Appello del 15 novembre 2004
Laboratorio di Algoritmi e Strutture Dati
Anno Accademico 2004/2005

Cognome e Nome: Docente:
Numero di Matricola:

Spazio riservato alla correzione

1	2	3	4	5	6	7	totale
/15	/15	/20	/10	/10	/10	/10	/90

1. Provare il seguente teorema. Se si fa riferimento ad altri lemmi e/o corollari è necessario enunciarli.

   Teorema: Se f è un flusso in una rete di flusso G con sorgente s e destinazione t, allora le seguenti condizioni sono equivalenti:

   1. f è un massimo flusso in G
   2. G_f non contiene augmenting path
   3. |f|= c(S,T) per un taglio (S,T)

2. Visita in profondità
   a. Scrivere lo pseudocodice dell'algoritmo per la visita in profondità di un grafo (DFS). Commentare l'uso delle strutture dati utilizzate. [8 punti]

   b. Indicare, giustificando la risposta, la complessità dell'algoritmo DFS. [7 punti]

3. Scrivere la funzione

   void updateKey(PriorityQueue Q, Object oldKey, Object newKey).

   La funzione deve aggiornare la chiave di tutti gli elementi della coda a priorità Q con chiave oldKey al valore indicato da newKey. La funzione deve utilizzare esclusivamente i metodi nell'interfaccia PriorityQueue ad eccezione del metodo decreaseKey. Si osservi che la funzione può modificare il contenuto di **Q** durante la sua esecuzione [15 punti]. Analizzare la complessità di tempo della funzione proposta [5 punti].

4. Aggiungere alla classe **LinkedTree** (implementa l'interfaccia **Tree** usando come nodo la classe **TreeNode**) il metodo: **allLeaves**() che restituisce un iteratore sugli elementi contenuti nelle foglie.

5. Scrivere una funzione Java **concat**(Sequence **a**, Sequence **b**) che riceve in input due sequenze e restituisce in output una sequenza che corrisponde alla concatenazione delle sequenze **a** e **b** senza ripetizione. Ad esempio se **a**=(2,1,6,3,8) e **b**=(6,4,2,9,10), in output viene restituita la sequenza (2,1,6,3,8,4,9,10).

6. Aggiungere alla classe **NodeList** (che implementa il TDA **List** mediante una lista doppiamente linkata) il metodo: **void RemoveOdd**( ) che rimuove dalla lista tutti gli elementi di ordine dispari.

7. Aggiungere alla classe **ArrayStack** il metodo **multiPop**(int **num**) che restituisce in un array i primi num elementi presenti nello stack. Il metodo deve cancellare dallo stack gli elementi restituiti. Inoltre, deve lanciare l'eccezione **NotEnoughElements** se nello stack vi sono meno di **num** elementi.