Cuaderno de ejercicios: Complejidad algorítmica

Solución: Cuaderno de ejercicios: Complejidad algorítmica

Ejercicio 1: Elemento único:

El siguiente ejercicio determina si un elemento no se repite en el arreglo.
¿Qué complejidad algorítmica tiene? ¿Expresalo en notación BigO?
Dificultad: ★

```
boolean unico(int datos[], int n){
   for(int i=0;i<n;i++){
       for(int j=i+1;j<n;j++){
           if(datos[i]==datos[j]){
               return false;
           }
       }
   }
   return true;
}
```

Ejercicio 2: 3 algoritmos:

¿Qué complejidad algorítmica tiene cada uno de estos 3 algoritmos? ¿Exprésalo en notación BigO?
Dificultad: ★

```
public class App {
```

```java
    int[] numeros = new {1,5,6,3,2};

    public void algoritmo1() {
        for (int a : numeros) {
            System.out.println(a);
        }
        for (int b : numeros) {
            System.out.println(b);
        }
    }

    public void algoritmo2() {
        for (int a : numeros) {
            for (int b : numeros) {
                System.out.println(b);
            }
        }
    }

    public int algoritmo3(int n) {
        if (n <= 1)
            return 1;
        return algoritmo3(n - 1) + algoritmo3(n + 1);
    }

}
```

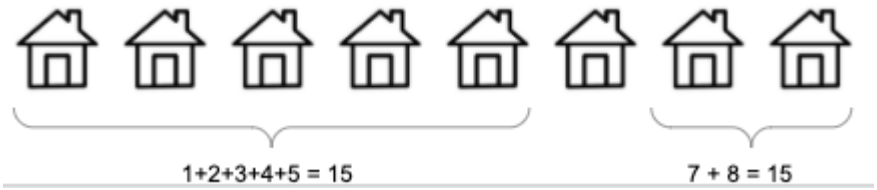
Ejercicio 3: Casas:

Crea el siguiente programa intentado usar la complejidad algorítmica menor.
Dificultad: ★ ★ ★ ★

Marta vive en un pueblo cuyas casas están numeradas del 1 al n
Siendo n el número total de casas de su ciudad.

Marta se pregunta: en qué casa debo vivir para que la suma de los números de las casas de mi izquierda, valgan igual que la suma de los números de las casas de mi derecha.
Si el problema no se puede resolver devolver -1

Ej si hay 8 casas, el resultado es la casa nº 6



Ejercicio 4: Suman X:

Crea el siguiente programa:
Dado un array de números cualesquiera
Determinar si hay 2 números del array que sumen 8 (o cualquier otro número dado)
¿Qué tipo de BigO has usado en la solución?

Dificultad: ★★★★★

```
public class Ejercicio {

   static boolean posee2ValoresSumenX(int[] array, int valor){
      <<< escribe aquí tu código >>>
       return ;
    }
    public static void main(String[] args) throws Exception {
        boolean r = posee2ValoresSumenX(new int[]{1,2,3,9}, 8);
        System.out.println(r); //FALSE
        r = posee2ValoresSumenX(new int[]{1,4,1,4}, 8);
        System.out.println(r);//TRUE
        r = posee2ValoresSumenX(new int[]{1,2,4,16}, 18);
        System.out.println(r);//TRUE
        r = posee2ValoresSumenX(new int[]{1,22,4,16,22,26,1,25}, 23);
        System.out.println(r);//TRUE
    }
}
```

Ejercicio 5: Cuántas veces se repita cada palabra:

Completa el siguiente ejercicio:
Si pide que realices un programa que cuente cuantas veces aparece cada palabra en el texto.
Tendrás que usar HashMaps para solucionar el ejercicio.
¿Qué BigO has usado?
Dificultad: ★★★★★

```
import java.util.HashMap;

public class App {

   static String[] limpiaPalabras(String frase) {
       String listaPalabras[] = frase.split(" ");
       for (int i = 0; i < listaPalabras.length; i++) {
           listaPalabras[i] = listaPalabras[i].toLowerCase().replaceAll("[,\\.]", "");
       }
       return listaPalabras;
   }

   // determinar las veces que se repite cada palabra en el texto
   static void contarPalabras(String listaPalabras[]) {
            <<< escribe aquí tu código >>>
   }

   public static void main(String[] args) throws Exception {
       String frase = "Mañana por la mañana voy a ir a la playa mañana";
       String listaPalabras[] = limpiaPalabras(frase);
       contarPalabras(listaPalabras); // {mañana=3, voy=1, a=2, playa=1, por=1, la=2,
ir=1}
   }
}
```

Ejercicio 6: Complejidad

¿Qué complejidad algorítmica tiene **estaCao**() y **moda()**.? Exprésala con notación BigO
Dificultad: ★

```
boolean estaCao(){
    if(ps <=0 || bloquedada) return true;
    return false;
}
```

```
String moda() {
    int maximaVecesQueSeRepite = 0;
    String moda = "";
    for (int i = 0; i < frases.length; i++) {
        int vecesQueSeRepite = 0;
        for (int j = 0; j < frases.length; j++) {
            if (frases[i] == frases[j])
                vecesQueSeRepite++;
        }
        if (vecesQueSeRepite > maximaVecesQueSeRepite) {
            moda = frases[i];
            maximaVecesQueSeRepite = vecesQueSeRepite;
        }
    }
    return moda;
}
```