ESERCIZIO DI ASD DEL 20 OTTOBRE 2008

Distanze Minime

Si osservi che in un vettore ordinato due elementi a distanza minima sono sicuramente consecutivi. MinDist(A) sfrutta questa proprietà ed opera nel seguente modo:

- copia il vettore A in un vettore B;
- ordina B;
- cerca in B due elementi consecutivi a distanza minima;
- ricerca in A le posizioni dei due elementi.

La copia in B è necessaria perchè l'esercizio chiede di ritornare le due posizioni degli elementi a distanza minima e non i due elementi.

Algorithm 1 MinDist(A)

```
1: Copia(A, B)
2: MergeSort(B, 1, length(B))
3: i ← CercaMinDist(B)
4: a ← Cerca(A, B[i])
5: b ← Cerca(A, B[i + 1])
6: return a and b
```

Algorithm 2 Copia(A, B)

```
1: for i ← 1 to length(A) do
2:     B[i] ← A[i]
3: end for
```

Algorithm 3 CercaMinDist(B)

```
1: min ← B[2] − B[1]
2: minpos ← 1
3: for i ← 2 to length(B) − 1 do
4:     if B[i + 1] − B[i] < min then
5:         min ← B[i + 1] − B[i]
6:         minpos ← i
7:     end if
8: end for
9: return minpos
```

Date: 20 Ottobre 2008.

Algorithm 4 CERCA(A, x)

```
1: i ← 1
2: while A[i] ≠ x do
3:     i ← i + 1
4: end while
5: return i
```

Correttezza. La correttezza della procedura COPIA è banale. La correttezza della procedura MERGESORT è stata dimostrata a lezione.

Dimostriamo la correttezza della procedura CERCAMINDIST(B). Iniziamo dimostrando l'invariante per il ciclo for.

Invariante 1. *All'inizio della i-esima iterazione del ciclo for di* CERCAMINDIST(B) *la variabile min contiene la distanza minima tra gli elementi in $B[1..i]$ e la distanza minima è raggiunta nelle posizioni minpos e $minpos + 1$, ovvero*

$$d(B[minpos], B[minpos + 1]) = min$$

Dimostrazione. Procediamo per induzione su i.
Base: $i = 2$

La variabile min contiene $B[2] - B[1] = d(B[1], B[2])$ e $minpos$ vale 1. In $B[1..2]$ ci sono solo gli elementi $B[1]$ e $B[2]$, quindi la tesi è soddisfatta.
Passo:

HpInd) All'inizio della $k-1$-esima iterazione del ciclo for la variabile min contiene la distanza minima in $B[1..k-1]$ e $d(B[minpos], B[minpos + 1]) = min$.

TsInd) All'inizio della k-esima iterazione del ciclo for la variabile min contiene la distanza minima in $B[1..k]$ e $d(B[minpos], B[minpos + 1]) = min$.

Dobbiamo analizzare ciò che accade durante l'esecuzione della $k-1$-esima iterazione. Se $B[k] - B[k-1] = d(B[k], B[k-1]) < min$, min prende valore $d(B[k], B[k-1])$ e $minpos$ prende valore $k-1$. Dall'ipotesi induttiva sappiamo che sicuramente non ci sono due elementi a distanza minore di min in $B[1..k-1]$. L'unica cosa da dimostrare è che non è possibile che esista $j < k-1$ tale che $d(B[k], B[j]) < min$. Il vettore B è ordinato quindi se $j < k-1$ abbiamo che $B[j] \leq B[k-1]$ e quindi $B[k] - B[j] \geq B[k] - B[k-1] = min$. Se $B[k] - B[k-1] = d(B[k], B[k-1]) \geq min$, allora min e $minpos$ non vengono aggiornati ed in modo analogo al caso precedente otteniamo la tesi. □

Lemma 1 (Correttezza di CERCAMINDIST(B))**.** CERCAMINDIST(B) *termina sempre ed al termine minpos è tale che $B[minpos]$ e $B[minpos + 1]$ sono a distanza minima.*

Dimostrazione. CERCAMINDIST(B) termina sicuramente dopo la terminazione del ciclo for. Il ciclo for termina quando i prende valore $length(B)$. Dall'invariante abbiamo che al termine del ciclo for min contiene la distanza minima tra gli elementi in $B[1..length(B)]$ e $d(B[minpos], B[minpos + 1]) = min$ che è la tesi. □

Lemma 2 (Correttezza di CERCA(A, x))**.** *Se x è un elemento di A, allora la procedura* CERCA(A, x) *termina ed al termine $A[i] = x$.*

Dimostrazione. Indichiamo con k la prima posizione in cui l'elemento x occorre in A. L'indice i parte da 1 e viene solo incrementato all'interno del while. Quindi dopo $k-1$ iterazioni del while l'indice i assume valore k. Alla k-esima iterazione

del while viene eseguito il test $A[k] \neq x$ che fallisce. Il ciclo while termina a questo punto ed il valore che viene restituito è k. □

Complessità. Sia $n = length(A) = length(B)$. MergeSort$(B, 1, length(B))$ ha complessità $\Theta(n \log n)$. Tutte le altre procedure utilizzate hanno complessità $\Theta(n)$. Quindi la complessità totale è $\Theta(n \log n)$.