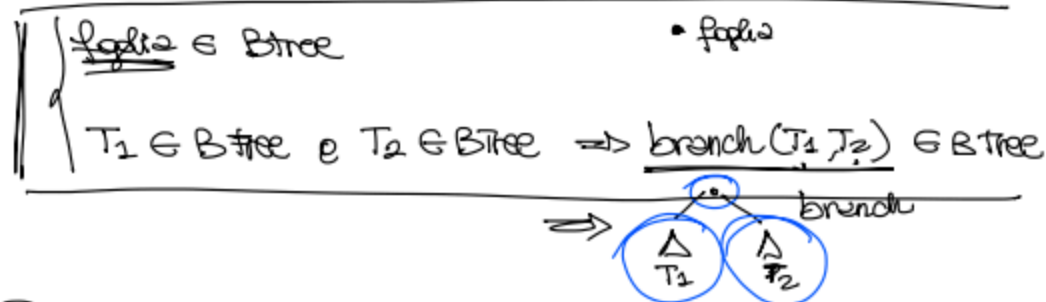


INDUZIONE STRUTTURALE

Definiamo BTree come



Definiamo $m : \text{BTree} \rightarrow \mathbb{N}$ conta i nodi di un albero binario

$$\begin{cases} m(\text{folia}) = 1 \\ m(\text{branch}(T_1, T_2)) = 1 + m(T_1) + m(T_2) \end{cases}$$

Definiamo $h : \text{BTree} \rightarrow \mathbb{N}$ calcola l'altezza dell'albero

$$\begin{cases} h(\text{folia}) = 0 \\ h(\text{branch}(T_1, T_2)) = 1 + \max(h(T_1), h(T_2)) \end{cases}$$

calcola il valore massimo tra due valori

Dimostrare per induzione ~~strutturale~~ strutturale che

$$(*) \quad \forall T \in \text{BTree}. m(T) \leq 2^{h(T)+1} - 1$$



Base: Dobbiamo dimostrare (*) per gli elementi folia

$$\begin{aligned} T = \text{folia} \quad m(T) &= m(\text{folia}) = 1 \\ h(T) &= h(\text{folia}) = 0 \end{aligned}$$

$$\text{Calcoliamo } 2^{h(T)+1} - 1 = 2^{0+1} - 1 = 1$$

$$m(T) = 2^{h(T)+1} - 1 \quad \checkmark$$

Poss. induttivo : $T = \text{branch}(T_1, T_2)$

Supponiamo per Hp. ind. che

$$\text{Hp. ind.} \rightarrow \begin{cases} \textcircled{*} m(T_1) \leq 2^{h(T_1)+1} - 1 \\ \textcircled{*} m(T_2) \leq 2^{h(T_2)+1} - 1 \end{cases}$$

dimostriamo che $m(T) \leq 2^{h(T)+1} - 1$

$$m(T) = 1 + m(T_1) + m(T_2) \quad \boxed{h(T)} = \textcircled{1} + \max(h(T_1), h(T_2))$$

$$m(T) = 1 + m(T_1) + m(T_2) \leq 1 + 2^{h(T_1)+1} - 1 + 2^{h(T_2)+1} - 1$$

per def di m

per $\textcircled{*}$

$$\text{Hp. ind.} \rightarrow 2(2^h) = 2^{h+1}$$

$$\leq 2^h + 2^h - 1$$

$$= 2^{h+1} - 1$$

$$= 2^{h(T)+1} - 1$$

$$h \geq h(T_1) + 1$$

$$h \geq h(T_2) + 1$$

BTree, definiamo

$\ell : \text{BTree} \rightarrow \mathbb{N}$ conta le foglie

$$\ell(\text{foglia}) = 1$$

$$\ell(\text{branch}(T_1, T_2)) = \ell(T_1) + \ell(T_2)$$

Dimostrare che $\ell(T) \leq 2^{h(T)} \quad \forall T \in \text{BTree}$

Base: $T = \text{foglia}$: $e(T) = 1$ $1 \leq 2^0 = 1$ ✓
 $h(T) = 0$

Passo Induttivo $T = \text{branch} (T_1, T_2)$

Hp. ind.: $e(T_1) \leq 2^{h(T_1)}$ $e(T_2) \leq 2^{h(T_2)}$

Dobbiamo dimostrare che $e(T) \leq 2^{h(T)}$

$$e(T) = e(T_1) + e(T_2) \quad h(T) = 1 + \max(h(T_1), h(T_2))$$

$$e(T) = e(T_1) + e(T_2) \leq 2^{h(T_1)} + 2^{h(T_2)}$$

per def di e

Hp. ind.

$$h = \max(h(T_1), h(T_2))$$

\Rightarrow

$$h(T_1) \leq h$$

$$h(T_2) \leq h$$

$$\leq 2^h + 2^h = 2(2^h) = 2^{h+1} = 2^{h(T)}$$

Exp (espressioni) definite induttivamente
 aritmetiche

$\begin{cases} m \in \text{Exp} \\ e_1, e_2 \in \text{Exp} \end{cases}$ def. di exp
 $e_1 + e_2 \in \text{Exp} \quad e_1 - e_2 \in \text{Exp}$

Def. val: $\text{Exp} \rightarrow \mathbb{N}$ numero di numeri m presenti in una exp

$$\begin{cases} \text{val}(m) = 1 \\ \text{val}(e_1 + e_2) = \text{val}(e_1 - e_2) = \text{val}(e_1) + \text{val}(e_2) \end{cases}$$

Def op: $\text{Exp} \rightarrow \mathbb{N}$ numero di operatori in exp $(+, -)$

$$\begin{cases} \text{op}(m) = 0 \\ \text{op}(e_1 + e_2) = \text{op}(e_1 - e_2) = \text{op}(e_1) + \text{op}(e_2) + 1 \end{cases}$$

Dimostrare che $\text{val}(e) = \text{op}(e) + 1 \quad \forall e \in \text{Exp}$

Base: $e = m$ $\text{val}(e) = \text{val}(m) = 1$
 $\text{op}(e) = \text{op}(m) = 0$ $1 = 0 + 1$ ✓

Passo ind: $e_1 + e_2$ ($e_1 - e_2$ analogo)
 perché la definizione di
 op e val non dipende
 dal tipo di operatore

Hp. ind $\text{val}(e_1) = \text{op}(e_1) + 1$
 $\text{val}(e_2) = \text{op}(e_2) + 1$

Dobbiamo dim. che $\text{val}(e) = \text{op}(e) + 1$

$$\begin{aligned} \text{val}(e) &= \text{val}(e_1) + \text{val}(e_2) = \text{op}(e_1) + 1 + \text{op}(e_2) + 1 \\ &\quad \uparrow \text{ per def di val} \qquad \text{Hp. ind.} \\ &= \boxed{\text{op}(e_1) + \text{op}(e_2) + 1} + 1 = \text{op}(e) + 1 \\ &\quad \text{op}(e) \\ &\quad \text{per def di op} \end{aligned}$$

```

{ int z=1;
  int y=2;
  void A() { int z = x+y; }
  void B() { int x=2;
             int w=4;
             void C() { int x = y+w;
                        A();
                        y = z+y;

```

main

$\begin{bmatrix} x \\ y \end{bmatrix}$	$A \begin{bmatrix} z \end{bmatrix}$	$\text{Sd}(\text{main}) = 0$
$\begin{bmatrix} x \\ w \end{bmatrix}$	$B \begin{bmatrix} z \\ w \end{bmatrix}$	$\text{Sd}(A) = \text{Sd}(B) = 1$
$\begin{bmatrix} x \end{bmatrix}$	$C \begin{bmatrix} x \end{bmatrix}$	$\text{Sd}(C) = 2$

B();
{

main $\rightarrow B \rightarrow C \rightarrow A$

Scoping statico

main

x	1
y	2

Chiamata main $\rightarrow B$

main

x	1
y	2

link
dinom.

x	2
w	4

chiamante chiamato

$$K_B = \text{Sd}(\text{main}) - \text{Sd}(B) + 1$$
$$= 0 - 1 + 1 = 0$$

Link Statico

$$CS(B) = \text{main}$$

Esecuzione di B

base = 2

$$y = x + y$$

la più vicina nella colonna di cui l'elemento statico che def. y

$$N_y = -\text{Sd}(\text{main}) + \text{Sd}(B)$$
$$\Rightarrow y_{\text{mem}} = 2$$
$$= 0 + 1 = 1$$

uso y

nel main

$$y = 2 + 2 = 4$$

Dobbiamo risalire di 1 la colonna statica per trovare il corretto riferimento a y

Chiamata B $\rightarrow C$

main

x	2
y	4

B

x	2
w	4

C

z	8
---	---

Link statico:

$$K_C = \text{Sd}(B) - \text{Sd}(C) + 1$$
$$= 1 - 2 + 1 = 0$$

$$CS(C) = B$$

base

$$z = y + w$$

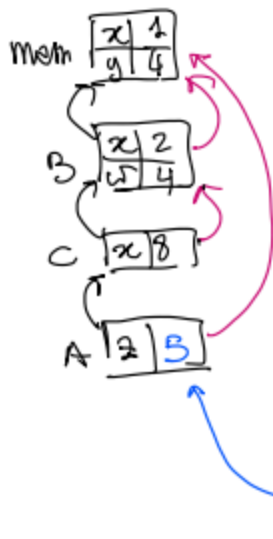
8

$$N_w = \text{Sd}(C) - \text{Sd}(B)$$
$$= 1 - 1 = 0 \text{ in } B \rightarrow 4$$

main

$$N_y = \text{Sd}(C) - \text{Sd}(\text{main}) = 2 \text{ in main}$$

Chiamata $C \rightarrow A$



Calcolo storico

$$V_A = \text{Sol}(C) - \text{Sol}(A) + 1$$

$$= 2$$

$$\text{CS}(A) = \text{CS}(\text{CS}(C)) =$$

$$= \text{CS}(B) = \text{main}$$

$$N_y = N_x = \text{Sol}(A) - \text{Sol}(\text{main})$$

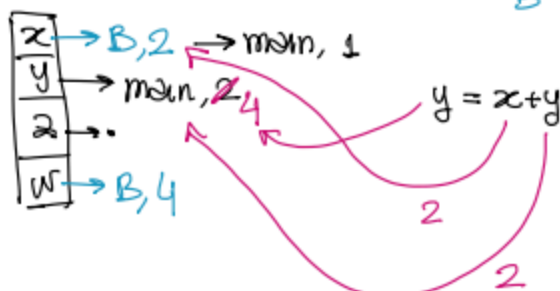
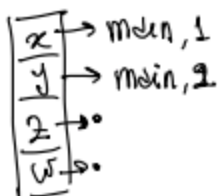
$$= 1 \rightarrow \text{main}$$

Scoping dinamico

CRT

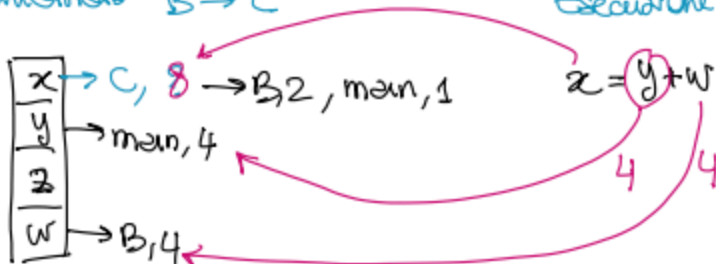
Chiamata $\text{main} \rightarrow B$

Esecuzione B

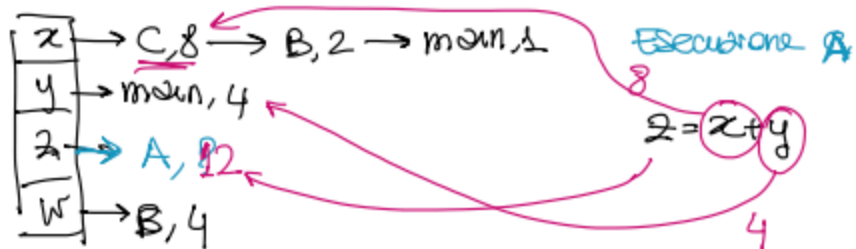


Chiamata $B \rightarrow C$

Esecuzione C



Chiamata $C \rightarrow A$



Ricorsione

lista function (lista list) {

if (length(list) == 0) then return list = ()

else return concat (function (cdr (list)), car (list))

}

function(3, 6, 9) \rightarrow (9, 6, 3)

\rightarrow concat (function(6, 9), (3))

\rightarrow concat (function(9), 6)

\rightarrow concat (function(), 9)

La funzione inverte gli elementi della lista

lista functionrc (lista list, lista res) {

if length(list) == 0 return res;

else return functionrc (cdr (list), concat (car (list), res))

}

lista function (lista list) { return functionrc (list, ()); }

function (3, 6, 9) ← (9, 6, 3)

↳ functionrc (369, ())

↳ functionrc (69, concat(3, ()))
(3)

↳ functionrc (9, concat(6, (3)))
(63)

↳ functionrc (, concat(9, (63)))
(9, 6, 3)