Algoritmi e Strutture Dati 1

Appello del 7/09/2011

Esercizio 1 (9 punti)

Scrivere un metodo **`public static int Selection (int [] A, int k)`** in Java che preso in input un array A di n numeri interi ed intero k compreso tra 1 ed n, restituisce in output il k-esimo elemento più grande presente nell'array A.

La procedura **NON può modificare** l'array A.

Ad esempio, se A=[1,5,2,5,-1,3] e k=2, il metodo deve restituire 5. Se invece A=[1,5,2,-1,3] e k=3, il metodo deve restituire 2.

La procedura deve avere tempo di esecuzione nel caso peggiore $\boldsymbol{O(n \log n)}$**,** e può sfruttare tutte le procedure viste a lezione (es: *InserctionSort, MergeSort, Quicksort, Binarysearch, LinearSearch, etc…*)

Esercizio 2 (9 punti)

Scrivere un algoritmo **ricorsivo** di ricerca "ternaria" **`public static boolean ricercaTernaria (int [] A, int i, int j, int x)`**, che prende in input un array A ordinato e un intero, e cerca l'intero x in A (tra le posizioni i e j, estremi inclusi) restituendo *true* se e solo se x è presente in A (tra le posizioni i e j, estremi inclusi). [si assuma che la prima volta il metodo è richiamato con i parametri *i* e *j* uguali a *0* e *A.length-1*, rispettivamente]

Ad ogni passo **`ricercaTernaria`**, quando invocato su un array di lunghezza L, confronta l'elemento x con quello in posizione L/3 ed eventualmente con quello in posizione 2L/3, ed eventualmente chiama ricorsivamente se stesso su un opportuno sottoarray lungo approssimativamente L/3.

Si dia la stima ricorsiva **T(n)** del tempo di esecuzione e, dopo averla risolta con un metodo a piacere, si confronti la ricerca ternaria con la ricerca binaria vista a lezione.

Esercizio 3 (9 punti)

Si consideri la classe SimpleList

```
public class SimpleList{
      int key;
      SimpleList next;
      ...
}
```

che rappresenta un elemento di una lista e possiede una chiave intera ed il puntatore all'elemento successivo. Si noti come una lista è identificata in modo naturale dal suo elemento di testa.

Scrivere un metodo **iterativo** **`public static booelan gemelli (SimpleList L)`** in Java che data una SimpleList L restituisce ***true*** se e solo se in L sono presenti due elementi consecutivi aventi la stessa chiave.

Esercizio 4 (9 punti)

a. Mostrare la tabella Hash di dimensione 17 con liste di trabocco che si ottiene a partire dalla tabella vuota inserendo nell'ordine indicato i seguenti elementi:
    - **13, 2, 60, 22, 67, 5, 25, 12, 3, 24, 28**
b. Si descriva brevemente come può essere risolto il problema delle collisioni nelle tabelle Hash (sia con lista di trabocco che con indirizzamento aperto)

Attenzione:

- Per svolgere il compito si hanno a disposizione **90** minuti.
- Scrivere **subito** nome, cognome, matricola e numero del compito su OGNI FOGLIO.
- Durante la prova scritta non è possibile abbandonare l'aula.
- Non è ammesso **per nessun motivo** comunicare in qualsiasi modo con altre persone
- **Non** è possibile consultare appunti, libri, dispense o qualsiasi altro materiale.
- Qualsiasi strumento elettronico di calcolo o comunicazione (telefoni cellulari, calcolatrici, palmari, computer, etc…) deve essere **completamente disattivato** e **depositato in vista sulla cattedra**