Algoritmi e Strutture Dati
&
Laboratorio di Algoritmi e Programmazione

Appello del 9 Febbraio 2006

Esercizio 1 (ASD)

Considerata la ricorrenza:

$$T(n) = 3T(\frac{n}{2}) + 4n^2\sqrt{n}$$

si richiede di:

- risolverla utilizzando il teorema principale;
- dire se $T(n) = O(n^3)$, giustificando la risposta.

Esercizio 2 (ASD)

Dire quali tra le seguenti affermazioni sono vere, giustificando la risposta:

1. L'altezza di un albero binario di ricerca con n nodi è nella classe $\Omega(\lg n)$.
2. L'altezza di un albero binario di ricerca con n nodi è nella classe $O(n)$.
3. L'altezza di un albero binario di ricerca con n nodi è nella classe $\Omega(n)$.
4. L'altezza di un albero binario di ricerca con n nodi è nella classe $O(\lg n)$.

Esercizio 3 (ASD)

Si consideri la struttura dati coda di min-priorità realizzata con un array che rappresenta un min-heap. Si aggiunga l'operazione terzomin(Q) che è definita solo se la coda contiene almeno tre elementi e restituisce il valore dell'elemento che ha la terza priorità minima. Scrivere lo pseudocodice di terzomin(Q).

Esercizio 4 (Laboratorio)

Si vuole realizzare il tipo di dato *insieme ordinato* mediante una lista semplice (singly-linked list). Si consideri quindi la seguente classe *SortedSet* appartenente al package *Sets*:

```
package Sets;
public class SortedSet {
    SetRecord head;        // riferimento alla testa della lista che rappresenta l'insieme
    int count;             // totale elementi nell'insieme

    // post: inserisce un nuovo record con chiave ob nella lista
    //       che rappresenta l'insieme, rispettando l'ordinamento
    //       crescente delle chiavi. Nel caso ob sia gia' presente
    //       nell'insieme non effettua l'inserimento e ritorna false.
    //       Altrimenti effettua l'inserimento, aggiorna count e
    //       ritorna true.
    public boolean insert(Comparable ob) {// non implementare!}

    // pre: insieme non vuoto
    // post: cancella il record con chiave ob dalla lista che rappresenta
    //       l'insieme e aggiorna count. Ritorna true se l'operazione e'
```

```
    //       andata a buon fine; ritorna false se ob non e' presente nella
    //       lista
    public boolean delete(Comparable ob) {// implementare!}
}
```

Si richiede di:

1. scrivere la classe *SetRecord.java* del package *Sets* che memorizza un singolo elemento dell'insieme;
2. implementare in modo efficiente il metodo *delete* della classe *SortedSet.java*, rispettando le pre- e post-condizioni assegnate;
3. dimostrare la correttezza del metodo *delete*

Esercizio 5 (Laboratorio)

Implementare nella classe *GenTree.java* del package *Trees* il seguente metodo usando la ricorsione:

```
// pre: k >= 0
// post: ritorna true sse tutti i nodi dell'albero  hanno AL PIU' k figli;
public boolean ktree(int k) {...}
```

L'algoritmo proposto è lineare rispetto al numero di nodi dell'albero? Giustificare la risposta.

Esercizio 6 (ASD)

1. Scrivere un algoritmo ricorsivo che stampa tutte le chiavi presenti in un BST (albero binario di ricerca) senza ripetizioni.
2. **(difficile)** Discutere la correttezza e la complessità dell'algoritmo proposto.

```
***************************** classe TreeNode.java ***************************
package Trees;
class TreeNode {

    Object key;       // valore associato al nodo
    TreeNode parent;    // padre del nodo
    TreeNode child;      // figlio sinistro del nodo
    TreeNode sibling;     // fratello destro del nodo


    // post: ritorna un albero di un solo nodo, con valore value e sottoalberi
    //       sinistro e destro vuoti
    TreeNode(Object ob) {
        key = ob;
        parent = child = sibling = null;
    }


    // post: ritorna un albero contenente value e i sottoalberi specificati
    TreeNode(Object ob,
        TreeNode parent,
        TreeNode child,
        TreeNode sibling) {
        key = ob;
        this.parent = parent;
        this.child = child;
        this.sibling = sibling;
    }

}
***************************** classe GenTree.java ***************************
package Trees;
import java.util.Iterator;
public class GenTree implements Tree{
    private TreeNode root;      // radice dell'albero
    private int count;           // numero di nodi dell'albero
    private TreeNode cursor;    // riferimento al nodo corrente

    // post: costruisce un albero vuoto
    public GenTree() {
        root = cursor = null;
        count = 0;
    }
    ....
    ....
}
```