Fundamentos de Análisis y Diseño de Algoritmos

Taller 3

“Algoritmos de ordenamiento”

Msc. Carlos Andrés Delgado Saavedra

Leider Santiago Cortés Hernández – 202159879

Miguel Ángel Rueda Colonia - 202159896

5 (quinto) semestre

Universidad del Valle sede Tuluá

02 de diciembre de 2023

Tuluá, Valle del cauca

1. Análisis algoritmo Divide y Vencerás:

a) Explicar la estrategia dividir y conquistar:

StoogeSort utiliza la técnica de divide y vencerás, En este caso, el problema es ordenar un arreglo de n elementos, y los subproblemas son ordenar las partes del arreglo. El algoritmo divide el arreglo en tres partes iguales, y ordena las primeras dos partes y las últimas dos partes recursivamente, hasta que el arreglo esté ordenado.

b) Implementación del algoritmo:

El algoritmo StoogeSort divide el segmento en tercios si tiene más de dos elementos, y realiza un ordenamiento recursivo en los primeros dos tercios, los últimos dos tercios, y de nuevo en los primeros dos tercios. Este proceso se repite hasta que el segmento a ordenar sea lo suficientemente pequeño, asegurando que el arreglo esté ordenado, ejecutando así la estrategia (D.C.V)

c) Calcular la complejidad teórica del algoritmo StoogeSort y compararla con la complejidad practica encontrada. Analizar lo encontrado:

n	Complejidad teórica O($n^{2.7}$)	Tiempo real (mili sg)	Constantes (tr/ct)
0	0	0	0
10	501.1872336	0.9872	0.00196972295
100	251188.6432	41.60332679748535	0.00016562582
1000	125892541.2	10671.868562698364	0.00008476966
10000	$6.309573445x10^{10}$	ind	----------
		Constante:	0.000124

Conclusiones de la Discusión: StoogeSort es un algoritmo interesante desde el punto de vista teórico debido a su naturaleza recursiva triple, pero su complejidad exponencial lo hace menos eficiente en comparación con algoritmos más eficientes y ampliamente utilizados. La estrategia de "Divide y Vencerás" está claramente presente en su estructura, pero en la práctica, su rendimiento puede no ser óptimo para conjuntos de datos de gran tamaño. La elección de algoritmos para situaciones del mundo real debe considerar la eficiencia práctica además de la complejidad teórica.

2. Diseño de algoritmo Divide y vencerás:

a) Explicar la estrategia dividir y conquistar:

La estrategia "divide y vencerás" se manifiesta en la división del problema original en subproblemas más pequeños (contar frecuencias y encontrar moda entre elementos únicos) y luego combinar las soluciones de estos subproblemas para obtener la solución general. La recurrencia y división del problema original en partes más pequeñas se realizan hasta llegar a casos base o soluciones triviales.

b) Implementación del algoritmo:

El algoritmo se dividió en dos partes fundamentales: la primera encargada de contar las frecuencias de cada elemento en el arreglo, y la segunda destinada a encontrar la moda entre los elementos únicos del conjunto.

c) Calcular la complejidad teórica del algoritmo y compararla con la complejidad practica encontrada. Analizar lo encontrado:

n	Complejidad teórica $O(n)$	Tiempo real (mili sg)	Constantes
0	0	0	0
10	10	0.00949880000007397	0.000949
100	100	0.05581989999882353	5.58x10-4
1000	1000	0.35131209999963176	3.51x10-41
10000	10000	2.800592499999766	0.00028

Constante: 4.71x10-5

Conclusiones de la Discusión: El diseño del algoritmo para encontrar la moda bajo la estrategia de "Divide y Vencerás" resultó en una implementación eficiente y escalable. La combinación de división, conquista y combinación permitió abordar el problema de manera estructurada y proporcionó un equilibrio entre complejidad teórica y eficiencia práctica. La estrategia demostró ser particularmente efectiva para conjuntos de datos de diversos tamaños, ofreciendo resultados coherentes con la complejidad teórica esperada.

3. Comparación de algoritmos:

Quick-sort:

n	Complejidad teórica $O(n \log n)$	Tiempo real (mili sg)	Constantes
10	10	0.00033179999991261866	3,318E-05
10	10	0.00025439999990339857	2,544E-05
50	84.94850022	0.0016585000012128148	1,9524E-13
50	84.94850022	0.09589290000076289	1,1288E-11
100	200	0.2991629999996803	0,00149581
100	200	0.22611589999905846	0,00113058
500	1349.485002	2.1533595000000787	1,5957E-09
500	1349.485002	1.889539900001182	1,4002E-09
1000	3000	4.767787599999792	0,00158926
1000	3000	4.770641299999625	0,00159021
2000	6602.059991	10.90026570000009	1,651E-09
2000	6602.059991	10.916516500001308	1,6535E-09
5000	18494.85002	47.867466600000625	2,5882E-08
5000	18494.85002	45.3402412000014	2,4515E-08
10000	40000	139.36378479999985	0,00348409
10000	40000	134.6403210999997	0,00336601

Constante: 1.60x10-9

Insertion-Sort:

n	Complejidad teórica $O(n^2)$	Tiempo real (mili sg)	Constantes
10	100	0.1<	0,011
10	100	0.1<	0,01
50	2500	0.1<	1,1772E-11
50	2500	0.1<	1,1772E-11
100	10000	0.5443096160888672	0,00272155
100	10000	1.0399818420410156	5,1999E+13
500	250000	4.504203796386719	3337720,53
500	250000	3.0100345611572266	22305061,2
1000	1000000	19.131183624267578	6,3771E+12
1000	1000000	19.023418426513672	6,3411E+12
2000	4000000	74.92399215698242	1134857,79
2000	4000000	76.17926597595215	1153871,16
5000	25000000	509.23705101013184	27533991,9
5000	25000000	495.58496475219727	26795835,8
10000	100000000	1956.6967487335205	4,8917E+11
10000	100000000	2005.9094429016113	5,0148E+11

Constante: 1.234x10-11

Merge-Sort:

n	Complejidad teórica $\theta(n \log n)$	Tiempo real (mili sg)	Constantes
10	10	1.000357867962451	1,0004E+14
10	10	1.004934310913086	1,0049E+14
50	84.94850022	1.0113716125488281	1190570,3
50	84.94850022	1.0008811950683594	1178221,15
100	200	1.003265380859375	5,0163E+12
100	200	1.001596450805664	5,008E+12
500	1349.485002	1.0247230529785156	7593437,88
500	1349.485002	0.9977817535400391	7,3938E-10
1000	3000	1.9989013671875	6663004557
1000	3000	2.616405487060547	8,7214E+11
2000	6602.059991	3.5390853881835938	5360577,45
2000	6602.059991	3.506898880004883	531182,523
5000	18494.85002	14.478445053100586	7828365,75
5000	18494.85002	11.47913932800293	620666,797
10000	40000	22.088289260864258	5,5221E+11
10000	40000	23.341894149780273	5,8355E+11

Constante: 9,4312x10-6

Conclusiones de la discusión:

En estas tablas, se muestran tiempos reales de ordenamiento para Quicksort, Insertion-Sort y Merge-Sort con diferentes tamaños de entrada. La complejidad temporal de cada algoritmo se proporciona junto con las constantes obtenidas al calcular el factor constante.

Es importante tener en cuenta que los tiempos reales pueden variar según el entorno de ejecución, y los valores presentados son ejemplos hipotéticos.

Además, el análisis del factor constante permite comparar la eficiencia relativa de los algoritmos en la práctica. Los resultados destacan que QuickSort y Merge-Sort, que tienen complejidades $O(n\log n)$, superan a Insertion-Sort, que tiene una complejidad $O(n2)$, especialmente a medida que el tamaño de entrada aumenta.