Algoritmi e Strutture Dati
&
Laboratorio di Algoritmi e Programmazione

— Appello del 17 Luglio 2007 —

Esercizio 1 (ASD)

1. Si discuta la verità o falsità di ciascuna delle seguenti affermazioni.
   (a) $n \lg n = O(n^2)$
   (b) $\sqrt{n} = O(n \lg n)$
   (c) $n^2 = \Omega(n \lg n)$
   (d) $\sqrt{n} + n^2 = \Theta(n^2 + \lg n)$
2. Si scriva la ricorrenza che descrive la conplessità asintotica di un algoritmo di tipo divide et impera che per risolvere un problema di dimensione n lo decompone in 3 sottoproblemi di dimensione $(n/3)$ ciascuno, e ricombina le loro soluzioni con un procedimento la cui complessità asintotica è quadratica.

Esercizio 2 (ASD)

Si consideri la complessità asintotica dell'algoritmo di cancellazione di una chiave in un albero R/N e si discuta la verità o falsità di ciascuna delle seguenti risposte.

(a) O(n).

(b) Ω(log n).

(c) O(n log n).

(d) Θ(n log n).

Esercizio 3 (ASD)

Si consideri la struttura dati *albero binario di ricerca* (`BST`) con gli attributi `key`, `left`, `right` associati a ciascun nodo. Si scriva lo pseudo codice di un algoritmo che dato un albero binario T memorizza le chiavi di T in un array A[1..n] che rappresenta un *max-heap*. Si discuta la complessità e la correttezza dell'algoritmo proposto.

Esercizio 4 (ASD)

Si consideri la struttura dati *albero generale* con gli attributi `key`, `child`, `sibling` associati a ciascun nodo. Si scriva lo pseudo codice di un algoritmo che applicato ad un albero generale T ritorna `true` se T contiene almeno un figlio unico il cui padre è pure figlio unico. (Nota: la radice viene considerata figlio unico)

[continua sul retro]

Esercizio 5 (LAB)

Considerate la seguente specifica della struttura dati *coda*.

```
class Queue {
    // POST: crea un coda vuota
    public Queue();
     // POST: true sse la coda non ha elementi
    public boolean empty();
    // POST: inserisce l'oggetto in coda
    public void enqueue(Object e);
    // POST: estrae l'oggetto che e' sulla testa della coda
    public Object dequeue() throws EmptyQueueException;
}
```

Implementate il metodo seguente.

```
 public static Queue replica(int n, Queue Q)
  /**
   * PRE:  Q != null, n >= 0
   *
   * POST: restituisce una nuova coda ottenuta replicando ciclicamente gli elementi
   *       di Q fino ad ottenere una coda di lunghezza n. Se la coda non ha
   *       elementi, restituisce la coda vuota. Esempi:
   *        replica(7, [a,b,c]) restituisce la coda [a,b,c,a,b,c,a]
   *        replica(2, [a,b,c]) restituisce la coda [a,b]
   *        replica(5, []) restituisce la coda []
   */
```

Esercizio 6 (LAB)

Considerate la seguente specifica della struttura dati *coda a priorità*.

```
class PQueue {
    // POST: restituisce una coda a priorita' vuota
    public PQueue();
    // POST: true sse la coda non ha elementi
    public boolean empty();
    // POST: inserisce l'oggetto e con chiave k nella coda
    public void insert(int k, Object e);
    // POST: restituisce l'elemento associato alla massima chiave della coda;
    public Object max() throws EmptyQueueException;
    // POST: estrae l'oggetto associato alla chiave massima
    public Object extractMax() throws EmptyQueueException;
}
```

Utilizzate `PQueue` per realizzare la struttura dati *pila* specificata come segue:

```
class Stack {
    // POST: crea uno stack vuoto
    public Stack();
    // POST: true sse lo stack  non ha elementi
    public boolean empty();
    // POST: inserisce l'oggetto e sullo stack
    public void push(object e);
    // POST: restituisce l'ultimo elemento inserito sullo stack
    public Object top() throws EmptyStackException;
    // POST: estrae l'ultimo elemento inserito sullo stack
    public Object pop() throws EmptyStackException;
}
```