Algoritmi e Strutture Dati 1

Appello del 6/07/2010

Esercizio 1

Si consideri la seguente classe Java:

```
public class Elem{
      public int blocco;
      public int valore;
}
```

Scrivere un metodo **`public static void sort (Elem [] A)`** in Java che preso in input un array **A** di ***n*** oggetti di tipo Elem, ordina l'array in modo che gli elementi con lo stesso numero di `blocco` siano tutti consecutivi nell'array, e ordinati in modo non decrescente rispetto a `valore`. Inoltre i blocchi devono essere ordinati in modo crescente rispetto al numero di blocco.

Suggerimento: scegliere un qualsiasi algoritmo di ordinamento visto a lezione (per esempio il Selection Sort riportato qui sotto) e modificarlo in modo che lavori correttamente con array di Elem. In particolare, sarà necessario definire un metodo statico che confronta due Elem identificando chi va prima nell'ordinamento (`public static boolean isLessOrEqual (Elem x, Elem y)`).

```
public static void SelectionSort (int [] A) {
      int n=A.length;
      for (int i=0; i<n; i++) {
            int posmin=i;
            for (int j=i+1; j<n; j++) {
                  if (A[j]<A[posmin]) {posmin=j;}
            }
            int aux=A[i];
            A[i]=A[posmin];
            A[posmin]=aux;
      }
}
```

Si dia una stima del tempo di esecuzione dell'algoritmo proposto nel caso peggiore.

Esercizio 2

Sia $T(n) = \begin{cases} 25T\left(n/5\right) + \mathrm{n}^2 & se\ n > 1 \\ c_1 & se\ n = 1 \end{cases}$

Si dia una stima esplicita (non ricorsiva) di T(n) facendo uso

- del **metodo di sostituzione ed induzione**;
- del **teorema generale** *(mostrare formalmente perché è valido il caso che si applica).*

INIZIO SECONDO PARZIALE

Esercizio 3

a. Mostrare l'heap che si ottiene a partire dall'heap vuoto inserendo nell'ordine indicato i seguenti elementi:
   - **18, 20, 60, 24, 35, 2, 25, 26, 23**
b. Aggiungere alla classe Heap un metodo **`public static int getsecondMax ()`** che, senza modificare l'heap su cui lavora, restituisce il secondo masssimo presente nell'heap, facendo uso di tutti gli altri metodi già presenti nella classe Heap ed implementati a lezione. Si stimi la complessità computazionale di getSecondMax in funzione di **n**, dove ***n*** è il numero di elementi nell'heap.

Esercizio 4

Si consideri una *tabella hash con lista di trabocco* contenente numeri interi, e si spieghi brevemente come una tale tabella possa essere implementata.

Data la funzione hash modulo, si mostri un esempio in cui l'inserimento di **n chiavi intere k_1, k_2, k_3, …, k_n,** in una tabella con **19** posizioni risulti in una tabella in cui la ricerca di una chiave **k** risulta avere il **miglior** tempo possibile di esecuzione a prescindere dalla politica implementata per l'inserimento in lista di nuovi elementi (in testa, in coda,…). In particolare, si fornisca per ogni i da 1 a n il valore di k_i, (in funzione dell'indice i), ed il valore **k** della chiave da cercare.

Esercizio 5

a. Mostrare l'albero binario di ricerca che si ottiene a partire dall'albero vuoto inserendo nell'ordine indicato i seguenti elementi:
   - **13, 2, 60, 22, 67, 5, 25, 12, 3, 24, 28**
b. Mostrare l'ordine di visita dei nodi dell'albero costruito al punto (a) rispetto ai seguenti algoritmi di visita:
   - Visita in ordine intermedio
   - Visita in preordine
   - Visita in postordine

Attenzione:

- Scrivere **subito** nome, cognome, matricola e numero del compito su OGNI FOGLIO.
- non è ammesso **per nessun motivo** l'uso di telefoni cellulari, calcolatrici, etc…
- **non** è possibile consultare appunti, libri, dispense.