

Algoritmi e Strutture Dati

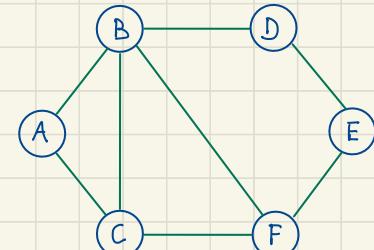
Lezione 21

16 novembre 2022

GRAFO $G = (V, E)$ / V VERTICI / NODI

$E \subseteq V \times V$ ARCHI / LATI / SPIGOLI

GRAFI
NON ORIENTATI: E SIMMETRICI

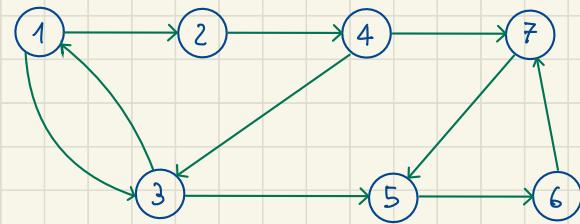


ORIENTATI / DIRETTI

CAMMINO, CICLO, CATENA, CIRCUITO

GRAFO CONNESSO, GRAFO FORTEMENTE CONNESSO

COMPONENTE FORTEMENTE CONNESSA, ALBERO, FORESTA, ERICCA ...



Rappresentazione di grafi

LISTA DI ARCHI

struttura /
vertici
archi:

$$G = (V, E)$$

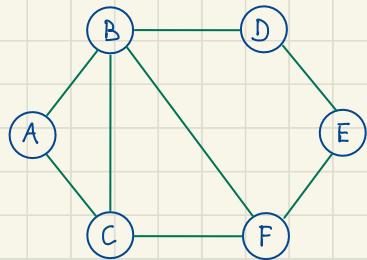
$$n = \# V \quad m = \# E$$

Spazio $O(n + m)$

$$0 \leq m \leq n^2$$

grafo connesso $m \geq n - 1$

LISTA DI ARCHI



(A, B)

(A, C)

(B, C)

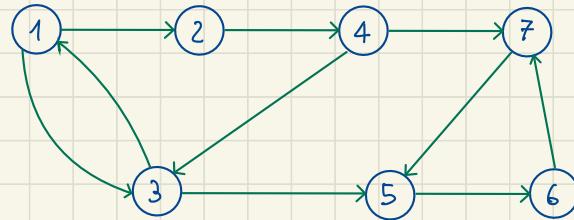
(B, D)

(B, F)

(C, F)

(D, E)

(E, F)



(1, 2)

(1, 3)

(2, 4)

(3, 1)

(3, 5)

(4, 3)

(4, 7)

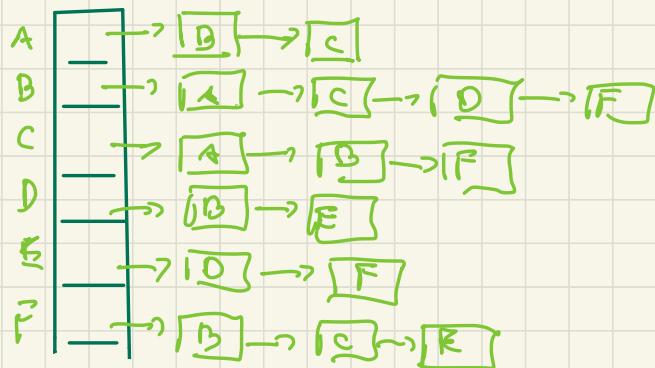
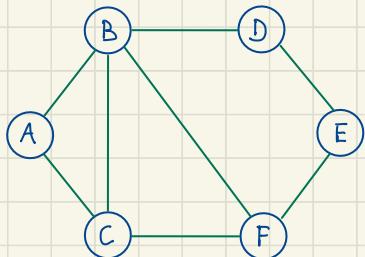
(5, 6)

(6, 7)

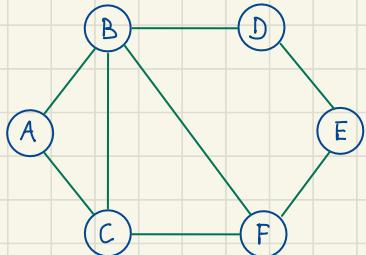
(7, 5)

LISTA DI ADIACENZA

struttura principale → vertici
per ogni vertice ↔ lista vertici adiacenti

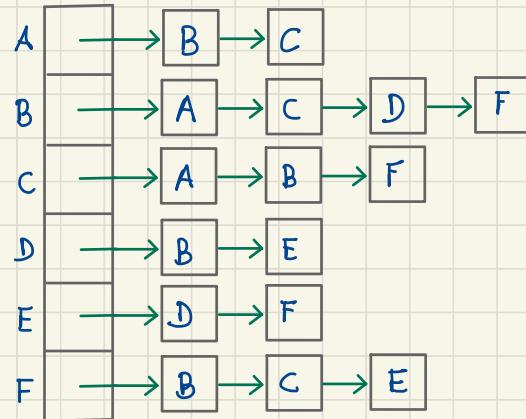


LISTA DI ADIACENZA



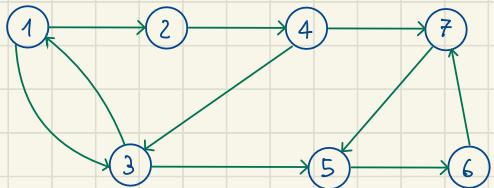
2 elementi per ogni arco

Σ lunghezze di弓p = 2m

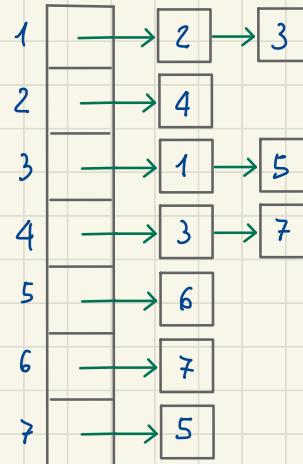


Spazio $\Theta(n+m)$

LISTA DI ADIACENZA



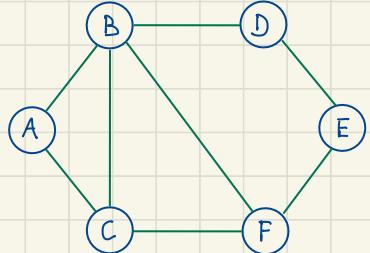
spazio $\Theta(n+m)$



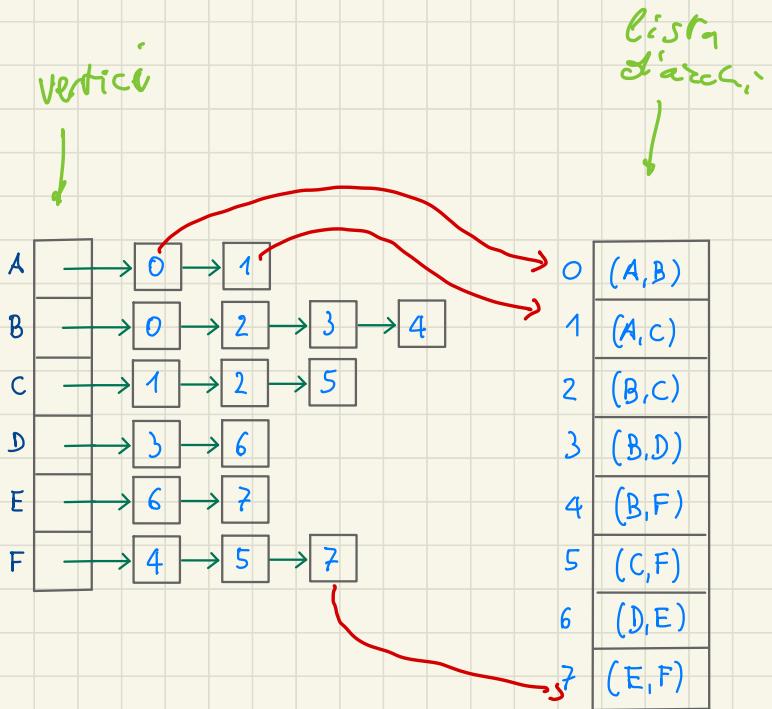
1 elemento per ogni arco

$$\sum \text{lunghezza liste} = m$$

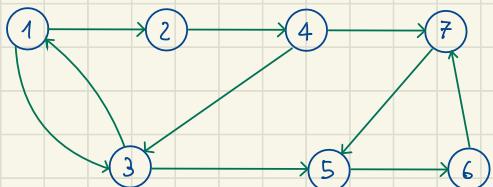
LISTA DI INCIDENZA



$$\text{Spec} \circ \Theta(n+m)$$



LISTA DE INCIDENZA



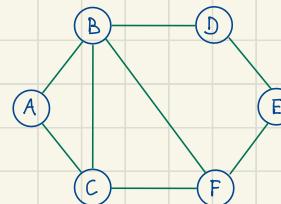
0	(1,2)
1	(1,3)
2	(2,4)
3	(3,1)
4	(3,5)
5	(4,3)
6	(4,7)
7	(5,6)
8	(6,7)
9	(7,5)

MATRICE DI ADIACENZA

Matrice quadrata non di 0/1

indici \leftrightarrow vertici

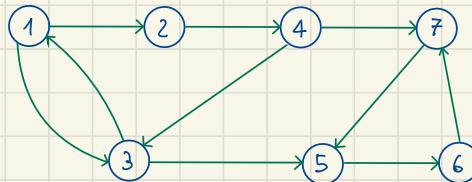
$M[u,v] = 1$ sse \exists arco da u a v



$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

	A	B	C	D	E	F
A	0	1	1	0	0	0
B	1	0	1	1	0	1
C	1	1	0	0	0	1
D	0	1	0	0	1	0
E	0	0	0	1	0	1
F	0	1	1	0	1	0

MATRICE DI ADIACENZA



Spazio $\Theta(n^2)$

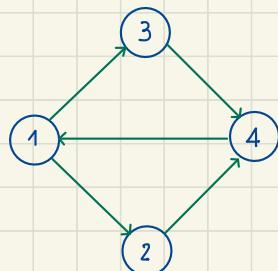
Se $m = \mathcal{O}(n)$ \rightarrow grafo sparso

Riga i →
archi
uscenti
da i

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Colonna j :
archi entranti in j

MATRICE DI ADIACENZA



$$M = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

\uparrow

$$M^2 = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

belen

$M^k[i, j] = 1$ se c'è cammino di lunghezza k da i a j

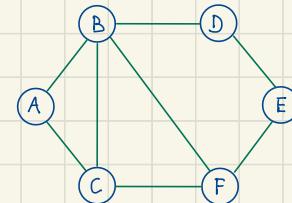
MATRICE DI INCIDENZA

grafi non orientati 0/1

spazio $\Theta(n \cdot m)$

vertici ↴ arco ↴

	(A,B)	(A,C)	(B,C)	(B,D)	(B,F)	(C,F)	(D,E)	(E,F)
A	1	1	0	0	0	0	0	0
B	1	0	1	1	1	0	0	0
C	0	1	1	0	0	1	0	0
D	0	0	0	1	0	0	1	0
E	0	0	0	0	0	0	1	1
F	0	0	0	0	1	1	0	1



MATRICE DI INCIDENZA

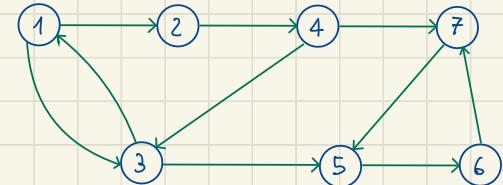
grafi orientari

$$-1, 0, 1$$

-1, 0, 1 ← 1 a user

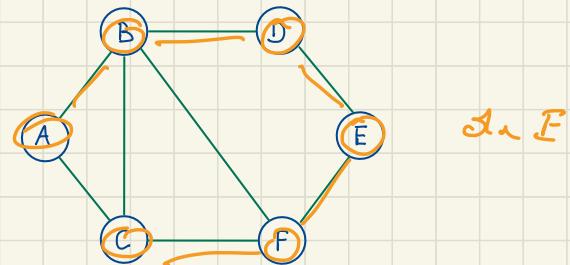
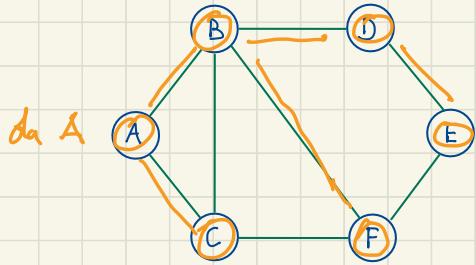
-1 arco invertido
1 arco invertido

$\text{Span}_{\mathbb{R}^n}$ $\Theta(n \cdot m)$



Attraversamento di grafi

VISITA IN AMPIEZZA (Breadth first search)



- Si inizia visitando un vertice s
- Si visitano i vertici adiacenti a s
- Si visitano i vertici adiacenti ai vertici adiacenti a s ,
non ancora visitati

...

ALGORITMO visitaInAmpicetta (grafo $G = (V, E)$, vertice s) \rightarrow Albero

$C \leftarrow$ coda vuota

$T \leftarrow$ albero formato solo da s

marca s come raggiunto

$C.\text{enqueue}(s)$

WHILE NOT $G.\text{isEmpty}()$ DO

$u \leftarrow C.\text{dequeue}()$

FOR EACH $(u, v) \in E$ DO

IF v non è marcato come raggiunto THEN

aggiungi v e (u, v) a T

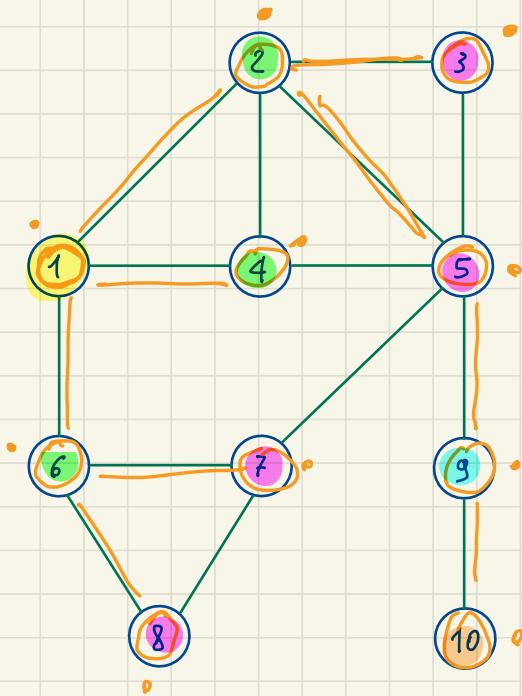
marca v come raggiunto

$C.\text{enqueue}(v)$

RETURN T

ESEMPIO

da 1



C
x 2 4 6 7 8 7 8 9 10

ALGORITMO $\text{visitaInAmpestra}(\text{grafo } G=(V,E), \text{ vertice } s) \rightarrow \text{Albero}$

$C \leftarrow \text{coda vuota}$

$T \leftarrow \text{albero formato solo da } s$

marca s come raggiunto

$C.\text{enqueue}(s)$

WHILE NOT $C.\text{isEmpty}()$ DO

$m \leftarrow C.\text{dequeue}()$

FOR EACH $(u,v) \in E$ DO

IF v non è marcato come raggiunto THEN

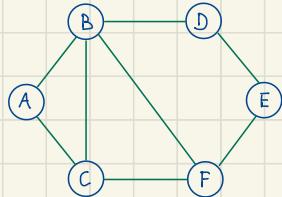
aggiungi v e (u,v) a T

marca v come raggiunto

$C.\text{enqueue}(v)$

RETURN T

VISITA IN AMPIEZZA: tempo



LISTA DI ARCHI

(A,B)

(A,C)

(B,C)

(B,D)

(B,F)

(C,F)

(D,E)

(E,F)

Tempo $O(m \cdot n)$

n
iterazion.

ALGORITMO visitaInAmpiezza (grafo $G = (V,E)$, vertice s) \rightarrow Albero

$C \leftarrow$ coda vuota

$T \leftarrow$ albero formato solo da s

marca s come raggiunto

$C.\text{enqueue}(s)$

WHILE NOT $C.\text{isEmpty}()$ DO

$u \leftarrow C.\text{dequeue}()$

FOR EACH $(u,v) \in E$ DO

IF v non è marcato come raggiunto THEN

aggiungi v e (u,v) a T

marca v come raggiunto

$C.\text{enqueue}(v)$

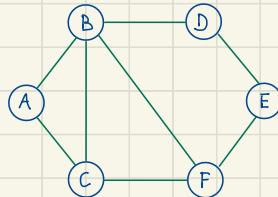
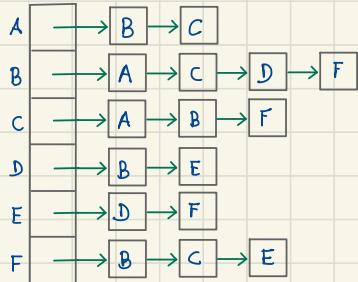
RETURN T

2° tipo
for + la
coda

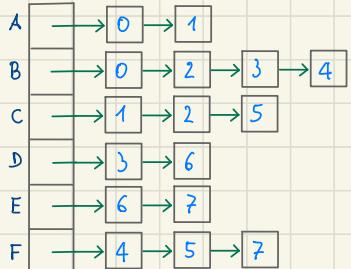
m passi:
per ogni
iterazione
del cerchio

VISITA IN AMPIEZZA: tempo

LISTA DI ADIACENZA



LISTA DI INCIDENZA



0	(A,B)
1	(A,C)
2	(B,C)
3	(B,D)
4	(B,F)
5	(C,F)
6	(D,E)
7	(E,F)

Tempo $O(n+m)$

ALGORITMO visitaInAmpiezza (grafo $G=(V,E)$, vertice s) \rightarrow Albero

$C \leftarrow$ coda vuota

$T \leftarrow$ albero formato solo da s

mark s come raggiunto

$C.enqueue(s)$

WHILE NOT $C.isEmpty()$ DO

$u \leftarrow C.dequeue()$

FOR EACH $(u,v) \in E$ DO

IF v non è marcato come raggiunto THEN

aggiungi v e (u,v) a T

mark v come raggiunto

$C.enqueue(v)$

RETURN T

n
iteration

fissato m :

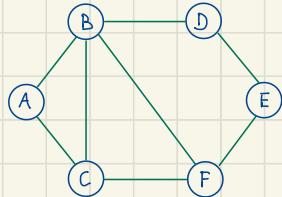
passi = # vertici adiacenti

$\Theta(m)$

in tutto
if WHILE

$\# passi = \sum \text{grado}(v) = 2m$

VISITA IN AMPIEZZA: tempo



MATRICE DI ADIACENZA

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

\uparrow
iterazione

Tempo $O(n^2)$

ALGORITMO visitaInAmpiezza (grafo $G = (V,E)$, vertice s) \rightarrow Albero

$C \leftarrow$ coda vuota

$T \leftarrow$ albero formato solo da s

marca s come raggiunto

$C.\text{enqueue}(s)$

WHILE NOT $C.\text{isEmpty}()$ DO

$m \leftarrow C.\text{dequeue}()$

FOR EACH $(u,v) \in E$ DO

"PUNTO DI RICERCA"

IF v non è marcato come raggiunto THEN

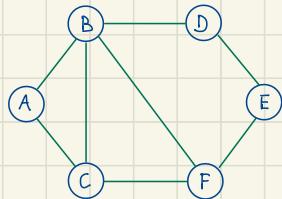
aggiungi v e (u,v) a T

marca v come raggiunto

$C.\text{enqueue}(v)$

RETURN T

VISITA IN AMPIEZZA: tempo



MATRICE DI INCIDENZA

	(A,B)	(A,C)	(B,C)	(B,D)	(B,F)	(C,F)	(D,E)	(E,F)
A	1	1	0	0	0	0	0	0
B	1	0	1	1	1	0	0	0
C	0	1	1	0	0	1	0	0
D	0	0	0	1	0	0	1	0
E	0	0	0	0	0	0	1	1
F	0	0	0	0	1	1	0	1

ALGORITMO visitaInAmpiezza (grafo $G = (V,E)$, vertice s) \rightarrow Albero

$C \leftarrow$ coda vuota

$T \leftarrow$ albero formato solo da s

marca s come raggiunto

$C.\text{enqueue}(s)$

WHILE NOT $C.\text{isEmpty}()$ DO

$m \leftarrow C.\text{dequeue}()$

FOR EACH $(u,v) \in E$ DO] *m non è
ogni ir*

IF v non è marcato come raggiunto THEN

aggiungi v e (u,v) a T

marca v come raggiunto

$C.\text{enqueue}(v)$

RETURN T

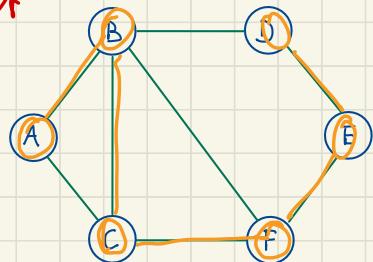
*n
itrazio-*

Temp. $O(n \cdot m)$

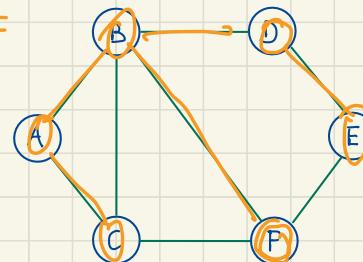
VISITA IN PROFONDITÀ (Depth first search)

- Si inizia visitando un vertice s
- Partendo da s si percorre un cammino di vertici non ancora raggiunti visitandoli tutti, terminando su un vertice privo di vicini non ancora raggiunti
- Si ripete dal passo precedente partendo dall'ultimo vertice sul cammino precedente che ha un vertice non ancora raggiunto

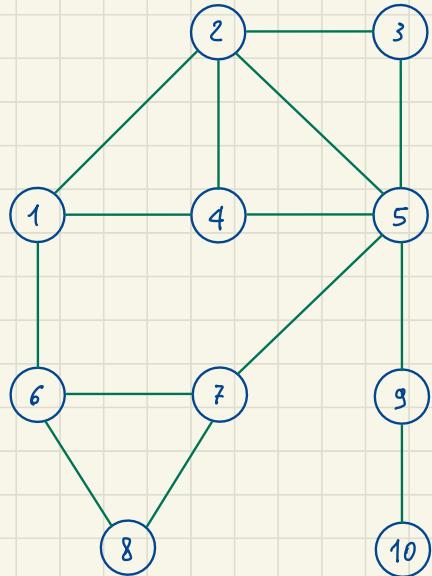
da A



da F



ESEMPIO



ALGORITMO visitaInProfondità (grafo $G = (V, E)$, vertice s) \rightarrow Albero

$T \leftarrow$ albero formato solo da s

visitaRicorsiva (G, s, T)

RETURN T

PROCEDURA visitaRicorsiva (grafo $G = (V, E)$, vertice u , albero T)

marca u come visitato

FOR EACH $(u, v) \in E$ DO

IF v non è marcato come visitato THEN

aggiungi v e (u, v) a T

visitaRicorsiva (G, v, T)