I Tarea Programada

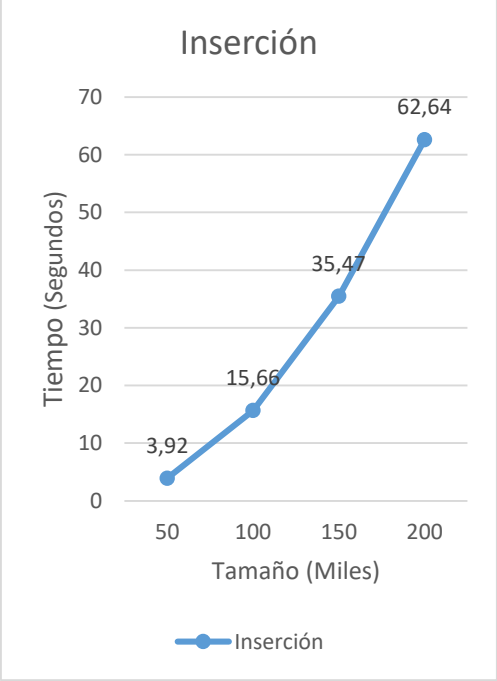
Cuadro 1

		Tiempo (s)			
Algoritmo	**Tam. (k)**	**Corrida 1**	**Corrida 2**	**Corrida 3**	**Promedio**
Selección	50	4.36	4.36	4.36	4.36
	100	17.3	17.3	17.4	17.33
	150	39.0	39.0	39.1	39.03
	200	69.4	69.4	69.4	69.4
Inserción	50	3.91	3.95	3.91	3.92
	100	15.78	15.60	15.61	15.66
	150	35.62	35.51	35.29	35.47
	200	62.7	62.55	62.68	62.64
Mezcla	50	0.014	0.014	0.015	0.014
	100	0.030	0.031	0.030	0.030
	150	0.046	0.045	0.045	0.045
	200	0.062	0.062	0.062	0.062
Heapsort	50	0.031	0.031	0.031	0.031
	100	0.046	0.046	0.046	0.046
	150	0.078	0.078	0.078	0.078
	200	0.094	0.094	0.094	0.094
Quicksort	50	0.012	0.012	0.012	0.012
	100	0.029	0.029	0.029	0.029
	150	0.041	0.041	0.042	0.041
	200	0.058	0.057	0.057	0.057
Radixsort	50	0.031	0.031	0.031	0.031
	100	0.062	0.062	0.062	0.062
	150	0.10	0.10	0.11	0.103
	200	0.14	0.14	0.15	0.143
Mediana	50	0.002	0.002	0.002	0.002
	100	0.005	0.005	0.005	0.005
	150	0.007	0.008	0.007	0.007
	200	0.011	0.011	0.011	0.011

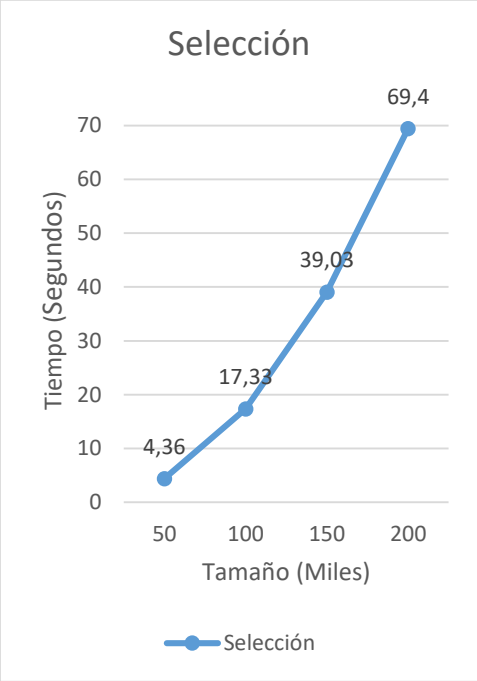
Con base en los tiempos anteriores podemos notar que al ordenar 50 mil elementos en cada corrida para cada algoritmo sus tiempos son muy similares, al pasar a 100 mil elementos cada algoritmo se comporta de manera similar en sus tres ejecuciones. Al pasar a 150 mil elementos el sesgo entre el tiempo promedio de ejecución de los algoritmos inserción y selección es un poco más grande que de los dos tamaños anteriores. Al ordenar 200 mil elementos vemos que en los algoritmos de inserción y selección su tiempo de duración en sus tres ejecuciones

es muy similar, pero comparando este tamaño con el primer tamaño establecido en cada uno de estos dos algoritmos podemos denotar la enorme diferencia en tiempos. De lo anterior podemos deducir que entre mayor cantidad de elementos se tengan que ordenar mayor diferencia en tiempos habrá, solo en el caso de los algoritmos Mezcla, Heapsort y Quicksort, Radixsort, Mediana notamos que sus tiempos promedios en sus cuatro corridas tienen un comportamiento muy lineal(similar).

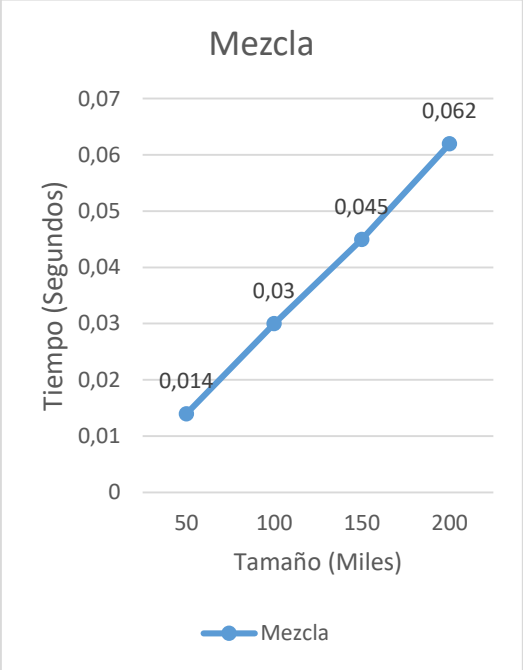
Cuadro 2



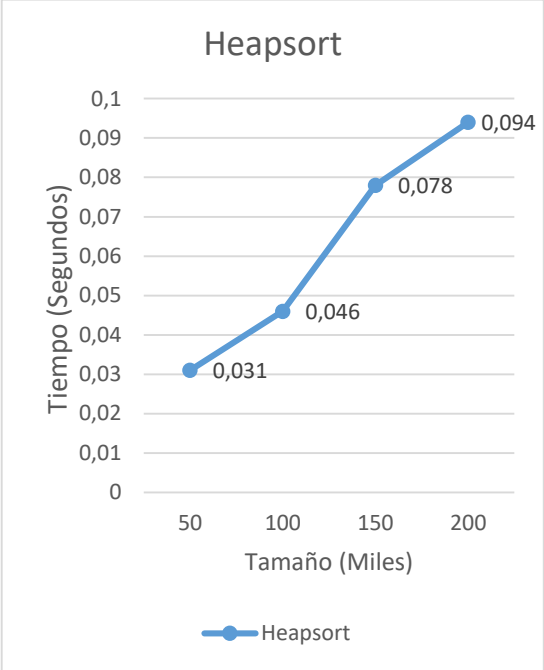
Cuadro3



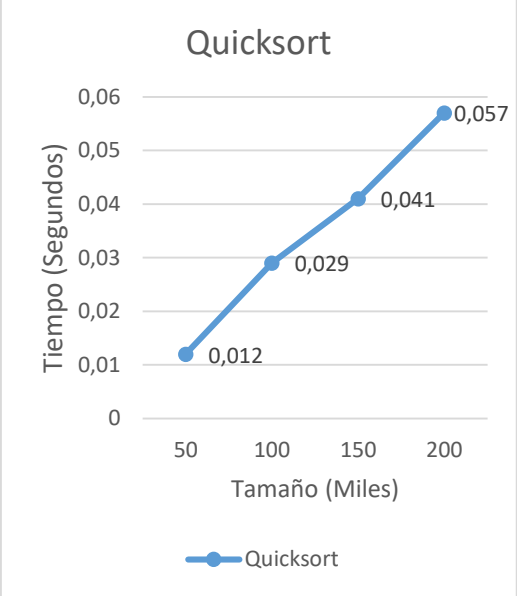
Cuadro 4



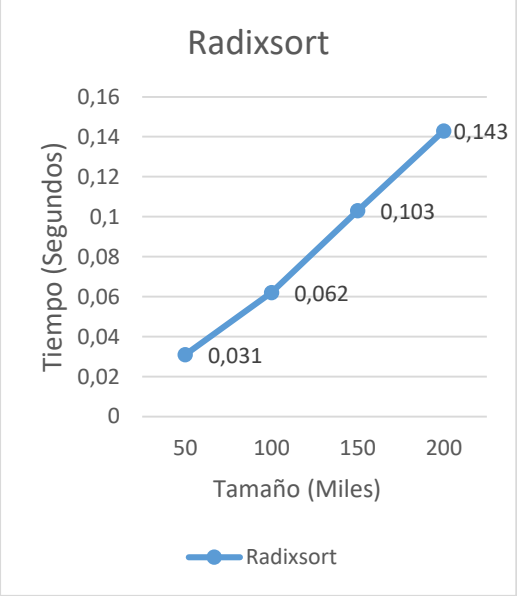
Cuadro 5



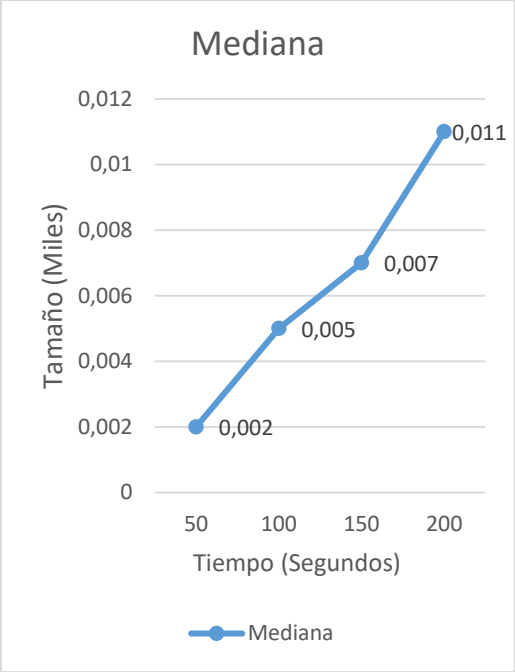
Cuadro 6



Cuadro 7



Cuadro 8



- Del cuadro 2 podemos notar como se forma una parte de una parábola, lo que comparando el tiempo de ejecución del algoritmo de selección que es $\theta(n^2)$ con la gráfica resultante se evidencia un comportamiento muy similar ya que la función n^2 es parabólica como el grafico del cuadro 2.

- Del cuadro 3 podemos notar como se forma una parte de una parábola, lo que comparando el tiempo de ejecución del algoritmo de inserción que es $\theta(n^2)$ con la gráfica resultante se evidencia un comportamiento muy similar ya que la función n^2 es parabólica como el grafico del cuadro 3.

- Del cuadro 4 podemos notar una gráfica casi lineal, la cual si la comparamos con el tiempo de ejecución del algoritmo de mezcla que es θ(n

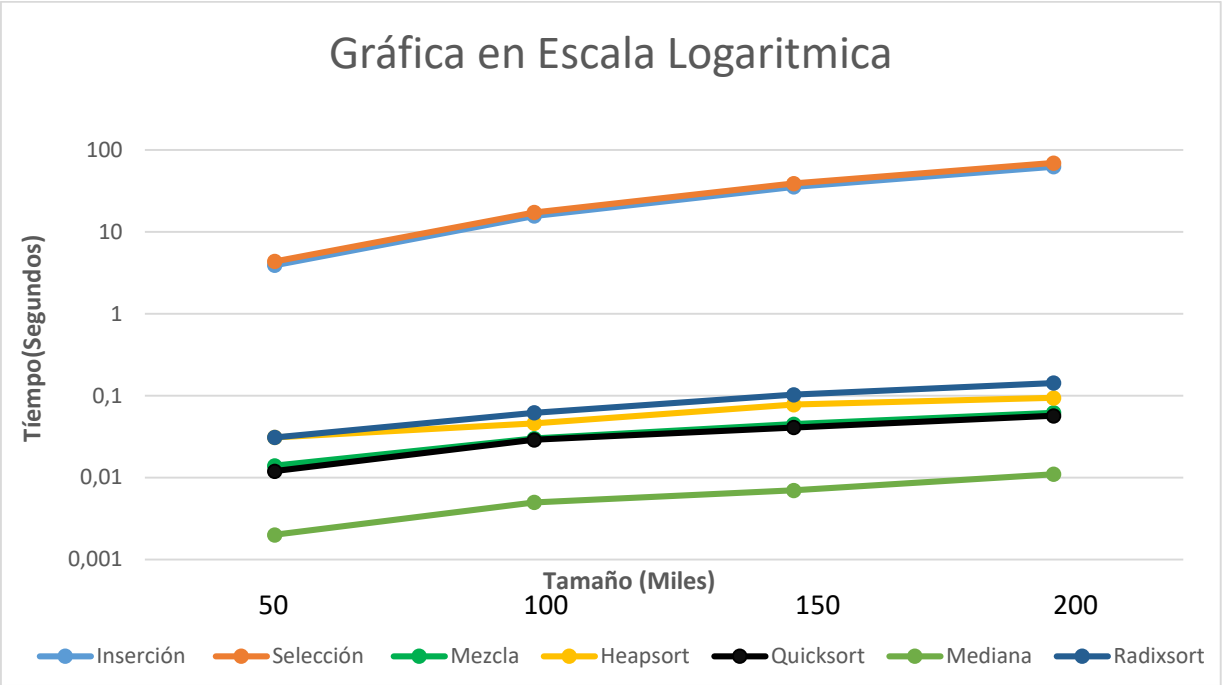
log n) evidenciamos que la gráfica y la función n log n se comportan de forma muy similar.

- Del cuadro 5 podemos notar una gráfica que tiende a ser lineal, la cual si la comparamos con el tiempo de ejecución del algoritmo de Heapsort que es θ(n log n) evidenciamos que la gráfica y la función n log n se comportan de forma muy similar.

- Del cuadro 6 podemos notar una gráfica que tiende a ser lineal, la cual si la comparamos con el tiempo de ejecución del algoritmo de Quicksort que es θ(n log n) evidenciamos que la gráfica y la función n log n se comportan de forma muy similar.

- Del cuadro 7 podemos notar una gráfica lineal, la cual si la comparamos con el tiempo de ejecución del algoritmo de Radixsort que es θ(n) evidenciamos que la gráfica y la función n se comportan de forma muy similar.

- Del cuadro 8 notamos que el algoritmo de búsqueda de la mediana es una función lineal la cual al compararse con el tiempo de ejecución de este algoritmo el cual es θ(n), se evidencia que la gráfica y la función n se comportan muy similares.



En el cuadro en escala logarítmica se ve como el algoritmo de selección es el que tarda más tiempo en promedio en ordenar los diversos tamaños de los vectores, el que le sigue en cuanto a duración es el algoritmo de inserción y el que ordena en menor tiempo de duración es el Quicksort ya que su grafica está por debajo de los otros algoritmos y notamos como todos los algoritmos $\theta(n \log n)$ se comportan muy similar es decir no hay una diferencia muy notable entre ellos, el más rápido en dar el resultado es el de búsqueda de la mediana, pero este puede que no ordene todo el arreglo.

Podemos ver que la relación entre las curvas de los algoritmos selección y inserción es la esperada ya que los algoritmos cuadráticos sobrepasan por mucho a los n log n.

La relación entre los algoritmos n log n es la que esperábamos ya que sus curvas como se puede observar están muy cerca unas de otras.

Por otro lado, podemos ver que la relación del Radixsort con la curva de la gráfica no es la esperada ya que su curva debería ser la más baja, sin embargo, por factores como el uso de la memoria cache hace que en la práctica se evidencie que no es el algoritmo más rápido.

Por ultimo si observamos la curva de la búsqueda de la mediana coincide con lo esperado ya que es una función lineal y como no precisamente necesita ordenar todo el arreglo es más rápido que todos los algoritmos anteriores.