CHƯƠNG 2
TÍNH LIÊN THÔNG CỦA ĐỒ THỊ

Trong rất nhiều trường hợp ta cần phải đi qua hay ghé thăm các đỉnh của đồ thị một cách có hệ thống, việc đi qua các đỉnh của đồ thị một cách có hệ thống như vậy được gọi là phép duyệt đồ thị. Nhờ vào phép duyệt đồ thị, ta có thể xác định được có đường đi từ đỉnh này đến đỉnh kia trong đồ thị hay không, ta có thể xác định được một đồ thị có liên thông hay không, hoặc đếm được số thành phần liên thông của đồ thị.

2.1. PHÉP DUYỆT ĐỒ THỊ

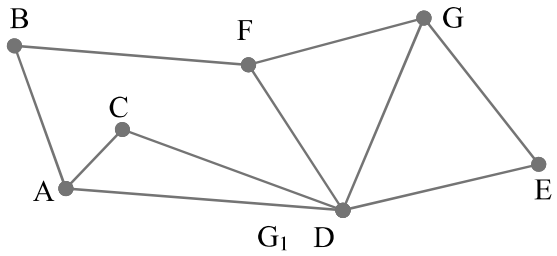
Phần này sẽ trình bày hai phép duyệt đồ thị phổ biến là duyệt theo chiều sâu và duyệt theo chiều rộng.

2.1.1. Duyệt theo chiều sâu (depth-first search)

Giả sử ta có một đồ thị G gồm n đỉnh, với các đỉnh ban đầu được đánh dấu là chưa duyệt (unvisited). Từ một đỉnh v nào đó ta bắt đầu duyệt như sau: đánh dấu v đã được duyệt, với mỗi đỉnh w kề với v chưa được duyệt, ta thực hiện đệ quy quá trình trên cho w. Cách duyệt này được gọi là duyệt theo chiều sâu vì nó sẽ duyệt theo một hướng nào đó sâu nhất có thể được.

Ví dụ 1:

Duyệt đồ thị G_1 theo chiều sâu



Hình 2.1. Minh họa duyệt đồ thị vô hướng theo chiều sâu

Giải:

Xét đồ thị G_1, giả sử bắt đầu duyệt từ đỉnh A:

Đỉnh A có các đỉnh kề là B, C, D, theo thứ tự này thì đỉnh B được duyệt.

Đỉnh B có một đỉnh kề chưa duyệt là F, nên đỉnh F được duyệt.

Đỉnh F có các đỉnh kề chưa duyệt là D và G, theo thứ tự này thì đỉnh D được duyệt.

Đỉnh D có các đỉnh kề chưa duyệt là C, E, G, theo thứ tự này thì đỉnh C được duyệt.

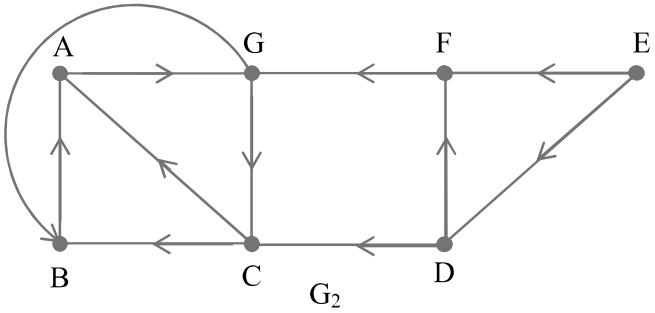
Các đỉnh kề với C đều đã được duyệt nên ta tiếp tục duyệt đỉnh E.

Đỉnh E có một đỉnh kề chưa duyệt là G nên đỉnh G được duyệt.

Vậy thứ tự các đỉnh của đồ thị G_1 được duyệt theo chiều sâu là ABFDCEG.

Ví dụ 2:

Duyệt đồ thị G_2 theo chiều sâu



Hình 2.2. Minh họa duyệt đồ thị có hướng theo chiều sâu

Giải:

Xét đồ thị G_2, giả sử bắt đầu duyệt từ đỉnh A:

Đỉnh A có một đỉnh kề là G nên đỉnh G được duyệt.

Đỉnh G có hai đỉnh kề với nó là B và C, theo thứ tự này thì đỉnh được duyệt kế tiếp là đỉnh B.

Đỉnh B có một đỉnh kề là A, mà A đã được duyệt. Nên ta tiếp tục xét đỉnh kề với G mà chưa được duyệt là đỉnh C.

Đỉnh C không có đỉnh kề nên ta tiếp tục xét các đỉnh khác. Còn lại các đỉnh chưa duyệt là D, E, F, theo thứ tự này thì đỉnh D được duyệt.

Tiếp theo ta duyệt đỉnh F.

Còn lại đỉnh E chưa duyệt, ta tiếp tục duyệt E và kết thúc.

Vậy thứ tự các đỉnh của đồ thị G_2 được duyệt theo chiều sâu là A G B C D F E.

✪ Giải thuật duyệt theo chiều sâu một đồ thị có thể được trình bày như bên dưới, trong đó ta dùng một mảng mark có n phần tử để đánh dấu các đỉnh của đồ thị là đã duyệt hay chưa.

```
// đánh dấu tất cả các đỉnh là chưa duyệt (n là số đỉnh)
for(v=0; v<n; v++)
    mark[v] = unvisited;
// duyệt theo chiều sâu từ đỉnh đầu tiên
```

```
for(v=0; v<n; v++)
        if(mark[v] == unvisited)
                DFS(v); // duyệt theo chiều sâu từ đỉnh v
```

Hàm DFS trong giải thuật trên viết như sau:

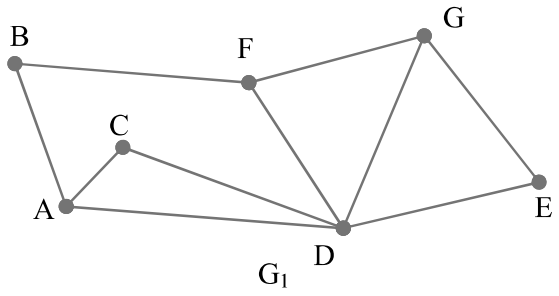
```
void DFS(vertex v)
{
        vertex w;
        mark[v] = visited;
        for (mỗi đỉnh w là đỉnh kề với v)
                if (mark[w] == unvisited)
                        DFS(w);
}
```

2.1.2. Duyệt theo chiều rộng (breadth-first search)

Giả sử ta có một đồ thị G gồm n đỉnh, với các đỉnh ban đầu được đánh dấu là chưa duyệt (unvisited). Từ một đỉnh v nào đó ta bắt đầu duyệt như sau: đánh dấu v đã được duyệt, rồi duyệt tất cả các đỉnh kề với v.

Khi ta duyệt đỉnh v rồi đến đỉnh w thì các đỉnh kề của v được duyệt trước các đỉnh kề của w, nên ta sử dụng một hàng đợi (queue) để lưu trữ các đỉnh theo thứ tự được duyệt để có thể duyệt các đỉnh kề với chúng.

Ví dụ 1: Duyệt đồ thị G_1 theo chiều rộng



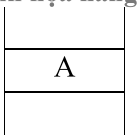
Hình 2.3. Minh họa duyệt đồ thị vô hướng theo chiều rộng

Giải:

Bắt đầu duyệt từ đỉnh A, tiếp theo duyệt tất cả các đỉnh kề với đỉnh A đó là B, C, D theo thứ tự đó. Tiếp đến là duyệt tất cả các đỉnh kề với B, C, D theo đúng thứ tự đó là F, E, G. Có thể minh họa hoạt động của hàng đợi trong phép duyệt trên như sau:

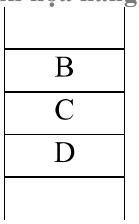
Đầu tiên ta duyệt đỉnh A và đưa A vào hàng đợi:

Bảng 2.1. Minh họa hàng đợi (bước 1)



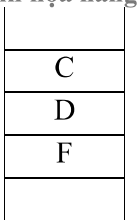
Tiếp theo duyệt tất cả các đỉnh kề với A mà chưa được duyệt đó là B, C, D. Xóa đỉnh A khỏi hàng đợi, đánh dấu duyệt các đỉnh B, C, D và đưa vào hàng đợi:

Bảng 2.2. Minh họa hàng đợi (bước 2)



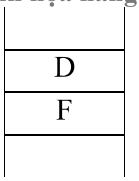
Kế đến B được lấy ra khỏi hàng đợi và các đỉnh kề với B mà chưa được duyệt đó là F. Đánh dấu duyệt F và đưa F vào hàng đợi:

Bảng 2.3. Minh họa hàng đợi (bước 3)



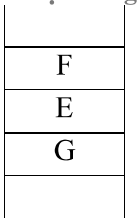
Đỉnh C được lấy ra khỏi hàng và xét duyệt tiếp các đỉnh kề với C mà chưa duyệt. Không có đỉnh nào như vậy nên bước này không có thêm đỉnh nào được duyệt và trạng thái của hàng đợi như sau:

Bảng 2.4. Minh họa hàng đợi (bước 4)



Tiếp theo D được lấy ra khỏi hàng và duyệt các đỉnh kề chưa duyệt của D là E, G. Nên E, G được đưa vào hàng đợi:

Bảng 2.5. Minh họa hàng đợi (bước 5)



Tiếp tục F được lấy ra khỏi hàng. Không có đỉnh nào kề với F chưa được duyệt. Vậy không duyệt thêm đỉnh nào.

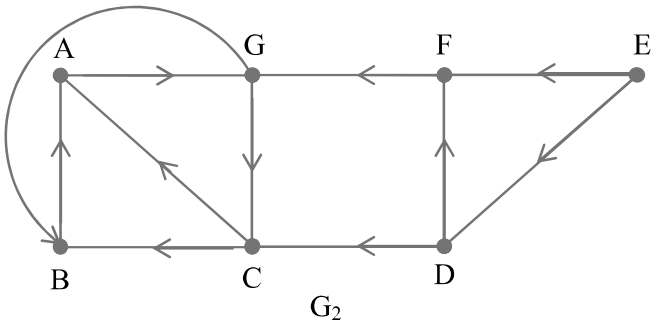
Bảng 2.6. Minh họa hàng đợi (bước 6)

E
G

Đỉnh E và đỉnh G lần lượt được lấy khỏi hàng đợi. Hàng đợi trở nên rỗng và giải thuật kết thúc.

Vậy thứ tự các đỉnh của đồ thị G_1 được duyệt theo chiều rộng là A B C D F E G.

Ví dụ 2: Duyệt đồ thị G_2 theo chiều rộng



G_2

Hình 2.4. Minh họa duyệt đồ thị có hướng theo chiều rộng

Giải:

Giả sử bắt đầu duyệt từ A. A chỉ có một đỉnh kề là G nên ta duyệt G. Tiếp theo duyệt tất cả các đỉnh kề của G là B, C. Tiếp theo duyệt tất cả các đỉnh kề với B, C theo thứ tự đó. Các đỉnh kề với B và C đều đã được duyệt, nên ta tiếp tục duyệt các đỉnh chưa được duyệt. Các đỉnh chưa được duyệt là D, E, F. Ta duyệt D, tiếp đến là F và cuối cùng là E.

Vậy thứ tự các đỉnh của đồ thị G_2 được duyệt theo chiều rộng là A G B C D F E.

✪ Giải thuật duyệt theo chiều rộng một đồ thị có thể được trình bày như bên dưới, trong đó ta dùng một mảng mark có n phần tử để đánh dấu các đỉnh của đồ thị là đã duyệt hay chưa.

```
// đánh dấu tất cả các đỉnh là chưa duyệt (n là số đỉnh)
for(v=0; v<n; v++)
        mark[v] = unvisited;
// duyệt theo chiều sâu từ đỉnh đầu tiên
for(v=0; v<n; v++)
        if(mark[v] == unvisited)
                BFS(v); // duyệt theo chiều sâu từ đỉnh v
```

Hàm BFS trong giải thuật trên viết như sau:

```
void BFS(vertex v)
{
    QUEUE of vertex Q;
    vertex x, y;
    mark[v] = visited;
    ENQUEUE(v, Q);
    while(!EMPTY_QUEUE(Q)){
        x = FRONT(Q);
        DEQUEUE(Q);
        for (mỗi đỉnh y là đỉnh kề với x)
            if (mark[y] == unvisited){
                mark[y] = visited;
                ENQUEUE(y, Q);
            }
    }
}
```

2.2. ĐƯỜNG ĐI, CHU TRÌNH

Đường đi độ dài n từ đỉnh u đến đỉnh v, trong đó n là số nguyên dương, trên đồ thị vô hướng G = (V, E) là dãy

$$x_0, x_1, \ldots, x_{n-1}, x_n$$

trong đó $u = x_0$, $v = x_n$, $(x_i , x_{i+1}) \in E$, i = 0, 1, 2, …, n-1.

Đường đi nói trên còn có thể biểu diễn dưới dạng dãy các cạnh:

$$(x_0, x_1), (x_1, x_2), \ldots, (x_{n-1}, x_n)$$

Đỉnh u gọi là đỉnh đầu, còn đỉnh v gọi là đỉnh cuối của đường đi.

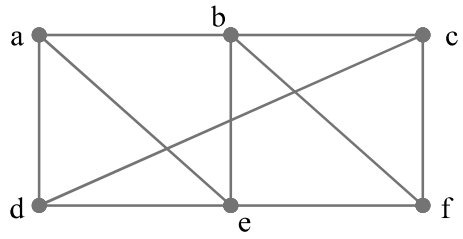
Chu trình là đường đi có đỉnh đầu trùng với đỉnh cuối (tức là u = v).

Đường đi hay chu trình được gọi là đơn nếu như không có cạnh nào bị lặp lại.

Đường đi hay chu trình được gọi là sơ cấp nếu như không có đỉnh nào bị lặp lại.

Ví dụ 1:

Xét đồ thị vô hướng sau:



Hình 2.5. Minh họa đường đi, chu trình trong đồ thị vô hướng

Trong đồ thị hình 2.5:

- Dãy a, d, c, f, e là đường đi đơn độ dài 4.

- Còn d, e, c, a không là đường đi, do (c,e) không phải là cạnh của đồ thị.

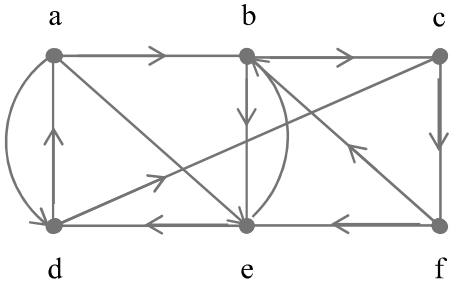
- Dãy b, c, f, e, b là chu trình độ dài 4.

- Đường đi a, b, e, d, a, b có độ dài là 5 không phải là đường đi đơn, do cạnh (a, b) có mặt trong nó 2 lần.

✍ Khái niệm đường đi và chu trình trên đồ thị có hướng được định nghĩa hoàn toàn tương tự như trong trường hợp đồ thị vô hướng, chỉ khác là ta có chú ý đến hướng trên các cung.

Ví dụ 2:

Xét đồ thị có hướng sau:



Hình 2.6. Minh họa đường đi, chu trình trong đồ thị có hướng

Trong đồ thị hình 2.6:

- Dãy a, d, c, f, e là đường đi đơn độ dài 4.

- Còn d, e, c, a không là đường đi, do (e, c) không phải là cung của đồ thị.

- Dãy b, c, f, e, b là chu trình độ dài 4.

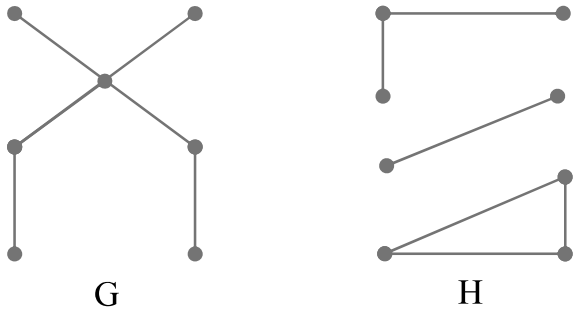
- Đường đi a, b, e, d, a, b có độ dài là 5 không phải là đường đi đơn, do cung (a, b) có mặt trong nó 2 lần.

2.3. TÍNH LIÊN THÔNG CỦA ĐỒ THỊ

2.3.1. Đồ thị liên thông

Đồ thị vô hướng G = (V, E) được gọi là liên thông nếu luôn tìm được đường đi giữa hai đỉnh bất kỳ của nó.

Ví dụ:



Hình 2.7. Minh họa đồ thị liên thông

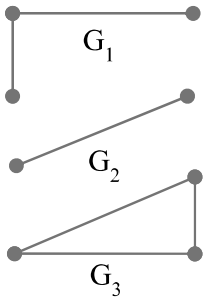
Trong các đồ thị trong hình 2.7:

- Đồ thị G là đồ thị liên thông.
- Đồ thị H không liên thông.

2.3.2. Thành phần liên thông

Gọi đồ thị con của đồ thị G = (V, E) là đồ thị H = (W, F), trong đó $W \subseteq V$ và $F \subseteq E$. Trong trường hợp đồ thị là không liên thông, nó sẽ rã ra thành một số đồ thị con liên thông đôi một không có đỉnh chung. Những đồ thị con liên thông như vậy ta sẽ gọi là các thành phần liên thông của đồ thị.

Ví dụ:



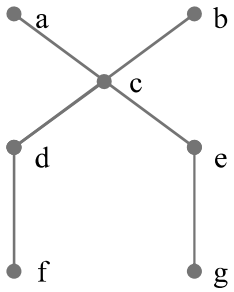
Hình 2.8. Minh họa thành phần liên thông

Đồ thị trong hình 2.8 gồm có 3 thành phần liên thông G_1, G_2, G_3.

Đỉnh rẽ nhánh: Đỉnh v được gọi là đỉnh rẽ nhánh nếu việc loại bỏ v cùng với các cạnh liên thuộc với nó khỏi đồ thị làm tăng số thành phần liên thông của đồ thị.

Cầu: Cạnh e được gọi là cầu nếu việc loại bỏ nó khỏi đồ thị làm tăng số thành phần liên thông của đồ thị.

Ví dụ:



Hình 2.9. Minh họa đỉnh rẽ nhánh, cầu

Đồ thị trong hình 2.9 có:

- Các đỉnh c, d, e là đỉnh rẽ nhánh.
- Các cạnh (d, f) và (e, g) là cầu.

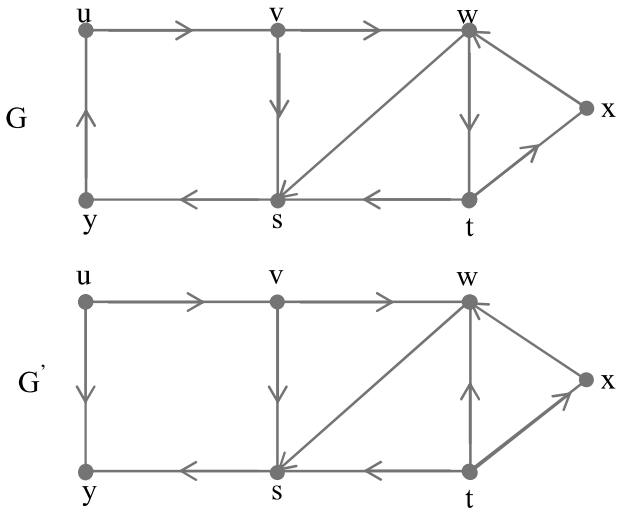
✍ Với đồ thị có hướng, ta có thêm hai khái niệm liên thông mạnh và liên thông yếu như định nghĩa bên dưới.

Định nghĩa:

Đồ thị có hướng G = (V, E) được gọi là liên thông mạnh nếu luôn tìm được đường đi giữa hai đỉnh bất kỳ của nó.

Đồ thị có hướng G = (V, E) được gọi là liên thông yếu nếu đồ thị vô hướng tương ứng với nó là liên thông.

Ví dụ:



Hình 2.10. Minh họa tính liên thông mạnh, yếu

Trong các đồ thị có hướng hình 2.10:

- Đồ thị G là liên thông mạnh.

- Đồ thị G' là liên thông yếu, do không có đường đi từ đỉnh u tới đỉnh x cũng như từ đỉnh x tới đỉnh u.

2.3.3. Thuật toán xác định các thành phần liên thông

Cho đồ thị G = (V, E) gồm n đỉnh. Hãy xác định các đỉnh của G thuộc thành phần liên thông nào (gán nhãn các đỉnh là thành phần liên thông tương ứng).

Bước 1. Khởi tạo biến label = 0 và gán nhãn 0 cho tất cả các đỉnh.

Bước 2. Duyệt qua tất cả các đỉnh i ∈ V

Nếu nhãn của i là 0

label = label + 1

gán nhãn cho tất cả các đỉnh cùng thuộc thành phần liên thông với i là label.

Thuật toán gán nhãn các đỉnh cùng thuộc thành phần liên thông với đỉnh i:

Cho đồ thị G = (V, E), đỉnh i, nhãn label. Hãy gán nhãn label cho các đỉnh thuộc cùng thành phần liên thông với i.

Bước 1. Gán nhãn label cho đỉnh i.

Bước 2. Duyệt qua tất cả các đỉnh j ∈ V và có cạnh nối với i

Nếu nhãn của j là 0

gán nhãn các đỉnh cùng thuộc thành phần liên thông với đỉnh j là label.

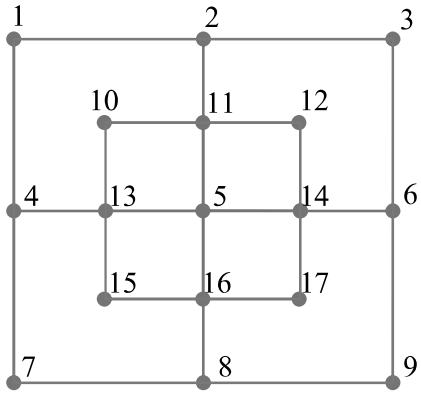
✪ Thuật toán trên có thể viết bằng ngôn ngữ C như sau:

```
int nhan[MAX]; // nhãn của các đỉnh
// Hàm gán nhãn một thành phần liên thông
void GanNhan(int i, int label)
{
      nhan[i] = label;
      for (int j=0; j<n; j++)
      {
            if((nhan[j]==0) && (A[i][j]||A[j][i]))
            {
                  GanNhan(j, label);
            }
      }
}
```

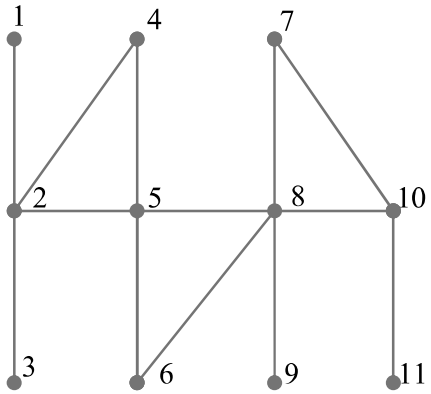
```
// Hàm đếm số thành phần liên thông
// và gán nhãn tất cả thành phần liên thông
int DemSoLienThong()
{
      int i, label;
      for (int i=0; i<n; i++)
      {
            nhan[i] = 0;
      }
      label = 0;
      for (int i=0; i<n; i++)
      {
            if (nhan[i] == 0)
            {
                  label ++;
                  GanNhan(i, label)
            }
      }
      return label;    // trả về số thành phần liên thông
}
```

BÀI TẬP CHƯƠNG 2

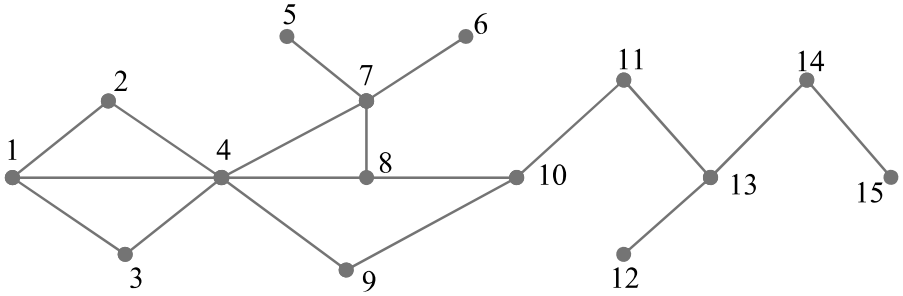
1. Duyệt các đồ thị sau theo chiều sâu và chiều rộng:



G_1

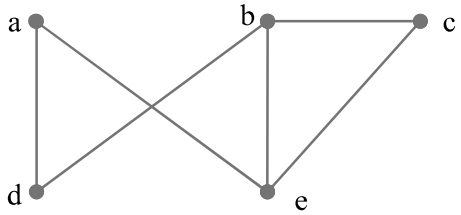


G_2



G_3

2. Cho đồ thị như hình vẽ. Hỏi danh sách các đỉnh đã cho có tạo nên đường đi không? Đường đi nào là đường đi đơn, đường đi nào là chu trình? Độ dài là bao nhiêu?



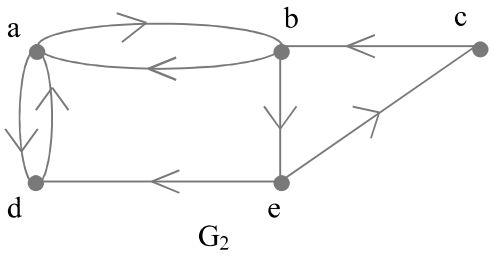
G_1

a) a, e, b, c, b

b) e, b, a, d, b, e

c) a, e, a, d, b, c, a

d) c, b, d, a, e, c



G_2

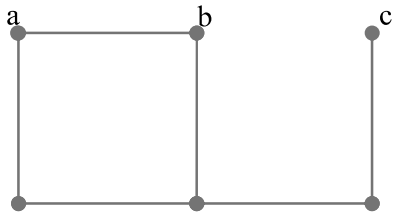
a) a, b, e, c, b

b) a, d, b, e, a

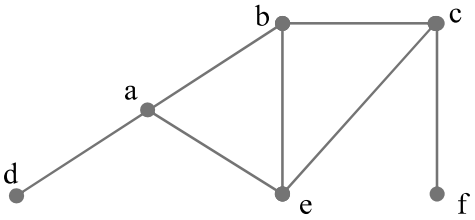
c) a, d, a, d, a

d) a, b, e, c, b, d, a

3. Xác định tất cả các đỉnh rẽ nhánh (đỉnh khớp, đỉnh cắt) và cầu của đồ thị sau:



G_1



G_2

4. Viết chương trình nhập vào một đồ thị vô hướng (tối đa 30 đỉnh), xác định xem đồ thị có liên thông hay không. Nếu đồ thị không liên thông, hãy cho biết có mấy thành phần liên thông và in ra các thành phần liên thông của đồ thị. Dữ liệu nhập được cho từ file.

5. Viết chương trình cho biết có tồn tại đường đi giữa 2 đỉnh s và t nào đó trên đồ thị không? Giả sử rằng ma trận kề được lưu trên file dạng text.