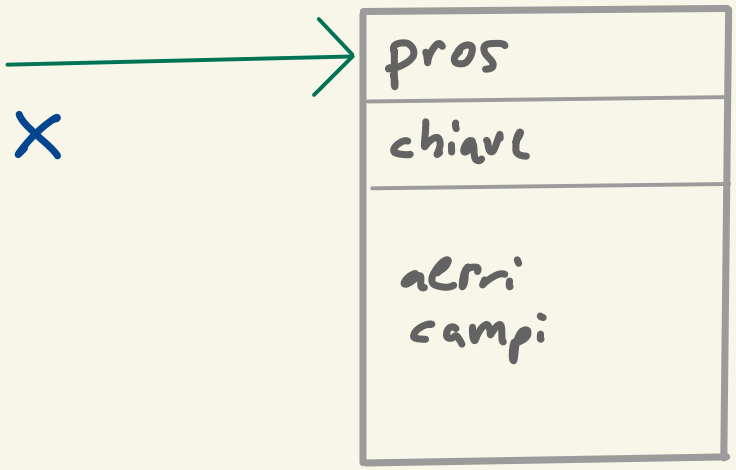
Algoritmi e Strutture Dati

Lezione 13

22 ottobre 2025

Liste concatenate

Accesso ai nodi tramite riferimenti (puntatori)



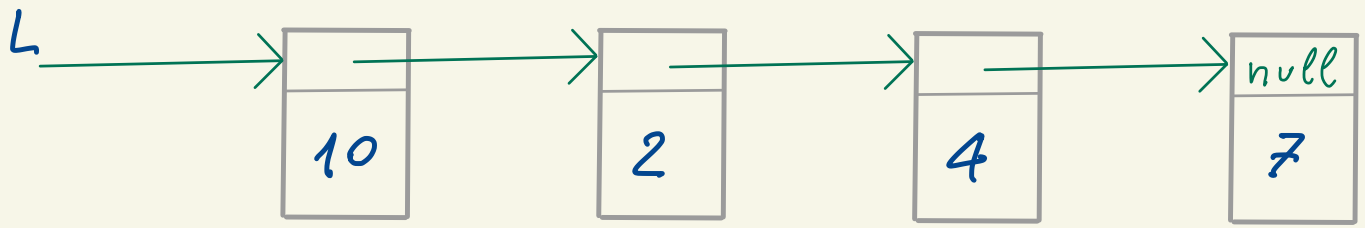
x ≡ riferimento al nodo

x.chiave ≡ campo chiave

x.pros ≡ riferimento a nodo successivo

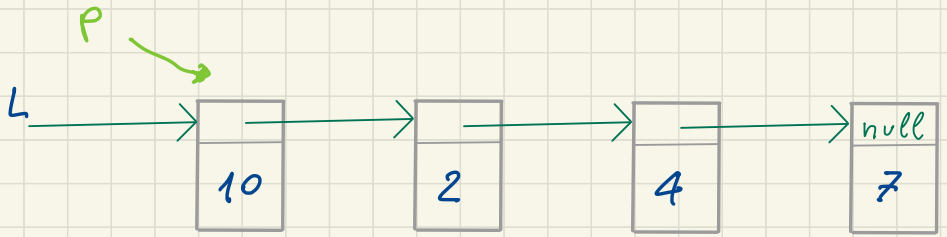
null ≡ riferimento nullo

Liste concatenate



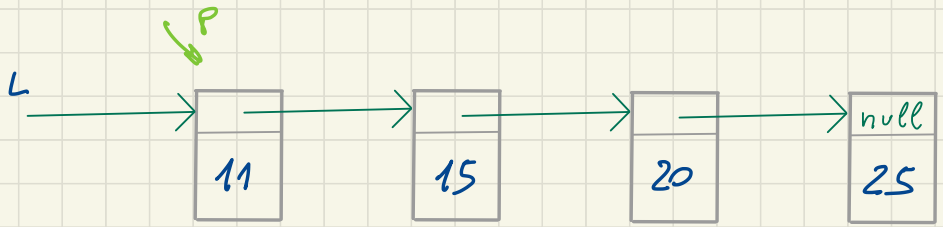
Ricerca elemento in base alla chiave

k [2,



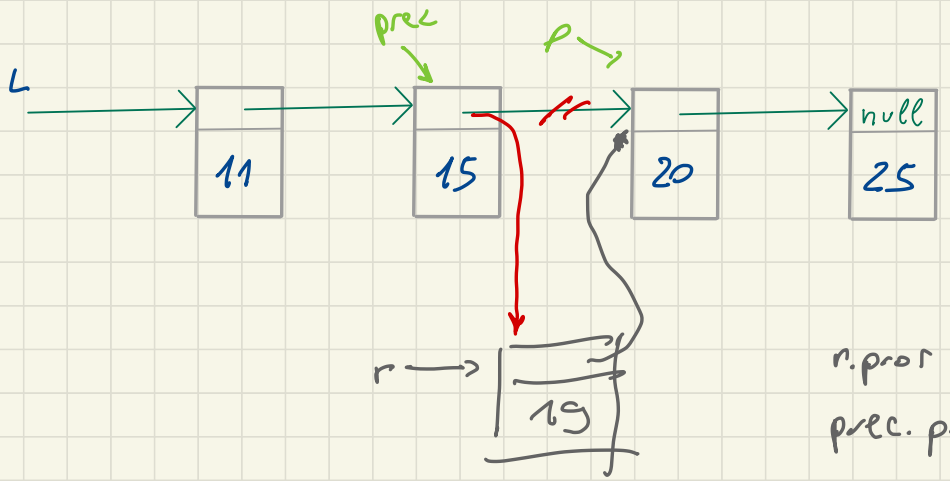

```
FUNZIONE trova (Lista L, tipoChiave K) → Nodo
    p ← L
    WHILE p ≠ null AND p.chiave ≠ K DO
        p ← p.pros
    RETURN p
```

Ricerca elemento in base alla chiave in una lista ORDINATA




```
FUNZIONE trova (ListaOrdinata L, tipoChiave K) -> Nodo
    p ← L
    WHILE p ≠ null AND p.chiave < K DO
        p ← p.pros
    IF p = null OR p.chiave > K THEN
        RETURN null
    ELSE RETURN p
```

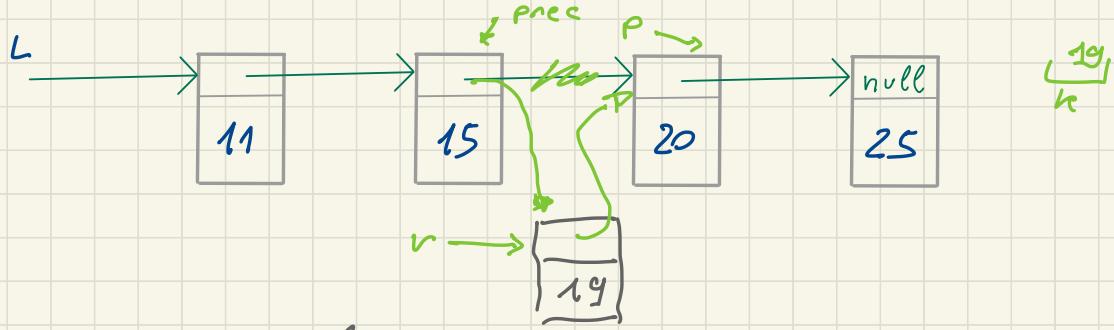
Inserimento in una lista ordinata



inserisci 19

r.pros ← p

prec.pros ← r

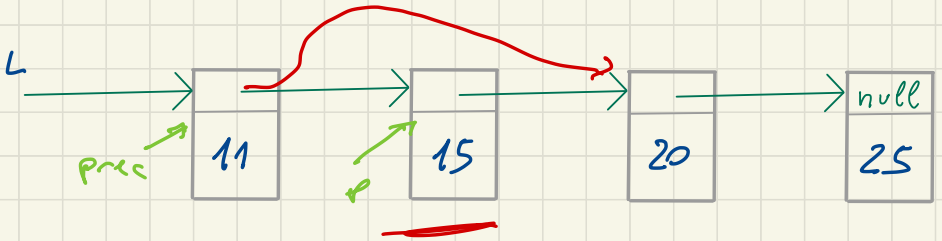


FUNZIONE inserisci (ListaOrdinata L, elemento d) → ListaOrdinata

```
k ← d.chiave
p ← L
prec ← null
WHILE p ≠ null AND p.chiave < k DO
    prec ← p
    p ← p.pros

r ← riferimento a nuovo nodo
r.chiave ← k
r.altri campi ← d.altri campi
r.pros ← p                          prec.pros ← r
IF prec = null THEN
    L ← r
ELSE prec.pros ← r
RETURN L
```

Cancellazione



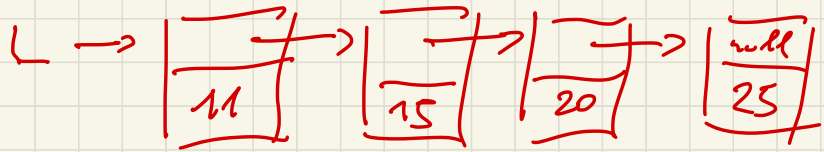
prec.pros ← p.pros ↝ rilasciare la memoria

caso particolare: cancellazione all'inizio

Implementazione tramite array

chiave	1	15	25	11	3	20	...
pros	7	5	-1	1	-1	2	...
altri campi	...	...	...	...	...	...	...
	0	1	2	3	4	5	

3



Implementazione tramite puntatori

Tipo Pila

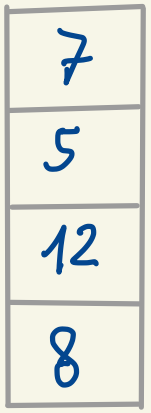
Stack

Collezione di dati con organizzazione *Last–In–First–Out*

LIFO

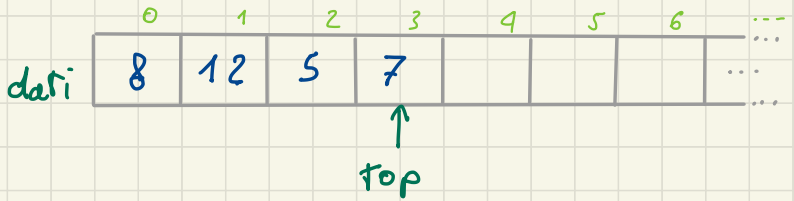
Operazioni

- isEmpty() → boolean
- push(elemento)
- pop() → elemento
- top() → elemento



PILA: implementazione mediante array

7
5
12
8



FUNZIONE isEmpty() -> boolean

IF top = -1 THEN RETURN true

ELSE RETURN false

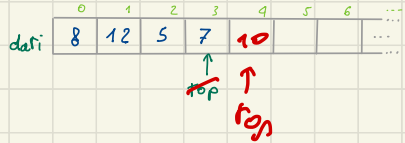
PROCEDURA push (elemento x)

top ← top+1

dati[top] ← x

Errore se array è pieno!

10
7
5
12
8



FUNZIONE top() → elemento

RETURN dati[top]

FUNZIONE pop() → elemento

x ← dati[top]

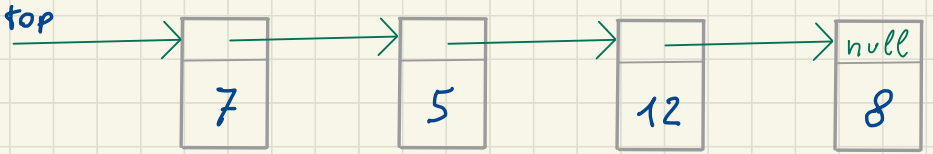
top ← top − 1

RETURN x

Errore se pila vuota!

PILA: implementazione mediante liste

lista vuota
null
top



elemento più in alto

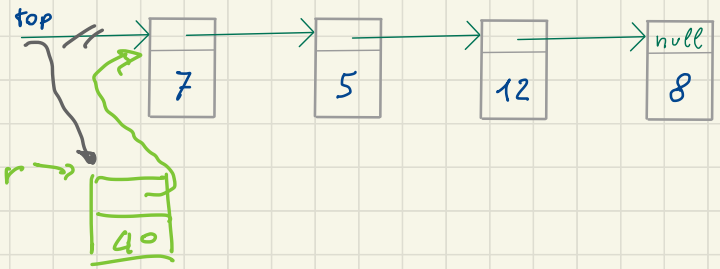
7
5
12
8

FUNZIONE isEmpty() → boolean

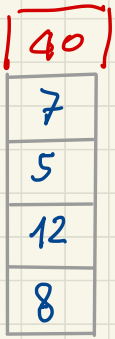
IF top = null THEN

RETURN true

ELSE RETURN false



push (40)



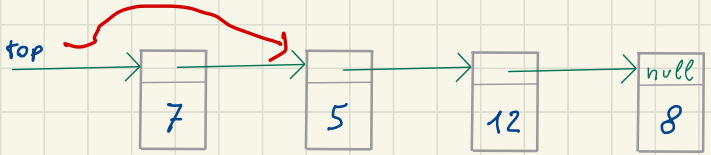
PROCEDURA push (elemento x)

r ← riferimento a un nuovo nodo

r.dato ← x

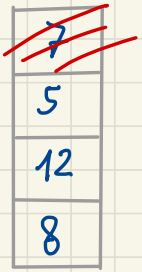
r.prox ← top

top ← r



FUNZIONE top() → elemento

RETURN top.data



FUNZIONE pop() → elemento

x ← top.data

top ← top.pros ; rilascio slack memoria

RETURN x

Tempo operazioni: $O(1)$

Tipo Coda

Collezione di dati con organizzazione *First–In–First–Out*

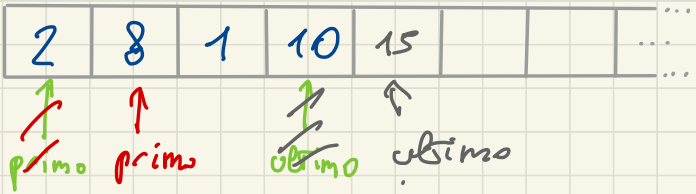
FIFO

Operazioni

- isEmpty() $\rightarrow$ boolean
- enqueue(elemento)
- dequeue() $\rightarrow$ elemento
- first() $\rightarrow$ elemento

4 10 15 7

CODA: implementazione mediante array

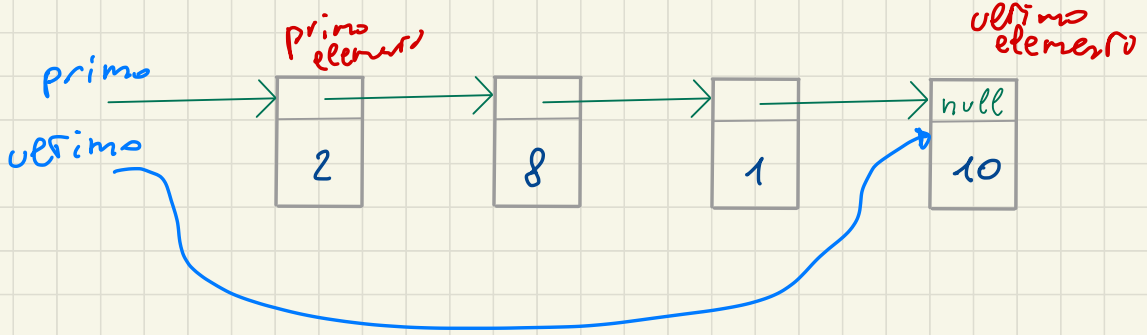


enqueue(15)

dequeue()

array "circolare"

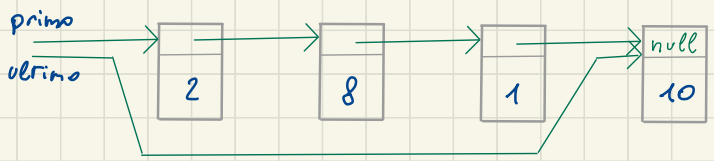
CODA: implementazione mediante liste



Code vuota

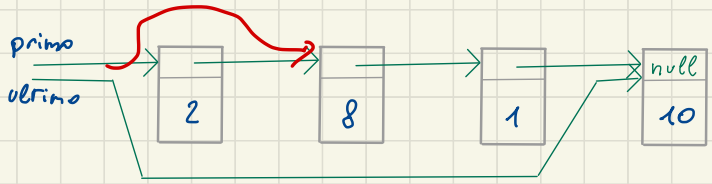
primo (null)

ultimo (null)

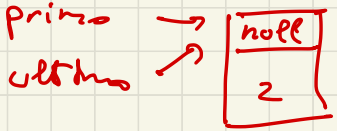



```
FUNZIONE   isEmpty() → boolean
    IF  primo = null THEN
        RETURN true
    ELSE RETURN false

FUNZIONE   first() → elemento
    RETURN  primo.dato
```



x 2



primo null

ultimo null

FUNZIONE dequeue () → elemento

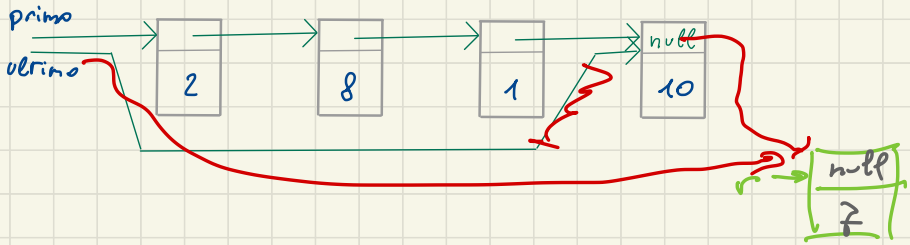
x ← primo.dato

primo ← primo.pros

IF primo = null THEN

ultimo ← null

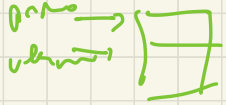
RETURN x




```
PROCEDURE enqueue (elemento x)
    r ← riferimento a un nuovo nodo
    r.dato ← x
    r.pros ← null
    IF primo = null THEN
        primo ← r
        ultimo ← r
    ELSE
        ultimo.pros ← r
        ultimo ← r
```

```
ultimo.pros ← r
ultimo ← r
```

caso di coda vuota

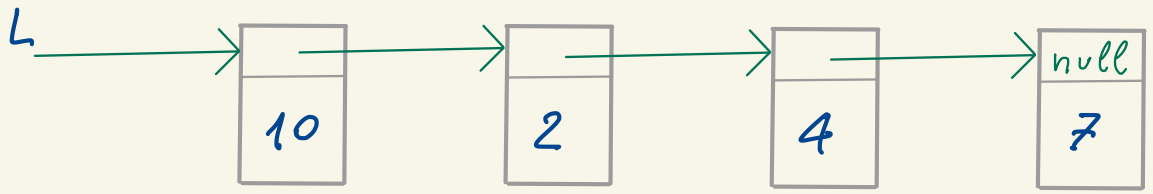


Tempo $O(1)$

Dalle liste agli alberi

Liste lineari

- Collezioni di nodi collegati tramite riferimenti



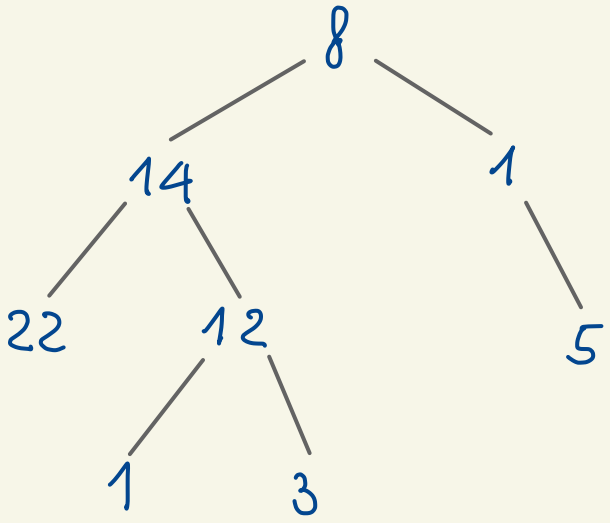
- Ad ogni nodo è possibile associare un *unico successore*

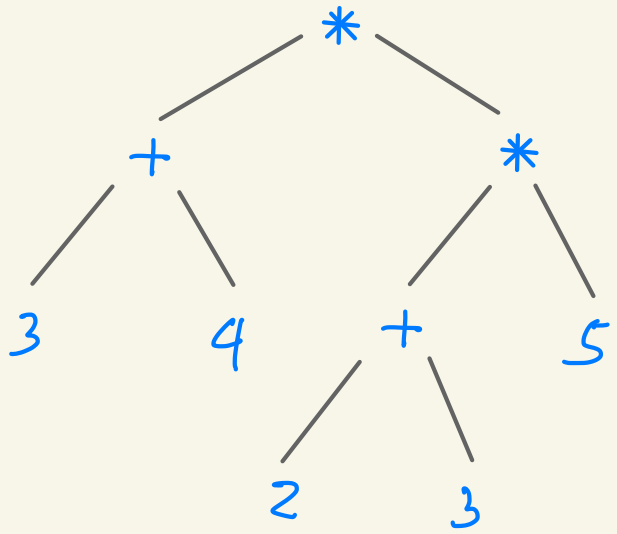
Mediante riferimenti/puntatori è possibile definire strutture più complicate, ma più flessibili

Dalle liste agli alberi

Alberi binari

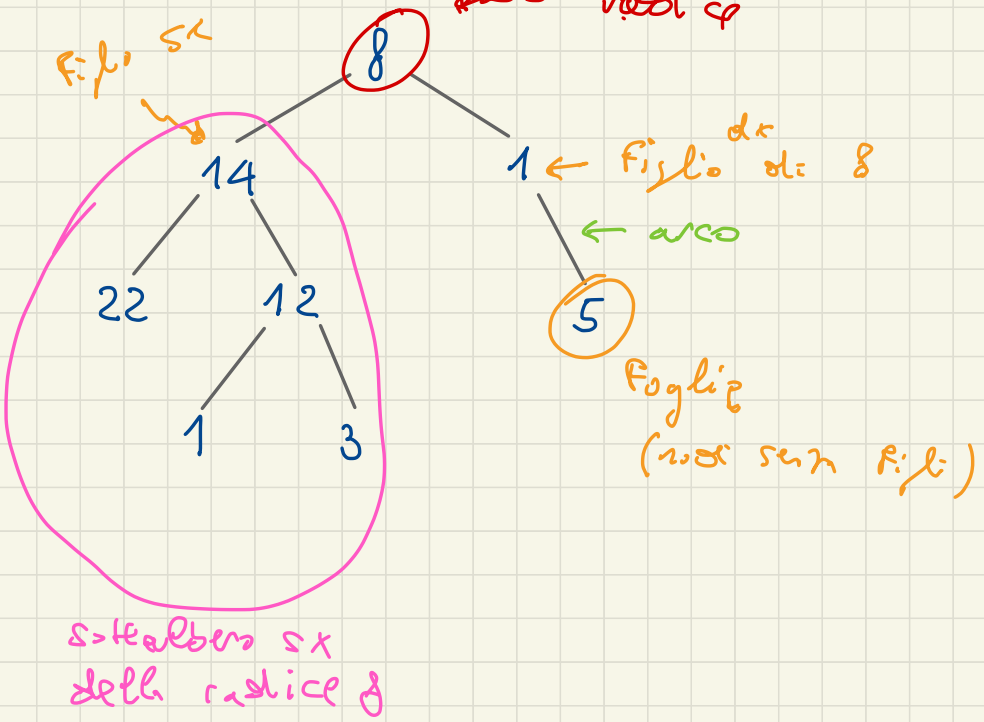
- Ad ogni nodo possono essere associati *due* "successori" detti *figlio sinistro* e *destro*





(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