Tính liên thông của đồ thị

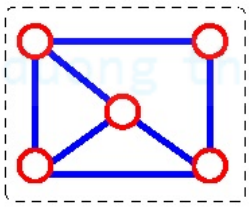
Tính liên thông của đồ thị

- Tính liên thông của đồ thị.
- Tính song liên thông.
- Đỉnh khớp
- Cầu

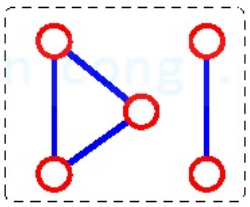
Nhắc lại một số khái niệm

Tính liên thông

- Đồ thị liên thông: một cặp đỉnh bất kỳ được nối với nhau bằng ít nhất một đường đi.



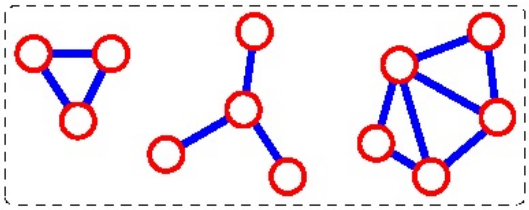
Đồ thị liên thông



Đồ thị không liên thông

Tính liên thông

- Thành phần liên thông: đồ thị con liên thông tối đại của G.



Quan hệ tương đương

- Một ***quan hệ*** trên tập hợp S là tập R các cặp có thứ tự các phần tử của S định nghĩa bởi một thuộc tính nào đó
- **Ví dụ:**
  - S ={ 1, 2, 3, 4 }
  - R ={ (i,j) ∈ S ´ S sao cho i < j }
    ={(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)}

Quan hệ tương đương

- Một ***quan hệ tương đương*** là quan hệ với các thuộc tính sau:
  - Tính phản xạ: (x,x) ∈ R, ∀ x ∈ S ***(reflexive)***
  - Tính đối xứng:(x,y) ∈ R ⇒ (y,x) ∈ R ***(symmetric)***
  - Tính bắc cầu :(x,y), (y,z) ∈ R⇒ (x,z) ∈ R ***(transitive)***

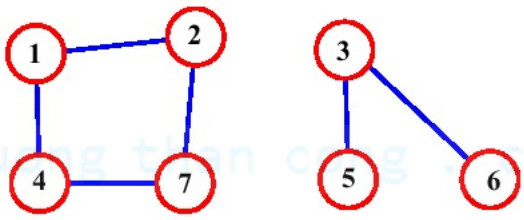
Quan hệ tương đương

- Quan hệ C trên tập các đỉnh của đồ thị:
  - (u,v) Ỵ C ***u*** và ***v*** thuộc cùng một thành phần liên thông

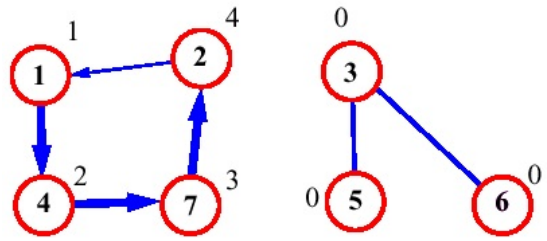
  là quan hệ tương đương.

DFS trên đồ thị không liên thông

DFS trên đồ thị không liên thông



DFS trên đồ thị không liên thông



Sau khi thực hiện DFS(1):

k	1	2	3	4	5	6	7
val[k]	1	4	0	2	0	0	3

DFS trên đồ thị không liên thông

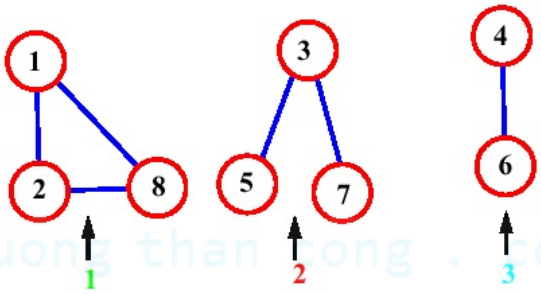
- Hàm đệ qui DFS thăm tất cả các đỉnh thuộc thành phần liên thông.
- Cần thêm vào một vòng lặp **for** để thăm tất cả các đỉnh của đồ thị:

```
for k = 1 to N do
   if (val[k] = 0) then dfs(k)
```

DFS trên đồ thị không liên thông

- Cách biểu diễn các thành phần liên thông:
  - Dùng mảng **Comp[1..N]** để biểu diễn:
    - Comp[k] = i nếu đỉnh k ∈ thành phần liên thông **i**

DFS trên đồ thị không liên thông



Sau khi thực hiện DFS(1):

k	1	2	3	4	5	6	7	8
val[k]	1	1	2	3	2	3	2	1

Thuật toán DFS xác định các thành phần liên thông

```
Algorithm DFS(v, id)
Input: Moät ñænh v cuûa ñoà thò, chæ soá id cuûa tplt
Output:  Gaùn nhaõn id cho taát caû caùc ñænh cuûa tplt
Comp[v]  =id;
for (moïi ñænh k keà vôùi v) do
   if Comp[k]= 0 then
      Goïi ñeä qui DFS(k, id);
```


Thuật toán DFS xác định các thành phần liên thông

```
Algorithm ThanhPhanLT
Input:   Ñoà thò G
Output:  Maûng Comp cho bieát caùc tplt
id   = 0;
for k = 1 to N do Comp[k] = 0;
for k = 1 to N do
   if (Comp[k] = 0) then
      id++;
      DFS(k, id);
```

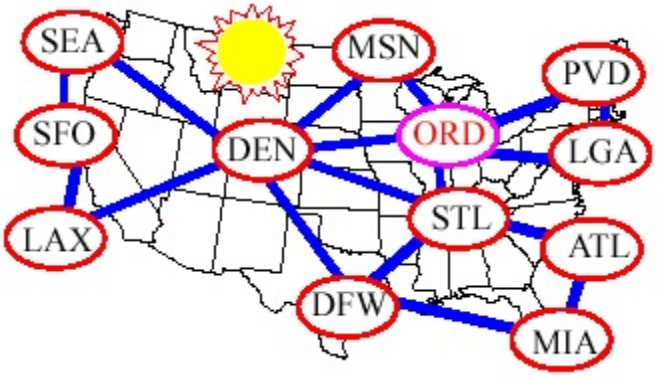
Tính song liên thông

Các đỉnh khớp

- **Đỉnh khớp** (**cutvertex**) là đỉnh nếu khi huỷ nó ra khỏi đồ thị sẽ làm tăng số thành phần liên thông của đồ thị
- Nếu G liên thông, w là đỉnh khớp của G thì sau khi huỷ w, G không còn liên thông.

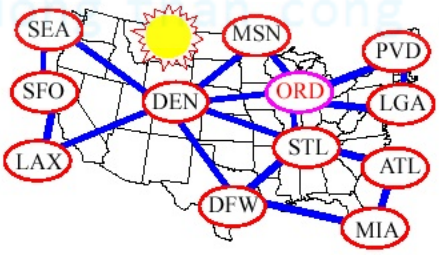
Các đỉnh khớp

- Ví dụ, ORD và DEN là các đỉnh khớp, MSD không phải.



Các đỉnh khớp

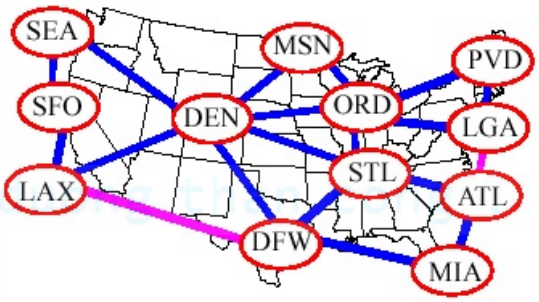
- Nếu sân bay DEN (Denver) bị đóng cửa, các tuyến bay nối miền đông và miền tây USA sẽ bị gián đoạn, nếu sân bay ORD (Chicago) bị đóng cửa, ta không thể đi từ Providence (PVD) đến Denver



Đồ thị song liên thông

- **Đồ thị song liên thông** (**biconnected Graph**) là đồ thị liên thông và không chứa đỉnh khớp.

Đồ thị song liên thông



- Nếu mở thêm hai tuyến bay **LGA-ATL** và **DFW-LAX**, mạng lưới sẽ trở thành đồ thị song liên thông.

Đồ thị song liên thông

- **Một số tính chất của đồ thị song liên thông:**
  - Có ít nhất hai đường đi khác nhau giữa hai đỉnh bất kỳ.
  - Giữa hai đỉnh bất kỳ, tồn tại chu trình sơ cấp đi qua chúng

Đồ thị song liên thông

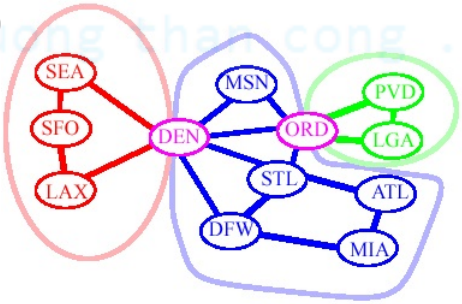
- Ta qui ước, đồ thị K_2 là đồ thị song liên thông, mặc dù nó không thoả hai tính chất trên

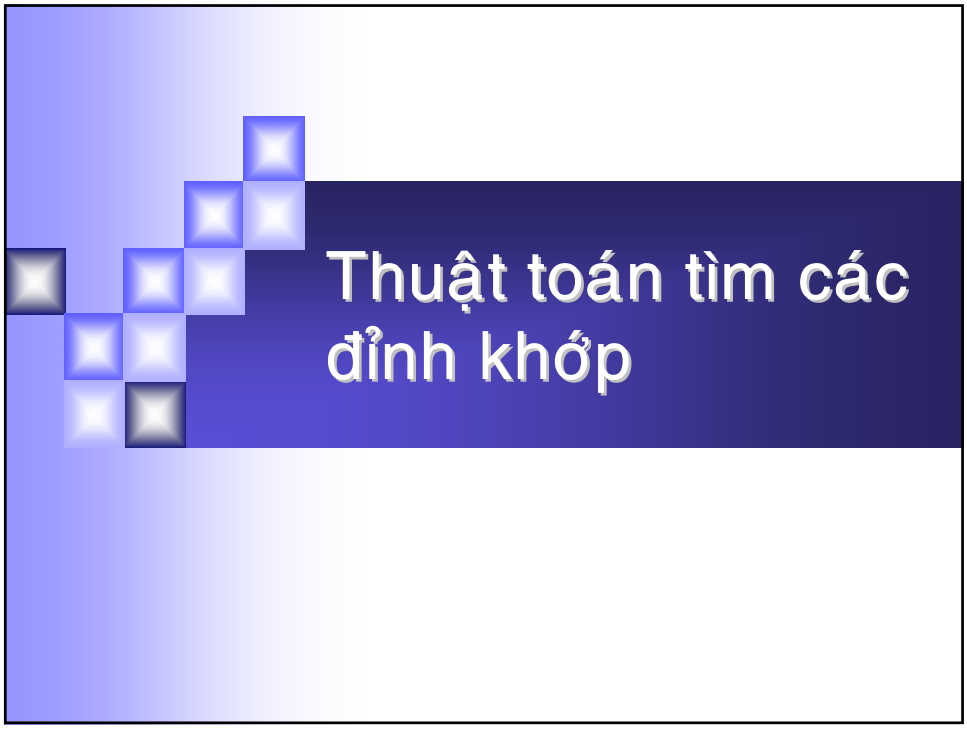
Thành phần song liên thông

- Thành phần song liên thông: đồ thị con song liên thông tối đại của G.
- Mấu chốt của việc xác định tính song liên thông cũng như các thành phần song liên thông là tìm cách xác định các đỉnh khớp.

Thành phần song liên thông

- Các thành phần song liên thông của một đồ thị không có cạnh chung nhưng một đỉnh có thể thuộc nhiều thành phần song liên thông (ví dụ DEN, ORD). Những đỉnh loại này chính là các đỉnh khớp





Thuật toán Brute-Force (vét cạn)

- Huỷ một đỉnh ra khỏi đồ thị, xác định tính liên thông của nó.
- Thử cho tất cả các đỉnh của đồ thị ta sẽ có kết quả mong muốn

Thuật toán Brute-Force (vét cạn)

```
Algorithm Brute-Force {tìm đỉnh khớp}
Input:    Ñoà thò lieân thoâng G
Output:   Danh saùch L caùc ñænh khôùp
L = ∅;
for (moïi ñænh k cuûa G) do
    Huyû k ra khoûi G;
    Kieåm tra tính lieân thoâng cuûa G;
    Neáu G khoâng lieân thoâng L = L +[k];
    Ñaët k vaøo laïi G;
```

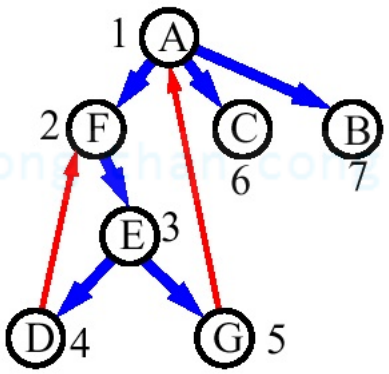

Thuật toán Brute-Force (vét cạn)

- Độ phức tạp của thuật toán:
  - Cần n lần kiểm tra tính liên thông.
  - Mỗi lần kiểm tra tốn chi phí O(n + m).
  - Như vậy, tổng chi phí sẽ là **$O(n^2 + nm)$**.
- Chi phí như vậy sẽ khoảng $O(n^3)$, rất cao.

Thuật toán đánh số DFS (DFS numbering)

- Nhắc lại là khi duyệt cây bằng DFS, các cạnh của đồ thị được chia làm hai loại: cạnh thuộc **cây DFS** và **backedge**.
  - (u,v) là cạnh thuộc **cây DFS** ⇔ val [u] < val [v]
  - (u,v) là backedge ⇔ val[u] > val[v]
- Ta sẽ xác định các đỉnh khớp thông qua cách đánh số DFS và hai thuộc tính ...

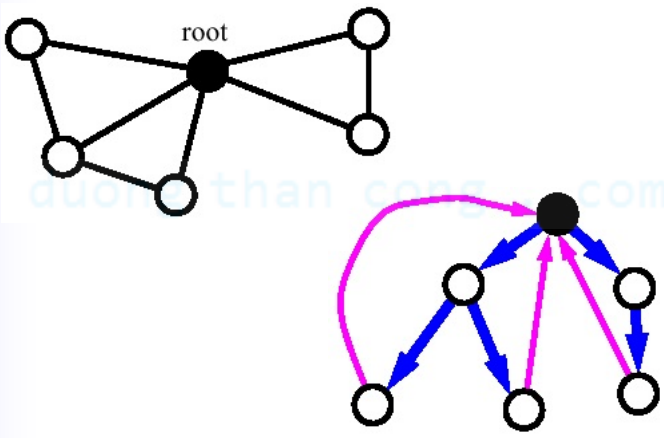
Thuật toán đánh số DFS (DFS numbering)



Thuật toán đánh số DFS (DFS numbering)

- **Thuộc tính 1 (gốc):** gốc (root) của cây DFS là đỉnh khớp nếu có 2 hoặc nhiều hơn cạnh trên cây đi ra từ nó.
  - Trong quá trình duyệt, ta đã phải quay lại gốc để đi đến những nút còn lại trong đồ thị
  - Khi đang đứng ở một cây con của Root, ta chỉ có thể đi đến cây con khác bằng cách băng qua gốc.

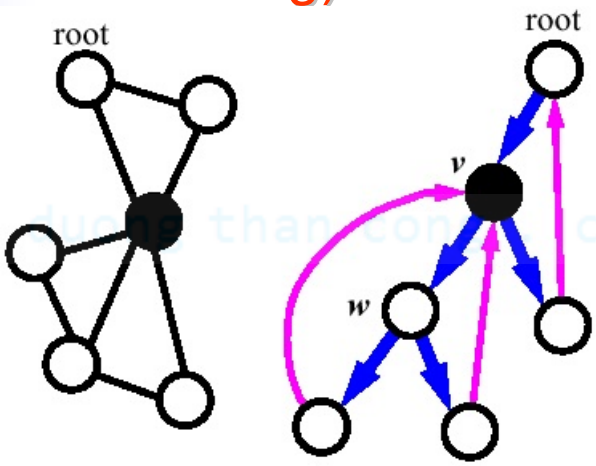
Thuật toán đánh số DFS (DFS numbering)



Thuật toán đánh số DFS (DFS numbering)

- **Thuộc tính 2 (phức hợp):** một đỉnh **v** không phải là gốc của cây DFS là đỉnh khớp nếu nó có đỉnh con **w** sao cho không có backedge nào nối **w** với các "**tổ tiên**" của **v**.

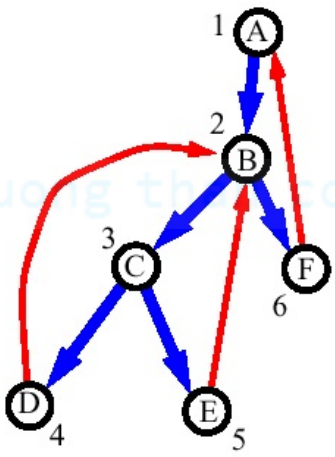
Thuật toán đánh số DFS (DFS numbering)



Thuật toán đánh số DFS (DFS numbering)

- Một số định nghĩa:
  - Low(v): đỉnh được đánh số với chỉ số nhỏ nhất có thể đi đến được từ v bằng cách dùng một con đường có hướng (tạo được trong quá trình duyệt DFS) mà trên đó **có tối đa một cạnh backedge**.
  - Min(v) = val(low(v)).

Thuật toán đánh số DFS (DFS numbering)



v	low(*v*)	Min(*v*)
A	A	1
B	A	1
C	B	2
D	B	2
E	B	2
F	A	1

Thuật toán DFS tìm đỉnh khớp

Algorithm DFS {tìm đỉnh khớp}

Input: Ñoà thò lieân thoâng G

Output: Danh saùch L caùc ñænh khôùp

L = ∅;

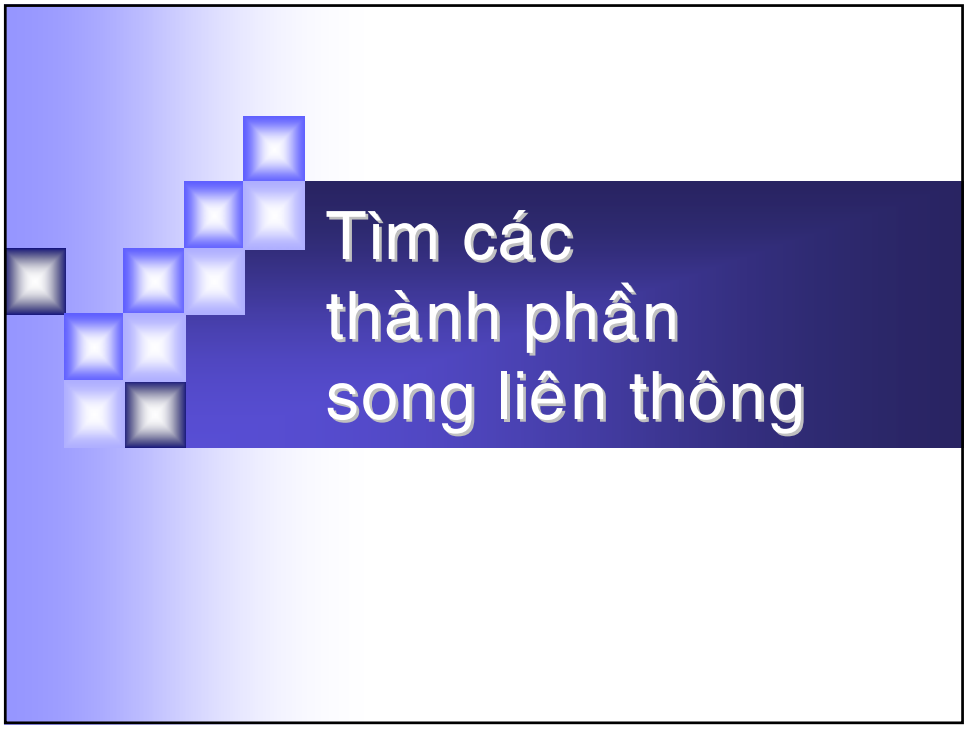
Thực hiện **DFS** trên đồ thị G

Kiểm tra nếu root của cây DFS có ít nhất 2 con (thoả mãn thuộc tính 1) thì L = L +[root]

Với mỗi đỉnh v, kiểm tra nếu có cạnh (v, w) của cây DFS thoả **Min(w) ≥ val[v]** thì L = L +[v]

Thuật toán DFS tìm đỉnh khớp

- Để xác định Min(v) ta có quan hệ sau: Min(v) = val(low(v)) là số nhỏ nhất trong các giá trị
  - Val[v]
  - số nhỏ nhất trong các số Min(w) với (v, w) là cạnh trên cây
  - số nhỏ nhất trong các số Val[z] với (v, z) là backedge.
- Như vậy, ta có thể xác đinh Min(v) bằng thuật toán đệ qui.



Tìm các thành phần song liên thông

- DFS thăm các đỉnh và các cạnh của mỗi thành phần song liên thông một cách liên tiếp
- Sử dụng một stack để lưu vết của thành phần song liên thông đang duyệt.

Tìm các thành phần song liên thông