Algoritmi e Strutture Dati (08/11/2021)

* Alberi di Ricerca

albero

→ nodi x $x.key$

→ operazioni $O(h)$

(se bilanciato : $O(\log n)$)

→ alberi binari di ricerca
→ alberi RED-BLACK
→ alberi AVL
→ B-alberi

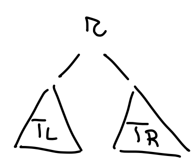
* Alberi binari di Ricerca (ABR / BST)

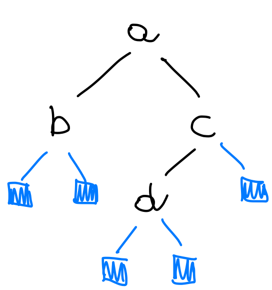
alberi binari ordinati (figlio sx, figlio dx)

definizione induttiva: albero binario

→ albero vuoto 

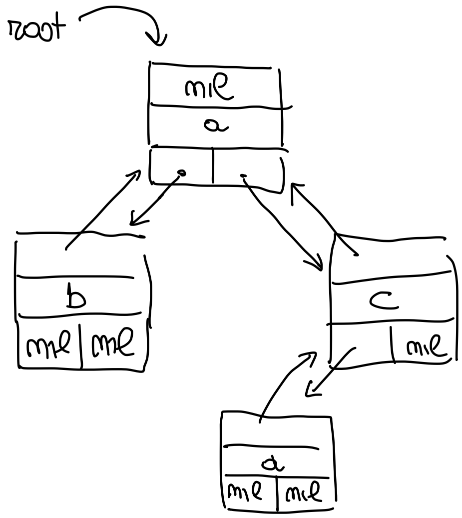
→ (r, T_L, T_R) dove r nodo
T_L, T_R alberi binari





in memoria

x — $x.key$, $x.p$, $x.left$, $x.right$ — nil (se vuoto)

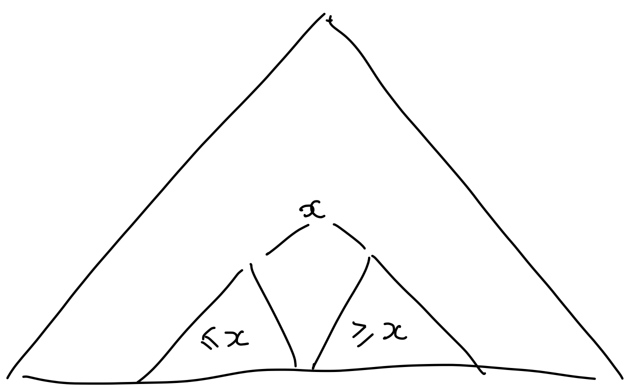


* Albero binario di Ricerca

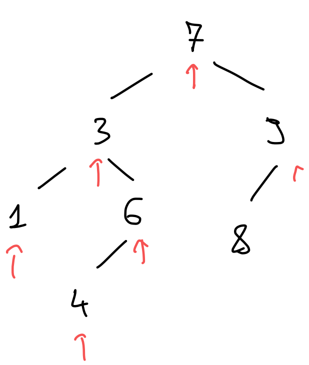
albero binario (ordinato) per ogni nodo x

→ ogni nodo y nel sottoalbero sx di x $\quad y.key \leq x.key$

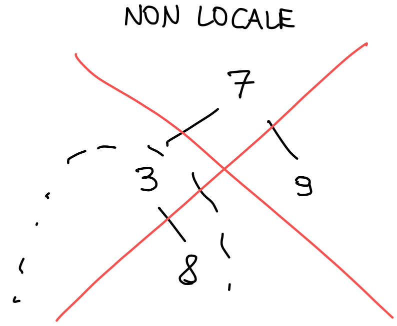
→ " " " " " " " " dx di x $\quad y.key \geq x.key$



Esempio :



PROPRIETÀ NON LOCALE

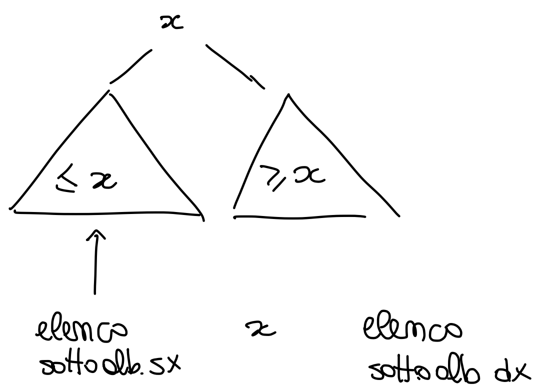


ESERCIZIO : Funzione IsABR (T) ritorna true / false a seconda che T sia o meno un ABR

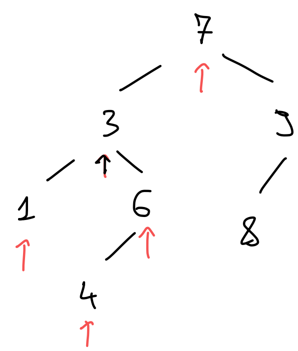
* Operazioni :

* In Order

elencare gli elementi del sottoalbero radicato in un nodo x
in ordine di chiave crescente




```
InOrder (x)
  if x ≠ nil
    InOrder (x.left)
    print x            Θ(1)
    InOrder (x.right)
```



1 3 4 6 7 . . .

Esercizio 1 Dato T albero binario ordinato

InOrder (T.root)
elenca gli elem. di
T in ordine crescente

sse T è un ABR

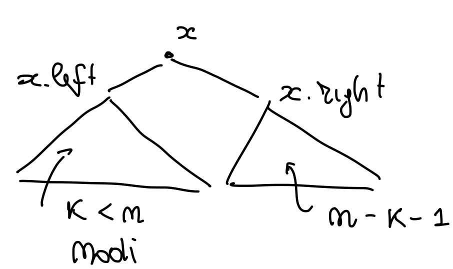
⇐

⇒

Complessità : $\Theta(n)$ (è una visita dell'albero)

n nodi

$$T(n) = T(k) + T(n-k-1) + \Theta(1)$$



$$T(n) = \begin{cases} d & n=0 \\ T(k)+T(n-k-1)+c & n>0 \end{cases}$$

ipotesi

$$= an+b$$

$(n=0)$ $\quad T(0) = d$

$\quad\quad = a\cdot 0 + b = b \quad \Big\} \Rightarrow \boxed{b = d}$

$(n>0)$ $\quad T(n) = T(k) + T(n-k-1) + c \qquad k,\ n-k-1 < n \downarrow$ ip. ind

$= (ak+b) + (a(n-k-1)+b) + c$

$= a(\cancel{k}+n-\cancel{k}-1) + 2b + c$

$= an \underbrace{-a+2b+c}$

$= an+b$

$\Big\} \Rightarrow b = -a+2b+c$

$a = b+c = d+c$

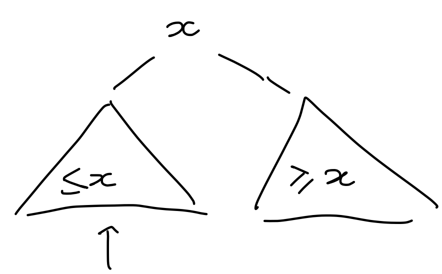
IPOTESI $\quad T(n) = an+b \quad$ verificata con $\quad a = d+c$, $b = d$

$= (c+d)n + d$

* Ricerca

dato k chiave, cerca nel sottoalbero radicato nel nodo x un nodo con chiave k

- se $x.key = k$ ⇝ ritorna x
- se $k < x.key$ ⇝ cerca in $x.left$
- se $k > x.key$ → cerca in $x.right$



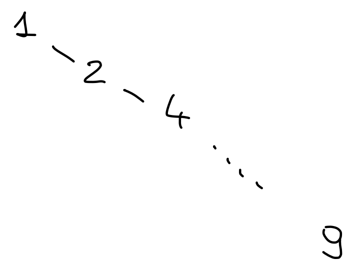

```
Search (x, k)
  if (x ≠ nil) and (x.key ≠ k)
      if k < x.key
          return Search (x.left, k)
      else return Search (x.right, k)
  else return x
```

Complessità : caso peggiore : continuo fino ad una foglia
e il cammino radice-foglia è
quello massimo } → h altezza

$O(h)$ h altezza albero

Nota h può essere $\Theta(n)$

1, 2, 4, 6, 7, 8, 9



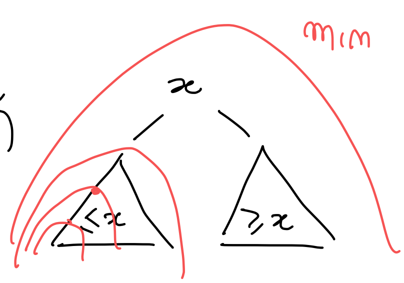
* iterativa

```
search (x, k)
    while (x ≠ nil) and (x.key ≠ k)
        if k < x.key
            x = x.left
        else
            x = x.right
    return x
```

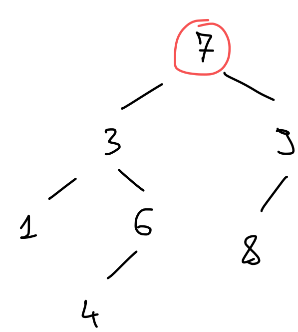
* Minimo

Dato x : minimo nel sottoalbero radicato in x

```
Min (x)        // x ≠ nil
    while (x.left ≠ nil)
        x = x.left
    return x
```



complessità $O(h)$

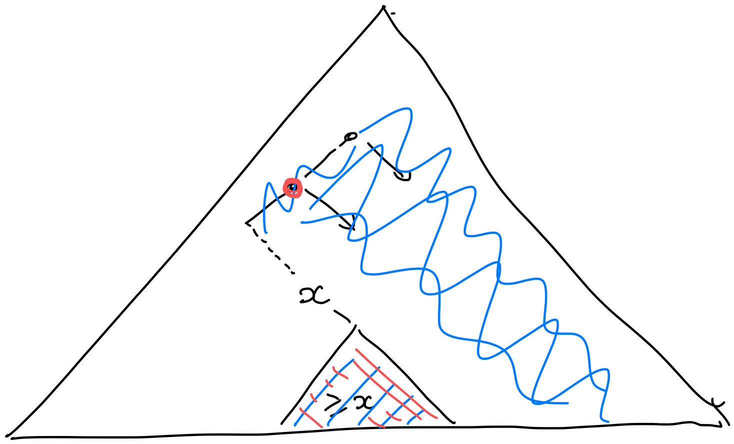


* Successore

Dato x in ABR

→ minimo tra i nodi più grandi di x

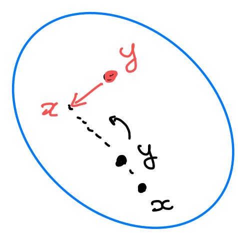
→ nodo che segue x in una visita in ordine

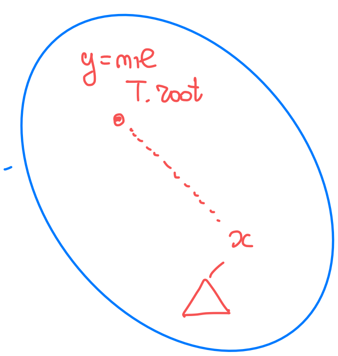


→ se x ha sottoalbero dx non vuoto ⇒ min (x.right)

→ se x. non ha sottoalbero dx ⇒ più vicino antenato di x di cui x è nel sottoalb. sx

```
       Succ (x)            // x ≠ nil
          if  x.right ≠ nil
O(h)          return Min (x.right)
          else
              y = x.p
O(h)          while (y ≠ nil) and (x = y.right)
                    x = y
                    y = y.p
              return y
```



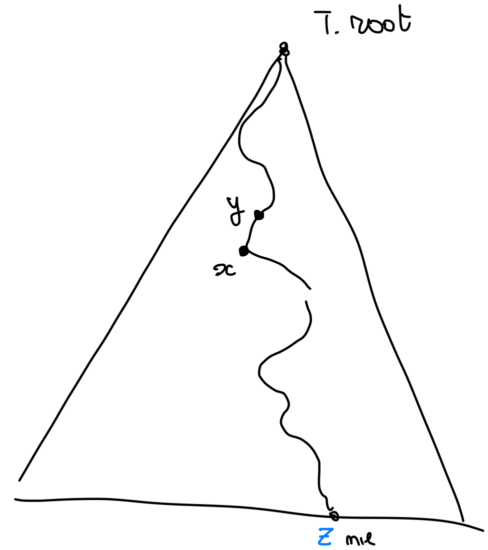


Complessità $O(h)$

* **Insert**

vogliamo inserire nodo z nell'albero ABR T (T.root)

z.key



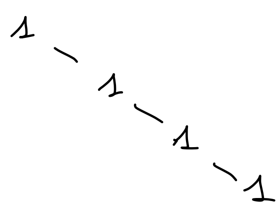

```
Insert (T, z)
    x = T.root
    y = nil
    while (x ≠ nil)
        y = x
        if z.key < x.key
            x = x.left
        else
            x = x.right

    z.p = y

    if y = nil
        T.root = z
    else
        if z.key < y.key
            y.left = z
        else
            y.right = z
```

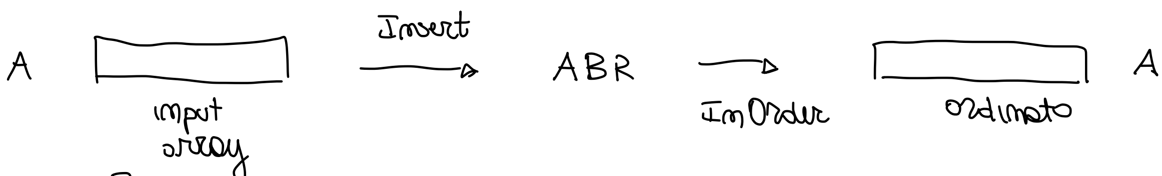
complessità $O(h)$

Ex. 1 , 1 , 1 , 1



ESERCIZI

① Usare ABR per ordinamento



complessità ?

② dato A $\rightsquigarrow$ trasf. in max-heap in $\Theta(n)$

è possibile trasformare A in un ABR quasi completo in tempo lineare ?

è possibile dato max-heap A estrarre gli elem. in ordine crescente in tempo lineare ?

③ Insert ricorsiva