Instituto Politécnico Nacional
Escuela Superior de Cómputo

Ejercicios 02: Implemente los algoritmos de ordenamiento

Materia: Algoritmos y Estructuras de Datos
Profesor: Martínez Edgardo Franco

Alumno: González Joshua

Grupo: 2CM8
Fecha: 18/03/2023 ISC

Introducción

La ordenación es el proceso de reorganizar un conjunto de objetos en un orden específico para facilitar la búsqueda de elementos. Existen varios algoritmos de ordenación con diferentes ventajas de eficiencia en tiempo de ejecución. Los métodos de ordenación buscan un uso eficiente de la memoria y generalmente se prefieren las permutaciones in situ para no ocupar memoria adicional. En la computación, la ordenación se puede realizar en elementos en memoria principal o memoria secundaria. La ordenación es fundamental para la mayoría de los algoritmos que almacenan, editan o acceden a información, y su objetivo es crear una relación de orden entre los elementos para facilitar su procesamiento, acceso, edición y eliminación.

Planteamiento del problema

El objetivo de este reporte es comparar el rendimiento de cinco algoritmos de ordenamiento diferentes: burbuja, inserción, selección, mezcla y rápido. Se evaluará el tiempo que tardan estos algoritmos en ordenar conjuntos de datos de diferentes tamaños: 1000, 5000 y 10000 elementos.

Diseño e implementación de la solución

Los algoritmos se codificaron en lenguaje C, apoyándose de los pseudocódigos vistos en clases. Aunque hubo una pequeña modificación en el algoritmo de mezcla, ya que el pseudocódigo propuesto en clase no contemplaba el arreglo auxiliar C para ordenar y mezclar los subarreglos con el arreglo original. También se cambió una condición en los ciclos de Quicksort, en su función pivote, de “< r” a “<= r”, ya que esto hacía que no se considerara el último elemento y por consecuencia el arreglo se ordenara de manera errónea al no estar correctamente ordenadas las particiones. Así mismo, para reducir el código toda la lógica se implementó dentro del archivo “main.c”, aquí se usaba el algoritmo que el usuario deseaba usando punteros de funciones a la función correcta al ingresar los argumentos al ejecutar el programa.

Marco teórico

El algoritmo burbuja es un método sencillo de ordenamiento que recorre el arreglo varias veces y en cada paso intercambia los elementos adyacentes si están en el orden incorrecto. Aunque es fácil de implementar, su complejidad en el peor caso es O(n^2), lo que lo hace poco eficiente para arreglos grandes.

El algoritmo de selección encuentra el elemento mínimo del arreglo y lo intercambia con el primer elemento. Luego encuentra el siguiente elemento mínimo del subarreglo restante y lo intercambia con el segundo elemento, y así sucesivamente. Tiene una complejidad en el peor caso de O(n^2), pero es más eficiente que el burbuja en promedio.

El algoritmo de inserción recorre el arreglo de izquierda a derecha, comparando cada elemento con los que ya están ordenados. Si el elemento es menor que el elemento

ordenado, se inserta en su posición correcta. Tiene una complejidad en el peor caso de O(n^2), pero es más eficiente que el burbuja y la selección en arreglos pequeños.

Merge sort es un algoritmo recursivo de ordenamiento que divide el arreglo en dos mitades iguales, ordena cada mitad recursivamente y luego las fusiona. Tiene una complejidad de O(n log n) en el peor caso, lo que lo hace eficiente para arreglos grandes.

Quick sort también es un algoritmo recursivo que divide el arreglo en dos subarreglos, uno con elementos menores que un pivote y otro con elementos mayores. Luego ordena recursivamente cada subarreglo y los fusiona. Tiene una complejidad de O(n log n) en el promedio, pero puede degradarse a O(n^2) en el peor caso si se elige un pivote malo.

Funcionamiento del programa

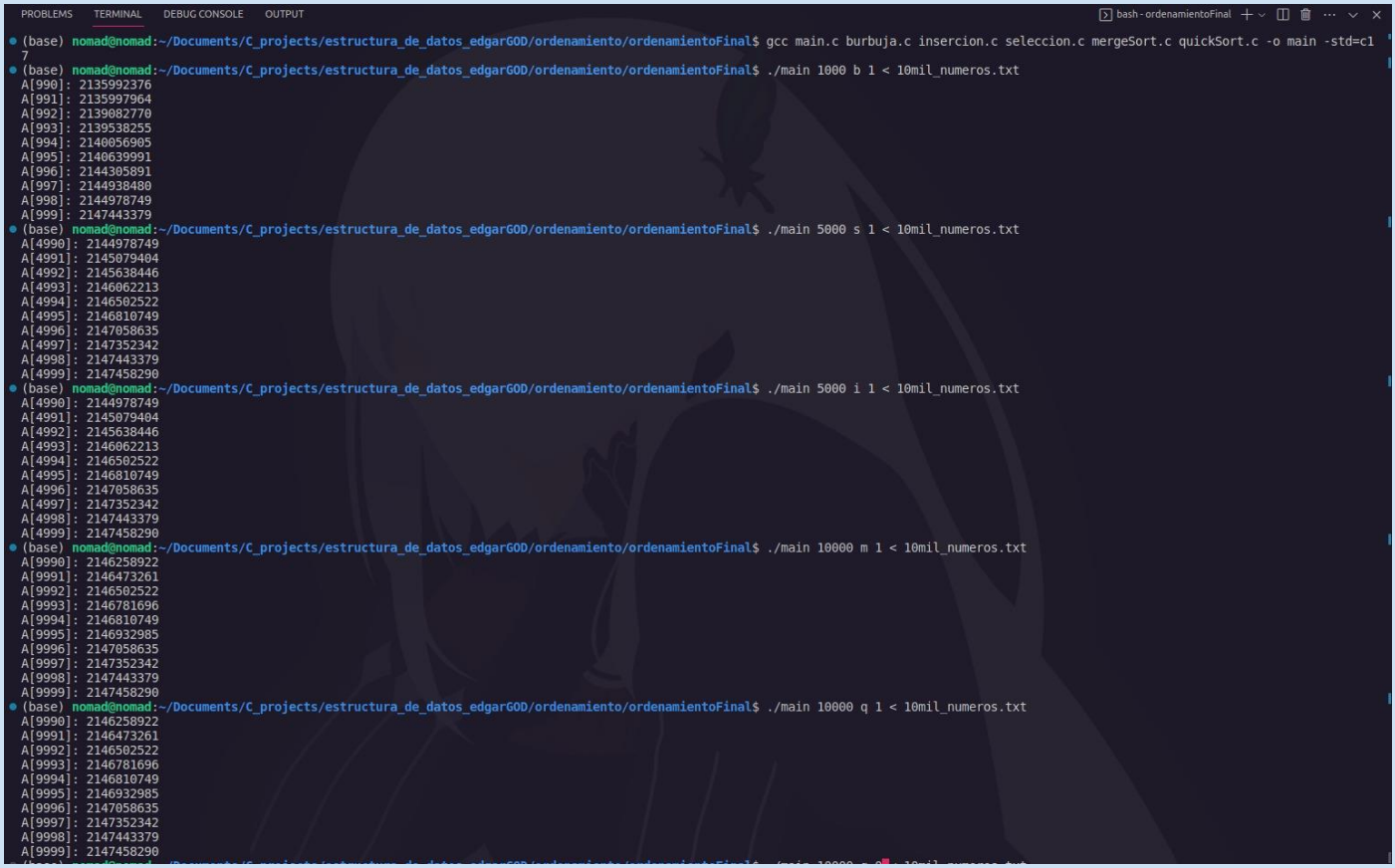
Instrucciones de compilación:

gcc main.c burbuja.c insercion.c seleccion.c mergeSort.c quickSort.c -o main -std=c17

Mi programa funcionaba tomando en cuenta los siguientes parámetros al ejecutarse:

1. TAMAÑO_DEL_ARREGLO
2. INICIAL_DEL_ALGORITMO
3. IMPRIMIR_ARREGLO_O_TIEMPO?
4. Tomaba la entrada de los números

A continuación se muestra un ejemplo de la ejecución imprimiendo los últimos 10 números de N datos de los arreglos ordenados y el tiempo que tarda en ejecutarse el algoritmo.



Reportes de marcas de tiempo

Se tomó la marca de 10 pruebas de tiempo para cada algoritmo y con base en ellas se obtuvo un promedio, estos fueron los resultados:

Algoritmo de ordenación de burbuja			
Marca de tiempo	**N=1000 (segundos)**	**N=5000 (segundos)**	**N=10000 (segundos)**
0	**0.0015160000**	**0.0367110000**	**0.1578600000**
1	**0.0015180000**	**0.0365110000**	**0.1587030000**
2	**0.0039560000**	**0.0380780000**	**0.1649540000**
3	**0.0019980000**	**0.0366220000**	**0.1573460000**
4	**0.0025820000**	**0.0365440000**	**0.1594300000**
5	**0.0015060000**	**0.0366310000**	**0.1585260000**
6	**0.0032870000**	**0.0375320000**	**0.1584250000**
7	**0.0031840000**	**0.0362200000**	**0.1658240000**
8	**0.0024140000**	**0.0362220000**	**0.1574620000**
9	**0.0027000000**	**0.0377920000**	**0.1580310000**
Promedio	**0.0024661**	**0.0368863**	**0.1596561**

Algoritmo de ordenación por Inserción			
Marca de tiempo	**N=1000 (segundos)**	**N=5000 (segundos)**	**N=10000 (segundos)**
0	**0.0004470000**	**0.0109450000**	**0.0431640000**
1	**0.0010070000**	**0.0109340000**	**0.0431780000**
2	**0.0004470000**	**0.0109250000**	**0.0446730000**
3	**0.0004470000**	**0.0109260000**	**0.0432650000**
4	**0.0004460000**	**0.0109280000**	**0.0432380000**
5	**0.0004480000**	**0.0114030000**	**0.0457880000**
6	**0.0004460000**	**0.0109400000**	**0.0431220000**
7	**0.0004470000**	**0.0109330000**	**0.0431620000**
8	**0.0011810000**	**0.0126410000**	**0.0452110000**
9	**0.0006370000**	**0.0109410000**	**0.0431500000**
Promedio	**0.0005953**	**0.0111516**	**0.0437951**

Alumno: González Joshua

Algoritmos y estructuras de datos 2CM18

Ejercicio 02: Implemente los algoritmos de ordenamiento

Algoritmo de ordenación por Selección			
Marca de tiempo	N=1000 (segundos)	N=5000 (segundos)	N=10000 (segundos)
0	0.0007410000	0.0174490000	0.0691930000
1	0.0007450000	0.0174420000	0.0692430000
2	0.0007500000	0.0174740000	0.0719290000
3	0.0007420000	0.0174510000	0.0692390000
4	0.0007410000	0.0174330000	0.0692120000
5	0.0007440000	0.0175040000	0.0691440000
6	0.0007450000	0.0185450000	0.0691520000
7	0.0007450000	0.0175320000	0.0691810000
8	0.0007420000	0.0174670000	0.0691660000
9	0.0007530000	0.0174330000	0.0692860000
Promedio	0.0007448	0.017573	0.0694745

Algoritmo de ordenación por mezcla			
Marca de tiempo	N=1000 (segundos)	N=5000 (segundos)	N=10000 (segundos)
0	0.0002440000	0.0006470000	0.0014190000
1	0.0001090000	0.0006480000	0.0013970000
2	0.0001260000	0.0006470000	0.0013790000
3	0.0001090000	0.0006470000	0.0013840000
4	0.0001080000	0.0006550000	0.0014190000
5	0.0001110000	0.0006460000	0.0014050000
6	0.0001090000	0.0006480000	0.0014040000
7	0.0001120000	0.0006460000	0.0013970000
8	0.0001090000	0.0006450000	0.0013830000
9	0.0001090000	0.0006490000	0.0013830000
Promedio	0.0001246	0.0006478	0.001397

Algoritmo de ordenación rápido			
Marca de tiempo	N=1000 (segundos)	N=5000 (segundos)	N=10000 (segundos)
0	0.0000720000	0.0004280000	0.0015170000
1	0.0000730000	0.0004450000	0.0009180000
2	0.0000730000	0.0006800000	0.0009180000
3	0.0000730000	0.0005430000	0.0016140000
4	0.0000720000	0.0004250000	0.0018670000
5	0.0000720000	0.0004250000	0.0016320000
6	0.0000730000	0.0004240000	0.0009190000
7	0.0000730000	0.0004280000	0.0009220000
8	0.0000720000	0.0004260000	0.0009150000
9	0.0000720000	0.0004240000	0.0009190000
Promedio	0.0000725	0.0004648	0.0012141

Algoritmo de ordenamiento	N=1000 (segundos)	N=5000 (segundos)	N=10000 (segundos)
Burbuja	0.0024661	0.0368863	0.1596561
Inserción	0.0005953	0.0111516	0.0437951
Selección	0.0007448	0.0175730	0.0694745
Mezcla	0.0001246	0.0006478	0.0013970
Rápido	0.0000725	0.0004648	0.0012141

Conclusión

El ordenamiento es un proceso fundamental en la programación y es esencial para muchas aplicaciones. Los algoritmos de ordenamiento son una herramienta esencial en la programación para clasificar una gran cantidad de datos. Sin embargo, cada algoritmo tiene sus propias características y es importante comprender su eficiencia y rendimiento en diferentes tamaños de datos. Es imprescindible saber cuándo usar un algoritmo u otro, que mejor ejemplo que la función de ordenación sort de la STL de C++, que se llama introsort, que combina el algoritmo de ordenamiento de inserción, heap y Quicksort, así la ordenación será más rápida dependiendo del estado actual del arreglo, es decir: si está casi ordenado, si el primero o último son el elemento desordenado, etc.

Bibliografía

[1]. S. Makarychev, "Sorting Algorithms Explained with Examples in Python, Java, and C++," freeCodeCamp.org, 02-Dec-2020. [Online]. Available: https://www.freecodecamp.org/news/sorting-algorithms-explained-with-examples-in-python-java-and-c/. [Accessed: 18-Mar-2023].

[2]. "Sorting Algorithms," GeeksforGeeks, [Online]. Available: https://www.geeksforgeeks.org/sorting-algorithms/. [Accessed: 18-Mar-2023].

[3]. "Sorting Algorithms in Data Structures," Programiz, [Online]. Available: https://www.programiz.com/dsa/sorting-algorithm. [Accessed: 18-Mar-2023].