PRIMA ESERCITAZIONE:

INDUZIONE STRUTTURALE

Si deve dim. una Prop. su un Insieme definito Induttivamente

(1) **Definiamo** RB-Tree come:
$$\begin{cases} \text{foglia} \in BTree \\ T_1 \in RBTree \text{ e } T_2 \in Tree \Rightarrow \underset{\substack{\wedge \\ Btre}}{Branch(T_1, T_2)} \end{cases}$$

Definiamo:

$n \to BTree \to \mathbb{N}$ che conta #Nodi Nell' Albero

$$n(\text{foglia}) = 1 \quad , \quad n(Branch(T_1, T_2)) = 1 + n(T_1) + n(T_2)$$

Definiamo: $h : BTree \to \mathbb{N}$ Calcola altezza (max Cammino)

$$\begin{cases} h(\text{foglia}) = 0 \\ h(Branch(T_1, T_2)) = 1 + \underset{\text{Massimo}}{\max}(h(T_1), h(T_2)) \end{cases}$$

DIMOSTRARE CHE $\forall T \in BTree : n(t) \leq 2^{h(t)+1} - 1$ (*)

- Base, dimostro per le foglie che Vale (*)

$$T = \text{foglia} \Rightarrow \quad \begin{aligned} n(t) &= 1 \\ h(t) &= 0 \end{aligned}$$

Calcolo (*) per foglia $\Rightarrow 2^{0+1} - 1 = 1$

ed Infatti è Vero visto che $\underset{1}{\underset{\shortparallel}{n(t)}} \leq 1$

~~M~~ Passo, Aggiungo il Branch e suppongo I.I. che $n(t_1) \leq 2^{h(t_1)+1} - 1$
e $n(t_2) \leq 2^{h(t_2)+1} - 1$

$T = Branch(T_1, T_2)$

Calcolo $n(t) = 1 + n(t_1) + n(t_2)$

$h(t) = 1 + \underbrace{\max(h(t_1), h(t_2))}_{\Rightarrow h}$

Applico I.I.

$n(t) = 1 + n(T_2) + n(T_1) \leq \cancel{1} + 2^{h(T_1)+1} - 1 + 2^{h(T_2)+1} \cancel{-1}$

Per DEFINIZIONE so che: $h \geq h(T_1) + 1$ ed Allora Posso Sostituire
$h \geq h(T_2) + 1$

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

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

Trucco è Stato fare la SOSTITUZIONE h

Sullo stesso Insieme definiamo:

$L : BTree \to N$ che conta le foglie

$$\begin{cases} L(foglia) = 1 \\ L(Branch(T_1, T_2)) = L(T_1) + L(T_2) \end{cases}$$

Va dimostrato che $L(T) \leq 2^{h(T)}$ $\forall T \in BTree$

- BASE $\Rightarrow$ $T = foglia$ e $L(T) = 1$ e $h(T) = 0$

  Trovo che $1 \leq 2^0 = 1$ che è VERO

- PASSO $\Rightarrow$ I.I. $L(T_1) \leq 2^{h(T_1)}$ e $L(T_2) \leq 2^{h(T_2)}$

con $T = Branch(T_1, T_2)$ devo dimostrare $L(T) \leq 2^{h(t)}$

$$L(T) = L(T_1) + L(T_2) \underset{I.I}{\leq} 2^{h(T_1)} + 2^{h(T_2)}$$

Prendo $h = \max(h(T_1), h(T_2))$ $\Rightarrow$ $h(T_1) \leq h$ e $h(T_2) \leq h$

$$\leq 2^h + 2^h = 2^{h+1}$$

è proprio UGUALE ad $2^{h(t)}$

Visto che la def era: $h(t) = 1 + h$

Insieme EXPR Aritmetiche definite Induttivamente

$$\begin{cases} e_1, e_2 \in EXPR . \quad e_1 + e_2 \in EXPR \\ \qquad\qquad\qquad\quad\; e_1 - e_2 \in EXPR \\ \\ n \in EXPR \end{cases}$$

Def. $Val: EXPR \rightarrow \mathbb{N}$ Numero di Numeri n Presenti in una e

$$\begin{cases} Val(n) = 1 \\ Val(e_1 + e_2) = Val(e_1) + Val(e_2) = Val(e_1 - e_2) \end{cases}$$

Def. $OP: Exp \rightarrow \mathbb{N}$ #operatori

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

DIMOSTRARE che $Val(e) = op(e) + 1 \quad \forall e \in EXP$

BASE $\Rightarrow$ $e = n$ e Quindi

$$\begin{cases} Val(e) = Val(n) = 1 \\ Op(e) = Op(n) = 0 \\ 1 = 0 + 1 \quad \text{ed è Verificato} \end{cases}$$

PASSO $\Rightarrow$ $e = e_1 + e_2$

I.I. $\Rightarrow$ $Val(e_1) = Op(e_1) + 1$

$Val(e_2) = Op(e_2) + 1$

Per def. $Val(e) = Val(e_1) + Val(e_2) \underset{I.I}{=} Op(e_1) + 1 + Op(e_2)$

$$= \underbrace{Op(e_1) + Op(e_2) + 1}_{op(e)} + 1$$

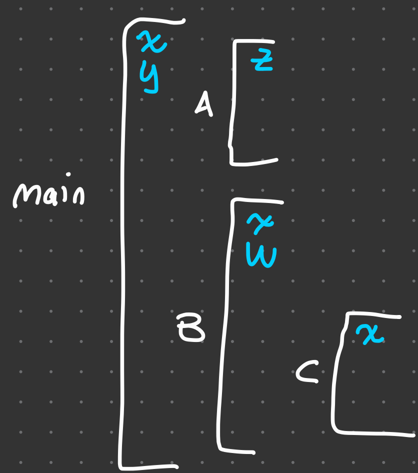
dalla definizione di OP

$$= Op(e) + 1$$

CATENA STATICA:

Viene fornito il Codice, dice Cosa Viene Calcolato. Si Costruisca la Catena Statica (Calcolando K) e Anche la RISOLUZIONE dei RIFERIMENTI NON Locali. In Caso DINAMICO Mostrare l'EVOLUZIONE della CRT.

```
int x=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=x+y;
           C();
           }
B();}
```



(1)

$Sd(Main) = 0$

$Sd(A) = Sd(B) = 1$

$Sd(C) = 2$

$Main \rightarrow B \rightarrow C \rightarrow A$

(2) SCOPING STATICO

$x = 1$
$y = 2$

MAIN Chiama B

$x = 1$
$y = 2$

link dinamico

x	2
w	4

link Statico

$K_B = Sd(Main) - Sd(B) + 1 = 0$

$CS(B) = main$

Eseguendo B ho: $x = w + y$

locale = 2

devo Calcolare N_y (di Quanto devo Risalire Catena Statica)

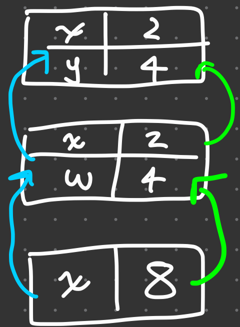
Più Vicina che definisce y

$N_y = -Sd(Main) + Sd(B) = 1$

Chi la USA

Risalgo di 1 Per Trovare y Corretta [MAIN]

Chiamata di C



$$K_C = Sd(B) - Sd(C) + 1 = 0$$

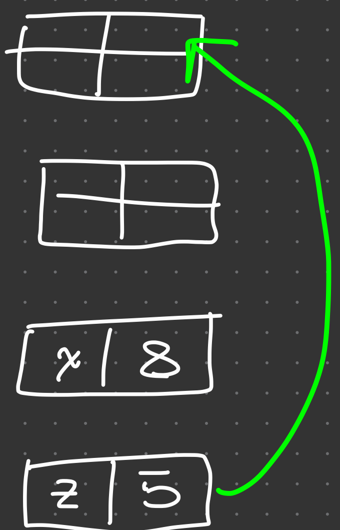
$$CS(C) = B$$

$$x = y + w = 8$$

$N_w = Sd(C) - Sd(B) = 1$ (Blocco Sopra) $\longrightarrow 4$

$N_y = Sd(C) - Sd(A) =$ Main $\longrightarrow 4$

Chiamata di A:



$$K_A = Sd(C) - Sd(A) + 1 = 2$$

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

$z = x + y$ Sono nel MAIN

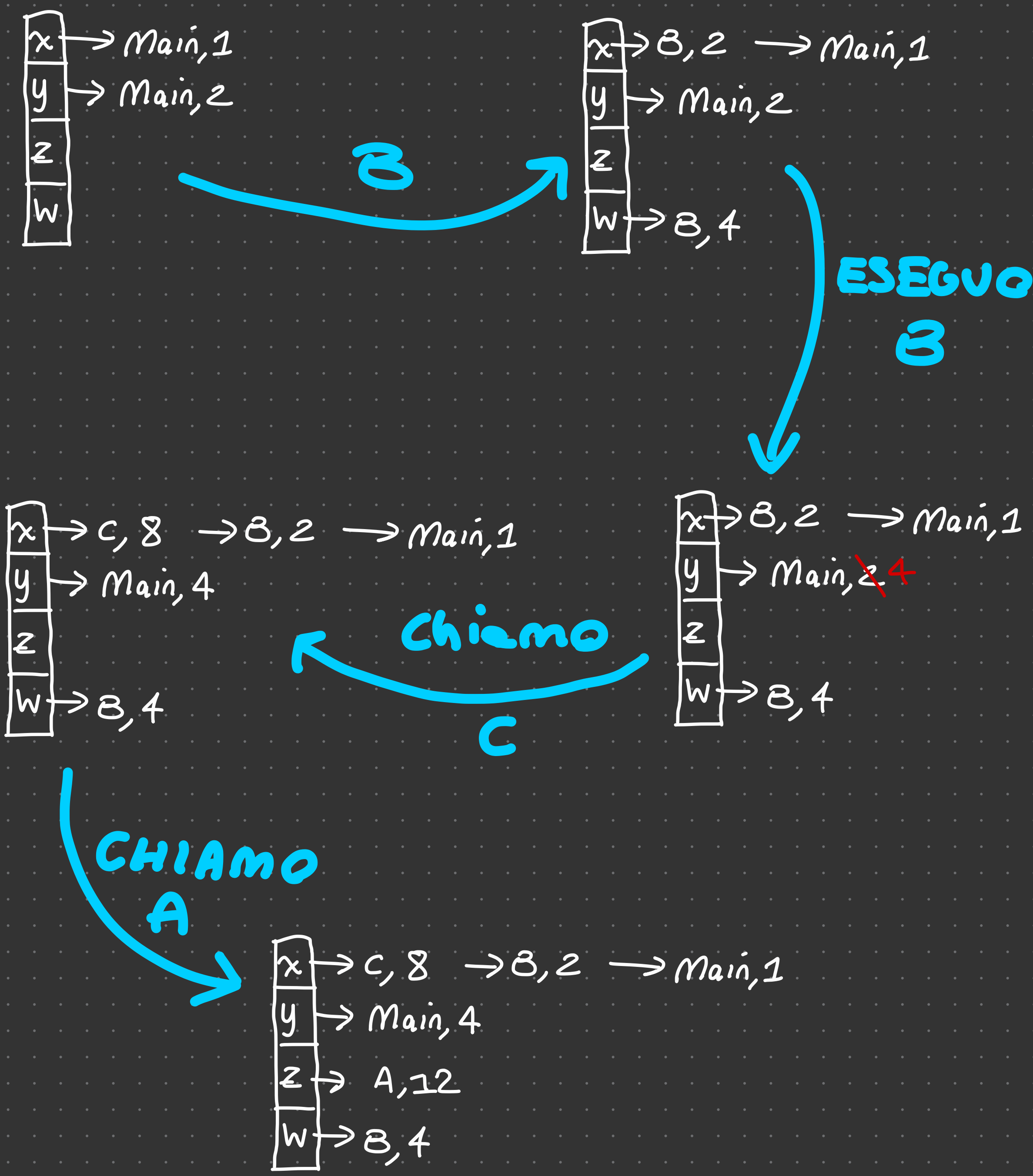
locale

$N_x = N_y = Sd(A) - Sd(MAIN) = 1$

$= 1 + 4 = 5$

Scoping Dinamico

Uso CRT per descrivere



Se fz. Terminassero Prima <u>NON</u> ricopiarle ed <u>Eliminare</u> i Valori che <u>NON</u> Sono più Attivi

RICORSIONE:

Sapere Concetti ⇒ Ricorsione e Ricorsione Di Coda

Fare Simulazione Su Esempio e Trasformarlo in Coda

```
Lista function (lista list){
    if(len(list) ==0) Then return list
    else
      return concat(function(cdr(list)), car(list))
}
```

Eseguo su [3,6,9] ← (9,6,3)

↳ concat(function([6,9]), 3) ← (9,6)

↳ concat(function(9,(6))) ← (9)

↳ concat(function(), 9)

↓

()

La funzione fa il REVERSE della LISTA

Trasformo RICORSIONE IN CODA

```
lista function (Lista list){ function RC ((list, ())}

lista functionRC (lista list, lista res){
    if (len(list) == 0)
        return res;
    else
        return functionRC (cdr(list), concat (car(list), res)
```

function(3,6,9)
↳ functionRC((3,6,9), ())
↳ functionRC((6,9), (3))
↳ functionRC((9) , (6,3))
↳ functionRC((), (9,6,3))

lista RISULTANTE