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

Appello del 15/2/2019

Esercizio 1 (4 punti)

Discutere i diversi tipi di complessità computazionale di un algoritmo.

Esercizio 2 (6 punti)

Siano dati i seguenti metodi Java:

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

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

1. Calcolare la complessità in passi base del metodo `f` nei termini dei parametri `M` e `N` (si distinguano i casi in cui `M` assume valori minori di `N` da quelli in cui assume valori maggiori o uguali).
2. Calcolare la complessità in passi base del metodo `g` nei termini dei parametri `M` e `N` (si supponga `M<N`).
3. Calcolare la complessità asintotica del metodo `g` nei termini del parametro `M`.

Esercizio 3 (5 punti)

Giorgio Antoni è titolare della ditta di auto-trasporti "Arrivo alle 14". Ogni giorno la sua flotta di mezzi percorre le strade per recapitare merci da Bologna verso ogni città d'Italia. Giorgio ha perciò deciso di informatizzare la gestione dei carichi trasportati ogni giorno. Per questo, per ogni carico, occorre memorizzare il peso (in Kg), una descrizione e la città di destinazione. Si scriva una classe `Carico` per Giorgio Antoni 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 carico.
4. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Carico` (la verifica va fatta su descrizione e città di destinazione).
5. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un oggetto `Carico` passato come parametro (per peso crescente e, a parità, in ordine alfabetico per descrizione).

Esercizio 4 (8 punti)

Si scriva una classe `Viaggio` che registri le informazioni riguardanti i carichi che verranno consegnati in un viaggio. Per ogni viaggio occorre memorizzare la data e la targa del mezzo, mentre i carichi vanno inseriti all'interno di un insieme. La classe `Viaggio` deve:

1. Presentare un opportuno costruttore con parametri (inizialmente, l'insieme dei carichi è vuoto).
2. Possedere opportuni metodi che permettano di accedere alle variabili d'istanza dell'oggetto.
3. Presentare il metodo `toString` che fornisca la descrizione del viaggio (inclusa la descrizione di tutti i carichi).
4. Possedere il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Viaggio` (la verifica va effettuata su targa e data).
5. Presentare il metodo `aggiungi` che, dato un oggetto `Carico`, lo inserisca all'interno dell'insieme, controllando che tale inserimento sia possibile.
6. Possedere il metodo `destinazioni` che restituisca un insieme contenente i nomi delle città destinazione dei carichi del viaggio.
7. Possedere il metodo `peso` che restituisca il peso totale dei carichi del viaggio.

Esercizio 5 (7 punti)

Si scriva un'applicazione per Giorgio Antoni che:

1. Crei una lista di oggetti `Viaggio`.
2. Crei un oggetto `Viaggio`, 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 `Carico`, lette da tastiera le informazioni necessarie.
5. Inserisca l'oggetto di cui al punto 4. all'interno del viaggio di cui al punto 2., controllando che tale inserimento sia possibile.
6. Stampi a video la targa del mezzo che ha il numero maggiore di città destinazione.

Soluzione Esercizio 2

Domanda 1:

2 assegnamenti	2	o 2
`++i<N`	$N-M$	o 1
`sum+=V[i]`	$N-M-1$	o 0
Totale	$2N-2M+1$	o 3

Domanda 2:

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

complessità. di `f`: $\sum_{j=0}^{M-1}(2M-2j+1)+\sum_{j=M}^{N-1}3 = 2M^2-M(M-1)+M+3(N-M) = M^2+3N-M$

Domanda 3:

Complessità asintotica: $O(M^2)$

Soluzione Esercizio 3

```
class Carico implements Comparable<Carico> {
  private String descrizione, citta;
  private int peso;

  public Carico(String descrizione, String citta, int peso) {
    this.descrizione = descrizione;
    this.citta = citta;
    this.peso = peso;
  }

  public String getDescrizione() { return descrizione; }
  public String getCitta() { return citta; }
  public int getPeso() { return peso; }

  public String toString() {
    return descrizione + " (" + peso + "): " + citta;
  }

  public boolean equals(Object o) { return equals((Carico) o); }
  public boolean equals(Carico c) {
    return descrizione.equals(c.descrizione) && citta.equals(c.citta);
  }

  public int compareTo(Carico c) {
    int ret = this.peso - c.peso;
    if(ret==0) ret =  this.descrizione.compareTo(c.descrizione);
    return ret;
  }
}
```

Soluzione Esercizio 4

```
import java.util.*;

class Viaggio {
  private String targa, data;
  private Set<Carico> s;

  public Viaggio(String targa, String data) {
    this.targa = targa;
    this.data= data;
    s = new TreeSet<Carico>();
  }

  public String getTarga() { return targa; }
  public String getData() { return data; }
  public String toString() { return targa + "(" + data + "): " + s;}
  public boolean equals(Object o) { return equals((Viaggio) o); }
  public boolean equals(Viaggio v){
    return targa.equals(v.targa) && data.equals(v.data);
  }
  public boolean aggiungi(Carico c) { return s.add(c); }

  public Set<String> destinazioni() {
    Set<String> result = new HashSet<String>();
    for(Carico c: s) result.add(c.getCitta());;
    return result;
  }

  public int peso() {
    int peso = 0;
    for(Carico c: s) peso += c.getPeso();
    return peso;
  }
}
```

Soluzione Esercizio 5

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

class Applicazione {
  public static void main(String[] args) {
    List<Viaggio> l = new ArrayList<Viaggio>();
    Viaggio v = new Viaggio(Lettore.in.leggiLinea(),
        Lettore.in.leggiLinea());
    l.add(v);
    Carico c = new Carico(Lettore.in.leggiLinea(), Lettore.in.leggiLinea(),
        Lettore.in.leggiInt());
    if(!v.aggiungi(c)) System.out.println("Inserimento non avvenuto!");
    Viaggio max = null;
    int maxD = 0;
    for(Viaggio x: l) {
      int numero = x.destinazioni().size();
      if(numero > maxD) {
        max=x;
        maxD=numero;
      }
    }
    System.out.println(max.getTarga());
  }
}
```