

# Alberi Binari di Ricerca

Vittorio Maniezzo - Università di Bologna

1

## Look-up table, dati ordinati

Collezione di dati nella forma di coppie attributo / valore, in cui i **campi chiave sono ordinabili**.

Caso semplice, array A ordinato (solo chiavi, niente valori), con  $A[1], \dots, A[n]$  crescenti.

Operazioni:

- Inserimento
- Cancellazione
- Ricerca

La ricerca può essere fatta tramite ricerca binaria.

Vittorio Maniezzo - Università di Bologna

2

2

## Ricerca binaria

Ricerca di una chiave  $k$  in un array ordinato  $A[l...r]$

**Algorithm** BinarySearch( $k, A[1...r]$ )

**if**  $l > r$  **then return** //  $k$  non presente

**if**  $l \leq r$  **then**  $i = \lfloor (l+r)/2 \rfloor$

**compare**  $m=A[i]$  with  $k$

• **if**  $k < m$  **then** BinarySearch( $k, A[1 \dots i-1]$ )

• **if**  $k = m$  **then return** //  $k$  is at  $i$

• **if**  $k > m$  **then** BinarySearch( $k, A[i+1 \dots r]$ )

**Complessità:**

Equazione ricorsiva:  $T(n) = O(1)$  se  $n=1$ ;  $T(n/2) + 1$  se  $n > 1$

BinarySearch è  $O(\log n)$

Per esempio, la ricerca in un array di 25000 dati richiede 15 passi

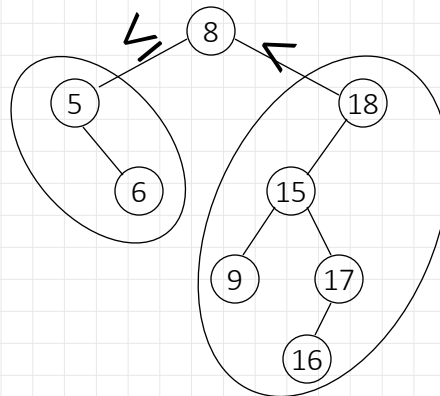
Vittorio Maniezzo - Università di Bologna

3

3

## Alberi di ricerca binari

- Tutti i nodi nel sottoalbero **sinistro** di un nodo hanno chiave **inferiore** a quella del nodo.
- Tutti i nodi nel sottoalbero **destro** di un nodo hanno chiave **superiore** a quella del nodo.

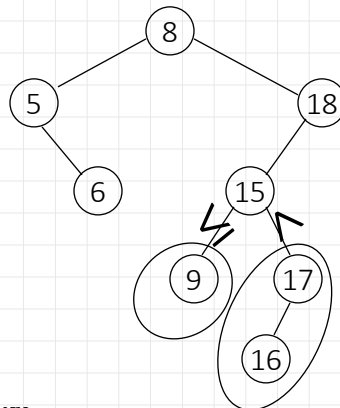


Vittorio Maniezzo - Università di Bologna

4

4

## Alberi di ricerca binari



Vittorio Maniezzo - Università di Bologna

5

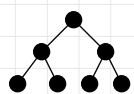
5

## BST: definizione formale

Sia  $x$  un nodo dell'albero:

- Se  $y$  è un nodo nel sottoalbero sinistro di  $x$  allora  $\text{key}[y] \leq \text{key}[x]$
- Se  $y$  è un nodo nel sottoalbero destro di  $x$  allora  $\text{key}[y] > \text{key}[x]$

Nota che un BST può essere molto sbilanciato !!!



bilanciato



sbilanciato

*Cfr. heap*: anche gli alberi binari possono essere rappresentati con un vettore, con la convenzione sugli indici dei figli.

Alberi binari non sono quasi completi → **ci sono celle vuote** all'interno dell'array

Vittorio Maniezzo - Università di Bologna

6

6

## Esercizi

Domande

- Presentare una min-heap che non è un BST
- Presentare un BST che non è una min-heap
- Elencare tutti i BST per le chiavi 1, 2, 3

Quanti BST con n nodi esistono? Numeri di Catalan.

$$C_n = \frac{1}{n+1} \binom{2n}{n}$$

Primi numeri di Catalan: 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420, 24466267020, 91482563640, 343059613650, 1289904147324, 4861946401452, ...

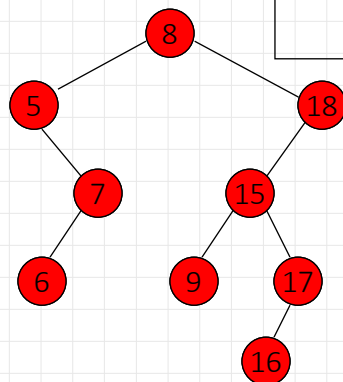
Vittorio Maniezzo - Università di Bologna

7

7

## Ordinamento

```
Inorder-tree-walk(x)
  if x ≠ nil
    then Inorder-tree-walk(left[x])
         print key[x]
         Inorder-tree-walk(right[x])
```



N.B: possibili anche:  
Preorder-tree-walk  
Postorder-tree-walk

Vittorio Maniezzo - Università di Bologna

8

8

## Traversal: complessità

Si visitano tutti i nodi  $\rightarrow \Omega(n)$

Equazione ricorsiva,  $k$  num nodi nel sottoalbero sinistro

$$T(0) = c$$

$$T(n) = T(k) + T(n-k-1) + d$$

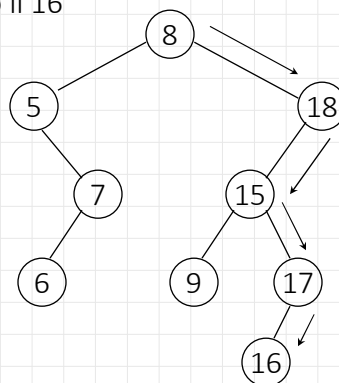
Si ricava:  $T(n) \leq (c+d) \cdot n + c$

Quindi:  $T(n)$  in  $O(n)$

Complessivamente:  $T(n) \in \Theta(n)$

## Ricerca

ricerchiamo il 16



$\Theta(h)$  confronti  
 $h$ =altezza albero

## Ricerca

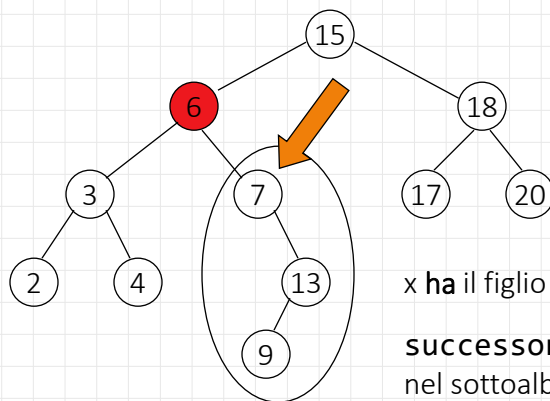
Tree-search( $x, k$ )

```
if  $x = \text{nil}$  or  $k = \text{key}[x]$ 
  then return  $x$ 
if  $k < \text{key}[x]$ 
  then return Tree-search(left[ $x$ ],  $k$ )
  else return Tree-search(right[ $x$ ],  $k$ )
```

Esercizio: dimostrare che il costo computazionale di Tree-search è  $\Theta(h)$

11

## Successore



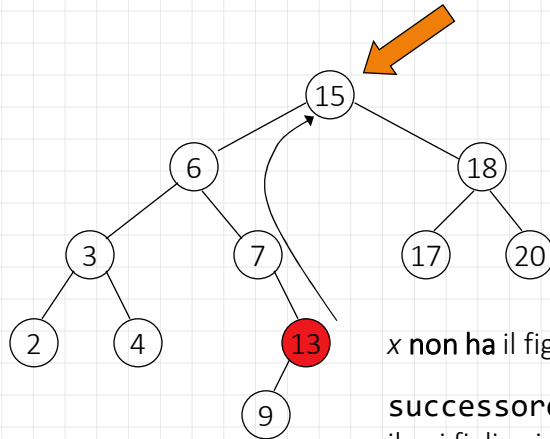
$x$  ha il figlio destro.

**successore( $x$ )** = minimo  
nel sottoalbero di destra

Dimostrazione: Esercizio.

12

## Successore



$x$  non ha il figlio destro.

**successore( $x$ )** = il più basso avo di  $x$   
il cui figlio sinistro è avo di  $x$

Dimostrazione: Esercizio.

## Minimo e massimo

```
algorithm treeMinimum(x)
while left(x)  $\neq$  nil
do x = left(x)
return x
```

```
algorithm treeMaximum(x)
while right(x)  $\neq$  nil
do x = right(x)
return x
```

## Operazioni su BST

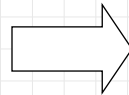
Ricerca

Minimo

Massimo

Predecessore

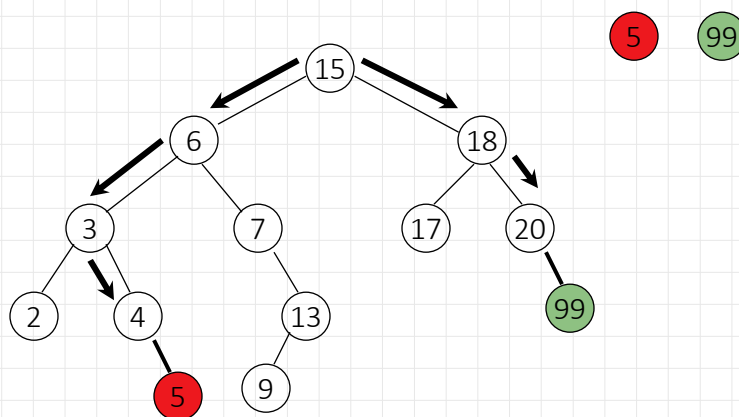
Successore



$\Theta(h)$   
confronti

15

## Inserimento




16



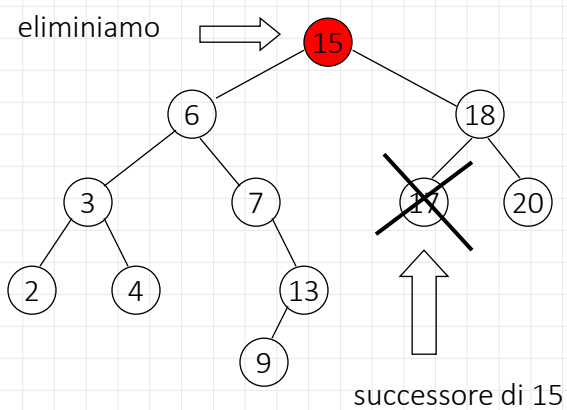
# Cancellazione

3 casi:

- x non ha figli: elimina x
- x ha un figlio: 
- x ha 2 figli:
  - Lemma: il successore di x sta nel sottoalbero destro e ha al massimo 1 figlio.
  - Dimostrazione: esercizio.

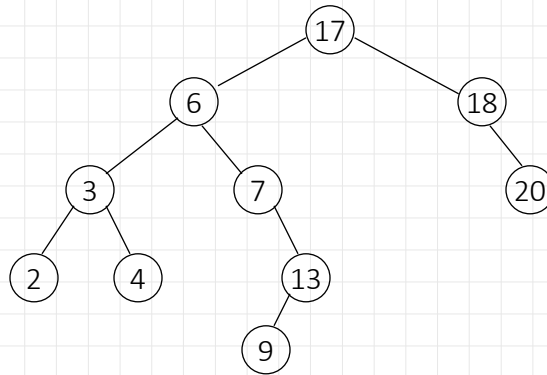
17

# Cancellazione



18

## Cancellazione



17 ha preso il posto di 15

## Cancellazione

Cancellazione di un nodo  $x$  con 2 figli:

1. sia  $y =$  successore di  $x$ .  
//  $y$  ha al Massimo un solo figlio
2. sostituisci  $x$  con  $y$ .
3. rimuovi  $y$ .

## Cancellazione

```
Algorithm treeDelete(T,z)
if (left(z) = nil)
then transplant(T,z,right(z))
else
  if (right(z) = nil)
  then transplant(T,z,left(z))
  else y = treeMinimum(z.right(z))
    if (p(y) ≠ z)
    then transplant(T,y,right(y))
      right(y) = right(z)
      p(right(y)) = y
  transplant(T,z,y)
  left(y) = left(z)
  p(left(y)) = y
```

Cortocircuita u

```
Algorithm transplant(T,u,v)
if p(u) = nil
then root(T) = v
else if (u = left(p(u)) )
then left(p(u)) = v
else right(p(u)) = v
if v ≠ nil then p(v) = p(u)
```

Vittorio Maniezzo - Università di Bologna

21

21

## Problema

Tutte le operazioni su BST hanno un costo lineare nell'altezza dell'albero.

Purtroppo, quando l'albero è sbilanciato,

$$h = n-1$$

Le operazioni hanno un costo lineare invece che logaritmico come speravamo !!! Soluzione ???

Vittorio Maniezzo - Università di Bologna

22

22

## Soluzione

---

Introduciamo alcune **proprietà aggiuntive** sui BST per mantenerli bilanciati.

Paghiamo in termini di una **maggiore complessità** delle operazioni dinamiche sull'albero.

Tali operazioni devono infatti preservare le proprietà introdotte per mantenere il bilanciamento.

Strutture dati di questo tipo sono gli *alberi rossoneri* (*red-black trees*), gli *alberi / heap binomiali*, le *heap di Fibonacci*, i *B-tree*, ecc. (non coperti dal corso)