RICORSIONE 2

Chiamata ricorsiva sul caso più piccolo, dobbiamo diminuire la grandezza del problema.
analisi del problema senza concentrarsi sul come.
(slide su teams completare)

Array = cella collegata all'array
obiettivo: avere più sottoarray

ITERATIVA

ris. | 1 | 4 | 2 | 3 | 1 | 2 |

Somma di un array
SOMMA(A,n)
S ← 0
for i = 0 to n-1 do
S ← S + A[i]
return S

qui so come risolvere il problema

RICORSIVA

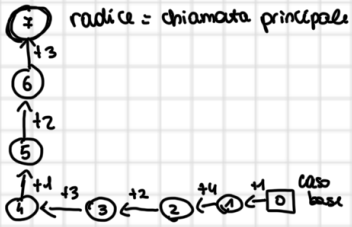
devo diminuire n di 1 ogni volta.

if n = 0 then return 0; (caso base)
return A[n-1] + somma(A,n-1)
↳ aggiungo l'ultimo elemento all'array.

qui non c'è nessuna soluzione, comprendo solo come funziona.

ALBERO DI RICORSIONE

è un albero dove ogni nodo è una chiamata ricorsiva, entra nel dettaglio di come funziona l'algoritmo per la somma di un array



= ciclo for

più facile capire la complessità, qui costante quindi O(n)

nello stack tante chiamate ricorsive quante ce ne sono nell'albero. | ricorsione più elegante

ASPETTI NEGATIVI della ricorsione

- la ricorsione occupa memoria, per le tante chiamate | è lineare
- algoritmo iterativo più veloce a livello pratico poiché non deve gestire chiamate nella memoria

RICORSIONE IN CODA

ciclo for che fa ripartire il ciclo per trasformarlo in modo iterativo

modifichiamo somma
SOMMA - BOUND(A,n,δ) → vogliamo sapere se la somma è maggiore o minore di δ
dobbiamo restituire V o F, non intero e possiamo accorgerci del risultato senza arrivare alla fine

δ = 10, andando a sommare c'è un punto in cui ho già la risposta.

ITERATIVO

S ← 0
i ← 0
condizioni di continuazione, quando una delle due non avviene più mi fermo
While (i < n AND S < δ) do
S ← S + A[i]
IF S < δ return FALSE
return true

oppure

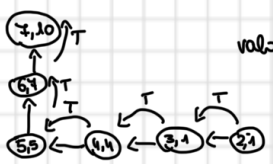
② i ← 0
while (i < n and δ > 0) do
δ ← δ - A[i]
if δ > 0 return FALSE
return true

RICORSIVO

② if δ ≤ 0 then TRUE (caso base)
if n = 0 then FALSE (caso base) (non sono riuscita a raggiungere δ)

return somma-bound(A, n-1, δ-A[n-1])
(restituisce lo stesso valore booleano di somma-bound)

ALBERO:



valore booleano ripropagato per tutto l'albero

Questo albero è variabile dipende da n | caso pessimo quando si deve arrivare ad n

EQUAZIONE DI RICORRENZA

↳ descrive il tempo di esecuzione di un algoritmo ricorsivo
T(n) il problema di un algoritmo su input n
T = tempo ↳ Quanto tempo ci metto.

T(n) = T(n-1) cioè il tempo di esecuzione di uno stesso problema su un sottoproblema più piccolo
+ O(1) ↳ tempo costante che corrisponde alla somma.

in questo esempio array
$T(n) = T(n-1)+1 = (T(n-2)+1)+1 = ((T(n-3)+1)+1)+1 = \dots = T(0) + \underbrace{1+1+1\dots+1}_{n} = n$

negli alberi binari non considero tutto l'input.

esempi
- $T(n) = 3T(\frac{n}{2}) + \log n$ dove 3 sottoproblemi di $\frac{n}{2}$ + costo di $\log n$ quando faccio le chiamate
- $T(n) = 2T(\frac{n}{4}) + n\log n$
- $T(n) = T(\frac{n}{2}) + 9T(\frac{n}{3}) + 2T(\frac{n}{4}) + n$ 12 sottoproblemi ognuno di questi tempo di esecuzione

MERGE-SORT $(A, \overset{i}{0}, \overset{j}{n-1})$

(11:2 floor = 5)

if $n \leq 1$ return

$m = \frac{n}{2}$ (floor) $\quad (i+j+1)$

$A = [0 \dots m-1]$ $\quad A[0 \dots 1]$ $\quad A[0 \dots 0]$

$A = [m \dots n-1]$ $\quad A[\,]$ $\quad A[1 \dots 1]$

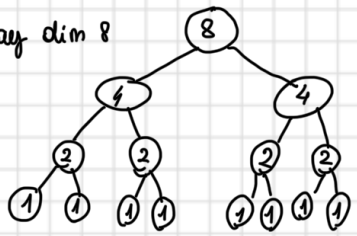
non è giusto

MERGESORT (A, i, m)

MERGESORT $(A, m+1, j)$

MERGE (A, i, j, m)

array dim 8



le chiamate si aprono in order, non tutte insieme

la complessità dipende dall'altezza dell'albero

$n \log n$

$T(n) = 2\left(\frac{n}{2}\right) + n \rightarrow n \log n$

PROBLEMI PIÙ COMPLESSI

- ZAINO (2 versioni)

Valori uguali

peso

① capienza K massima

$A = \{a_0, a_1, a_2, \dots, a_n\}$

- $p: A \rightarrow R$ $\quad p(a_i)$ = peso di a_i

  $0 \leq i \leq n$ $\quad a_i \in A$

- $V: A \rightarrow R$ $\quad V(a_i) = 1$ ogni oggetto uguale

  ↳ valore

APPROCCIO RICORSIVO

ZAINO (n, K)

IF $(n = 0)$ or $p(a_{n-1}) > K$ then return 0

(oppure più oggetto supera la capienza)

RETURN $1 + \text{ZAINO}(n-1, K - p(a_{n-1}))$

Valori diversi x ogni oggetto

②

$p(a_i)$ $\quad V(a_i)$ $\quad S \subseteq A$ $\quad \sum_{a \in S} V(a_i) = \max$

$\sum_{a \in A} p(a) \leq K \rightarrow$ vincolo sullo zaino

es. $A = \{a_0, a_1, a_2, a_3, a_4\}$ $\quad K = 9$

p	2	1	7	5	2
V	3	3	4	2	4

p 212 soluzione migliore

peso più basso e valore maggiore

$$\text{ZAINO}(n,k) = \begin{cases} 0 & n=0, K=0 \\ -\infty & K<0 \rightarrow \text{soluzione inappropriata} \\ \max\left(\begin{matrix} \text{ZAINO}(n-1, K-p(a_{n-1}) + V(a_{n-1}) \\ \text{ZAINO}(n-1,k) \end{matrix}\right) & \end{cases}$$

un elemento in meno

ultimo elemento preso - peso

preso a_n SI → ZAINO$(n-1, K-p(a_n))$

NO → ZAINO$(n-1, k)$ l'oggetto ho tutto ma ho un elemento in meno

ALGORITMO RICORSIVO

IF $n == 0$ or $K == 0$ return 0

if $x < 0$ return $-\infty$

return $\max\left(\text{ZAINO}(n-1, K-p(a_{n-1}) + V(a_{n-1}), \text{ZAINO}(n-1,k)\right)$

↳ ritorna il più grande dei valori

---

esercizi su teams