

## CORRETTEZZA BELL' ALGORITMO TOPOLOGICAL-SORT

TOPOLOGICAL-SORT prende in input un grafo orientato aciclico e restituisce in output un ordinamento topologico, ovvero un ordinamento lineare di  $V$  tale che  $(u, v) \in E \Rightarrow u < v$

## TOPOLOGICAL-SORT

1. Invocare DFS
2. al termine dell'esplorazione di un vertice inserirlo in una lista collegate
3. return le liste collegate

RUNNING-TIME  $\Theta(|V| + |E|)$  dato che DFS impiega  $\Theta(|V| + |E|)$  e l'implemento in lista è  $O(1)$

LEMMA 1:  $G$  è un DAG  $\Leftrightarrow$  una ricerca DF non genera archi all'indietro

Supponiamo che una ricerca produce un arco indietro. Allora  $v$  è predecessore di  $u$  nelle foreste DF. Quindi  $G$  contiene un cammino da  $v$  a  $u$  e l'arco  $(v, u)$  completa il ciclo

Supponiamo che  $G$  contiene un ciclo finito

Supponiamo che  $v$  contenga un arco. Se  $v$  è il primo vertice ad essere scoperto in  $C$ , sia  $(u, v)$  l'arco precedente in  $C$ . Al tempo v.d i vertici formano un cammino di vertici bianchi.

Allora del White-Peth-Theorem sappiamo che  $v$  è un discendente di  $u$  nelle foreste DF. Pertanto  $(u, v)$  è un arco all'indietro.

THM. topological sort produce un ordinamento topologico di un DAG ricevuto in input

E' sufficiente mostrare che  $\forall u, v \in V (u \neq v)$   
Se  $(u, v) \in E$  allora  $v.f < u.f$

Considerando un qualunque arco in DFS

Quando quest'arco viene esplorato  $v$  non può essere grigio altrimenti sarebbe un arco all'indietro contraddicendo il LEMMA 1.

Perciò  $v$  può essere bianco o nero

Se  $v$  è bianco lo esploriamo e diventa discendente di  $u$  allora  $v.f < u.f$

Se  $v$  è nero, già è stato esplorato e  $v.f < u.f$

CORRETTEZZA DELL'ALGORITMO BELLMAN-FORD

Poniamo definire il peso di un cammino con il peso degli archi che lo compongono

OSS: Potrebbero esserci pesi negativi. Se  $G$  non

contiene cicli di peso negativo raggiungibili da  $s$ , allora il peso del cammino minimo  $\delta(s, v)$  è ben definito  $\forall v \in V$ . Altrimenti i pesi dei cammini minimi non sono ben definiti. E in tal caso  $\delta(s, v) = \infty$

L'algoritmo Bellman-Ford risolve SSSP nel caso generale in cui i pesi possono essere negativi.

### BELLMAN-FORD ( $G, s, w$ )

1. INIZIALIZZAZIONE SSSP
2. FOR  $i = 1$  TO  $|V| - 1$
3. FOR EACH  $(u, v) \in E$
4.     RELAX ( $u, v, w$ )
5. FOR EACH EDGE  $(u, v) \in E$
6.     IF  $v.d > u.d + w(u, v)$
7.         RETURN FALSE
8. RETURN TRUE

CODICE  
COMPLETO  
APPUNTI FARO

### PROPRIETA' DEI CAMMINI MINIMI

1) Disegliente triangolare

$$\forall (u, v) \in E, \delta(s, v) \leq \delta(s, u) + w(u, v)$$

2) Upper-bound

Per tutte le duretta abbiamo  $v.d \geq \delta(s, v)$  e una volta che  $v.d = \delta(s, v)$  non cambierà più

3) Algoritmo

### 3) No-path

Se non ci sono cammini da  $s$  a  $v$  allora

$$\delta(s, v) = \infty$$

### 4) Convergence

Se  $s \rightsquigarrow u \rightarrow v$  è un cammino minimo da  $s$  a  $v$  e se  $u.d = \delta(s, u)$  in qualche istante precedente il rilevamento di  $(u, v)$ , allora  $v.d = \delta(s, v)$  a tutti gli istanti successivi.

### 5) Path-relextion

Se  $\rho = \langle s, v_1, \dots, v_m \rangle$  cammino minimo e gli archi di  $\rho$  vengono rilevati nell'ordine  $(s, v_1), (v_1, v_2), \dots, (v_{m-1}, v_m)$ . Allora  $v_m.d = \delta(s, v_m)$

### 6) Predecessor - Subgraph

Una volta che  $v.d = \delta(s, v) \quad \forall v \in V$  il sottografo dei predecessori  $G_P$  è un albero dei cammini minimi radicato su  $s$ .

TEMPO COMPUTAZIONALE :  $O(|V|^2 + |V| \cdot |E|)$ , quando il grafo è rappresentato da liste di adiacenze

- INIZIALIZZAZIONE :  $\Theta(|V|)$

- $|V| - 1$  peraggi sugli archi nelle linee 2-4 e ognuno di questi impiega  $\Theta(|V| + |E|)$

• il ciclo for delle linee 5-7 impiega  $\Theta(|V|+|E|)$

In realtà scriviamo  $O(|V|^2 + |V| \cdot |E|)$  piuttosto che  $\Theta(|V|^2 + |V| \cdot |E|)$  perché in genere non è necessario fare tutte le  $|V|-1$  iterazioni del primo ciclo

**LEMMA 2:** Sia  $G = (V, E)$  un grafo orientato,  $s \in V$  un vertice sorgente,  $w : E \rightarrow \mathbb{R}$  funzione di peso. Assumendo di non avere cicli di peso negativo. Dopo le iterazioni avremo i cammini minimi de  $s$  verso tutti i vertici  $v$  raggiungibili

Utilizzeremo la PATH-RELAXATION-PROPERTY

Si consideri un vertice  $v$  raggiungibile da  $s$  e un cammino minimo de  $s$  e  $v$ . Alle  $i$ -esime iterazione viene rilasciato  $(v_{i-1}, v_i)$  e questo vale per ogni  $i \in \{1, \dots, k\}$ . Per le proprietà  $v.d = v_k.d = \delta(s, v_k) = \delta(s, v)$

**COROLLARIO:** Avendo un grafo per ogni vertice  $v \in V$  c'è un cammino minimo de  $s$  e  $v$   
 $\iff$

Bellman-Ford termina con  $v.d < \infty$

**THM CORRETTEZZA:** Se  $G$  non contiene cicli di peso negativo che siano raggiungibili da  $s$ , allora l'algoritmo di Bellman-Ford restituisce TRUE,  $v.d = \delta(s, v) \forall v \in V$

il sottografo dei predecessori.

Se  $G$  contiene un ciclo di peso negativo l'algoritmo restituisce FALSE.

**DIM:** Supponiamo che non ci siano cicli di peso negativo. Osserviamo che al termine dell'algoritmo  $v.d = \delta(s, v) \quad \forall v \in V$

Se  $v$  è raggiungibile da  $s$  allora per il LEMMA 2,  $v.d = \delta(s, v)$

Se  $v$  non è raggiungibile da  $s$  allora per la no-path-property,  $v.d = \infty = \delta(s, v)$

Allora se  $v.d = \delta(s, v) \quad \forall v \in V$  abbiamo l'albero dei predecessori secondo le subgraph-pr.

Al termine dell'esecuzione,  $\forall (u, v) \in E$   
 $v.d = \delta(s, v) \leq \delta(s, v) + w(u, v) = u.d + w(u, v)$

Quindi nessun test restituisce TRUE

Ora supponiamo per assurdo che esendo un ciclo di peso negativo e restituiscce nessun TRUE

un ciclo di peso negativo indice  $\exists c = \langle v_0, v_1, \dots, v_k \rangle$   
 $+ v_0$

$$\sum_{i=1}^k w(v_{i-1}, v_i) < 0$$

L'algoritmo restituisce true se  $v_i.d \leq v_{i-1} + w(v_{i-1}, v_i)$

$$\Rightarrow \sum_{i=1}^k v_i \cdot d \leq \sum_{i=1}^k (v_{i-1} \cdot d + w(v_{i-1}, v_i))$$

Siccome  $v_0 = \infty$  ogni vertice del ciclo appare esattamente una volta in ognuna delle somme

$$\sum_{i=1}^k v_i \cdot d = \sum_{i=1}^k v_{i-1} \cdot d$$

allora si avrebbe che:

$$\cancel{\sum_{i=1}^k v_i \cdot d} \leq \cancel{\sum_{i=1}^k v_{i-1} \cdot d} + \sum_{i=1}^k w(v_{i-1}, v_i)$$

Ne segue che:  $\sum_{i=1}^k w(v_{i-1}, v_i) \geq 0$  che è una contraddizione

Quindi l'algoritmo se  $\exists$  un ciclo di peso negativo restituisce TRUE