Appello di Programmazione e Algoritmi 1
13 luglio 2021

Esercizio 1

Cosa stampa il seguente frammento di codice Java?

```
public static void main(String[] args) {
          boolean condizione = true;
          int []a = {6,9,10,12,1,4};
          int conto=a.length;
          for (int i=0; i<a.length-1 && condizione; i++)
               {
                    if (a[i] <a[i+1])
                         {System.out.println(a[i+1]-a[i]);}
               else {condizione = false; conto=conto-i;}

               }
          System.out.println(conto);
}
```

NON SARANNO CONSIDERATE VALIDE LE RISPOSTE NON GIUSTIFICATE DAI CALCOLI EFFETTUATI PER OTTENERE IL RISULTATO

Il frammento stampa

```
3
1
2
3
```

Le risposte vanno giustificate dai calcoli

Cosa stampa il seguente programma Java?

```
public static void main(String[] args) {
               System.out.println(enigma(4));
               System.out.println(enigma(11));
               System.out.println(enigma(3));
          }
          static int enigma (int x){
               if (x<0 || x>15) return 2*x;
               if (x<=3) return enigma(-x+2);
               return x+enigma(x+4);
          }
     }
```

NON SARANNO CONSIDERATE VALIDE LE RISPOSTE NON GIUSTIFICATE DAI CALCOLI o DAI RAGIONAMENTI EFFETTUATI PER OTTENERE IL RISULTATO

Il programma stampa

```
56
64
-2
```

Le risposte vanno giustificate dai calcoli. Ad esempio per il primo risultato si può motivare come segue: *enigma*(4)

Il test dei due comandi if nel corpo del metodo enigma non sono soddisfatti (analogamente per le chiamate enigma(8) e enigma(12)), si avrà quindi

```
4+ enigma(8) =
4+(8+enigma(12))=
4+(8+(12+enigma(16)))=
4+(8+(12+(16*2)))= 56
```

Nella chiamata enigma(16) è soddisfatto il test del primo if e si considera quindi il blocco corrispondente.

Esercizio 2

Scrivere un metodo iterativo

public static int maxCaduta(int[] a)

che preso come parametro l'array a di numeri interi, restituisce la massima differenza tra 2 elementi (non necessariamente consecutivi) dell'array, tali per cui il maggiore appare prima nell'array (ha un indice minore).

Nel caso nell'array **a** non siano presenti almeno 2 elementi, in cui il maggiore appare prima nell'array, il risultato sarà 0.

Ad esempio, se *a* = {-10, -15, 8, 2, 3, - 4}, l'output è 12 (ovvero la differenza fra l'elemento 8 e l'elemento -4). Altro esempio, se *a* = {-10, 5, 8, 10, 13, 14} la differenza è 0 (non ci sono 2 elementi in a tali per cui il primo è maggiore del secondo). Ultimo esempio se *a* = {-10, 15, 8, 4, 13, 2}, allora l'output è 13, ovvero la differenza fra l'elemento 15 e l'elemento 2.

Si analizzi la complessità temporale del metodo proposto (soluzioni con complessità minore verranno valutate maggiormente).

```
public static int maxCaduta(int[] a)
     {if (a==null|| a.length<2) return 0;
     int max=0;
     for (int i=0; i<a.length; i++)
         {for (int j=i+1; j<a.length; j++)
           {if (a[i]>a[j] && max<a[i]-a[j]) max =a[i]-a[j];}
         }
     return max;}
```

Compessità $0(n^2)$, dove n è la lunghezza dell'array

Esercizio 3
Scrivere un metodo
static int[] compattak (int[] a, int k)
che, presi come parametri un array **a** di numeri interi ed un intero **k**, restituisce un nuovo array ottenuto eliminando eventualmente i doppioni del valore che si trova in a[k]. Se l'array **a** è null o se **k** non è un valore ammissibile come indice dell'array **a** dovrà essere restituito null

Esempio se *a* = {-10, 5, 8, 2, 8, 13, 8} e k=2 il metodo dovrà restituire *a* = {-10, 5, 8, 2, 13} (perché a[2]=8 e di conseguenza tutti i doppioni del valore 8 vanno eliminati).
Altro esempio, se *a* = {1, 5, 8, 10, 13, 14} e k=2 il metodo dovrà restituire un array uguale ad a, perché non ci sono doppioni del valore a[2]=8). Ultimo esempio se *a* = {-10, 15, 8} e k=4 (oppure k=-2) allora il metodo dovrà restituire null perché k non è un valore ammissibile come indice dell'array a.

```
static int[] compattak (int[] a, int k)
      {(if a==null|| k<0||k>=a.length) return null;}
      int conta=0;
      for (int i=0; i<a.length; i++)
      {if (i !=k && a[i]== a[k]) conta++;}
      int[] res=new int [a.length-conta];
      int j=0;
      for (int i=0; i<a.length; i++)
      {if (i =k || a[i]!= a[k]) {b[j]=a[i]; j++;}
      }
      return b;
}
```

Si analizzi la complessità temporale del metodo proposto.
Complessità O(n) dove n è la lunghezza di a

Esercizio 4
Si consideri il seguente array di numeri interi:
{24,21,19,2,14,20,11,16,9,7}
Mostrare **passo-passo l'esecuzione del merge-sort** per ordinare l'array in modo non decrescente.
Si descriva la complessità computazionale del merge-sort.

{24,21,19,2,14,20,11,16,9,7}
{24,21,19,2,14} **{20,11,16,9,7}**
{24,21,19} **{2,14}** **{20,11,16}** **{9,7}**
{24,21} **{19}** **{2}** **{14}** **{20,11}** **{16}** **{9}** **{7}**
{24} **{21}** **{19}** **{2,14}** **{20}** **{11}** **{16}** **{7,9}**
{21,24} **{19}** **{2,14}** **{11,20}** **{16}** **{7,9}**
{19,21,24} **{2,14}** **{11,16,20}** **{7,9}**
{2,14,19,21,24} **{7,9,11,16,20}**
{2,7,9,11,14,16,19,20,21,24}

[Esercizio di Laboratorio] In relazione al progetto di laboratorio, si implementino i seguenti due metodi:

- int ultimaFigurina(boolean[] album)

restituisce il numero dell'ultima figurina presente nell'album (quella di indice maggiore). Se l'album è vuoto restituisce -1.

- boolean[] creaAlbum(int n, int[] figurine)

restituisce un nuovo album di lunghezza n che contenga le figurine i cui indici sono presenti nell'array figurine. Ad esempio, con input n=7 e l'array {2,3,5} il metodo restituisce l'album:
{false, false, true, true, false, true, false};

Non è necessario implementare il metodo main, ma solamente i due metodi richiesti.
Nell'implementazione dei metodi è possibile sfruttare altri metodi del progetto, se lo ritenete utile.