TALLER 3

Orrego Alfonso Daniela 2259579-2724

Vélez Ospina Juan Camilo 2259635-2427

PUNTO 1

A. Divide: se realiza una operación dividiendo el intervalo [i, h] en tres partes: [i, (h-i+1)/3], [(h-i+1)/3, h - ((h-i +1) /3)] y [h-((h-i+1)/3), h]. Luego se aplican a los dos nuevos segmentos y el segmento medio se utiliza como punto de referencia para garantizar la alineación correcta de los elementos en la lista. En caso de que el índice i sea mayor o igual al h, no se realiza ninguna operación indicando que la lista esta ordenada.

   Conquista – Combinación: esta operación se realiza en las subpartes, es una función contenedora con temporizador incluido para medir el tiempo de ejecución del algoritmo en diferentes arreglos de tamaño y contenido.

B. El tiempo de ejecución de cada llamada se mide y se muestra en la consola. Como era de esperar, el tiempo de ejecución aumenta significativamente a medida que aumenta el tamaño de la matriz.

C.

10	0.00003
100	0.038552284240722656
1000	9.170341491699219
10000	16.734187126159668

D. El tiempo de ejecución aumenta significativamente a medida que aumenta el tamaño de la matriz. Dado que la complejidad práctica del algoritmo StoogeSort es significativamente mayor que la complejidad teórica

PUNTO 2

A. Dividir: Primero dividimos el arreglo en dos partes iguales. Esto se hace una y otra vez hasta que el arreglo se divide en diferentes partes del mismo elemento.

   Conquistar: después de dividir el arreglo, calculamos la moda de cada parte del arreglo. Para hacer esto, contamos la frecuencia de cada elemento en el elemento del arreglo y almacenamos los elementos con la frecuencia promedio (medias).

Combinar: Finalmente, combinamos las modas de dos partes del arreglo para obtener la moda de todo el sistema.

B. La función moda divide el vector en dos partes, encuentra las modas de ambas partes y las combina. Se crea un diccionario para contar la frecuencia de cada elemento en los modos combinados. Finalmente, se buscan los elementos con la frecuencia máxima (modas) y se devuelven.

C.

10	0.0000564628
100	0.0045434343
1000	0.00201392173834743
10000	0.0145165992025756836

D. En las pruebas realizadas con 10, 100, 1000 y 10000 elementos, el tiempo de ejecución del algoritmo varió desde aproximadamente 0,0001 segundos hasta aproximadamente 0,023 segundos.

El algoritmo implementado para encontrar la moda en este código tiene una complejidad temporal de O(n log n), porque divide el vector en dos partes y utiliza una técnica similar a la selección rápida para encontrar los elementos más frecuentes.
En términos prácticos, el tiempo de ejecución de este algoritmo disminuye rápidamente a medida que aumenta el tamaño del vector, porque los tiempos de ejecución se promedian en segundos y el algoritmo no muestra signos de ser lento a medida que aumenta el tamaño del vector.

PUNTO 3

Insert Sort

Entrada (n)	Tiempo real (seg)	Complejidad	constantes
10	0,3	100	0,003000
10	0	100	0,000000
10	0	100	0,000000
50	0	2500	0,000000
50	0	2500	0,000000
50	0	2500	0,000000
100	0	10000	0,000000
100	0	10000	0,000000
100	0	10000	0,000000
500	500	250000	0,000000
500	0	250000	0,000000
500	0	250000	0,000000
1000	1000	100000	0,000000
1000	1000	100000	0,000000
1000	1000	100000	0,000000

2000	4000	400000	0,000000
2000	0	400000	0,000000
2000	4000	400000	0,000000
5000	75000	2500000	0,000000
5000	50000	2500000	0,000000
5000	800000	2500000	0,000000
1000	800000	10000000	0,000000
1000	900000	10000000	0,000000
1000	900000	10000000	0,000000