Corso di Laurea in INFORMATICA

Algoritmi e Strutture Dati

a.a. 2012-2013

MODULO 12

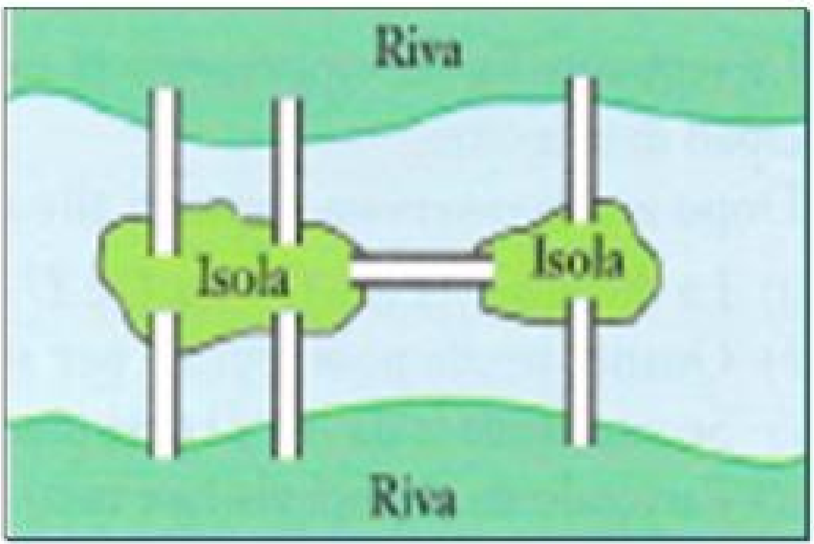
STRUTTURE NON LINEARI

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

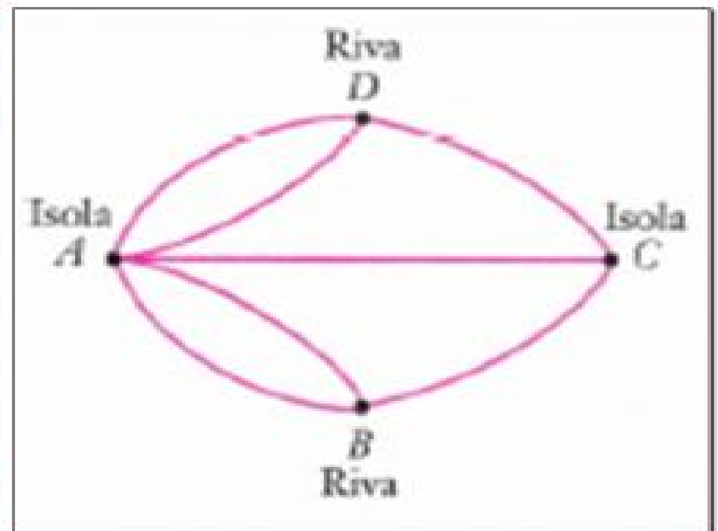
Grafi

Lo studio dei grafi risale a una curiosità matematica: il problema dei ponti di KÖNISBERG, una città attraversata dal fiume Prevel. Nel mezzo del fiume vi sono due isole collegate alla terraferma da ponti e tra loro da un altro un ponte. Il problema posto da Eulero era quello di determinare se fosse possibile attraversare tutti i ponti una sola volta e tornare al punto di partenza.

I ponti di KÖNISBERG



I sette ponti di Königsberg



Schematizzazione mediante una rete topologica

Esempi di grafi esistenti ‘in natura’

- Una carta stradale può essere presentata come un grafo i cui nodi sono le città e i cui archi sono le strade fra una città ed un’altra.
- Una molecola può essere rappresentata come un grafo i cui nodi sono gli atomi che la compongono e i cui spigoli sono i legami fra gli atomi stessi, determinati dalle valenze.
- Un circuito elettrico può essere rappresentato come un grafo in cui i nodi sono i componenti (generatori, resistenze, interruttori) e i cui archi sono i fili elettrici fra un componente e l’altro.

Definizione di Grafo

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

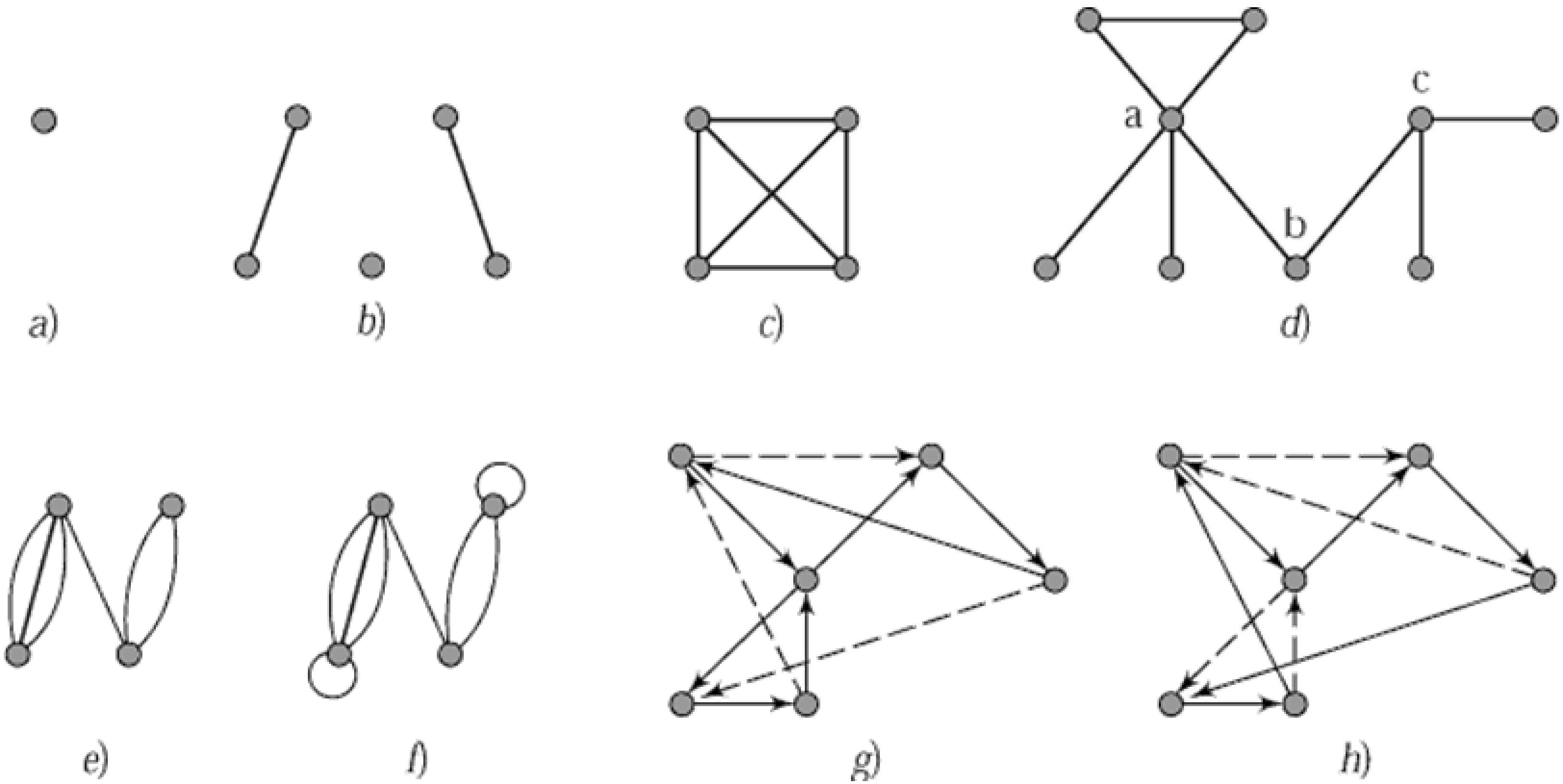
In generale i nodi sono usati per rappresentare entità e gli archi per rappresentare relazioni tra entità

Esempio 1: N={persone che vivono in Italia}, A={coppie {x,y} tali che x e y si sono stretta la mano}

Esempio 2: N={persone che vivono in Italia}, A={coppie (x,y) tale che x ha inviato una mail a y}

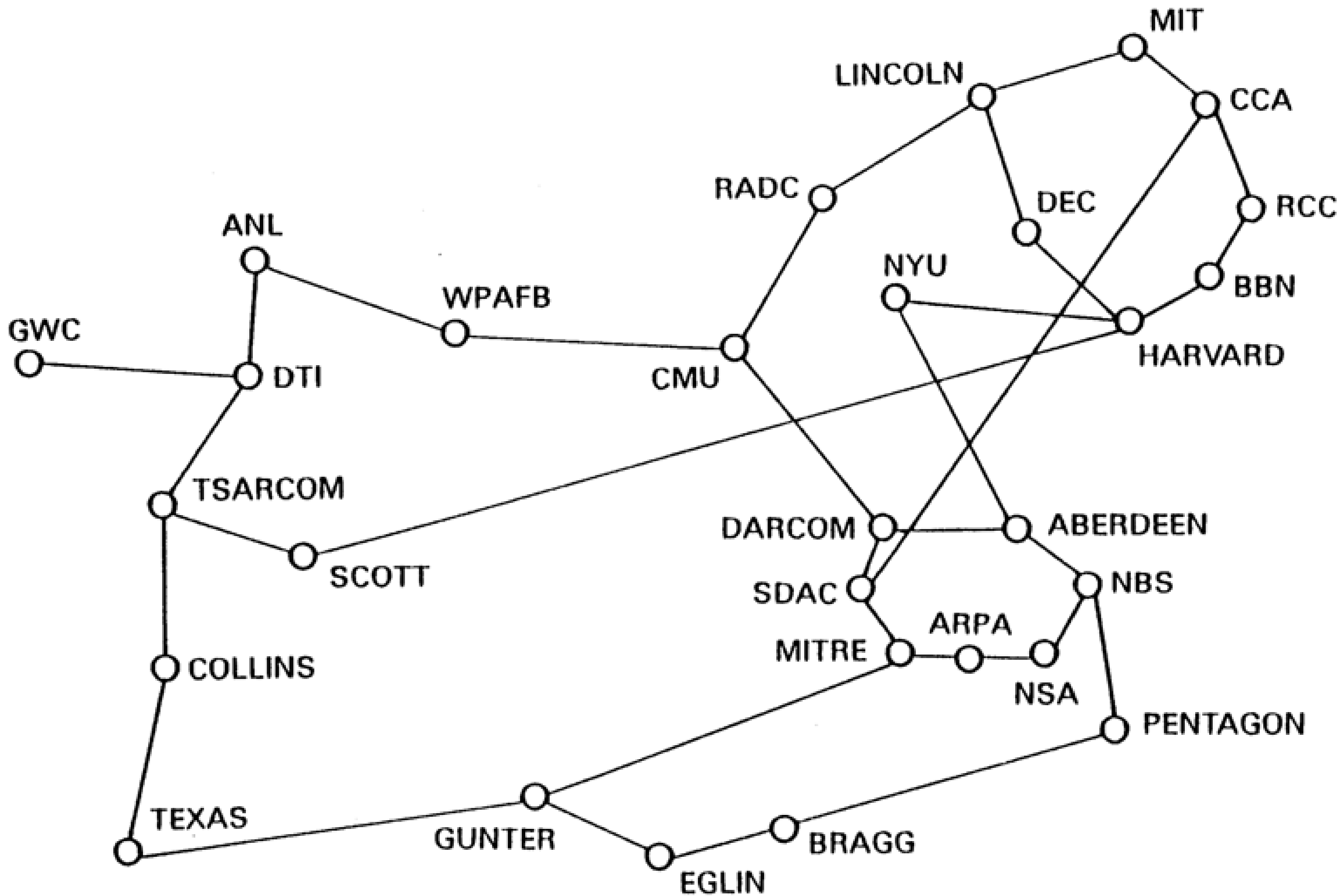
Esempi

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

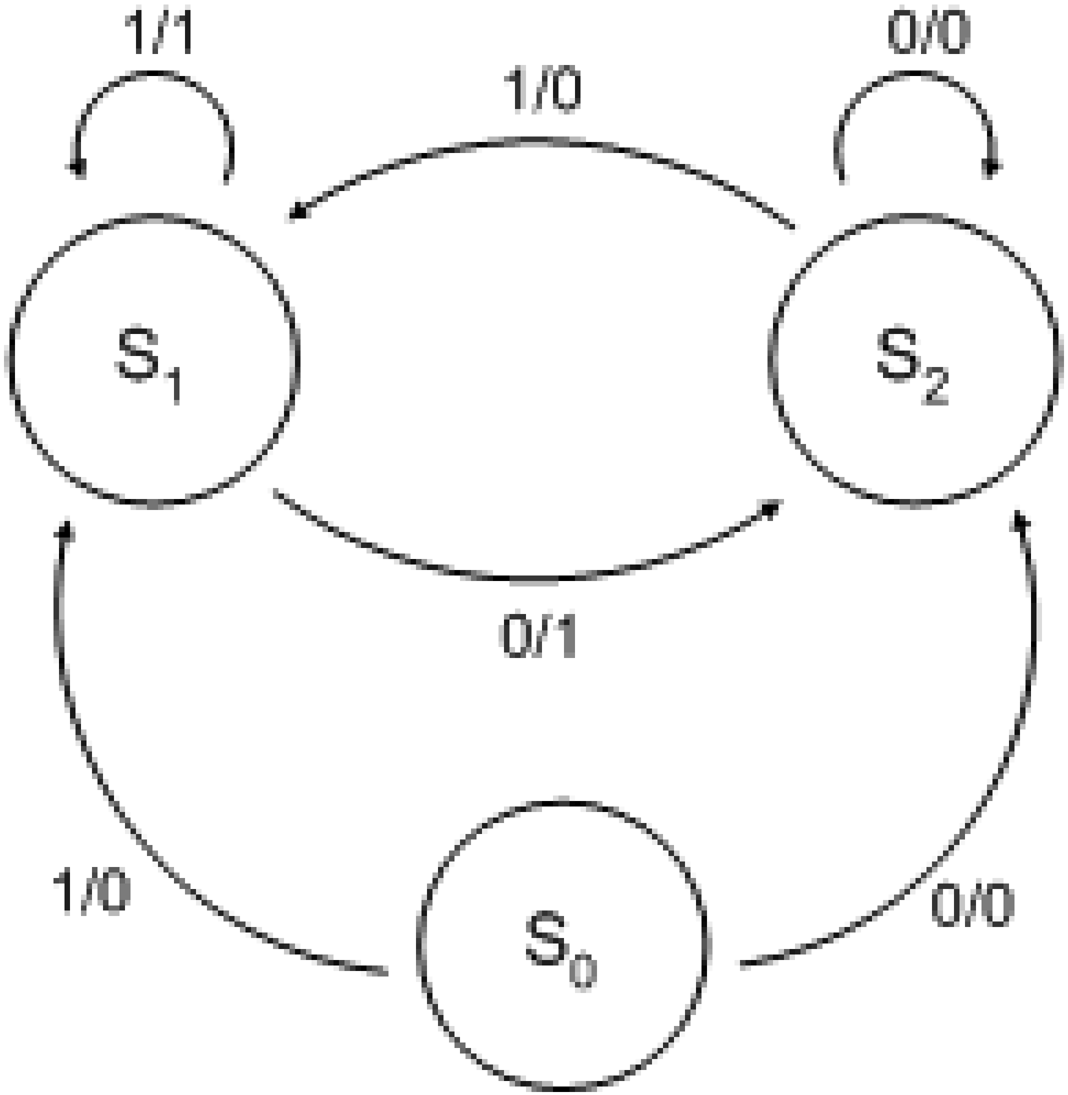


Esempi di grafi

UN ESEMPIO DI GRAFO



Parte della mappa di ARPANET.



Grafo di rappresentazione di un automa di Mealy

GRAFI E ALBERI

LA DIFFERENZA TRA LE DUE STRUTTURE:

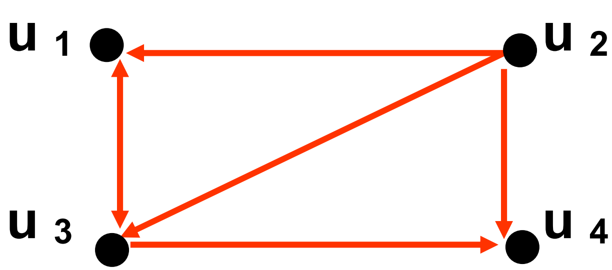
- **GENERALMENTE NELL'ALBERO VI E' UNA DIREZIONALITA' CHE UTILIZZIAMO PER RAPPRESENTARE PARTIZIONI SUCCESSIVE O TASSONOMIE GERARCHICHE.**

- **NEL GRAFO QUESTA DIREZIONALITA', SE ESISTE, NON E' PREDEFINITA NE' UTILIZZATA PER RAPPRESENTARE UN RANGO NELLA ORGANIZZAZIONE DEI DATI MA PIUTTOSTO LA DIREZIONE DELLA RELAZIONE TRA I NODI COLLEGATI.**

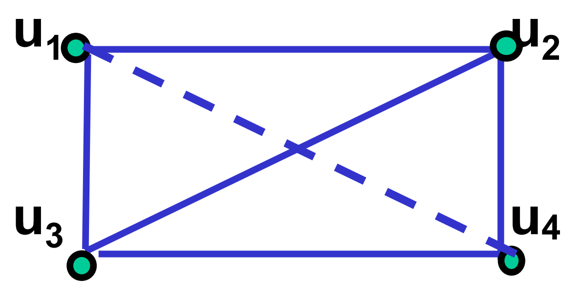
TUTTI GLI ELEMENTI (*nodi*) DEL GRAFO SONO SULLO STESSO PIANO E LA STRUTTURA DATI RAPPRESENTA L'ESISTENZA DI UNA CONNESSIONE TRA ELEMENTI.

GRAFI ORIENTATI E NON ORIENTATI

IN UN GRAFO ORIENTATO (u_i,u_j) e (u_j,u_i) INDICANO DUE ARCHI DISTINTI, IN UN GRAFO NON ORIENTATO INDICANO LO STESSO ARCO CHE INCIDE SUI DUE NODI.



1. Grafo orientato

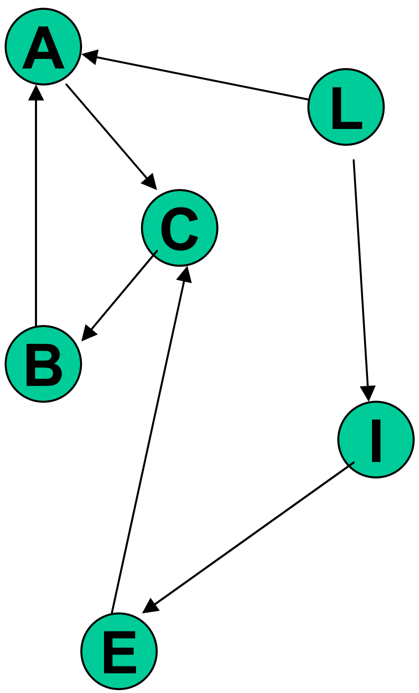


2. Grafo non orientato

IN UN GRAFO NON ORIENTATO I NODI CONGIUNTI DA UN ARCO SONO DETTI ADIACENTI. NELL'ESEMPIO I NODI u_1 E u_3 SONO ADIACENTI MA u_1 E u_4 NON LO SONO.

UN GRAFO E' DETTO COMPLETO SE PER OGNI COPPIA DI NODI $u_i, u_j \in N$ ESISTE UN ARCO CHE VA DA u_i AD u_j, APPARTENENTE A $(A = N \times N)$.

Grafo orientato (Directed Graph) Terminologia



< L , I , E, C, B, A > è un cammino nel grafo orientato di lunghezza 5

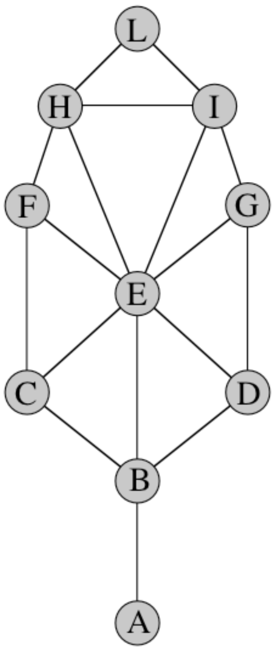
Il cammino deve rispettare il verso di orientamento degli archi

<A, C, B> 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

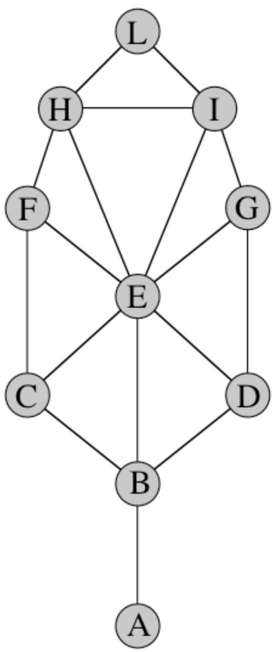


n = numero di nodi
m = numero di archi

L ed I sono adiacenti

(L,I) è incidente ad L e ad I

Grafo non orientato: Terminologia



< L , I , E, C, B, A > 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

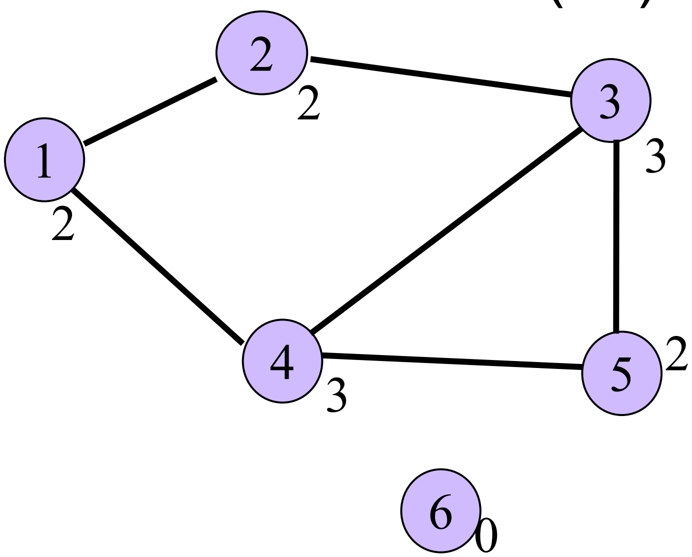
Grado del Grafo

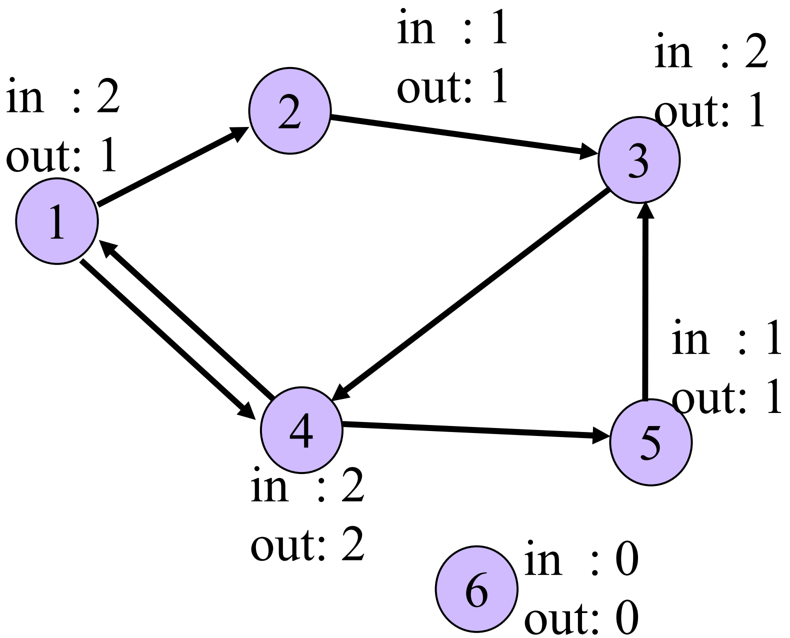
In un grafo non orientato

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

In un grafo orientato

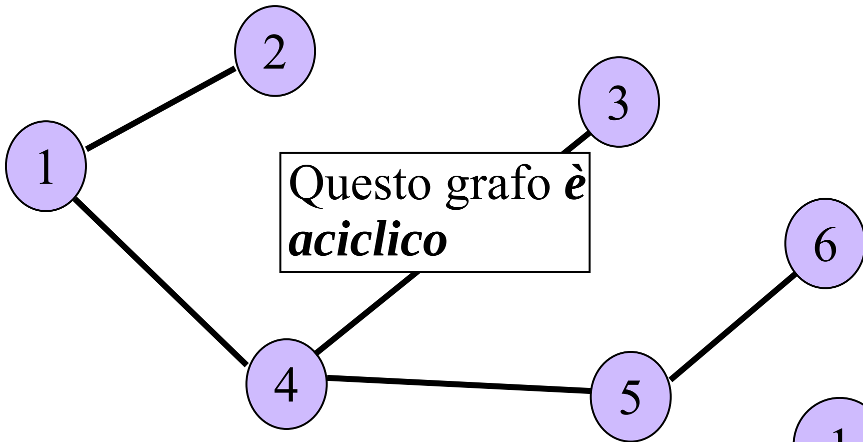
- il *grado entrante* (*uscente*) di un vertice è il numero di archi incidenti in (da) esso



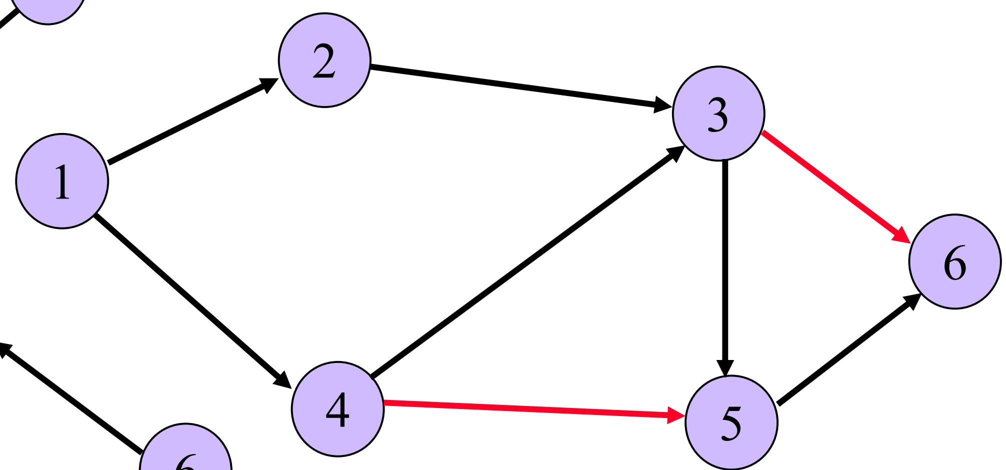


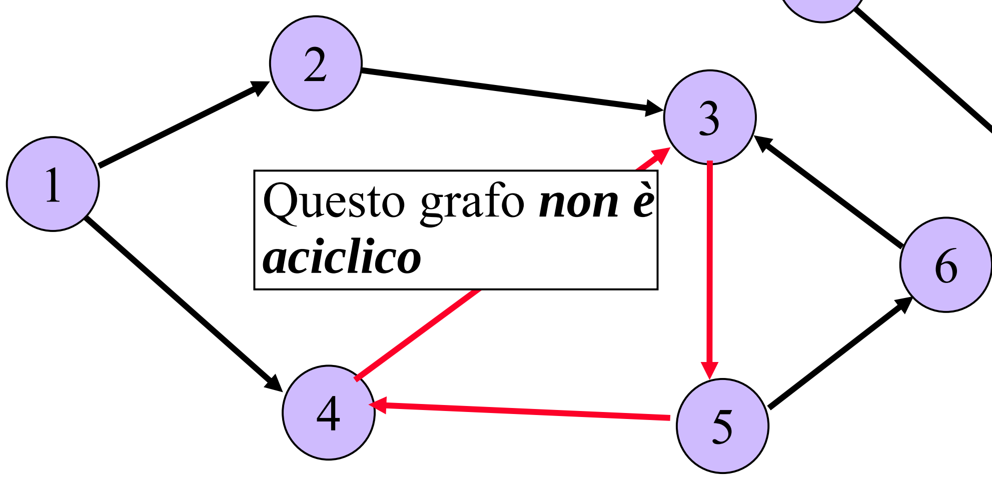
Grafi aciclici

Un grafo senza cicli è detto aciclico



Un ***grafo orientato aciclico*** è chiamato **DAG (*<u>D</u>irected <u>A</u>cyclic <u>G</u>raph*)**

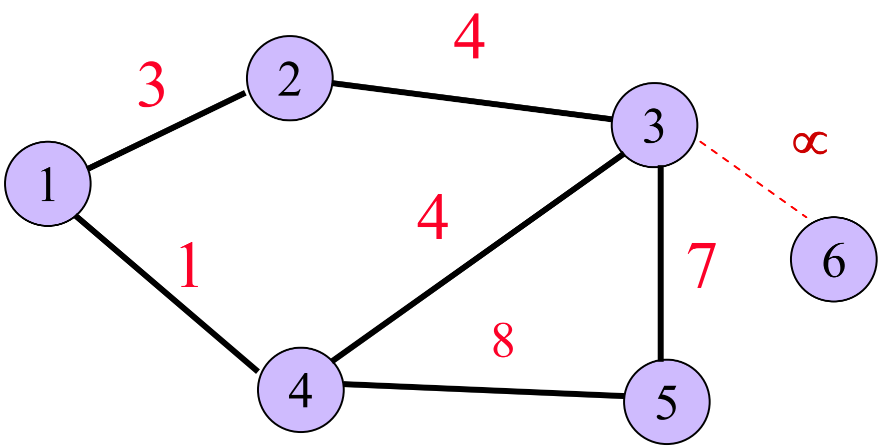




Grafi pesati

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

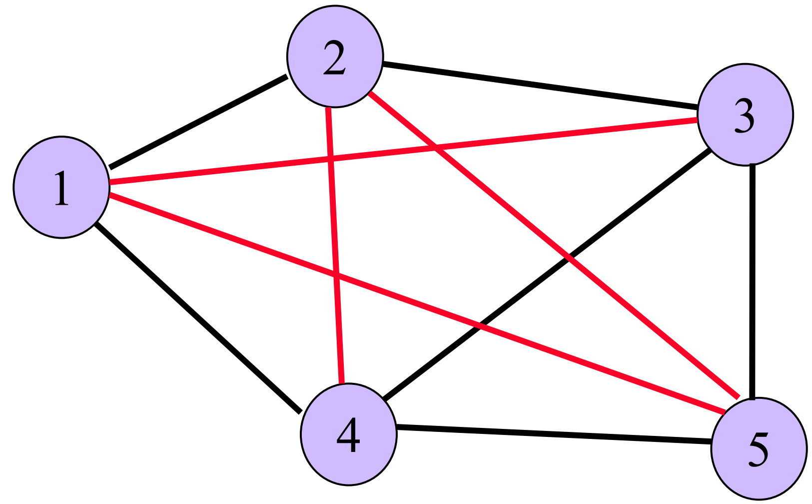
- Il peso può essere determinato tramite una funzione di costo
  p: $N \times N \rightarrow$ R, dove R è l'insieme dei numeri reali
- Quando tra due vertici non esiste un arco, il peso è infinito



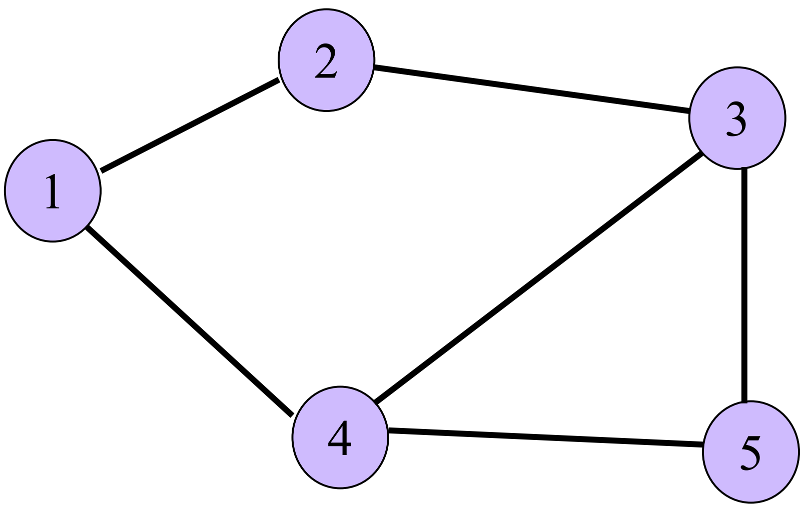
Grafo completo

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

Questo grafo ***è completo***



Questo grafo ***non è completo***

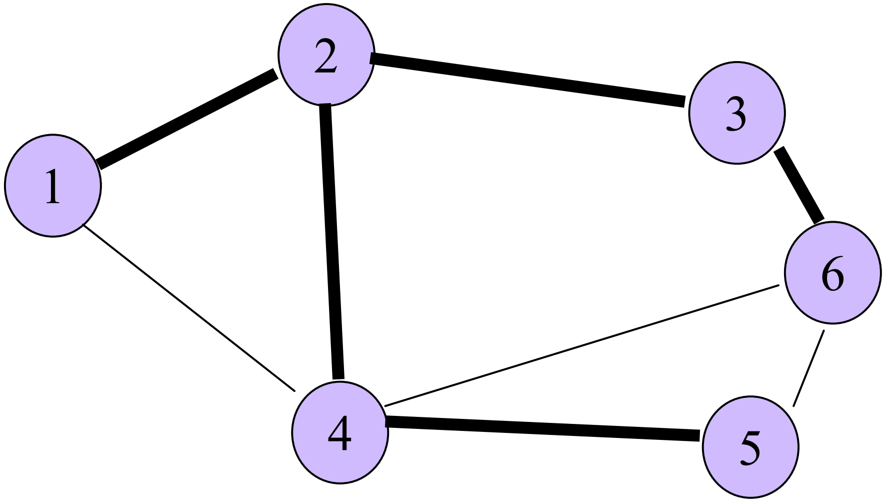


$$m = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

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', CONNETTENDO I NODI u_i,u_j CON I DUE ARCHI E (u_j,u_i) E (u_i,u_j).

SPECIFICA

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

NODO: INSIEME FINITO QUALSIASI

LISTA(o SET): LISTA (o INSIEME) DI ELEMENTI DI TIPO NODO

BOOLEAN: INSIEME DEI VALORI DI VERITA’

OPERATORI:

CREAGRAFO: () → GRAFO

CREAGRAFO = G

PRE: NESSUNA

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

GRAFOVUOTO: (GRAFO) → BOOLEAN

GRAFOVUOTO (G) = b

PRE: NESSUNA

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

b=FALSO ALTRIMENTI

INSNODO: (NODO,GRAFO) → GRAFO

INSNODO (u,G) = G'

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

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

INSARCO: (NODO,NODO,GRAFO) → GRAFO

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)}

ESISTENODO: (NODO,GRAFO) → BOOLEAN

ESISTENODO (u,G) = b

PRE: G = (N,A)

POST: b=VERO SE u $\in$N

b=FALSO ALTRIMENTI

ESISTEARCO: (NODO,NODO,GRAFO) → BOOLEAN

ESISTEARCO (u,v,G) = b

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

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

b=FALSO ALTRIMENTI

CANCNODO: (NODO,GRAFO) → GRAFO

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: (NODO,NODO,GRAFO) → GRAFO

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: (NODO,GRAFO) → LISTA

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 | (u,v) $\in$ A}

Talvolta è utile avere l'insieme di tutti i nodi validi

LISTANODI: (GRAFO) → LISTA

LISTANODI (G) = S

PRE: G = (N,A)

POST: RESTITUISCE UNA LISTA CHE CONTIENE TUTTI GLI ELEMENTI DI N

QUANDO 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 TIPI (TIPOETICHETTA, TIPOPESO) E NUOVI OPERATORI PER SCRIVERE E LEGGERE I NODI

LEGGINODO(u,G): (NODO, GRAFO) → TIPOETICHETTA

SCRIVINODO(a, u, G): (TIPOETICHETTA, NODO, GRAFO) → GRAFO

MENTRE PER RITROVARE O MODIFICARE LE INFORMAZIONI ASSOCIATE AGLI ARCHI USEREMO LEGGIARCO E SCRIVIARCO.

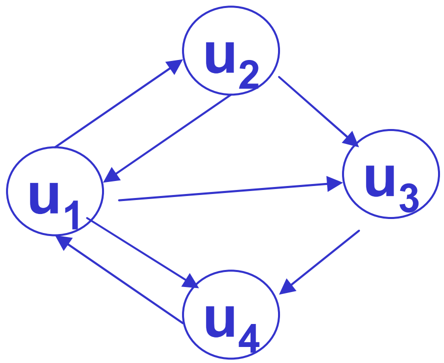
INOLTRE, POSSONO RISULTARE UTILI OPERATORI CHE CONTROLLANO LA CARDINALITA' DEGLI INSIEMI N E A

NUMNODI (G): (GRAFO) → INTEGER

NUMARCHI (G): (GRAFO) → INTEGER

RAPPRESENTAZIONE CON MATRICE DI ADIACENZA

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



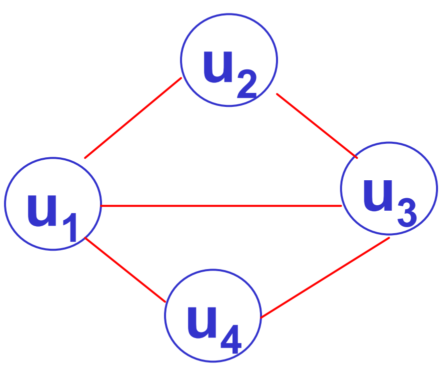
	1	2	3	4
1	0	1	1	1
2	1	0	1	0
3	0	0	0	1
4	1	0	0	0

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

$$e_{ij} = \begin{cases} P_{ij} & \text{SE } (i,j) \in A \\ +\infty \ (-\infty) & \text{SE } (i,j) \notin A \end{cases}$$

RAPPRESENTAZIONE CON MATRICE DI ADIACENZA

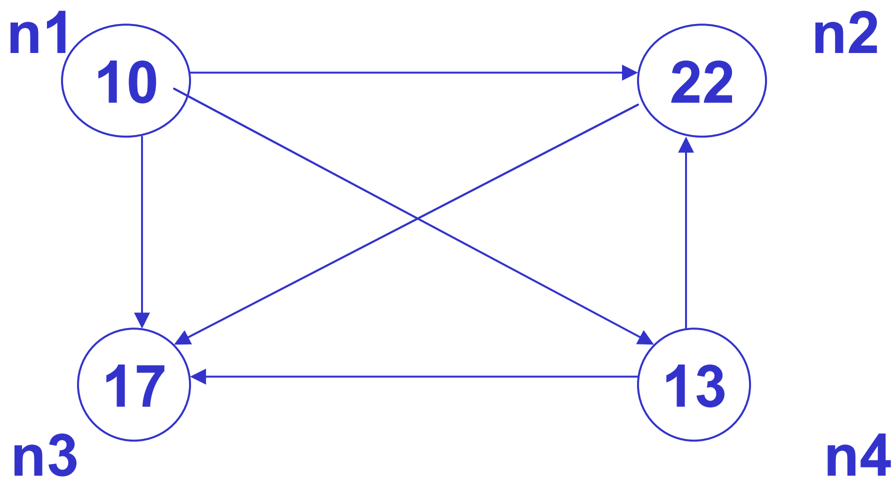
NATURALMENTE E' POSSIBILE UTILIZZARE LA MEDESIMA RAPPRESENTAZIONE PER GRAFI NON ORIENTATI: NE RISULTERA' UNA MATRICE SIMMETRICA RISPETTO ALLA DIAGONALE PRINCIPALE CON $e_{ij} = e_{ji}$



	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 x m), $B = [b_{ik}]$, NELLA QUALE CIASCUNA RIGA RAPPRESENTA UN NODO E CIASCUNA COLONNA RAPPRESENTA UN ARCO.

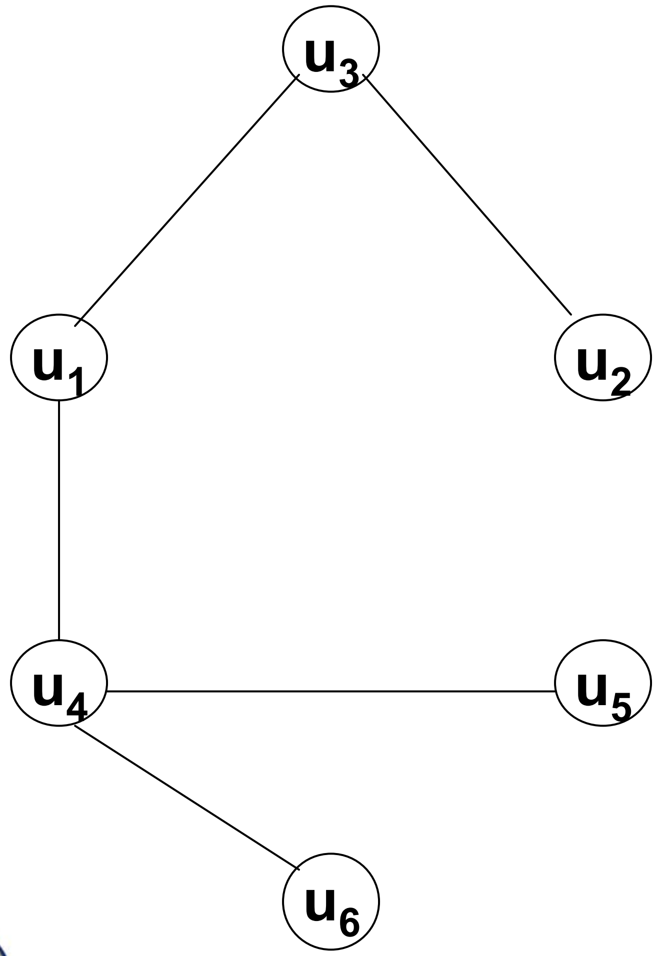
PER UN GRAFO NON ORIENTATO

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

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

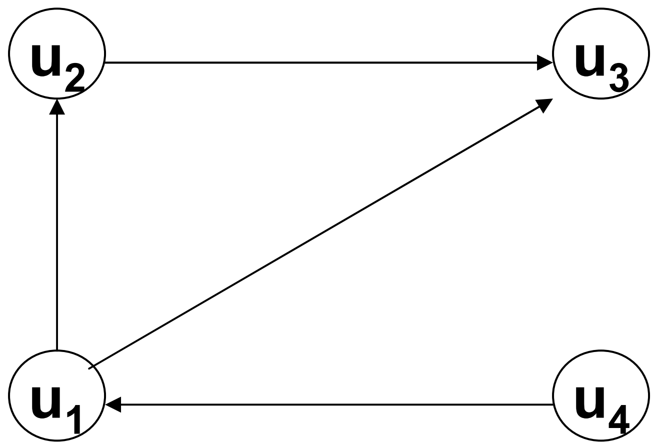
$$b_{ij} = \begin{cases} +1 & \text{SE L'ARCO } j\text{-ESIMO ENTRA NEL NODO } i \\ -1 & \text{SE L'ARCO } j\text{-ESIMO ESCE DAL NODO } i \\ 0 & \text{ALTRIMENTI} \end{cases}$$

RAPPRESENTAZIONE CON MATRICI D'INCIDENZA GRAFO NON ORIENTATO



	(u_1,u_3) 1	(u_1,u_4) 2	(u_2,u_3) 3	(u_4,u_5) 4	(u_4,u_6) 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.

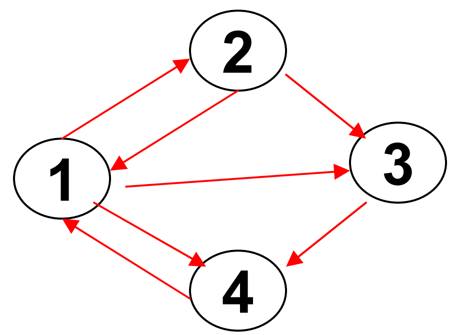
PER CALCOLARE A(u) E' NECESSARIO SCANDIRE LA RIGA u DI B ALLA RICERCA DELLE COLONNE k ∍′ $b_{u\,k}$ = -1, E PER OGNI COLONNA k SCANDIRE L'INDICE DI RIGA i ∍′ $b_{i\,k}$ = +1.

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.

1	1
2	4
3	6
4	7

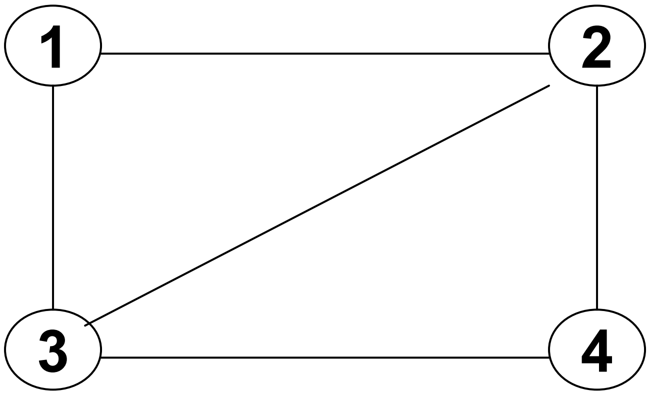
1	2
2	3
3	4
4	1
5	3
6	4
7	1



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)

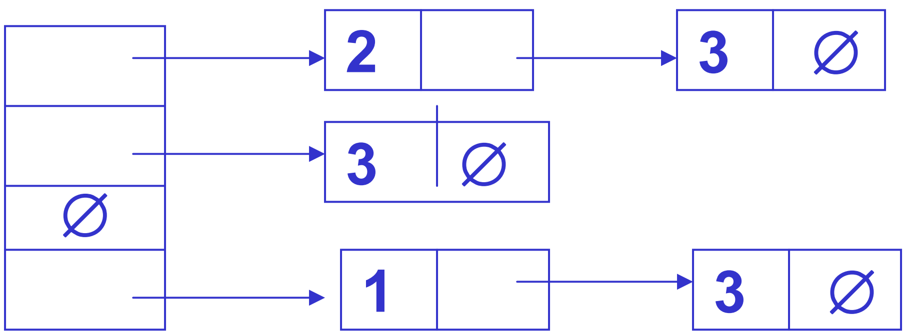
1	1
2	3
3	6
4	9

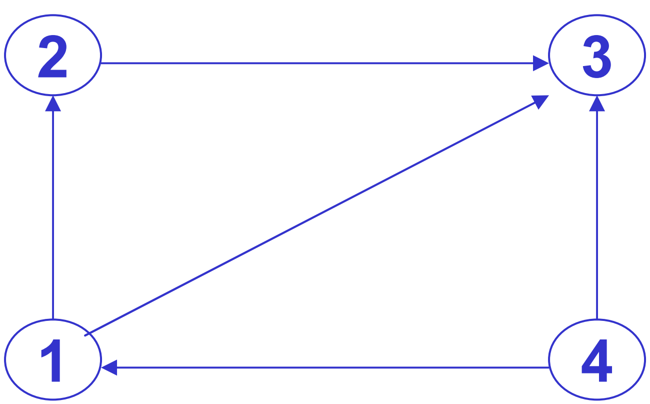
1	2
2	3
3	1
4	3
5	4
6	1
7	2
8	4
9	2
10	3

RAPPRESENTAZIONE CON LISTE DI ADIACENZA

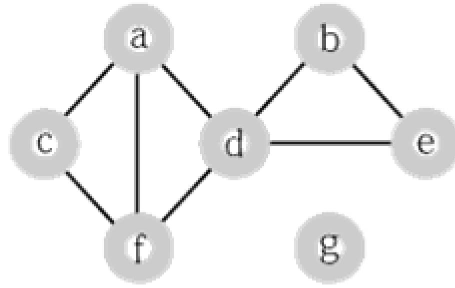
E' POSSIBILE ANCHE UTILIZZARE UN VETTORE DI NODI A (1..n) ED n LISTE.

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





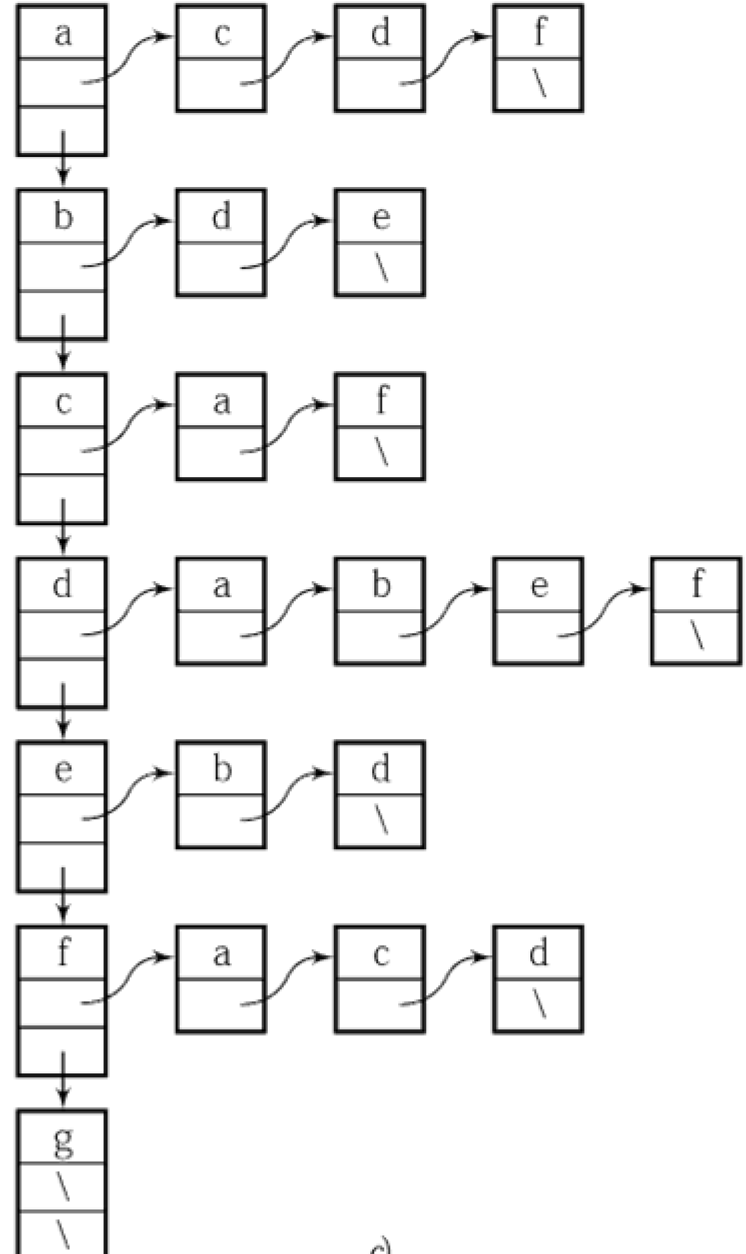
Un grafo (a) rappresentato con vettore che implementa una lista (statica) di adiacenze (b) e con una lista di liste di adiacenza (c)



a)

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

b)



c)

	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

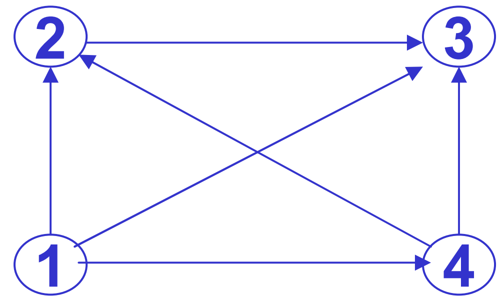
e)

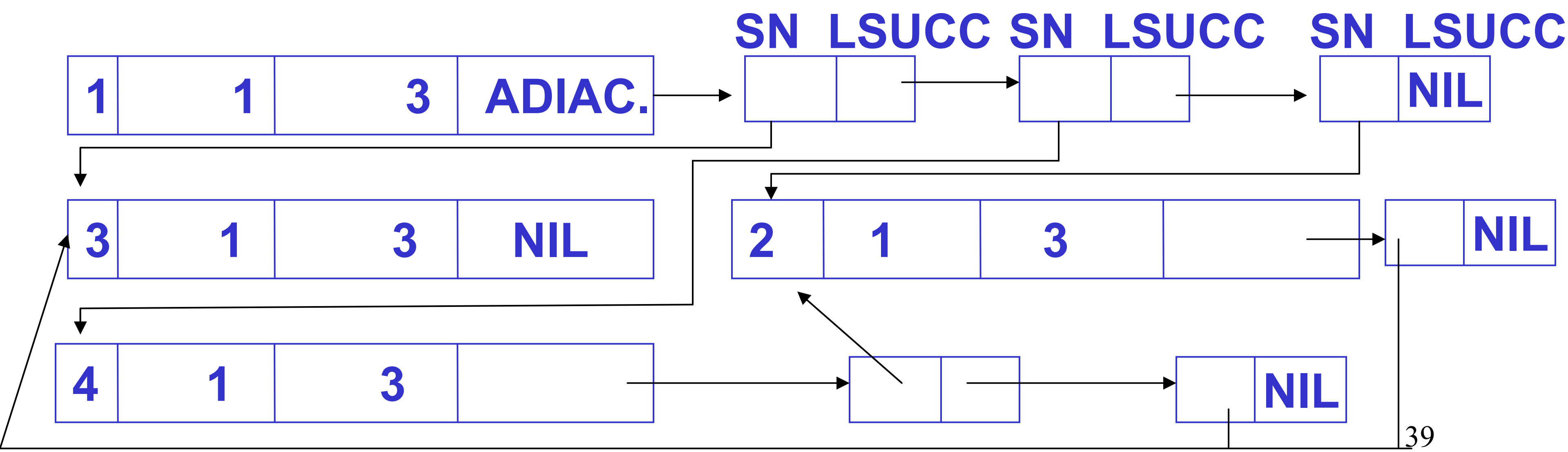
Lo stesso grafo (a) rappresentato come una matrice di adiacenze (*d*) e come una matrice d’incidenza (*e*)

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 $O(n+m)$

Grafi connessi e grafi sconnessi

- Si può dimostrare che un grafo è connesso se non ha sconnessioni.
- Questo criterio non è però efficiente in quanto se il grafo ha n nodi le possibili partizioni dell'insieme dei nodi sono 2^n.
- Vedremo in seguito un metodo più efficiente per determinare se un grafo è connesso.

Algoritmo per riconoscere se un grafo non orientato è connesso

- Inizia una lista(insieme) con un nodo (vertice) V a caso.
- Aggiungi alla lista(insieme) tutti i nodi adiacenti a V.
- Ripeti ricorsivamente la procedura a partire da ciascuno dei nuovi nodi introdotti.
- Se ad un certo punto la lista contiene tutti i nodi del grafo, il grafo è connesso.
- Se ripetendo il procedimento non si ottengono nuovi elementi nella lista, ma la lista non contiene tutti i nodi, il grafo non è connesso.

Esempio

Sia G il grafo la cui matrice di adiacenza è

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

Il nodo 1 è unito al nodo 3, il nodo 2 è unito al nodo 4, e i nodi 1,2,3,4 sono uniti a se stessi.

Applicando l'algoritmo partendo da 1 aggiungo il nodo 3 alla lista, ma al passo successivo posso solo aggiungere i nodi 1 e 3 che già fanno parte della lista. Quindi il grafo non è connesso

Esempio

Sia G il grafo la cui matrice di adiacenza è

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

Applicando l'algoritmo partendo da 1, aggiungiamo alla lista il vertice 3.

3 è unito a 2 da due archi, quindi aggiungiamo 2 alla lista.

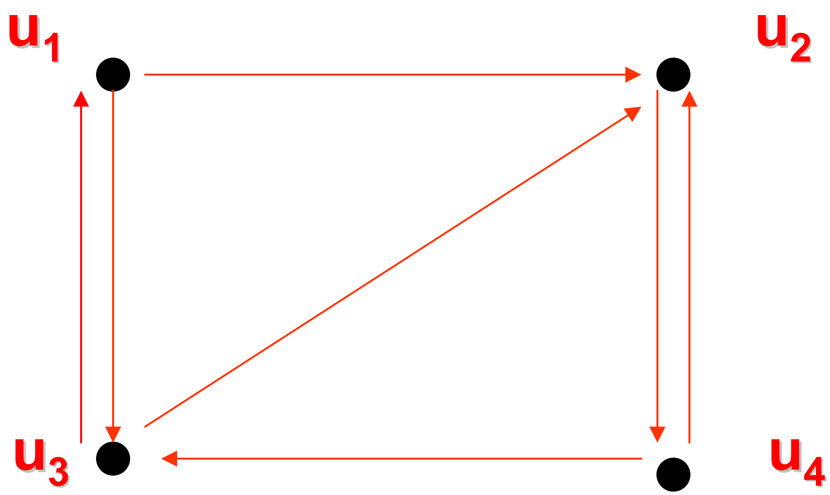
2 è unito a 4 da un arco, quindi aggiungiamo anche 4.

Ora la lista contiene tutti i vertici, e il grafo è connesso.

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 = <N,A> 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

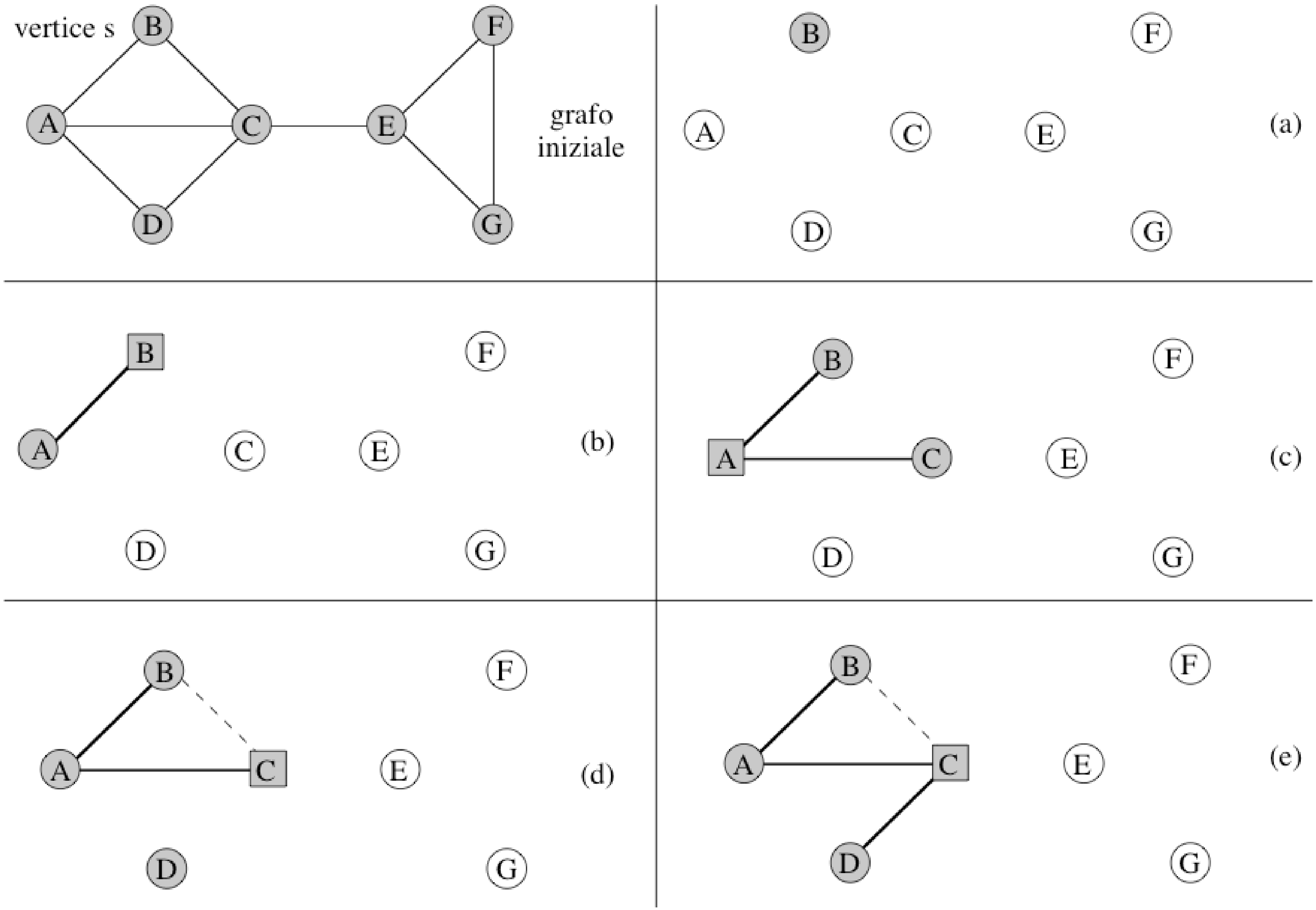
IL PRIMO SCHEMA DI VISITA E' QUELLO IN PROFONDITA'

(DFS ovvero DEPTH-FIRST-SEARCH)

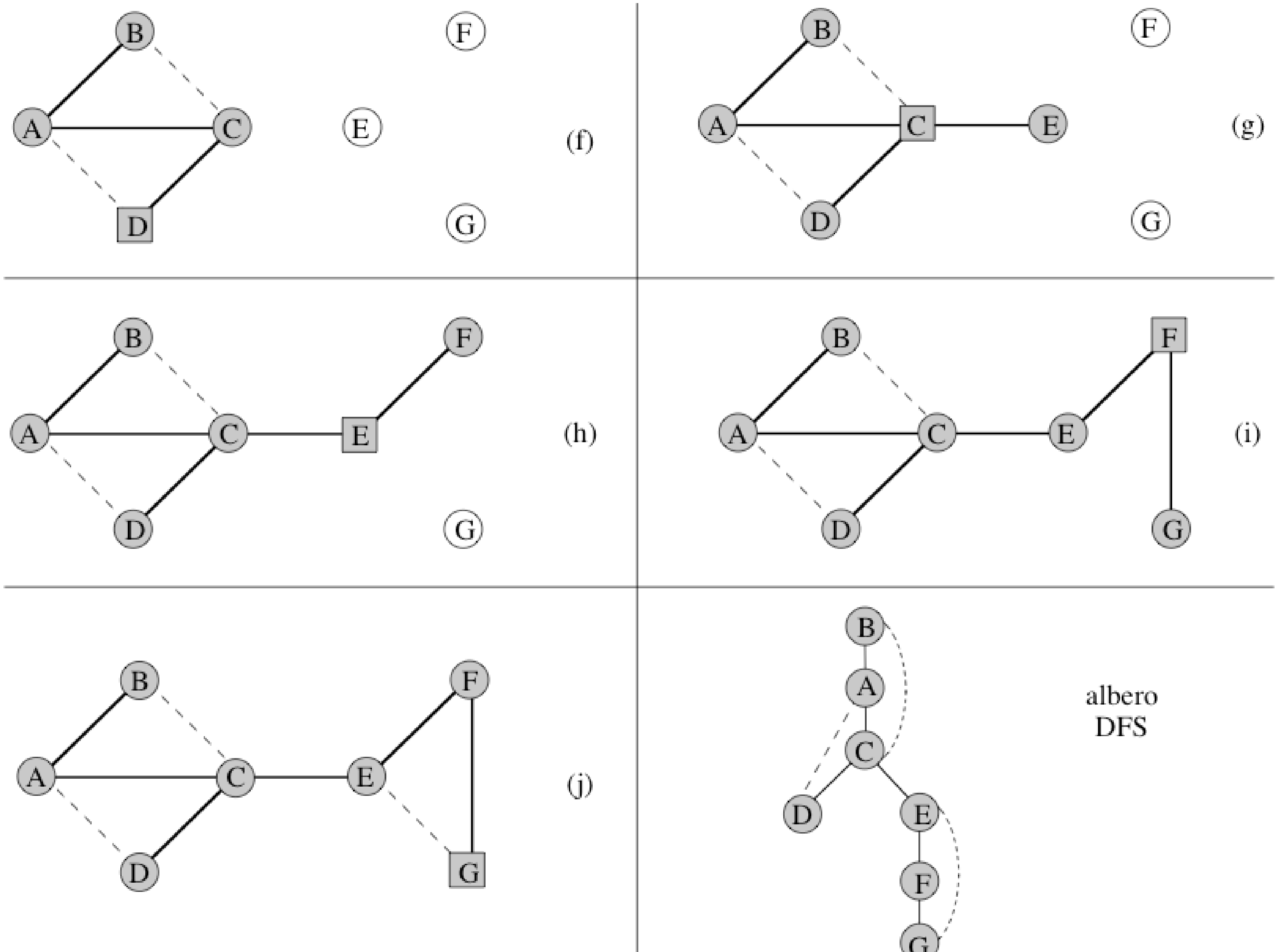
L'ALGORITMO PREVEDE CHE SI "*CONTRASSEGNI*" UN NODO APPENA LO SI VISITA E POI CI SI SPOSTI IN UN VERTICE ADIACENTE NON CONTRASSEGNATO.

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

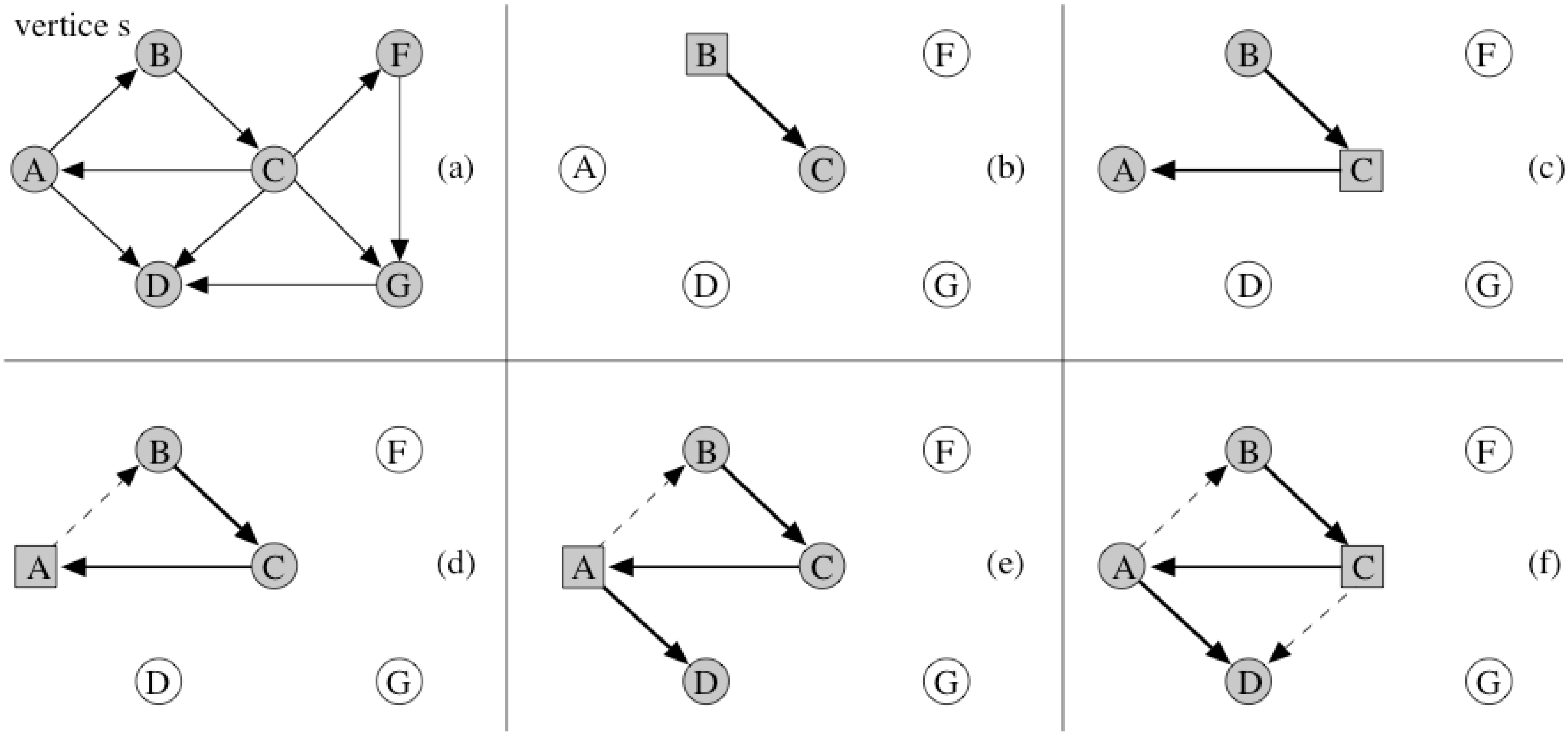
Esempio (DFS): grafo non orientato (1/2)



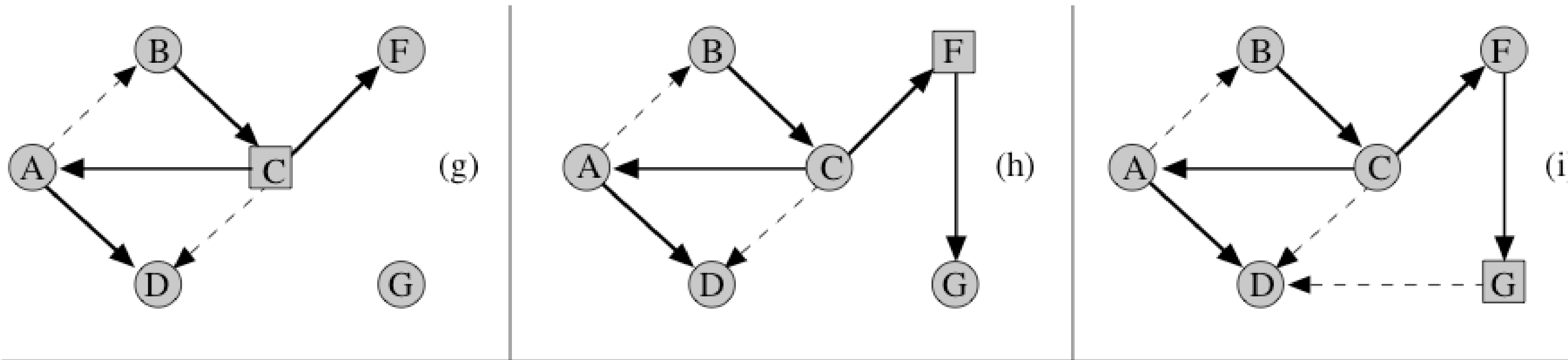
Esempio (DFS): grafo non orientato (2/2)

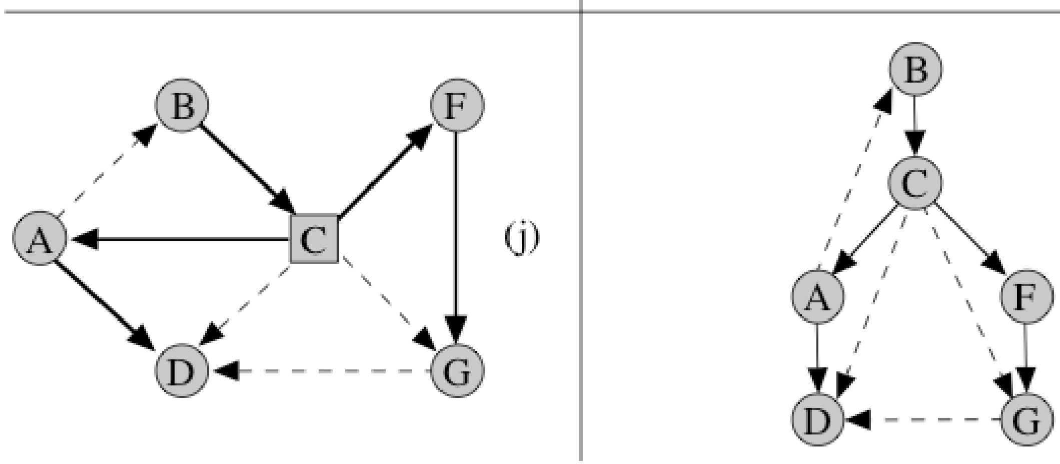


Esempio (DFS): grafo orientato (1/2)



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)

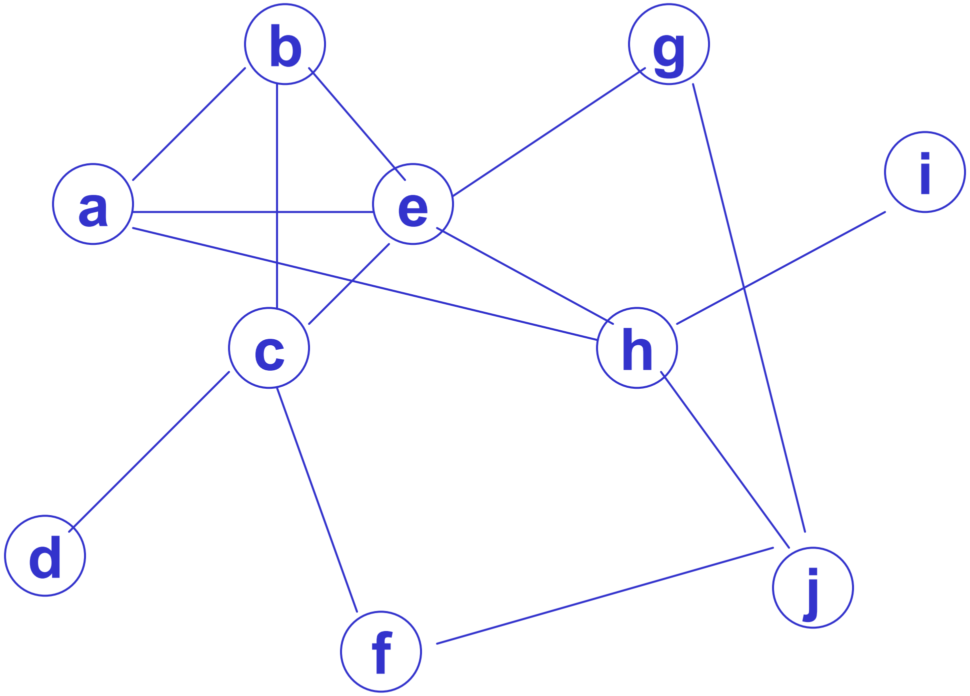
ESAMINA IL NODO u E MARCALO “VISITATO”

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

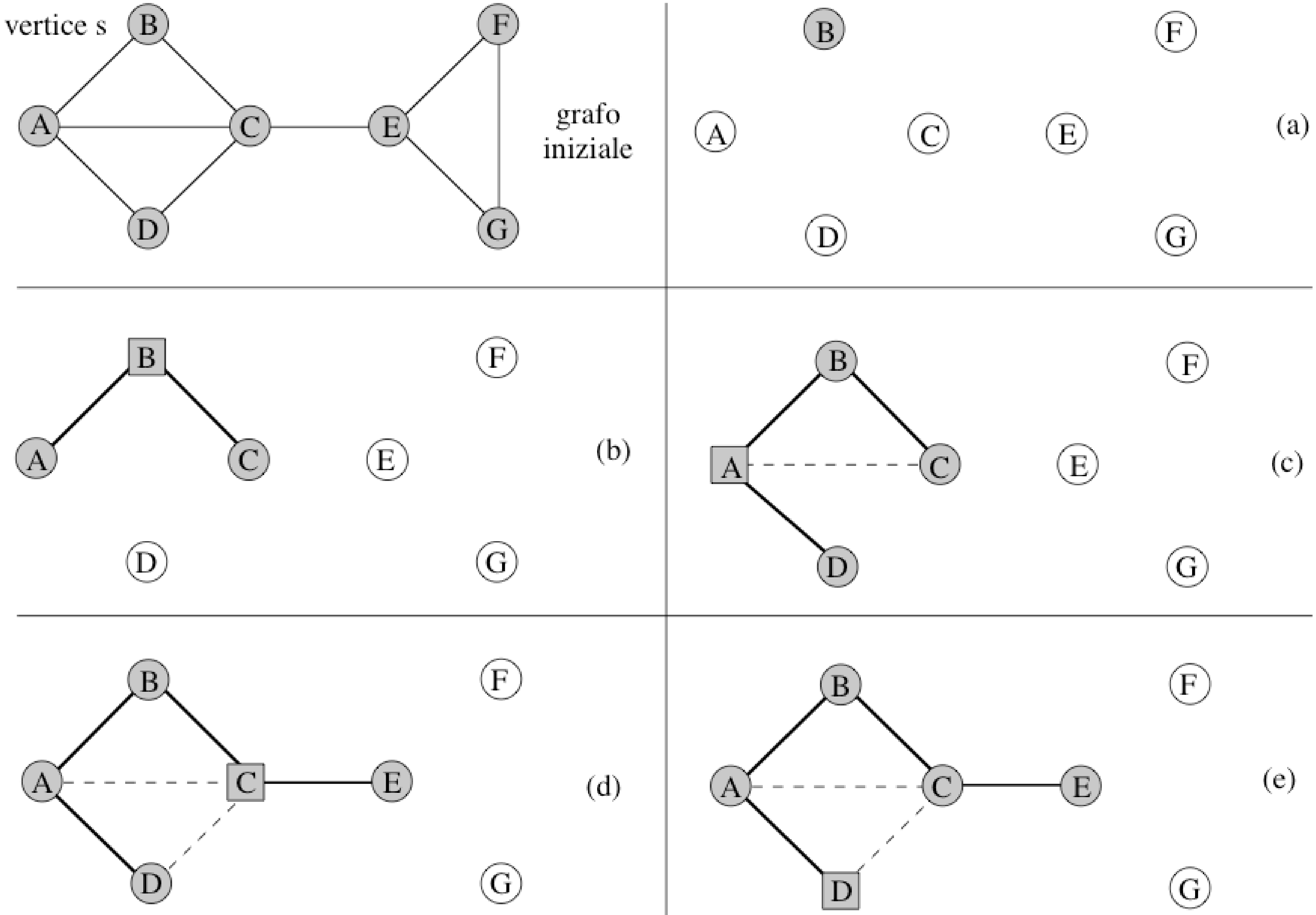
(BFS ≡ BREADTH-FIRST-SEARCH)

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

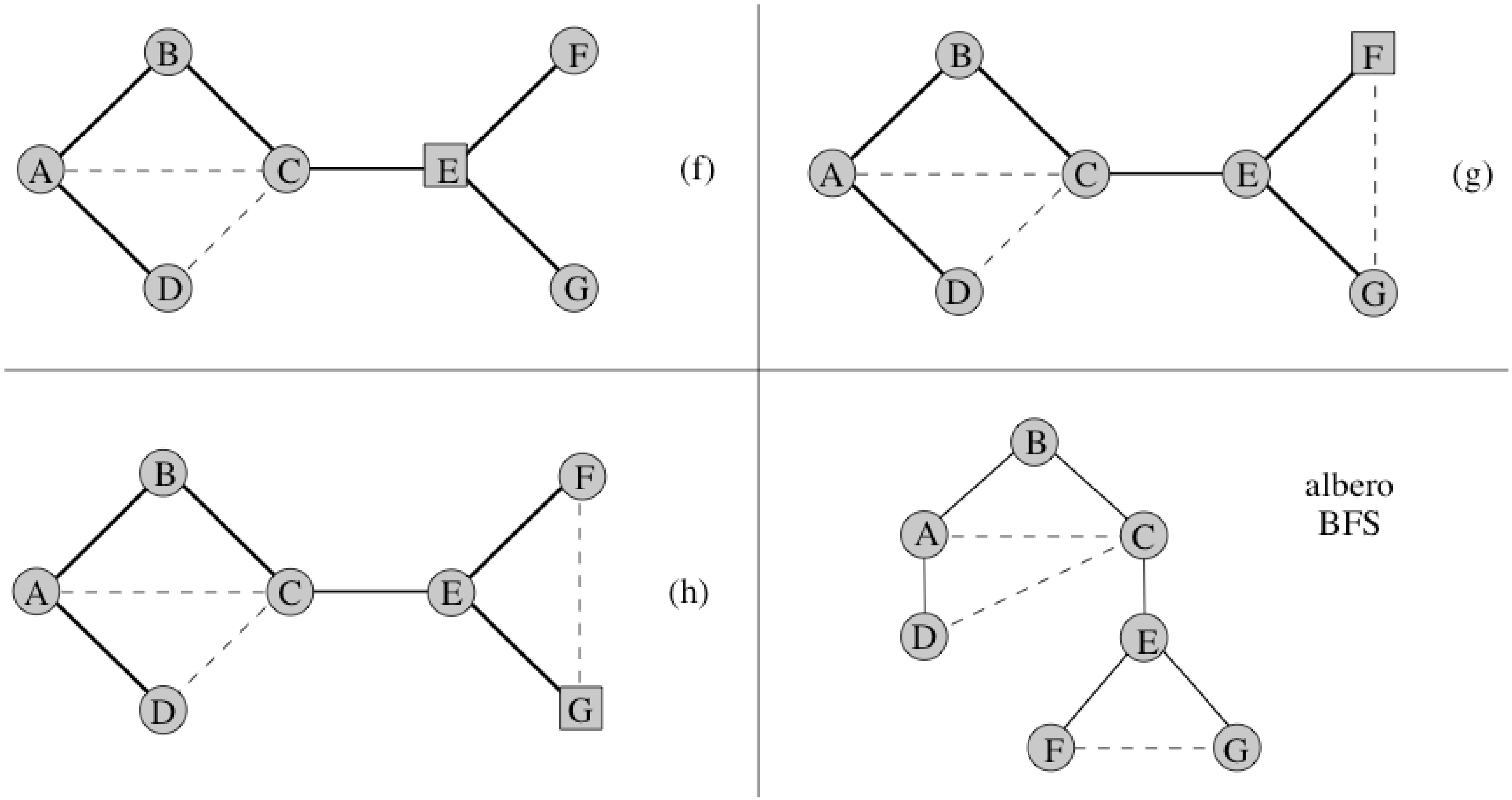
I NODI SONO VISITATI IN ORDINE DI *DISTANZA* CRESCENTE DAL NODO DI PARTENZA **u**, DOVE LA *DISTANZA* DA **u** AD UN GENERICO NODO **v** E' IL **MINIMO NUMERO DI ARCHI IN UN CAMMINO DA u A v.**

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.

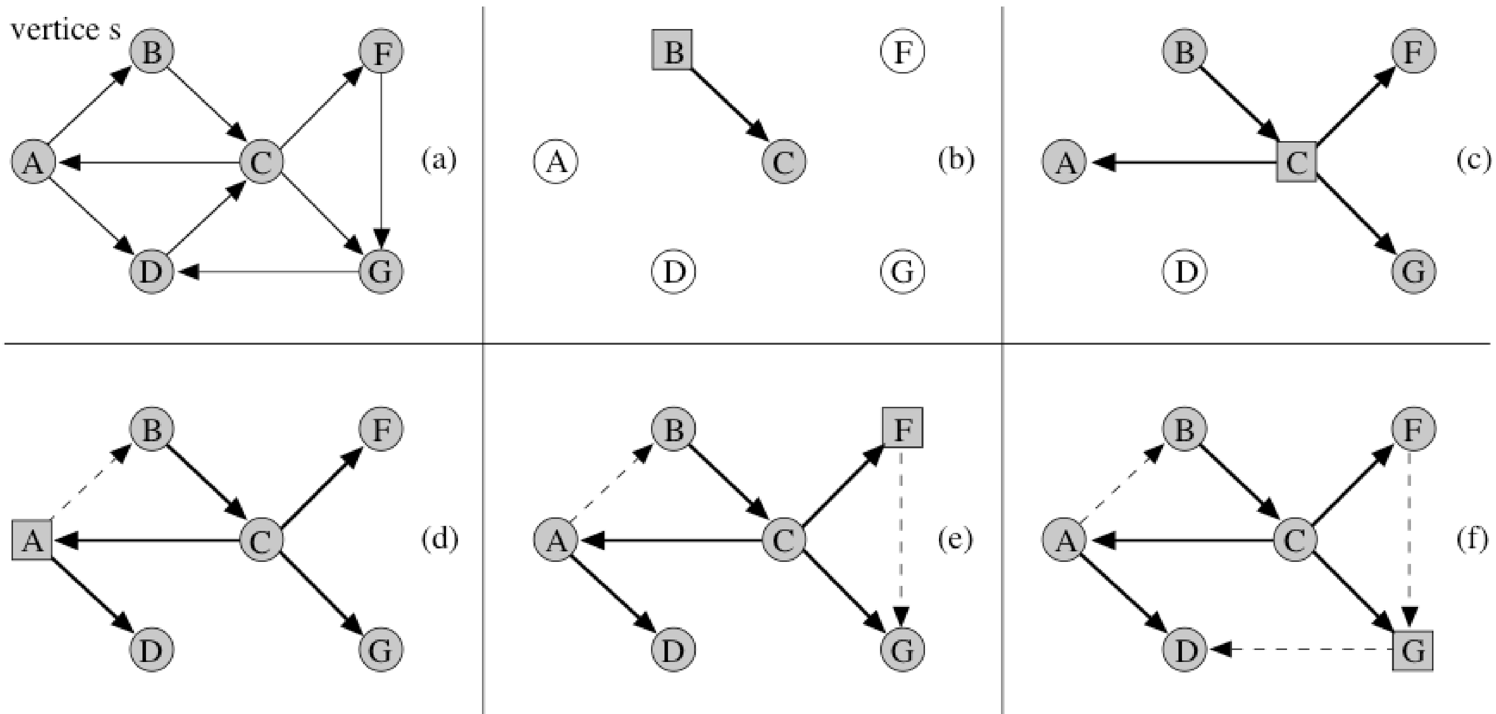
Esempio (BFS): grafo non orientato (1/2)

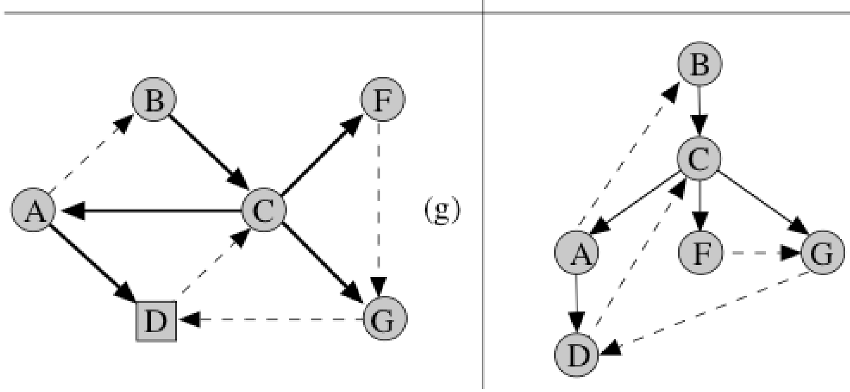


Esempio (BFS): grafo non orientato (2/2)



Esempio (BFS): grafo orientato





archi con estremi nello stesso livello: (F,G)

archi da un livello a quello immediatamente successivo: (G,D)

archi da un livello a uno precedente: (A,B) e (D,C)

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

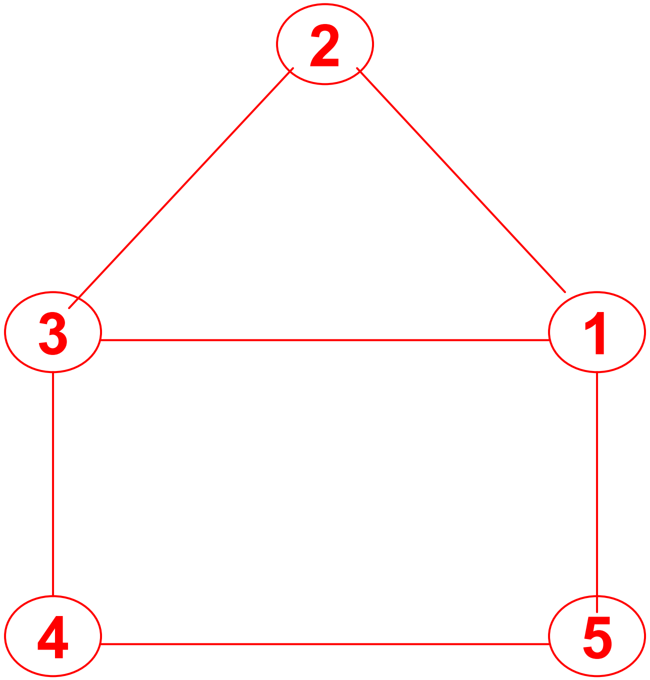
ALGORITMO DELLA BFS

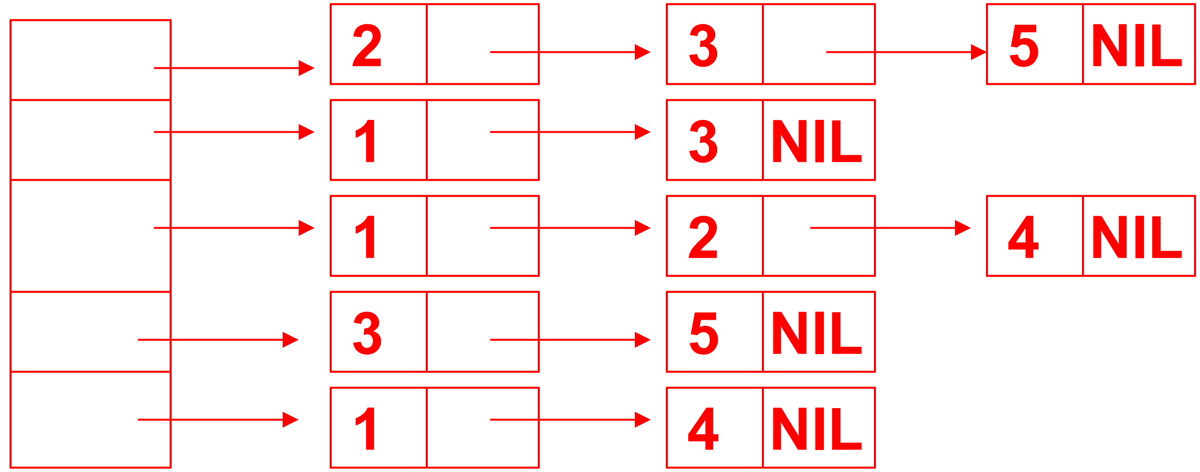
```
BFS (G:GRAFO; u:NODO)
  CREACODA(Q)
  INCODA(u,Q)
  while not CODAVUOTA(Q) do
    u ← LEGGICODA(Q)
    FUORICODA(Q)
    esamina u e marcalo “visitato”
    PER TUTTI I NODI v ADIACENTI A u (corrisponde ad un ciclo
       sulla  lista Adiacenti (u,G))
       esamina l’arco (u,v)
      if v non è marcato “visitato” and v ∉ Q then
        INCODA(v,Q)
```

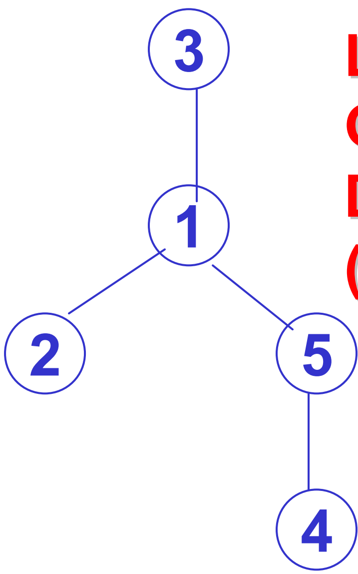
Costo della visita in ampiezza

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

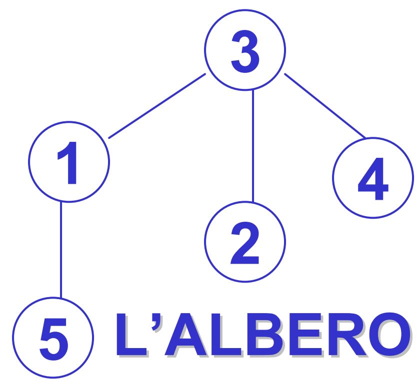
- Liste di adiacenza: $O(m+n)$
- Matrice di adiacenza: $O(n^2)$







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)***

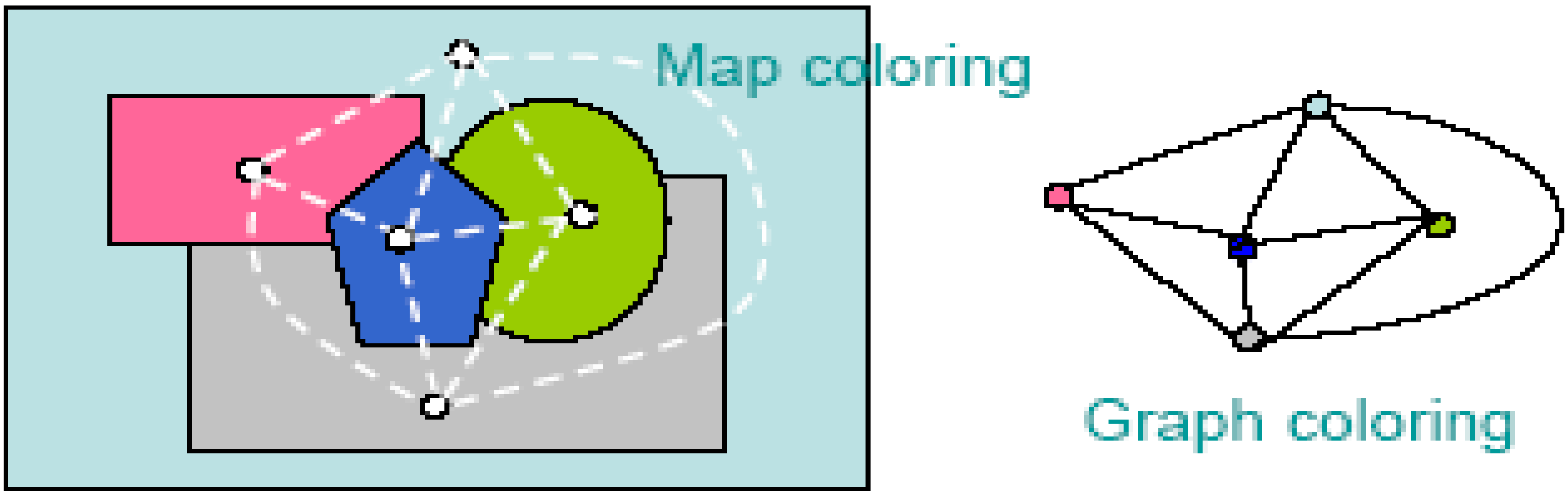
QUESTO RISULTA INTERESSANTE QUANDO SI VOGLIA RISOLVERE UN PROBLEMA CHE RICHIEDE DI SELEZIONARE, TRA TUTTI I PERCORSI CHE CONNETTONO DUE NODI IN UN GRAFO, QUELLO CHE RISULTA OTTIMO, AD ESEMPIO IN BASE AL CRITERIO DI MINIMIZZAZIONE DELLA SOMMA DEI PESI ASSOCIATI AGLI ARCHI.

***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.*

La colorazione di un grafo

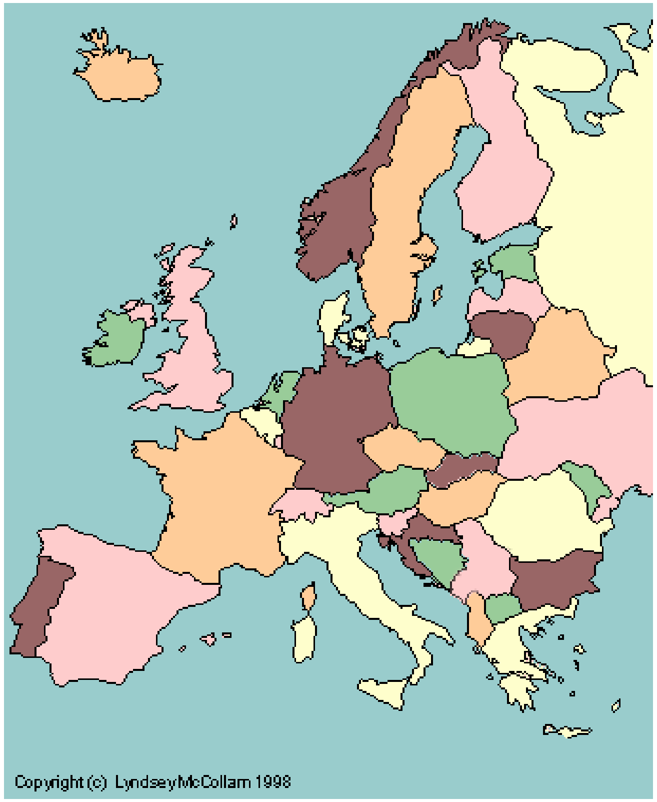
- Un famoso problema matematico è quello di dimostrare che una qualsiasi carta geografica può essere colorata con quattro colori in modo che due qualunque stati confinanti abbiano colori diversi. (Problema dei quattro colori).
- Se rappresentiamo gli stati con dei nodi e i confini fra due stati come uno spigolo che unisce i nodi corrispondenti, il problema diventa quello di colorare i nodi di un grafo con quattro colori in modo che due nodi uniti da uno spigolo abbiano colori diversi.

Il problema della colorazione di mappe



Il problema della colorazione di mappe

- Colora le nazioni con il minor numero di colori
- I colori devono essere differenti per regioni adiacenti
- 4 colori bastano sempre

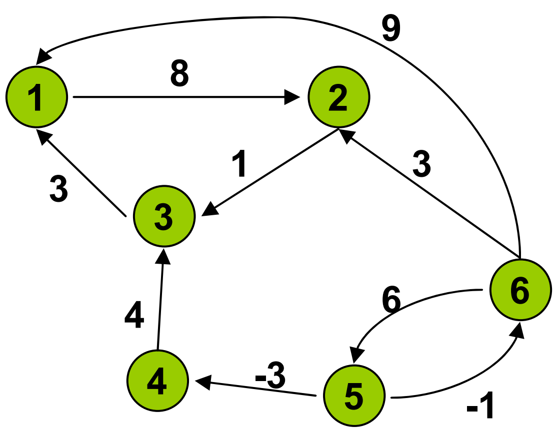


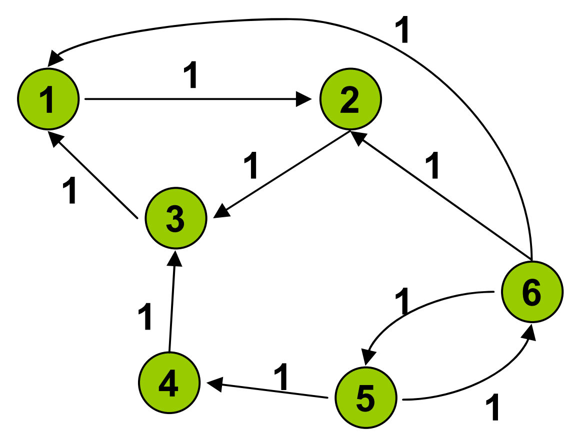
Problema generale

- Siano dati un grafo G ed un numero n. E' possibile colorare G con n colori in modo che due qualunque vertici uniti da uno spigolo abbiano colori diversi? Se si può il grafo G viene detto n-colorabile.
- Per ogni n, esiste un grafo che non è n colorabile. Basta prendere n+1 vertici e unirli a due a due in tutti i modi possibili. Per colorarlo ho bisogno di n+1 colori.

Problema dei cammini minimi

Input: un grafo G=(N,A) **orientato** e **pesato**, con una funzione peso w: A → 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= <n_1, n_2, …,n_k> è la somma dei pesi degli archi che lo costituiscono.

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

p1= <PO, FI, SI> =

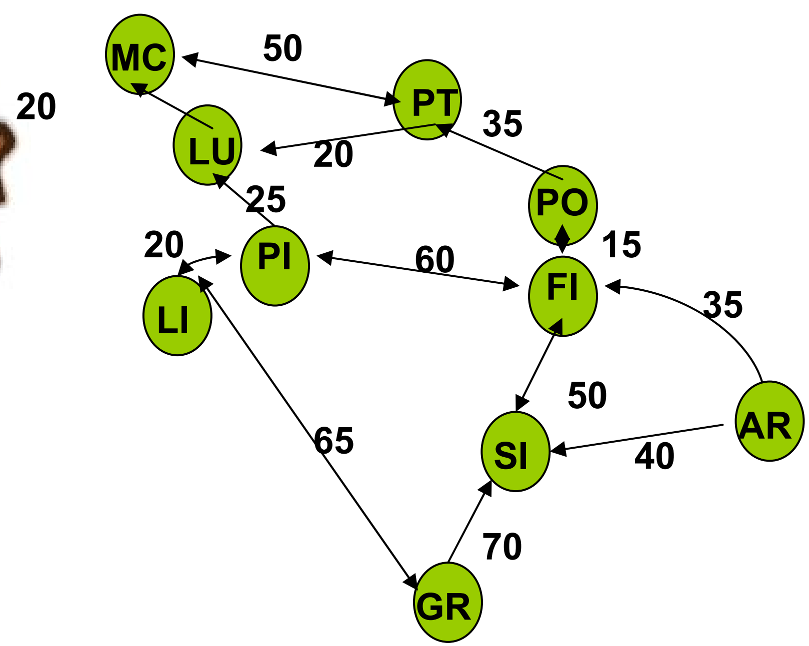
15 min + 50 min = 65 min

p2 = <PO, FI, AR, SI> =

15 min + 35 min + 40 min =

= 90 min

p3 = <PO, FI, PI, LI, GR, SI> = 160 min



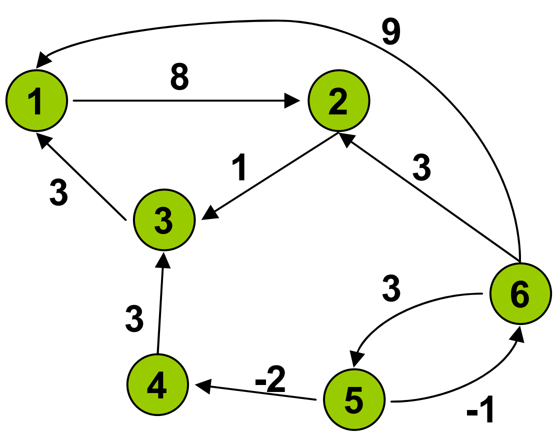
Problema dei cammini minimi

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

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

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!



$\delta(6,1) = 7$

$p_1 = <6, 2, 3, 1>$ $\quad w(p_1) = 7$

$p_2 = <6, 1>$ $\quad w(p_2) = 9$

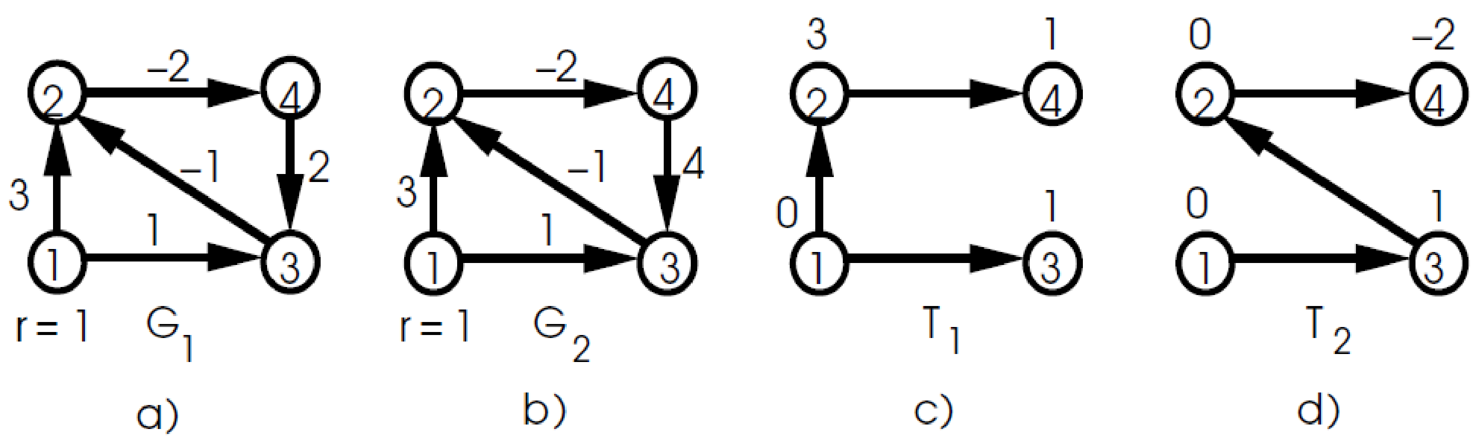
$p_3 = <6, 5, 6, 2, 3, 1>$ $\quad w(p_3) = 9$

$p_4 = <6, 5, 4, 3, 1>$ $\quad w(p_4) = 7$

Esempio

- Nella figura
  - un grafo con un ciclo negativo
  - un grafo senza cicli negativi
  - una soluzione ammissibile per
  - una soluzione ottima per G_2

- **Esempio di pesi negativi**
  - Proprietario TIR
  - Viaggiare carico → profitto
    - Peso negativo
  - Viaggiare scarico → perdita
    - Peso positivo



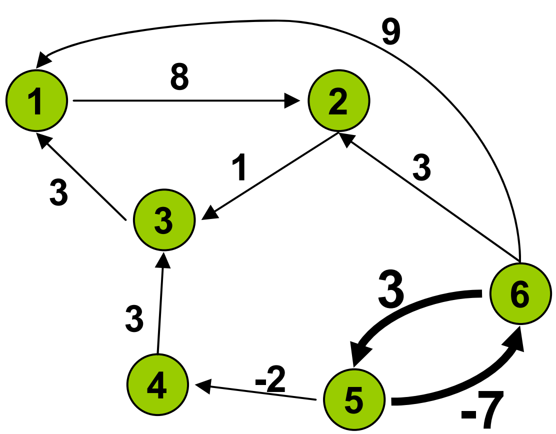
Vari problemi

- **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.

Archi 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 (-∞).**



Ciclo <6,5> negativo.

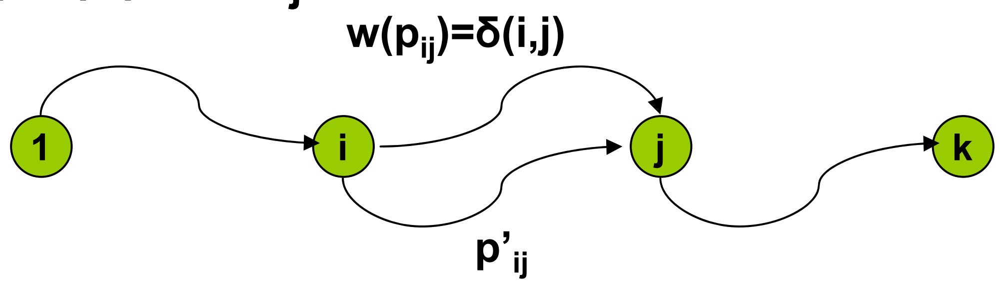
Ogni volta che compio un giro diminuisco il peso del cammino che passa per il ciclo.

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

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 $p= <v_1, v_2, \dots,v_k>$ 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 $p_{ij}= <v_i, v_{i+1}, \dots,v_j>$ è un cammino minimo**.

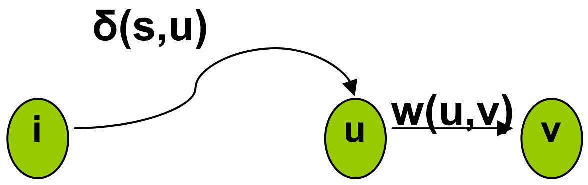


Dato un altro sottocammino da i a j p'_{ij}, necessariamente $w(p_{ij}) \leq w(p'_{ij})$, altrimenti il cammino minimo passa per p'_{ij}.

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 minino da s a v è **δ(s,v) = δ(s,u) + w(u,v).**



cammino minimo tra u e v

δ(s,v) = δ(s,u) + w(u,v).

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

δ(s,v) ≤ δ(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.

---

prototipoCamminiMinimi(GRAPH G, NODE r)

% inizializza T ad una foresta di copertura composta da nodi isolati

% inizializza d con una sovrastima della distanza (0 per r, $+\infty$ altrimenti)

while $\exists(u, v) : d_u + w(u, v) < d_v$ **do**

  $d_v \leftarrow d_u + w(u, v)$

  % Sostituisci il padre di v in T con u

Note

Se al termine dell'esecuzione qualche nodo mantiene una distanza infinita,
esso non è raggiungibile da *r*

IL METODO BASE

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.

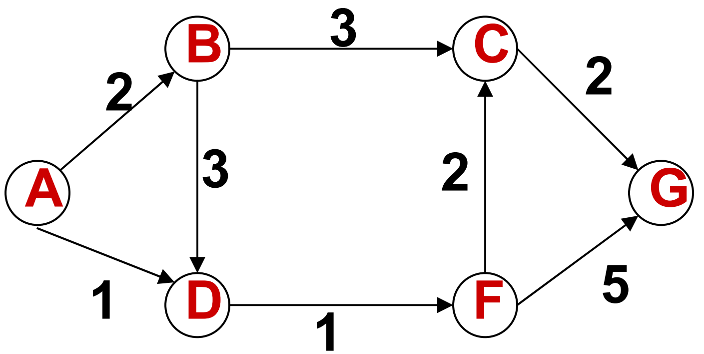
OSSERVIAMO CHE SE IL PROSSIMO CAMMINO MINIMO DA GENERARE C È DA r AL NODO u, TUTTI I NODI SONO IN S.

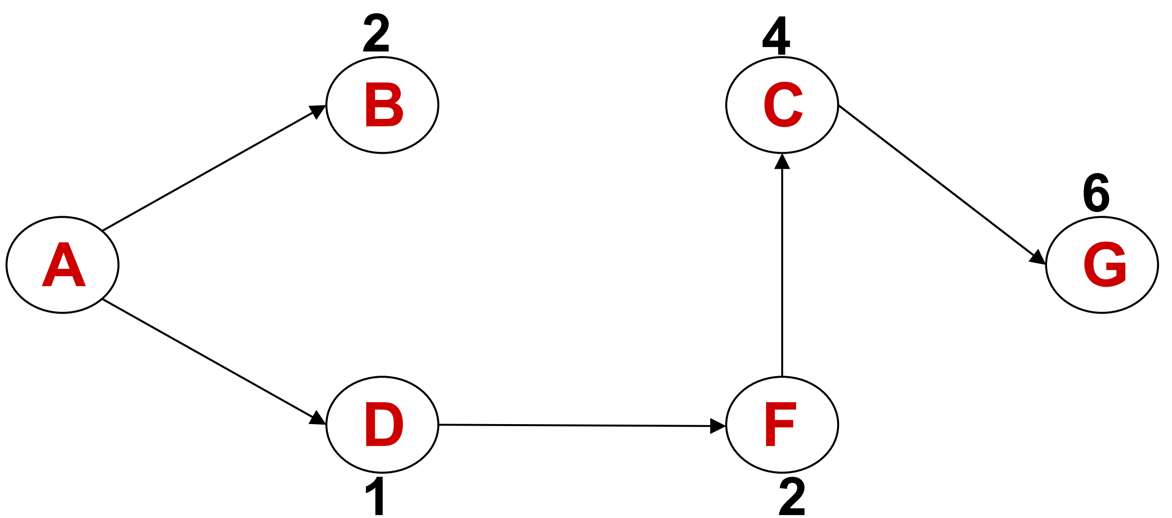
INFATTI SE UN NODO k DI C NON APPARTENESSE A S VI SAREBBE UN CAMMINO DA r A UN NODO k NON CONTENUTO IN S DI LUNGHEZZA MINORE A QUELLA DI C, CONTRADDICENDO L'IPOTESI CHE IL PROSSIMO CAMMINO DA GENERARE SIA C.

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 <u,z> 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.


A
B
C
D
F
G
2
3
2
3
2
1
1
5


2
B
4
C
A
6
G
D
F
1
2

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).

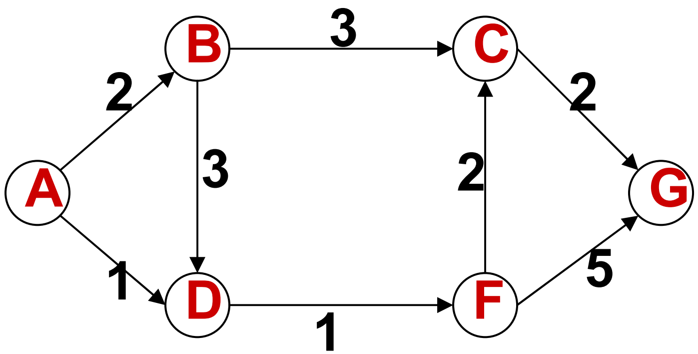
UNA SOLUZIONE AMMISSIBILE T È OTTIMA SE E SOLO SE

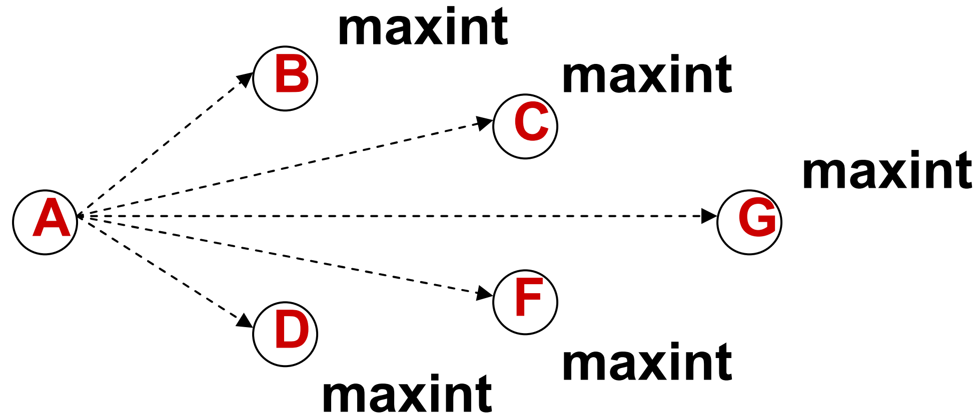
$$\mathrm{DIST}(i) + C_{ij} = \mathrm{DIST}(j)\ \forall\ (i,j) \in T \quad \text{E}$$

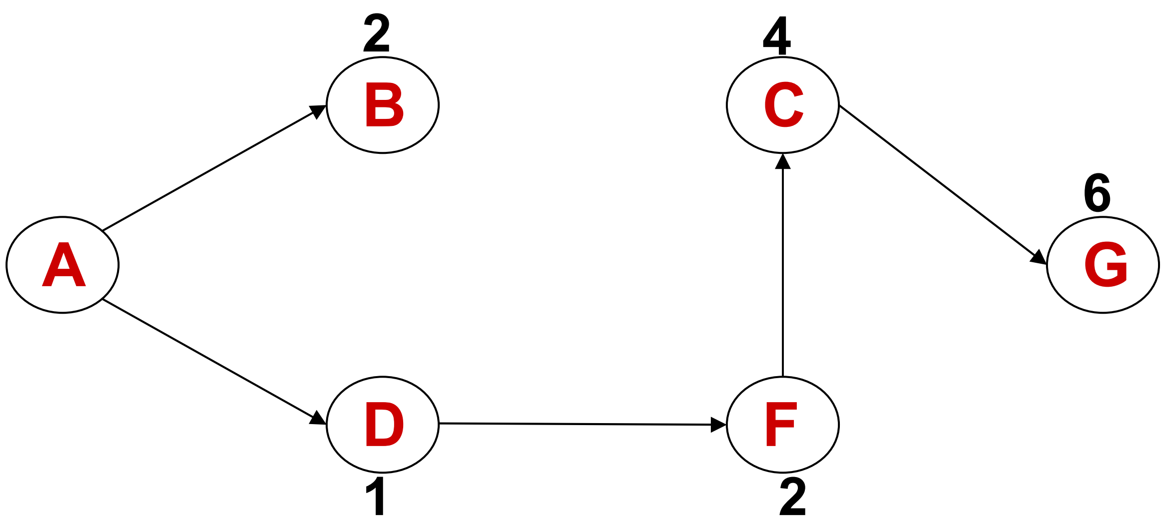
$$\mathrm{DIST}(i) + C_{ij} \geq \mathrm{DIST}(j)\ \forall\ \text{ARCO}(i,j) \in A$$

(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*


A
B
C
D
F
G
2
1
3
3
2
2
1
5


A
B
C
D
F
G
maxint
maxint
maxint
maxint
maxint


A
B
C
D
F
G
2
4
6
1
2

```
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) ←0; d(r) ←0;
for k=1 to n do
        if k≠r then
                T(k) ← r; d(k) ← maxint
INSERISCI (r,S)
while not INSIEMEVUOTO(S) do
        i ←LEGGI (S); CANCELLA (i, S);
        for each j ∈ ADIACENTI(i, G) l’insieme degli adiacenti di i do
                if d(j)+ c_ij < d(j)
                        T(j) ←i
                        d(j) ←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:

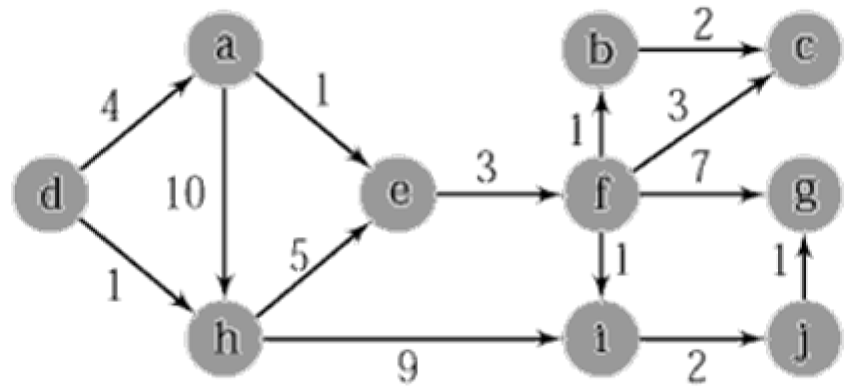
• **S**: dove d[v] = δ(s,v), quindi un cammino minimo tra s e v è stato determinato.

• **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]=∞.

Algoritmo di Dijkstra

```
DIJKSTRA(G,w,s)
    for ogni nodo u in N[G]                  // inizializzazione di ogni nodo
        do d[u] ← ∞
           p[u] ← NIL                        // predecessore di u indefinito
    d[s] ← 0                                 // si comincia dal nodo s
    Q ← N[G]                                 // coda a priorità
    S ← Ø                                    // insiemi dei cammini minimi trovati
    while Q≠Ø                                // termina quando la coda Q è vuota
        do u ← EXTRACT-MIN(Q)                // prendi il cammino in Q più
                                                piccolo
           S ← S ∪ {u}                       // inserisci u in S
           for ogni nodo v tra gli adiacenti di u considera l'arco (u.v)
        // aggiorna cammini minimi in Q con v adiacente a u
               do if d[v] > d[u] + w(u,v)
                  then d[v] ← d[u] + w[u,w]
                       p[v] ← u
```



a)

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	∞	4	4								
b	∞	∞	∞	∞	∞	9					
c	∞	∞	∞	∞	∞	11	11	11			
d	0										
e	∞	∞	6	5							
f	∞	∞	∞	∞	8						
g	∞	∞	∞	∞	∞	15	15	15	15	12	
h	∞	1									
i	∞	∞	10	10	10	9	9				
j	∞	∞	∞	∞	∞	∞	∞	11	11		

b)

Un’esecuzione di DijkstraAlgorithm

UN ALTRO PROBLEMA:

PROBLEMA DEL MINIMO ALBERO DI COPERTURA

DATO UN GRAFO <u>NON ORIENTATO E CONNESSO</u> 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.