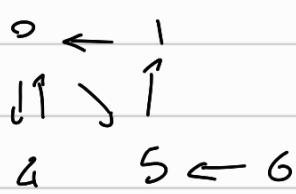


Componenti fortemente connesse / 28-03



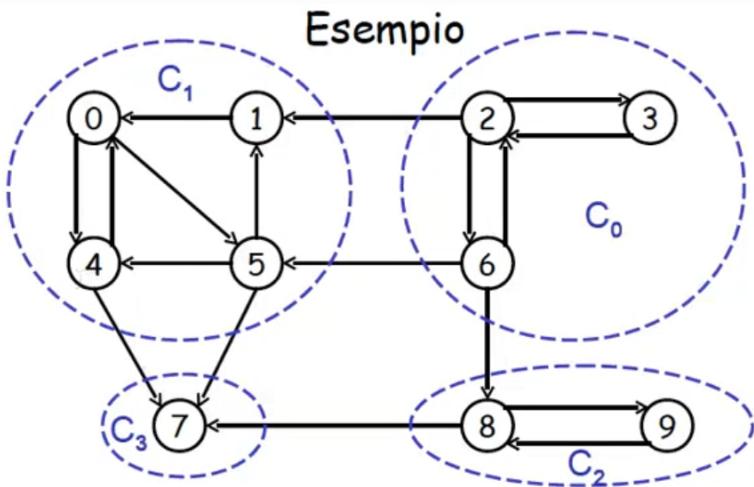
I nodi 0 e 1 si dicono **mutuamente raggiungibili**, perché da 0 posso raggiungere 1 ($0 \xrightarrow{*} 1$) e da $1 \xrightarrow{*} 0$ (dunque $0 \xleftrightarrow{*} 1$). Si può dire che 0 e 1 stanno sullo stesso ciclo. Da 6 posso raggiungere 0 , ma da 6 non posso raggiungere 0 .

Traffiamo grafici orientati.

Due nodi sono **fortemente connessi**: se sono mutuamente raggiungibili (es. 0 e 1)

È una relazione di equivalenza, \downarrow
riflessiva, simmetrica e transitiva
CFC = classi di equivalenza \uparrow
mutuamente

Le componenti fortemente connesse sono sottografi massimali
(non posso aggiungere altri nodi, non sarebbero fortemente connesse):
i cui nodi sono fortemente connessi. Inoltre non devono essere pochi.



C_0, C_1, C_2, C_3
sono CFC del
grafo

Se il grafo è aclico le CFC sono i singoli nodi

Tale relazione è esplicitata con \longleftrightarrow^* (relazione di equivalenza)

Si può costruire un grafo i cui nodi sono le cfc, esso è detto:
grafo quoziente:



gli archi sono presenti se lo dei nodi che collegano il nodo di una cfc a un altro di un'altra cfc

Sicuramente il grafo quoziente è aciclico

(Se avessi ciclo allora avrei un'unica cfc che vecchierebbe (e fortemente connesse). Se avessi un G aciclico, esso è già grafo quoziente

Quindi ha un ordine-topologico perché ha un grafo aciclico nel grafo quoziente: $c_0 c_1 c_2 c_3$, $c_0 c_2 c_1 c_3$

L'output ottenuto da un algoritmo è:

$\{2, 3, 6\}$, $\{0, 1, 4, 5\}$, $\{8, 9\}$, $\{7\}$ ovvero le cfc
in ordine topologico

L'algoritmo è una generalizzazione di quello topologico

Una cfc è una sorgente o pozzo solo nel grafo quoziente

Il tempo di fine visita end (C) di una cfc C è il max dei tempi delle visite dei suoi nodi

Nota: l'algoritmo non calcola il grafo quoziente

Dato $C \in C'$ c.d.c di G , se \exists un arco fra nodi di C e C' allora
 $\text{end}(C') < \text{end}(C)$

Provare di correttificare:

- Il primo nodo visitato è $u \in C$, con $C \subset C'$ connesso.

Allora tutti i nodi di $C \subset C'$ sono visitati prima di terminare la visita
di u : $\text{end}(C') < \text{end}(u) = \text{end}(C)$

- Il primo nodo visitato è in C' ; \exists cammino $C' \rightarrow C$ quindi
necessariamente tutti i nodi di C' sono visitati prima di quelli di C
(seguendo da $C \rightarrow C'$), quindi $\text{end}(C') < \text{start}(C)$ quindi $\text{end}(C') < \text{end}(u)$

T₁

In una visita in profondità di un grafo orientato, il modo avente il massimo
 $\text{end}(u) \in \text{c.d.c sorgente}$

Si dimostra seguendo che per assurdo se C non fosse sorgente avrei un
arco da $C' \rightarrow C$ ma allora $\text{end}(C) < \text{end}(C')$ e quindi $\text{end}(u)$
non sarebbe il max come supposto nell'ipotesi.

T₂

Inoltre se faccio DFS di un nodo v e.g. C posso visitare tutti e soli i nodi di C
(non ho archi uscenti, quindi un limite a: nodi di C)

O

Una c.d.c sorgente in G è c.d.c pozzo in G^T dove il **grafo trasposto**
inverte l'orientamento degli archi di G .

T₁ ci permette di trovare le c.d.c sorgenti, ovvero un ord. top

T₂ ci permette di visitare tutti i nodi di una c.d.c pozzo (che lo contiene)

O ci permette di vedere i sorgenti come pozzi (collega T₁ → T₂)

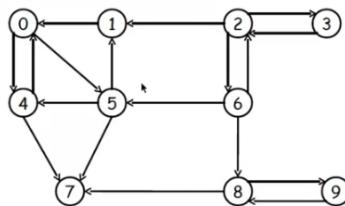
Note: un modo con $\text{end}[u]$ minimo (pozzo) non puo' appartenere
a una c.d.c sorgente (nell'es $0 \rightarrow 5 \rightarrow 1$, 1 ha end min, ma è C , e C , non è sorgente)

L'algoritmo è una DFS conservando i nodi in una sequenza S .

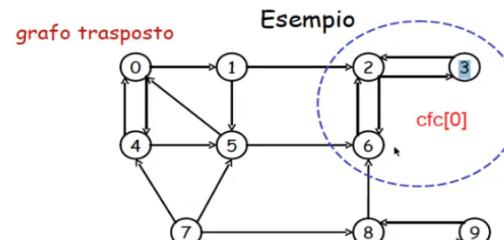
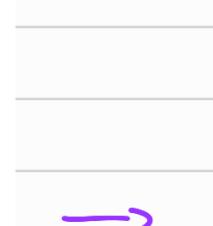
Estraggo da S l'ultimo nodo u che per T_1 è in una cfc sorgente.

Per trovare tutti i nodi di tale cfc, considero per 0 il G^T e quindi le visite dei nodi raggiungibili da u nel G^T .

Non è necessario modificare il grafo.

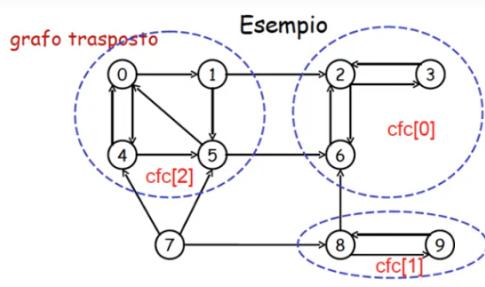


- visita in profondità a partire dal nodo **0**: la sequenza delle chiamate ricorsive e dei rispettivi ritorni può essere
 $(0 \ (5 \ (1) \ (7) \ (4))) \ (2 \ (3) \ (6 \ (8 \ (9))))$
- la corrispondente sequenza **S** dei nodi in ordine crescente di fine visita è:
1 7 4 5 0 3 9 8 6 2



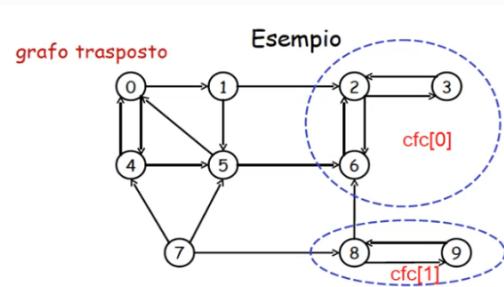
1 7 4 5 0 3 9 8 6 2

- prendo l'ultimo nodo della sequenza **S**, cioè **2**
- nel grafo trasposto visito tutti i nodi raggiungibili da esso, e li inserisco nel primo cfc: **cfc[0] = {2, 3, 6}**



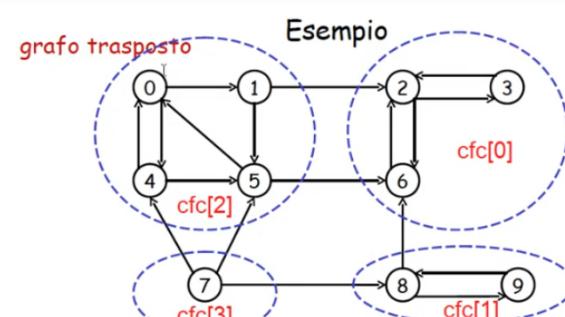
1 7 4 5 0 3 9 8 6 2

- percorrendo la sequenza **S** all'indietro, prendo il successivo nodo non visitato, cioè **0**
- nel grafo trasposto visito tutti i nodi raggiungibili da esso, e li inserisco nel terzo cfc: **cfc[2] = {0, 1, 5, 4}**



1 7 4 5 0 3 9 8 6 2

- percorrendo la sequenza **S** all'indietro, prendo il successivo nodo non visitato, cioè **8**
- nel grafo trasposto visito tutti i nodi raggiungibili da esso, e li inserisco nel secondo cfc: **cfc[1] = {8, 9}**



1 7 4 5 0 3 9 8 6 2

- percorrendo la sequenza **S** all'indietro, prendo il successivo nodo non visitato, cioè **7**
- nel grafo trasposto visito tutti i nodi raggiungibili da esso, e li inserisco nel **cfc[3]**: **cfc[3] = {7}**



Algoritmo

SCG (G)

$O(n+m)$

DFS (G , ord) // aggiunge i nodi visitati a Ord in ordine di fine visita

// non serve colorare i tempi di fine visita

G^T = grafo trasposto di G $O(n+m)$

Ord^{\leftrightarrow} = seq. vuota // ordine topologico delle CFC, risultato finale, \neq Ord

while (Ord non vuoto)

u = ultimo nodo non visitato in Ord (stà in CFC sorgente) $\xrightarrow{\text{per } T_i}$

C = insieme di nodi visitati (nuova CFC)

DFS (G^T , u , C) // aggiunge nodi visitati in C

$Ord^{\leftrightarrow} \leftarrow ord(C)$ // aggiunge in fondo a

return Ord^{\leftrightarrow}

altra visita completa

$O(n+m)$

Complessità:

\rightarrow tet

$O(n+m)$

Se il grafo di potenza è ciclico, l'algoritmo si riduce al calcolo dell'ordine topologico con la restituzione dei nodi in tale ordine
(le CFC sono i singoli nodi stessi)