Laboratorio 1 – Análisis empírico de costes de algoritmos de ordenación

GUIÓN DEL LABORATORIO

1.- Objetivos del laboratorio

- Comprensión de un programa en C++
- Diseño de bucles
- Análisis experimental del coste de un algoritmo
- Razonar sobre datos experimentales
- Implementar algoritmos de ordenación

2.- Antes de asistir al laboratorio

Antes de asistir al laboratorio debes realizar las siguientes tareas:

1. Leer alguna de las lecturas recomendadas y los apuntes de clase sobre análisis de algoritmos y algoritmos de ordenación de vectores.
2. Revisar el guión del laboratorio (este documento)
3. Analizar el código de "`lab1_act1.cpp`" que está en la página web de la asignatura.
4. Contestar a la cuestiones 1 a 6 de la Actividad 0 de tu hoja de trabajo. Recuerda que es preciso entregar las respuestas a esta actividad antes de la fecha prevista. En caso contrario, no podrás acceder al laboratorio ni entregar de ninguna manera la solución a esta práctica.

3.- Actividades a realizar

Actividad 1: Calcular empíricamente el coste del algoritmo de ordenación por Inserción mediante la ejecución del programa que se proporciona ya escrito.

1. Crear un nuevo directorio llamado `Lab1` en el escritorio
2. Copiar el archivo "`lab1_act1.cpp`" de la página web de la asignatura en el directorio `Lab1`.
3. Abrir el archivo "`lab1_act1.cpp`" con *DevC++*
4. Compilar el programa "`lab1_act1.cpp`" y verificar que no hay errores.
5. Ejecutar el programa para distintos tamaños del vector y completar la tabla 1.1 (cuestión 7) de la Actividad 1 de tu hoja de trabajo.

Actividad 2: Modificar el código del programa de la actividad 1 para que ordene vectores de distinto tamaño, de manera que en una sola ejecución del programa se puedan obtener los resultados necesarios para la tabla anterior.

1. Guardar el programa de la actividad anterior como "`lab1_act2.cpp`".
2. Introducir un bucle en el programa principal de "`lab1_act2.cpp`" de manera que se repitan las tareas del programa para los siguientes tamaños de vector: 10, 100, 1000, 10000, y 20000
3. Completar la tabla 2.1 (cuestión 8) de la Actividad 2 de tu hoja de trabajo.

Actividad 3: Para que los resultados sobre el número de operaciones realizadas por el algoritmo de ordenación cuando el vector está aleatoriamente ordenado (situación esperada más probable) sean más fiables, se debe modificar el programa anterior para que el resultado de asignaciones y comparaciones calculado para el vector generado aleatoriamente sea la media de los resultados

obtenidos tras ordenar 100 vectores de este tipo.

1. Guardar el programa de la actividad anterior como "`lab1_act3.cpp`".
2. Modifica el programa principal de "`lab1_act3.cpp`" para que, cuando la organización inicial del vector sea de tipo aleatorio (no en los otros dos casos) se repita 100 veces el proceso con 100 vectores diferentes y el resultado de asignaciones y comparaciones calculado sea la media de las 100 ordenaciones. (NOTA: Como para valores grandes de tamaño de vector se puede acumular un número elevado de asignaciones y comparaciones, se recomienda elegir `double` como tipo de datos adecuado para las variables encargadas de acumular las sumas necesarias para el cálculo de la media).
3. Completar la tabla 3.1 (cuestión 9) de la Actividad 3 de tu hoja de trabajo.

Actividad 4: Implementar 2 métodos más de ordenación, *Selección* y *Quicksort*

1. Guardar el programa de la actividad anterior como "`lab1_act4.cpp`".
2. Basándote en los apuntes de clase o en los libros recomendados, escribe dos funciones nuevas en el programa:
   a. `OrdenarSeleccion (Vector, int, int&, int&)`, que implementa el método de ordenación por Selección.
   b. `OrdenarQuicksort (Vector, int, int&, int&)`, que implementa el método de ordenación Quicksort.
3. Introduce en estas dos funciones contadores para las asignaciones y comparaciones de elementos del vector allí donde sea necesario (ver el código de `OrdenarInsercion`).
4. Modifica el programa principal de "`lab1_act4.cpp`" para que todos los vectores generados se ordenen siempre mediante los tres métodos de ordenación implementados. Para realizar correctamente este proceso, hay que tener la precaución de que al aplicar cualquiera de los métodos no se esté ordenando un vector previamente ordenado por otro método. Ya que se estaría ordenando un vector inicialmente ordenado y los resultados obtenidos serían incorrectos. La estrategia a seguir para evitar esta situación pasa por realizar copias de vectores, lo que implicará implementar una función que permita copiar un vector en otro.
5. Completar las tabla 4.1 a 4.5 (cuestión 10) de la Actividad 4 de tu hoja de trabajo.

Los archivos con los programas correspondientes a las actividades 2, 3 y 4 se deberán entregar, a través del aula virtual, al finalizar la sesión de prácticas, junto con la Hoja de Trabajo del estudiante.

5.- Después de asistir al laboratorio

Una vez escritos los programas y obtenidos resultados numéricos, es el momento de analizar e interpretar los valores obtenidos y verificar si el comportamiento práctico de los algoritmos se corresponde con el comportamiento teórico descrito en clase (y en la bibliografía). Para ello, contesta a las cuestiones 11 a 19 correspondientes a las Actividad 5 de tu hoja de trabajo.

6.- Lecturas recomendadas

- Capítulos 2 y 5 de "*Introducció a l'anàlisi i disseny d'algorismes*", F. Ferri, J. Albert, G. Martín. Servei de Publicacions de la Universitat de València (1998).
- Capítulo 12 de "*Programación en C++. Algoritmos, estructuras de datos y objetos (2ª ed.)*", L. Joyanes. McGrawHill (2006)