Práctica 1: Evaluación de algoritmos de ordenación

En esta práctica hay que implementar diferentes algoritmos de ordenación y estudiar su complejidad computacional. Para ello se suministran los dos archivos siguientes:

- **Java.zip**: código fuente con todo lo necesario para implementar algoritmos de ordenación de listas y lanzar experimentos para evaluar los tiempos de ejecución de estos algoritmos en función del número de elementos.

  - La clase `SortingAlgorithm` es una superclase de la que heredan los algoritmos de ordenación a implementar.

  - Las clases `BubbleSort`, `InsertionSort` y `SelectionSort` heredan de la clase `SortingAlgorithm` y corresponden a los algoritmos de ordenación por el método de la burbuja, por inserción y por selección, respectivamente. **Hay que implementar** el método `sort(List<E> A, int l, int r)` de cada algoritmo, siendo `l` y `r` los índices de los elementos extremos (izquierdo y derecho) de la lista a ordenar `A`.[1]

  - La clase `TestSortingAlgorithms` permite ejecutar un algoritmo de ordenación sobre un conjunto de números aleatorios. Esta clase tiene un método `Main` (ejecutable) que recibe los siguientes parámetros:

    - Método de ordenación: 'bubble', 'insertion' o 'selection'.
    - Número de elementos inicial de la lista.
    - Número de veces a duplicar el número de elementos.
    - Número de veces que se repite cada ordenación (para estadísticas).

    Por ejemplo, con los parámetros "selection, 2000, 3, 10" el `Main` ejecuta el algoritmo de ordenacion por selección para 2000, 4000 y 8000 elementos alelatorios. Esto se repite 10 veces para cada uno de estos tamaños.

    Como salida, este método genera un fichero de texto que contiene los tiempos de ejecución del algoritmo de ordenación pasado como parámetro.

- **Matlab_Octave.zip**: código Matlab (compatible con Octave) para (1) leer los ficheros generados por el método `Main` de la clase `TestSortingAlgorithms`, (2) realizar un ajuste a n^2 o nlogn y (3) representarlo todo gráficamente para facilitar el análisis de la complejidad de estos algoritmos. **Ejecutar el script analize1.m** una vez ejecutados los algoritmos de ordenación implementados.

IMPORTANTE:

- Entregar al final de clase el código de los algoritmos implementados (tarea en CV).
- Hay que entregar una memoria final en los próximos días (consultar CV).

1 Para validar la correcta implementación de los algoritmos, pasar a la VM de Java el argumento "-ea".