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

Appello del 10/6/2016

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 i, sum=0;
  for(i=0; ++i<N/2;)
    sum+=V[i];
  return sum;
}

public static int g(int V[], int N) {
  int j=N, sum=0;
  do
    sum+=f(V, --j);
  while(j>0);
  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)

Maurizio (Maurone per gli amici) Mariani è un intraprendente imprenditore a capo di una piccola ditta di lavorazioni meccaniche. In vista della ripresa al termine delle vacanze estive, Mariani ha deciso di memorizzare le informazioni relative ai turni di lavoro all'interno di un calcolatore. In particolare, per ogni lavoratore occorre registrare nome e cognome, la matricola e la qualifica (es. tornitore, rettificatore). Si scriva una classe `Operaio` per Maurizio Mariani che:

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

Esercizio 4 (7 punti)

Si scriva una classe `Turno` che memorizzi le informazioni riguardanti gli operai inseriti in ciascun turno di lavoro (ogni giorno si lavora su un solo tipo di pezzo). Per ogni turno occorre memorizzare la data, il nome del pezzo da lavorare e il numero di pezzi prodotti, mentre gli operai vanno inseriti all'interno di un insieme. La classe `Turno` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, il turno non contiene operai).
2. Possedere opportuni metodi che permettano di accedere alle variabili d'istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione del turno (inclusa la descrizione di tutti gli operai).
4. Possedere il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Turno` (la verifica va effettuata unicamente sulla data).
5. Presentare il metodo `aggiungi` che, dato un oggetto `Operaio`, lo inserisca all'interno dell'insieme, controllando che tale inserimento sia possibile.
6. Presentare il metodo `qualifica` che, data una qualifica, indichi se esiste un operaio con tale qualifica all'interno del turno.

Esercizio 4 (8 punti)

Si scriva un'applicazione per Maurone Mariani che:

1. Crei una lista di oggetti `Turno`.
2. Crei un oggetto `Turno`, lette da tastiera le informazioni necessarie.
3. Inserisca l'oggetto di cui al punto 2. in coda alla lista di cui al punto 1.
4. Crei un oggetto `Operaio`, lette da tastiera le informazioni necessarie.
5. Inserisca l'oggetto di cui al punto 4. all'interno del turno di cui al punto 2., controllando che tale inserimento sia possibile.
6. Letta da tastiera una qualifica, stampi a video la data di tutti i turni che non presentano operai aventi tale qualifica.
7. Letto da tastiera il nome di un pezzo, stampi a video la descrizione del turno che, tra tutti quelli della lista di cui al punto 1., ha prodotto tale pezzo nel numero maggiore.

Soluzione Esercizio 2

Domanda 1:

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

Domanda 2:

2 assegnamenti	2
`sum+=f(V, --j)`	`N`
`j>0`	`N`
complessità di `f`	$N^2/2 + 1/2$
Totale	$N^2/2 + 2N + 5/2$

compl. di `f`: $\sum_{\substack{j=0 \\ j \text{ pari}}}^{N-1} (j+1) + \sum_{\substack{j=0 \\ j \text{ dispari}}}^{N-1} j = \sum_{i=0}^{N-1} j + \sum_{\substack{j=0 \\ j \text{ pari}}}^{N-1} 1 = \frac{N(N-1)}{2} + \frac{N+1}{2} = \frac{N^2+1}{2}$

Domanda 3:

Complessità asintotica: $O(N^2)$

Soluzione Esercizio 3

```
class Operaio implements Comparable<Operaio> {
  private String nome, cognome, qualifica;
  private int matr;

  public Operaio(String nome, String cognome, String qualifica, int matr) {
    this.nome = nome;
    this.cognome = cognome;
    this.qualifica = qualifica;
    this.matr = matr;
  }

  public String getNome() { return nome; }
  public String getCognome() { return cognome; }
  public String getQualifica () { return qualifica; }
  public int getMatricola() { return matr; }

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

  public boolean equals(Object o) { return equals((Operaio) o); }
  public boolean equals(Operaio o) { return matr==o.matr; }

  public int compareTo(Operaio o) {
    int ret = this.cognome.compareTo(o.cognome);
    if(ret==0) ret = this.nome.compareTo(o.nome);
    if(ret==0) ret = this.matr - o.matr;
    return ret;
  }
}
```

Soluzione Esercizio 4

```
import java.util.*;

class Turno {
  private Set<Operaio> s;
  private String pezzo;
  private int a, m, g, pezzi;

  public Turno(String pezzo, int g, int m, int a, int pezzi) {
    this.pezzo = pezzo;
    this.g = g; this.m = m; this.a = a;
    this.pezzi = pezzi;
    s = new TreeSet<Operaio>();
  }

  public String getPezzo() { return pezzo; }
  public String getData() { return g + "/" + m + "/" + a; }
  public int getNumeroDiPezzi() { return pezzi; }
  public String toString() {
    return pezzi + " " + pezzo + "(" + getData()+ "):" + s;
  }

  public boolean equals(Object o) { return equals((Turno) o); }
  public boolean equals(Turno t){ return getData().equals(t.getData()); }
  public boolean aggiungi(Operaio o) { return s.add(o); }
  public boolean qualifica(String qualifica) {
    for(Operaio o: s)
    if(o.getQualifica().equals(qualifica)) return true;
    return false;
  }
}
```

Soluzione Esercizio 5

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

class Applicazione {
  public static void main(String[] args) {
    List<Turno> l = new ArrayList<Turno>();
    Turno t = new Turno(Lettore.in.leggiLinea(), Lettore.in.leggiInt(),
      Lettore.in.leggiInt(), Lettore.in.leggiInt(), Lettore.in.leggiInt());
    l.add(t);
    Operaio o = new Operaio(Lettore.in.leggiLinea(),
        Lettore.in.leggiLinea(), Lettore.in.leggiLinea(),
        Lettore.in.leggiInt());
    if(!t.aggiungi(o)) System.out.println("Operaio gà presente!");
    String qualifica = Lettore.in.leggiLinea();
    for(Turno x: l)
      if(!x.qualifica(qualifica)) System.out.println(x.getData());
    String pezzo = Lettore.in.leggiLinea();
    Turno max=null;
    for(Turno x: l)
      if(x.getPezzo().equals(pezzo) &&
        (max==null||max.getNumeroDiPezzi()<x.getNumeroDiPezzi())) max=x;
    if(max!=null) System.out.println(max);
  }
}
```