Esercizi riassuntivi
(Fondamenti di Informatica 2 – Walter Didimo)
Soluzioni

Esercizio 1 Dire quale è la complessità temporale del seguente metodo, espressa con notazione asintotica O(.) (con la migliore approssimazione possibile) rispetto alle dimensioni della matrice a. Si motivi brevemente la risposta.

```
public static int esercizio (int[][] a){
  int m = a.length;
  int n = a[0].length;
  int i, somma=0;
  if (m <= n)
     i=m;
  else
     i=n;
  for (j=0; j<i; j++)
     somma += a[j][j];
  return somma;
}
```

Soluzione

L'operazione dominante del metodo è l'operazione di somma eseguita nel ciclo for. Tale operazione viene svolta un numero di volte pari al minimo tra m ed n. Possiamo dunque scrivere che la complessità è O(min{n,m}). Ovviamente, in modo meno preciso, questo implica anche che la complessità è O(n) e O(m) allo stesso tempo.

Esercizio 2 Esprimere la complessità computazionale dei due metodi seguenti, usando la notazione asintotica O(.) (con la migliore approssimazione possibile) rispetto al numero n passato come parametro. Si motivi brevemente la risposta.

```
public double metodo1 (int n){
      double x = 0;
      for (int j=1; j<n; j++){
            int h = 0;
            while (h < j){
                  x = x + h;
                  h++;
            }
      }
      return x;
}

public double metodo2 (int n){
      double x = n;
      for (int j=1; j<3500; j++){
            int h = 0;
            while (h < j){
                  x = x + h;
                  h++;
            }
      }
      return x;
}
```

Soluzione

- Nel metodo1, le operazione dominanti sono le due operazioni eseguite all'interno del ciclo più annidato, cioè il ciclo while. Ciascuna di queste operazioni ha costo costante e viene eseguita complessivamente un numero di volte $O(n^2)$. Infatti, ciascuna di esse è eseguita un numero di volte che è ordine della somma dei primi n numeri naturali.
- Nel metodo2, le operazioni dominanti sono ancora quelle all'interno del ciclo while. Tuttavia, in questo caso, indipendentemente da n, il numero totale di esecuzioni di queste operazioni è costante (ordine di 3500 al quadrato). Il costo del metodo è dunque O(1).

Esercizio 3 Scrivere un metodo di classe ricorsivo che riceve in ingresso una stringa e calcola il numero di vocali contenute in tale stringa (una soluzione non ricorsiva è considerata errata).

Soluzione

Data una stringa s, e indicando con c il primo carattere di s e con s' il resto della stringa, la funzione ricorsiva alla base del metodo può essere scritta come segue:

numeroVocali (s) = 0	se s è la stringa vuota
numeroVocali (s) = 1 + numeroVocali (s')	se c è una vocale
numeroVocali (s) = numeroVocali (s')	se c non è una vocale

Segue l'implementazione del metodo ricorsivo.

```
public static int numeroVocali (String s){
     int vocali = 0;
     if (s.length() > 0){
          char c = (s.toLowerCase()).charAt(0);
          if (c=='a' || c=='e' || c=='i' || c=='o' || c=='u')
               vocali = 1 + numeroVocali (s.substring(1));
          else
               vocali = numeroVocali (s.substring(1));
     }
     return vocali;
}
```

Esercizio 4 Scrivere un metodo di classe ricorsivo che riceve in ingresso una stringa ed un carattere e restituisce **true** se il carattere compare nella stringa e **false** altrimenti (una soluzione non ricorsiva è considerata errata).

Soluzione

La funzione ricorsiva alla base del metodo dipende da due variabili, una stringa s e un carattere car. Indicando con c il primo carattere di s e con s' il resto della stringa, e la funzione ricorsiva alla base del metodo può essere scritta come segue:

contiene (s,car) = false	se s è la stringa vuota
contiene (s,car) = true	se c = car
contiene (s,car) = contiene (s',car)	se c $\neq$ car

Segue l'implementazione del metodo.

```
public static boolean contiene (String s, char car){
     boolean b;
     if (s.length() == 0)
          b = false;
     else{
          char c = s.charAt(0);
          if (c == car)
               b = true;
          else
               b = contiene (s.substring(1),car);
     }
     return b;
}
```

Esercizio 5 La classe Arrays è una classe dell'API di Java già definita e pronta per l'uso. La classe Arrays si trova nel package java.util, ed ha un metodo statico (cioè di classe), avente il seguente prototipo: *public static void sort (int[ ] seq)*

L'invocazione *Arrays.sort (seq)* eseguirà l'ordinamento dei numeri nell'array seq.

Utilizzando il metodo *Arrays.sort (seq)*, completare il codice della seguente classe:

```
import java.util.*;

class Sequenza{

     /* ordina la sequenza seq e restituisce la massima differenza
     tra due elementi consecutivi */
     public static int maxDelta (int[] seq){
          ...
     }

     /* verifica il corretto funzionamento del metodo maxDelta */
     public static void main (String[] args){
          InputWindow in = new InputWindow ();
          int dim = in.readInt("NUmero elementi?");
          int[] seq = new int[dim];
          for (int i=0; i<seq.length; i++)
               seq[i] = in.readInt();
          int max = Sequenza.maxDelta(seq);
          System.out.println ("Max delta = " + max);
     }

}
```

Soluzione

```
public static int maxDelta (int[] seq){
     int max = 0;
     Arrays.sort (seq);
     for (int i=1; i<seq.length; i++){
          System.out.println (seq[i]);
          int diff = seq[i]-seq[i-1];
          if (diff > max)
```

```
                max = diff;
        }
        return max;
}
```

Esercizio 6 Un oggetto della classe Insieme, permette di rappresentare un insieme di numeri di dimensione arbitraria. Un oggetto della classe Insieme, rappresentante inizialmente un insieme vuoto, può essere istanziato con questo costruttore:

- `public Insieme ( )`

Inoltre, su un oggetto Insieme, possono essere invocati questi due metodi:

- `public void add (int k);  // aggiunge l'intero k all'insieme`
- `public int removeMin ();  // rimuove un elemento di valore minimo`
  `// dall'insieme e lo restituisce.`

Supponendo che la classe Insieme sia già stata implementata, si chiede di spiegare cosa fa il seguente metodo di classe.

```
public static int[] met (int [] a){
        int[] s = new int[a.length];
        Insieme ins = new Insieme ();
        for (int i=0; i<a.length; i++)
                ins.add (a[i]);
        for (int i=0; i<a.length; i++)
                s[i] = ins.removeMin();
        return s;
}
```

Supponendo inoltre che, su un insieme di *m* elementi, una singola esecuzione del metodo *add* o del metodo *removeMin* richieda tempo O(log *m*), dire quale è la complessità asintotica del metodo *met* in funzione della dimensione *n* dell'array *a*.

Soluzione

Il metodo *met* prende in ingresso un array di numeri interi *a* e restituisce un array con gli stessi elementi di *a*, ma ordinati in senso non decrescente.
Assumendo che un singola esecuzione del metodo *add* e del metodo *removeMin* richieda tempo O(log m) su un insieme di m elementi, la complessità del metodo *met* è O(n log n).

Esercizio 7 Considera il seguente array di numeri interi:

10	2	5	14	3	7

Mostrare come l'array si modifica nelle prime tre passate dei seguenti algoritmi di ordinamento: selection sort, insertion sort, bubble sort.

Soluzione

Selection sort

1 passata

2	10	5	14	3	7

2 passata	2	3	5	14	10	7
3 passata	2	3	5	14	10	7

Insertion sort

1 passata	2	10	5	14	3	7
2 passata	2	5	10	14	3	7
3 passata	2	5	10	14	3	7

Bubble sort

1 passata	2	5	10	3	7	14
2 passata	2	5	3	7	10	14
3 passata	2	3	5	7	10	14

Esercizio 8 Con riferimento al modello run time della Java Virtual Machine, dire quali delle seguenti affermazioni è vera o falsa, cerchiando la lettera V oppure la lettera F.

Soluzione

Una variabile di istanza corrisponde ad una locazione di memoria allocata nello heap	V	
Una variabile di istanza corrisponde ad una locazione di memoria allocata nello stack		F
Un oggetto istanziato si trova nello stack		F
Il punto di ritorno di un metodo è un indirizzo nello heap		F
Ogni volta che viene invocato un metodo si alloca un record di attivazione	V	
La deallocazione della memoria è un processo automatico eseguito dalla JVM	V	

Esercizio 9 Dire cosa visualizza il seguente programma, mostrando lo sviluppo del calcolo:

```
class Esame{
      public static void main (String[]args){
            int x = 5;
            int y = f(x);
            System.out.println(y);
      }
      public static int f(int n){
            int z;
            if (n <=1) z = 1;
            else       z = 3 * f(n/2) + 4 * f (n%2);
```

```
            return z;
      }
}
```

Soluzione

Il risultato è 25. Il calcolo è riassunto da questo sviluppo:

y = f(5) = 3 f(2) + 4 f(1) = 3 (3 f(1) + 4 f(0)) + 4 = 3 (3 + 4) + 4 = 21 + 4 = 25

Esercizio 10 Scrivere un metodo ricorsivo che calcola il resto della divisione intera tra un numero intero non negativo **i** e un numero intero positivo **j** (una soluzione non ricorsiva è considerata errata).

Soluzione

```
public static int resto (int i, int j){
          int r;
          if (i < j)
               r = i;
          else // i >= j
               r = resto (i-j,j);
          return r;
}
```