

# **Corso di Laurea in INFORMATICA**

## **Algoritmi e Strutture Dati a.a. 2011-2012 MODULO 12 STRUTTURE NON LINEARI**

**Il tipo astratto grafo: specifiche sintattiche e  
semantiche. Realizzazioni. Visita di un grafo.**

**Questi lucidi sono stati preparati per uso didattico. Essi contengono materiale originale di proprietà dell'Università degli Studi di Bari e/o figure di proprietà di altri autori, società e organizzazioni di cui e' riportato il riferimento. Tutto o parte del materiale può essere fotocopiato per uso personale o didattico ma non può essere distribuito per uso commerciale. Qualunque altro uso richiede una specifica autorizzazione da parte dell'Università degli Studi di Bari e degli altri autori coinvolti.**



# Definizione

Un **grafo**  $G=(N,A)$  consiste in:

- un insieme  $N$  di **nodi** (o **vertici**),  $|N|=n$
- un insieme  $A$  di coppie di nodi, detti **archi** o **spigoli**: ogni arco connette due vertici

$$A=\{(v_i, v_j): v_i, v_j \in N\}$$

In particolare:

$(v_i, v_j)=(v_j, v_i)$ : Grafo semplice

$(v_i, v_j) \neq (v_j, v_i)$ : Grafo diretto o orientato

In generale i **nodi** sono usati per rappresentare **entità** e gli **archi** per rappresentare **relazioni** tra entità



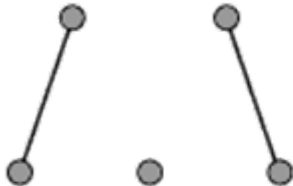
# Esempi

- Relazioni tra classi nei linguaggi OO
- Grafo del Web
- Assetti societari
- Reti di trasporto
- Social Network
- .....

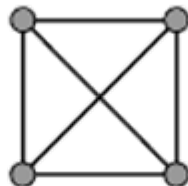




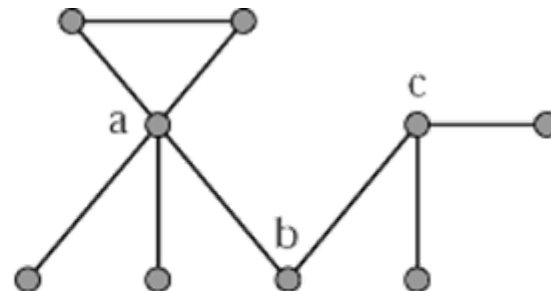
a)



b)



c)



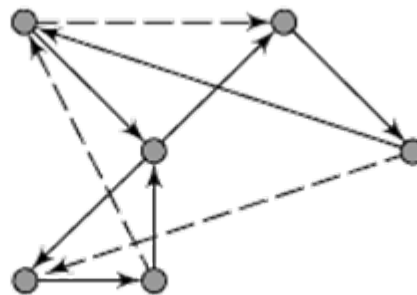
d)



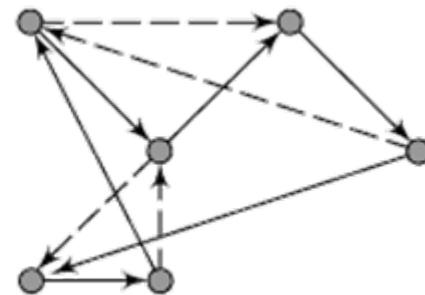
e)



f)



g)



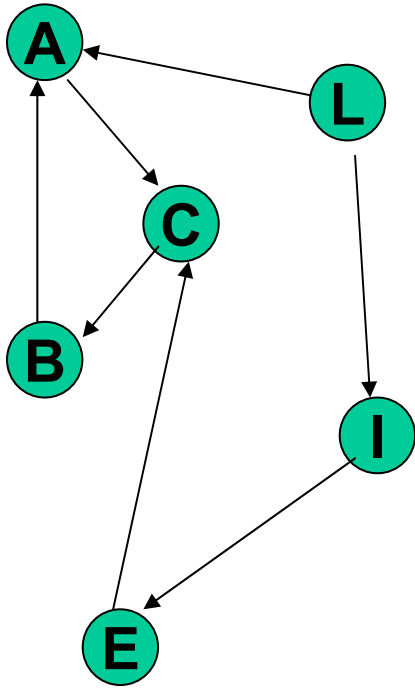
h)

Esempi di grafi



# Grafo orientato (**D**irected **G**raph)

## Terminologia



$\langle L, I, E, C, B, A \rangle$  è un **cammino** nel grafo orientato di lunghezza 5

Il **cammino** deve rispettare il verso di **orientamento** degli archi

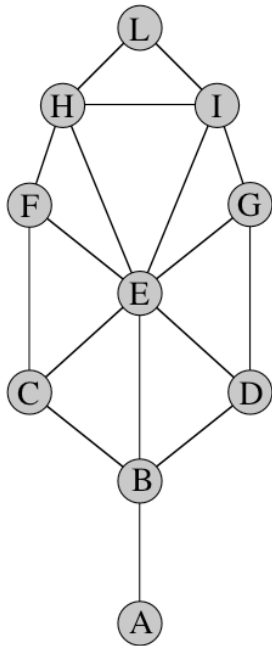
$\langle A, C, B \rangle$  costituiscono un **anello** o **ciclo**

Dati due nodi **A** e **I** se esiste un cammino da **A** ad **I** oppure da **I** ad **A** il grafo orientato si dice **connesso**. Se esiste un cammino da **A** ad **I** e da **I** ad **A** si dice **fortemente connesso**



# Grafo non orientato: Terminologia

relazione simmetrica → grafo non orientato



$n$  = numero di nodi

$m$  = numero di archi

L ed I sono **adiacenti**

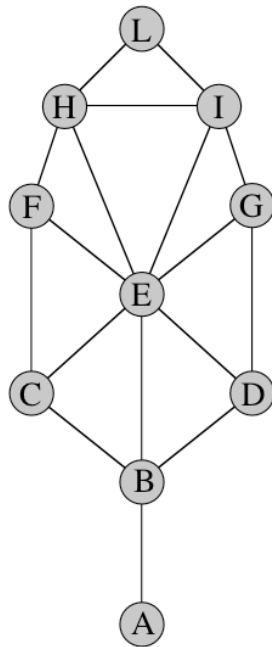
$(L, I)$  è **incidente** ad L e ad I



# Grafo non orientato: Terminologia

$\langle L, I, E, C, B, A \rangle$  è una **catena** nel grafo (non orientato) di lunghezza 5

**Circuito** è il concetto analogo a ciclo



La lunghezza del più corto cammino tra due vertici si dice **distanza** tra i due vertici: L ed A hanno distanza 4

Se esiste un cammino per ogni coppia di vertici, allora il grafo si dice **connesso**

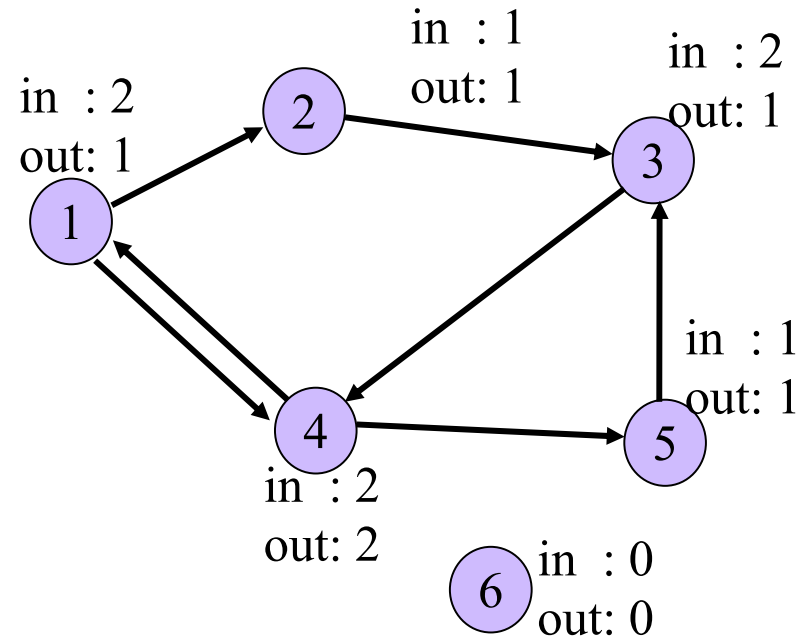
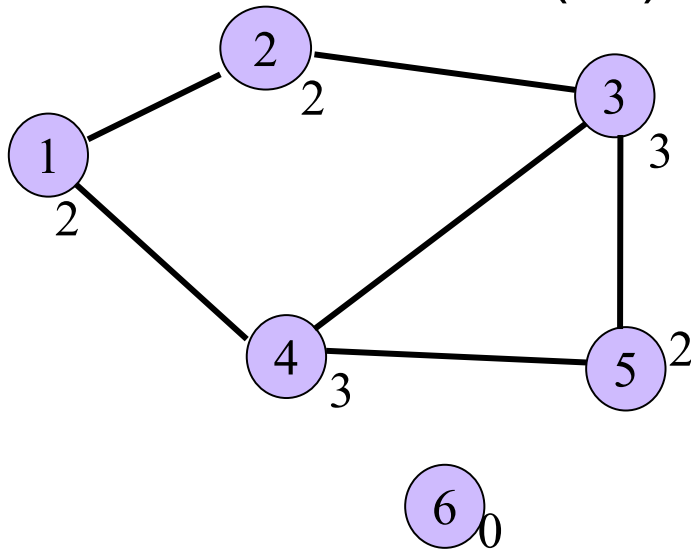
# Grado del Grafo

In un grafo non orientato

- il **grado** di un vertice è il numero di archi che partono da esso

In un grafo orientato

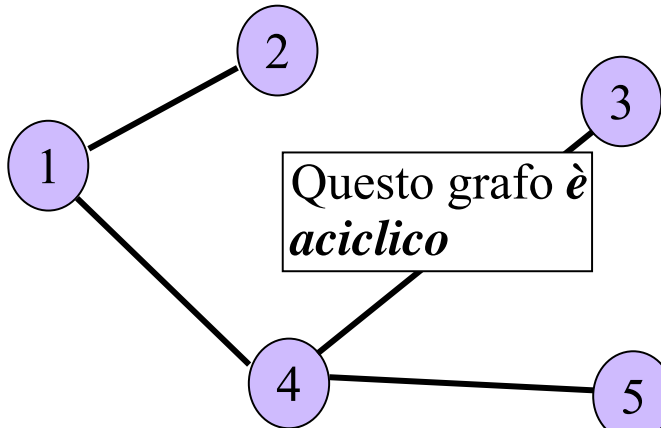
- il **grado entrante (uscente)** di un vertice è il numero di archi incidenti in (da) esso



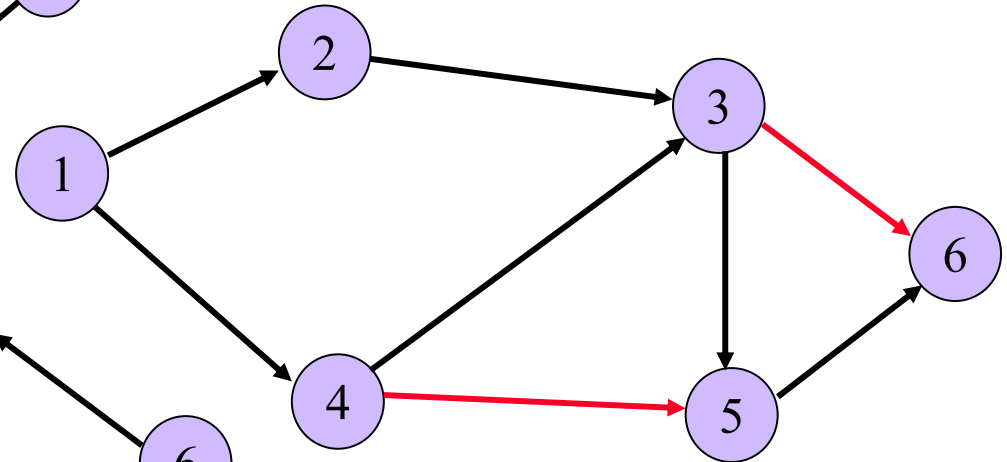
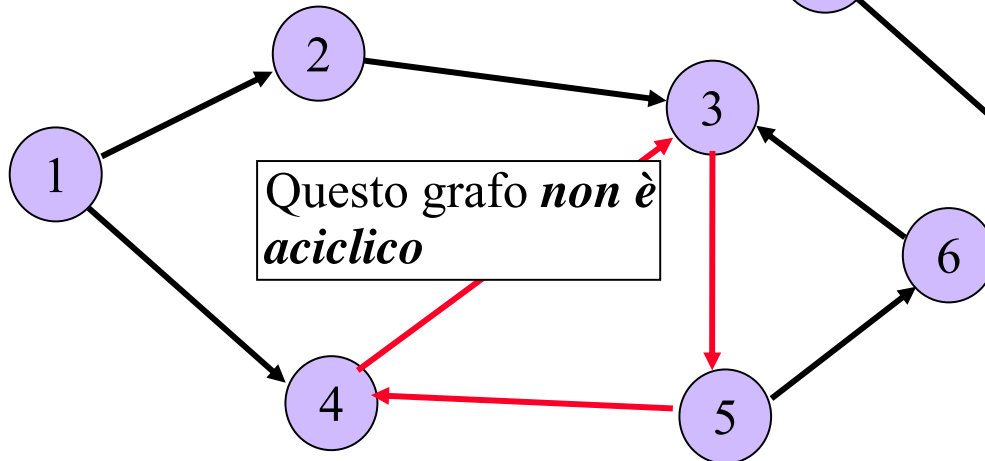


# Grafi aciclici

Un grafo senza cicli è detto aciclico



Un *grafo orientato aciclico* è chiamato **DAG** (*Directed Acylic Graph*)



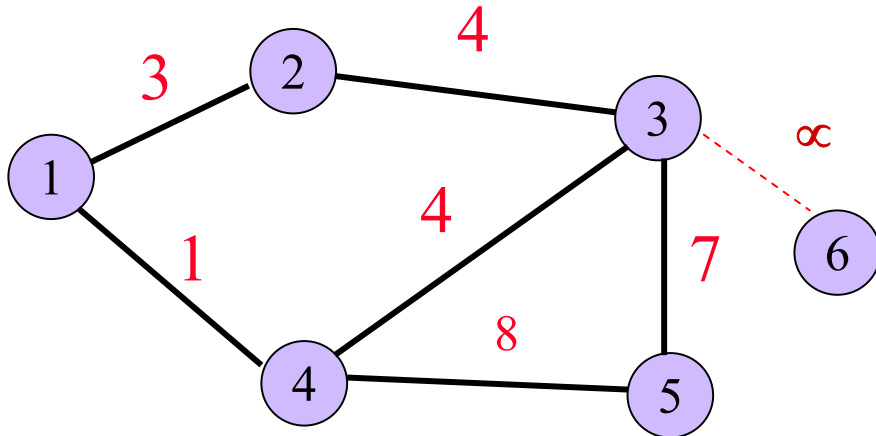
# Grafi pesati

In alcuni casi ogni arco ha un *peso* (*costo*, *guadagno*) associato

- Il peso può essere determinato tramite una funzione di costo

$p: N \times N \rightarrow R$ , dove  $R$  è l'insieme dei numeri reali

- Quando tra due vertici non esiste un arco, il peso è infinito

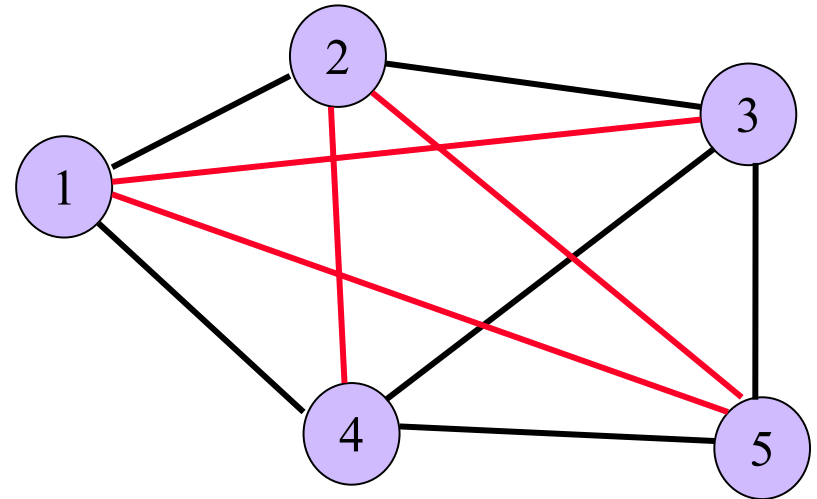
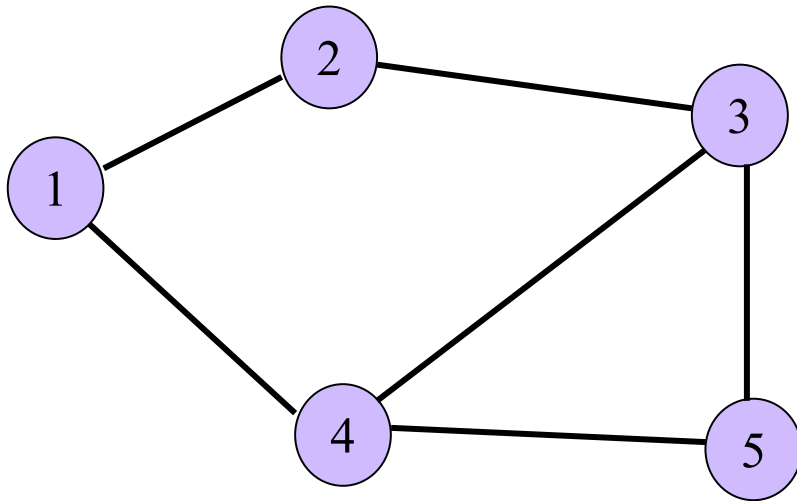


# Grafo completo

Un *grafo completo* è un grafo che ha un arco tra ogni coppia di vertici.

Questo grafo *è completo*

Questo grafo *non è completo*



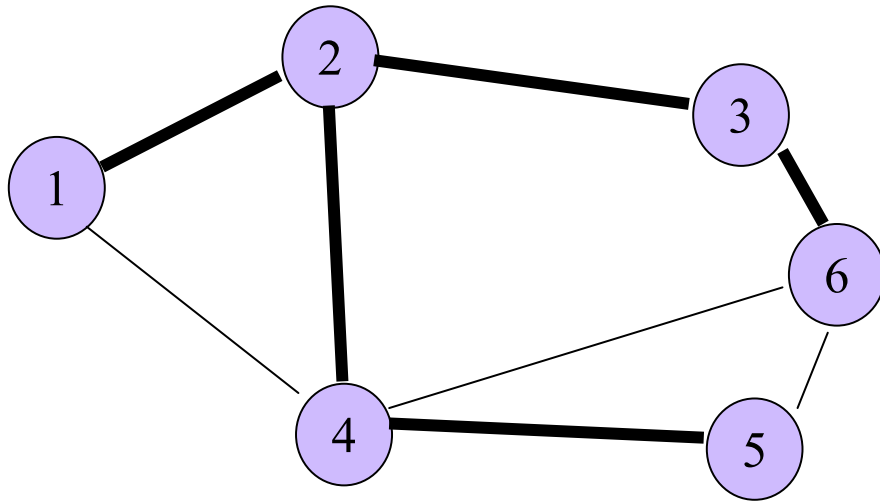
$$m = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$



# Alberi di copertura

In un grafo non orientato  $G=(N, A)$

un albero di copertura  $T$  è un albero libero  $T = (N, A')$  composto da tutti i nodi di  $N$  e da un sottoinsieme degli archi ( $A' \subseteq A$ ), tale per cui tutte le coppie di nodi del grafo sono connesse da una sola catena nell'albero.



## IL TIPO ASTRATTO GRAFO

PER DEFINIRE L'ALGEBRA CONSIDERIAMO SOLO  
**GRAFI ORIENTATI O DIRETTI.**

GLI ARCHI HANNO UNA DIREZIONE DA UN CERTO NODO (DI PARTENZA) A UN ALTRO NODO (DI ARRIVO). IL GRAFO  $G = (N, A)$ , DOVE  $N$  E' L'INSIEME FINITO DEI NODI, PREVEDE CHE  $A$  SIA UN INSIEME FINITO DI **COPPIE ORDINATE** DI NODI, RAPPRESENTANTI GLI ARCHI ORIENTATI.

OGNI GRAFO NON ORIENTATO  $G'$  PUO' ESSERE VISTO COME UN GRAFO ORIENTATO  $G$ , OTTENUTO DA  $G'$ , SOSTITUENDO OGNI ARCO  $(i, j)$  CON I DUE ARCHI  $(j, i)$  E  $(i, j)$ .



# SPECIFICA SINTATTICA

**TIPI:** GRAFO, BOOLEAN, NODO, LISTA, **TIPOELEM**

**OPERATORI:**

CREAGRAFO:	()	→	GRAFO
GRAFOVUOTO:	(GRAFO)	→	BOOLEAN
INSNODO:	(NODO, GRAFO)	→	GRAFO
INSARCO:	(NODO, NODO, GRAFO)	→	GRAFO
CANCNODO:	(NODO, GRAFO)	→	GRAFO
CANCARCO:	(NODO, NODO, GRAFO)	→	GRAFO
ADIACENTI:	(NODO, GRAFO)	→	LISTA
ESISTENODO:	(NODO, GRAFO)	→	BOOLEAN
ESISTEARCO:	(NODO, NODO, GRAFO)	→	BOOLEAN



# SPECIFICA SEMANTICA

**TIPI:** **GRAFO:** INSIEME  $G = (N, A)$  CON  $N$  SOTTOINSIEME FINITO DI ELEMENTI DI TIPO “NODO” E  $A \subseteq N \times N$

**NODO:** INSIEME **FINITO** QUALSIASI

**LISTA:** LISTA DI ELEMENTI DI TIPO NODO

**BOOLEAN:** INSIEME DEI VALORI DI VERITA’

**OPERATORI:**

**CREAGRAFO** =  $G$

**PRE:** NESSUNA

**POST:**  $G = (N, A)$  CON  $N = \emptyset$  E  $A = \emptyset$



**GRAFOVUOTO (G) = b**

**PRE: NESSUNA**

**POST: b=VERO SE  $N = \emptyset$  E  $A = \emptyset$**

**b=FALSO ALTRIMENTI**

**INSNODO (u,G) = G'**

**PRE:  $G = (N,A)$   $u \notin N$  (E' DI TIPO "NODO")**

**POST:  $G' = (N',A)$ ,  $N' = N \cup \{u\}$**

**INSARCO (u,v,G) = G'**

**PRE:  $G = (N,A)$ ,  $u \in N$ ,  $v \in N$ ,  $(u,v) \notin A$**

**POST:  $G' = (N,A')$ ,  $A' = A \cup \{(u,v)\}$**

**CANCNODO (u,G) = G'**

**PRE:  $G = (N,A)$ ,  $u \in N$**

**NON ESISTE  $v \in N \ni (u,v) \in A$  OPPURE  $(v,u) \in A$**

**POST:  $G' = (N',A)$ ,  $N' = N - \{u\}$**





**CANCARCO  $(u,v,G) = G'$**

**PRE:  $G = (N,A), u \in N, v \in N, (u,v) \in A$**

**POST:  $G' = (N,A'), A' = A - \{(u,v)\}$**

**ADIACENTI  $(u,G) = L$**

**PRE:  $G = (N,A), u \in N$**

**POST: L E' UNA LISTA CHE CONTIENE UNA E UNA SOLA VOLTA GLI ELEMENTI DI  $A(u) = \{v \mid (u,v) \in A\}$**

**ESISTENODO  $(u,G) = b$**

**PRE:  $G = (N,A)$**

**POST:  $b = \text{VERO}$  SE  $u \in N$**

**$b = \text{FALSO}$  ALTRIMENTI**



**ESISTEARCO**  $(u,v,G) = b$

**PRE:**  $G = (N,A)$ ,  $u \in N$ ,  $v \in N$ ,

**POST:**  $b$  SE  $(u,v) \in A$

$b = \text{FALSO}$  ALTRIMENTI

TALVOLTA AI **NODI** E AGLI **ARCHI** SONO ASSOCIATE INFORMAZIONI (**ETICHETTE**, **PESI**), SI PARLA DI **GRAFI ETICHETTATI** NEI **NODI** / **PESATI** NEGLI **ARCHI** IN TAL CASO VANNO INTRODOTTI NUOVI OPERATORI PER SCRIVERE E LEGGERE I NODI

**LEGGINODO:**  $(\text{NODO}, \text{GRAFO}) \rightarrow \text{TIPOELEM}$

**SCRIVINODO:**  $(\text{TIPOELEM}, \text{NODO}, \text{GRAFO}) \rightarrow \text{GRAFO}$

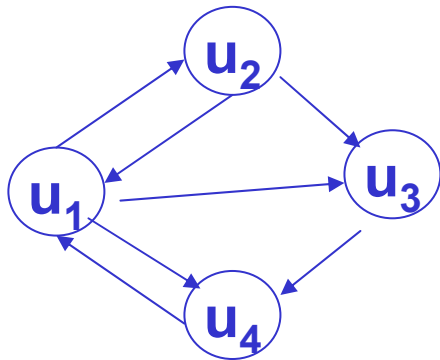
E PER RITROVARE O MODIFICARE LE INFORMAZIONI ASSOCIATE AGLI ARCHI USEREMO **LEGGIARCO** E

**SCRIVIARCO**



# RAPPRESENTAZIONE CON MATRICE DI ADIACENZA

LA PIU' SEMPLICE RAPPRESENTAZIONE UTILIZZA UNA MATRICE  $N \times N$ ,  $E = [e_{ij}]$ , TALE CHE  $e_{ij} = 1$  NEL CASO  $(i,j) \in A$ , MENTRE  $e_{ij} = 0$  SE  $(i,j) \notin A$ .



	1	2	3	4
1	0	1	1	1
2	1	0	1	0
3	0	0	0	1
4	1	0	0	0

SE IL GRAFO E' PESATO, NELLA MATRICE SI UTILIZZANO I PESI DEGLI ARCHI AL POSTO DEGLI ELEMENTI BINARI. SE  $P_{ij}$  E' IL PESO DELL'ARCO  $(i,j)$  ALLORA L'ELEMENTO DELLA MATRICE E DIVENTA

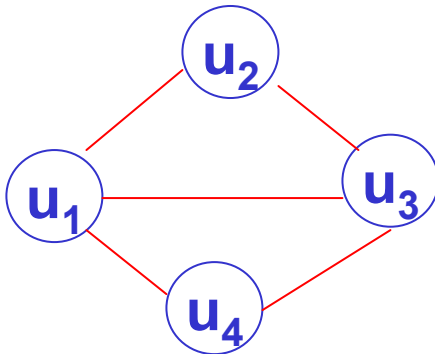
$e_{ij} =$

$$\left\{ \begin{array}{ll} P_{ij} & \text{SE } (i,j) \in A \\ +\infty \text{ } (-\infty) & \text{SE } (i,j) \notin A \end{array} \right.$$



# RAPPRESENTAZIONE CON MATRICE DI ADIACENZA

NATURALMENTE E' POSSIBILE UTILIZZARE LA MEDESIMA RAPPRESENTAZIONE PER GRAFI NON ORIENTATI: NE RISULTERA' UNA MATRICE SIMMETRICA RISPETTO ALLA DIAGONALE PRINCIPALE CON  $e_{ij} = e_{ji}$

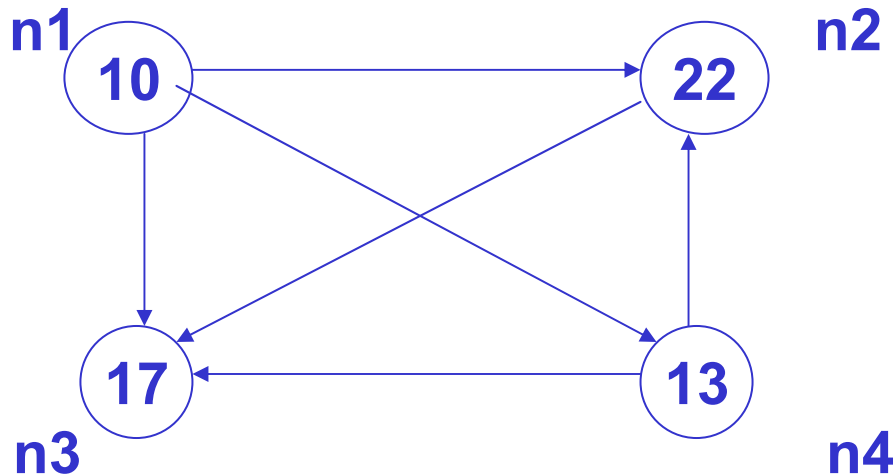


	1	2	3	4
1	0	1	1	1
2	1	0	1	0
3	1	1	0	1
4	1	0	1	0



# LA MATRICE DI ADIACENZA PER GRAFI ETICHETTATI

NEL CASO IL GRAFO SIA ETICHETTATO POSSIAMO ASSOCIARE AL NODO ALTRE INFORMAZIONI.



POSSIAMO SEMPRE UTILIZZARE LA RAPPRESENTAZIONE CON MATRICE DI ADIACENZA.



# LA MATRICE DI ADIACENZA (UNA ESTENSIONE)

	LABEL	MARK	ARCHI	RIGA				
n=1	10	1	3	0	1	1	1	
n=2	22	1	3	0	0	1	0	
n=3	17	1	3	0	0	0	0	
n=4	13	1	3	0	1	1	0	
n=5	24	0	0					

**MARK** E' UN FLAG CHE HA VALORE FALSO O 0 SE IL NODO E' STATO RIMOSSO.

**ARCHI** CONTIENE IL NUMERO SOMMA DEGLI ENTRANTI E USCENTI k DAL GENERICO NODO.



# RAPPRESENTAZIONE CON MATRICI D'INCIDENZA

UN GRAFO  $G = (N,A)$  PUO' ANCHE ESSERE RAPPRESENTATO MEDIANTE UNA MATRICE  $(n \times m)$ ,  $B = [b_{ik}]$ , NELLA QUALE CIASCUNA RIGA RAPPRESENTA UN NODO E CIASCUNA COLONNA RAPPRESENTA UN ARCO.

PER UN **GRAFO NON ORIENTATO**

$$b_{ij} = \begin{cases} 1 & \text{SE L'ARCO } j\text{-ESIMO E' INCIDENTE} \\ & \text{NEL NODO } i \\ 0 & \text{ALTRIMENTI} \end{cases}$$

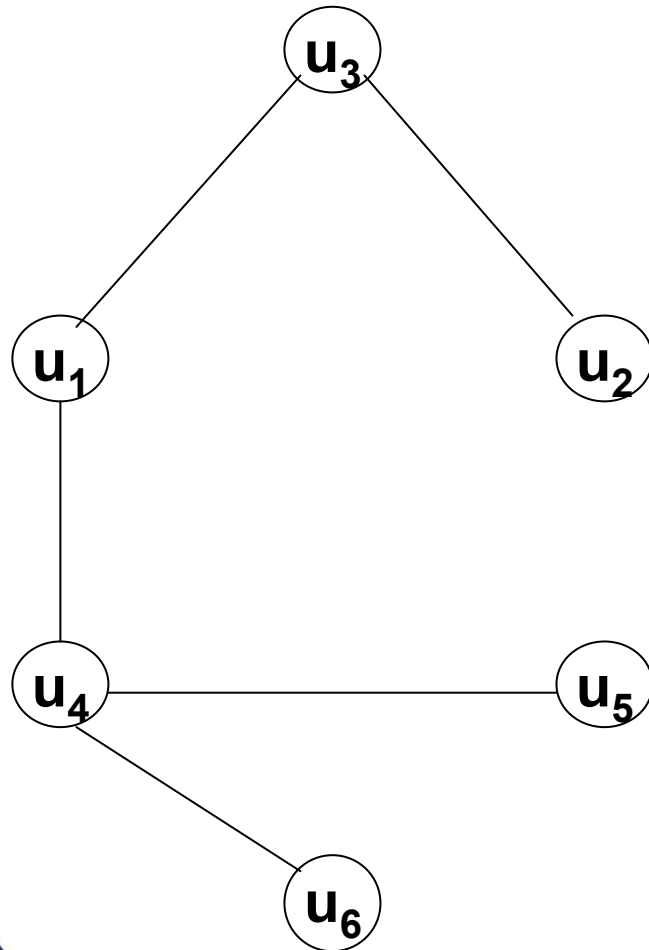
NEL CASO DI **GRAFI ORIENTATI** O DIRETTI IL GENERICO ELEMENTO DI B DIVIENE

$$b_{ij} = \begin{cases} +1 & \text{SE L'ARCO } j\text{-ESIMO ENTRA NEL} \\ & \text{NODO } i \\ -1 & \text{SE L'ARCO } j\text{-ESIMO ESCE DAL} \\ & \text{NODO } i \\ 0 & \text{ALTRIMENTI} \end{cases}$$



# RAPPRESENTAZIONE CON MATRICI D'INCIDENZA

## GRAFO NON ORIENTATO

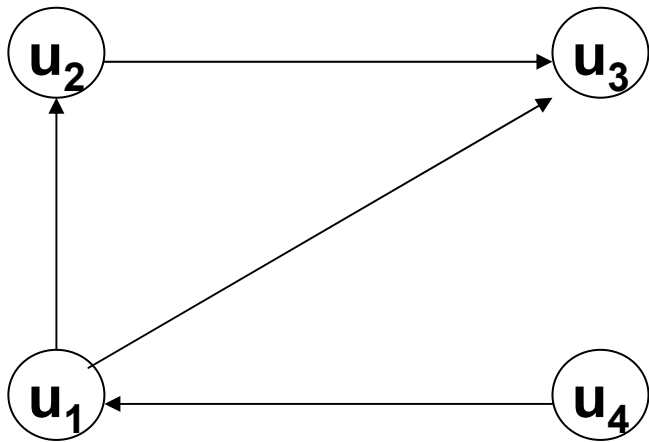


	$(u_1, u_3)$	$(u_1, u_4)$	$(u_2, u_3)$	$(u_4, u_5)$	$(u_4, u_6)$
	1	2	3	4	5
1	1	1	0	0	0
2	0	0	1	0	0
3	1	0	1	0	0
4	0	1	0	1	1
5	0	0	0	1	0
6	0	0	0	0	1





# RAPPRESENTAZIONE CON MATRICI D'INCIDENZA GRAFO ORIENTATO



	$(u_1, u_2)$	$(u_1, u_3)$	$(u_2, u_3)$	$(u_4, u_1)$
1	-1	-1	0	+1
2	+1	0	-1	0
3	0	+1	+1	0
4	0	0	0	-1

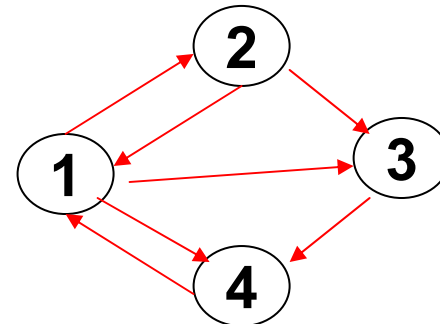
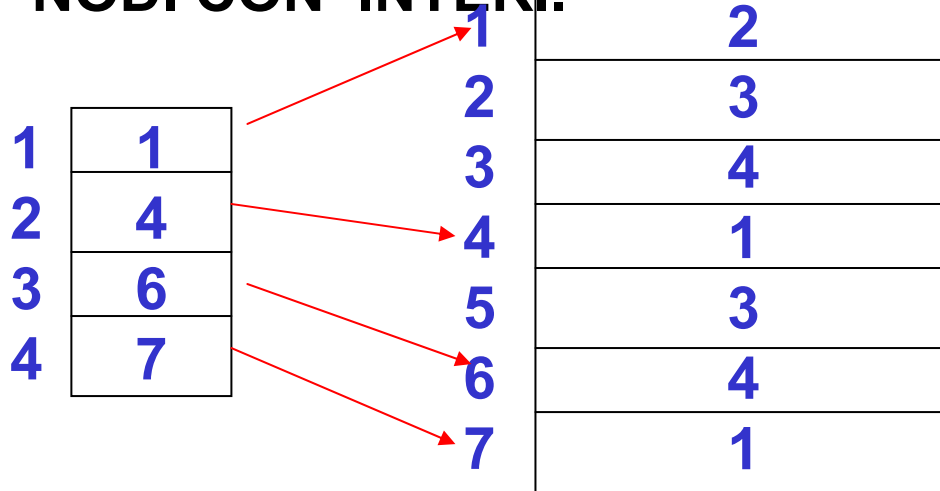
DATO UN NODO NON E' FACILE RICAVARE L'INSIEME DI ADIACENZA.

PER CALCOLARE  $A(u)$  E' NECESSARIO SCANDIRE LA RIGA  $u$  DI  $B$  ALLA RICERCA DELLE COLONNE  $k$   $\ni$   $b_{uk} = -1$ , E PER OGNI COLONNA  $k$  SCANDIRE L'INDICE DI RIGA  $i$   $\ni$   $b_{ik} = +1$ .



# RAPPRESENTAZIONE CON VETTORI DI ADIACENZA (GRAFO ORIENTATO)

E' POSSIBILE RAPPRESENTARE IL GRAFO  $(N,A)$  CON DUE VETTORI, IL VETTORE **NODI** E IL VETTORE **ARCHI**. IL VETTORE **NODI** E' FORMATO DA N ELEMENTI E **NODI(i)** CONTIENE UN CURSORE ALLA POSIZIONE DI **ARCHI** A PARTIRE DALLA QUALE E' MEMORIZZATO  $A(i)$ . PER SEMPLICITA' DENOTIAMO I NODI CON INTERI.

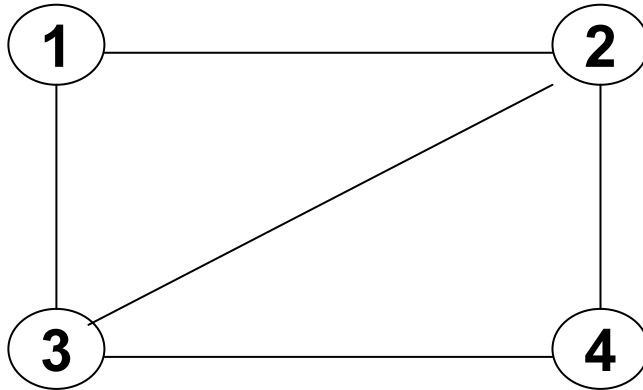


NEL CASO IL GRAFO SIA ETICHETTATO POSSIAMO ASSOCIARE AL NODO ALTRE INFORMAZIONI.

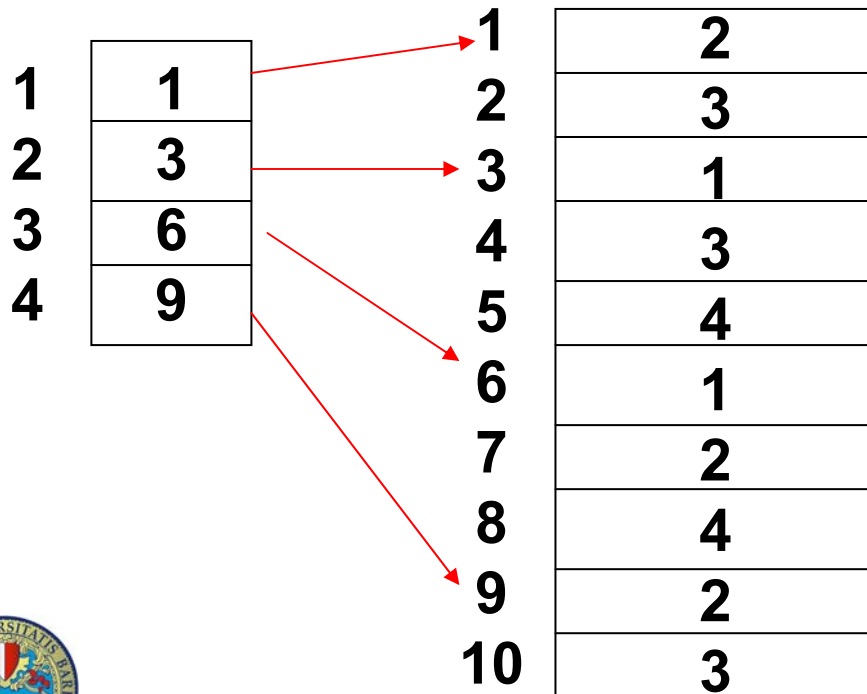


# RAPPRESENTAZIONE CON VETTORI DI ADIACENZA (GRAFO NON ORIENTATO)

E' NECESSARIO RAPPRESENTARE OGNI ARCO  $[i,j]$  DUE VOLTE.



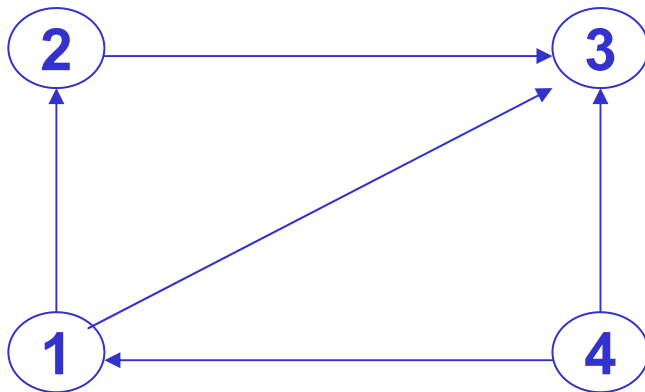
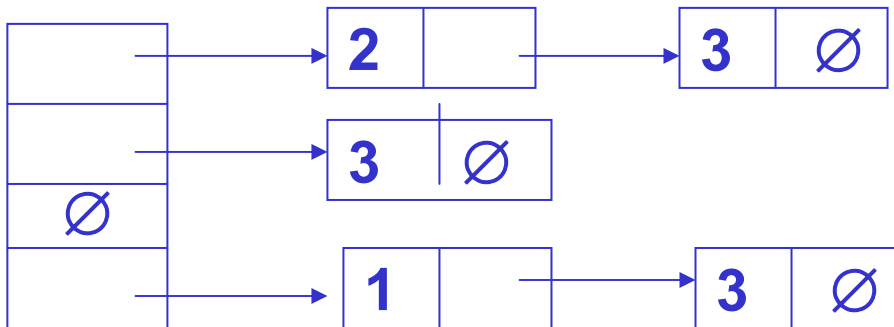
*SE I GRAFI SONO ETICHETTATI  
SUI NODI E/O SUGLI ARCHI I PESI  
POSSONO ESSERE MEMORIZZATI  
IN VETTORI LABELNODI(n) E  
PESIARCHI (m)*



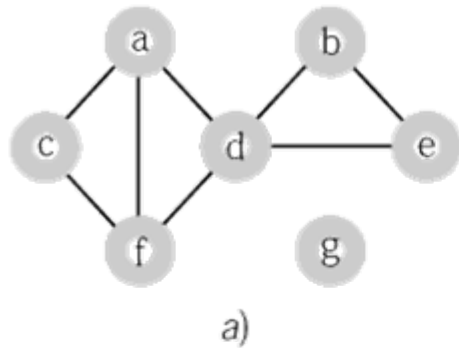
# RAPPRESENTAZIONE CON LISTE DI ADIACENZA

E' POSSIBILE ANCHE UTILIZZARE UN VETTORE DI NODI A (1..n) ED n LISTE.

UNA GENERICA COMPONENTE  $A(i)$  DEL VETTORE E' IL PUNTATORE ALLA LISTA  $i$ -ESIMA IN CUI SONO MEMORIZZATI I NODI ADIACENTI A  $i$ .

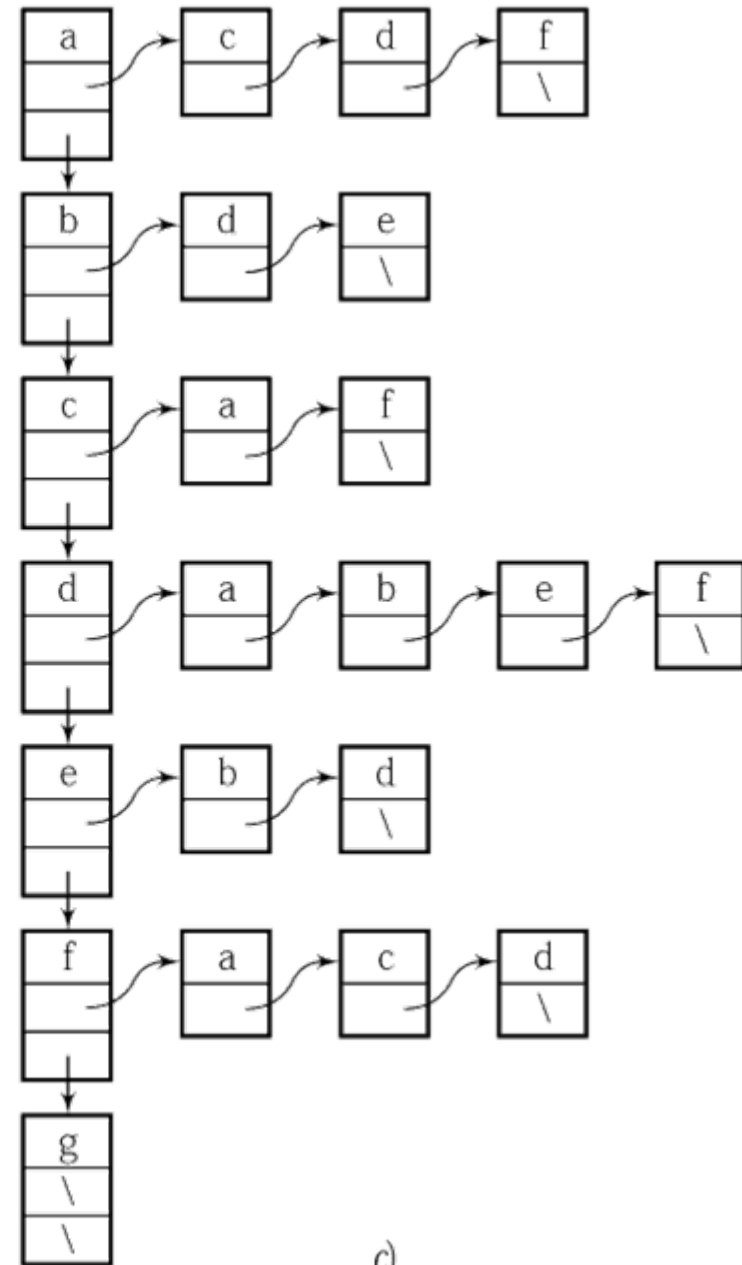


Un grafo (a)  
rappresentato  
con vettore che  
implementa  
una lista  
(statica) di  
adiacenze  
(b) e con una  
lista di liste di  
adiacenza (c)



a	c	d	f	
b	d	e		
c	a	f		
d	a	b	e	f
e	b	d		
f	a	c	d	
g				

b)



	a	b	c	d	e	f	g
a	0	0	1	1	0	1	0
b	0	0	0	1	1	0	0
c	1	0	0	0	0	1	0
d	1	1	0	0	1	1	0
e	0	1	0	1	0	0	0
f	1	0	1	1	0	0	0
g	0	0	0	0	0	0	0

d)

	ac	ad	af	bd	be	cf	de	df
a	1	1	1	0	0	0	0	0
b	0	0	0	1	1	0	0	0
c	1	0	0	0	0	1	0	0
d	0	1	0	1	0	0	1	1
e	0	0	0	0	1	0	1	0
f	0	0	1	0	0	1	1	0
g	0	0	0	0	0	0	0	0

e)

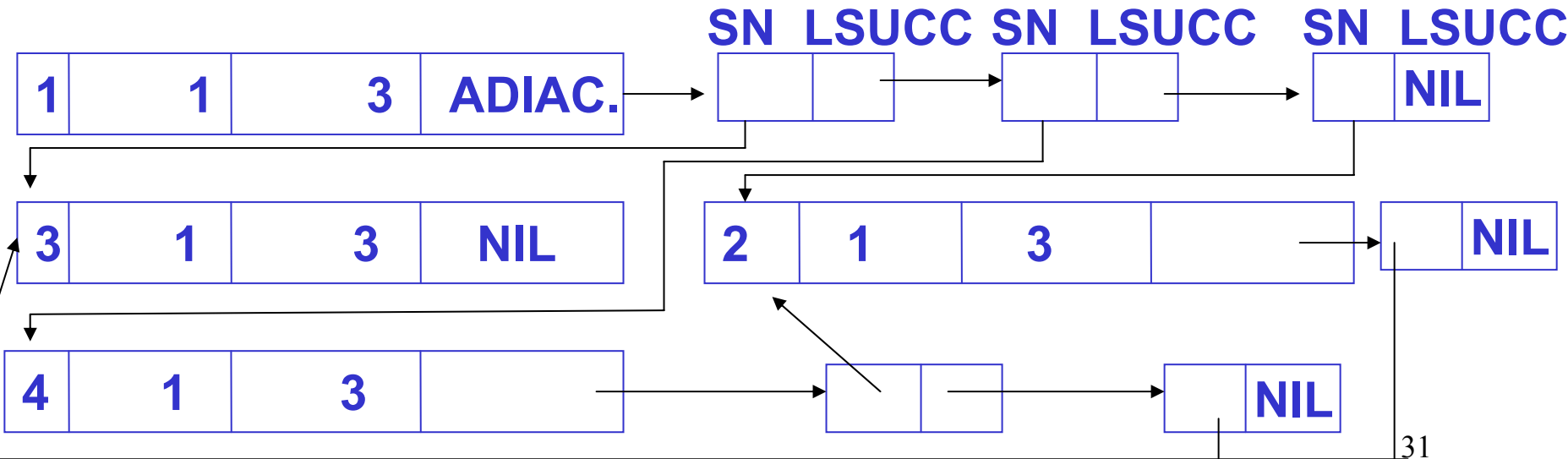
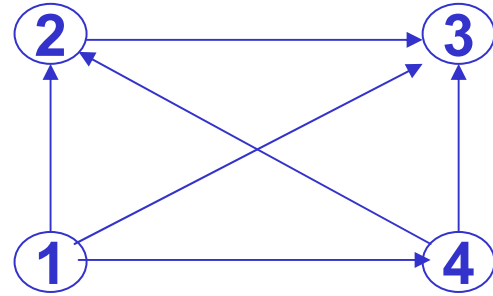
Lo stesso grafo (a) rappresentato come una matrice di adiacenze (d) e come una matrice d'incidenza (e)

# RAPPRESENTAZIONE CON STRUTTURA A PUNTATORI

**E' LA VERSIONE DINAMICA DELLA MATRICE DI ADIACENZA ESTESA. IL NODO  $v$  CON ETICHETTA  $e$ ,  $h$  ARCHI ENTRANTI E  $k$  ARCHI USCENTI E' RAPPRESENTATO MEDIANTE UN RECORD**



**SUCCESSIVO CONTIENE UN RIFERIMENTO ALLA LISTA DEI SUCCESSORI DEL NODO. OGNI ELEMENTO DI QUESTA LISTA CORRISPONDE AD UN ARCO USCENTE.**



# Complessità delle rappresentazioni

Poniamo

□  $n = |N|$  numero di nodi

□  $m = |A|$  numero di archi

## Matrice di adiacenza

□ Spazio richiesto  $O(n^2)$

□ Verificare se il nodo  $u$  è adiacente a  $v$  richiede tempo  $O(1)$

□ Elencare tutti gli archi costa  $O(n^2)$

## Liste di adiacenza

□ Spazio richiesto  $O(n+m)$

□ Verificare se il nodo  $u$  è adiacente a  $v$  richiede tempo  $O(n)$

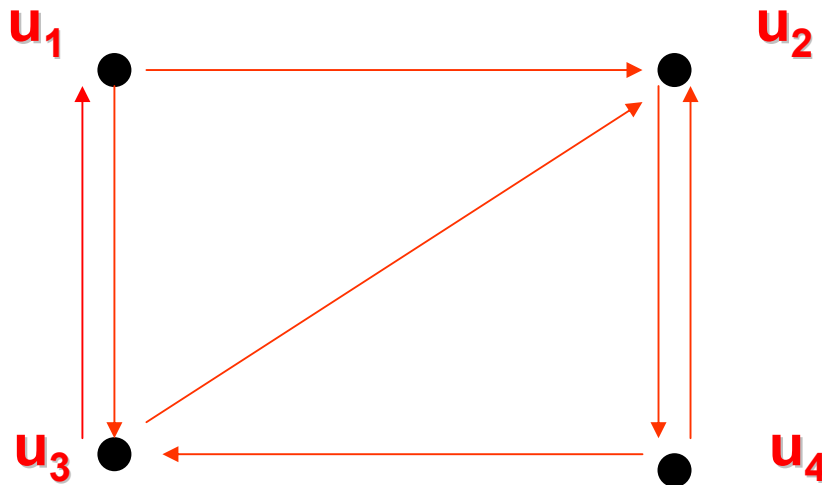
□ Elencare tutti gli archi costa  $O(n+m)$



# ESPLORAZIONE DI UN GRAFO

ESISTONO DEI METODI SISTEMATICI PER **ESPLORARE** UN GRAFO **“VISITANDO”** ALMENO UNA VOLTA OGNI NODO ED OGNI ARCO DI UN **GRAFO NON ORIENTATO E CONNESSO** OPPURE **ORIENTATO E FORTEMENTE CONNESSO**.

RICORDIAMO CHE **GRAFO CONNESSO** E' UN GRAFO  $G = \langle N, A \rangle$  IN CUI, DATI  $u$  E  $v \in N$  ESISTE UN CAMMINO DA  $u$  A  $v$  O UN CAMMINO DA  $v$  AD  $u$ .  $G$  E' DETTO **FORTEMENTE CONNESSO** SE PER OGNI COPPIA DI NODI  $u$  E  $v$  ESISTE ALMENO UN CAMMINO DA  $u$  A  $v$  ED ALMENO UN CAMMINO DA  $v$  AD  $u$ .



# Scopo e tipi di visita

- Una visita (o attraversamento) di un grafo  $G$  permette di esaminare i nodi e gli archi di  $G$  **in modo sistematico** (supporremo  $G$  connesso)
- Problema di base in molte applicazioni
- Esistono vari tipi di visite con diverse proprietà: in particolare, **visita in profondità** (DFS=depth first search) e **visita in ampiezza** (BFS=breadth first search)



# VISITA DI UN GRAFO

IL PRIMO SCHEMA DI VISITA E' QUELLO **IN PROFONDITA'**

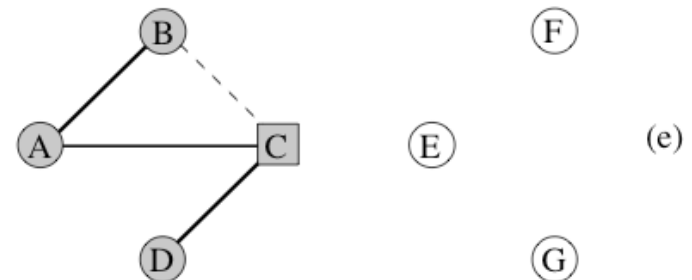
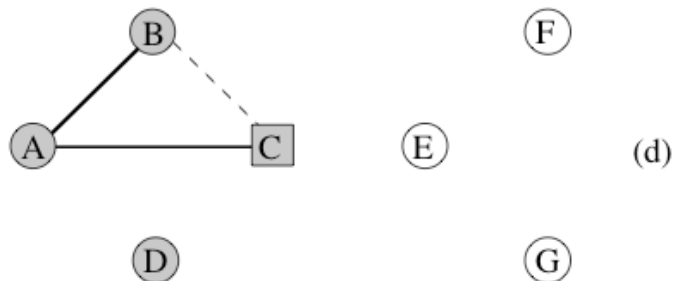
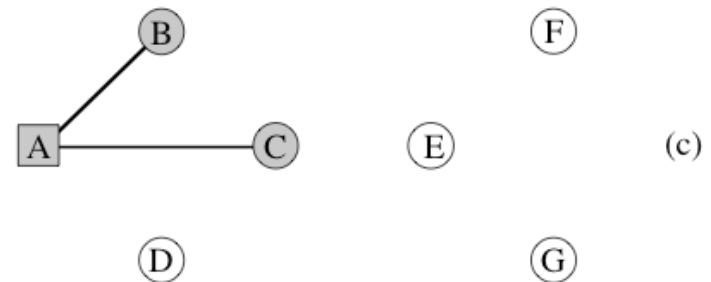
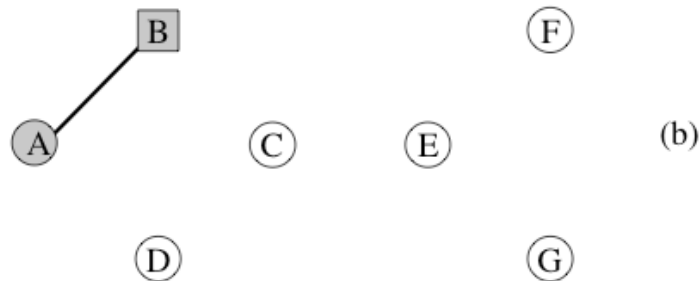
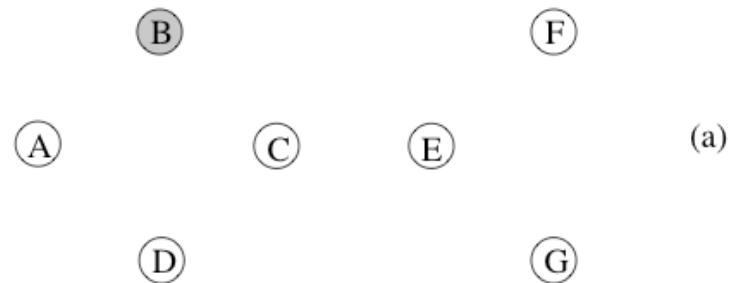
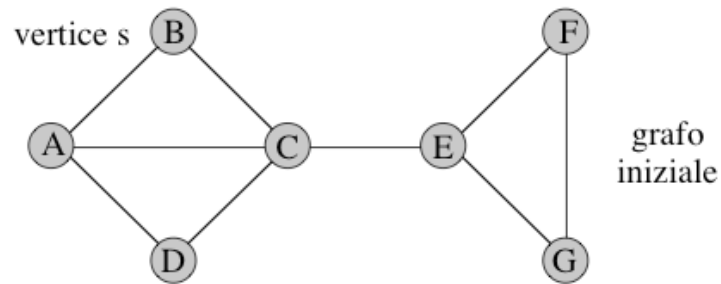
(**DFS** ovvero **DEPTH-FIRST-SEARCH**)

L'ALGORITMO PREVEDE CHE SI “**CONTRASSEGNI**” UN NODO APPENA LO SI VISITA E POI CI SI SPOSTI IN UN VERTICE ADIACENTE NON CONTRASSEGNAO.

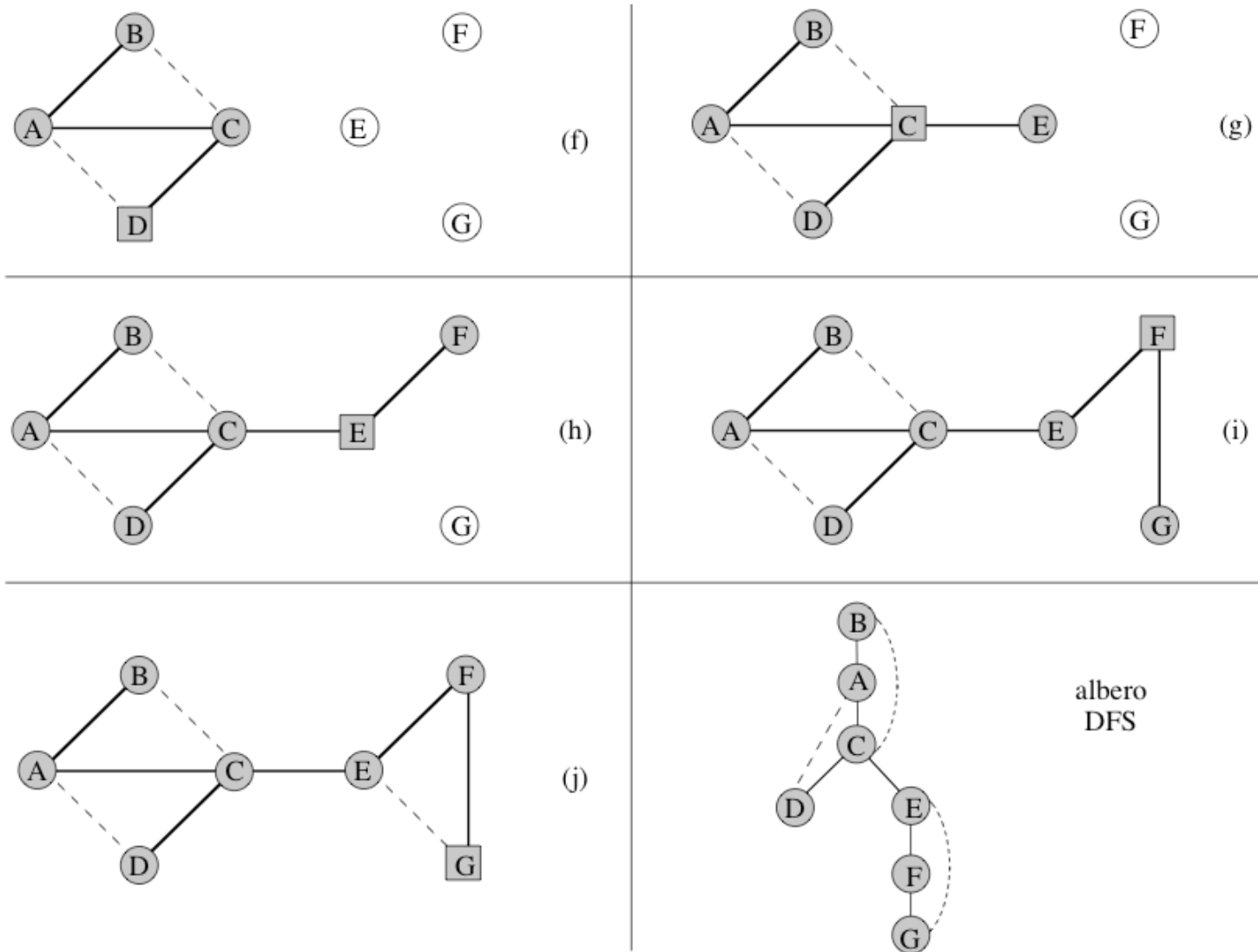
SE NON CI SONO NODI DA VISITARE (NON MARCATI) SI INDIETREGGIA LUNGO I NODI GIA' VISITATI FINCHE' SI ARRIVA AD UN NODO CHE RISULTA ADIACENTE AD UNO NON VISITATO E POI SI CONTINUA IL PROCEDIMENTO FINCHE' NON CI SONO PIU' NODI DA VISITARE



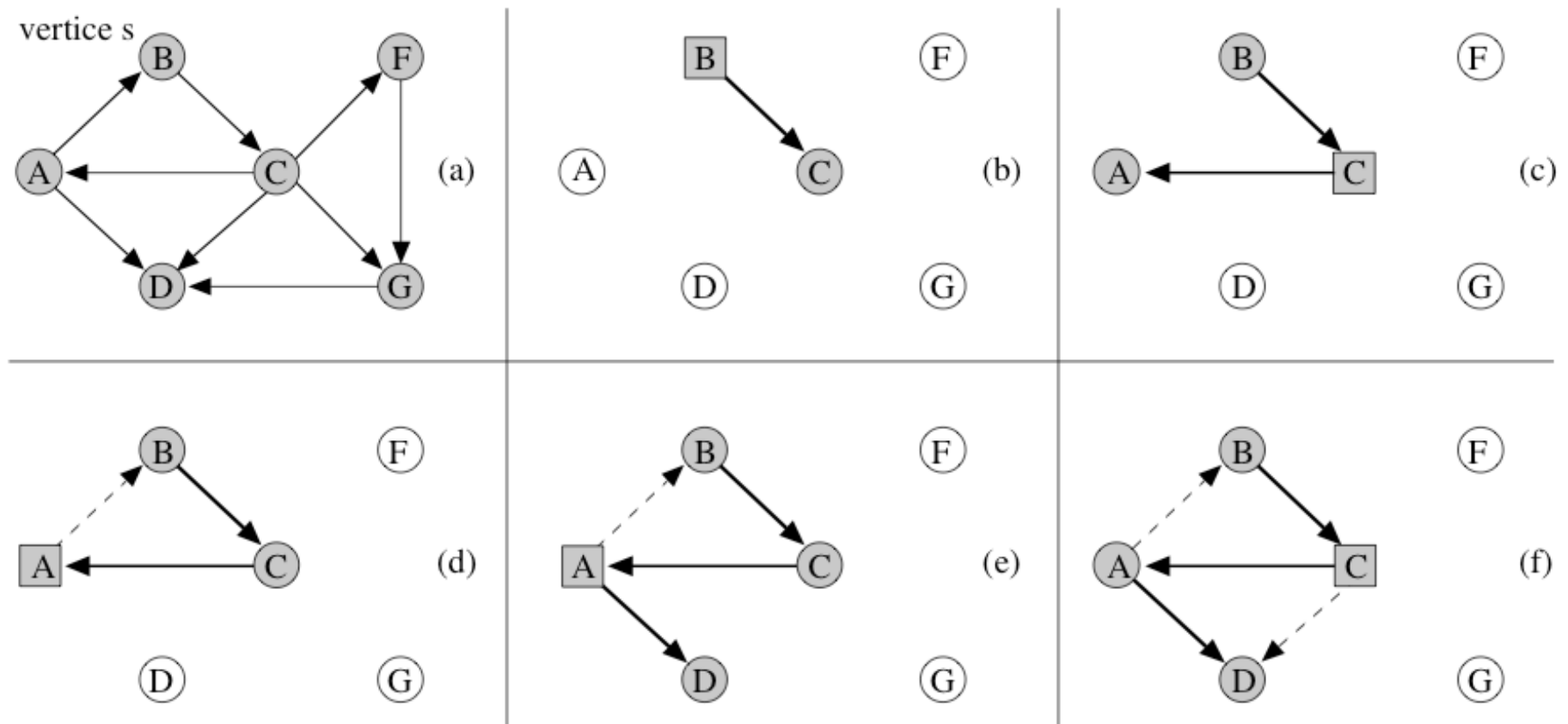
# Esempio (DFS): grafo non orientato (1/2)



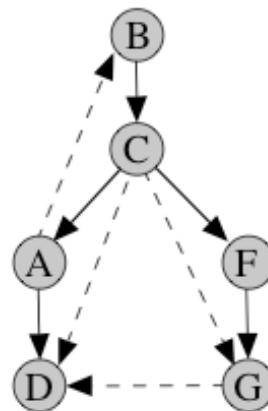
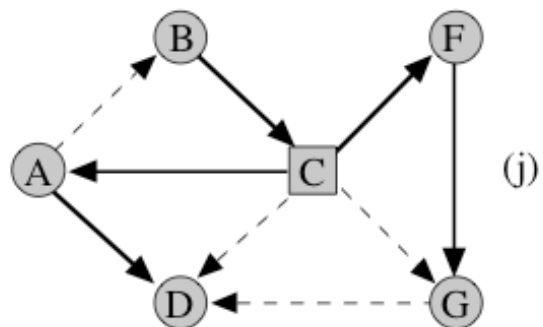
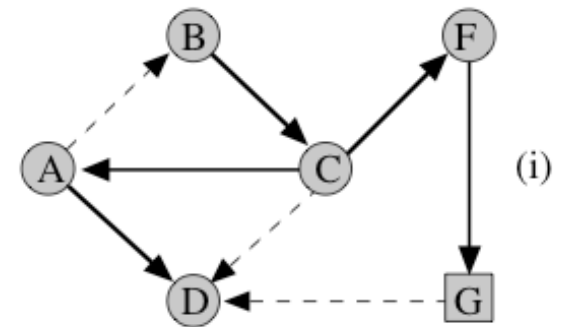
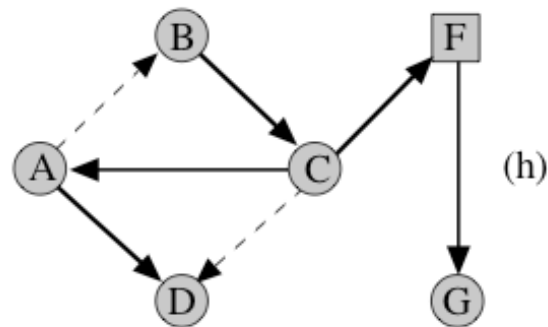
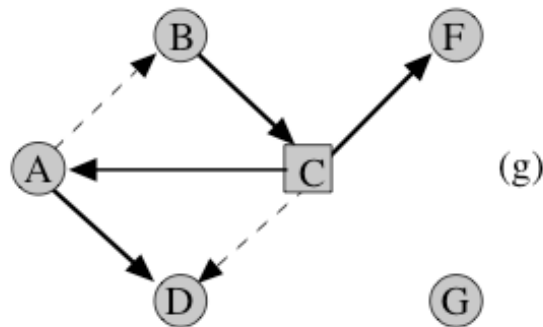
# Esempio (DFS): grafo non orientato (2/2)



# Esempio (DFS): grafo orientato (1/2)



# Esempio (DFS): grafo orientato (2/2)



archi in avanti: (C,D) e (C,G)

archi all'indietro: (A,B)

archi trasversali a sinistra: (G,D)

# Proprietà dell'albero DFS generato

- Sia  $(u,v)$  un arco di un **grafo non orientato**. Allora:
  - $(u,v)$  è un ramo dell'albero DFS,
  - i nodi  $u$  e  $v$  sono l'uno discendente/antenato dell'altro
- Sia  $(u,v)$  un arco di un **grafo orientato**. Allora:
  - $(u,v)$  è un ramo dell'albero DFS,
  - i nodi  $u$  e  $v$  sono l'uno discendente/antenato dell'altro,
  - $(u,v)$  è un arco **trasversale a sinistra**, ovvero il vertice  $v$  è in un sottoalbero visitato precedentemente ad  $u$





# L'ALGORITMO DELLA DFS

DFS (G di tipo GRAFO, u di tipo NODO)

ESAMINA IL NODO  $u$  E MARCALO “VISITATO”

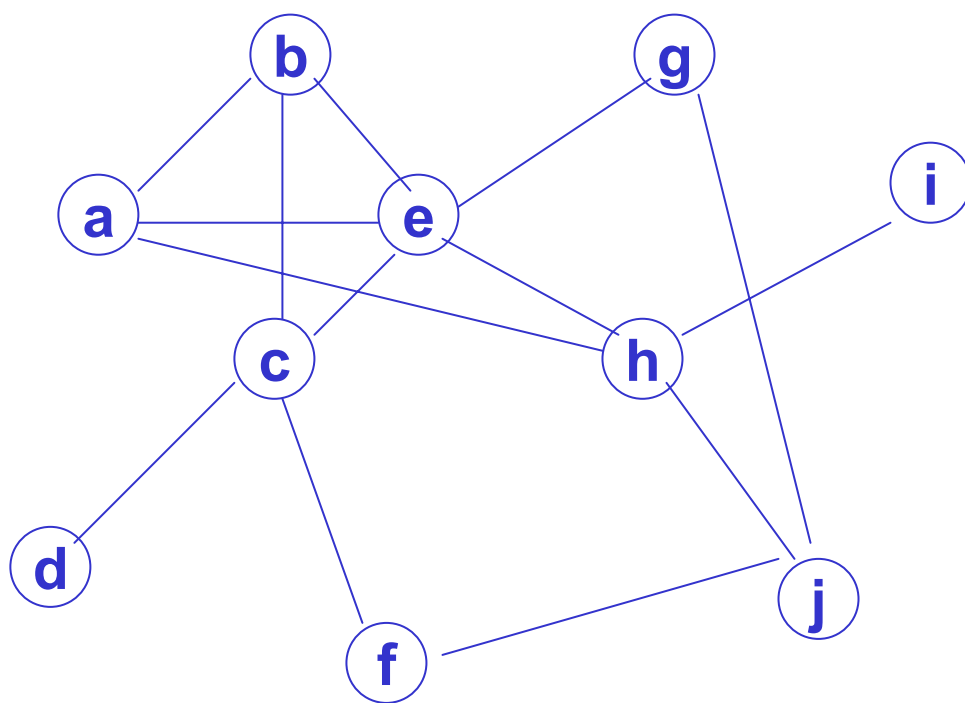
PER TUTTI I NODI ADIACENTI A  $u$  (*esplicitare il ciclo sulla lista Adiacenti  $(u, G)$* )

ESAMINA L'ARCO  $(u, v)$

**if  $v$  NON E' “VISITATO” then**

**DFS( $G, v$ )**





## LE CHIAMATE RICORSIVE

**a**

**AD{b,e,h}**

**b**

**AD{(a),c,e}**

**c**

**AD{(b),d,e,f}**

**d**

**AD{(c)}**

**e**

**AD{(a),(b),(c),g,h}**

**g**

**AD{(e),j}**

**j**

**AD{f,(g),h}**

**f**

**AD{(c),(j)}**

**h**

**AD{(a),(e),i,(j)}**

**i**

**AD{(h)}**



# Implementazione iterativa della DFS

- L'implementazione iterativa usa una pila per memorizzare gli archi uscenti da un nodo visitato.
- Ad ogni passo si estrae l'arco  $(v,u)$  sulla cima della pila.
- La visita prosegue dal nodo adiacente  $u$  se marcato non visitato.

# Costo della visita in profondità

Il tempo di esecuzione dipende dalla struttura dati usata per rappresentare il grafo (e dalla connettività o meno del grafo rispetto ad **s**):

- Liste di adiacenza:  $O(m+n)$
- Matrice di adiacenza:  $O(n^2)$



# LA VISITA IN AMPIEZZA

(BFS  $\equiv$  BREADTH-FIRST-SEARCH)

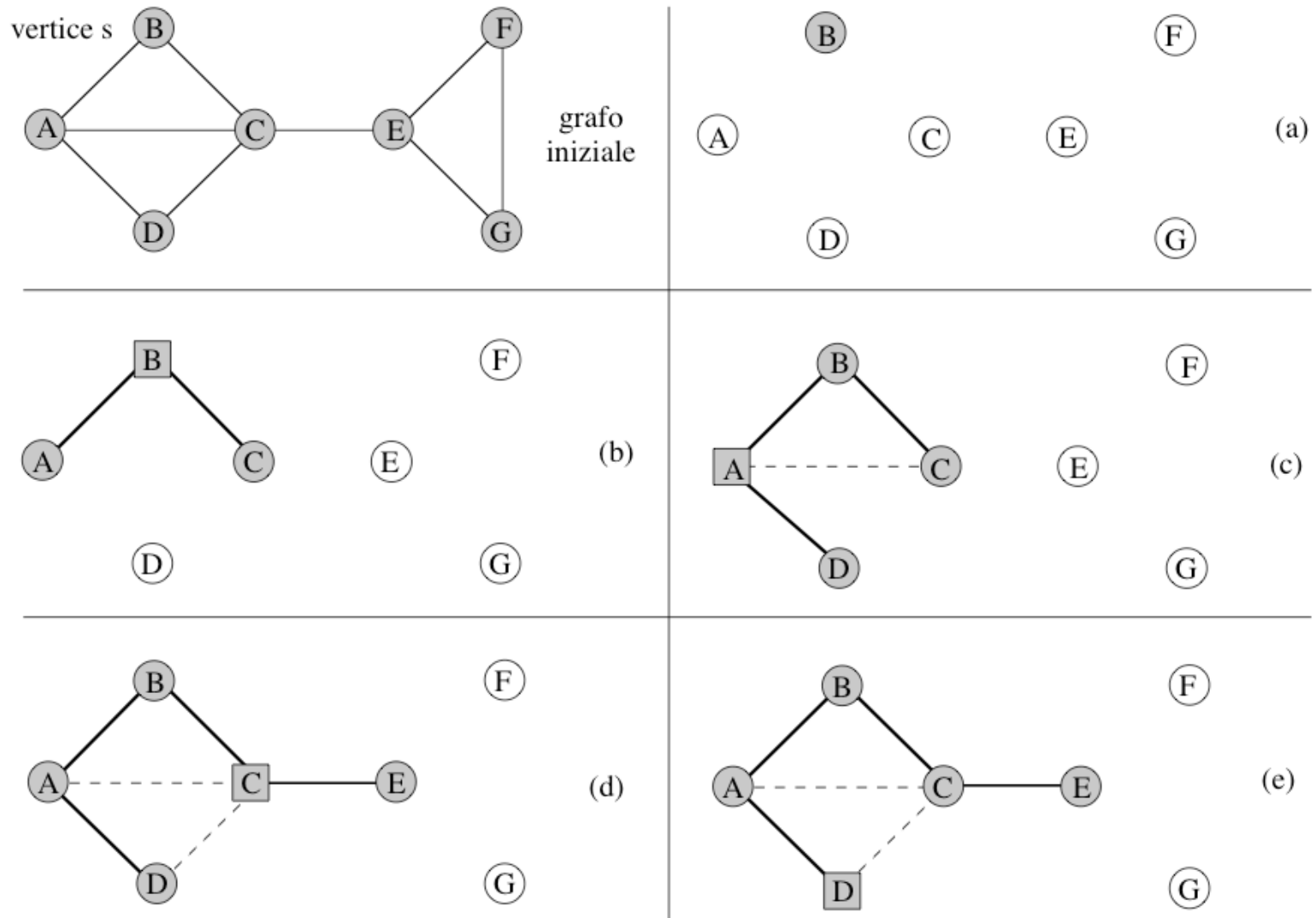
IN QUESTO SCHEMA TUTTI I NODI ADIACENTI AL NODO CORRENTE VENGONO VISITATI PRIMA DI SPOSTARSI DAL NODO CORRENTE STESSO.

I NODI SONO VISITATI IN ORDINE DI **DISTANZA** CRESCENTE DAL NODO DI PARTENZA  $u$ , DOVE LA **DISTANZA** DA  $u$  AD UN GENERICO NODO  $v$  E' IL **MINIMO NUMERO DI ARCHI IN UN CAMMINO DA  $u$  A  $v$ .**

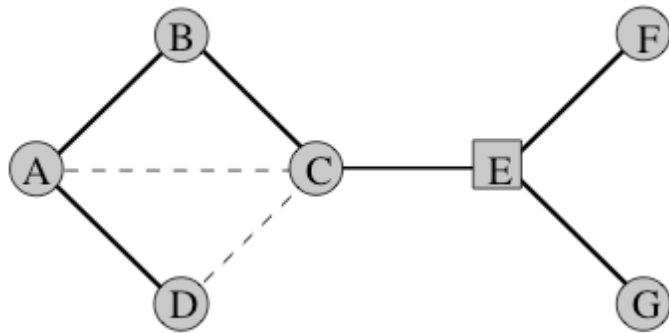
CONVIENE TENERE IN UNA CODA I NODI VISITATI MA NON COMPLETAMENTE “ESAMINATI” COSI' CHE, QUANDO SI E' PRONTI A PASSARE AD UN NODO ADIACENTE AL CORRENTE, SI PUO' RITORNARE AL VECCHIO NODO CORRENTE DOPO IL MOVIMENTO.



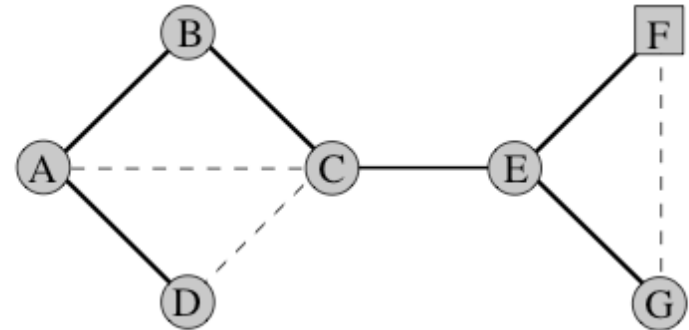
# Esempio (BFS): grafo non orientato (1/2)



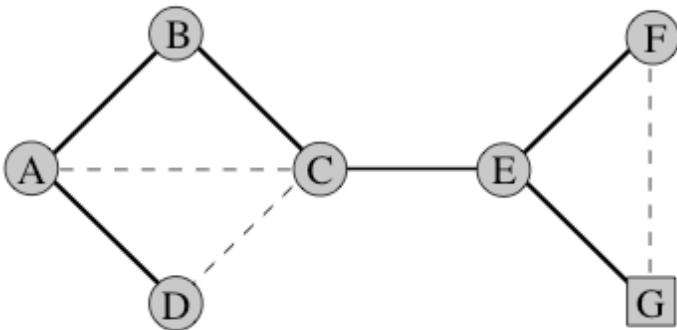
# Esempio (BFS): grafo non orientato (2/2)



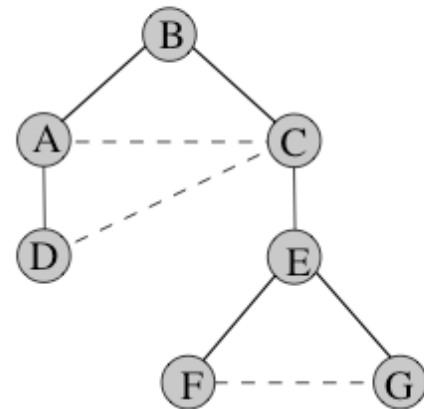
(f)



(g)



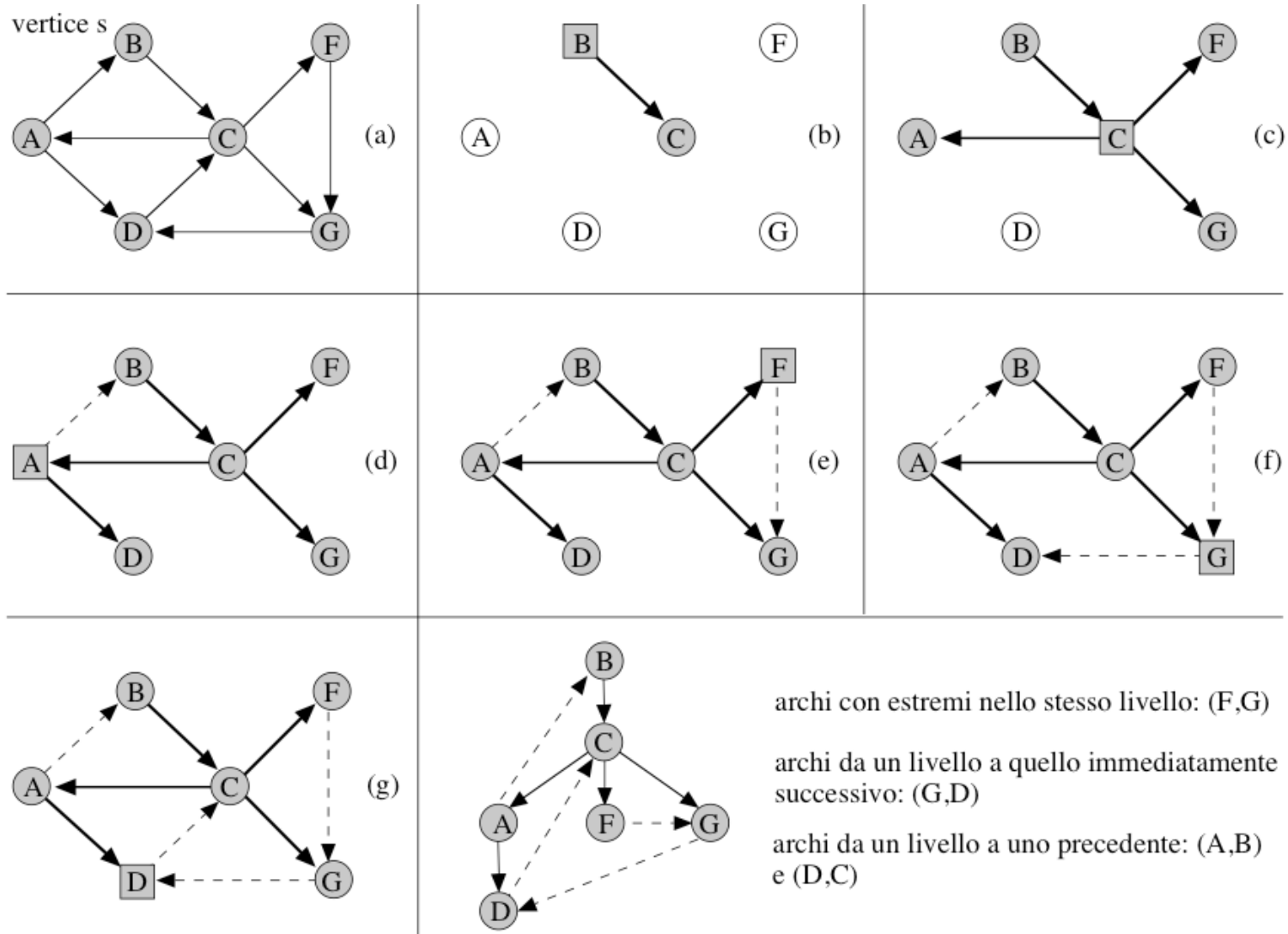
(h)



albero  
BFS



# Esempio (BFS): grafo orientato





# Proprietà dell'albero BFS generato

- ❑ Per ogni nodo  $v$ , il livello di  $v$  nell'albero BFS è pari alla distanza di  $v$  dalla sorgente  $s$
- ❑ Per ogni arco  $(u,v)$  di un **grafo non orientato**, gli estremi  $u$  e  $v$  appartengono allo stesso livello o a livelli consecutivi dell'albero BFS
- ❑ Se il **grafo è orientato**, possono esistere archi  $(u,v)$  che attraversano all'indietro più di un livello



# ALGORITMO DELLA BFS

BFS (G: GRAFO; u: NODO)

CREACODA(Q)

INCODA(u, Q)

while not CODAVUOTA(Q) do

u ← LEGGICODA(Q)

FUORICODA(Q)

*esamina u e marcalo "visitato"*

PER TUTTI I NODI v ADIACENTI A u (*corrisponde ad un ciclo sulla lista Adiacenti (u, G)*)

*esamina l'arco (u, v)*

if v non è marcato "visitato" and  $v \notin Q$  then

INCODA(v, Q)

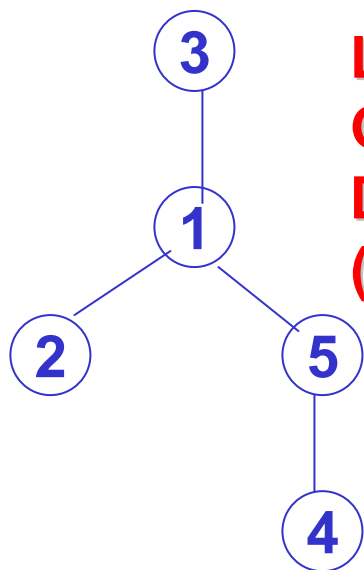
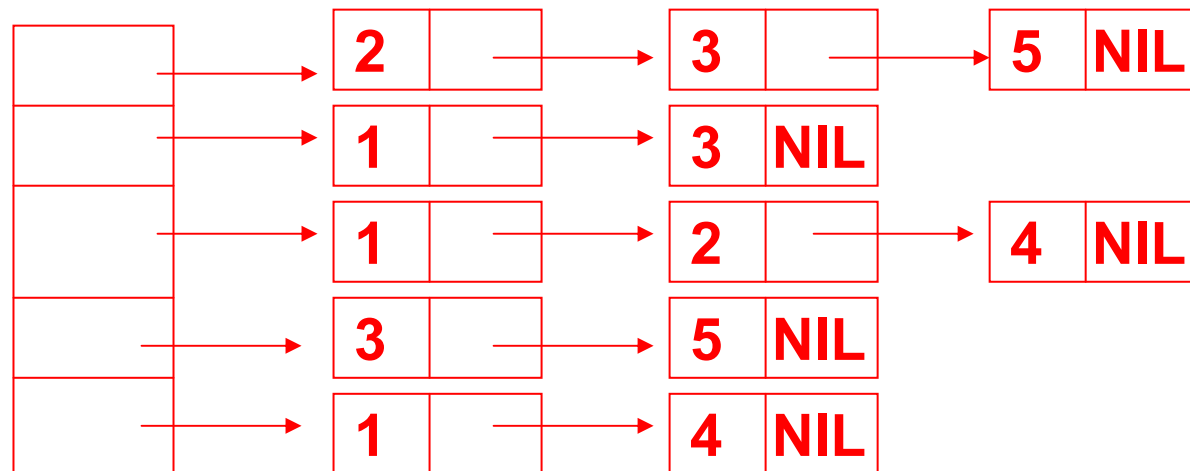
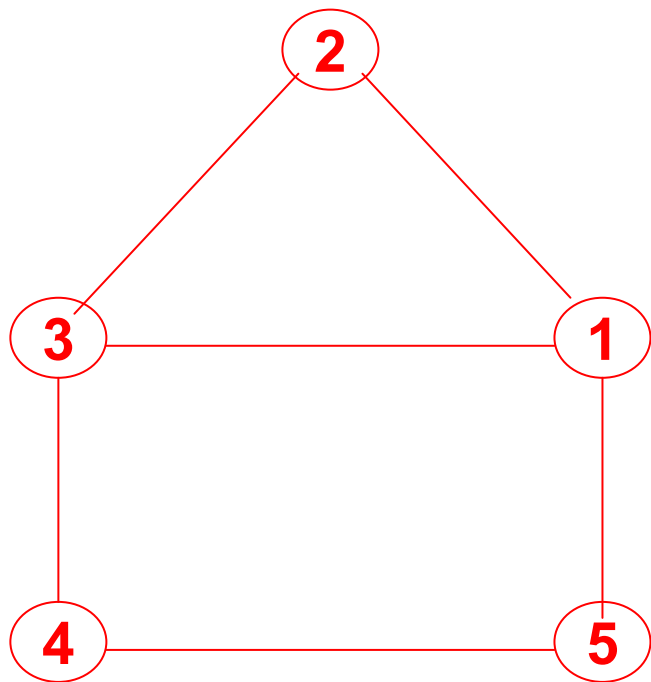


# Costo della visita in ampiezza

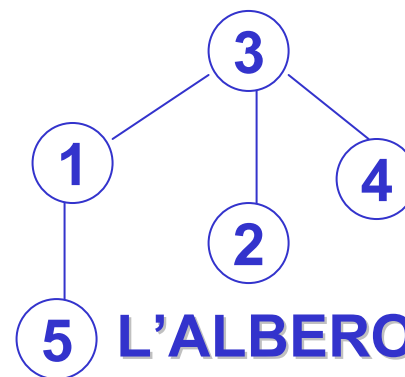
Il tempo di esecuzione dipende dalla struttura dati usata per rappresentare il grafo (e dalla connettività o meno del grafo rispetto ad **s**):

- Liste di adiacenza:  $O(m+n)$
- Matrice di adiacenza:  $O(n^2)$





**L'ALBERO  
GENERATO IN  
DEPTH-FIRST  
( $r = 2$ )**



**L'ALBERO  
GENERATO IN  
BREADTH-FIRST  
( $r = 3$ )**



LE PROCEDURE **DFS** E **BFS** SONO METODI DI VISITA SISTEMATICA CHE VENGONO USATI PER RISOLVERE MOLTI PROBLEMI. AD ESEMPIO:

- *VERIFICARE SE UN GRAFO NON ORIENTATO E' CONNESSO OPPURE NO*
- *TROVARE TUTTI I SOTTOGRAFI CONNESSI DI UN GRAFO NON ORIENTATO (COMPONENTI CONNESSE)*

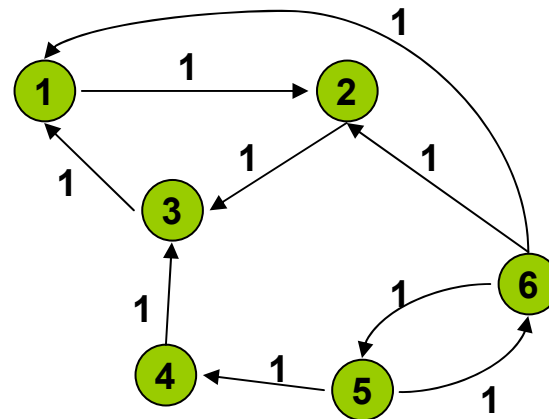
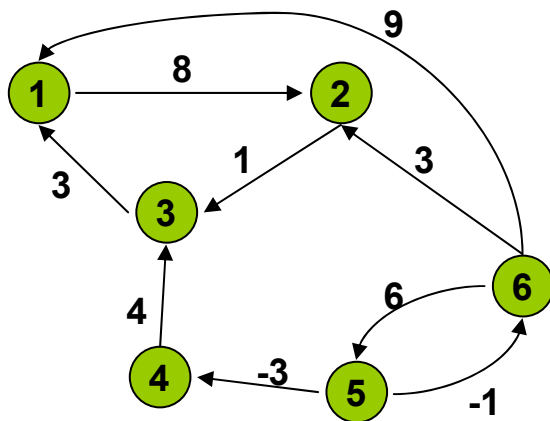
QUESTO RISULTA INTERESSANTE QUANDO SI VOGLIA RISOLVERE UN PROBLEMA CHE RICHIEDE DI SELEZIONARE, TRA TUTTI I PERCORSI CHE CONNETTONO DUE NODI IN UN GRAFO, QUELLO CHE RISULTA OTTIMO, AD ESEMPIO IN BASE AL CRITERIO DI MINIMIZZAZIONE DELLA SOMMA DEI PESI ASSOCIATI AGLI ARCHI.

*ESEMPIO: I NODI SONO CITTÀ, GLI ARCHI SONO LE STRADE CHE LE COLLEGANO, LE ETICHETTE SUGLI ARCHI SONO LE DISTANZE: TROVARE IL PERCORSO PIÙ BREVE TRA TUTTI QUELLI CHE CONNETTONO LA CITTÀ A ALLE ALTRE CITTÀ.*



# Problema dei cammini minimi

Input: un grafo  $G=(N,A)$  orientato e pesato, con una funzione peso  $w: A \rightarrow \mathbb{R}$ , che associa ad ogni arco in  $A$  un peso  $a$  con valore nei reali.



# Problema dei cammini minimi

Il **peso di un cammino**  $p = \langle n_1, n_2, \dots, n_k \rangle$  è la somma dei pesi degli archi che lo costituiscono.

$$w(p) = \sum_{i=2}^k w(n_{i-1}, n_i)$$

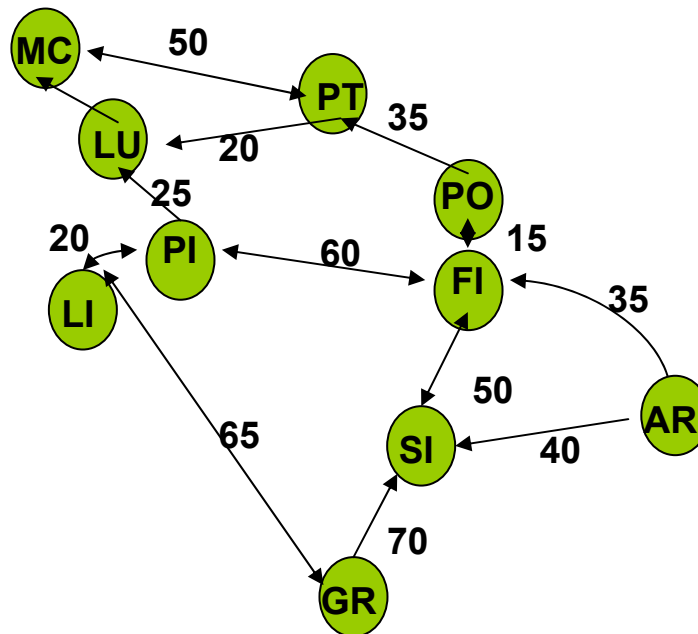
$$p_1 = \langle PO, FI, SI \rangle =$$

$$15 \text{ min} + 50 \text{ min} = 65 \text{ min}$$

$$p_2 = \langle PO, FI, AR, SI \rangle =$$

$$15 \text{ min} + 35 \text{ min} + 40 \text{ min} = 90 \text{ min}$$

$$p_3 = \langle PO, FI, PI, LI, GR, SI \rangle = 160 \text{ min}$$



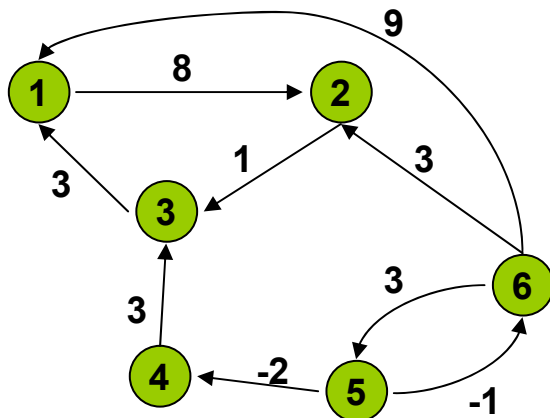
# Problema dei cammini minimi

Il **peso di un cammino minimo** dal nodo  $u$  al nodo  $v$  è definito da:

$$\delta(u, v) = \begin{cases} \min\{w(p) : u \xrightarrow{p} v\} & \text{se esiste un cammino da } u \text{ a } v \\ \infty & \text{altrimenti} \end{cases}$$

Un **cammino minimo** dal  $u$  al nodo  $v$  è definito come un qualunque cammino  $p$  con peso  $w(p) = \delta(u, v)$ .

**Può non essere unico!**



$$\delta(6, 1) = 7$$

$$p_1 = \langle 6, 2, 3, 1 \rangle$$

$$w(p_1) = 7$$

$$p_2 = \langle 6, 1 \rangle$$

$$w(p_2) = 9$$

$$p_3 = \langle 6, 5, 6, 2, 3, 1 \rangle$$

$$w(p_3) = 9$$

$$p_4 = \langle 6, 5, 4, 3, 1 \rangle$$

$$w(p_4) = 7$$





# Vari problemi

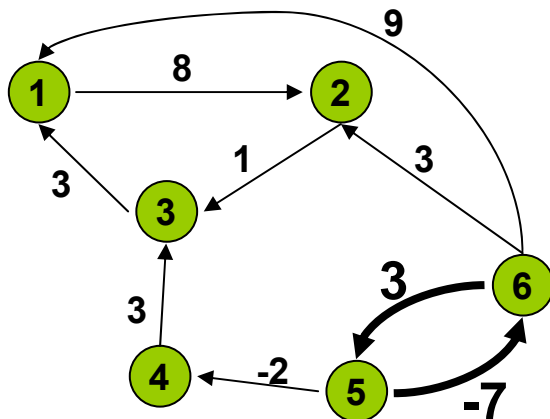
- **Problema di cammini minimi con sorgente singola:** si vuole trovare un cammino minimo da un dato nodo sorgente  $s$  ad ogni nodo  $v$  in  $N$ .
- **Problema di cammini minimi con destinazione singola:** si vuole trovare da ogni nodo  $v$  in  $N$  un cammino minimo ad un dato nodo destinazione  $t$ .
- **Problema di cammini minimi tra una coppia:** si vuole trovare un cammino minimo da  $u$  a  $v$ .
- **Problema di cammini minimi tra tutte le coppie:** determinazione di un cammino minimo per ogni coppia di nodi  $u$  e  $v$ .



# Archivi con pesi negativi

Un possibile problema può essere rappresentato dalla presenza di pesi negativi sugli archi e di **cicli che contengano archi con pesi negativi**.

Se il peso di un ciclo è negativo, allora tutti i nodi raggiungibili dal ciclo hanno **un cammino minimo infinitamente negativo ( $-\infty$ )**.



Ciclo  $\langle 6, 5 \rangle$  negativo.

Ogni volta che compio un giro diminuisco il peso del cammino che passa per il ciclo.

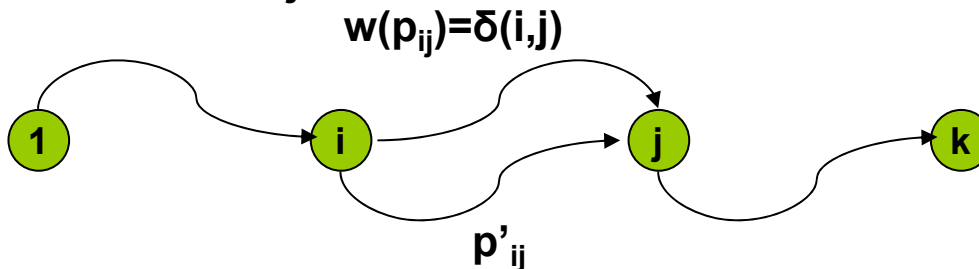
$$\delta(6, 1) = -\infty$$



# Sottostruttura ottima di un cammino minimo

## Sottocammini di cammini minimi sono cammini minimi

Dato un grafo  $G=(N,A)$  con funzione peso  $w:A \rightarrow R$ , sia  $p = \langle v_1, v_2, \dots, v_k \rangle$  un cammino minimo da  $v_1$  a  $v_k$ . Per ogni  $i$  e  $j$  tali che  $1 \leq i \leq j \leq k$ , ha che **il sottocammino  $p_{ij} = \langle v_i, v_{i+1}, \dots, v_j \rangle$  è un cammino minimo.**



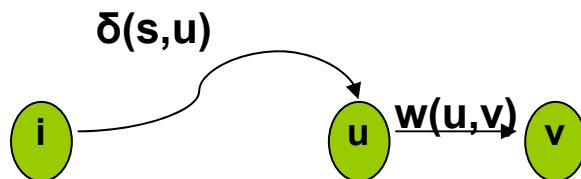
Dato un altro sottocammino da  $i$  a  $j$   $p'_{ij}$ , necessariamente  $w(p_{ij}) \leq w(p'_{ij})$ , altrimenti il cammino minimo passa per  $p'_{ij}$ .



# Sottostruttura ottima di un cammino minimo

**Di conseguenza:**

Si supponga che **un cammino minimo  $p$**  da un nodo sorgente ad un nodo  $v$  passi per l'arco  $(u,v)$  con peso  $w(u,v)$ . Il peso del cammino minimo da  $s$  a  $v$  è  **$\delta(s,v) = \delta(s,u) + w(u,v)$** .



cammino minimo tra  $u$  e  $v$

$$\delta(s,v) = \delta(s,u) + w(u,v).$$

Più **in generale**, se esiste un arco  $(u,v)$ , allora si ha:

$$\delta(s,v) \leq \delta(s,u) + w(u,v).$$



## IL METODO BASE

L'IDEA E' DI CALCOLARE, IN ORDINE CRESCENTE, LA LUNGHEZZA DEI CAMMINI MINIMI DA  $r$  A TUTTI I NODI DEL GRAFO.

INDICHIAMO CON  $S$  L'INSIEME DEI NODI DI CUI, AD UN DATO ISTANTE, SI È GIÀ CALCOLATO LA LUNGHEZZA DEL CAMMINO MINIMO DA  $r$ .

UTILIZZIAMO UN VETTORE  $DIST$  CON TANTE COMPONENTI QUANTI SONO I NODI DEL GRAFO, IN MODO CHE  $DIST(i)$  RAPPRESENTI LA LUNGHEZZA DEL CAMMINO MINIMO TRA QUELLI CHE VANNO DA  $r$  A  $i$  PASSANDO SOLO PER NODI CONTENUTI IN  $S$  (A PARTE  $i$  STESSO). L'IPOTESI DI FONDO E' CHE LE DISTANZE SIANO INTERI POSITIVI.

OSSERVIAMO CHE SE IL PROSSIMO CAMMINO MINIMO DA GENERARE  $C$  È DA  $r$  AL NODO  $u$ , TUTTI I NODI SONO IN  $S$ .



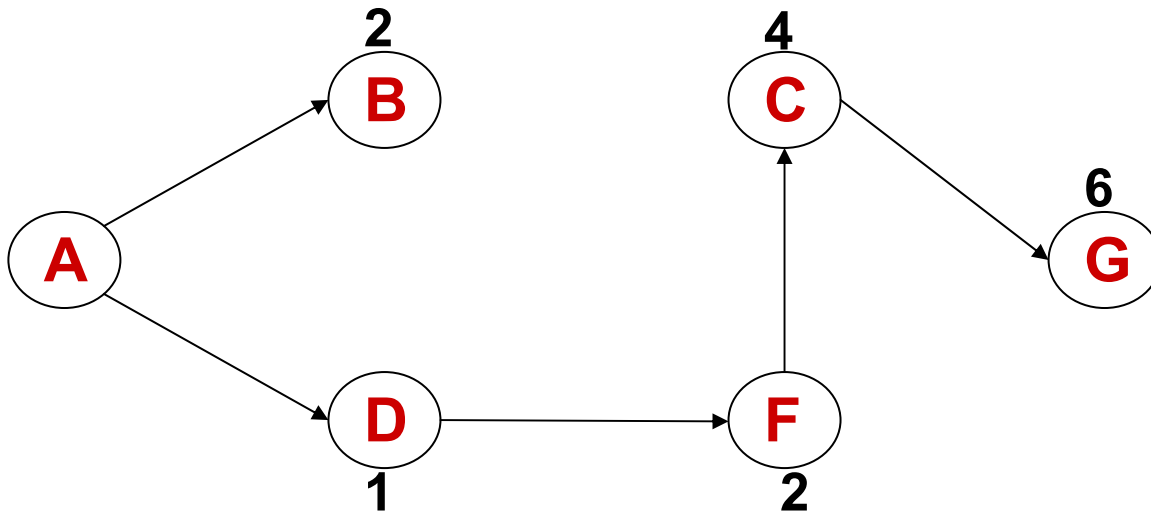
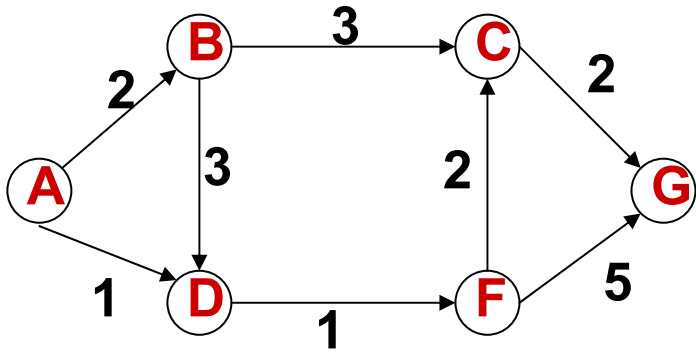
INFATTI SE UN NODO  $k$  DI  $C$  NON APPARTENESSE A  $S$  VI SAREBBE UN CAMMINO DA  $r$  A UN NODO  $k$  NON CONTENUTO IN  $S$  DI LUNGHEZZA MINORE A QUELLA DI  $C$ , CONTRADDICENDO L'IPOTESI CHE IL PROSSIMO CAMMINO DA GENERARE SIA  $C$ .

LA LUNGHEZZA DI  $C$  E IL NODO  $u$  SONO FACILMENTE INDIVIDUABILI; BASTA CALCOLARE IL VALORE MINIMO DI  $DIST(i)$  PER  $i \notin S$ .

INDIVIDUATO  $u$  SI INSERISCE IN  $S$  E SI AGGIORNA  $DIST$  PER I NODI CHE  $\notin S$ .

IN PARTICOLARE, SE PER UN CERTO NODO  $z$  CONNESSO A  $u$  DA  $\langle u, z \rangle$  CON PESO  $E$ , LA SOMMA  $DIST(u)+E$  È MINORE DI  $DIST(z)$  ALLORA A  $DIST(z)$  VA ASSEGNATO IL NUOVO VALORE  $DIST(u)+E$ .





VIENE GENERATO UN **ALBERO DI COPERTURA T**,  
RADICATO IN **r**, CHE INCLUDE UN CAMMINO DA **r** AD OGNI  
**ALTRO NODO**.

L'ALBERO RADICATO **T** PUÒ ESSERE RAPPRESENTATO  
CON UN VETTORE DI PADRI, INIZIALIZZATO AD UN  
**ALBERO "FITTIZIO"** IN CUI TUTTI I NODI SONO **FIGLI DI r**  
CONNESSI DA UN **ARCO FITTIZIO PESATO** CON UN  
VALORE MAGGIORE DI TUTTI GLI ALTRI PESI (**MAXINT**).

UNA **SOLUZIONE AMMISSIBILE T È OTTIMA SE E SOLO SE**

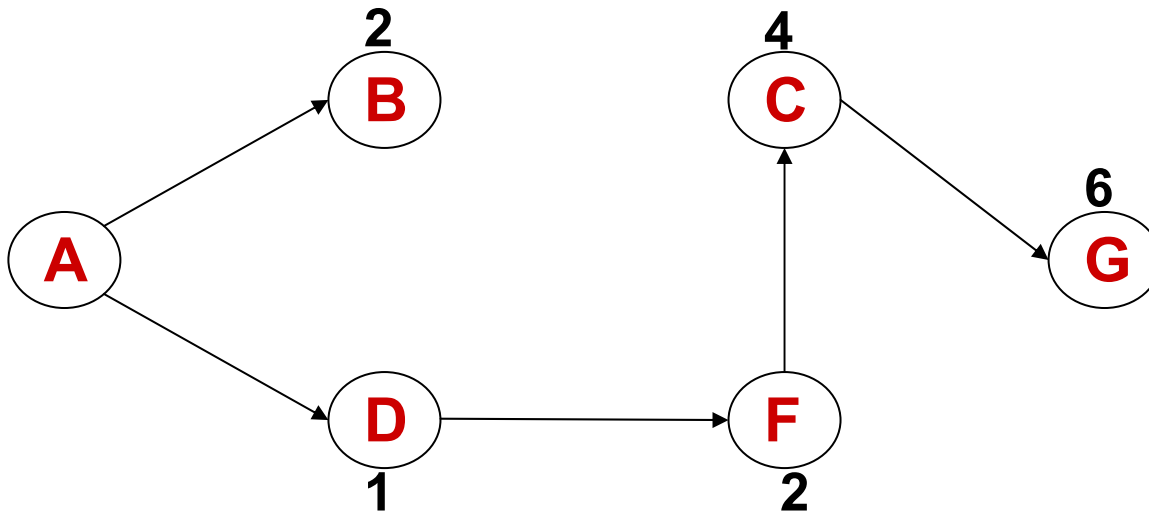
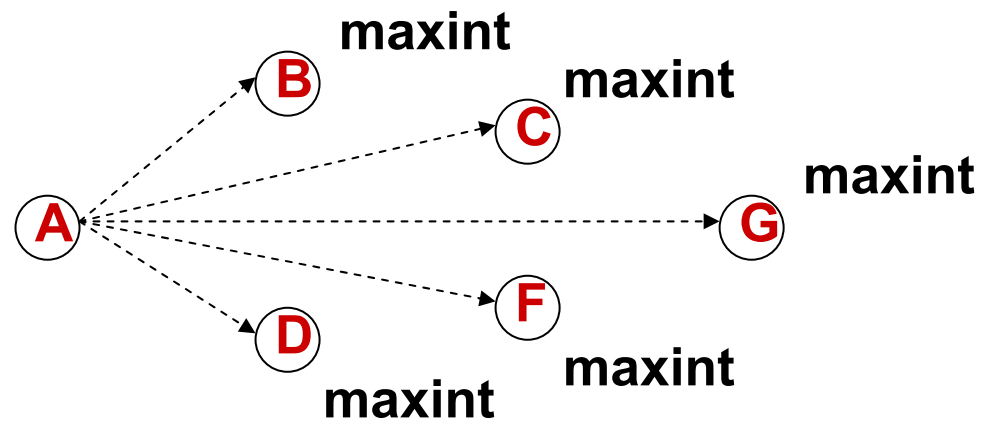
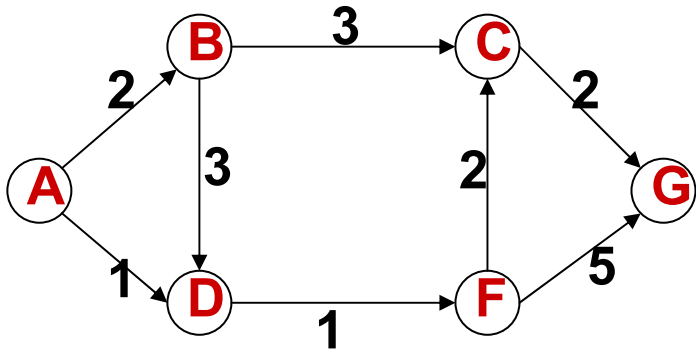
$$\begin{aligned} \text{DIST}(i) + C_{ij} &= \text{DIST}(j) \quad \forall (i,j) \in T \quad \text{E} \\ \text{DIST}(i) + C_{ij} &\geq \text{DIST}(j) \quad \forall \text{ARCO}(i,j) \in A \end{aligned}$$

**(CONDIZIONE DI BELLMAN)**

**NOTA: UN ALTRO ALGORITMO (BELLMAN-FORD) RISOLVE IL  
PROBLEMA DEI CAMMINI MINIMI NEL CASO PIU' GENERALE IN  
CUI I PESI DEGLI ARCHI POSSONO ESSERE NEGATIVI**







**PROCEDURA** Camminiminimi (G di tipo grafo, r di tipo nodo)

S di tipo insieme, T di tipo vettore, i e j di tipo nodo, K intero

CREAINSIEME (S)

$T(r) \leftarrow 0$ ;  $d(r) \leftarrow 0$ ;

**for** k=1 **to** n **do**

**if**  $k \neq r$  **then**

$T(k) \leftarrow r$ ;  $d(k) \leftarrow \maxint$

INSERISCI (r,S)

**while not** INSIEMEUVUOTO(S) **do**

$i \leftarrow \text{LEGGI}(S)$ ; CANCELLA (i, S);

**for each**  $j \in \text{ADIACENTI}(i, G)$  l'insieme degli adiacenti di i **do**

**if**  $d(j) + c_{ij} < d(j)$

$T(j) \leftarrow i$

$d(j) \leftarrow d(j) + c_{ij}$

**if not** APPARTIENE (j,S) **then**

            INSERISCI (j,S)

**end**



***anno 1959!!!***

# Algoritmo di Dijkstra

L'algoritmo di Dijkstra risolve **il problema dei cammini minimi con sorgente singola** su un grafo orientato e pesato  $G=(N,A)$  nel caso in cui **tutti i pesi** degli archi siano **non negativi**.

Ci sono due insiemi:

- **S**: dove  $d[v] = \delta(s,v)$ , quindi un cammino minimo tra  $s$  e  $v$  è stato determinato.
- **Q = A-S**: una coda a priorità dove  $d[v]$  ha il valore del cammino con peso minore finora scoperto.

**All'inizio**,  $S$  contiene solo  $s$ ,  $d[s]=0$ , mentre  $Q=A-\{s\}$  con  $d[v]=\infty$ .

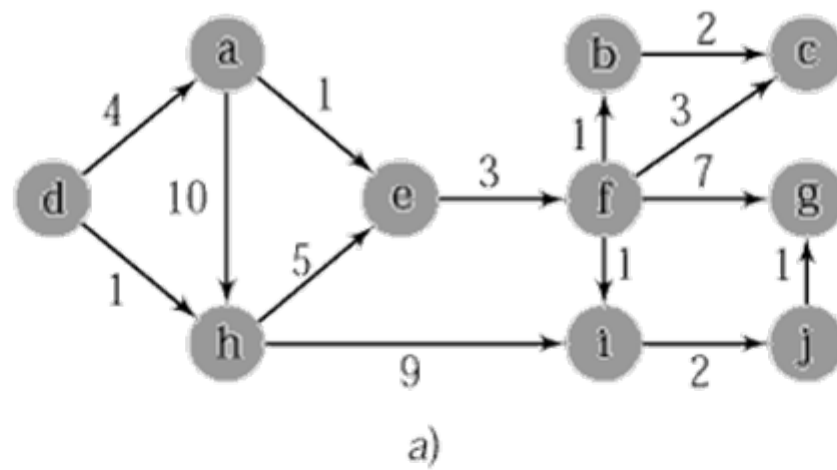


# Algoritmo di Dijkstra

DIJKSTRA( $G, w, s$ )

1.     **for** ogni nodo  $u$  in  $N[G]$                      *// inizializzazione di ogni nodo*
2.         **do**  $d[u] \leftarrow \infty$
3.              $p[u] \leftarrow NIL$                      *// predecessore di  $u$  indefinito*
4.      $d[s] \leftarrow 0$                      *// si comincia dal nodo  $s$*
5.      $Q \leftarrow N[G]$                      *// coda a priorità*
6.      $S \leftarrow \emptyset$                      *// insiemi dei cammini minimi trovati*
7.     **while**  $Q \neq \emptyset$                      *// termina quando la coda  $Q$  è vuota*
8.         **do**  $u \leftarrow \text{EXTRACT-MIN}(Q)$              *// prendi il cammino in  $Q$  più piccolo*
9.              $S \leftarrow S \cup \{u\}$                      *// inserisci  $u$  in  $S$*
10.             **for** ogni nodo  $v$  tra gli adiacenti di  $u$  considera l'arco  $(u, v)$   
               *// aggiorna cammini minimi in  $Q$  con  $v$  adiacente a  $u$*
11.                 **do if**  $d[v] > d[u] + w(u, v)$
12.                     **then**  $d[v] \leftarrow d[u] + w[u, v]$
13.                      $p[v] \leftarrow u$





	iterazione:	init	1	2	3	4	5	6	7	8	9	10
	vertice attivo:		d	h	a	e	f	b	i	c	j	g
a		$\infty$	4	4								
b		$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	9					
c		$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	11	11	11			
d		0										
e		$\infty$	$\infty$	6	5							
f		$\infty$	$\infty$	$\infty$	$\infty$	8						
g		$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	15	15	15	15	12	
h		$\infty$	1									
i		$\infty$	$\infty$	10	10	10	9	9				
j		$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	11	11		

b)

Un'esecuzione  
di  
DijkstraAlgorithm



## UN ALTRO PROBLEMA:

### PROBLEMA DEL MINIMO ALBERO DI COPERTURA

**DATO UN GRAFO NON ORIENTATO E CONNESSO  $G=(N,A)$ , CON PESI (NON NEGATIVI) SUGLI ARCHI, TROVARE UN ALBERO DI COPERTURA PER  $G$ , CIOÈ UN ALBERO AVENTE TUTTI I NODI IN  $N$ , MA SOLO ALCUNI ARCHI IN  $A$ , IN MODO TALE CHE SIA MINIMA LA SOMMA DEI PESI ASSOCIATI AGLI ARCHI.**

