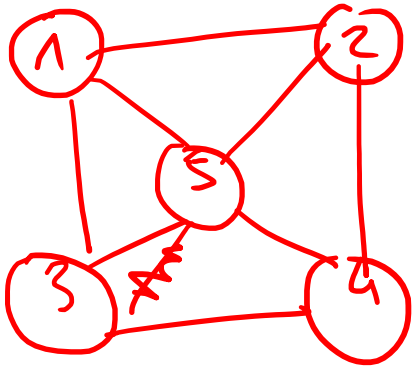
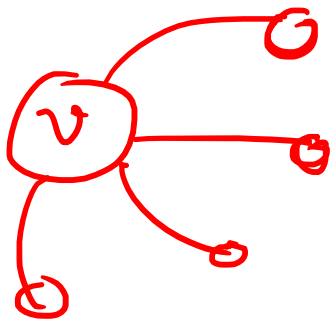
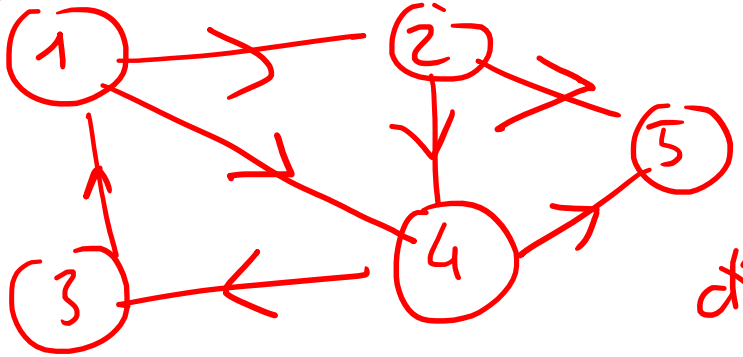


$V = \{1,2,3,4,5\}$

$E = \{(1,2),(1,3),(1,5),(2,4),(2,5),(3,4),(3,5),(4,5)\}$

đồ thị vô hướng





$V = \{1,2,3,4,5\}$

cung (arc) $E = \{(1,2),(1,4),(2,4),(2,5),(3,1),(4,3),(4,5)\}$

đồ thị có hướng.

Đồ thị thưa (số cạnh nhỏ hơn rất nhiều so với bình phương số đỉnh)

⇓

D/S kề: A[v]: d/s các đỉnh (cạnh/cung) kề với v.

Đồ thị đầy (complete graph) → ma trận kề...

D/S cạnh:

Dùng thư viện vector để thể hiện DS kề.

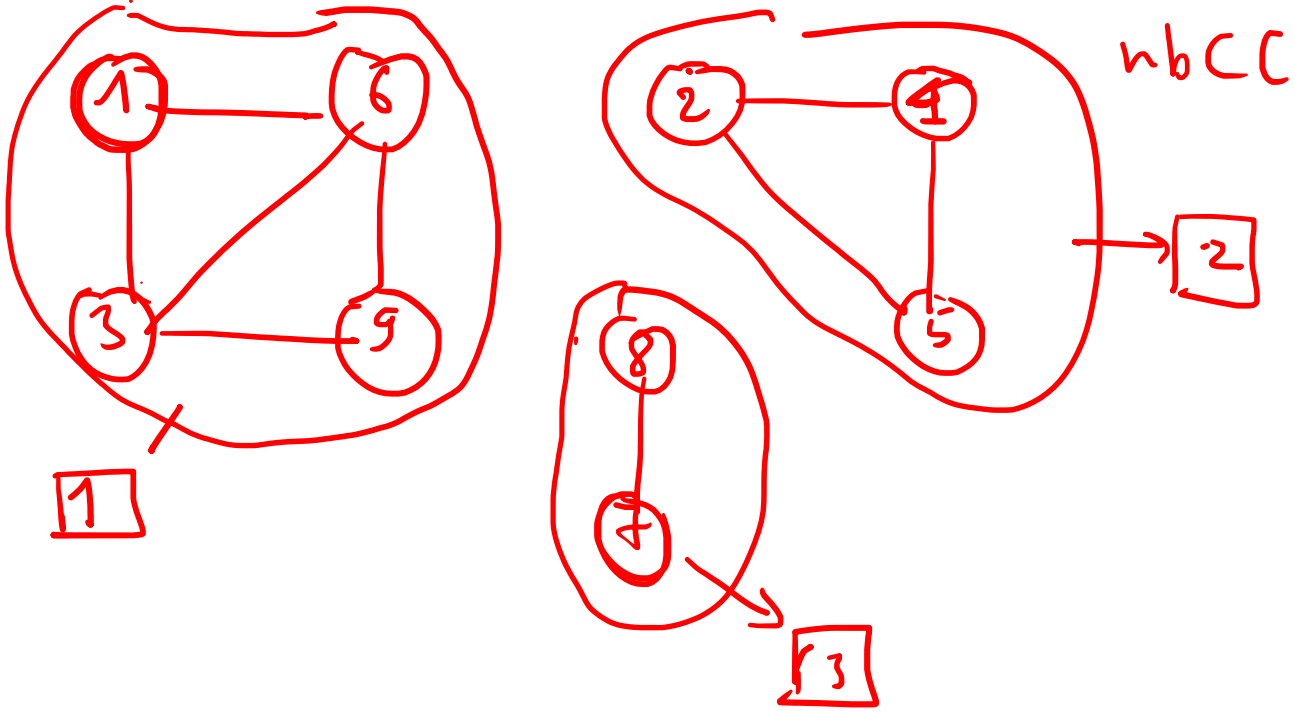
vector<int> A[N]; // A[v]: list các đỉnh kề với v.

```
struct Edge{
    int nod; // other endpoint
    int weight; // trọng số.

};
vector<Edge> A[N];
```

Minh họa 1: Lập trình liệt kê các connected component (thành phần liên thông của đồ thị vô hướng)

Dùng DFS(...)



nbCC = 3

DFS(1) → nbCC = 1

cc[3] = cc[1] = cc[6] = cc[9] = nbCC = 1.

cc[1] = 1 cc[3] = 1

cc[2] = 2 cc[4] = 2

cc[7] = cc[8] = 3

DFS(2) | nbCC = nbCC + 1 = 1 + 1 = 2

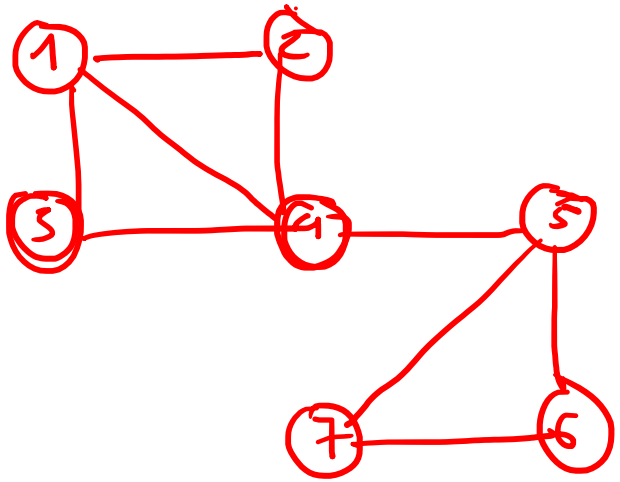
⇓

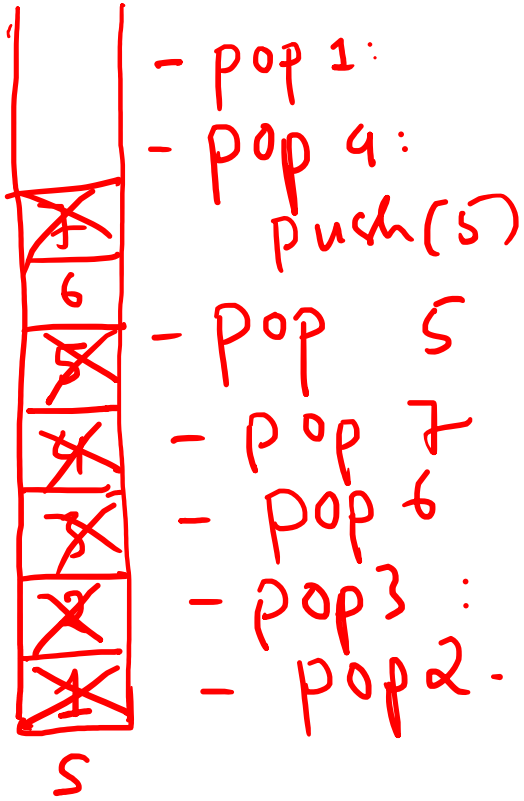
visit 4, 5 ⇒ cc[4] = cc[5] = nbCC = 2.

DFS(7) → nbCC = nbCC + 1 = 2 + 1 = 3

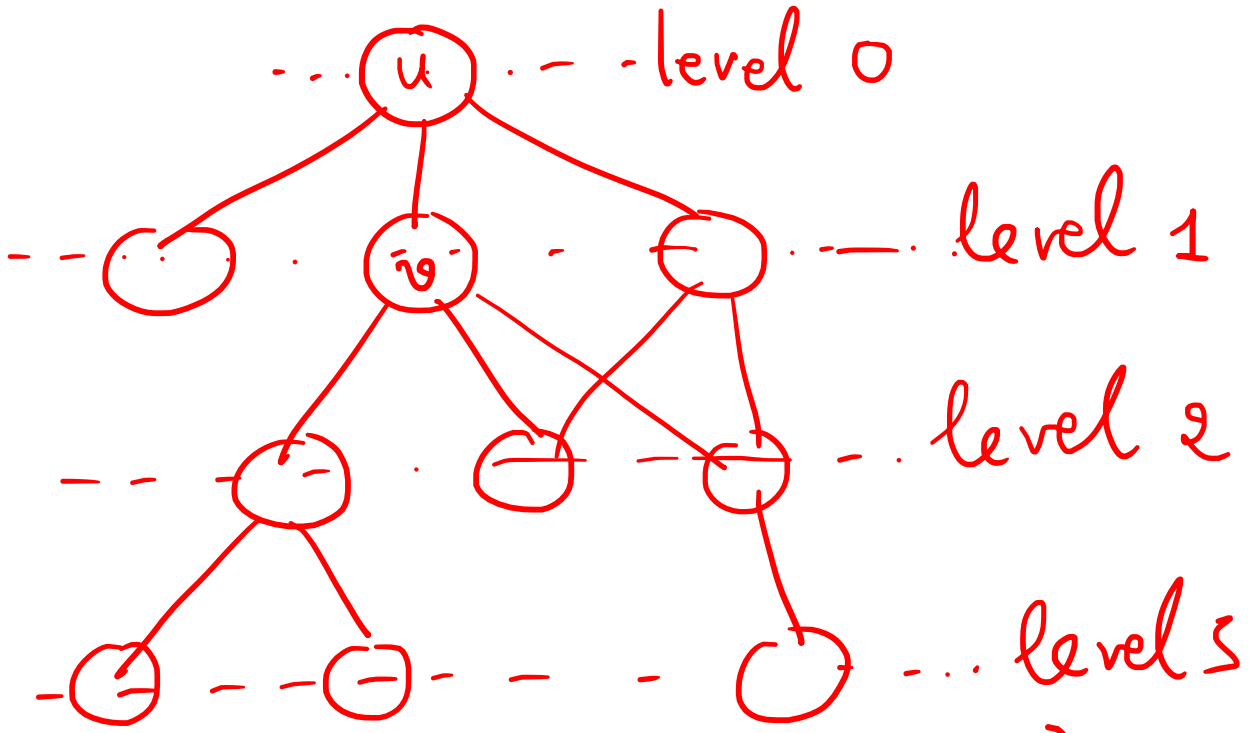
cc[7] = cc[8] = nbCC = 3

```
DFS(u)   // use stack -
[ stack S;
  S.push(u); visited[u] = True; cc[u] = nbCC;
  while S not empty do
  [ v = S.pop();
    for x ∈ A[v] do
      if not visited[x] then
      [ S.push(x);
        visited x = True;
```





Duyệt theo chiều rộng BFS(u) => Dùng Queue.

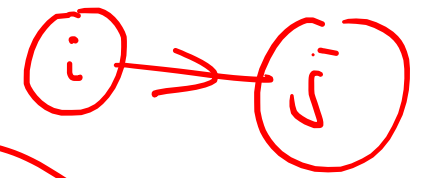


Duyệt theo thứ tự từ điển
→ Sort A[v] theo thứ tự tăng dần.

```
BFS(u)
[ Q.push(u);
  visited[u] = True; level[u] = 0;
  while Q not empty do
  [ v = Q.pop();
    for x ∈ A[v] do
      if not visited[x] then
      [ Q.push(x);
        visited[x] = True; Level[x] =
                          level[v] + 1
```

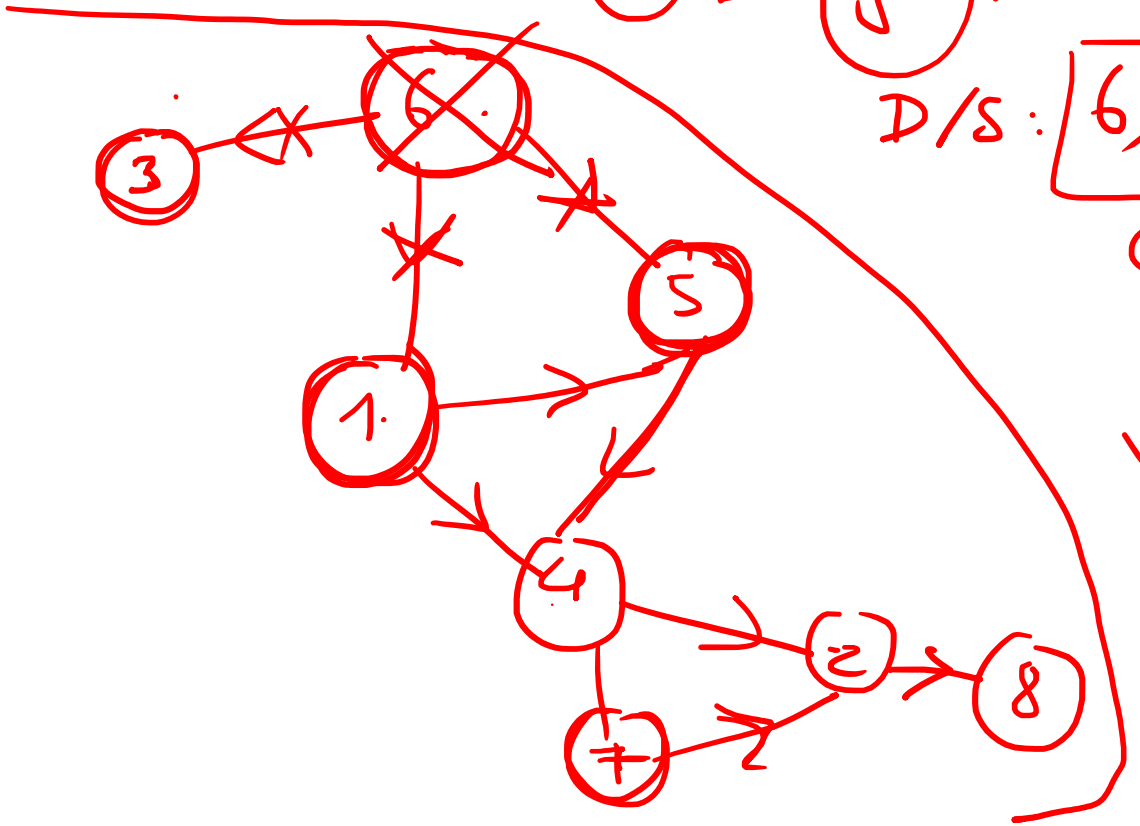
Sắp xếp TOPO:

Input: đồ thị có hướng và không có chu trình: cần chỉ ra dãy thứ tự các đỉnh, sao cho với mỗi cung (i,j) thì i đứng trước j trong D/S.



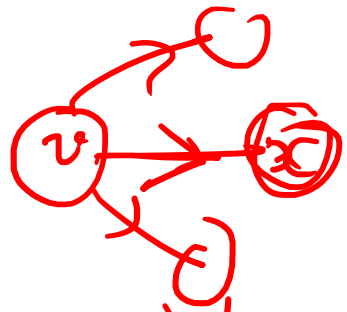
: muốn i thực hiện trước muốn j

D/S: 6, 3, 1, 5, 4, 7, 2, 8

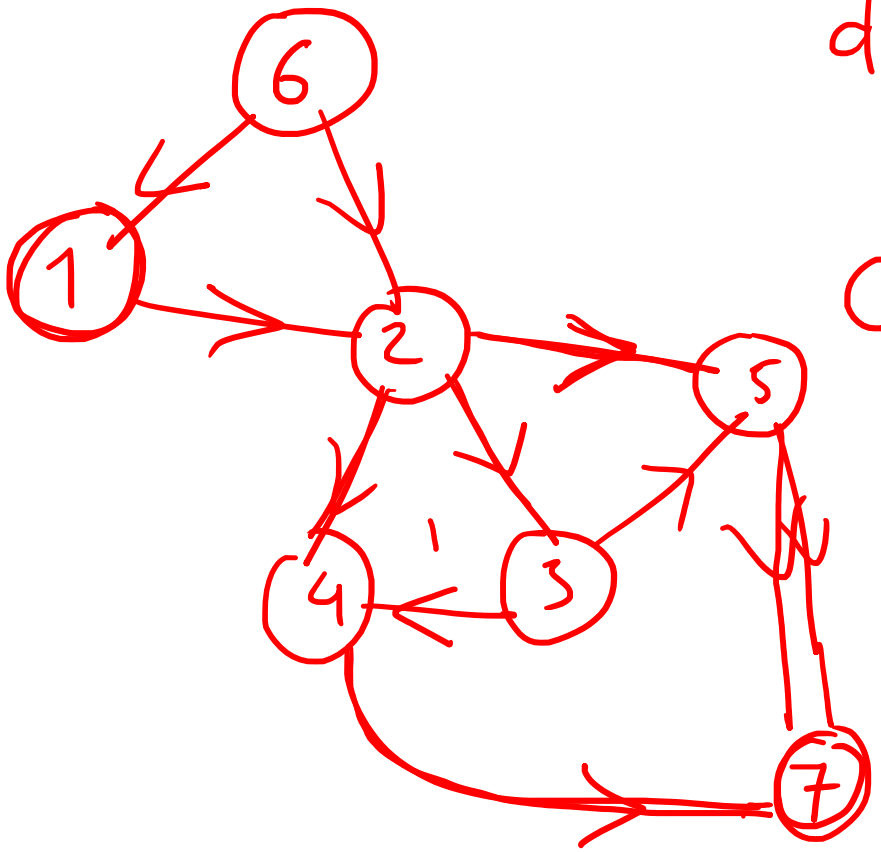


d[v]: bậc vào của đỉnh v.

d[v]=0 ⇒ sẽ đưa vào D/S TOPO



Queue Q;
Tính bậc vào d[v], ∀v∈V;
đưa các đỉnh v có d[v]=0 vào queue.
while Q not empty do
[v = Q.pop();
đưa vào sau mới D/S TOPO;
for x∈A[v] do
[d[x] = d[x] - 1
if d[x]=0 then Q.push(x)



d

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

read (u,v)
d[v] = d[v]+1

Q

TOPO | 6 | 1 | 2 | 3 | 4 | 5 | 7 |

6
1
2
3
4
5
7