CHƯƠNG 1. BIỂU DIỄN ĐỒ THỊ

Chương này cung cấp cho sinh viên các kiến thức liên quan đến đồ thị, thực hành các cách biểu diễn đồ thị trên máy tính; cách thực hiện các thao tác cơ bản trên đồ thị

1.1 CÁC KHÁI NIỆM CƠ BẢN

Lý thuyết đồ thị là một lĩnh vực đã có từ lâu và có nhiều ứng dụng. Những tư tưởng cơ bản của lý thuyết đồ thị được đề xuất vào những năm đầu của thế kỷ 18 bởi nhà toán học lỗi lạc người Thụy Sỹ Lenhard Eurler. Chính ông là người đã sử dụng đồ thị để giải bài toán nổi tiếng về các cái cầu ở thành phố Konigsberg.

Đồ thị được sử dụng để giải các bài toán trong nhiều lĩnh vực khác nhau. Chẳng hạn, đồ thị có thể sử dụng để xác định các mạch vòng trong vấn đề giải tích mạch điện. Chúng ta có thể phân biệt các hợp chất hóa học hữu cơ khác nhau với cùng công thức phân tử nhưng khác nhau về cấu trúc phân tử nhờ đồ thị. Chúng ta có thể xác định hai máy tính trong mạng có thể trao đổi thông tin được với nhau hay không nhờ mô hình đồ thị của mạng máy tính. Đồ thị có trọng số trên các cạnh có thể sử dụng để giải các bài toán như: Tìm đường đi ngắn nhất giữa hai thành phố trong mạng giao thông. Chúng ta cũng còn sử dụng đồ thị để giải các bài toán về lập lịch, thời khóa biểu, và phân bố tần số cho các trạm phát thanh và truyền hình…

1.1.1 Định nghĩa đồ thị:

Một **đồ thị (graph)** G = (V, E) gồm một tập không rỗng V là tập hợp các đỉnh (Vertex) của G và một tập E là tập hợp gồm các cạnh (Edge)**,** mỗi cạnh nối một hoặc hai đỉnh của đồ thị.

Lưu ý tập V và E có thể là tập vô hạn, khi đó ta gọi đồ thị là đồ thị vô hạn (infinite graph). Trong giáo trình này ta chỉ đề cập đến đồ thị hữu hạn (finite graph), là đồ thị có tập đỉnh V và tập cạnh E hữu hạn.

1.1.2 Cạnh của đồ thị:

Gồm có các loại sau: cạnh đơn, cạnh song song và khuyên.

-Cạnh đơn là cạnh duy nhất nối hai đỉnh của đồ thị.

-Cạnh song song: giữa hai đỉnh có nhiều cạnh nối thì các cạnh nối đó gọi là các cạnh song song.

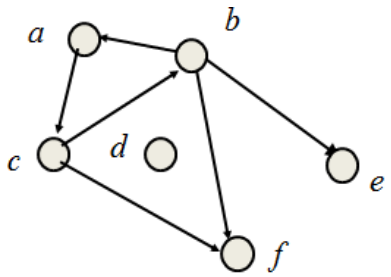
-Khuyên: Đường nối có đỉnh đầu và đỉnh cuối trùng nhau được gọi là khuyên.

Cạnh đơn Cạnh song song Khuyên

1.1.3 Đỉnh

-Đỉnh treo: đỉnh trong đồ thị có duy nhất một cạnh nối với đỉnh đó .

-Đỉnh cô lập: đỉnh không có cạnh nào nối với nó



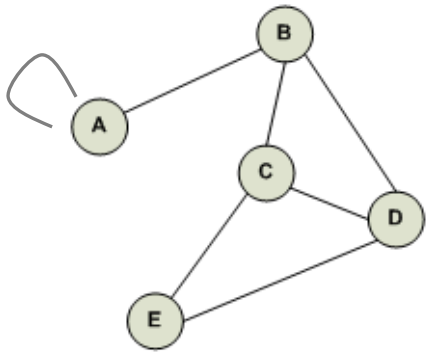
Đồ thị G

Trong đồ thị G ở trên đỉnh e là đỉnh treo và đỉnh d là đỉnh cô lập.

1.2 PHÂN LOẠI ĐỒ THỊ

1.2.1 Đồ thị vô hướng:

Một **đồ thị vô hướng (undirected graph)** G = (V, E) gồm một tập không rỗng V là tập hợp các đỉnh (Vertex) của G và một tập E là tập hợp gồm các cạnh (Edge) **không** sắp thứ tự của hai đỉnh.

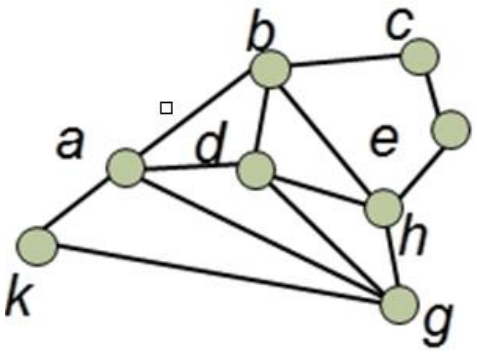


Tập đỉnh V = {A, B, C, D, E}

Tập cạnh E = {AA, AB, BA, BC, CB, BD, DB, CD, DC, CE, EC}

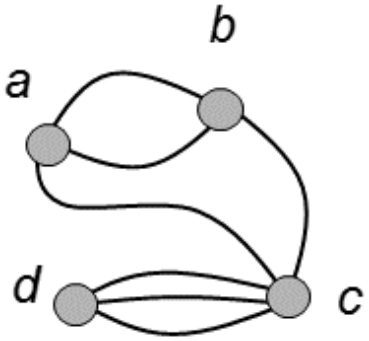
1.2.2 Đơn đồ thị vô hướng:

Đồ thị vô hướng không có cạnh song song và không có khuyên gọi là đơn đồ thị vô hướng



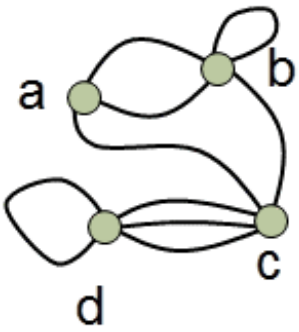
1.2.3 Đa đồ thị vô hướng:

Đồ thị vô hướng cho phép có cạnh song song nhưng không có khuyên gọi là ***đa đồ thị vô hướng***



1.2.4 Giả đồ thị:

Đồ thị vô hướng cho phép có cạnh song song và có khuyên gọi là ***giả đồ thị.***



1.2.5 Đa đồ thị có hướng:

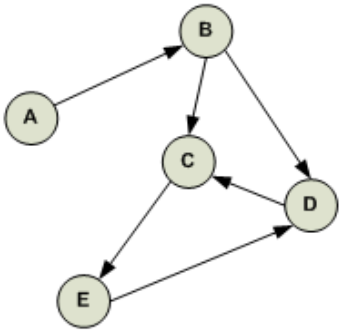
Một **đa đồ thị có hướng** G = (V,E) gồm một tập không rỗng V là tập hợp các đỉnh (Vertex) của G và một tập E là tập hợp gồm các cạnh (Edge) có sắp thứ tự của hai đỉnh, ký hiệu uv.

- Nếu uv là một cung thì ta nói u và v kề nhau. Hay Đỉnh u gọi là đỉnh đầu, đỉnh v gọi là đỉnh cuối (ngọn) của cung uv. Hay đỉnh v là đỉnh sau của u.

- Hai cung có cùng gốc và ngọn gọi là cung song song
- Cung có điểm gốc và ngọn trùng nhau gọi là khuyên

1.2.6 Đồ thị có hướng:

Đa đồ thị có hướng không chứa các cạnh song song gọi là **đồ thị có hướng**



Đồ thị có hướng

Tập đỉnh V = {A, B, C, D, E}

Tập cạnh E = {AB, BC, BD, DC, CE, ED}

1.3 BIỂU DIỄN ĐỒ THỊ

1.3.1 Dùng ma trận kề

Cho G=(V,E) với tập đỉnh V={0,1,2, . . ., n-1}, tập cạnh E={e_0, e_1, e_2,…e_{m-1}}. Ma trận kề của G là ma trận A={a_{ij}, i,j = 0,...n-1} . Với aij là số cạnh (số cung) đi từ đỉnh i đến đỉnh j

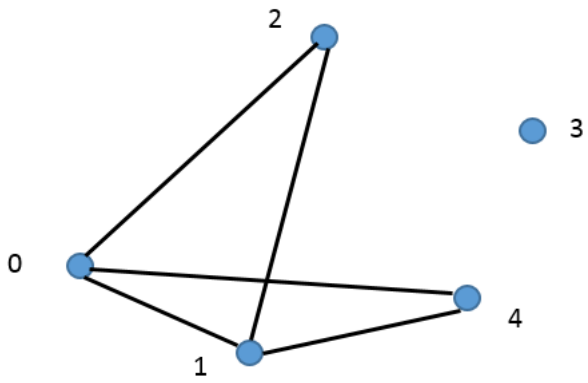
$$a_{ij} = \begin{cases} 1 \ nếu \ (i,j) \in E \\ 0 \ nếu \ (i,j) \notin E \end{cases}$$

Lưu ý là ma trận kề của một đồ thị tùy thuộc vào thứ tự liệt kê của các đỉnh. Do vậy có tới n! Ma trận liền kề khác nhau của một đồ thị có n đỉnh.

Ma trận kề của một đơn đồ thị là đối xứng, nghĩa là $a_{ij} = a_{ji}$.

Do là đơn đồ thị không có khuyên nên $a_{ij} = 0$ với i = 1, 2, 3 …n.

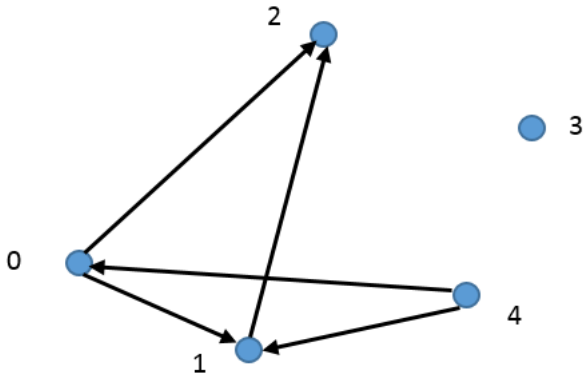
Ví dụ1: Cho đơn đồ thị vô hướng G1 như sau:



Ma trận kề của đồ thị G1 như sau:

	0	1	2	3	4
0	0	1	1	0	1
1	1	0	1	0	1
2	1	1	0	0	0
3	0	0	0	0	0
4	1	1	0	0	0

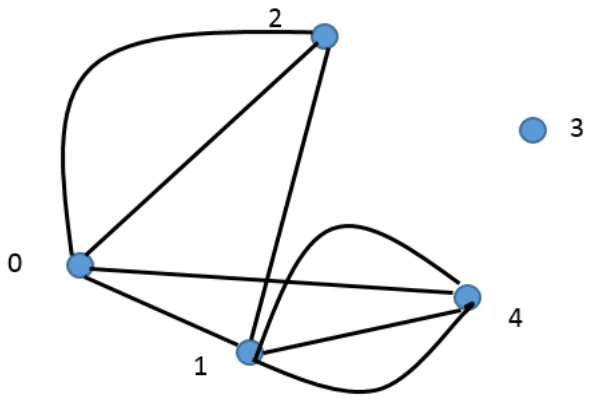
Ví dụ2: Cho đơn đồ thị vô hướng G2 như sau:



Ma trận kề của đồ thị G2 như sau:

	0	1	2	3	4
0	0	1	1	0	0
1	0	0	1	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0

Ví dụ3: Cho đa đồ thị vô hướng G3 như sau:



Ma trận kề của đồ thị G3 như sau:

	0	1	2	3	4
0	0	1	2	0	1
1	1	0	1	0	3
2	2	1	0	0	0
3	0	0	0	0	0
4	1	3	0	0	0

1.3.2 Ma trận trọng số:

Giả sử G= (V, E) là một đơn đồ thị với V= {0, 1, 2… n-1}, tập cạnh E= {e_0, e_1… e_{m-1}}.

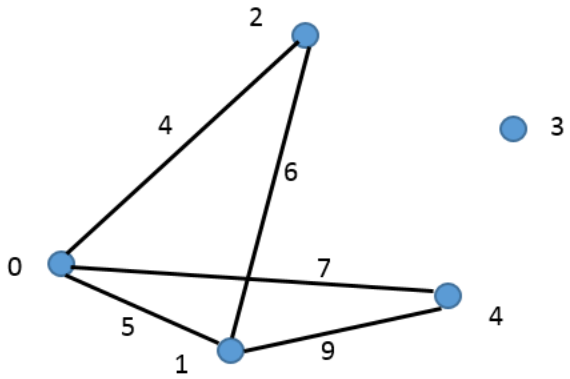
Ta gọi ma trận kề trọng số (gọi tắt là ma trận trọng số) của G là:

$$A= \{a_{ij}, i=\overline{0..n-1}, j=\overline{0..m-1}\}.$$

Trong đó: $a_{ij} = c_{ij}$ nếu (i, j) $\in$E

a_{ij} = b nếu (i, j) $\notin$E. với b là giá trị đặc biệt 0, ∞ $hoặc$ $-\infty$.

Ví dụ:



Đồ thị G

Ma trận trọng số của đồ thị G như sau:

	0	1	2	3	4
0	0	5	4	0	7
1	5	0	6	0	9
2	4	6	0	0	0
3	0	0	0	0	0
4	7	9	0	0	0

Các tính chất của ma trận kề:

-Ma trận kề của đồ thị vô hướng là ma trận đối xứng, tức là a[i,j]=a[j,i], i,j=1,2,. . .,n.

-Tổng các phần từ trên dòng i (cột j) của ma trận kề chính bằng bậc của đỉnh i (đỉnh j).

Ưu điểm lớn nhất của phương pháp biểu diễn đồ thị bằng ma trận kề (hoặc ma trận trọng số) là để trả lời câu hỏi: Hai đỉnh u,v có kề nhau trên đồ thị hay không, chúng ta chỉ phải thực hiện một phép so sánh. nhược điểm lớn nhất của phương pháp này là: không phụ thuộc vào số cạnh của đồ thị, ta luôn phải sử dụng n^2 đơn vị bộ nhớ để lưu trữ ma trận kề của nó

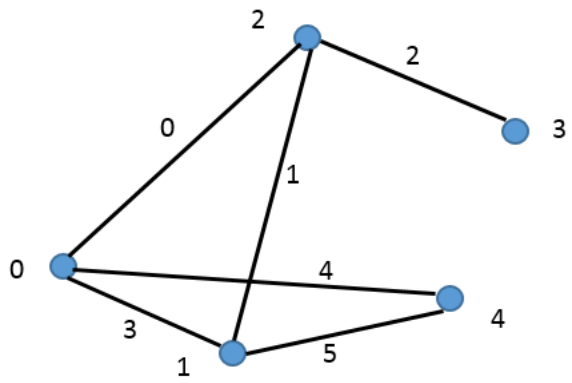
1.3.3 Danh sách cạnh:

Đối với các đồ thị thưa n đỉnh, m cạnh (m < 6n) thường dùng cách biểu diễn danh sách cạnh để tiết kiệm không gian lưu trữ

-Lưu các cạnh e=(u, v) của đồ thị trong một danh sách

-Danh sách có thể được cài đặt bằng mảng 1 chiều hoặc danh sách liên kết.

Ví dụ: đơn đồ thị vô hướng G



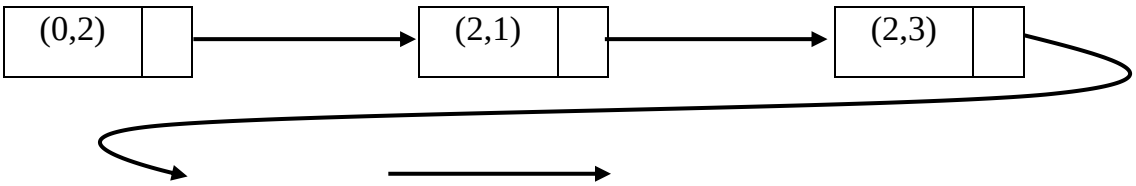
Đồ thị G

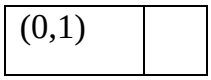
Cạnh	Đầu 1	Đầu 2
0	0	2
1	2	1
2	2	3
3	0	1
4	0	4
5	1	4

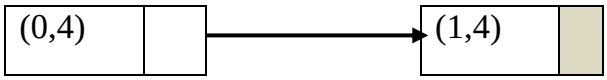
Biểu diễn bằng mảng 1 chiều:

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

Biểu diễn bằng danh sách liên kết







1.3.4 Danh sách cung:

Trong trường hợp đồ thị có hướng thì mỗi phần tử của danh sách (gọi là danh sách cung) là một cung e=(u, v). Trong đó u là đỉnh đầu, v là đỉnh cuối của cung.

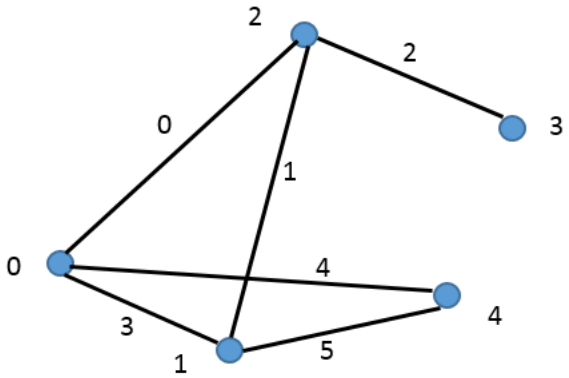
1.3.5 Danh sách kề:

Tương ứng với mỗi đỉnh v của đồ thị, ta lưu trữ một danh sách các đỉnh kề với nó.

Ke(v)= { u∈ V: (v,u)∈ E}

Biểu diễn danh sách bằng mảng 1 chiều, hoặc danh sách liên kết

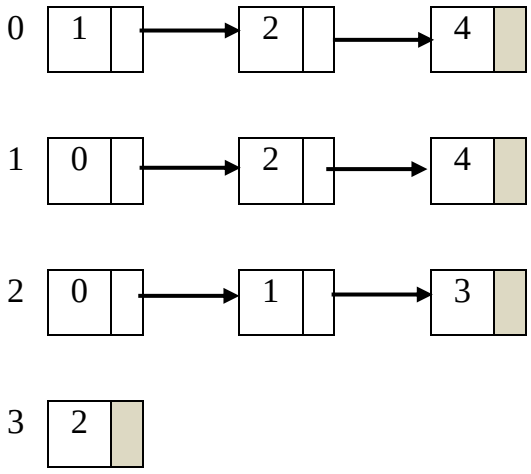
Ví dụ: đơn đồ thị vô hướng G



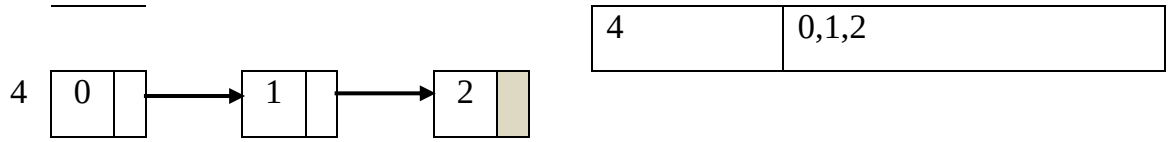
Đồ thị G

Đỉnh V	Đỉnh kề
0	1,2,4
1	0,2,4
2	0,1,3
3	2
4	0,1,2

Biểu diễn bằng danh sách kề liên kết



Đỉnh V	Đỉnh kề
0	1,2,4
1	0,2,4
2	0,1,3
3	2



Ưu điểm và hạn chế của danh sách kề

– Ưu điểm chi phí duyệt và lưu trữ khá tối ưu. Đặc biệt là danh sách kề trong mảng.

– Hạn chế: cài đặt bài toán bằng danh sách kề tương đối dài hơn so với ma trận kề và danh sách cạnh.

– Đối với từng bài toán cụ thể, tùy vào dữ liệu bài toán cho, hãy lựa chọn cách tổ chức dữ liệu phù hợp nhất, không nhất thiết lúc nào cũng tổ chức danh sách kề.

1.4 CÁC THAO TÁC CƠ BẢN TRÊN ĐỒ THỊ

Khi giải quyết các vấn đề trong lý thuyết đồ thị, ta luôn phải duyệt qua tất cả các đỉnh của đồ thị. Cho nên, cần có thuật toán duyệt toàn bộ các đỉnh của đồ thị. Thông thường sử dụng hai thuật toán duyệt theo chiều sâu và duyệt theo chiều rộng.

1.4.1 Duyệt theo chiều sâu Depth First Search – DFS

1.4.1.1 Ý tưởng thuật toán

Cho G = (V,E) là đồ thị có tập các đỉnh V và tập các cạnh E. v là một đỉnh trong V và u là đỉnh kề của v, sao cho u cũng thuộc V.

-Dán nhãn cho tất cả các đỉnh của đồ thị là 0(đánh dấu tình trạng: chưa xét).

-Chọn một đỉnh v thuộc tập V để bắt đầu duyệt. Gán nhãn đỉnh v này là 1(v đã được duyệt).

-Chọn đỉnh u trong tập V kề với đỉnh v mà nhãn là 0. Duyệt qua đỉnh u và gán nhãn u là 1.

-Tiếp tục quá trình duyệt đến khi tất cả các đỉnh đồ thị có nhãn là 1.

1.4.1.2 Mã giả thuật toán

```
Void DFS (int v)
{
        Gắn nhãn v đã duyệt;
        For (u = 0; u < n; u++)
        {
                If(u tồn tại trong danh sách kề V)
                {
                        If(u có nhãn là 0)
                        {
                                Xử lý đỉnh u; //Gắn nhãn 1
                                DFS (u);
                        }
                }
        }
}
```

1.4.1.3 Ứng dụng:

DFS thường được dùng trong các bài toán:

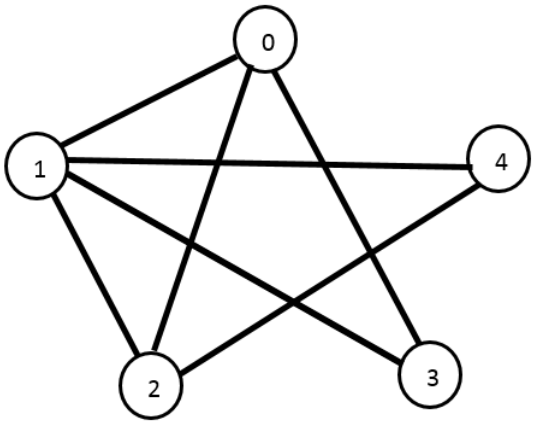
-Kiểm tra đường đi giữa 2 đỉnh

-Chia đồ thị thành các thành phần liên thông

-Xây dựng cây khung của đồ thị

-Kiểm tra xem đồ thị có chu trình hay không

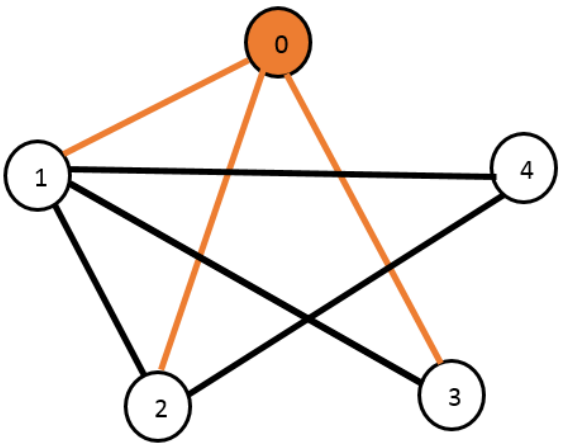
Ví dụ:



Đỉnh V	Danh sách kề	Nhãn
0	1,2,3	0
1	0,2,3,4	0
2	0,1,4	0
3	0,1	0
4	1,2	0

Chọn một đỉnh từ danh sách các đỉnh của G. Ở đây ta chọn đỉnh 0 làm đỉnh đầu tiên để bắt đầu duyệt.

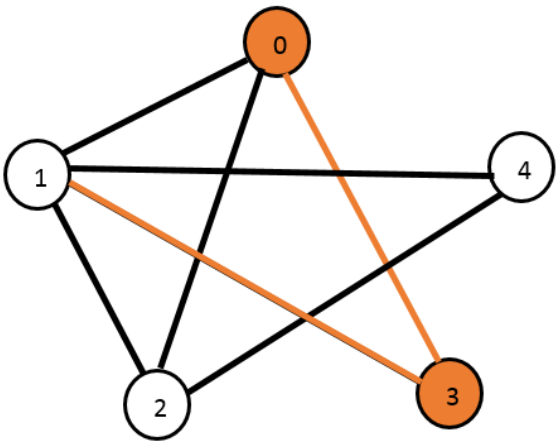
Thực hiện DFS(0):



Đỉnh V	Danh sách kề	Nhãn
0	1,2,3	1
1	0,2,3,4	0
2	0,1,4	0
3	0,1	0
4	1,2	0

Gán nhãn cho đỉnh 0 là 1 khi duyệt qua nó, tìm thấy trong các đỉnh kề của đỉnh 0 có đỉnh 3 có nhãn là 0(chưa được duyệt).Lặp lại quá trình với đỉnh 3.

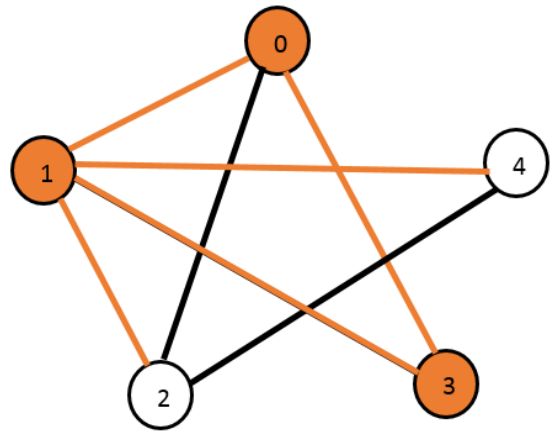
Thực hiện DFS(3):



Đỉnh V	Danh sách kề	Nhãn
0	1,2,3	1
1	0,2,3,4	0
2	0,1,4	0
3	0,1	1
4	1,2	0

Gán nhãn cho đỉnh 3 là 1 khi duyệt qua nó, tìm thấy trong các đỉnh kề của đỉnh 3 có đỉnh 1 có nhãn là 0(chưa được duyệt).Lặp lại quá trình với đỉnh 1.

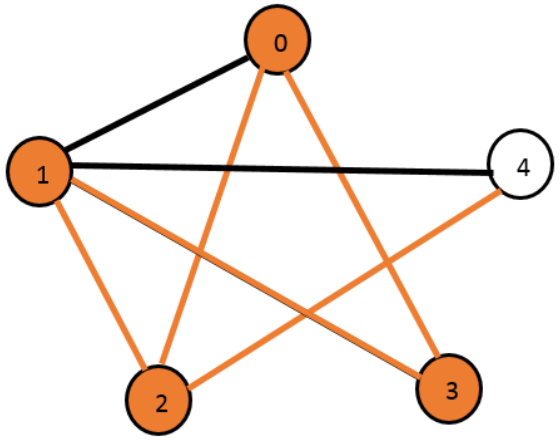
Thực hiện DFS(1):



Đỉnh V	Danh sách kề	Nhãn
0	1,2,3	1
1	0,2,3,4	1
2	0,1,4	0
3	0,1	1
4	1,2	0

Gán nhãn cho đỉnh 1 là 1 khi duyệt qua nó, tìm thấy trong các đỉnh kề của đỉnh 1 có đỉnh 2 có nhãn là 0(chưa được duyệt).Lặp lại quá trình với đỉnh 2.

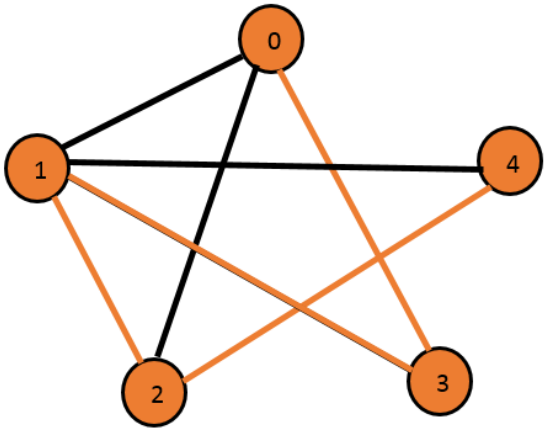
Thực hiện DFS(2):



Đỉnh V	Danh sách kề	Nhãn
0	1,2,3	1
1	0,2,3,4	1
2	0,1,4	1
3	0,1	1
4	1,2	0

Gán nhãn cho đỉnh 2 là 1 khi duyệt qua nó, tìm thấy trong các đỉnh kề của đỉnh 2 có đỉnh 4 có nhãn là 0(chưa được duyệt).Lặp lại quá trình với đỉnh 4.

Thực hiện DFS(4):



Đỉnh V	Danh sách kề	Nhãn
0	1,2,3	1
1	0,2,3,4	1
2	0,1,4	1
3	0,1	1
4	1,2	1

Gán nhãn cho đỉnh 4 là 1 khi duyệt qua nó.Tất cả các đỉnh đã được duyệt.Dừng.

1.4.2 Duyệt theo chiều rộng

1.4.2.1 Ý tưởng thuật toán

Cho G = (V,E) là đồ thị có tập các đỉnh V và tập các cạnh E. v là một đỉnh trong V và u là đỉnh kề của v, sao cho u cũng thuộc V.

Xuất phát từ 1 đỉnh v bất kỳ, lưu các đỉnh u kề của v.vào tập đang xét

Tiếp tục đem 1 đỉnh khác (từ tập đỉnh đang xét đã được lưu) ra xét và đi cho đến khi không còn đỉnh nào có thể đi.

Trong quá trình đi từ đỉnh này sang đỉnh kia, tiến hành lưu lại đỉnh cha của đỉnh kề, để khi đi ngược lại từ đỉnh kết thúc đến đỉnh xuất phát, ta có được kết quả.

1.4.2.2 Mã giả thuật toán

```
Void BFS(int v)
{
    QUEUE = ∅;
    DinhDaXet[v] = true;/*Đánh dấu true cho đỉnh v*/
    DuongDi[v] = -1;//đánh dấu đỉnh đã qua trước đỉnh v là -1
    Thêm v vào QUEUE;
    while ( QUEUE ≠ ∅ )
    {
        p = phần tử đầu QUEUE;
        Duyệt đỉnh p;
        for ( u ∈ Ke(p) )
        if ( DinhDaXet[u] == false)
        {
            Thêm u vào QUEUE;
            DinhDaXet[u] = true;/*Đánh dấu đã xét đỉnh u*/
            DuongDi[u] = p;//đánh dấu đỉnh đã qua trước đỉnh u là p
        }
```

}

}

1.4.2.3 Ứng dụng

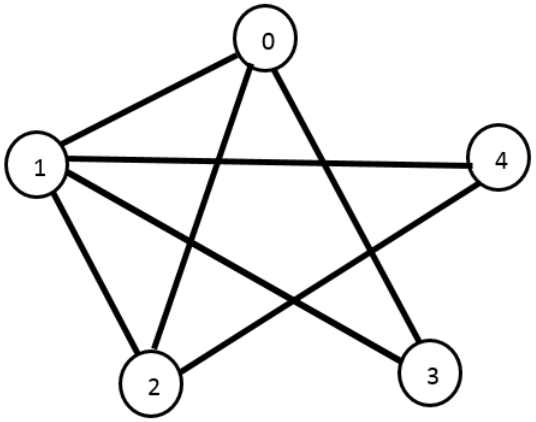
-Kiểm tra đường đi giữa 2 đỉnh.

-Chia đồ thị thành các thành phần liên thông.

-Xây dựng cây khung của đồ thị

-Tìm đường đi ngắn nhất từ 1 đỉnh đến các đỉnh còn lại

Ví dụ:



Đỉnh V	Danh sách kề
0	1,2,3
1	0,2,3,4
2	0,1,4
3	0,1
4	1,2

-Bước 0: Chuẩn bị dữ liệu:

+Mảng danh sách đỉnh kề của G

danhSachKe[]	0	1	2	3	4
	1,2,3	0,2,3,4	0,1,4	0,1	1,2

+Mảng đánh dấu các đỉnh đã xét

dinhDaXet[]	0	1	2	3	4
	false	false	false	false	false

+Mảng lưu vết đường đi

duongDi[]	0	1	2	3	4
	-1	-1	-1	-1	-1

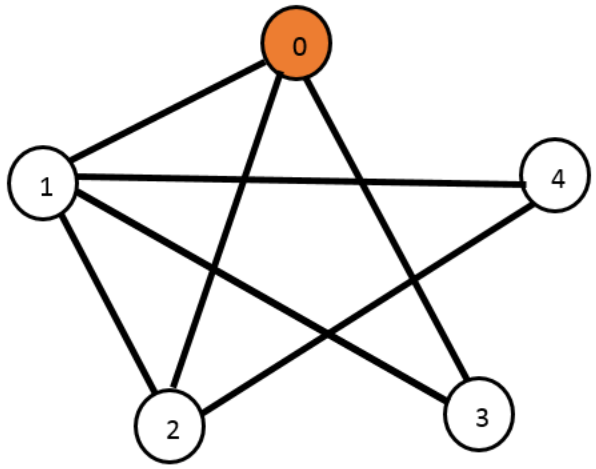
+Hàng đợi lưu các đỉnh đang xét

Queue

…

-Bước 1: Chạy thuật toán lần 1:

Đỉnh 0 là đỉnh bắt đầu đi, bỏ đỉnh 0 vào hàng đợi, đánh dấu đã xét đỉnh 0.



+Mảng danh sách đỉnh kề của G

danhSachKe[]	0	1	2	3	4
	1,2,3	0,2,3,4	0,1,4	0,1	1,2

+Mảng đánh dấu các đỉnh đã xét

dinhDaXet[]	0	1	2	3	4
	true	false	false	false	false

+Mảng lưu vết đường đi

duongDi[]	0	1	2	3	4
	-1	-1	-1	-1	-1

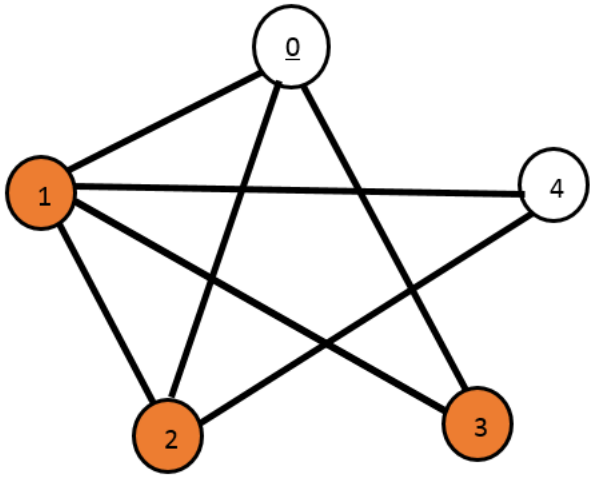
+Hàng đợi lưu các đỉnh đang xét

Queue

0

-Bước 2: Chạy thuật toán lần 2:

Lấy đỉnh 0 ra xét, tìm các đỉnh chưa xét kề với đỉnh 0 (1,2,3) bỏ vào hàng đợi; đánh dấu đã xét các đỉnh 1,2,3; lưu vết đường đi trước khi đến 1,2,3 là 0.



+Mảng danh sách đỉnh kề của G

danhSachKe[]	0	1	2	3	4
	1,2,3	0,2,3,4	0,1,4	0,1	1,2

+Mảng đánh dấu các đỉnh đã xét

dinhDaXet[]	0	1	2	3	4
	true	true	true	true	false

+Mảng lưu vết đường đi

duongDi[]	0	1	2	3	4
	-1	0	0	0	-1

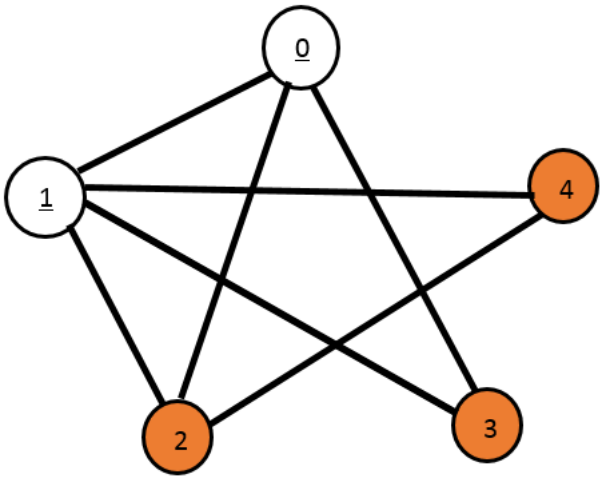
+Hàng đợi lưu các đỉnh đang xét

Queue

1	2	3

-Bước 3: Chạy thuật toán lần 3:

Lấy đỉnh 1 ra xét, tìm các đỉnh chưa xét kề với đỉnh 1 (4) bỏ vào hàng đợi; đánh dấu đã xét đỉnh 4; lưu vết đường đi trước khi đến 4 là 1.



+Mảng danh sách đỉnh kề của G

danhSachKe[]	0	1	2	3	4
	1,2,3	0,2,3,4	0,1,4	0,1	1,2

+Mảng đánh dấu các đỉnh đã xét

dinhDaXet[]	0	1	2	3	4
	true	true	true	true	true

+Mảng lưu vết đường đi

duongDi[]	0	1	2	3	4
	-1	0	0	0	1

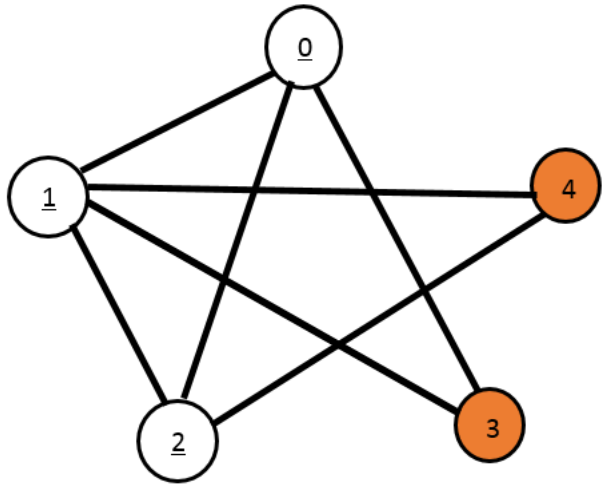
+Hàng đợi lưu các đỉnh đang xét

Queue

2	3	4

-Bước 4: Chạy thuật toán lần 4:

Lấy đỉnh 2 ra xét, tìm các đỉnh chưa xét kề với đỉnh 2 (không có) bỏ vào hàng đợi



+Mảng danh sách đỉnh kề của G

danhSachKe[]	0	1	2	3	4
	1,2,3	0,2,3,4	0,1,4	0,1	1,2

+Mảng đánh dấu các đỉnh đã xét

dinhDaXet[]	0	1	2	3	4
	true	true	true	true	true

+Mảng lưu vết đường đi

duongDi[]

0	1	2	3	4
-1	0	0	0	1

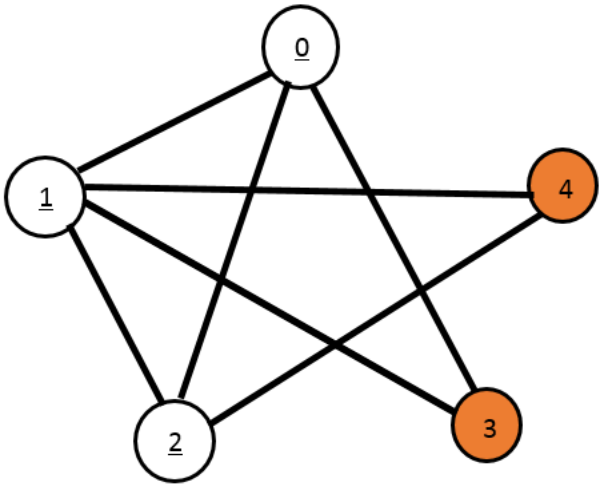
+Hàng đợi lưu các đỉnh đang xét

Queue

3	4

-Bước 5: Chạy thuật toán lần 5:

Lấy đỉnh 3 ra xét, tìm các đỉnh chưa xét kề với đỉnh 3 (không có) bỏ vào hàng đợi



+Mảng danh sách đỉnh kề của G

danhSachKe[]

0	1	2	3	4
1,2,3	0,2,3,4	0,1,4	0,1	1,2

+Mảng đánh dấu các đỉnh đã xét

dinhDaXet[]

0	1	2	3	4
true	true	true	true	true

+Mảng lưu vết đường đi

duongDi[]	0	1	2	3	4
	-1	0	0	0	1

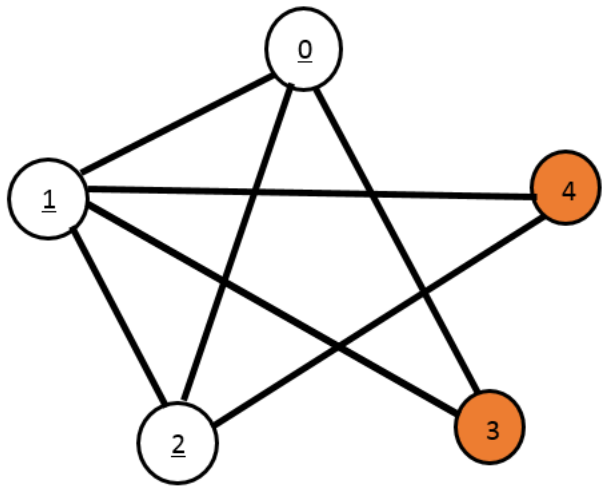
+Hàng đợi lưu các đỉnh đang xét

Queue

4

-Bước 6: Chạy thuật toán lần 6:

Lấy đỉnh 4 ra xét, tìm các đỉnh chưa xét kề với đỉnh 4 (không có) bỏ vào hàng đợi



+Mảng danh sách đỉnh kề của G

danhSachKe[]	0	1	2	3	4
	1,2,3	0,2,3,4	0,1,4	0,1	1,2

+Mảng đánh dấu các đỉnh đã xét

dinhDaXet[]	0	1	2	3	4
	true	true	true	true	true

+Mảng lưu vết đường đi

duongDi[]	0	1	2	3	4
	-1	0	0	0	1

+Hàng đợi lưu các đỉnh đang xét

Queue

…

-Dừng thuật toán

Kết quả:

duongDi[]	0	1	2	3	4
	-1	0	0	0	1

1.4.3 Kiểm tra tính liên thông của đồ thị:

1.4.3.1 Tính liên thông của đồ thị:

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

Định lý:

-Định lý về đường đi giữa 2 đỉnh bậc lẻ:

Nếu một đồ thị G (không quan tâm liên thông hay không) có đúng 2 đỉnh bậc lẻ, chắc chắn sẽ có một đường đi nối 2 đỉnh này.

-Định lý về số cạnh của đồ thị (Định Lý bắt tay)

Số cạnh tối đa của một đơn đồ thị không liên thông G gồm n đỉnh và k thành phần là: (n-k)*(n-k+1)/2

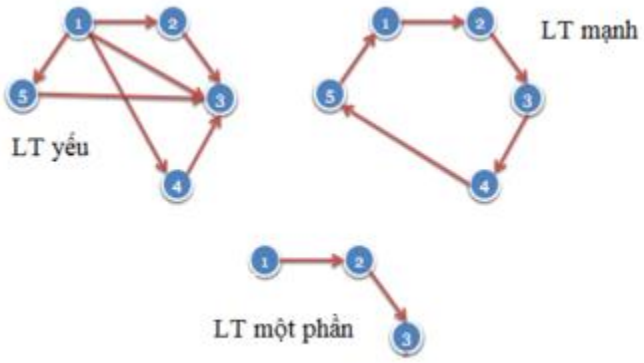
Tính chất:

Đồ thị liên thông có hướng:

-*Liên thông mạnh* (strongly connected): Đồ thị có hướng gọi là liên thông mạnh nếu có đường đi từ a tới b và từ b tới a với mọi cặp đỉnh a và b của đồ thị.

-*Liên thông yếu* (weakly connected): Đồ thị có hướng gọi là liên thông yếu nếu có đường đi giữa 2 đỉnh bất kỳ của đồ thị vô hướng tương ứng với đồ thị đã cho. Tức là hủy bỏ các hướng của các cạnh trong đồ thị

-*Liên thông một phần* (unilaterally connected): Đồ thị có hướng gọi là liên thông một phần nếu với mọi cặp đỉnh a, b bất kỳ, có ít nhất một đỉnh đến được đỉnh còn lại



Đỉnh khớp (cut vertex/ articulation point): của một đồ thị vô hướng là đỉnh mà nếu xóa đỉnh này khỏi đồ thị và các cạnh nối đến nó thì số thành phần liên thông của đồ thị sẽ tăng thêm.

Cạnh cầu (bridge): của một đồ thị vô hướng là cạnh mà nếu xóa đi khỏi đồ thị thì số thành phần liên thông của đồ thị sẽ tăng thêm.

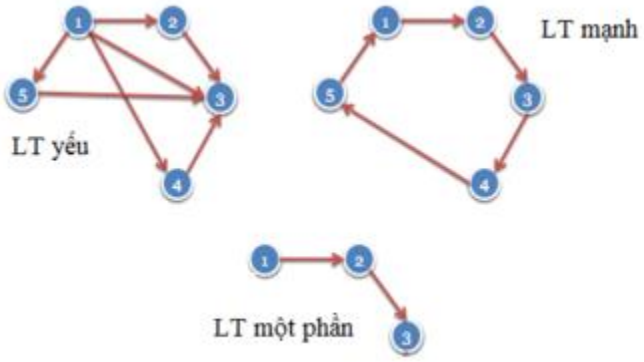
Đồ thị song liên thông (biconnectivity): là đồ thị không chứa đỉnh khớp.

Đồ thị liên thông có hướng:

-*Liên thông mạnh* (strongly connected): Đồ thị có hướng gọi là liên thông mạnh nếu có đường đi từ a tới b và từ b tới a với mọi cặp đỉnh a và b của đồ thị. Xem thêm thành phần liên thông mạnh.

-*Liên thông yếu* (weakly connected): Đồ thị có hướng gọi là liên thông yếu nếu có đường đi giữa 2 đỉnh bất kỳ của đồ thị vô hướng tương ứng với đồ thị đã cho. Tức là hủy bỏ các hướng của các cạnh trong đồ thị

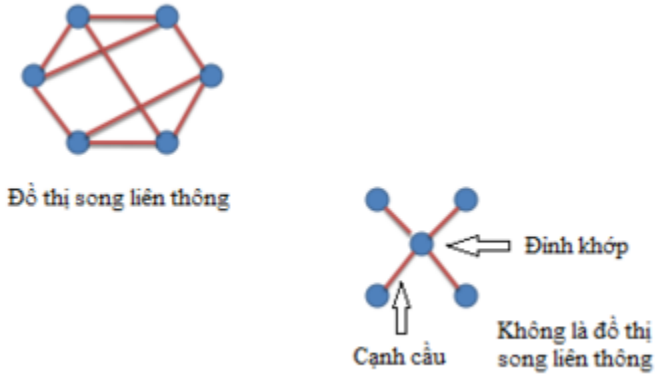
-*Liên thông một phần* (unilaterally connected): Đồ thị có hướng gọi là liên thông một phần nếu với mọi cặp đỉnh a, b bất kỳ, có ít nhất một đỉnh đến được đỉnh còn lại



Đỉnh khớp (cut vertex/ articulation point): của một đồ thị vô hướng là đỉnh mà nếu xóa đỉnh này khỏi đồ thị và các cạnh nối đến nó thì số thành phần liên thông của đồ thị sẽ tăng thêm.

Cạnh cầu (bridge): của một đồ thị vô hướng là cạnh mà nếu xóa đi khỏi đồ thị thì số thành phần liên thông của đồ thị sẽ tăng thêm.

Đồ thị song liên thông (biconnectivity): là đồ thị không chứa đỉnh khớp.



1.4.3.2 Ý tưởng thuật toán:

Sử dụng kỹ thuật loang.

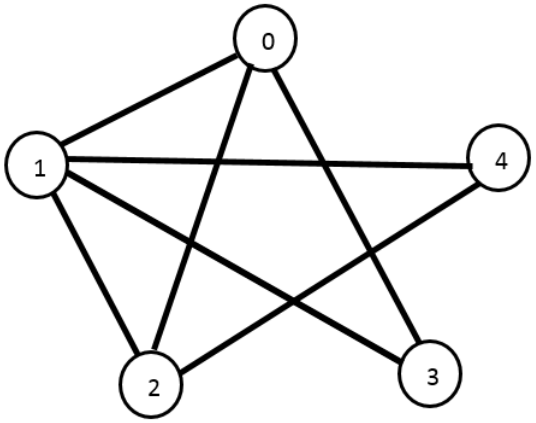
Bước 1: Xuất phát từ một đỉnh bất kỳ của đồ thị. Ta đánh dấu đỉnh xuất phát và chuyển sang bước 2.

Bước 2: Từ một đỉnh i đã đánh dấu, ta đánh dấu tất cả các đỉnh j nếu A[i,j] = 1 và j chưa được đánh dấu và chuyển sang bước 3.

Bước 3: Thực hiện bước 2 cho đến khi không còn thực hiện được nữa chuyển sang bước 4.

Bước 4: Kiểm tra nếu số đỉnh đánh dấu nhỏ hơn n (tồn tại ít nhất một đỉnh chưa được đánh dấu) đồ thị sẽ không liên thông và ngược lại đồ thị liên thông.

Ví dụ: kiểm tra tính liên thông của đồ thị



Đỉnh V	Danh sách kề
0	1,2,3
1	0,2,3,4
2	0,1,4
3	0,1
4	1,2

-Bước 0: Chuẩn bị dữ liệu:

+Mảng danh sách đỉnh kề của G

danhSachKe[]	0	1	2	3	4
	1,2,3	0,2,3,4	0,1,4	0,1	1,2

+Mảng đánh dấu các đỉnh đã xét khi chưa xét gán giá trị 0, ngược lại gán giá trị 1

dinhDaXet[]	0	1	2	3	4
	0	0	0	0	0

+Hàng đợi lưu các đỉnh đã xét

Queue

…

-Bước 1:

+Đánh dấu đỉnh đầu tiên của đồ thị (dinhDaXet[0]=1).

dinhDaXet[]	0	1	2	3	4
	1	0	0	0	0

+Hàng đợi lưu các đỉnh đang xét

Queue

0

-Bước 2+3:

+Đỉnh 0 đã được đánh dấu, xét đỉnh 0, các đỉnh kề của 0 (1,2,3),và chưa đánh dấu đã xét→ đánh dấu

dinhDaXet[]	0	1	2	3	4

1	1	1	1	0

Hàng đợi lưu các đỉnh đang xét

Queue

1	2	3

+Đỉnh 1 đã được đánh dấu, xét đỉnh 1,các đỉnh kề của 1(0,2,3,4), có đỉnh 4 chưa đánh dấu đã xét→ đánh dấu

dinhDaXet[] 0 1 2 3 4

1	1	1	1	1

Hàng đợi lưu các đỉnh đang xét

Queue

2	3	4

+Đỉnh 2 đã được đánh dấu, xét đỉnh 2,các đỉnh kề của 2(0,1,4), đã đánh dấu

dinhDaXet[] 0 1 2 3 4

1	1	1	1	1

Hàng đợi lưu các đỉnh đang xét

Queue

3	4

+Đỉnh 3 đã được đánh dấu, xét đỉnh 3,các đỉnh kề của 3(0,1), đã đánh dấu

dinhDaXet[] 0 1 2 3 4

1	1	1	1	1

Hàng đợi lưu các đỉnh đang xét

Queue

4

+Đỉnh 4 đã được đánh dấu, xét đỉnh 4,các đỉnh kề của 4(1,2), đã đánh dấu

dinhDaXet[]	0	1	2	3	4
	1	1	1	1	1

Hàng đợi lưu các đỉnh đang xét

Queue

…

Dừng xét.

Bước 4:

Nếu tất cả phần tử dinhDaXet[] = 1: đồ thị liên thông, ngược lại: không liên thông