Algoritmi e Strutture Dati

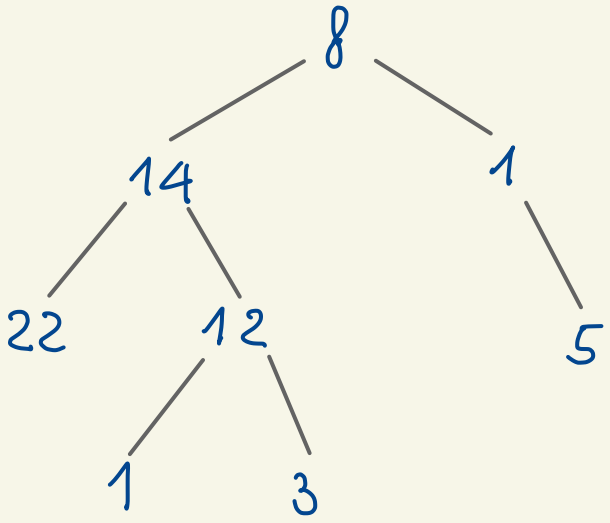
Lezione 14

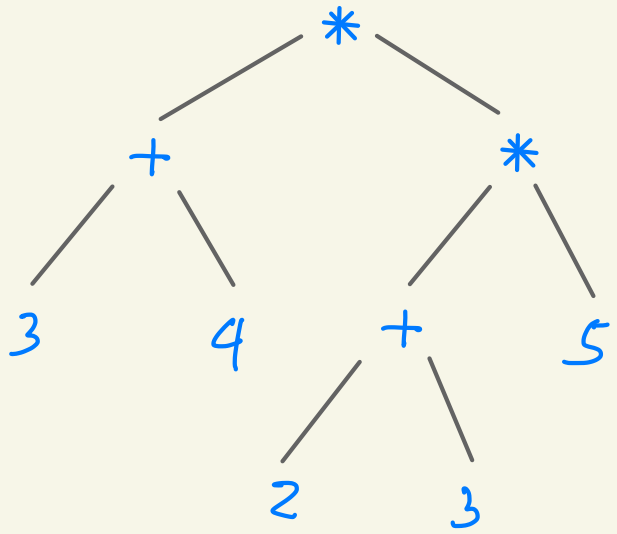
24 ottobre 2025

Dalle liste agli alberi

Alberi binari

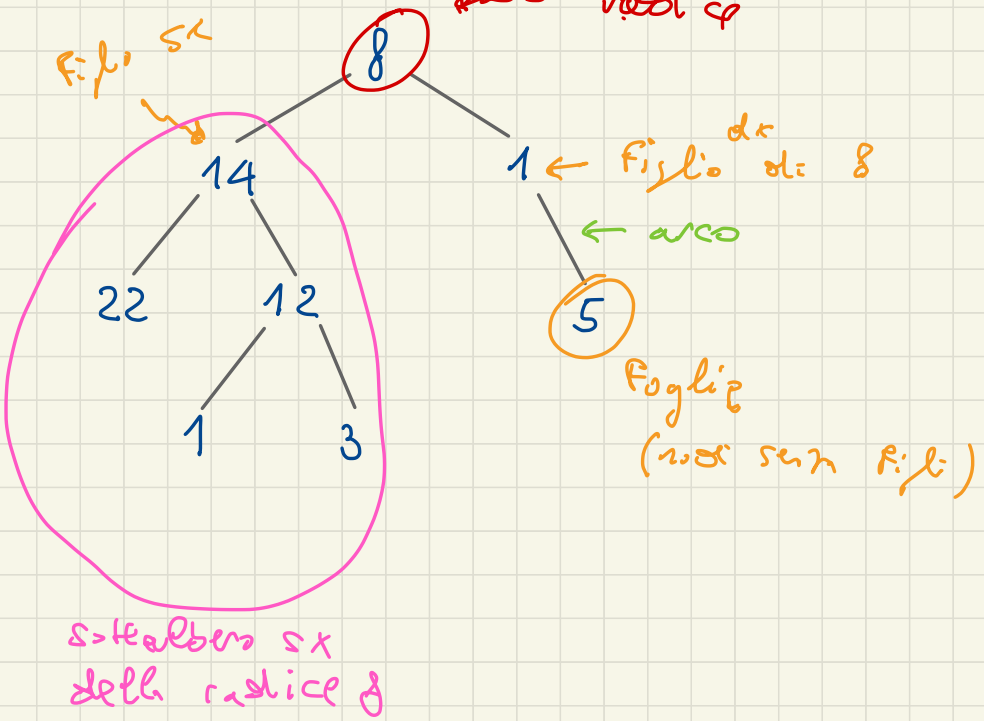
- Ad ogni nodo possono essere associati *due* "successori" detti *figlio sinistro* e *destro*





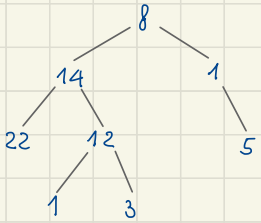
(3 + 4) * (2 + 3) * 5

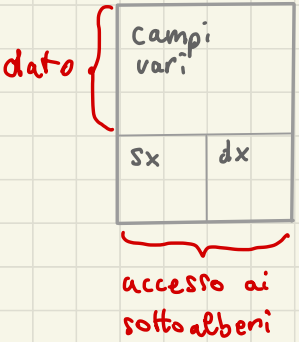
Alberi con radice

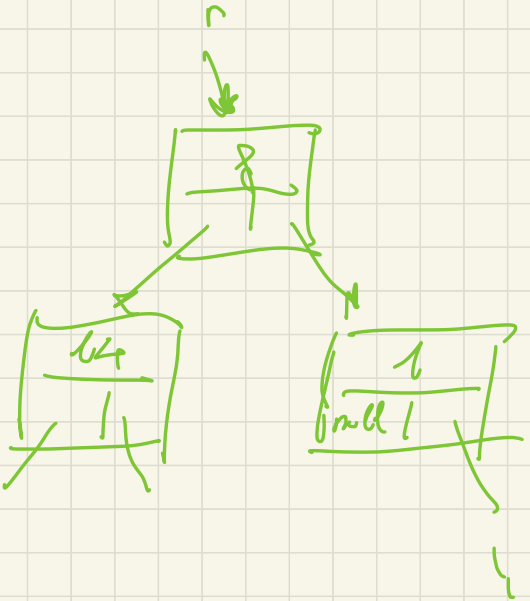


Profondità o livello
- radice prof. 0
- figli di un nodo a profondità k hanno prof. k+1

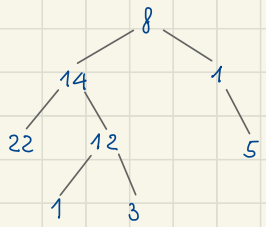
Altezza albero:
max prof di un nodo


8
14
1
22
12
5
1
3


dato
campi vari
sx
dx
accesso ai sottoalberi


r
8
14
1
22
12
5
1
3

VISITE AD ALBERI BINARI



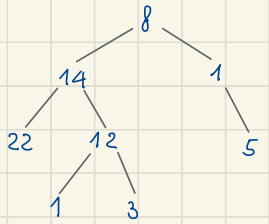
insieme S

ALGORITMO visitaGenerica (AlberoBinario radice)

    S ← {radice}
    WHILE S ≠ ∅
        preleva un nodo n da S
        visita n
        S ← S ∪ {figli del nodo n}

VISITA in AMPIEZZA

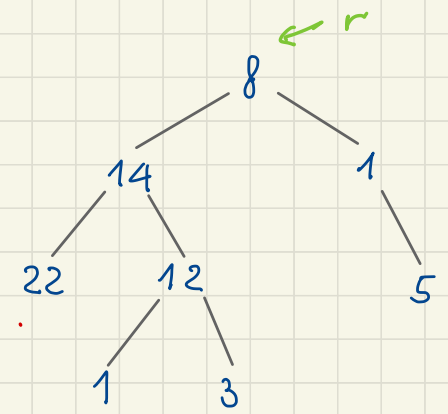
S → CODA




```
ALGORITMO visitaInAmpiezza (AlberoBinario r)
    C ← coda vuota
    C.enqueue(r)
    WHILE NOT (C.isEmpty()) DO
        n ← C.dequeue()
        IF n ≠ null THEN
            visita il nodo associato a n
            C.enqueue(n.sx)
            C.enqueue(n.dx)
```

```
ALGORITMO visitaGenerica (AlberoBinario radice)
    S ← {radice}
    WHILE S ≠ ∅
        preleva un nodo n da S
        visita n
        S ← S ∪ {figli del nodo n}
```

```
ALGORITMO visitaInAmpiezza (AlberoBinario r)
    C ← coda vuota
    C.enqueue(r)
    WHILE NOT (C.isEmpty()) DO
        n ← C.dequeue
        IF n ≠ null THEN
            visita il nodo associato a n
            C.enqueue(n.sx)
            C.enqueue(n.dx)
```



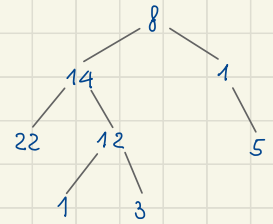
8 14 1 22 12 5 1 3

C ~~8~~ ~~14~~ 1 ~~22~~ 12 null 5 null null ...

costo visita $O(n)$

VISITA in PROFONDITA'

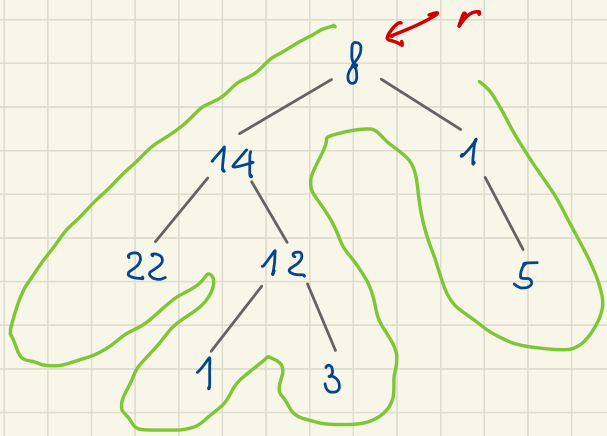
S ← pila



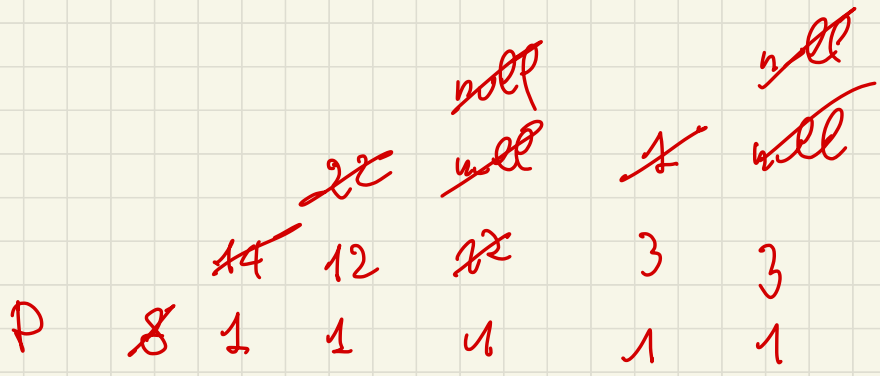

```
ALGORITMO visitaInProfondita(AlberoBinario r)
    P ← pila vuota
    P.push(r)
    WHILE NOT(P.isEmpty()) DO
        n ← P.pop()
        IF n ≠ null THEN
            visita il nodo associato a n
            P.push(n.dx)
            P.push(n.sx)
```

```
ALGORITMO visitaGenerica(AlberoBinario radice)
    S ← {radice}
    WHILE S ≠ ∅
        preleva un nodo n da S
        visita n
        S ← S ∪ {figli del nodo n}
```

```
ALGORITMO visitaProfondità(albero binario r)
    P ← pila vuota
    P.push(r)
    WHILE NOT(P.isEmpty()) DO
        n ← P.pop()
        IF n ≠ null THEN
            visita il nodo associato a n
            P.push(n.dx)
            P.push(n.sx)
```



8 14 22 12 1 3 1 5

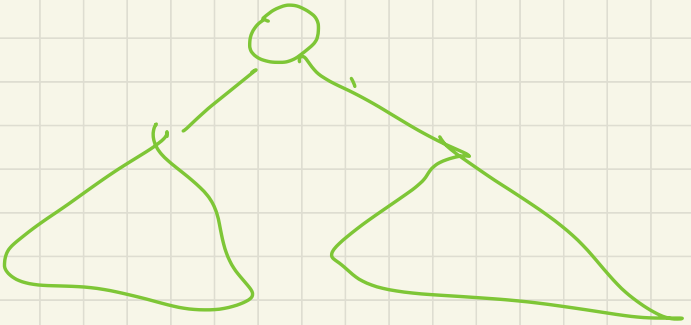


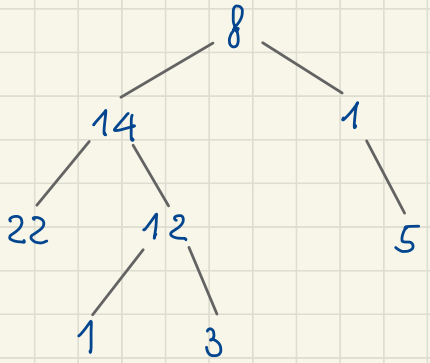
Albero binario { vuoto / oppure }

nodo (radice) + 2 sottoalberi

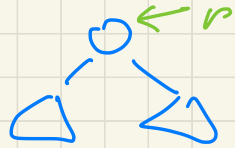
sottoalbero sx

" dx



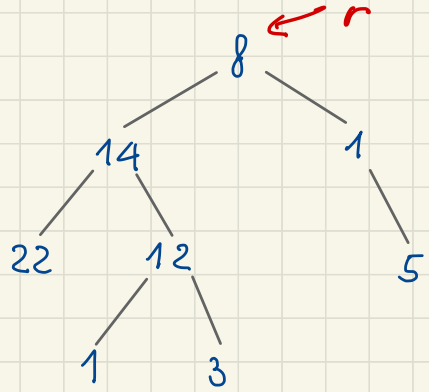


Albero binario: vuoto oppure nodo (radice) + 2 sottoalberi (sottoalbero sx, sottoalbero dx)




```
ALGORITMO visitaInProfondita(AlberoBinario r)
    IF r ≠ null THEN
        visita la radice
        visitaInProfondita(r.sx)
        visitaInProfondita(r.dx)
```

```
ALGORITMO visitaInProfondita (AlberoBinario r)
  IF r ≠ null THEN
    visita la radice
    visitaInProfondita (r.sx)
    visitaInProfondita (r.dx)
```



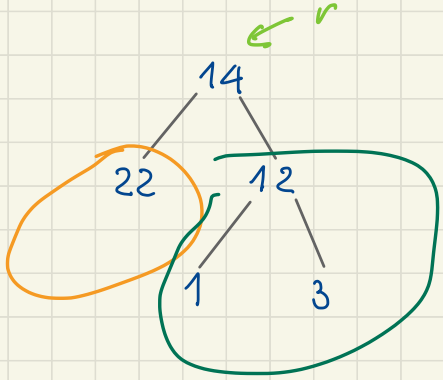
8 14 22 12 1 3 1 5

sottoalbero sx

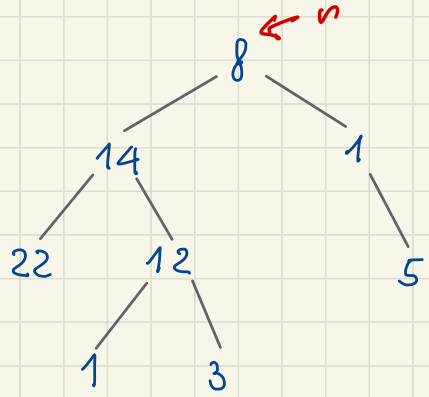
sottoalbero dx

VISITA IN ORDINE ANTICIPATO

(Pre-ordine)

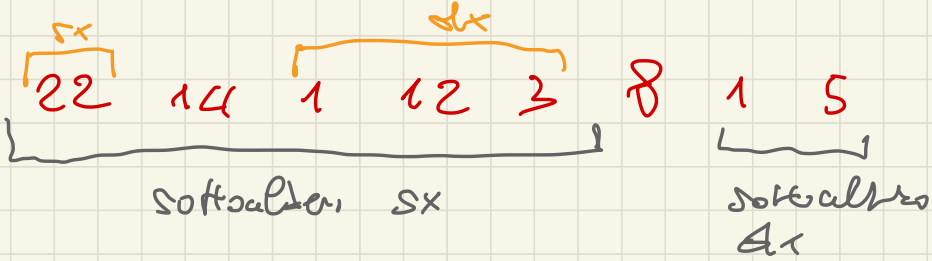


```
ALGORITMO visitaInProfondita (AlberoBinario r)
  IF r ≠ null THEN
    visitaInProfondita (r.sx)
    visita la radice
    visitaInProfondita (r.dx)
```

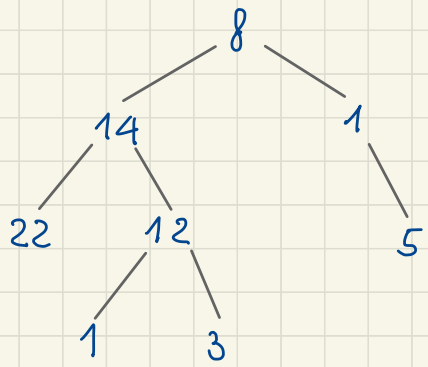


VISITA IN ORDINE SIMMETRICO

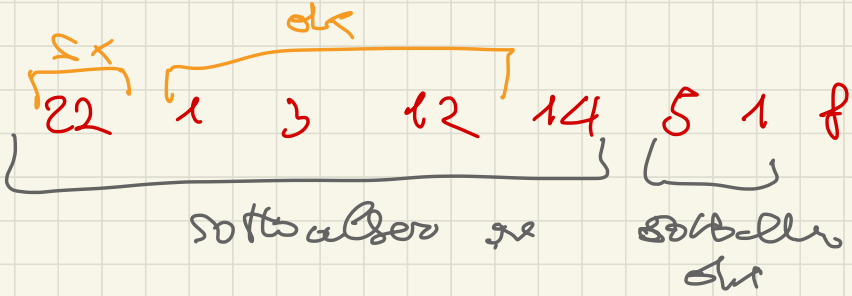
(In-order)



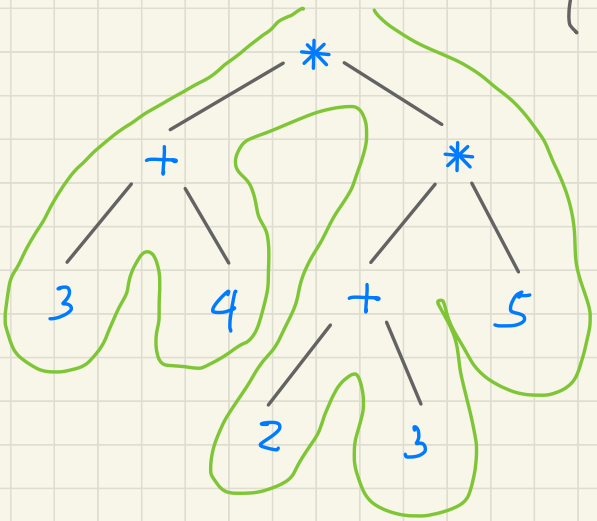
```
ALGORITMO visitaInProfondita (AlberoBinario r)
  IF r ≠ null THEN
    visitaInProfondita (r.sx)
    visitaInProfondita (r.dx)
    visita la radice
```



ORDINE POSTICIPATO (Post-order)



$(3+4)*(2+3)*5$



VISITA IN AMPIEZZA

* + * 3 4 + 5 2 3

ANTICIPATO * + 3 4 * + 2 3 5

SIMMETRICO 3 + 4 * 2 + 3 * 5

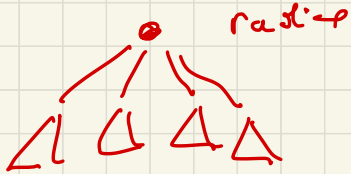
POSTICIPATO 3 4 + 2 3 + 5 * *

notazione postfissa (polacca inversa)

3 5
4 2 5 25
3 7 7 7 175

Alberi generici

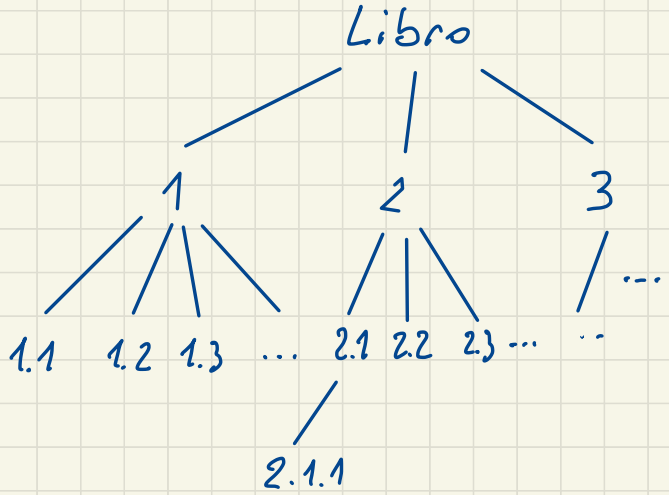
ALBERI (con radice)

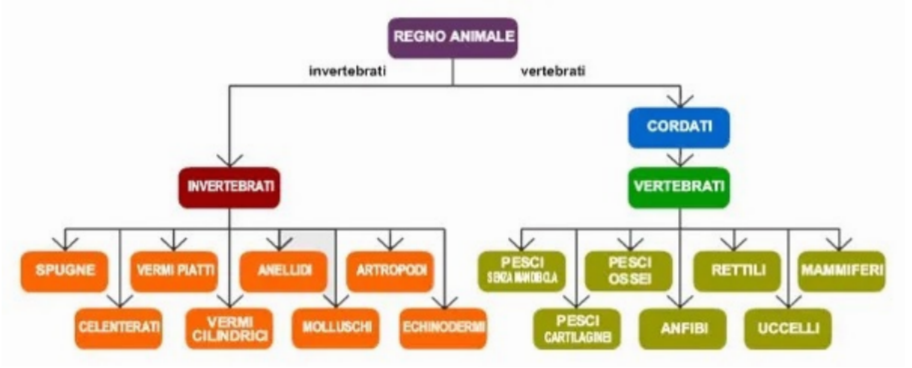


n° arbitrario di figli

Esempio: indice di un libro

1. Introduzione
   1.1 Numeri di Fibonacci
   1.2 Algoritmo numerico
   1.3 Algoritmo ricorsivo
   ...
2. Modelli e metodologie
   2.1 Modelli di calcolo
   2.1.1 Criteri di costo
   2.2 Notazioni asintotiche
   2.3 ...
   ...
3. Strutture dati elementari



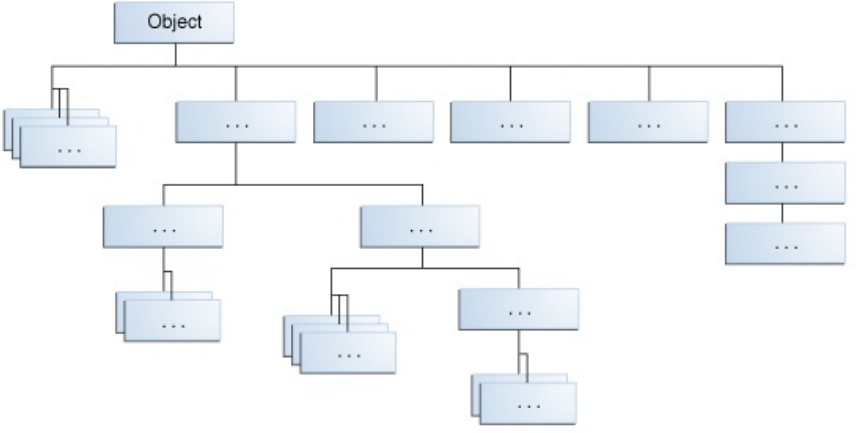


Mergesort (0, 8)

Mergesort (0,4)

Mergesort (4,8)

Gerarchia delle classi Java



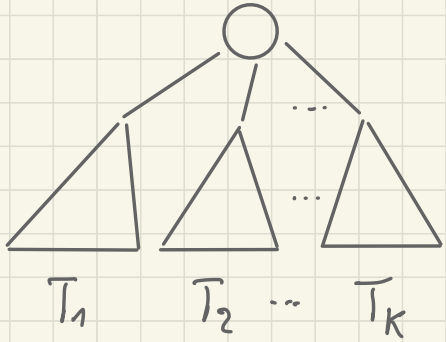
Albero con radice : definizione ricorsiva

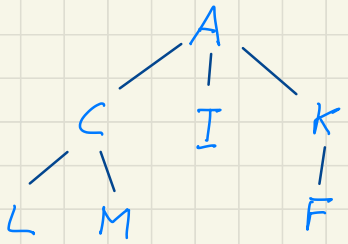
Un albero (con radice) è:

- la struttura vuota

oppure

- un nodo cui sono associati $K \geq 0$ alberi





C, I, K fratelli

grado di un nodo (albero con radici) → # figli

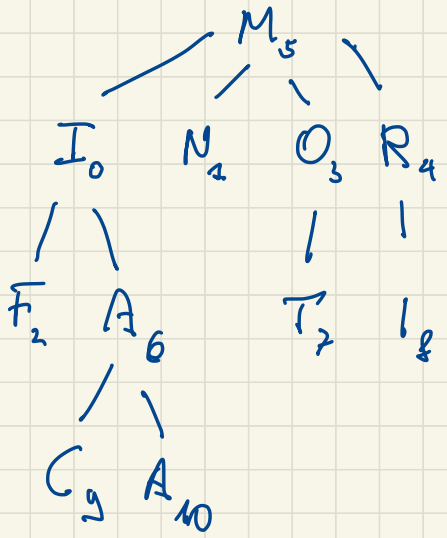
A: grado 3

C: grado 2

grado albero = max grado dei nodi

Rappresentazione: vettore dei padri: / array con i valori contenuti nei nodi
\ array con la posizione del padre di ciascun nodo

	0	1	2	3	4	5	6	7	8	9	10
info	I	N	F	O	R	M	A	T	I	C	A
padre	5	5	0	5	5	-1	0	3	4	6	6



Rappresentazione: vettore dei figli

array con i valori contenuti nei nodi

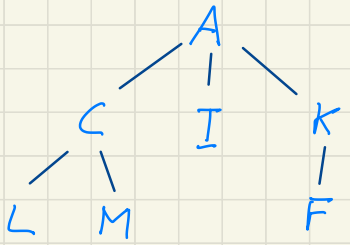
d array contenenti le posizioni dei figli di ogni nodo

grado dell'albero

	0	1	2	3	4	5	6	7	8	9	10
info	I	N	F	O	R	M	A	T	I	C	A
figlio1	2	-1	-1	7	8	0	9	-1	-1	-1	-1
figlio2	6	-1	-1	-1	-1	1	10	-1	-1	-1	-1
figlio3	-1	-1	-1	-1	-1	3	-1	-1	-1	-1	-1
figlio4	-1	-1	-1	-1	-1	4	-1	-1	-1	-1	-1

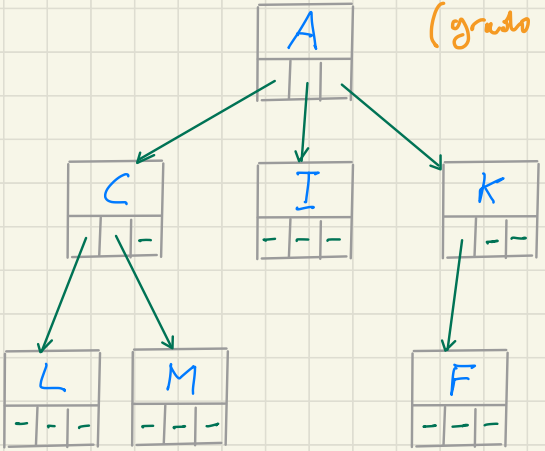
radice 5

Rappresentazioni collegate

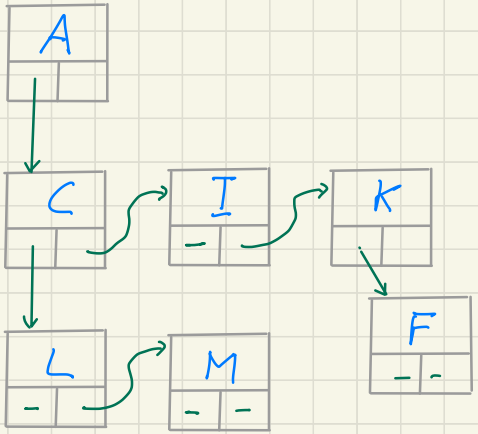


puntatori ai figli

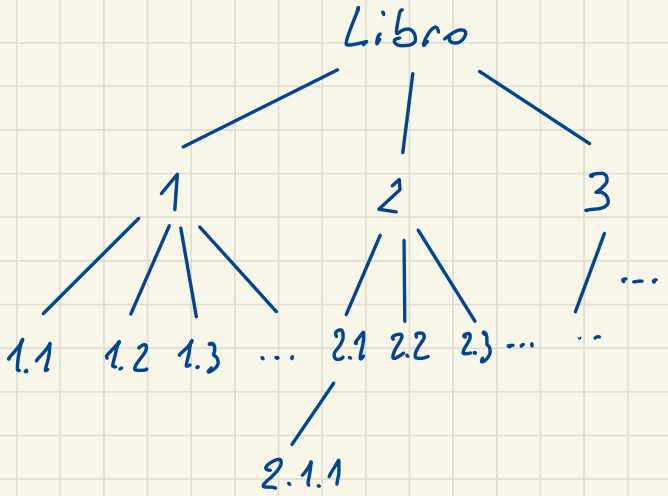
(grado 3)

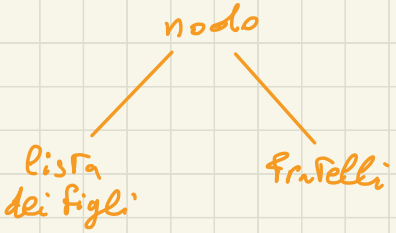


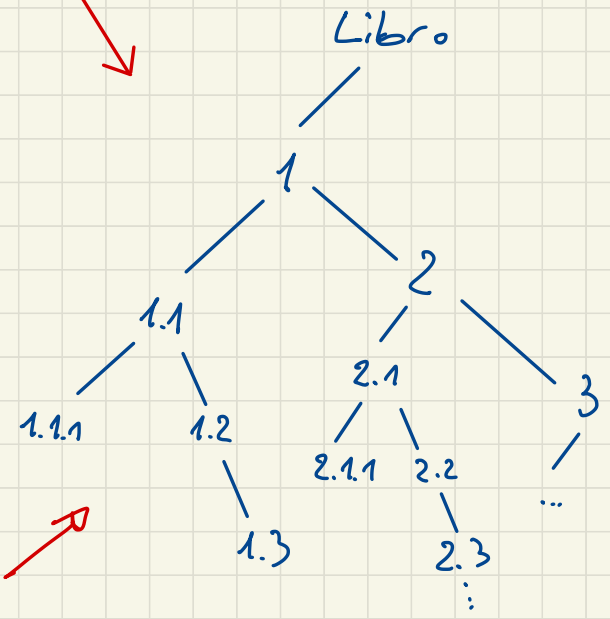
lista dei fratelli



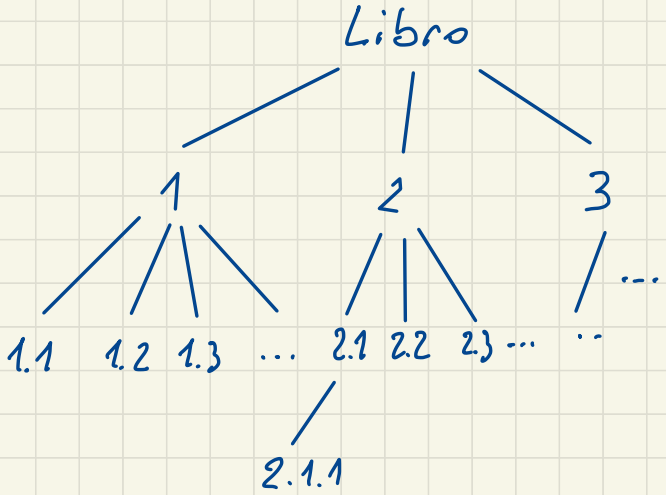
ALBERI GENERICI → ALBERI BINARI

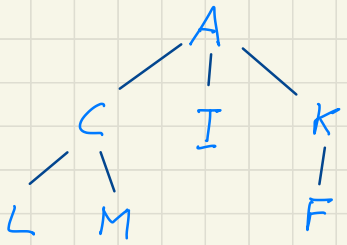






indice ≡
visita in
ordine anticipato





Visite
- ampiezza
- profondità

Alberi binari: n° nodi vs altezza

n

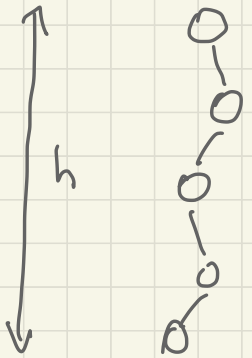
h

Che relazioni ci sono tra numero di nodi e altezza in un albero binario?

Alberi binari: n° nodi vs altezza

n h

- Numero minimo di nodi per alberi di altezza h

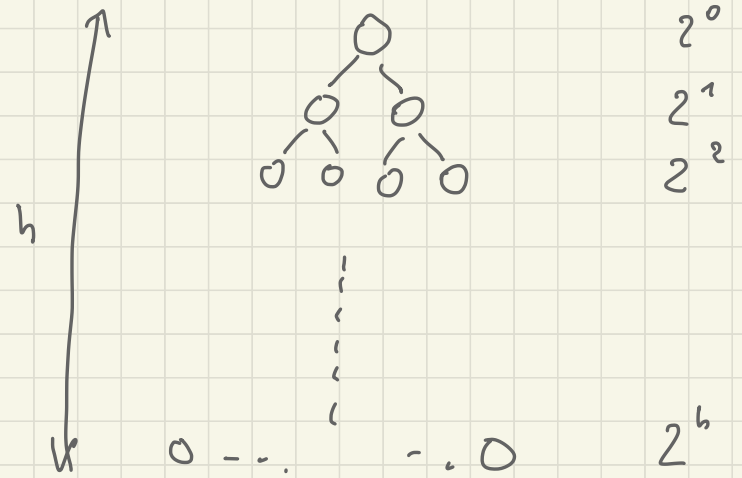


$n = h + 1$

Alberi binari: n° nodi vs altezza

n h

- Numero massimo di nodi per alberi di altezza h



$$\sum_{i=0}^{h} 2^i = 2^{h+1} - 1$$

Alberi binari: n° nodi vs altezza

n h

$$h+1 \leq n < 2^{h+1}$$

$h < n$ $\quad \log_2 n < h+1 \quad \log_2 n \leq h$

$$\Rightarrow \log_2 n \leq h < n$$