LISTE

8 * → 12 → 15 ⊣ NIL
Nodo Nodo Nodo

LISTA SEMPLICEMENTE LINKATA
LISTA DOPPIAMENTE LINKATA

NIL ← 8 ⇄ 12 ⇄ 14 → NIL

HEAD *Nodo
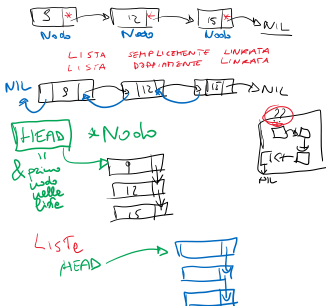il
& primo
Nodo
della
Lista

9
12
15

Liste →
HEAD

COSA POSSO FARE CON UNA LISTA?

1. INSERIRE UN NODO NUOVO ELEMENTO
2. ELIMINARE UN ELEMENTO
3. RICERCA DI UN ELEMENTO
3a. RICERCA DEL MASSIMO
3b. RICERCA DEL MINIMO
4. MODIFICA DI UN ELEMENTO

EDITING   SEARCH

HEAD = NIL
a. INSERT   10
b. INSERT   15
c. INSERT   20
d. INSERT   5
e. DELETE   15
f. EDIT   10→12

a. HEAD → NIL
I) Creo un nodo con valore 10
II) HEAD punta al nuovo nodo
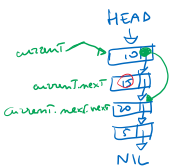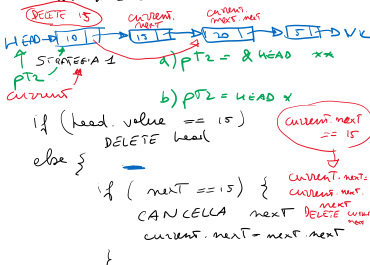b. 1) Creo un nodo con valore 15
II) il nodo precedente punta a 15

HEAD
↓
10 → 15 → 20 → 5 ⊣ NIL

V==15?   V==15

ELIMINAZIONE

STRATEGIA 1
previsione del futuro
look forward list next
Nodo corrente ≤ == 15
Nodo successivo == 15

STRATEGIA 2
aiuto da casa
Nodo temporaneo che è
sempre un passo indietro
al nodo che sto interrogando

DELETE 15   current.next   current.next.next
HEAD → 10 → 15 → 20 → 5 ⊣ DVK
  PT2   STRATEGIA 1
current

a) PT2 = & HEAD **
b) PT2 = HEAD *

current.next == 15

if (head.value == 15)
    DELETE head
else {
    if (next == 15) {
        current.next =
        current.next.next
        CANCELLA next   DELETE next
        current.next = next.next
    }
}

HEAD
↓
10
current → 15
current.next → 20
current.next.next → 5
NIL

DELETE 15
1. head.value == 15?
2. current = head
3. current.next.value == 15? SI
4. current.next = current.next.next
5. DELETE 15

STRATEGIA 2   Aiuto da casa
prev = NULL
current
HEAD
↓
10
15
20
5
NIL
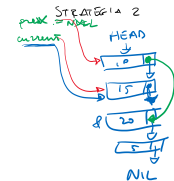
1. current.value == 15?
   prev = current
   current = current.next   NO
2. prev.next == current.next
   next   SI
   DELETE 15

NODO	LISTA

- value : T
- next : Nodo *

+ Nodo ()
+ Nodo (T value)
+ Nodo (T value, Nodo* next)
+ next() : Nodo*
+ value() : T
+ setNext (Nodo *) : void
+ setValue (T value) : void
+ toString () : std::string

LISTA
- head : Nodo<T>*
- lunghezza : int

+ Lista ()
+ Lista (Nodo<T>* head)
+ insert (Nodo<T> * n) : void
+ insert (T value) : void
+ search (T value) : Nodo<T>*
+ delete (T value) : void
+ min ()
+ max ()

current HEAD
→ 10
NIL
n ⊣
current.next == NULL