TA1

Ejercicio 1)

1.

Para i =0 hasta nElementos hacer:

pos <- i

aux <- arreglo[i]

mientras pos > 0 y arreglo[pos - 1] > aux hacer

arreglo[pos] <- arreglo[pos - 1]

pos—

fin mientras

arreglo[pos] <- aux

Fin Para

2.

Comparaciones = 34

Movimientos = 27

256 - 458 - 655 - 298 - 043 - 648 - 778 - 621 - 655 - 019 - 124 – 847

256 – **298** – 458 – 655- 043 - 648 - 778 - 621 - 655 - 019 - 124 – 847

**043** - 256 – 298 – 458 – 655 - 648 - 778 - 621 - 655 - 019 - 124 – 847

043 - 256 – 298 – 458 - **648** - 655 - 778 - 621 - 655 - 019 - 124 – 847

043 - 256 – 298 – 458 - **621** - 648 - 655 - 778 - 655 - 019 - 124 – 847

043 - 256 – 298 – 458 - 621 - 648 - 655 - **655** - 778 - 019 - 124 – 847

**019** - 043 - 256 – 298 – 458 - 621 - 648 - 655 - 655 - 778 - 124 – 847

019 - 043 - **124** - 256 – 298 – 458 - 621 - 648 - 655 - 655 - 778 – 847

3.

c

a

d

Ejercicio 2)

1.

Comparaciones: 44

Movimientos: 14

**256** - 458 - 655 - 298 - 043 - **648** - 778 - 621 - 655 - 019 - 124 – 847

256 - **458** - 655 - 298 - 043 - 648 - **778** - 621 - 655 - 019 - 124 – 847

256 - 458 - **655** - 298 - 043 - 648 - 778 - **621** - 655 - 019 - 124 – 847

256 - 458 - **621** - 298 - 043 - 648 - 778 - **655** - 655 - 019 - 124 – 847

256 - 458 - 621 - **298** - 043 - 648 - 778 - 655 - **655** - 019 - 124 – 847

256 - 458 - 621 - 298 - **043** - 648 - 778 - 655 - 655 - **019** - 124 – 847

256 - 458 - 621 - 298 - **019** - 648 - 778 - 655 - 655 - **043** - 124 – 847

256 - 458 - 621 - 298 - 019 - **648** - 778 - 655 - 655 - 043 - **124** – 847

256 - 458 - 621 - 298 - 019 - **124** - 778 - 655 - 655 - 043 - **648** – 847

**256** - 458 - 621 - 298 - 019 - **124** - 778 - 655 - 655 - 043 - 648 – 847

**124** - 458 - 621 - 298 - 019 - **256** - 778 - 655 - 655 - 043 - 648 – 847

124 - 458 - 621 - 298 - 019 - 256 - **778** - 655 - 655 - 043 - 648 – **847**

**---------------------------------------------------------------------------------**

**124** - 458 - 621 - 298 - **019** - 256 - 778 - 655 - 655 - 043 - 648 – 847

**019** - 458 - 621 - 298 - **124** - 256 - 778 - 655 - 655 - 043 - 648 – 847

019 - **458** - 621 - 298 - 124 - **256** - 778 - 655 - 655 - 043 - 648 – 847

019 - **256** - 621 - 298 - 124 - **428** - 778 - 655 - 655 - 043 - 648 – 847

019 - 256 - **621** - 298 - 124 - 428 - **778** - 655 - 655 - 043 - 648 – 847

019 - 256 - 621 - **298** - 124 - 428 - 778 - **655** - 655 - 043 - 648 – 847

019 - 256 - 621 - 298 - **124** - 428 - 778 - 655 - **655** - 043 - 648 – 847

019 - 256 - 621 - 298 - 124 - **428** - 778 - 655 - 655 - **043** - 648 – 847

019 - **256** - 621 - 298 - 124 - **043** - 778 - 655 - 655 - 428 - 648 – 847

019 - **043** - 621 - 298 - 124 - **256** - 778 - 655 - 655 - 428 - 648 – 847

019 - 043 - 621 - 298 - 124 - 256 - **778** - 655 - 655 - 428 - **648** – 847

019 - 043 - 621 - 298 - 124 - 256 - **648** - 655 - 655 - 428 - **778**– 847

019 - 043 - **621** - 298 - 124 - 256 - **648** - 655 - 655 - 428 - 778– 847

019 - 043 - 621 - 298 - 124 - 256 - 648 - **655** - 655 - 428 - 778– **847**

**--------------------------------------------------------------------------------**

**019** - 043 - **621** - 298 - 124 - 256 - 648 - 655 - 655 - 428 - 778– 847

019 - **043** - 621 - **298** - 124 - 256 - 648 - 655 - 655 - 428 - 778– 847

019 - 043 - **621** - 298 - **124** - 256 - 648 - 655 - 655 - 428 - 778– 847

019 - 043 - **124** - 298 - **621** - 256 - 648 - 655 - 655 - 428 - 778– 847

**019** - 043 - **124** - 298 - **621** - 256 - 648 - 655 - 655 - 428 - 778– 847

019 - 043 - 124 - **298** - 621 - **256** - 648 - 655 - 655 - 428 - 778– 847

019 - 043 - 124 - **256** - 621 - **298** - 648 - 655 - 655 - 428 - 778– 847

019 - **043** - 124 - **256** - 621 - **298** - 648 - 655 - 655 - 428 - 778– 847

019 - 043 - 124 - 256 - **621** - 298 - **648** - 655 - 655 - 428 - 778– 847

019 - 043 - 124 - 256 - 621 - **298** - 648 - **655** - 655 - 428 - 778– 847

019 - 043 - 124 - 256 - 621 - 298 - **648** - 655 - **655** - 428 - 778– 847

019 - 043 - 124 - 256 - 621 - 298 - 648 - **655** - 655 - **428** - 778– 847

019 - 043 - 124 - 256 - 621 - **298** - 648 - **428** - 655 - **655** - 778– 847

019 - 043 - 124 - 256 - 621 - 298 - 648 - 428 - **655** - 655 - **778**– 847

019 - 043 - 124 - 256 - 621 - 298 - 648 - 428 - 655 - **655** - 778– **847**

**--------------------------------------------------------------------------------**

**019 - 043** - 124 - 256 - 621 - 298 - 648 - 428 - 655 - 655 - 778– 847

019 - **043 - 124** - 256 - 621 - 298 - 648 - 428 - 655 - 655 - 778– 847

019 - 043 - **124 - 256** - 621 - 298 - 648 - 428 - 655 - 655 - 778– 847

019 - 043 - 124 - **256 - 621** - 298 - 648 - 428 - 655 - 655 - 778– 847

019 - 043 - 124 - 256 - **621 - 298** - 648 - 428 - 655 - 655 - 778– 847

019 - 043 - 124 - 256 **- 298** - **621** - 648 - 428 - 655 - 655 - 778– 847

019 - 043 - 124 - **256** **- 298** - **621** - 648 - 428 - 655 - 655 - 778– 847

019 - 043 - 124 - 256 - 298 - **621** - **648** - 428 - 655 - 655 - 778– 847

019 - 043 - 124 - 256 - 298 - 621 - **648 - 428** - 655 - 655 - 778– 847

019 - 043 - 124 - 256 - 298 - 621 - **428 - 648** - 655 - 655 - 778– 847

019 - 043 - 124 - 256 - 298 - **621** - **428 - 648** - 655 - 655 - 778– 847

019 - 043 - 124 - 256 - **298** - **428** - **621 - 648** - 655 - 655 - 778– 847

019 - 043 - 124 - 256 - **298** - **428** - **621 - 648** - 655 - 655 - 778– 847

019 - 043 - 124 - 256 - 298 - 428 - 621 - **648 - 655** - 655 - 778– 847

019 - 043 - 124 - 256 - 298 - 428 - 621 - 648 - **655 - 655** - 778– 847

019 - 043 - 124 - 256 - 298 - 428 - 621 - 648 - 655 - **655 - 778**– 847

019 - 043 - 124 - 256 - 298 - 428 - 621 - 648 - 655 - 655 - **778– 847**

019 - 043 - 124 - 256 - 298 - 428 - 621 - 648 - 655 - 655 - 778– 847

2.

Orden O(N^2)

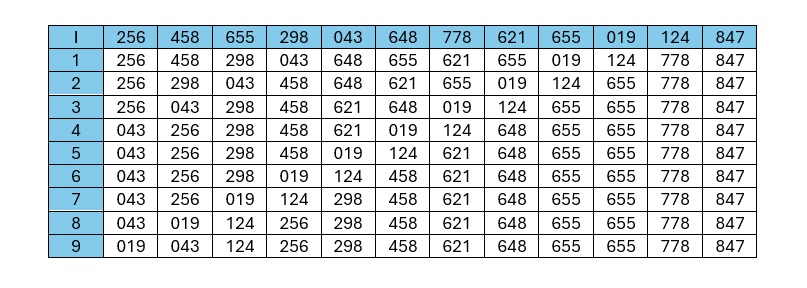
3.

b. Entre 10 y 20

Ejercicio 3)

1.





2.

O(N^2)

3.

Una forma de reducir el tiempo de ejecución del método de ordenación burbuja es implementar una bandera que verifique si se realizaron intercambios en la última pasada. Si no se hicieron intercambios, significa que el arreglo ya está ordenado y se puede terminar el algoritmo prematuramente sin necesidad de más iteraciones.

Ejercicio 4)

Parte 1)

¿Cómo se genera un vector monótonamente ascendente?

Se utiliza el método generarDatosAscendentes(), donde se asigna a cada posición del arreglo un valor creciente desde 0 hasta TAMANIO\_MAX - 1.

¿Cómo se genera un vector monótonamente descendente?

Se utiliza el método generarDatosDescendentes(), donde se asigna a cada posición del arreglo un valor decreciente desde TAMANIO\_MAX hasta 1.

¿Cómo se genera un vector con valores aleatorios? ¿Pueden existir claves repetidas? ¿Cuál es el orden del tiempo de ejecución de este método?

Se genera con generarDatosAleatorios(). Puede haber claves repetidas, pero se evitan usando un arreglo booleano para controlar qué posiciones ya se han utilizado. El orden de tiempo de ejecución es O(n)O(n)O(n), donde n es TAMANIO\_MAX.

¿Cuántos elementos contiene el vector de datos generado? ¿Cómo se puede modificar esta clase para que la cantidad de elementos del vector sea parametrizable?

Contiene TAMANIO\_MAX elementos. Para hacerlo parametrizable, se puede cambiar TAMANIO\_MAX a una variable tamanioMaximo, que se pasa como parámetro al constructor de la clase.

¿Cómo podemos verificar que un conjunto está ordenado? ¿Cuál sería el orden del tiempo de ejecución de un algoritmo que lo haga?

Se recorre el arreglo y se verifica que cada elemento sea menor o igual al siguiente (en el caso de orden ascendente). El orden de tiempo de ejecución es O(n)O(n)O(n), donde n es el tamaño del arreglo, ya que se debe revisar cada elemento al menos una vez.

Parte 2)

En la carpeta “Ta1”