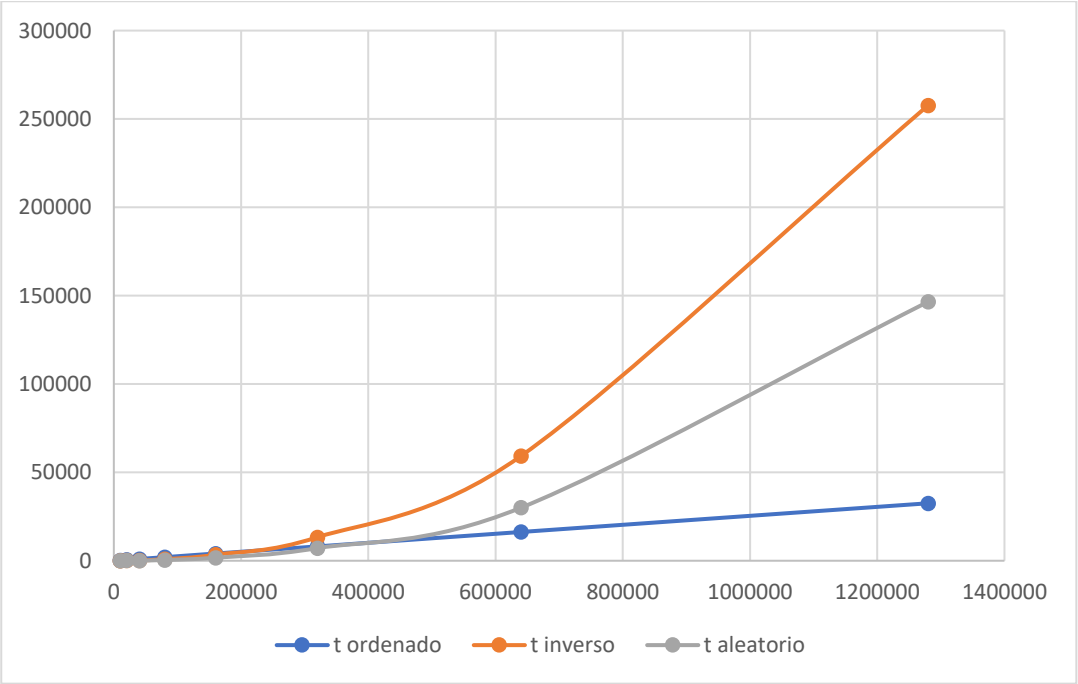
PANICO GIANPIETRO

***El tiempo de toda las operaciones se midió en un procesador intel core i7-1185G7, y 16,0 GB de memoria. El tiempo se expresa siempre en milisegundos.*

ALGORITMO INSERCION DIRECTA

nVeces= 1000000

N	t ordenado	t inverso	t aleatorio
10000	272	75	41
20000	512	254	126
40000	1014	213	116
80000	2017	885	416
160000	4084	3420	1734
320000	8102	13316	7130
640000	16271	59270	30012
1280000	32513	257695	146546



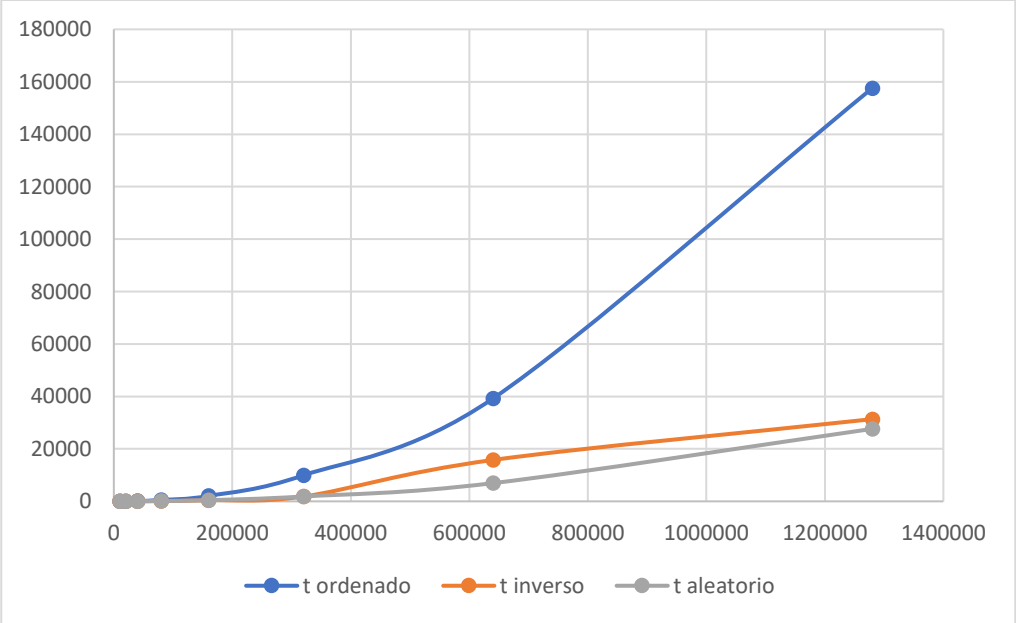
ALGORITMO SELECTION

- Opcion: ordenado. nVeces= 1000 de n=1000 a 2000, nVeces=100 de 4000 en adelante. Asi que los valores 1000 y 2000 se han dividido por 10. (114/10 = 11.4, por eso es un valor <50).

- Opcion: inversa. nVeces= 1000 de n=1000 a 4000, nVeces=10 de 8000 en adelante. Por lo tanto, los valores 1000 y 4000 se han dividido por 100. (228/100 = 2,28, por lo que es un valor <50).

- Opcion: aleatoria. nVeces= 1000 de n=1000 a 4000, nVeces=10 de 8000 en adelante. Asi que los valores 1000 y 4000 se han dividido por 100. (185/100 = 1.85, por eso es un valor <50).

N	t ordenado	t inverso	t aleatorio
10000	11,4	2,28	1,85
20000	39	6,84	7,11
40000	126	27,79	28,41
80000	520	118	140
160000	2071	438	453
320000	9936	1837	1859
640000	39249	15786	6947
1280000	157565	31343	27641



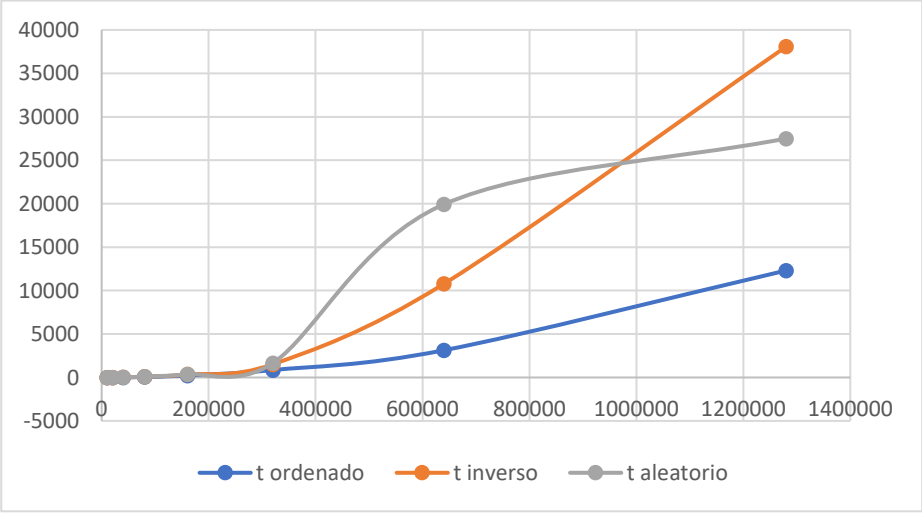
ALGORITMO BURBUJA

- Opcion: ordenado. nVeces= 1000 de n=1000 a 4000, nVeces=10 de 8000 en adelante. Asi que los valores 1000 y 2000 se han dividido por 10. (105/100 = 1.05, por eso es un valor <50).

- Opcion: inversa. nVeces= 1000 de n=1000 a 4000, nVeces=10 de 8000 en adelante. Entonces los valores 1000 y 2000 se dividieron por 10. (175/100 = 1.75, por esta razón es un valor <50).

- Opcion: inversa. nVeces= 1000 de n=1000 a 4000, nVeces=10 de 8000 en adelante. Entonces los valores 1000 y 2000 fueron divididos por 10. (121/100 = 1.21, por esta razón es un valor <50).

N	t ordenado	t inverso	t aleatorio
10000	1,05	1,75	1,21
20000	3,17	5,46	3,63
40000	12,61	22,27	14,22
80000	64	93	105
160000	233	345	377
320000	857	1518	1653
640000	3132	10774	19927
1280000	12308	38073	27496



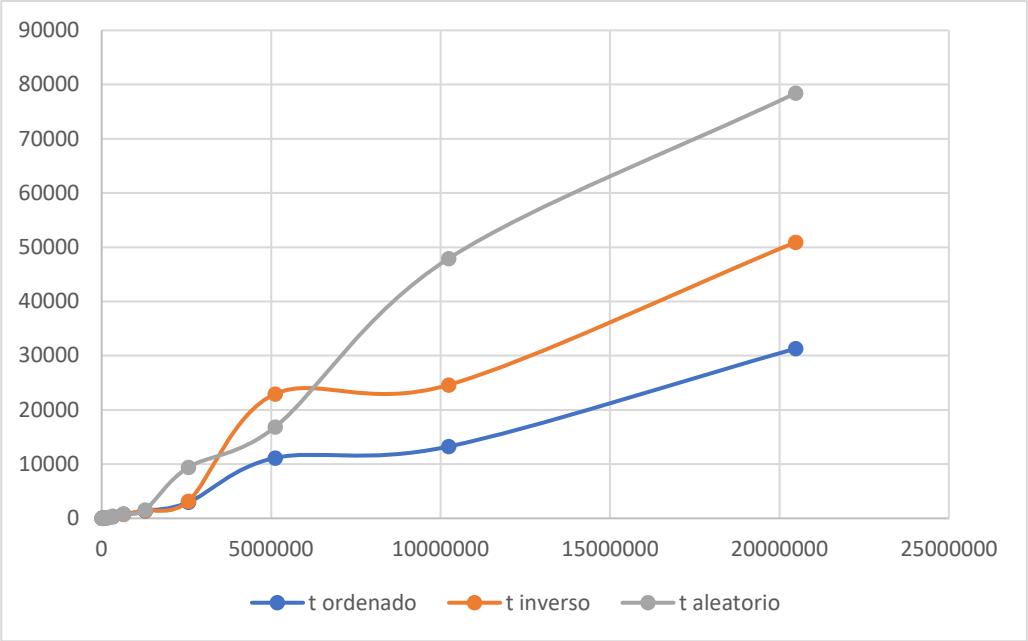
ALGORITMO QUICKSORT

- Opcion: ordenado. nVeces= 10000 de n=1000 a 4000, nVeces=1000 de 8000 en adelante. Entonces los valores 1000,2000 y 4000 se han dividido por 10. (93/10 = 9.3, por esta razon es un valor <50).

- Opcion: inversa. nVeces= 10000 de n=1000 a 4000, nVeces=1000 de 8000 en adelante. Entonces los valores 1000,2000 y 4000 se han dividido por 10. (112/10 = 11.2, por esta razón es un valor <50).

- Opcion: aleatoria. nVeces= 10000 de n=1000 a 4000, nVeces=1000 de 8000 en adelante. Entonces los valores 1000,2000 y 4000 han sido divididos por 10. (107/10 = 10.7, por esta razón es un valor <50).

N	t ordenado	t inverso	t aleatorio
10000	9,3	11,2	10,7
20000	18,3	19,9	20,7
40000	37,2	40,5	44,9
80000	86	87	90
160000	153	169	164
320000	344	368	392
640000	799	747	787
1280000	1369	1423	1582
2560000	2965	3182	9432
5120000	11146	22953	16857
10240000	13246	24613	47921
20480000	31309	50933	78415



-Describe en el documento en que consiste este método de selección, cuándo funciona mal y cuando no y que efecto tiene en el tiempo de ejecución.

La implementación de la clase RapidoFatal utiliza el primer elemento del array como pivote, lo que puede no ser representativo de la distribución de elementos. Esto puede provocar un deterioro del rendimiento, especialmente cuando el array ya está ordenado o casi ordenado.
El tiempo de ejecución del algoritmo QuickSort depende, por tanto, de la elección del pivote y de la distribución de los elementos en la matriz. Existen varias técnicas para mejorar la elección del pivote, por ejemplo, eligiéndolo aleatoriamente o utilizando un algoritmo de selección de la mediana.
Sin embargo, para matrices de tamaño moderado, el algoritmo 'RapidoFatal' puede seguir siendo eficiente, especialmente si la matriz no está ya ordenada o casi ordenada.