

BFS SU GRAFO

Implementiamo all' algoritmo di BFS una funzione che esplicita la distanza tra due nodi in cui uno dei due sarà fissato, ovvero il nostro nodo sorgente.

Negliamo inoltre trovare l' insieme dei nodi raggiungibili da s .

Calcoleremo le seguenti funzioni:

- $\delta(s, v) = \text{distanza tra } s \text{ e } v$
- $\{(s, x) \mid (s, x) \in \text{Reach}\}$ dove Reach è l' insieme dei raggiungibili.

Introduciamo una funzione auxiliaria "color" che attribuisce un colore a un nodo che indichi lo stato di visita:

bianco (non incontrato), grigio (incontrato e non visitato), nero (visitato)

Un'altra funzione che useremo è quella che dato un nodo ci dà la distanza del nodo dalla sorgente e chiameremo "dist".

L'ultima funzione che useremo è "pred(v)" che dato un nodo ci indica attraverso quale nodo siamo arrivati ad esso, vedremo dopo meglio a cosa serve questa funzione e de tipo di struttura costruisce

INIT(G)

FOR EACH $v \in V$ DO

COLOR(v) = b

dist(v) = ∞

pred(v) = NIL

Funzione che prepara il grafico

Setta per ogni nodo il colore "b",
la distanza a " ∞ " e il precedente a
NIL (pred restituisce il riferimento a
un nodo)

BFS(G, S)

INIT(G)

DIST(S) = 0

Q = ACCODA(Q, S)

COLOR(S) = g

WHILE Q ≠ NIL DO

X = TESTA(Q)

FOR EACH $y \in \text{Adj}(x)$ DO

IF COLOR(y) = b THEN

Q = ACCODA(Q, y)

DIST(y) = DIST(x) + 1

COLOR(y) = g

PRED(y) = x

La sorgente ovviamente ha distanza 0

Aggiungi S alla coda

Gli oggetti della coda sono sempre
grigi perché sono quelli da visitare

Prendiamo il primo elemento da visitare

Per ogni nodo (y) adiacente ad x

Il nodo non è stato mai incontrato.
(non è in coda)

Dovreemo visitare y dopo (e i suoi adiacenti)

y è adiacente a x

Aggiunto alla coda, è grigio

Siamo arrivati da x a y

$Q = \text{DECODE}(Q)$

Tagliamo il modo visitato (x)

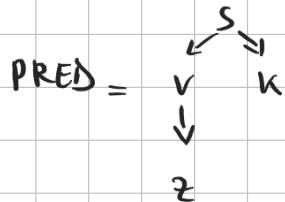
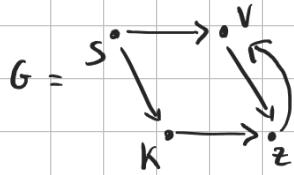
$\text{COLOR}(x) = m$

x è stato visitato, e' meno

$\text{DIST}(y)$ e $\text{COLOR}(x)$ non sono strettamente necessari al funzionamento, per color basterebbero due colori, pero' ci danno informazioni interessanti

BFT

Costruiamo uno schema che preso un grafo mostra colori e distanze



	s	v	z	k
$Q = \{s\}$	s	b	b	b
	0	∞	∞	∞
$Q = \{v, k\}$	m	s	b	s
	0	1	∞	1
$Q = \{k, z\}$	m	m	s	s
	0	1	2	1
$Q = \{z\}$	m	m	s	m
	0	1	2	1
$Q = \emptyset$				

Si noti che da pred deriva un albero, in particolare un sottografo di G , ed è detto **BFT** (breadth first tree), ci mostra i percorsi fissata una sorgente e inoltre risolvono i percorsi minimi

TEMPO DI ESECUZIONE

Considerando la visita dell' albero binario sul numero di nodi, ci aspettiamo qualcosa di simile a

$$T(|G|) = \Theta(|G|) = \Theta(|V| + |E|)$$

Questo è verificato:

- Ogni nodo viene visitato solo una volta (garantito dal colore).
- Il ciclo while viene eseguito al più $|V|$ volte

Il for each costa lineare sugli adiacenti del nodo v , in particolare esegue tante ripetizioni quanti sono gli archi uscenti dal nodo, moltiplicato per tutti i nodi abbiamo il numero di archi totali $|E|$.

$$T(|G|) = \Theta(|E| + |V|)$$

PROPIETÀ INVARIANTI (del ciclo for)

Per dimostrare la correttezza dell'algoritmo ci aiuteremo con alcune proprietà dette **INVARIANTI**, ovvero che valgono in OGNI momento.

Sfrutteremo inoltre una proprietà di δ (distanza) che ci dice che

$$(u, v) \in E \Rightarrow \delta(s, v) \leq \delta(s, u) + 1$$

Le proprietà invarianti sono:

① $\forall v \in V \quad \delta(s, v) \leq \text{dist}[v]$

La distanza da s di v è una stima per eccesso della distanza effettiva (in più al termine vogliamo sia un'egualfiorita).

Dimostrazione

La distanza viene settata subito dopo l'accodamento, quindi ragioniamo per induzione sul numero di accodamenti e vediamo cosa succede.

• 0 accodamenti (caso base, situazione iniziale)

Le distanze sono tutte 0 quindi è banalmente verificato.

• k accodamenti

Sia x un nodo già accodato precedentemente. Per ipotesi induttiva $\delta(s, x) \leq \text{dist}(x)$.

Sia y il $k+1$ -esimo accodamento nonché adiacente di x .

Per la proprietà di δ vale $\delta(s, y) \leq \delta(s, x) + 1$

Possiamo dire: $\delta(s, y) \leq \delta(s, x) + 1 \leq \text{dist}(x) + 1$

Ma per definizione di adiacenti $\text{dist}(x) + 1 = \text{dist}(y)$

Per transitività $\delta(s, y) \leq \text{dist}(sy)$ ✓

② Se $Q = [v_1, \dots, v_k]$ allora

- 1) $\text{dist}(v_i) \leq \text{dist}(v_{i+1}) \quad \forall 1 \leq i < k$
- 2) $\text{dist}(v_k) \leq \text{dist}(v_1) + 1$

DIMOSTRAZIONE

Essendo una proprietà sulla coda Q , dobbiamo sì lavorare per induzione sulle operazioni di coda, ma dobbiamo verificare le proprietà anche dopo decodarla (perché appunto la coda varia in due momenti)

● 1 operazione di coda (caso base)

E' stato eseguito solo un accodamento, abbiamo solo un elemento in coda quindi le due proprietà sono banalmente verificate

● k -esima operazione in coda.

Non sappiamo se quest'operazione e' accoda o decoda quindi analizziamo entrambi i casi

► Se la k -esima operazione e' di accodamento, avremo la coda della $k-1$ -esima operazione (che verifica le proprietà) + 1 nodo aggiunto

Guardando l'algoritmo, dopo l'accodamento l'unico modo per cui cambia dist e l'ultimo, quindi se i precedenti verificavano le proprietà lo continueranno a fare. Se v_{z+1} è il nodo appena inserito, dobbiamo verificare che $\text{dist}(v_z) \leq \text{dist}(v_{z+1})$ (prima proprietà).

► Per l'algoritmo $\text{dist}(v_{z+1}) = \underline{\text{dist}(v_z) + 1}$ (v_z è x nell'algoritmo, v_{z+1} è y). Possiamo applicare l'ipotesi induttiva (la seconda proprietà) su v_z e dire che $\text{dist}(v_z) \leq \underline{\text{dist}(v_1) + 1} \rightarrow \text{dist}(v_z) \leq \text{dist}(v_{z+1})$. Quindi la proprietà 1 è verificata.

► La proprietà 2 sulla nostra coda è verificata di conseguenza perché abbiamo dimostrato che $\text{dist}(v_{z+1}) = \text{dist}(v_z) + 1$

► Se la k -esima operazione è di decodamento, avremo la coda della $k-1$ esima meno v_1 .

► Nell'algoritmo dopo il decodamento non vengono modificate le distanze di alcun vertice, quindi se prima valeva la proprietà 1 continuerà a valere per tutti i vertici.

► Per ipotesi induttiva vale $\underline{\text{dist}(v_1) \leq \text{dist}(v_2)}$ (prima proprietà) e $\underline{\text{dist}(v_z) \leq \text{dist}(v_1) + 1}$ e vogliamo anche dopo il decodamento perché le distanze non vengano modificate. Per transitività avremo

$\text{dist}(v_2) \leq \text{dist}(v_1) + 1 \leq \text{dist}(v_2) + 1$ (la maggiorazione non cambia la relazione). In definitiva $\text{dist}(v_2) \leq \text{dist}(v_1) + 1$ che è proprio la proprietà 2 dopo il decodamento.

OSS

la prima proprietà della recombinazione invariante è quella che ci interessa maggiormente, ciò che ci dice semplicemente è che non è possibile che un nodo inserito in coda abbia distanza (da uno stesso nodo) minore di un vertice inserito precedentemente

PROPRIETÀ DI CORRETTEZZA DELL'ALGORITMO

Suddividiamo i vertici in classi così definite:

$$V_i = \{v \mid \delta(s, v) = i\} \quad \forall i \geq 0$$

Se ai V_i uniamo V_∞ (ugualmente definito) otteniamo una partizione di V .

Per dimostrare la correttezza dell'algoritmo vogliamo dimostrare che:

$\forall i \in \mathbb{N}$, Se $v \in V_i$ allora ad un certo istante vamo'

- $\text{color}(v) = g$
- $v \in Q$
- $\text{pred}(v) \in V_{i-1} \quad \forall i > 0$
- $\text{dist}(v) = \delta(s, v)$

DI MOSTRAZIONE

Anzitutto lavoreremo su V_i perché V_{∞} non viene mai modificato.

(sono i vertici non raggiungibili da s) e non ci interessa.

Ragioniamo per induzione su i .

- Caso base

Per $i=0$ è tutto verificato perché consideriamo solo la radice stessa dopo averla accodata. Il terzo punto non va neanche verificato.

- $i > 0$

Consideriamo un $v \in V_i$.

Per ipotesi induttiva, $V_{j < i}$ verifica tutte le proprietà in un certo istante.

Sappiamo che per definizione di V_i , esiste un percorso (minimo) che va da s a v . In questo percorso ci sarà un predecessore di v che è direttamente collegato a v attraverso un arco, chiamiamo questo modo u . $u \in V_{i-1}$ (perché il percorso da s a u è minimo).

Sappiamo che $\delta(s, u) = i-1 < \delta(s, v)$.

Dobbiamo quindi garantire che u venga inserito in coda prima di v così v potrà essere scoperto ($v \in Q$).

Questo e' garantito perch' $v \in V_{i-1}$, e verificare tutte le proprietà e
in particolare $\text{dist}(u) = i-1$. $\text{dist}(v) \geq \delta(s, v)$ per la prima
proprietà invariante, ma $\delta(s, v) = i$ quindi $\text{dist}(v) \geq i > i-1$.

Così sappiamo che se v entra in coda lo fa con dist corretta.

v entra sicuramente in coda perch' u e' in coda per ipotesi
induttiva, prima o poi sarà testa ed inevitabilmente scoprirà v in quanto
adiacente, e verrà quindi messa in coda ($v \in Q$) con :

$\text{color}(v) = "g"$, $\text{pred}(v) = u \in V_{i-1}$, $\text{dist}(v) = \text{dist}(u) + 1$ (per l'algoritmo) ma
 $\text{dist}(u) + 1 = i = \delta(s, v)$ perch' $v \in V_i$ quindi sono verificate tutte le proprietà.

