PRÁCTICA 2: ORDENACIÓN

Información del sistema:

Procesador:	11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.8GHz
Memoria:	16 GB

Toma de tiempos del algoritmo de **Inserción**:

N	nVeces	t ordenado		t ordenado/nVeces
		miliseg	seg	
10000	100000	180	0,18000	0,0000018000
20000		300	0,30000	0,0000030000
40000		618	0,61800	0,0000061800
80000		1193	1,19300	0,0000119300
160000		2387	2,38700	0,0000238700
320000		4877	4,87700	0,0000487700
640000		27583	27,58300	0,0002758300
1280000		60807	60,80700	0,0006080700
2560000		129589	129,58900	0,0012958900
5120000		280268	280,26800	0,0028026800
10240000		570010	570,01000	0,0057001000
20480000	10	72	0,07200	0,0072000000
40960000		111	0,11100	0,0111000000
81920000		219	0,21900	0,0219000000
163840000		451	0,45100	0,0451000000
327680000		906	0,90600	0,0906000000
Java heap space				

N	nVeces	t inverso		t inverso/nVeces
		miliseg	seg	
10000	100	63	0,06300	0,0006300000
20000		214	0,21400	0,0021400000
40000		689	0,68900	0,0068900000
80000		2729	2,72900	0,0272900000
160000		24394	24,39400	0,2439400000
320000		55937	55,93700	0,5593700000
640000		174716	174,71600	1,7471600000
1280000	Tarda más de 10min con nVeces= 1			

N	nVeces	t aleatorio		t aleatorio/nVeces
		miliseg	seg	
10000	10000	114	0,11400	0,0000114000
20000		178	0,17800	0,0000178000
40000		414	0,41400	0,0000414000
80000		1102	1,10200	0,0001102000
160000		7464	7,46400	0,0007464000
320000		30323	30,32300	0,0030323000
640000		116417	116,41700	0,0116417000
1280000		459373	459,37300	0,0459373000
2560000	Tarda más de 10min con nVeces= 1			

Toma de tiempos del algoritmo de **Selección**:

N	nVeces	t ordenado		t ordenado/nVeces
		miliseg	seg	
10000	10	83	0,08300	0,0083000
20000		306	0,30600	0,0306000
40000		1225	1,22500	0,1225000
80000		4869	4,86900	0,4869000
160000		49440	49,44000	4,9440000
320000		140870	140,87000	14,0870000
640000	1	64732	64,73200	64,7320000
1280000		112961	112,96100	112,9610000
2560000	Tarda más de 10min con nVeces= 1			

N	nVeces	t inverso		t inverso/nVeces
		miliseg	seg	
10000	10	246	0,24600	0,0246000000
20000		827	0,82700	0,0827000000
40000		6278	6,27800	0,6278000000
80000		26562	26,56200	2,6562000000
160000		67469	67,46900	6,7469000000
320000		334028	334,02800	33,4028000000
640000	1	115083	115,08300	115,0830000000
1280000		478460	478,46000	478,4600000000
2560000	Tarda más de 10min con nVeces= 1			

N	nVeces	t aleatorio		t aleatorio/nVeces
		miliseg	seg	
10000	10	211	0,21100	0,0211000000
20000		686	0,68600	0,0686000000
40000		2708	2,70800	0,2708000000
80000		26451	26,45100	2,6451000000
160000		63699	63,69900	6,3699000000
320000		179886	179,88600	17,9886000000
640000	1	73804	73,80400	73,8040000000
1280000		271274	271,27400	271,2740000000
2560000	Tarda más de 10min con nVeces= 1			

Toma de tiempos del algoritmo **Burbuja**:

N	nVeces	t ordenado		t ordenado/nVeces
		miliseg	seg	
10000	10	134	0,13400	0,0134000
20000		362	0,36200	0,0362000
40000		1363	1,36300	0,1363000
80000		5478	5,47800	0,5478000
160000		66222	66,22200	6,6222000
320000		269681	269,68100	26,9681000
640000	Tarda más de 10min con nVeces= 1			

N	nVeces	t inverso		t inverso/nVeces
		miliseg	seg	
10000	1	67	0,06700	0,0670000
20000		244	0,24400	0,2440000
40000		898	0,89800	0,8980000
80000		7410	7,41000	7,4100000
160000		43628	43,62800	43,6280000
320000		175307	175,30700	175,3070000
640000	Tarda más de 10min con nVeces= 1			

N	nVeces	t aleatorio		t aleatorio/nVeces
		miliseg	seg	
10000	1	94	0,09400	0,0940000
20000		386	0,38600	0,3860000
40000		1586	1,58600	1,5860000
80000		17335	17,33500	17,3350000
160000		28333	28,33300	28,3330000
320000		162997	162,99700	162,9970000
640000	Tarda más de 10min con nVeces= 1			

Toma de tiempos del algoritmo **Quicksort (Mediana a tres)**:

N	nVeces	t ordenado		t ordenado/nVeces
		miliseg	seg	
10000	1000	105	0,10500	0,0001050
20000		205	0,20500	0,0002050
40000		392	0,39200	0,0003920
80000		891	0,89100	0,0008910
160000		1717	1,71700	0,0017170
320000		3819	3,81900	0,0038190
640000		7305	7,30500	0,0073050
1280000		16213	16,21300	0,0162130
2560000		32345	32,34500	0,0323450
5120000		72078	72,07800	0,0720780
10240000		141991	141,99100	0,1419910
20480000		319765	319,76500	0,3197650
40960000	10	682	0,68200	0,0682000
81920000		1646	1,64600	0,1646000
163840000		3504	3,50400	0,3504000
327680000		7048	7,04800	0,7048000
Java heap space				

N	nVeces	t inverso		t inverso/nVeces
		miliseg	seg	
10000	1000	112	0,11200	0,0001120
20000		212	0,21200	0,0002120
40000		417	0,41700	0,0004170
80000		966	0,96600	0,0009660

160000		1898	1,89800	0,0018980
320000		4816	4,81600	0,0048160
640000		9012	9,01200	0,0090120
1280000		21303	21,30300	0,0213030
2560000		40732	40,73200	0,0407320
5120000		87766	87,76600	0,0877660
10240000		152638	152,63800	0,1526380
20480000		360498	360,49800	0,3604980
40960000	1	1256	1,25600	1,2560000
81920000		2862	2,86200	2,8620000
163840000		6315	6,31500	6,3150000
327680000		13213	13,21300	13,2130000
Java heap space				

N	nVeces	t aleatorio		t aleatorio/nVeces
		miliseg	seg	
10000	1000	114	0,11400	0,0001140
20000		217	0,21700	0,0002170
40000		425	0,42500	0,0004250
80000		975	0,97500	0,0009750
160000		2007	2,00700	0,0020070
320000		5064	5,06400	0,0050640
640000		10536	10,53600	0,0105360
1280000		24010	24,01000	0,0240100
2560000		52512	52,51200	0,0525120
5120000		102775	102,77500	0,1027750
10240000		205813	205,81300	0,2058130
20480000		440210	440,21000	0,4402100
40960000	1	3438	3,43800	3,4380000
81920000		7310	7,31000	7,3100000
163840000		17022	17,02200	17,0220000
327680000		47565	47,56500	47,5650000
Java heap space				

El método **Quicksort RapidoFatal** comienza con el pivote como el primer elemento de la lista. Su caso fatal es cuando se le pasa una lista de elementos ya ordenados pues su complejidad aumenta, O(n^2).