

18/07/14: dato  $G$  vedi se esiste un vertice  $a, b, e$

che ha almeno una delle seguenti proprietà:

i)  $a$  non raggiungibile; ii)  $b$  non raggiungibile; iii)  $e$  non raggiungibile

Alg ( $G$ )

DFS-VISIT ( $G, a, e_1, e_2, e_3$ )

for each  $a, b, e \in V$  do

  | DFS-VISIT ( $G, a, e_1$ )

  | DFJ-VISIT ( $G, b, e_2$ )

  | DFJ-VISIT ( $G, e, e_3$ )

  | if  $e_1(s) = B$  or  $e_2(e) = B$  or  $e_3(a) = B$  then

  |   return true

return false

23/02/18: Dat.  $G = (V, E)$ ,  $A, B \subseteq V$  &  $k \in \mathbb{N}$

Si vogliamo che ogni percorso da parte di un vertice in  $A$  e raggiunge un vertice in  $B$  ha lunghezza  $\geq k$

$\text{DM}_k(G)$

for each  $v \in V$  do

color(v) =  $\infty$

pred(v) = null

d(v) = 0

l'idea è quella di

definire false quantità

$\rightarrow$   $V : a \sim b \vee c$

la sua lunghezza è  $\leq k$

mentre si va avanti

alla BFS

BFS( $G, s, Q$ )

$\text{DM}_k(G)$

$Q = \{s\}$

color(s) = 0

while  $Q \neq \emptyset$  do

$n = \text{Degree}(Q)$

for each  $u \in \text{Adj}[n]$  do

if color(u) =  $\infty$  then

enqueue(Q, u)

color(u) = 0

pred(u) =  $n$

$d(u) = d(n) + 1$

color(n) =  $N$

Alg ( G, A, B, k )

for each  $a \in A$  do

if  $\text{color}(a) = B$  then  
    BFS (G, a)

for each  $b \in B$  do

if  $\text{color}(b) = N$  then  
    if  $d(b) < k$  then  
        return false

return true

( une jonsse qui l'eeone . . . )

28/06/18: Dato  $G = (V, E)$  e  $A \subseteq V$  si deve su algoritmo  
che dato  $G, A$  e' in rete e n' dellez l'insieme  $Z \subseteq V$  t.c.  
 $\forall z \in V \quad z \in Z \Leftrightarrow$  esiste un percorso da  $z$  a  $v$  che passa  
per qualche  $a \in A$ . Quer se  $\exists a \in A : z \text{ non a n'}$ .

Algo ( $G, A, v$ )

$$Z = \emptyset$$

Init ( $G, e_1, e_2$ ) // tutto bianco

BFS ( $G^T, v, e_1$ )

for each  $a \in A$  do

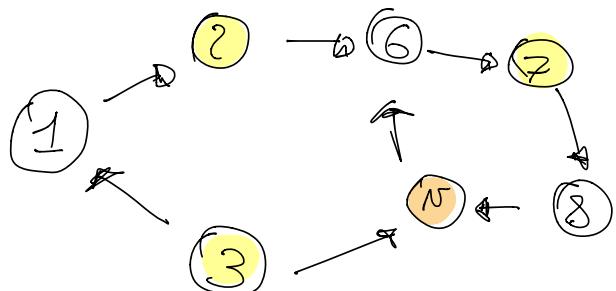
```

  if  $e_1[a] = N$  then
    DFS-Visit( $G^T, a, e_2$ )
  
```

for each  $N \in V$  do

```

  if  $e_2[N] = N$  then
     $Z = Z \cup \{N\}$ 
  
```



$$A = \{2, 5, 7\}$$

25/01/28: Dado o grafo G e os vértices S, V, U, reflexe  
se busca os vértices de fonte de S, pôrém perdeu o vértice V  
fomos para U

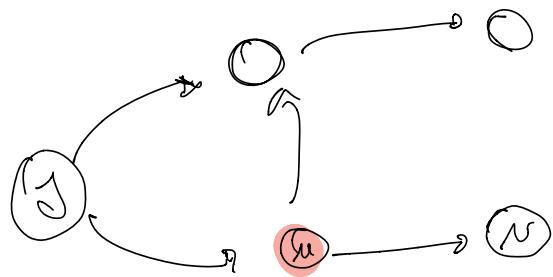
Algo (G, S, U, V)

Amostra (G) // todos os vértices

colorir [U] = N

BFS (G, S)

return colorir [V] = N



26/10/2018: Dati  $G$ ,  $A, B \subseteq V$  cerca  $Z \subseteq V$  t.c.  $A \cap Z = \emptyset$

$Z \in \mathcal{Z} \Leftrightarrow$  esistono due percorri da  $Z$  che l'uno passa  
a qualche vertice  $a \in A$  senza passare per  $b \in B$ ; l'altro  
passa a qualche vertice di  $B$  e mai passa per quelli di  $A$

Alg (  $G$ ,  $A, B$  )

$Z = \emptyset$

$\text{DFS}(G, B, e_1)$  // tutto binario tranne  $b \in B$  di meno  
for each  $a \in A$  do

if  $e_1(a) = B$  then  
   $\text{DFS\_right}(G^T, a, e_1)$

$\text{DFS}(G, A, e_2)$  // tutto binario tranne  $a \in A$  meno

for each  $b \in B$  do

if  $e_2(b) = B$  then  
   $\text{DFS\_left}(G^T, b, e_2)$

for each  $z \in V$  do

if  $e_1(z) = N$  and  $e_2(z) = N$  then  
   $Z = Z \cup \{z\}$

ritorna  $Z$

03/07/19: Dato  $G$ , vedere se è possibile partizione in due  
due parti vette in due sottoset V<sub>1</sub>, V<sub>2</sub>  $\subseteq V$   $\forall c$

i)  $V_1 \cap V_2 = \emptyset$

ii)  $V = V_1 \cup V_2$

iii)  $u \in V_1 \Leftrightarrow \forall (u, v) \in E \text{ tal } v \in V_2$

Algo (a)

V<sub>1</sub> = Colorazione legale ( $v$ )  $\parallel \forall x \in V$

V<sub>2</sub> = V - V<sub>1</sub>

for each  $v \in V$  do  $\parallel$  colorare uguali per distinzione

if  $v \in V_1$  then

    color(v) = GIALLO

else

    color(v) = VERDE

for each  $u \in V_1$  do

    if  $v \in \text{Adj}(u)$  and color(v) ≠ VERDE

        V<sub>1</sub> = V<sub>1</sub> ∪ {u}

        V<sub>2</sub> = V<sub>2</sub> ∪ {v}

Colorazione legale ( $V$ )

return V12

12/02/19: Dado  $G$  e  $k \in \mathbb{N}$ , se  $V$ ,  $A \subseteq V$  verifica se  
ogni percorso che parte da  $s$  e raggiunge qualche vertice di  $A$   
ha lunghezza  $\leq k$

Algo( $G, k, s, A$ )

Init( $G$ )

BFS( $s$ )

for each  $a \in A$  do

| if  $\text{dist}(a) \leq k$  then  
| | return false

return true

04/09/19: Dado  $G$  e  $A \subseteq V$  encontra o vértice  $v \in A$  tal que  $\deg(v)$  é o menor para  $v \in A$

Algo ( $G, A, u, v$ )

$$z = \emptyset$$

Init ( $G, e_1, e_2$ )

DFS-Visit ( $G, u, e_1$ )

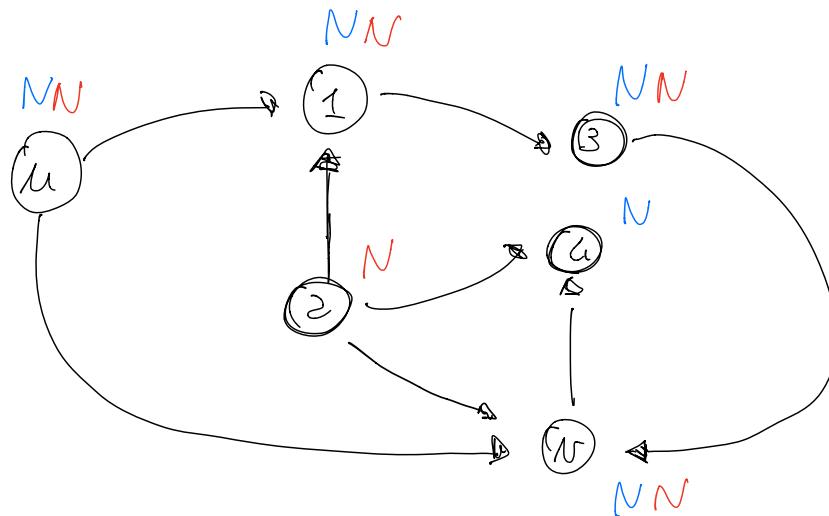
if  $e_1[v] = N$  then

DFS-Visit ( $G^T, v, e_2$ )

for each  $z \in A$  do

if  $e_1[z] = B$  or  $e_2[z] = B$  then  
 $z = z \cup \{z\}$

return  $z$



29/03/19A: Dado  $G$  e  $A \subseteq V$  e que retíeli  $\mu, \nu$  sevem  $\mu$  algoritmo de execução  $A' \subseteq A$  s.t.  $\forall a' \in A'$   
 $a \in A' \Leftrightarrow$  existe  $m$  pares de ele fonte de  $\mu$ , omive a  $\nu$   
 Passando por  $a$ , logo  $\mu \rightsquigarrow a \rightsquigarrow \nu$

Algo ( $G, A, \mu, \nu$ )

$$A' = \emptyset$$

Init ( $G, c_1, c_2$ )

DFS-Vis( $G, \mu, c_1$ )

if  $c_1[\nu] = N$  then ||  $\mu$  rejege  $\nu$

    DFS-Vis( $G^T, \nu, c_2$ )

    for each  $a \in A$  do

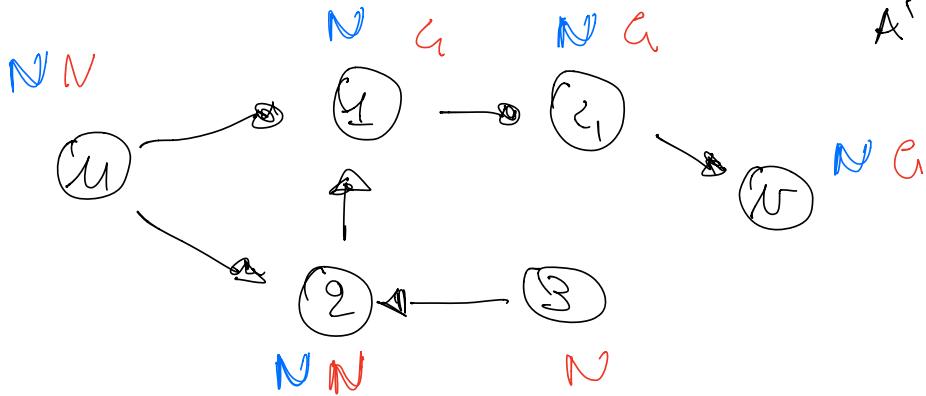
        if  $c_1[a] = N$  and  $c_2[a] = N$  then

$$A' = A' \cup \{a\}$$

Retir  $A'$

$$A = \{1, 2, 3, 4\}$$

$$A' = \{1, 2, 4\}$$



29/03/19 B : Dado  $G$  e  $B \subseteq V$  e ovo retiroi  $\mu, \nu$  e  
 $\mu$  inteo  $k$ , sevee u agommo de determini C'usione  
 $B' \subseteq B$   $\forall b \in B$

$b \in B' \Leftrightarrow b$  e' negjigible olo  $\nu$  sero pomone per  $\mu$  e  
 $b$  negjige  $\mu$  ee u puceto lungo  $< k$

Alg ( $G, B, \mu, \nu, k$ )

$$B' = \emptyset$$

Dinit ( $G, e_1, e_2$ )

$$e_1[\mu] = N$$

DFS-Visiq ( $G, e_1, N$ )

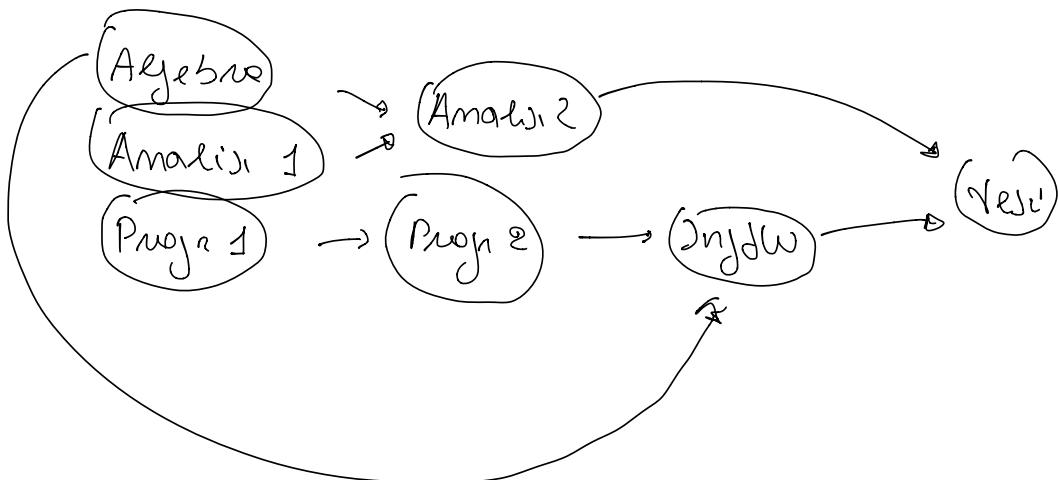
BFS ( $G^T, e_2, \mu$ )

for each  $b \in B$  olo

[ if  $e_1[b] = N$  and  $e_2[b] = N$  and  $d[b] < k$  then  
 [      $B' = B' \cup \{b\}$  ] ]

return  $B'$

13/06/19: Si consideri un grafo  $G$  i cui nodi rappresentano  
dei corsi e gli ordini di riuscita dei prospettivi esistente per  
i relativi esami. Scrivere un algoritmo che, dato  $G$ ,  
calcola il numero minimo di generazioni necessarie per  
completare tutti i corsi, se ogni corso è selezionato



Algo ( $G$ )

Init ( $G$ ) // Inizializza tutti bianchi

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

for each element  $\in S$  do

    Counter = Counter + 1

return Counter

DFS ( $G$ ) // ord-topologico

$S = \emptyset$

for each  $v \in V$  do

    if colore( $v$ ) =  $\beta$  the

$S = \text{DFS-Visita}(G, v, S)$

return  $S$

DFS-VISIT( $G, N, S$ )

color( $v$ ) =  $G$

for each  $u \in \text{Adj}[N]$  do

    if  $\text{color}(u) = B$  then

$S = \text{DFS-VISIT}(G, u, S)$

color( $v$ ) =  $N$

Push( $S, N$ )

Scenari: Dato un gioco  $G$ ,  $I \subseteq V$

$F_G := \{N \in V : N \text{ non ha mai uscito}\}$ , scrivere un algoritmo che verifica se  $\forall v \in I \exists u \in F_G : u$  è raggiungibile da  $v$

Algo ( $G, I, F_G$ )

Init ( $G$ )

returning false quando

for each  $u \in F_G$  do

usare un vertice di  $I$  che

| if colore( $u$ ) = B then

non è raggiungibile da nessun

| | DFS\_Visit( $G^T, u$ )

vertice di  $F_G$

for each  $N \in I$  do

| if colore( $N$ ) = B then

| | return false

return true

**Scenarii:** Dati  $G, s \in V, B, C \subseteq V$ , cercare un algoritmo

che calcola, in una lista  $L$ , i vertici  $N \in V \setminus C$

i)  $N \in B$  e raggiungibili da  $s$

ii) esiste un percorso da  $s$  a  $v$  che non passa per alcun vertice di  $C$

Algo ( $G, s, B, C$ )

$$L = \emptyset$$

Impr ( $G, C, e_1, e_2$ ) ||  $\forall v \in C \quad e_2(v) = N$

DFS-Visit ( $G^T, s, e_1$ )

for each  $b \in B$  do

    if  $e_1(b) = N$  then

        DFS-Visit ( $G, s, e_2$ )

        if  $e_2(s) = N$  then

            Impr ( $L, b$ )

Return  $L$

Impr ( $L, key$ )

if  $L = \text{null}$  then

$L = \text{AllocateElement}$

$L \rightarrow \text{key} = \text{key}$

$L \rightarrow \text{next} = \text{null}$

else

$L \rightarrow \text{next} = \text{AllocateElement}(\rightarrow \text{key} = \text{key})$

return  $L$

Lemma 7: DDo G. refines  $\pi$  if and only if vertex a, b, c

$\forall c \in \text{non-adjacent to } a \cup \text{non-adjacent to } b \cup \text{non-adjacent to } c$  we have  $a \sim c$

Alg. (a)

$\text{Dm.}(\pi, c_1, c_2, c_3)$

for each  $a, b, c \in V$  do

$\text{DFS-Vis.}(\pi, c_1, a, c_2)$

$\text{DFS-Vis.}(\pi, c_2, b, c_3)$

$\text{DFS-Vis.}(\pi, c_3, c, c_1)$

if  $c_1(S) \neq N$  or  $c_2(S) \neq N$  or  $c_3(S) \neq N$  then

return false

else true

**Scomponendo:** Dato  $G$  e  $X \subseteq V$  e due vertici  $u, v$ ,  
 deve un algoritmo che risponda se valgono le seguenti:  
 i)  $\exists$  un percorso da  $v$  a  $u$  passante per un vertice di  $X$   
 ii)  $\forall u \in X$   $\exists v \in V$  tale che  $v$  è vicino a  $u$

Alg( $G, X, u, v$ )

$\Delta_{M, Y}(G, e_1, e_2, X)$  // tutti binari tranne  $x \in X$

$\text{DFS-}V_{ij}(G, v, e_1)$

$\text{DFS-}V_{ij}(G, u, e_2)$

if  $e_1(u) = N$  or  $e_2(v) = N$  return false

else

return true

$\Delta_{M, Y}(G, e_1, e_2, X)$

for each  $v \in V$

$e_1(v) = e_2(v) = B$   
 if  $v \in X$  then  
 $e_1(v) = e_2(v) = N$

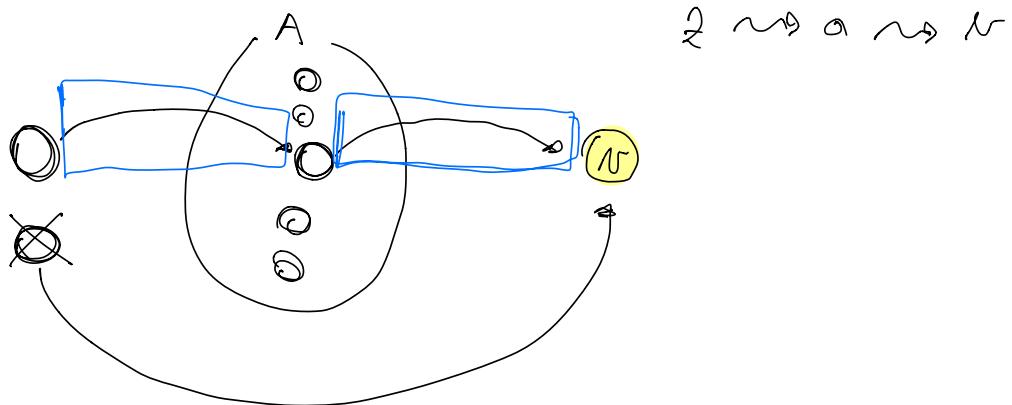
return false

esiste un percorso da  $v$  a  $u$  che non passa per  $X$   
 oppure

esiste un percorso da  $u$  a  $v$  che non passa per  $X$

1. Annalisa quale di  $X$  per la prima e seconda visita
2. Se esiste  $v \neq u$  o  $u \neq v$ , ret false

Desmonstro: Dado  $G \in A \subseteq Y$  e d  $v \in V(G)$ , se deve  
 provar que existem  $z \in V(G)$  tais que  $H = G \setminus z$



$\forall x \in A \exists y \in A$

1. Kefre se k moy
  2. Se y mo z, ay yi n ole' cipile e poice'  
Dee 'tewi'wir'e, u moy not.

1. Int ( a )

for each  $x \in Y$  do

If  $\text{color}(v) = B$  then

$\Delta FJ_1, V_{1,2}, \gamma \subset G_1, u_1$

for each year old

if  $\text{color}(y) = N$  then

11 21 ~~now~~ y

2.  $\text{Jmi}(G)$

$\text{DFS-visit}(G^T, z)$

for each  $y \in A$  do

if  $e_2(y) = N$  then

$y \rightsquigarrow z$

$\text{Ago}(G, A, z)$

$Z = \emptyset$

$\text{Jmi}(G, e_1, e_2)$  // tutto biomeo

$\text{DFS-visit}(G^T, z, e_1)$

for each  $y \in A$  do

    if  $e_1(y) = N$  then // solo sui  $y \in e_1$   $y \rightsquigarrow z$

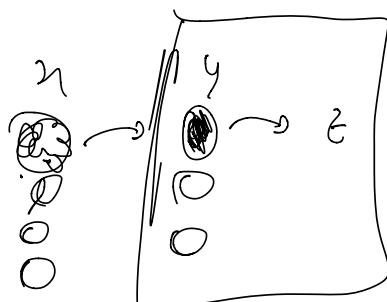
$\text{DFS-visit}(G^T, y, e_2)$

for each  $u \in V$  do

    if  $e_2(u) = N$  then

$Z = Z \cup \{u\}$

return  $Z$



Sequencia: Dado un grafo  $G$ , sección en algo el  
área de vértices que es cíclico

Ajgo ( $G$ )

ret = false

Dinit ( $G$ )

for each  $v \in V$  do

    if  $\text{color}(v) = B$  then

        ret = DFS-visit( $G, v$ )

        if ret = True

            return True

return ret

DFS-visit ( $G, v$ )

ret = false

color ( $v$ ) = G

for each  $u \in \text{Adj}(v)$  do

    if  $\text{color}(u) = B$  then

        DFS-visit ( $G, u$ )

    if  $\text{color}(u) = G$  then

        ret = True

color ( $v$ ) = N

return ret

Sequenza: Dato  $G \in A \Sigma V$ , serve un algoritmo che  
colora le varie  $z \subseteq A$  in  $\forall z \in A$

$z \in Z \Leftrightarrow$  ogni pezzo minore ha forme per  $z$

Alg ( $G, A$ )

$$Z = A$$

Dmit ( $G, A$ ) / tutto brano viene colorato  $A$

DFS-visit ( $G, u$ )

if colore( $v$ ) =  $N$  then

for each  $z \in Z$  do

if colore( $z$ ) =  $N$  then

$$Z = Z \cup \{z\}$$

return  $Z$

Regime al contrario:  $Z \leftarrow A$  e volgo jà el.

Per esistere un pezzo minore non deve formare per  $Z$

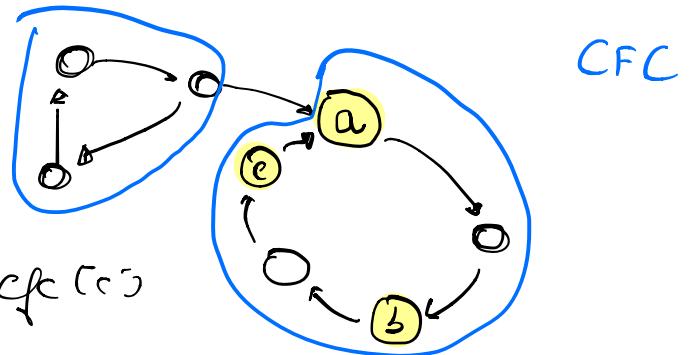
**Definizione:** Dato  $G$  e tre vertici  $a, b, c$  deve un algoritmo che verifica l'esistenza di un percorso infinito  $T$  che contiene  $a, b, c$ .

L'idea è quella di verificare se  $a, b, c$  sono tutti nello stesso CFC, anche sono mutuamente reg.

Algo( $G, a, b, c$ )

$$CFC = CFC(G)$$

$$\text{return } CFC(a) = CFC(b) = CFC(c)$$



$CFC(G)$

$$L = DFS_1(G)$$

$$\text{return } DFS_2(G, L)$$

$DFS_1(G)$

$In.V(G)$

for each  $v \in V$  do

| if  $color(v) = B$  then

$$| \quad L = DFS_1 - v, V \setminus \{v\}(G, v)$$

return  $L$

$DFS_2(G, L)$

for each  $v \in L$  do

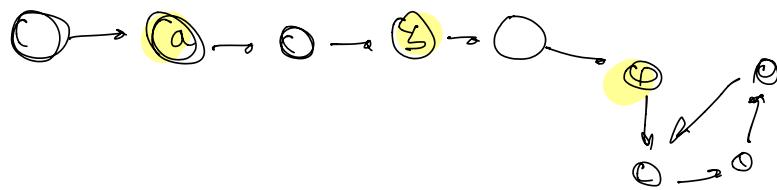
| if  $color(v) = B$  then

$$| \quad CFC = DFS_2 - v, V \setminus \{v\}(G, v)$$

new  $CFC$

Il may be another  
and some  $v$   
be EFC or be  
openable

28/06/12f: Dato un grafo e tre vertici  $a, b, c$  devono essere in  
 algoritmo che verifica se esiste un percorso infinito che  
 forma solo una volta prima per  $a$ , poi  $b$  e poi  $c$ ,  
 e così via; i tre vertici  $a, b, c$  restano inizialmente in  
 un solo ciclo e non vengono divisi in due  
 infiniti



- Verifica prima che tutto sia un buone.
- Se ok, verifica l'esistenza di un solo ciclo  
 dove tutti i adiacenti di  $c$

Algo ( $G, a, b, c$ )

$\text{DMIT}(G, e_1, e_2, e_3)$  // biondo

$\text{DFS\_VISIT}(G, a, e_1)$

$\text{DFJ\_VISIT}(G, b, e_2)$

if  $e_1(b) = N$  and  $e_2(c) = N$  then // anche  
 $\Rightarrow$  anche  
 return  $\text{DFS\_VISIT}(G, e, e, \text{rosso})$

Altre false

DFS-Visit-Von ( $G, v, e, \text{color}$ )

$\text{color}(v) = a$

for each  $u \in \text{Adj}[v]$  do

if  $\text{color}(u) = b$  then

$\text{ref} = \text{false}$

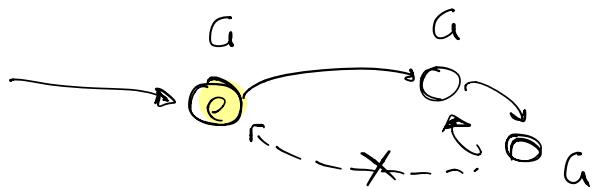
$\text{DFS-Visit-Von}(G, u, \text{color})$

if  $\text{color}(u) = a$  and  $u \neq e$  then

$\text{ref} = \text{true}$

$\text{color}(v) = N$

$\text{return ref}$



13/01/21 : Dato un grafo e un vertice  $s$  e deve risolvere  $B \subseteq V$   
 chever un algoritmo che dà i vertici  $G, s, B, C$  colle cui  
 siamo lista ( $v_i$  è un vertice  $v$  se)

i)  $v \in B$  e può raggiungere  $s$  tramite un percorso

ii) esiste un solo percorso da  $s$  a  $v$  che non passa per  
 alcun vertice di  $C$

Algo( $G, s, B, C$ )

$$L = \emptyset$$

IMPL( $G, e_1, e_2$ ) // tutti binari tranne  $e_2[v] = N \quad \forall v \in C$

DFS-Visit( $G^T, s, e_1$ )

DFS-Visit( $G, s, e_2$ )

for each  $v \in B$  do

if  $e_1[v] = N$  and  $e_2[v] = N$  then  
      $L = \text{Insert}(L, v)$

return  $L$

SMJN( $L, key$ )

if  $L = \text{null}$  then

$L = \text{AllocElement}()$

$L \rightarrow key = key$

$L \rightarrow next = \text{null}$

else

$L \rightarrow next = \text{AllocElement}() \rightarrow key = key$

return  $L$

22/02/21: Dato un grafo  $G$  e due vertici  $u, v$  e  
un insieme  $A \subseteq V$ , scrivere un algoritmo che dà  
in ingresso il grafo, i due vertici  $u, v$ , l'insieme  $A$   
e uscita  $Z \subseteq E$  tali che solo se

$z \in A$ , allora  $z \subseteq A$

22') Ogni percorso che porta da  $u$  a  $v$  non può passare  
per  $z$

Algo( $G, u, v, A$ )

$$Z = A$$

Δmit( $G$ )

DFS-visit( $G, u, \epsilon_1$ )

DFS-visit( $G^T, v, \epsilon_2$ )

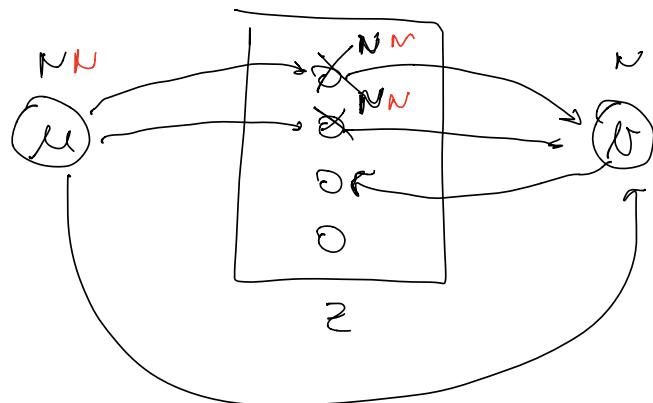
for each  $z \in Z$  do

```

  if  $\epsilon_1[z] = N$  and  $\epsilon_2[z] = N \quad \forall u$ 
     $Z = Z \setminus \{z\}$ 
  
```

return  $Z$

Rimuovo da  $Z$  gli el. di  $A$  tali che  
esiste un percorso da  $u$  a  $v$  che passa per  $z$



22/02/21: Dato un ABR  $T$ , se  $h \geq 0$ ,  $h_1 \leq h$   
 Albero ha algoritmo che cancella gli  $\tau$  tutti e due i modi sono  
 i) ho un elenco compreso in  $[h_1; h_2]$   
 ii) il settable in cui non c'è contiene un modo  
 che elenco compreso in  $[h_1; h_2]$  che sia a  
 profondità  $\geq h$

L'algoritmo deve restituire ad ogni suo predecessore  
 la radice dello max profondità dei nodi con elenco  
 in  $[h_1; h_2]$  nel settable che deve essere riportato.

Alg ( $\tau, h, h_1, h_2$ )

if  $\tau \neq \text{null}$  and  $h \geq 0$  and  $h_1 \leq h \leq h_2$  then

if  $\tau \rightarrow \kappa \in [h_1; h_2]$  then

PS = Alg02 ( $\tau \rightarrow \text{rx}, h_1, h_2, h, \tau$ )

PD = Alg02 ( $\tau \rightarrow \text{ax}, h_1, h_2, h, \tau$ )

if  $PS \geq h$  or  $PD \geq h$  then

$\tau = \text{cancella}(\tau)$

return  $\tau$

Alg02 ( $\tau$ ,  $k_1$ ,  $k_2$ ,  $h$ ,  $P$ )

if  $\tau \neq \text{null}$  then

if  $\tau \rightarrow h \in \tau k_1 ; h \in \tau$  then

$ps = \text{Alg02}(\tau \rightarrow sx, k_1, k_2, h, \tau)$

$pD = \text{Alg02}(\gamma, dx, k_1, k_2, h, \tau)$

if  $ps \geq h$  or  $pD \geq h$  then

if  $\tau = P \rightarrow sx$  then

$P \rightarrow sx = \text{Parallelo}(\tau)$

else

$P \rightarrow dx = \text{Parallelo}(\tau)$

return  $\max(pD, ps) + 1$

return 0

(310612) : Dato un grafo  $G$  e  $A \subseteq V$  serve a definire che  
Hanno, se esistono, i due usciti  $B, C \subseteq V$  t.c.  
i)  $B \cap C = A$  e' l'unico solo  $i$  vertice di  $A$  è comune  
ii) ogni vertice  $v \in B$  ha un precezio che non ha alcun  
altra rete in  $A$   
iii) ogni vertice  $v \in C$  è raggiungibile da alcuno  
altro in  $A$ .

Se gli usciti non esistono, l'algo delle segnalate l'anche  
di una soluzione.

Il caso è le seguenti:

1. Considero  $B$  raggiungendo quelli sottostanti i vertici che  
raggiungono almeno un vertice in  $A$  facendo una  
DFS-vis. Sul grafo risposto su questo  
vertice dell'usciere  $A$  è
2. Considero  $C$  in modo simile, ma sul grafo  
non risposto
3. Pongo i vertici di  $A$  in alto a destra, qui  
es.  $c_3 = \text{Verde}$ . Se c'è sottostante  
sono già in  $B$  o in  $C$ , ovvero che sono i primi  
le condizioni preseletive, allora se non sono  
lementi di  $A$  restano false poiché ci  
sono elementi in comune a  $B$  e  $C$  o si hanno  
noi stessi elementi di  $A$

Alg0( $G, A$ )

$$B = C = \emptyset$$

Init( $G, A$ ) // If both biomes have green at  $A$ , color set  
at step 0 in while

for each  $a \in A$  do

  DFS-Visit( $GT, a, e_1$ )

  DFS-Visit( $G, a, e_2$ )

for each  $v \in V$  do

  if  $e_1[v] = N$  then

$B = B \cup \{v\}$

  if  $e_2[v] = N$  then

$C = C \cup \{v\}$

for each  $v \in V$  do

  if  $e_1[v] = N$  and  $e_2[v] = N$  and  $e_3[v] \neq \text{Vhole}$  then

    print "I see 'yellow' over edition"

    return false

return true

Init( $G, A$ )

for each  $v \in V$  do

$c_1[v] = c_2[v] = B$

for each  $a \in A$  do

$c_3[a] = \text{False}$

13/04/24 : Dato come unico input in gesso  $a$ ,  
selezio un algoritmo che risolvo l'equazione di due  
Vetori  $a, b \in V$  se almeno uno delle seg. è vero:

- i)  $a$  non raggiunge  $b$ , oppure
- ii)  $b$  non raggiunge  $a$

Ago ( $a$ )

Smis ( $a$ )

$$efc = CFC(a)$$

$$\text{return } efc(a) \neq efc(b)$$

Posso applicare l'algoritmo CFC al cubo  
le comp. fisi. comuni, senza la usura massimale  
di vedi es.  $\mathbb{S}^2 V \otimes \mathbb{H}_{\text{MNC}}$  una  $N$  e  $N$  u.

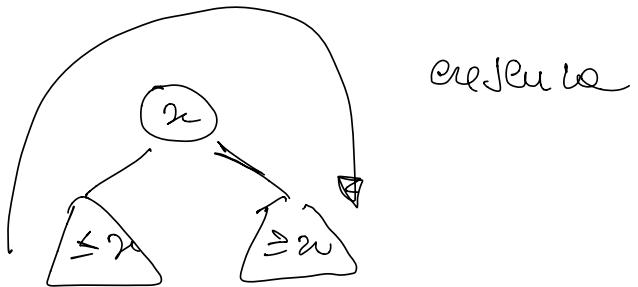
Noto che per risolvere il mio problema,  
è sufficiente vedere la completezza del CFC,  
risultando forse a MNC se  
si ha  $N$  oppure  $N$  non si

18/02/21: Dato un ABR T, generare algoritmo che restituisce una lista ordinata in modo efficiente delle chiavi nel T. (In modo da T possa essere già contenuto in un intervallo  $[k_1; k_2]$ )

Algo( $T, k_1, k_2$ )

$L = \emptyset$

if  $T \neq \text{null}$  then



cerca(w)

Algo( $T \rightarrow \text{sx}, k_1, k_2$ )

if  $T \rightarrow \text{key} \% 2 = 0$  and  $k_1 \leq T \rightarrow \text{key} \leq k_2$  then

$L = \text{insert}(L, T \rightarrow \text{key})$

Algo( $T \rightarrow \text{dx}, k_1, k_2$ )

return  $L$

insert( $L, \text{key}$ )

if  $L = \text{null}$  then

$L = \text{AllocElement}()$

$L \rightarrow \text{key} = \text{key}$

$L \rightarrow \text{next} = \text{null}$

else

$L \rightarrow \text{next} = \text{AllocElement}(\rightarrow \text{key} = \text{key})$

return  $L$

05/02/20 : Dato  $G$ , un insieme  $A$  tale che  $|A| \leq |V|$

Tale che  $A[i] \in V$  per ogni  $i \in \{1, \dots, n\}$  ed un vertice  $v$ , si deve un algoritmo che dati  $v$  e  $A$

$G, A, v$  eseguire l'insieme  $Z \subseteq V$  tale

$z \in Z \Leftrightarrow$  esiste un percorso da  $z$  a  $v$  tale percorso

di: raggiungere  $v$  prima per arrivare al vertice di  $A$

Algo  $(G, A, v)$

$$Z = \emptyset$$

$\text{Init} \leftarrow G, e_1, e_2$

$\text{DFS\_Visit}(G^T, v, e_1)$

for each  $a \in A$  do

  if  $e_1[a] = N$  then

$\text{DFS\_Visit}(G^T, a, e_2)$

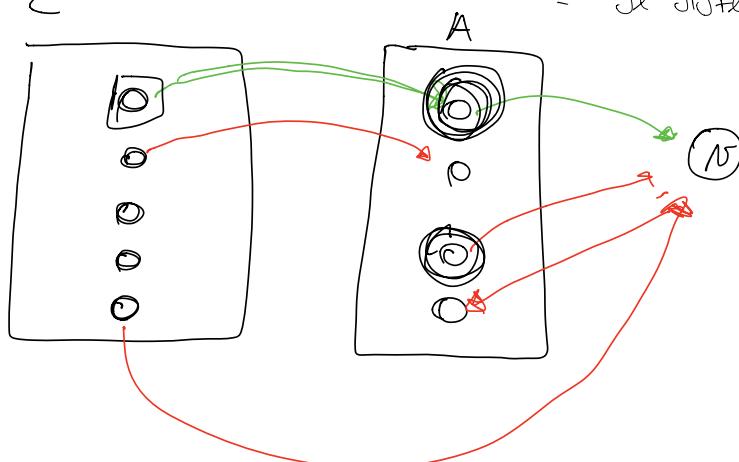
for each  $v \in V$  do

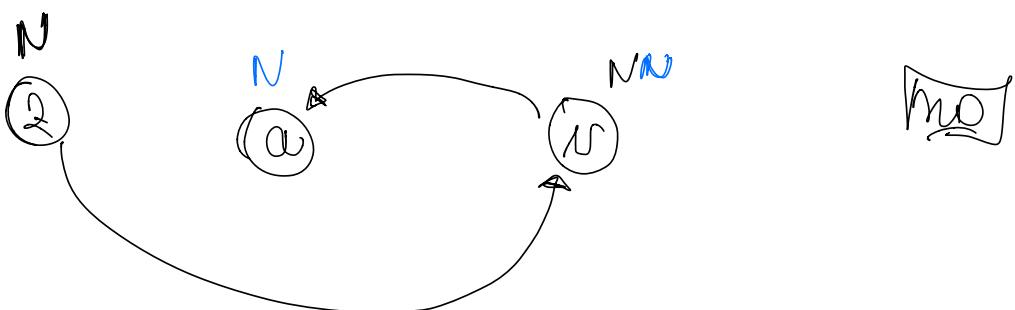
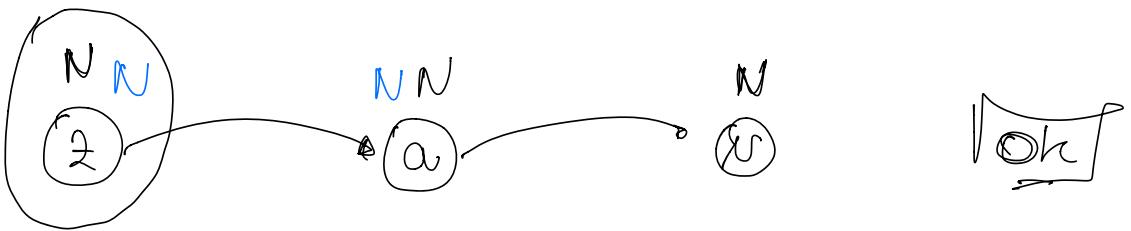
  if  $e_2[v] = N$  then

$$Z = Z \cup \{v\}$$

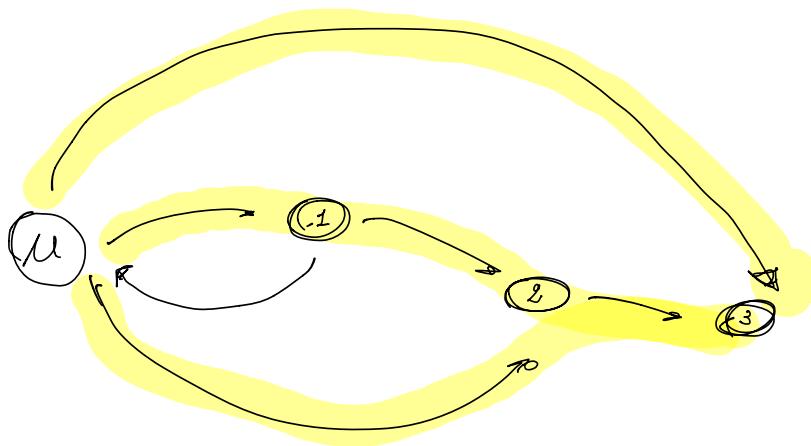
return  $Z$

- Verifico che  $A$  raggiunge  $v$  (arrivo)
- Verifico che  $A$  è raggiunto da  $Z$  ( $Z$  arriva)
- Se si tiene conto:  $Z$  arriva a  $v$



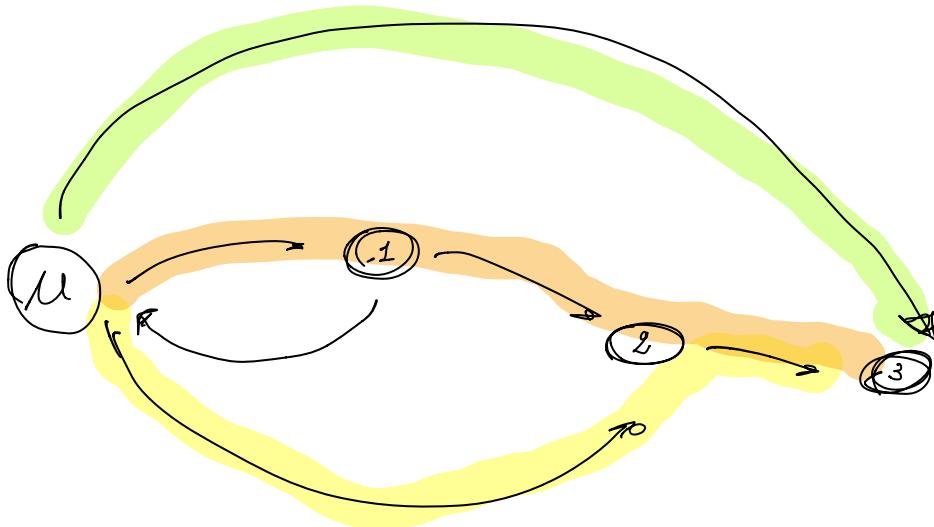


18/11/19: Dato un grafo  $G$  e un algoritmo  $VALCS$  che  
mostra ad un gen: vertice  $u$  numero, deve un  
algoritmo che visualizza le Tutte i percorsi semplici  
massimi (cioè non etichettabili) che partono  
da un vertice  $u$  e lo passano per due vertici  
Eci' c'è mostrata una delle possibili.

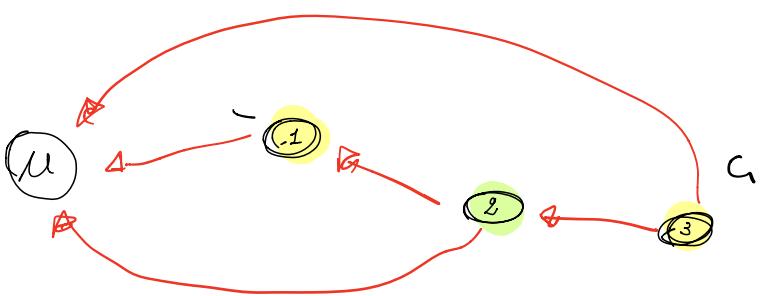


è un percorso  
tempo è un massimo

possibile a  
un vertice  
perito



1. Ambedue siamo liste i vertici però soffrono della loop
2. Ju c'è un problema d'ordine, oppure no DFN ridit  
Vedendo che esistono due vertici con diversi  
punti (true)



$\text{Alg}_0(G, \text{VALS}, \mu)$

$\Delta_{\text{MIX}}(G, e_1, e_2)$

$L = \text{BFS}(G, \mu, e_1)$

for each  $v \in L$  do

if  $e_1(v) \neq v$  then

$L = L \cup \{v\}$

if  $e_2(v) = B$  then

$w = \text{DFS\_VERIFY}(G^T, v, e_2, \text{VAL})$

return  $w$

$\text{DFS\_VERIFY}(G, v, e_2)$

$e_2(v) = G$

for each  $u \in \text{Adj}(v)$  do

if  $e_2(u) = B$  then

    if  $\text{EqualPoint}(VAL(v), VAL(u))$  then

        return false

$\text{DFS\_VERIFY}(G, u)$

$\text{Count}(v) = N$

return true

$\text{EqualPoint}(x, y)$

return  $x, y$  poss' on  $x, y$  identical

22/02/21: Dato un ABR  $T$  e una chiave  $h \geq 0$ ,  $k_1 \leq k_2$

Severo in alto alle cancellazioni da  $T$  tanti e più di modo che

i) hanno chiave in  $[k_1, k_2]$

ii) il sottoalbero in essi nascono contiene almeno un  
nodo con chiave minima  $\text{key}_1, \text{key}_2$  che  
è a profondità maggiore di  $h$

Ogni operazione all'alto deve restituire il val. delle  
min. profondità dei nodi con chiave minima nei  
sottoalberi ( $r, h, k_1, k_2$ )

if  $T \neq \text{null}$  then

$T \rightarrow \text{sx} = \text{Parella}(T \rightarrow \text{sx}, h, k_1, k_2)$

$T \rightarrow \text{dx} = \text{Parella}(T \rightarrow \text{dx}, h, k_1, k_2)$

if  $T \rightarrow \text{key} \in [k_1, k_2]$  then

$T = \text{Parella}(\text{Reduce}(T))$

$T \rightarrow \text{sx} = \text{Alegor}(T \rightarrow \text{sx}, h, 0)$

refin  $T$

Alegor( $T, h, \text{alt}$ )

if  $T \neq \text{null}$  then

if  $T \rightarrow \text{key} \in [k_1, k_2]$  then

if  $\text{alt} > h$  then

$\text{ret} = \text{true}$



else

ref = Alloc ( $\gamma \rightarrow \text{sx}$ , h, ++alf)

return ref

Dealloc ( $\gamma$ )

if  $\gamma \neq \text{null}$  then

if  $\gamma \rightarrow \text{sx} = \text{null}$  then

$\gamma = \gamma \rightarrow \text{dx}$

else if  $\gamma \rightarrow \text{dx} = \text{null}$  then

$\gamma = \gamma \rightarrow \text{sx}$

else

min = StackMin ( $\gamma \rightarrow \text{dx}$ )

Swap ( $\gamma \rightarrow \text{ty}$ , min  $\rightarrow \text{ty}$ )

Dealloc (min)



return  $\gamma$

StackMin ( $\gamma$ )

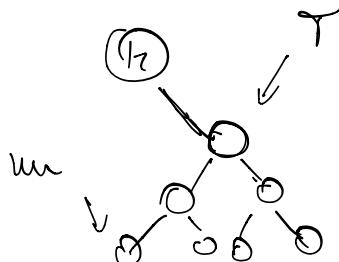
if  $\gamma \neq \text{null}$  then

if  $\gamma \rightarrow \text{sx} = \text{null}$  then

min =  $\gamma$

else

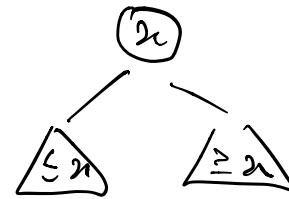
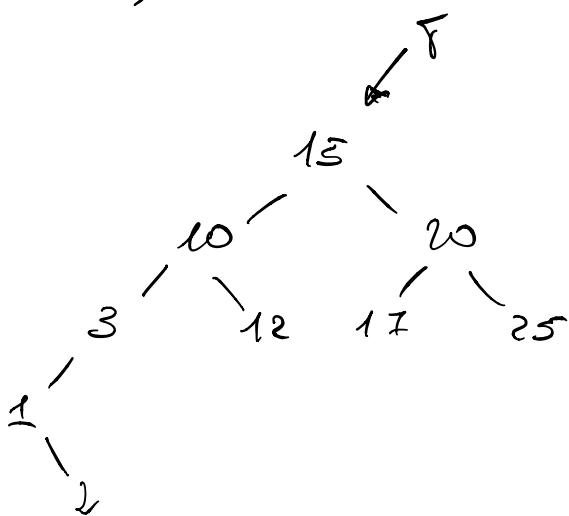
min = StackMin ( $\gamma \rightarrow \text{sx}$ )



return min

**Sequencia:** Dado un ABR  $\tau$  e una rama  $K_1$  e  $K_2$ ,  
 Devuelve un algoritmo iterativo que encuerre el  $\tau$  i modo  
 COM el que  $h \in K_1 \wedge h_1 \leq h \leq h_2$

$(S; 10)$



- Si  $\gamma \rightarrow \text{key} > h_2$ , el val de enc. Juega a  $S_x$
- Averni se  $\gamma \rightarrow \text{key} < h_1$ , el val de enc. Juega a  $D_x$
- Averni, sij, es  $\gamma \rightarrow \text{key} \in [h_1, h_2]$ , guarda  
 $\gamma \rightarrow \Delta x = \text{Enc}_c$ ;  $\gamma \rightarrow \alpha x = \text{Enc}_c$ ;  $\gamma = \text{EncRoot}(\gamma)$

- Operar -

$\gamma \rightarrow \Delta x = \text{Encelle}(\gamma \rightarrow S_x, h_1, h_2)$

$\gamma \rightarrow \alpha x = \text{Encelle}(\gamma \rightarrow \alpha x, h_1, h_2)$

if  $\gamma \rightarrow \text{key} \in [h_1; h_2]$  then  
 {  $\gamma = \text{EncelleReduce}(\gamma)$

return  $\gamma$

if  $\gamma \neq \text{null}$  then

$\gamma \rightarrow \delta x = \text{CancellingIntervals}(\gamma \rightarrow s_x, h_1, h_2)$

$\gamma \rightarrow \alpha x = \text{CancellingIntervals}(\gamma \rightarrow \alpha x, h_1, h_2)$

if  $\gamma \rightarrow y \in [h_1, h_2]$  then

$\gamma = \text{cancelRoot}(\gamma)$

return  $\gamma$

CancellingIntervals( $\gamma, h_1, h_2$ )

$CT = \gamma$

$ST = \text{last} = \text{null}$

while  $CT \neq \text{null}$  or  $ST \neq \emptyset$  do

if  $CT \neq \text{null}$  then

$\text{push}(ST, CT)$

$CT = CT \rightarrow s_x$

else

$CT = \text{top}(ST)$

if  $\text{last} \neq CT \rightarrow \alpha x$  and  $CT \rightarrow \alpha x \neq \text{null}$  then

$CT \rightarrow \delta x = \text{ref}$

$\text{push}(ST, CT)$

$CT = CT \rightarrow \alpha x$

else

if  $CT \rightarrow \alpha x = \text{null}$  then

$\text{ref} = \text{null}$

$CT \rightarrow s_x = \text{ref}$

$c\ell \rightarrow \alpha x = \text{rec}$

if  $c\ell \rightarrow by \in [u_1; u_1]$  then

|  $c\ell = \text{ParallelFor}(c\ell)$

Pop ( $\alpha u_0$ )

last =  $c\ell$

$c\ell = \text{rec}$

else

$c\ell \rightarrow \alpha x = \text{rec}$

if  $c\ell \rightarrow by \in [u_1; u_1]$  then

|  $c\ell = \text{ParallelFor}(c\ell)$

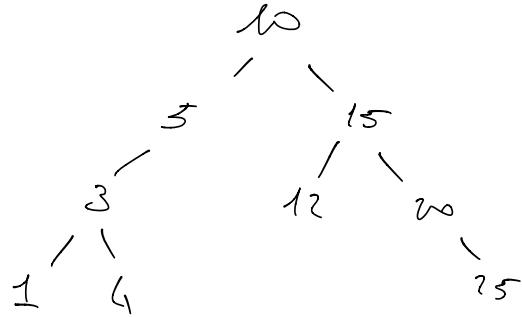
Pop ( $\alpha u_0$ )

last =  $c\ell$

$c\ell = \text{rec}$

return  $c\ell$

17/07/15: Dato un ABR T trovare un albero minimo che  
degli  $T \in \mathbb{Z}$  risulti che il modo in cui ha le  
numerazione più vicina possibile a  $k$  ma diverso da  $T$



$\text{Ago}(\gamma, k)$

$\text{ref} = \text{null}$

if  $\gamma \neq \text{null}$  then

    if  $\gamma \rightarrow \text{key} = k$  then

$\text{ref} = \top$

        if  $\gamma \rightarrow \delta x = \text{null}$  then

$\text{ref} = \gamma \rightarrow \delta x$

        else if  $\gamma \rightarrow \delta x = \text{null}$  then

$\text{ref} = \gamma \rightarrow \delta x$

    else

$\text{ref} = \text{Closest}(k, \gamma \rightarrow \delta x \rightarrow \gamma_y, \gamma \rightarrow \delta x \rightarrow \delta y)$

else if  $\gamma \rightarrow \text{key} < k$  then

$\text{ref} = \text{Ago}(\gamma \rightarrow \delta x, k)$

else

$\text{ref} = \text{Ago}(\gamma \rightarrow \delta x, k)$

return  $\text{ref}$

$\text{Closest}(n, y, z)$

if  $y - n < z - n$

    return  $y$

else

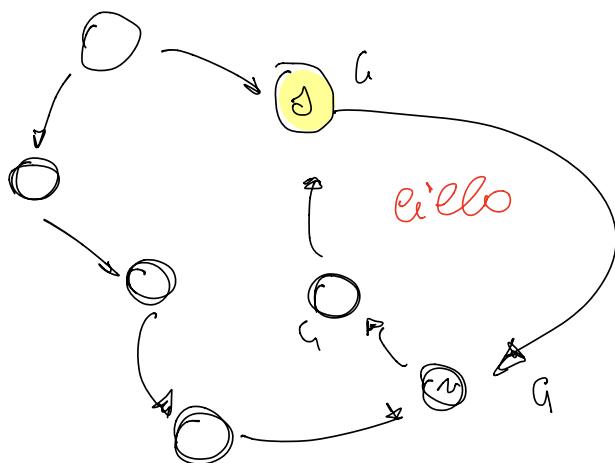
    return  $z$

26/01/19: Davi due grafi  $G_1$  e  $G_2$  t.c.  $G_1 = (V, E_1)$   
 $G_2 = (V, E_2)$  e deve vedere se sono sovrapposte.

i) Mentre percorro infatti  $G_1$  de fronte alle  $\delta$   
 posso per  $V$

ii) Tuttavia i percorsi infatti in  $G_2$  de fronte  
 alle  $\delta$  formano per  $V$

$G_1$ :



- Applico una DFS-VISIT de  $\delta$
- Se incontra n. bille e colorano in  $L_1$  i vertici incontrati
- Se  $v \notin L_1 \Leftrightarrow$  rett. true i condiz. vire

DFS\_VISIT ( $G, v, C$ )

colora [ $v$ ] =  $C$

per each  $u \in \text{Adj}[v]$  do

if colora [ $u$ ] =  $B$  then

DFS\_VISIT ( $G, u$ )

```

    else if color( $\mu$ ) = a then
    {
        L = Amalg(L,  $\mu$ )
    }

color( $\nu$ ) = N
return L

```

```

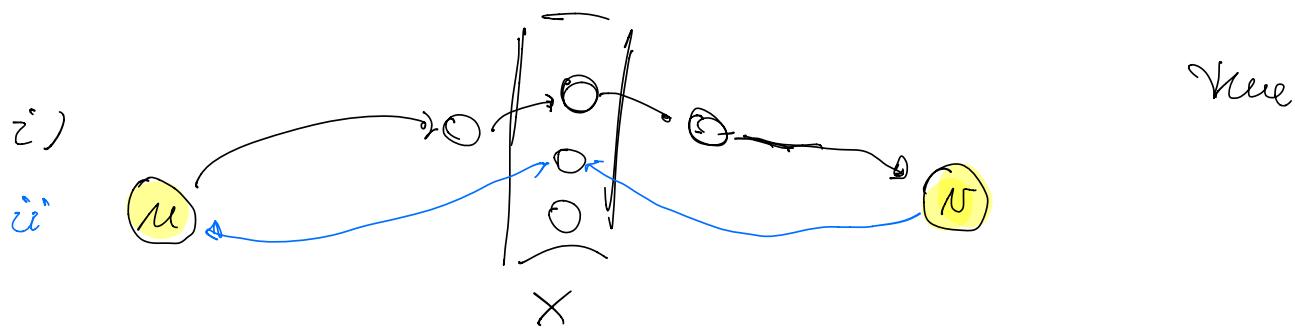
Ago(G1, G2, S, N)
Amalg(G1, G2)
L1 = DFS-VISIT(G1, S, L1)
L2 = DFS-VISIT(G2, S, L2)
for each  $x \in L_1$  do
{
    if  $x = \nu$  then
        return false           // 1st cond. mem fail case
    for each  $y \in L_2$  do
        if

```

13/01/20: Dado  $G$  e  $X \subseteq V$  o  $\mu, \nu \in V$  seveu m  
algoritmo de reflexão se  $\nu$  é sg. soluçao satisfatória

i) Dada  $\mu$  procede a parte da  $\mu$  e  $\nu$  em  $V$   
para  $\nu$  igual a  $x \in X$

ii)  $\Rightarrow \nu \neq \mu =$



false se existe  $x$  procedo de  $\mu$  a  $\nu$  o  
de  $\nu$  a  $\mu$  ele mai pômo fu qualde  $x \in X$

- Amplaço  $x \in X$
- Se  $\mu$  negye  $\nu$  o  $\nu$  negye  $\mu$ , false

Algô ( $G, X, \mu, \nu$ )

dm.v ( $G, X$ )

dm.v ( $G, X$ )

for each  $v \in V$

DFS- $\gamma_{\text{WIT}}(G, \mu)$

if  $\text{color}(\bar{v}) = \emptyset$

if  $\text{color}(\bar{v}) = \bar{v}$  then

for each  $x \in X$  ob

dm.v ( $G, X$ )

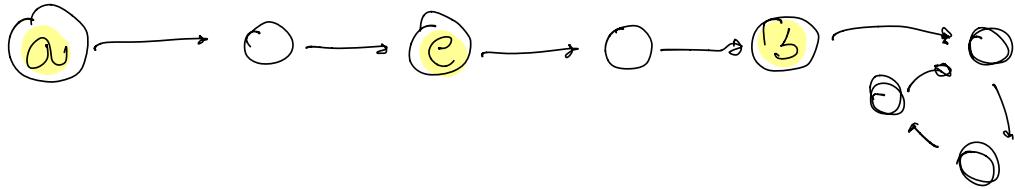
if  $\text{color}(x) = \bar{x}$

DFS- $\gamma_{\text{WIT}}(G, \nu)$

return  $\text{color}(\bar{\mu}) = \bar{\mu}$

return false

18/07/17: Dato  $G$  e  $a, b, c \in V$  scrivere un algoritmo che verifica se ogni percorso infinto da parte di  $a$  è formato da 6 passi per e più di  $c$ .



Algo ( $G, a, b, c$ )

dim $\nabla(G, e_1, e_2, e_3)$

DFS- $\nabla_{ij, \gamma}(G^T, e, e_2)$

DFS- $\nabla_{ij, \gamma}(G, e, e_2)$

if  $e_1(a) = N$  and  $e_2(b) = N$  then

return DFS- $\nabla_{ij, \gamma}(G, b, e_2)$

return false

DFS- $\nabla_{ij, \gamma}$ -color ( $G, n, \text{color}$ )

color( $v_N$ ) =  $G$

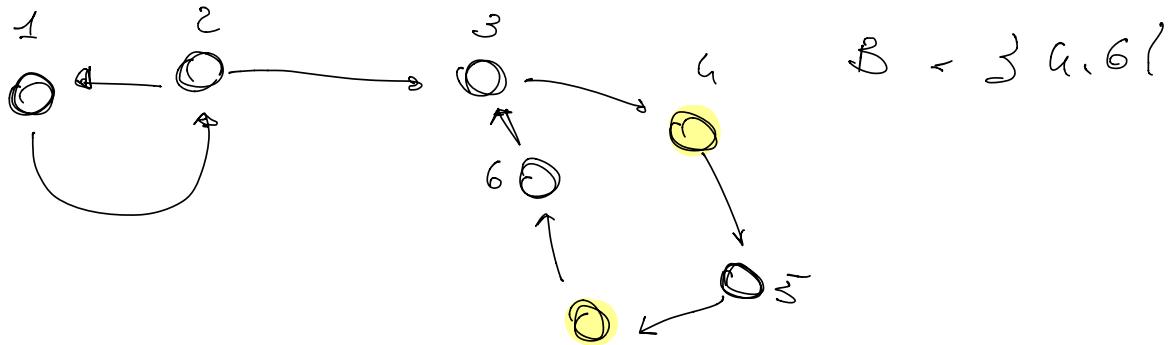
for each  $u \in \text{Adj}(v_N)$  do

if color( $u_N$ ) = $B$ then     DFS- $\nabla_{ij, \gamma}$ -color( $G, u, \text{color}$ )
if color( $u_N$ ) = $G$ then     return false

color( $v_N$ ) =  $N$

return false

26/02/17: Dato  $A$  e  $B \subseteq V$  serve un algoritmo che  
rispetti l'esistenza di un percorso infinito nel  
quale ciascun vertice di  $B$  riceve infinite visite.



Algo ( $A, B$ )

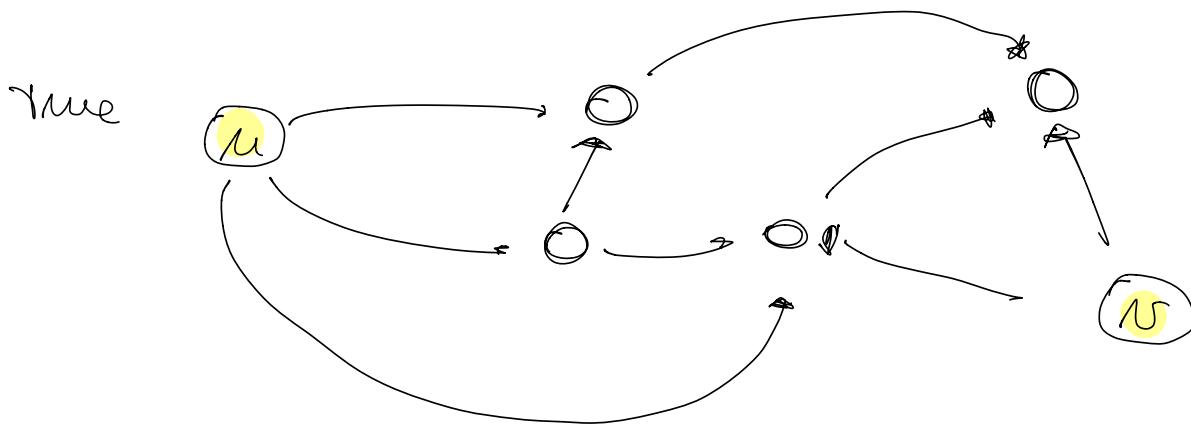
$$qfC = \text{CFC}(A)$$

for each  $b, b' \in B$  do

[ if  $qfC(b) \neq qfC(b')$  then  
[ [ return false

return true

26/01/17: Dato a Java un algoritmo che verifica  
l'esistenza di due vertici  $u, v$  tale che ogni percorso  
di ponte da  $u$  non passa per  $v$



True false se esiste un percorso di ponte  
da  $u$  che passa per  $v$

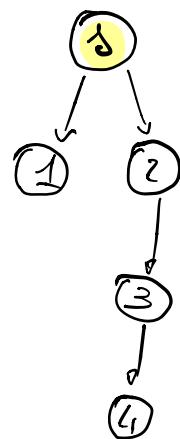
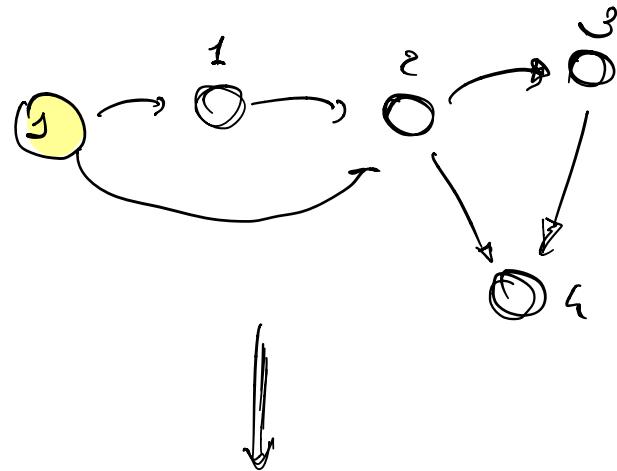
Algo ( $G$ )

```

    AMT(G)
    for each  $u, v \in V$  do
        DFS-Vis( $G, u$ )
        if color( $v$ ) = B then
            return True
    return False
  
```

return False

12/01/09. Dato  $G$  è un grafico  $N$  serve da algoritmo trasformare  $G$  in un altro realizzato in  $N$  dai percorsi minimi che si dipanano da  $v$



- Per ogni vertice costituzionale del PN di  $N$

$$PH(s, 1) = 5 \rightarrow 1$$

$$PH(s, 2) = 5 \rightarrow 2$$

$$PH(s, 3) = 5 \rightarrow 2 \rightarrow 3$$

$$PH(s, 4) = 5 \rightarrow 2 \rightarrow 3 \rightarrow 4$$

$PM(s, v, pred)$

if  $v = s$  return  $s$

else

return  $PM(s, pred[v])$   
 $\cup pred[s, v]$

$AGG(G, s)$

$A = \emptyset$

$Smir(G)$

$BFS(G, s) \quad // \text{ per gettare } pred[G]$

for each  $v \in V$  do

$A = A \cup PM(s, v, pred[v])$

return  $A$

26/06/16: Dato  $G$  è un vertice  $N$ . Si definisce  
 $G' = (r, \epsilon')$  di  $G$  un sottogrado minimale  
di  $G$  che ammette un ord. topologico che parte da  $N$   
se

i)  $\epsilon' \subseteq \epsilon$

ii)  $\forall (u, u') \in \epsilon - \epsilon'$  il grfo  $(n, \epsilon \cup \{(u, u')\})$   
non ammette un ord. top. che parte da  $N$

Se ne un algoritmo che restituisce un sottogrado minimale  $G'$   
di  $G$  che ammette un ord. top. che parte da  $N$

15/02/10 : Dato  $G = \langle S, V \rangle$  sevele in algoritmo che verifica se  $\forall x, y \in S \quad \text{efc}[x] = \text{efc}[y]$

Algo ( $G, \times$ )

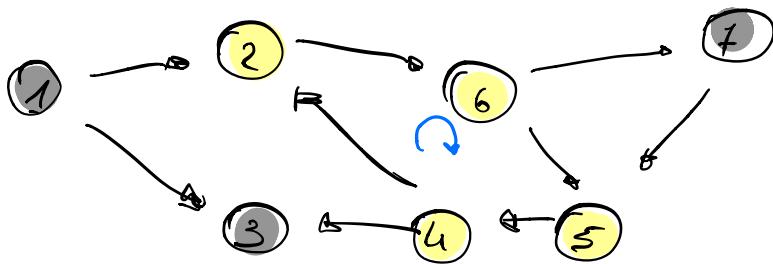
$$\text{efc} = \text{efc}(G)$$

for each  $x \in S$  do

| if  $\text{efc}[x] \neq \text{efc}[A[0]]$  then  
| | return false

return true

**Lemma 5:** Dato  $G$  e  $A \subseteq V$  scrivere un algoritmo che verifica l'esistenza di un percorso  $\pi = \underline{N_1} \rightarrow N_2 \rightarrow \dots \rightarrow \underline{N_k}$  tale che  $\{N_1, \dots, N_k\} \subseteq A$



$$A = \{2, 4, 5, 6\}$$

Abbiamo dei vertici i vertici di  $G$  nonne quei di  $A$ .

Poi: verifico l'esistenza di un ciclo su un quattro vertice di  $A$

Algo ( $G, A$ )

Jm.  $\gamma(G, A)$

for each  $a \in A$  do

| if DFS\_Visit( $G, a, a$ ) then  
 | | return true  
 | end if

return false

Jm.  $\gamma(G, A)$

for each  $N \in V$  do

| choose  $CNS = N$

for each  $a \in A$  do

| choose  $CON = B$

DFS-Ring(G, N, a)

Color(N) = a

for each  $u \in \text{Adj}(N)$  do

if  $\text{Color}(u) = \beta$  then

return  $\text{DFS-}\underline{\text{Ring}}(G, u, a)$

if  $\text{Color}(u) = a$  and  $u = a$  then

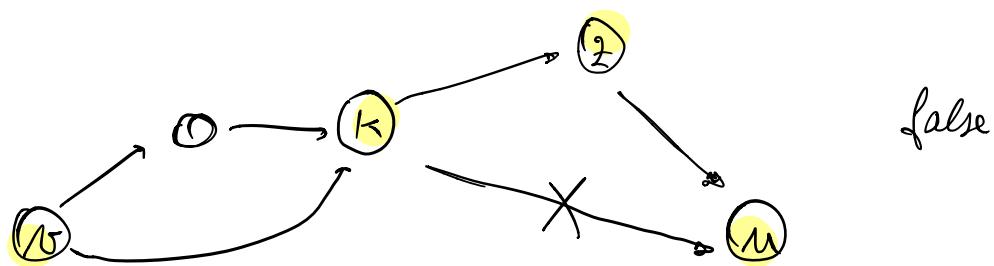
return true

Color(N) = N

return false

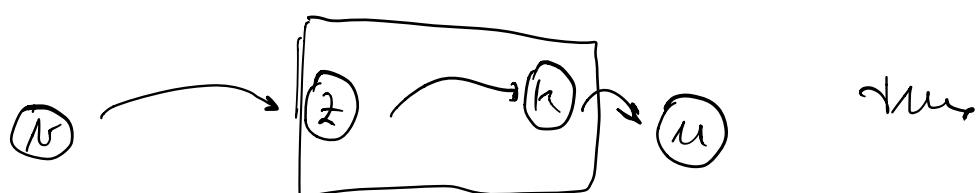
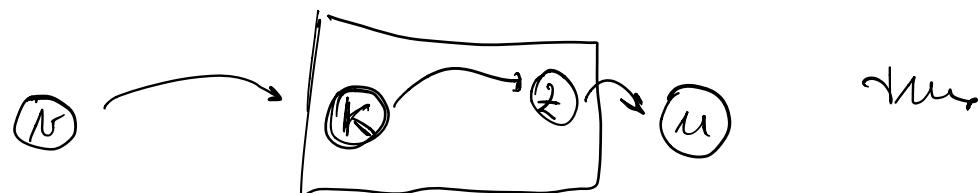
Dg 109116: Dato  $G$  e quattro vertici  $u, v, k, z$ . Scrivere un algoritmo che verifica se ogni percorso, non messo a scuola, da  $v$  alle  $k$  e  $u$  soddisfa le seguenti:

- i) Se  $\pi$  passa per  $k$  allora passa prima o dopo  $u$ , ovvero per  $z$



- Verifica rimanendo se  $v \rightarrow u$ .
- Se sì, verifica i.

Se  $v \rightarrow u$ :



Ora verificare se  $k \rightarrow z$  o viceversa  
e se  $v \rightarrow z/k$  e  $h/z \rightarrow u$

Ovvero

DFS- $\pi_{1,2}(G, k, e_1)$

DFS- $\pi_{1,2}(G, z, e_2)$

if  $e_1[2] = N$  or  $e_2[k] = N$  then  
 $k \neq 2$  o wieder

else pos!

DFS- $\gamma_{1,2}(G^T, k, e_1)$

..  $(e_1, 2, e_2)$

if  $e_3[N] = N$  or  $e_4[N] = N$  then  
 $N \neq 2 \neq k$

end while

DFS- $\gamma_{1,2}(G^T, N, e_3)$

if  $e_5[2] = N$  or  $e_6[k] = N$  then  
 $k \neq 2 \neq N$

Alg0 ( $G, N, u, k, z$ )

$\gamma_{u,z}(G)$

DFS- $\gamma_{1,2}(G, N, e_1)$

if  $e_1[u] = N$  then  $|N \neq u$

DFS- $\gamma_{u,z}(G, k, e_2)$

DFJ- $\gamma_{1,2}(G, 2, e_3)$

if  $e_2[2] = N$  or  $e_3[k] = N$  then

DFS- $\gamma_{1,2}(G^T, k, e_4)$

DFS- $\gamma_{1,2}(G^T, z, e_5)$

if  $e_6[N] = N$  or  $e_7[N] = N$  then

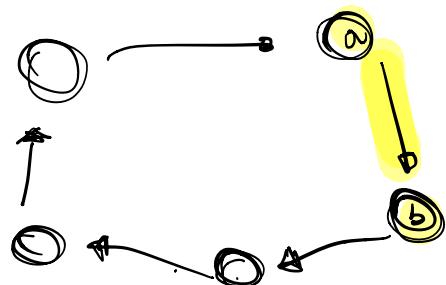
DFS- $\gamma_{1,2}(G^T, N, e_6)$

if  $e_8[2] = N$  or  $e_9[k] = N$  then

return true

return false

domanda: Data a serie di algoritmi che  
verifica se  $a \cdot b$  è un mero di qualche vila



DFS- $\text{YIJ.Y}(G, a, b, \text{false})$

color(v) = G

for each  $u \in \text{Adj}(v)$  do

if color(u) = S then

if  $u = b$  then

pomatoPerb = true

DFS- $\text{YIJ.Y}(G, u)$

if color(u) = G or pomatoPerb = true then

visit u

color(v) = N

true false