Laboratorio #1: Algoritmos de ordenación

Jenny Johanna Vargas Oliveros

Escuela Superior de Ingeniería y Tecnología, Fundación Universitaria Internacional de la Rioja

Ingeniería informática

Algoritmia y Complejidad

Rafael Lizcano Reyes

20 de septiembre 2024

Introducción:

Este laboratorio está destinado a poner en práctica el diseño e implementación de clases en Java y el diseño e implementación de interfaces de usuario. Se trabajará con AWT/SWING.

Desarrollo del Laboratorio

Vamos a realizar el análisis empírico de algunos algoritmos de ordenación. Para medir el tiempo de ejecución y número de operaciones básicas de los algoritmos tenemos que analizar el código y hacer los ajustes que correspondan.

Archivos recibidos:

- «Entrada-800.txt»
- «Entrada-8000.txt»
- «Entrada-80000.txt»

Implementación:

- Python

1. En el siguiente informe se da respuesta a las siguientes preguntas:

A. ¿Funciona bien el programa?
   - Sí, no obstante al momento de hacer el análisis con la función burbuja en «Entrada-80000.txt» tarda más tiempo en responder.
   - Tanto las funciones burbuja, inserción y selección son menos eficientes, especialmente con grandes conjuntos de datos.

B. ¿Qué fallos existen?
   - Al momento de hacer la ejecución con entrada burbuja en «Entrada-80000.txt» el algoritmo tarda más en ejecutar la orden por lo que se estima menos eficiente en grandes volúmenes de datos a analizar «Entrada-80000.txt».

C. ¿Cuál fue la mayor dificultad en el desarrollo de las funciones?

- Los tiempos de espera ejecutando los algoritmos (burbuja, selección e inserción) ya que tardan más al procesar altos volúmenes de datos
- Realizar el código para que corriera adecuadamente y según los parámetros establecidos.
- Emplear las entradas correctas.

2. Se realiza la selección de una de las funciones de algoritmos de ordenación explicando la funcionalidad desarrollada.

Algoritmo Radix Sort:

Algoritmo de ordenación no comparativo que clasifica los números procesando sus dígitos de forma individual. Ordenación por conteo. A diferencia de los algoritmos que comparan valores como Quick Sort o Merge Sort, Radix Sort organiza los datos basándose en los dígitos que componen los números. Se utiliza principalmente para ordenar números enteros o cadenas de caracteres.

El algoritmo comienza ordenando los números por el dígito menos significativo (unidades) y sigue hacia los dígitos más significativos. Este enfoque asegura que, al final del proceso, los números estén completamente ordenados.

Ventajas:

Eficiencia en grandes volúmenes de datos. Tiene una complejidad casi lineal en términos de tiempo de ejecución para datos con un número fijo de dígitos, con una complejidad de $O\ (d\ x\ (\ n\ +\ k))$, donde:

- d es el número de dígitos (o bits) en el número más grande.
- n es el número de elementos.
- k es el rango de los dígitos (como 10 para decimales o 2 para bits en binario).

Para grandes bases de datos con enteros de longitud fija, Radix Sort puede superar a otros algoritmos comparativos como Quick Sort o Merge Sort.

Radix Sort puede ser muy productivo para grandes bases de datos, especialmente si los datos son numéricos de longitud fija o de rango limitado, ya que tiene un tiempo de ejecución cercano a $O\ (\ n)$. Sin embargo, su productividad disminuye si el rango de los

números es muy amplio, si los datos tiene mucha variabilidad en su tamaño o si hay limitaciones de memoria.

Tablas de análisis

3. La siguiente tabla contiene los tiempos empíricos obtenidos de ejecutar los distintos algoritmos con las entradas, estimando empíricamente el número de operaciones básicas que ejecuta cada algoritmo según el número de entradas que se analiza para cada caso (n=800, n=8000 y n=80000).

ANÁLISIS EMPÍRICO			
Algoritmo	**n**	**Tiempo de ejecución (seg)**	**Número de operaciones básicas**
Burbuja	800	0.05	798122
Burbuja	8000	4.95	80047431
Burbuja	80000	524.45	
Selección	800	0.01	320394
Selección	8000	1.2	32003990
Selección	80000	193.81	3200039981
Inserción	800	0.01	320234
Inserción	8000	1.41	32134853
Inserción	80000	221.13	3200625845
Mergesort	800	0.0	6745
Mergesort	8000	0.02	93642
Mergesort	80000	0.22	1203564
Quicksort	800	0.0	8391
Quicksort	8000	0.01	124178
Quicksort	80000	0.12	1260981

Gráfico 1: Tabla análisis empírico.

4. La siguiente tabla muestra los cálculos realizados con las fórmulas que estiman el número de intercambios y operaciones básicas de cada uno de los tipos de algoritmos de ordenación. A partir de las entradas analizadas para cada caso (n=800, n=8000 y n=80000).

ANÁLISIS MATEMÁTICO		
Algoritmo	**Fórmula cerrada número medio de intercambios**	**Número estimado de operaciones básicas para n=800, n=8000 y n=80000**
Burbuja	$\frac{n\,(n-1)}{4} =$	n=800:159600 n=8000:15,996,000 n=80000:1,599,960,000
Selección	$n - 1 =$	n=800:799 n=8000:7,999 n=80000:79,999
Inserción	$\frac{n\,(n-1)}{4} =$	n=800:159600 n=8000:15,996,000 n=80000:1,599,960,000
Mergesort	$o(n\ long\ n) =$	n=800:0 n=8000:0 n=80000:0
Quicksort	$o(n\ long\ n)$ =	n=800:7,712 n=8000:104,000 n=80000:1,305,600

Gráfico 2: Tabla análisis matemático.

5. A continuación se realiza el análisis de las diferencias entre el número de operaciones básicas calculadas empíricamente con la ejecución de los programas (valores experimentales) y los valores teóricos calculados a partir de las fórmulas. Indicando sus diferencias.

- Los algoritmos más eficientes corresponde a:

- Merge Sort (por mezcla).
- Quick Sort (Rápido).

- Los algoritmos menos eficientes corresponde a:

- Burbuja, Inserción y selección, siendo burbuja el más pesado para altos volúmenes de datos ya que son directamente proporcional a los datos (aplica para los diferentes tipos de algoritmos).

- Los algoritmos de ordenación nos permiten entender cómo cambia el tiempo de ejecución a medida que aumenta la cantidad de los datos.
- Teniendo en cuenta lo anterior se debe considerar la eficiencia de los algoritmos Merge Sort y Quick Sort para trabajar con grandes volúmenes de datos.
- De los laboratorios más complicados a realizar ya que tuve que esperar hasta la clase del 13 de septiembre para poder despejar las dudas en relación a la tabla de análisis empírico

el apartado número de operaciones básicas y aun asi no tengo la certeza de que esa sea la información solicitada ya que no comprendí muy bien ese apartado.

Entregables:

- Informe con análisis del laboratorio.
- Código fuente y entregables https://github.com/Johavaroli/Algoritmia-Lab1

Webgrafía

- Consulta ¿qué es un algoritmo de ordenación? https://www.freecodecamp.org/espanol/news/algoritmos-de-ordenacion-explicados-con-ejemplos-en-javascript-python-java-y-c/
- Análisis comparativo de los organismos de ordenamiento https://www.pereiratechtalks.com/analisis-de-algoritmos-de-ordenamiento/
- Consulta ¿cómo funciona Radix Sort? https://www.youtube.com/watch?v=W_euZjKoHkM
- Consulta ¿Cómo funcionan los algoritmos de ordenamiento para grandes volúmenes de datos? https://www.mheducation.es/bcv/guide/capitulo/8448198441.pdf