Universidad Simón Bolívar

Laboratorio de Algoritmos y Estructuras 2

# Desarrollo y prueba de 7 algoritmos de ordenamiento para el Laboratorio de Algoritmos y Estructuras 2


Realizado por:

- César Colina 13-10299
- Ian Goldberg 14-10406


Caracas, Febrero 2017

# ÍNDICE

# INTRODUCCIÓN

Un algoritmo de ordenamiento es un algoritmo que reordena o permuta elementos de una lista en una secuencia dada por una relación de orden, teniendo como salida del algoritmo una lista “ordenada”. El objetivo del proyecto es desarrollar “desde cero” 7 algoritmos de ordenamientos (ya existentes) totalmente funcionales y probar los mismos con múltiples pruebas. Los nombres de los algoritmos a desarrollar son: Mergesort, Heapsort, Quicksort Aleatorio, Median of 3 quicksort,Introsort, Quicksort with 3 way partitioning y Dual pivot Quicksort. Las pruebas a realizar sobre los algoritmos son 5, las cuales consisten en otorgarle como entrada al algoritmo listas de tamaño 4096,8192,16384,32768,65536 respectivamente, además cada prueba debe ejecutarse 3 veces y se reportará como tiempo de ejecución del algoritmo, el tiempo promedio de esas tres pruebas.

Finalmente los 7 algoritmos de ordenamiento se deben ejecutar en cada uno de los 7 conjuntos de datos, los cuales son: Enteros aleatorios, Orden inverso, Cero-Uno,Ordenado,Reales aletorios,Mitad,Casi ordenado. Para la realización de todas estas pruebas hicimos uso de las computadoras del Laboratorio de Sala “A” de la Universidad Simon Bolivar las cuales poseían un CPU Intel core i7-3770 3.4Ghz en una arquitectura de 64 bits con 8Gb de ram en la distribución de GNU Linux Ubuntu. Con dicho informe el lector podrá tener un panorama general del tiempo de ejecución de los algoritmos con diferentes conjuntos de datos y listas con gran cantidad de elementos y una explicación breve detallada de porque los algoritmos pueden ser eficientes o no en cada prueba en particular. Finalmente el informe se divide en portada, índice, introducción, análisis de los resultados, conclusiones.

## ANÁLISIS DE LOS RESULTADOS

A continuación se publicarán una serie de tablas donde se ven reflejados el tiempo (en segundos) de ejecución de los 7 algoritmos en las 5 pruebas en un determinado conjunto de datos.

Nota: Si alguna tabla no posee su grafica correspondiente es que lastimosamente no nos dio tiempo de terminar la ejecución del algoritmo la cual suelta la grafica al finalizar todo el proceso, sin embargo esto no es de impedimento para hacer el análisis de las pruebas.

**Conjunto de datos:**

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

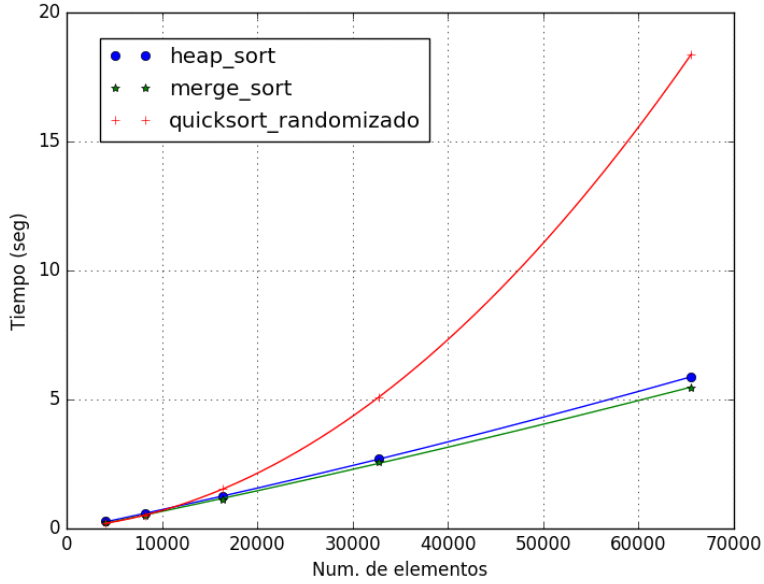
| N | MERGESORT | HEAPSORT | QS RANDOM | MED OF 3 QS | INTROSORT | DUAL PIVOT QS | QS 3 WAY |
|---|---|---|---|---|---|---|---|
| **4096** | 0,228 | 0,255 | 0,192 | 18,147 | 18,142 | 9,04 | 13,77 |
| **8192** | 0,539 | 0,558 | 0,505 | 76,632 | 73,617 | 36,388 | 32,73 |
| **16384** | 1,072 | 1,211 | 1,481 | 292,099 | 291,129 | 145,122 | 90,642 |
| **32778** | 2,46 | 2,622 | 4,84 | 1170,027 | 1165,892 | 587,527 | 615,484 |
| **65536** | | 5,649 | | | | 2311,885 | 892,992 |

Del conjunto de datos de enteros aleatorios, los algoritmos que presentaron un orden de crecimiento lineal o cuasi lineal fueron mergesort, heapsort y quicksort randomizado y los que presentaron un orden de crecimiento cuadrático fueron median of 3 quicksort, introsort, dual pivot quicksort y quicksort with 3 partitioning.

Este conjunto de datos quizás sea el más común de encontrar, una serie de datos que no expresan el peor caso de ningún algoritmo, si no que demuestra sencillamente el tiempo de corrida esperado de cada algoritmo. Nos damos cuenta que la función Partition de quicksort randomizado es muchísimo más eficiente que la función Partition de median of 3 quicksort, quicksort with 3 partitioning y dual pivot quicksort debido a que al elegir un pivote randomizado el algoritmo es mucho menos propenso a

un peor caso y hacer ¨splits¨ malos, es decir es capaz de tener particiones mucho mas balanceadas y más eficientes en tiempo que el particionamiento con una mediana, el doble pivote o dividir el arreglo en 3 particiones. Introsort de la misma manera que median of 3 quicksort su método de particionamiento no fue más eficiente que el particionamiento randomizado. Por otra parte mergesort con su algoritmo de tipo dividir y conquistar, divide la secuencia en dos partes iguales y soluciona cada secuencia de forma recursiva y luego con la función Merge es capaz de unir las soluciones en una secuencia ordenada de forma muy eficiente y asintóticamente optima. Finalmente heapsort con su ordenamiento ¨in place ¨ y el uso de de la estructura de dato de heap binario hace que el algoritmo sea asintóticamente optimo.

**Gráfica sólo con los algoritmos de orden lineal o cuasi lineal.**



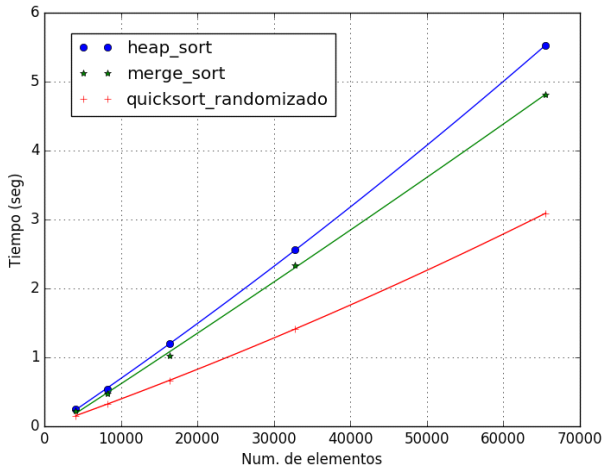
**Conjunto de datos:**

**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** | 0,212 | 0,242 | 0,141 | 18,242 | 18,271 | 18028 | 62,229 |
| **8192** | 0,459 | 0,529 | 0,316 | 73,219 | 73,477 | 72,598 | 249,443 |
| **16384** | 0,983 | 1,147 | 0,67 | 291,911 | 292,773 | 290,909 | 992,092 |
| **32778** | 2,235 | 2,492 | 1,53 | 1160,045 | 1176,109 | 1160,669 | 4012,18 |
| **65536** | | | | | | | |

Del conjunto de datos orden inverso podemos destacar principalmente el peor caso de quicksort with 3 partitioning el cual es muy poco eficiente en la mayoría de los casos en especial en arreglos que están ordenados o casi ordenados. Los algoritmos que presentaron un orden lineal o cuasi lineal fueron mergesort, heapsort y quicksort randomizado mientras que los que tuvieron un desempeño cuadrático fueron median of 3 quicksort, introsort, dual pivot quicksort y quicksort with 3 partitioning.Este conjunto de datos no suele afectar el desempeño de la mayoría de algoritmos, sino únicamente a quicksort with 3 partitioning.

**Gráfica sólo con los algoritmos de orden lineal o cuasi lineal**




**Conjunto de datos:**

**Cero-uno:** Ceros y unos generados aleatoriamente.

| N | MERGESORT | HEAPSORT | QS RANDOM | MED OF 3 QS | INTROSORT | DUAL PIVOT QS | QS 3 WAY |
|---|---|---|---|---|---|---|---|
| **4096** | 0,204 | 0,128 | 14,549 | 8,904 | 8,897 | 4,34 | 0,072 |
| **8192** | 0,445 | 0,276 | 58,436 | 35,561 | 35,451 | 18,242 | 0,135 |
| **16384** | 0,962 | 0,593 | 232,917 | 141,511 | 140,969 | 71,074 | 0,27 |
| **32778** | 2,203 | 1,285 | 933,716 | 564,041 | 562,185 | 284,094 | 0,578 |
| **65536** | 4,462 | 2,706 | 3784,822 | 2268,192 | 2252,309 | 1122,966 | 1,046 |

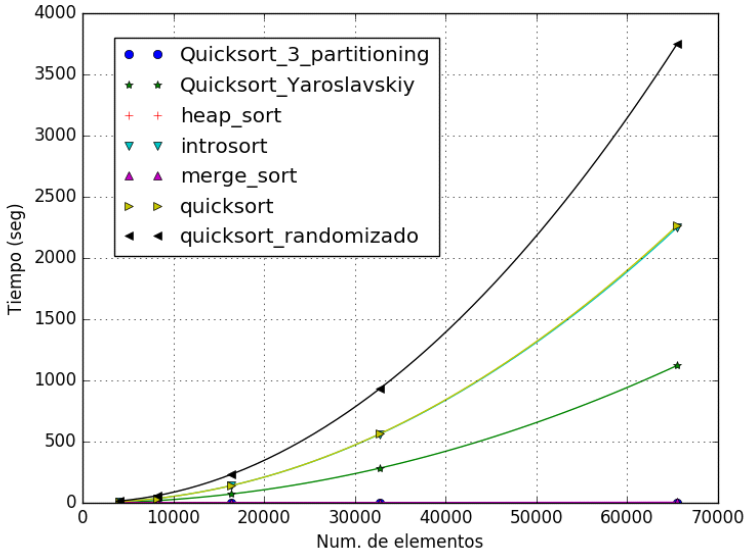
Analizamos los resultados para todos los algoritmos excluyendo a Merge Sort y Heap Sort ya que, como mencionamos anteriormente, el desempeño de éstos no se ve afectado por la manera en que llenamos los arreglos a ordenar.

Para este conjunto de datos es obvio afirmar, por la teoría de cada uno de los algoritmos implementados, que 3-way es el algoritmo de ordenamiento más eficiente. Ésto se debe a que esta versión de quicksort aprovecha la condición de algunos arreglos de tener una gran cantidad de elementos iguales (como el arreglo utilizado que sólo posee 0's y 1's), además no

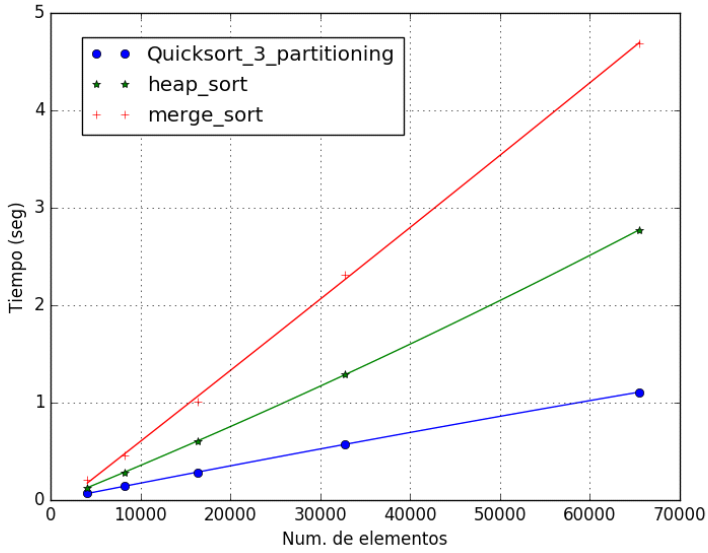
sólo se repite mucho cada uno de los elementos sino que además tenemos que sólo hay 2 elementos distintos, lo cual aumenta aun más su eficiencia.

Para el resto de las versiones de quicksort podemos notar que su desempeño no es tan bueno como el de los demás algoritmos y ésto se debe al pivote, ya que es muy probable que éstos agarren como pivote el uno, el cual sería el peor de los casos. La única ventaja de Dual Pivot, y que se refleja en los resultados, es el hecho de que como agarra 2 pivotes aumenta la probabilidad de que uno de éstos sea un cero, el cual permitiría a esta versión de quicksort desarrollarse con una mejor eficiencia. Para el caso de Median of 3 se podría pensar que como agarra 3 candidatos a pivote, los compara entre sí, y agarra el valor del medio, como las comparaciones se realizaron contemplando la igualdad existe aun una alta probabilidad de que agarre como pivote un elemento que tenga valor uno.En este conjunto de datos obtuvimos que Merge Sort, Heap Sort y 3-way poseen una eficiencia de nlog(n), mientras que los demás algoritmos son de tipo cuadrático.

**Gráfica con todos los algoritmos.**




**Gráfica sólo con los algoritmos de orden lineal o cuasi lineal**



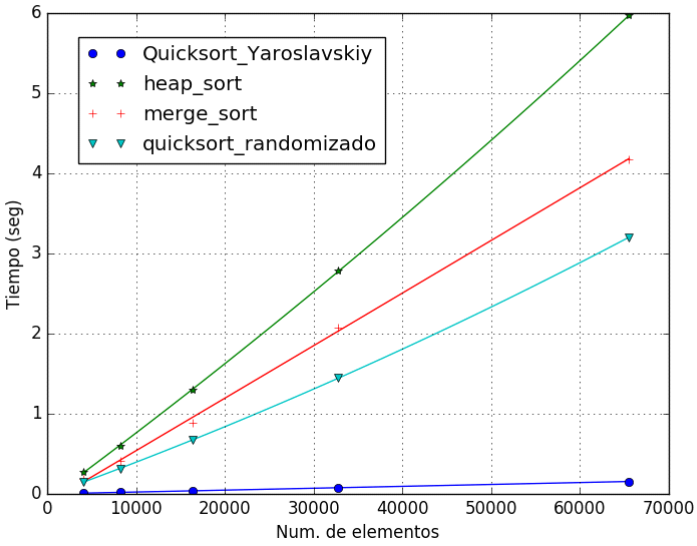

**Conjunto de datos:**

**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** | 0,196 | 0,282 | 0,166 | 19,284 | 19,189 | 0,01 | 46,809 |
| **8192** | 0,423 | 0,621 | 0,324 | 76,93 | 77,216 | 0,02 | 188,183 |
| **16384** | 0,911 | 1,339 | 0,722 | 307,804 | 307,503 | 0,04 | 751,112 |
| **32778** | 2,125 | 2,884 | 1,46 | 1231,305 | 1237,059 | 0,078 | 3006,668 |
| **65536** | | | | | | | |

Del conjunto de datos ordenado destacamos principalmente el mejor caso y destacada eficiencia de dual pivot quicksort, debido a que el particionamiento no tiene que hacer ningún swap entre elementos del arreglo, debido a que el valor de todos los elementos están comprendidos entre ambos pivotes, es decir ya se encuentra particionado y ordenado el arreglo. Los algoritmos que presentaron un orden lineal o cuasi lineal fueron mergesort, heapsort, quicksort randomizado y dual pivot quicksort mientras que los que tuvieron un desempeño cuadrático fueron median of 3 quicksort, introsort y quicksort with 3 partitioning.

**Gráfica sólo con los algoritmos de orden lineal o cuasi lineal.**



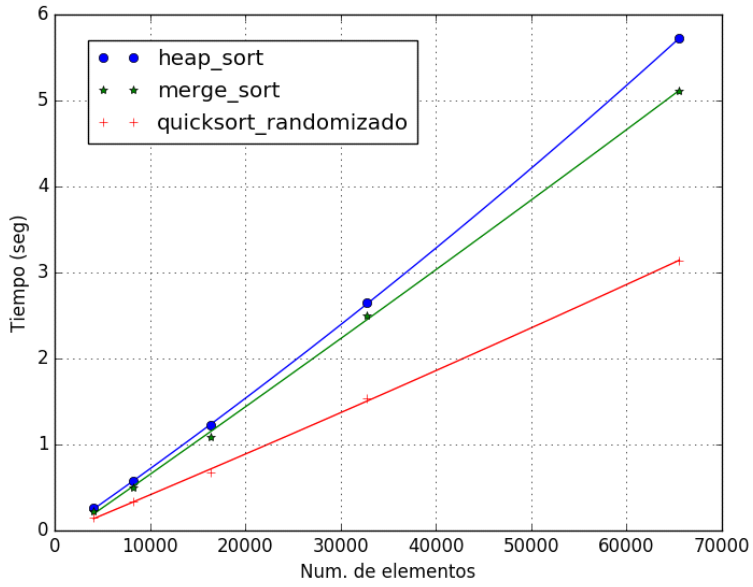

**Conjunto de datos:**

**Reales aleatorios:** Números reales comprendidos en el intervalo [0, 1) generados aleatoriamente.

| N | MERGESORT | HEAPSORT | QS RANDOM | MED OF 3 QS | INTROSORT | DUAL PIVOT QS | QS 3 WAY |
|---|---|---|---|---|---|---|---|
| **4096** | 0,244 | 0,266 | 0,154 | 19,513 | 19,419 | 9,695 | 51,847 |
| **8192** | 0,524 | 0,579 | 0,313 | 77,413 | 77,713 | 39,106 | 184,293 |
| **16384** | 1,126 | 1,252 | 0,752 | 307,856 | 308,348 | 154,01 | 805,967 |
| **32778** | 2,588 | 2,706 | 1,454 | 1240,326 | 1240,861 | 618,275 | 2822,453 |
| **65536** | | | | | | | |

Del conjunto de datos Reales aleatorios podemos ver como siempre un desempeño lineal o cuasi lineal de mergesort, heapsort, quicksort randomizado,sin embargo dual pivot quicksort a pesar de no correr en tiempo lineal con este conjunto de datos, se comportó mucho mas eficiente que los algoritmos median of 3 quicksort, introsort y quicksort with 3 partitioning que corrieron en tiempo cuadrático debido a que el particionamiento de dual pivot es favorecido a que los pivotes serian 0 y un numero cerca de 1 y todos los demás elementos tendrán valores que están en dicho rango.

**Gráfica sólo con los algoritmos de orden lineal o cuasi lineal.**




**Conjunto de datos:**

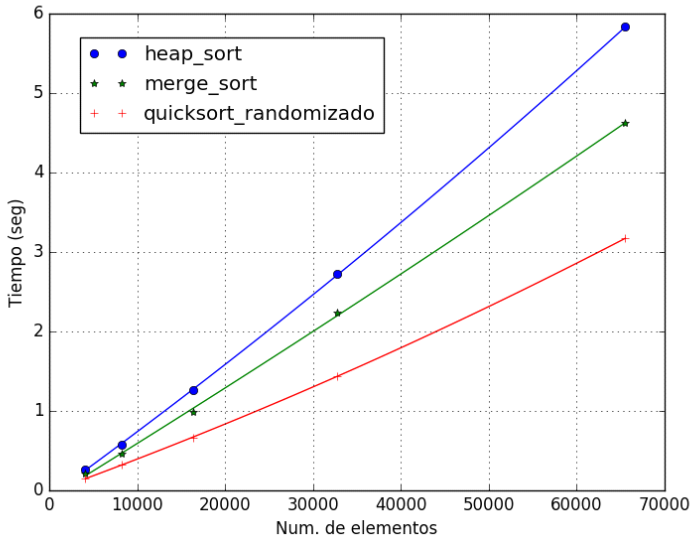
**Mitad:** Dado un arreglo de tamaño N, el arreglo contiene como elementos la secuencia de la forma 1, 2, . . . , N/2, N/2, . . . , 2, 1.

| N | MERGESORT | HEAPSORT | QS RANDOM | MED OF 3 QS | INTROSORT | DUAL PIVOT QS | QS 3 WAY |
|---|---|---|---|---|---|---|---|
| **4096** | 0,198 | 0,258 | 0,145 | 17,926 | 14,565 | 8,873 | 30,201 |
| **8192** | 0,427 | 0,564 | 0,309 | 71,967 | 53,859 | 35,73 | 121,68 |
| **16384** | 0,919 | 1,217 | 0,664 | 288,969 | 252,35 | 144,571 | 489,144 |
| **32778** | 2,093 | 2,632 | 1,396 | 1160,986 | 1162,461 | 575,248 | 1956,662 |
| **65536** | | | | | | | |

Para este conjunto de datos podemos observar que no se registró algún resultado que resalte, con ésto nos referimos a que no hay ningún algoritmo que se salga de su desempeño habitual para un arreglo de características comunes. Con ésto queremos decir que este tipo específico de arreglo no es el mejor caso para ninguno de los algoritmos a estudiar.

En la prueba 6 obtuvimos que Merge Sort, Heap Sort y Quicksort Randomizado poseen una eficiencia de nlog(n), mientras que los demás algoritmos son de tipo cuadrático.

**Gráfica sólo con los algoritmos de orden lineal o cuasi lineal.**




**Conjunto de datos:**

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 | 0,198 | 0,271 | 0,14 | 18,636 | 18,529 | 0,022 | 22,488 |
| 8192 | 0,429 | 0,591 | 0,304 | 74,438 | 74,193 | 0,043 | 90,939 |
| 16384 | 0,923 | 1,281 | 0,688 | 300,363 | 298,125 | 0,187 | 365,344 |
| 32778 | 2,142 | 2,744 | 1,457 | 1196,817 | 1119,861 | 0,174 | 1449,744 |
| 65536 | | | | | | | |

Para este conjunto de datos, como para todas, podemos afirmar que Merge Sort y Heap Sort se comportan de forma idónea ya que éstos no dependen del tipo de arreglo sino de la cantidad

de elementos, es decir, no importa con qué método llenemos los arreglos ellos siempre realizarán la misma cantidad de comparaciones.

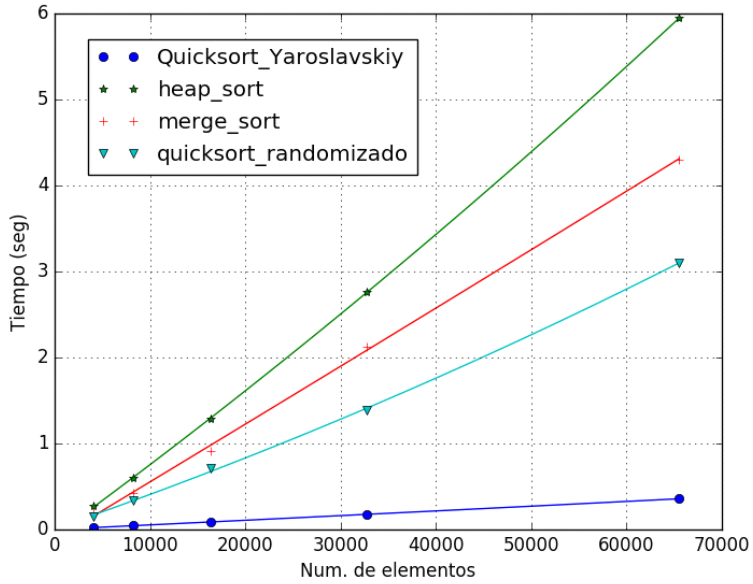
Sabiendo lo antes mencionado podemos proceder a analizar los resultados de los demás algoritmos:

En el caso de Dual Pivot observamos que es el que mejor comportamiento tiene, con respecto a los demás algoritmos de ordenamiento. Esto se debe a que el el segundo pivote ayuda a que el hecho de estar semiordenado no afecte en tanta medida al algoritmo como lo hace con el Quicksort traidicional. Con ésto nos referimos a que, como se implementa un segundo pivote, cada uno de los pivotes no realiza tantas comparaciones como lo haría si tuviese uno solo. Además también se reduce la cantidad de comparaciones gracias a que cada recurrencia de Dual Pivot en sí mismo trabaja con secciones del arreglo de menor tamaño que como lo haría con un solo pivote, a los cuales se le aplica la misma ventaja del doble pivote.

Para el caso de introsort, Median of 3 y 3-way notamos que no hay gran diferencia entre el desempeño de estos tres. Más adelante explicaremos cuál es el caso óptimo de cada uno de éstos y por qué con este tipo de arreglos no se nota una gran eficiencia.

En este conjunto de datos obtuvimos que Merge Sort, Heap Sort y Dual Pivot poseen una eficiencia de nlog(n), mientras que los demás algoritmos son de tipo cuadrático.

**Gráfica sólo con los algoritmos de orden lineal o cuasi lineal.**



## CONCLUSIONES

Con la ejecución de todos los algoritmos en cada una de las pruebas y conjuntos de datos utilizados podemos concluir que Mergesort, un algoritmo de tipo dividir y conquistar , el cual divide la secuencia de entrada en dos secuencias de igual tamaño y resuelve recursivamente dichas secuencias para luego unirlas de forma muy eficiente y resolviendo el problema de ordenamiento en un tiempo nlog(n) sin importar el arreglo de entrada es un algoritmo asintóticamente optimo y muy eficiente en la práctica. Heapsort a diferencia del algoritmo Mergesort ordena los elementos “in place” y utiliza una estructura de datos llamada heap binario el cual hace que este algoritmo sea asintóticamente optimo corriendo en un tiempo nlog(n) para cualquier tamaño de arreglo como entrada, muy útil también en la práctica.

El algoritmo quicksort randomizado también es un algoritmo muy útil en la práctica, en todas las pruebas y conjunto de datos fue asintóticamente optimo excepto cuando el arreglo sólo tiene 0 y 1. Este algoritmo debido a su pivote randomizado evita un peor caso con muchas probabilidades, sin embargo no está exento al peor caso donde corre en tiempo cuadratico. El algoritmo Median-of-3 quicksort el cual implementa una función de Partition basándose en la mediana de 3 elementos del arreglo, presentó un desempeño cuadrático haciendo que el

algoritmo no sea muy útil al momento de la práctica, pero de todas formas logró obtener mejores resultados en muchos casos que otros algoritmos.

El algoritmo introsort a pesar de que en teoría este algoritmo, el cual es una versión de quicksort que limita la recursión y llama a heapsort cuando pasa cierto límite de recursiones, corre en tiempo nlog(n) inclusive en su peor caso, realmente al momento de las pruebas su desempeño fue cuadrático y concluimos que no es útil en la práctica.

La implementación de Dual Pivot Quicksort mostró mejor desempeño que las demás variantes de quicksort, a excepción de Quicksort Randomizado, ésto se debe al uso de un segundo pivote, con el cual logramos dividir el arreglo en tres secciones, a diferencia del quicksort tradicional que sólo lo divide en dos. Gracias a ésto el algoritmo realiza menos cantidad de comparaciones, lo que produce la eficiencia obtenida. A pesar de ser una mejor opción, en la mayoría de los casos, que 3-way, Median of 3 e Introsort obtuvimos que su desempeño es de tipo cuadrático.

Por último el algoritmo de 3-way programado obtuvo muy malos resultados en la mayoría de los casos, exceptuándose ésto al ordenar el arreglo que contiene sólo 0's y 1's. Ésto se debe a la forma en que dicho algoritmo trabaja, es decir, de manera excepcional con arreglos los cuales cuentan con una gran cantidad de elementos iguales. Obviando este tipo de arreglos obtuvimos que 3-way siempre fue el método de ordenamiento más ineficiente, con un desempeño de tipo cuadrático.