



**Alumno—**

Jesus Octavio Amarillas Amaya

**ID—**

207653

**Asignación—**

Algoritmo Burbuja

**Materia—**

Análisis de Algoritmos

**Profesor—**

Sergio Castellanos Bustamante

## 1. Se declara el arreglo y sus elementos

```
9  * Santhor Jesus
10 */
11 public class Pruebas {
12     public static void main(String[] args) {
13         int[] a = {9, 7, 8}; // Arreglo desordenado
14         Ordenamientos ado = new Ordenamientos(a);
15         ado.mostrar();
16         ado.metodoBurbuja(a);
17         ado.mostrar();
18     }
19 }
20
21
```

Name	Type	Value
Static		
args	String[]	#46(length=0)
a	int[]	#48(length=3)
[0]	int	9
[1]	int	7
[2]	int	8

## 2. Se accede al metodo burbuja y se compara entrando al dentro del for:

```
23     this.a = arr;
24 }
25
26 /**
27  * Metodo que utiliza el algoritmo de ordenamiento de burbuja
28  */
29 public void metodoBurbuja(int[] a) {
30     for (int i = 0; i < a.length - 1; i++) { // asignacion: 1, comparacion: n-1, incremento: n-1
31         for (int j = 0; j < a.length - 1 - i; j++) { // asignacion: 1, comparacion: n-1-i, incr =n-2
32             if (a[j] > a[j + 1]) { // 1 comparacion
33                 //3 asignaciones
34                 int aux = a[j]; //1 asignacion
35                 a[j] = a[j + 1]; //1 asignacion
36                 a[j + 1] = aux; //1 asignacion
37             }
38         }
39     } // n^2 +n +4
40 }
41
42 /**
43  * Metodo donde se aplica el algoritmo de insercion
44  */
45 public void metodoInsercion(int[] a) {
46 }
47
```

Name	Type	Value
Static		
this	Ordenamientos	#51
a	int[]	#48(length=3)
[0]	int	9
[1]	int	7
[2]	int	8
i	int	0
j	int	0

## 3. Se accede al for anidado y se asignan los valores de acuerdo a su orden

```
24 }
25
26 /**
27  * Metodo que utiliza el algoritmo de ordenamiento de burbuja
28  */
29 public void metodoBurbuja(int[] a) {
30     for (int i = 0; i < a.length - 1; i++) { // asignacion: 1, comparacion: n-1, incremento: n-1
31         for (int j = 0; j < a.length - 1 - i; j++) { // asignacion: 1, comparacion: n-1-i, incr =n-2
32             if (a[j] > a[j + 1]) { // 1 comparacion
33                 //3 asignaciones
34                 int aux = a[j]; //1 asignacion
35                 a[j] = a[j + 1]; //1 asignacion
36                 a[j + 1] = aux; //1 asignacion
37             }
38         }
39     } // n^2 +n +4
40 }
41
42 /**
43  * Metodo donde se aplica el algoritmo de insercion
44  */
45 public void metodoInsercion(int[] a) {
46 }
47
```

4. Se realizan varias sobre escrituras y se vuelve a entrar al for por la cantidad de iteraciones necesarias

```
5  /**
6  * Metodo que utiliza el algoritmo de ordenamiento de burbuja
7  */
8  */
9  public void metodoBurbuja(int[] a) {
10
11      for (int i = 0; i < a.length - 1; i++) { // asignacion: 1, comparacion: n-1, incremento: n-1
12
13          for (int j = 0; j < a.length - 1 - i; j++) { //asignacion: 1, comparacion: n-1-i, incr =n-2
14
15              if (a[j] > a[j + 1]) { // 1 comparacion
16                  //3 asignaciones
17                  int aux = a[j]; //1 asignacion
18                  a[j] = a[j + 1]; //1 asignacion
19                  a[j + 1] = aux; //1 asignacion
20              }
21          }
22      }
23      // n^2 +n +4
24  }
```

put	Variables x	Name	Type	Value
	<Enter new watch>			
+	this		Ordenamientos	#51
+	a		int[]	#48(length=3)
+	i		int	0
+	j		int	1
+	aux		int	9

5. Se finaliza el metodoBurbuja y se llama al metodo mostrar.

```
1  /**
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5  package algoritmos;
6
7  /**
8  *
9  * @author Jesus
10 */
11 public class Pruebas {
12     public static void main(String[] args) {
13         int[] a = {9, 7, 8}; // Arreglo desordenado
14         Ordenamientos ado = new Ordenamientos(a);
15         ado.mostrar();
16         ado.metodoBurbuja(a);
17         ado.mostrar();
18     }
19 }
20
21
```

Output x

AA\_UC2\_207653 (debug) x | Debugger Console x |

debug:

Arreglo:

[ 9 ]

[ 7 ]

[ 8 ]

Arreglo:

[ 7 ]

[ 8 ]

[ 9 ]

BUILD SUCCESSFUL (total time: 2 minutes 13 seconds)