

Laboratorio di Algoritmi e Strutture Dati

Esercitazioni del 13 Novembre 2012

Esercizio 3: ricerca binaria

Si tratta di implementare l'algoritmo per la ricerca binaria di un elemento in un array ordinato di interi. Assumiamo di aver definito una macro:

```
#define DIM 200
```

Si supponga che l'array **a** sia un array ordinato (in senso crescente) di DIM interi: per ogni $i = 0, \dots, \text{DIM} - 2$ deve valere $a[i] \leq a[i+1]$.

Sia dato un intero **n** da cercare nell'array **a** nell'intervallo (nel sottoarray) costituito dagli elementi $a[1], a[1+1], \dots, a[r-2], a[r-1]$, per opportuni interi **l, r** con $l < r$.

Allora il generico passo dell'algoritmo deve:

1. Calcolare l'indice **m** dell'elemento centrale dell'intervallo considerato, vale a dire **m** deve essere tale che $(m - l)$ differisca al più di un'unità da $(r - m)$.
(N.B.: Se x e y sono interi, l'espressione x/y è l'approssimazione dal basso della frazione $\frac{x}{y}$. Esempio: $12/5 == 2$.)
2. Se $a[m] == n$ allora l'elemento **n** è stato trovato in posizione **m** nell'array $a[1], a[1+1], \dots, a[r-2], a[r-1]$, e la ricerca termina con successo.
3. Altrimenti:
 - (a) $a[m] > n$ implica che se l'elemento **n** sta nell'array $a[1], a[1+1], \dots, a[r-2], a[r-1]$, allora deve stare nel sottoarray $a[1], a[1+1], \dots, a[m-2], a[m-1]$.
Dunque si itera dal passo 1 ponendo $r = m$.
 - (b) $a[m] < n$ implica che se l'elemento **n** sta nell'array $a[1], a[1+1], \dots, a[r-2], a[r-1]$, allora deve stare nel sottoarray $a[m+1], a[m+2], \dots, a[r-2], a[r-1]$.
Dunque si itera dal passo 1 ponendo $l = m+1$.
4. L'algoritmo termina se trova l'elemento **n**, oppure quando il sottoarray esaminato contiene un solo elemento e tale elemento non è **n**.

Ovviamente i parametri per il passo iniziale dell'algoritmo sono $l = 0$ e $r = \text{DIM}$.

Si richiede di implementare la funzione

```
int binsearch(int *a, int l, int r, int n)
```

dove **a** è l'array ordinato di interi, **l** e **r** sono gli indici dell'estremo sinistro e dell'estremo destro (+1) del sottoarray da ordinare (vedi sopra), e **n** è l'elemento da ricercare.

binsearch deve restituire la posizione **m** dell'elemento **n**, oppure -1 se l'elemento **n** non compare nell'array.

Il programma per testare l'algoritmo di ricerca binaria deve:

1. Leggere DIM interi dal file `numeri.txt` all'array `a`, locale a `main`.
2. Ordinare l'array `a`, ad esempio utilizzando uno dei metodi di ordinamento elementari implementati nell'esercizio della lezione dell'8 Novembre (importate il codice necessario. Una soluzione proposta dell'esercizio è nel file `elesort0.c`).
3. Stampare il messaggio `Immetti elemento da ricercare:.`
4. Leggere da tastiera il numero intero `n` da ricercare (non si richiede alcun controllo).
5. Se l'elemento `n` si trova in posizione `m` nell'array ordinato, allora si deve stampare il messaggio `Elemento n trovato in posizione m.`
6. Altrimenti stampare il messaggio `Elemento n non trovato.`

Facoltativo

Implementate `binsearch` come funzione *ricorsiva*.

(Se invece avete già implementato `binsearch` come funzione ricorsiva, scrivetene una versione *iterativa*.)