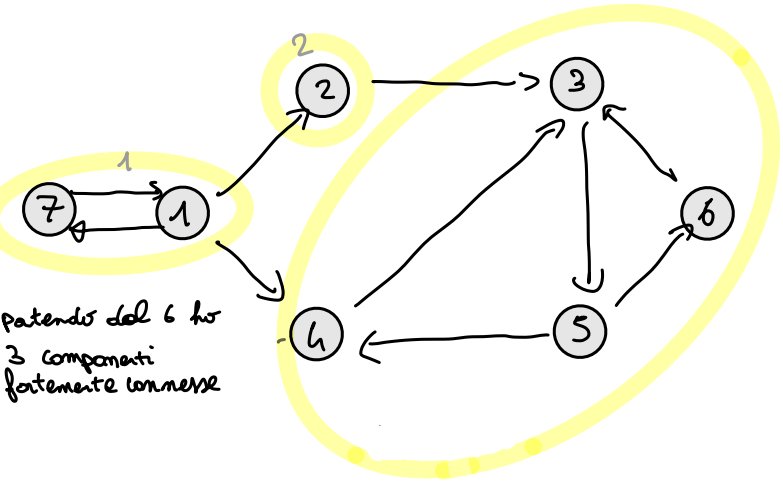
COMPONENTI FORTEMENTE CONNESSE

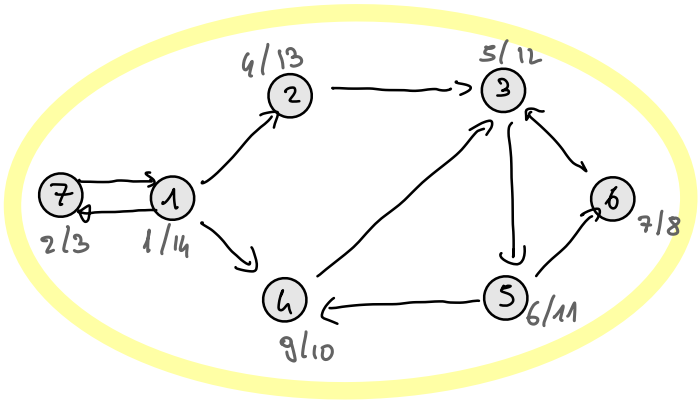


partendo dal 6 ho 3 componenti fortemente connesse

molti algoritmi su grafi devono lavorare su tutto il grafo anche se composti da componenti fortemente connesse

- cercano le componenti fortemente connesse
- eseguono l'algoritmo su ogni componente indipendentemente
- combinano i risultati

FORTEMENTE CONNESSO: Considerando 2 nodi A e B c'è un cammino che collega A e B e un cammino che collega B con A.

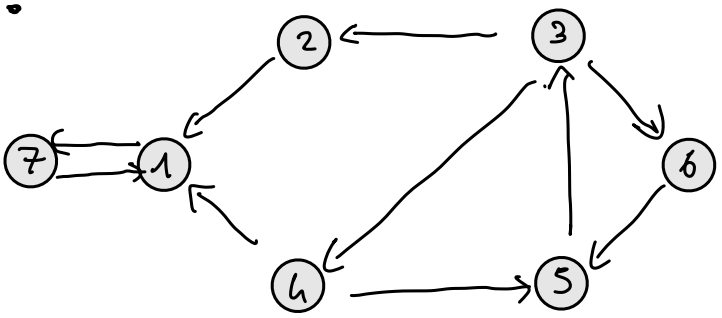


partendo invece dal nodo 1 ho un'unica componente connessa

PROBLEMA: occorre trovare un modo per capire qual è il nodo esatto per trovare componenti fortemente connesse

TRASPOSTO: stesso grafo ma con le direzioni delle frecce invertite

Dopo aver fatto una DFS e averle ordinati in modo inverso in base al post si calcola il TRASPOSTO

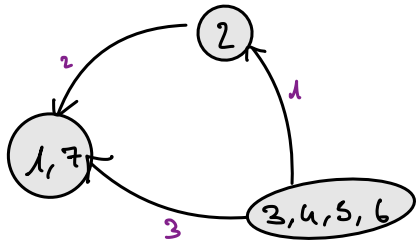


Perché funziona questo algoritmo?

PROPRIETÀ: il grafo delle C.F.C del TRASPOSTO e NON sono uguali. Cambia solo la direzione delle frecce

- Si parte dal grafo delle componenti connesse del trasposto

GRAFO C.F.C TRASPOSTO



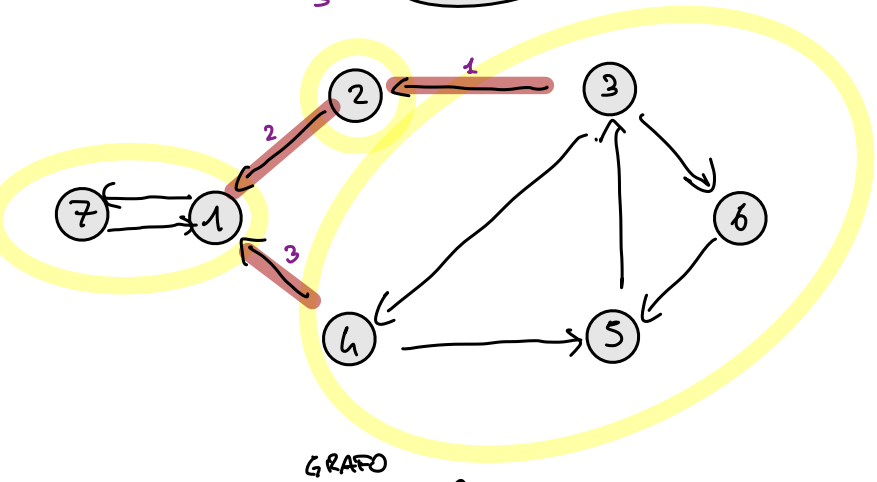
Considero gli archi che, nel trasposto, connettono le componenti fortemente connesse

⇓

NON posso avere un ciclo perché altrimenti le componenti fortemente connesse si unirebbero in una sola

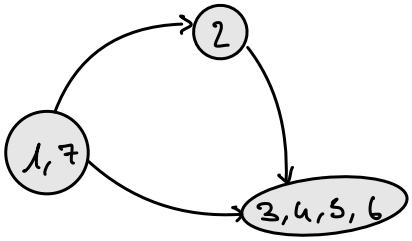
⇓

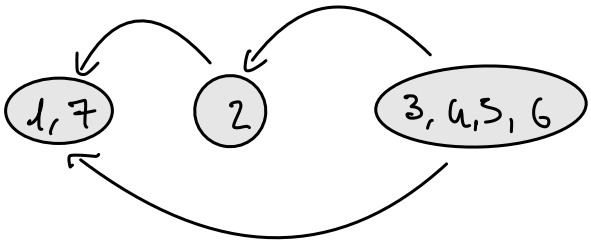
si tratta di un DAG.



GRAFO DEL TRASPOSTO

GRAFO C.F.C





Facendo una DFS sul Trasposto ottengo le componenti fortemente connesse.

Dopo aver linearizzato quello trasposto si può vedere che

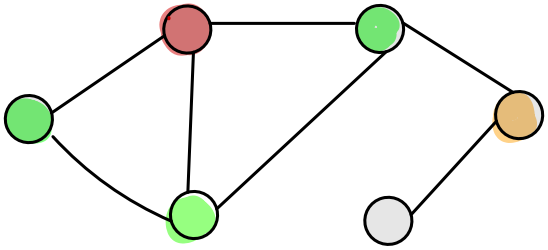
1) Si parte dal nodo con post più alto (1) e, dato che si tratta di una componente fortemente connessa visito il 7. Mi accorgo che la C.F.C. è stata visitata tutta
2) Passo al secondo nodo con la post più alta (2). Visito la componente fortemente connessa.
3) Passo all'ultimo nodo con la post più alta e visito la totalità della C.F.C.

⇓

La cosa funziona perché abbiamo un DAG.

BFS sui GRAFI

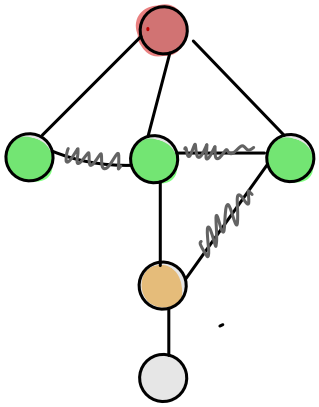
- Partendo da un nodo si visitano i vicini poi i vicini dei vicini e così via.
- Si parte da un nodo sorgente



vicini della sorgente

vicini dei vicini della sorgente

vicini dei vicini dei vicini della sorgente



ALBERO DI COPERTURA

La BFS, a differenza della DFS, non viene fatta ripetine; si parte solo da una sorgente

- GRAFO INDIRETTO: ottengo tutti i nodi di una componente fortemente connessa
- GRAFO DIRETTO: se non fortemente connesso allora alcuni nodi potrebbero rimanere fuori

NON si utilizza la RICORSIONE ma una serie di iterazioni e una CODA.

```
BFS(G,s):
    for all v∈V
        visited[v]:=0
O(1) { visited[s]:=1
       Q:=new_queue()
       enqueue(Q,s)
O(m) { while NOT is_empty(Q) do
           u:=dequeue(Q)
           for all (u,v)∈E do
               if visited[v]=0
                   then
                       enqueue(Q,v)
                       visited[v]:=1
```

// esploro i vicini di u per metterli in fondo alla coda. Controllo quindi gli archi incidenti in u.

1 volta per ogni nodo

- dist[u] memorizza la distanza da s a u
- Distanza tra u e v: numero di archi nel cammino più breve in G tra u e v
- prev[u] memorizza il padre di u nell'albero di copertura della visita BFS con sorgente s.

Ogni nodo viene inserito e estratto O(|V|) (O(n))

Ogni arco viene considerato
- 1 volta se il grafo è diretto per O(1) op
- 2 volte se il grafo è indiretto

$\Rightarrow O(|E|)$

$$\Downarrow$$

$$O(|V|+|E|)$$

Costa O(n) perché sono il numero di nodi che vado a inserire

Si può modificare per calcolare la distanza tra un nodo e la sorgente ⇒ numero di archi più breve tra u e v (cammino più breve) allo stesso costo