

TEOREMA DEL PERCORSO BIANCO

Dato un grafo G e un $\text{DFS}(G)$, al termine verrà:

True ✓

v è discendente di u all'istante $d(u)$ esiste un percorso π
 \iff nella FDF da u a v fatto da vertici bianchi

la conseguenza di questo teorema e' che la DFS percorre da u a v un percorso di soli nodi bianchi costruendo un albero della FDF

DIMOSTRAZIONE

⇒

Per ipotesi esiste un albero della FDF in cui v discenda da u
Segliamo come percorso per la tesi proprio quello della DFS

Prendiamo un modo w nel percorso tra u e v del DFT.

Questo modo discende da u , quindi vale $d(u) < d(w) < f(u) < f(w)$.

Ha quindi all'istante d(u) u non e' ancora stato scoperto quindi e' bianco. Questo vale per ogni modo tra u e v quindi la propriet e' verificata.

\Leftarrow

Per ipotesi esiste un percorso π tutto bianco al tempo $d(u)$.

Ora non possiamo dimostrare che $\pi \in G$ sia il percorso nel DFT perché non per forza sarà così, potremmo avere un altro percorso nel DFT essendoci più percorsi in G che vanno da u a v .

Per assurdo non tutti i vertici di π diventino discendenti di u

Sia w il primo nodo non discendente. Sia z il nodo precedente appartenente a π che discende ancora da u . Sappiamo che

$$d(u) < d(z) < f(z) < f(w)$$

w dev'essere scoperto dopo $d(u)$ perché un qual momento deve essere "b" e sicuramente dopo $f(u)$ per il teorema delle parentesi.

Se fosse scoperto dopo $f(u)$, lo sarebbe anche dopo $f(z)$ per transitività.

Ma w è adiacente a z , per l'algoritmo il momento $f(z)$ è dopo la scoperta di tutti gli adiacenti, quindi non può essere scoperto dopo $f(u)$. Ma questo è un **ASSURDO** perché non potrebbe essere mai memorizzato

TEMPI DI ESECUZIONE DFS

L'algoritmo esegue una chiamata su `DFS_VISIT` una sola volta per nodo.

Quindi partiamo da $T_{\text{DFS}} = |V|$

Ognuna di queste chiamate su un nodo effettua una chiamata su tutti gli adiacenti che sono tanti quanto gli archi uscenti del nodo.

Se consideriamo l'insieme di tutti gli archi uscenti da tutti i nodi

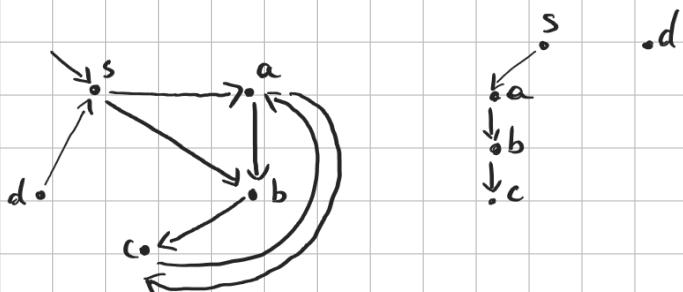
Questo ha proprio cardinalità $|E|$. In particolare dovremmo dire $|E|+1$ perché c'è il controllo di vuotezza dell'array.

Quindi sommando al costo delle chiamate esterne avremo

$$T_{\text{DFS}} = \Theta(|E| + |V|)$$

CLASSIFICAZIONE DEGLI ARCHI

Prendiamo il seguente grafo e costruiamo la FDF



Gli archi di G si possono suddividere in 4 classi:

● ARCHI DELL' ALBERO

Sono gli archi che sono presenti anche nella foresta

● ARCHI DI RITORNO

Sono gli archi che formano i cicli, quelli che se fossero inseriti nella foresta collegherebbero un discendente al suo antenato (ad esempio $c \rightarrow a$)

● ARCHI IN AVANTI

Sono il contrario degli archi di ritorno, collegano un antenato con un discendente nella foresta (ma non sono presenti, ad esempio $a \rightarrow c$)

● ARCHI DI ATTRAVERSAMENTO

Sono archi che collegano nodi che non hanno relazione nella foresta ma sono effettivamente collegati nel grafo (ad esempio $d \rightarrow s$)

Per verificare quale tipo di arco stiamo attraversando dobbiamo effettuare dei controlli sui colori dei nodi d'arrivo:

● ARCHI DELL'ALBERO

Il modo d'arrivo non è stato ancora scoperto quindi dev'essere bianco.

● ARCHI DI RITORNO

Il modo d'arrivo già è stato scoperto ma non visitato, quindi dev'essere grigio

● ARCHI IN AVANTI

Il modo d'arrivo è nero. Se pensiamo alla relazione tra i tempi e la foresta notiamo che (nel caso visto prima) preso $a \rightarrow c$ come arco in avanti, a ha già visitato c quando ha trovato b quindi c è già nero. Bisogna osservare i tempi di scoperta:

c (modo d'arrivo) viene scoperto dopo a, quindi d(partenza) < d(arrivo). Vediamo come questa proprietà sia discriminante

● ARCHI DI ATTRAVERSAMENTO

Anche in questo caso i modi d'arrivo sono neri però la visita del modo d'arrivo è già terminata quando lo troviamo del nostro modo (ad esempio nell'arco $d \rightarrow s$, s ha già completato la sua visita quando d' si collega) e soprattutto già è stato scoperto, quindi d(arrivo) < d(partenza).

Implementando questo controllo nell'algoritmo avremo:

DFS-VISIT (G, s)

COLOR(s) = g

FOR EACH $v \in \text{Adj}(s)$ DO

IF COLOR(v) = b THEN

- E' UN ARCO DELL' ALBERO

DFS-VISIT (G, v)

ELSE IF COLOR(v) = g THEN

- E' UN ARCO DI RITORNO

ELSE

IF $d(v) > d(s)$

- E' UN ARCO D' AVVIAZIONE

ELSE

- E' UN ARCO IN AVANTI

...

PROPRIETA'

Un grafo G è acilico \Leftrightarrow DFS non incontra mai archi di ritorno

DI MOSTRAZIONE

<=

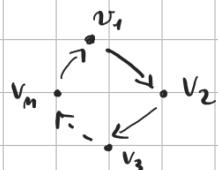
Dimostriamo che se DFS incontra un arco di ritorno allora G contiene cicli. Questo è evidente perché se $u \rightarrow v$ è un arco di ritorno, allora v è antenato di u e quindi c'è un percorso da v a u .

Questo è un ciclo

=>

Se G contiene un ciclo la DFS trova un arco di ritorno.

Immaginiamo il seguente ciclo in G



Supponiamo che DFS scopra v_1 come primo nodo.

All'istante $d(v_1)$ tutti i vertici sono bianchi perché ancora non scoperti, quindi esiste un percorso bianco tra v_1 e v_2 , v_2 e v_3 e così via.

Per il teorema del percorso bianco tutti i nodi discendono da v_1 .

Questo significa che nella foresta avremo un sottobosco di G radicato in v_1 che ha come figli tutti i nodi fino a v_n .

Quando la DFS arriverà a v_n , traverserà proprio v_1 che è grigio

ALGORITMO DI VERIFICA DELLA ACICLICITÀ

L'algoritmo viene implementato nella DFS-VISIT

DFS-VISIT (G, s)

$\text{COLOR}(s) = g$

FOR EACH $v \in \text{Adj}(s)$ DO

IF $\text{COLOR}(v) = b$ THEN

$\text{RET} = \text{DFS-VISIT}(G, v)$

IF $\text{RET} = \text{TRUE}$ THEN

RETURN TRUE

C'è un ciclo

ELSE

IF $\text{COLOR}(v) = g$ THEN

RETURN TRUE

$\text{COLOR}(s) = n$

RETURN FALSE

Non sono stati trovati archi di ritorno

ACICLICO (G)

INT (G)

FOR EACH $v \in V$ DO

IF COLOR (v) = b THEN

RET = DFS-VISIT (G, v)

IF RET = TRUE THEN

Sono state trovate violazioni

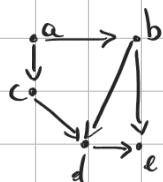
RETURN FALSE

RETURN TRUE

ORDINAMENTO TOPOLOGICO

È una permutazione di vertici (ordinamento) tale che

$\forall (u, v) \in E$ u sta prima di v nella permutazione



a b c d e ✓
a c b d e ✓

Un grafo senza archi ammette tutte le permutazioni come ordinamenti topologici. Viceversa, i grafici ciclici non ammettono alcun ordinamento topologico

PROPRIETA'

- ① Se G e' ACICLICO allora $\exists v \in V \mid v$ non ha archi entranti.
- ② Se G e' ACICLICO allora esiste almeno un ordinamento topologico di G .

DIMOSTRAZIONE

- ① Per assurdo $\forall v \in V$ v ha almeno un arco entrante.

Per rispettare l'ipotesi, inserendo n nodi avremo

$$v_n \rightarrow \dots \rightarrow v_2 \rightarrow v_1$$

Ma v_n deve avere un arco entrante, **assurdo**

- ② Sia v_1 il nodo che non ha archi entranti e poniamolo primo senza violazioni.

Dato il grafo aciclico $G < V, E >$ costruiamo $G' < V', E' >$ con

$$V' = V \setminus \{v_1\} \quad e \quad E' = E \setminus \{(v_1, u) \mid u \in V\}$$

G' e'必然是一个子图，且是无向的

G' ha un nodo che non ha archi entranti, chiamiamolo v_2 .

v_2 potenzialmente aveva come arco entrante in G proprio quello di v_1 .

Costruiamo quindi il seguente ordinamento

$$v_1 \ v_2$$

Itero su G' per ottenere G'' e ricavare v_3 , itero fino al grafo vuoto

Avremo a questo punto costruito una sequenza per la quale ogni nodo ha come possibile arco entrante solo dal precedente nell'ordinamento.

Abbiamo quindi costruito un ordinamento topologico partendo da G .