DOMANDA: Illustrare l'Algoritmo di Fusione (AFUS)

RISPOSTA:
L'Algoritmo di Fusione risolve il cosiddetto problema del "merging", ovvero la fusione di due array ORDINATI di dati (dello stesso tipo) in un terzo array ORDINATO che ha come elementi gli elementi dei due array, e quindi ha un size uguale alla somma dei size dei due array. I dati possono essere numeri, caratteri, stringhe di caratteri o qualsiasi altro insieme di dati su cui sia definita una relazione di ordine.

L'AFUS ha come dati di input due array ordinati, che chiamiamo `A` e `B`, e i loro size che chiamiamo rispettivamente `n_A` e `n_B`, e ha come dato di output il terzo array che chiamiamo `C`, il cui size è noto ed è `n_A+n_B`.

L'AFUS è basato sull'idea di effettuare un numero di passi uguale al size del terzo array `C`, cioè `n_A+n_B` passi; a ogni passo, si determina un elemento dell'array `C`.
Detto `i_C` l'indice per accedere agli elementi dell'array `C`, al generico passo `i_C` lo scopo è determinare l'elemento di `C` di indice `i_C`.
Come si costruisce l'elemento `C(i_C)` ?
Per prima cosa, chiamiamo `i_A` e `i_B` i due indici per accedere rispettivamente agli elementi dell'array `A` e agli elementi dell'array `B`. Al primo passo `i_A`, `i_B` e `i_C` indicano il primo elemento (dei rispettivi array).
Al passo `i_C`, ci sono tre possibili scenari:
Il primo è quello per così dire "normale", cioè c'è almeno un elemento di `A` e un elemento di `B` da considerare, ovvero non ancora inseriti nell'array `C`; gli indici `i_A` e `i_B` indicano l'elemento di `A` e l'elemento di `B` che devono essere considerati al fine di determinare quale deve essere inserito in C; è facile convincersi che il più piccolo tra `A(i_A)` e `B(i_B)` deve essere inserito in `C`, nella posizione `i_C` (nel caso gli elementi `A(i_A)` e `B(i_B)` fossero uguali, si inserirebbe uno qualunque dei due); L'indice dell'array il cui elemento viene inserito in `C` deve essere incrementato.
Il secondo scenario è quello in cui tutti gli elementi dell'array `A` sono stati già inseriti in `C` e rimangono da considerare solo gli elementi dell'array `B`; in tal caso, l'elemento `B(i_B)` deve essere inserito in `C`, nella posizione `i_C`, e l'indice `i_B` deve essere incrementato.
Il terzo scenario è quello in cui tutti gli elementi dell'array `B` sono stati già inseriti in `C` e rimangono da considerare solo gli elementi dell'array `A`; in tal caso, l'elemento `A(i_A)` deve essere inserito in `C`, nella posizione `i_C`, e l'indice `i_A` deve essere incrementato.
Si noti che, per una data istanza del problema, solo uno tra lo scenario 2 e lo scenario 3 può verificarsi.

Un esempio aiuta a capire la dinamica dell'AFUS. Si consideri un array `A` ordinato di `n_A`=7 elementi interi e un array `B` ordinato di `n_B`=4 elementi interi

27	36	41	48	52	66	68
i _A=0	1	2	3	4	5	6

31	33	49	50
i _B=0	1	2	3

L'array `C` avrà 11 (cioè 7+4) elementi

0	0	0	0	0	0	0	0	0	0	0
i _C=0	1	2	3	4	5	6	7	8	9	10

Poiché 27 è minore di 31, allora 27 è assegnato a `C(0)` e gli indici `i_A` e `i_C` incrementati

27	0	0	0	0	0	0	0	0	0	0
0	**i _C=1**	2	3	4	5	6	7	8	9	10

27	36	41	48	52	66	68
0	**i _A=1**	2	3	4	5	6

31	33	49	50
i _B=0	1	2	3

Al secondo passo, poiché 31 è minore di 36 allora 31 è assegnato a `C(1)` e gli indici `i_B` e `i_C` incrementati

27	31	0	0	0	0	0	0	0	0	0
0	1	**i _C=2**	3	4	5	6	7	8	9	10

27	36	41	48	52	66	68
0	**i _A=1**	2	3	4	5	6

31	33	49	50
0	**i _B=1**	2	3

Al terzo passo, poiché 33 è minore di 36, allora 33 è assegnato a `C(2)` e gli indici `i_B` e `i_C` incrementati

27	31	33	0	0	0	0	0	0	0	0
0	1	2	**i _C=3**	4	5	6	7	8	9	10

27	36	41	48	52	66	68
0	**i _A=1**	2	3	4	5	6

31	33	49	50
0	1	**i _B=2**	3

E così via, fino a quando tutti gli elementi di uno dei due array sono stati inseriti in `C` (nell'esempio, l'array `B`)

27	31	33	36	48	49	50	0	0	0	0
0	1	2	3	4	5	6	**i _C=7**	8	9	10

27	36	41	48	52	66	68
0	1	2	3	**i _A=4**	5	6

31	33	49	50	
0	1	2	3	**i _B=4**

A questo punto non è più necessario effettuare alcun confronto tra elementi di `A` e di `B`, ma si deve solo assegnare 52 a `C(1)` e incrementare gli indici `i_A` e `i_C`.
L'algoritmo termina quando viene inserito l'elemento nella posizione 10 dell'array `C`.

Questa è l'implementazione in C dell'AFUS, per due array di tipo `char`:

```
void fusioneC(char a[],int n_a,char b[],int n_b,char c[])
{
  int i_a=0,i_b=0,i_c;
  for (i_c=0; i_c < n_a+n_b; i_c++)
    {
     if(i_a<n_a && i_b<n_b)
      {
       if(a[i_a] < b[i_b]){
         c[i_c] = a[i_a];
         ia++;}
       else {
         c[i_c] = b[i_b];
         ib++;}
      }
      else if(i_b >= n_b) {
         c[i_c] = a[i_a];
         ia++;}
       else {
         c[i_c] = b[i_b];
         ib++;}
    }
}
```

Un breve commento al codice. Il `for` su `i_c` da 0 a `n_a+n_b-1` gestisce i passi dell'AFUS. Lo scopo del passo `i_c` è inserire nella posizione `i_c` dell'array `C` uno tra i due elementi `A(i_a)` o `B(i_b)`. La selezione **`if(i_a<n_a && i_b<n_b)`** individua il primo scenario, mentre la selezione **`if(i_b >= n_b) .. else ..`** individua gli altri due scenari.

Una implementazione più elegante dell'AFUS è la seguente, in cui ognuno dei tre scenari è realizzato da un ciclo `while`

```
void fusioneC(char a[],int n_a,char b[],int n_b,char c[])
{
  int i_a=0,i_b=0,i_c=0;
  while (i_a < n_a && i_b <n_b)
    {
     if(a[i_a] < b[i_b])
         c[i_c++] = a[i_a++];
       else
         c[i_c++] = b[i_b++];
     }
  while (i_a < n_a)
     c[i_c++] = a[i_a++];
  while (i_b < n_b)
     c[i_c++] = b[i_b++];
}
```

Analizziamo la complessità dell'AFUS. Chiamiamo n la dimensione computazionale (n = n_A+n_B) del problema. La complessità di spazio è 2 n, perché l'algoritmo costruisce un terzo array di size n.

Passiamo alla complessità di tempo. L'operazione dominante è l'operazione di confronto tra un elemento dell'array `A` e un elemento dell'array `B`.
L'AFUS effettua n = n_A+n_B passi, cioè quanti sono gli elementi di `C` che devono essere determinati. A ogni passo si costruisce un elemento dell'array `C`.
Se si è nello scenario 1, allora è necessario esattamente un confronto per determinare un elemento di `C`.
Se si è nello scenario 2 o nello scenario 3, allora non si effettua alcun confronto.
Quindi si può concludere che il numero totale di confronti è al più n = n_A+n_B.
Si noti che si tratta di una complessità di caso peggiore.
In conclusione la complessità di tempo dell'AFUS è
$T(n)=T(n\_A,+n\_B)=n=O(n)$ confronti, al più; con $n=n\_A+n\_B$.