

## Interrogazioni su un albero binario di ricerca

- **Interrogazioni** - Operazioni che restituiscono informazioni su un insieme dinamico. Non modificano l'insieme.

**Search( $S, k$ )** - dato un insieme  $S$  ed un valore chiave  $k$  restituisce un puntatore  $x$  ad un elemento in  $S$  tale che  $key[x] = k$

**Maximum( $S$ )** - dato un insieme  $S$  totalmente ordinato restituisce l'elemento di  $S$  con chiave piu' grande

**Minimum( $S$ )** - dato un insieme  $S$  totalmente ordinato restituisce l'elemento di  $S$  con chiave piu' piccola

**Successor( $S, x$ )** - dato un insieme  $S$  totalmente ordinato ed un elemento  $x$  di  $S$ , restituisce il successivo elemento piu' grande in  $S$  (o NIL se  $x$  e' l'elemento massimo).

**Predecessor( $S, x$ )** - dato un insieme  $S$  totalmente ordinato ed un elemento  $x$  di  $S$ , restituisce il successivo elemento piu' piccolo in  $S$  (o NIL se  $x$  e' l'elemento minimo).

## Ricerca in un ABR

```
Tree-search(x,k)
If (x=NIL) o (k = key[x])
    then return x
If (k < key[x])
    then Tree-Search(left[x],k)
    else Tree-Search(right[x],k)
```

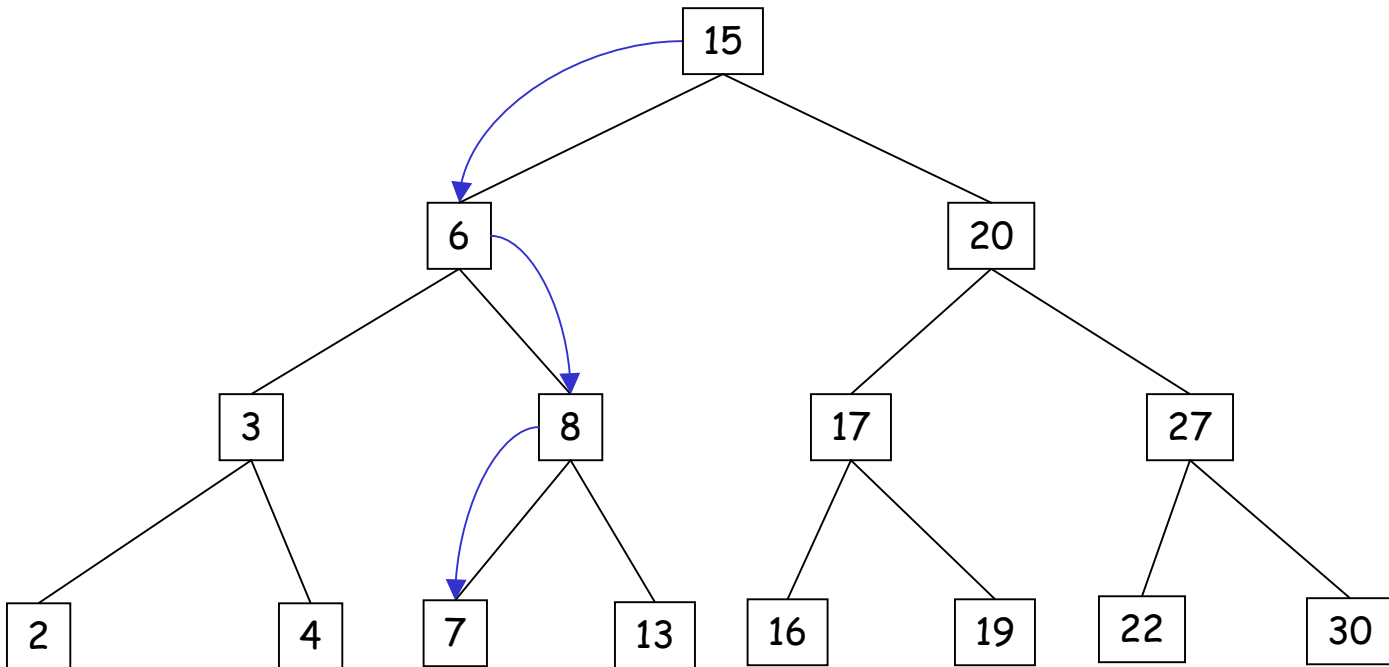
```
Iterative-Tree-search(x,k)
While (x≠NIL) e (k ≠ key[x])
    do if k < key[x]
        then x ← left[x]
        else x ← right[x]
Return x
```

Qual'è la complessità della operazione di ricerca in un ABR ?

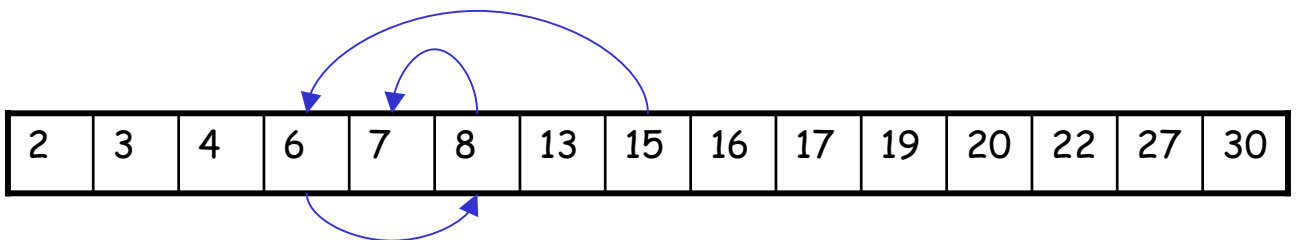
$T(n) = O(h)$       $h = \text{Altezza dell'albero}$

## Confronto con il metodo dei dimezzamenti successivi

Ipotizziamo che l'albero sia **completo** -  
Supponiamo di eseguire  $\text{tree-search}(\text{root}(t), 7)$ .



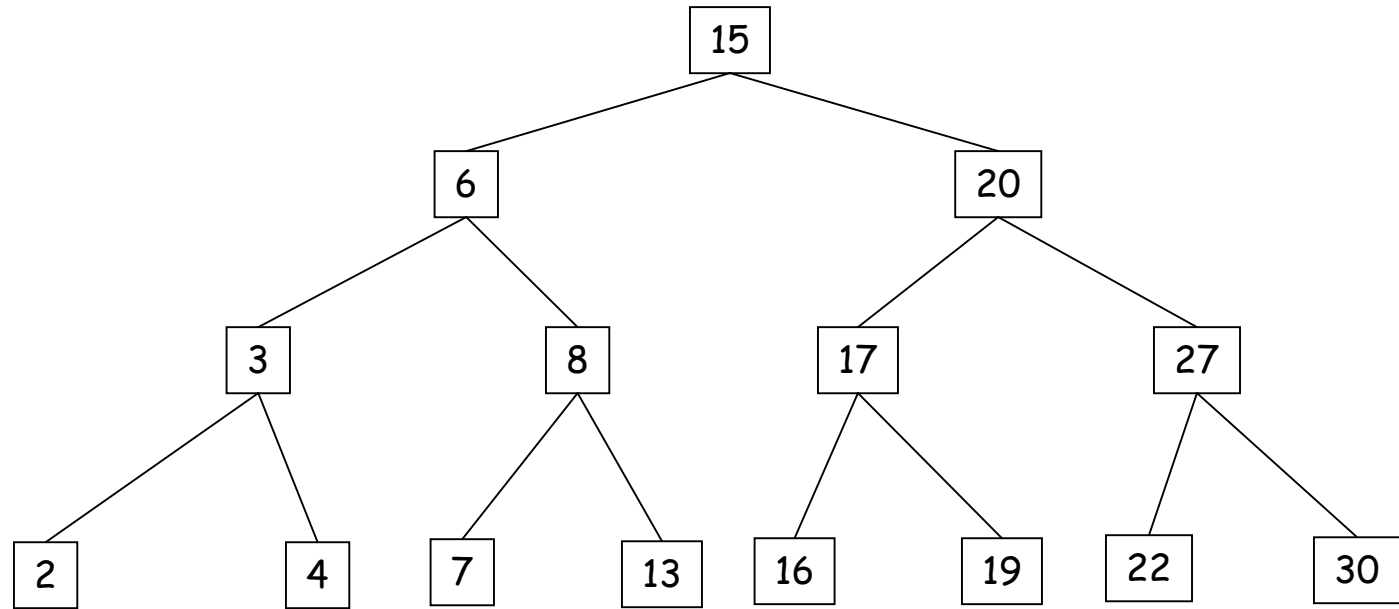
Supponiamo di avere implementato il dizionario con un array  $A$  ordinato. Eseguiamo  $\text{cerca2}(A, 7, 1, 15)$



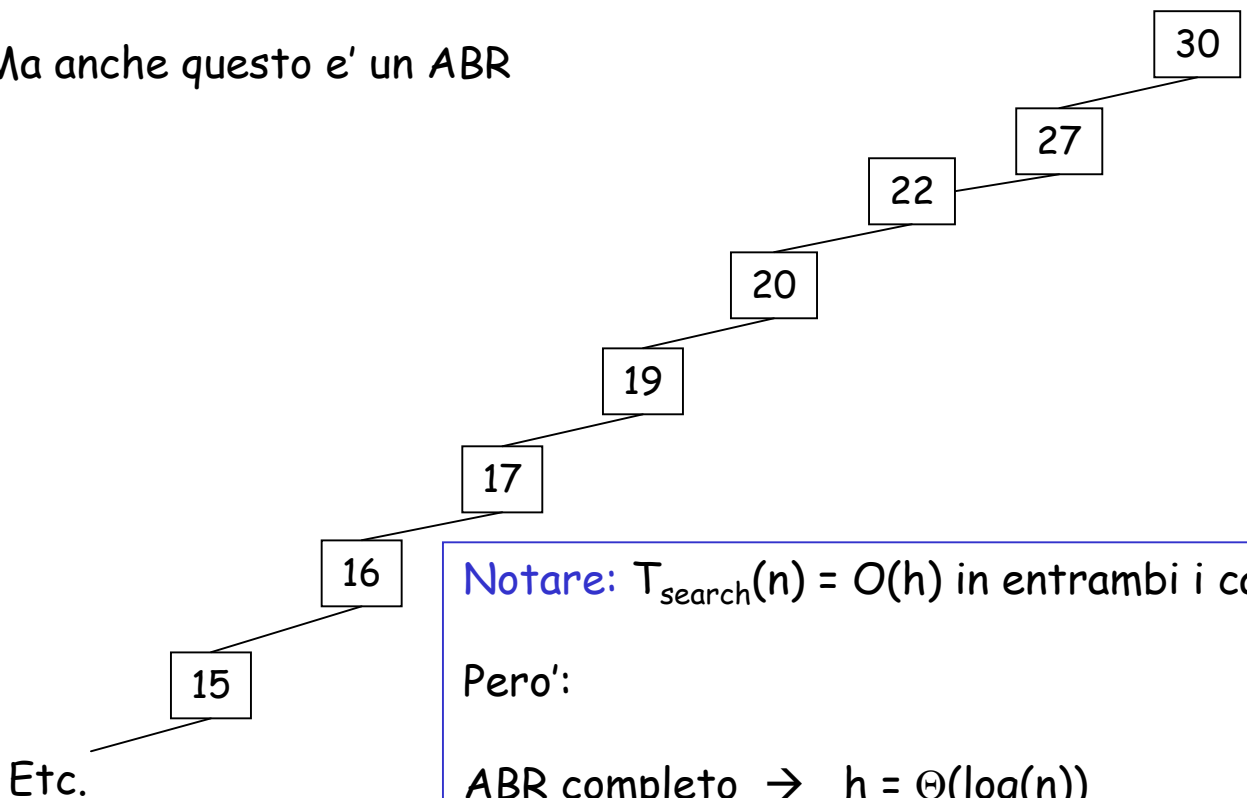
Per le proprietà dell'ABR, un elemento  $x$  (quando l'albero è completo) è l'elemento "mediano" nell'insieme ordinato composto da  $x$ , dal sottoalbero sinistro di  $x$  e dal sottoalbero destro di  $x$

## Notare ...

Non e' detto che un ABR sia completo. Dipende da come lo abbiamo costruito. Ad esempio:



Ma anche questo e' un ABR



**Notare:**  $T_{\text{search}}(n) = O(h)$  in entrambi i casi

Pero':

ABR completo  $\rightarrow h = \Theta(\log(n))$

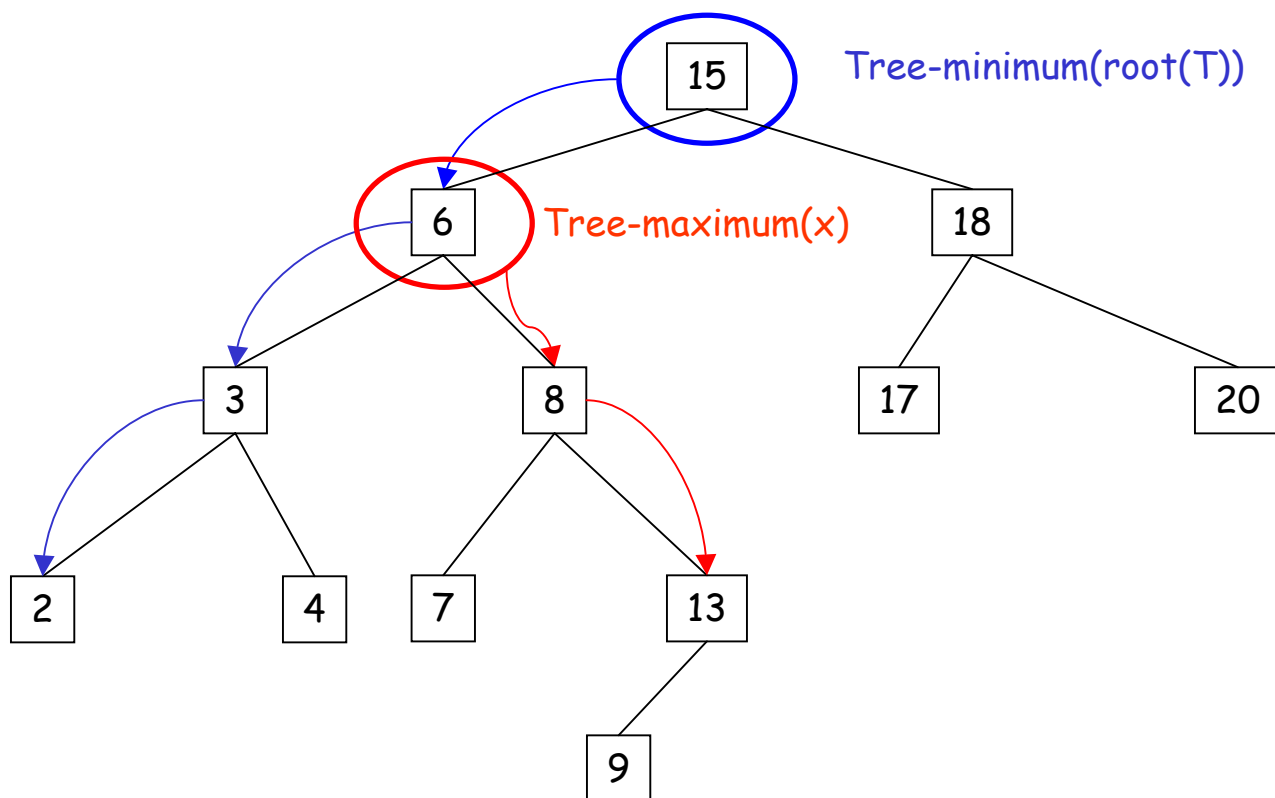
ABR "linearizzato"  $\rightarrow h = \Theta(n)$

## Minimo e Massimo

```
Tree-minimum(x)
while (left[x] ≠ NIL)
    do x ← left[x]
Return[x]
```

```
Tree-maximum(x)
while (right[x] ≠ NIL)
    do x ← right[x]
Return[x]
```

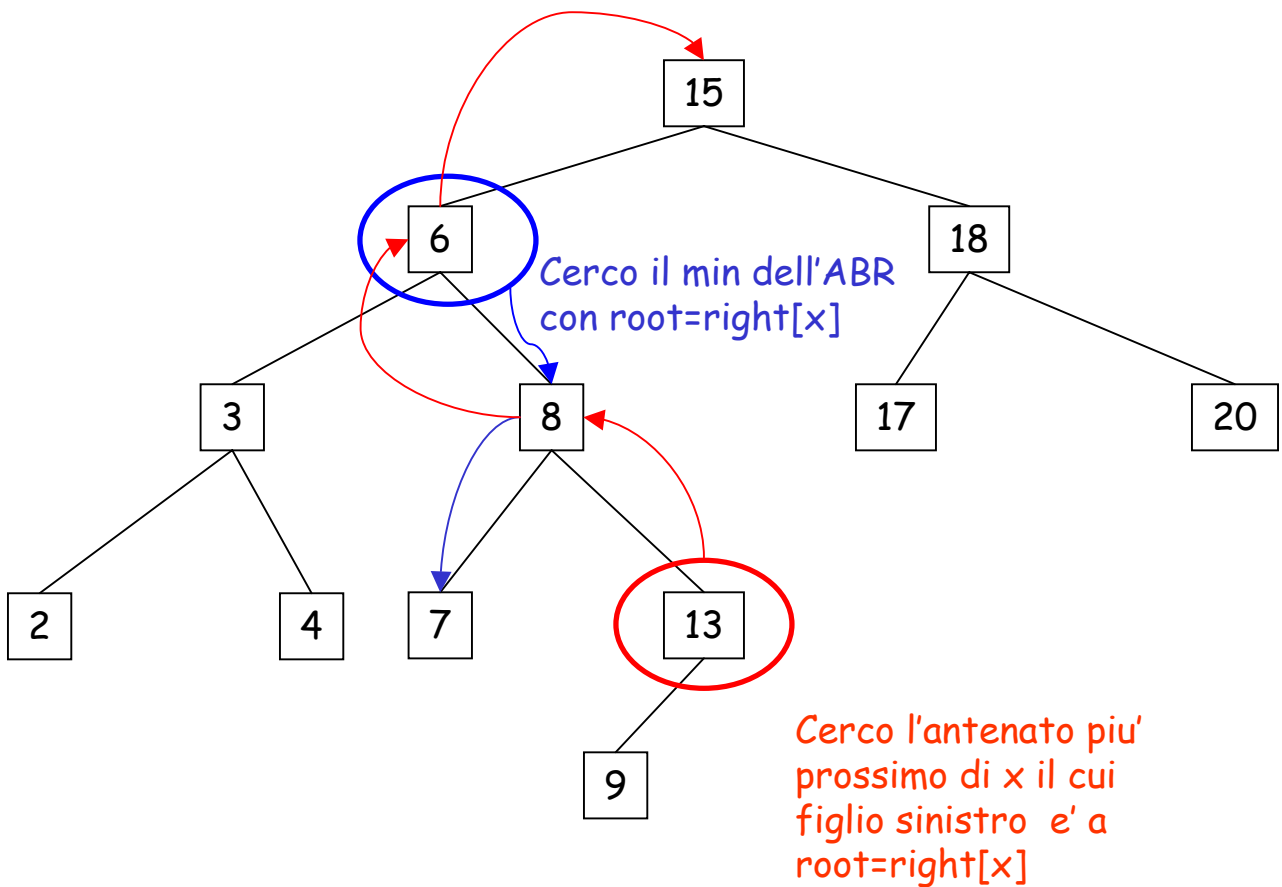
Complessita' -  $T(n) = O(h)$   
h = altezza dell'albero



## Successore e predecessore

```
Tree-successor(x)
If (right[x] ≠ NIL)
  then return Tree-Minimum(right[x])
y ← p[x]
While (y ≠ NIL) e (x=right[y])
  do x ← y
  y ← P[y]
Return y
```

$$T(n) = O(h)$$



In conclusione:

Le operazioni di interrogazione *Search*, *Minimum*, *Maximum*, *Successor* e *Predecessor* possono essere eseguite in un albero binario di ricerca di altezza  $h$  in un tempo  $O(h)$ .