

ESERCIZI TIPOLOGIA 1

Esercizio

Si consideri il seguente programma Java:

```
public class MainClass {
    public static void main(String[] args) {
        int[] a = {10, 20, 2, 10, 9, 10, 2};
        System.out.println(enigma(a));
    }
    static int enigma(int[] a) { int ris = 0;
    for (int i=0; i<a.length; i++) { boolean flag=false;
        for (int j=0; j<i; j++) { if (a[j] == a[i]) {
            flag=true;
        }
        if (!flag) {
            System.out.println("a[" + i + "]= " + a[i]);
            ris++;
        }
    }
    return ris;
}
}
```

- 1 Cosa stampa il programma?
- 2 Si analizzi la complessità computazionale del metodo enigma.

Esercizio

Si consideri il seguente programma Java:

```
public class MainClass {
    public static void main(String[] args)
    { int[] a = {10, 2, 20, 10, 9, 10, 2};
        System.out.println(enigma(a));
    }
    static int enigma(int[] a) { int ris = 0, i=0;
        while (ris<2 && i<a.length) { int conto=0;
            for (int j=0; j<i; j++) { if (a[j] < a[i]) {
                conto++;
            }
            if (conto>1) {
                System.out.println ("a[" + i + "]= " + a[i]);
                ris++;
            }
            i++;
        }
        return ris;
    }
}
```

- 1 Cosa stampa il programma?
- 2 Si analizzi la complessità computazionale del metodo enigma.

Esercizio

Cosa stampa il seguente frammento di codice Java?

```
int conto=0;
int[] a = {-3, -6, 5, 10, -4};
int n=-30, i, j;
for (i=a.length-1; i>=0; i--) {
    for (j=i-1; j>=0; j--) {
        if (a[i]*a[j]==n){ conto++;
        System.out.println(i + " e " + j);
        }
    }
}
System.out.println ("conto = " + conto);
```

Esercizio

Cosa stampa il seguente programma Java?

```
public class Main {
    public static void main(String[] args) {
        System.out.println(enigma(8,1));
        System.out.println(enigma(7,3));
        System.out.println(enigma(500,4));
    }
    static int enigma (int x, int y){
        if (x<=0 || y<=0) return 0;
        return 2+enigma(x-y,y);
    }
}
```

Esercizio

Cosa stampa il seguente frammento di codice Java?

```
int conto=0;
int[] a = {1, 81, 27, 9, 100};
int n=-30, i, j;
for (i=1; i<=10; i++) {
    for (j=0; j<a.length; j++) { if (i*i==a[j]){
        conto++;
        System.out.println ( "a[" + j + "]" = " + a[j] );
    }
    }
}
System.out.println ("conto = " + conto);
```

Esercizio

Cosa stampa il seguente programma Java?

```
public class Main {
    public static void main(String[] args) {
        System.out.println(enigma(8,2)); System.out.println(enigma(8,3));
        System.out.println(enigma(1000,4));
    }
    static int enigma (int x, int y){ if (x<=0 || y<=0) return 0;
        return 1+enigma(x-y,y);
    }
}
```

ESERCIZI TIPOLOGIA 2

Esercizio

Scrivere un metodo in Java

static void ordinamentoRelativo (int[] a, int rif)

che, preso come parametro un array **a** di numeri interi e un intero **rif**, ordina gli elementi di **a** in modo non decrescente rispetto alla loro distanza (in valore assoluto) dall'intero **rif**.

Ad esempio, se **a**={8, 1, 3, 1, 5, 10, 1, 3} e **rif**=4, **a** dovrà essere ordinato così: {3, 5, 3, 1, 1, 1, 8, 10}.

Si analizzi la complessità temporale del metodo proposto: tale metodo deve avere complessità temporale **O(n log n)** nel caso peggiore (*soluzioni con complessità temporale peggiore danno luogo a una valutazione minore*), dove **n** è la lunghezza dell'array **a** in input.

Esercizio

Scrivere un metodo in Java

static boolean equivalenti (int[] a, int[] b)

che, presi come parametro due array **a** e **b** di numeri interi, restituisce *true* se e solo se **a** e **b** contengono gli stessi elementi (non tenendo in conto le ripetizioni degli elementi e/o l'ordine in cui essi compaiono).

Se **a** e **b** valgono entrambi *null*, viene restituito *true*; se solo uno dei due array vale *null*, viene restituito *false*. Ad esempio, se **a**={1, 8, 3, 1, 5, 10, 1, 4} e **b**={8, 1, 5, 10, 4, 3, 3}, il metodo deve restituire *true*.

Si analizzi la complessità temporale del metodo proposto: tale metodo può richiamare qualsiasi algoritmo di ordinamento e/o di ricerca visto a lezione e deve avere complessità temporale **O(n log n)** nel caso peggiore (*soluzioni con complessità temporale peggiore danno luogo a una valutazione minore*), dove **n** è la lunghezza dell'array **a** in input.

Esercizio

L'array delle ripetizioni di un dato array **a** di numeri interi è definito come un array avente come lunghezza il numero di elementi distinti dell'array **a**, e contenente in posizione **i** (**i**=0,1,...) il numero di ripetizioni nell'array **a** dell'(**i**+1)-esimo elemento più piccolo di **a**. Ad esempio, se **a**={8, 1, 3, 1, 5, 10, 1, 3}, il suo array di ripetizioni è {3,2,1,1,1} in quanto in **a** ci sono 5 elementi distinti, e l'elemento più piccolo di **a** (1) compare 3 volte, il secondo elemento più piccolo (3) compare 2 volte, e i rimanenti elementi compaiono solo una volta.

Scrivere un metodo in Java

static int[] ripetizioni (int[] a)

che, preso come parametro un array **a** di numeri interi, crea e restituisce il suo array delle ripetizioni. Se **a** vale *null*, viene restituito *null*; se **a** è vuoto, viene restituito un array vuoto.

Si analizzi la complessità temporale del metodo proposto: tale metodo può richiamare qualsiasi algoritmo di ordinamento e/o di ricerca visto a lezione e deve avere complessità temporale **O(n log n)** nel caso peggiore (*soluzioni con complessità temporale peggiore danno luogo a una valutazione minore, pari a un massimo di 5 punti totali*), dove **n** è la lunghezza dell'array **a** in input.

Esercizio

Scrivere un metodo in Java

static void ordinamentoRelativo (int[] a)

che, preso come parametro un array **a** di numeri interi, ordina gli elementi di **a** in modo non decrescente **dando però precedenza agli elementi pari rispetto a quelli dispari**. In altre parole, nell'array ordinato dovranno comparire prima tutti gli elementi pari (ordinati in ordine non decrescente) e poi tutti i dispari (ordinati in ordine non decrescente).

Ad esempio, se **a**={8, 1, 3, 1, 5, 10, 1, 3}, dovrà essere ordinato così: {8, 10, 1, 1, 1, 3, 3, 5}.

Si analizzi la complessità temporale del metodo proposto: tale metodo deve avere complessità temporale **O(n log n)** nel caso peggiore (*soluzioni con complessità temporale peggiore danno luogo a una valutazione minore*), dove **n** è la lunghezza dell'array **a** in input.

ESERCIZI TIPOLOGIA 3

Esercizio

Si consideri il seguente array di numeri interi:

{10, 20, 5, 50, 31, 14, 2, 6, 1, 8, 25}

Mostrare **passo-passo l'esecuzione del merge-sort** per ordinare l'array in modo non decrescente.

Si descriva la complessità computazionale del merge-sort (anche scrivendo e resolvendo la ricorrenza $T(n)$).

Esercizio

Si consideri il seguente array di numeri interi:

{10, 20, 5, 50, 31, 14, 2, 6, 1, 8, 25}

Mostrare **passo-passo l'esecuzione del merge-sort** per ordinare l'array in modo non decrescente