

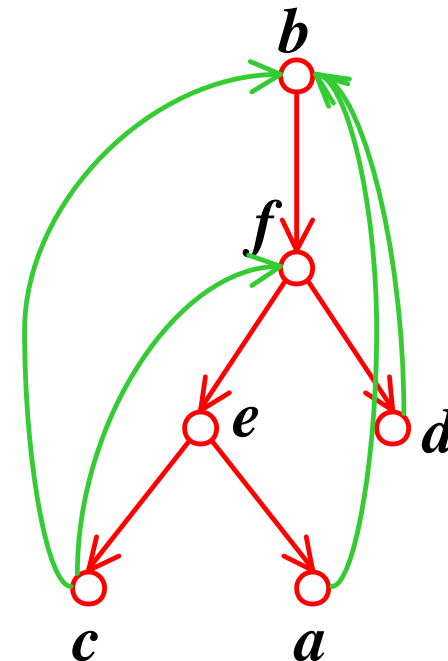
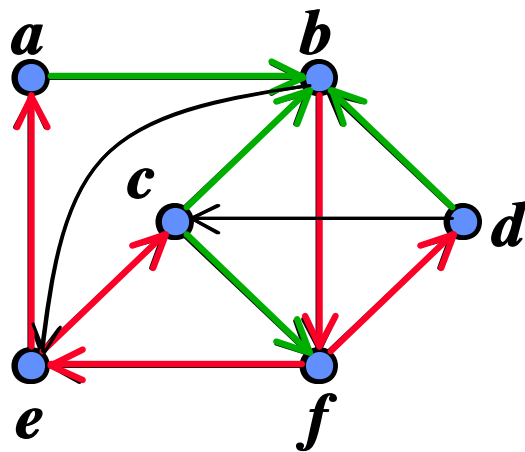
Algoritmi e Strutture Dati (Mod. B)

Algoritmi su grafi Ricerca in profondità (Depth-First Search) Parte II

Classificazione degli archi

Sia G_p la *foresta DF* generata da DFS sul grafo G .

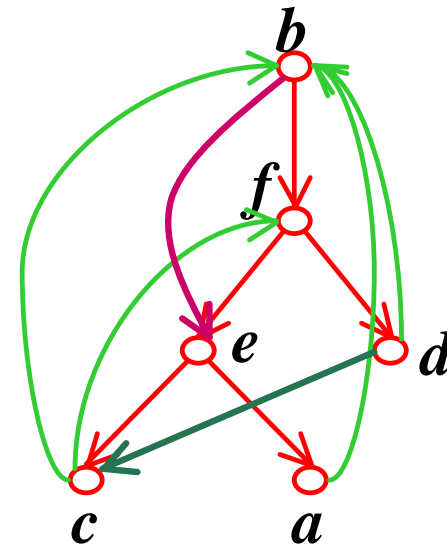
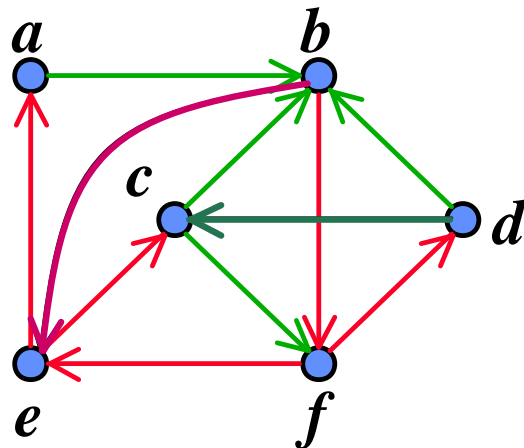
- **Arco d'albero**: gli archi della foresta G_p , tali che l'arco $(u,v) \in E_p$ se v è stato scoperto esplorando l'arco (u,v) .
- **Arco di ritorno**: gli archi (u,v) che connettono un vertice u con un *antenato* v nell'*albero DF*.



Classificazione degli archi

Sia G_p la *foresta DF* generata da DFS sul grafo G .

- **Arco in avanti**: archi (u,v) *non* appartenenti all'*albero DF* che connettono l'arco u con un *discendente* v
- **Arco di attraversamento (cross)**: tutti gli altri archi. Possono connettere *vertici nello stesso albero DF* (a patto che un vertice non sia antenato dell'altro nell'albero) o vertici in *alberi DF differenti*.

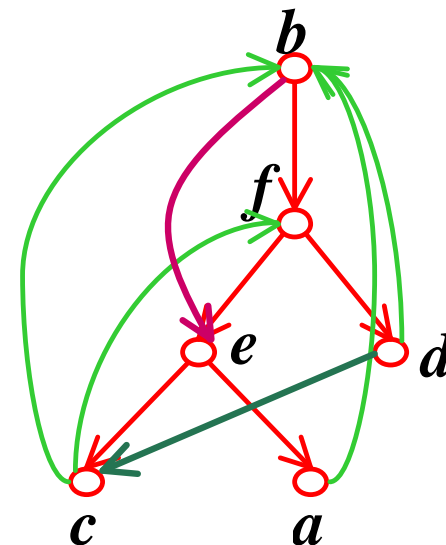
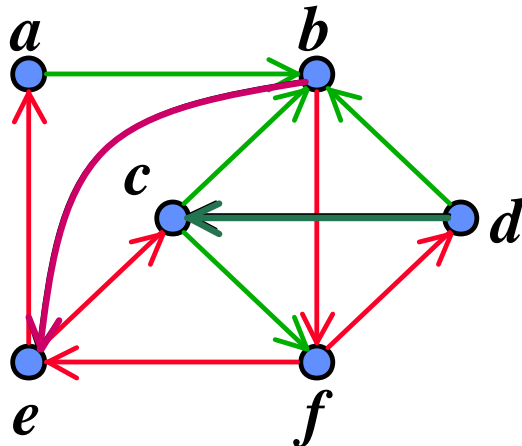


DFS per la classificazione degli archi

DFS può essere usata per classificare gli archi di un grafo G .

Si utilizza il colore del vertice che si raggiunge durante la visita dell'arco (u,v) :

- se v è *bianco*: allora l'arco è un *arco d'albero*
- se v è *grigio*: allora l'arco è un *arco di ritorno*
- se v è *nero*: allora l'arco è un *arco in avanti* o un *arco di attraversamento*

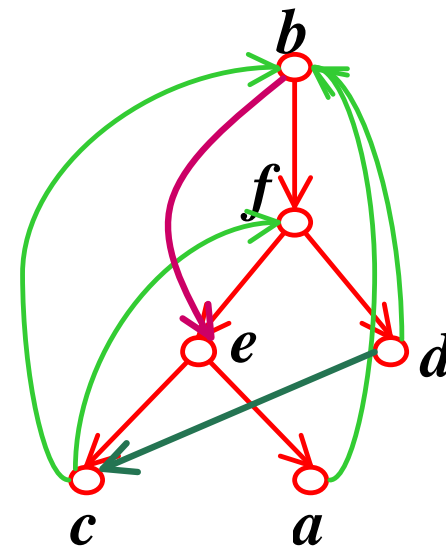
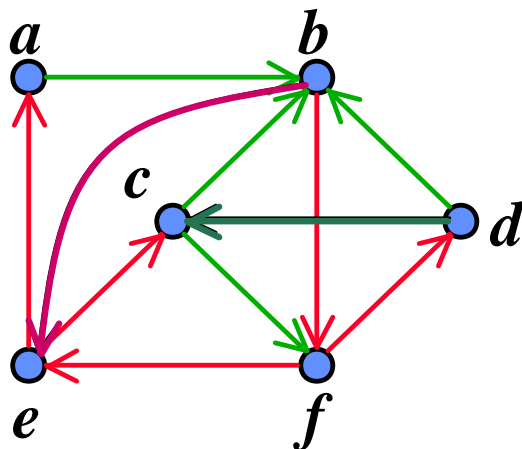


DFS per la classificazione degli archi

DFS può essere usata per classificare gli archi di un grafo G .

Si utilizza il colore del vertice che si raggiunge durante la visita dell'arco (u,v) :

- v è nero: allora l'arco è un *arco in avanti* o un *arco di attraversamento*
 - se inoltre $d[u] < d[v]$ allora è un *arco in avanti*
 - se $d[v] < d[u]$ allora è un *arco di attraversamento*



Proprietà di DFS

Teorema: Durante la DFS di un *grafo non orientato* G , ogni arco è un *arco dell'albero* o un *arco di ritorno*.

Dimostrazione: Sia (u,v) un arco arbitrario di G . Consideriamo il caso in cui $d[u] < d[v]$ (il caso $d[v] < d[u]$ è simmetrico).

Se $d[u] < d[v]$, v deve venir scoperto e visitato prima che si finisca di visitare u (v è nella lista di adiacenza di u).

Ora, se l'arco (u,v) viene esplorato prima da u a v , allora diventa un *arco dell'albero*.

Se l'arco (u,v) viene esplorato prima da v a u , allora diventa un *arco di ritorno*, poiché u è ancora grigio quando l'arco viene esplorato per la prima volta.

Esercizi

Dal libro di testo:

- **Es. 23.1-3** (calcolo del grafo trasposto G^T di G)
- **Es. 23.3-4**
- **Es. 23.3-6**
- **Es. 23.3-7**
- **Es. 23.3-8**

Applicazioni di DFS

Due problemi:

- calcolare l'ordinamento topologico indotto da un *grafo aciclico*.
- calcolare le componenti (fortemente) connesse (CFC) di un *grafo (non) orientato*.

Vedremo che entrambi i problemi possono essere risolti *impiegando* opportunamente l'*algoritmo* di DFS

Ordinamento topologico

Definizione: Dato un *grafo orientato aciclico* G (un *DAG*), un *ordinamento topologico* su G è un ordinamento lineare dei suoi vertici tale che:

- se G contiene l'arco (u,v) , allora u compare prima di v nell'ordinamento.

Ordinamento dei vertici in un *DAG* tale che

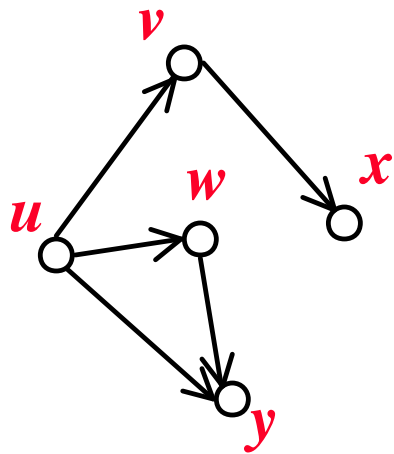
- se esiste un *percorso* da u a v , allora u compare prima di v nell'ordinamento

Ordinamento topologico

Ordinamento dei vertici in un **DAG** tale che

- se esiste un percorso da u a v , allora u compare prima di v nell'ordinamento

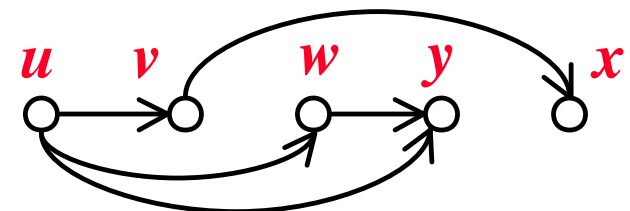
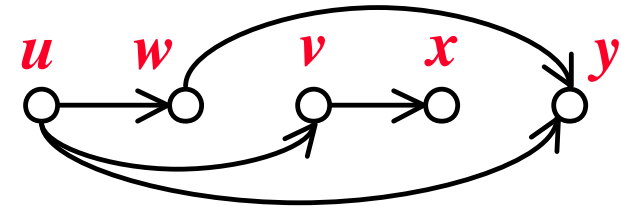
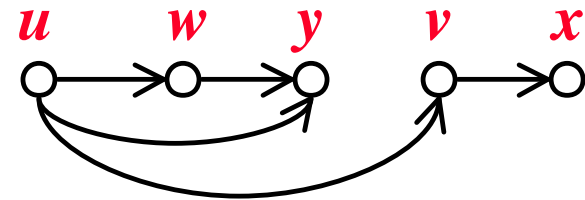
Ci possono essere *più ordinamenti topologici*.



$u w y v x$

$u w v x y$

$u v w y x$



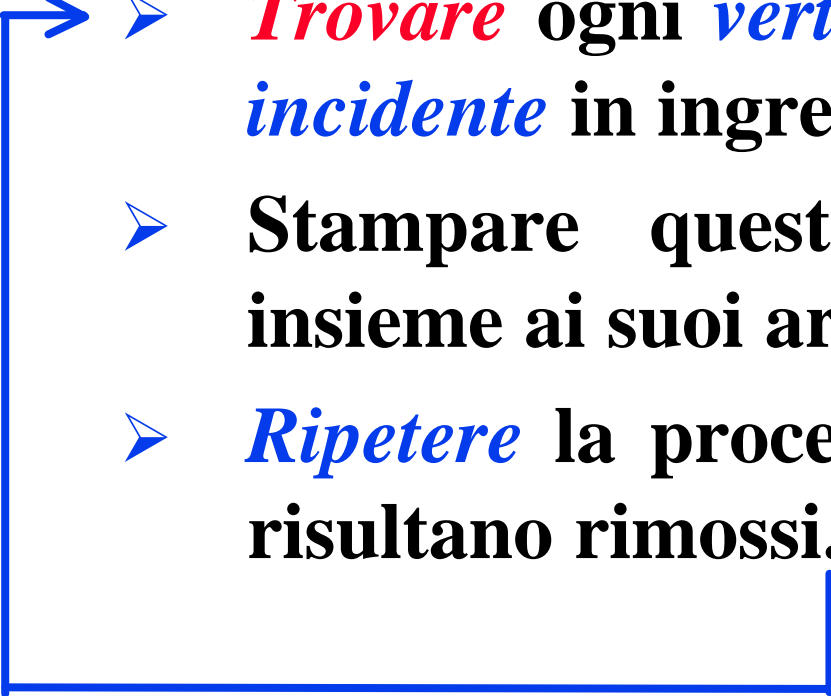
Ordinamento topologico

Problema: Fornire un algoritmo che *dato un grafo orientato aciclico*, ne calcoli e ritorni un *ordinamento topologico*.

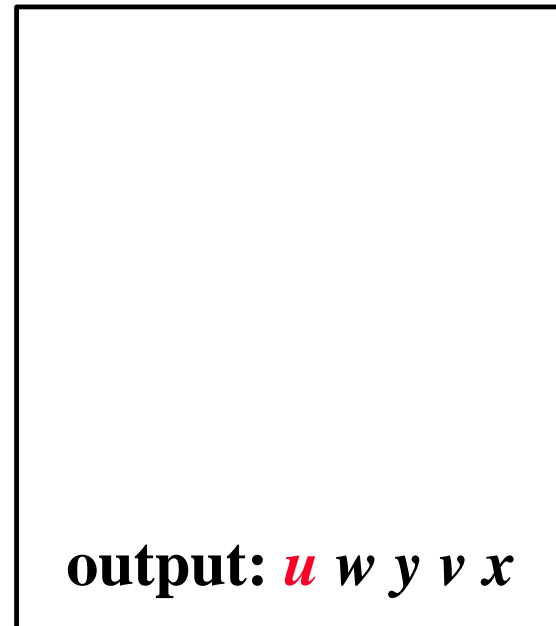
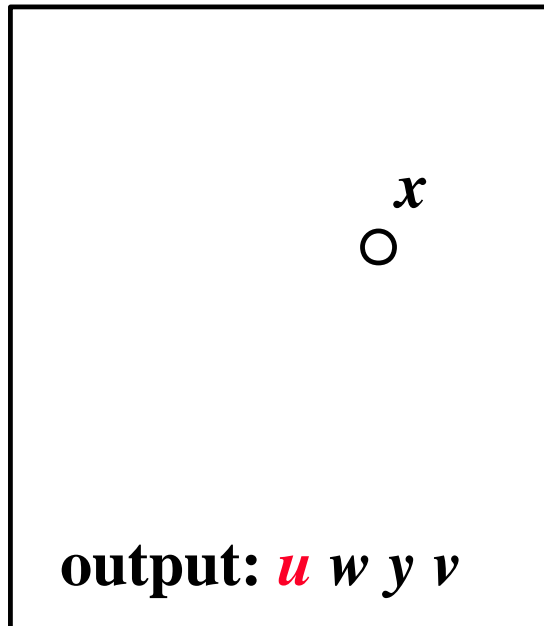
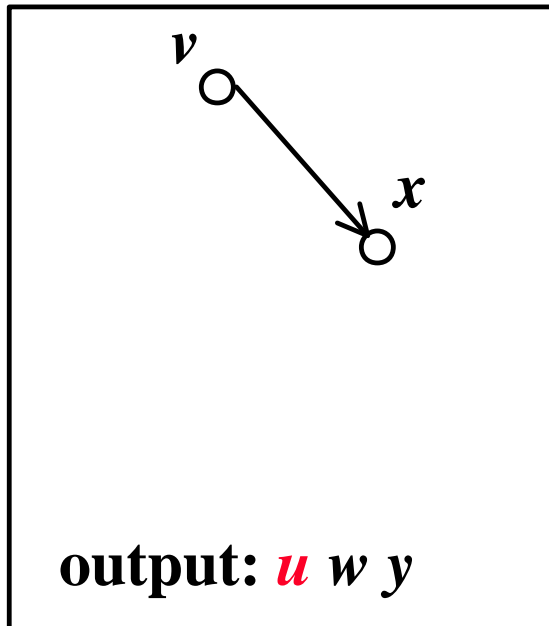
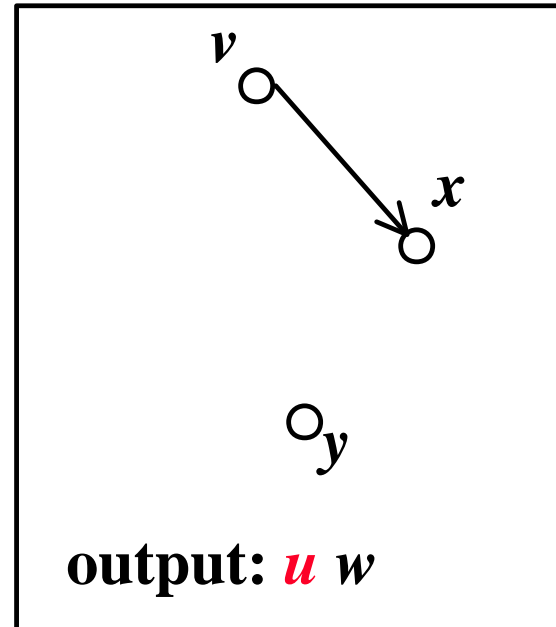
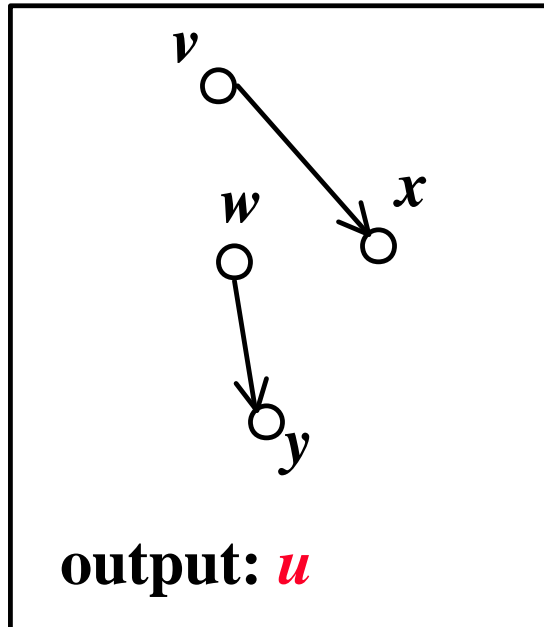
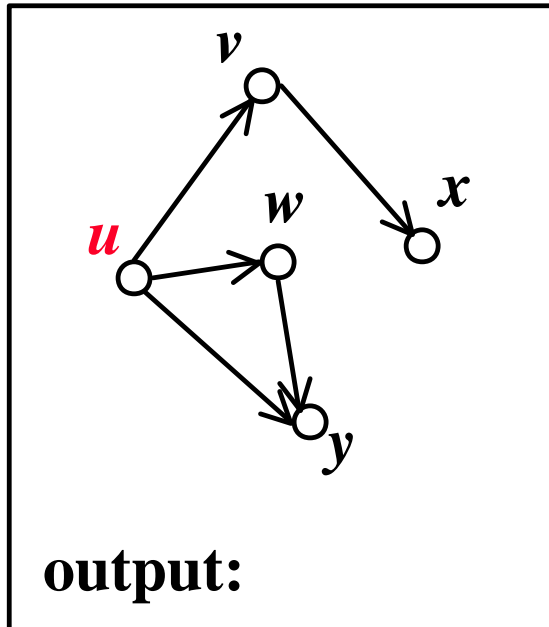
Soluzioni:

- Soluzione *diretta*
- Soluzione che utilizza *DFS*

Ordinamento topologico: algoritmo I

- 
- ***Trovare*** ogni *vertice* che *non* ha *alcun arco incidente* in ingresso
 - Stampare questo vertice e *rimuoverlo*, insieme ai suoi archi
 - *Ripetere* la procedura finché tutti i vertici risultano rimossi.

Ordinamento topologico: algoritmo I



Esercizio

Es 23.4-5: Terminare l'esercizio fornendo un *algoritmo* che in tempo $O(V+E)$ computa l'*Ordinamento Topologico* di un grafo G secondo l'idea appena illustrata.

Ordinamento topologico

Teorema: Un grafo orientato è *aciclico* se e solo se *DFS* su *G* non trova alcun *arco di ritorno*.

Dimostrazione:

se: Supponiamo che *G* contenga un ciclo *c*.

Allora *DFS* necessariamente troverà un *arco di ritorno*.

Infatti, se *v* è il *primo* vertice che viene scoperto in *c*, e (u,v) è l'arco che lo precede in *c*,

allora al tempo $d[v]$ c'è un percorso bianco da *v* a *u*.

Per il *teorema del percorso bianco*, sappiamo che *u* diventa un discendente di *v* nella *foresta DF*.

Perciò, (u,v) deve essere un *arco di ritorno*.

Ordinamento topologico

Teorema: Un grafo orientato è *aciclico* se e solo se *DFS* su *G* non trova alcun *arco di ritorno*.

Dimostrazione:

solo se: Supponiamo che *DFS* incontri un *arco di ritorno* (u,v) .

Allora il vertice v è un *antenato* di u nella *foresta DF*. Quindi esiste un percorso da v a u in G , e l'*arco di ritorno* (u,v) completa il *ciclo*, quindi G non è *aciclico*.

Ordinamento topologico: algoritmo II

Ordinamento-Topologico(G : grafo)

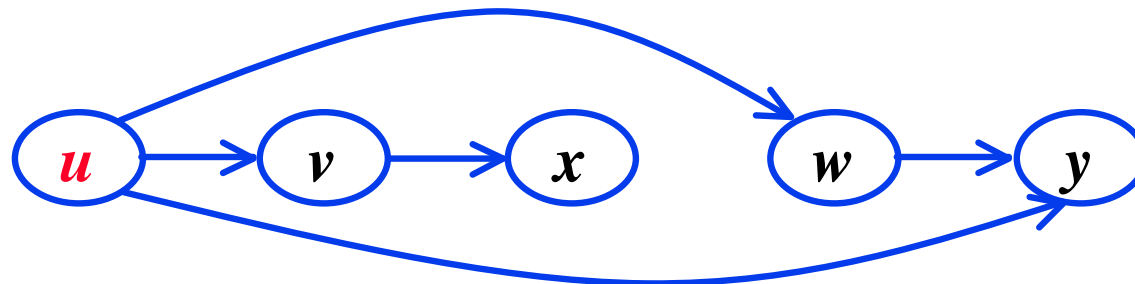
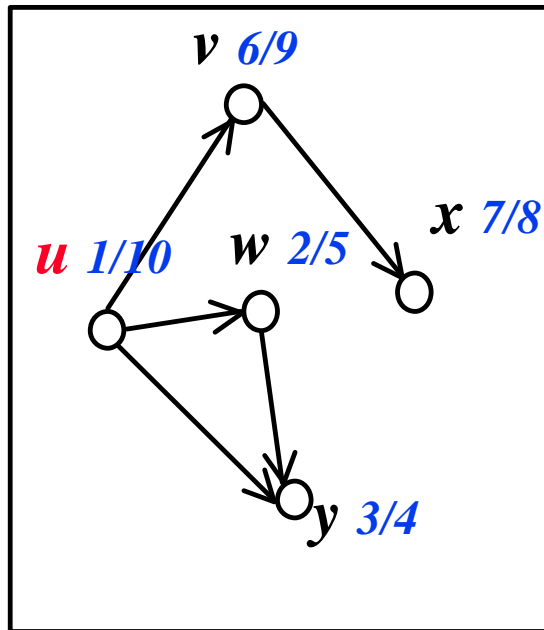
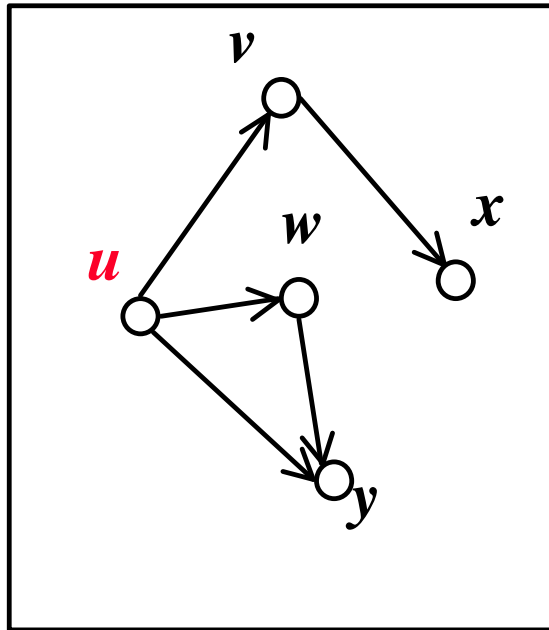
1 *DFS*(G) per calcolare i tempi $f[v]$

2 Durante la *DFS*, ogni volta che un vertice è terminato, *aggiungerlo* in testa ad una *lista*

3 Ritornare la *lista* di vertici

In altre parole, *l'algoritmo ordina i vertici in ordine inverso rispetto ai tempi di fine visita.*

Ordinamento topologico: algoritmo II



output: $u v x w y$

Correttezza dell'algoritmo II

Teorema: *Ordinamento-Topologico(G)* calcola correttamente l'ordinamento topologico di un grafo aciclico G .

Dimostrazione: *Dobbiamo dimostrare che vale la proprietà di ordinamento topologico: per ogni arco $(u,v) \hat{\in} E$, u precede v nell'ordinamento.*

Ma questo equivale a dimostrare che, *dopo la DFS, per ogni coppia di vertici u e v , se abbiamo che $(u,v) \hat{\in} E$, allora $f[v] < f[u]$.*

In tal caso abbiamo appunto che *u precederà v nell'ordinamento* (vedi Algoritmo).

Correttezza dell'algoritmo II

Teorema: *Ordinamento-Topologico(G)* calcola correttamente l'ordinamento topologico di un grafo aciclico G .

Dimostrazione: Dimostriamo che, dopo *DFS*, per ogni coppia di vertici u e v , se $(u,v) \in E$, allora $f[v] < f[u]$.

Preso un qualsiasi arco $(u,v) \in E$ esplorato da *DFS*, quando l'arco viene esplorato, v non può essere grigio,

altrimenti v sarebbe un antenato di u e (u,v) un arco di ritorno, contraddicendo il teorema precedente.

Quindi v o è bianco o è nero.

Correttezza dell'algoritmo II

Teorema: *Ordinamento-Topologico*(G) produce correttamente l'ordinamento topologico di un grafo aciclico G .

Dimostrazione: il vertice v è o bianco o nero.

Se v è bianco, allora diventa un *discendente* di u e
 $f[v] < f[u]$

Se v è nero, allora ovviamente sarà $f[v] < f[u]$.

In conclusione, dato l'ordine di inserimento nella lista e l'aciclicità di G , segue la *correttezza*.