Algoritmi e Strutture Dati

Lezione 20

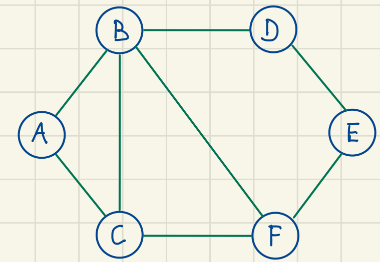
7 novembre 2025

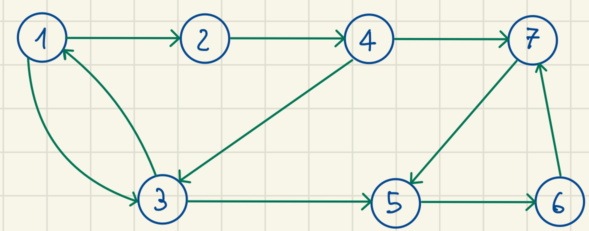
GRAFO $G=(V,E)$

- V VERTICI / NODI
- $E \subseteq V \times V$ ARCHI / LATI / SPIGOLI

GRAFI

- NON ORIENTATI: E SIMMETRICA
- ORIENTATI / DIRETTI





CAMMINO, CICLO, CATENA, CIRCUITO

GRAFO CONNESSO, GRAFO FORTEMENTE CONNESSO

COMPONENTE FORTEMENTE CONNESSA

Rappresentazione di grafi

LISTA DI ARCHI

spesso è un array

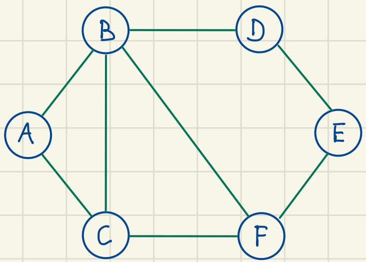
- vertici → spazio $\Theta(n)$
- archi → spazio $\Theta(m)$

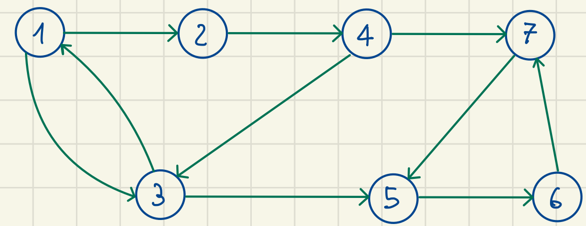
$G = (V, E)$

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

spazio $O(n+m)$

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)

(4,3)

(3,5)

(4,7)

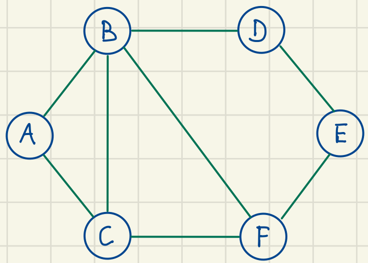
(5,6)

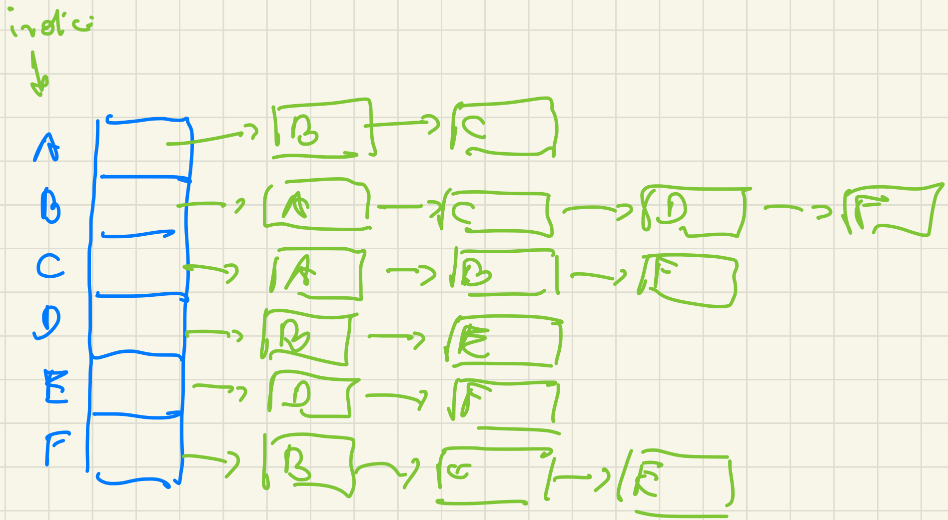
(6,7)

(7,5)

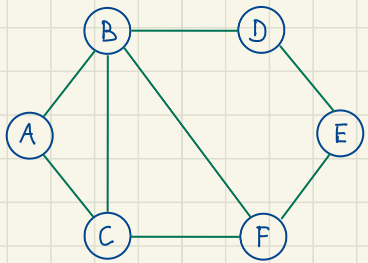
LISTA DI ADIACENZA

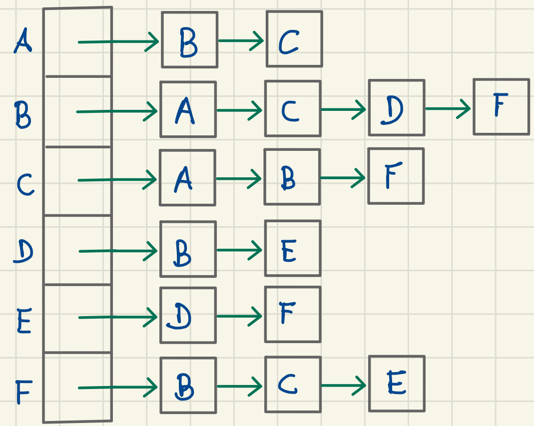
- struttura principale (Array) → vertici
- per ogni vertice: lista vertici adiacenti





LISTA DI ADIACENZA



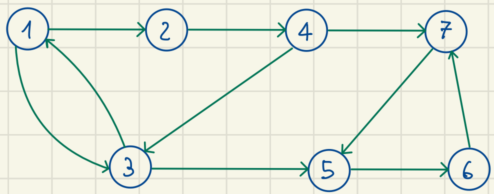


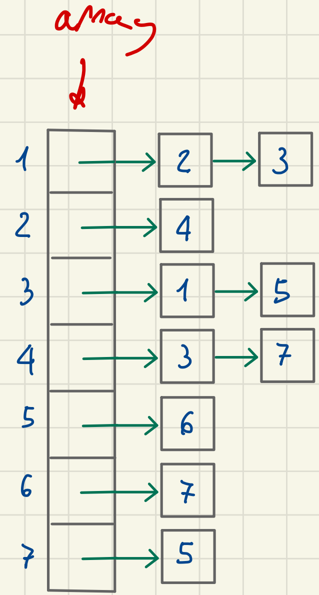
lunghezza di ciascuna lista
= grado vertice corrispondente

$\sum$ lunghezze delle liste $= 2m$ (grafi non orientati)

Spazio $\Theta(n+m)$

LISTA DI ADIACENZA





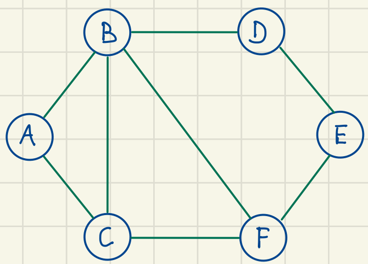
grafo orientato

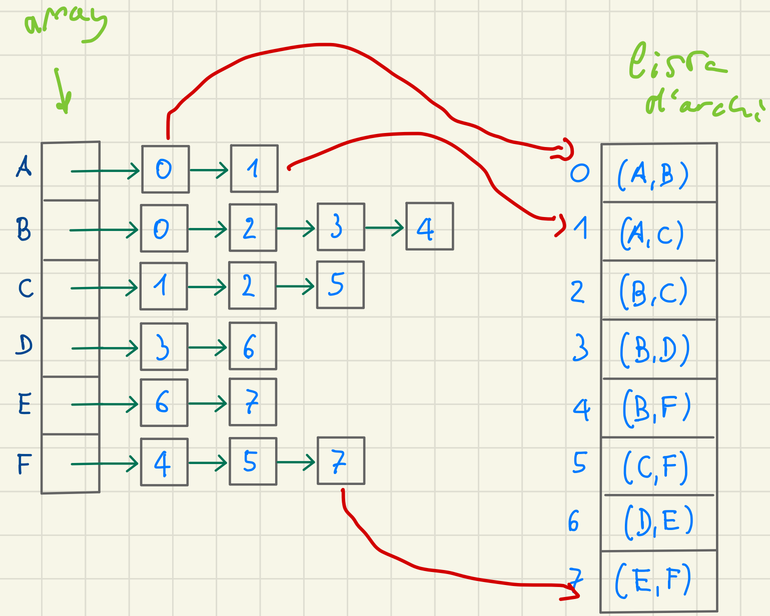
lunghezza lista associata a vertice $= \delta_{out}(v)$

$\sum$ lunghezze delle liste $= \sum_v \delta_{out}(v) = m$

Spazio $\Theta(n+m)$

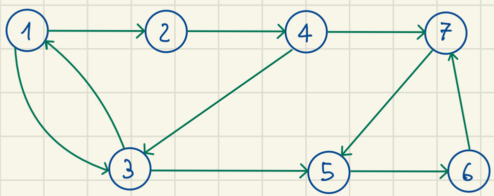
LISTA DI INCIDENZA

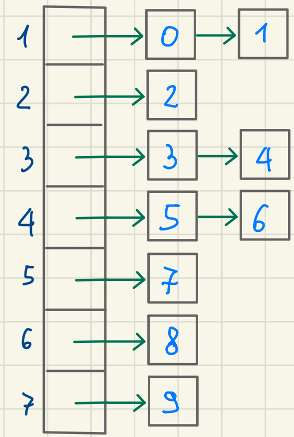


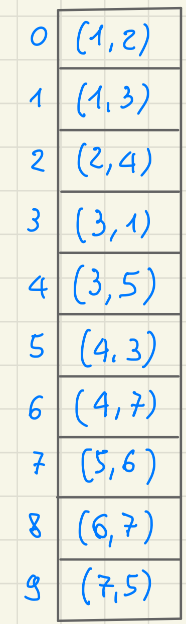


Spazio $\Theta(n+m)$

LISTA DI INCIDENZA







Spazio $\Theta(n+m)$

MATRICE DI ADIACENZA

Matrice booleana $n \times n$

indici $\longleftrightarrow$ vertici del grafo

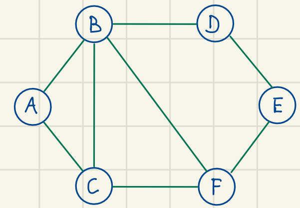
$M[u,v] = 1$ sse $(u,v) \in E$

grafo non orientato

$\downarrow$

matrice simmetrica

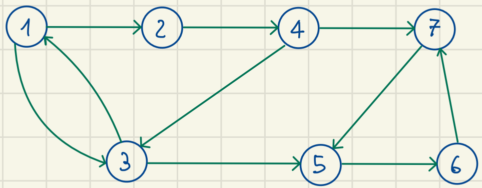
spazio $\Theta(n^2)$



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

	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

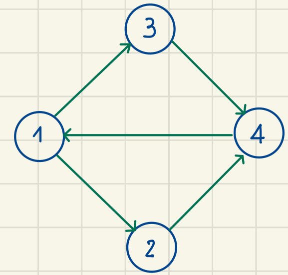


$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

archi uscenti dal vertice $i \rightarrow$ riga i

archi entranti nel vertice $i \rightarrow$ colonna i

MATRICE DI ADIACENZA



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

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

or booleano

$M^2[i,j] = 1$ sse $\exists$ cammino formato da 2 archi da i e j

$M^k[i,j] = 1$ sse $\exists$ cammino di lunghezza k da i e j

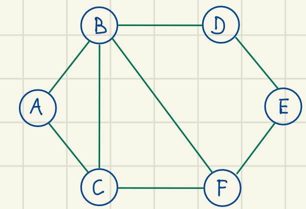
MATRICE DI INCIDENZA

righe → vertici
colonne → archi

0 / 1

grafi non orientati

Spazio $O(n \cdot m)$



	(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

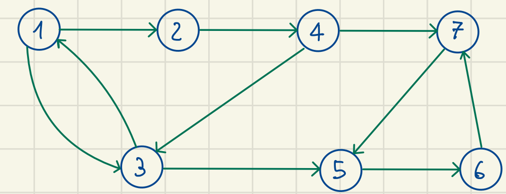
MATRICE DI INCIDENZA

grafi orientati

0

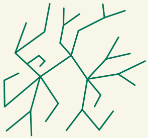
−1 arco entrante

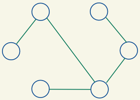
1 " uscente

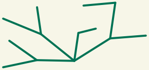


	(1,2)	(1,3)	(2,4)	(3,1)	(3,5)	(4,3)	(4,7)	(5,6)	(6,7)	(7,5)
1	1	1	0	−1	0	0	0	0	0	0
2	−1	0	1	0	0	0	0	0	0	0
3	0	−1	0	1	1	−1	0	0	0	0
4	0	0	−1	0	0	1	1	0	0	0
5	0	0	0	0	−1	0	0	1	0	−1
6	0	0	0	0	0	0	0	−1	1	0
7	0	0	0	0	0	0	−1	0	−1	1

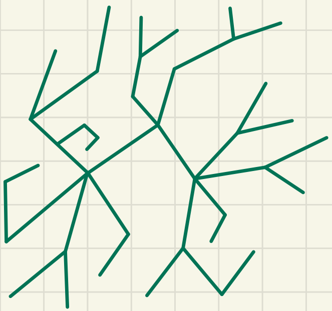
Alberi







ALBERI

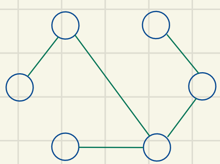


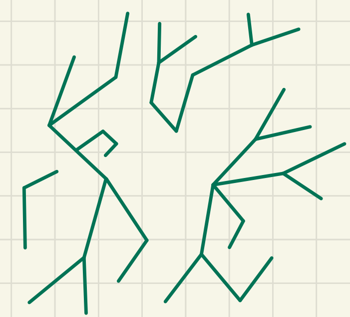
Def.

ALBERO: grafo NON ORIENTATO, CONNESSO e PRIVO DI CICLI

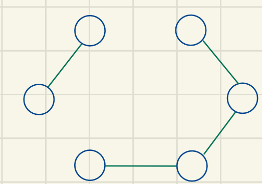
↓

tra ogni coppia di vertici ∃ uno e un solo cammino

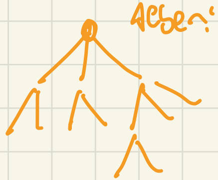




FORESTA: insieme di alberi



Alberi con radice

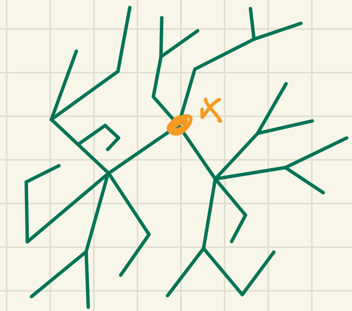


Propr. 1 Sia $G=(V,E)$ un albero. Allora $\#E = \#V - 1$

Dim $n = \#V$, induzione su n

$\boxed{n=1}$ • 1 vertice 0 archi

$\boxed{n>1}$ scelgo un vertice x qualunque



elimino x e i suoi archi incidenti da G

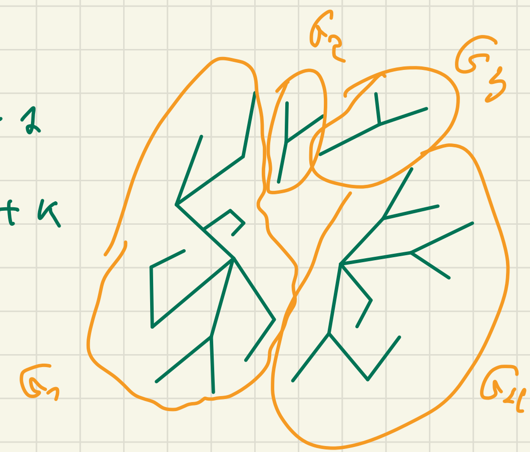
Siano $G_1=(V_1,E_1) \dots G_k=(V_k,E_k)$ gli alberi ottenuti

$\#V_1, \dots, \#V_k < n \rightarrow$ ip. ind:

$$\#E_1 = \#V_1 - 1, \quad \#E_2 = \#V_2 - 1, \dots \quad \#E_k = \#V_k - 1$$

$$\#E = \#E_1 + \dots + \#E_k + \underbrace{k}_{\text{archi incidenti a } x} = \#V_1 - 1 + \dots + \#V_k - 1 + k$$

$$= \#V_1 + \#V_2 + \dots + \#V_k = \#V - \underbrace{1}_{\text{vertice } x}$$

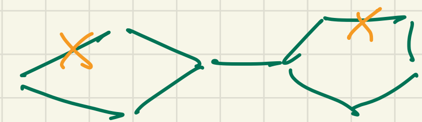


Prop. 2 Sia $G = (V, E)$ non orientato e connesso

Se $\# E = \# V - 1$ allora G è un albero

Dim Per assurdo supponiamo che G non sia un albero

allora G contiene almeno un ciclo



Elimino un arco dal ciclo

e ripeto finché non ottengo un albero

$\to$ il grafo risultante:

connesso

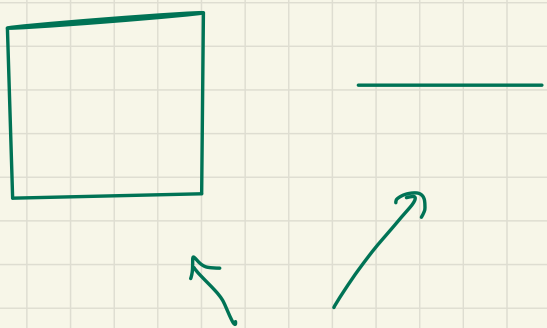
privo di cicli e ha meno di $\# V - 1$ archi

albero

ASSURDO

Teo (Prop 1 + Prop. 2)

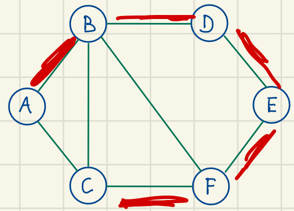
Un grafo $G=(V,E)$ non orientato e connesso è un albero sse $\#E = \#V - 1$

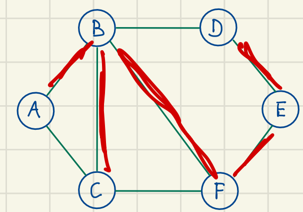


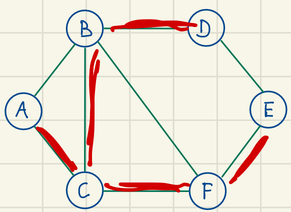
$n = 6$

$m = 5$

non connesso



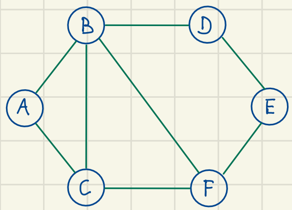




Alberi ricoprenti

ALBERO di SUPPORTO o RICOPRENTE (Spanning tree)

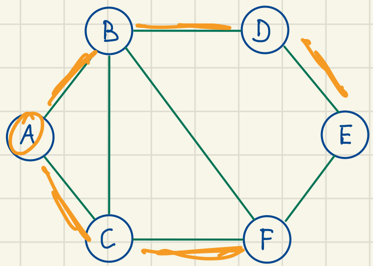
Dato $G=(V,E)$ grafo non orientato connesso,
un albero ricoprente di G è un albero $G'=(V',E')$
con $V'=V$ e $E'\subseteq E$



Attraversamento di grafi

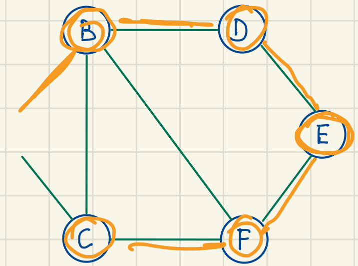
VISITA IN AMPIEZZA (Breadth first search)

da A



A, B, C, D, F, E

da E



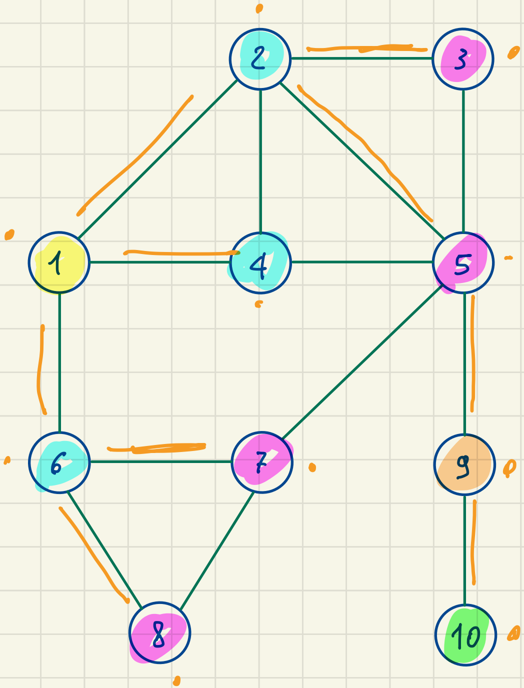
E, D, F, B, C

- 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 visitaInAmpiezza (grafo G=(V,E), vertice s) → Albero
  C ← coda vuota
  T ← albero formato solo da s
  marca s come raggiunto
  C.enqueue(s)
  WHILE NOT C.isEmpty() DO
    u ← C.dequeue()
    FOR EACH (u,v) ∈ E DO
      IF v non è marcato come raggiunto THEN
        aggiungi v e (u,v) a T
        marca v come raggiunto
        C.enqueue(v)
  RETURN T
```

ESEMPIO da 1




```
ALGORITMO visitaInAmpiezza (grafo G=(V,E), vertice s) → Albero
  C ← coda vuota
  T ← albero formato solo da s
  marca s come raggiunto
  C.enqueue(s)
  WHILE NOT C.isEmpty() DO
    u ← C.dequeue()
    FOR EACH (u,v) ∈ E DO
      IF v non è marcato come raggiunto THEN
        aggiungi v e (u,v) a T
        marca v come raggiunto
        C.enqueue(v)
  RETURN T
```

~~1~~ ~~2~~ ~~4~~ ~~6~~ ~~3~~ ~~5~~ ~~7~~ ~~8~~ ~~9~~ ~~10~~