

E.s.

Convertiamo il seguente algoritmo in versione iterativa

L'algoritmo elimina dall'albero + tutti i nodi con valore compreso  
tra a e b

CANCINT ( $T, a, b$ )

IF  $T \neq \text{NIL}$  THEN

IF  $T \rightarrow \text{key} < a$  THEN

$T \rightarrow dx = \text{CANCINT}(T \rightarrow dx, a, b)$

IF  $T \rightarrow \text{key} > b$  THEN

$T \rightarrow sx = \text{CANCINT}(T \rightarrow sx, a, b)$

ELSE

$T \rightarrow sx = \text{CANCINT}(T \rightarrow sx, a, b)$

$T \rightarrow dx = \text{CANCINT}(T \rightarrow dx, a, b)$

$T = \text{CANCROOT}(T)$

Un'eliminazione post order è la soluzione più efficiente perché se cancelliamo prima la radice sostituendola con il minimo o l'unico potenzialmente potremo dover ricancellare la radice e così via.

RETURN T

CANCINT ( $T, a, b$ )

$CT = T$

$STT = NIL$

WHILE ( $CT \neq NIL \text{ || } STT \neq NIL$ ) DO

IF  $CT \neq NIL$  THEN

$STT = PUSH (STT, CT)$

IF  $CT \rightarrow key < a$  THEN

$CT = CT \rightarrow dx$

ELSE

$CT = CT \rightarrow sx$

ELSE

$CT = TOP (STT)$

IF  $CT \rightarrow key < a$  THEN

$CT \rightarrow dx = RET$

$RET = CT$

$LAST = CT$

$STT = POP (STT)$

$CT = NIL$

ELSE IF  $CT \rightarrow key > b$  THEN

In tutti e 3 i casi effettuiamo una chiamata quindi solviamo il contesto

Solviamo la chiamata su  $T \rightarrow dx$

Sia nel second if che nel 3° il primo comando è lo stesso

Discriminiamo i casi: i casi sono esclusivi quindi per capire da quale tamponio basta fare di nuovo gli stessi controlli di prima

$CT \rightarrow s_x = RET$

$RET = CT$

$LAST = CT$

$STT = POP(STT)$

$CT = NIL$

ELSE

IF  $CT \rightarrow d_x \neq LAST \ \& \& \ CT \rightarrow d_x \neq NIL$  THEN

$CT \rightarrow s_x = RET$

$CT = CT \rightarrow d_x$

ELSE IF  $CT \rightarrow d_x \neq NIL$  THEN

utomo della 2<sup>a</sup> chiamata

$CT \rightarrow d_x = RET$

$LAST = CT$

$CT = CANCROOT(CT)$

CANCROOT potrebbe cambiare CT  
e invalidare LAST

$RET = CT$

$STT = POP(STT)$

$CT = NIL$

ELSE

$CT \rightarrow s_x = RET$

$LAST = CT$

$CT = CANCROOT (\alpha)$

$RET = CT$

$STT = \text{POP} (S\bar{T})$

$CT = NIL$

$\text{RETURN } CT$

## ALGORITMI RICORSIVI IN CODA

Vediamo un algoritmo ricorsivo per il fattoriale che sia ottimale

•  $FATT (m)$

Return  $FATTRC (m, 1)$

•  $FATTRC (m, r)$

IF  $m = 1$  THEN

    return  $r$

ELSE

    return  $FATTRC (m-1, r \cdot m)$

In questo algoritmo calcoliamo il fattoriale "scendendo" da  $m$  a 1  
e **NON** necessita di alcun salvataggio del contesto perché non

non vengono mai letti i valori dopo le chiamate, in particolare la chiave sta nel fatto che dopo la chiamata non vengono effettuate operazioni di lettura e scrittura con i parametri passati.

Questa è la caratteristica principale degli algoritmi **RICORSIVI IN CODA**

### RICERCA (RICORSIVA IN CODA)

RICERCA ( $T, k$ )

IF  $T \neq \text{NIL}$  THEN

IF  $T \rightarrow \text{Key} < k$  THEN

$\text{ret} = \text{CERCA}(T \rightarrow \text{dx}, k)$

ELSE IF  $T \rightarrow \text{Key} > k$  THEN

$\text{ret} = \text{CERCA}(T \rightarrow \text{sx}, k)$

ELSE

$\text{ret} = T$

ELSE

return  $\text{ret}$

## INSERIMENTO RICORSIVO IN CODA

Dobbiamo assicurare che il caso base sia terminale (dopo non facciamo nulla). Li ponteremo il padre di ogni modo e cercando la posizione di K faremo puntare il padre al nuovo nodo.

INSRC ( $T, k, p$ )

IF  $T \neq \text{NIL}$  THEN

IF  $T \rightarrow \text{Key} < k$  THEN

INSRC ( $T \rightarrow dx, k, T$ )

ELSE IF  $T \rightarrow \text{Key} > k$  THEN

INSRC ( $T \rightarrow dx, k, T$ )

ELSE

$T = \text{allocando}()$

$T \rightarrow \text{key} = k$

IF  $p \rightarrow \text{key} > k$  THEN

$p \rightarrow sx = T$

ELSE

$p \rightarrow dx = T$

RETURN  $T$

Quest' algoritmo tuttavia non funziona se l'albero è vuoto.

La gestione di questo caso la lasciamo all'algoritmo d'interfaccia

INS( $T, k$ )

IF  $T \neq \text{NIL}$  THEN

  INSRC( $T, k, \text{NIL}$ )

ELSE

$T = \text{allocando}()$

$T \rightarrow \text{key} = k$

RETURN  $T$

### CANCELLAZIONE RICORSIVA IN CODA

E' l' analogo dell' inserimento, con qualche piccola differenza

CANCRC( $T, k, p$ )

IF  $T \neq \text{NIL}$  THEN

  IF  $T \rightarrow \text{Key} < k$  THEN

    INSRC( $T \rightarrow dx, k, T$ )

  ELSE IF  $T \rightarrow \text{Key} > k$  THEN

    INSRC( $T \rightarrow dx, k, T$ )

ELSE IF  $T = P \rightarrow s_x$  THEN

$P \rightarrow s_x = CANCROOT(T)$

ELSE

$P \rightarrow d_x = CANCROOT(T)$

I casi di violazione ora sono se l'albero e' vuoto e se dobbiamo eliminare proprio la radice (per via del padre).

Vediamo come l'algoritmo d'interfaccia gestisce questi casi.

$CANC(T, K)$

IF  $T \neq NIL$  THEN

IF  $T \rightarrow Key \neq Key$  THEN

$CANCR(T, K, NIL)$

ELSE

$T = CANCROOT(T)$

return T

Si noti che  $CANCROOT$  richiede  $STACCATIN$  che e' a sua volta ricorsivo in coda, mantenendo la validita'.