

Il problema del dizionario

Oggetto

k
---	-----	-----	-----	-----

chiave

dati satellite

x = puntatore all'oggetto

Dizionario = insieme dinamico che consente di effettuare le Operazioni di ricerca, inserimento, rimozione.

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

Insert(S,x) - inserisce in S l'elemento puntato da x

Delete(S,x) - rimuove da S l'oggetto puntato da x

Proviamo a realizzare un dizionario utilizzando una struttura dati chiamata **albero binario di ricerca**.

Alberi binari di ricerca (ABR)

Nodo

k	p	left	right
---	---	------	-------

chiave

x = puntatore all'oggetto

Albero binario

insieme di oggetti (nodi) caratterizzato da quattro campi:

K[x] (chiave),

P[x] (puntatore al parente),

Left[x] (puntatore al figlio sinistro),

Right[x] (puntatore al figlio destro).

Root[T] → Puntatore alla radice dell'albero

X = puntatore nodo Left[x] = NIL (il nodo non ha figlio sinistro)
Right[x] = NIL (il nodo non ha figlio destro)

Un **albero binario di ricerca (ABR)** e' una albero binario in cui le chiavi dei vari nodi soddisfano la seguente proprieta':

PROPRIETA' dell'ABR

Sia x un nodo generico di un ABR.

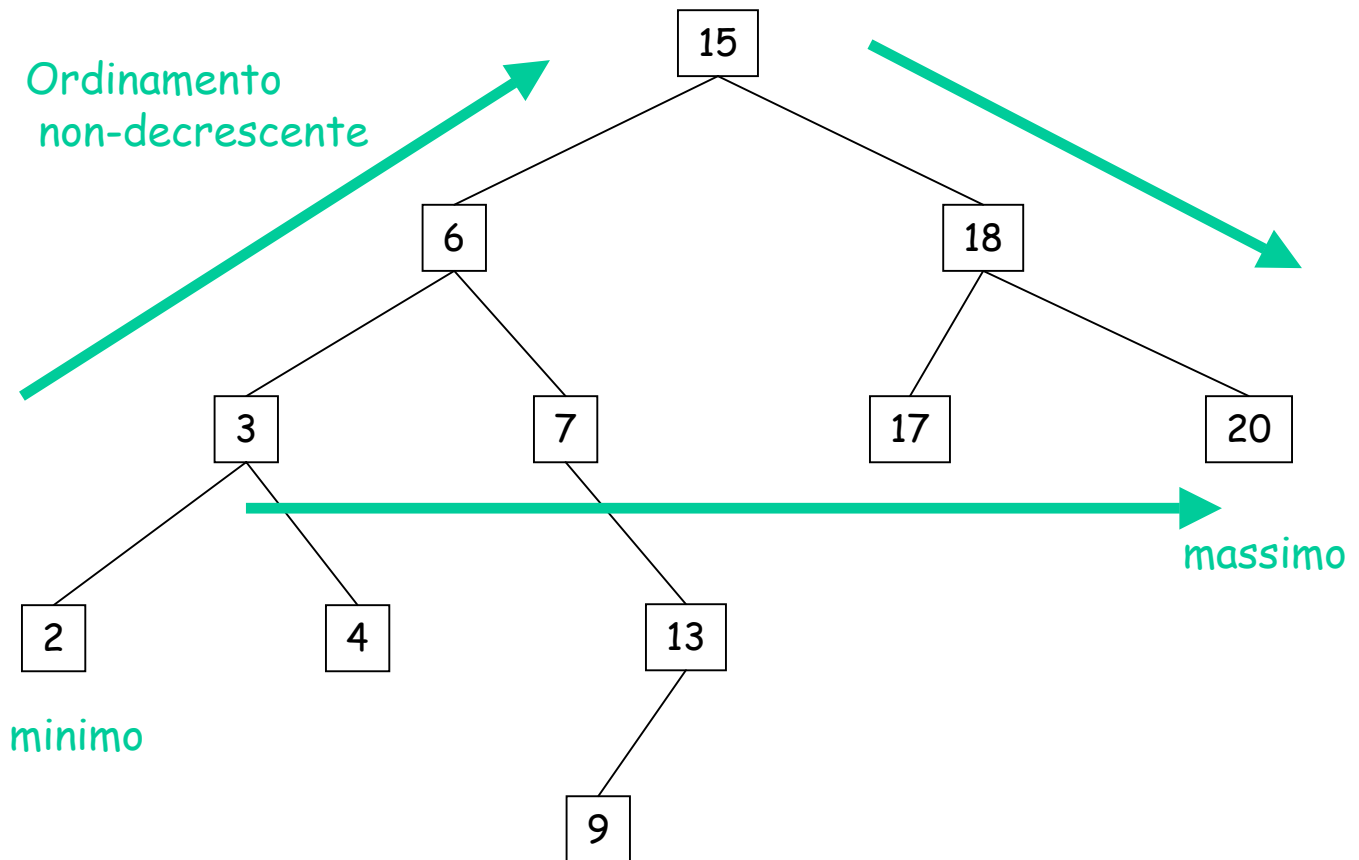
Se y e' un nodo del **sotto-albero sinistro** di x allora:

$$k[y] \leq k[x]$$

Se y e' un nodo del **sotto-albero destro** di x allora:

$$k[y] \geq k[x]$$

Alberi binari di ricerca



Un **albero binario di ricerca (ABR)** e' una albero binario in cui le chiavi dei vari nodi soddisfano la seguente proprieta':

PROPRIETA' dell'ABR

Sia x un nodo generico di un ABR.

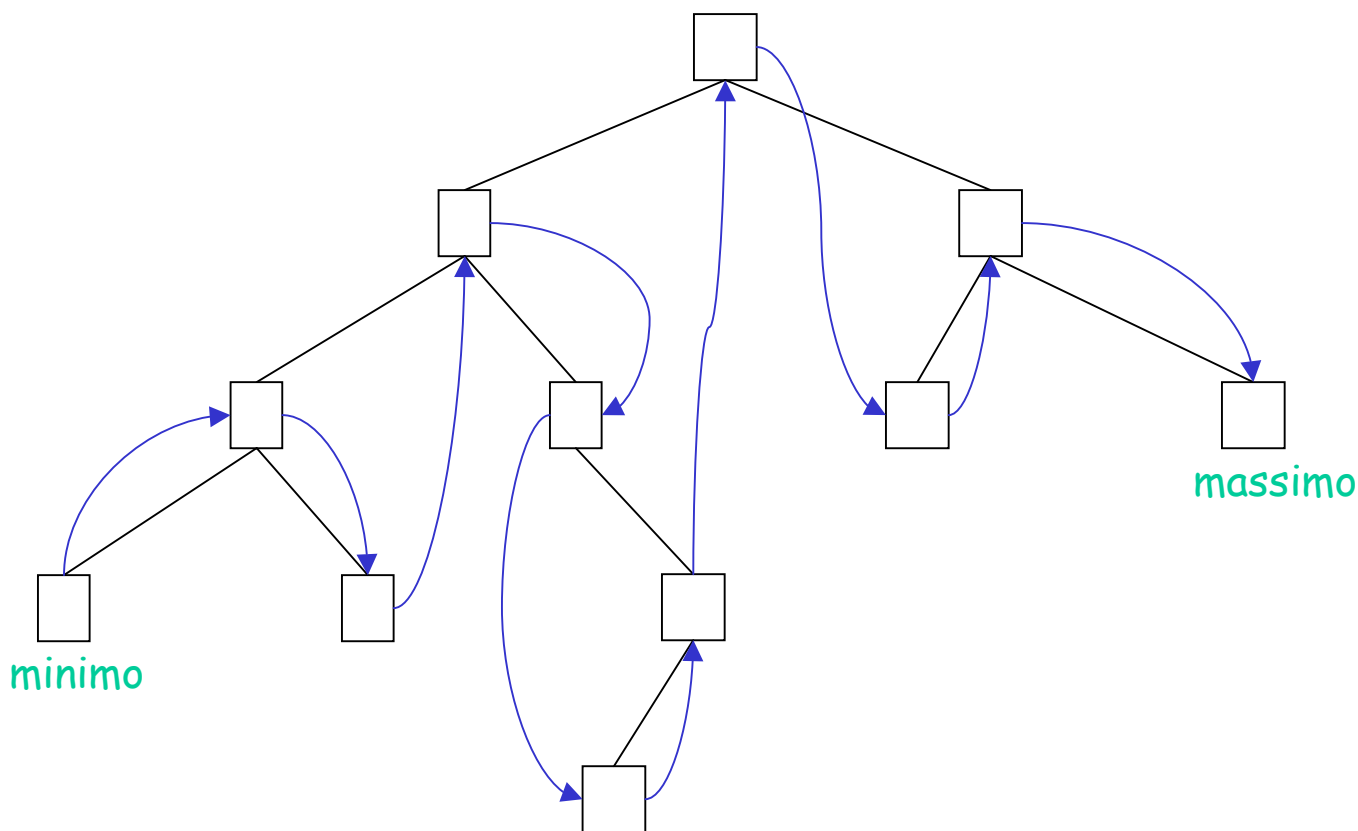
Se y e' un nodo del **sotto-albero sinistro** di x allora:

$$k[y] \leq k[x]$$

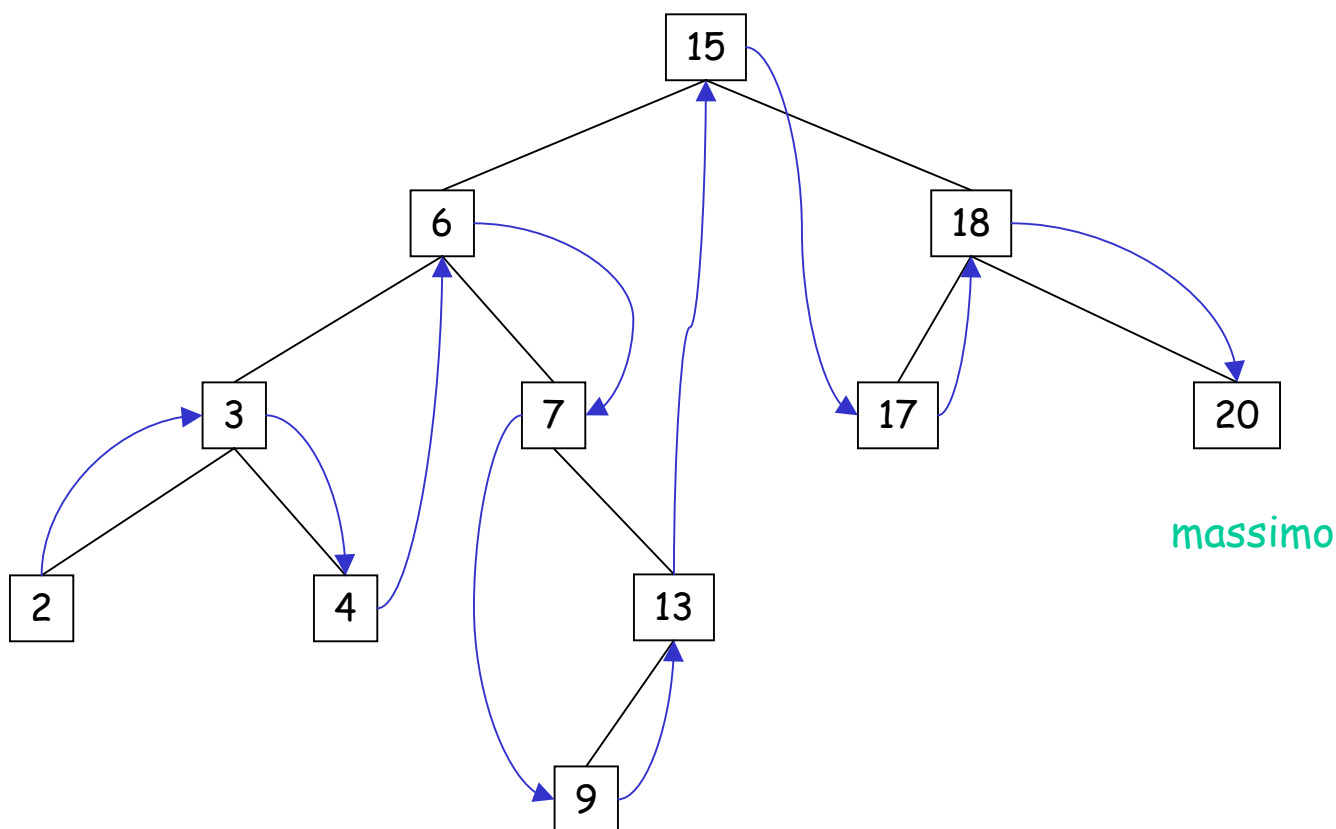
Se y e' un nodo del **sotto-albero destro** di x allora:

$$k[y] \geq k[x]$$

Le proprietà di un ABR determinano un'ordinamento totale ...



verifichiamo



Visita di un ABR

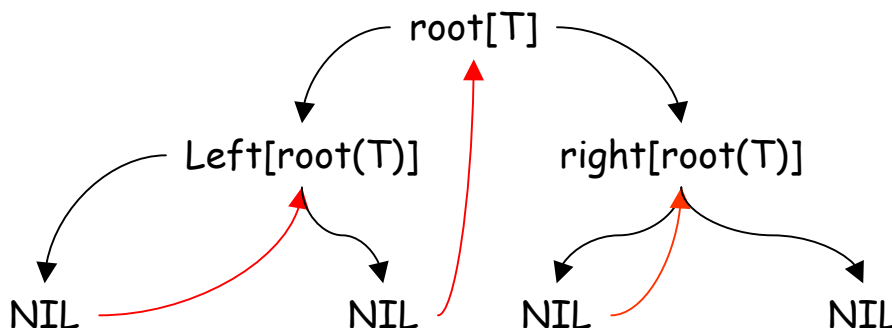
Visita in ordine simmetrico - dato un nodo x , elenco prima il sotto-albero sinistro di x , poi il nodo x , poi il sotto-albero destro.

```
Inorder-tree-walk(x)
If ( $x \neq \text{NIL}$ )
  then Inorder-tree-walk(left[x])
       stampa k[x]
       Inorder-tree-walk(right[x])
```

Verifica di correttezza - Supponiamo, per semplicità, che l'albero sia completo. Indichiamo con h l'altezza dell'albero. Vogliamo mostrare che.

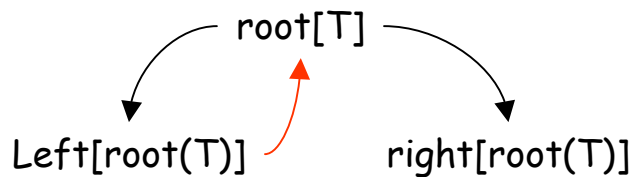
Per ogni h , la procedura $\text{Inorder-tree-walk}(\text{root}(T))$ è corretta

$h=1 \rightarrow$ Successione parametri chiamate



Verifica correttezza (continua ...)

h = generico (ipotizzo che la procedura sia corretta per $h-1$)



Albero di altezza $h-1$. Tutti i suoi elementi sono minori o uguali della radice

Albero di altezza $h-1$. Tutti i suoi elementi sono maggiori o uguali della radice

Analisi complessita'

La complessita' della procedura considerata e' $T(n) = \Theta(n)$.

Matematicamente

$$T(n) = 2 T(n/2) + c$$

Intuitivamente

La procedura e' chiamata un numero di volte pari al numero di nodi dell'albero (e ad ogni chiamata effettua un numero costante di operazioni).