Complejidad de un algoritmo

Introducción

La complejidad algorítmica es aquella que dicta cual es la cantidad de recursos que necesita un algoritmo para resolver un problema, logrando el poder determinar cuándo es eficiente el algoritmo o cuando no lo es. Sin embargo, la principal preocupación del análisis del algoritmo es el tiempo o el rendimiento requeridos.

La complejidad de un algoritmo se puede dividir en dos tipos. La complejidad del tiempo y la complejidad del espacio.

1. Complejidad de tiempo de un algoritmo

La complejidad del tiempo se define como el proceso de determinar una fórmula para el tiempo total requerido para la ejecución de ese algoritmo. Este cálculo es totalmente independiente de la implementación y el lenguaje de programación.

2. Complejidad espacial de un algoritmo

La complejidad espacial se define como el proceso de definir una fórmula para la predicción de cuánto espacio de memoria se requiere para la ejecución exitosa del algoritmo. El espacio de memoria generalmente se considera como la memoria primaria.

En este trabajo, se presenta el comparativo de los siguientes algoritmos de ordenamiento:

Ordenamiento por selección (Selection Sort): La clasificación por selección es uno de los algoritmos básicos para ordenar datos, su simplicidad resulta útil para ordenar pequeñas cantidades de datos. La ordenación por selección funciona comenzando primero en el vector inicial (índice 0) y recorre todo el vector comparando cada valor con el índice actual, si es más pequeño que el índice actual de lo que se guarda ese índice. Una vez que el bucle ha atravesado todos los datos y si se encontró un valor más pequeño que el índice actual, se realiza un intercambio entre el índice actual en el índice donde se encontró el valor más pequeño. El índice actual se incrementa, ahora al índice 1, el algoritmo se repite.

Ordenamiento burbuja (Bubble Sort)

Bubble sort es un algoritmo de comparación. Compara los elementos adyacentes del vector e intercambia los elementos para mover los elementos a la posición correcta. En cada iteración, empuja el elemento más alto al final del vector. Para la primera iteración, empuja el elemento más alto hasta el final, para la segunda iteración, empuja el segundo elemento más alto hasta el final, etc.

Ordenamiento rápido (Quick Sort)

Es uno de los algoritmos de clasificación más rápidos en programación. Ordena el vector de tal manera que el punto de pivote entra en el medio y a la izquierda del punto de pivote se generan elementos más pequeños y a la derecha del punto de pivote se generan elementos más grandes.

Ordenamiento por mezclas (Merge Sort)

Ésta algoritmo ordena cada vector, divide la lista sin ordenar en *n* sublistas, cada una de las cuales contiene un elemento (una lista de un elemento se considera ordenada).

Después fusiona repetidamente las sublistas para producir nuevas sublistas ordenadas hasta que solo quede una sublista. Esta será la lista ordenada.

Con el objetivo de evaluar y comparar la complejidad computacional de cada uno de los algoritmos mencionados, en el archivo T_3.cpp se presenta el código que explora el desempeño. Dichos algoritmos ordenan de manera ascendente un arreglo. La complejidad de cada algoritmo mide el tiempo de ejecución con diferentes arreglos aleatorios de diferentes tamaños**.** Se realizaron 15 réplicas por cada tamaño, los tamaños son los siguientes: 100,500,1000,1500,2000,2500.

Resultados

A continuación, se presentan las 4 tablas en las que se reportan los tiempos de ejecución de las 15 réplicas para cada uno de los tamaños especificados:

Selection Sort

100	500	1000	1500	2000	2500
0	0.001	0.003	0.002	0.004	0.007
0	0	0.001	0.003	0.004	0.006
0	0.001	0	0.002	0.006	0.007
0	0	0.001	0.002	0.006	0.006
0	0	0.001	0.002	0.006	0.007
0	0	0.001	0.003	0.007	0.01
0	0.002	0.001	0.003	0.007	0.007
0	0	0.001	0.002	0.005	0.006
0	0.001	0.001	0.002	0.005	0.007
0	0	0.002	0.003	0.004	0.01
0	0.001	0.001	0.002	0.004	0.007
0	0	0.001	0.003	0.004	0.007
0	0	0.001	0.002	0.004	0.006
0	0	0.001	0.002	0.004	0.007
0	0	0.001	0.003	0.005	0.007

Bubble Sort

100	500	1000	1500	2000	2500
0	0.001	0.003	0.003	0.006	0.01
0	0.001	0.002	0.004	0.006	0.01
0	0.001	0.002	0.003	0.007	0.009
0	0.001	0.002	0.004	0.006	0.009
0	0.001	0.002	0.003	0.007	0.011
0	0.001	0.002	0.004	0.009	0.009
0	0.001	0.001	0.003	0.008	0.01
0	0.001	0.002	0.003	0.006	0.009
0	0.001	0.001	0.004	0.007	0.009
0	0.001	0.002	0.004	0.006	0.009
0	0	0.002	0.003	0.006	0.01
0	0.001	0.001	0.004	0.009	0.01
0	0	0.002	0.003	0.008	0.01
0	0.001	0.001	0.004	0.007	0.01
0	0	0.002	0.003	0.006	0.009

Quick Sort

100	500	1000	1500	2000	2500
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0.001
0	0	0	0	0	0
0	0	0	0.001	0	0
0	0	0	0	0.001	0
0	0	0	0	0	0
0	0	0.001	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0.001
0	0	0	0	0	0

Merge Sort

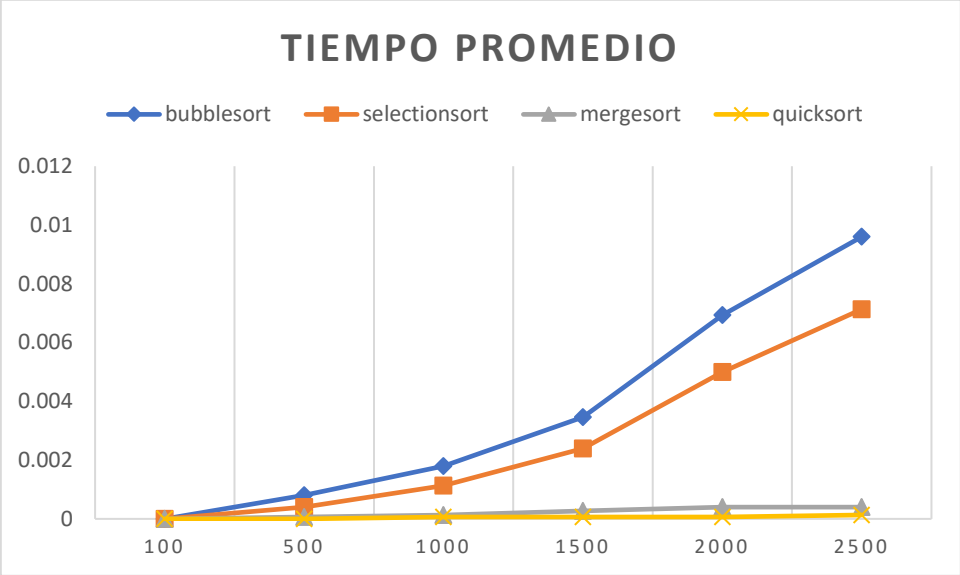
100	500	1000	1500	2000	2500
0	0	0	0.001	0.001	0
0	0	0	0	0	0.001
0	0	0.001	0	0	0
0	0	0	0.001	0.001	0.001
0	0	0	0	0	0
0	0	0	0	0.001	0
0	0	0	0	0	0.001
0	0.001	0	0.001	0	0
0	0	0.001	0	0.001	0.001
0	0	0	0	0	0
0	0	0	0	0	0.001
0	0	0	0.001	0.001	0
0	0	0	0	0	0
0	0	0	0	0	0.001
0	0	0	0	0.001	0

Estos resultados también pueden ser observados en los archivos de Excel adjuntos.

Al realizar una gráfica de comparación entre el tiempo promedio que tarda en ordenar los datos podemos concluir dos cosas:

Escogiendo el Selection Sort y el Bubble Sort podemos observar que tienden a tardar más tiempo en ordenar los datos conforme aumenta el tamaño de estos.

Mientras que, por otro lado, con los ordenamientos Merge Sort y Quick Sort el tiempo no es tan diferente cuando su tamaño incrementa.



Para más detalle de los resultados, revisar el archivo Excel adjunto:

analisis%20resultados.xlsx

En otras palabras, de acuerdo con el experimento presentado, se puede concluir que los algoritmos Selection Sort y Bubble Sort son bastante complejos, en termino de tiempo, conforme el tamaño de elementos a ordenar incrementa.

Por otra parte, la complejidad del algoritmo Quick Sort y Merge Sort tiende a no tener incrementos tan significativos conforme el tamaño del vector crece.