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

Appello del 14/1/2009

Esercizio 1 (4 punti)

Illustrare il concetto di complessità di un algoritmo.

Esercizio 2 (6 punti)

Siano date le seguenti funzioni C:

```
int f(int V[], int M) {
  int j=M, sum=0;

  while(j>0)
    sum+=V[--j];
  return sum;
}

void g(int U[], int V[], int N) {
  int j;

  for(j=1; j<=N; j++) U[j]=f(V,j-1);
}
```

1. Calcolare la complessità in passi base della funzione `f` nei termini del parametro `M`.
2. Calcolare la complessità in passi base della funzione `g` nei termini del parametro `N`.
3. Calcolare la complessità asintotica della funzione `g` nei termini del parametro `N`.

Esercizio 3 (8 punti)

L'aeroporto "Santa Barbara" della capitale dello stato caraibico di St. Marquez desidera informatizzare le informazioni sui voli in partenza dallo scalo. In particolare, per ogni aereo in partenza vengono indicati la compagnia aerea che organizza il volo, il codice del volo (es. AZ475, LH393), la città di destinazione e gli orari di partenza e di arrivo del volo. Si tenga conto che, visto il regime totalitaristico dello stato, non sono possibili né ritardi né tantomeno scioperi. Si scriva una classe `Volo` per l'aeroporto "Santa Barbara" che:

1. Possieda un opportuno costruttore con parametri.
2. Presenti opportuni metodi che permettano di accedere alle variabili di istanza della classe.
3. Possieda un metodo `durata` che calcoli la durata totale del volo in minuti (supponendo che il volo non attraversi fusi orari e che il volo stesso si concluda nella stessa giornata della partenza).
4. Presenti il metodo `toString` che fornisca la descrizione del volo.
5. Possieda il metodo `equals` per stabilire l'uguaglianza con un altro oggetto `Volo` (l'uguaglianza va verificata unicamente sul codice del volo).
6. Implementi l'interfaccia `Comparable`, definendo il metodo `compareTo` per stabilire la precedenza con un `Volo` passato come parametro (l'ordine è dato dall'orario di partenza).

Esercizio 4 (7 punti)

Si scriva una classe `Pista` che memorizzi le informazioni relative ai vari voli in partenza da una certa pista di decollo dell'aeroporto. In particolare, oltre al numero della pista di decollo, gli aerei in partenza vanno memorizzati all'interno di una lista. La classe `Pista` deve inoltre:

1. Possedere un opportuno costruttore (inizialmente la pista non contiene alcun volo).
2. Possedere un metodo `getNumero` che restituisca il numero della pista.
3. Presentare un metodo `cercaVolo` che, dato il nome di una destinazione, restituisca il primo oggetto `Volo` in partenza per tale destinazione, se questo esiste.
4. Possedere un metodo `aggiungiVolo` che, dato un oggetto `Volo`, lo inserisca all'interno della lista di partenza che va mantenuta ordinata; nel caso che l'orario di partenza del volo si sovrapponga a quello di uno già esistente, l'inserimento non va effettuato (suggerimento: si utilizzi il metodo `add(int i, Volo v)` della classe `List<Volo>`).
5. Possedere il metodo `toString` che restituisca una stringa che fornisca una descrizione della pista, comprendendo anche tutti i voli in partenza.

Esercizio 5 (6 punti)

Si scriva un'applicazione per l'aeroporto "Santa Barbara" che:

1. Crei un vettore di 5 oggetti `Pista`, inserendovi i 5 oggetti relativi (con numero progressivo).
2. Crei un oggetto `Volo`, lette da tastiera le informazioni necessarie.
3. Provveda ad inserire l'oggetto creato al punto 2., all'interno della prima pista del vettore di cui al punto 1. in grado di contenerlo.

Per la lettura di dati da tastiera è possibile utilizzare l'oggetto `Lettore.in`, definito all'interno del package `fiji.io`, che possiede i seguenti metodi:

- `boolean leggiBoolean()` Legge un boolean (delimitato da spazi).
- `char leggiChar()` Legge un singolo carattere.
- `double leggiDouble()` Legge un numero razionale (delimitato da spazi).
- `float leggiFloat()` Legge un numero razionale (delimitato da spazi).
- `int leggiInt()` Legge un intero (delimitato da spazi).
- `String leggiLinea()` Legge una linea di testo.
- `String leggiString()` Legge una parola senza spazi al suo interno.

Soluzione Esercizio 2

Domanda 1:

2 assegnamenti	2
`while`	`M` + 1
`sum+=V[i]`	`M`
Totale	2 `M` + 3

Domanda 2:

1 assegnamento	1
`for`	`N` + 1
chiamata di `f`	`N`
complessità di `f`	N^2 + 2`N`
`j++`	`N`
Totale	N^2 + 5`N` + 2

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

Soluzione Esercizio 3

```
class Volo implements Comparable<Volo> {
  private String compagnia, codice, dest;
  private int orap, minp, oraa, mina;

  public Volo(String compagnia, String codice, String dest,
    int orap, int minp, int oraa, int mina) {
    this.compagnia = compagnia; this.codice = codice; this.dest = dest;
    this.orap = orap; this.minp = minp;
    this.oraa = oraa; this.mina = mina;
  }

  public String getCodice() { return codice; }
  public String getDest() { return dest; }

  public int durata() { return (oraa-orap)*60+(mina-minp); }

  public String toString() {
    return codice + " " + dest + "(" + orap + ":" + minp;
  }

  public boolean equals(Object o) { return equals((Volo) o); }
  public boolean equals(Volo v) { return (codice.equals(v.codice)); }

  public int compareTo(Volo v) {
    return (orap-v.orap)*60+(minp-v.minp);
  }
}
```

Soluzione Esercizio 4

```
import java.util.*;
class Pista {
  private int numero;
  private List<Volo> voli;

  public Pista(int numero) {
    this.numero = numero;
    voli = new ArrayList<Volo>();
  }

  public int getNumero() { return numero; }

  public Volo cercaVolo(String dest) {
    for(Volo v:voli)
      if(v.getDest().equals(dest)) return v;
    return null;
  }

  public boolean aggiungiVolo(Volo v) {
    int i=0;
    while((i<voli.size())&&(voli.get(i).compareTo(v)<0))
      i++;
    if((i==voli.size())||(voli.get(i).compareTo(v)>0)) {
      voli.add(i, v);
      return true;
    }
    return false;
  }

  public String toString() {
    return "Pista numero: "+numero+"\n"+voli;
  }
}
```

Soluzione Esercizio 5

```
import fiji.io.*;

class Applicazione {
  public static void main(String[] args) {
    Pista []p=new Pista[5];
    int i;
    for(i=0; i<5; i++) p[i]=new Pista(i); // domanda 1
    Volo v=new Volo(Lettore.in.leggiString(), Lettore.in.leggiString(),
      Lettore.in.leggiString(), Lettore.in.leggiInt(),
      Lettore.in.leggiInt(), Lettore.in.leggiInt(),
      Lettore.in.leggiInt()); // domanda 2
    i=0;
    while(!p[i].aggiungiVolo(v)) i++; // domanda 3
  }
}
```