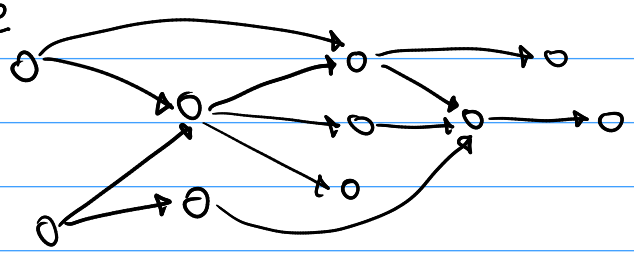
GRAFI DIRETTI ACICLICI (DAG) & ORDINAMENTO TOPOLOGICO

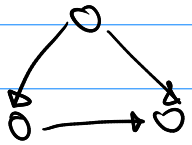
Def Un grafo $G = (V, E)$ è un DAG se è
- un grafo diretto
- che non contiene cicli

Directed Acyclic Graph ← lo dice anche il nome!!!

Esempio



oppure



Ordinamento topologico

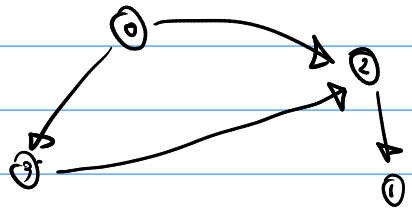
- Un ordinamento topologico di un grafo diretto $G=(V,E)$ è un <u>ordinamento</u> σ dei vertici tale che

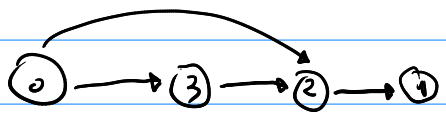
per ogni arco (u,v) in $E(G)$ si ha che

$$\sigma(u) < \sigma(v)$$

ovvero: un ordinamento dei vertici tale che tutti gli archi del grafo vadano da un vertice di posizione minore ad uno di posizione maggiore

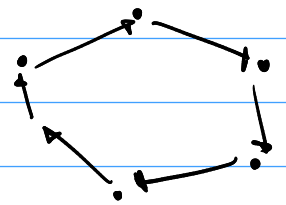
E.g.





tutti gli archi vanno da sinistra a destra

OSSERVAZIONE Un grafo diretto contenente cicli non ha un ordinamento topologico.



ESERCIZIO dimostrare questa osservazione

Thm Un grafo diretto ha un ordinamento topologico se e solo se è un DAG

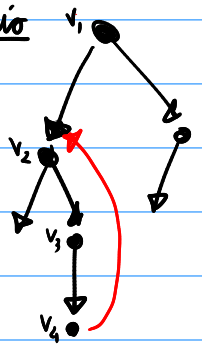
dim (A) Se ha un ciclo → non ha un ordinamento topologico come da OSSERVAZIONE precedente

(B) Se il grafo è un DAG → ha un ordinamento topologico
La dimostrazione di questa parte segue da quello che mostreremo in questa parte

La DFS può essere usata per calcolare un ordinamento topologico

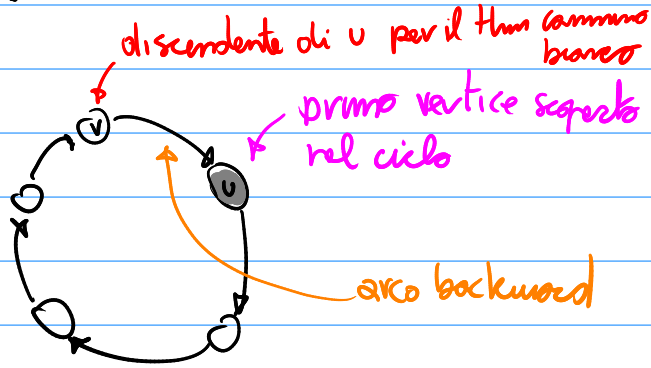
INTUIZIONE 1 Se la visita DFS produce un arco backward allora esiste un ciclo e quindi NO ORDINE TOPOLOGICO

Esempio



Il ciclo $v_2 \to v_3 \to v_4$ impedisce l'ordinamento topologico

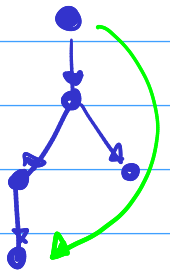
INTUIZIONE 2 Se esiste un ciclo allora il primo vertice trovato dalla visita nel ciclo ha un cammino bianco fino al suo predecessore. Quindi viene trovato un arco backward



Lemma : DFS(G) trova un arco backward se e solo se G non è DAG

- La domanda allora è: come definiamo l'ordinamento topologico se non ci sono cicli (e quindi archi all'indietro in una DFS)?

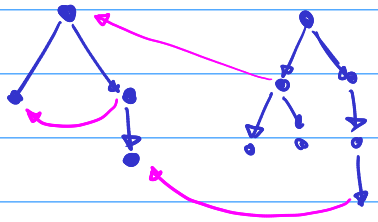
INTUIZIONE 3



- I discendenti di un vertice devono venire dopo i relativi antecedenti
- Questo fa sì che l'ordinamento rispetti tutti gli archi dell'albero e gli archi forward.

INTUIZIONE 4

- Diciamo che i discendenti diretti sono enumerati da sx a dx in base all'ordine di visita e che vale lo stesso per alberi nella foresta ottenuta



- I vertici più a dx devono venire prima di quelli a sx, nell'ordinamento
- questo fa sì che gli archi crossing siano rispettati

OSSERVAZIONE Dati due vertici u e v

- fine[v] > fine[u] se u discendente di v (thm delle parentesi)
- Se u non è discendente di v o viceversa, gli intervalli inizio–fine sono disgiunti (thm delle parentesi)

qualunque arco crossing  implica inizio[u] > inizio[v] e quindi fine[u] > fine[v]

IN CONCLUSIONE Se ordino tutti i vertici di G in base al valore di fine[v] in modo decrescente ottengo che
- tutti gli archi dell'albero, forward, crossing sono rispettati
- se G è un DAG, non ci sono archi backward

Questo ci fornisce un algoritmo per l'ordine topologico di un DAG

- INPUT: $G=(V,E)$ grafo diretto
- output: ordine topologico

1. Esegui la DFS su G
2. Se la visita produce un arco all'indietro, segnala che esiste un ciclo
3. Altrimenti, ordina $V(G)$ in modo decrescente in base al valore fine[v]

OSS Non è necessario ordinare $V(G)$ esplicitamente: è sufficiente aggiungere ad una sequenza inizialmente vuota ogni vertice quando questo diventa nero
- in questo modo i vertici sono ordinati in modo crescente rispetto a fine[v]
- è sufficiente invertire la sequenza alla fine

Complessità totale: una DFS più un'inversione

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

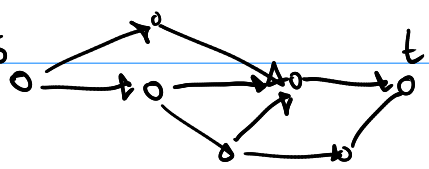
Esercizio 22.4-2

Descrivete un algoritmo di complessità $O(|V|+|E|)$ che riceve un grafo orientato aciclico

$$G=(V,E)$$ e due vertici $s,t \in V$

e calcola il numero di cammini distinti da s a t.

E.g.



ci sono 4 cammini

- Esercizio 22.4-3

Descrivere un algoritmo che determina se un grafo non orientato $G=(V,E)$ contiene un ciclo oppure no.

La complessità deve essere $O(|V|)$, quindi indipendente da $|E|$.

INDIZIO Se G ha un ciclo, quanti archi esplora la DFS prima di trovare un arco backward.

- Esercizio 22.4-5

Un altro modo di eseguire un ordinamento topologico su DAG, $G=(V,E)$

1. trovare un vertice di grado entrante 0
2. aggiungerlo alla sequenza ordinata
3. eliminarlo dal grafo
4. ripetere

Usando questo schema trovate un algoritmo alternativo di complessità

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