Đồ thị
(Graph 2)

Nguyễn Phương Thái
Bộ môn Khoa Học Máy Tính – Khoa CNTT
Đại Học Công Nghệ - ĐHQGHN
Email: thainp@vnu.edu.vn

Đồ thị (graph)

- G = (V, E)
  - V: Tập đỉnh
  - E = { (u,v) | u, v ∈ V}: Tập cạnh

Ví dụ: Biểu diễn bản đồ đường đi trong thành phố bằng đồ thị G = (V, E)

- V: Tập hợp các điểm trong thành phố
- E: Tập hợp các đường đi trong thành phố, mỗi đường đi nối hai điểm

Đi qua đồ thị theo chiều rộng (Breadth first search)

- Đi qua tất cả các đỉnh của đồ thị, mỗi đỉnh đúng một lần
- Bắt đầu xuất phát từ một đỉnh **s**, lần lượt thăm các đỉnh liền kề với **s**. Tiếp tục quá trình thăm các đỉnh theo nguyên tắc: Đỉnh nào được thăm trước thì các đỉnh liền kề với đỉnh đó sẽ được thăm trước
- Xem ví dụ

  http://www.cs.princeton.edu/~wayne/cs423/lectures.html

Đi qua đồ thị theo chiều sâu (Depth first search)

```
//Đi qua đồ thị theo chiều sâu xuất phát từ v
DepthFirstSearch (v) {
    for (mỗi đỉnh u kề với v)
        if (u chưa được thăm) {
            thăm u và đánh dấu u đã được thăm
            DepthFirstSearch (u)
        }
}
```

Xem ví dụ

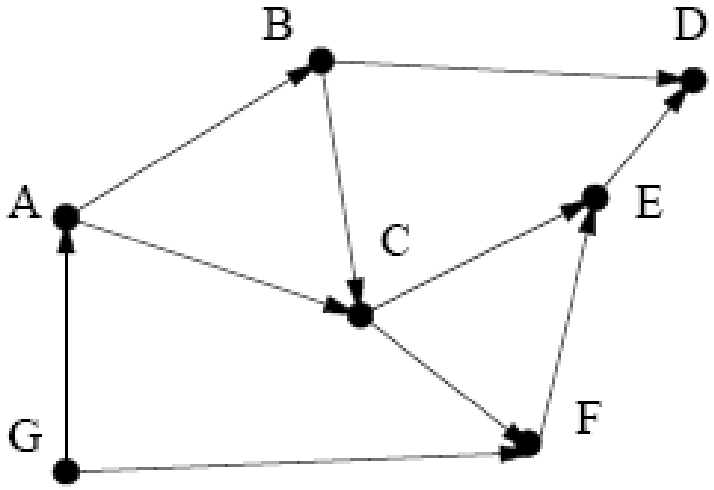
http://www.cs.princeton.edu/~wayne/cs423/lectures.html

Sắp xếp topo

Cho đồ thị có hướng nhưng không có chu trình G = (V, E)

(Directed acylic graph / DAG)

Sắp xếp các đỉnh của đồ thị G thành một danh sách sao cho nếu có cung $(u,v) \in E$, thì đỉnh ***u*** phải đứng trước đỉnh ***v***.



G A B C F E D

Sắp xếp topo

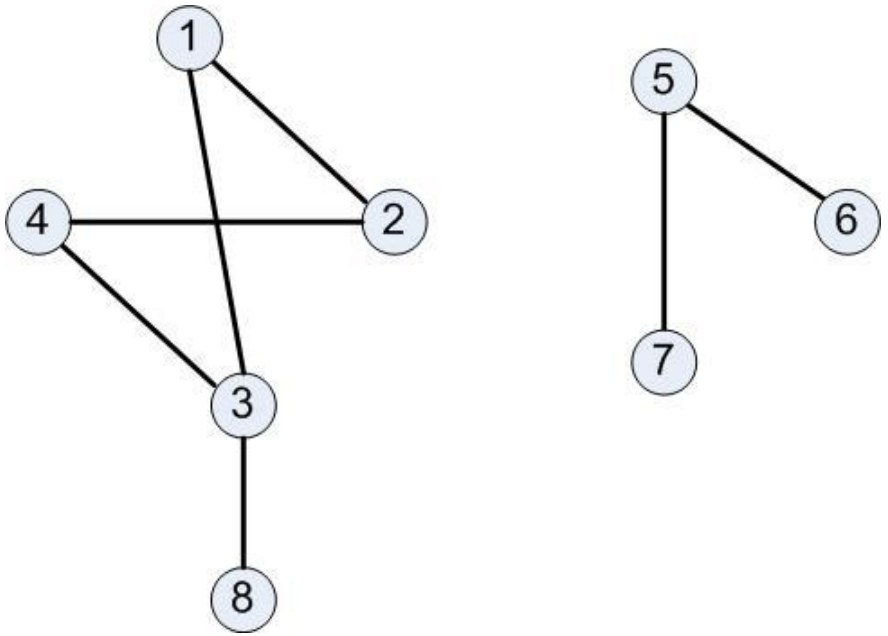
```
TopoSort (u) {
    for (mỗi đỉnh v kề u)
        if ( v chưa thăm)
            TopoSort(v);
     Xen  u  vào đầu danh sách T;
    Đánh dấu u đã thăm;
}

//Sắp xếp các đỉnh của đồ thị định hướng
//không có chu trình G =(V,E) thành danh sách topo.
 TopoSortGraph(G) {
        for (mỗi đỉnh u ∈ V)
            Đánh dấu u chưa được thăm;

        Khởi tạo danh sách topo T rỗng;
        for (mỗi đỉnh u ∈ V)
            if (u chưa thăm)
                TopoSort (u);
}
```

Đường đi giữa hai điểm

Cho đồ thị G = (V, E)

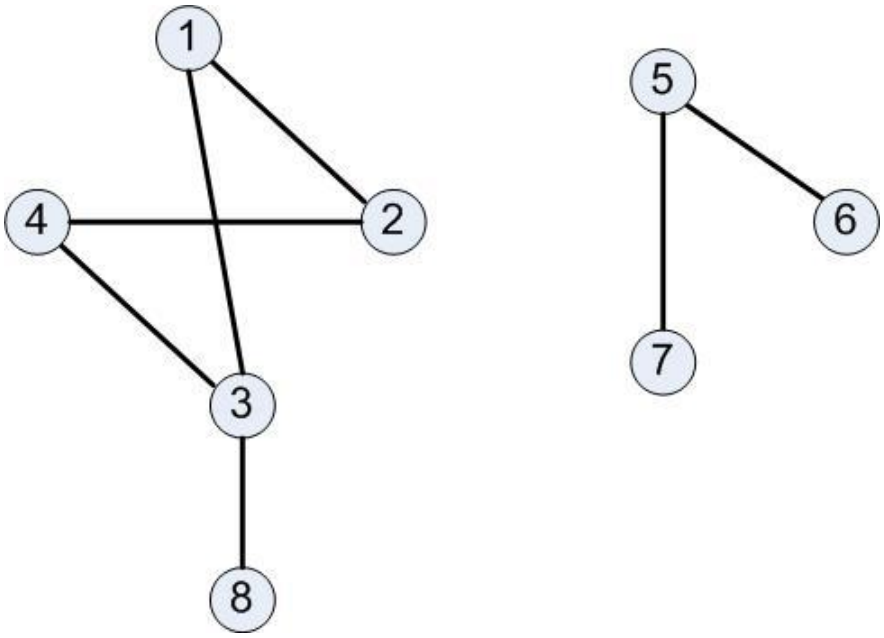
Giữa hai đỉnh (x_0, x_k) có đường đi, nếu tồn tại (x_1,…,x_{k-1}) thỏa mãn

$(x_i, x_{i+1}) \in E, \quad \forall i = 0 \ldots (k-1)$



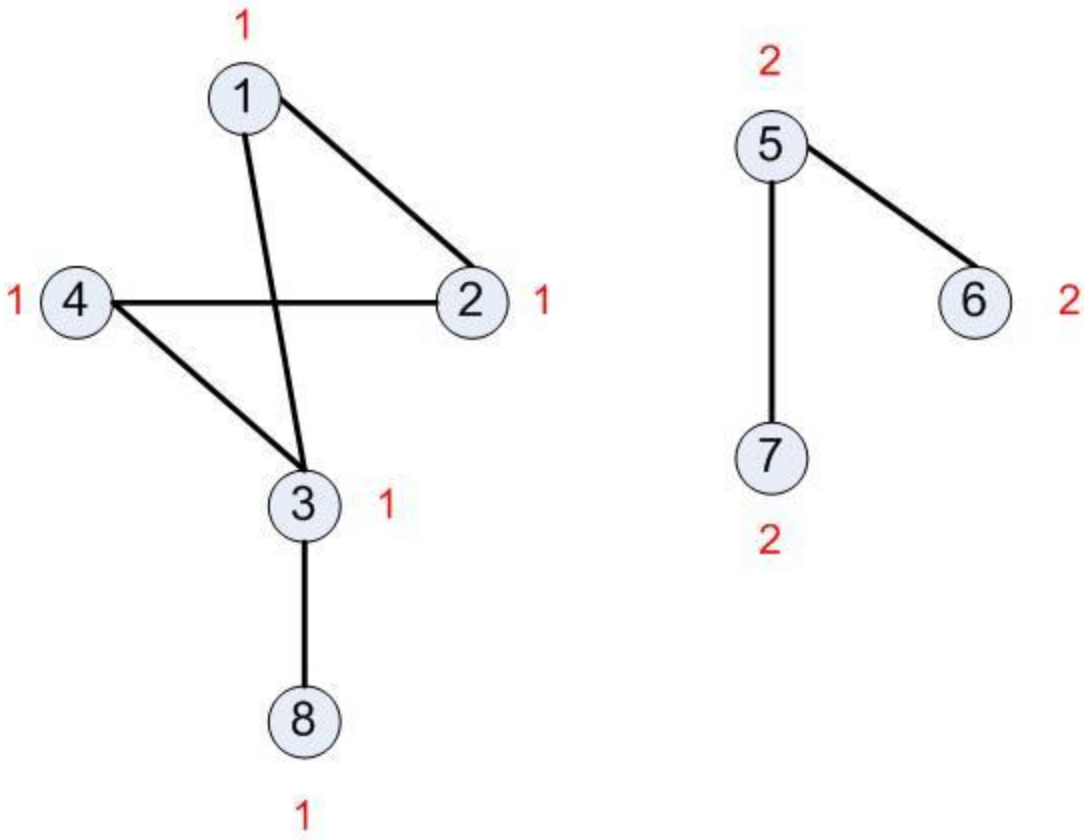
Đồ thị liên thông
(Connected graph)

Cho đồ thị G = (V, E), G được gọi là liên thông nếu tồn tại đường đi giữa hai đỉnh bất kì của đồ thị



Tìm tất cả các thành phần liên thông

Cho đồ thị G = (V, E), tìm tất cả các thành phần liên thông của đồ thị. Đỉnh *i* của đồ thị được gán nhãn c_i cho biết thuộc miền liên thông c_i.



Tìm các thành phần liên thông

```
//Đi qua đồ thị theo bề rộng xuất phát từ v
FindConnectedComponent (v, ci)  {
  (1)   Khởi tạo hàng đợi Q rỗng;
  (2)   Xen v vào hàng đợi Q;
  (3)   Đánh dấu đỉnh v đã được thăm, và gán nhãn v bằng ci
  (4)   while (hàng đợi Q không rỗng) {
  (5)         Lấy đỉnh w ở đầu hàng đợi Q;
  (6)         for (mỗi đỉnh u kề w)
  (7)               if ( u chưa được thăm) {
  (8)                     Xen u vào đuôi hàng đợi Q;
  (9)                     Đánh dấu u đã được thăm, và gán nhãn u bằng ci
                    }
  (10)        Loại w ra khỏi hàng đợi Q
      }  // hết vòng lặp while.
}
```