Practica Opcional 1 Estructura de Datos y Algoritmos

*Algoritmos de ordenación*

**Nombre de los integrantes del grupo:**

* Daniel Serrano Torres
* Juan Luis Pérez Valbuena

**Instrucciones para compilar el código**

Se ha utilizado un entorno de desarrollo integrado como NetBeans. De todas maneras puede ser compilado mediante el uso del compilador gcc .

Las únicas dependencias son las propias al uso de las librerías internas del propio lenguaje C++.

**Método empleado para obtener los tiempos y generar las gráficas**

Mediante la salida del programa, se ha trasladado los datos a Microsoft Excel y con la utilidad de gráficas que dispone se han realizado las tres gráficas.

**Análisis de gráficas**

Como era de esperar, los algoritmos de orden (n log n) tienen unos tiempos significativamente **más bajos** que los n^2 , lógicamente por el ‘número de accesos’ que realizan sobre los vectores.

Entre los de O(n^2), se encuentra que el algoritmo de la **burbuja** crece bastante más rápido que los demás. Esto es debido a que su caso mejor y peor coincide en orden asintótico.

En el caso peor sería si le pasáramos un vector en orden inverso y el mejor sería si se lo pasáramos ya ordenado. En cualquier caso nos da un orden O(n^2).

En el algoritmo de ordenación por **inserción** el caso peor y mejor coinciden (entrada en orden inverso como peor caso y entrada ordenada como mejor caso), solo que en el mejor caso el bucle interno no se ejecuta así que quedaría un O(n). En el peor caso deberá realizar el máximo de iteraciones en el bucle interno, esto nos queda en un O(n^2).

En el algoritmo de **selección** tiene solo un caso, porque la comparación interna se ejecuta siempre más o menos el mismo número de veces. Caso mejor, caso promedio y caso peor del O(n^2).

En el algoritmo **QuickSort** se basa en elegir una posición llamada pivote y colocar los elementos del array de forma que queden los mayores a la derecha y los menores a la izquierda. Dependiendo de cómo elijamos este pivote nos dará un orden de complejidad u otro.

En el mejor de los casos, el pivote queda en el centro de la lista, la divide en dos listas de igual tamaño, esto resulta en el orden O(n log n).

En el peor de los casos, la entrada viene ordenada, con lo cual elegirá el pivote en el último elemento y nos generara dos listas, una con todos los elementos y otra vacía, con orden O(n^2).

En el caso promedio el orden es de O(n log n).

El algoritmo **Mergesort** se basa en dividir el array en dos y ordenar cada parte de forma recursiva. Es el único algoritmo que requiere memoria adicional para ser ejecutado, por ello añade cierto tiempo a la ejecución. En cuanto al análisis de casos todos sus casos son de O(n log n).