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

Appello del 18/9/2014

Esercizio 1 (4 punti)

Descrivere le modalità di studio della complessità temporale di un algoritmo.

Esercizio 2 (6 punti)

Siano dati i seguenti metodi Java:

```
public static int f(int V[], int N) {
  int sum=0, i=N;
  do
    sum+=V[i--];
  while(--i>0);
  return sum;
}

public static int g(int V[], int N) {
  int sum=0;
  for(j=1; ++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 (5 punti)

Per l'edizione di quest'anno di Miss St. Marquez, il patron Vicente El Fraile ha deciso di informatizzare la gestione dei concorsi locali che qualificheranno le aspiranti miss alle finali che si svolgeranno nella capitale. Le informazioni relative a ogni ragazza comprendono, oltre al nome e al cognome, la data di nascita e il punteggio assegnatole dalla giuria. Si scriva una classe `Miss` che:

1. Possieda un opportuno costruttore con parametri.
2. Presenti opportuni metodi che permettano di accedere alle variabili di istanza dell'oggetto.
3. Presenti il metodo `toString` che fornisca una descrizione della miss.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Miss` (la verifica va fatta su nome, cognome e data di nascita).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Miss` passato come parametro (in ordine crescente di età e, a parità, per ordine alfabetico di cognome e nome).

Esercizio 4 (7 punti)

Si scriva una classe `Concorso` che memorizzi le informazioni relative a un concorso locale. Per ogni concorso occorre memorizzare il nome della città sede, la data di svolgimento e le aspiranti miss che vi partecipano (all'interno di una lista). La classe `Concorso` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, la lista delle miss 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 del concorso (inclusa la descrizione di tutte le miss).
4. Possedere il metodo `aggiungi` che, dato un oggetto `Miss`, lo inserisca all'interno della lista, mantenendo la lista ordinata secondo il punto 5. dell'esercizio 3.
5. Presentare il metodo `vincitrice` che restituisca la miss che ha ottenuto il punteggio maggiore (a parità di punteggio va scelta la ragazza più giovane).
6. Possedere il metodo `cerca` che, dato il nome e il cognome di una miss, indichi se tale ragazza partecipa o meno al concorso.
7. Possedere il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Concorso` (la verifica va fatta sul nome della città e sulla data di svolgimento).

Esercizio 5 (8 punti)

Si scriva un'applicazione per il concorso Miss St. Marquez che:

1. Crei un insieme di oggetti `Concorso`.
2. Crei un oggetto `Concorso`, 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 `Miss`, lette da tastiera le informazioni necessarie.
5. Inserisca la miss creata al punto 4. tra quelle partecipanti al concorso di cui al punto 2.
6. Letti da tastiera il nome e il cognome di una miss, stampi a video le sedi dei concorsi cui tale miss ha partecipato.
7. Crei l'insieme `finaliste` che contenga tutte le ragazze vincitrici dei concorsi all'interno dell'insieme di cui al punto 1.
8. Stampi a video le informazioni della miss più giovane vincitrice di un concorso.

Soluzione Esercizio 2

Domanda 1:

2 assegnamenti	2	o 2
`sum+=V[i--]`	N/2	o (N + 1)/2
`--i>0`	N/2	o (N + 1)/2
Totale	N + 2	o N + 3

Domanda 2:

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

complessità di `f`: $\sum_{\substack{j=2 \\ j \text{ pari}}}^{N-1} (j+2) = \sum_{i=1}^{(N-1)/2} (2i+2) = \frac{N^2}{4} + N - \frac{5}{4}$

Domanda 3:
Complessità asintotica: $O(N^2)$

Soluzione Esercizio 3

```
class Miss implements Comparable<Miss> {
  private String nome, cognome;
  private int g, m, a, punti;

  public Miss(String nome, String cognome, int g, int m, int a, int punti) {
    this.nome=nome;
    this.cognome=cognome;
    this.g=g; this.m=m; this.a=a;
    this.punti=punti;
  }

  public String getNome() { return nome; }
  public String getCognome() { return cognome; }
  public String getData() { return g+"/"+m+"/"+a; }
  public int getPunti() { return punti; }

  public String toString() {
    return nome + " " + cognome + ": " + getData() + " - " + punti;
  }

  public boolean equals(Object o) { return equals((Miss) o); }
  public boolean equals(Miss m) {
    return cognome.equals(m.cognome) && nome.equals(m.nome)
      && getData().equals(m.getData();
  }

  public int compareTo(Miss m) {
    int ret = m.a-this.a;
    if(ret==0) ret = m.m-this.m;
    if(ret==0) ret = m.g-this.g;
    if(ret==0) ret = cognome.compareTo(m.cognome);
    if(ret==0) ret = nome.compareTo(m.nome);
    return ret;
  }
}
```

Soluzione Esercizio 4

```
import java.util.*;
class Concorso {
  private String sede, organizzatore;
  private int a, m, g; private List<Miss> l;
  public Concorso(String sede, String organizzatore, int a, int m, int g) {
    this.sede=sede; this.organizzatore=organizzatore;
    this.a=a; this.m=m; this.g=g;
    l=new ArrayList<Miss>();
  }
  public String getSede() { return sede; }
  public String getOrganizzatore () { return organizzatore; }
  public String getData() { return g+"/"+m+"/"+a; }
  public String toString() {
    return sede + " (" + getData() + "): " + l.toString();
  }
  public void aggiungi(Miss m) {
    int i=0;
    while((i<l.size())&&(l.get(i).compareTo(m)<0)) i++;
    l.add(i, m);
  }
  public Miss vincitrice() {
    Miss v = null;
    for(Miss m:l) if(v==null || m.getPunti()>v.getPunti() ||
      (m.getPunti()==v.getPunti() && m.compareTo(v)<0)) v = m;
    return v;
  }
  public boolean cerca(String nome, String cognome) {
    for(Miss m:l)
      if(m.getNome().equals(nome) && m.getCognome().equals(cognome))
        return true;
    return false;
  }
  public boolean equals(Object o) { return equals((Concorso) o); }
  public boolean equals(Concorso c) {
    return sede.equals(c.sede) && getData().equals(c.getData());
  }
}
```

Soluzione Esercizio 5

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

class Applicazione {
  public static void main(String[] args) {
    Set<Concorso> s=new TreeSet<Concorso>();
    Concorso c = new Concorso(Lettore.leggiLinea(), Lettore.leggiLinea (),
      Lettore.leggiInt(), Lettore.leggiInt(), Lettore.leggiInt());
    if(!s.add(c)) System.out.println("Concorso già esistente!");
    Miss m = new Miss(Lettore.leggiLinea(),Lettore.leggiLinea(),
      Lettore.leggiInt(), Lettore.leggiInt(), Lettore.leggiInt(),
      Lettore.leggiInt());
    c.aggiungi(m);
    String nome = Lettore.leggiLinea(), cognome = Lettore.leggiLinea();
    for(Concorso x: s) if(x.cerca(nome, cognome))
      System.out.println(x.getSede());
    Set<Miss> finaliste = new HashSet<Miss>();
    for(Concorso x: s) finaliste.add(x.vincitrice());
    Miss max = null;
    for(Miss v: finaliste) if(max == null || v.compareTo(max)<0) max = v;
    System.out.println(max);
  }
}
```