

## NUMERO DI SCC IN UN GRAFO

Un grafo ha **ALMENO** una componente (deve avere almeno un nodo).

Un grafo ha **AL PIÙ**  $|V|$  componenti. Nel caso di un grafo aciclico sono esattamente  $|V|$ .

## PROPRIETÀ

DATA UNA SCC  $C$  E DUE VERTICI  $x, y \in C$ , OGNI PERCORSO DA  $x$  A  $y$  CONTIENE SOLO VERTICI DI  $C$ .

## DIMOSTRAZIONE

Per assurdo  $\exists z \notin C$  che appartiene ad un percorso da  $x$  a  $y$ .

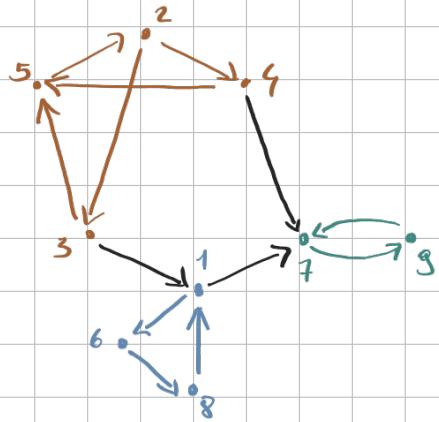
Possiamo arrivare da  $x$  a  $z$ , da  $z$  a  $y$ . Per ipotesi abbiamo due percorsi da  $x$  a  $y$  e viceversa. Ma allora possiamo effettuare  $z \rightarrow y \rightarrow x \rightarrow z$  quindi possiamo arrivare da  $x$  a  $z$  e viceversa, quindi  $z \in C$ .

## TEOREMA

DATO  $G = \langle V, E \rangle$ , AL TERMINE DI UNA DFS ( $G$ ), TUTTI I VERTICI DI UNA STESSA COMPONENTE APPARTENGONO ALLO STESSO ALBERO DELLA FDF.

E' possibile che uno stesso albero contenga più SCC

Ese.



In questo caso la DFS costruirebbe un solo albero (perché non ci sono nodi isolati) però il grafo non è una SCC, bensì ne contiene 3.

### Dimostrazione

Prendiamo  $G$  e una sua componente  $C$ .

La DFS visita tutti i nodi in momenti diversi.

Fissiamone un primo grigio  $u$ , gli altri sono ancora bianchi all'istante di  $u$ . Sia  $u \in C$ , tutti i percorsi da  $u$  agli altri vertici di  $C$  appartengono a  $C$ . Ma quindi tutti i percorsi di  $C$  sono bianchi e per il teorema del percorso bianco sono tutti discendenti di  $u$ , quindi appartengono tutti all'albero radicato in  $u$ .

### TRASPOSTA DI UN GRAFO

Dato  $G = \langle V, E \rangle$  grafo, la trasposta di  $G$  è  $G^T = \langle V, E^T \rangle$

$$\text{dove } E^T = \{(v, u) \mid (u, v) \in E\}$$

OSS

la trasposizione conserva la reciproca raggiungibilità.

### INDIVIDUARE LE SCC IN UN GRAFO

Il problema principale sta nel fatto che un albero può contenere più SCC e noi in qualche modo dobbiamo isolare.

Sfrutteremo gli alberi perché ci danno informazioni sulla raggiungibilità dei nodi, in particolare preso un nodo  $s$  se eseguiamo la DFS partendo da questo, i suoi discendenti (nell'albero) sono i nodi da esso raggiungibili. I potremmo poi gli eseguire una seconda DFS sui nodi dell'albero, potremmo verificare quali nodi sono mutuamente raggiungibili, ci sono però altre considerazioni da fare.

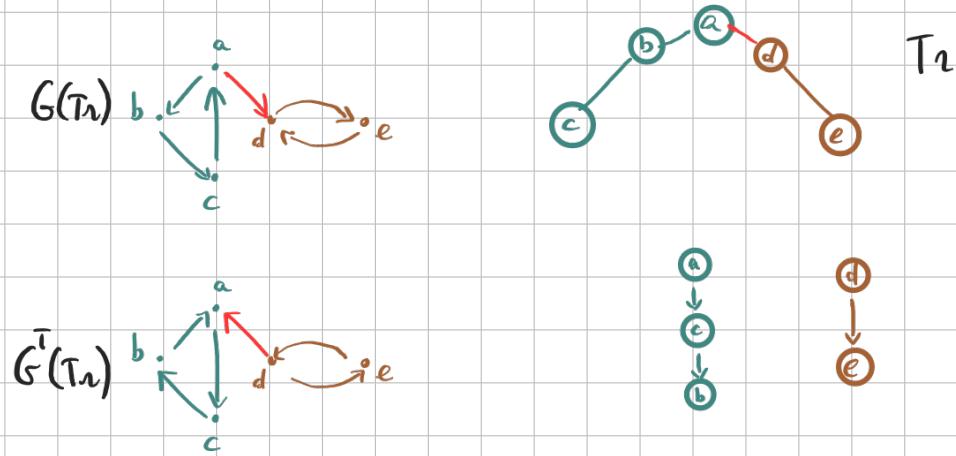
Sia  $T_1$  un albero della FDF, e la sua radice  $r$  lo sua radice e  $G(T_1)$  grafo indotto su  $T_1$ , ovvero il grafo dal quale deriva  $T_1$ .

Vogliamo dimostrare che se un nodo  $x$  è raggiungibile da  $r$  in  $T_1$  (e quindi sicuramente anche in  $G(T_1)$ ), se  $x$  è ancora raggiungibile da  $r$  in  $G^T(T_1)$  allora sono mutuamente connesse.

Se  $x \in T_1$  sicuramente **NON** appartiene alla stessa SCC per la proprietà precedente.

Vediamo un esempio:

Partiamo da a



la prima DFS ci dà la raggiungibilità in un verso, la seconda nell'altro verso.

Chiamiamo quindi  $\text{DFS}_1$  la visita su  $G$  e  $\text{DFS}_2$  quella su  $G'$ .

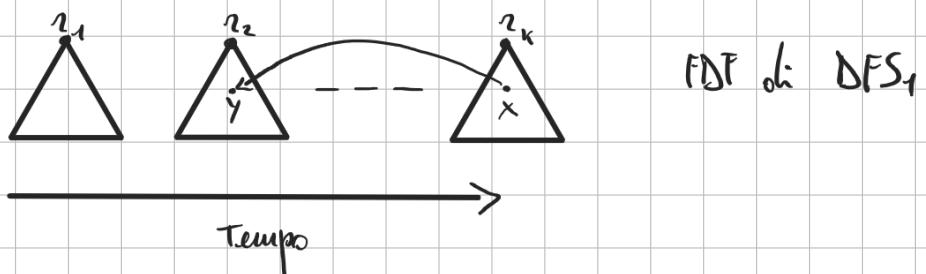
La foresta generata da  $\text{DFS}_1$  può presentare **archi di attraversamento**, questo è un problema per  $\text{DFS}_2$  perché quando incontrerà quell'arco cambierà albero impedendo di trovare tutte le SCC di  $T_1$ .

Dobbiamo quindi forzare  $\text{DFS}_2$  a stare in  $T_1$ .

La DFS termina quando, in un percorso, arriviamo in un modo che ha solo adiacenti non binormali.

Quindi se mai ci assicuriamo che il modo di arrivare dell'arco di attraversamento di  $\text{DFS}_2$  è già stato visitato, faranno  $\text{DFS}_2$  a continuare in  $T_2$ .

Facciamo una rappresentazione grafica di come viene costruita FDF nel tempo

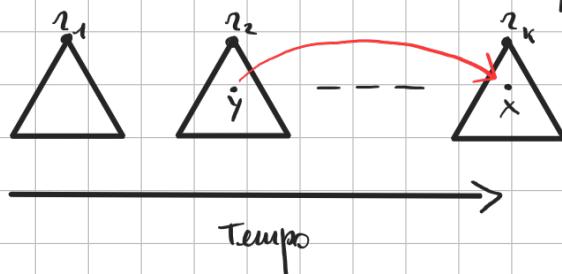


Gli archi di attraversamento vanno solo "all'indietro nel tempo"

se modi già scoperti, all'istante  $d(y) < e'$  binario quando  $(y, x)$  non può essere d'attraversamento. Ci sono due possibilità per l'arco  $(y, x)$ : dopo  $d(y)$ , quando viene attraversato l'arco  $x$  è ancora binario (quindi  $x$  è figlio di  $y$  e starà nel suo stesso albero) oppure  $x$  è grigio perché sono stati attraversati altri archi.

Quindi  $(x, y)$  è un arco dell'albero o di attraversamento.

Ma allora nella FDF di  $\text{DFS}_2$  accadrà questo:



Quindi nell' albero radicato in  $r_K$  possono solo entrare archi e non uscire.

Quindi se in  $G^T$  facciamo partire la DFS<sub>2</sub> dall'albero radicato in  $r_K$ , avremo risolto il problema degli archi d'attraversamento e in più sappiamo che i nodi di quest'albero sono per forza mutuamente raggiungibili, in particolare sono gli unici.

Quindi ricapitolando,  $\text{DFS}_2(G^T, r_K)$  individua tutte le SCC.

Prendiamo in esame l'albero in  $r_K$



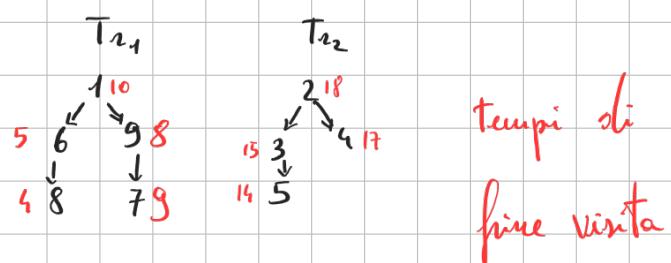
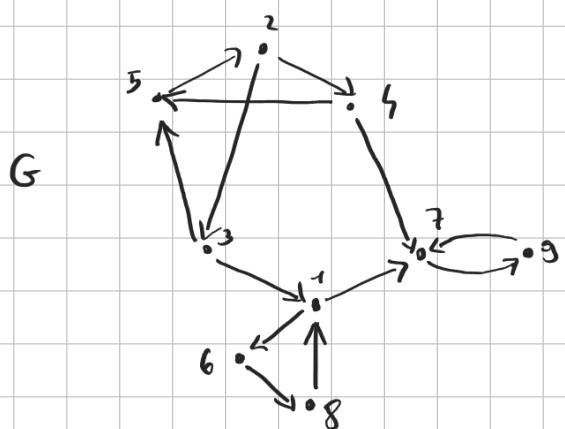
Al termine della DFS su  $r_K$  ci saranno tutti i vertici neri di una SCC, ma sappiamo che in un albero sono contenuti potenzialmente più sott'alberi di SCC. Immaginiamo ci siano due sott'alberi che hanno tra loro un arco d'attraversamento (che è stato trasposto in maniera analoga). Prendiamo la radice dell'sott'albero costituito per ultimo, chiamiamola  $i'$ . Possiamo applicare a questo vertice lo stesso ragionamento di  $r_K$  e proseguire ricorsivamente.

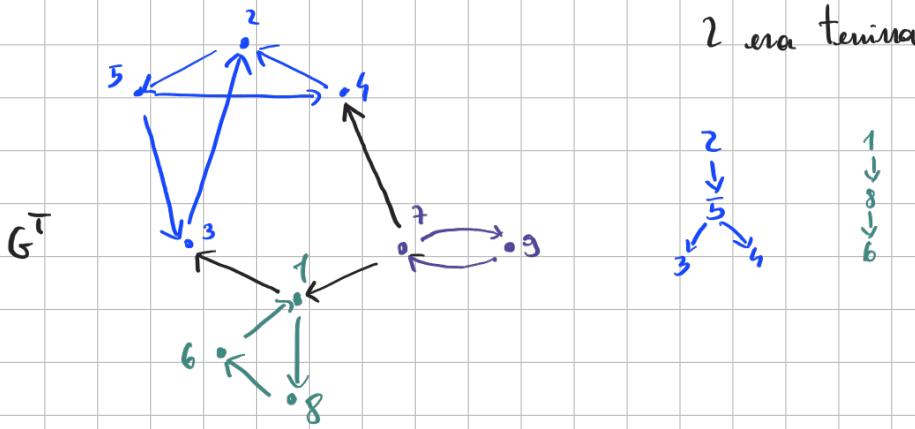
L'idea è quella di eseguire  $\text{DFS}_2$  su  $G^T$  ma scegliere opportunamente le sorgenti delle visite sul trasposto ( $v, v''..$ ), a differenza di  $\text{DFS}_1$ .

$f(v_n)$  è l'ultimo tempo registrato in  $\text{DFS}_1$ , sia rispetto agli altri alberi che rispetto al suo stesso albero (per il Teorema della struttura a parentesi). Quindi  $v_k$  è il vertice che termina per ultimo in  $\text{DFS}_1$ , ed è proprio secondo questo criterio che sceglieremo le sorgenti. I passaggi dell'algoritmo quindi sono:

- ① Esegue  $\text{DFS}_1(G)$  per calcolare i tempi di fine visita
- ② Calcola  $G^T$
- ③ Esegue  $\text{DFS}_2(G^T)$  selezionando in ordine decrescente i tempi di fine visita

Vediamo un esempio per chiarire:





2 era terminato per ultimo



I vecchi archi d'attraversamento  $(1,3)$ ,  $(7,4)$  sono stati invertiti  
e non vengono attraversati perché trovano l'arrivo nero

## ALGORITMO DI RICERCA SCC

- $\text{DFS1}(G)$

$\text{INIT}(G)$

$S = \emptyset$

$S$  contiene i vertici ordinati, e' uno stack

FOR EACH  $v \in V$  DO

IF  $\text{COLOR}(v) = b$

$S = \text{DFSVISIT1}(G, v, S)$

RETURN  $S$

## ● DFSVISIT<sub>1</sub>(G, v, S)

COLOR(v) = g

FOR EACH  $u \in \text{Adj}_G(v)$

IF COLOR(u) = b

S = DFSVISIT<sub>1</sub>(G, u, S)

COLOR(v) = m

S = PUSH(S, v)

RETURN S

Ogni volta che viene ammesso un vertice viene inserito nello stack, alla fine avremo l'ultimo vertice concluso come primo dello stack

## ● DFS2(G, S)

G è il grafo trasposto

INIT(G)

CFC =  $\emptyset$

Array delle componenti

C = 0

Contatore per le componenti

FOR EACH  $v \in S$

dall'ultimo inserito fino al primo

IF COLOR(v) = b

DFSVISIT<sub>2</sub>(G, v, CFC, c)

c = c + 1

RETURN CFC

## ● DFSVISIT2( $G, v, CFC, c$ )

COLOR( $v$ ) =  $g$

CFC( $v$ ) =  $C$

FOR EACH  $u \in \text{Adj}(v)$

IF COLOR( $u$ ) =  $b$

CFC = DFSVISIT2( $G, u$ )

COLOR( $v$ ) =  $m$

RETURN CFC

## ● SCC( $G$ )

$S = \text{DFS1}(G)$

$G^T = \text{GRAFOTRASPOSTO}(G)$

CFC = DFS2( $G^T, S$ )

RETURN CFC

E' l'algoritmo generale

crea l'ordine con DFS1

crea il trasposto

segue DFS2 su  $G^T$

CFC e' l'array che salva le  
componenti di ogni elemento

## ● GRAFOTRASPOSTO (G)

$$E^T = \emptyset$$

FOR EACH  $v \in V$

FOR EACH  $u \in \text{Adj}(v)$

$$E^T = E^T \cup (u, v)$$

RETURN  $(V, E^T)$

Il tempo di esecuzione è  $\Theta(|V| + |E|)$  perché:

● GRAFOTRASPOSTO impiega  $|E|$  ad eseguire il doppio for  
(implementando l'assegnazione come inserimento in testa) +  $|V|$   
per l'assegnamento  $V^T = V$ . Costo  $\Theta(|V| + |E|)$

● DFS<sub>1</sub> e DFS<sub>2</sub> impiegano  $\Theta(|V| + |E|)$

OSS