Universidad Nacional de San Agustín de Arequipa
Escuela Profesional de Ciencia de la Computación
Curso: Estructura de Datos Avanzados

LABORATORIO 1

Algoritmos y Costo Computacional

Docente: Rolando Jesús Cárdenas Talavera

1 Competencia del Curso

Comprende la importancia e impacto de los algoritmos estudiados y las nuevas propuestas.

2 Competencia del Laboratorio

- Describir, implementar y analizar algoritmos de ordenamiento.
- Interpretar el costo computacional en algoritmos de estudio.

3 Equipos y Materiales

- Un computador.
- Lenguaje de Programación (c++, python, java, c)

4 Marco Teórico

4.1 Algoritmos de ordenamiento

Debido a que las estructuras de datos son utilizadas para almacenar información, para poder recuperar esa información de manera eficiente es deseable que aquella esté ordenada. Existen varios métodos para ordenar las diferentes estructuras de datos básicas.

En general los métodos de ordenamiento no son utilizados con frecuencia, en algunos casos sólo una vez. Hay métodos muy simples de implementar que son útiles en los casos en donde el número de elementos a ordenar no es muy grande (ej, menos de 500 elementos). Por otro lado hay métodos sofisticados, más difíciles de implementar pero que son más eficientes en cuestión de tiempo de ejecución.

Los métodos sencillos por lo general requieren de aproximadamente $n \times n$ pasos para ordenar $n$elementos. Los métodos simples son: **insertion sort** (o por inserción directa) **selection sort**, **bubble sort**, y **shellsort**, en dónde el último es una extensión al insertion sort, siendo más rápido. Los métodos más complejos son el **quicksort**, el **heap sort**, **radix** y **address-calculation sort**. El ordenar un grupo de datos significa mover los datos o sus referencias para que queden en una secuencia tal que represente un orden, el cual puede ser numérico, alfabético o incluso alfanumérico, ascendente o descendente.

Se ha dicho que el ordenamiento puede efectuarse moviendo los registros con las claves. El mover un registro completo implica un costo, el cual se incrementa conforme sea mayor el tamaño del registro. Es por ello que es deseable evitar al máximo el movimiento de los registros. Una alternativa es el crear una tabla de referencias a los registros y mover las referencias y no los datos. A continuación se mostrarán los métodos de ordenamiento empezando por el más sencillo y avanzando hacia los más sofisticados.

La eficiencia de los algoritmos se mide por el número de comparaciones e intercambios que tienen que hacer, es decir, se toma n como el número de elementos que tiene el arreglo a ordenar y se dice que un algoritmo realiza $O(n^2)$ comparaciones cuando compara n veces los n elementos.

4.2 Costo computacional

El primer paso para resolver un problema informático dado es encontrar un algoritmo que lo resuelva. Posterior a ello nos surge la interrogantes si existen "mejores" algoritmos para su solución.

Se sabe que un algoritmo posee una sucesión finita de operaciones elementales, que ejecutará sobre una serie de datos. Un algoritmo, será mejor que otro en tiempo si, actuando sobre los mismos datos, el tiempo de ejecución es menor, y será mejor que otro en espacio sí, actuando sobre los mismos datos, la memoria, principal o secundaria, que utiliza es menor. Pero, tanto el tiempo como el espacio utilizado dependen de la máquina en la que ejecutemos el algoritmo, por tanto hay que determinar una forma para estudiar la eficiencia de los algoritmos independiente de la máquina que se utilice.

Por lo cual una forma correcta de comparar algoritmos será contar el número de operaciones que realiza en su ejecución, calculando su complejidad computacional.

5 Actividad

1. Utilizando los archivos adjuntos (*DataGen1*, *DataGen05*, *DataGen025*), utilice los datos para las pruebas de los algoritmos de ordenamiento. Tenga en cuenta la cantidad de datos de cada uno.
2. Implemente los siguientes algoritmos:
   - Bubble sort
   - Heap sort
   - Insertion sort
   - Selection sort
   - Shell sort
   - Merge sort
   - Quick sort
3. Analizar la complejidad computacional de cada uno.
4. Evaluar y comparar sus algoritmos usando los archivos de datos y elabore una(s) gráfica(s) comparativa(s). De utilizar c++, mida el tiempo de ejecución con la función *std::chrono::high_resolution_clock::now();*

6 Entregables

Al finalizar el estudiante deberá:

- Elaborar un documento, en donde se registre los algoritmos elaborados, el análisis realizado y las gráficas elaborados
- Deberá de incluir el código en formato de texto (no coloque imágenes de los códigos empleados)
- Deberán de subir a la plataforma Classroom el documento elaborado en **formato PDF** (se recomienda el uso de *LaTeX*) y los códigos elaborados.
- **IMPORTANTE** En caso de copia o plagio o similares todos los alumnos implicados tendrán sanción en toda la evaluación del curso.