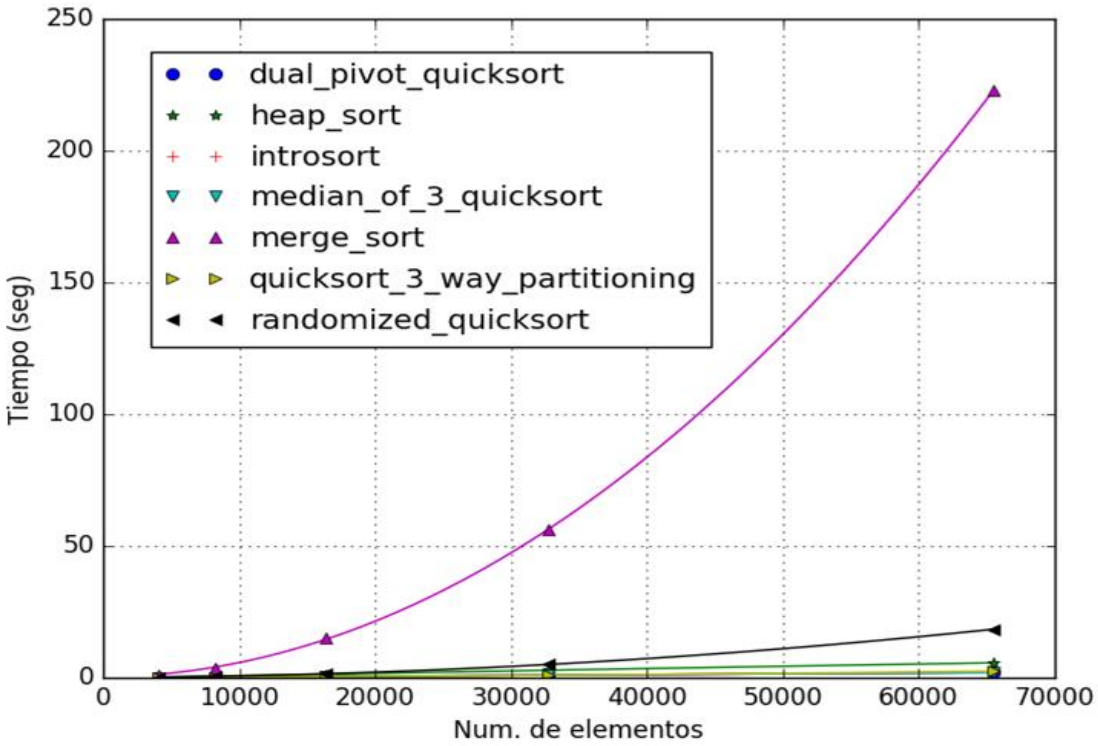
Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI-2692 - Laboratorio de Algoritmos y Estructuras II
Trimestre Enero-Marzo 2017
Integrantes: Angelica Acosta, 14-10005 y Yeferson Licet, 14-10572

Las pruebas fueron realizadas en una laptop Lenovo, modelo Yoga 510, Core i5, 8 GB RAM, con sistema operativo Ubuntu 16.04.

Conjunto de Datos 1:

Números enteros comprendidos en el intervalo [0; 500] generados aleatoriamente.

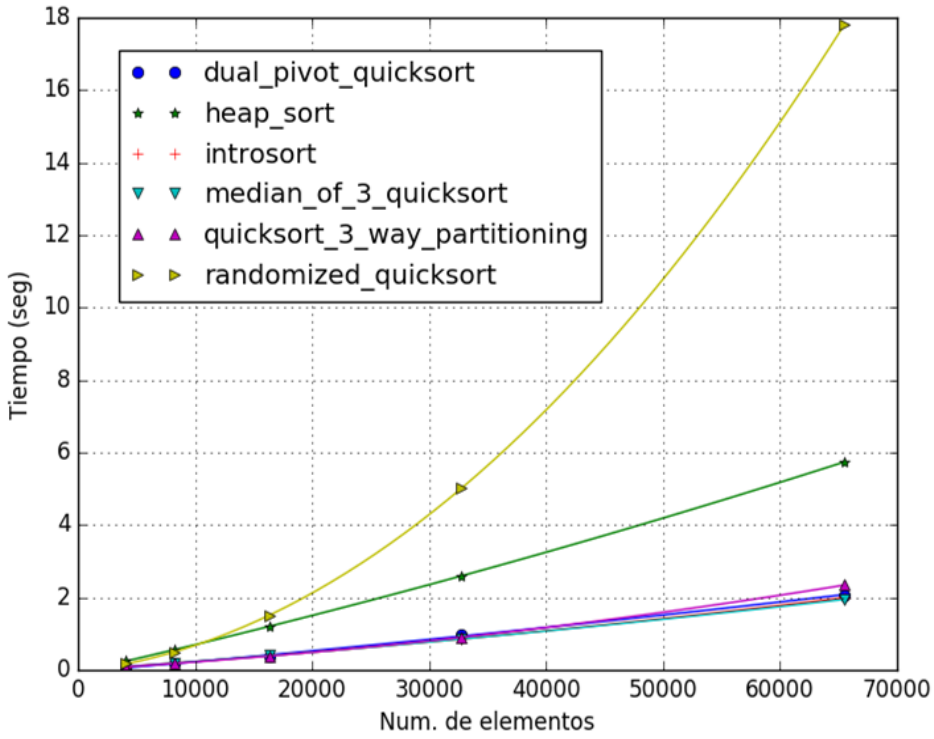
N	Mergesort	Heapsort	QS Random	Med-of-3 QS	Introsort	Dual pivot QS	QS 3-way
4096	1.095	0.254	0.197	0.088	0.088	0.096	0.088
8192	3.903	0.554	0.485	0.177	0.176	0.193	0.190
16384	15.083	1.207	1.558	0.390	0.389	0.420	0.414
32768	56.028	2.903	4.964	0.989	0.973	0.952	0.928
65536	223.169	5.620	18.400	1.892	1.899	1.984	2.497



Como podemos ver Mergesort presenta comportamiento semi-cuadratico mientras que los demás algoritmos presentan comportamientos lineales. Esto porque mergesort no está optimizado y realiza todo el proceso de dividir en arreglo en muchos subarreglos de tamaño 1, mientras que los demás algoritmos realizan solo los intercambios necesarios.

Zoom 1

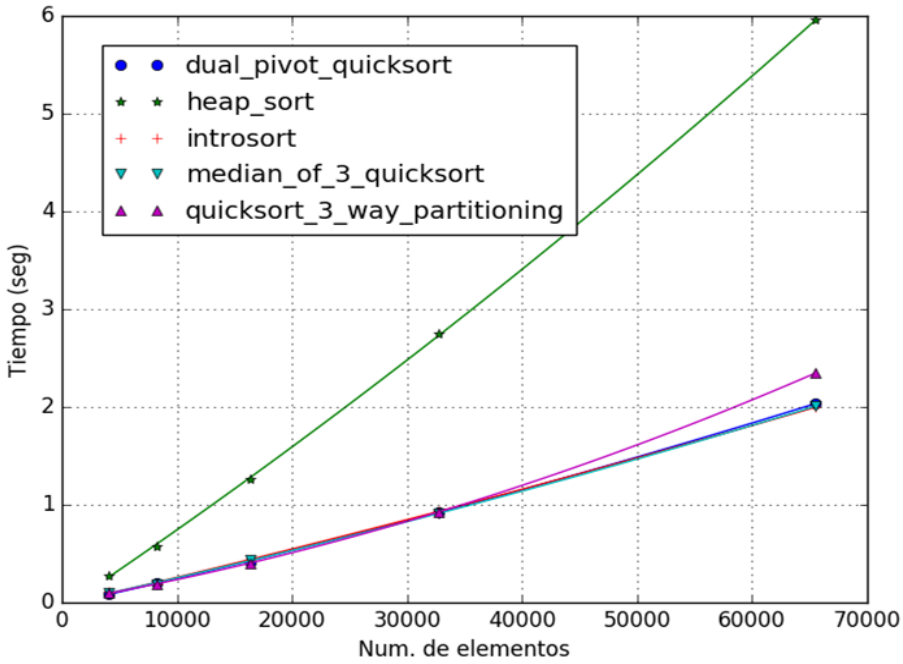
N	Heapsort	QS Random	Med-of-3 QS	Introsort	Dual pivot QS	QS 3-way
4096	0.253	0.182	0.090	0.087	0.083	0.087
8192	0.552	0.495	0.195	0.198	0.174	0.181
16384	1.196	1.483	0.401	0.401	0.374	0.386
32768	2.604	5.034	0.851	0.852	0.965	0.902
65536	5.743	17.822	1.951	1.994	2.077	2.345



Randomized quicksort presenta el peor comportamiento si lo comparamos con el resto de los algoritmos de tiempo lineal, esto porque quicksort no esta optimizado para la repetición de números. En este caso los arreglos son de tamaños mayores a 500, y como los números del arreglo son enteros entre 0 y 500 se puede asegurar que se repetirán números y así quicksort perderá tiempo intercambiando valores.

Zoom 2

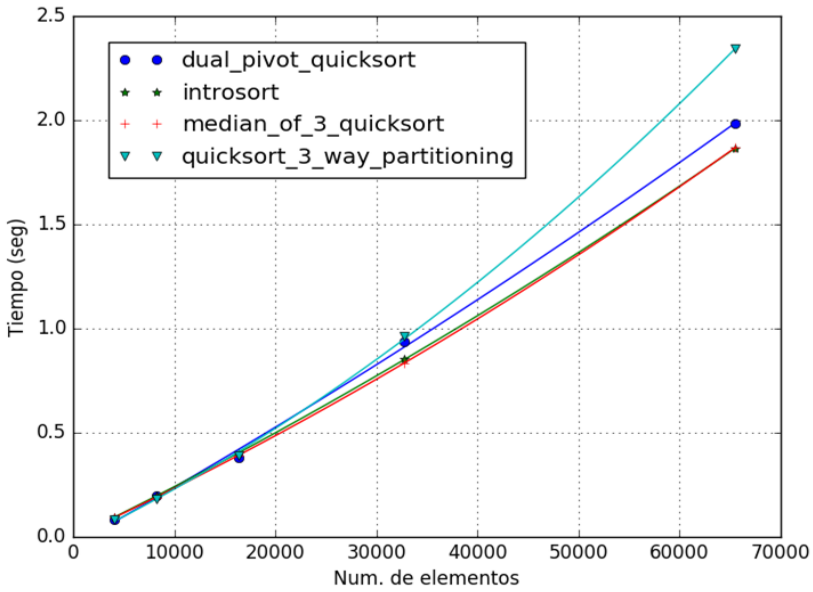
N	Heapsort	Med-of-3 QS	Introsort	Dual pivot QS	QS 3-way
4096	0.279	0.105	0.109	0.086	0.100
8192	0.580	0.183	0.189	0.198	0.189
16384	1.266	0.432	0.431	0.415	0.406
32768	2.744	0.913	0.942	0.929	0.928
65536	5.958	2.009	1.997	2.035	2.348



Aquí podemos ver que introsort, median_ of_3_quicksort y dual_pivot quicksort presentan comportamientos similares. Estos son algoritmos mucho más optimizados para la repetición de valores.

Zoom 3

N	Med-of-3 QS	Introsort	Dual pivot QS	QS 3-way
4096	0.094	0.094	0.083	0.084
8192	0.189	0.199	0.200	0.181
16384	0.392	0.402	0.379	0.391
32768	0.835	0.852	0.935	0.962
65536	1.870	1.867	1.984	2.344

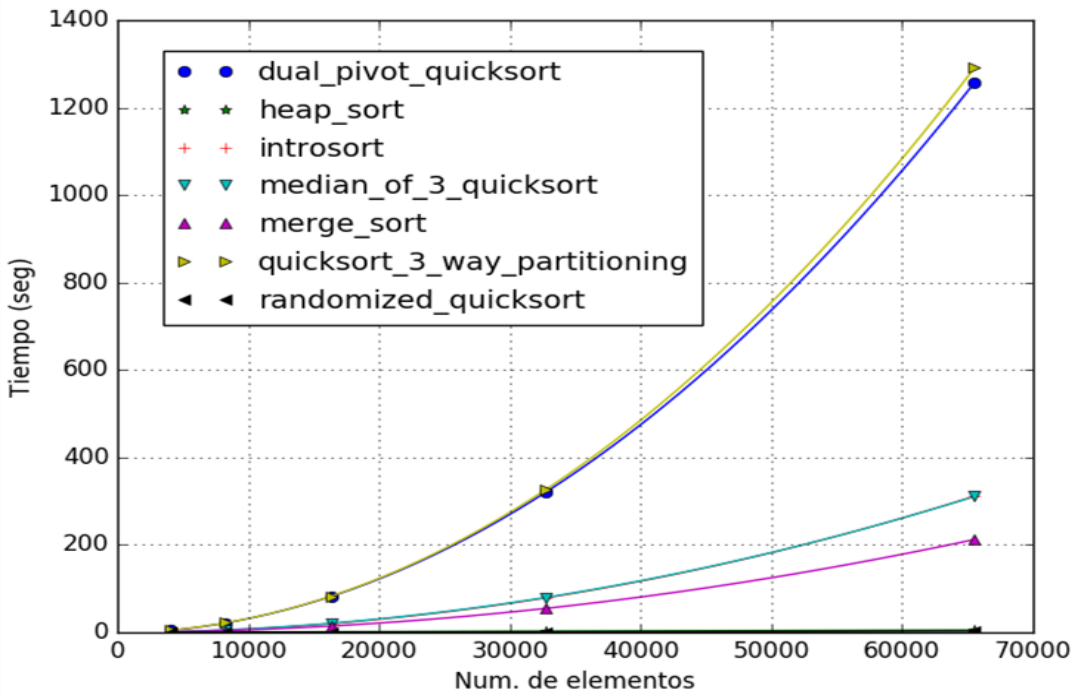


Aquí se aprecia como Med-of-3 QS , Introsort y Dual pivot QS tienen el mejor rendimiento sin embargo QS 3 way tiene un rendimiento ligeramente menor, sin dejar de ser una buena opción, indudablemente estos algoritmos son los mejores para ordenar números aleatorios.

Conjunto de Datos 2:

Si el tamaño del arreglo es N, entonces el arreglo contendrá la secuencia N; N - 1; . . . ; 2; 1.

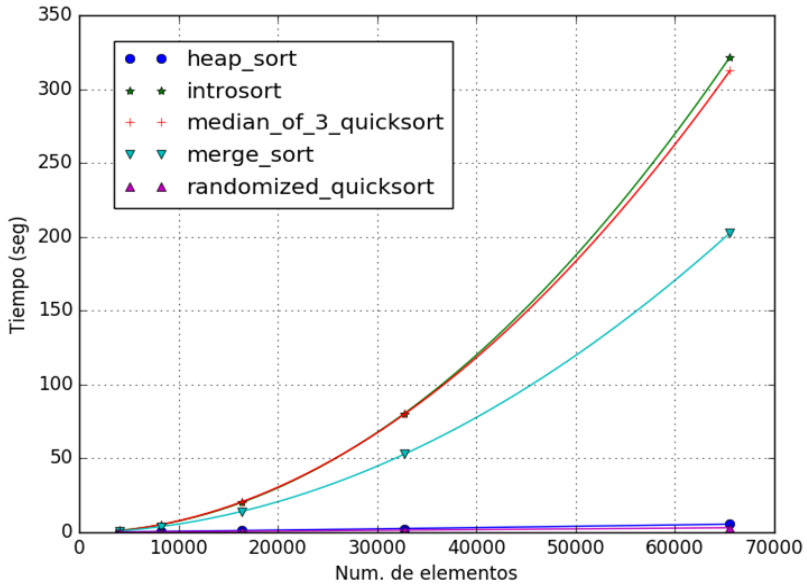
N	Mergesort	Heapsort	QS Random	Med-of-3 QS	Introsort	Dual pivot QS	QS 3-way
4096	1.093	0.248	0.139	1.295	1.322	5.202	5.329
8192	4.076	0.555	0.305	5.320	5.331	20.916	21.963
16384	14.531	1.153	0.673	21.758	20.187	81.916	82.497
32768	54.852	2.539	1.399	79.708	79.178	321.390	327.276
65536	212.370	5.287	3.197	310.834	311.345	1257.416	1291.321



Este representa el peor caso para dual_pivot_quicksort y para quicksort_3_way_partitioning. Como el arreglo esta ordenado descendientemente, independientemente del pivote que se escoja, el procedimiento partition realizara muchos intercambios.

Zoom 1

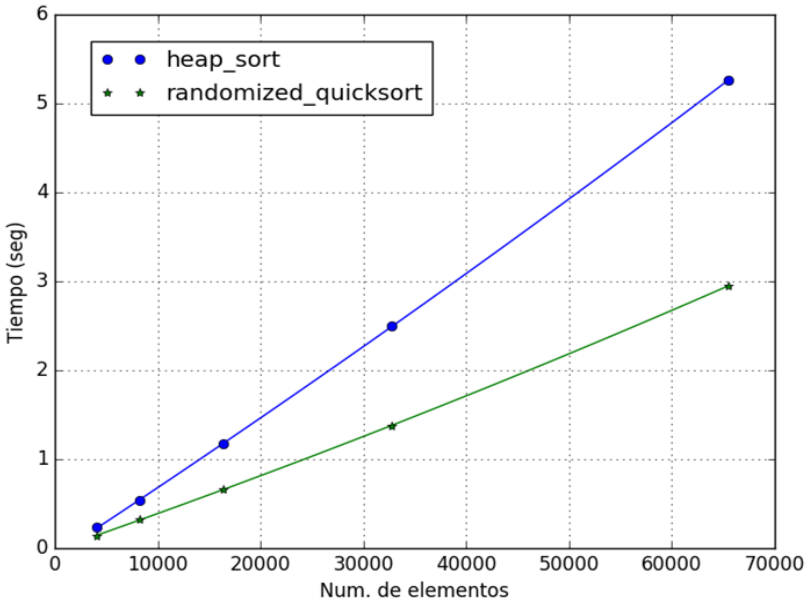
N	Mergesort	Heapsort	QS Random	Med-of-3 QS	Introsort
4096	1.045	0.244	0.151	1.296	1.296
8192	3.647	0.531	0.315	5.0543.647	5.087
16384	14.017	1.219	0.688	20.225	20.197
32768	53.078	2.549	1.480	80.279	80.522
65536	202.631	5.496	3.071	312.455	321.767



Aquí podemos observar que mergesort, median_of_3_quicksort e introsort son los que tienen peor desempeño dentro de los algoritmos cuasi-lienales. Esto porque también deben realizar muchos intercambios en cada corrida.

Zoom 2

N	Heapsort	QS Random
4096	0.233	0.140
8192	0.537	0.321
16384	1.169	0.660
32768	2.499	1.379
65536	5.266	2.952



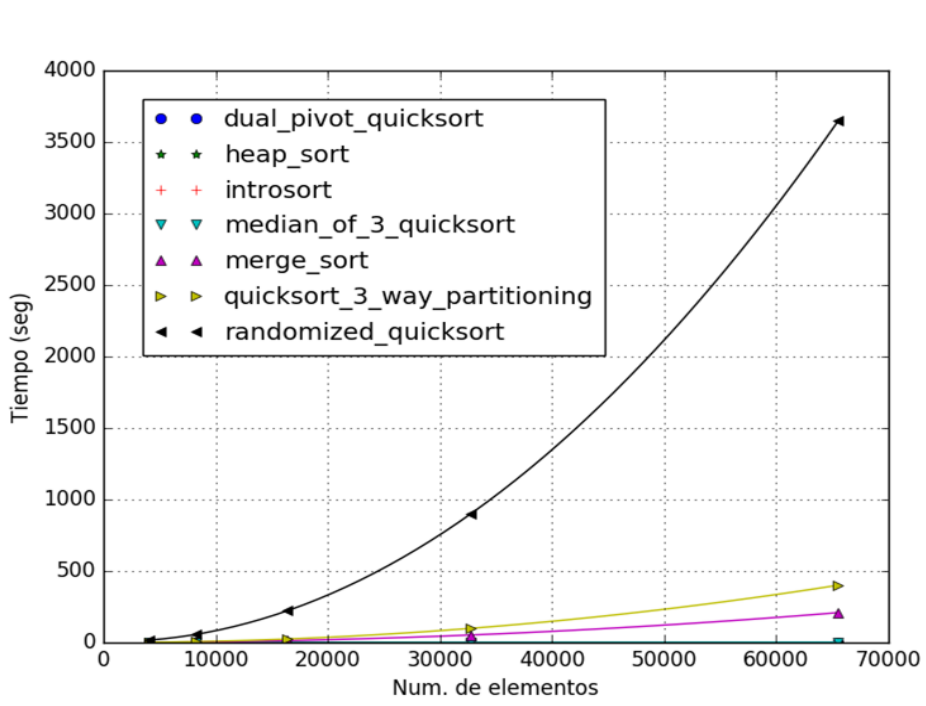
Randomized_quicksort es el mejor algoritmo para el ordenamiento de elementos que se encuentran ordenados descendientemente. Esto porque al escoger un pivote al azar, como en este caso no hay repeticiones de elementos, las particiones son balanceadas.

Conjunto de Datos 3:

Ceros y unos arreglados aleatoriamente

N	Mergesort	Heapsort	QS Random	Med-of-3 QS	Introsort	Dual pivot QS	QS 3-way
4096	1.025	0.125	14.072	0.077	0.077	0.099	1.559
8192	3.639	0.274	56.293	0.165	0.165	0.206	6.158
16384	13.629	0.579	226.661	0.389	0.388	0.450	24.546
32768	52.291	1.221	897.101	0.842	0.843	0.964	98.986

65536	209.078	2.623	3649.703	1.907	1.822	2.184	400.847

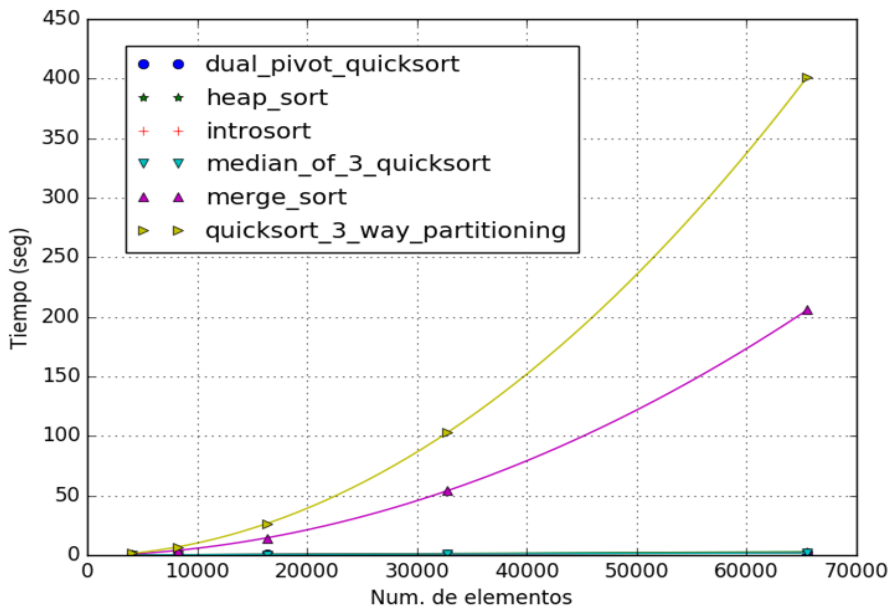


En este caso el peor algoritmo es randomized_quicksort, porque se repiten 0 y 1 en todo el arreglo e independientemente del pivote que se elija se realizaran muchas comparaciones e intercambios innecesarios. Como se habló anteriormente, quicksort no está optimizado para arreglos con repetición de elementos.

Zoom 1

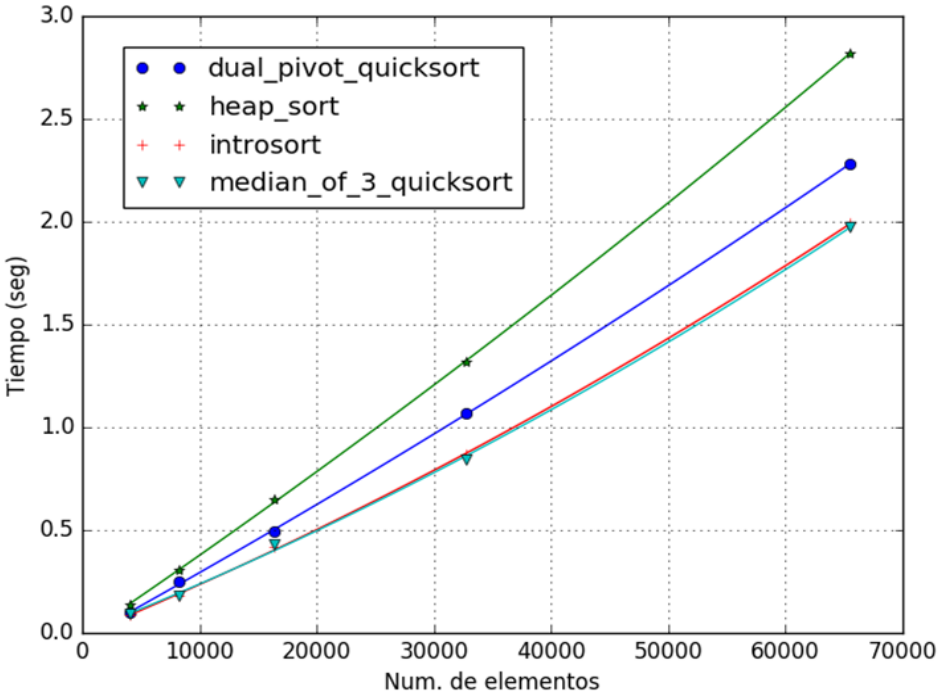
N	Mergesort	Heapsort	Med-of-3 QS	Introsort	Dual pivot QS	QS 3-way
4096	1.117	0.149	0.095	0.098	0.097	1.703
8192	3.817	0.289	0.190	0.217	0.217	6.612
16384	14.217	0.603	0.406	0.441	0.482	26.446
32768	54.280	1.323	0.873	0.889	1.037	102.958
65536	205.812	2.723	1.794	1.793	2.211	401.495

Aquí notamos que el siguiente peor algoritmo es quicksort_3_way_partitioning. Al ser este una versión más optimizada de quicksort vemos que presenta mejor comportamiento que el randomizado, sin embargo sigue en desventaja con los demás algoritmos en este caso de repetición de elementos. Mergesort es el tercer algoritmo más lento, sin embargo sigue teniendo un tiempo consistente con lo visto en teoría.



Zoom 2

N	Heapsort	Med-of-3 QS	Introsort	Dual pivot QS
4096	0.139	0.092	0.088	0.098
8192	0.303	0.178	0.181	0.248
16384	0.650	0.433	0.418	0.491
32768	1.316	0.844	0.867	1.069
65536	2.821	1.976	1.992	2.282

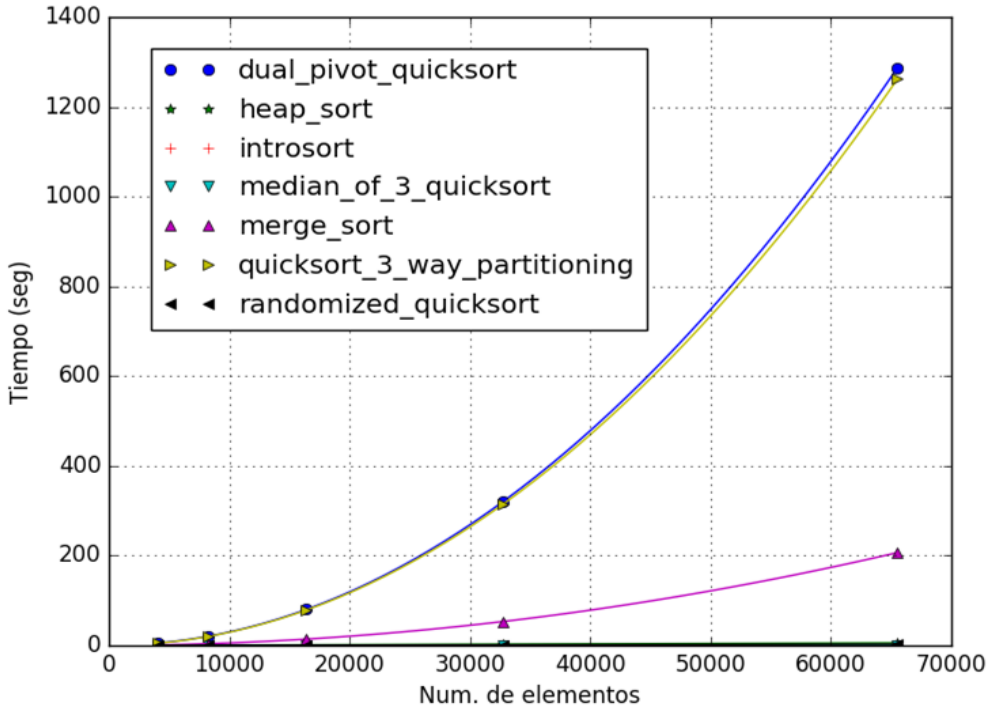


Median_of_3_ quicksort e introsort son los mejores algoritmos. Estos están muy optimizados para la repetición de elementos.

Conjunto de Datos 4:

Si el tamaño del arreglo es N, entonces el arreglo contendrá la secuencia 1; 2; . . . ; N-1; N.

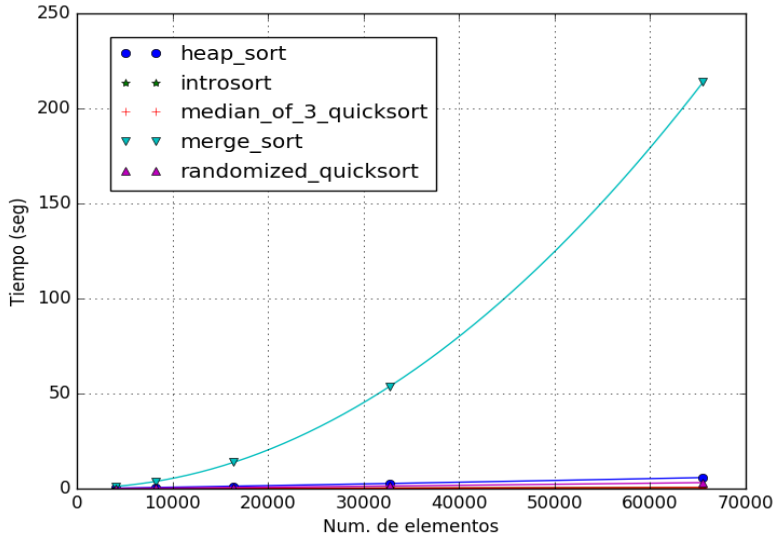
N	Mergesort	Heapsort	QS Random	Med-of-3 QS	Introsort	Dual pivot QS	QS 3-way
4096	1.116	0.264	0.135	0.029	0.029	5.071	5.059
8192	3.853	0.572	0.290	0.062	0.062	20.117	19.948
16384	13.749	1.272	0.644	0.130	0.139	81.115	78.087
32768	53.527	2.684	1.380	0.282	0.281	320.064	315.210
65536	206.836	5.683	2.926	0.597	0.598	1287.313	1261.941



Este debería ser el mejor caso pues el arreglo ya está ordenado, sin embargo vemos que dual_pivot_quicksort y quicksort_3_way_partitioning presentan tiempo semi-cuadratico. Esto porque al elegir un pivote, realiza todas las comparaciones y luego vuelve a colocar el pivote en su posición original, en este proceso se pierde tiempo.

Zoom 1

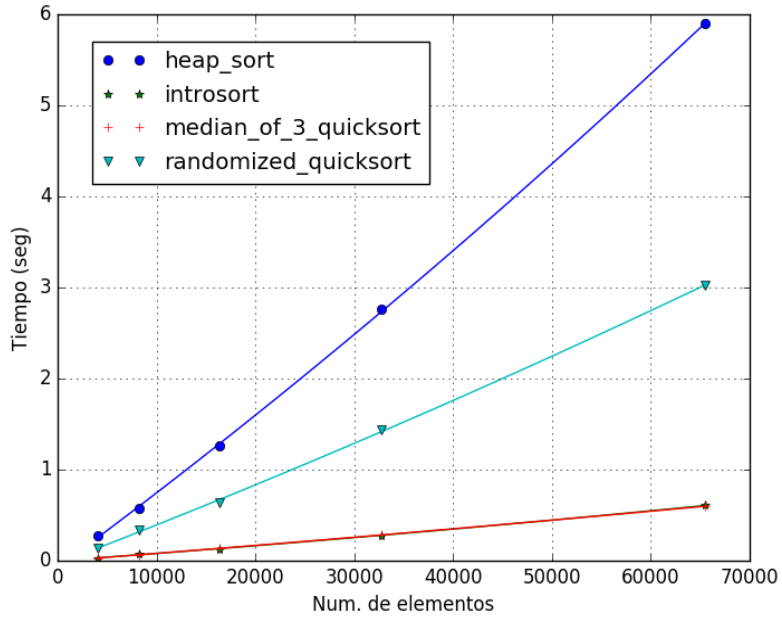
N	Mergesort	Heapsort	QS Random	Med-of-3 QS	Introsort
4096	1.071	0.273	0.142	0.032	0.033
8192	3.767	0.581	0.297	0.060	0.062
16384	14.008	1.278	0.611	0.130	0.166
32768	53.702	2.726	1.405	0.283	0.286
65536	213.852	5.797	3.157	0.596	0.606



Mergesort es el peor de los cuasi-lineales, esto porque independientemente de si esta ordenado o no el arreglo, mergesort realizará las mismas comparaciones sobre los subarreglos de tamaño 1.

Zoom 2

N	Heapsort	QS Random	Med-of-3 QS	Introsort
4096	0.280	0.140	0.029	0.030
8192	0.581	0.338	0.068	0.069
16384	1.258	0.632	0.132	0.130
32768	2.756	1.438	0.284	0.280
65536	5.904	3.028	0.599	0.607

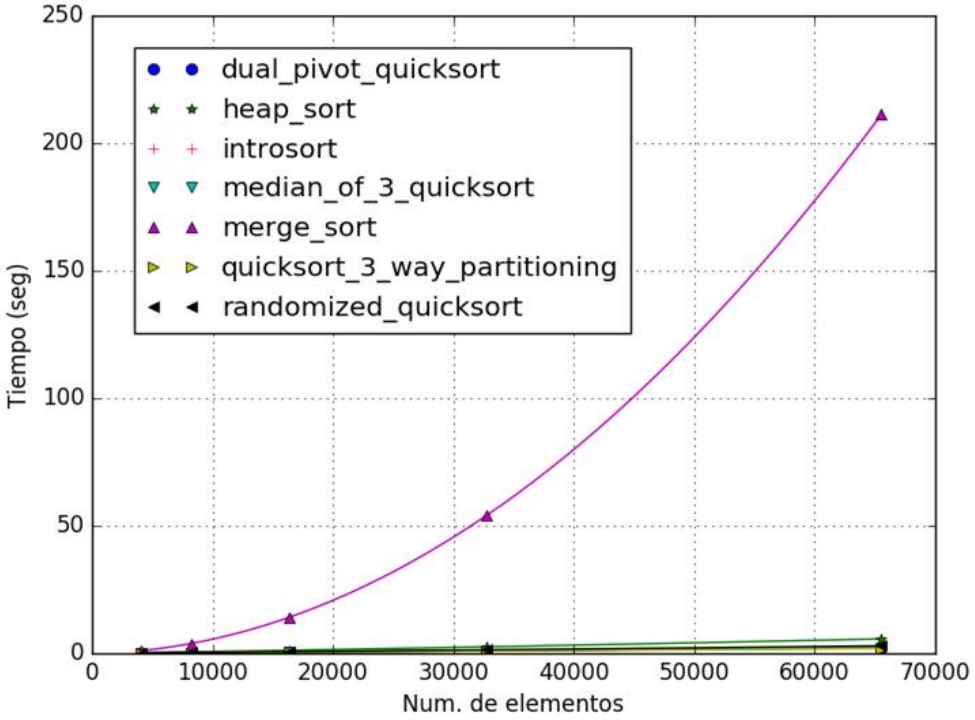


Los mejores algoritmos son median_of_3_quicksort e introsort. Estos algorimos se dan cuenta de que el arreglo esta ordenado.

Conjunto de Datos 5:

Números reales comprendidos en el intervalo [0; 1) generados aleatoriamente.

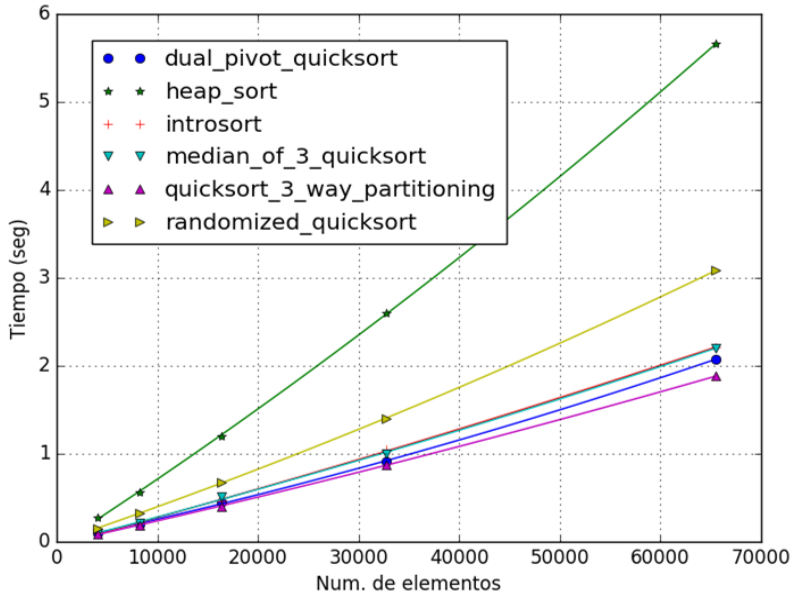
N	Mergesort	Heapsort	QS Random	Med-of-3 QS	Introsort	Dual pivot QS	QS 3-way
4096	1.056	0.248	0.206	0.100	0.100	0.088	0.096
8192	3.731	0.541	0.297	0.221	0.222	0.202	0.191
16384	14.173	1.171	0.653	0.475	0.489	0.407	0.399
32768	54.060	2.563	1.347	0.979	1.107	0.995	0.846
65536	211.254	5.638	2.925	2.133	2.152	2.045	1.812



En este caso como los números son reales aleatorios entre 0 y 1, las posibilidades de que se repitan los elementos son bajas. Además, no están ordenados de ninguna forma lo que representa en caso estándar para todos los algoritmos.

Zoom 1

N	Heapsort	QS Random	Med-of-3 QS	Introsort	Dual pivot QS	QS 3-way
4096	0.273	0.146	0.097	0.104	0.096	0.088
8192	0.560	0.330	0.212	0.213	0.206	0.190
16384	1.200	0.674	0.508	0.461	0.452	0.394
32768	2.602	1.400	1.003	1.044	0.909	0.879
65536	5.662	3.087	2.204	2.211	2.078	1.882

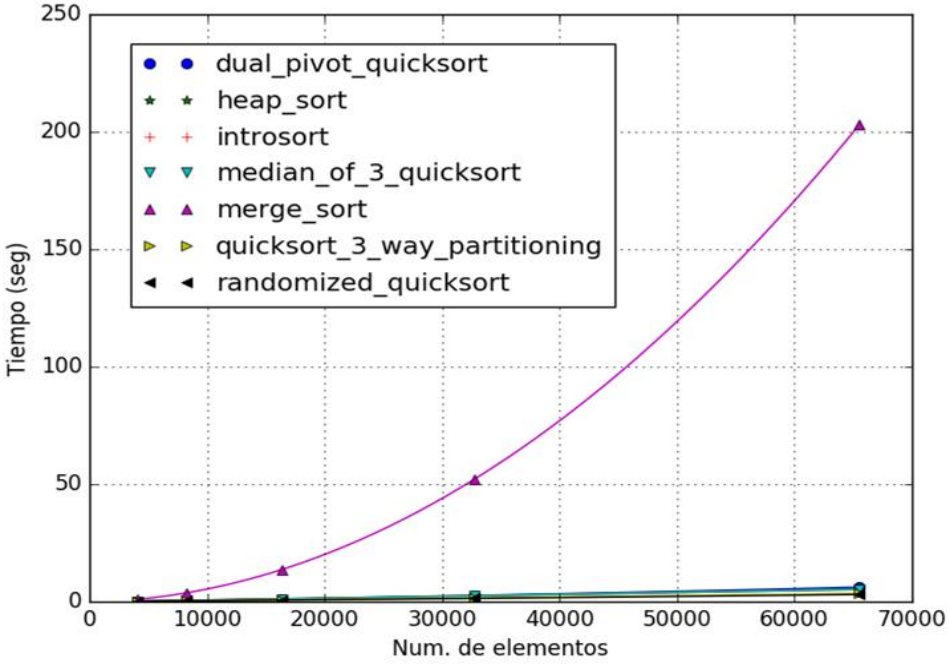


Casi todos los algoritmos se comportan de manera parecida, siendo el mejor quicksort_3_way _partitioning. En este caso las particiones son balanceadas. El que representa el algoritmo más lento de los lineales es heap_sort, sin embargo tiene un desempeño óptimo.

Conjunto de Datos 6:

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

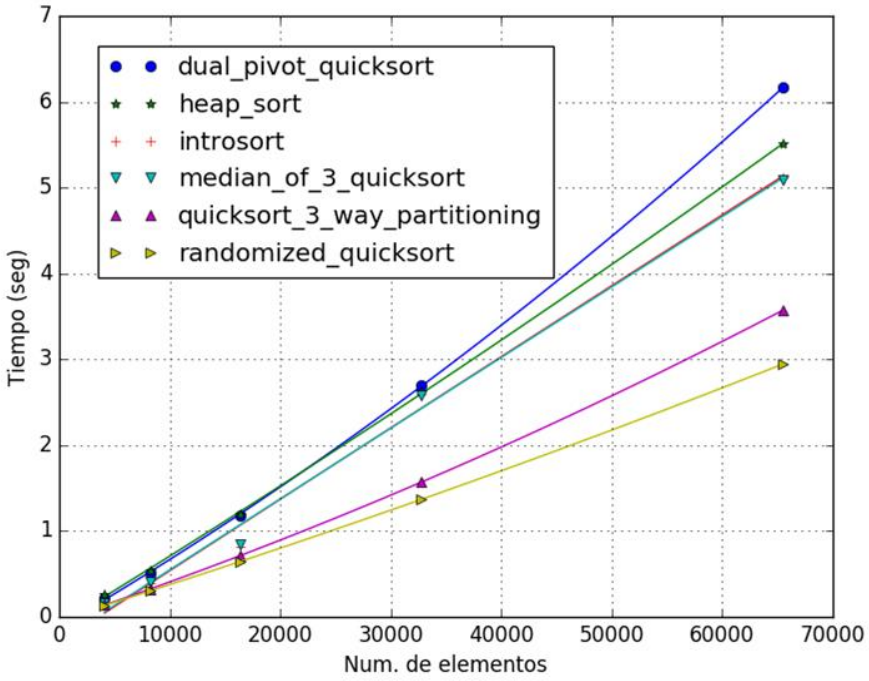
N	Mergesort	Heapsort	QS Random	Med-of-3 QS	Introsort	Dual pivot QS	QS 3-way
4096	1.030	0.254	0.135	0.154	0.154	0.216	0.141
8192	3.635	0.547	0.313	0.389	0.389	0.495	0.316
16384	13.587	1.190	0.603	0.818	0.820	1.161	0.712
32768	52.183	2.547	1.351	2.526	2.521	2.635	1.581
65536	203.042	5.544	3.037	5.126	5.073	6.134	3.496



En este caso todos los algoritmos se comportan de forma cuasi-lineal a excepción de mergesort que sigue comportándose como esperado en teoría.

Zoom 1

N	Heapsort	QS Random	Med-of-3 QS	Introsort	Dual pivot QS	QS 3-way
4096	0.259	0.135	0.156	0.156	0.215	0.142
8192	0.546	0.299	0.413	0.391	0.515	0.320
16384	1.202	0.643	0.840	0.821	1.174	0.708
32768	2.620	1.370	2.574	2.589	2.705	1.574
65536	5.514	2.948	5.093	5.113	6.168	3.575

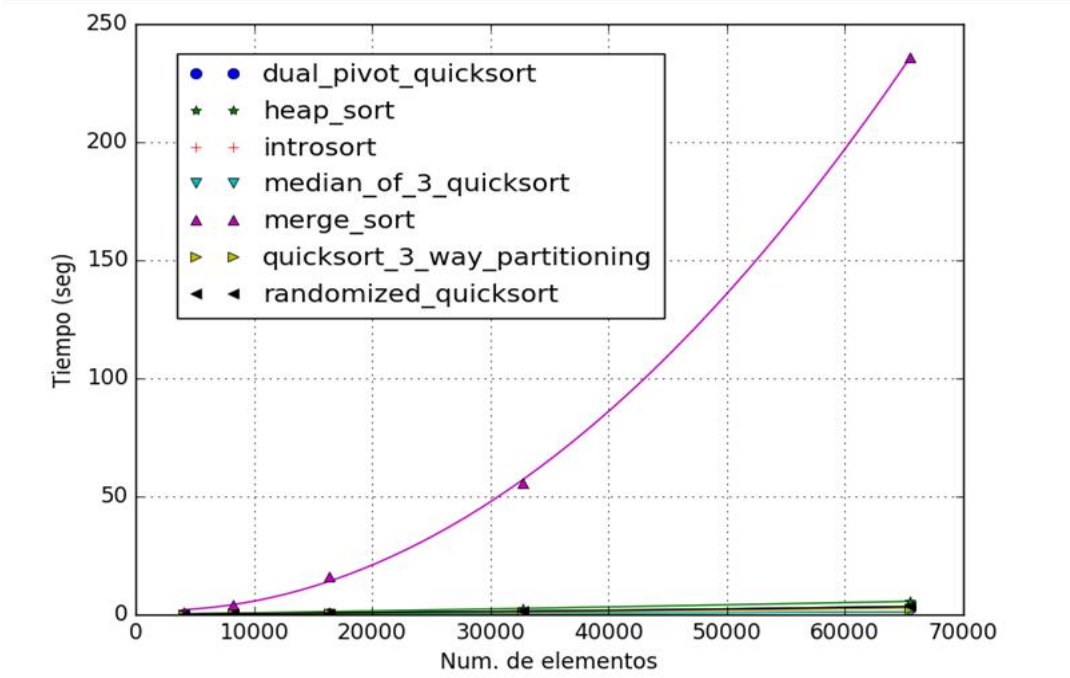


El mejor algoritmo en este caso es randomized _quicksort, como solo se repiten dos elementos por cada elemento en el arreglo entonces se obtienen particiones balanceadas.

Conjunto de Datos 7:

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.

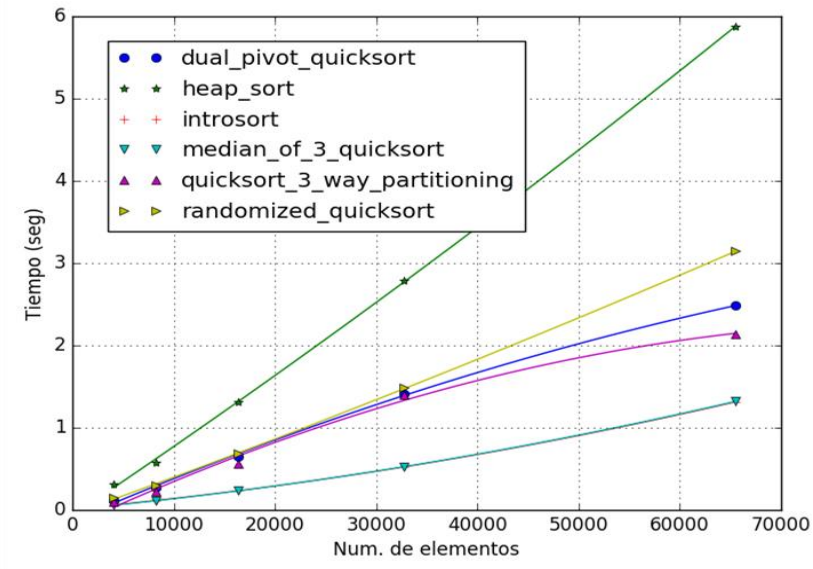
N	Mergesort	Heapsort	QS Random	Med-of-3 QS	Introsort	Dual pivot QS	QS 3-way
4096	1.191	0.252	0.140	0.043	0.044	0.120	0.101
8192	4.124	0.574	0.282	0.098	0.098	0.453	0.365
16384	16.013	1.239	0.665	0.214	0.214	0.604	0.512
32768	55.890	2.570	1.369	0.470	0.473	1.226	1.060
65536	236.157	5.586	3.391	1.071	1.118	3.381	2.664



En este caso los elementos están semi ordenados, a excepción de ciertos intercambios que se realizaron en la creación del arreglo. Así los algoritmos solo deben revertir dichos intercambios, de ahí la velocidad de los mismos.

Zoom 1

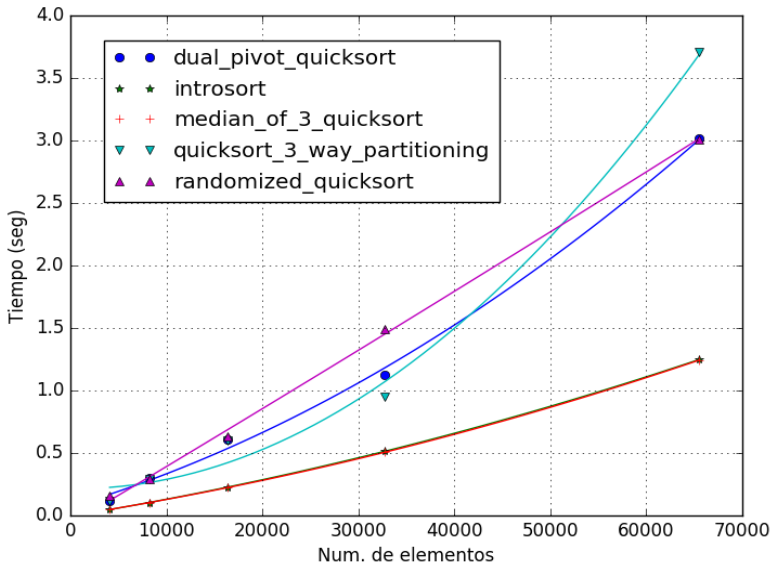
N	Heapsort	QS Random	Med-of-3 QS	Introsort	Dual pivot QS	QS 3-way
4096	0.313	0.150	0.058	0.060	0.112	0.103
8192	0.572	0.297	0.108	0.112	0.274	0.220
16384	1.307	0.684	0.238	0.241	0.650	0.560
32768	2.790	1.484	0.524	0.520	1.417	1.402
65536	5.879	3.144	1.322	1.316	2.481	2.137



Heapsort es el más lento de los algoritmos de tiempo lineal en este caso. Sin embargo, sigue presentando un buen desempeño

Zoom 2

N	QS Random	Med-of-3 QS	Introsort	Dual pivot QS	QS 3-way
4096	0.157	0.048	0.048	0.120	0.119
8192	0.291	0.103	0.104	0.299	0.293
16384	0.636	0.224	0.227	0.612	0.603
32768	1.489	0.505	0.514	1.129	0.948
65536	3.010	1.244	1.249	3.017	3.710



Los mejores algoritmos son median_of_3_quicksort e introsort, quienes tienen prácticamente el mismo desempeño en este caso. Por otro lado, quicksort_3_way_ partitioning funciona mejor con arreglos de menor tamaño.

Conclusiones

Finalmente podemos ver que hay algoritmos que mantienen un tiempo estable en todas las pruebas como es el caso de mergesort. Así podemos ver que en mergesort se ejecutaran las mismas acciones independientemente del arreglo de entrada. Otro algoritmo que mantiene un tiempo predecible en todas las pruebas es heapsort, cuyo mejor caso fue en el conjunto de datos 3, cuando el arreglo tenía 0 y 1 en sus elementos. Sin embargo ese fue el peor caso para randomized quicksort, porque aquí se repiten mucho las cifras y las particiones no quedan balanceadas. En línea general todos los resultados coinciden con los vistos en teoría, pues el desempeño de los algoritmos para cada caso fue el esperado.