

## MERGESORT ITERATIVO

Ricordiamo l'algoritmo ricorsivo

MERGESORT ( $A, p, r$ )

IF  $p < r$  THEN

$$q = \lfloor \frac{p+r}{2} \rfloor$$

MERGESORT ( $A, p, q$ )

MERGESORT ( $A, q+1, r$ )

MERGE ( $A, p, q, r$ )

In questo caso dobbiamo salvare tutti e 3 i parametri perché vengono utilizzati dopo una chiamata o funzione.

Potremo calcolare  $q$  tramite  $p, r$  a tempo costante, quindi non lo salviamo

MERGESORT ( $A, p, r$ )

$CP = P$

$CR = R$

$STP = STR = NIL$

INIZIO            RIPRESA  
WHILE ( $CP \leq CR \text{ || } STP \neq NIL$ ) DO

lo stack vuoto e' sempre la ripresa

IF CP≤CR THEN

Eseguiamo la simulazione

IF CP < CR THEN

$$q = \lfloor \frac{CP+CR}{2} \rfloor$$

STR = PUSH(STR, CP)

Solviamo il contesto

STR = PUSH(STR, CR)

CR = Q

Li preparammo alla prima chiamata

ELSE

Se CP=CR l'algoritmo non fa nulla, in questo momento assegniamo eventuali valori gli ritorno e facciamo INTRO a false

CR=CP-1

Forzare risalita

ELSE

Da dove dobbiamo riprendere? Abbiamo 2 possibilità (due chiamate a merge sort)

CP = TOP(STR)

CR = TOP(STR)

Le due chiamate a merge sort differiscono sul secondo e terzo parametro.

Ne possiamo scegliere uno come discriminante per capire da quale chiamata provengono. Per fare ciò introduciamo una variabile LAST che salva le variabili prese in input nella chiamata in modo che la chiamata figlia comandi in qualche modo con la chiamata padre

$$q = \left\lfloor \frac{cp+cr}{2} \right\rfloor$$

$CR=R$  IF  $LAST=CR$  THEN

ci serve sicuramente q

Seconda chiamata, dobbiamo chiamare MERGE

MERGE( $A, cp, q, cr$ )

$STP = \text{POP}(STP)$

Puliamo gli stack

$STR = \text{POP}(STR)$

$LAST = CR$

$CR = CP - 1$

}

Risalita

ELSE

$CP = Q + 1$

Prima chiamata, vogliamo simulare la 2°

$CR$  non è stata toccata, i valori già sono  
nello stack quindi non ci serve il push

OSS

Non è possibile realizzare un algoritmo iterativo per il mergesort  
che non utilizzi uno stack, questo è dovuto alla doppia chiamata  
ricorsiva

## FIBONACCI ITERATIVO

Ricorsivamente un esempio e'

$\text{FIB}(n)$

IF  $n \leq 1$  THEN

$$z = n$$

ELSE

$$z = \text{FIB}(n-1)$$

$$z = z + \text{FIB}(n-2)$$

RETUR  $z$

In questo caso dobbiamo solvare  $n$  ed  $z$  che vengono  
modificate e poi lette

$\text{FIB}(n)$

$$CN = n$$

$$STN = STR = \text{NIL}$$

WHILE ( $CN \geq 0$  ||  $STN \neq \text{NIL}$ ) DO

IF  $CN \geq 0$  THEN

IF  $CN \leq 1$  THEN

$$z = CN$$

LAST = CN

RET = r

RET è il valore di ritorno della chiamata

CN = -1

Forziamo inizio false

ELSE

STN = PUSH(STN, CN)

li serve salvare ca solo dopo la seconda chiamata

CN = CN - 1

chiamata

ELSE

CN = TOP(STN)

Ripristino della chiamata padre

Le due chiamate differiscono su n

IF (LAST = CN - 1) THEN

Torniamo della prima

r = RET

STR = PUSH(STR, r)

CN = CN - 2

ELSE

r = TOP(STR)

r = r + RET

STR = POP(STR) STN = POP(STN)

Pulizia degli stack

RET = r

LAST = CN

CN = -1

RETURN r

## COUNT NODI DISPARI ITERATIVO

Consideriamo un algoritmo che conta i nodi dispari di un albero binario

### CONTADISPARI ( $T$ )

$$z=0$$

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

$$z = \text{CONTADISPARI} (T \rightarrow s_x)$$

IF  $T \rightarrow \text{key} \% 2 = 1$  THEN

$$z=z+1$$

$$z = z + \text{CONTADISPARI} (T \rightarrow d_x)$$

return  $z$

### CONTADISPARI-ITERATIVO ( $T$ )

$$CT=T$$

$$STT=ST R=\text{NIL}$$

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

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

$$z=0$$

$STT = PUSH(STT, CT)$

$CT = CT \rightarrow S_x$

Non ci serve salvare  $x$  dopo la prima chiamata

ELSE

$CT = TOP(CTT)$

I due casi differiscono su  $T$ , ma puo' succedere che  $T \rightarrow dx$  e  $T \rightarrow S_x$  sono entrambi NIL.

Per risolvere questo problema svilupperemo l'algoritmo in modo che se  $T \rightarrow dx$  e' NIL la chiamata non viene effettuata, siccome non possiamo prevedere il valore della funzione se  $T$  e' NIL, avremo 0.

IF ( $LAST \neq CT \rightarrow dx$  &  $CT \rightarrow dx \neq NIL$ )

Torniamo dalla prima chiamata

$z = RET$

IF  $CT \rightarrow Key \neq 2=1$  THEN

$z = z + 1$

$STR = PUSH(STR, z)$

$CT = CT \rightarrow dx$

ELSE

IF  $CT \rightarrow dx \neq NIL$  THEN

Dobbiamo capire in quale dei due casi dell'ultimo IF siamo entrati

Veniamo da sinistra

$r = \text{TOP}(\text{STR})$

$r = r + \text{RET}$

$\text{ST} R = \text{POP}(\text{STR})$

$\text{STT} = \text{POP}(\text{STT})$

$\text{RET} = r$

$\text{LAST} = CT$

$CT = \text{NIL}$

$\text{ELSE}$

$r = \text{RET}$

Vengo da sinistra ma destra e' NIL  
quindi non voglio eseguirlo

$\text{IF } CT \rightarrow \text{Key} \geq r \text{ THEN}$

$r = r + 1$

$r$  non ha mai pushato quindi non puliamo STR

$\text{STT} = \text{POP}(\text{STT})$

$\text{RET} = r$

$\text{LAST} = r$

$\text{LAST} = CT$

$CT = \text{NIL}$

Return  $r$

## ALGORITMO PAZZO

ALGO( $T$ )

$R = 0$

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

$X = \text{RND}() \% 2$

IF  $X = 1$  THEN

$R = \text{ALGO}(T \rightarrow d_x) + 1$

$R = R + \text{ALGO}(T \rightarrow s_x)$

ELSE

$R = \text{ALGO}(T \rightarrow s_x) + 1$

$R = \text{ALGO}(T \rightarrow d_x) - R$

RETURN  $R$

In questo caso non possiamo assumere semplicemente di non eseguire le chiamate a NIL a destra o a sinistra perché i due rami dell' $x$  si disambiguano in modo che uno esegue prima la chiamata a sinistra e poi a destra.

Potremo scindere i casi (e decidere di non eseguire la seconda chiamata) a seconda del valore di  $x$ , che quindi dovremo salvare nonostante non venga letto dopo la chiamata.

## ALGO - RICORSIVO (T)

$CT = T$

$STT = STX = NIL$

WHILE ( $CT \neq NIL \quad || \quad STT \neq NIL$ ) DO

IF  $CT \neq NIL$  THEN

$z=0$

$x = RND() \% 2$

IF  $x = 1$  THEN

$STT = PUSH(STT, CT)$

$STX = PUSH(STX, x)$

$CT = CT \rightarrow dx$

ELSE

$STT = PUSH(STT, CT)$

$STX = PUSH(STX, x)$

$CT = CT \rightarrow S_x$

ELSE

Risolita

$CT = TOP(STT)$

$x = TOP(STX)$

IF  $x = 0$  THEN

}

Ripristino del contesto

Seconda coppia di chiamate

| IF  $LAST \neq CT \rightarrow dx \quad \& \quad CT \rightarrow dx \neq NIL$  THEN

Stessa tecnica di prima

$\text{L} = \text{RET} + 1$

Prima chiamata

$\text{STR} = \text{PUSH}(\text{STR}, \text{L})$

$\text{CT} = \text{CT} \rightarrow \text{dx}$

ELSE

IF  $(\text{CT} \rightarrow \text{dx} \neq \text{NIL})$  THEN N

Seconda chiamata

$\text{L} = \text{RET} - 1$

$\text{STT} = \text{POP}(\text{STT}) ; \text{STX} = \text{POP}(\text{STX}) ; \text{STR} = \text{POP}(\text{STR})$

$\text{RET} = \text{L}$

$\text{LAST} = \text{CT}$

$\text{CT} = \text{NIL}$

ELSE

Caso di disambiguità

$\text{L} = \text{RET} + 1$

$\text{L} = 0 - \text{L}$

$\text{STT} = \text{POP}(\text{STT}) ; \text{STX} = \text{POP}(\text{STX})$

$\text{RET} = \text{L}$

$\text{LAST} = \text{CT}$

$\text{CT} = \text{NIL}$

ELSE

IF  $\text{LAST} \neq \text{CT} \rightarrow \text{Sx}$  & &  $\text{CT} \rightarrow \text{Sx} \neq \text{NIL}$  THEN

$\text{L} = \text{RET} + 1$

$\text{STR} = \text{PUSH}(\text{STR})$

$\text{CT} = \text{CT} \rightarrow \text{S}_x$

ELSE

IF  $\text{CT} \rightarrow \text{S}_x \neq \text{NIL}$  THEN

$\text{l} = \text{TOP}(\text{STR})$

$\text{l} = \text{l} + \text{RET}$

$\text{STT} = \text{POP}(\text{STT}) ; \text{STX} = \text{POP}(\text{STX}) ; \text{STR} = \text{POP}(\text{STR})$

$\text{RET} = \text{l}$

$\text{LAST} = \text{CT}$

$\text{CT} = \text{NIL}$

ELSE

stiamo tornando dalla prima chiamata

$\text{l} = \text{RET} + 1$

a dx ma  $\text{s}_x$  e' NIL

$\text{STT} = \text{POP}(\text{STT})$

$\text{STX} = \text{POP}(\text{STX})$

$\text{RET} = \text{l}$

$\text{LAST} = \text{CT}$

$\text{CT} = \text{NIL}$

Return  $\text{l}$