# Grafi

## Il tipo astratto grafo: specifiche sintattiche e semantiche. Realizzazioni. Visite.

**Algoritmi e Strutture Dati + Lab**
A.A. 14/15

Informatica
Università degli Studi di Bari “Aldo Moro”

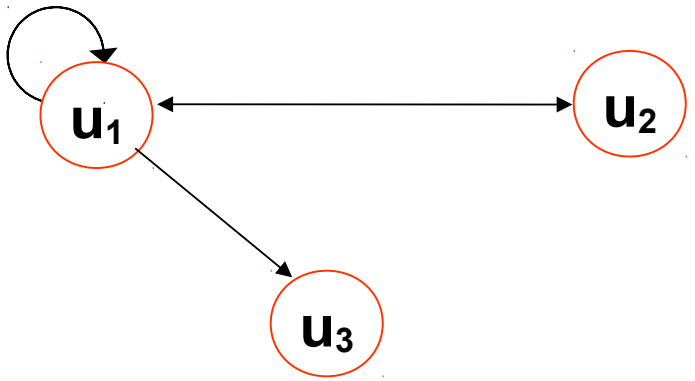
Nicola Di Mauro

# Il tipo astratto grafo

- Il grafo è una struttura composta da nodi e archi che rappresenta una relazione binaria sull'insieme costituito dai nodi.
- In generale, i nodi sono usati per rappresentare oggetti e gli archi per rappresentare relazioni tra coppie di oggetti.
- Consideriamo di seguito solo grafi orientati o diretti, nei quali gli archi hanno una direzione da un certo nodo (di partenza) a un altro nodo (di arrivo).
  - In tal caso il grafo G = (N, A), dove n è l'insieme finito dei nodi, prevede che a sia un insieme finito di coppie ordinate di nodi, rappresentanti gli archi orientati.
- Poiché ogni grafo non orientato G' può essere visto come un grafo orientato G, ottenuto da G', sostituendo ogni arco (i,j) con i due archi e (j,i) e (i,j)

# Grafi orientati

- Un grafo orientato G è una coppia <N, A> dove N è un insieme finito non vuoto (insieme di nodi) e A $\subseteq$ NxN è un insieme finito di coppie ordinate di nodi, detti archi (o spigoli o linee).
  - Se $< u_i, u_j > \in$ A nel grafo vi è un arco da $u_i$ ad $u_j$.
- Nell'esempio N = $\{u_1 1, u_2, u_3\}$, A = $\{(u_1, u_2), (u_1, u_2), (u_2, u_1), (u_1, u_3)\}$.



# Grafi non orientati

- Definiremo grafo non orientato un grafo G = <N, A> nel quale gli archi sono formati da coppie non ordinate. Mentre 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.
- I nodi congiunti da un arco sono adiacenti. Anche nei grafi non orientati troviamo nozioni analoghe a quelle di cammino (catena) e di ciclo (circuito).

# Specifica sintattica

- Tipi: grafo, boolean, nodo, lista, tipoelem
- Operatori:
  - crea: () → grafo
  - vuoto: (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 operatori
  - legginodo: (nodo, grafo) → tipoelem aggiuntivi
  - scrivinodo: (tipoelem, nodo, grafo) → grafo

# 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 verità
- Operatori:
  - crea = G
    - pre: nessuna
    - post: G = (N, A) con $N = \emptyset$ e $A = \emptyset$
  - vuoto (G) = b
    - pre: nessuna
    - post: b=vero se $N = \emptyset$ e $A = \emptyset$ ; b=falso altrimenti

# Specifica semantica /2

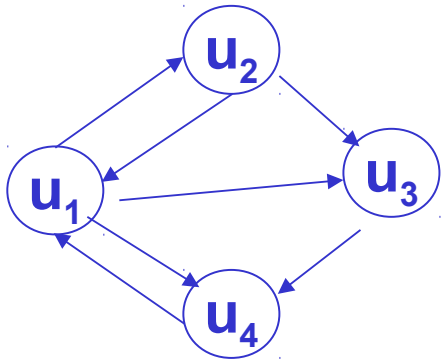
- insnodo (u, G) = G'
  - pre: G = (N, A) $u \notin N$
  - post: G' = (N', A), $N' = N \cup \{u\}$
- insarco (u, v, G) = G'
  - pre: G = (N, A), $u \in N$, $v \in N$, $(u,v) \notin A$
  - post: G' = (N, A'), $A' = A \cup \{(u,v)\}$
- cancnodo (u, G) = G'
  - pre: G = (N, A), $u \in N$ e non esiste $v \in N$ tale che $(u,v) \in A$ oppure $(v,u) \in A$
  - post: G' = (N', A), $N' = N \setminus \{u\}$
- cancarco (u, v, G) = G'
  - pre: G = (N, A), $u \in N$, $v \in N$, $(u,v) \in A$
  - post: G' = (N, A') , $A' = A \setminus \{(u,v)\}$
- adiacenti: (u, G) = L
  - pre: G = (N, A), $u \in N$
  - post: L è una lista che contiene una e una sola volta gli elementi di $A(u) = \{v \in N \mid (u,v) \in A\}$
-

# Specifiche

- Quando ai nodi e/o agli archi sono associate informazioni (etichette, pesi), si parla di grafi etichettati: in tal caso vanno introdotti nuovi operatori per ritrovare o modificare le informazioni associate ai nodi e/o agli archi.

# Rappresentazione con matrice di adiacenza

- La più 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.

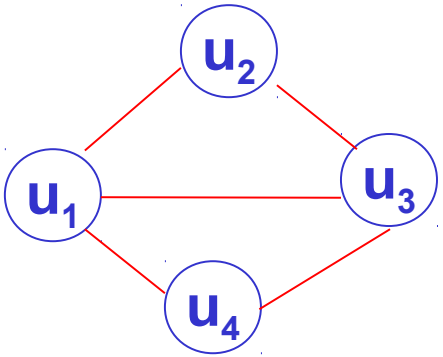



| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 |
| 2 | 1 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 1 |
| 4 | 1 | 0 | 0 | 0 |

- Se il grafo è pesato, nella matrice si utilizzano i pesi degli archi al posto degli elementi binari. Se $p_{ij}$ è il peso dell'arco (i,j) allora l'elemento della matrice E diventa $p_{ij}$ se (i,j) $\in$ A, e $+\infty$ ($-\infty$) se (i,j) $\notin$ A

# Rappresentazione con matrice di adiacenza /2

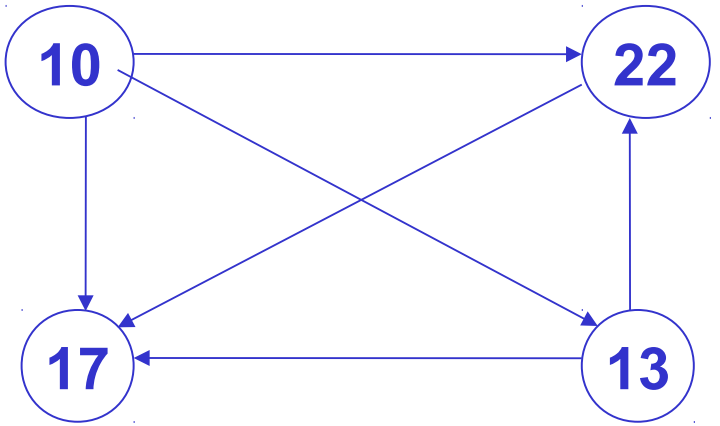
- Naturalmente è possibile utilizzare la medesima rappresentazione per grafi non orientati: ne risulterà una matrice simmetrica rispetto alla diagonale principale ($e_{ij} = e_{ji}$)




| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 |
| 2 | 1 | 0 | 1 | 0 |
| 3 | 1 | 1 | 0 | 1 |
| 4 | 1 | 0 | 1 | 0 |

# Rappresentazione con matrice di adiacenza /3

- Nel caso il grafo sia etichettato possiamo associare al nodo altre informazioni.



# Matrice di adiacenza estesa

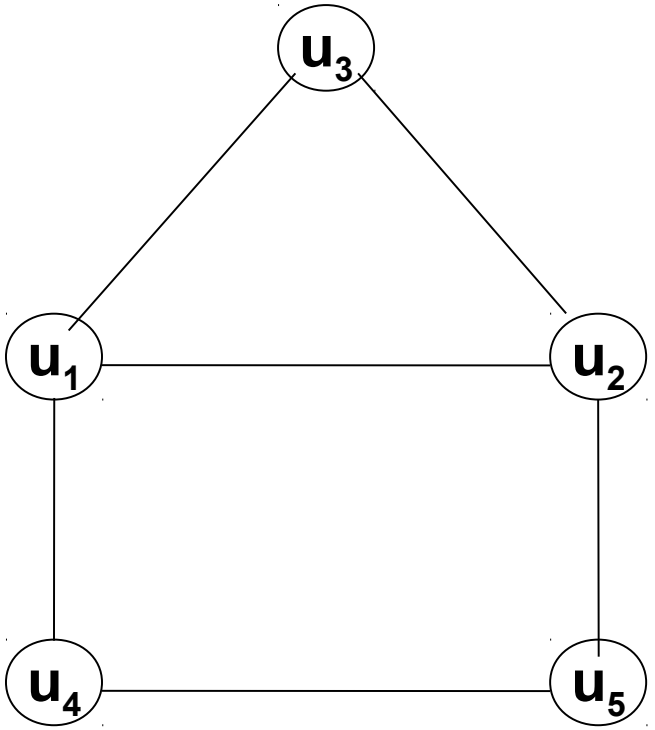
| | LABEL | MARK | ARCHI | RIGA | | | | |
|---|---|---|---|---|---|---|---|---|
| n=1 | 10 | 1 | 3 | 0 | 1 | 1 | 1 | |
| n=2 | 22 | 1 | 3 | 0 | 0 | 1 | 0 | |
| n=3 | 17 | 1 | 3 | 0 | 0 | 0 | 0 | |
| n=4 | 13 | 1 | 3 | 0 | 1 | 1 | 0 | |
| n=5 | 24 | 0 | 2 | | | | | |

- MARK è un flag che ha valore falso o 0 se il nodo è stato rimosso.
- ARCHI contiene il numero somma degli entranti h e uscenti k dal generico nodo.

# Rappresentazione con matrici di incidenza

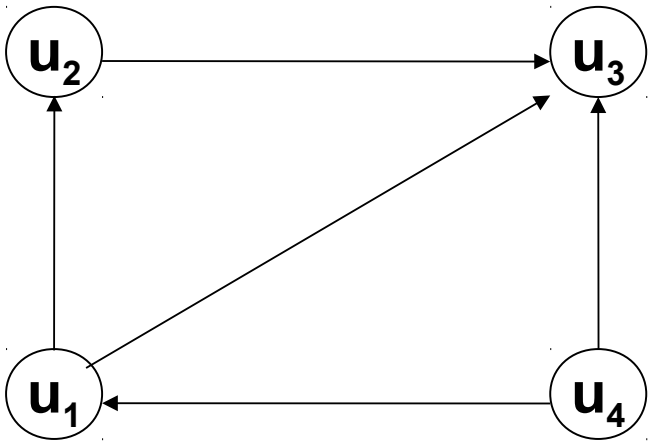
- Un grafo G = (N, A) può anche essere rappresentato mediante una matrice (N x M), B = [$b_{ik}$], nella quale ciascuna riga rappresenta un nodo e ciascuna colonna rappresenta un arco.
- Per un **grafo non orientato** $b_{ik}$ = 1 se l'arco j-esimo è incidente nel nodo i, 0 altrimenti
- Nel caso di **grafi orientati** o diretti il generico elemento $b_{ij}$ di B diviene a) +1 se l'arco j-esimo entra nel nodo i, b) -1 se l'arco j-esimo esce dal nodo i, c) 0 altrimenti

# Rappresentazione con matrici di incidenza: grafo non orientato




| | $(u_1,u_2)$ | $(u_1,u_3)$ | $(u_1,u_4)$ | $(u_2,u_3)$ | $(u_2,u_5)$ | $(u_4,u_5)$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 1 | 1 | 0 |
| 3 | 0 | 1 | 0 | 1 | 0 | 0 |
| 4 | 0 | 0 | 1 | 0 | 0 | 1 |
| 5 | 0 | 0 | 0 | 0 | 1 | 1 |

# Rappresentazione con matrici di incidenza: grafo orientato

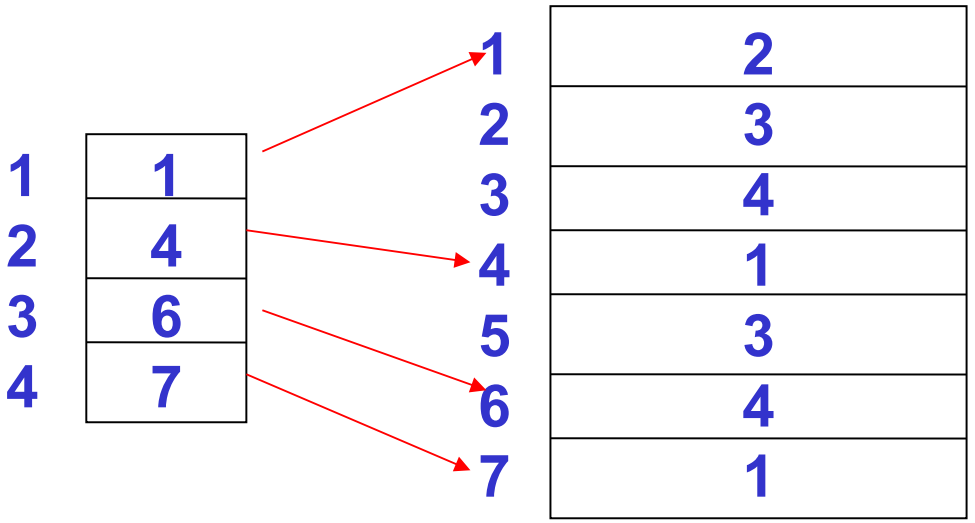


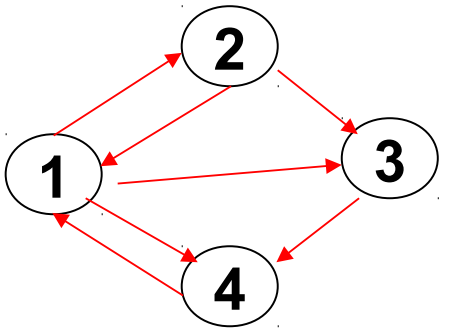

| | $(u_1,u_2)$ | $(u_1,u_3)$ | $(u_2,u_3)$ | $(u_4,u_1)$ | $(u_4,u_3)$ |
|---|---|---|---|---|---|
| 1 | -1 | -1 | 0 | +1 | 0 |
| 2 | +1 | 0 | -1 | 0 | 0 |
| 3 | 0 | +1 | +1 | 0 | +1 |
| 4 | 0 | 0 | 0 | -1 | -1 |

- Dato un nodo non è facile ricavare l'insieme di adiacenza.
- Per calcolare A(u) è necessario scandire la riga u di B alla ricerca delle colonne k t.c. $b_{uk} = -1$, e per ogni colonna k scandire l'indice di riga i t.c. $b_{ik} = +1$.

# Rappresentazione con vettore di adiacenza: grafo orientato

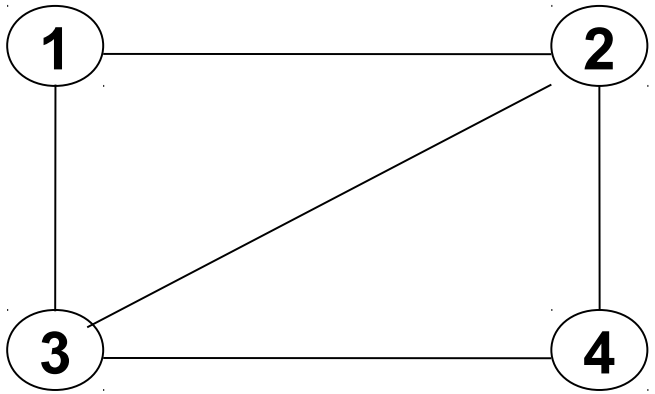
- E' possibile rappresentare il grafo (N, A) con due vettori, il vettore NODI e il vettore ARCHI.
  - Il vettore NODI è formato da N elementi e NODI(i) contiene un cursore alla posizione di ARCHI a partire dalla quale è memorizzato A(i).
    - Per semplicità denotiamo i nodi con interi. Nel caso il grafo sia etichettato possiamo associare al nodo altre informazioni.

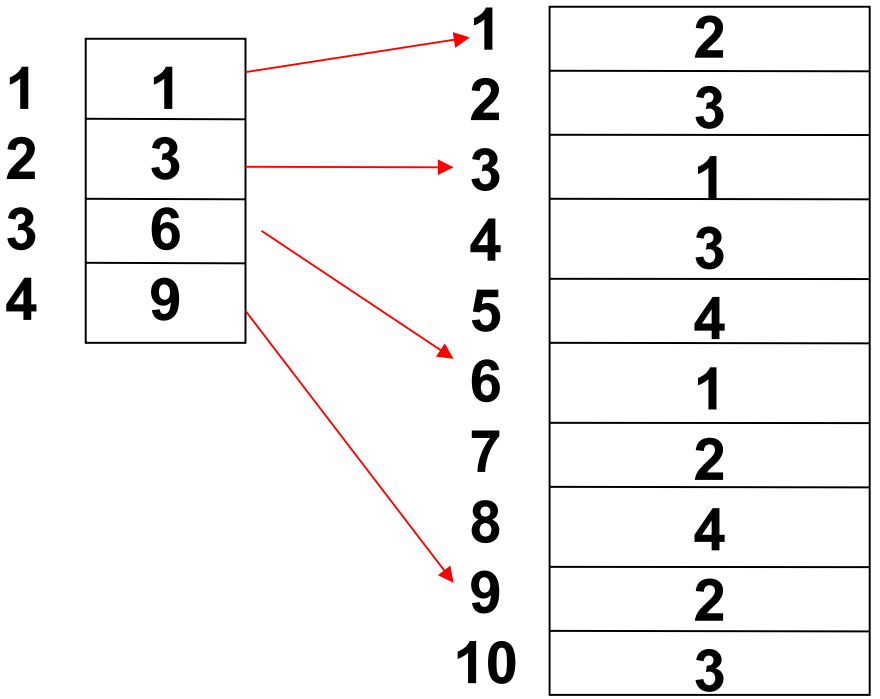


# Rappresentazione con vettore di adiacenza: grafo non orientato

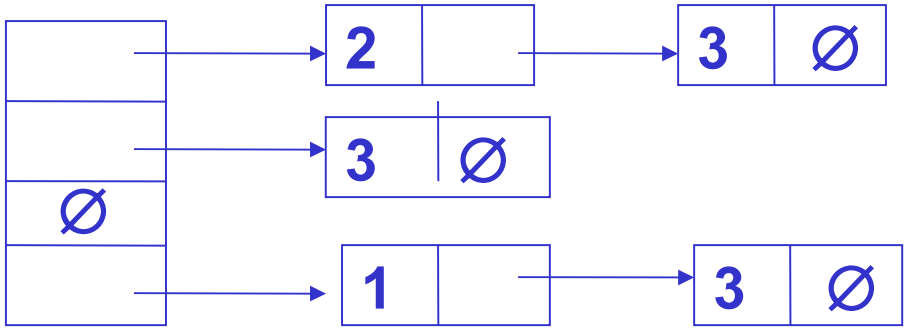
- E' necessario rappresentare ogni arco due volte
- Se i grafi sono etichettati sui nodi e/o sugli archi i pesi possono essere memorizzati in vettori PESINODI(n) e PESIARCHI(m)

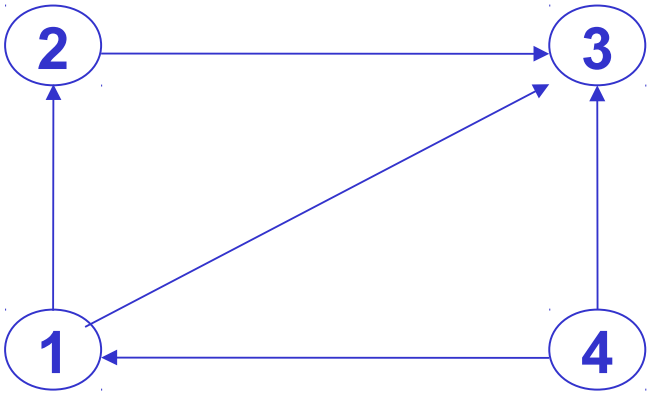


# Rappresentazione con lista di adiacenza

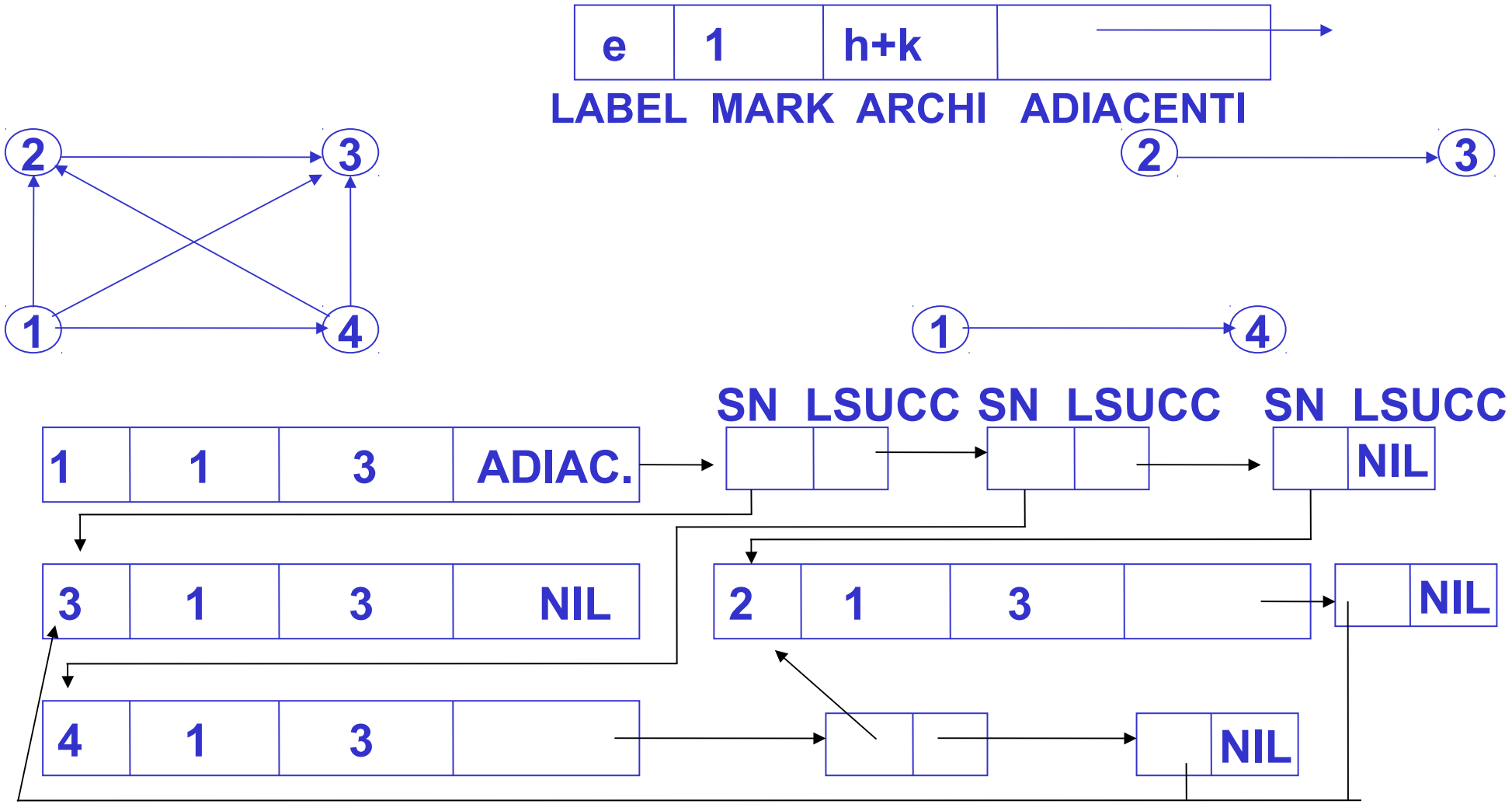
- E' possibile anche utilizzare un vettore di nodi A ed n liste.
  - una generica componente A(i) del vettore è il puntatore alla lista i-esima in cui sono memorizzati i nodi adiacenti a i.



# 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 è rappresentato mediante un record



# Visita 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 è 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 è 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.

# Depth first search (DFS)

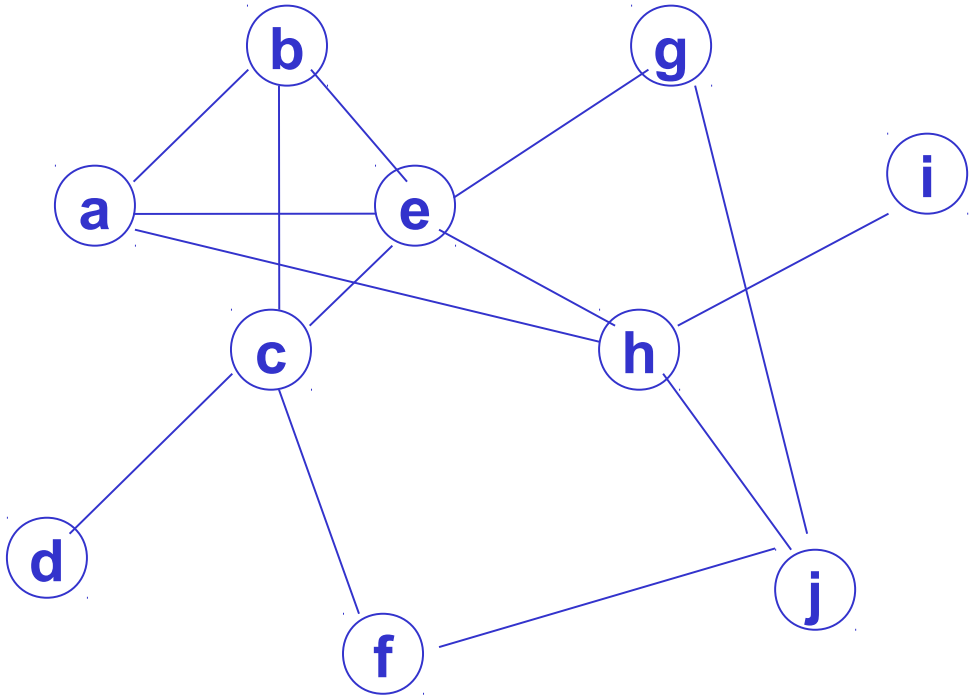
- L'algoritmo prevede che contrassegnamo un vertice appena lo visitiamo e poi cerchiamo di spostarci in un vertice adiacente non contrassegnato. Se non ci sono vertici non marcati indietreggiamo lungo i vertici già visitati finché arriviamo ad un vertice  che risulta adiacente ad uno non visitato e continuiamo così il procedimento.

```
DFS (G: GRAFO; u:NODO)
    ESAMINA IL NODO u E MARCALO “VISITATO”
    for (TUTTI I v ϵ A(u)) do
        ESAMINA L’ARCO (u,v)
        if v NON E' “VISITATO” then
            DFS(G,v)
```

# Depth first search (DFS) /2

Chiamate ricorsive

```
a                       A(b,e,h)
   b                    A([a],c,e)
      c                 A([b],d,e,f)
         d              A([c])
         e              A([a,b,c],g,h)
            g           A([e],j)
               j        A([g],f,h)
                  f     A([c,j])
                  h     A([a,e,j],i)
                     i  A([h])
```



# Breadth first search (BFS)

- In questo schema ogni vertice adiacente al vertice corrente è visitato prima di spostarsi dal vertice 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 è il minimo numero di archi in un cammino da u a v.
- Conviene tenere in una coda i vertici visitati ma non completamente “esaminati” così che, quando siamo pronti a spostarci su un vertice adiacente al corrente, possiamo ritornare al vecchio vertice corrente dopo il nostro movimento.

# Breadth first search (BFS) /2

```
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”
    for (TUTTI I NODI v ∈ A(u)) do
      esamina l'arco (u,v)
      if v non è marcato “visitato” and v ∉ Q then
        INCODA(v,Q)
```

# Algoritmo generale di visita

- Esaminato un nodo iniziale i, si visita un nodo j collegato a i e, successivamente, si considera un nuovo nodo k, diverso da i e da j, e adiacente a i oppure  a j.
- Si utilizza un insieme R dei nodi visitati e un insieme Q dei nodi utili per il proseguimento della ricerca.
- Infine è utilizzato l'insieme B in cui vengono memorizzati gli archi utilizzati per la scoperta dei nuovi nodi

# Algoritmo generale di visita /2

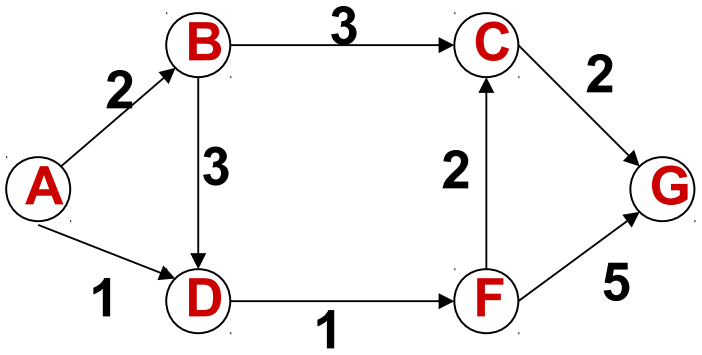
```
Q = N
B = {}
repeat
    “SCEGLI UN NODO j da Q”
    if (ESISTE UN NODO NON VISITATO ADIACENTE A j) then
        “VISITA E PONI IN R E IN Q UNO O PIU’ NODI k1,k2,…
         ADIACENTI A j E NON ANCORA VISITATI”
        “PONI IN B GLI ARCHI (j, k1),(j, k2),…”
    else “ELIMINA j DA Q”
until (Q = {})
```

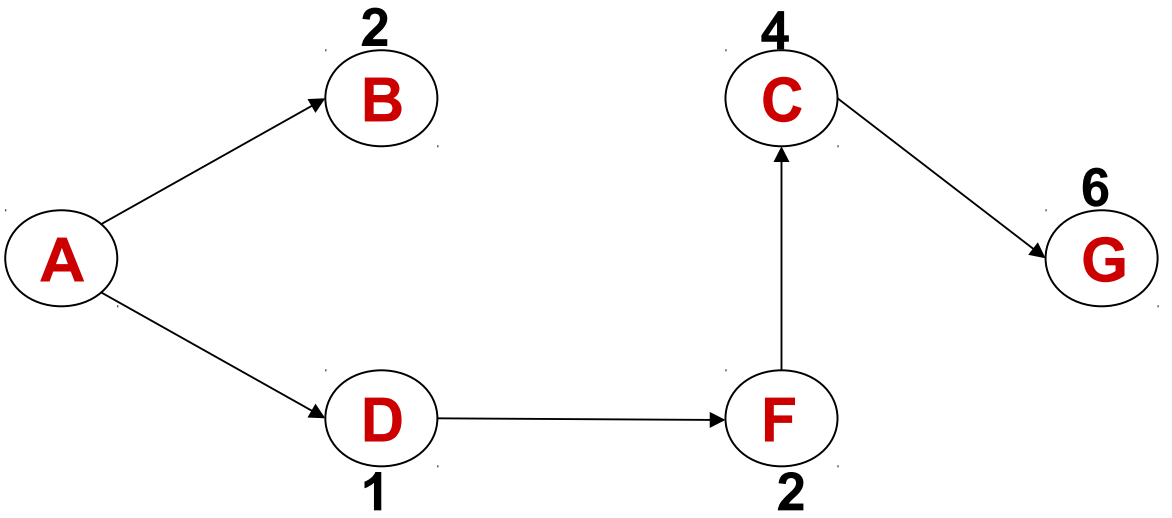
# Algoritmo generale di visita: note

- Il criterio di scelta del nodo stabilisce la modalità di visita.
- Nella ricerca in profondità (depth-first), che è diretta estensione della visita in ordine anticipato di un albero radicato, tra i vari nodi utili per il proseguimento della ricerca viene sempre scelto quello visitato più recentemente.
- Nella ricerca in ampiezza (breadth-first) i nodi sono visitati in ordine di distanza crescente dal nodo di partenza r, dove la distanza da r ad un generico nodo u è il numero di archi in un cammino da r ad u che sia il più corto possibile. Dunque la ricerca in ampiezza fornisce sempre un cammino “ottimo”.

- 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 è connesso oppure no
  - trovare tutti i sottografi connessi di un grafo non orientato (componenti connesse)
- Inoltre, sia la visita DFS che la visita BFS di un grafo, connettendo i nodi marcati ad uno ad uno a partire da un generico nodo i, generano un albero radicato nel nodo iniziale i.
  - 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 (ricerca dell'albero minimo di copertura o di attraversamento).

- Un problema classico è quello della individuazione in un grafo orientato del percorso più breve che unisce due generici nodi del grafo.
- Sia G = (N, A) un grafo orientato etichettato negli archi con valori interi positivi (pesi). Partendo dal nodo r $\in$ N, per ogni nodo u $\in$ N, si calcola il percorso (cammino o catena) che connette r ad u tale che la somma dei pesi associati agli archi che compongono il cammino sia minima.
  - Esempio: i nodi sono città, gli archi sono le strade che le collegano, le etichette sugli archi sono le distanze.
  - Il problema consiste nel trovare il percorso più breve tra tutti quelli che connettono la città a alle altre città.


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


A
B
C
D
F
G
2
4
6
1
2