

10.1 CORRETTEZZA DELL'ALGORITMO TOPOLOGICAL-SORT

Topological-Sort: prende in input un grafo orientato acchilico (DAG) $G = (V, E)$ e restituisce un ordinamento topologico, ovvero un ordinamento lineare \prec dei vertici su V su modo tale che $(u, v) \in E \Rightarrow u \prec v$.

TOPOLOGICAL SORT

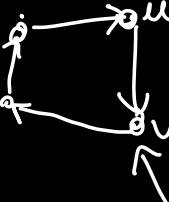
1. Invocare $DFS(G)$ per calcolare $v.f \quad \forall v \in V$;
2. al termine dell'esplorazione di un vertice, inserirlo in una lista collegati;
3. Return: la lista collegati dei vertici.

Tempo computazionale: $\Theta(|V| + |E|)$. Infatti DFS impiega $\Theta(|V| + |E|)$ e l'inserimento di ogni vertice nella lista avviene in $O(1)$.

Lemma 1: G è un DAG \Leftrightarrow una ricerca DF di G non genera archi all'indietro.

Proof: (\Rightarrow): Supponiamo che una ricerca DF produca un arco all'indietro (u, v) , allora v è un predecessore di u nella foresta DF. Quindi, G contiene un circuito

da v a u , e l'arco all'indietro (u, v) complete
il ciclo $\Rightarrow G$ non è un DAG.

(\Leftarrow): Supponiamo che G contiene un ciclo c . Se v è il
primo vertice ad essere scoperto in c . Scegli (u, v)
l'arco precedente in c . Al tempo v.d., i vertici
 di c formano un circuito di vertici bianchi
da v a u . Allora, supposemo del White-Peth-
Theorem (lez. 9) che u è un discendente di v
nella foresta DF. Quindi (u, v) è un arco all'indietro. \square

THM (CORRETEZZA DI TOPOLOGICAL-SORT): Topological-Sort produce
un ordinamento topologico di un DAG in input. $G = (V, E)$

Proof: Dobbiamo provare che $\forall u, v \in V (u \neq v)$ se $(u, v) \in E$
allora $v.f < u.f$.

Consideriamo un qualunque arco (u, v) esplorato dalla
DFS (G). Quando questo arco viene esplorato,

v non può essere grigio, perché allora v avrebbe un predecessore di u e (u,v) avrebbe un arco all'indietro, contraddicendo il Lemma 1. Però, v è bianco o nero.

Se v è bianco, allora è un discendente di $u \Rightarrow v.f < u.f$

Se v è nero allora è già stato esplorato e $v.f$ è già determinato. Poiché DF sta ancora esplorando i cennini che partono da u , allora $u.f$ deve essere ancora assegnato $\Rightarrow v.f < u.f$. \square

10.2 CORRETEZZA DELL'ALGORITMO DI BELLMAN-FORD

SINGOLE-SOURCE SHORTEST PATH PROBLEM

input: un grafo $G = (V, E)$, una funzione di peso $w: E \rightarrow \mathbb{R}$,
 $s \in V$

goal: trovare un cammino minimo da s a ogni altro vertice $v \in V$.

cammino : $P = \langle v_0, v_1, \dots, v_k \rangle$, $(v_{i-1}, v_i) \in E \quad \forall i = 1, \dots, k$

$$\delta(s, v) = \min_{\substack{w(p) \\ p \in P}} w(p) = \sum_{i=1}^k w(v_{i-1}, v_i)$$

OSS: Potrebbero esserci pesi negativi. Se il $G = (V, E)$ non contiene cicli di peso negativo che sono raggiungibili da s , allora $\delta(s, v)$ è ben definito.

Un caso contrario : i pesi dei cammini infiniti non sono ben definiti. In tal caso, non siamo concordi se s è un vertice può essere infatti un perché possono sempre trovare un cammino di peso superiore seguendo il supposto cammino infinito e poi ripetendo il ciclo.

Se esiste un ciclo con peso negativo su un cammino da s a v , allora $\delta(s, v) = \infty$.

Two algorithms of Bellman-Ford solve Single-Source Shortest Path.

BELLMAN-FORD (G, ω, s)

1. INITIALIZE-SINGLE-SOURCE (G, s)
2. for $i = 1$ to $|V| - 1$
3. for each $(u, v) \in E$
4. RELAX (u, v, ω)
5. for each edge $(u, v) \in E$
6. if $v.d > u.d + \omega(u, v)$
7. return FALSE
8. return TRUE

INITIALIZE-SINGLE-SOURCE (G, s)

1. for each $v \in V$
2. $v.d = \infty$
3. $v.\pi = \text{Nil}$

4. $v.d = \infty$

RELAX (u, v, ω)

1. if $v.d > u.d + \omega(u, v)$
2. $v.d = u.d + \omega(u, v)$
3. $v.\pi = u$

10. 2.1. PROPRIETÀ DEI CAMMINI MINIMI E DEI RILASSAMENTI

(Disegualanza triangolare): $\forall (u, v) \in \mathcal{E}, \delta(s, v) \leq \delta(s, u) + \omega(u, v)$

(Upper-bound property): Per tutte le diverse abbassate
 $v.d \geq \delta(s, v) \quad \forall v \in V$ e una volta
che $v.d = \delta(s, v)$ non cambia più.
(lez. 11)

(No-Path Property): Se non ci sono cammini da s a v , allora
 $v.d = \delta(s, v) = \infty$.
(lez. 11)

(Convergence Property): Se $s \sim u \rightarrow v$ è un cammino connesso
in G per qualche $u, v \in V$ e $u.d = \delta(s, u)$
a qualunque stante precedente il predecessore
di (u, v) , allora $v.d = \delta(s, v)$ a tutti
gli stanti successivi. (lez. 11)

(Path-Reflection Property): Se $P = \langle v_0, v_1, \dots, v_k \rangle$ è
gli archi
di P sono alberi nell'ordine
 $(v_0, v_1), (v_1, v_2), \dots, (v_{k-1}, v_k)$ allora
 $v_k.d = \delta(s, v_k)$. (lez. 10)

Questo proprietà vale a prescindere degli
altri passi di riflessione che possono intercorr.

(Predecessor-Subgraph): Una volta che $v.d = \delta(s, v) \forall v \in V$,
allora G_π è un albero dei cammini
connessi radicato a s . (lez. 10, 11)

L'algoritmo di Bellman-Ford impiega $\Theta(|V|^2 + |V||E|)$ -time, quando il grafo è rappresentato dalle due liste di adiacenza.

Un'altra realizzazione: $\Theta(|V|)$; ognuno dei $|V|-1$ passaggi segnali archi alle liste 2-4 impiega $\Theta(|V| + |E|)$ -time (dobbiamo eseguire $|V|$ liste di adiacenza per trovare gli archi); il ciclo for alle liste 5-7 impiega $\Theta(|V| + |E|)$ -time.

In realtà, oh solito sono sufficienti $|V|-1$ passaggi segnali archi. Ecco perché abbiamo scritto $O(|V|^2 + |V||E|)$, piuttosto che $\Theta(|V|^2 + |V||E|)$.

LEMMA 2: Sia $G = (V, E)$ grafo orientato, $v \in V$, $w: E \rightarrow \mathbb{R}$ funzione di peso. Si assume che G non contenga alcun ciclo oh pesi negativi che sia raggiungibile da v .

Allora, dopo $|V|-1$ stereozoom del ciclo for alle
bruse 2ζ , $v.d = \delta(\zeta, v)$ & v raggruppabile ok 3.

Proof: Utilizziamo la PATH-RELAXATION property. Si consideri
un qualunque vertice v che sia raggiungibile da s ,
e se $P = \{v_0, v_1, \dots, v_k\}$ un cammino cicolico da s
a v .

Poiché i commenti anteriori sono semplici, per la al più
 $|V|-1$ archi, perchò $k \leq |V|-1$. Ognuna delle
 $|V|-1$ iterazioni del ciclo for (linee 2-4) mantiene fatti
gli archi in E . Tra gli archi che sono
mossi alle i -esime iterazioni c'è (v_{i-1}, v_i)
 $\forall i \in \{1, \dots, k\}$. Quindi gli archi di E vengono mossi
nell'ordine $(v_0, v_1), (v_1, v_2), \dots, (v_{k-1}, v_k)$ e per le
path-relabeling property

$$v \cdot d = v_k \cdot d = \delta(s, v_k) = \delta(s, v). \quad \square$$

COR 3: Sia $G = (V, E)$ un grafo orientato, $s \in V$, $w: E \rightarrow \mathbb{R}$ una funzione di peso. Allora $\forall v \in V \exists$ un cammino da $s \rightarrow v \Leftrightarrow$ BELLMAN-FORD termina con $v.d < \infty$.

Proof: Giunto dritto. (Dettagli per esercizio).

THI (CORRETTEZZA DELL'ALGORITMO DI BELLMAN-FORD): Sia $G = (V, E)$ un grafo, $s \in V$, $w: E \rightarrow \mathbb{R}$ funzione peso.

Se G non contiene cicli di peso negativo raggiungibili da s , allora BELLMAN-FORD restituisce TRUE, $v.d = \delta(s, v) \quad \forall v \in V$ e G_n è un albero dei cammini minimi radicato a s .

Se G contiene un ciclo negativo raggiungibile da s allora l'algoritmo restituisce FALSE.

Proof: Supponiamo che G non contenga cicli di peso negativo raggiungibili da s .

Per prima cose osserviamo che al termine dell'algo-

altrio, $v.d = d(s, v) \quad \forall v \in V$.

Inoltre, se v è raggiungibile da s , il LEMMA 2 ci dice che $v.d = d(s, v)$. Se invece, v non è raggiungibile da s , allora per la no-path property, $v.d = \infty = d(s, v)$.

Quindi in tal caso (se non ci sono cicli negativi raggiungibili da s) $v.d = d(s, v)$. Per la predecessor-subgraph property, G_T è un albero dei precedenti predecessori non visitati ad s .

Ora osserviamo che, al termine dell'esecuzione, $\forall (u, v) \in E$
 $v.d = d(s, v) \leq d(s, u) + w(u, v) \Leftrightarrow$ $\begin{cases} \text{disegualanza} \\ \text{tra angoli} \end{cases}$
 $\Leftrightarrow u.d + w(u, v)$.

Quindi se non ci sono cicli altri percorsi negativi raggi. da s , nessuno dei test alle linee 6 è soddisfatto e l'algoritmo restituisce TRUE.

Invece, se (per errore)

\exists un ciclo di peso negativo $\Rightarrow \exists C = \langle v_0, v_1, \dots, v_k \rangle : \sum_{i=1}^k \omega(v_{i-1}, v_i) < 0$

\wedge
l'algoritmo restituisce \Leftarrow $v_i.d \leq v_{i-1}.d + \omega(v_{i-1}, v_i) \Rightarrow$
TRUE $\forall i = 1, \dots, k$

$$\sum_{i=1}^k v_i.d \leq \sum_{i=1}^k (v_{i-1}.d + \omega(v_{i-1}, v_i)) \quad \text{F}$$

(Siccome $v_0 = v_k$, ogni vertice del ciclo appare effettivamente una volta in ogni delle scene $\sum_{i=1}^k v_i.d$ e $\sum_{i=2}^k v_{i-1}.d$)

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

$$\underbrace{\sum_{i=1}^k v_i.d}_{\text{sono le } v_i \text{ per il cor. 3}} \leq \underbrace{\sum_{i=1}^k v_{i-1}.d}_{v_i < \infty} + \underbrace{\sum_{i=1}^k \omega(v_{i-1}, v_i)}_{\text{perché } v_i \text{ è lo}}$$

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

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

contradiction!

Quindi l'algoritmo (se \exists ciclo negativo raggiungibile da s) restituisce FALSE. \square