Algoritmi e Strutture Dati (Mod. B)

Algoritmi su grafi

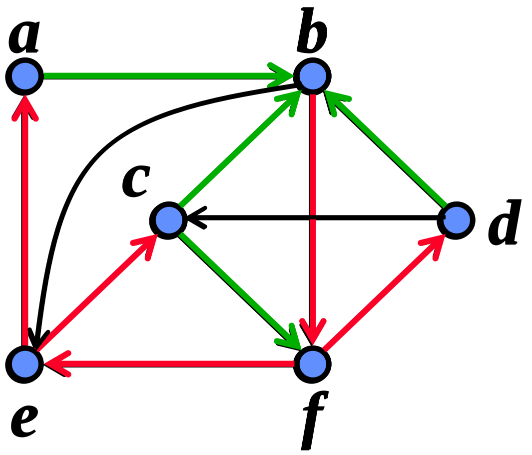
Ricerca in profondità

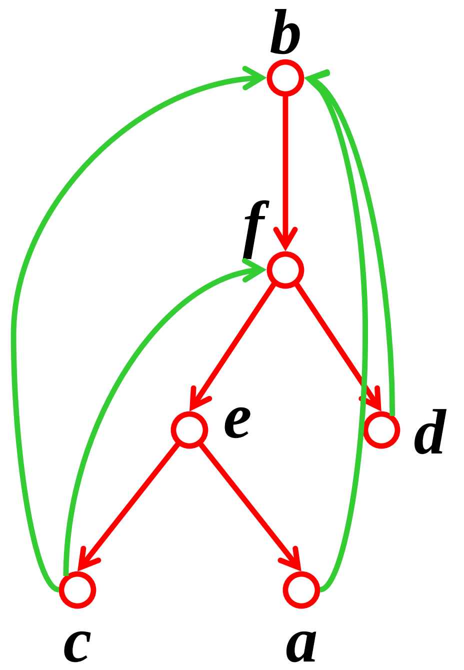
(<u>D</u>epth-<u>F</u>irst <u>S</u>earch) Parte II

Classificazione digli archi

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

- *Arco d'albero*: gli archi della foresta G_π, tali che l'arco $(u,v)\in E_\pi$ 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*.

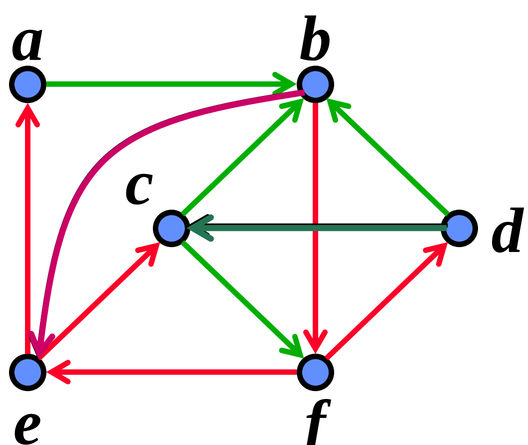


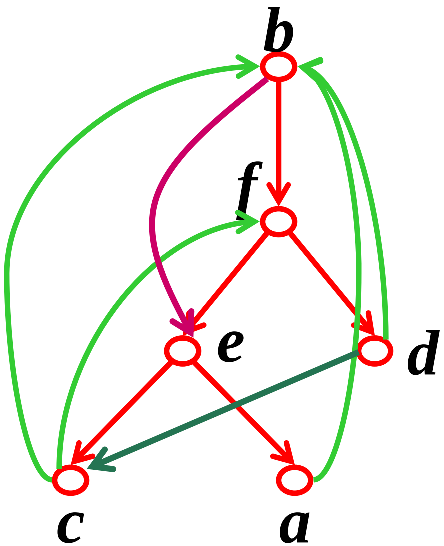


Classificazione digli archi

Sia G_π 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*.**



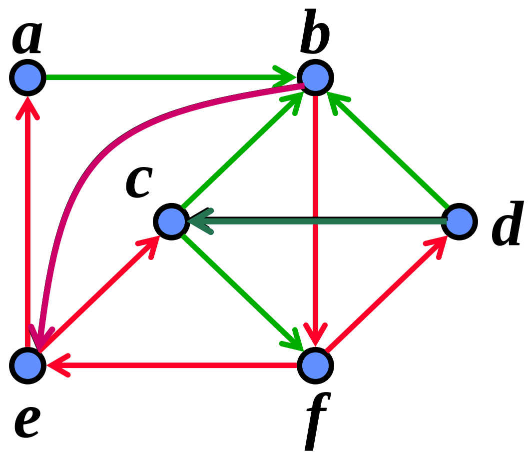


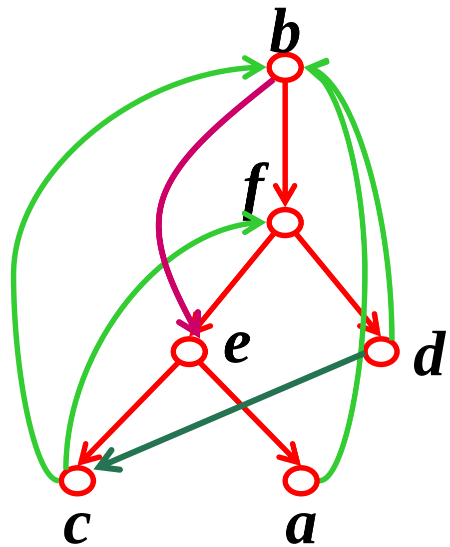
DFS per la classificazione digli archi

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***



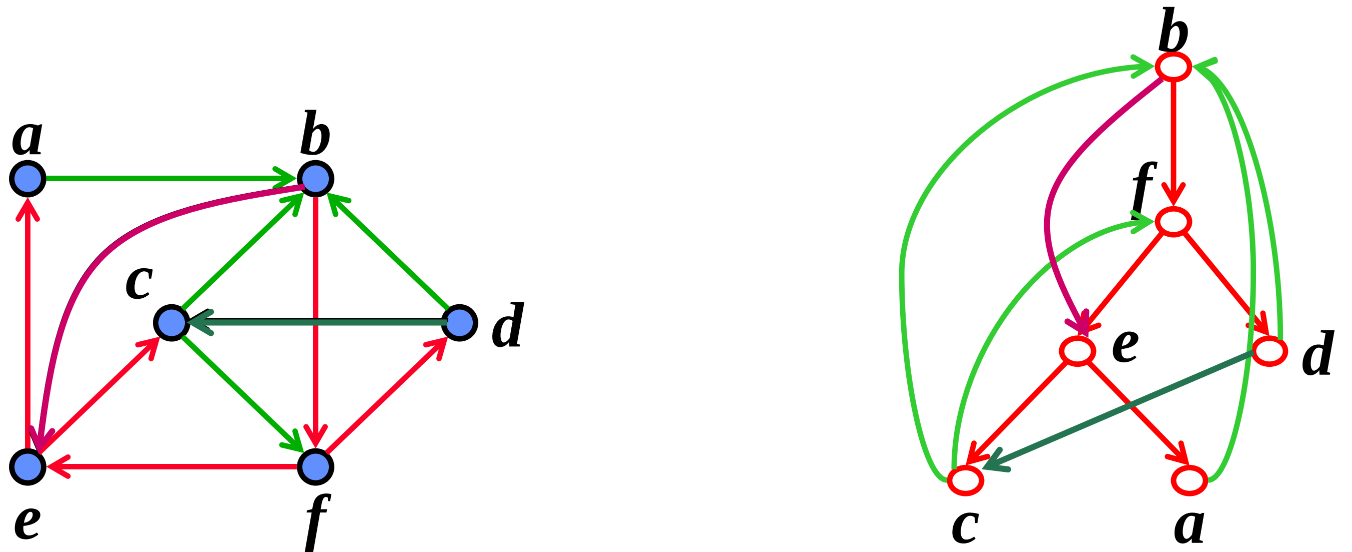


DFS per la classificazione digli archi

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 attraversa-mento***
  - **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 essere stato scoperto e visitato prima che si termini u (poiché v è nella lista di adiacenza di u).

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

Se invece l'arco (u,v) viene esplorato prima nella direzione da v a u, allora l'arco (u,v) diventa un *arco di ritorno*, poiché u è già grigio quando l'arco viene attraversato.

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**

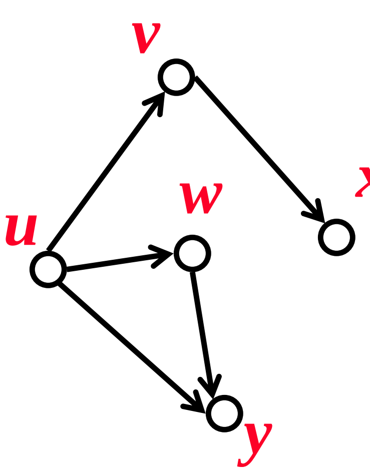
- **più in generale, se esiste un *percorso* da *u* a *v*, allora *u* compare prima di *v* nell'ordinamento**

Ordinamento topologico

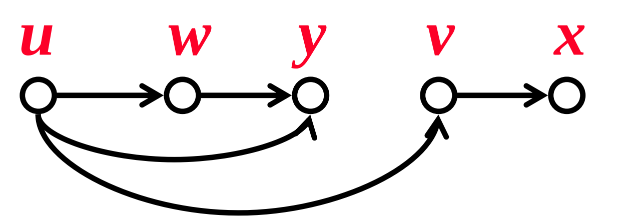
***Ordinamento* dei vertici in un *DAG* tale che**

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

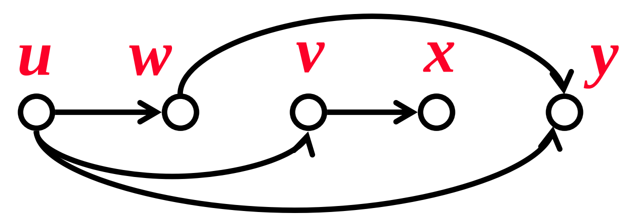
Ci possono essere *più ordinamenti topologici.*



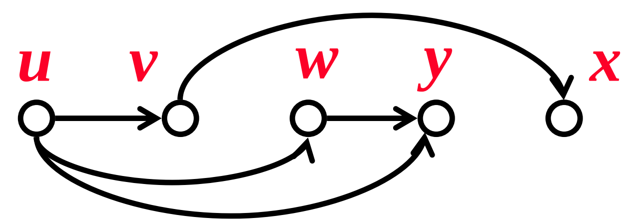
u w y v x



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*.**

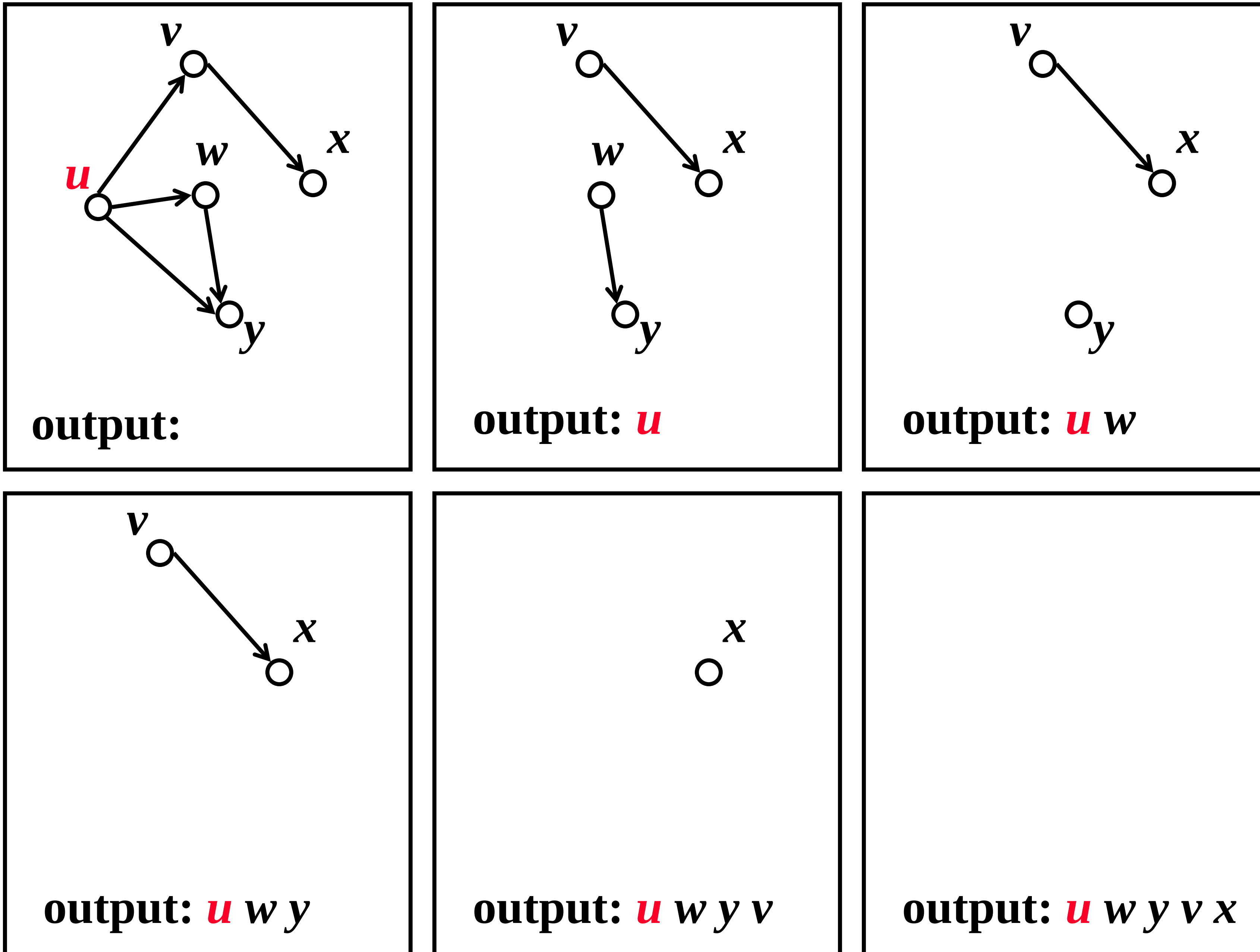
Soluzioni:

- **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*” (virtualmente) insieme ai suoi archi**
- ***Ripetere*** **la procedura finché tutti 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, sia *v* è il *primo* vertice che viene scoperto in *c*, e (*u*,*v*) l'arco del ciclo *c* che entra in *v*.

Poiché *v* è il *primo* vertice di *c* che viene scoperto, al tempo *d*[*v*] c'è un percorso bianco da *v* a *u* (quello che segue *c* fino ad *u*).

Per il *teorema del percorso bianco*, *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 certamente un percorso che va da *v* a *u* nel grafo *G*.

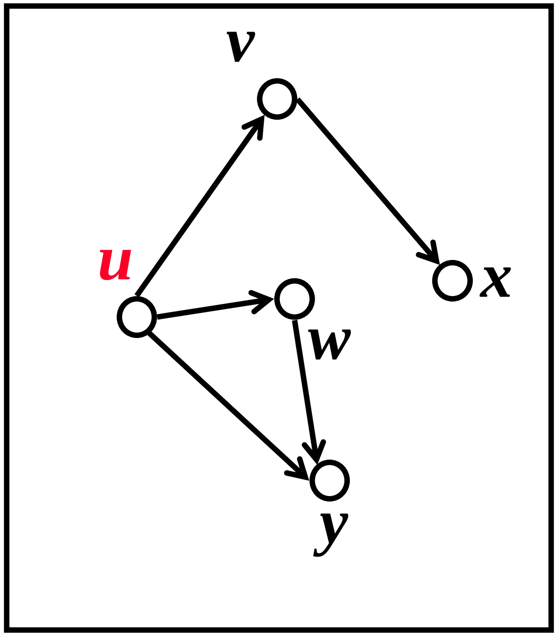
Tale percorso, concatenato con l'*arco di ritorno* (*u*,*v*), forma un ciclo, quindi il grafo *G non è aciclico*.

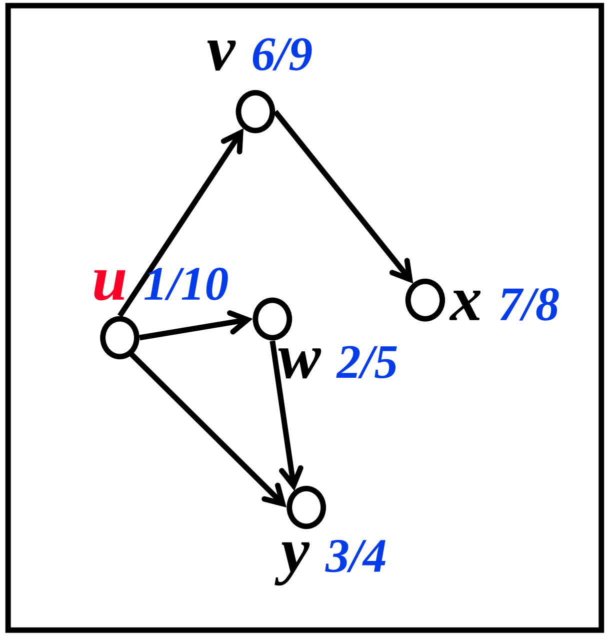
Ordinamento topologico: algoritmo II

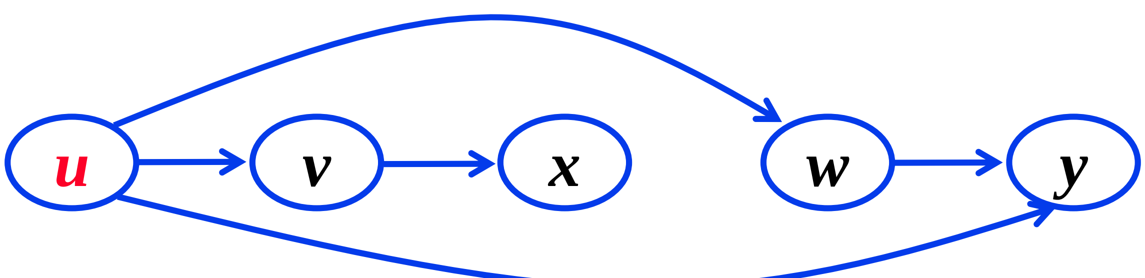
```
Ordinamento-Topologico(G: grafo)

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

2 Durente la DFS, ogni volta che un ver-
  tice è  terminato, aggiungerlo in testa
  ad una lista

3 Ritornare la lista di vertici
```

In altre parole, ***l'algoritmo costruisce una lista di vertici seguendo l'ordine inverso dei 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)*** $\in$ ***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)*** $\in$ ***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.**

a) Se *v* è bianco, allora diventa un *discendente* di *u* e *f*[*v*]<*f*[*u*] (per il teorema della Struttura a parentesi)

b) 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*.