

RAPPRESENTAZIONE GRAFI NORMALI

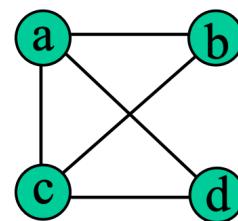
POSSIAMO RAPPRESENTARLI CON:

- **MATRICE DI ADIACENZA**

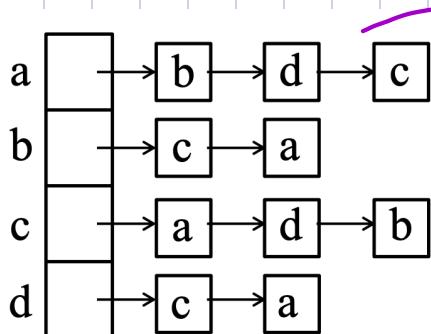
	a	b	c	d
a	0	1	1	1
b	1	0	1	0
c	1	1	0	1
d	1	0	1	0

1 SE C'È UN ARCO CHE COLLEGA
I DUE NODI, 0 ALTRIMENTI

COMPLESSITÀ SPAZIALE: $O(m^2)$



- **LISTE DI ADIACENZA**



LISTA DI ADIACENZA DI A (ARCHI INCIDENTI AD A)

COMPLESSITÀ SPAZIALE: $O(2m + m)$

OGNI ARCO È INSERITO 2 VOLTE
UNA VOLTA X U E UNA X V

$O(m + m)$ → LINEARE NELLA DIM. DEL GRAFO

ESEMPIO

DATO UN NODO V VOGLIO SAPERE TUTTI GLI ARCHI INCIDENTI

↳ MATRICE: ACCEDO A V IN $O(1)$ E IERO LA RIGA $\rightarrow O(m)$

↳ LISTA: ACCEDO A V IN $O(1)$ E IERO LA LISTA $\rightarrow O(\delta(v))$

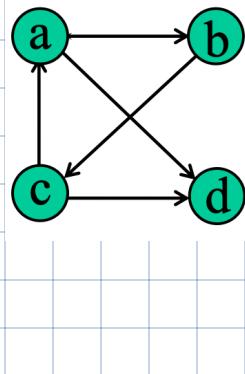
GRADO

VOGLIO SAPERE SE C'È UN ARCO (u, v) :

↳ MATRICE: ACCEDO ALLA ENTRY $[u][v]$ E CONTROLO $\rightarrow O(1)$

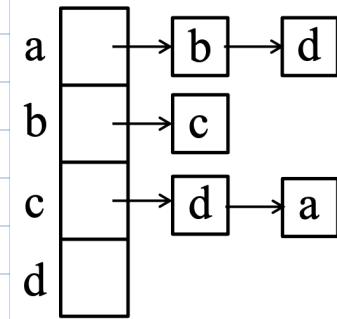
↳ LISTA: ACCEDO ALLA ENTRY DI u, v , SCELGO QUELLO CON GRADO MINORE, COMPLESSITÀ $\rightarrow O(\min\{\delta(u), \delta(v)\})$

RAPPRESENTAZIONE GRAFI ORIENTATI



	a	b	c	d
a	0	1	0	1
b	0	0	1	0
c	1	0	0	1
d	0	0	0	0

ENTRY IMPOSTATA A 1 SE C'È
L'ARCO USCENTE (u,v)



INSEGUISCO NELLA LISTA DI UN
NODO TUTTI I NODI V RAGGIUNGIBILI
DA U TRAMITE UN ARCO USCENTE

STESSI COSTI



ESEMPIO

DATO UN NODO V VOGLIO SAPERE TUTTI GLI ARCHI USCENTI

↳ MATERICE : ACCEDO A V IN $O(1)$ E ITERO LA RIGA $\rightarrow O(m)$

↳ LISTA : ACCEDO A V IN $O(1)$ E ITERO LA LISTA $\rightarrow O(\delta(v))$

GRADO

VOGLIO SAPERE SE C'È UN NODO (u,v) :

↳ MATERICE : ACCEDO ALLA ENTRY $[u][v]$ E CONTROLLO $\rightarrow O(1)$

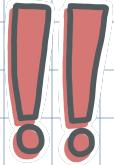
↳ LISTA : ACCEDO ALLA ENTRY DI U E ITERO $\rightarrow O(\delta(u))$

SCOPI E TIPI DI VISITA

- ESAMINIAMO TUTTI I NODI IN MODO SISTEMATICO
- GENERA UN ALBERO DI VISITA
- CI SONO VARI TIPI DI VISITA
 - ↳ BFS (VISITA IN AMPIEZZA)
 - ↳ DFS (VISITA IN PROFONDITÀ)

VISITA IN AMPIEZZA BFS

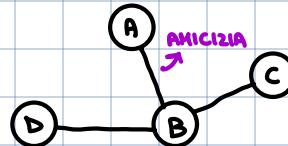
DATO UN GRAFO E UN NODO S (SORGENTE), TROVA TUTTE LE DISTANZE / CAMMINI MINIMI VERSO OGNI ALTRO NODO V.



APPLICAZIONI:

- web crawling
 - come google trova nuove pagine da indicizzare
- social networking
 - trovare gli amici che potresti conoscere
- network broadcast
 - un nodo manda un messaggio a tutti gli altri nodi della rete
- garbage collection
 - come scoprire memoria non più raggiungibile che si può liberare
- model checking
 - verificare una proprietà di un sistema
- risolvere puzzle
 - risolvere il Cubo di Rubik con un numero minimo di mosse

ESEMPIO: SUGGERIMENTO AMICIZIA FACEBOOK



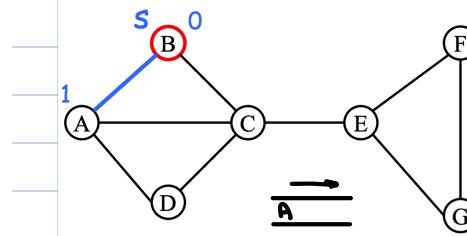
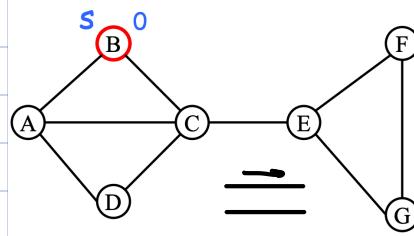
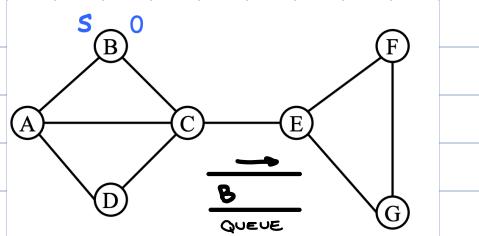
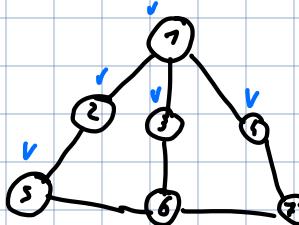
- AMICI DI A = NODI A DISTANZA 1
- AMICI DI AMICI DI A = NODI A DISTANZA 2

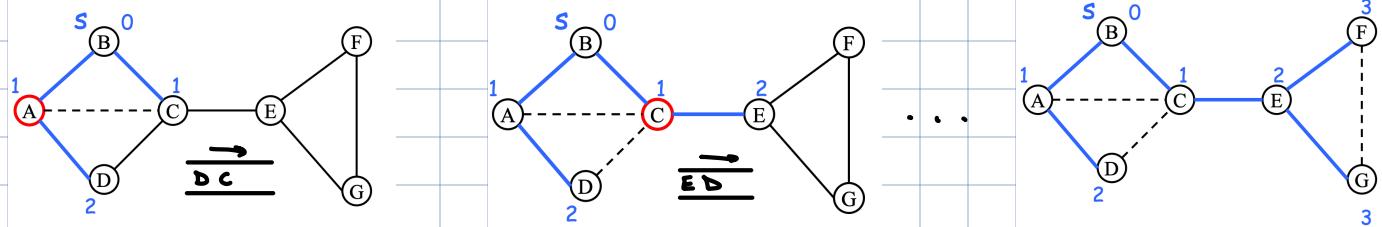
SUGGERISCO

PSEUDOCODICE:

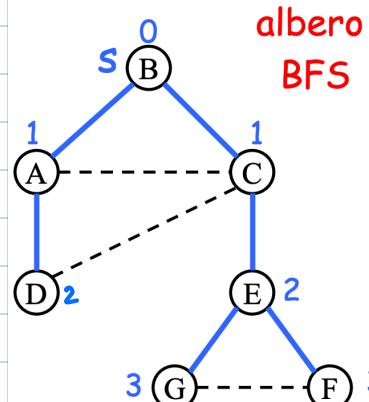
algoritmo visitaBFS(vertice s) → albero

1. rendi tutti i vertici non marcati
2. $T \leftarrow$ albero formato da un solo nodo s
3. Coda F
4. marca il vertice s ; $\text{dist}(s) \leftarrow 0$
5. $F.\text{enqueue}(s)$
6. **while (not F.isempty()) do**
7. $u \leftarrow F.\text{dequeue}()$ ESTRAE DALLA CODA IL NODO DA VISITARE
8. **for each (arco (u, v) in G) do** PER OGNI ARCO INCIDENTE AD U
9. **if (v non è ancora marcato) then** SE IL NODO VICINO NON È STATO VISITATO
10. $F.\text{enqueue}(v)$ METTO IN CODA IL NODO VICINO
11. marca il vertice v ; $\text{dist}(v) \leftarrow \text{dist}(u)+1$ MARCO IL VICINO COME LA DISTANZA DAL NODO U IN CUI SONO + 1 CHE SEMBRA DA U a V
12. rendi u padre di v in T IL VICINO SARÀ FIGLIO DI U NELL'ALBERO
13. **return T**





ALBERO FINALE



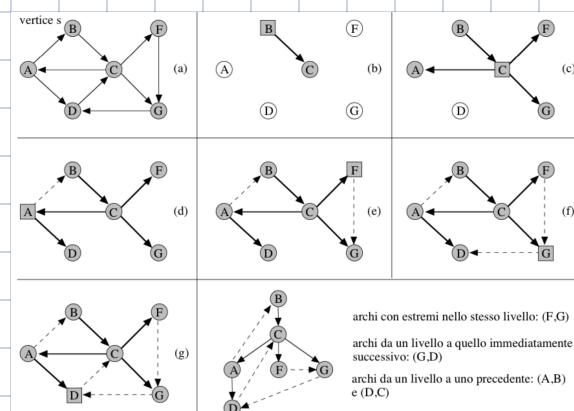
OGNI VOLTA CHE DA UN NODO U INCONTRA UN VICINO V, MARCO U CON LA DISTANZA DI U + 1.

U È IL NODO ESTRATTO AD OGNI ITERAZIONE.

AD ESEMPIO LA RADICE ALLA PRIMA ITER. SARÀ MARCATA COME U CON DIST = 0, I NODI VICINI SARANNO MARCATI CON DIST(U) + 1, METTIAMO CASO CHE IL VICINO Z SIA MARCATO CON DISTANZA U + 1 = 0 + 1 = QUINDI 1, ALLA PROSSIMA ITERAZIONE QUANDO Z VERRÀ ESTRATTO DALLA CODA DIVENTERÀ IL NUOVO U, SCOPRENDO I VICINI QUESTI VERRANNO MARCATI CON LA DIST(U) + 1, CHE SAREBBE LA DIST. DEL VECCHIO Z + 1.

BFS SU GRAFI ORIENTATI

POSSIAMO EFFETTUARE LA BFS ANCHE SU GRAFI ORIENTATI MA QUANDO ANDIAMO A VEDERE I NODI ADIACENTI DOBBIANO SOLAMENTE VEDERE GLI ARCHI USCENTI.



ANALISI DELLA COMPLESSITÀ USANDO MATRICI DI ADIACENZA



```

algoritmo visitaBFS(vertice s) → albero
1. rendi tutti i vertici non marcati
2.  $T \leftarrow$  albero formato da un solo nodo s
3. Coda F
4. marca il vertice s;  $\text{dist}(s) \leftarrow 0$ 
5. F.enqueue(s)
6. while (not F.isEmpty()) do
7.    $u \leftarrow F.\text{dequeue}()$ 
8.   for each (arco  $(u, v)$  in G) do
9.     if (v non è ancora marcato) then
10.      F.enqueue(v)
11.      marca il vertice v;  $\text{dist}(v) \leftarrow \text{dist}(u)+1$ 
12.      rendi u padre di v in T
13. return T

```

OGNI NODO VIENE INSERITO/ESTRATTO UNA VOLTA:

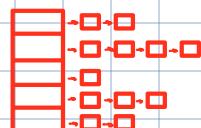
 $O(m)$

PER OGNI NODO CONTROLLO I VICINI, QUINDI GLI ARCHI INCIDENTI, USANDO LE MATERICI DI ADIACENZA IL COSTO È:

 $O(m)$

COSTO COMPLESSIVO: $O(m^2)$

ANALISI DELLA COMPLESSITÀ USANDO LISTE DI ADIACENZA



```

algoritmo visitaBFS(vertice s) → albero
1. rendi tutti i vertici non marcati
2.  $T \leftarrow$  albero formato da un solo nodo s
3. Coda F
4. marca il vertice s;  $\text{dist}(s) \leftarrow 0$ 
5. F.enqueue(s)
6. while (not F.isEmpty()) do
7.    $u \leftarrow F.\text{dequeue}()$ 
8.   for each (arco  $(u, v)$  in G) do
9.     if (v non è ancora marcato) then
10.      F.enqueue(v)
11.      marca il vertice v;  $\text{dist}(v) \leftarrow \text{dist}(u)+1$ 
12.      rendi u padre di v in T
13. return T

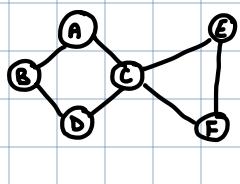
```

OGNI NODO VIENE ESAMINATO UNA VOLTA. QUANDO VISITIAMO OGNI NODO PERCORRIAMO LA SUA LISTA DI ADIACENZA. $\rightarrow O(|\text{S}(u)|)$

LA SOMMA DI TUTTE LE LUNGHEZZE DELLE LISTE DI ADIACENZA È $|E|$ SE G È ORIENTATO, $2 \cdot |E|$ SE NON È ORIENTATO.

PERCHÉ NON È $O(|V| \cdot |E|)$? NELLA BFS NON STIAMO CONTROLLANDO TUTTI GLI ARCHI PER OGNI NODO MA SOLO GLI ARCHI INCIDENTI.

QUANDO SIAVO SU UN NODO u E "SCOPRIAVO" I SUOI VICINI QUESTI VENGONO MARCATI PER NON ESSERE VISTI PIÙ VOLTE. QUINDI PER OGNI NODO AGGIUNGIAMO I VICINI **NON MARCATI**, QUESTO EQUIVALE AL NUMERO COMPLESSIVO DI ARCHI cioè m .



$$\begin{aligned} A &\rightarrow (A,B), (A,C) \\ B &\rightarrow (B,D) \\ C &\rightarrow (C,E), (C,F) \\ E &\rightarrow (E,F) \end{aligned}$$

PER OGNI NODO VISITIAMO GLI ARCHI INCIDENTI E SE CI FACCIAMO CASO È PROPRIO $m = |E|$

AVENDO QUINDI UNA COMPLESSITÀ TOTALE DI: $O(m + m)$

OSSERVAZIONI:

IL NUMERO MINIMO DI ARCHI IN UN GRAFO È $m-1$, OVVERO IN UN ALBERO.

- SE G È CONNESSO ALLORA $m \geq m-1 \rightarrow O(m)$ MINIMO * DI ARCHI
- SE G È COMPLETO ALLORA $m \leq \frac{m(m-1)}{2} \rightarrow O(m^2)$ MASSIMO * DI ARCHI

VISITA IN PROFONDITÀ (DFS)

GARANTISCE DI VISITARE TUTTI I NODI E GLI ARCHI DI UN GRAFO CONNESSO, FORNENDO UNA CONOSCENZA COMPLETA DELLA STRUTTURA.
SEGUE UN SENTIERO FINO IN FONDO PRIMA DI VISITARNE ALTRI.



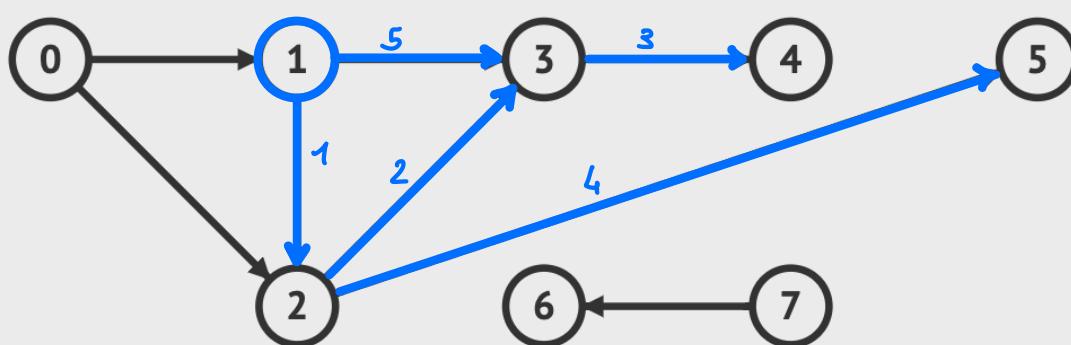
procedura visitaDFSRicorsiva(vertice v , albero T)

1. marca e visita il vertice v → MARCA IL VERTICE VISITATO
2. for each (arco (v, w)) do → PER OGNI ARCO INCIDENTE, QUINDI PER OGNI VICINO
3. if (w non è marcato) then → AGGIUNGI L'ARCO ALL'ALBERO
4. aggiungi l'arco (v, w) all'albero T → RICHIAMA LA DFS SUL VICINO
5. visitaDFSRicorsiva(w, T) → RICHIAMA LA DFS SUL VICINO

algoritmo visitaDFS(vertice s) → albero

6. $T \leftarrow$ albero vuoto
7. visitaDFSRicorsiva(s, T)
8. return T

INIZIAMO LA DFS DA UN NODO u , LO MARCHIAMO E PROCEDIAMO AD ITERARE I SUOI VICINI, O MEGLIO ARCHI INCIDENTI, PER OGNI VICINO RICHIAMIAMO LA DFS.



① DFS SUL NODO 1, CHIAMA DFS SUL VICINO 2, RIMANGONO I VICINI [3]

② DFS SUL NODO 2, CHIAMA DFS SUL VICINO 3, RIMANGONO I VICINI [5]

③ DFS SUL NODO 3, CHIAMA DFS SUL VICINO 4, RIMANGONO I VICINI [7]

DFS SUL NODO 3, CHIAMO DFS SUL VICINO 4, RIMANGONO I VICINI []

- DFS SUL NODO 4, TERMINA E Torna indietro alla chiamata fatta dal nodo 3
- SONO TORNATO INDIETRO SUL NODO 3, NON CI SONO ALTRI VICINI, Torno al 2
- ④ SONO TORNATO INDIETRO SUL NODO 2, CHIAMO DFS SUL NODO 5
- DFS SUL NODO 5, TERMINA E Torna indietro alla chiamata fatta dal nodo 2
- SONO TORNATO INDIETRO SUL NODO 2, NON CI SONO ALTRI VICINI, Torno al 1
- SONO TORNATO AL 1, ERA rimasto il vicino 3 ha été stato marcato, termino.

ANALISI COMPLESSITÀ

USANDO LE LISTE DI ADIACENZA



$$O(m + m)$$

USANDO LA MATRICE DI ADIACENZA

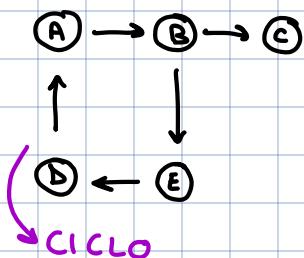
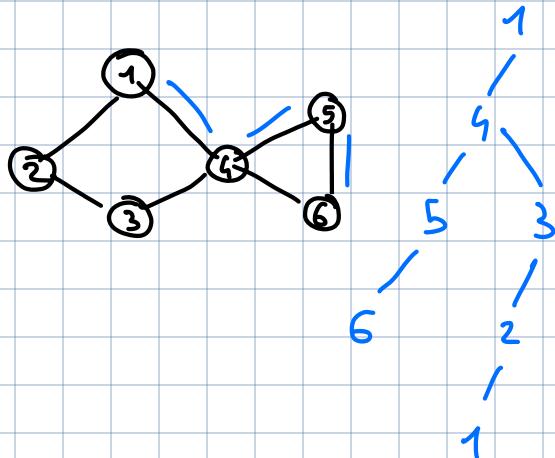


$$O(m^2)$$

PROPRIETÀ DELL' ALBERO DFS

SE IL GRAFO È **NON ORIENTATO**, PER OGNI ARCO (u, v) NEL GRAFO:

- (u, v) È UN ARCO DELL' ALBERO DFS (n. 1, 4)
- OPPURE
- $u = v$ SONO ANTESTITO/DISCENDENTE DELL' ALTRO NELL' ALBERO DFS (n. 1, 2)



SE IL GRAFO È **ORIENTATO**, PER OGNI ARCO (u, v) SI HA:

- (u, v) È UN ARCO NELL' ALBERO DFS

OPPURE

- U E V SONO ANTENUATO/DISCENDENTE DELL'ALTRO NELL'ALBERO DFS
- OPPURE
- V È STATO ESPLORATO PRIMA DI U, V APPARTIENE AD UN SOTTALBERO

GRAFI

→ $G = (|V| + |E|)$ POSSONO ESSERE RAPPRESENTATI:

- MATRICI $O(m^2)$
- LISTE DI ADIACENZA $O(m+m)$

BFS

→ VISITA IN AMPIEZZA CON COSTO

- MATRICI $O(m^2)$
- LISTE DI ADIACENZA $O(m+m)$
- GRAFO CONNESSO $O(m)$

DFS

→ VISITA IN PROFONDITÀ CON COSTO

- MATRICI $O(m^2)$
- LISTE DI ADIACENZA $O(m+m)$