

**Prova d'esame di Laboratorio di Programmazione
per il corso di laurea in Matematica
9 Febbraio 2004**

Tema d'esame 2: Metodo di ricerca binaria su un elenco ordinato; inserimento di nuovi termini nell'elenco e riordinamento

Descrizione dell'algoritmo (inizio)

Supponiamo che gli elementi di una successione finita di n numeri interi a_0, \dots, a_{n-1} siano ordinati in modo non decrescente; vogliamo aggiungere l'elemento a_n a questo "elenco": ci chiediamo in che posizione debba essere inserito a_n (che quindi cambierà il valore del suo indice, assieme ad altri elementi dell'"elenco"), affinché sia rispettato l'ordine non decrescente per tutti i primi $n + 1$ elementi della successione finita.

L'*indice di inserimento* (d'ora in avanti *i.i.*) del nuovo elemento a_n in un elenco ordinato a_0, \dots, a_{n-1} può essere determinato utilizzando l'algoritmo di ricerca binaria, che possiamo riassumere come segue:

- (I) se a_n è minore di a_0 , allora l'*i.i.* è uguale a 0;
- (II) se a_n è maggiore o uguale di a_{n-1} , allora l'*i.i.* è uguale a n ;
- (III) se entrambi i tests ai punti (I) e (II) sono falsi, allora definiamo l'*indice estremo inferiore* (d'ora in avanti *i.e.i.*) della ricerca binaria uguale a 0 e l'*indice estremo superiore* (d'ora in avanti *i.e.s.*) della ricerca binaria uguale a $n - 1$;
- (IV) iteriamo l'esecuzione dei punti seguenti da (V) fino a (VI), mentre è verificata la condizione che l'*i.e.s.* **non è immediatamente successivo** all'*i.e.i.*; quando questa condizione non è verificata, si passi al punto (VII);
- (V) definiamo l'*indice medio* (d'ora in avanti *i.m.*) della ricerca binaria uguale al valore medio tra l'*i.e.i.* e l'*i.e.s.*;
- (VI) confrontiamo a_n con l'elemento della successione corrispondente all'*i.m.*: se a_n è il minore dei due, allora poniamo l'*i.e.s.* uguale all'*i.m.*, altrimenti, poniamo l'*i.e.i.* uguale all'*i.m.*;
- (VII) poniamo il valore dell'*i.i.* uguale a quello dell'*i.e.s.*

Al fine di comprendere meglio l'algoritmo di ricerca binaria, può essere sicuramente utile considerare un caso concreto. Per esempio, siano $n = 5$,

$$a_0 = 2, \quad a_1 = 5, \quad a_2 = 9, \quad a_3 = 11, \quad a_4 = 14,$$

sia, infine, $a_5 = 8$ l'elemento che vogliamo inserire in modo opportuno.

Si può facilmente verificare che l'esecuzione dell'algoritmo di ricerca binaria, che abbiamo descritto sopra, termina attribuendo all'*i.i.* il valore 2, che è evidentemente corretto.

Obiettivo (intermedio) 1:

si scriva un programma in linguaggio **C** che determina l'*i.i.* all'interno di un vettore formato da elementi ordinati in modo non decrescente, fatta ovviamente eccezione per l'ultimo elemento (che è quello che si deve inserire). Il programma deve contenere:

- (a) la definizione di un vettore di N (N fissato a piacere) elementi di tipo **char**; non occorre che ci sia una fase di input da tastiera, invece gli elementi del vettore possono essere semplicemente definiti all'interno del main;
- (b) una prima funzione che effettua la stampa su video di tutti gli elementi del vettore;

- (c) una seconda funzione che ha come argomenti l'indice n (del nuovo elemento che si vuole inserire) e il vettore, questa seconda funzione deve restituire il valore dell'*i.i.*;
- (d) la chiamata dal main alla prima e alla seconda funzione (**attenzione** al valore del parametro n con cui si effettua la chiamata); con la stampa finale sul video del valore dell'*i.i.* restituito dalla seconda funzione.

Descrizione dell'algoritmo (fine)

A questo punto dovrebbe essere chiaro come è possibile effettuare il “riordinamento parziale” degli elementi della successione finita a_0, \dots, a_n in modo che tutti questi $n+1$ elementi siano disposti in modo non decrescente. Possiamo descrivere l'algoritmo di “riordinamento parziale” come segue:

- (I) si esegua preventivamente l'algoritmo che permette di determinare il valore dell'*i.i.*;
- (II) iteriamo l'esecuzione del punto seguente per j che va dal valore dell'*i.i.* fino a $n-1$;
- (III) scambiamo i valori di a_j e di a_n .

Infine, possiamo pensare di utilizzare l'algoritmo di “riordinamento parziale” che abbiamo appena descritto per effettuare una semplicissima (ma inefficiente) procedura di “ordinamento totale”. Consideriamo gli elementi di una successione finita di N numeri interi a_0, \dots, a_{N-1} , che ora supponiamo disposti in un modo qualsiasi.

L'“ordinamento totale” si ottiene semplicemente iterando l'esecuzione del “riordinamento parziale” rispetto ad n (**attenzione** alla scelta degli estremi di iterazione).

Obiettivo (intermedio) 2:

al programma richiesto dall'obiettivo 1 si aggiunga la procedura di “riordinamento parziale”, in modo tale che

- (a) vi sia una terza funzione che ha come argomenti l'indice n e il vettore, questa funzione deve chiamare quella richiesta al punto (c) dell'obiettivo 1 e poi eseguire i punti (II) e (III) dell'algoritmo di “riordinamento parziale” descritto sopra;
- (b) nel main vi sia una nuova stampa di tutti gli elementi del vettore dopo la chiamata della terza funzione richiesta al punto precedente.

Obiettivo (intermedio) 3:

si modifichi il programma richiesto dall'obiettivo 2 in modo tale che venga effettuato l'“ordinamento totale”, come descritto in precedenza. Si ricordi di modificare la definizione del vettore all'inizio del main, in modo tale che i suoi elementi siano disposti in modo disordinato.

Obiettivo (finale) 4:

si scriva un nuovo programma, in linguaggio **C**, che apre il file `belelenco.dat` (contenente il login name, cognome, nome e gruppo di accesso al laboratorio degli studenti che hanno frequentato il corso di L.d.P. dell'a.a. 2003/04), di cui si può richiedere copia al docente^(⊙). Questo nuovo programma deve

- (a) leggere tutto l'elenco compreso nel suddetto file;
- (b) effettuare l'ordinamento completo (utilizzando l'algoritmo descritto sopra) delle righe dell'elenco in modo crescente rispetto al numero all'interno del login name;
- (c) stampare l'elenco ordinato in un file chiamato `ordelenco.dat`.

(⊙) Per gli studenti che hanno scaricato il presente tema d'esame dalla rete, il suddetto file si trova nella stessa directory del tema d'esame stesso.