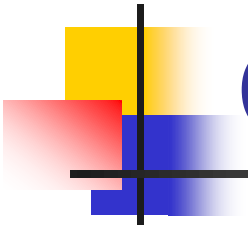
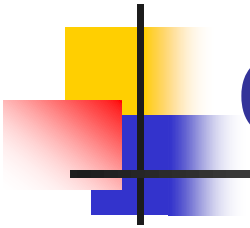
Richiami di matematica discreta: grafi e alberi

Paolo Camurati
Dip. Automatica e Informatica
Politecnico di Torino



Grafi

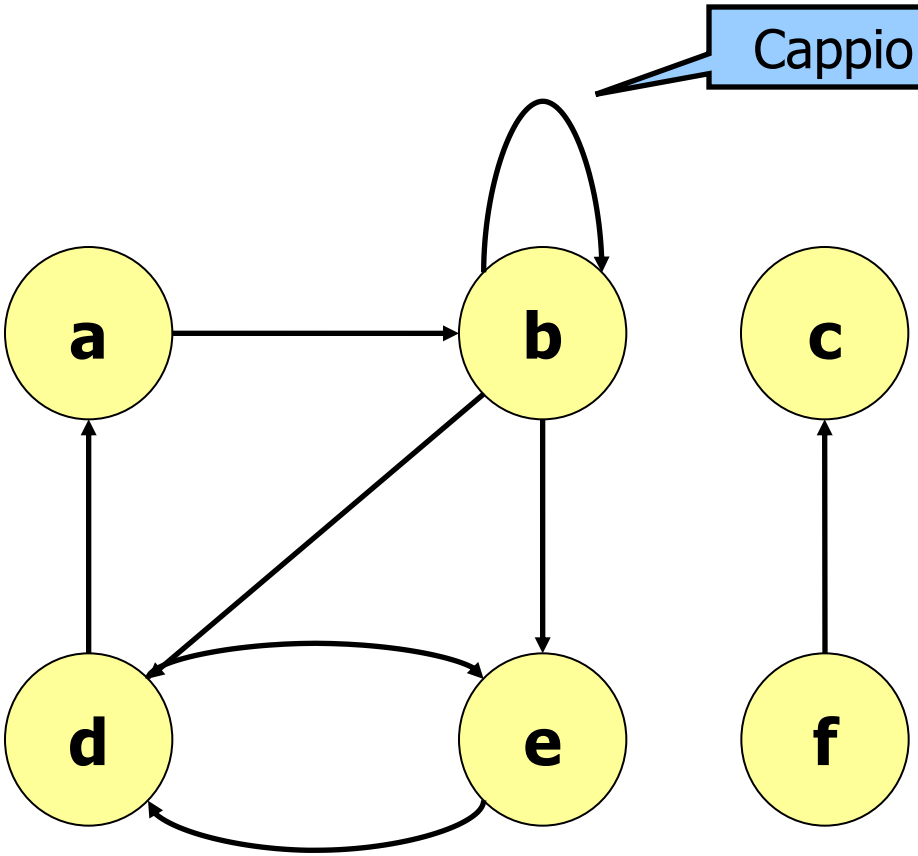
- Definizione: G = (V,E)
  - V: insieme finito e non vuoto di vertici (dati semplici o composti)
  - E: insieme finito di archi, che definiscono una relazione binaria su V
- Grafi orientati/non orientati:
  - orientati: arco = coppia ordinata di vertici $(u, v) \in E$ e $u, v \in V$
  - non orientati: arco = coppia non ordinata di vertici $(u, v) \in E$ e $u, v \in V$



Grafi come modelli

Dominio	Vertice	Arco
comunicazioni	telefono, computer	fibra ottica, cavo
circuiti	porta, registro, processore	filo
meccanica	giunto	molla
finanza	azioni, monete	transazioni
trasporti	aeroporto, stazione	corridoio aereo, linea ferroviaria
giochi	posizione sulla scacchiera	mossa lecita
social networks	persona	amicizia
reti neurali	neurone	sinapsi
composti chimici	molecola	legame

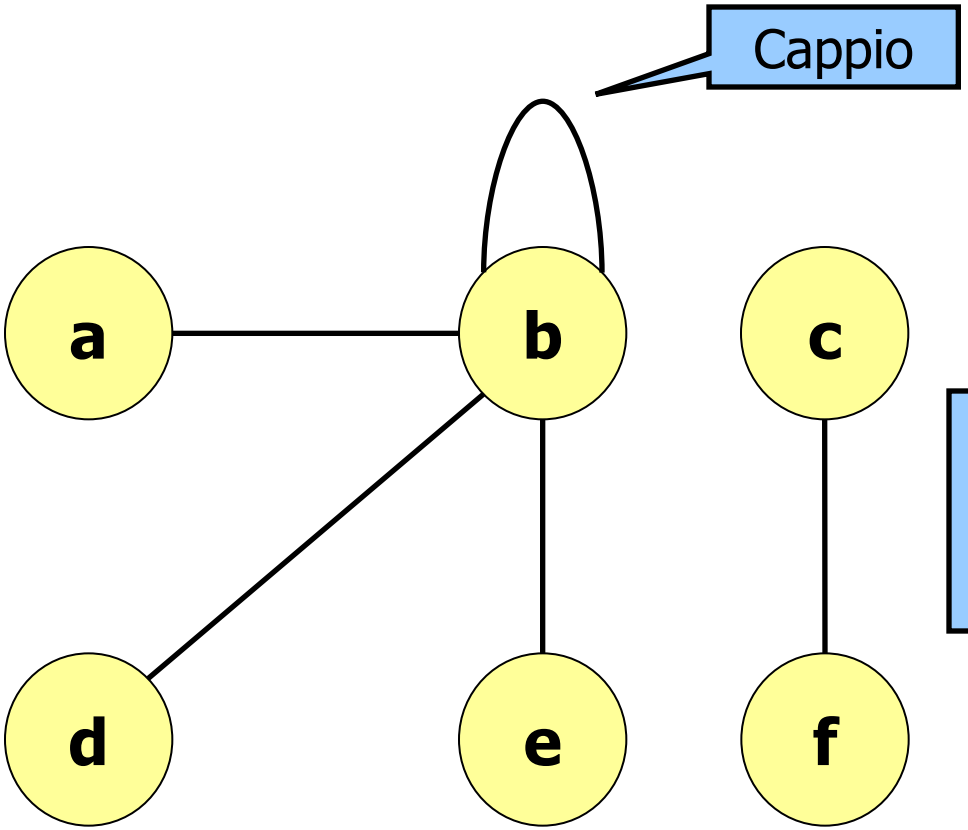
Esempio: grafo orientato



G={V, E}
V={a, b, c, d, e, f}
E={(a,b),(b,b),(b,d),(b,e),
(d,a),(d,e),(e,d),(f,c)}

NB: in alcuni contesti i cappi possono essere vietati.
Se il contesto ammette i cappi, ma il grafo ne è privo, esso si dice SEMPLICE.

Esempio: grafo non orientato



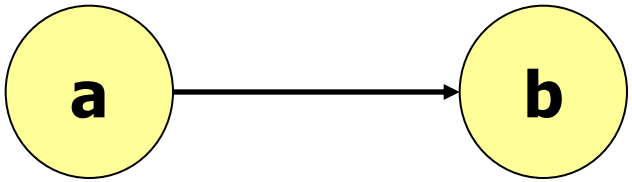
G={V, E}
V={a, b, c, d, e, f}
E={(a,b),(b,b)(b,d),(b,e),(c,f)}

NB: in alcuni contesti i cappi possono essere vietati.
Se il contesto ammette i cappi, ma il grafo ne è privo, esso si dice SEMPLICE.

Incidenza e adiacenza

Arco (a, b):

- incidente da vertice a
- incidente in vertice b
- incidente (insistente) sui vertici a e b



Vertici a e b adiacenti:

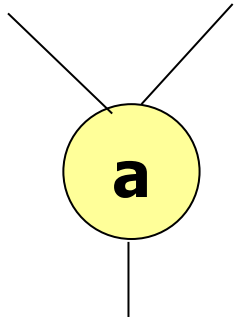
$$a \rightarrow b \Leftrightarrow (a, b) \in E$$

Grado di un vertice

grafo non orientato:

- degree(a) = numero di archi incidenti

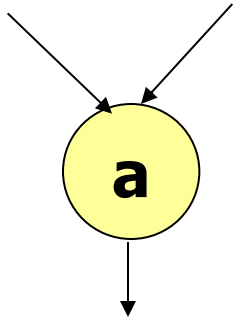
degree(a) = 3



grafo orientato:

- in_degree(a)=numero di archi entranti
- out_degree(a)=numero di archi uscenti
- degree(a)=in_degree(a) + out_degree(a).

in_degree(a) = 2
out_degree(a) = 1
degree(a) = 3



Cammini e raggiungibilità

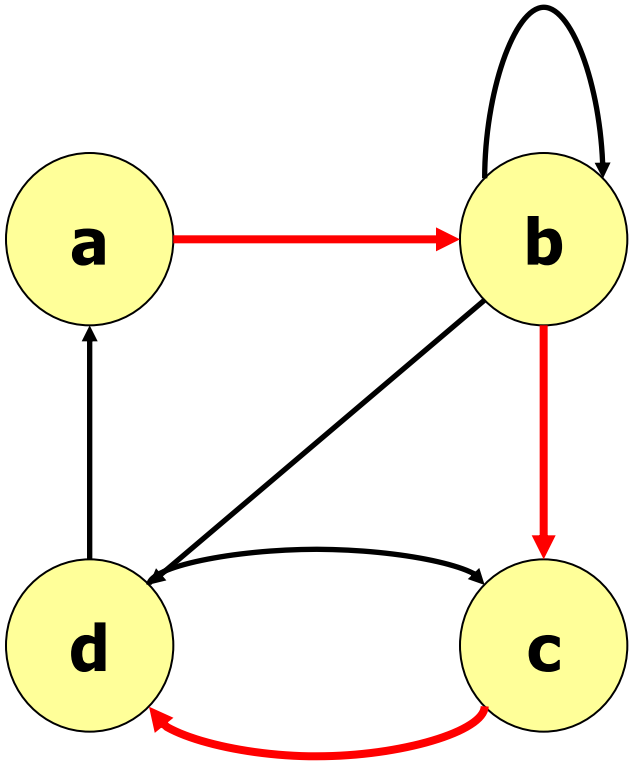
Cammino p: $u \rightarrow_p u'$ in G=(V,E):

$\exists (v_0, v_1, v_2, \ldots, v_k)$ |

$u=v_0$, $u'=v_k$, $\forall i = 1,2,\ldots,k$ $(v_{i-1}, v_i) \in E$.

- k = lunghezza del cammino.
- u' è raggiungibile da u $\Leftrightarrow$ $\exists$ p: $u \rightarrow_p u'$
- cammino p semplice: $(v_0, v_1, v_2, \ldots, v_k) \in p$ distinti.

Esempio



G = (V, E)

p: $a \rightarrow_p d$: (a, b), (b, c), (c, d)

k = 3

d è raggiungibile da a

p semplice.

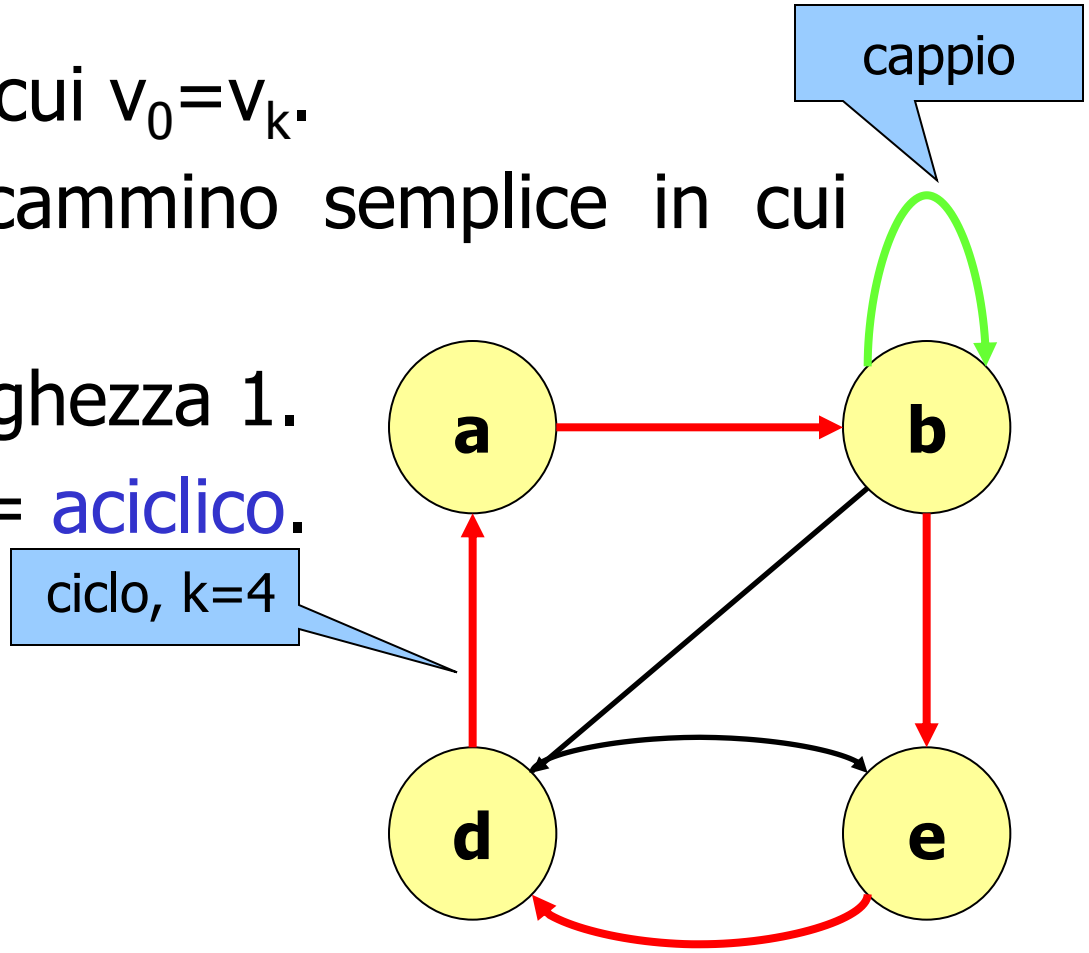
Cicli

Ciclo = cammino in cui $v_0=v_k$.

Ciclo semplice = cammino semplice in cui $v_0=v_k$.

Cappio = ciclo di lunghezza 1.

Un grafo senza cicli = aciclico.



Connessione nei grafi non orientati

Grafo non orientato connesso:

$$\forall v_i, v_j \in V \qquad \exists p \quad v_i \rightarrow_p v_j$$

Componente connessa: sottografo connesso massimale (= $\nexists$ sottoinsiemi per cui vale la proprietà che lo includono).

Grafo non orientato connesso: una sola componente connessa.

Connessione nei grafi orientati

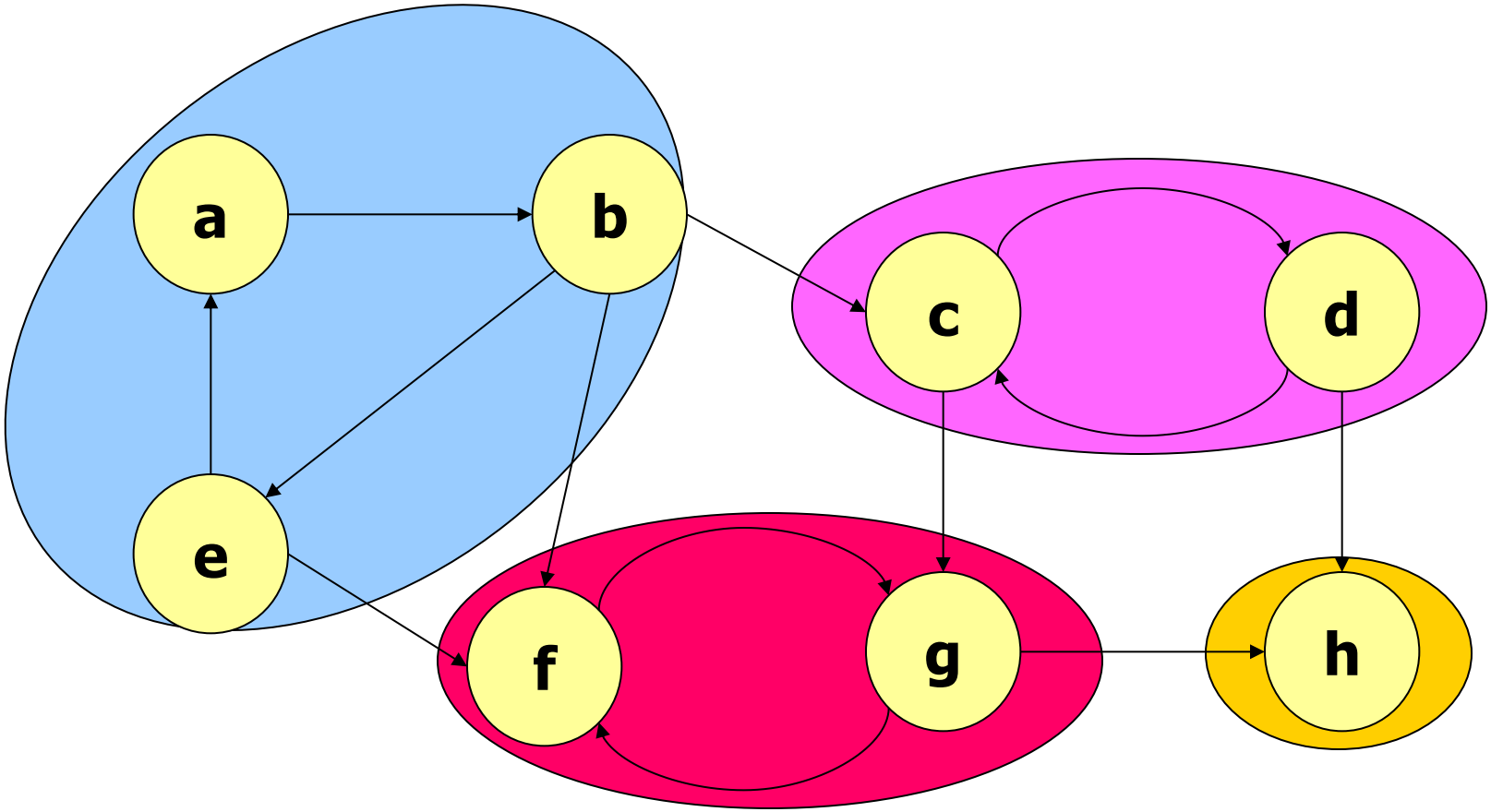
Grafo orientato fortemente connesso:

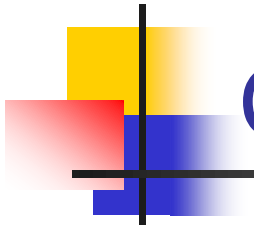
$$\forall v_i, v_j \in V \quad \exists p, p' \; v_i \rightarrow_p v_j \; \text{ e } \; v_j \rightarrow_{p'} v_i$$

Componente fortemente connessa: sottografo fortemente connesso massimale.

Grafo orientato fortemente connesso: una sola componente fortemente connessa.

Esempio





Grafi densi/sparsi

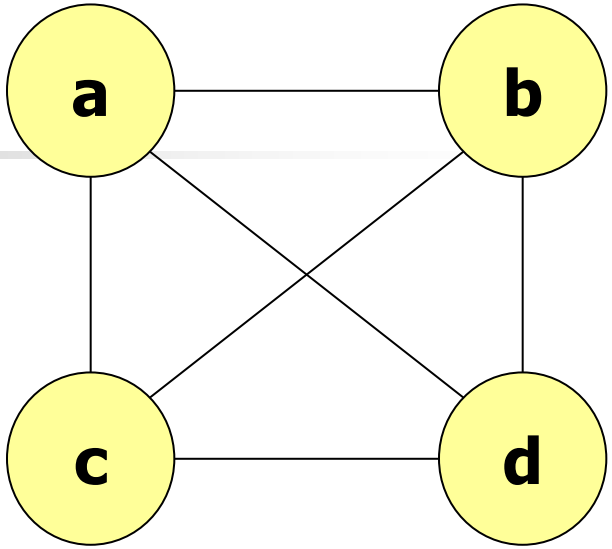
Dato grafo G = (V, E)

|V| = cardinalità dell'insieme V

|E| = cardinalità dell'insieme E

- grafo denso: $|E| \cong |V|^2$
- grafo sparso: $|E| << |V|^2$

Grafo completo



Definizione:

$\forall v_i, v_j \in V \quad \exists (v_i, v_j) \in E$

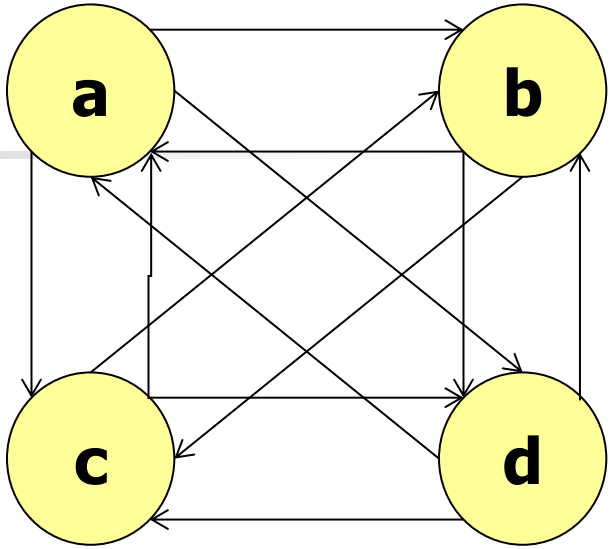
Quanti archi?

grafo completo non orientato:

|E| = numero di **combinazioni** di |V| elementi a 2 a 2

L'ordine non conta

$$|E| = \frac{|V|!}{(|V|-2)!*2!} = \frac{|V|*(|V|-1)*(|V|-2)!}{(|V|-2)!*2!} = \frac{|V|*(|V|-1)}{2}$$



Quanti archi?

grafo completo orientato:

|E| = numero di **disposizioni** di |V| elementi a 2 a 2

L'ordine conta

$$|E| = \frac{|V|!}{(|V|-2)!} = \frac{|V|*(|V|-1)*(|V|-2)!}{(|V|-2)!} = |V| * (|V|-1)$$

Grafo bipartito

Definizione:

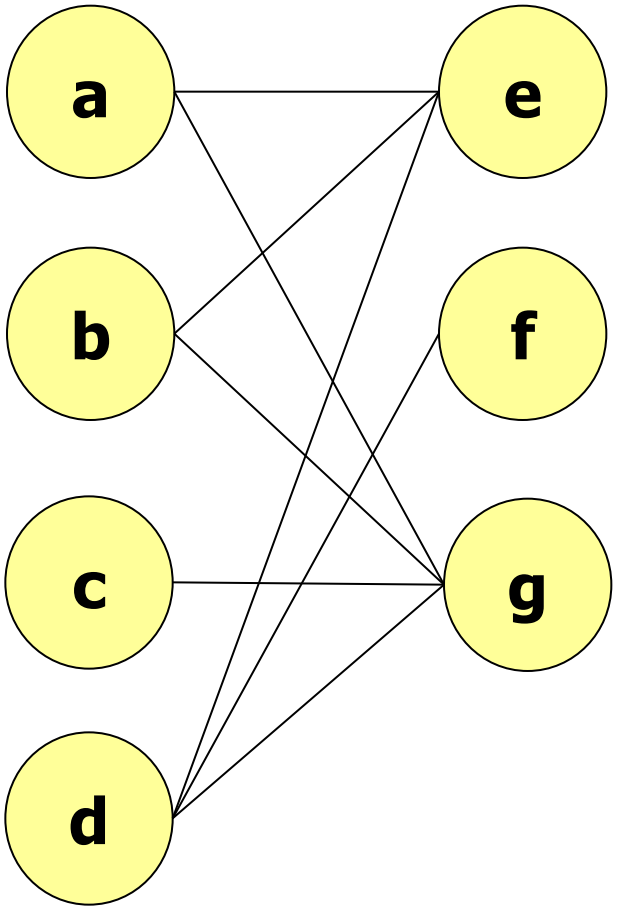
Grafo non orientato in cui l'insieme V può essere partizionato in 2 sottoinsiemi V_1 e V_2, tali per cui

$\forall (v_i, v_j) \in E$

$v_i \in V_1 \;\&\&\; v_j \in V_2$

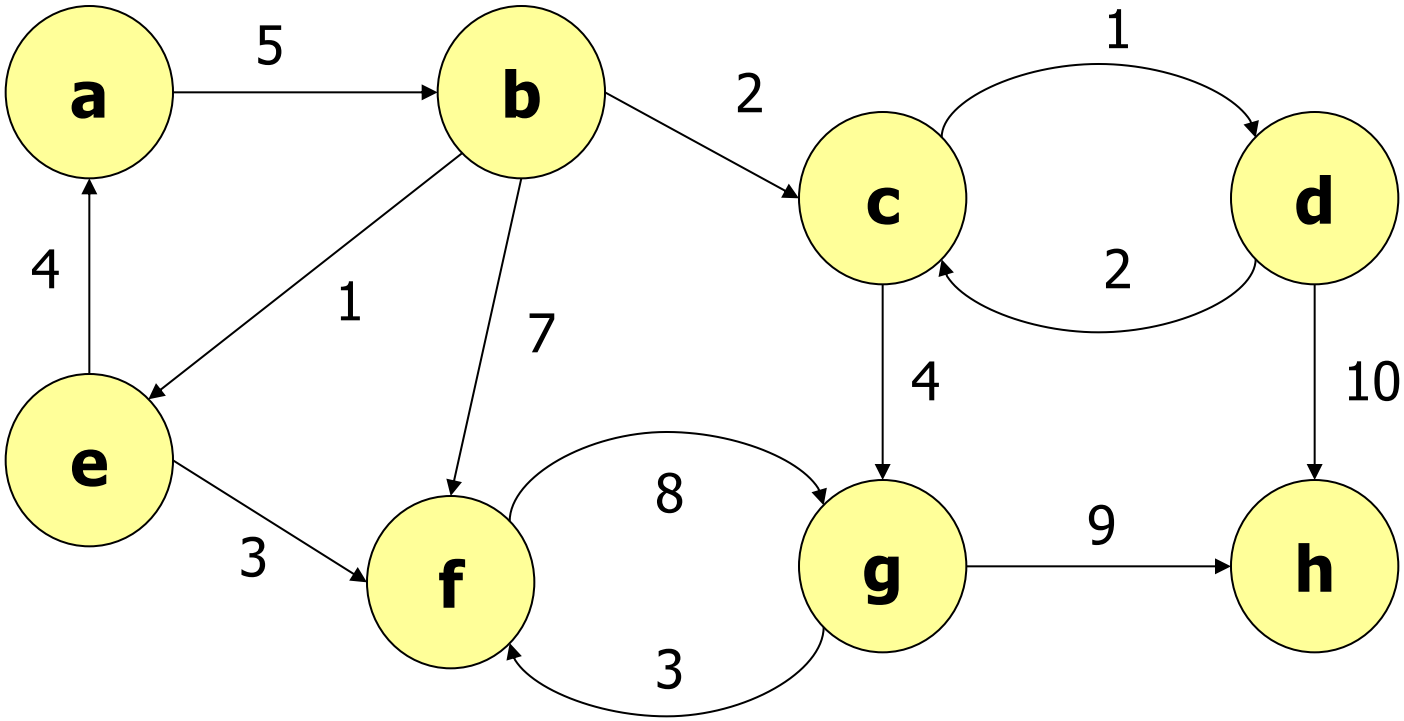
$\|$

$v_j \in V_1 \;\&\&\; v_i \in V_2$

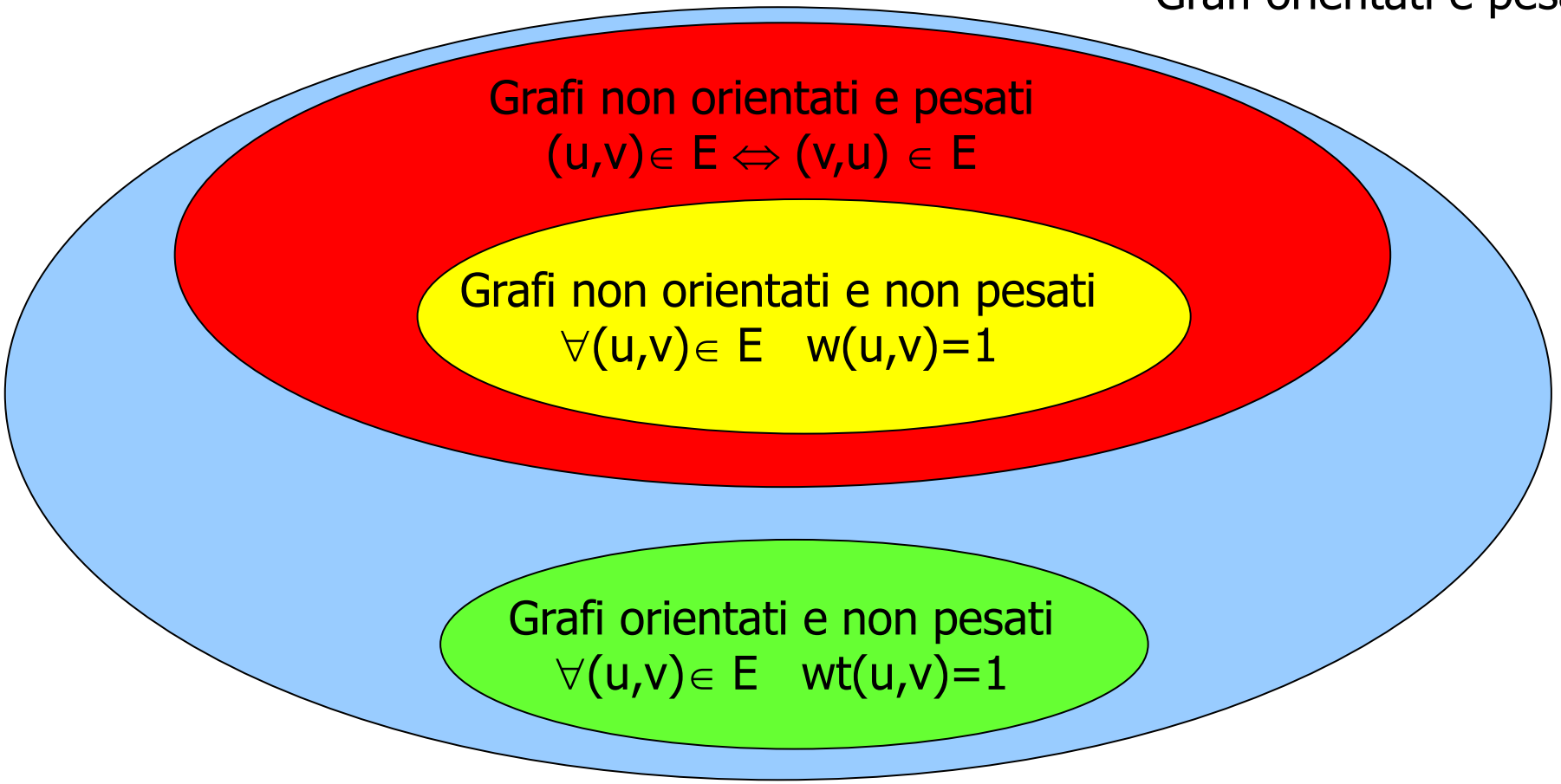


Grafo pesato

$\exists$ wt : E $\rightarrow$ R $\cup$ {-∞, + ∞} | wt(u,v) = peso dell'arco (u, v)



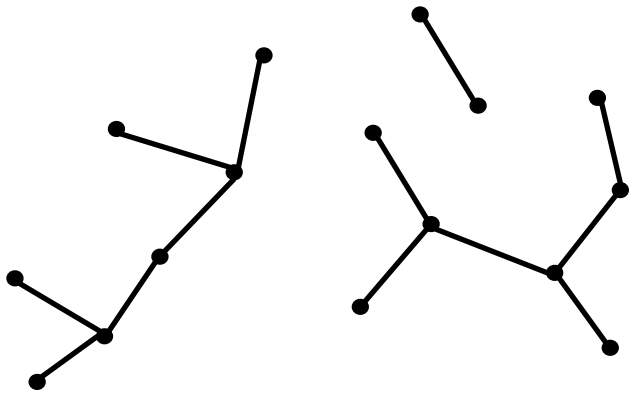
Riassunto Tipologie di Grafo



Alberi non radicati (o liberi)

Albero non radicato (o libero) = grafo non orientato, connesso, aciclico

Foresta = grafo non orientato, aciclico



Proprietà

G = (V, E) grafo non orientato $|E|$ archi, $|V|$ nodi:

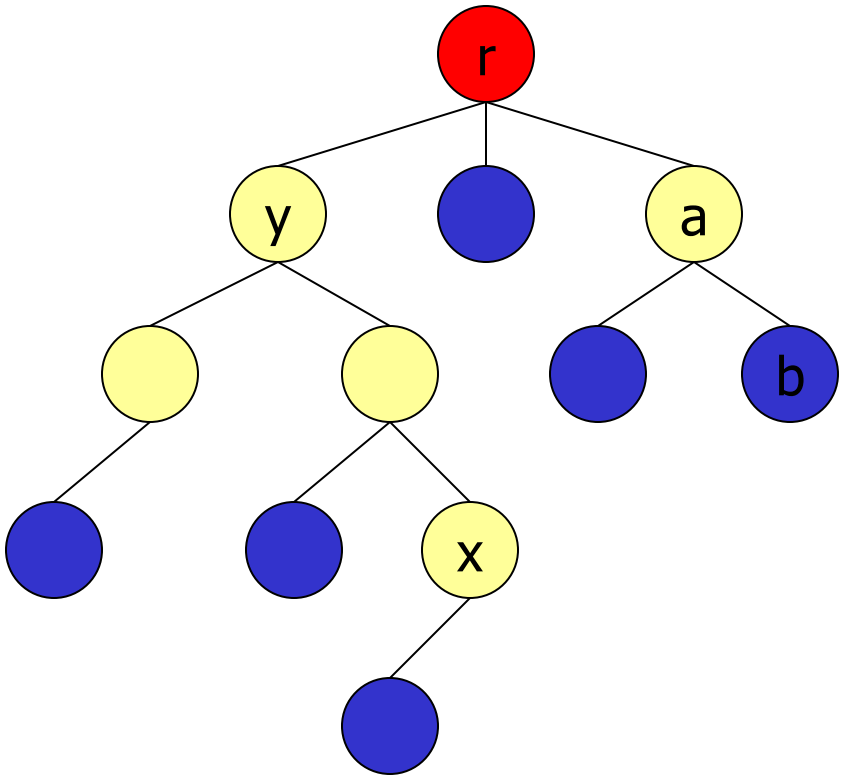
- G = albero non radicato
- ogni coppia di nodi connessa da un unico cammino semplice
- G connesso, la rimozione di un arco lo sconnette
- G connesso e $|E| = |V| - 1$
- G aciclico e $|E| = |V| - 1$
- G aciclico, l'aggiunta di un arco introduce un ciclo.

Alberi radicati

$\exists$ nodo r detto radice

- relazione di parentela
  - y antenato di x se y appartiene al cammino da r a x. x discendente di y
  - antenato proprio se $x \neq y$
  - padre/figlio: nodi adiacenti
- radice: no padre 
- foglie no figli 

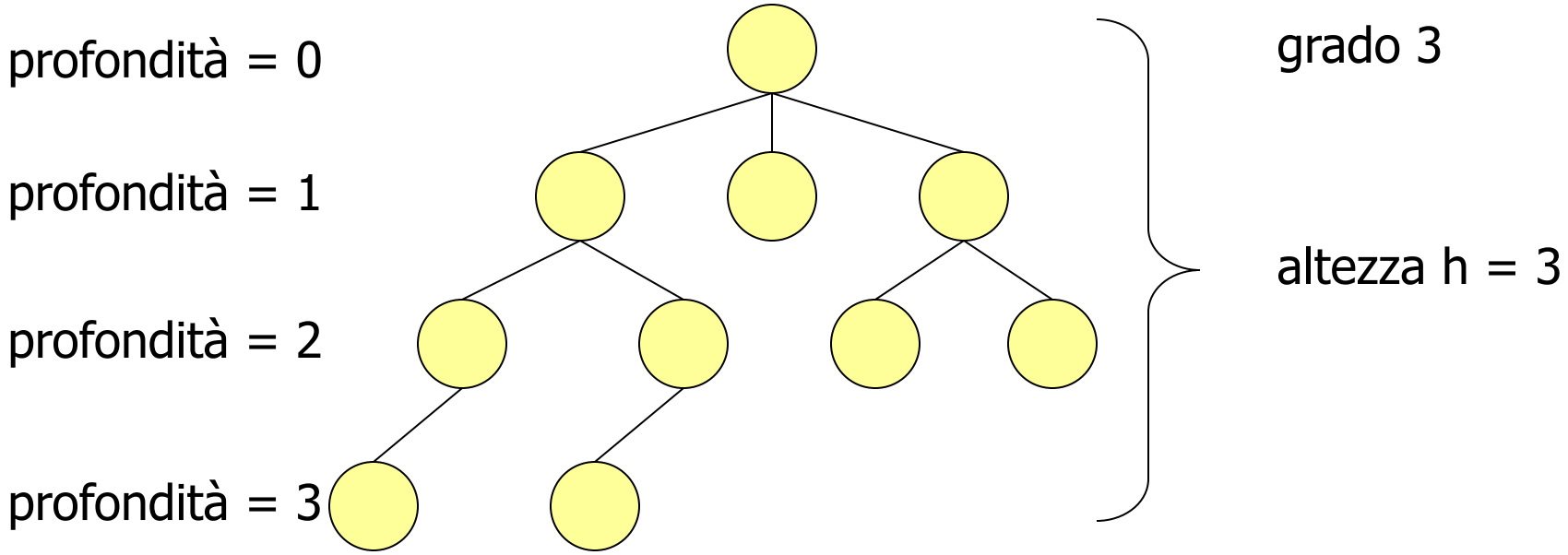
Esempio



y antenato di x
x discendente di y
a padre di b
b figlio di a

Proprietà di un albero T

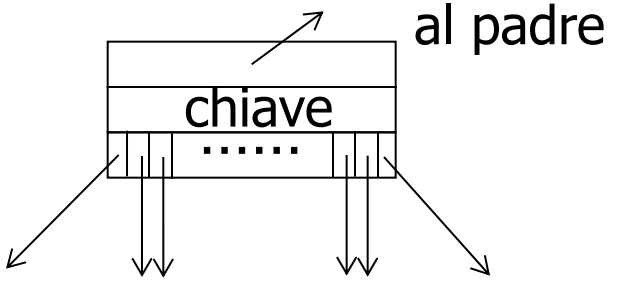
- grado(T) = numero max di figli
- profondità(x) = lunghezza del cammino da r a x
- altezza(T) = profondità massima.



Rappresentazione di alberi

Rappresentazione di un nodo di un albero di grado(T) = k

- puntatore al padre, chiave, k puntatori ai k figli



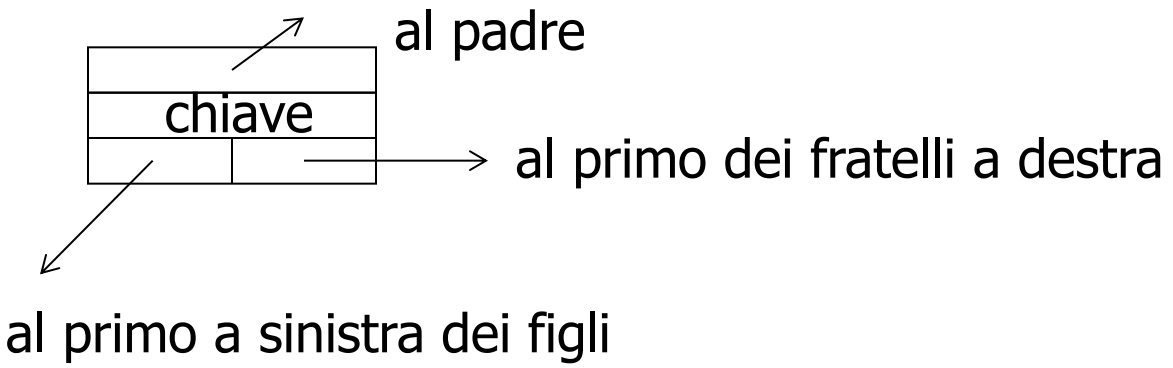
k puntatori ai k figli, eventualmente a NULL

Inefficiente se solo pochi nodi hanno davvero grado k (spazio per tutti i k puntatori allocato, ma molti a NULL).

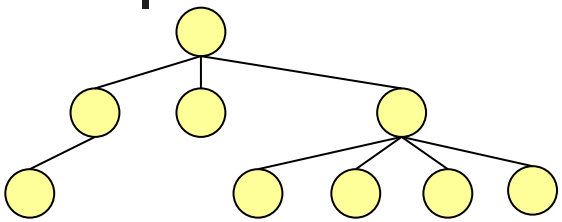
Rappresentazione left-child right-sibling

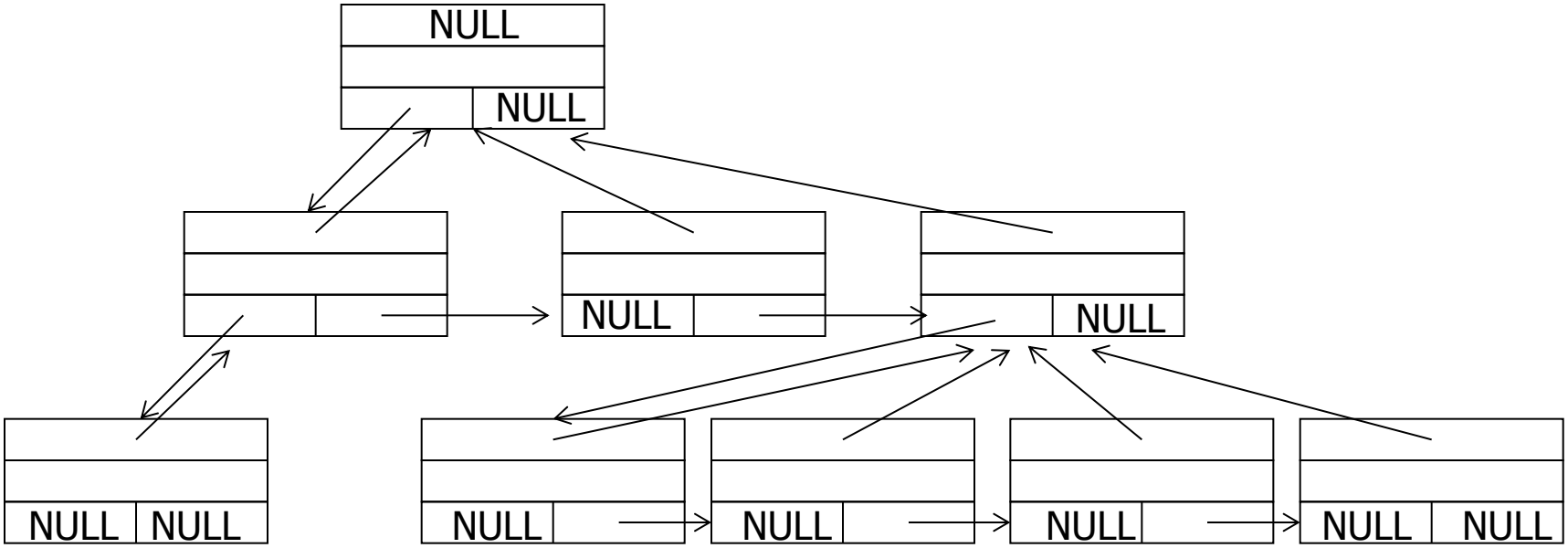
Rappresentazione di un nodo di un albero di grado(T) = k

- puntatore al padre, chiave, 1 puntatore al figlio sinistro, 1 puntatore al fratello a destra



Efficiente: sempre solo 2 puntatori, indipendentemente dal grado dell'albero

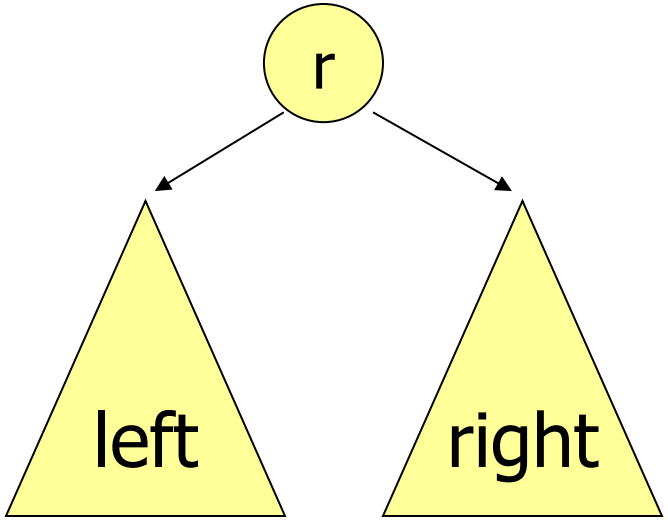



NULL
NULL
NULL
NULL
NULL
NULL
NULL
NULL
NULL
NULL

Alberi binari

Definizione:

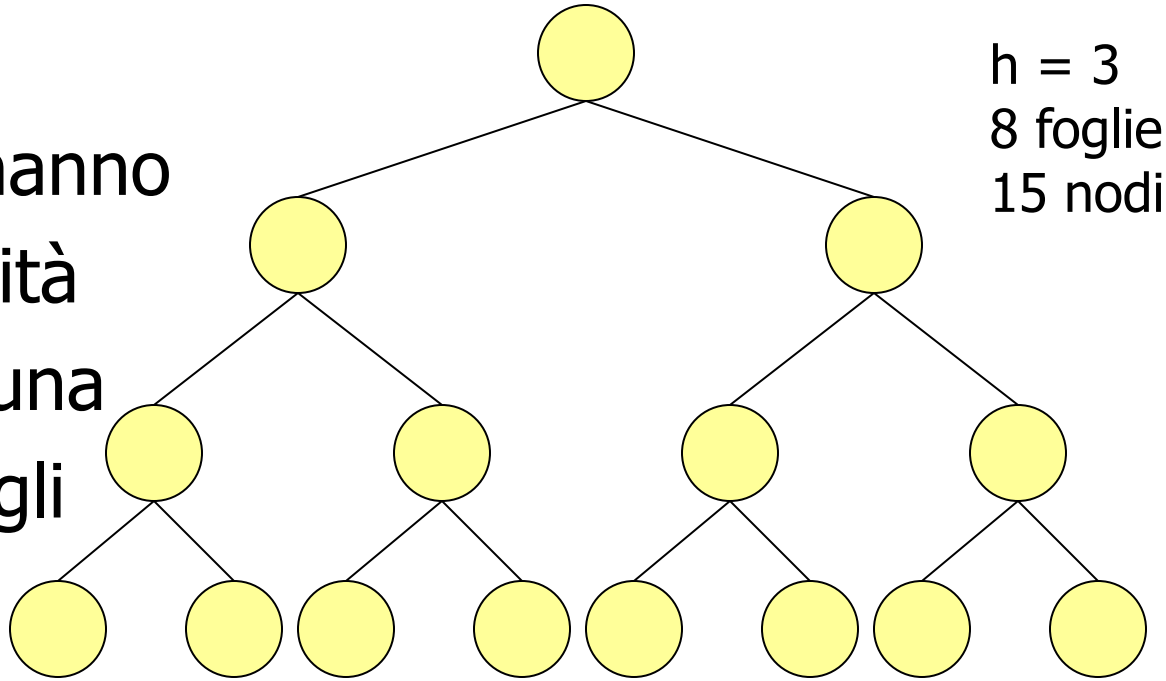
- Albero di grado 2
- Ricorsivamente T è:
  - insieme di nodi vuoto
  - radice, sottoalbero sinistro, sottoalbero destro.



Albero binario completo

Due condizioni:

- tutte le foglie hanno stessa profondità
- ogni nodo o è una foglia o ha 2 figli



h = 3
8 foglie
15 nodi

Albero binario completo di altezza h:

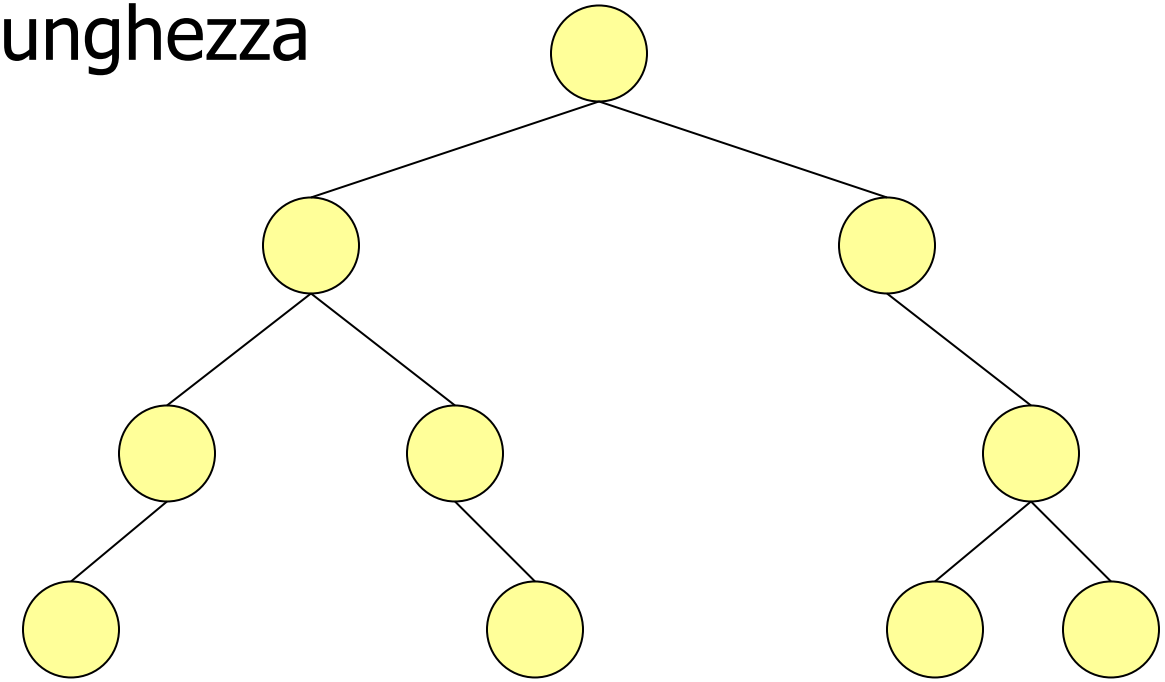
- numero di foglie 2^h
- numero di nodi = $\Sigma_{0 \le i \le h} 2^i = 2^{h+1} - 1$

progressione geometrica finita di ragione 2


Hyphaene Compressa - Doum Palm
© Shlomit Pinter

Albero binario bilanciato

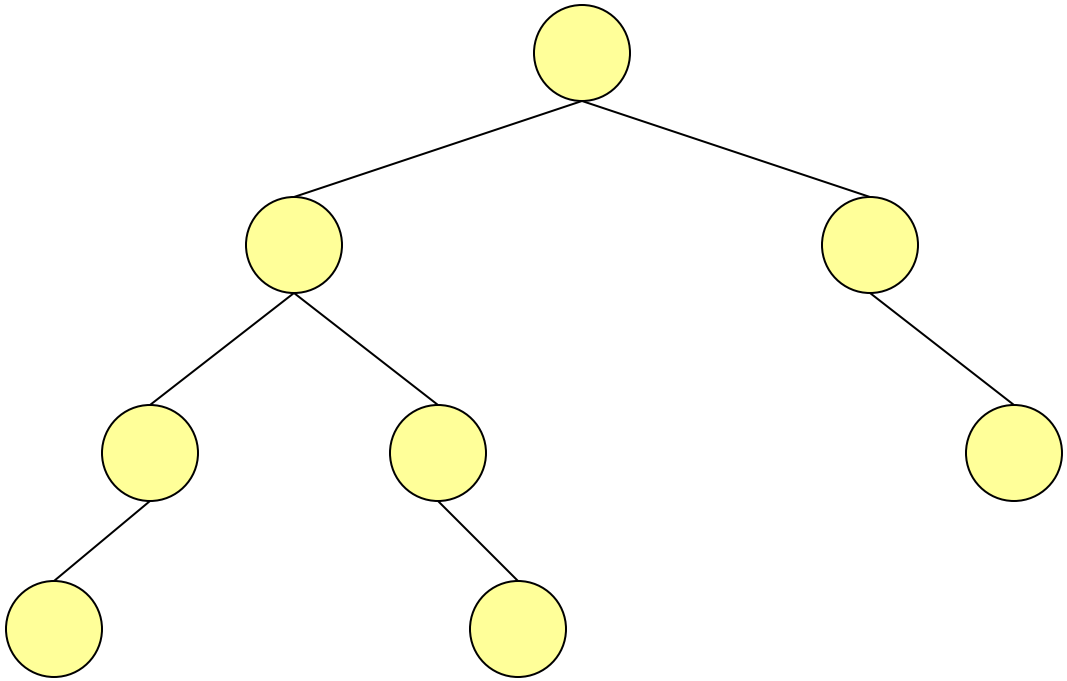
- Tutti i cammini radice-foglie sono di ugual lunghezza



- Se T è completo, allora T è bilanciato, non vale necessariamente il viceversa.

Albero binario quasi bilanciato

- La lunghezza di tutti i cammini radice-foglie differisce al massimo di 1.



Sequenze lineari

- Insieme finito di elementi disposti consecutivamente in cui a ogni elemento è associato univocamente un indice

$$a_0, a_1, \ldots, a_i, \ldots, a_{n-1}$$

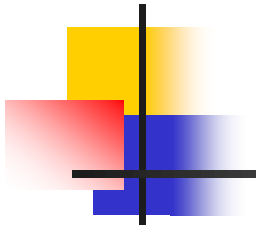
- Sulle coppie di elementi è definita una relazione predecessore/successore:

$$a_{i+1} = succ(a_i) \qquad a_i = pred(a_{i+1})$$

Memorizzazione e accesso

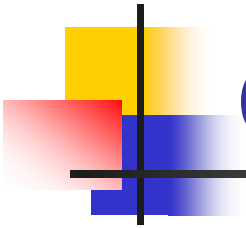
Vettori o array:

- Modalità di memorizzazione: dati contigui in memoria
- Accesso diretto:
  - dato l'indice i, si accede all'elemento a_i senza dover scorrere la sequenza lineare
  - il costo dell'accesso non dipende dalla posizione dell'elemento nella sequenza lineare, quindi è O(1)



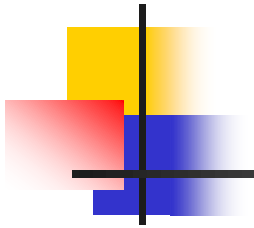
Liste:

- Modalità di memorizzazione: dati non contigui in memoria
- Accesso sequenziale:
  - dato l'indice i, si accede all'elemento a_i scorrendo la sequenza lineare a partire da uno dei suoi 2 estremi, solitamente quello di SX
  - il costo dell'accesso dipende dalla posizione dell'elemento nella sequenza lineare, quindi è O(n) nel caso peggiore

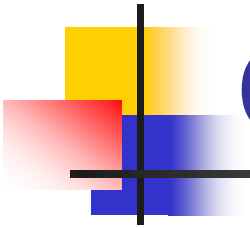


Operazioni sulle liste

- **ricerca** di un elemento il cui campo chiave di ricerca è uguale a una chiave data
- **inserzione** di un elemento:
  - in testa alla lista non ordinata
  - in coda alla lista non ordinata
  - nella posizione tale da garantire l'invarianza della proprietà di ordinamento per una lista ordinata



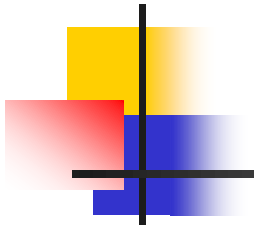
- **estrazione** di un elemento:
  - che si trova in testa alla lista non ordinata
  - che ha un campo con contenuto uguale a quello di una chiave di cancellazione (tale operazione richiede solitamente una ricerca preventiva dell'elemento da cancellare).



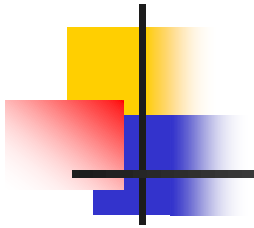
Collezioni di dati

Code generalizzate: collezioni di oggetti (dati) con operazioni principali:

- Insert: inserisci un nuovo oggetto nella collezione
- Search: ricerca se un oggetto è nella collezione
- Delete: cancella un oggetto della collezione



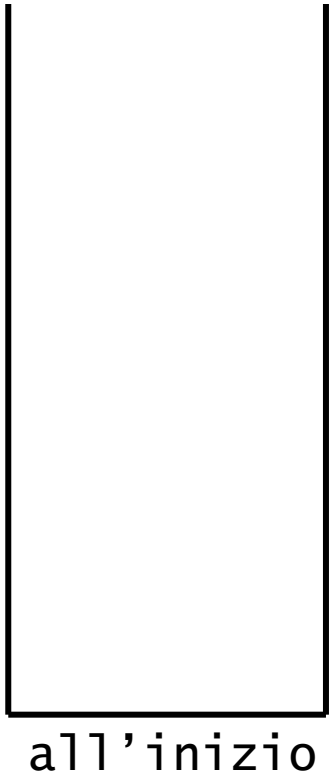
- Altre operazioni:
  - Inizializzare la coda generalizzata
  - Conteggio oggetti (o verifica collezione vuota)
  - Distruzione della coda generalizzata
  - Copia della coda generalizzata

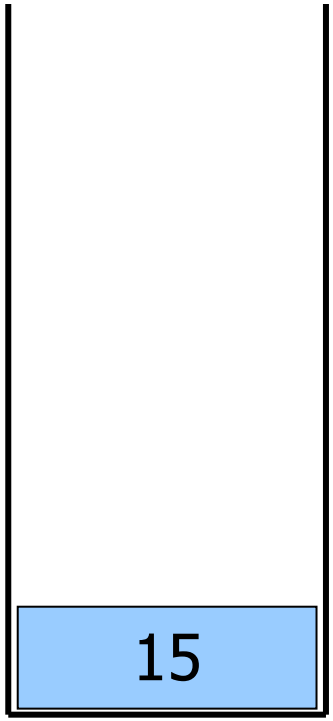


Criteri per operazione di Delete:

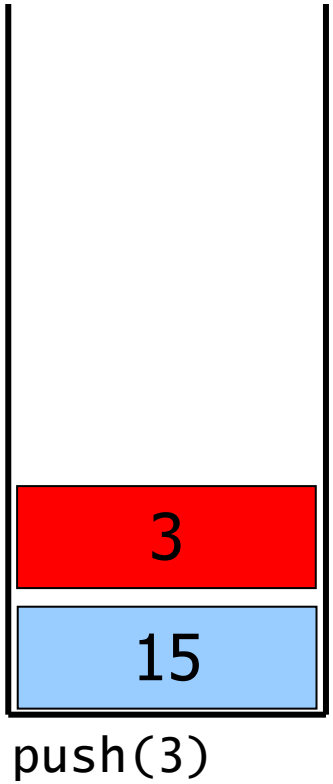
- cronologico:
  - estrazione dell'elemento inserito più recentemente
    - politica LIFO: Last-In First-Out
    - **stack** o **pila**
    - inserzione (push) ed estrazione (pop) dalla testa

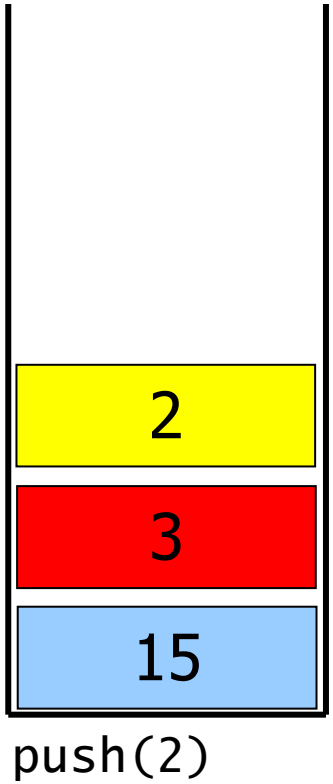
Esempio

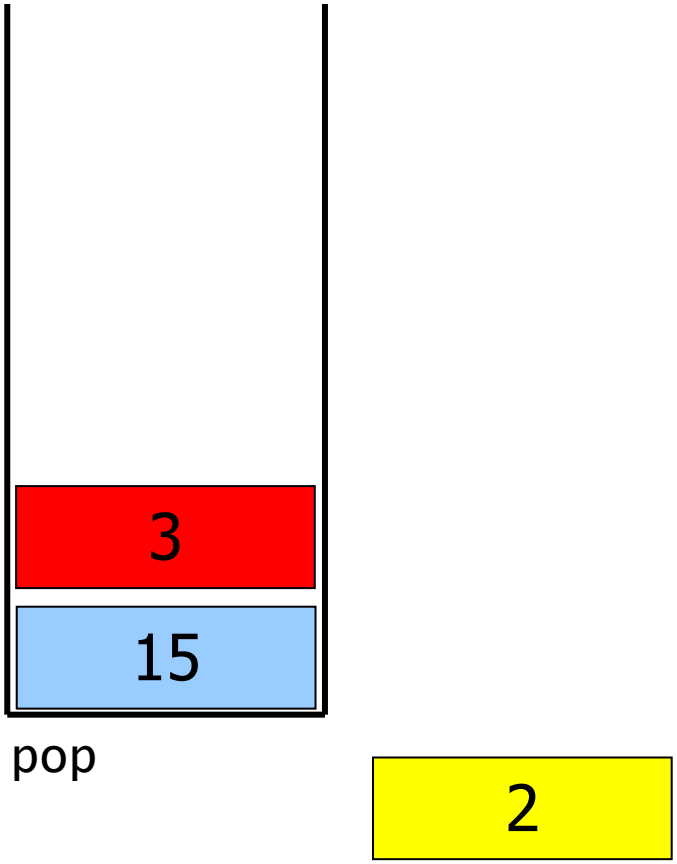


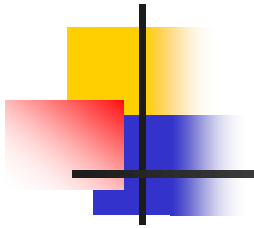


push(15)


3
15
push(3)

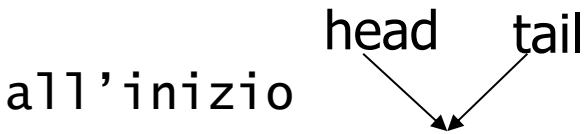

2
3
15
push(2)

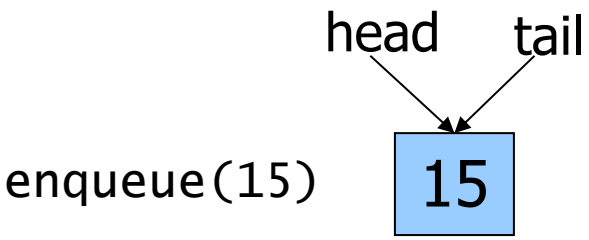

3
15
pop
2

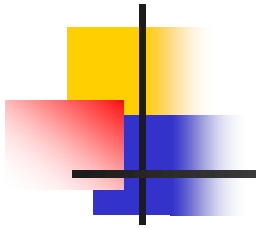


- estrazione dell'elemento inserito meno recentemente
  - politica FIFO: First-In First-Out
  - **queue** o **coda**
  - inserzione (enqueue) in coda (tail) ed estrazione (dequeue) dalla testa (head)

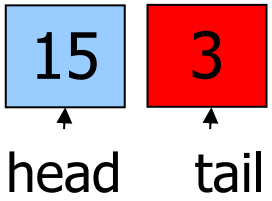
Esempio

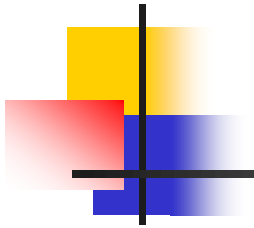



head
tail
enqueue(15)
15

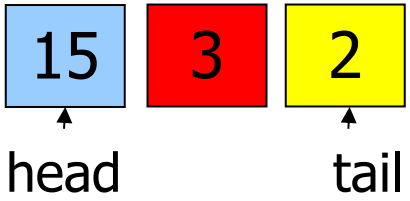


`enqueue(3)`

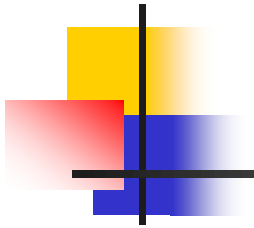




`enqueue(2)`

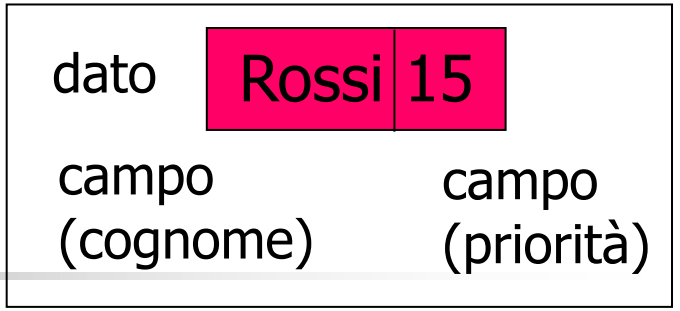



dequeue
15
3
2
head
tail

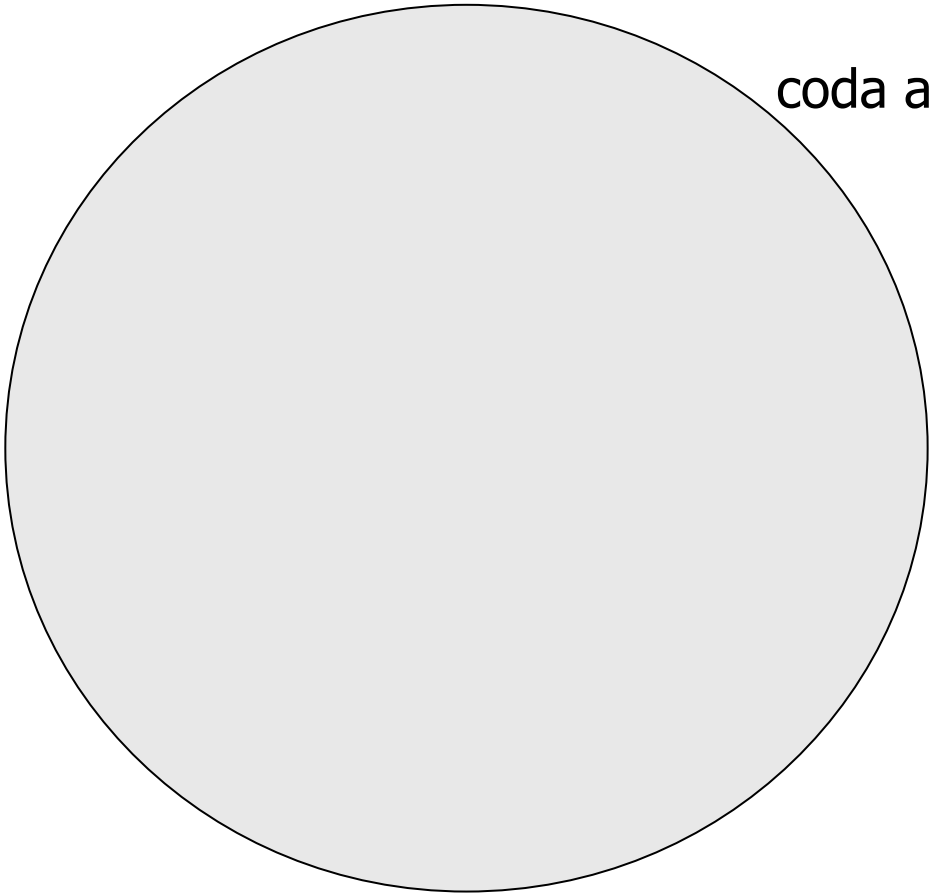


- priorità:
  - l'inserzione garantisce che, estraendo dalla testa, si ottenga il dato a priorità massima (o minima)
  - **coda a priorità**

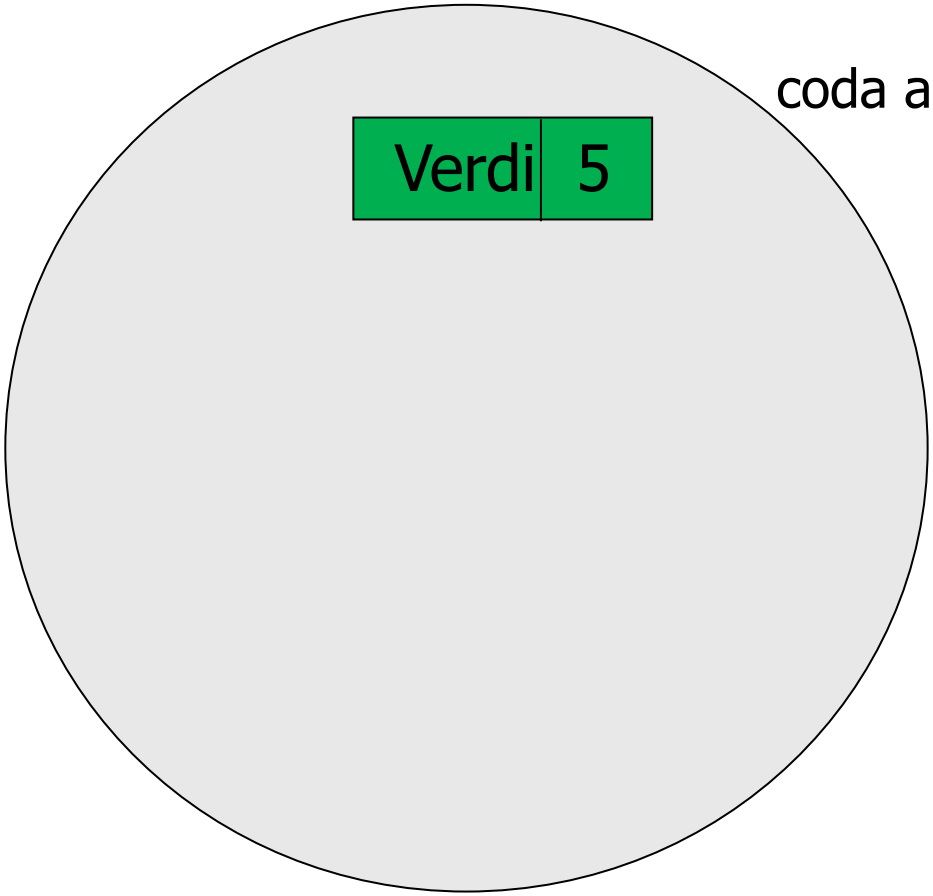
Esempio



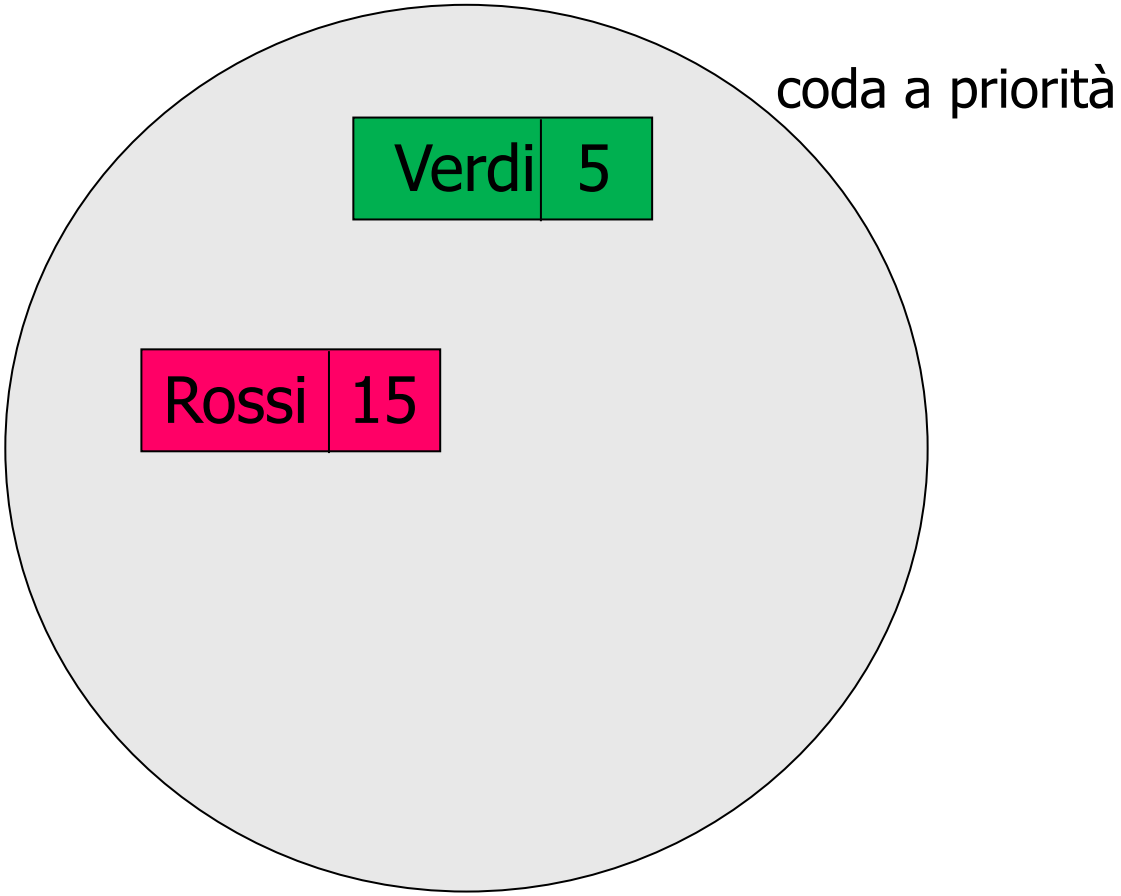
all'inizio



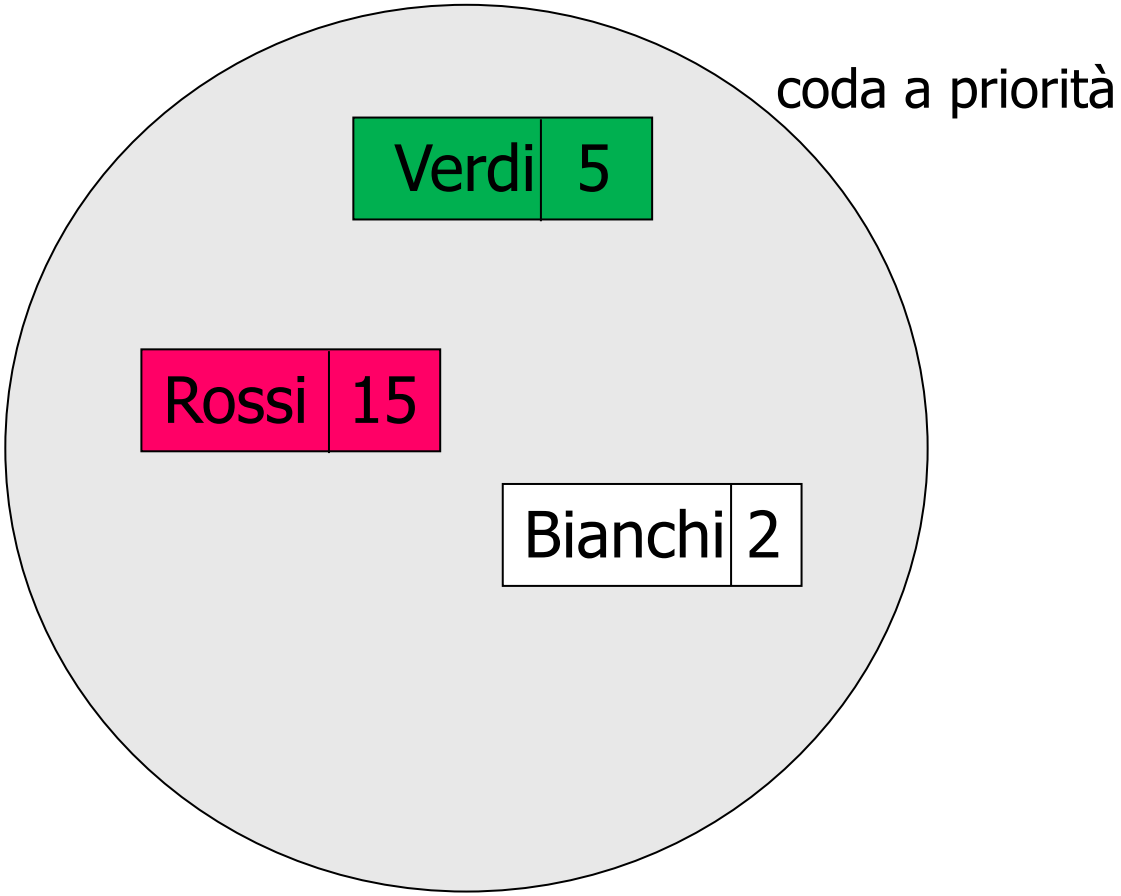
```
insert(Verdi, 5)
```

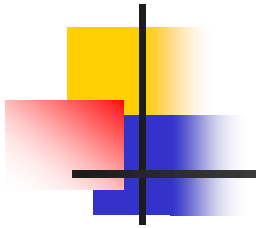


```
insert(Rossi, 15)
```

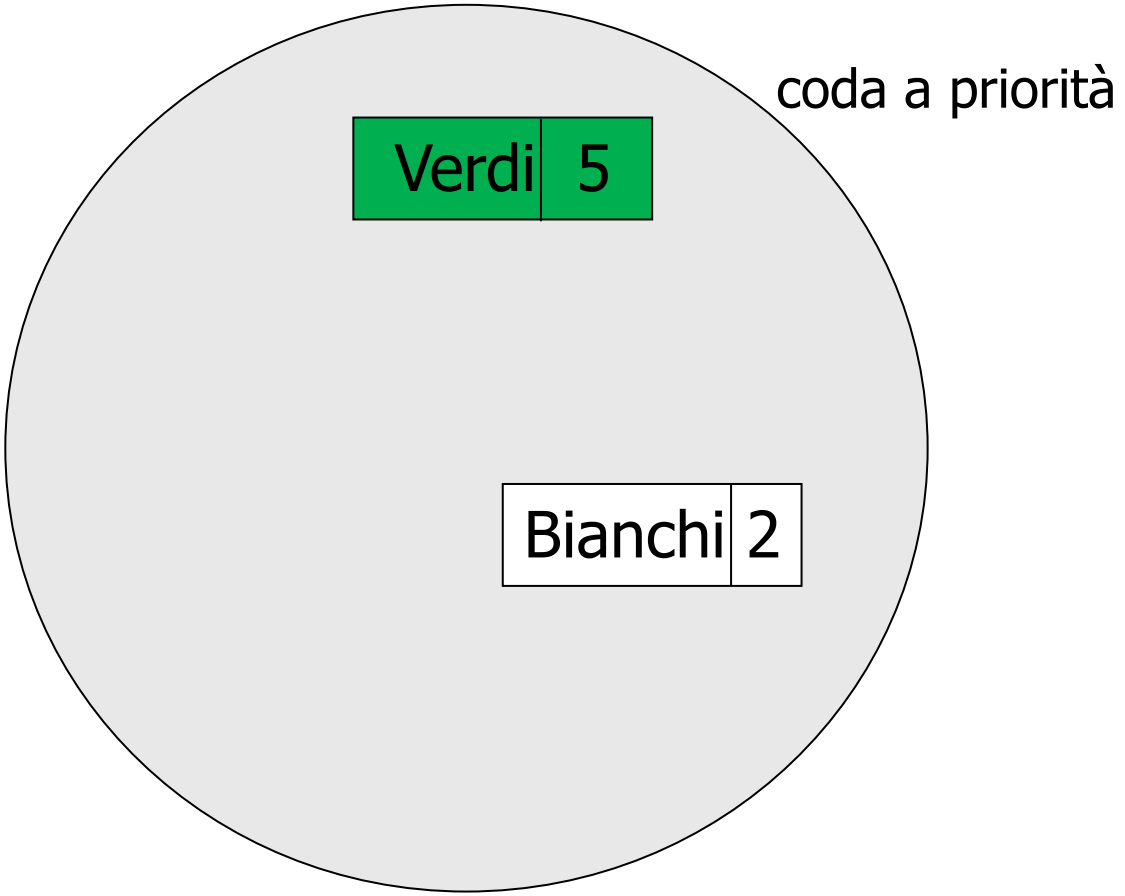


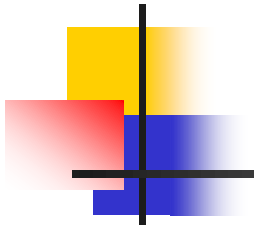
```
insert(Bianchi, 2)
```



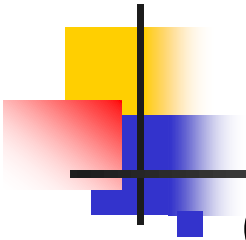


`extract`





- contenuto:
  - l'estrazione ritorna un contenuto secondo determinati criteri
  - **tabella di simboli**



Riferimenti

- Grafi:
  - Cormen 5.4
  - Sedgewick Part 5 17.1
- Alberi:
  - Cormen 5.5
  - Sedgewick 5.4
- Liste, pile, code, code a priorità:
  - Cormen 10.1, 10.2, 6.5
  - Sedgewick 3.3, 4.2, 4.6, 9
- Code generalizzate:
  - Sedgewick 4.1