

La visita fornisce informazioni sul grafo visitato

IDEA DI BASE

Vertici divisi in

- bianchi → non ancora visitati
- grigi → visitati ma di cui non sono ancora stati scoperti tutti gli adiacenti
- neri → scoperti con anche gli adiacenti

ALGORITMO

INIZIALIZZA(G)

for $\forall u \in V$ do
 $u.\text{color} \leftarrow \text{bianco}$

→ all'inizio saranno tutti da visitare

VISITA(G, s)
nodo di partenza

$s.\text{color} \leftarrow \text{grigio}$
while \exists nodo grigio do
 $u \leftarrow$ nodo grigio
 if $\exists v$ bianco $\in \text{adj}[u]$ then
 $v.\text{color} \leftarrow \text{grigio}$
 else
 $u.\text{color} \leftarrow \text{black}$

scelgo un nodo grigio (vedremo come)

Se il grafo è connesso, l'algoritmo visita tutti i nodi e partire da s .

Se G non è connesso → visita completa

PROPRIETA' E INVARIANTI

- il colore può passare solo da bianco a grigio a nero.
- se $(u, v) \in E$ e u è nero : v è grigio o nero
- tutti i vertici grigi o neri sono raggiungibili da s .
- qualunque cammino da s ad un nodo bianco deve contenere almeno un grigio

Al termine dell'algoritmo :

v è nero $\Leftrightarrow v$ raggiungibile da s

COSTRUZIONE DEL SOTTOGRAFO DEI PREDECESSORI

A vertice scoperto, voglio ricordare quale vertice grigio ha permesso di scoprirla

⇒ Associo ad ogni nodo un attributo che memorizza il nodo dal quale è stato scoperto

ALGORITMO :

```
INIZIALIZZA( $G$ )
for  $\forall u \in V$  do
     $u.\text{color} \leftarrow \text{bianco}$ 
     $u.\pi \leftarrow \text{nil}$ 
```

```
VISITA( $G, s$ )
 $s.\text{color} \leftarrow \text{grigio}$ 
while  $\exists$  nodo grigio do
     $u \leftarrow$  nodo grigio
    if  $\exists v \text{ bianco} \in \text{adj}[u]$  then
         $v.\text{color} \leftarrow \text{grigio}$ 
         $v.\pi \leftarrow u$ 
    else
         $u.\text{color} \leftarrow \text{black}$ 
```

Al termine della visita,
l'unico vertice nero con predecessore $\pi = \text{null}$ è s

N.B. il sottografo dei predecessori è un albero

Come accennato sopra, per visitare un grafo non连通, bisogna effettuare una visita completa

ALGORITMO : VISITA-TUTTI-VERTICI(G)

```
INIZIALIZZA( $G$ )
for  $\forall u \in V$  do
    if  $u.\text{color} = \text{bianco}$  then
        VISITA( $G, u$ )
```

↳ il sottografo dei predecessori diventa una foresta!
foresta di scoperta

VERSIONE "CONCRETA" DELL'ALGORITMO

- Uso struttura D per gestire l'insieme dei vertici grigi
- Operazioni necessarie :
 - ▶ **MAKE-EMPTY**: crea una struttura nuova
 - ▶ **FIRST(D)**: restituisce il primo elemento (senza modificare D)
 - ▶ **ADD(D, x)**: aggiunge l'elemento x a D
 - ▶ **REMOVE-FIRST(D)**: toglie da D il primo elemento
 - ▶ **NOT-EMPTY(D)**: restituisce true se D non è vuoto, false altrimenti

VISITA(G, s)

```
 $D \leftarrow \text{MAKE-EMPTY}$ 
 $s.\text{color} \leftarrow \text{grigio}$ 
 $\text{ADD}(D, s)$ 
while  $\text{NON-EMPTY}(D)$  do
     $u \leftarrow \text{FIRST}(D)$ 
    if  $\exists v \text{ bianco} \in \text{adj}[u]$  then
         $v.\text{color} \leftarrow \text{grigio}$ 
         $v.\pi \leftarrow u$ 
         $\text{ADD}(D, v)$ 
    else
         $u.\text{color} \leftarrow \text{black}$ 
         $\text{REMOVE-FIRST}(D)$ 
```

Come viene implementato ?

- il ciclo parte da dove e' arrivato l'ultima volta
- associa ad ogni vertice il valore corrente del puntatore alla lista di adiacenti
⇒ lista di adj percorse una sola volta

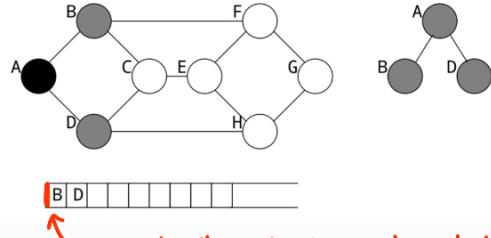
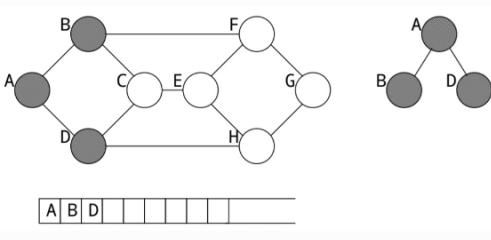
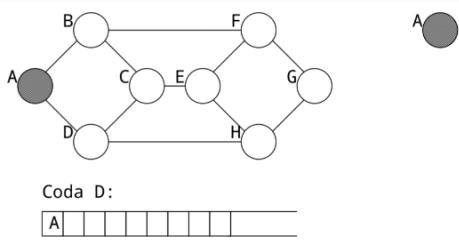
Complessità

- Inizializzazione : $O(|V|)$
 - Tempo dedicato alle operazioni su D : $O(|V|)$
 - Tempo dedicato alla ricerca di nodi bianchi : $O(|E|)$
- Tempo Totale : $O(|V| + |E|)$

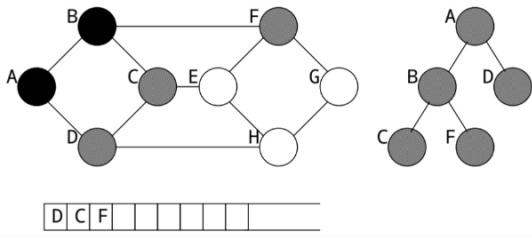
VISITA IN AMPIEZZA (BFS)

La struttura D e' una coda \Rightarrow FIFO

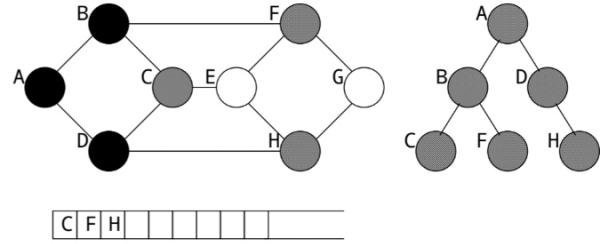
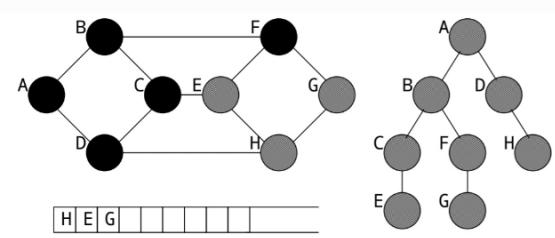
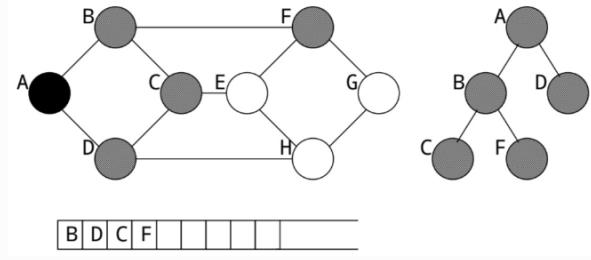
esempio



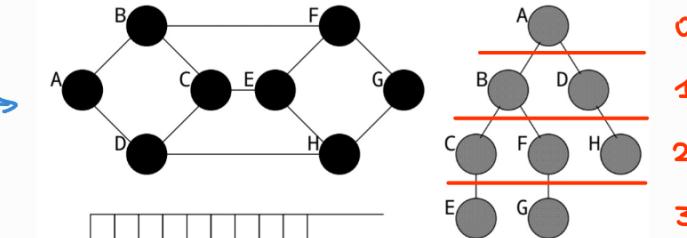
quando il nodo A e' diventato nero
(esplorati gli adiacenti) e' stato
rimosso dalle coda



procedo visitando
gli adj del nodo in
testa



...



N.B. il livello indica quanti archi di distanza ci sono da A

\Rightarrow trova le minime distanze di un nodo dalla radice
(cammino minimo con costi unitari)

VISITA(G, s)

$D \leftarrow \text{MAKE-EMPTY}$

$s.\text{color} \leftarrow \text{grigio}$

$\text{ADD}(D, s)$

while $\text{NON-EMPTY}(D)$ **do**

$u \leftarrow \text{FIRST}(D)$

if $\exists v$ bianco $\in \text{adj}[u]$ **then**

$v.\text{color} \leftarrow \text{grigio}$

$v.\pi \leftarrow u$

$\text{ADD}(D, v)$

else

$u.\text{color} \leftarrow \text{nero}$

$\text{REMOVE-FIRST}(D)$

SECONDA VERSIONE

```
VISITA( $G, s$ )
 $D \leftarrow \text{MAKE-EMPTY}$ 
 $s.\text{color} \leftarrow \text{grigio}$ 
 $\text{ADD}(D, s)$ 
while NON-EMPTY( $D$ ) do
     $u \leftarrow \text{FIRST}(D)$ 
    for  $\forall v : v \text{ è bianco ed è } \in \text{adj}[u]$  do
         $v.\text{color} \leftarrow \text{grigio}$ 
         $v.\pi \leftarrow u$ 
         $\text{ADD}(D, v)$ 
     $u.\text{color} \leftarrow \text{black}$ 
     $\text{REMOVE-FIRST}(D)$ 
```

il primo elemento della coda non cambia finché ci sono adiacenti bianchi

TERZA VERSIONE

```
VISITA( $G, s$ )
 $D \leftarrow \text{MAKE-EMPTY}$ 
 $s.\text{color} \leftarrow \text{grigio}$ 
 $\text{ADD}(D, s)$ 
while NON-EMPTY( $D$ ) do
     $u \leftarrow \text{FIRST}(D)$ 
     $ptr \leftarrow \text{adj}[u]$ 
    while  $ptr \neq \text{nil}$  do
         $v \leftarrow ptr.\text{vtx}$ 
        if  $v.\text{color} = \text{bianco}$  then
             $v.\text{color} \leftarrow \text{grigio}$ 
             $v.\pi \leftarrow u$ 
             $\text{ADD}(D, v)$ 
         $ptr \leftarrow ptr.\text{next}$ 
     $u.\text{color} \leftarrow \text{black}$ 
     $\text{REMOVE-FIRST}(D)$ 
```

Ogni elemento nella lista di adiacenti ha due campi:

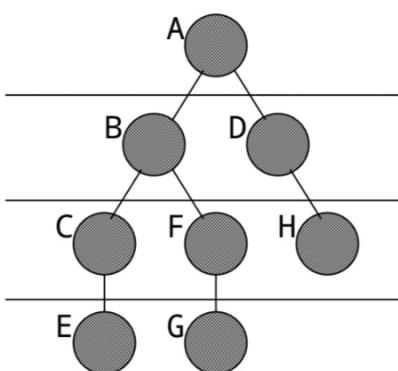
vtx = vertice

next = prossimo elem.
nella lista

ALBERO DI BFS

- viene costruito a livelli
- la costruzione del livello $n+1$ comincia solo dopo aver concluso la costruzione del livello n

→ associamo ad ogni nodo un attributo d che ricorda il livello



```
INIZIALIZZA( $G$ )
for  $\forall u \in V$  do
     $u.\text{color} \leftarrow \text{bianco}$ 
     $u.\pi \leftarrow \text{nil}$ 
     $u.d \leftarrow \infty$ 
```

```
VISITA( $G, s$ )
 $D \leftarrow \text{MAKE-EMPTY}$ 
 $s.\text{color} \leftarrow \text{grigio}$ 
 $s.d \leftarrow 0$ 
 $\text{ADD}(D, s)$ 
while NON-EMPTY( $D$ ) do
     $u \leftarrow \text{FIRST}(D)$ 
    for  $\forall v : v \text{ è bianco ed è } \in \text{adj}[u]$  do
         $v.\text{color} \leftarrow \text{grigio}$ 
         $v.\pi \leftarrow u$ 
         $v.d \leftarrow u.d + 1$ 
         $\text{ADD}(D, v)$ 
     $u.\text{color} \leftarrow \text{black}$ 
     $\text{REMOVE-FIRST}(D)$ 
```

VISITA IN PROFONDITÀ (DFS)

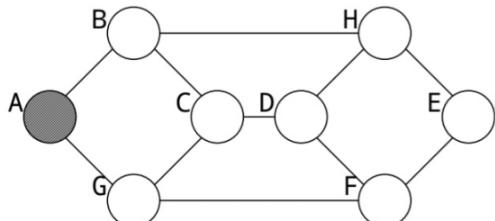
D è una PILA (LIFO)

VISITA(G, s)

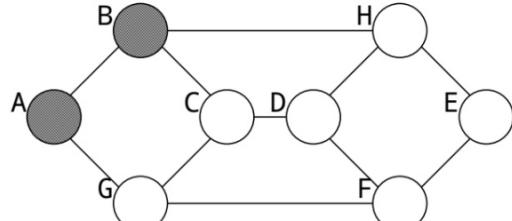
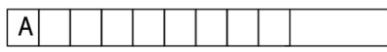
```

 $D \leftarrow \text{MAKE-EMPTY}$ 
 $s.\text{color} \leftarrow \text{grigio}$ 
 $\text{ADD}(D, s)$ 
while NON-EMPTY( $D$ ) do
     $u \leftarrow \text{FIRST}(D)$ 
    if  $\exists v \text{ bianco} \in \text{adj}[u]$  then
         $v.\text{color} \leftarrow \text{grigio}$ 
         $v.\pi \leftarrow u$ 
         $\text{ADD}(D, v)$ 
    else
         $u.\text{color} \leftarrow \text{black}$ 
         $\text{REMOVE-FIRST}(D)$ 
```

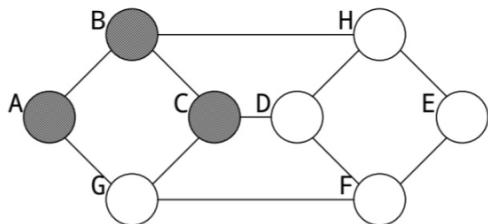
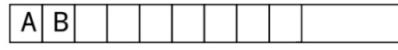
esempio



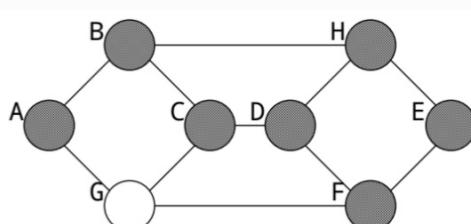
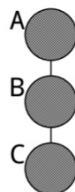
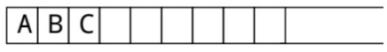
Pila:



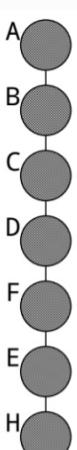
Pila D:



Pila D:

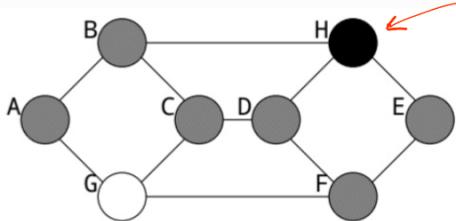


Pila D:



non ci sono piu' adiacenti bianchi
⇒ comincio a risalire

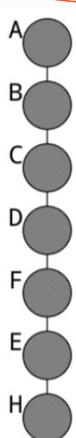
finche' non mi trovo in un nodo con adiacenti bianchi



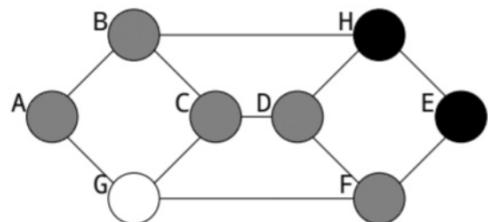
Pila D:



H ora è nero ed è stato
rimosso dalla pila



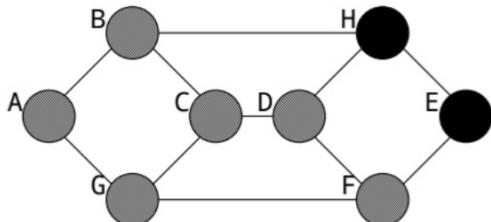
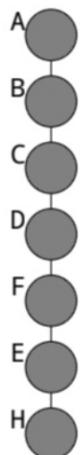
continua nella
pagina seguente



Pila D:

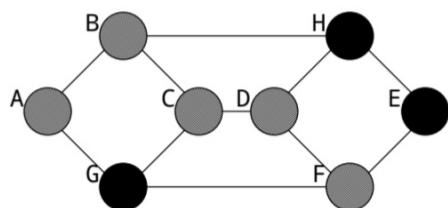
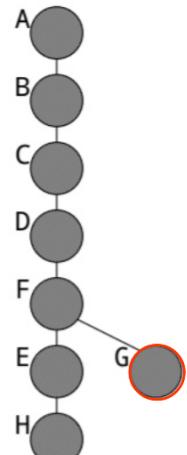
A	B	C	D	F			
---	---	---	---	---	--	--	--

ha adiacenti bianchi



Pila D:

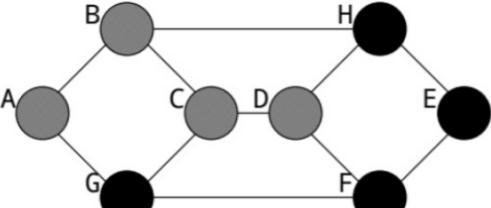
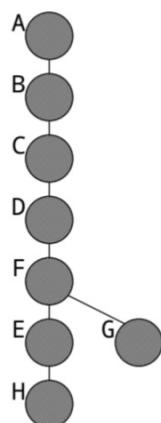
A	B	C	D	F	G		
---	---	---	---	---	---	--	--



Pila D:

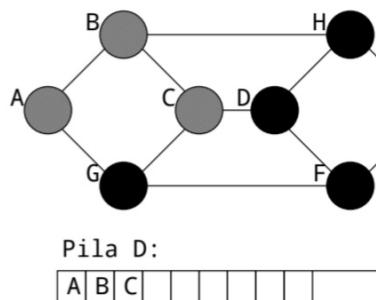
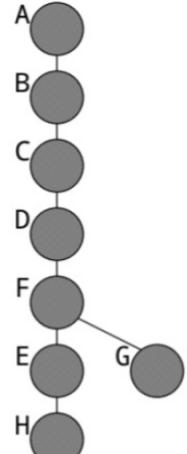
A	B	C	D	F			
---	---	---	---	---	--	--	--

Ricomincio la risalita



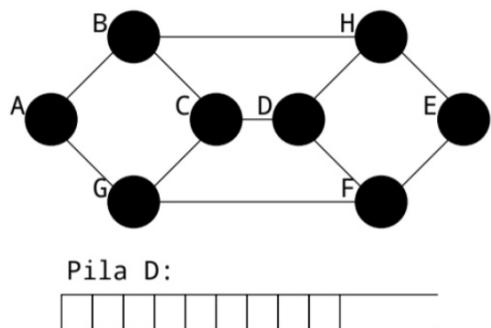
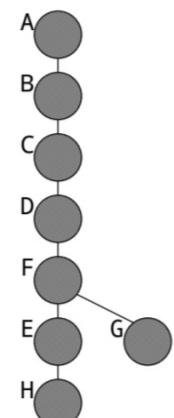
Pila D:

A	B	C	D				
---	---	---	---	--	--	--	--



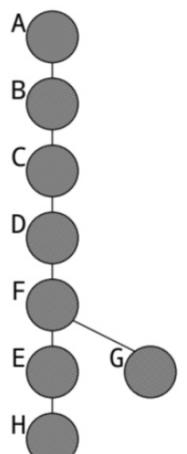
Pila D:

A	B	C					
---	---	---	--	--	--	--	--

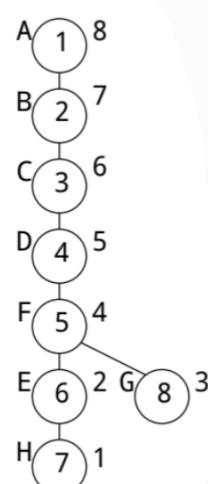
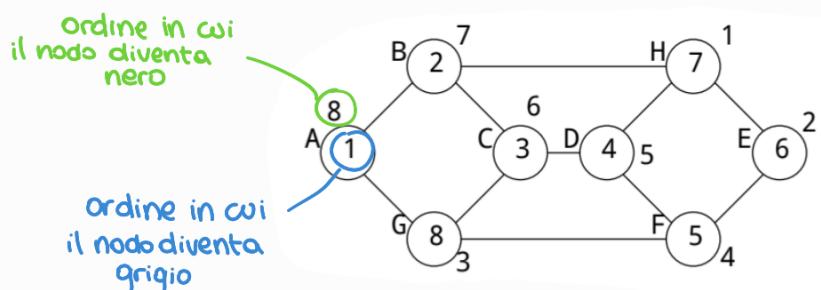


Pila D:

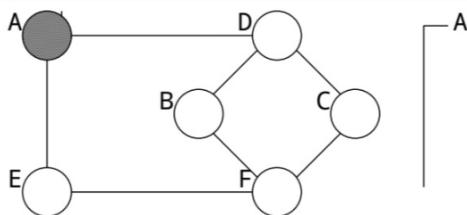
--	--	--	--	--	--	--	--



Ordine di scoperta dei nodi :



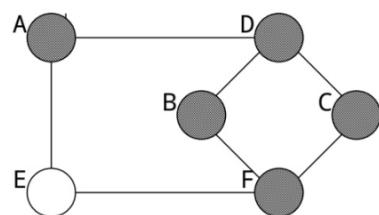
Altro esempio :



Pila:

A						
---	--	--	--	--	--	--

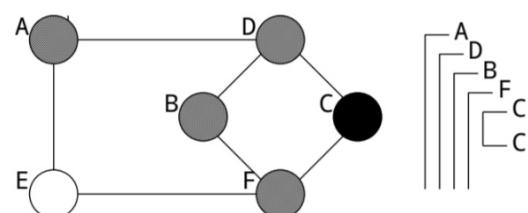
A



Pila:

A	D	B	F	C		
---	---	---	---	---	--	--

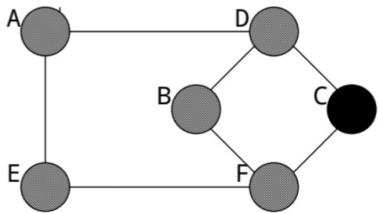
A
D
B
F
C



Pila:

A	D	B	F			
---	---	---	---	--	--	--

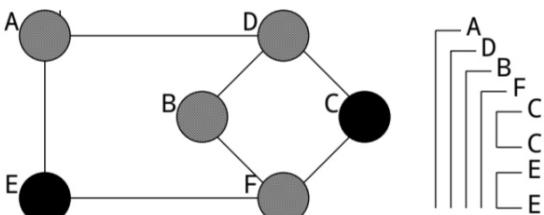
A
D
B
F
C
C



Pila:

A	D	B	F	E		
---	---	---	---	---	--	--

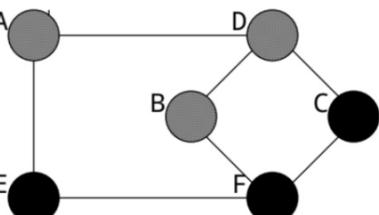
A
D
B
F
C
C
E



Pila:

A	D	B	F			
---	---	---	---	--	--	--

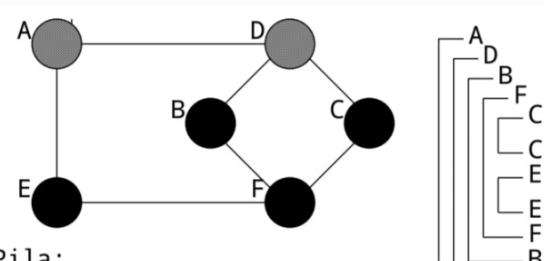
A
D
B
F
C
C
E
E
F
B



Pila:

A	D	B				
---	---	---	--	--	--	--

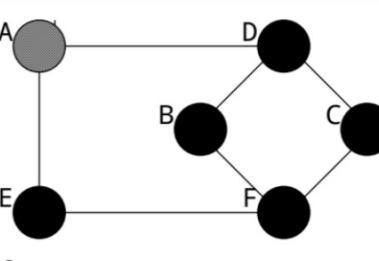
A
D
B
F
C
C
E
E
F
B
D



Pila:

A	D					
---	---	--	--	--	--	--

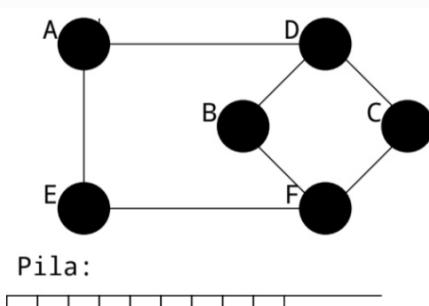
A
D
B
F
C
C
E
E
F
B



Pila:

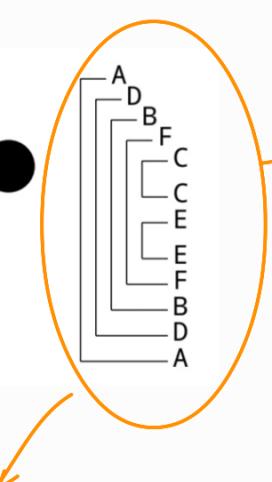
A						
---	--	--	--	--	--	--

A
D
B
F
C
C
E
E
F
B
D



Pila:

--	--	--	--	--	--	--



i livelli di attivazione di una coppia di nodi sono:

- disgiunti oppure
- uno interamente contenuto nell'altro

un vertice non viene disattivato finché non sono stati attivati e disattivati tutti i suoi discendenti

→ suggerisce una procedura ricorsiva

VERSIONE RICORSIVA DI DFS

```
VISITA( $G, s$ )
 $s.\text{color} \leftarrow \text{grigio}$ 
while  $\exists v : v \text{ è bianco ed è } \in \text{adj}[s]$  do
     $v.\pi \leftarrow s$ 
    VISITA( $G, v$ )
     $s.\text{color} \leftarrow \text{nero}$ 
```

\rightarrow non serve più la pila D

\downarrow versione estesa per raccogliere informazioni

```
VISITA( $G, s$ )
 $s.\text{color} \leftarrow \text{grigio}$ 
 $s.d \leftarrow \text{time}$   $\rightarrow$  inizio visita
 $\text{time} \leftarrow \text{time} + 1$ 
while  $\exists v : v \text{ è bianco ed è } \in \text{adj}[s]$  do
     $v.\pi \leftarrow s$ 
    VISITA( $G, v$ )
     $s.f \leftarrow \text{time}$   $\rightarrow$  fine visita
     $\text{time} \leftarrow \text{time} + 1$ 
     $s.\text{color} \leftarrow \text{nero}$ 
```

time = contatore per ricordare l'ordine delle disattivazioni

\downarrow

```
INIZIALIZZA( $G$ )
for  $\forall u \in V$  do
     $u.\text{color} \leftarrow \text{bianco}$ 
     $u.\pi \leftarrow \text{nil}$ 
     $u.d \leftarrow \infty$ 
     $u.f \leftarrow \infty$ 
     $\text{time} \leftarrow 1$ 
```

i tempi di un nodo non visitato rimangono infiniti

ATTENZIONE \rightarrow visita completa solo se il grafo è connesso !

VISITA INTERA DI UN GRAFO NON CONNESSO :

VISITA-TUTTI-VERTICI(G)

INIZIALIZZA(G)

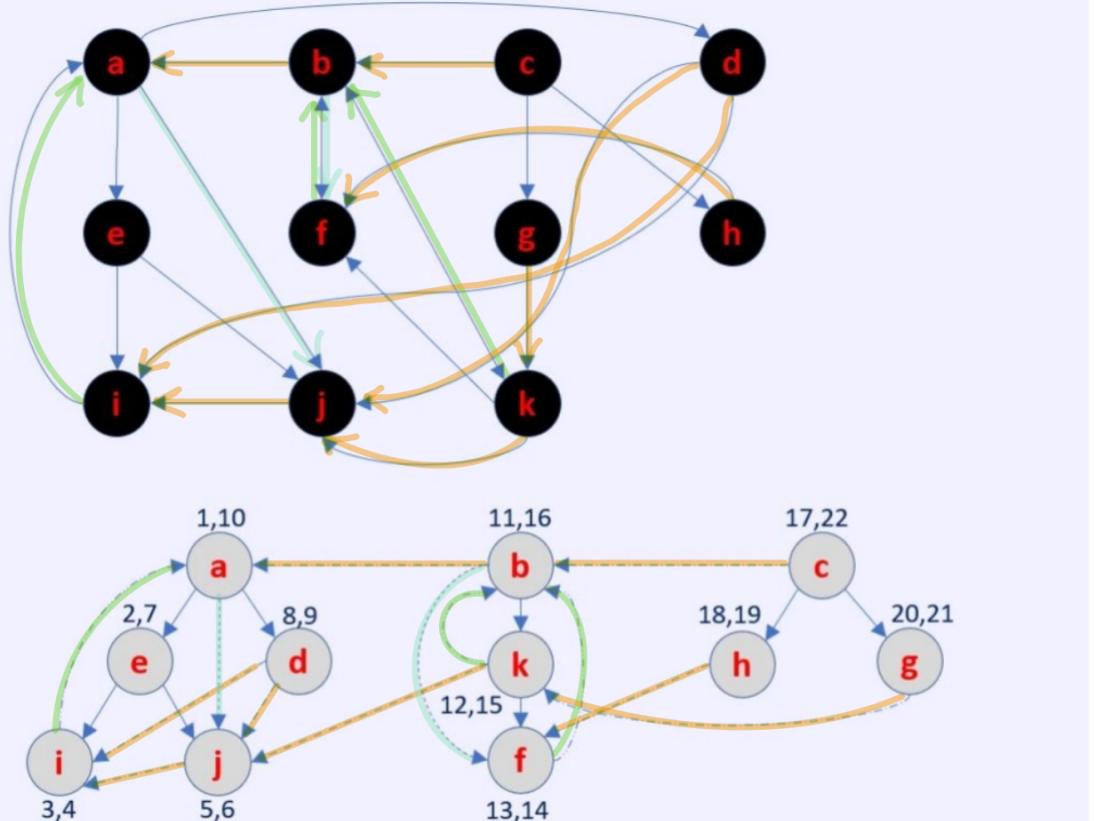
for $\forall u \in V$ **do**

if $u.\text{color} = \text{bianco}$ **then**

VISITA(G, u)

CLASSIFICAZIONE DEGLI ARCHI

a:	e	d	j
b:	a	k	f
c:	b	h	g
d:	i	j	
e:	i	j	
f:	b		
g:	k		
h:	f		
i:	a		
j:	i		
k:	b	j	f



- Sono tutti da dx a sx

• sono gli archi del grafo originale che non sono stati usati per la visita

da nodo a suo avo → mai presente se il grafo è aciclico

↳ lo capisco quando vado da nodo grigio a nodo grigio

da nodo ad adiacente (usato nella visita)