Introduzione agli algoritmi
Proff. E. Fachini – S. Caminiti
5 maggio 2020

Es 4: Si vuole ordinare un array A che contiene interi arbitrariamente grandi ma disposti in modo tale che la sequenza dei valori di indice pari è crescente e la sequenza dei valori di indice dispari è decrescente. La richiesta è di ordinare A in tempo O(n) nel caso peggiore. È possibile utilizzare array di appoggio.

Esempio:

A=

10	85	60	65	70	60	80	40	90	25	95	20	100
0	1	2	3	4	5	6	7	8	9	10	11	12

Una possibilità consiste nell' ottenere due array ordinati crescenti l'uno con gli elementi presi dalle posizioni pari e l'altro da quello delle posizioni dispari e poi utilizzare la funzione fondi per ordinare i due array. Il tempo complessivo è in θ(n), dovuto sia alla fase di generazione dei due array di appoggio sia alla funzione fondi.

E' possibile costruire i due array con una unica scansione dell'array dato, A, esaminando contemporaneamente da sinistra e da destra l'array e inserendo gli elementi di indice pari della scansione da sinistra e quelli dispari in quella da destra.

Se l'array contiene un numero pari di elementi i due array con gli elementi delle posizioni pari e dispari hanno lo stesso numero di elementi altrimenti quello dei pari ne ha uno in più, visto che 0 è pari.

In alternativa si può pensare di modificare la funzione fondi in modo da adattarla alle ipotesi su A. Si potrebbe copiare A in un array d'appoggio X e poi ricopiare gli elementi in A scorrendo X da sinistra e da destra, confrontando i due elementi e copiando in A l'elemento più piccolo. Come nel caso della fondi si dovrà completare copiando eventuali elementi rimasti in uno dei due array.

Quale dei due approcci è meglio? Dal punto di vista dell'occupazione in memoria abbiamo in entrambi i casi un'occupazione lineare nella dimensione dell'input. Dal punto di vista asintotico entrambi hanno tempo di esecuzione in Θ(n). Nel dettaglio però nel primo caso si fanno (almeno) due scansioni dell'array, una per copiare gli elementi nei due array di appoggio e una per la fondi, mentre nel secondo caso si risolve con un'unica scansione.

Scrivo tre versioni dello Pseudocodice, per mettere in chiaro che durante l'esame non ci si aspetta una soluzione "ottimizzata" dal punto di vista del codice, che è invece necessaria in fase di programmazione. Ciò non toglie che le soluzioni più efficienti possono anche essere le prime che vengono in mente, quindi ecco le tre opzioni, tutte valide ma in ordine crescente di efficienza.

La funzione fondi che viene utilizzata è quella vista a lezione:

Fondi(L,R,A)

Input: L,R ed A sono array di m, n e n+m elementi
prec: L e R sono ordinati, cioè
L[0] ≤ ...≤ L[m-1] e R[0]≤…≤R[n-1]
postc: A contiene gli elementi di L e di R e
A[0] ≤ ...≤ A[m+n-1]

OrdinaPariDispari1(A)

input: A è un array di interi

prec: in A la sequenza dei valori di indice pari è crescente e la sequenza dei valori di indice dispari è decrescente

output l'array A ordinato

n = A.length

if n è pari then crea due array B e C di n/2 elementi

else crea due array B di n/2+1 elementi e C di n/2 elementi

Copia gli elementi di A in posizione pari in B e quelli di posizione dispari in C in un' unica scansione da sinistra

inverti C per avere gli elementi in ordine crescente

Fondi(B,C,A)

OrdinaPariDispari2(A)

input: A è un array di interi

prec: in A la sequenza dei valori di indice pari è crescente e la sequenza dei valori di indice dispari è decrescente

output l'array A ordinato

```
n = A.length

if n è pari  then j = n-1

// j è l'indice di scorrimento da destra dell'array e quindi deve corrispondere alla prima entrata dispari dell'array da destra

                    e  crea due array B e C   di n/2 elementi

               else j = n-2

                    e crea due array  B di n/2+1 elementi e C  di n/2 elementi

i=0; k=0;h=0;

while i < n do

      B[k] = A[i] k++

      i = i+2

      if j > 0  then C[h] = A[j] h++

      j = j-2

Fondi(B,C,A)
```

```
OrdinaPariDispari3(A)

input: A è un array di interi

prec: in  A la sequenza dei valori di indice pari è crescente e la sequenza dei valori di indice dispari è decrescente

output l'array A ordinato

n = A.length

crea un array X di n elementi

copia A in X

if n è pari  then m = n-1 else m = n-2

// m è l'indice dell'ultimo elemento di indice dispari

i = 0; j = m; k = 0

while i < n and j > 0 do
        if X[i] ≤ X[j]  then
                        A[k] = X[i]
                         i = i+2
                        else
                        A[k] = X[j]
                        j = j-2
        k++
while i < n do
        A[k] = X[i]
        k++
        i=i+2
while j > 0 do
        A[k] = X[j]
        k++
        j = j - 2
```