Algoritmos y Estructura de Datos

Grupo No. 5

Jonathan Aguirre 14349

Boggdan Barrientos 14484

José Corona 14417

Julio Diéguez 14475

Laboratorio No.3:

Para este laboratorio se nos pedía seleccionar y programar cuatro tipos distintos de sorts de ordenamiento, estos fueron Insertion sort, Selection sort, Mergesort y Quicksort. Tomando en cuenta los datos de la página de Big-O se pueden analizar el mejor caso, el caso promedio y el peor de los casos para cada uno de los cuatro sorts.

Como marguen de calificación se utiliza la siguiente escala:

Algoritmo	Tiempo de ejecución			Uso de memoria
	Mejor caso	Caso promedio	Peor caso	Peor caso
Quicksort	Malo	Malo	Horrible	Bueno
Mergesort	Malo	Malo	Malo	Justo
Insertion Sort	Bueno	Horrible	Horrible	Bueno
Selection sort	Horrible	Horrible	Horrible	Bueno

Utilizando estos datos como valores teóricos del tiempo de ejecución y uso de memoria de cada uno de los cuatro sort se puede calificar al Mergesort como el mejor de los cuatro sort, a pesar de que es el que mayor cantidad de memoria utiliza en el peor de los casos de ejecución del programa. Como el menos eficiente se puede tomar el Selecction sort que no es mejor que ninguno de los otros tres sort en el mejor, promedio o peor caso de ejecución.

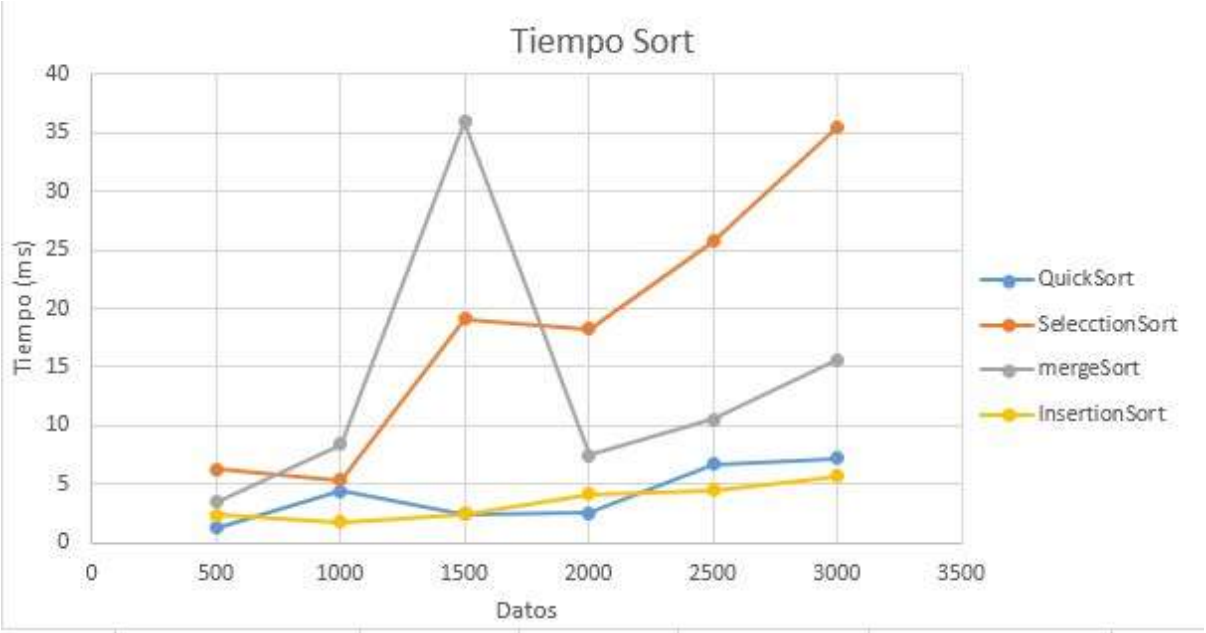
En la gráfica no.1 podemos observar que el QuickSort y el insertionSort fueron los mejores en ordenar los 3000 números esto se debe a que el QuickSort funciona mejor cuando no se repiten los números y como se utilizó un random que generara números entre 0 a 3000, no habían tantos números repetidos. Es por esto el QuickSort es considerado como el más rápido algoritmo de ordenamiento en este caso.

En el caso del Selectionsort es el que presenta los perores tiempos, los magas grandes, ya que tiene un orden en la Big-O de n^2.

En el MergeSort para 1500 datos se tuvo el tiempo más altos para el ordenamiento, esto se debe que este tipo de sort es más estable y se utiliza para el manejo de medios secuenciales, es decir, que es

mejor para el ordenamiento de listas enlazadas. El manejo de números repetidos para cada uno de los métodos pudo haber afectado al tiempo de ejecución.

Grafica 1. Tempo de cada Sort



Cuadro No 1. Tiempo de los sort para 100000 datos

	No.Datos	Tiempo (ms)
QuickSort	100000	74
SelectionSor	100000	38,376
MergeSort	100000	285
InsertionSor	100000	5770

En el cuadro 1 podemos observar que el Quicksort es el mejor para sort para ordenar los datos ya que tiene n log n en la Big-O.

Referencias:

Rowel Eric, Array Sorting Algoritm. [Recuperado el 6/8/2015] de: http://bigocheatsheet.com/