Universidad del Valle

Taller 3 ADA

Análisis y diseño de algoritmos

Carlos Andres Delgado

Integrantes:

Lemus Agudelo Maria Paula - 202259570
Ramirez Mesa Jandi Yandubery – 202259594
Viedma Rueda Camilo Andres - 202259461

Punto 1

A) Explique la estrategia: Dividir, conquistar y combinar

En este algoritmo:

- Si el tamaño del arreglo es 1 o menos, ya está ordenado y se devuelve el arreglo.
- Se realiza un intercambio si el primer elemento es mayor que el último para asegurarse de que el arreglo esté parcialmente ordenado.
- Se calcula el índice t para dividir el arreglo en tres partes aproximadamente iguales.
- Se aplica el algoritmo Stooge Sort de manera recursiva en las tres partes del arreglo.
- Se combina el resultado final.

Luego, el código principal prueba el algoritmo Stooge Sort con arreglos aleatorios de diferentes tamaños y verifica la corrección del algoritmo utilizando la función `test_sort_algorithm`. Es importante tener en cuenta que Stooge Sort no es un algoritmo eficiente en términos de tiempo de ejecución y generalmente se utiliza con fines educativos más que prácticos. Su complejidad temporal es alta y no es recomendable para conjuntos de datos grandes. En el código, se incluye una prueba de rendimiento para ilustrar esto al ordenar arreglos de diferentes tamaños.

Listing 1: Código de Stooge Sort

```
def stooge_sort(arr):
    if len(arr) <= 1:
        return arr

    # Intercambia los elementos si el primer elemento es mayor que el último
    if arr[0] > arr[-1]:
        arr[0], arr[-1] = arr[-1], arr[0]

    if len(arr) > 2:
        # Calcula el índice para dividir en tres partes aproximadamente iguales
        t = len(arr) // 3

        # Aplica el algoritmo Stooge Sort recursivamente en las tres partes
        arr[:t] = stooge_sort(arr[:t])
        arr[-t:] = stooge_sort(arr[-t:])
        arr[:t] = stooge_sort(arr[:t])

    return arr
```

B -

C -

D. Calcule la complejidad teórica del algoritmo y compárela con la complejidad práctica encontrada. Analice lo que encuentra.

- La complejidad de tiempo del algoritmo Stooge Sort es aproximadamente $O(n^{2.7095})$. La fórmula exacta para la complejidad de tiempo de Stooge Sort es $T(n) = 3 \cdot T(2n/3) + O(1)$, que resuelve a aproximadamente $O(n^{\log_3 3}) = O(n^{2.7095})$.

  Esta fórmula se deriva del hecho de que el algoritmo realiza tres llamadas recursivas en 2/3 del tamaño del arreglo original y una cantidad constante de trabajo adicional (intercambiando elementos si es necesario).

Punto 2

A) Estrategia

Consiste en utilizar el algoritmo de ordenamiento Quicksort y organizar la lista de menor a mayor. Se implementa un algoritmo que recorre la lista una sola vez preguntando si el índice actual es igual al siguiente. En caso afirmativo, se incrementa una variable de contador que cuenta la cantidad de veces que se repite el valor en la lista y se ve reemplazado cuando aparece un elemento que se repite más veces. En caso de que se repitan la misma cantidad de veces y sean los mayores, se concatenan y se muestran en la salida.

B -

C -

D) Complejidad

La complejidad encontrada es $O(n)$ ya que depende en gran medida de los valores ingresados y el algoritmo recorrerá solo un bucle `for`.

Entrada	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
10	45	41	38	37	36	37	41	38	40	40
100	319	323	320	319	321	321	323	318	316	318
1000	3026	3038	3034	3040	3039	3042	3041	3052	3035	3065
10000	30047	30047	30083	30051	30053	30054	30064	30051	30045	30062
100000	300046	300045	300049	300038	300059	300050	300054	300055	300132	300081
1000000	3000131	3000108	3000110	3000123	3000107	3000098	3000091	3000087	3000107	3000096

Table 1: Complejidad.

Entrada	**Promedio**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
10	39.3	32.49	2.89	1.69	5.29	10.89	5.29	2.89	1.69	0.49	0.49
100	320.8	3.24	4.84	0.64	3.24	0.04	0.04	4.84	7.84	23.04	51.84
1000	3041.2	231.04	10.24	51.84	1.44	4.84	0.64	0.04	116.64	98.44	566.44
10000	30055.7	75.69	75.69	745.29	22.09	7.29	2.89	68.89	22.09	114.49	39.69
100000	300060.9	222.01	252.81	141.61	524.41	3.61	118.81	47.61	34.81	5055.21	404.01
1000000	3000105.8	635.04	4.84	17.64	295.84	1.44	60.84	219.04	353.44	1.44	96.04

Table 2: Desviación estándar por entrada.

Entrada	**Suma**	**1/n**	**Desviación**
10	64.1	7.12	2.67
100	99.6	11.07	3.33
1000	1021.6	113.51	10.65
10000	1174.1	130.46	11.42
100000	6804.9	756.1	27.50
1000000	1685.6	187.29	13.69

- Se calcula la desviación estándar por que los resultados varían dependiendo de los valores de la entrada razón por la que salen resultados distintitos en la complejidad, porque los resultados varían dependiendo de los valores de la entrada razón por la que salen resultados distintitos en la complejidad.

Punto 3

Implemente los algoritmos QuickSort, Insertion-Sort y Merge-Sort y realice una comparaci´on entre los utilizando una tabla para entradas aleatorias de tama˜no 10, 50, 100, 500, 1000, 2000, 5000 y 10000.

- En esta sección, se presentan los resultados de los algoritmos de ordenamiento QuickSort, Insertion-Sort y Merge-Sort en diferentes tamaños de entrada. Cada tabla muestra el tiempo de ejecución, la complejidad y la constante asociada a cada algoritmo.

TAMAÑO ENTRADA	TIEMPO	COMPLEJIDAD	CONSTANTE
10	9.30e-06	100	4.04e-07
10	8.11e-06	100	3.52e-07
10	1.07e-05	100	4.66e-07
50	0.00012	2500	6.36e-07
50	0.00011	2500	5.39e-07
50	9.82e-05	2500	5.02e-07
100	0.00038	10000	8.34e-07
100	0.00041	10000	8.96e-07
100	0.00041	10000	8.93e-07
500	0.01027	250000	3.31e-06
500	0.01024	250000	3.29e-06
500	0.01123	250000	3.62e-06
1000	0.04427	1000000	6.41e-06
1000	0.04274	1000000	6.19e-06
1000	0.04209	1000000	6.09e-06
2000	0.18329	4000000	1.21e-05
2000	0.19918	4000000	1.31e-05
2000	0.35386	4000000	2.33e-05
5000	2.41	25000000	5.66e-05
5000	2.06	25000000	4.85e-05
5000	1.14	25000000	2.68e-05
10000	4.68	100000000	5.09e-05
10000	5.60	100000000	6.08e-05
10000	6.15	100000000	6.68e-05

Table 3: Resultados de QuickSort.

TAMAÑO ENTRADA	TIEMPO	COMPLEJIDAD	CONSTANTE
10	2.22e-05	100	9.63e-07
10	2.36e-05	100	1.03e-06
10	2.31e-05	100	1.00e-06
50	9.94e-05	195.60	5.08e-07
50	9.68e-05	195.60	4.95e-07
50	9.13e-05	195.60	4.67e-07
100	0.00020	460.52	4.31e-07
100	0.00018	460.52	4.01e-07
100	0.00021	460.52	4.45e-07
500	0.000999	3107.30	3.22e-07
500	0.00102	3107.30	3.29e-07
500	0.00167	3107.30	5.39e-07
1000	0.00211	6907.76	3.06e-07
1000	0.00235	6907.76	3.41e-07
1000	0.00219	6907.76	3.18e-07
2000	0.00450	15201.80	2.96e-07
2000	0.00495	15201.80	3.26e-07
2000	0.00475	15201.80	3.12e-07
5000	0.01020	42585.97	2.39e-07
5000	0.01216	42585.97	2.86e-07
5000	0.01607	42585.97	3.77e-07
10000	0.01749	92103.40	1.90e-07
10000	0.02256	92103.40	2.45e-07
10000	0.01862	92103.40	2.02e-07

Table 4: Resultados de Insertion-Sort.

TAMAÑO ENTRADA	TIEMPO	COMPLEJIDAD	CONSTANTE
10	2.50e-05	23.03	2.50e-07
10	3.34e-05	23.03	3.34e-07
10	3.27e-05	23.03	3.27e-07
50	0.00012	195.60	4.98e-08
50	0.00012	195.60	4.96e-08
50	0.00012	195.60	4.90e-08
100	0.00029	460.52	2.86e-08
100	0.00029	460.52	2.86e-08
100	0.00039	460.52	3.85e-08
500	0.00170	3107.30	6.80e-09
500	0.00153	3107.30	6.11e-09
500	0.00154	3107.30	6.15e-09
1000	0.00327	6907.76	3.27e-09
1000	0.00327	6907.76	3.27e-09
1000	0.00338	6907.76	3.38e-09
2000	0.00752	15201.80	1.88e-09
2000	0.00714	15201.80	1.79e-09
2000	0.00754	15201.80	1.89e-09
5000	0.02140	42585.97	8.56e-10
5000	0.02289	42585.97	9.16e-10
5000	0.02150	42585.97	8.60e-10
10000	0.04634	92103.40	4.63e-10
10000	0.04848	92103.40	4.85e-10
10000	0.04344	92103.40	4.34e-10

Table 5: Resultados de Merge-Sort.