Algoritmi e Strutture Dati 1

Appello del 22/06/2011

Esercizio 1 (9 punti)

Scrivere un metodo **`public static int piuFrequente (int [] A)`** in Java che preso in input un array A di numeri interi restituisce in output uno degli interi presenti in A il maggior numero di volte. La procedura **NON può modificare** l'array A.

Ad esempio, se A=[1,5,2,5,5,1,3], il metodo deve restituire 5.

La procedura deve avere tempo di esecuzione nel caso peggiore $\boldsymbol{O(n^2)}$, e NON può sfruttare nessuna delle procedure viste a lezione.

Esercizio 2 (9 punti)

Scrivere un metodo **`public static int piuFrequenteBis (int [] A)`** in Java che preso in input un array A di numeri interi restituisce in output uno degli interi presenti in A il maggior numero di volte. La procedura **NON può modificare** l'array A.

Ad esempio, se A=[1,5,2,5,5,1,3], il metodo deve restituire 5.

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 3 (9 punti)

Scrivere un algoritmo **ricorsivo** di ricerca "ternaria" **`public static int 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 metodi 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)

a. Mostrare l'heap di **minimo** che si ottiene a partire dall'heap vuoto inserendo nell'ordine indicato i seguenti elementi:
    - **24, 2, 6, 15, 8, 90, 5, 22, 3**
b. Mostrare come è possibile implementare un heap tramite array (in particolare illustrare la corrispondenza tra nodi dell'albero dell'heap e posizioni dell'array).
c. Indicare, **giustificando adeguatamente le risposte**, la complessità delle seguenti operazioni per l'heap di massimo **in funzione del numero n di chiavi presenti nell'heap**:
    - Individuazione del massimo
    - Estrazione del massimo
    - Inserimento