UNIVERSIDAD SIMÓN BOLÍVAR

DEPARTAMENTO DE COMPUTACIÓN Y TECNOLOGÍA DE LA INFORMACIÓN

LABORATORIO DE ALGORITMOS Y ESTRUCTURAS II (CI2692)

PROYECTO 1:

Estudio experimental de algoritmos de ordenamiento

POR:
Mario Quintero (13-11148)
Carolina Rivas (13-11209)

Enero-Marzo 2017

El presente proyecto pide la implementación y prueba de los algoritmos de ordenamiento ***dual pivot quicksort***, ***quicksort with three-way-partition***, ***introsort*** y ***median-of-three quicksort***, junto a los ya anteriormente implementados, ***quicksort*** **aleatorio**, ***mergesort*** y ***heapsort*** para distintos tipos de arreglos (de números aleatorios, ordenado ascendente o descendentemente, de ceros y unos, de un número real entre cero y uno, etc.) y distintos números de elementos (desde 4096 hasta 65536).

En el siguiente informe se muestran tantos cuadros como tipos de pruebas, donde se expresan, en función del número de elementos y del tipo del algoritmo, el tiempo promedio de corrida del programa. Además de estos cuadros de datos, se mostrarán gráficas a partir de los resultados de la tabla donde se evidenciará la tendencia de la eficiencia de estos algoritmos.

A partir de estas matrices de datos y gráficas, se hará un análisis de cada una de las pruebas realizadas y otro análisis global de todos los resultados obtenidos, al igual que una comparación de los resultados del proyecto con lo esperado por la teoría.

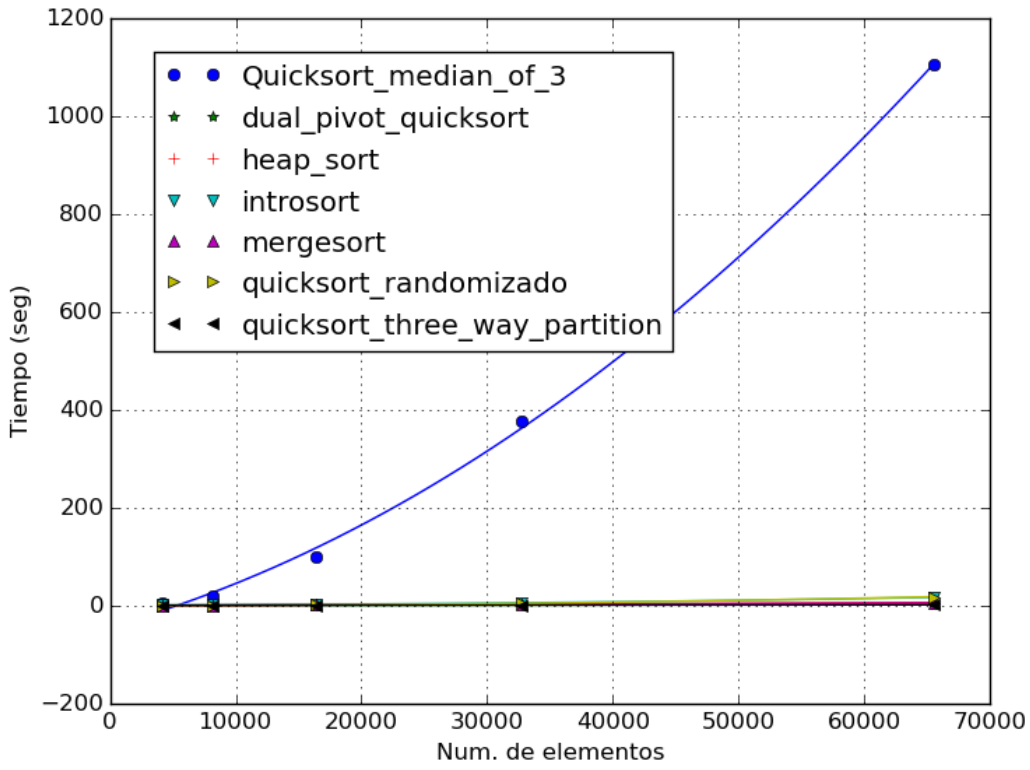
El equipo en el que se realizaron las pruebas fue una laptop Toshiba Satellite C55-C, con memoria RAM de 7,7 GiB, procesador Intel® Core™ i5-5200 U CPU de 2.20GHz × 4, y sistema operativo Ubuntu 15.10 de 64 bits.

RESULTADOS Y ANÁLISIS POR CASO

A continuación, se mostrarán los cuadros y las gráficas por cada conjunto de datos:

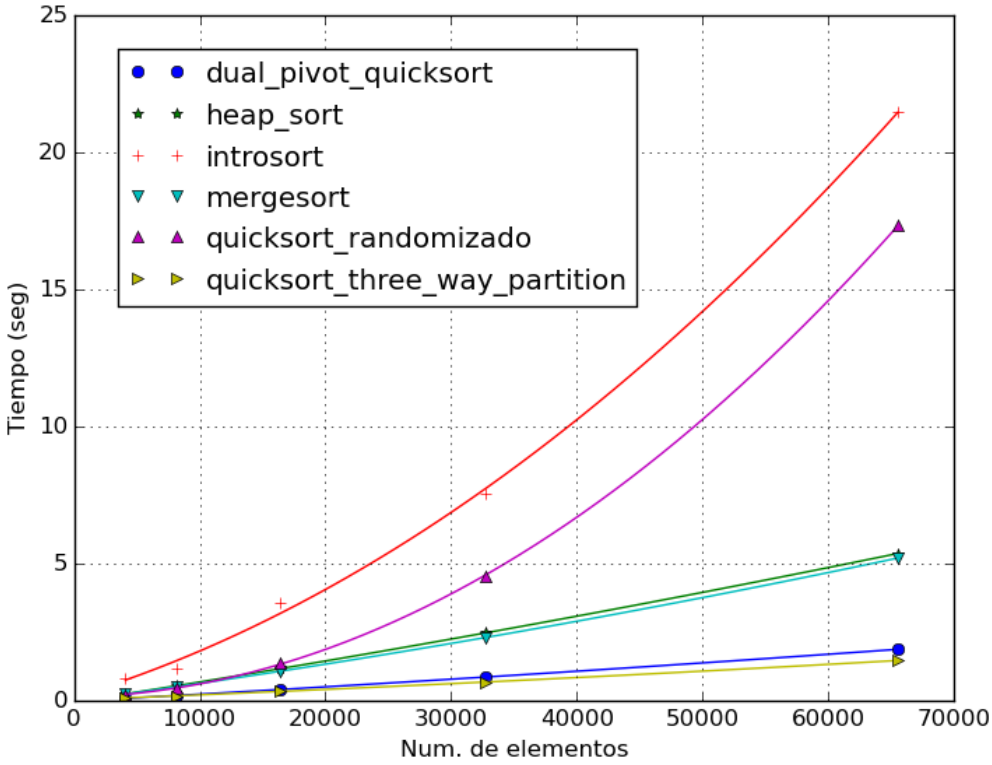
1. **Enteros aleatorios:** números enteros comprendidos en el intervalo [0,500] generados aleatoriamente

N	Mergesort	Heapsort	QS Random	Med-of-3 QS	Introsort	Dual pivot QS	QS 3-way
4096	0.243 s	0.236 s	0.167 s	3.953 s	0.771 s	0.083 s	0.081 s
8192	0.482 s	0.507 s	0.454 s	17.659 s	1.126 s	0.170 s	0.163 s
16384	1.090 s	1.115 s	1.390 s	68.098 s	3.484 s	0.376 s	0.331 s
32768	2.800 s	2.673 s	4.886 s	316.924 s	5.745 s	0.812 s	0.721 s
65536	4.845 s	5.311 s	16.628 s	1254.716	21.012 s	1.966 s	1.345 s



Para la prueba de enteros aleatorios, el único algoritmo que presentó un orden de crecimiento cuadrático fue *quicksort median of 3*, mientras que todo los demás algoritmos a primera vista parecen haberse comportado de manera lineal o cuasi-lineal. El comportamiento de median of 3 puede deberse a la manera en que fue implementado el cálculo de la media de tres números y *partition*, ya que no hay una causa genérica para tal comportamiento.

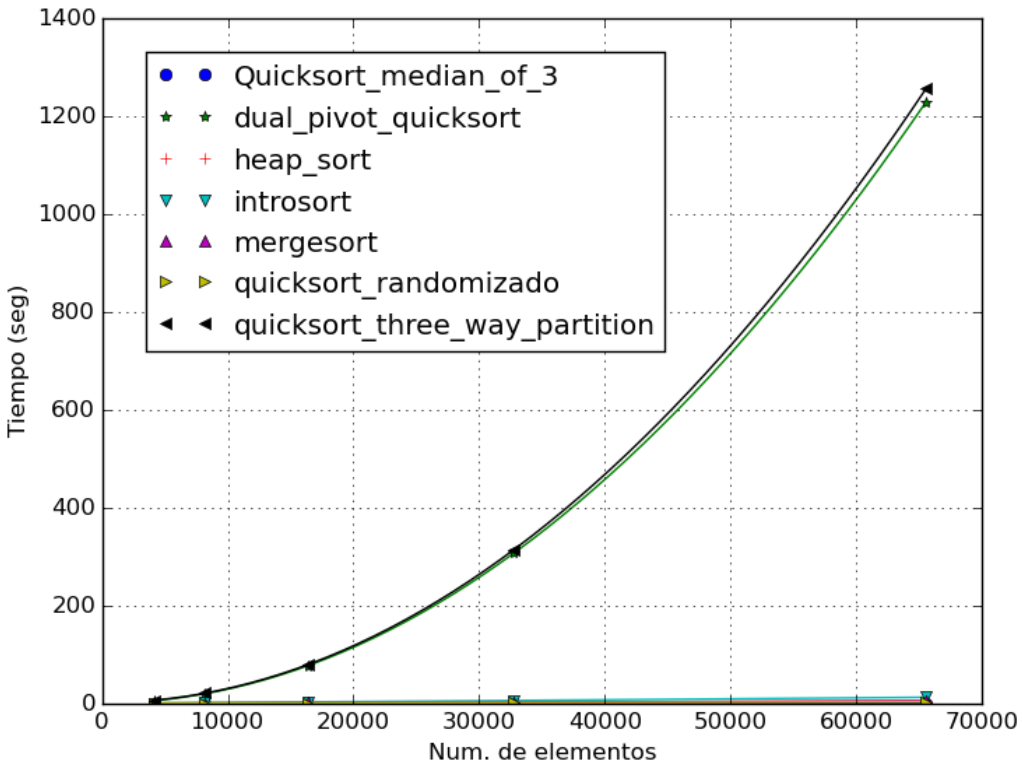
A continuación se muestra una gráfica del comportamiento de los algoritmos que se comportaron casi linealmente:



Se puede observar que entre estos algoritmos hay unos que no se comportan de manera O(nlogn), con *quicksort three way partition* teniendo la mejor eficiencia, muy de cerca al *dual pivot quicksort*, al igual que *mergesort* y *heapsort*.

2. **Orden inverso:** si el tamaño del arreglo es N, entonces el arreglo contendrá la secuencia N, N-1,…, 2, 1.

N	Mergesort	Heapsort	QS Random	Med-of-3 QS	Introsort	Dual pivot QS	QS 3-way
4096	0.349 s	0.222 s	0.131 s	0.026 s	0.564 s	5.144 s	4.944 s
8192	0.611 s	0.496 s	0.299 s	0.052 s	1.269 s	19.487 s	19.751 s
16384	1.275 s	1.084 s	0.643 s	0.105 s	2.705 s	78.381 s	77.579 s
32768	2.173 s	2.293 s	1.299 s	0.208 s	5.647 s	311.454	311.519 s
65536	4.552 s	5.121 s	2.880 s	0.414 s	12.160 s	1223.636 s	1296.936 s

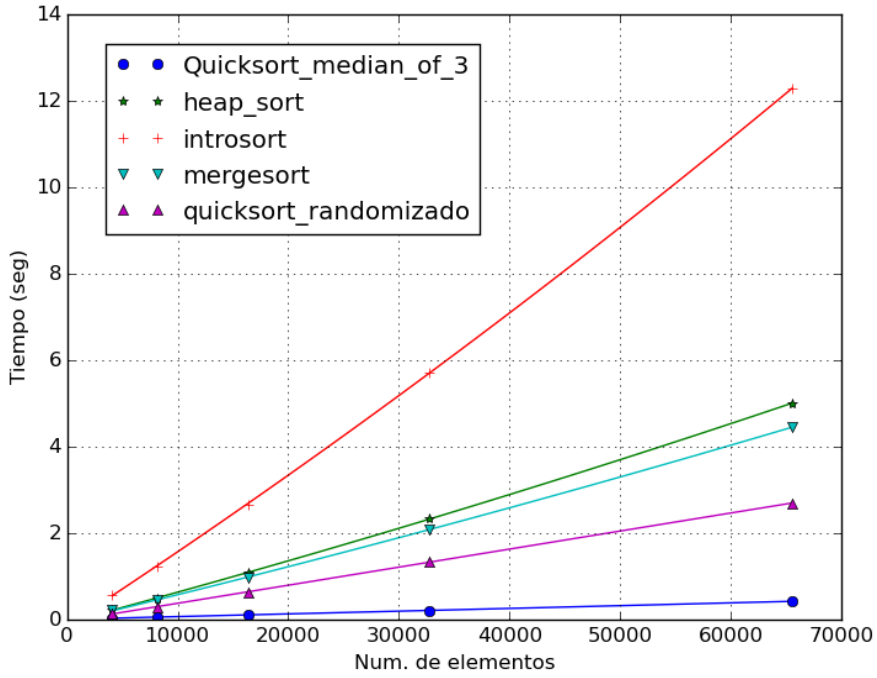


Para este caso, los algoritmos con un orden de crecimiento cuadrático son *dual pivot quicksort* y *quicksort three way partition*.

Se sabe que el *dual pivot quicksort* en su peor caso tiene complejidad cuadrática y que esta ocurre cuando el pivote es el elemento más pequeño o más grande del subarreglo. Como el arreglo está ordenado inversamente, efectivamente se tuvo que haber escogido como pivote, esta vez, el elemento más pequeño de cada subarreglo.

Con *quicksort three way partition*, se puede inferir que su comportamiento se debe a la elección de un pivote que resulta ser el menor número de todo el arreglo, ya que el arreglo está en el orden inverso y por ende, realiza un mayor número de comparaciones dentro de la corrida.

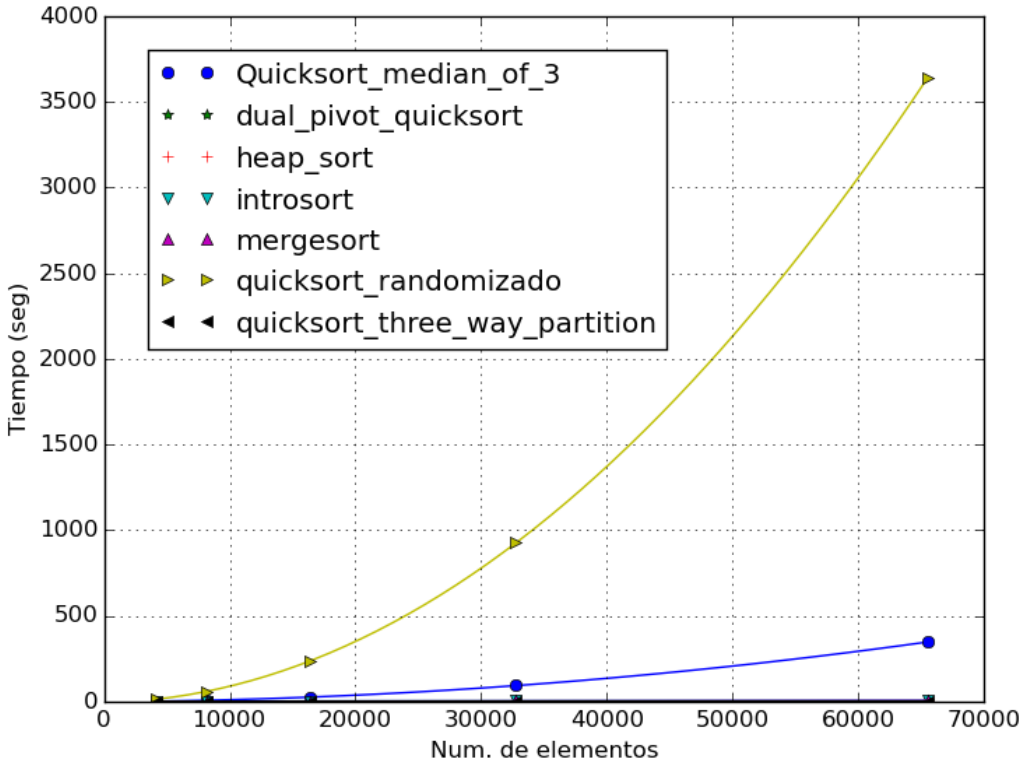
A continuación se observa detalladamente el comportamiento de los algoritmos que aparentan ser lineales:



Se puede observar que nuevamente *introsort* fue el algoritmo de ordenamiento más ineficiente entre estos algoritmos, sin embargo, en este caso se comporta de manera O(nlogn) al igual que todos los demás. En este tipo de arreglo el más eficiente fue el *quicksort median of 3* que tiene un comportamiento aparentemente lineal (O(n)) debido a que en cierto punto el algoritmo evita un problema de comportamiento cambiando a *heapsort* cuando el número de subproblemas llega a cierto límite.

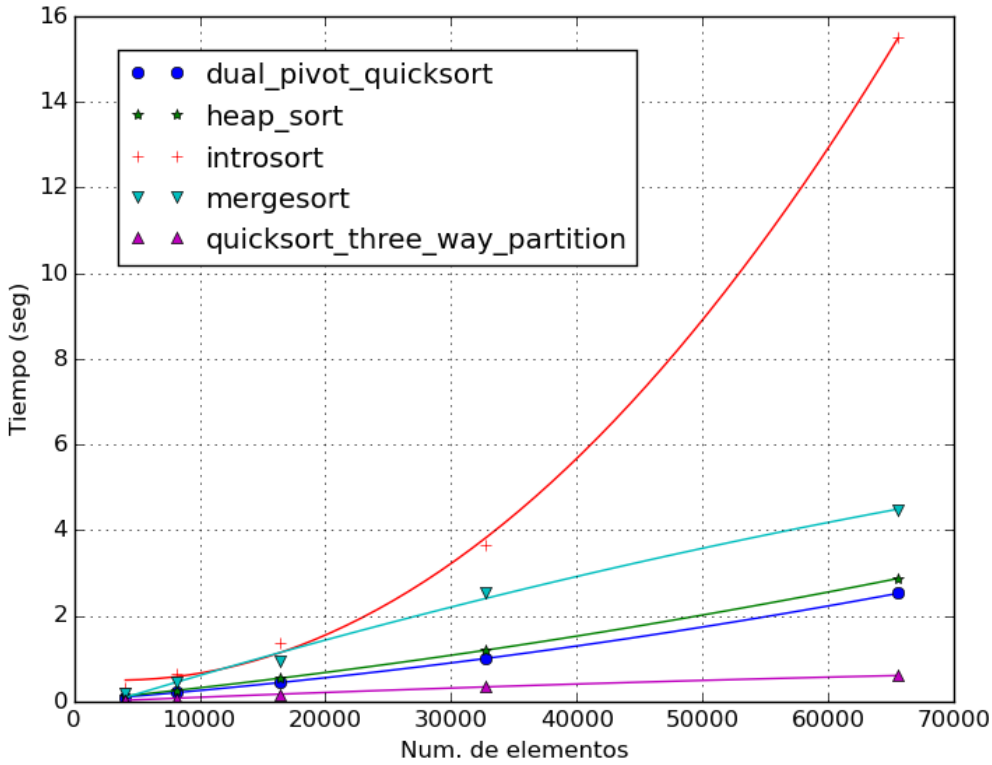
3. **Cero-uno:** ceros y unos generados aleatoriamente.

N	Mergesort	Heapsort	QS Random	Med-of-3 QS	Introsort	Dual pivot QS	QS 3-way
4096	0.219 s	0.117 s	13.739 s	1.337 s	0.302 s	0.094 s	0.036 s
8192	0.432 s	0.276 s	55.912 s	5.234 s	0.419 s	0.203 s	0.072 s
16384	0.993 s	0.555 s	222.665 s	21.553 s	1.577 s	0.441 s	0.145 s
32768	2.095 s	1.198 s	882.291 s	85.905 s	3.148 s	0.984 s	0.289 s
65536	4.465 s	2.568 s	3536.847 s	346.391	6.639 s	2.117 s	0.590 s



Para este tipo de arreglos el *quicksort* randomizado fue el que tuvo una tendencia de crecimiento cuadrática junto al *quicksort median of 3*, sólo que este mucho más por debajo que el randomizado. El comportamiento del *quicksort* aleatorio se debe a que habrán muchos elementos dentro del arreglo con el mismo valor (0 o 1).

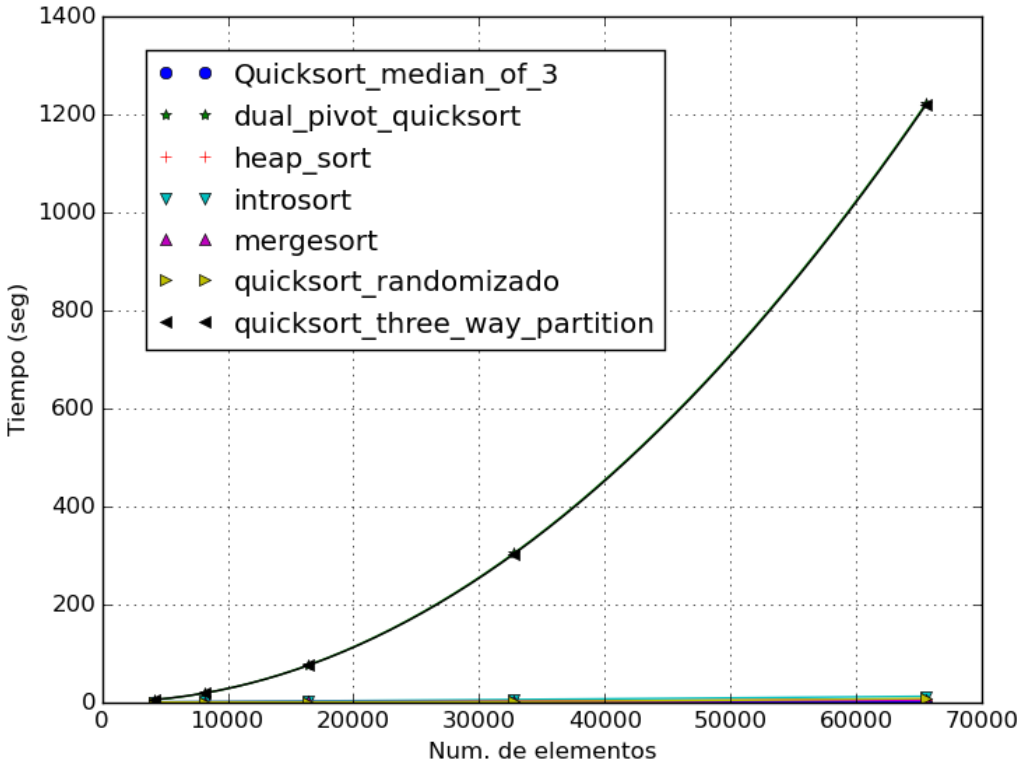
A continuación, se detallará el comportamiento de los algoritmos más eficientes:



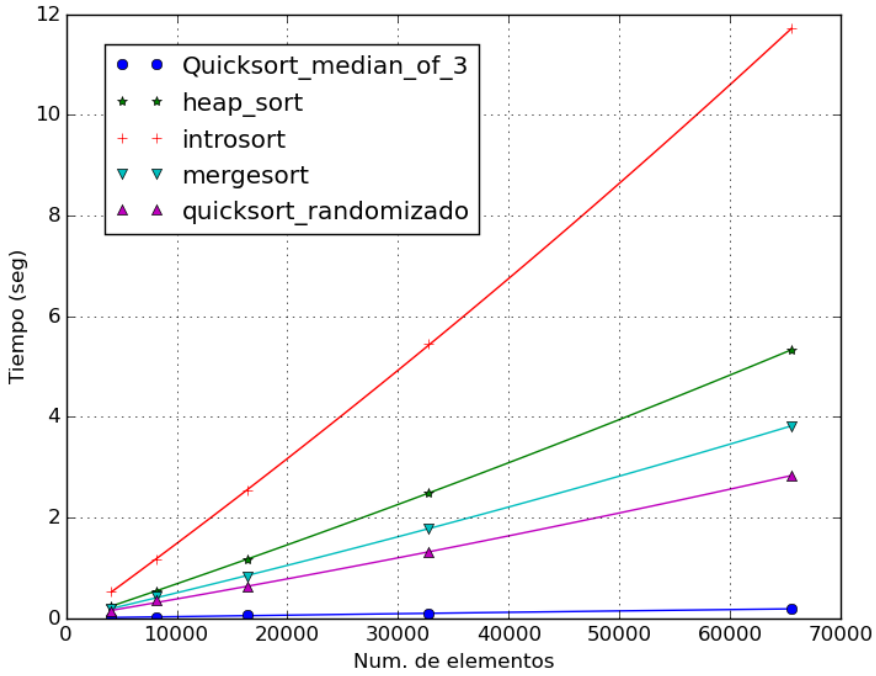
Se constata que efectivamente casi todos los algoritmos se comportan de manera O(nlogn), exceptuando *introsort* y *quicksort three way partition*, que poseen un orden de crecimiento cuadrático ($O(n^2)$, peor caso) y lineal (O(n), mejor caso) respectivamente.

4. **Ordenado:** si el tamaño del arreglo es N, entonces el arreglo contendrá la secuencia 1, 2,…, N-1, N.

N	Mergesort	Heapsort	QS Random	Med-of-3 QS	Introsort	Dual pivot QS	QS 3-way
4096	0.202 s	0.245 s	0.124 s	0.012 s	0.530 s	4.887 s	4.695 s
8192	0.426 s	0.543 s	0.291 s	0.024 s	1.174 s	20.712 s	18.871 s
16384	1.097 s	1.153 s	0.601 s	0.047 s	2.517 s	79.474 s	74.718 s
32768	1.984 s	2.561 s	1.321 s	0.093 s	5.414 s	318.348 s	312.084 s
65536	4.058 s	5.423 s	2.815 s	0.188 s	11.821 s	1225.031 s	1214.766 s



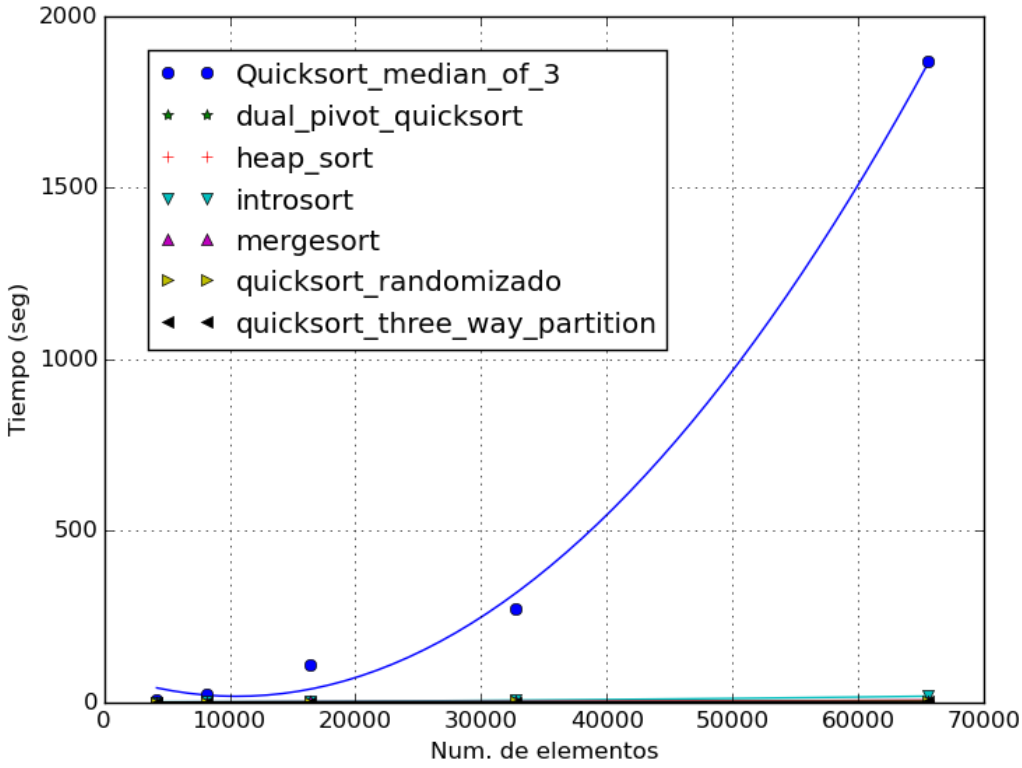
Para la prueba de enteros ordenados, los algoritmos con orden de crecimiento cuadrático fueron *dual pivot quicksort* (evidentemente por la elección de un pivote máximo con respecto al resto de elementos de los subarreglos) y *quicksort three way partition.* Todos los demás algoritmos fueron mucho más eficientes en esta corrida. Obsérvese en detalle el comportamiento de éstos:



Todos estos algoritmos tienen comportamiento O(nlogn) excepto *quicksort median of 3*, con orden de crecimiento lineal (O(n)). El peor comportamiento entre estos algoritmos está en introsort que sin embargo sí se comporta con O(nlogn).

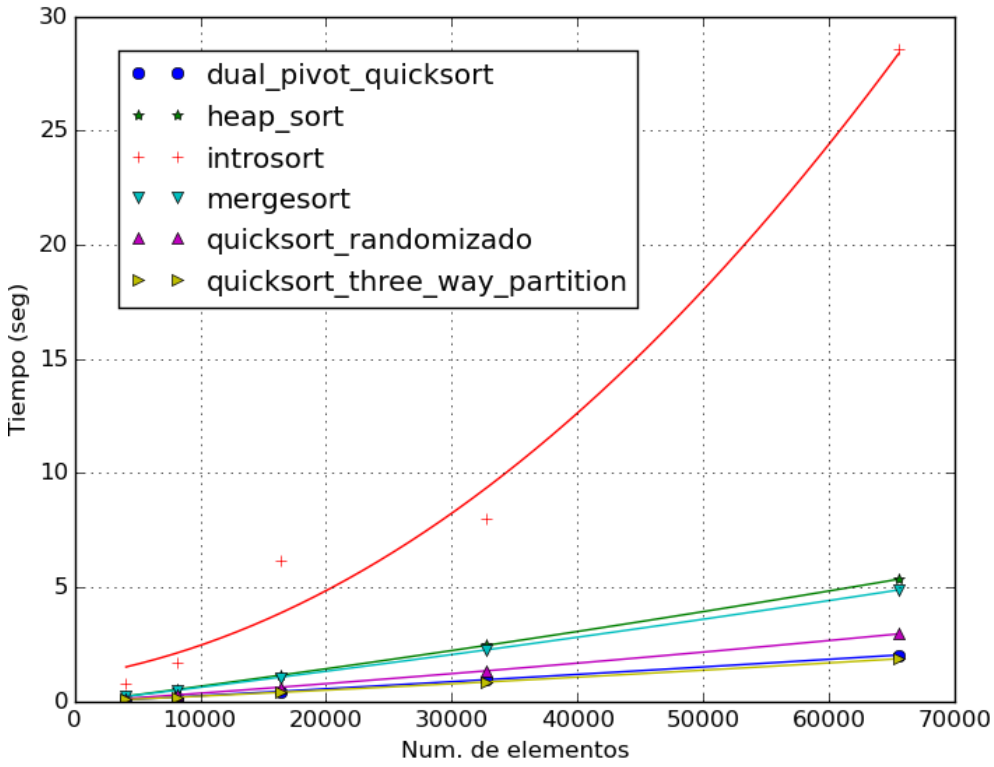
5. **Reales aleatorios:** números reales comprendidos en el intervalo [0,1)

N	Mergesort	Heapsort	QS Random	Med-of-3 QS	Introsort	Dual pivot QS	QS 3-way
4096	0.246 s	0.237 s	0.136 s	4.223 s	0.544 s	0.094 s	0.085 s
8192	0.476 s	0.526 s	0.291 s	17.599 s	1.195 s	0.189 s	0.185 s
16384	1.032 s	1.130 s	0.637 s	80.315 s	3.712 s	0.439 s	0.390 s
32768	2.231 s	2.650 s	1.402 s	299.249 s	8.531 s	0.910 s	0.905 s
65536	4.783 s	5.304 s	2.876 s	1156.739 s	17.260 s	2.037 s	1.806 s



En este caso, *quicksort median of 3* resultó ser (aparentemente) el único algoritmo con comportamiento cuadrático. Esto se puede deber a que dentro de la implementación de *quicksort median of 3* se hace llamada a un *quicksort_loop* donde adentro, calcula la mediana de 3 números reales aleatorios entre 0 y 1. Estas últimas comparaciones pueden llevar a un caso muy ineficiente, por el número de cifras significativas que pueda tener los valores del arreglo a ordenar.

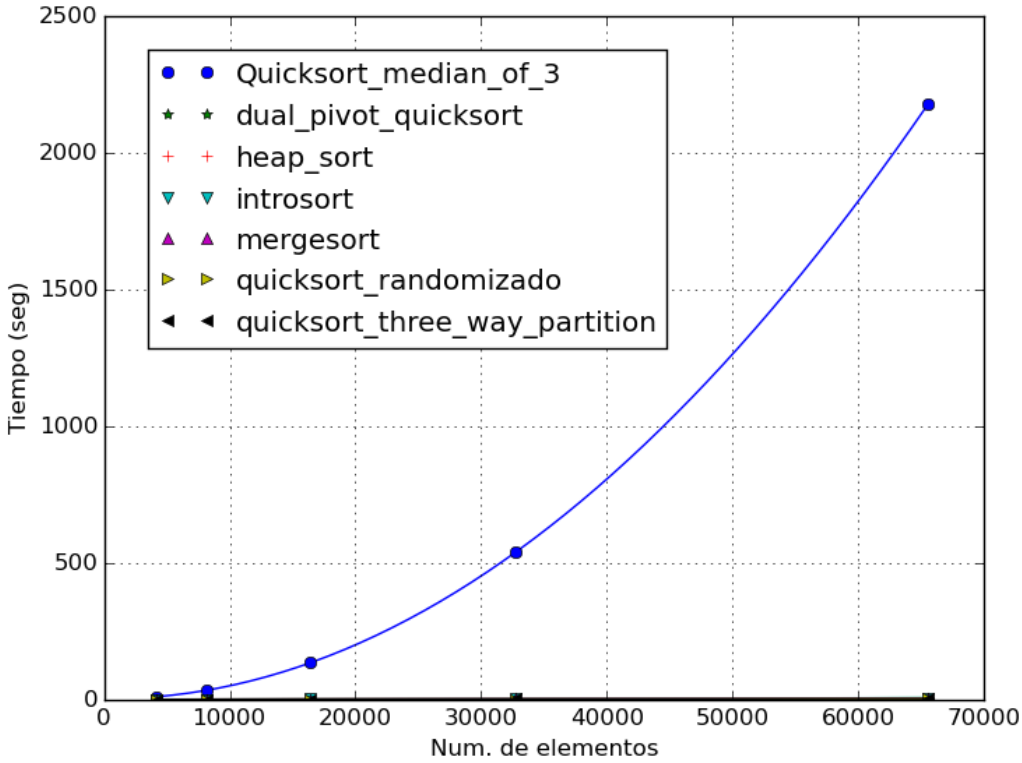
Detállese el comportamiento de los demás algoritmos:



Los demás algoritmos tuvieron un orden crecimiento lineal, excepto introsort que fue cuadrático. Para este tipo de arreglos, el *quicksort with three way partition* y el *dual pivot quicksort* resultaron ser los más eficientes, casi a la par del *quicksort* randomizado.

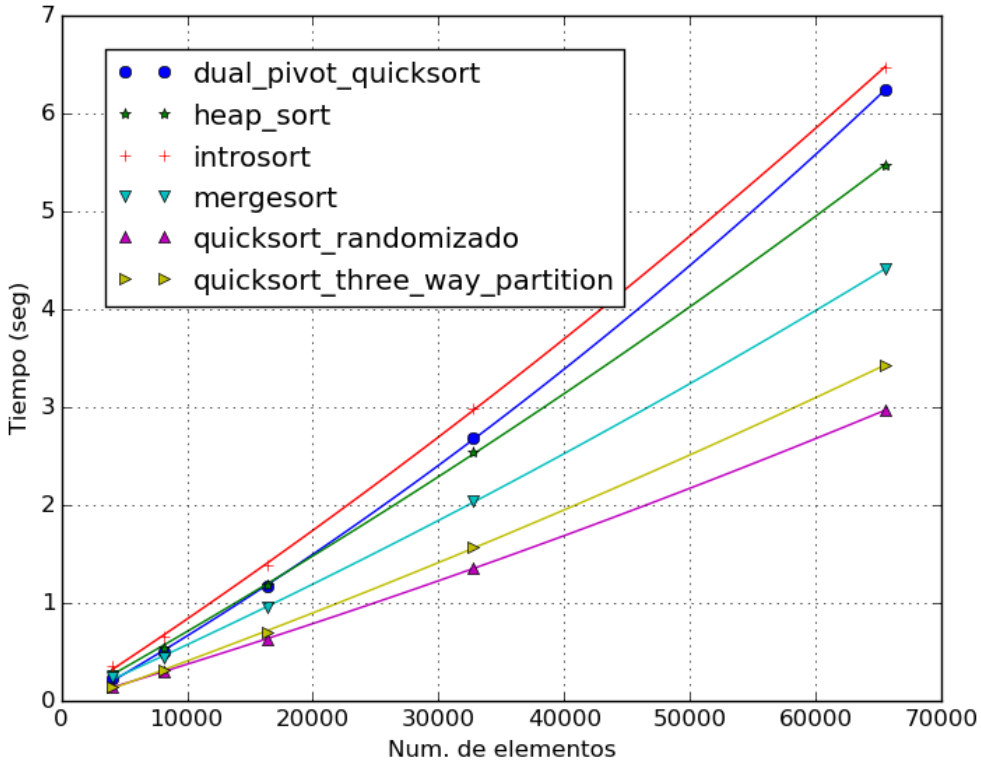
6. **Mitad:** dado un arreglo de tamaño N, el arreglo contiene como elementos la secuencia de la forma 1, 2,…, N/2, N/2,…, 2, 1

N	Mergesort	Heapsort	QS Random	Med-of-3 QS	Introsort	Dual pivot QS	QS 3-way
4096	0.218 s	0.236 s	0.130 s	7.888 s	0.281 s	0.210 s	0.133 s
8192	0.484 s	0.519 s	0.265 s	32.146 s	0.616 s	0.486 s	0.298 s
16384	1.293 s	1.207 s	0.629 s	131.917 s	1.443 s	1.135 s	0.713 s
32768	2.209 s	2.636 s	1.465 s	575.659 s	3.104 s	2.580 s	1.602 s
65536	4.401 s	6.190 s	2.852 s	2123.322 s	6.190 s	6.042 s	3.426 s



Para este caso, todos los algoritmos se comportaron de manera lineal, exceptuando *quicksort median of three* que se comporta de manera cuadrática.

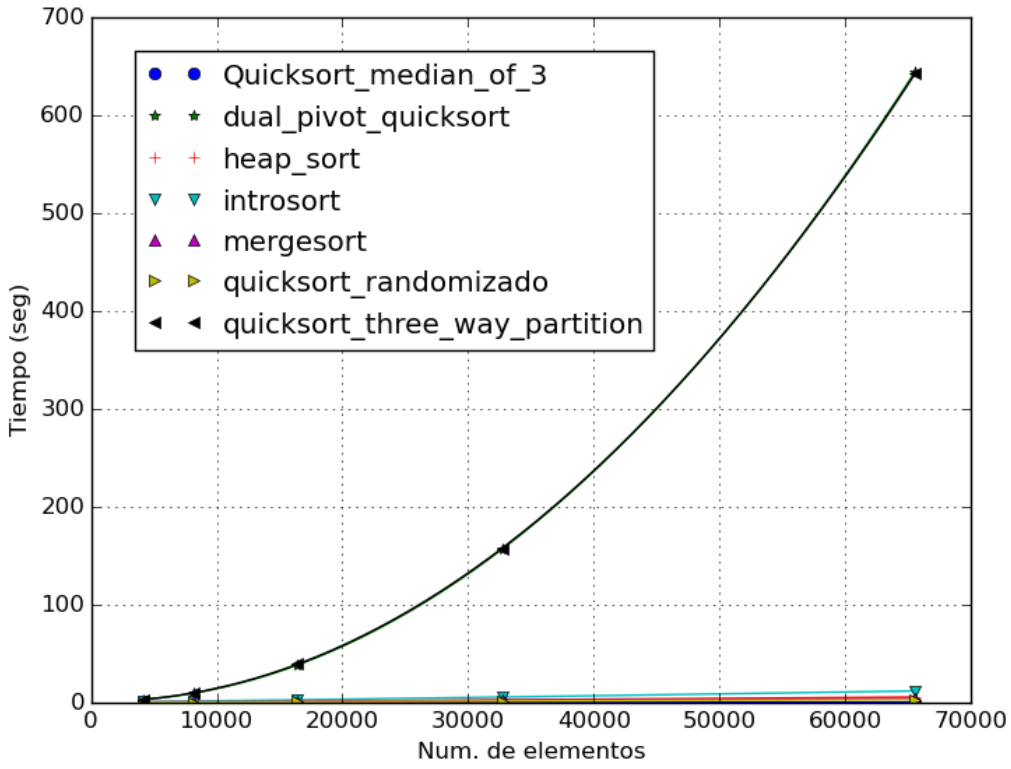
A continuación se verá un detalle del comportamiento de los otros algoritmos:



En este caso, casi todos los algoritmos tuvieron un comportamiento asintótico O(nlogn), exceptuando por el *dual pivot quicksort*, que tiene comportamiento cuadrático. Esto puede deberse al hecho que en los subarreglos puede elegirse como pivote tanto el mínimo como el máximo número de éste.

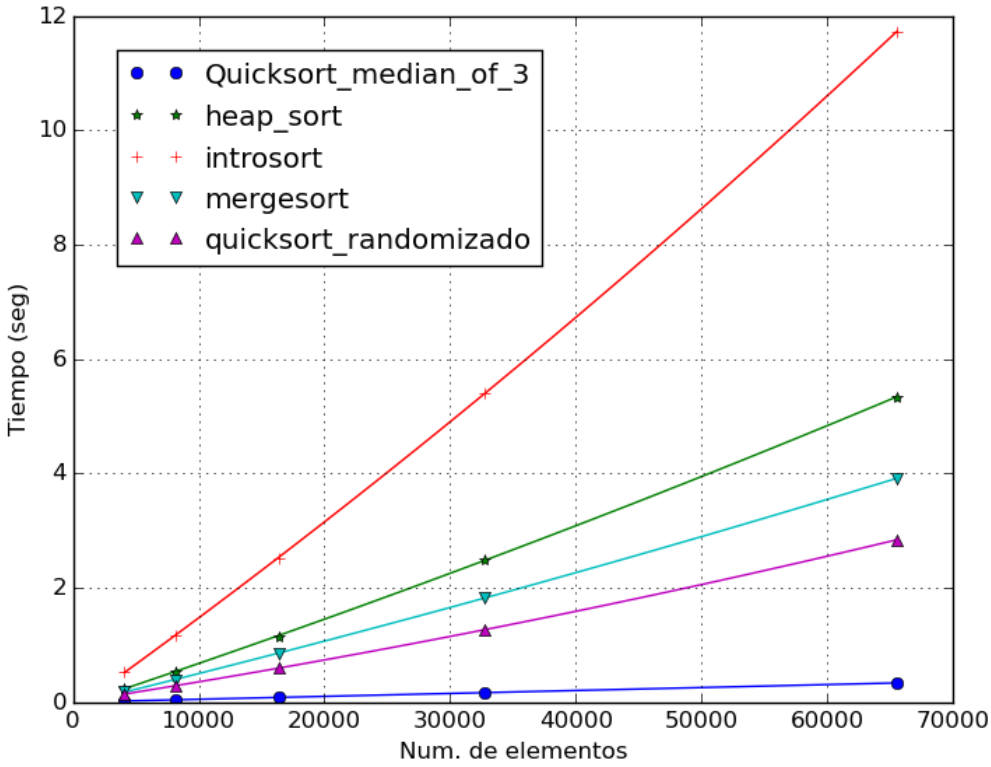
7. **Casi ordenado:** dado un conjunto ordenado de N elementos de tipo entero, se escogen al azar n/4 pares de elementos que se encuentran separados 4 lugares, entonces se intercambian los pares.

N	Mergesort	Heapsort	QS Random	Med-of-3 QS	Introsort	Dual pivot QS	QS 3-way
4096	0.202 s	0.276 s	0.140 s	0.022 s	0.557 s	2.420	2.620 s
8192	0.425 s	0.547 s	0.291 s	0.043 s	1.199 s	10.000 s	9.806 s
16384	0.906 s	1.229 s	0.677 s	0.096 s	2.667 s	39.365 s	43.093 s
32768	1.954 s	2.637 s	1.380 s	0.181 s	5.718 s	161.314s	169.871 s
65536	4.168 s	5.939 s	3.290 s	0.381 s	13.141 s	650.018 s	635.789 s



Para este caso, los algoritmos de peor comportamiento fueron *quicksort three way partition* y *dual pivot quicksort* (véase el caso 4).

Detállese el comportamiento del resto de los algoritmos:



Esta vez *introsort* vuelve a ser el algoritmo menos eficiente de entre estos, pero manteniendo su comportamiento O(nlogn). *Quicksort median of 3* otra vez vuelve a correr en tiempo lineal (O(n)) y en consecuencia es el algoritmo más eficiente para esta prueba.

ANÁLISIS GLOBAL

En general, los algoritmos de ordenamiento sí respondieron de distintas maneras a las diferentes pruebas a los que fueron sometidos, es decir, que dependiendo de los valores dentro del arreglo a ordenar, pueden poseer un orden de crecimiento muy distinto al que pueden presentar con ciertos otros arreglos.

Nótese que *mergesort* y *heapsort* efectivamente tuvieron un orden de crecimiento O(nlogn) en todos los tipos de arreglo, manteniendo un récord de tiempos prácticamente constante para cada caso y número de elementos, ignorando la manera en que estaba dada el arreglo a ordenar. Aunque nunca llegaron a ser los algoritmos más eficientes para alguna prueba, en general se pueden decir que fueron los algoritmos más eficientes para todos ya que en ningún tipo de arreglo corrieron a tiempo cuadrático y estaban casi siempre a la par.

Igualmente, *introsort* mantuvo un comportamiento O(nlogn) para casi todos los algoritmos (exceptuando el arreglo de ceros y unos, y el de reales aleatorios entre 0 y 1). Esto confirma el hecho de que este algoritmo corre la mayoría de las veces en tiempo O(nlogn), debido a la llamada de heapsort dentro de este algoritmo.

Ahora, en el caso del *dual pivot*, la corrida se apega al comportamiento que debería tener de acuerdo a los arreglos a ordenar (O(nlogn)), sin embargo, para ciertos casos, como el primero y el cuarto, tuvo un orden de crecimiento cuadrático mucho peor que el de los demás algoritmos. En cada prueba se detalló el por qué.

Fue el *quicksort median of 3* quien tuvo una corrida en tiempo cuadrático en cuatro de las siete pruebas, haciéndolo en promedio, el algoritmo más ineficiente de todos. Sin embargo, para las pruebas en las que no tuvo complejidad de tiempo cuadrática, poseyó una lineal O(n) haciéndolo increíblemente más eficiente que todos los demás.

El *quicksort three way partition* fue ineficiente en casi la mitad de las pruebas (tres) al igual que *dual pivot quicksort* por lo que fueron los segundos más ineficientes durante las pruebas, aunque cabe destacar que en los arreglos que no tuvieron este comportamiento, igualmente funcionaron de la manera esperada (en tiempos O(nlogn)).

En cuanto al *quicksort* aleatorio, casi siempre fue eficiente exceptuando en el caso de arreglos llenos de ceros y unos, debido al hecho de que muchos números podían ser iguales. Se puede

concluir que, después *mergesort* y *heapsort* este fue el segundo algoritmo más eficiente de todos por número de pruebas en que no fue el peor caso.