Taller en clase 1

Objetivos

1. Implementar y comparar en la práctica algoritmos de ordenamiento con diferentes complejidades
2. Entender el patrón de diseño algoritmo (o estrategia) y la introspección
3. Practicar ejecución en linea de comandos de programas en java

Ordenamiento de arreglos

1. Descargar el archivo zip adjunto, descomprimirlo e importarlo como un proyecto de Eclipse. Explorar las clases y las interfaces contenidas en el proyecto.

2. Desarrollar las tres clases que implementen la interfaz NumbersArraySorter para ordenar un arreglo dado de números enteros implementando los siguientes algoritmos:

- BubbleNumbersArraySorter: Ordenamiento burbuja (Bubble Sort)
- MergeNumbersArraySorter: Ordenamiento por mezcla (Merge Sort)
- QuickNumbersArraySorter: Ordenamiento rápido (Quick Sort)

3. Utilizar el programa disponible en la clase NumbersSortingExample para probar los algoritmos y compararlos con el algoritmo de ordenamiento ya disponible en java. El programa recibe los siguientes parámetros:

args[0]: Ruta al archivo con los números a ordenar
args[1]: Algoritmo utilizado para ordenar. Puede ser “Bubble”, “Merge” o “Quick”. Si no se provee un algoritmo, el programa utiliza por defecto el algoritmo implementado en la clase java.util.Arrays.

Probar los diferentes algoritmos con listas de 100, 1000, 10000, 100000 y un millon de números. Como ejemplo de archivo de entrada se provee una lista de 100000 números reales aleatorios en el archivo data/numbers.txt. Generar una tabla con el tiempo que necesitó cada algoritmo para ordenar los números en cada caso.

Extra para la casa: Implementar también los algoritmos radix sort y count sort, los cuales permiten ordenar números naturales en tiempo lineal. Comparar los tiempos de ejecución con los obtenidos por merge sort y quick sort. Una animación del

funcionamiento de estos algoritmos se puede encontrar en este enlace: https://visualgo.net/en/sorting