

# Roberto Verdugo Beltran - 248285

## AA - Reporte de depuración del algoritmo de burbuja

Reporte de debug del algoritmo de burbuja:

**Paso 1:** Los elementos del arreglo son inicializados en el método, además de comenzar a ejecutar el for:

```
31 public void burbuja(int[] a) {
32     // 1
33     for (int i = 0; i < a.length - 1; i++) { // (n - 1)
34         // 1
35         for (int j = 0; j < a.length - 1 - i; j++) { //(n - 1 - i)
36             // 1
37             if (a[j] > a[j + 1]) {
38                 // 3
39                 int aux = a[j]; // 1
40                 a[j] = a[j + 1]; // 1
41                 a[j + 1] = aux; // 1
42             }
43         }
44     }
45     // n^2 + n + 4
46 }
```

Name	Type	Value
this	AlgoritmosDeOrdenamiento	#66
a	int[]	#67(length=3)
a	int[]	#67(length=3)
a[2]	int	5
a[1]	int	1
a[0]	int	9

**Paso 2:** Se inicializan las variables de los for's y se comprueba si se cumple la condición:

```
34     for (int i = 0; i < a.length - 1; i++) { // (n - 1)
35         // 1
36         for (int j = 0; j < a.length - 1 - i; j++) { //(n - 1 - i)
37             // 1
38             if (a[j] > a[j + 1]) {
39                 // 3
40                 int aux = a[j]; // 1
41                 a[j] = a[j + 1]; // 1
42                 a[j + 1] = aux; // 1
43             }
44         }
45     }
46     // n^2 + n + 4
47 }
```

Name	Type	Value
this	AlgoritmosDeOrdenamiento	#66
a	int[]	#67(length=3)
j	int	0
i	int	0
a	int[]	#67(length=3)
a[2]	int	5
a[1]	int	1
a[0]	int	9

**Paso 3:** Se crea la variable aux con un valor de 9 ya que  $a[j]$  es menor a  $a[j+1]$  la variable toma el valor de  $a[j]$  que es de 9:

```

// 1
for (int i = 0; i < a.length - 1; i++) { // (n - 1)
    // 1
    for (int j = 0; j < a.length - 1 - i; j++) { //(n - 1 - i)
        // 1
        if (a[j] > a[j + 1]) {
            // 3
            int aux = a[j]; // 1
            a[j] = a[j + 1]; // 1
            a[j + 1] = aux; // 1
        }
    }
} // n^2 + n + 4

/**
 * Metoda para mostrar el arreglo en las pruebas
 */

```

Name	Type	Value
this	AlgoritmosDeOrdenamiento	#66
a	int[]	#67(length=3)
j	int	0
i	int	0
aux	int	9
a[2]	int	5
a[1]	int	1
a[0]	int	9

**Paso 4:** Aquí se realizan varias sobre escrituras,  $a[j]$  pasa a tener el valor de  $a[j+1]$  y  $a[j+1]$  pasa a tener el valor del auxiliar, esto provoca que el índice [0] del arreglo pase a tener el valor de 1:

```

public void burbuja(int[] a) {
    // 1
    for (int i = 0; i < a.length - 1; i++) { // (n - 1)
        // 1
        for (int j = 0; j < a.length - 1 - i; j++) { //(n - 1 - i)
            // 1
            if (a[j] > a[j + 1]) {
                // 3
                int aux = a[j]; // 1
                a[j] = a[j + 1]; // 1
                a[j + 1] = aux; // 1
            }
        }
    } // n^2 + n + 4
}

```

Name	Type	Value
this	AlgoritmosDeOrdenamiento	#66
a	int[]	#67(length=3)
j	int	0
i	int	0
aux	int	9
a[2]	int	5
a[1]	int	1
a[0]	int	1

**Paso 5:** Se refleja el cambio de valores, como se mencionó antes: el índice [0] y [1] intercambiaron valor. Se vuelve a ejecutar el for para realizar otra iteración:

```
31 public void burbuja(int[] a) {
32     // 1
33     for (int i = 0; i < a.length - 1; i++) { // (n - 1)
34         // 1
35         for (int j = 0; j < a.length - 1 - i; j++) { //(n - 1 - i)
36             // 1
37             if (a[j] > a[j + 1]) {
38                 // 3
39                 int aux = a[j]; // 1
40                 a[j] = a[j + 1]; // 1
41                 a[j + 1] = aux; // 1
42             }
43         }
44     }
45     // n^2 + n + 4
46 }
```

algoritmos.AlgoritmosDeOrdenamiento > burbuja > for (int i = 0; i < a.length - 1; i++) >

Name	Type	Value
this	AlgoritmosDeOrdenamiento	#66
j	int	0
i	int	0
a	int[]	#67(length=3)
a[2]	int	5
a[1]	int	9
a[0]	int	1

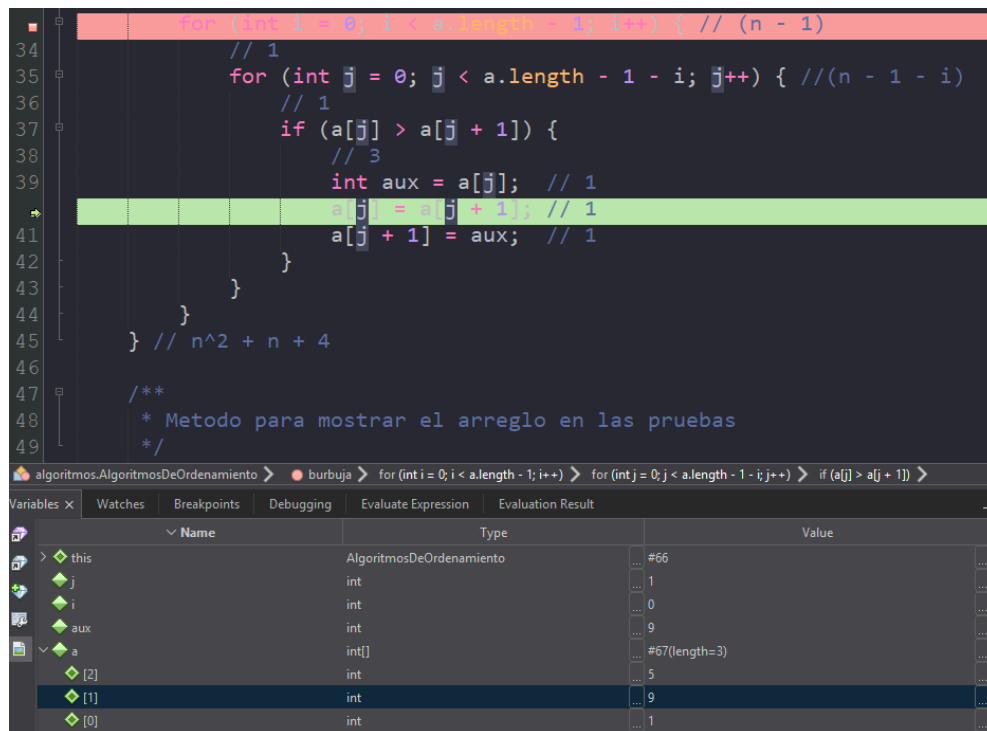
**Paso 6:** El valor de j del for sube a 1. Se vuelve a comprobar la condición.

```
32     // 1
33     for (int i = 0; i < a.length - 1; i++) { // (n - 1)
34         // 1
35         for (int j = 0; j < a.length - 1 - i; j++) { //(n - 1 - i)
36             // 1
37             if (a[j] > a[j + 1]) {
38                 // 3
39                 int aux = a[j]; // 1
40                 a[j] = a[j + 1]; // 1
41                 a[j + 1] = aux; // 1
42             }
43         }
44     }
45     // n^2 + n + 4
46 }
```

algoritmos.AlgoritmosDeOrdenamiento > burbuja > for (int i = 0; i < a.length - 1; i++) >

Name	Type	Value
this	AlgoritmosDeOrdenamiento	#66
j	int	1
i	int	0
a	int[]	#67(length=3)
a[2]	int	5
a[1]	int	9
a[0]	int	1

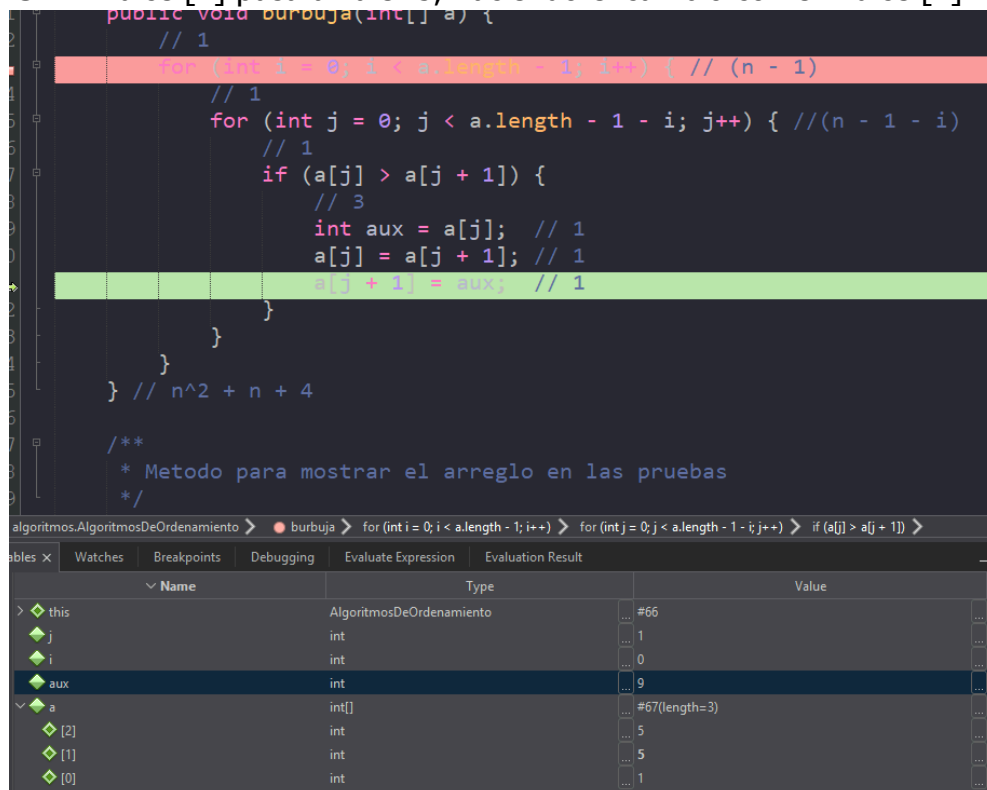
**Paso 7:** Vuelve a pasar lo mismo que el paso 3, se crea la variable aux con un valor de 9.



```
34 for (int i = 0; i < a.length - 1; i++) { // (n - 1)
35     // 1
36     for (int j = 0; j < a.length - 1 - i; j++) { //(n - 1 - i)
37         // 1
38         if (a[j] > a[j + 1]) {
39             // 3
40             int aux = a[j]; // 1
41             a[j] = a[j + 1]; // 1
42             a[j + 1] = aux; // 1
43         }
44     }
45 } // n^2 + n + 4
46
47 /**
48  * Metodo para mostrar el arreglo en las pruebas
49  */
```

Name	Type	Value
this	AlgoritmosDeOrdenamiento	#66
j	int	1
i	int	0
aux	int	9
a	int[]	#67(length=3)
a[2]	int	5
a[1]	int	9
a[0]	int	1

**Paso 8:** El indice [1] pasa a valer 5, haciendo el cambio con el indice [2].



```
2 public void burbuja(int[] a) {
3     // 1
4     for (int i = 0; i < a.length - 1; i++) { // (n - 1)
5         // 1
6         for (int j = 0; j < a.length - 1 - i; j++) { //(n - 1 - i)
7             // 1
8             if (a[j] > a[j + 1]) {
9                 // 3
10                int aux = a[j]; // 1
11                a[j] = a[j + 1]; // 1
12                a[j + 1] = aux; // 1
13            }
14        }
15    } // n^2 + n + 4
16
17    /**
18     * Metodo para mostrar el arreglo en las pruebas
19     */
20 }
```

Name	Type	Value
this	AlgoritmosDeOrdenamiento	#66
j	int	1
i	int	0
aux	int	9
a	int[]	#67(length=3)
a[2]	int	5
a[1]	int	5
a[0]	int	1

**Paso 9:** Los índices [1] y [2] intercambian de valor y se vuelve a ejecutar el for.

```

34  for (int i = 0; i < a.length - 1; i++) { // (n - 1)
35  // 1
36  for (int j = 0; j < a.length - 1 - i; j++) { //(n - 1 - i)
37  // 1
38  if (a[j] > a[j + 1]) {
39  // 3
40  int aux = a[j]; // 1
41  a[j] = a[j + 1]; // 1
42  a[j + 1] = aux; // 1
43  }
44  }
45  } // n^2 + n + 4
46
47  /**
48   * Metodo para mostrar el arreglo en las pruebas
49   */

```

Name	Type	Value
this	AlgoritmosDeOrdenamiento	#66
j	int	1
i	int	0
a	int[]	#67(length=3)
a[2]	int	9
a[1]	int	5
a[0]	int	1

**Paso 10:** Se cierra el 2do for del método.

```

34  for (int i = 0; i < a.length - 1; i++) { // (n - 1)
35  // 1
36  for (int j = 0; j < a.length - 1 - i; j++) { //(n - 1 - i)
37  // 1
38  if (a[j] > a[j + 1]) {
39  // 3
40  int aux = a[j]; // 1
41  a[j] = a[j + 1]; // 1
42  a[j + 1] = aux; // 1
43  }
44  }
45  } // n^2 + n + 4
46
47  /**
48   * Metodo para mostrar el arreglo en las pruebas
49   */

```

Name	Type	Value
this	AlgoritmosDeOrdenamiento	#66
j	int	0
i	int	0
a	int[]	#67(length=3)
a[2]	int	9
a[1]	int	5
a[0]	int	1

**Paso 11:** El valor del contador i sube a 1.

```

public void burbuja(int[] a) {
    // 1
    for (int i = 0; i < a.length - 1; i++) { // (n - 1)
        // 1
        for (int j = 0; j < a.length - 1 - i; j++) { //(n - 1 - i)
            // 1
            if (a[j] > a[j + 1]) {
                // 3
                int aux = a[j]; // 1
                a[j] = a[j + 1]; // 1
                a[j + 1] = aux; // 1
            }
        }
    } // n^2 + n + 4 S

    /**
     * Metodo para mostrar el arreglo en las pruebas
     */
}

```

Name	Type	Value
this	AlgoritmosDeOrdenamiento	#66
i	int	1
a	int[]	#67(length=3)
a[2]	int	9
a[1]	int	5
a[0]	int	1

**Paso 12:** se vuelve a abrir el 2do for para volver a hacer una iteracion y se comprueba la condicion.

```

31 public void burbuja(int[] a) {
32     // 1
33     for (int i = 0; i < a.length - 1; i++) { // (n - 1)
34         // 1
35         for (int j = 0; j < a.length - 1 - i; j++) { //(n - 1 - i)
36             // 1
37             if (a[j] > a[j + 1]) {
38                 // 3
39                 int aux = a[j]; // 1
40                 a[j] = a[j + 1]; // 1
41                 a[j + 1] = aux; // 1
42             }
43         }
44     }
45 } // n^2 + n + 4 S
46
47 /**
48  * Metodo para mostrar el arreglo en las pruebas
49  */

```

Name	Type	Value
this	AlgoritmosDeOrdenamiento	#66
j	int	0
i	int	1
a	int[]	#67(length=3)
a[2]	int	9
a[1]	int	5
a[0]	int	1

**Paso 13:** Al ya estar ordenada la lista, se vuelve a cerrar el for.

```
public void burbuja(int[] a) {
    // 1
    for (int i = 0; i < a.length - 1; i++) { // (n - 1)
        // 1
        for (int j = 0; j < a.length - 1 - i; j++) { //(n - 1 - i)
            // 1
            if (a[j] > a[j + 1]) {
                // 3
                int aux = a[j]; // 1
                a[j] = a[j + 1]; // 1
                a[j + 1] = aux; // 1
            }
        }
    } // n^2 + n + 4 S

    /**
     * Metodo para mostrar el arreglo en las pruebas
     */
}
```

algoritmos.AlgoritmosDeOrdenamiento > burbuja >

Name	Type	Value
this	AlgoritmosDeOrdenamiento	#66
i	int	1
a	int[]	#67(length=3)
a[2]	int	9
a[1]	int	5
a[0]	int	1

**Paso 14:** Se acaba el ciclo del método de ordenamiento.

```
public void burbuja(int[] a) {
    // 1
    for (int i = 0; i < a.length - 1; i++) { // (n - 1)
        // 1
        for (int j = 0; j < a.length - 1 - i; j++) { //(n - 1 - i)
            // 1
            if (a[j] > a[j + 1]) {
                // 3
                int aux = a[j]; // 1
                a[j] = a[j + 1]; // 1
                a[j + 1] = aux; // 1
            }
        }
    } // n^2 + n + 4 S

    /**
     * Metodo para mostrar el arreglo en las pruebas
     */
}
```

algoritmos.AlgoritmosDeOrdenamiento > burbuja >

Name	Type	Value
this	AlgoritmosDeOrdenamiento	#66
a	int[]	#67(length=3)
a[2]	int	9
a[1]	int	5
a[0]	int	1