

PERCORSO MINIMO TRA DUE NODI

Un percorso minimo da s a v con s sorgente, e' ottenibile concatenando un percorso da s a $\text{pred}(v)$ con l'arco $(\text{pred}(v), v)$.

Immaginiamo di aver già richiamato BFS.

STAMPA PERCORSO MINIMO (s, v, pred)

IF $s = v$ THEN

PRINT s

ELSE

STAMPA PERCORSO MINIMO ($s, \text{pred}(v), \text{pred}$)

PRINT v

Implementando in un algoritmo

PERCORSO MINIMO (G, s, v)

$\text{pred} = \text{BFS}(G, s)$

Immaginiamo che BFS ritorni il BFT

STAMPA PERCORSO MINIMO (s, v, pred)

Il costo di quest' algoritmo e' limitato superiormente dal numero di nodi.

DFS SU GRAFO

L'algoritmo di ricerca in profondità su un grafo si applica per necessita' diverse dall'BFS. Per profondità intendiamo esplorare tutto un percorso per poi tornare agli adiacenti di s e così via (anziché per livelli di distanza).

Per evitare cicli infiniti implementeremo anche qui i colori indicativi dello stato di visita.

La struttura che useremo per salvare i nodi da visitare sarà lo stack di attivazione dell'algoritmo, essendo l'algoritmo ricorsivo.

DFS-VISIT (G, s)

COLOR (s) = 'g'

partiamo colorando ormai il nostro nodo di "g"

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

IF COLOR(u) = 'b'

dobbiamo incontrare l'adiacente

pred(u) = s

continuiamo la struttura pred

DFS-VIST (G, u)

COLOR(s) = 'm'

ho visitato tutti gli adiacenti di s che sicuramente non sono bianchi, e lui è stato visitato

L'inizializzazione dei colori dev'essere fatta all'esterno:

DFS (G)

INIT (G)

FOR EACH $v \in V$ DO

IF COLOR(v) = b THEN

DFS_VISIT (G, v)

Chiamiamo DFS su tutti i nodi che non sono ancora stati visitati

OSS

La differenza principale tra DFS e BFS e' che lo DFS visita tutti i nodi, anche quelli isolati. Inoltre DFS costruisce un insieme di alberi detta FDF (FOREST DEPTH FIRST)

TEMPI DI VISITA

Implementiamo due funzioni che ci danno informazioni aggiuntive sui nodi. In particolare una funzione che tiene conto (costruendo un array) del momento in cui il nodo diventa grigio (TEMPO DI INIZIO VISITA) e una funzione che tiene conto del momento in cui il nodo diventa nero (TEMPO DI FINE VISITA).

DFS-VISIT (G, s)

$T = T + 1$

$d(s) = t$

COLOR (s) = 'g'

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

IF COLOR (u) = 'b'

$\text{pred}(u) = s$

DFS-VISIT (G, u)

COLOR (s) = 'm'

$T = T + 1$

$f(s) = T$

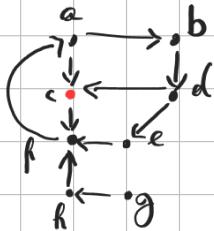
incrementiamo di un istante

Inseriamo il tempo di inizio visita

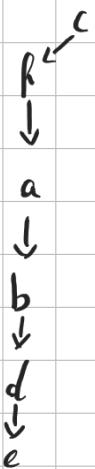
OSS

L'ordine di visita tra DFS e BFS è diverso, e anche il tipo di rappresentazione cambia il risultato finale

Eso.



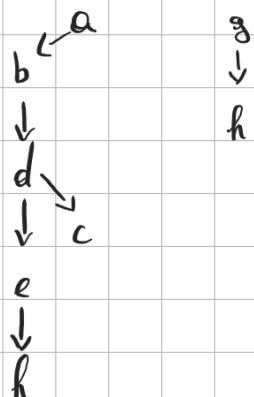
Prendiamo c come sorgente



	a	b	c	d	e	f	g	h
d	3	4	1	5	6	2	13	14
f	10	3	12	8	7	11	15	16

A seconda della sorgente cambierà la FDF e lo schema dei tempi.

Proviamo a partire da a



Il DFT **NON** ci dà informazioni sui percorsi minimi (ad esempio c'è raggiungibile con due archi da a, ma il DFT ci dice che il percorso è lungo 3).

N.B.

Se dobbiamo calcolare le distanze o percorsi minimi possiamo usare **Solo** la BFS

TEOREMA DELLA STRUTTURA A PARENTESI

Ad ogni vertice associamo un tempo di inizio e fine visita che sono le nostre parentesi.

① Al termine della $\text{DFS}(G)$, $\forall u, v \in V$ valgono una delle seguenti proprietà:

- $d(v) < f(u) < d(u) < f(v)$ (vale il duale con v)

- $d(v) < d(u) < f(u) < f(v)$ //

Se consideriamo le parentesi antitetiche, il primo caso è $()[]$, il secondo caso è $[()]$

NON valgono $d(v) < d(u) < f(v) < f(u)$ oppure $d(u) < d(v) < f(v) < f(u)$

② Se torniamo sul secondo caso possiamo affermare che:

$$d(v) < d(u) < f(u) < f(v) \iff u \text{ è discendente di } v \text{ nella foresta}$$

DIMOSTRAZIONE ①

Dimostriamo che per ogni caso mai possiamo mai ottenere i casi da scartare. Procediamo per assurdo, supponiamo vero $d(v) < d(u) < f(v) < f(u)$.

Se inseriamo $d(v)$ significa che nello stack di attivazione abbiamo proprio v . Successivamente inseriamo u nello stack.

Per chiudere v significa che nello stack abbiamo v in cima (ogni chiamata modifica solo d e f dei nodi in cima allo stack, ovvero le variabili d'input). Ipoteticamente ci sono due cose per cui v è in cima allo stack dopo $d(u)$:

- 1) v è stato reinserito nello stack sopra u (che non è stato rimosso)
- 2) sono state chiuse tutte le chiamate successive (che quindi erano sopra nello stack) e v quindi ora è di nuovo in cima

Non si verifica nessuna delle due in realtà perché:

- 1) Non possiamo visitare v due volte grazie ai coloni
- 2) Se u è stata chiusa, l'algoritmo successivamente chiude v il che va contro la nostra ipotesi.

Questo genera un **ASSURDO**.

Vale analogamente per l'espressione duale

DIMOSTRAZIONE ②

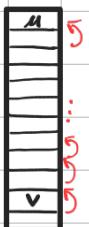
=>

È facile vedere che u discende da v perché supponendo $d(v) < d(u)$

Sappiamo che u è in cima e v è sotto, ci saranno potenzialmente

altri nodi in mezzo. Due nodi sono consecutivi nello stack se sono adiacenti nel grafo (perché la chiamata ricorsiva, con conseguente inserimento nello stack, avviene sugli adiacenti).

Se sono adiacenti esiste un arco che li collega.



Questo è proprio il percorso di visita della DFS, quindi u è raggiungibile da v , e sarà proprio questo il percorso inserito nella foresta perché l'algoritmo assegna il predecessore proprio prima di effettuare la chiamata ricorsiva

\Leftarrow

Dimostriamo per induzione sulla lunghezza dei percorsi tra v e u



- Caso base (lunghezza 1)

essendo ci un arco tra v e u nella foresta, sono l'uno il predecessore dell'altro. In particolare $\text{pred}(u)=v$ viene assegnato

nella chiamata di v prima di richiamarsi su u .

In questo momento u è bianco e v è grigio.

Vale quindi sicuramente $d(v) < d(u)$. Sappiamo dall'algoritmo che $f(v)$ verrà assegnato come ultimo comando quindi $f(u)$ sarà per forza prima. $d(v) < d(u) < f(u) < f(v)$

• $|T|=k>1$

Prendiamo z che forma un percorso di lunghezza $k-1$ da v

Per ipotesi induttiva sappiamo che $d(v) < d(z) < f(z) < f(v)$

Inoltre sappiamo che $(z, u) \in E$ e quindi $\text{pred}(u)=z$.

Questo assegnamento viene effettuato durante la chiamata su z .

ma quindi $d(u)$ e $f(u)$ sono compresi tra $d(z)$ e $f(z)$.

Per transitività avremo

$$d(v) < d(z) < d(u) < f(u) < f(z) < f(v) \Rightarrow d(v) < d(u) < f(u) < f(v) \quad \checkmark$$