Estructura de datos y Algoritmos.
2º Ingeniería de la Salud

Nombre y Apellidos: Alejandro Domínguez Recio

Práctica 3: Divide y vencerás

Objetivos

El objetivo de esta práctica es la implementación de los algoritmos de ordenación Mergesort y Quicksort. Para el desarrollo del algoritmo Mergesort utilizamos dos métodos sort y mezclar. En el metodo sort introducimos por parametros la lista a ordenar y los extremos donde queremos que se efectue la ordenación. Dentro del metodo sort nos ayudamos de tres listas las cuales llamadas recursivamente marcaran la mitad derecha e izquierda de nuestra lista original como una tercera que sera nuestra lista de apoyo a la ordenación. El segundo método que utilizamos el mezclar es donde se realizan las comparaciónes caracteristicas del algoritmo y el cual va ordenando las listas dadas comos parámetros. En el algoritmo Quicksort utilizamos igualmente dos métodos el sort y el partir. En el método sort igualmente que en el anterior se puede introducir por parámetros la lista a ordenar y los extremos donde se pide que se realize la ordenación pero con la diferencia que la llamada recursiva no vendra marcada siempre por una división de las listas por la mitad si no que dependerá de la partición correspondiente marcada por el metodo partir. En el método partir se efectuaran las comparaciones correspondientes con el fin de devolver un entero que sera la posición por la que se debe dividir nuestros arrays para realizar la ordenación, dejando a la izquierda los valores menores que este y a la izquiera los mayores.

Configuración

-Características de la máquina en la que se han ejecutado los diferentes algoritmos:

N.º de cores CPU = 1
Frecuencia = 2.20Ghz
HDD = 916Gb
RAM = 8,00 Gb
Versión de Java = JavaSE-13

-Parámetros con los que se han ejecutado los algoritmos:

En cada uno de los diferentes algoritmos he utilizado los mismos parámetros definidos por defecto:

Tamaño inicial de array : 2000
Número de veces que se dobla el tamaño del array : 6
Número de test por cada tamaño de array : 10

Resultados

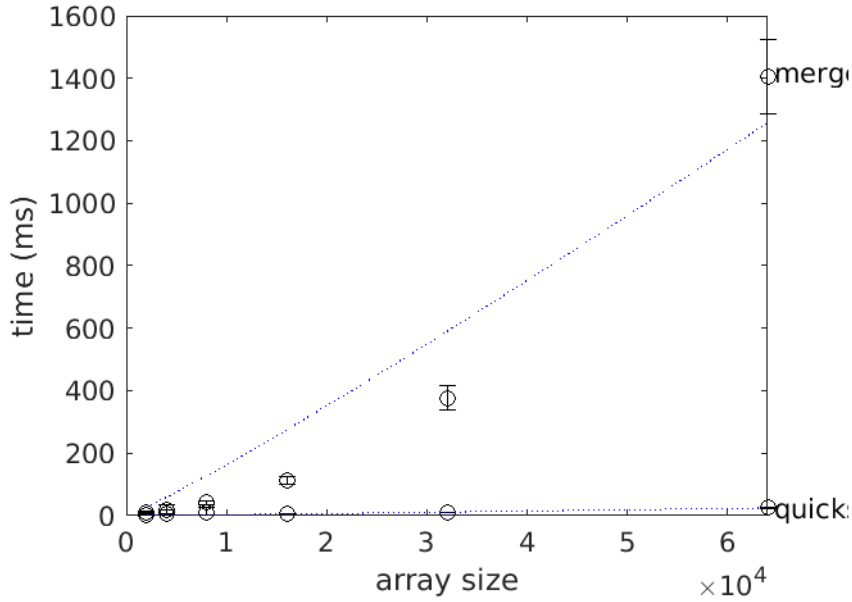
Tabla del tiempo de ejecución del algoritmo Quicksort:

2000	6	3	2	4	20	6	3	1	2
4000	1	1	0	1	2	1	0	1	1
8000	9	2	2	3	2	2	3	2	2
16000	4	6	4	5	6	7	5	6	5
32000	12	10	11	10	9	9	9	10	9
64000	22	20	20	18	16	16	22	17	20

Tabla del tiempo de ejecución del algoritmo Mergesort:

2000	42	17	58	54	10	7	8	7	13
4000	19	17	18	17	25	18	16	17	20
8000	70	50	50	57	51	48	43	40	43
16000	133	133	129	138	131	132	119	135	130
32000	374	438	358	298	424	382	478	449	382
64000	1703	1364	1368	1348	1589	1674	1704	1484	1673

Representación gráfica de los tiempos de ejecución de los algoritmos Mergesort y Quicksort.



Conclusiones

A simple vista podemos apreciar que ambos algoritmos se comportan de una manera similar cuando el tamaño de nuestro input es relativamente pequeño pero cuando este alcanza tamaños considerablemente grandes Quicksort es mucho mas eficiente. Respecto a las complejidades de ambos algoritmos podemos comentar que aunque teniendo diferencias de complejidad en el peor caso de comparaciones $O(n^2)$ para

quicksort y o(nlogn) para mergesort, el quicksort resulta más eficiente y esto es porque la mayoría de las veces se mueve en su caso promedio o(nlogn) o en su mejor caso O(n) debido a que por una parte no utiliza tres arrays solo dos disminuyendo la utilización de memoria como tambien porque por su forma de comparar se darán casos en los que no sea necesario recorrer uno de los dos arrays debiado a que ya se encuentre ordenado.

Bonus opcional:

Respecto a la comparación con los algoritmos de ordenación bubblesort, insertion y selection podemos decir claramente que estos tienen un tiempo de ejecución mucho mayor como tambien una complejidad superior independientemente del tamaño del input impuesto en nuestras pruebas, mostrando la diferencia respecto a eficiencia que aporta el utilizar el enfoque divide y vencerás en este tipo de implementaciones.