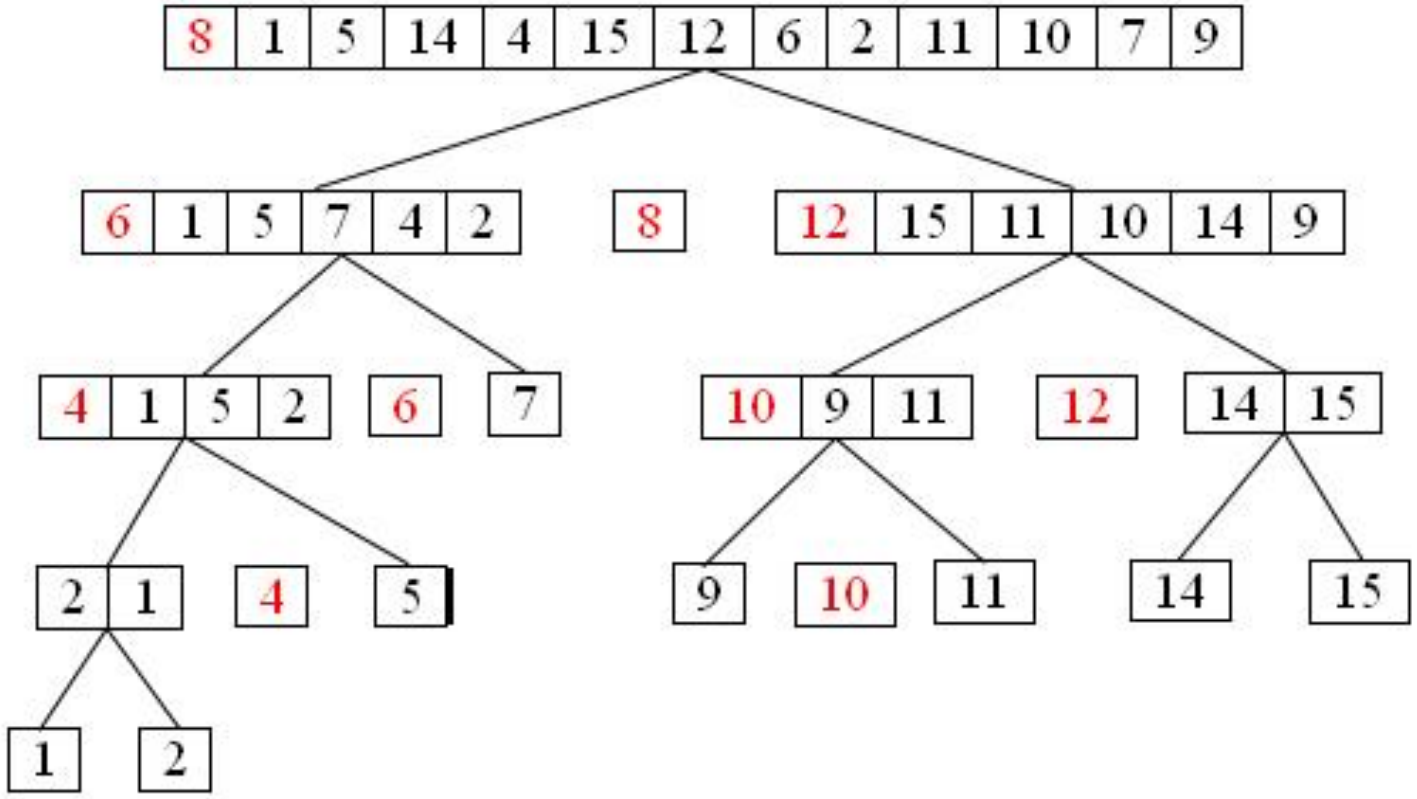
QuickSort

- QuickSort aplica el principio de “Divide y venceras” divide el problema, en problemas más pequeños de forma recursiva y luego combina todas las soluciones de forma automática



Juntando los elementos, el arreglo quedaría ordenado

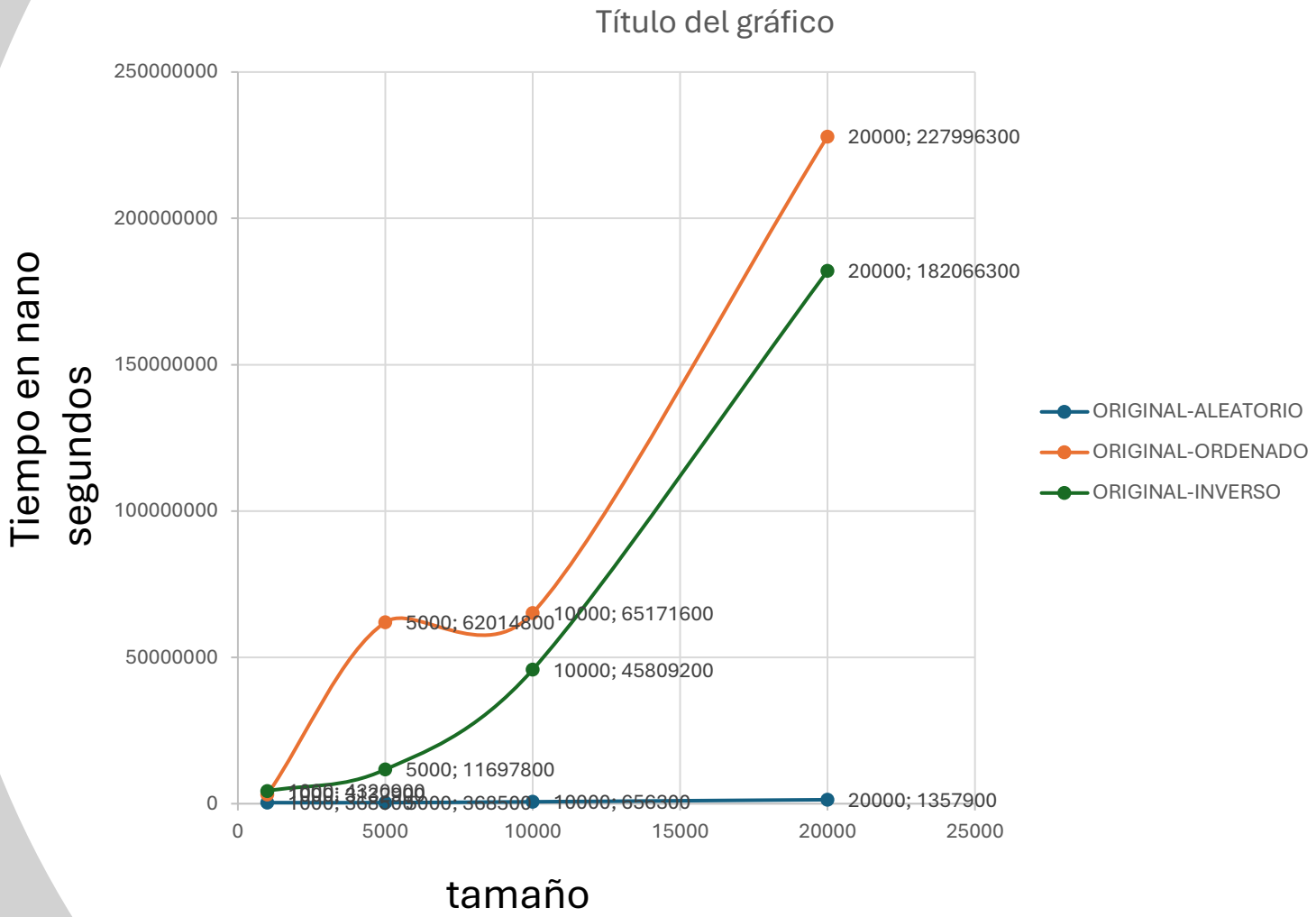
1	2	4	5	6	7	8	9	10	11	12	14	15

QuickSort

A continuación realizaremos la prueba del algoritmo de ordenamiento (QuickSort) en Java utilizando tamaños de datos en los arreglos diferentes estos serán 1000, 5000,10000 y 20000, este consiste básicamente en colocar un pivote al final del arreglo y el sub arreglo como se vio anteriormente tras los tes se pudo obtener la tabla siguiente.

TAMAÑO	ORIGINAL-ALEATORIO	ORIGINAL-ORDENADO	ORIGINAL-INVERSO
1000	368500	3132900	4320900
5000	368500	62014800	11697800
10000	656300	65171600	45809200
20000	1357900	227996300	182066300

Tiempo en nano segundos

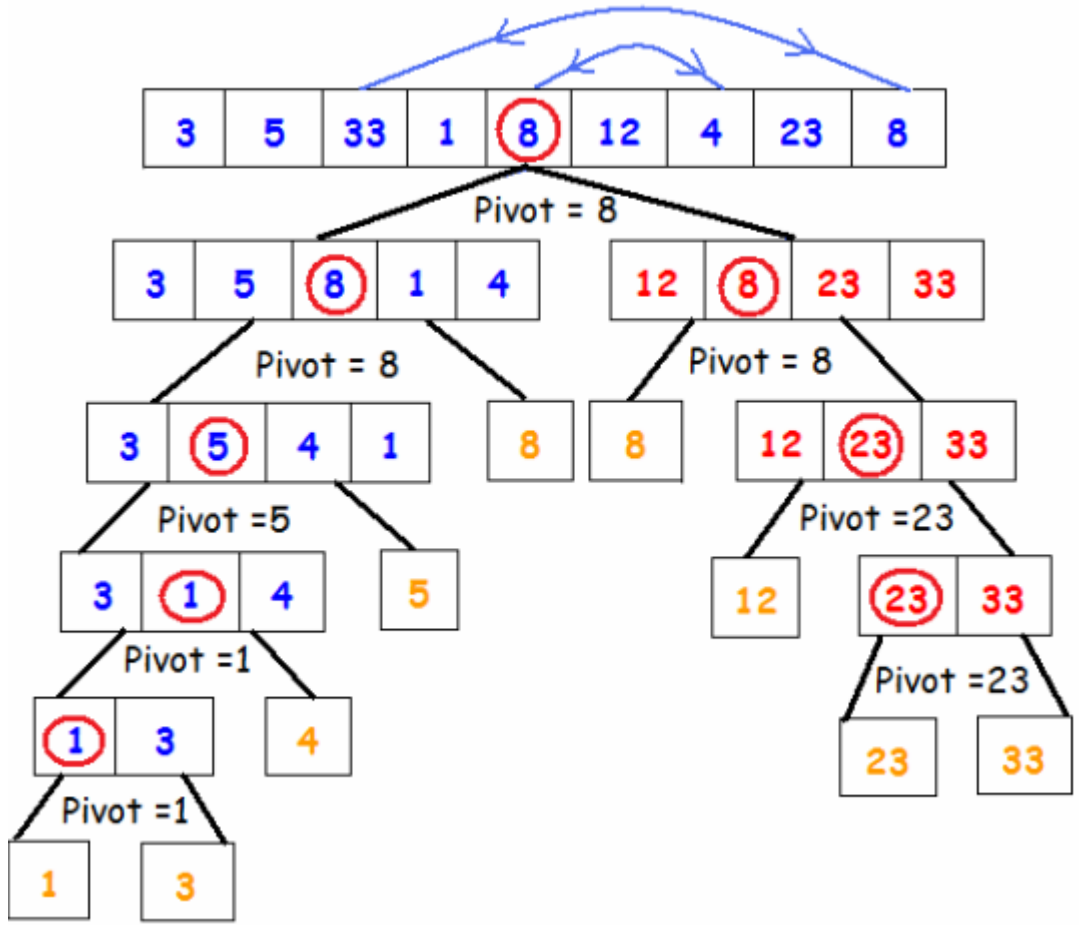

Título del gráfico
Tiempo en nano segundos
tamaño
250000000
200000000
150000000
100000000
50000000
0
0
5000
10000
15000
20000
25000
20000; 227996300
20000; 182066300
5000; 62014800
10000; 65171600
10000; 45809200
5000; 11697800
10000; 656300
20000; 1357900
ORIGINAL-ALEATORIO
ORIGINAL-ORDENADO
ORIGINAL-INVERSO

QuickSort

Como podemos observar en los diferentes escenarios el ordenamiento más rápido es el ordenamiento aleatorio y el que le sigue es el ordenamiento inverso, de alguna forma el ordenado es el más lento

QuickSort con pivote en medio

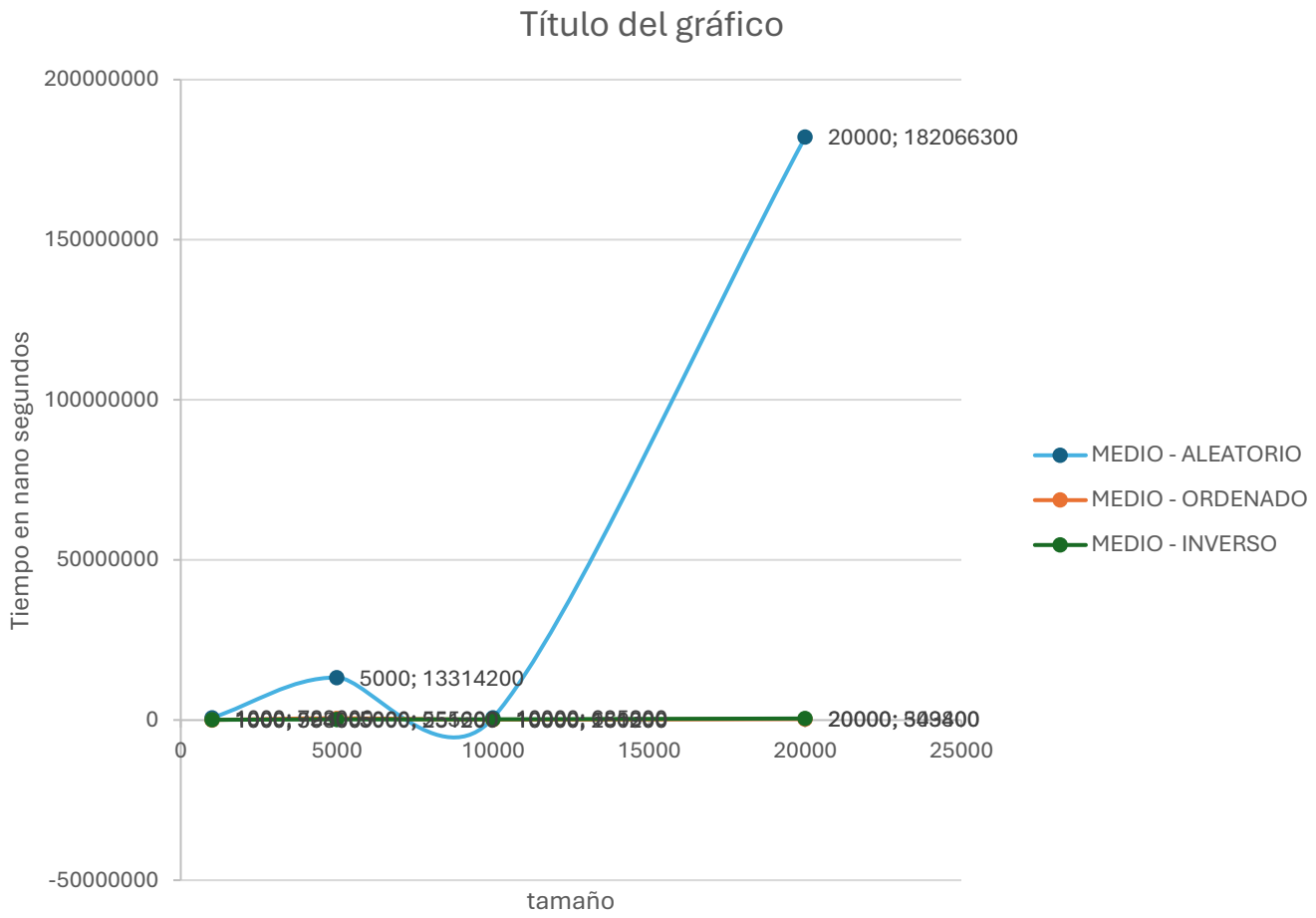
- Esta alternativa consiste en seleccionar como pivote el elemento que esta en la posición media del arreglo o sub arreglo



QuickSort pivote en la posición media

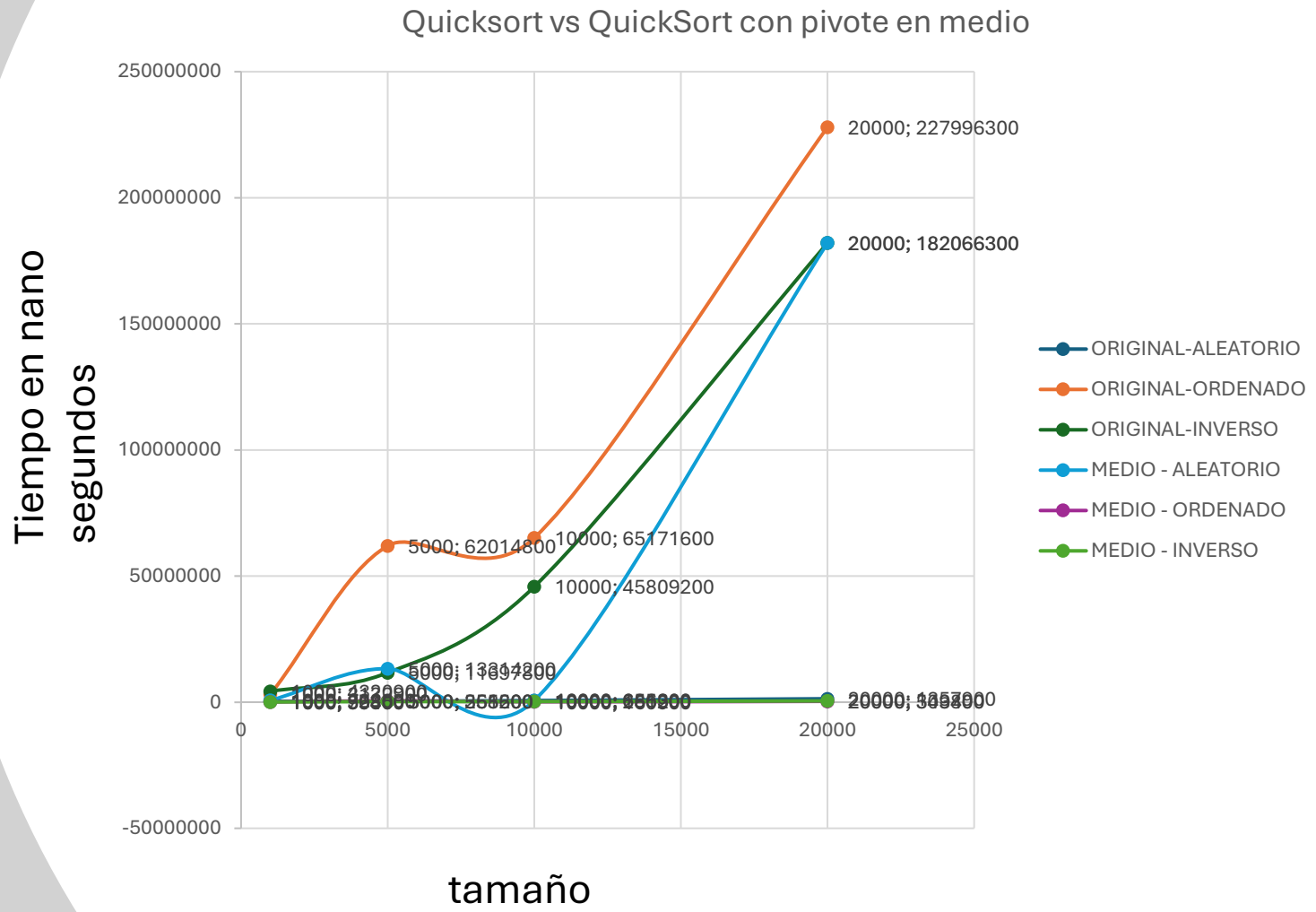
A continuación realizaremos la prueba del algoritmo de ordenamiento (QuickSort con pivote en la posición media) en Java utilizando tamaños de datos en los arreglos diferentes estos serán 1000, 5000,10000 y 20000, este consiste básicamente en colocar un pivote en medio del arreglo y el sub arreglo como se vio anteriormente tras los tes se pudo obtener la tabla siguiente.

TAMAÑO	MEDIO - ALEATORIO	MEDIO - ORDENADO	MEDIO - INVERSO
1000	723000	59400	90800
5000	13314200	555600	251200
10000	685300	160900	251200
20000	182066300	343800	509400


Título del gráfico
200000000
150000000
100000000
50000000
0
-50000000
Tiempo en nano segundos
0
5000
10000
15000
20000
25000
tamaño
20000; 182066300
5000; 13314200
MEDIO - ALEATORIO
MEDIO - ORDENADO
MEDIO - INVERSO

QuickSort con pivote intemedio

Como podemos observar esta alternativa de algoritmo es mucho más eficiente en la gran parte de los casos, excepto en el ordenamiento de datos aleatorios ahí se queda un poco corto.


Quicksort vs QuickSort con pivote en medio
Tiempo en nano segundos
tamaño
250000000
200000000
150000000
100000000
50000000
0
-50000000
0
5000
10000
15000
20000
25000
20000; 227996300
20000; 182066300
5000; 62014800
10000; 65171600
10000; 45809200
ORIGINAL-ALEATORIO
ORIGINAL-ORDENADO
ORIGINAL-INVERSO
MEDIO - ALEATORIO
MEDIO - ORDENADO
MEDIO - INVERSO

comparación

En conclusión el ordenamiento QuickSort con pivote en la posición media es más eficiente que el QuickSort normal, más eficiente en listas ordenadas e inversas no tanto con aleatorias, y el QuickSort normal es más eficiente con listas aleatorias y no tanto con ordenadas e inversas.