Esame di		
	Fondamenti di informatica II - Algoritmi e strutture dati	**12 CFU**
	Algoritmi e strutture dati V.O.	**5 CFU**
	Algoritmi e strutture dati (Nettuno)	**6 CFU**

Appello (straordinario) del 2-11-2017 – a.a. 2016-17
Tempo a disposizione: *4 ore* – somma punti: 34+2

Istruzioni

Lanciare la macchina virtuale Oracle VirtualBox e lavorare all'interno della cartella `ESAME`, avendo cura di creare all'interno della cartella stessa:

- un file `studente.txt` contenente, una stringa per riga, cognome, nome, matricola, email; in tutto quattro righe, memorizzando il file nella cartella `ESAME`;
- una cartella `java.<matricola>`, o `c.<matricola>`, ove al posto di `<matricola>` occorrerà scrivere il proprio numero di matricola, **contenente i file prodotti per risolvere il Problema 2** (in tale cartella si copi il contenuto dell'archivio `c-aux.zip` o `java-aux.zip`); tale cartella va posizionata nella cartella `ESAME`;
- altri file `probl1.<matricola>.txt`, `probl3.<matricola>.txt`, `probl4.<matricola>.txt`, ecc. contenenti, rispettivamente, gli svolgimenti degli altri problemi; tali file vanno posti nella cartella `ESAME` e debbono avere estensione `.txt`.

È possibile consegnare materiale cartaceo integrativo, che verrà esaminato solo a condizione che risulti ben leggibile e sia solo un'integrazione dei contributi scritti nei file di cui sopra.

Per l'esercizio di programmazione (Problema 2) è possibile usare qualsiasi ambiente di sviluppo disponibile sulla macchina virtuale. Si raccomanda però di controllare che i file vengano salvati nella cartella `java.<matricola>`, o `c.<matricola>`. Si consiglia inoltre per chi sviluppa in `c` di compilare da shell eseguendo il comando `make` e poi eseguire `driver` per verificare la correttezza dell'implementazione. Analogamente si raccomanda per chi sviluppa in `java` di compilare da shell eseguendo il comando `javac *.java` e poi eseguire `java Driver` per verificare la correttezza dell'algoritmo.

N.B. Le implementazioni debbono essere compilabili. In caso contrario, l'esame non è superato.

Problema 1 Analisi algoritmo

Si considerino i metodi Java di seguito illustrati.

```
static int[] modIncr(int[] arr) {
  return modIncr(arr, arr.length-1);
}

static int[] modIncr(int[] arr, int i) {
  if(i < 0) return arr;
  if(arr[i] == 0) {
    arr[i] = 1;
    return arr;
  } else {
    arr[i] = 0;
    return modIncr(arr, i-1);
  }
}
```

Sviluppare, *argomentando adeguatamente* (il 50% del punteggio dell'esercizio sarà sulle argomentazioni addotte), quanto segue:

(a) Determinare il costo temporale asintotico di caso peggiore dell'algoritmo descritto da `modIncr(int[])` in funzione della dimensione dell'input. [2/30]

(b) Quanto costano $2^{\texttt{arr.length}}$ invocazioni consecutive di `modIncr(int[])`, assumendo che la prima venga effettuata su una stringa di zeri? (Notare che durante tali invocazioni il caso peggiore determinato al punto (a) si verificherà raramente). [4/30]

Problema 2 Progetto algoritmi C/Java [soglia minima: 4.5/30]

Con riferimento agli alberi binari di ricerca (BST), impiegati per realizzare un dizionario ordinato con chiavi `int` *non negative*, risolvere al computer quanto segue, in Java o in C. Si impieghi la rappresentazione basata su classe/struttura `BST` e classe/struttura `BinNode`. Per le specifiche esatte di rappresentazione si vedano i file forniti a supporto.

- realizzare una funzione/metodo `BST_insert` per inserire in un BST una nuova chiave, avendo cura di gestire correttamente la possibile presenza di chiavi duplicate; la funzione/metodo deve restituire un riferimento/puntatore al nodo appena inserito; [3/30]
- realizzare una funzione/metodo `isBST` che, dato un albero, determini se questo è un BST. In particolare, l'output (di tipo `int`) deve essere così organizzato:

output	significato
-1 :	funzione non implementata
0 :	non è un BST
$+1$:	è un BST

[3/30]

- realizzare una funzione/metodo `isBalanced` che, dato un albero, determini se questo è bilanciato. In particolare, l'output (di tipo `int`) deve essere così organizzato:

output	significato
-1 :	funzione non implementata
0 :	non è bilanciato
$+1$:	è bilanciato

[3/30]

Opzionalmente è possibile sviluppare, al posto di `isBalanced`, una funzione/metodo `isAVL` che, attraverso una unica vista dell'albero, determini in tempo lineare se questo è un BST e se è bilanciato; l'output previsto è un intero, con significato analogo a quanto visto nelle tabelle precedenti. Per tale funzione/metodo verranno riconosciuti fino a 5 punti.

È necessario implementare i metodi in `bst.c` o `Tree.java` identificati dal commento `/*DA IMPLEMENTARE*/`. In tali file è permesso sviluppare nuovi metodi se si ritiene necessario. *Non è assolutamente consentito modificare metodi e strutture già implementati.* È invece possibile modificare il file `driver` per poter effettuare ulteriori test.

Problema 3 Progetto di una mappa

Un progettista che deve realizzare una mappa ordinata può scegliere fra: array ordinato, lista collegata disordinata, BST ed AVL. Partendo da una mappa iniziale vuota verranno fatte le seguenti operazioni, nell'ordine in cui sono descritte: $\sqrt{m}$ inserimenti, m^2 ricerche, $\sqrt[4]{m}$ cancellazioni e $\log m$ range queries, ove m è un parametro che può variare arbitrariamente.

Avendo l'obiettivo di minimizzare il costo temporale dell'intera sequenza di operazioni, determinare il costo totale (funzione di m) derivante dalla migliore scelta possibile fra le strutture dati sopra elencate (il 50% del punteggio dell'esercizio sarà sulle argomentazioni addotte). [6/30]

Problema 4 Rete stradale

La rete stradale di una grande regione è modellata attraverso un *grafo stradale*, ove i nodi rappresentano incroci fra due o più strade e gli archi rappresentano tratti stradali privi di incroci (che sono presenti solo alle loro estremità). Si assume per semplicità che non ci siano strade a senso unico, che per ogni coppia di incroci esista al più un tratto stradale che li collega e che non ci siano strade che, partendo da un incrocio, conducano all'incrocio stesso. Ciascun arco è pesato con la lunghezza del tratto stradale che rappresenta (reale positivo).

Il governo regionale intende dotarsi di uno strumento che gli consenta di individuare, a partire da un incrocio dato s, il tratto stradale più critico, definito come quel tratto che, in caso di blocco, incrementa maggiormente la distanza media fra s e gli altri incroci. Qualora questo tratto critico non sia unico occorre individuarli tutti.

Ciò premesso, si chiede di sviluppare quanto segue.

(a) Descrivere un algoritmo (pseudo-codice) che, preso in input un grafo stradale $G = (V, E)$ e un vertice $s \in V$, determini il valore medio della distanza fra s e gli altri vertici. In formule, indicando con $d(x, y)$ la distanza[1] fra x ed y, l'algoritmo deve determinare la quantità $\overline{d(s)}$ definita come:

$$\overline{d(s)} = \frac{\sum_{t \in V \setminus \{s\}} d(s,t)}{|V| - 1}$$

Valutare il costo dell'algoritmo. [6/30]

N.B. Lo pseudo-codice privo di indentazione riceve penalizzazione del 20%.

[1] È opportuno rammentare che per *distanza* si intende la lunghezza del percorso più breve.

(b) Descrivere un algoritmo (pseudo-codice) che, preso in input un grafo stradale $G = (V, E)$ e un vertice $s \in V$, determini e restituisca l'arco $e \in E$ tale che, in caso di blocco della circolazione sul tratto stradale corrispondente ad e, il valore $\overline{d(s)}$ incrementi maggiormente. Nel caso e non sia unico, l'algoritmo deve determinare e restituire l'elenco di tutti tali archi. Valutare il costo dell'algoritmo. [6/30]
N.B. Lo pseudo-codice privo di indentazione riceve penalizzazione del 20%.