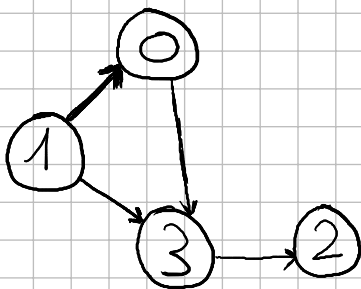


# Come si rappresenta un grafo?

Esistono 3 modi per rappresentare i grafi:

- Elenco di Archi
- Matrice di Adiacenza
- Lista di Adiacenza

## Elenco di Archi



num archi:

4

$(1,0)$   
 $(1,3)$   
 $(0,3)$   
 $(3,2)$

per realizzare questa rappresentazione  
potresti fare un array di  
struct, tipo:

```
{  
    arco uscente: x  
    arco entrante: y  
}
```

In generale hai:

n

$(v_1, v_1)$

$(v_2, v_2)$

...

$(v_n, v_n)$

Costo operazioni:

esiste l'arco  $(u,v)$ ?  $O(m)$  perché devi scorrere tutti gli archi

quali sono i vicini di un nodo  $v$ ?  $O(m)$  // vicina di 3? 2

$\delta_{out}(v)$ ?  $O(m)$

$\delta_{in}(v)$ ?  $O(m)$

$\delta(v)$ ?  $O(m)$

Aggiunta di un arco  $(u,v)$ ?  $O(1)$

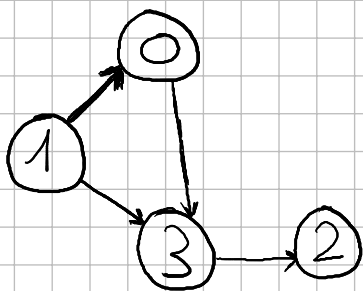
Eliminazione di un arco  $(u,v)$ ?  $O(m)$  // bisogna prima trovare l'arco

Contare num archi  $(|E|)$ ?  $O(m)$

Queste operazioni risultano costose in tempo perché prima bisogna trovare l'arco o nodo interessato.

Costo in spazio:  $O(m)$

## Matrice di Adiacenza



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

righe = archi uscenti

colonne = archi entranti

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

Matrice di dimensione  $n \times n$

num nodi  $\times$  num nodi

↑  
sempre zero su questa diagonale perché non esiste

## Costo operazioni:

esiste l'arco  $(u,v)$ ?  $O(1)$  // basta fare accesso posizionale

quali sono i vicini di un nodo  $v$ ?  $O(n)$  // Scorro colonna del nodo  
avere  $n$  invece  
che  $n$  è ottimo, soprattutto  
se il grafo è denso

$\delta_{out}(v)$ ?  $O(n)$

$\delta_{in}(v)$ ?  $O(n)$

$\delta(v)$ ?  $O(n)$

Aggiunta di un arco  $(u,v)$ ?  $O(1)$

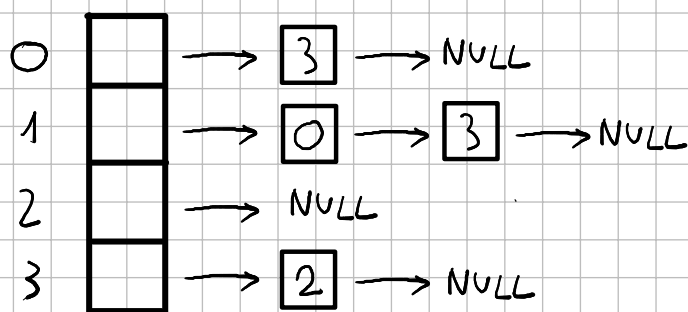
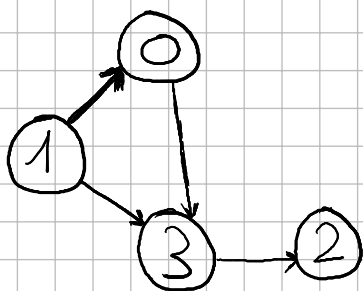
Eliminazione di un arco  $(u,v)$ ?  $O(1)$

Contare num archi  $(|E|)$ ?  $O(n^2)$  // unico caso in cui scorro  
tutta la matrice

Confronto all'elenco di archi abbiamo un netto miglioramento,  
dato che è sempre meglio avere costi nell'ordine di  $n$  e non di  $m$ ,  
soprattutto se un grafo è denso.

Costo in Spazio:  $O(n^2)$   $\rightarrow$  questo costo è fisso, anche  
se il grafo non ha archi.

## Lista di Adiacenza



Gli elementi della lista sono gli archi uscenti del rispettivo nodo.

## Costo operazioni:

esiste l'arco  $(u,v)$ ?  $O(\sum_{out}(u))$  // questo è dovuto dal fatto che la dim della lista di un nodo è uguale al grado uscente ( $\sum_{out}$ )

quali sono i vicini di un nodo  $v$ ?  $O(\sum_{out}(v))$

$\sum_{out}(u)$ ?  $O(\sum_{out}(u))$  // significa che per sapere il grado uscente di un nodo devi scorrere tutta la lista, che ha dimensione  $\sum_{out}(u)$

$\sum_{in}(u)$ ?  $O(n+m)$  // perché devi scorrere tutte le liste di ogni nodo

$\delta(u)$ ?  $O(n+m)$

Aggiunta di un arco  $(u,v)$ ?  $O(1)$  // head insert

Eliminazione di un arco  $(u,v)$ ?  $O(\sum_{out}(u))$

Contare num archi  $(|E|)$ ?  $O(n+m)$

Costo Spazio:  $O(n+m)$  // se grafo sparso lista meglio di matrice

se grafo denso lista praticamente uguale alla matrice  
però nella matrice le operazioni costano meno

# Quando conviene una rappresentazione rispetto ad un'altra?

L'elenco di archi non conviene praticamente mai, l'unico vantaggio è il costo in spazio ridotto.

La matrice di adiacenza ha costi ottimi per le operazioni, ma è onerosa in termini di spazio.

La lista di adiacenza è un mix bilanciato, perché se il grafo non è particolarmente denso  $\text{Spazio}(u)$  è un buon costo, ma se il grafo è denso  $\text{Spazio}(u)$  tende a  $n+m$ , quindi peggiore rispetto alla matrice.

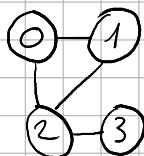
Per la complessità in spazio vale la stessa cosa.

## Tutto molto bello, ma i grafi non ordinati?

Anche per i grafi non ordinati si possono usare queste 3

rappresentazioni, semplicemente duplichi gli archi, scrivendoli da entrambi i versi. I costi ovviamente non variano.

Esempio



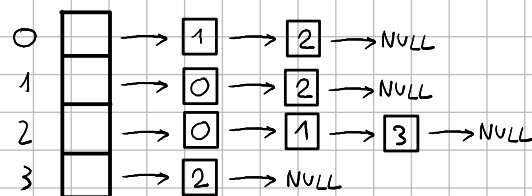
Elenco di archi:

8  
(0,1)  
(1,0)  
(0,2)  
(2,0)  
(1,2)  
(2,1)  
(2,3)  
(3,2)

Matrice di adiacenza:

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

Lista di adiacenza:

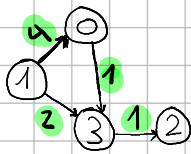


Se noti è una matrice simmetrica.

# E per i pesi?

- elenco di archi: si aggiunge un parametro

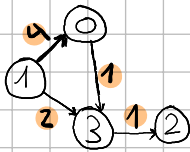
Es:



4  
(1,0,4)  
(1,3,2)  
(0,3,1)  
(3,2,1)

- matrice di adiacenza: invece di mettere 1 si mette il peso.

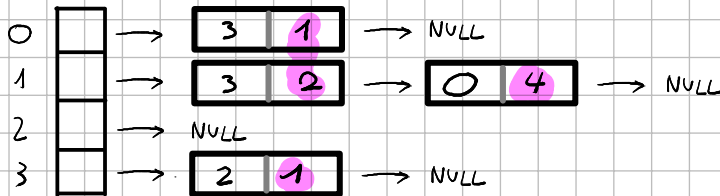
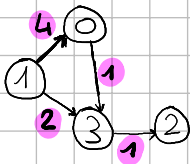
Es:



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

- lista di adiacenza: invece di mettere il numero del nodo metto una struct con nodo + peso.

Es:



Esiste un quarto modo per rappresentare i grafi, ovvero le liste di incidenza.

Si salvano gli archi invece che i nodi.

Es:

