Instituto Tecnológico de Costa Rica

Escuela de Computación

Campus Tecnológico Local San José

Taller de Programación

Tarea #5

Algoritmos de Ordenamiento

Profesor:

Mauricio Avilés

Estudiante:

José Manuel Granados Villalobos (2022049007)

Algoritmos de Ordenamiento

1. Bubble Sort:

Es uno de los más sencillos, sin embargo, es ineficiente comparado al resto. Este, al trabajar mediante comparaciones, lo que realiza es una comparación entre el primer elemento de la lista (i = x) y el siguiente (i + 1 = y) buscando si "i" es menor o igual a "i + 1", si es de esta forma, pues están ordenados, de lo contrario es necesario realizarse un intercambio (i = y, i + 1 = x) y se continúa de esta forma, repitiéndose constantemente hasta el final de la lista. En las pruebas se llega hasta la cantidad de elementos que no sobrepasen los 10 segundos por prueba.

Características:

- In-place
- Estable
- Utiliza comparaciones
- Basado en intercambios
- Implementa la iteración

Bubble Sort	
Cantidad de elementos	**Tiempo promedio de 10 pruebas**
100	0.000712442398071289
600	0.035685038566589354
1100	0.14932260513305665
1600	0.3600119352340698
2100	0.5657344341278077
2600	1.038810920715332
3100	1.3559424877166748
3600	1.8546303510665894
4100	2.348458671569824
4600	2.9581933498382567
5100	3.9686230182647706
5600	4.8704866647720335
6100	4.5480228662490845
6600	5.94755322933197
7100	6.75989408493042
7600	8.004982590675354

2. Insertion Sort:

Este es eficiente con listas que ya estén en su mayoría ordenadas o cortas. Este funciona dividiendo la lista en parte ordenada y desordenada, así, se irán haciendo comparaciones de los números que están en la parte desordenada con los que están en la ordenada; de esta forma, la parte ordenada irá creciendo y la desordenada disminuirá hasta que no haya más números que ordenar. En las pruebas se llega hasta la cantidad de elementos que no sobrepasen los 10 segundos por prueba.

Características:

- In-place
- Estable
- Utiliza comparaciones
- Basado en intercambios
- Implementa la iteración

Insertion Sort	
Cantidad de elementos	**Tiempo promedio de 10 pruebas**
100	0.0005983352661132812
600	0.0368973970413208
1100	0.1409754276275635
1600	0.2795954465866089
2100	0.4748647451400757
2600	0.8068522691726685
3100	1.2611597299575805
3600	1.7814953565597533
4100	2.244008755683899
4600	3.125412440299988
5100	3.258039712905884
5600	3.7487046241760256
6100	4.593561339378357
6600	4.85471920967102
7100	6.168131041526794
7600	6.710570764541626
8100	7.818587422370911
8600	9.028725075721741

3. Selection Sort:

Un algoritmo que de igual forma tampoco es muy eficiente, más que todo para listas de gran tamaño. Este trabaja buscando el menor de la lista, una vez se tiene, se requiere la posición de este y se intercambia con el primero de la lista, esto se repetirá con los sobrantes de la lista hasta llenar nuevamente esta lista, pero ordenada. En las pruebas se llega hasta la cantidad de elementos que no sobrepasen los 10 segundos por prueba.

Características:

- In-place
- Estable
- Utiliza comparaciones
- Basado en intercambios
- Implementa la iteración

Selection Sort	
Cantidad de elementos	**Tiempo promedio de 10 pruebas**
100	0.0006958246231079102
600	0.020000624656677245
1100	0.06812078952789306
1600	0.165253210067749
2100	0.2785155773162842
2600	0.3974995851516724
3100	0.5372956752777099
3600	0.7281533718109131
4100	0.9447955131530762
4600	1.3766152381896972
5100	1.9164446592330933
5600	2.320447015762329
6100	2.6114237785339354
6600	3.0048901557922365
7100	2.8640136241912844
7600	3.668128418922424
8100	3.985926961898804
8600	4.849565100669861
9100	5.08129050731659
9600	5.556162309646607
10100	6.602257537841797

4. Merge Sort:

Está basado en la técnica de divide y vencerás, por lo tanto, divide la cantidad de sus elementos y los asigna en dos sublistas, así consecutivamente hasta que solo existan sublistas de un elemento, en este punto se comienza a devolver para poder rellenar cada sublista en orden, se unen. En las pruebas se llega hasta la cantidad de elementos que no sobrepasen los 10 segundos por prueba.

Características:

- No es In-place
- Estable
- Utiliza comparaciones
- Basado en intercambios
- Implementa la recursión

Merge Sort	
Cantidad de elementos	**Tiempo promedio de 10 pruebas**
100	0.0002958059310913086
1100	0.005093026161193848
2100	0.01026928424835205
3100	0.01650371551513672
4100	0.02422952651977539
5100	0.028622889518737794
6100	0.03765826225280762
7100	0.05542147159576416
8100	0.05515244007110596
9100	0.08144323825836182
10100	0.07255415916442871
11100	0.0920494794845581
12100	0.09843027591705322
13100	0.09905068874359131
14100	0.1186997652053833
15100	0.12931745052337645
16100	0.14696946144104003
17100	0.1411673307418823
18100	0.14943060874938965
19100	0.16945037841796876
20100	0.16474432945251466
21100	0.17444732189178466
22100	0.22825486660003663

5. Quick Sort:

Nuevamente es basado en la técnica de divide y vencerás, consiste en seleccionar un pivote, puede ser el primer, último o medio elemento (este último más recomendable); a partir de este pivote, se crean dos sublistas, una con los menores e iguales y otra con los mayores, se sigue repitiendo de forma recursiva hasta llegar a sublistas de un elemento, a partir de acá se concatenan creando una lista del tamaño original pero ordenada. En las pruebas se llega hasta la cantidad de elementos que no sobrepasen los 10 segundos por prueba, además, como el quick sort es mucho más eficiente que el resto, se comenzará desde una cantidad de elementos más alta y el incremento será de 5000 elementos.

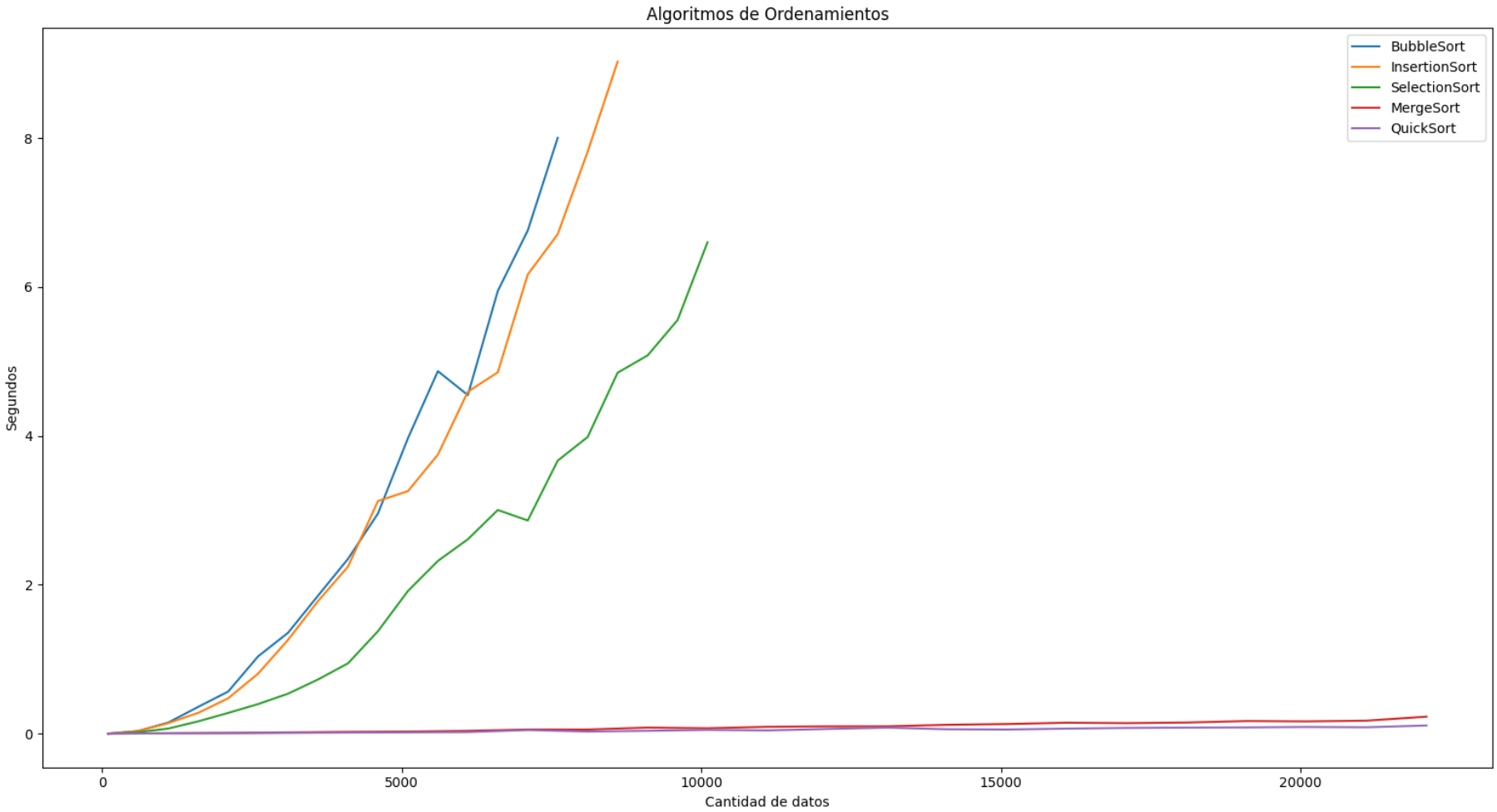
Características:

- No In-place
- Inestable
- Utiliza comparaciones
- Basado en intercambios
- Implementa la recursión

Quick Sort	
Cantidad de elementos	**Tiempo promedio de 10 pruebas**
100	0.0001035451889038086
1100	0.004188108444213867
2100	0.005894136428833008
3100	0.010275721549987793
4100	0.014898324012756347
5100	0.017302846908569335
6100	0.02121114730834961
7100	0.0484844446182251
8100	0.02841627597808838
9100	0.03837199211120605
10100	0.05039424896240234
11100	0.043699049949646
12100	0.06291835308074951
13100	0.08056116104125977
14100	0.05893640518188477
15100	0.055700135231018064
16100	0.06751608848571777
17100	0.0768383264541626
18100	0.08120915889739991
19100	0.08400278091430664
20100	0.0898050308227539

21100	0.0856848955154419
22100	0.10981917381286621

Gráfico de rendimiento



Análisis

Es claro que los tres primeros (bubble, insertion y selection sort) son algoritmos que, aunque presentan distintas formas de ordenar listas, terminan siendo muy semejantes con respecto a eficiencia. De acuerdo con los promedios generados podemos notar gran similitud en los resultados del bubblesort y el insertionsort, teniendo números con milésimas de segundo más bajas comparadas con el selectionsort.

Además, recordemos que se asignó un tiempo límite para las pruebas, el cual era de 10 segundos, por lo tanto, si nos fijamos en la cantidad final de elementos analizados, encontramos que, de estos tres algoritmos, el que se comportó de manera más eficaz fue el selectionsort, llegando a analizar cinco veces más que el bubblesort y tres veces más que el insertionsort.

Por otra parte, el mergesort y quicksort, de antemano se sabe que son varias veces mejores que los tres anteriores, pues estos son capaces de analizar hasta 100.000 elementos en poco tiempo (en el caso del quicksort, hasta millones). En nuestro caso es sencillo de ver la diferencia, pues mientras que los tres primeros algoritmos no eran capaces de superar los 10 mil elementos sin excederse de 10 segundos, el merge y quicksort podían duplicar la cantidad de elementos sin siquiera llegar a 1 segundo.

Hipótesis

Realizaremos distintas hipótesis incluyendo la capacidad de hardware del equipo en el que el experimento fue realizado.

- De primera entrada, los datos promedio adquiridos tienen un valor usualmente correcto, pues presentan tiempos que se presentarían en la mayoría de los equipos, sin embargo, podemos notar desfases en algunos algoritmos, esto supondremos que será debido al procesador de tan baja gama que contiene el equipo, pues es una bastante antiguo (Intel Core i3).
- Por otra parte, podremos también culpar a la gran cantidad de programas que se mantenían abiertos a la hora de realizar el estudio, situación que tampoco pudo ser mejorada por el procesador.
- Por último, en este caso se podría llegar a creer que una de las causas de ciertos fallos y cambios en los diversos promedios sea causado simple y sencillamente por el programa utilizado.