INFORMÁTICA (30717)

Práctica 6: Resolución de problemas en Java (III)

Grado en Estudios en Arquitectura

Universidad de Zaragoza

Material de prácticas del curso 2025–2026

Universidad Zaragoza

Tabla de contenidos

1. Objetivos de la práctica

Los objetivos de la sexta práctica de la asignatura son los siguientes:

- Desarrollar algoritmos en Java.
- Ejecutar algoritmos en el entorno *Eclipse*.

2. Algoritmos de ordenación

Dado un vector de n números enteros, el objetivo de esta práctica es implementar dos algoritmos de ordenación: el algoritmo de **ordenación de burbuja** (*bubble sort*) y el algoritmo de **selección** (*selection sort*). Ambos algoritmos son métodos sencillos para ordenar una lista de elementos, aunque no son los más eficientes para grandes conjuntos de datos. Sin embargo, son útiles para entender los conceptos básicos de ordenación y complejidad algorítmica.

3. *Bubble sort* u ordenación de burbuja

El algoritmo de **ordenación de burbuja** funciona **comparando cada par de elementos adyacentes** e **intercambiándolos si están en el orden incorrecto**. Este proceso se repite hasta que la lista está completamente ordenada.

Para un vector de n números enteros, el proceso es:

- Comparamos $v[0]$ con $v[1]$ y, si es mayor, los intercambiamos.
- Repetimos la comparación con cada par de elementos adyacentes hasta llegar a $v[n-2]$ y $v[n-1]$.
- En ese momento, la última posición ya está ordenada.
- Repetimos el proceso hasta tener todo el vector ordenado.
- Si en un recorrido completo no hay ningún intercambio, el vector ya está ordenado.

Figura 1: Idea visual de una iteración de ordenación de burbuja. La versión web y las diapositivas permiten avanzar paso a paso.

4. *Selection sort* u ordenación por selección

El algoritmo de ordenación por **selección** funciona **dividiendo la lista en dos partes**: la parte **ordenada** y la parte **no ordenada**. En cada iteración, el algoritmo selecciona el elemento más pequeño de la parte no ordenada y lo intercambia con el primer elemento de esa parte.

El proceso es:

- Buscamos el elemento más pequeño en el vector y lo intercambiamos con el primer elemento.
- Después buscamos el elemento más pequeño en el resto del vector y lo intercambiamos con el segundo elemento.
- Repetimos este proceso hasta que todo el vector esté ordenado.

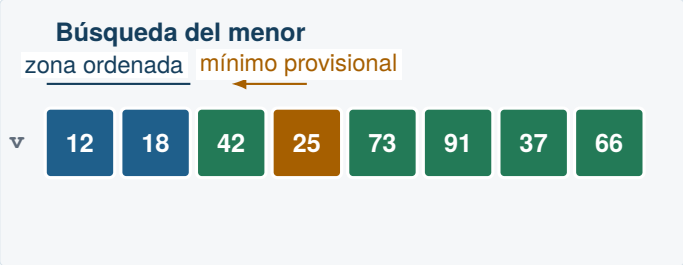

Figura 2: Idea visual de una iteración de ordenación por selección. La versión web y las diapositivas muestran las comparaciones paso a paso.

5. Ejercicios

Implementa los algoritmos de ordenación de burbuja y selección en Java utilizando dos clases diferentes, `OrdenacionBurbuja.java` y `OrdenacionSeleccion.java`. La siguiente estructura puede servir como punto de partida para la clase de burbuja:

```
import java.util.*;

public class OrdenacionBurbuja {
  public static void ordenar(int[] v) {
    // Tu codigo aqui.
    // Recuerda que puedes acceder al tamano del vector
    // con v.length y a cada elemento con v[i].
  }

  public static void main(String[] args) {
    Random r = new Random();
    Scanner sc = new Scanner(System.in);

    System.out.print("Dimension de la lista de numeros: ");
    int size = sc.nextInt();

    int[] v = new int[size];
    for (int i = 0; i < size; i++) {
      int x = r.nextInt();
      x = Math.abs(x) % 50;
      v[i] = x;
    }

    System.out.println("Vector inicial");
    for (int i = 0; i < size; i++)
      System.out.print(v[i] + "\t");

    ordenar(v);

    System.out.println();
```

```
    System.out.println("Vector ordenado");
    for (int i = 0; i < size; i++)
      System.out.print(v[i] + "\t");
  }
}
```

Ejecuta paso a paso los algoritmos utilizando el depurador de *Eclipse* para entender cómo funciona el proceso de ordenación. Utiliza un tamaño de vector pequeño, por ejemplo 20 elementos, para facilitar la visualización del proceso. Además, debes **medir el tiempo de ejecución** de los algoritmos utilizando cuatro valores de n: 10^2, 10^3, 10^4 y 10^5.

5.1. Tiempos de ejecución

n	Burbuja	Selección
100	0	0
1 000	3	3
10 000	120	83
100 000	11 032	1 106

Cuadro 1: Tabla de tiempos de ejecución en milisegundos de los algoritmos de ordenación.

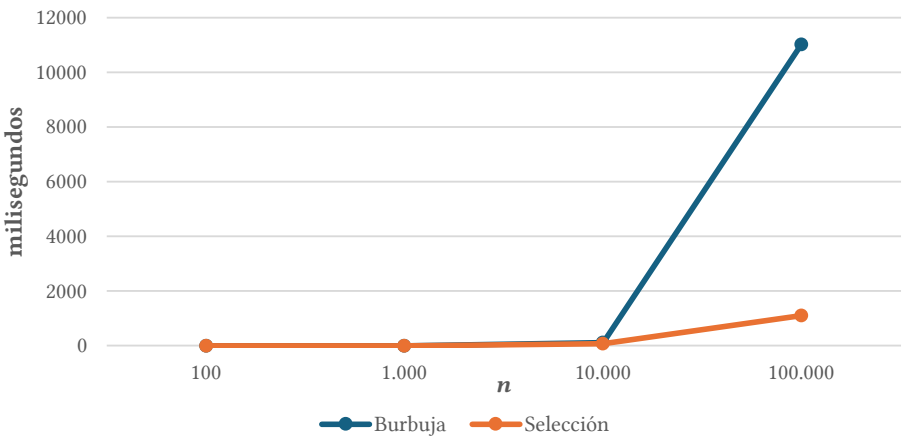

Figura 3: Comparación de tiempos de ejecución de los algoritmos de burbuja y selección.

Posible falta de memoria

Para valores de n muy grandes, es posible que el programa no tenga suficiente memoria para ejecutar el algoritmo de ordenación de burbuja. Puedes aumentar la memoria asignada a Java desde `Run > Run Configurations...`, pestaña `Arguments`, añadiendo `-Xmx1024m` en `VM arguments`.

5.2. Medición de tiempo de ejecución

Para medir el tiempo de ejecución de cada implementación, utiliza la clase `System` de Java antes y después de ejecutar el algoritmo:

```
long startTime = System.currentTimeMillis();

ordenar(v);

long endTime = System.currentTimeMillis();
long duration = endTime - startTime;
System.out.println("Tiempo de ejecucion: " + duration + " milisegundos");
```

6. Entrega de la solución

Sólo debe realizar la entrega un miembro del grupo. Sube a Moodle un fichero comprimido `practica6.zip` que contenga:

- `OrdenacionBurbuja.java`: implementación del algoritmo de ordenación de burbuja.
- `OrdenacionSeleccion.java`: implementación del algoritmo de ordenación por selección.
- `tiempos.xlsx`: archivo de Excel con los tiempos de ejecución medidos para ambos algoritmos y los diferentes tamaños de vector.