$G = (V, E)$

$E \subseteq V \times V$

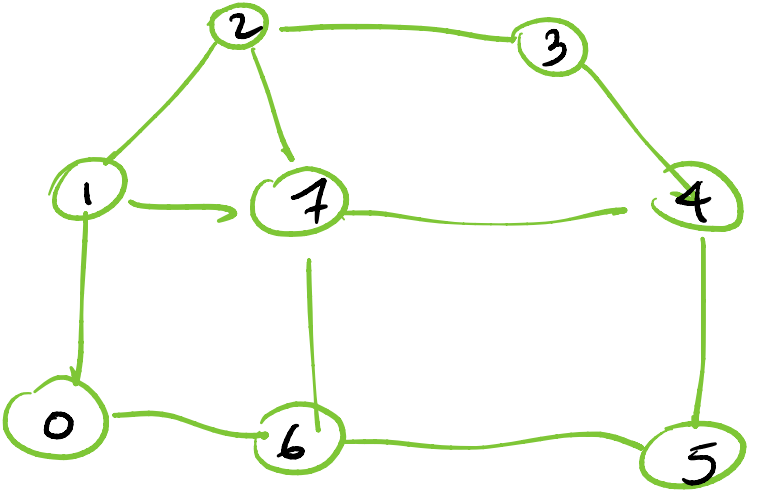
self-loop NO: (x, x)

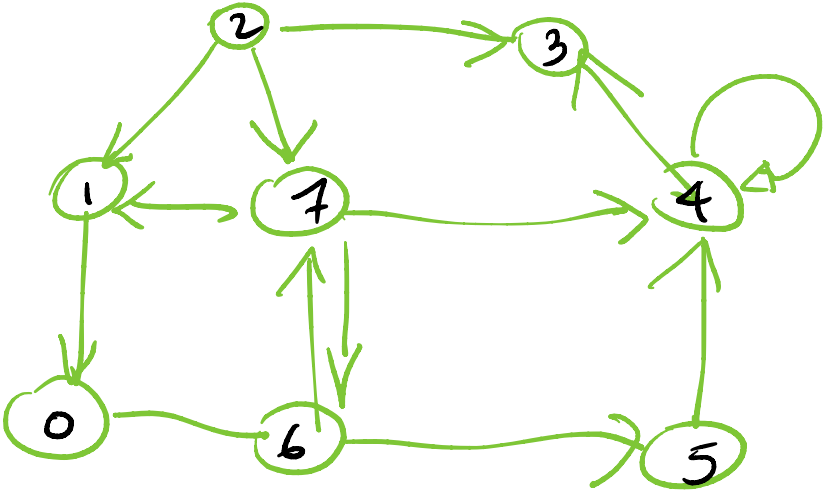
$G \leftrightarrow$ relazione binaria

$(u,v) \neq (v,u)$ orientato

$(u,v) = (v,u)$ non orientato
(rel. simmetrica)

$\{u,v\}$ oppure uv



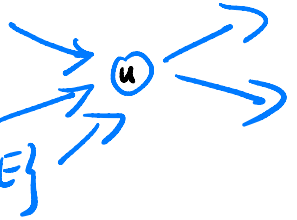


$d_u = \deg(u) = \text{grado}(u) = |N(u)|$

G non orientato

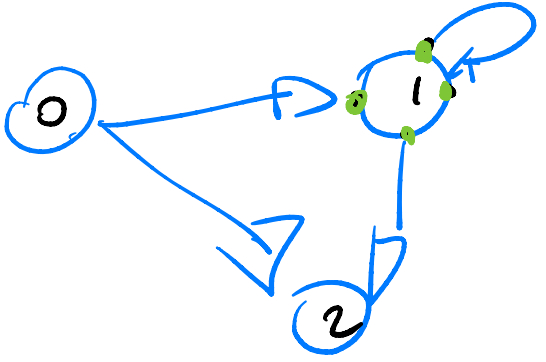
$N^-(u)$, $N^+(u)$

$\{v \in V : (v,u) \in E\}$ $\{v \in V : (u,v) \in E\}$



G orientato

$d_u^- = |N^-(u)|$, $d_u^+ = |N^+(u)|$



$d^-(1) = 2 = |\{(0,1),(1,1)\}|$

$d^+(1) = 2 = |\{(1,1),(1,2)\}|$

$d(1) = d^-(1) + d^+(1)$

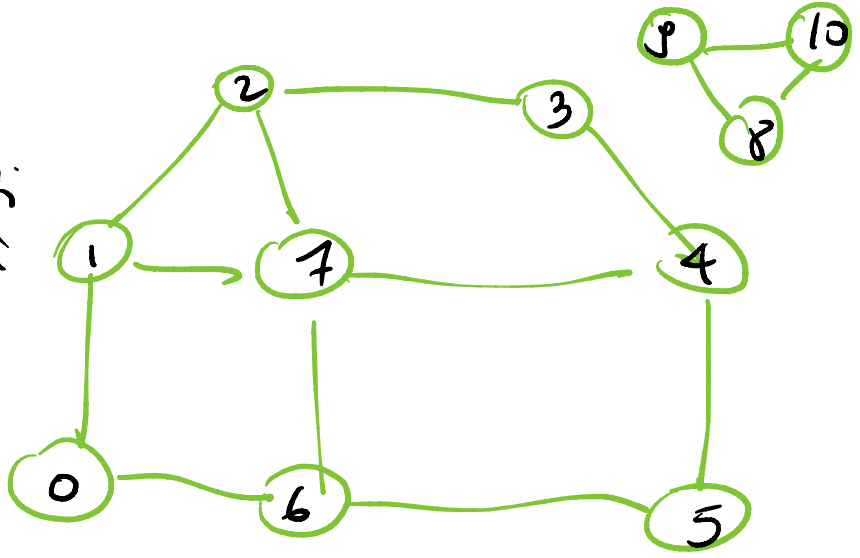
Handshaking lemma:

- G non orientato: $\sum_{u \in V} d_u = 2|E|$ e ci sono un numero pari di nodi u aventi d_u dispari

- G orientato: $\sum_{u \in V} d_u^- = \sum_{u \in V} d_u^+ = |E|$

- cammino [walk, trail, path]
  $u_1 u_2 \dots u_k$ sequenza di nodi in cui nodi adiacenti
  $(u_j, u_{j+1}) \in E$ sono collegati da archi

  $k-1$ = lunghezza cammino
  = # archi attraversati



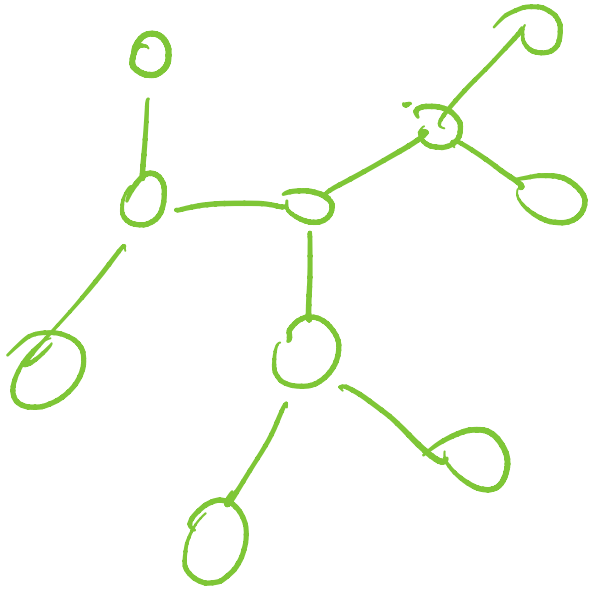
- ciclo quando $u_1 = u_k$ e $k > 1$

- G è **connesso** se esiste un cammino per ogni coppia di nodi

  (se non è connesso, prendiamo le sue componenti connesse)
  cc

- G è **ciclico** se contiene un ciclo; **aciclico** altrimenti

G è aciclico e connesso sse G è un albero



Domanda: come faccio a stabilire ① G connesso ?
② aciclico

VISITA IN PROFONDITA' : DFS depth-first search

```
1  Scansione( G ):
2    FOR (s = 0; s < n; s = s + 1)
3      raggiunto[s] = FALSE;
4    FOR (s = 0; s < n; s = s + 1) {
5      IF (!raggiunto[s]) DepthFirstSearchRicorsiva( s );
6    }
```

G è connesso sse istruzione 5 lancia DFS solo con s=0

```
1  DepthFirstSearchRicorsiva( u ):
2    raggiunto[u] = TRUE;
3    FOR (x = listaAdiacenza[u].inizio; x != null; x = x.succ) {
4      v = x.dato;
5      IF (!raggiunto[v]) DepthFirstSearchRicorsiva(v);
6    }
```

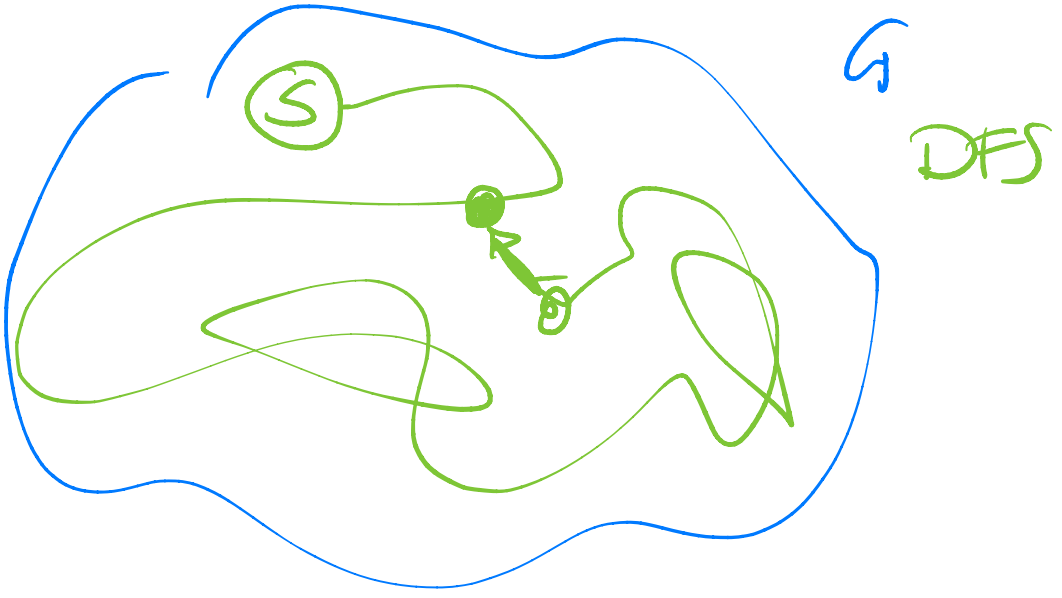
raggiunto ↔ visitato

for (auto v: adj[u])

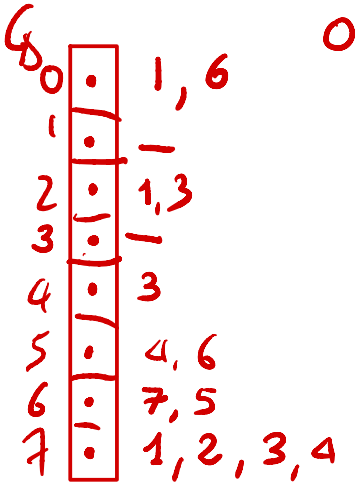
F completato = T

raggiunto[v] = TRUE ⇔ G ciclico
se
(u,v) arco back

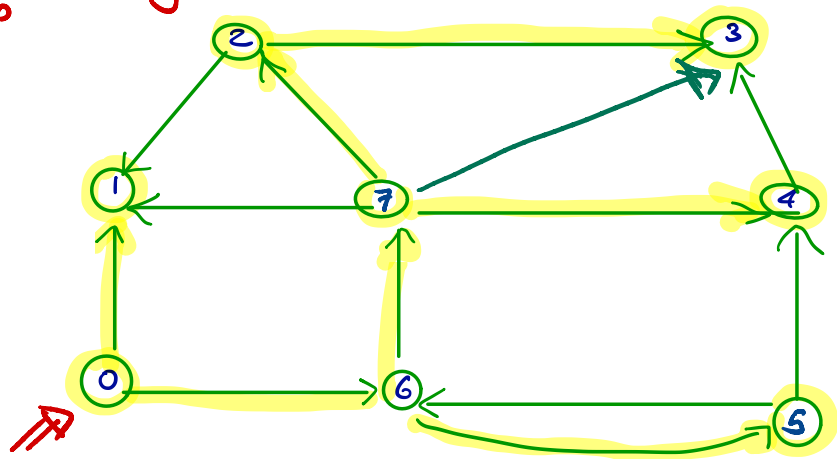
raggiunto[v] = TRUE
se
completato[v] = FALSE

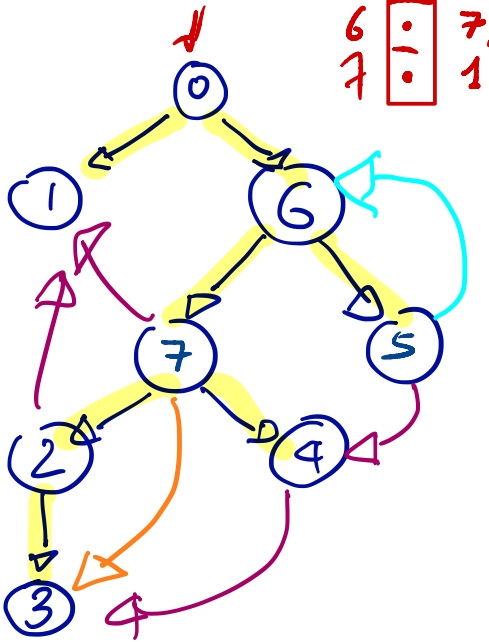

S
G
DFS

Esempio
(liste ordinate)



$O(m+n)$ tempo perché ogni arco viene attraversato 2 volte





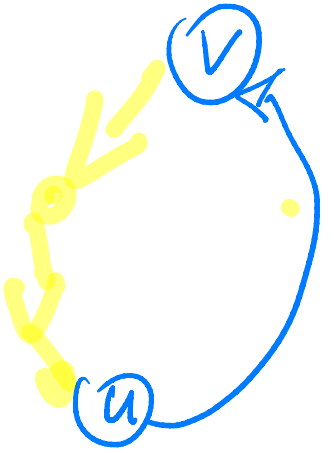
ALBERO DFS

- tree edge (archi dell'albero)
- forward edge (archi in avanti)
- backward edge (archi all'indietro)
- cross edge (archi trasversali)
  vanno da nodo scoperto "dopo" a nodo scoperto "prima"

G ciclico $\Leftrightarrow$ $\exists$ arco (u,v) back

($\Leftarrow$) $\exists$ back

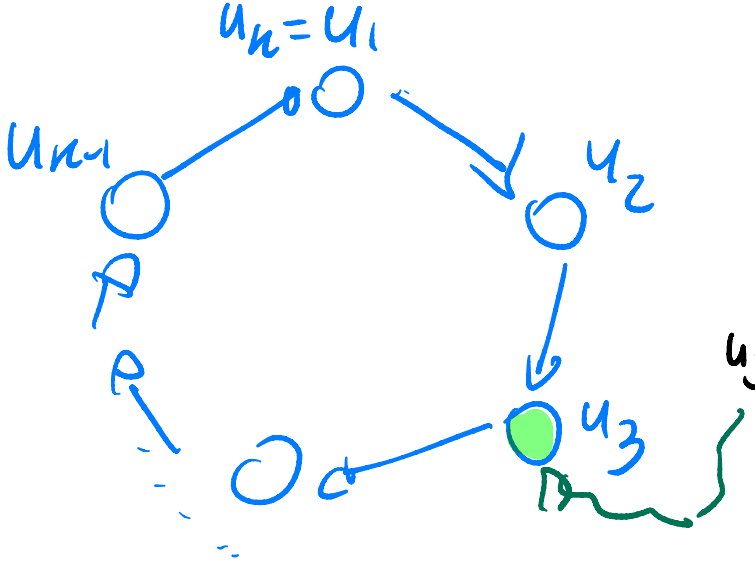
questo implica un cammino di tree edge da v a u: tale cammino diventa un ciclo estendolo con l'arco (u,v)



v antenato u nell'albero DFS

($\Rightarrow$) G ciclico

sia $u_1, u_2 \dots u_k = u_1$ un ciclo in G



Hp. almeno uno tra $u_1, \dots, u_{k-1}$ è il nodo raggiunto dalla DFS

DFS(u_j) scopre tutti gli altri nodi $u_1, \dots, u_{k-1}$ ($\neq u_j$) prima o poi in un certo ordine

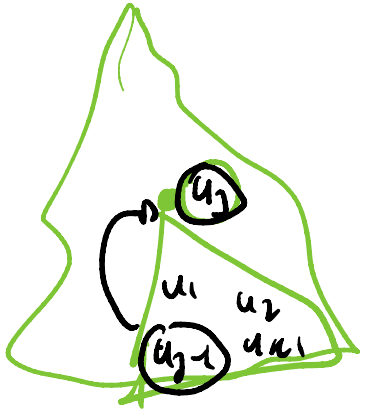
sono tutti quindi discendenti di u_j nell'albero DFS

ALBERO DFS

tra questi c'è anche u_{j-1}, predecessore di u_j nel ciclo $\left(u_j = u_1 \Rightarrow u_{j-1} = u_{k-1}\right)$

quindi (u_{j-1}, u_j) è un arco che collega un discendente a un suo antenato

$\Rightarrow$ è un arco back ∎

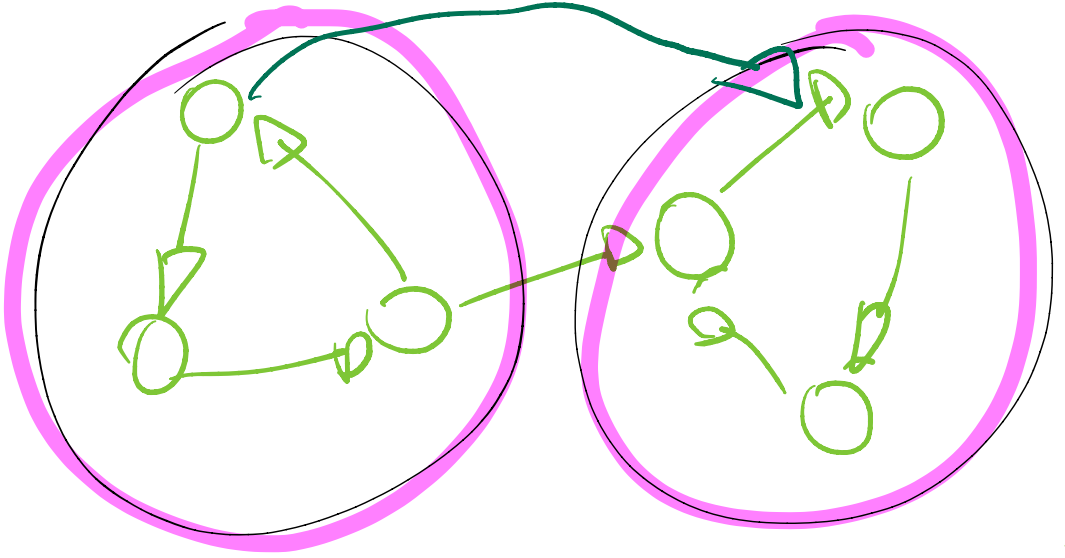


Su G non orientato: ∃ solo BACK

(u, v) back se collega u discendente proprio di v
(cioè v non è il padre di u nell'albero DFS)

G orientato : componente fortemente connessa

SCC

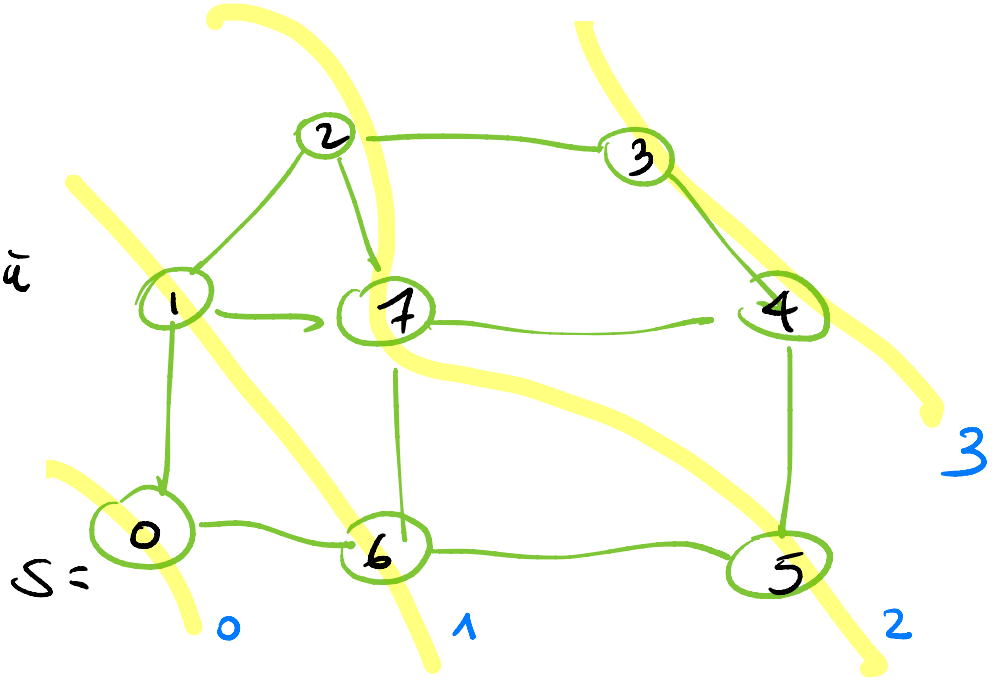


DAG
=
DIRECTED
ACYCLIC
GRAPH

sulle SCC

BFS = Breadth-first search
(visita in ampiezza)

$d(x,y)$ = lunghezza del cammino più breve tra x e y

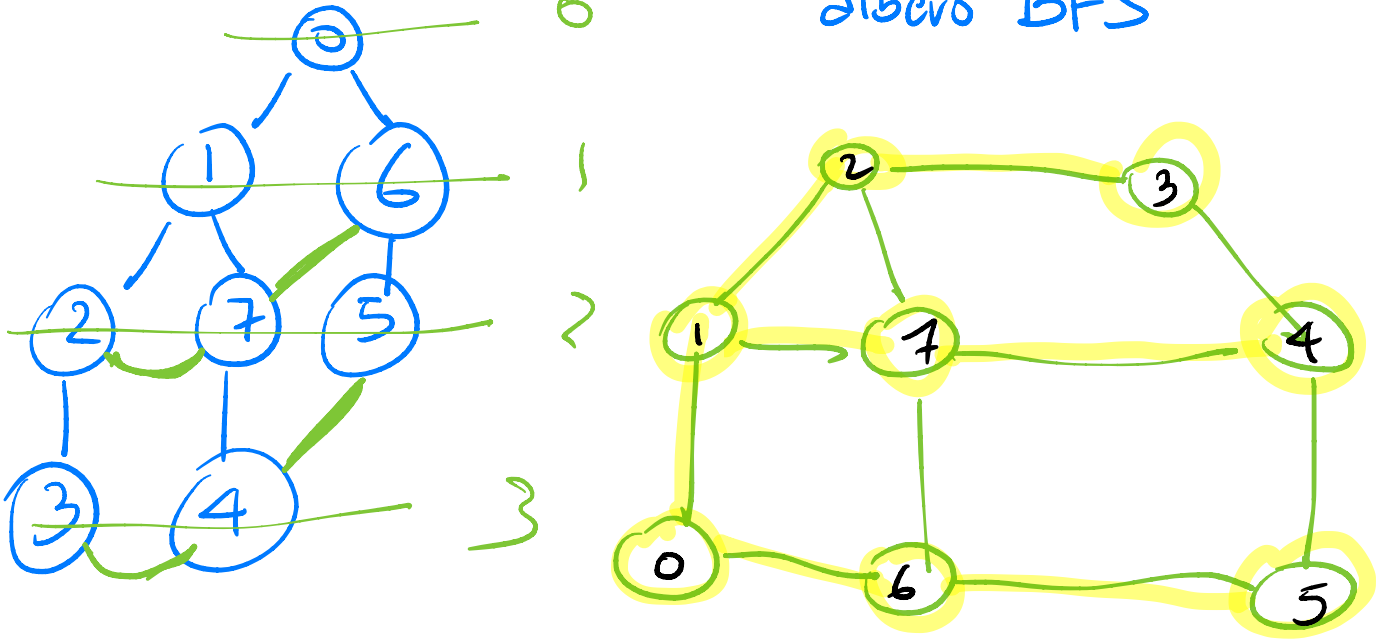


L'ordine di visita dei nodi è compatibile con l'ordine non-decrescente basato su $d(s,-)$

BFS(s):

```
1  for (u=0; u<n; u++) visitato[u] = false;
2  C.enqueue(s); visitato[s] = true;   d_s[s] = 0;
3  while (!C.empty()){
4        u = C.dequeue();        // d_s[u] è ben definito
5        for (auto v : adj[u]){
6              if (!visitato[v]) {C.enqueue(v); visitato[v] = true;}
   }
```

$d_s[v] = d_s[u] + 1$



$(u,v) \Rightarrow |d_s[u] - d_s[v]| \leq 1$

Complessità

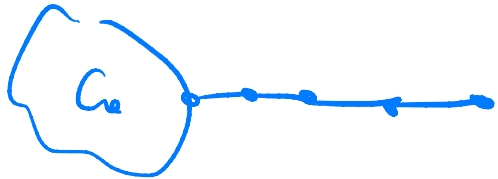
DFS e BFS richiedono tempo lineare $O(n+m)$

"ogni lista di adiacenza viene incrementalmente scandita solo in avanti, sempre nella stessa direzione"

distanza media $\sum_{u \neq v} \frac{d(u,v)}{n(n-1)}$

(esperimento di MILGRAM "6 gradi di separazione")

diametro $\max_{u \neq v} d(u,v)$

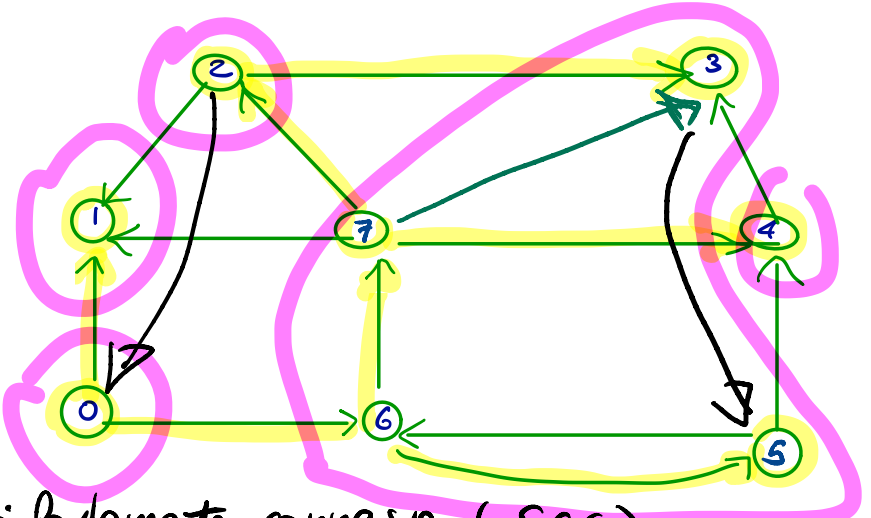


↳ $\max_{u \neq v} d(u,v) = \max_{s} \underbrace{\left(\max_{v} d(s,v) \right)}_{BFS(s)}$

$O(n \cdot BFS) = O(n^2 + nm)$ tempo

↳ se G è denso ⇒ metodi basati sulla moltiplicazione tra matrici

↳ se G è sparso ⇒ $O(n^2)$ tempo NON si può fare meno di così SETH



Componenti fortemente connesse (SCC)

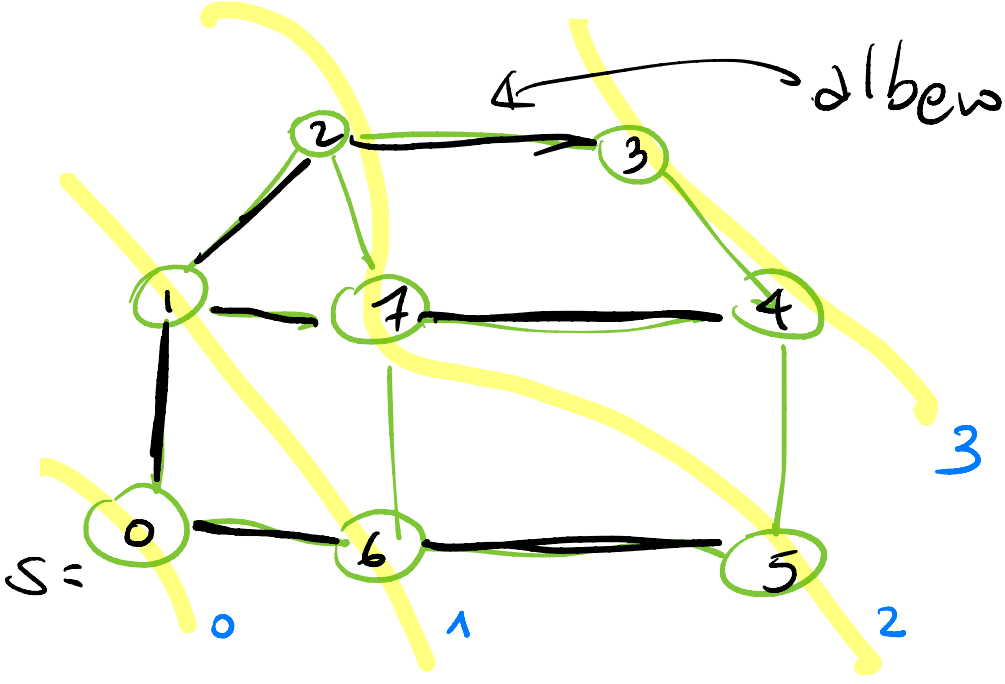
DFS aiuta a calcolare le SCC in $O(m+n)$ tempo

$\forall u,v \in SCC \Leftrightarrow \exists$ cammino $u \rightsquigarrow v$ e cammino $v \rightsquigarrow u$ (orientati $\forall$)

$\Leftrightarrow$ u, v sono su un ciclo orientato

C

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



- tree edge
- non-tree edge
  - collegano nodi che sono
    - stesso livello oppure
    - il loro livello differisce di 1