Esame di Fondamenti di Informatica L-B
Ingegneria Gestionale e dei Processi Gestionali

Appello del 20/6/2014

Esercizio 1 (4 punti)

Illustrare il concetto di complessità di un algoritmo.

Esercizio 2 (6 punti)

Siano dati i seguenti metodi Java:

```
public static int f(int V[], int N) {
  int sum=0;
  for(int i=0; i++<N; ++i)
    sum+=V[i];
  return sum;
}

public static int g(int V[], int N) {
  int j=0, sum=0;
  while (++j<N)
    sum+=f(V, j);
  return sum;
}
```

1. Calcolare la complessità in passi base del metodo `f` nei termini del parametro `N` (si distinguano i casi in cui `N` assume valori pari da quelli in cui assume valori dispari).
2. Calcolare la complessità in passi base del metodo `g` nei termini del parametro `N` (si supponga `N` dispari).
3. Calcolare la complessità asintotica del metodo `g` nei termini del parametro `N`.

Esercizio 3 (6 punti)

In vista degli esami di maturità, il liceo “Vicente ElFraile” dello stato caraibico di St. Marquez vuole informatizzare la gestione dei propri studenti. Le informazioni relative a ogni studente comprendono, oltre al nome e al cognome, il punteggio (da 1 a 10) ottenuto nelle tre prove di spagnolo, matematica e quiz. Si scriva una classe `Studente` che:

1. Possieda un opportuno costruttore con parametri.
2. Presenti opportuni metodi che permettano di accedere alle variabili di istanza dell’oggetto.
3. Possieda il metodo `punteggio` che restituisca il punteggio totale ottenuto dallo studente nelle tre prove.
4. Presenti il metodo `toString` che fornisca una descrizione dello studente.
5. Possieda il metodo `equals` per stabilire l’uguaglianza con un altro oggetto `Studente` (la verifica va fatta su nome e cognome).
6. Implementi l’interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Studente` passato come parametro (in ordine decrescente di punteggio totale e, a parità, per ordine alfabetico di cognome e nome).

Esercizio 4 (7 punti)

Si scriva una classe `Classe` che memorizzi le informazioni relative a una classe del liceo “Vicente ElFraile”. Per ogni classe occorre memorizzare la sezione (una singola lettera) e gli studenti che la compongono (all’interno di una lista). La classe `Classe` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, la lista degli studenti è vuota).
2. Possedere opportuni metodi che permettano di accedere alle variabili di istanza dell’oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione della classe (inclusa la descrizione di tutti gli studenti).
4. Possedere il metodo `aggiungi` che, dato un oggetto `Studente`, lo inserisca all’interno della lista, mantenendo la lista ordinata secondo il punto 6. dell’esercizio 3.
5. Presentare il metodo `migliore` che restituisca lo studente della classe che ha ottenuto il punteggio totale massimo (se nella classe è presente almeno uno studente).
6. Possedere il metodo `cerca` che, dato il nome e il cognome di uno studente, indichi se tale studente appartiene o meno alla classe.
7. Presentare il metodo `quanti` che restituisca il numero totale di studenti presenti nella classe.
8. Possedere il metodo `equals` per stabilire l’uguaglianza con un altro oggetto `Classe` (la verifica va fatta esclusivamente sulla sezione).

Esercizio 5 (7 punti)

Si scriva un’applicazione per il liceo “Vicente ElFraile” che:

1. Crei un insieme di oggetti `Classe`.
2. Crei un oggetto `Classe`, lette da tastiera le informazioni necessarie.
3. Inserisca l’oggetto di cui al punto 2. all’interno dell’insieme di cui al punto 1., controllando che tale inserimento sia possibile.
4. Crei un oggetto `Studente`, lette da tastiera le informazioni necessarie.
5. Inserisca lo studente creato al punto 4. tra quelli della classe di cui al punto 2.
6. Stampi a video il nome e cognome dello studente che ha ottenuto il punteggio totale massimo tra tutti gli studenti del liceo.
7. Stampi a video il numero totale di studenti iscritti al liceo.

Soluzione Esercizio 2

Domanda 1:

2 assegnamenti	2	o 2
`i++<N`	`N`/2 + 1	o (`N` + 1)/2 + 1
`sum+=V[i]`	`N`/2	o (`N` + 1)/2
`++i`	`N`/2	o (`N` + 1)/2
Totale	3`N`/2 + 3	o 3`N`/2 + 9/2

Domanda 2:

2 assegnamenti	2
`++j<N`	`N`
`sum+=f(V, j)`	`N` – 1
complessità di `f`	$3N^2/4 + 3N - 15/4$
Totale	$3N^2/4 + 5N - 11/4$

complessità di `f`: $\sum_{\substack{j=1 \\ (j \text{ pari})}}^{N-1} \left(\frac{3j}{2}+3\right) + \sum_{\substack{j=1 \\ (j \text{ dispari})}}^{N-1} \left(\frac{3j}{2}+\frac{9}{2}\right) = \sum_{j=1}^{N-1}\left(\frac{3j}{2}+3\right) + \sum_{\substack{j=1 \\ (j \text{ dispari})}}^{N-1} \frac{3}{2} = \frac{3}{4}N^2 + 3N - \frac{15}{4}$

Domanda 3:

Complessità asintotica: $O(N^2)$

Soluzione Esercizio 3

```
class Studente implements Comparable<Studente> {
  private String nome, cognome;
  private int spagnolo, matematica, quiz;

  public Studente(String nome, String cognome, int spagnolo, int matematica,
      int quiz) {
    this.nome = nome;
    this.cognome = cognome;
    this.spagnolo= spagnolo;
    this.matematica = matematica;
    this.quiz = quiz;
  }

  public String getNome() { return nome; }
  public String getCognome() { return cognome; }
  public int getSpagnolo() { return spagnolo; }
  public int getMatematica() { return matematica; }
  public int getQuiz() { return quiz; }
  public int punteggio() { return spagnolo + matematica + quiz; }

  public String toString() {
    return nome + " " + cognome + ": " + punteggio();
  }

  public boolean equals(Object o) { return equals((Studente) o); }
  public boolean equals(Studente s) {
    return cognome.equals(s.cognome) && nome.equals(s.nome);
  }

  public int compareTo(Studente s) {
    int ret = s.punteggio() – punteggio();
    if(ret==0) ret = cognome.compareTo(s.cognome);
    if(ret==0) ret = nome.compareTo(s.nome);
    return ret;
  }
}
```

Soluzione Esercizio 4

```
import java.util.*;

class Classe {
  private char sezione;
  private List<Studente> l;

  public Classe(char sezione) {
    this.sezione = sezione;
    l=new LinkedList<Studente>();
  }

  public char getSezione() { return sezione; }
  public String toString() {
    return sezione + ":" + l.toString();
  }

  public void aggiungi(Studente s) {
    int i=0;
    while((i<l.size())&&(l.get(i).compareTo(s)<0)) i++;
    l.add(i, s);
  }

  public Studente migliore() {
    if(l.size()>0) return l.get(0); else return null;
  }
  public boolean cerca(String nome, String cognome) {
    for(Studente s: l)
      if(s.getNome().equals(nome) & s.getCognome().equals(cognome))
        return true;
    return false;
  }

  public int quanti() { return l.size(); }
  public boolean equals(Object o) { return equals((Classe) o); }
  public boolean equals(Classe c) { return sezione == c.sezione; }
}
```

Soluzione Esercizio 5

```
import java.util.*;
import fiji.io.*;

class Applicazione {
  public static void main(String[] args) {
    Set<Classe> s = new TreeSet<Classe>();
    Classe c = new Classe(Lettore.in.nextChar());
    if(!s.add(c)) System.out.println("Classe già esistente!");
    Studente t = new Studente(Lettore.in.leggiLinea(),
      Lettore.in.leggiLinea(), Lettore.in.leggiInt(),
      Lettore.in.leggiInt(), Lettore.in.leggiInt());
    c.aggiungi(t);
    Studente max = null;
    for(Classe x: s) {
      Studente migliore = c.migliore();
      if(max == null || migliore.compareTo(max) < 0) max = migliore;
    }
    System.out.println(max.getNome() + " " + max.getCognome());
    int totale = 0;
    for(Classe x: s)
      totale += c.quanti();
    System.out.println(totale);
  }
}
```