Titolo nota 26/02/2016

BFS-esplora(s)

```
Q = nuova Coda()
D = nuovoDizionario()
Enqueue(Q, s)
Insert(D, s)
while (Q ≠ ∅) {
    u = Dequeue(Q)
    * esamina u;
    ∀ vertice v ∈ Adj(u) {
        if( Search(D, v) == NIL) {
            Enqueue(Q, v)
            Insert(D, v);
        }
    }
}
```

D = AVL

$T(n, m) = O((n+m)\log n)$

n = # vertici raggiungibili dalla sorgente s

m = # archi esaminati durante la visita

$\Theta(n)$ I/O sulla coda Q $\rightarrow \Theta(1)$ tempo

$\Theta(n)$ Insert nel dizionario D

$\Theta(m)$ ricerche in D

$|D| = n$

D = tabelle hash

$\Rightarrow T(n, m) = \Theta(n+m)$ MEDIO

$T_{WORST}(n, m) = O((n+m) \cdot n)$

Titolo nota 26/02/2016

DFS(G)

```
for ogni v ∈ V
    v.color = B
    v.π = NIL
time = 0
for ogni u ∈
    if (u.color == B) DFS-visit(G, u)
```

$\Theta(\sum_{u \in V} |Adj[u]|)$

$\Theta(|E|)$

$T(|V|, |E|) = \Theta(|V| + |E|)$

ottimo

DFS-visit si esegue una ed una sola volta su ogni vertice del grafo ($B \to G$)

DFS-visit(G, u)

```
time++
u.d = time
u.color = G
for ogni v ∈ Adj[u] {
    // ispeziona arco (u, v)
    if (v.color == B) {
        v.π = u
        print "l'arco (u,v) è dell'albero"
        DFS-visit(G, v)
    }
    else if (v.color == G) print "(u,v) è all'indietro";
}
u.color = N
time++;
u.f = time
```

Titolo nota 26/02/2016

$G = (V, E)$

PROPRIETÀ

$G_\pi = (V, E_\pi)$

$E_\pi = \{ (v.\pi, v) \mid v \in V, v.\pi \neq NIL \}$

è una FORESTA (DF)

v è un discendente di u
(v è sottoalbero di radice u nella foresta DF) $\Leftrightarrow$ v è stato scoperto quando u era GRIGIO
(visita di $Adj(u)$ ancora in corso)

Titolo nota 26/02/2016

CLASSIFICAZIONE degli ARCHI (Grafi ORIENTATI)

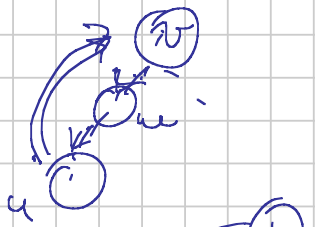
$G = (V, E)$

$\forall (u, v) \in E$,

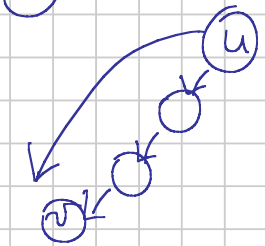
(u, v) può essere catalogato come:

1. **arco dell'albero**
   nel corso della visita, quando si ispeziona l'arco (u, v), $v \in B$.

2. **arco all'indietro**
   v è GRIGIO quando l'arco viene ispezionato
   $\Rightarrow$ v è un antenato di u in un albero DF
   u è discendente di v



3. **archi in avanti**
   v è un discendente di u in un albero DF
   v è NERO, e u è stato scoperto prima di v
   $(u.d < v.d)$



Titolo nota 26/02/2016

(4) arco trasversale

u e v non sono l'uno contenuto dell'altro.

$v.color = N$, $u.d > v.d$

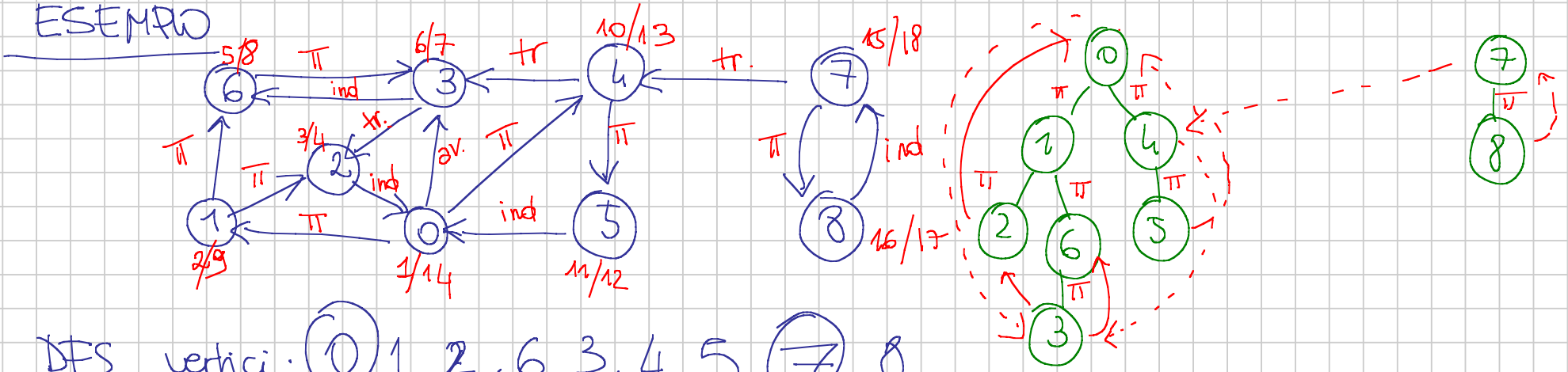
(u scoperto dopo v)

Titolo nota 26/02/2016

```
DFS-visit(G,u)
time++
u.d = time
u.color = G
for ogni v ∈ Adj[u] {
    // ispeziona arco (u,v)
    if (v.color == B) {
        v.π = u
        print "l'arco (u,v) è dell'albero"
        DFS-visit(G,v)
    }
    else if (v.color == G) {
        print "l'arco (u,v) è all'indietro"
    }
    else if (v.color == N) {
        if (u.d < v.d)
            print "(u,v) è un arco in avanti";
        else // u.d > v.d
            print "(u,v) è un arco trasversale"
    }
u.color = N
time++;
u.f = time;
```

Titolo nota 26/02/2016

ESEMPIO



DFS vertici: 0, 1, 2, 6, 3, 4, 5, 7, 8

archi: (0,1), (1,2), (2,0), (1,6), (6,3), (3,2), (3,6), (0,3),
(0,4), (4,3), (4,5), (5,0), (7,4), (7,8), (8,7)

π π indietro π π tr. ind av.

π tr. π ind tr. π ind

Titolo nota 26/02/2016

TEOREMA

In una DFS di un grafo non orientato G, gli archi di G possono essere solo archi dell'albero o archi all'indietro.

DIM.

$(u,v) \in E$
$u.d < v.d$

u è GRIGIO quando v viene scoperto $(B \to G)$
u è ancora GRIGIO quando v diventa NERO

CASO 1 l'arco (u,v) viene esplorato la prima volta da u verso v
$\Rightarrow v \in B \Rightarrow$ l'arco (u,v) è un arco dell'albero.

CASO 2 l'arco (u,v) viene esplorato la prima volta da v verso u.
~~$\Rightarrow$~~ ma $u \in G \Rightarrow (u,v)$ è un arco all'indietro.

□

Titolo nota 26/02/2016

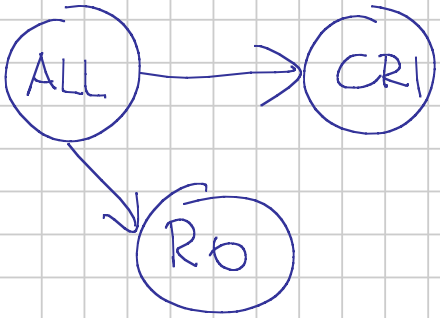
ORDINAMENTO TOPOLOGICA di un GRAFO ORIENTATO ACICLICO

DAG

(grafi diretti e aciclici)

trovare un ordinamento dei vertici t.c.

$\forall (u, v) \in E$, u preceda v nell'ordinamento



→ si fa usando la DFS

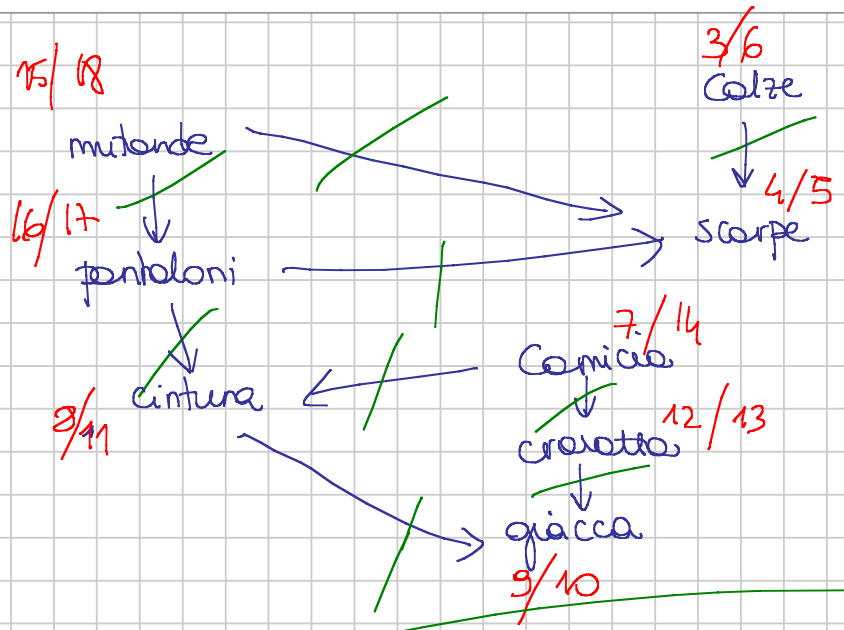
Titolo nota 26/02/2016

TOPOLOGICAL-SORT (G) // G è un DAG

- DFS(G) calcolando i tempi di fine visita di ogni $v \in V$
- quando un vertice diventa Nero, si inserisce in testa a una lista concatenata
- ritorna lista con ordine dei vertici

$$T(|V|, |E|) = \Theta(|V| + |E|)$$

Titolo nota 26/02/2016



mutande, pantaloni, camicia, cravatta, cintura, giacca, calze, scarpe, orologio

Titolo nota 26/02/2016

LEMMA: Un grafo è aciclico $\Leftrightarrow$ non ci sono archi all'indietro.

TEOREMA: Topological_Sort(G) produce un ordinamento topologico di un DAG G.

Dimostrazione

OT: $\forall (u,v) \in E$, u deve precedere v nell'OT.

$\Rightarrow$ u deve essere inserito in lista DOPO v (inserimenti in testa)

$\Rightarrow$ deve valere $v.f < u.f$

(v va in lista prima di u)

$\Rightarrow$ dobbiamo dimostrare che

$\forall (u,v) \in E$, $v.f < u.f$

Titolo nota 26/02/2016

Sia $(u,v) \in E$.

quando si ispeziona l'arco (u,v) v NON può essere GRIGIO
(G è un grafo aciclico, non può contenere archi all'indietro).

1° caso — v è BIANCO

u è in G, scopro v, chiamo DFS-Visit(G, v)
$\Rightarrow$ v diventa NERO prima di u
$\Rightarrow$ $v.f < u.f$ ✓
(e v segue u nell'OT)

2° caso — v è NERO

la visita di v è già terminata, mentre quella di u è ancora in corso $\Rightarrow$ $v.f < u.f$ ✓

Titolo nota 26/02/2016

ESERCIZIO

Progettare un algoritmo efficiente per verificare se G e $\bar{G}$ un grafo CONNESSO.

CONNESSO(G)

- scegli una sorgente $s \in V$
- BFS(G, s)
- for all $v \in V$ {
  if(v.colore == B) return FALSE; // G non è connesso
  }
- return TRUE // G è connesso