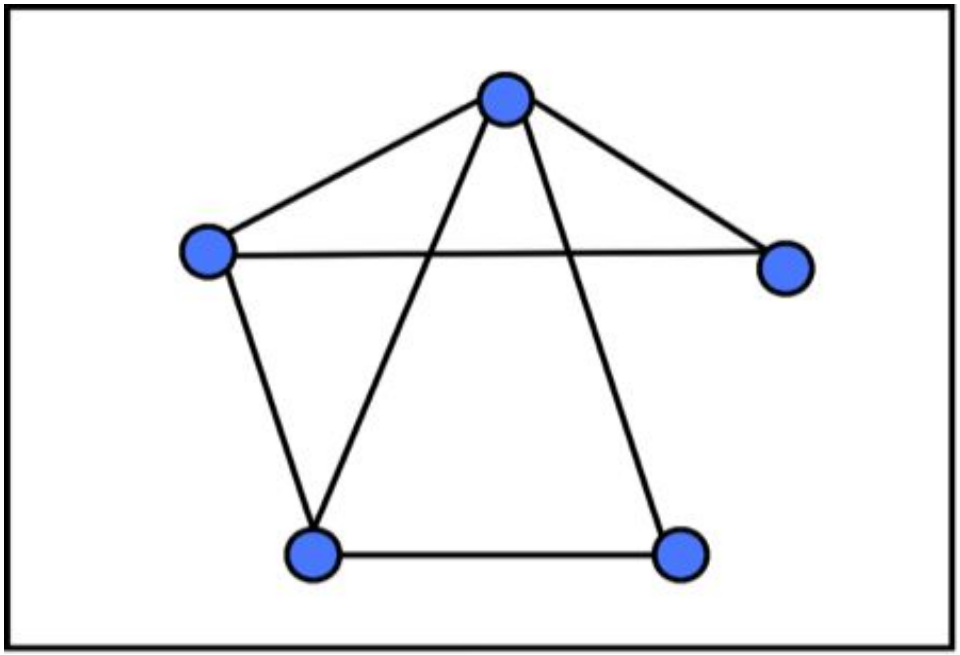
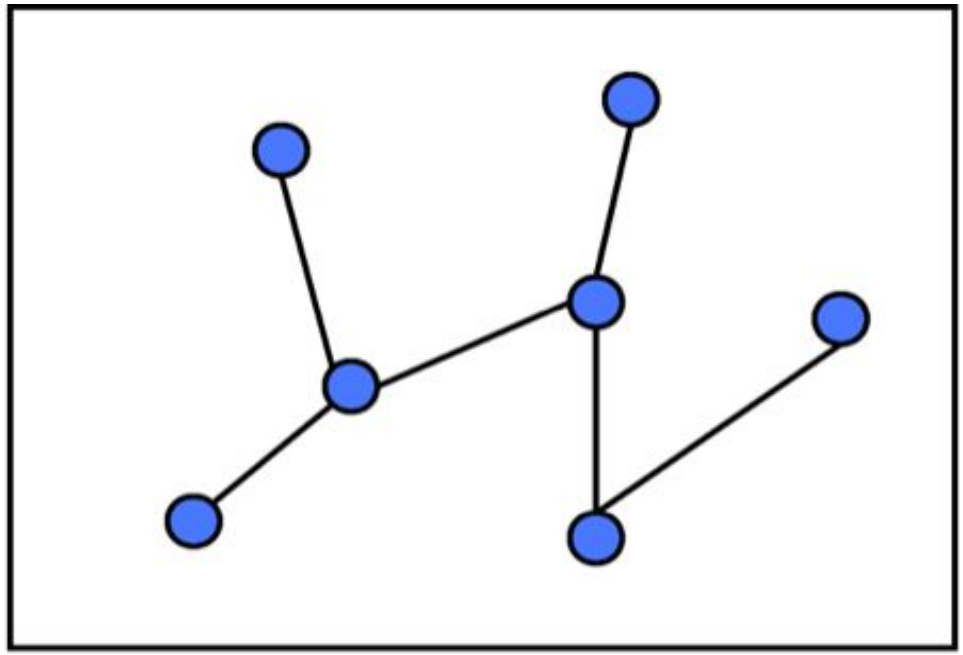
Algoritmi e Strutture di Dati

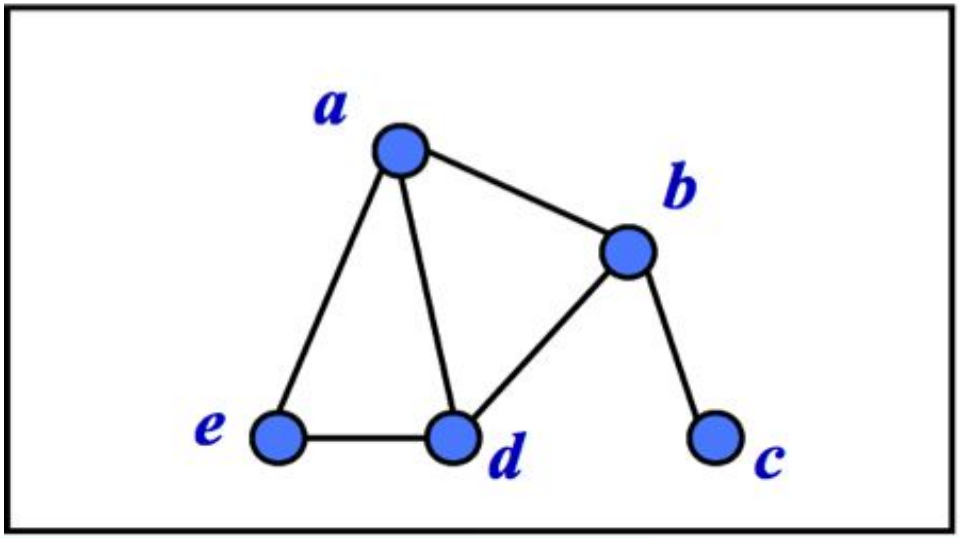
Grafi

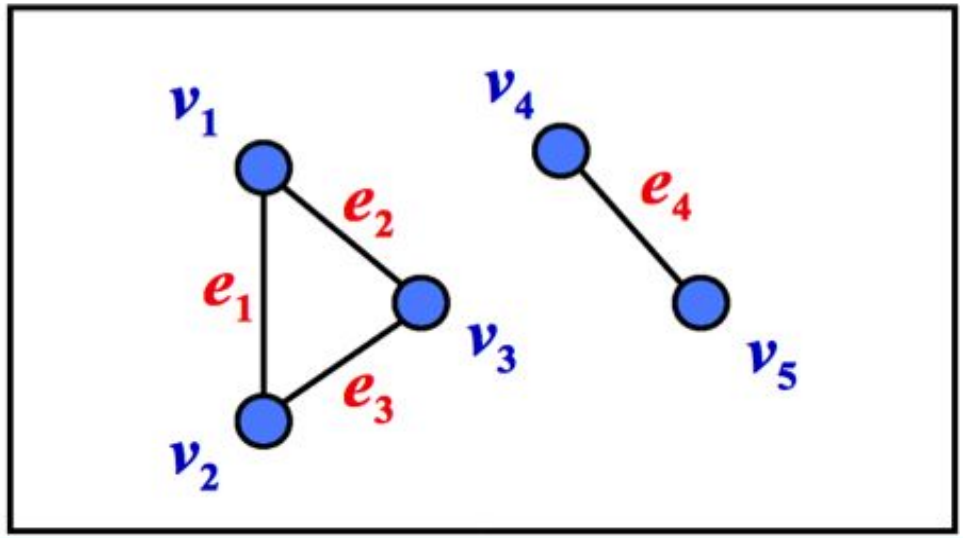
Jocelyne Elias, Alberto Montresor e Gianluigi Zavattaro

Esempi di grafi



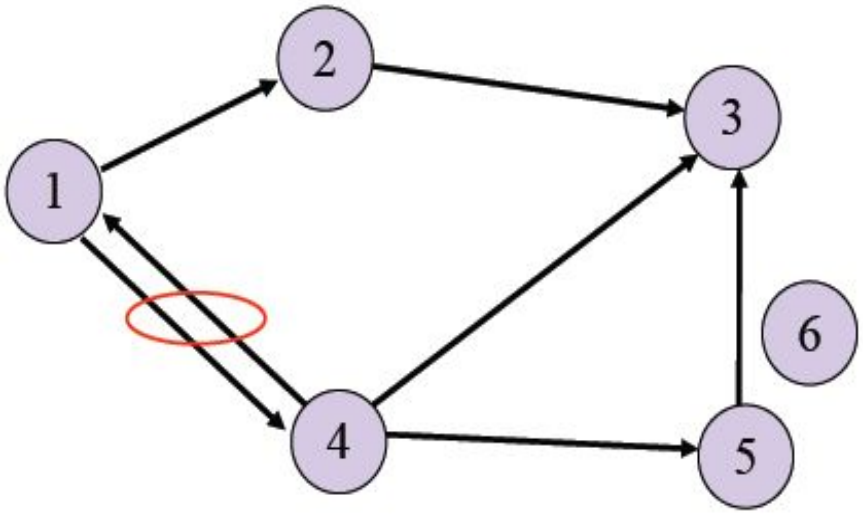






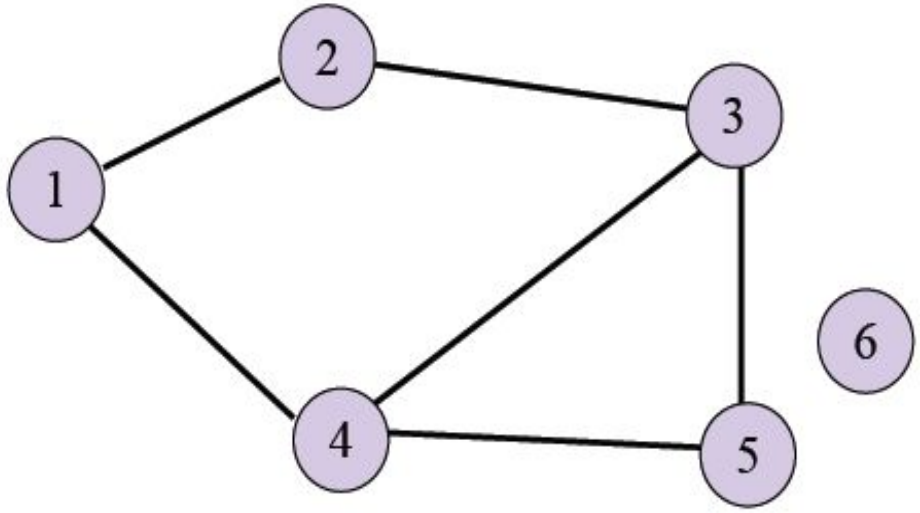
Grafi orientati e non orientati: definizione

- Un *grafo orientato* G è una coppia (V, E) dove:
  - Insieme finito dei vertici V
  - Insieme degli archi E: relazione binaria tra vertici



$V = \{ 1, 2, 3, 4, 5, 6 \}$
$E = \{ (1,2), (1,4), (2,3), (4,1), (4,3), (4,5), (5,3) \}$

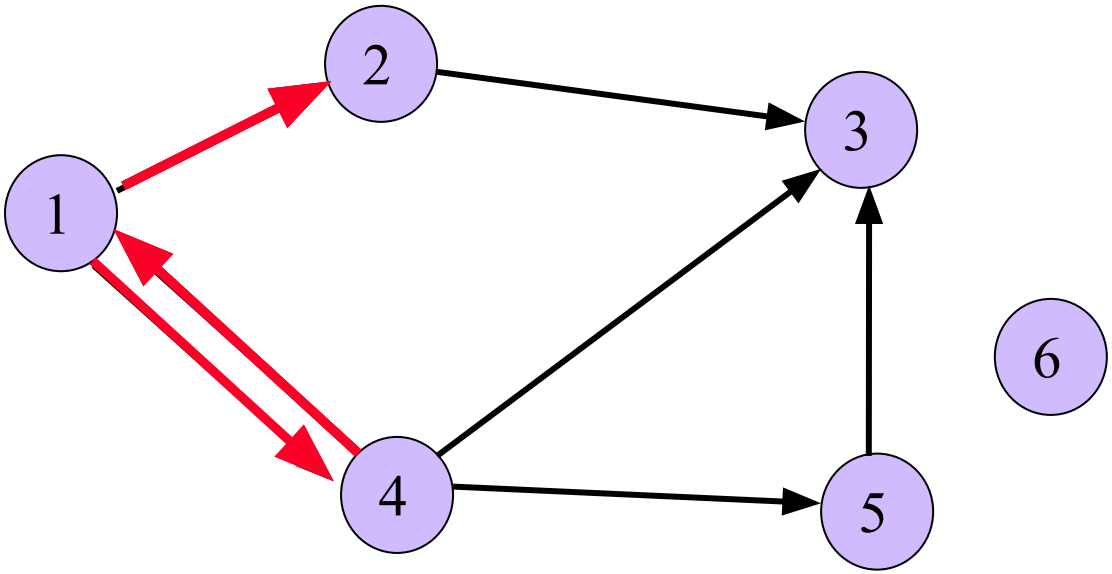
- Un *grafo non orientato* G è una coppia (V, E) dove:
  - Insieme finito dei vertici V
  - Insieme degli archi E: coppie non ordinate



$V = \{1, 2, 3, 4, 5, 6 \}$
$E = \{ [1,2], [1,4], [2,3], [3,4], [3,5], [4,5] \}$

Definizioni: incidenza e adiacenza

- **In un grafo orientato un arco (u,v) si dice *incidente* da u in v**
- **In un grafo non orientato la relazione di adiacenza tra vertici è simmetrica**
- **Un vertice v si dice *adiacente* a u se e solo se $(u, v) \in E$**



L'arco (1,2) è incidente da 1 a 2
(1,4) è incidente da 1 a 4
(4,1) è incidente da 4 a 1

2 è adiacente ad 1
3 è adiacente a 2, 4, 5
1 è adiacente a 4 e viceversa
2 non è adiacente a 3,4
6 non è adiacente ad alcun vertice

Rappresentazione grafi

- Poniamo
  - $n = |V|$ numero di vertici (o nodi) di un grafo
  - $m = |E|$ numero di archi

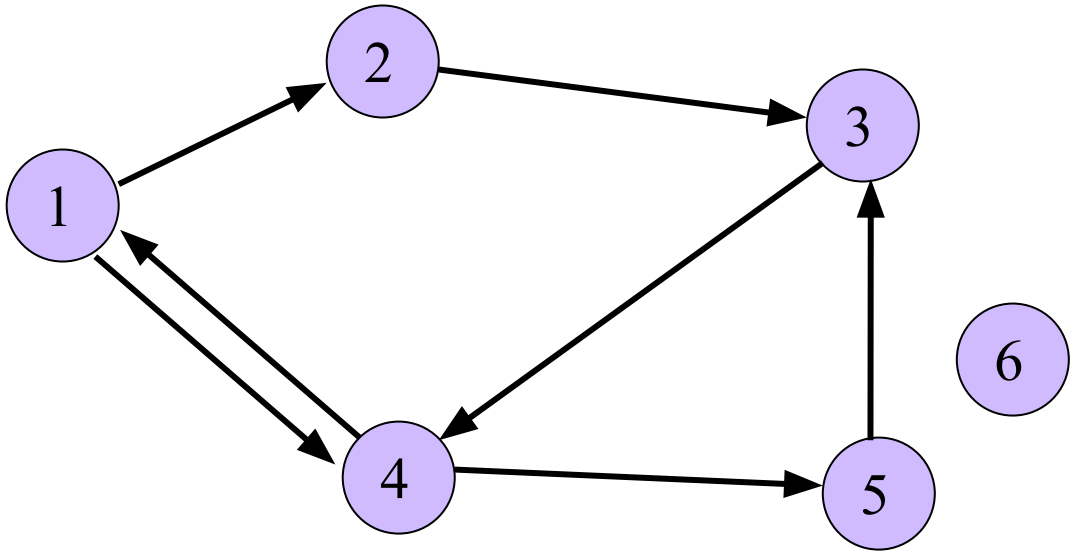
- Matrice di adiacenza
  - Spazio richiesto $O(n^2)$
  - Verificare se il vertice u è adiacente a v richiede tempo $O(1)$
  - Elencare tutti gli archi costa $O(n^2)$

- Liste di adiacenza
  - Spazio richiesto $O(n+m)$
  - Verificare se il vertice u è adiacente a v richiede tempo $O(n)$
  - Elencare tutti gli archi costa $O(n+m)$

Matrice di adiacenza: grafo orientato o non orientato

$$m_{uv} = \begin{cases} 1, & \text{se } (u,v) \in E, \\ 0, & \text{se } (u,v) \notin E. \end{cases}$$

Spazio: n^2

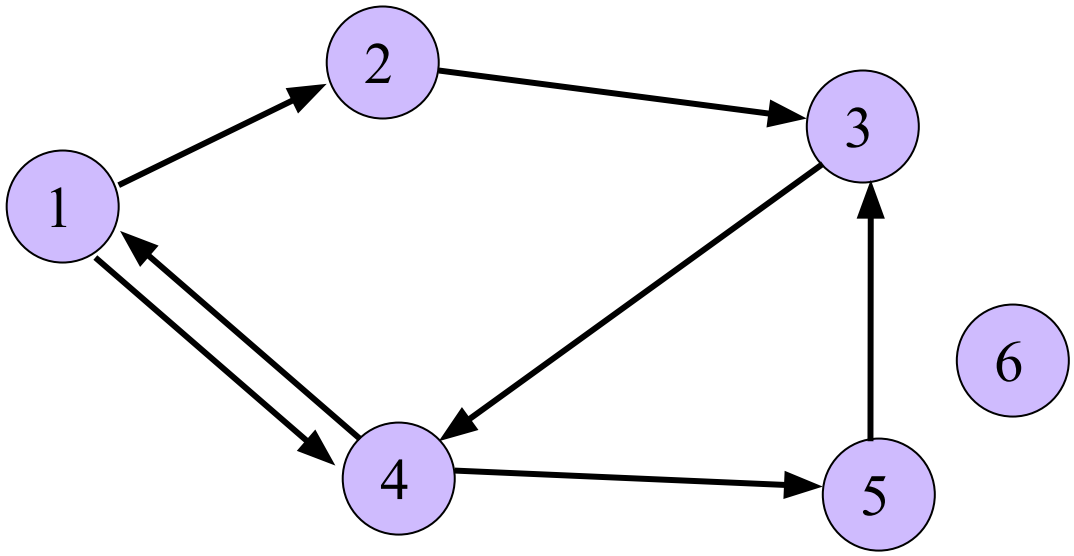


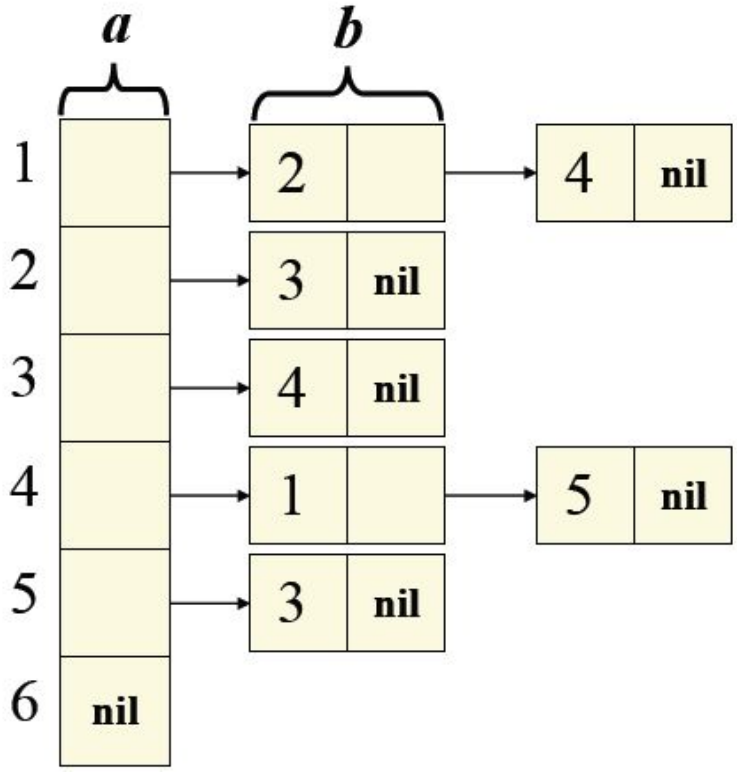
	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	1	0	0	0
3	0	0	0	1	0	0
4	1	0	0	0	1	0
5	0	0	1	0	0	0
6	0	0	0	0	0	0

Lista di adiacenza: grafo orientato

$G.adj(u) = \{ v \mid (u,v) \in E \}$

Spazio: $a \cdot n + b \cdot m$



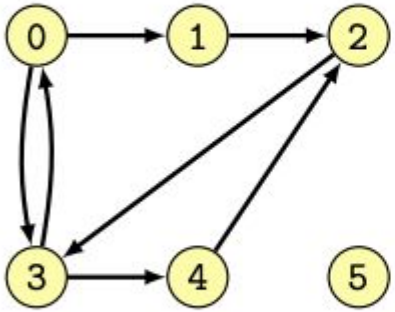


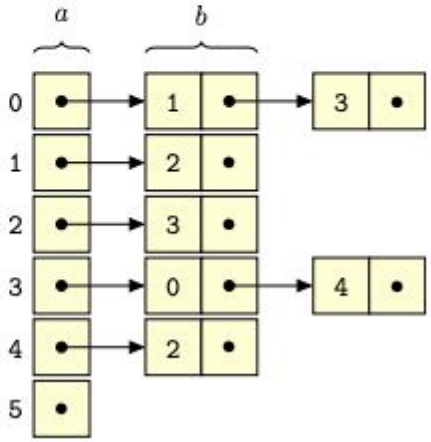
Lista e matrice di adiacenza: altri esempi …

Liste di adiacenza: grafi orientati

$G.\text{adj}(u) = \{v | (u,v) \in E\}$

Spazio $= an + bm$ bit

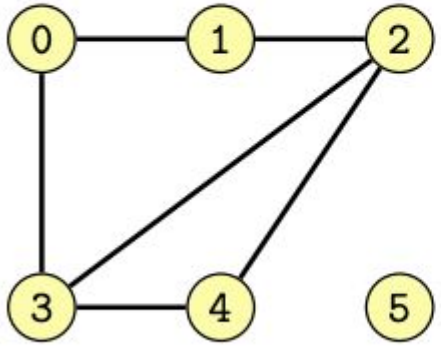




Matrice di adiacenza: grafi non orientati

$$m_{uv} = \begin{cases} 1 & (u,v) \in E \\ 0 & (u,v) \notin E \end{cases}$$

Spazio $= n^2$ oppure $n(n-1)/2$ bit

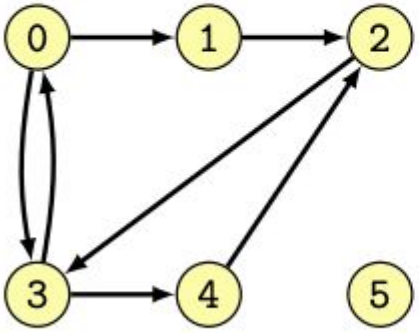


	0	1	2	3	4	5
0		1	0	1	0	0
1			1	0	0	0
2				1	1	0
3					1	0
4						0
5						

Matrice di adiacenza: grafi orientati

$$m_{uv} = \begin{cases} 1 & (u,v) \in E \\ 0 & (u,v) \notin E \end{cases}$$

Spazio $= n^2$ bit

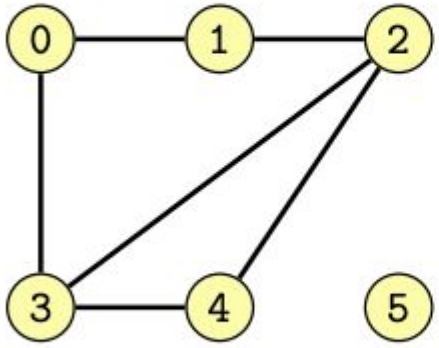


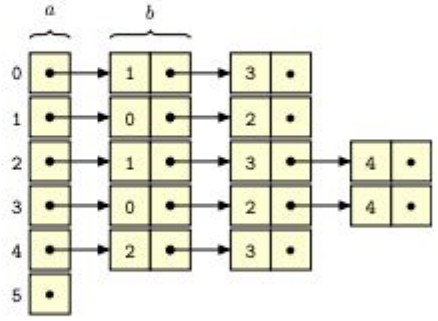
	0	1	2	3	4	5
0	0	1	0	1	0	0
1	0	0	1	0	0	0
2	0	0	0	1	0	0
3	1	0	0	0	1	0
4	0	0	1	0	0	0
5	0	0	0	0	0	0

Liste di adiacenza: grafo non orientato

$G.\text{adj}(u) = \{v | (u,v) \in E\}$

Spazio $= an + 2 \cdot bm$

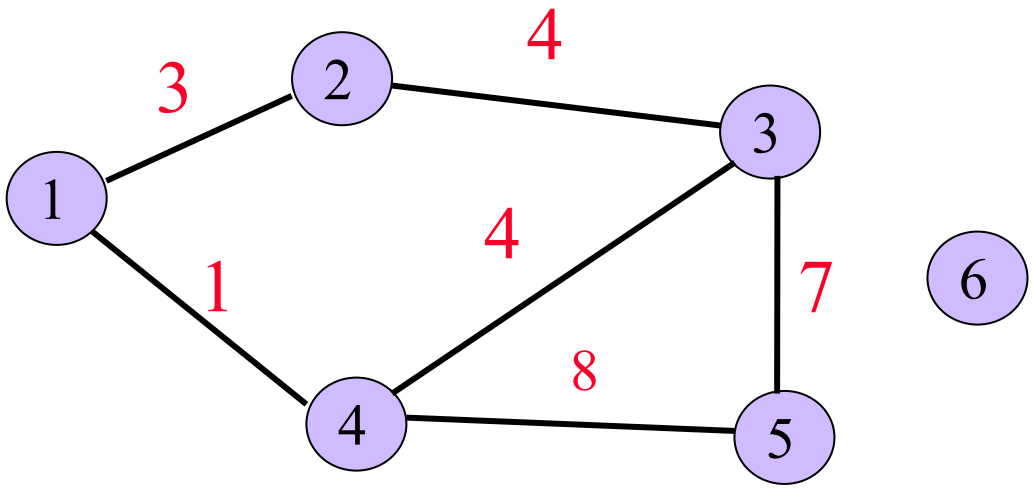




Grafi pesati

- In alcuni casi ogni arco ha un *peso* (*costo*, *lunghezza*, *guadagno*) associato
  - Il peso può essere determinato tramite una funzione
    p: $V \times V \rightarrow$ R, dove R è l'insieme dei numeri reali
  - Quando tra due vertici non esiste un arco, il peso è infinito

$$m_{uv} = \begin{cases} p_{uv}, & \text{se } (u,v) \in E, \\ +\infty \text{ (oppure } -\infty) & \text{se } (u,v) \notin E. \end{cases}$$



	1	2	3	4	5	6
1	*	3	*	1	*	*
2	3	*	4	*	*	*
3	*	4	*	4	7	*
4	1	*	4	*	8	*
5	*	*	7	8	*	*
6	*	*	*	*	*	*

Specifica

GRAPH

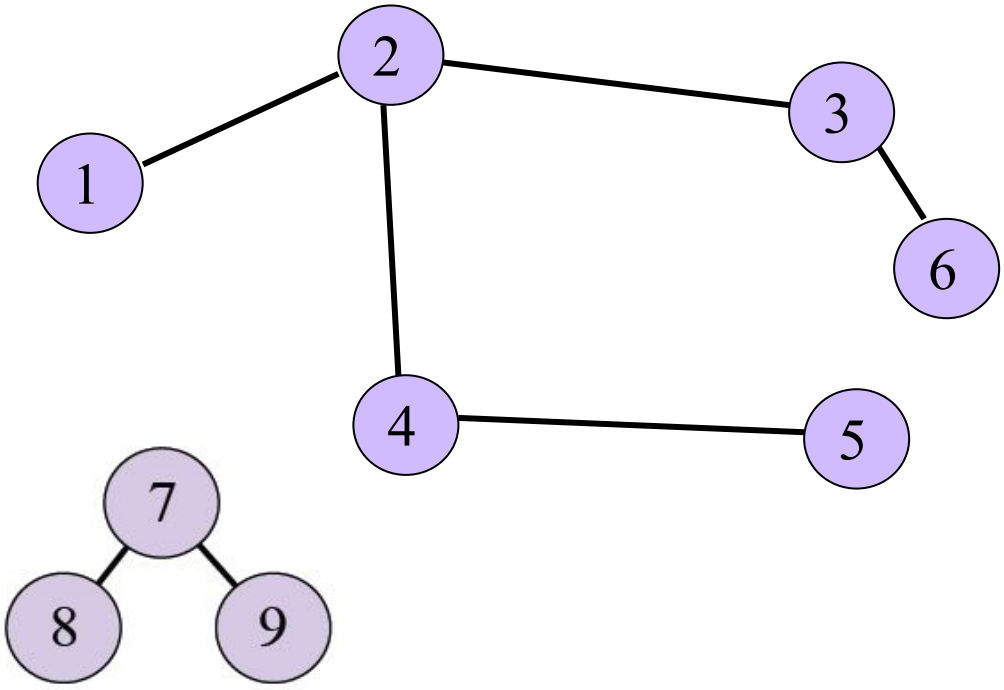
Graph( )	% Crea un grafo vuoto
insertNode(NODE u)	% Aggiunge il nodo u al grafo
insertEdge(NODE u, NODE v)	% Aggiunge l'arco (u, v) al grafo
deleteNode(NODE u)	% Rimuove il nodo u dal grafo
deleteEdge(NODE u, NODE v)	% Rimuove l'arco (u, v) nel grafo
SET adj(NODE u)	% Restituisce l'insieme dei nodi adiacenti ad u
SET V()	% Restituisce l'insieme di tutti i nodi

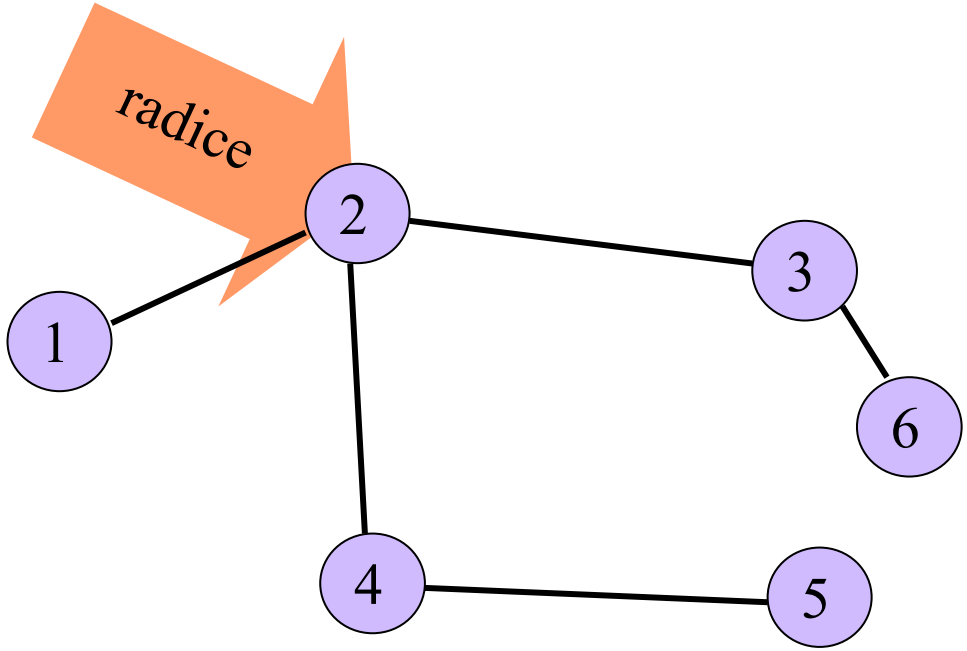
```
foreach u ∈ G.V() do
    foreach v ∈ G.adj(u) do
        fai un'operazione sull'arco (u, v)
```

- **Complessità**
  - O(n+m) liste di adiacenza
  - O(n^2) matrice di adiacenza
  - O(m) "operazioni"

Definizioni: Alberi

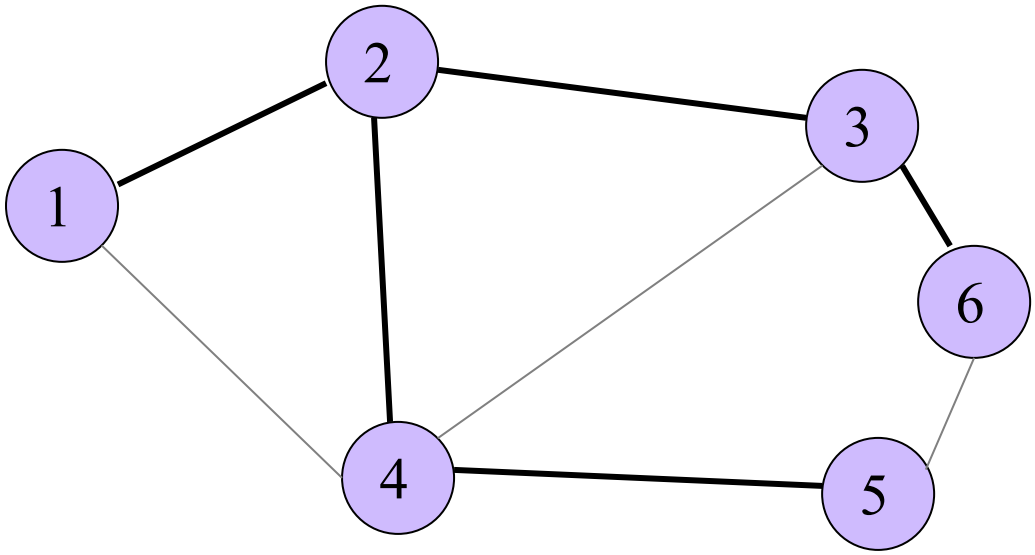
- Un *albero libero* è un grafo non orientato connesso e aciclico (⇔ connesso col minimo numero di archi ⇔ connesso con una sola catena tra ogni coppia di nodi)
- Se qualche vertice è detto radice, otteniamo un *albero radicato*
- Un insieme di alberi è detta *foresta*





Definizioni: Alberi di copertura

- In un grafo non orientato $G=(V, E)$
  - un albero di copertura T è un albero libero $T = (V, E')$ composto da tutti i nodi di V e da un sottoinsieme di [$n - 1$] archi ($E' \subseteq E$) [ogni coppia di nodi del grafo è connessa da una sola catena nell'albero]



Problema: Attraversamento grafi

- **Definizione del problema**
  - Dato un grafo $G=(V, E)$ ed un vertice r di V (detto *sorgente* o *radice*), visitare ogni vertice raggiungibile nel grafo dal vertice r
  - Ogni nodo deve essere visitato una volta sola
- **Visita in ampiezza (breadth-first search)**
  - Visita i nodi "espandendo" la frontiera fra nodi scoperti / da scoprire
  - Esempi: Cammini più brevi da singola sorgente
- **Visita in profondità (depth-first search)**
  - Visita i nodi andando il "più lontano possibile" nel grafo
  - Esempi: Ordinamento topologico (per DAG, iniziando la visita dai nodi senza archi entranti)