Análisis de tiempos de ejecución de algoritmos de ordenamiento

Antony Leonardo Picado Alvarado
C15939

***Resumen* – En este trabajo se realiza un análisis de tiempos de ejecución de varios algoritmos, para esto se utilizó el lenguaje de programación C++ para representar dichos algoritmos, los resultados fueron los esperados en cuanto al comportamiento de los algoritmos, por lo tanto, se concluye que el algoritmo *QuickSort* como su nombre lo indica es en cuanto a tiempo de ejecución, es el más rápido de todos.**

I. Introducción

En este trabajo se realizó un análisis del tiempo de ejecución de tres algoritmos de ordenamiento, *SelectionSort*, *InsertionSort* y *MergeSort*, HeapSort, QuickSort y RadixSort los cuales mediante un programa en C++ se les toma el tiempo de ejecución de varias corridas, almacenando estos datos en una tabla (ver cuadro I) y se utilizan estos datos para crear gráficos en donde se logra apreciar la diferencia entre estos algoritmos en cuanto al tiempo de ejecución.

II. Metodología

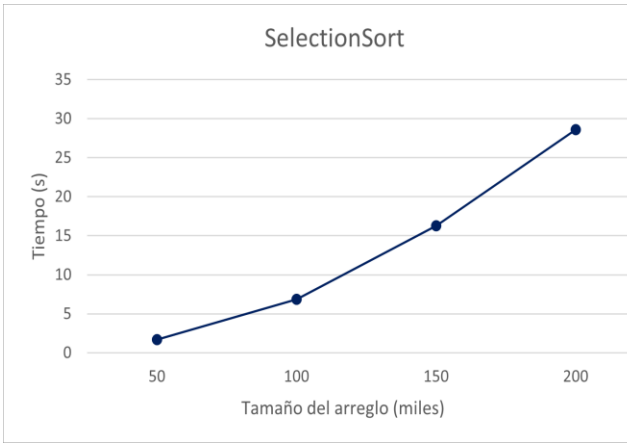
Para lograr lo propuesto, se utilizó el lenguaje de programación C++ para recrear los algoritmos utilizando el pseudocódigo propuesto en el libro *Introduction to Algorithms* de Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest y Clifford Stein, y para obtener su tiempo de ejecución a la hora de ordenar arreglos de números aleatorios de tamaño 50 000, 100 000, 150 000 y 200 000, una vez que se obtiene el tiempo de ejecución de los algoritmos se prosigue a representar la curva de tiempo de cada algoritmo al someterlo a ordenar arreglos de números aleatorios de dichos tamaños.

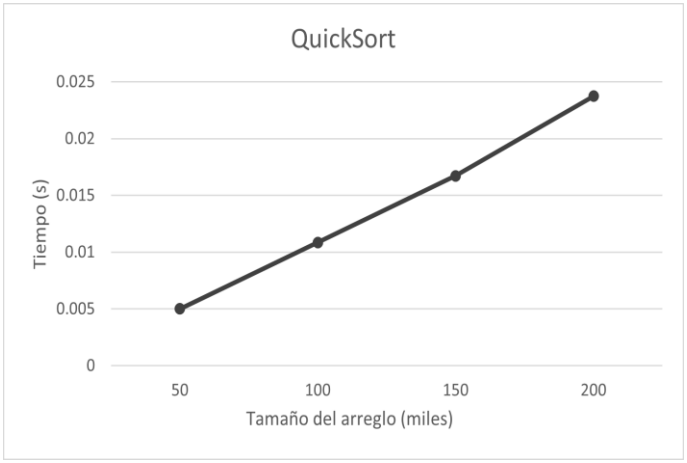
Cuadro I
TIEMPO DE EJECUCIÓN DE LOS ALGORIMTOS

		Tiempo (s)			
		Corrida			
Algoritmo	Tam(k)	1	2	3	Prom.
InsertionSort	50	1.79733219	1.795582801	1.829665125	1.80752671
	100	7.387323845	7.345180294	7.288742683	7.34041561
	150	16.38680335	16.44529797	16.43396295	16.4220214
	200	29.27701881	29.41890797	29.20967833	29.3018684
SelectionSort	50	1.696252785	1.680788214	1.657647053	1.67822935
	100	6.730436145	6.760718556	7.083140523	6.85809841
	150	16.97464658	16.70009971	15.160805	16.2785171
	200	28.19512821	29.71600643	27.87325163	28.5947954
MergeSort	50	0.007068701	0.006935561	0.007184437	0.0070629
	100	0.01484104	0.014886255	0.01496905	0.01489878
	150	0.022840706	0.023071168	0.022988864	0.02296691
	200	0.031437812	0.03154311	0.0313943	0.03145841
HeapSort	50	0.009617732	0.009811114	0.009759488	0.00972944
	100	0.020679057	0.021632001	0.023313899	0.02187499
	150	0.032277697	0.032075269	0.034606396	0.03298645
	200	0.050985765	0.0455178	0.050180638	0.04889473
QuickSort	50	0.005006862	0.004916002	0.005116718	0.00501319
	100	0.010695099	0.011255299	0.010604149	0.01085152
	150	0.017071774	0.016392995	0.016719158	0.01672798
	200	0.023587739	0.024057831	0.023571691	0.02373909

III. Resultados

Los tiempos de ejecución obtenidos luego de 3 corridas con arreglos de tamaño 50 000, 100 000, 150 000 y 200 000 que se muestran en el cuadro I, se logra observar un comportamiento similar entre los algoritmos *InsertionSort* y *SelectionSort* esto correspondiendo a su tiempo de ejecución de $\theta(n^2)$ en ambos algoritmos, por otra parte, los algoritmo *MergeSort, QuickSort y RadixSort* obtiene una reducción importante en su tiempo de ejecución, no durando más de 1 segundo ordenando 200 000 números aleatorios almacenados en un arreglo de número aleatorios, lo cual obedece a su tiempo de ejecución de $\theta(n \log n)$.





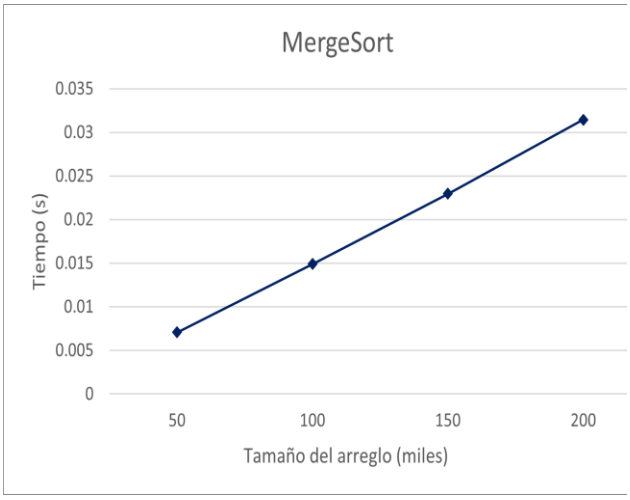


Figura 1. Tiempos promedio de ejecución de los algoritmos *SelectionSort*, *InsertionSort* y *MergeSort*, *HeapSort, QuickSort y RadixSort*

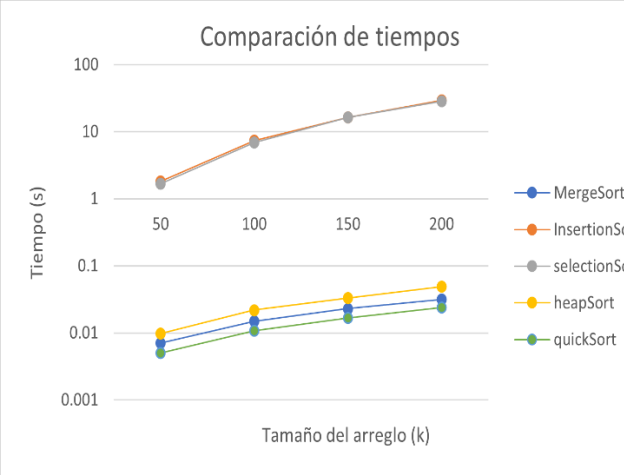


Figura 2. Gráfico comparativo de los tiempos promedio de ejecución de los algoritmos.

IV. Conclusiones

A partir de los resultados se concluye que los 6 algoritmos presentados en esta primera parte concuerdan con los resultados esperados, en donde, tanto los algoritmos InsertionSort y SelectionSort tienen un comportamiento parecido al obedecer ambos a su tiempo de ejecución $\theta(n^2)$, en el caso de los tiempos promedios de ambos algoritmos, SelectionSort llega a ser algo superior en promedio hablando del tiempo de ejecución (véase el cuadro I) en comparación con InsertionSort, en donde este tiene mejores tiempos promedios, por otra parte, si se habla de superioridad en cuando a rapidez, MergeSort, QuickSort y RadixSort son muy superiores a los demás algoritmos antes analizados, en donde no llega a durar siquiera un segundo (0.02s) en el caso de QuickSort en ordenar 200 000 números aleatorios, en comparación del promedio de 28.948 segundos de los primeros dos algoritmos analizados, de estos últimos 4 QuickSort es algo superior en tiempos de ejecución en comparación a MergeSort, HeapSort y RadixSort obteniendo tiempos promedios menores ordenando 50k, 100k, 150k y 200k elementos aleatorios.