DOMANDA: Illustrare l'Algoritmo ricorsivo di somma di un array, approccio incrementale (ASOMRICAI)

RISPOSTA:
L'algoritmo ricorsivo di somma di un array, approccio incrementale, calcola la somma degli elementi di un array di dati. I dati possono essere numeri interi, numeri reali o qualsiasi altro insieme di dati su cui sia definita una operazione di somma.
L'ASOMRICAI ha come dati di input un array e il suo size, e ha come dato di output un dato scalare, che è il valore della somma di tutti gli elementi dell'array.
L'ASOMRICAI non fa uso di altri array o strutture dati.
L'ASOMRICAI è basato sull'applicazione del classico approccio incrementale al problema della somma degli elementi di un array.
L'approccio incrementale è una metodologia di progetto di algoritmi in cui la soluzione del problema viene determinata risolvendo iterativamente, a partire dall'istanza banale, istanze di dimensione crescente (incrementata ogni volta di 1) del problema, fino ad arrivare all'istanza in input. A ogni iterazione, la soluzione dell'istanza è calcolata utilizzando la soluzione (già calcolata) dell'istanza precedente.
La tecnica di programmazione ricorsiva, basata sull'autoattivazione di una function, consente di descrivere facilmente un algoritmo basato sull'approccio incrementale, usando la cosiddetta "tail recursion".
Nel caso del problema della somma degli elementi di un array, la soluzione del problema, cioè la somma, viene calcolata come la somma tra l'ultimo elemento dell'array e la somma degli elementi della porzione dell'array che va dal primo al penultimo elemento.
Si noti che si tratta di una sorta di "variante" rispetto all'approccio Divide et impera: qui, la porzione "di sinistra" invece di avere size dimezzato (come nel DI) ha size diminuito di 1 e la porzione "di destra" invece di avere size dimezzato (come nel DI) è costituita da un unico elemento (in particolare, l'ultimo elemento). Di qui il nome "tail recursion".
Ogni autoattivazione suddivide una data istanza in due istanze: una di size 1, la cui soluzione è proprio l'unico elemento che la costituisce, e una di dimensione diminuita di 1; crea un processo che risolve l'istanza del problema della somma sulla porzione di size diminuito di 1; e quindi restituisce la somma tra il valore dell'ultimo elemento della porzione e il valore ottenuto da tale processo. A ogni autoattivazione, il processo creato agisce su una "porzione" di size diminuito di 1 rispetto al precedente, fino ad arrivare al size 1, che è il size dell'istanza banale del problema della somma.
Diamo una rapida analisi della meccanica dell' ASOMRICAI. Alla prima attivazione si crea un processo che agisce sull'intero array A, di size n; esso crea a sua volta un processo che effettua la somma sulla porzione dell'array A costituita dai primi (n-1) elementi; il valore calcolato da questo processo è sommato all'ultimo elemento e il risultato è restituito. La porzione è individuata dall'indirizzo base e dal size (n-1).
Il caso base della ricorsione è costituito dalla porzione di size 1, e in tal caso si deve restituire il valore dell'unico elemento della porzione.
Se l'istanza che il processo sta risolvendo non rientra nel caso base, allora si deve autoattivare sulla porzione 0..(n-2), cioè la porzione che ha lo stesso indirizzo base della porzione in input e size diminuito di 1, e sommare il risultato di tale autoattivazione con l'elemento in ultima posizione della porzione in input, restituendo tale somma al chiamante.
Un esempio aiuta a capire la dinamica dell'ASOMRICAI. Si consideri un array A di n=8 elementi interi.
Poiché il size 8 è diverso da 1, non siamo nel caso base.

41	31	27	28	52	36	11	13
0	1	2	3	4	5	6	7

Quindi, l'algoritmo deve continuare, cioè deve effettuare una autoattivazione che crea un processo che agisce sulla porzione di inizio 0 e size 7 (cioè la porzione 0..6); il valore restituito da tale autoattivazione

deve essere sommato al valore dell'elemento in ultima posizione, cioè il valore 13, che è il valore dell'elemento in posizione 7.
Il nuovo processo "vede" la seguente porzione di array
Processo 1

41	31	27	28	52	36	11
0	1	2	3	4	5	6

Poiché il size 7 è diverso da 1, non siamo nella prima situazione del caso base. Si attiva un nuovo processo che agisce sulla porzione di inizio 0 e size 6 (cioè la porzione 0..5); il valore restituito da tale autoattivazione deve essere sommato al valore dell'elemento in ultima posizione, cioè il valore 11, che è il valore dell'elemento in posizione 6.
Il nuovo processo "vede" la seguente porzione di array
Processo 2

41	31	27	28	52	36
0	1	2	3	4	5

Poiché il size 6 è diverso da 1, non siamo nella prima situazione del caso base. Il processo attiva un nuovo processo che agisce sulla porzione di inizio 0 e size 5 (cioè la porzione 0..4); il valore restituito da tale autoattivazione deve essere sommato al valore dell'elemento in ultima posizione, cioè il valore 36, che è il valore dell'elemento in posizione 5.
Il nuovo processo "vede" la seguente porzione di array
Processo 3

41	31	27	28	52
0	1	2	3	4

Poiché il size è 5, non siamo nella prima situazione del caso base. Si crea un nuovo processo e così via, fino al raggiungimento del caso base.
A questo punto ognuno dei processi sospesi 7,6,5,4,3,2,1 somma il valore ricevuto con elemento in ultima posizione della porzione di loro competenza e restituisce tale somma; al termine, il processo sospeso iniziale ottiene il valore dal processo 1, effettua la somma con l'ultimo elemento dell'array A e restituisce la soluzione del problema.

Questa è l'implementazione in C dell'ASOMRICAI, per un array di tipo `int`:

```
int somma_a_ricAI(int a[],int n)
{
 if (n == 1)
     /* soluzione del caso base */
       return a[0];
    else
     /* autoattivazione */
       return a[n-1] + somma_a_ricAI(a,n-1);
}
```

Un breve commento al codice. La function ha la classica struttura delle function ricorsive, cioè un `if-the-else` che distingue il caso base dalle istanze non banali.
Il blocco `then` gestisce il caso base, restituendo il valore dell'unico elemento di una porzione di size 1.
Il blocco `else` gestisce l'autoattivazione. L'autoattivazione agisce sulla porzione che ha lo stesso indirizzo base della porzione ricevuta e size uguale a **`n-1`**, cioè un size diminuito di 1 rispetto al size **`n`** della porzione ricevuta. La soluzione dell'istanza è calcolata sommando il valore dell'ultimo elemento della porzione ricevuta con il valore restituito dall'autoattivazione, ed è restituita al chiamante.

L’esecuzione dell’ASOMRICAI dà luogo alla seguente sequenza di attivazione/sospensione dei processi nello stack (ogni colonna “fotografa” lo stack dopo ogni autoattivazione):

							P(0)			
						P(0..1)	P(0..1)	P(0..1)		
					P(0..2)	P(0..2)	P(0..2)	P(0..2)	P(0..2)	
				P(0..3)	P(0..3)	P(0..3)	P(0..3)	P(0..3)	P(0..3)	P(0..3)
			P(0..4)	P(0..4)	P(0..4)	P(0..4)	P(0..4)	P(0..4)	P(0..4)	P(0..4)
		P(0..5)	P(0..5)	P(0..5)	P(0..5)	P(0..5)	P(0..5)	P(0..5)	P(0..5)	P(0..5)
	P(0..6)	P(0..6)	P(0..6)	P(0..6)	P(0..6)	P(0..6)	P(0..6)	P(0..6)	P(0..6)	P(0..6)
P(0..7)	P(0..7)	P(0..7)	P(0..7)	P(0..7)	P(0..7)	P(0..7)	P(0..7)	P(0..7)	P(0..7)	P(0..7)

P(0..4)			
P(0..5)	P(0..5)		
P(0..6)	P(0..6)	P(0..6)	
P(0..7)	P(0..7)	P(0..7)	P(0..7)

Analizziamo la complessità dell’ASOMRICAI. La complessità di spazio è n, perché l’algoritmo opera completamente “in place”.

Passiamo alla complessità di tempo. L’operazione dominante è l’operazione di somma tra due numeri.

C’è un’unica operazione di somma nell’ASOMRICAI, che è effettuata nel blocco `else` a ogni autoattivazione, con esclusione di quella relativa al caso base. Poiché il numero totale di attivazioni, esclusa quella del caso base, è n-1, è immediato concludere che il numero di totale di somme è n-1.

In conclusione, la complessità di tempo dell’ASOMRICAI è $T(n) = n - 1$ somme.

Si noti che si tratta di una complessità assoluta, cioè indipendente dal valore degli elementi dell’array, e che tale complessità è identica a quella del classico algoritmo non ricorsivo di somma degli elementi di un array basato sull’approccio incrementale.

In particolare, in entrambi gli algoritmi (ricorsivo e iterativo) l’ordine secondo cui sono sommati i vari numeri è esattamente lo stesso. In altre parole, i due algoritmi descrivono la medesima sequenza computazionale di operazioni.