

*ALBERI & RICORSIONE

(SEZ. 5 - DOMANDA 27)

Domanda 27 Realizzare una procedura `Level(T)` che dato un albero binario T , con radice $T.root$, e nodi x con campi $x.left$, $x.right$ e $x.key$, rispettivamente figlio destro, figlio sinistro e chiave intera, ritorna il numero di nodi per i quali la chiave $x.key$ è minore o uguale al livello del nodo (la radice ha livello 0, i suoi figli livello 1 e così via). Valutare la complessità.

SOLUZIONE:

→ per ciascun livello i , contare # nodi y al liv. i tali che $y.key \leq i$.

`Level(x, level):`

```
if x == nil: // foglie  
    return 0
```

`else`

```
    left = Level(x.left, level + 1)
```

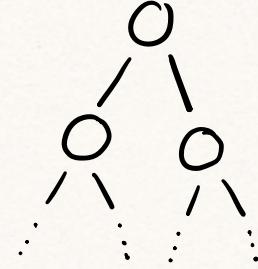
```
    right = Level(x.right, level + 1)
```

```
    if x.key <= level:
```

```
        return left + right + 1
```

```
    else
```

```
        return left + right
```



`Level(T):`

```
    return Level(T.root, 0)
```

• **COMPLESSITÀ:** $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + c \Rightarrow \Theta(n)$ (HT)

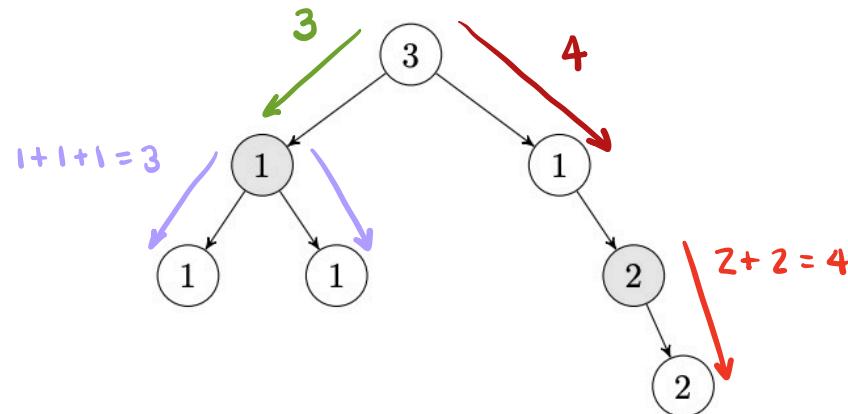
*ricorro su $\frac{1}{2}$ albero dx e $\frac{1}{2}$ albero sx
assumendo che, in media, l'albero è
bilanciato*

→ visita dell'albero: devo passare per ogni nodo 1 volta
 $\rightarrow \Theta(n)$

(SEZ. 5 - ESERCIZIO 10 FILE)

Esercizio 10 Un nodo x di un albero binario T si dice *fair* se la somma delle chiavi nel cammino che conduce dalla radice dell'albero al nodo x (escluso) coincide con la somma delle chiavi nel sottoalbero di radice x (con x incluso). Realizzare un algoritmo ricorsivo $printFair(T)$ che dato un albero T stampa tutti i suoi nodi fair. Supporre che ogni nodo abbia i campi $x.left$, $x.right$, $x.p$, $x.key$. Valutare la complessità dell'algoritmo.

Un esempio: i nodi grigi sono fair



SOLUZIONE:

```

printFair(x, path): // x modo corrente
    if x == nil: // foglie
        return 0
    left = printFair(x.left, path + x.key)
    right = printFair(x.right, path + x.key)
    sumTree = left + right + x.key
    if path == sumTree
        print x
    return sumTree

```

```

printFair(T)
    return printFair(T.root, 0)

```

• **COMPLESSITÀ:** $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + c \Rightarrow \Theta(n)$ (HT)
 ↳ visita dell'alters come sopra

(SEZ. 5 - DOMANDA 29)

Domanda 29 Scrivere una funzione $complete(T)$ che dato in input un albero binario verifica se è completo (ovvero ogni nodo interno ha due figli e tutte le foglie hanno la stessa distanza dalla radice).

SOLUZIONE:

$complete(x)$: \rightsquigarrow ritorna distanza delle foglie dal nodo corrente

```
if x == nil: // foglie
    return 0
else:
    countl = complete(x.left)
    countr = complete(x.right)
    if un sottocaso è già fallito or questo nodo fallisce ( $countr == -1$ ) or ( $countl == -1$ ) or ( $countr != countl$ ):
        return -1
    else
        return countl + 1
```

• COMPLESSITÀ: $T(n) = 2 \cdot \underbrace{T\left(\frac{n}{2}\right)}_{\text{caso "immedio" di }} + c \Rightarrow \Theta(n)$

$T(n) = T(k) + T(n-k) + c$



* HASH TABLES

(SEZ. 6 - DOMANDA 31)

Domanda 31 Si consideri una tabella hash di dimensione $m = 8$, e indirizzamento aperto con doppio hash basato sulle funzioni $h_1(k) = k \bmod m$ e $h_2(k) = 1 + k \bmod (m - 2)$. Si descriva in dettaglio come avviene l'inserimento della sequenza di chiavi: 12, 3, 22, 14, 38.

RECAP

- OPEN ADDRESSING: memorizza elementi della struttura direttamente nella tabella
- $h(K, i)$ funzione di hashing, K chiave, $i \#$ tentativo
- DOPPIO HASHING: date $h_1(K)$, $h_2(K)$
 - $h(K, i) = (h_1(K) + i \cdot h_2(K)) \bmod m$
 - inserimento nel primo spazio vuoto disponibile

SOLUZIONE:

$$m = 8 \quad h(K, i) = [(K \bmod m) + i \cdot (1 + k \bmod (m-2))] \bmod m$$

→ da inserire: 12, 3, 22, 14, 38

0	
1	
2	
3	
4	12
5	
6	
7	

$$h(12, 0) = 12 \bmod 8 + 0 \\ = 4$$

0	
1	
2	
3	3
4	12
5	
6	
7	

$$h(3, 0) = 3 \bmod 8 + 0 \\ = 3$$

0	
1	
2	
3	3
4	12
5	
6	22
7	

$$h(22, 0) = 22 \bmod 8 + 0 \\ = 6$$

0	
1	14
2	
3	3
4	12
5	
6	22
7	

$$h(14, 0) = 14 \bmod 8 + 0 \\ = 6!$$

$$h(14, 1) = (6 + 1 + 14 \bmod 6) \bmod 8 \\ = (7 + 2) \bmod 8 \\ = 1$$

0	
1	14
2	
3	3
4	12
5	
6	22
7	38

$$h(38,0) = 38 \bmod 8 + 0 \\ = 6 !$$

$$h(38,1) = (6 + 1 + 38 \bmod 6) \bmod 8 \\ = (7 + 2) \bmod 8 \\ = 1 !$$

$$h(38,2) = [6 + 2 \cdot (3)] \bmod 8 \\ = 4 !$$

$$h(38,3) = [6 + 3 \cdot 3] \bmod 8 \\ = 15 \bmod 8 \\ = 7$$

(SEZ. 6 - DOMANDA 34)

Domanda 34 Si consideri una tabella hash di dimensione $m = 8$, gestita mediante chaining (liste di trabocco) con funzione di hash $h(k) = k \bmod m$. Si descriva in dettaglio come avviene l'inserimento della sequenza di chiavi: 14, 10, 22, 18, 19.

• CHAINING: $T[j]$ è lista di elementi con la stessa chiave

SOLUZIONE:

$$m = 8 \quad h(k) = k \bmod m \quad \Rightarrow \text{da inserire: } 14, 10, 22, 18, 19$$

0	
1	
2	
3	
4	
5	
6	
7	

$\rightarrow 10$

$$14 \bmod 8 = 6$$

$$10 \bmod 8 = 2$$

$\rightarrow 14$

0	
1	
2	
3	
4	
5	
6	
7	

$\rightarrow 18$

$\rightarrow 19$

$\rightarrow 22 \rightarrow 14$

$$22 \bmod 8 = 6$$

$$18 \bmod 8 = 2$$

$$19 \bmod 8 = 3$$