, 7 |
| 9 | 8, 10 | 1, 2, 3, 4, 6, 5, 7, 12 |
| 10 | 10 | 1, 2, 3, 4, 6, 5, 7, 12, 8 |
| 11 | 9, 11, 13 | 1, 2, 3, 4, 6, 5, 7, 12, 8, 10 |
| 12 | 11, 13 | 1, 2, 3, 4, 6, 5, 7, 12, 8, 10, 9 |
| 13 | 13 | 1, 2, 3, 4, 6, 5, 7, 12, 8, 10, 9, 11 |
| 14 | Ø | 1, 2, 3, 4, 6, 5, 7, 12, 8, 10, 9, 11, 13 |

Kết quả duyệt: 1, 2, 3, 4, 6, 5, 7, 12, 8, 10, 9, 11, 13

# Lưu ý

- Với đồ thị vô hướng
  - Nếu DFS(u) = V hoặc BFS(u) = V, ta có thể kết luận đồ thị liên thông
- Với đồ thị có hướng
  - Nếu DFS(u) = V hoặc BFS(u) = V, ta có thể kết luận đồ thị liên thông yếu

# Cài đặt thuật toán

- Hàm Init(): đọc dữ liệu theo khuôn dạng từ file dothi.in và thiết lập mảng chuaxet[u] =True (u=1, 2,..,n).
- Hàm BFS_Dequi: Cài đặt thuật toán BFS(u) bằng hàng đợi.

**dothi.in**

```
10
0  1  1  1  0  0  0  0  0  0
1  0  0  1  1  0  0  0  0  0
1  0  0  1  0  1  0  0  0  0
1  1  1  0  1  1  0  0  1  0
0  1  0  1  0  0  0  1  0  0
0  0  1  1  0  0  1  0  0  0
0  0  0  0  0  1  0  0  1  1
0  0  0  0  1  0  0  0  1  1
0  0  0  1  0  0  1  1  0  1
0  0  0  0  0  0  1  1  1  0
```

BFS(3) = 3, 1, 4, 6, 2, 5, 9, 7, 8, 10.

Xem code minh họa

# Ứng dụng của thuật toán DFS và BFS

# Vài ứng dụng cơ bản của DFS và BFS

- Duyệt tất cả các đỉnh của đồ thị.
- Duyệt tất cả các thành phần liên thông của đồ thị.
- Tìm đường đi từ đỉnh s đến đỉnh t trên đồ thị.
- Duyệt các đỉnh trụ trên đồ thị vô hướng.
- Duyệt các đỉnh trụ trên đồ thị vô hướng.
- Duyệt các cạnh cầu trên đồ thị vô hướng.
- Định chiều đồ thị vô hướng.
- Duyệt các đỉnh rẽ nhánh của cặp đỉnh s, t.
- Xác định tính liên thông mạnh trên đồ thị có hướng.
- Xác định tính liên thông yếu trên đồ thị có hướng.
- Thuật toán tìm kiếm theo chiều rộng trên đồ thị.
- Xây dựng cây khung của đồ thị vô hướng liên thông…

# Xác định thành phần liên thông của đồ thị

- Phát biểu bài toán:
  - Cho đồ thị đồ thị vô hướng G=<V,E>, trong đó V là tập đỉnh, E là tập cạnh. Xác định các thành phần liên thông của G =<V,E>?
- Thuật toán:

```
Thuật toán Duyet-TPLT:
    Bước 1 (Khởi tạo):
        solt = 0; //Khởi tạo số thành phần liên thông ban đầu là 0
    Bước 2 (Lặp):
        for ( u =1; u≤ n; u++)  do //lặp trên tập đỉnh
            if (chuaxet[u] ) then
                solt = solt + 1; //Ghi nhận số thành phần liên thông
                <Ghi nhận các đỉnh thuộc TPLT>;
                BFS (u); //DFS(u); //
            endif;
        endfor;
    Bước 3 (Trả lại kết quả):
        Return(solt);
end.
```

# Kiểm nghiệm thuật toán

- Cho đồ thị được biểu diễn dưới dạng ma trận kề:

$$\begin{vmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{vmatrix}$$

- Kết quả:
  - Thành phần liên thông 1: BFS(1) = { 1, 3, 5, 7, 9, 11, 13}.
  - Thành phần liên thông 2: BFS(2) = {2, 4, 6, 8, 10, 12}.

# Cài đặt thuật toán

- Hàm Init(): Đọc dữ liệu theo khuôn dạng và khởi đầu mảng chuaxet[u] = True ($1 \le i \le n$).
- Hàm BFS (u), DFS(u) : Hai thuật toán duyệt theo chiều rộng và duyệt theo chiều sâu được sử dụng để xác định các thành phần liên thông.

Xem code minh họa

# Tìm đường đi giữa các đỉnh trên đồ thị

- Phát biểu bài toán
  - Cho đồ thị G =<V, E> (vô hướng hoặc có hướng), trong đó V là tập đỉnh, E là tập cạnh. Tìm đường đi từ đỉnh s∈V đến đỉnh t∈V?
- Mô tả thuật toán
  - Nếu t ∈ DFS(s) hoặc t ∈ BFS(s) thì ta có thể kết luận có đường đi từ s đến t trên đồ thị, ngược lại sẽ không có đường đi
  - Để ghi nhận đường đi ta sử dụng mảng truoc[] gồm n n phần tử (n= |V|)
    - Khởi tạo ban đầu truoc[u]=0 với mọi u.
    - Mỗi khi đưa v ∈ Ke(u) vào ngăn xếp (nếu sử dụng DFS) hoặc hàng đợi (nếu sử dụng BFS) ta ghi nhận truoc[v]=u.
    - Nếu DFS và BFS không duyệt được đến đỉnh t, khi đó truoc[t]=0 thì ta kết luận không có đường đi từ s đến t.

# Tìm đường đi giữa các đỉnh trên đồ thị Dùng DFS

**Thuật toán DFS(s):**
**Begin**
**Bước 1 (Khởi tạo):**
stack = ∅; *//Khởi tạo stack là ∅*
Push(stack, s); *//Đưa đỉnh s vào ngăn xếp*
chuaxet[s] = False; *//Xác nhận đỉnh u đã duyệt*
**Bước 2 (Lặp) :**
while ( stack ≠ ∅ ) do
u = Pop(stack); *//Loại đỉnh ở đầu ngăn xếp*
for each v∈ Ke(u) do *//Lấy mỗi đỉnh u ∈Ke(v)*
if ( chuaxet[v] ) then *//Nếu v đúng là chưa duyệt*
chuaxet[v] = False; *// Xác nhận đỉnh v đã duyệt*
Push(stack, u);*//Đưa u vào stack*
Push(stack, v); *//Đưa v vào stack*
**truoc[v] = u;** *//Ghi nhận truoc[v] là u*
break; *//Chỉ lấy một đỉnh t*
EndIf;
EndFor;
EndWhile;
**Bước 3 (Trả lại kết quả):**
Return(<Tập đỉnh đã duyệt>);
End.

# Tìm đường đi giữa các đỉnh trên đồ thị
# Dùng BFS

**Thuật toán BFS(s):**

**Bước 1(Khởi tạo):**

Queue = ∅; Push(Queue,s); chuaxet[s] = False;

**Bước 2 (Lặp):**

```
while (Queue ≠ ∅ ) do
    u = Pop(Queue);
    for each v∈Ke(u) do
        if ( chuaxet[v] ) then
            Push(Queue, v);chuaxet[v]=False;truoc[v]=u;
        EndIf ;
    EndFor ;
EndWhile ;
```

**Bước 3 (Trả lại kết quả) :**

Return(<Tập đỉnh được duyệt>) ;

**End.**

# Tìm đường đi giữa các đỉnh trên đồ thị
## Ghi nhận đường đi

```
Thuật toán Ghi-Nhan-Duong-Di (s, t) {
        if ( truoc[t] == 0 ) {
                <Không có đường đi từ s đến t>;
        }
        else {
                <Đưa ra đỉnh t>; //Đưa ra trước đỉnh t
                u = truoc[t]; //u là đỉnh trước khi đến được t
                while (u ≠ s ) { //Lặp nếu u chưa phải là s
                        <Đưa ra đỉnh u>; //Đưa ra đỉnh u
                        u = truoc[u]; // Lần ngược lại đỉnh truoc[u].
                }
                <Đưa ra nốt đỉnh s>;
        }
}
```

# Kiểm nghiệm thuật toán

- Xác định đường đi từ đỉnh 1 đến đỉnh 13 trên đồ thị được biểu diễn dưới dạng ma trận kề bên:

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

# Kiểm nghiệm thuật toán DFS(1)

| STT | Trạng thái stack | Truoc[s]=? |
|---|---|---|
| 1 | 1 | 0 |
| 2 | 1, 2 | truoc[2] =1 |
| 3 | 1, 2, 3 | truoc[3] = 2 |
| 4 | 1, 2, 3, 4 | truoc[4] =3 |
| 5 | 1, 2, 3, 4, 7 | truoc[7] =4 |
| 6 | 1, 2, 3, 4, 7, 5 | truoc[5] =7 |
| 7 | 1, 2, 3, 4, 7, 5, 6 | truoc[6] =5 |
| 8 | 1, 2, 3, 4, 7, 5, 6, 12 | truoc[12] =6 |
| 9 | 1, 2, 3, 4, 7, 5, 6, 12, 8 | truoc[8] =12 |
| 10 | **1, 2, 3, 4, 7, 5, 6, 12, 10** | truoc[10] =12 |
| 11 | 1, 2, 3, 4, 7, 5, 6, 12, 10, 9 | truoc[9] =10 |
| 12 | 1, 2, 3, 4, 7, 5, 6, 12, 10, 9, 11 | truoc[11] =9 |
| 13 | 1, 2, 3, 4, 7, 5, 6, 12, 10, 9, 11, 13 | truoc[13] =11 |
| 14 | ∅ | |

**Kết quả đường đi từ đỉnh 1 đến đỉnh 13:** *13->11-9-10-12-6-5-7-4-3-2-1.*

# Kiểm nghiệm thuật toán BFS(1)

| STT | Trạng thái Queue | Truoc[s]=? |
|---|---|---|
| 1 | 1 | truoc[1]=0 |
| 2 | 2, 3, 4 | truoc[2]=1; truoc[3]=1; truoc[4]=1; |
| 3 | 3, 4, 6 | truoc[6]= 2 |
| 4 | 4, 6, 5 | truoc[5]=3 |
| 5 | 6, 5, 7 | truoc[7]= 4 |
| 6 | 5, 7, 12 | truoc[12]=6 |
| 7 | 7, 12, 8 | truoc[8]=12 |
| 8 | 12, 8 | |
| 9 | 8, 10 | truoc[10]=12; |
| 10 | 10 | |
| 11 | 9, 11, 13 | truoc[9]=10; truoc[11]=10; truoc[13]=10; |
| 12 | 11, 13 | |
| 13 | 13 | |
| 14 | ∅ | |

Kết quả đường đi: 13-10-12-6-2-1.

- **Lưu ý:**
  - Đường đi từ đỉnh s đến đỉnh t theo thuật toán BFS đi qua ít nhất các cạnh của đồ thị (có độ dài nhỏ nhất).

# Cài đặt thuật toán

- Xem code minh họa

# Tính liên thông mạnh trên đồ thị có hướng

- Phát biểu bài toán:
  - Đồ thị có hướng G=<V,E> liên thông mạnh nếu giữa hai đỉnh bất kỳ của nó đều tồn tại đường đi.
  - Cho trước đồ thị có hướng G = <V,E>. Kiểm tra xem G có liên thông mạnh hay không?
- Thuật toán:

```
Boolean      Strong-Connective ( G =<V, E>) {
        ReInit(); //∀u∈V: chuaxet[u] = True;
        for each u∈V do {//Lấy mỗi đỉnh thuộc V
                if (DFS(u)≠V ) then //Nếu DFS(u) ≠V hoặc BFS(u) ≠V
                        return(False); //Đồ thị không liên thông mạnh
                endif;
                ReInit();//Khởi tạo lại các mảng chuaxet[]
        endfor;
        return(True);//Đồ thị liên thông mạnh
}
```

# Kiểm nghiệm thuật toán

- Xác định đồ thị có hướng G =<V,E> được biểu diễn dưới dạng ma trận kề bên có liên thông mạnh hay không?

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |

# Kiểm nghiệm thuật toán kiểm tra tính liên thông mạnh

| Đỉnh u∈V | DFS(u)=?//BFS(u)=? | DFS(u) =V? |
|---|---|---|
| 1∈V | DFS(1) = 1, 6, 10, 2, 3, 9, 5, 7, 11, 8, 4, 12, 13 | Yes |
| 2∈V | DFS(2) = 2, 3, 9, 5, 7, 11, 8, 4, 1, 6, 10, 12, 13 | Yes |
| 3∈V | DFS(3) = 3, 9, 5, 7, 11, 2, 8, 4, 1, 6, 10, 12, 13 | Yes |
| 4∈V | DFS(4) = 4, 1, 6, 10, 2, 3, 9, 5, 7, 11, 8, 12, 13 | Yes |
| 5∈V | DFS(5) = 5, 7, 11, 2, 3, 9, 13, 8, 4, 1, 6, 10, 12 | Yes |
| 6∈V | DFS(6) = 6, 10, 2, 3, 9, 5, 7, 11, 8, 4, 1, 12, 13 | Yes |
| 7∈V | DFS(7) = 7, 11, 2, 3, 9, 5, 13, 8, 4, 1, 6, 10, 12 | Yes |
| 8∈V | DFS(8) = 8, 4, 1, 6, 10, 2, 3, 9, 5, 7, 11, 13, 12 | Yes |
| 9∈V | DFS(8) = 9, 5, 7, 11, 2, 3, 13, 8, 4, 1, 6, 10, 12 | Yes |
| 10∈V | DFS(10) = 10, 2, 3, 9, 5, 7, 11, 8, 4, 1, 6, 12, 13 | Yes |
| 11∈V | DFS(11) = 11, 2, 3, 9, 5, 7, 13, 8, 4, 1, 6, 10, 12 | Yes |
| 12∈V | DFS(12) = 12, 4, 1, 6, 10, 2, 3, 9, 5, 7, 11, 8, 13 | Yes |
| 13∈V | DFS(13) = 13, 9, 5, 7, 11, 2, 3, 8, 4, 1, 6, 10, 12 | Yes |

# Cài đặt thuật toán

- Thủ tục Read-Data(): Đọc ma trận kề biểu diễn đồ thị trong file dothi.in.
- Thủ tục ReInit(): Khởi tạo lại giá trị cho mảng chuaxet[].
- Thủ tục DFS(u): Thuật toán DFS bắt đầu tại đỉnh u.
- Thủ tục BFS(u): Thuật toán BFS bắt đầu tại đỉnh u.
- Hàm Strong-Conective(): Kiểm tra tính liên thông mạnh của đồ thị.

Xem code minh họa

# Duyệt các đỉnh trụ

- Phát biểu bài toán
  - Cho đồ thị vô hướng liên thông G =<V, E>. Đỉnh u∈V được gọi trụ nếu loại bỏ đỉnh u cùng với các cạnh nối với u làm tăng thành phần liên thông của đồ thị.
  - Cho đồ thị vô hướng liên thông G =<V, E> , tìm các đỉnh trụ của G?
- Mô tả thuật toán
  - Trong DFS() hoặc BFS(), thiết lập giá trị chuaxet[u] = False.
  - Quá trình duyệt sẽ được thực hiện tại một đỉnh bất kỳ v≠u.
    - Nếu DFS(v) = V\{u} hoặc BFS(v) = V\{u}:
      - Đồ thị mới nhận được cũng chỉ có 1 thành phần liên thông
      - Kết luận v không là trụ.
    - Nếu DFS(v) ≠ V\{u} hoặc BFS(v) ≠ V\{u}:
      - Khi đó v chính là trụ vì số thành phần liên thông của đồ thị đã tăng lên.

# Thuật toán duyệt các đỉnh trụ của đồ thị

```
Thuật toán    Duyet-Tru ( G =<V, E>) {
        ReInit(); //∀u∈V: chuaxet[u] = True;
        for each v∈V do {//Lấy mỗi đỉnh u tập đỉnh V
                chuaxet[v] = False; //Cấm DFS hoặc BFS duyệt vào đỉnh v
                if (DFS(u) ≠ V\{v} ) then //Duyệt DFS hoặc BFS tại đỉnh u≠v
                        <Ghi nhận v là trụ>;
                endif ;
                ReInit();//Khởi tạo lại các mảng chuaxet[]
        endfor;
}
```

# Kiểm nghiệm thuật toán

- Xác định đồ thị vô hướng G=<V,E> được biểu diễn dưới dạng ma trận kề bên có những đỉnh trụ nào?

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |

# Kiểm nghiệm thuật toán duyệt các đỉnh trụ của đồ thị

| Đỉnh v∈V | DFS(u)=?//BFS(u)=? u≠v. | DFS(u) ≠V\v? |
|---|---|---|
| 1∈V | DFS(2) = 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 | No |
| 2∈V | DFS(1) = 1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 | No |
| 3∈V | DFS(1) = 1, 2, 4 | Yes |
| 4∈V | DFS(1) = 1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13 | No |
| 5∈V | DFS(1) = 1, 2, 3, 4 | Yes |
| 6∈V | DFS(1) = 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13 | No |
| 7∈V | DFS(1) = 1, 2, 3, 4, 5, 6, 9, 8, 10, 11, 12, 13 | No |
| 8∈V | DFS(1) = 1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13 | No |
| 9∈V | DFS(1) = 1, 2, 3, 4, 5, 6, 7, 8 | Yes |
| 10∈V | DFS(1) = 1, 2, 3, 4, 5, 6, 7, 8, 9 | Yes |
| 11∈V | DFS(1) = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13 | No |
| 12∈V | DFS(1) = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13 | No |
| 13∈V | DFS(1) = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13 | No |

# Cài đặt thuật toán

- Thủ tục Read-Data(): Đọc ma trận kề biểu diễn đồ thị trong file dothi.in.
- Thủ tục ReInit(): Khởi tạo lại giá trị cho mảng chuaxet[].
- Thủ tục DFS(u): Thuật toán DFS bắt đầu tại đỉnh u.
- Thủ tục BFS(u): Thuật toán BFS bắt đầu tại đỉnh u.

Xem code minh họa

# Duyệt các cạnh cầu

- Phát biểu bài toán
  - Cho đồ thị vô hướng G =<V,E>. Cạnh e$\in$E được gọi là cầu nếu loại bỏ e làm tăng thành phần liên thông của đồ thị.
  - Cho trước đồ thị vô hướng liên thông G = <V,E>, tìm tất cả các cạnh cầu của đồ thị.
- Mô tả thuật toán
  - Đối với đồ thị được biểu diễn dưới dạng ma trận kề:
    - Loại bỏ cạnh e=(u,v)$\in$E ra khỏi đồ thị ta thực hiện bằng cách cho các phần tử A[u][v]=0 và A[v][u]=0.
  - Đối với đồ thị được biểu diễn dưới dạng danh sách kề:
    - Danh sách kề của đỉnh u ta bớt đi đỉnh v, Ke(u) = Ke(u)\{v}
    - Danh sách kề của đỉnh v ta bớt đi đỉnh u, Ke(v) = Ke(v)\{u}.

# Thuật toán duyệt các cạnh cầu của đồ thị

```
Thuật toán   Duyet-Cau ( G =<V, E>) {
        ReInit(); //∀u∈V: chuaxet[u] = True;
        for each e∈E do {//Lấy mỗi cạnh thuộc đồ thị
                E = E\{e}; //Loại bỏ cạnh e ra khỏi đồ thị
                if (DFS(1) ≠ V ) then //Nếu tăng thành phần liên thông của đồ thị
                        <Ghi nhận e là cầu>;
                endif ;
                E = E∪{e} ; //Hoàn trả lại cạnh e
                ReInit();//Khởi tạo lại các mảng chuaxet[]
        endfor;
}
```

# Kiểm nghiệm thuật toán

- Xác định đồ thị vô hướng G=<V,E> được biểu diễn dưới dạng ma trận kề bên có những cạnh cầu nào?

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |

# Kiểm nghiệm thuật toán duyệt các cạnh cầu của đồ thị

| Cạnh e∈E | DFS(u)=?//BFS(u)=? | DFS(u) ≠V? |
|---|---|---|
| (1,2)∈E | DFS(1) = 1, 3, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 | No |
| (1,3)∈E | DFS(1) = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 | No |
| (1,4)∈E | DFS(1) = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 | No |
| (2,3)∈E | DFS(1) = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 | No |
| (2,4)∈E | DFS(1) = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 | No |
| (3,4)∈E | DFS(1) = 1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 4 | No |
| (3,5)∈E | DFS(1) = 1, 2, 3, 4, 5 | Yes |
| (5,6)∈E | DFS(1) = 1, 2, 3, 4, 5, 7, 6, 8, 9, 10, 11, 12, 13 | No |
| (5,7)∈E | DFS(1) = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 | No |
| (5,8)∈E | DFS(1) = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 | No |
| (5,9)∈E | DFS(1) = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 | No |
| (6,7)∈E | DFS(1) = 1, 2, 3, 4, 5, 6, 9, 8, 7, 10, 11, 12, 13 | No |
| (6,9) ∈E | DFS(1) = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 | No |
| (7,8) ∈E | DFS(1) =1, 2, 3, 4, 5, 6, 7, 9, 8, 10, 11, 12, 13 | No |
| (8,9) ∈E | DFS(1) =1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 | No |
| (9,10) ∈E | DFS(1) =1, 2, 3, 4, 5, 6, 7, 8, 9 | Yes |
| (10,11) ∈E | DFS(1) =1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 11, 13 | No |
| (10,12) ∈E | DFS(1) =1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 | No |
| (10,13) ∈E | DFS(1) =1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 | No |
| (11,12) ∈E | DFS(1) =1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 12 | No |
| (11,13) ∈E | DFS(1) =1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 | No |
| (12,13) ∈E | DFS(1) =1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 | No |

Kết luận: cạnh (3,5), (9,10) là cầu

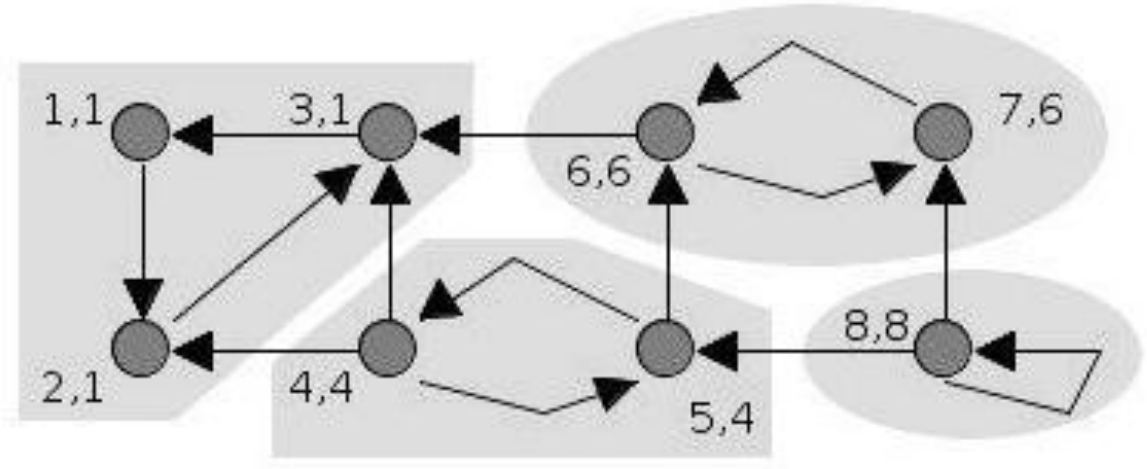
# Cài đặt thuật toán

- Thủ tục Read-Data(): Đọc ma trận kề biểu diễn đồ thị trong file dothi.in.
- Thủ tục ReInit(): Khởi tạo lại giá trị cho mảng chuaxet[].
- Thủ tục DFS(u): Thuật toán DFS bắt đầu tại đỉnh u.
- Thủ tục BFS(u): Thuật toán BFS bắt đầu tại đỉnh u.

Xem code minh họa

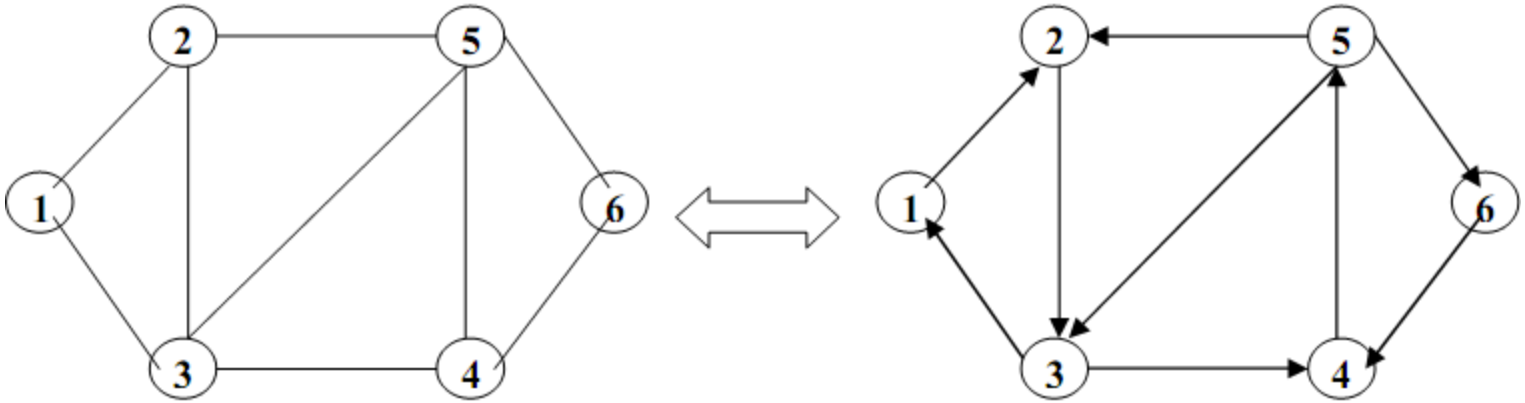
# Duyệt các thành phần liên thông mạnh của đồ thị

- Mỗi thành phần liên thông mạnh của đồ thị là một đồ thị con của G mà giữa hai đỉnh bất kỳ của đồ thị con đều có đường đi.
- Bài toán:
  - Liệt kê tất cả các thành phần liên thông mạnh của đồ thị có hướng G=<V,E>.



# Bài toán định chiều đồ thị (1/2)

- Định nghĩa
  - Phép định chiều đồ thị vô hướng liên thông là phép biến đổi đồ thị vô hướng liên thông thành đồ thị có hướng liên thông mạnh.
  - Đồ thị vô hướng G =<V,E> có thể dịch chuyển được thành đồ thị có hướng liên thông mạnh bằng cách định chiều mỗi cạnh vô hướng thành một cung có hướng được gọi là đồ thị định chiều được.



# Bài toán định chiều đồ thị (2/2)

- Định lý
  - Đồ thị vô hướng liên thông G =<V, E> định chiều được khi và chỉ khi tất cả các cạnh e$\in$E của G đều không phải là cầu.
- Bài toán.
  - Cho đồ thị vô hướng liên thông G = <V,E>. Hãy định chiều đồ thị G sao cho ta có thể nhận được đồ thị có hướng với ít nhất thành phần liên thông mạnh.
- Một số vấn đề cần quan tâm
  - Chứng minh một đồ thị vô hướng là định chiều được.
  - Viết chương trình kiểm tra một đồ thị vô hướng có định chiều được hay không?
  - Chỉ ra một phép định chiều trên một đồ thị vô hướng.

# Bài tập

- Làm các bài tập từ 1 – 10 trong Bài giảng Toán rời rạc 2.