MAC0121 – EP3

Nome: Adriano Elias Andrade
NUSP: 13671682

Introdução:

Nesse trabalho, pretende-se testar empiricamente diferentes algoritmos de ordenação de palavras em C, a fim de compará-los em questão de eficiência. Para isso, serão analisados dados como número de comparações, número de trocas (movimentações), tempo de execução e o limite de elementos suportados para cada um deles.

Descrição dos métodos testados:

Serão testados quatro algoritmos de ordenação diferentes: insertionSort, quicksort (esolhendo o último elemento como pivô), mergesort e heapsort (corrigindo o heap descendo).

Descrição do tipos de instâncias:

Para cada algoritmo, serão testados 3 diferentes tipos de instâncias: aleatórias, ordenadas e parcialmente ordenadas. Para as instâncias parcialmente ordenadas, foram sorteados 20% dos elementos para trocar de lugar entre si, em pares de índice aleatório.

Esses 3 tipos de instâncias serão testados para listas de 250 palavras até o limite que o programa for capaz de suportar, ou até demorarem um tempo muito grande.

Resultados:

Para uma entrada com mais de 1024000 palavras, não foi possível ordenar em nenhum dos algoritmos devido a problemas de escaneamento de entrada, e por isso, as tabelas vão somente até 1024000 palavras.

Obs: nos gráficos, o mergesort e o heapsort se comportam de modo semelhante, e por isso a linha amarela do heapsort acaba sobrepondo a verde do mergesort.

Comparações (aleatório):

Tamanho	insertionSort	quicksort	mergesort	heapsort
250	16402	3803	2510	3207
500	60811	11353	5827	7380
1000	250113	29840	13137	16778
2000	996238	86401	29207	37638
4000	3942606	290951	64300	83289
8000	15816441	997497	140576	182234
16000	64300795	3202632	305543	396779
32000	255827079	11542533	659211	858209
64000	1025048650	44650349	1414065	1844972
128000	4094667031	44650349	3021049	3946683
256000	16365429736	695003988	6428459	8406082
512000	muito tempo	2726575806	13618186	17826500
1024000		10587224360	28770696	37709124

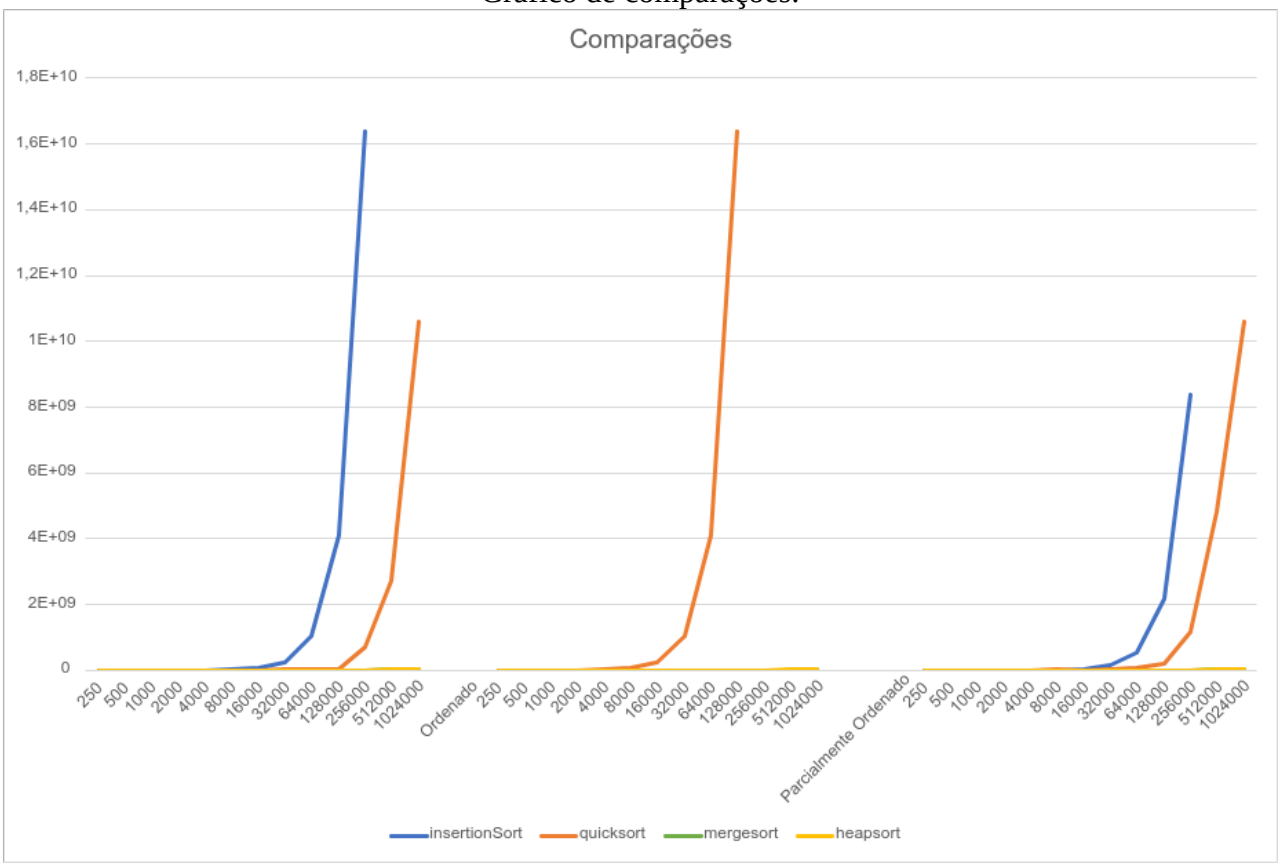
Comparações (ordenado):

Tamanho	insertionSort	quicksort	mergesort	heapsort
250	249	62250	2022	744
500	499	249500	4544	1494
1000	999	999000	10088	4602
2000	1999	3998000	22176	24129
4000	3999	15996000	48352	67790
8000	7999	63992000	104704	162447
16000	15999	255984000	225408	367680
32000	31999	1023968000	482816	802944
64000	63999	4095936000	1029632	1725966
128000	127999	16383872000	2187264	3688237
256000	255999	segmentation fault	4630528	7891810
512000	511999		9773056	16667970
1024000	1023999		20570112	35011774

Comparações (parcialmente ordenado):

Tamanho	insertionSort	quicksort	mergesort	heapsort
250	9743	10144	2067	1849
500	38288	38909	4633	3863
1000	145353	153010	10260	9355
2000	579374	597225	22502	30080
4000	2239158	2364364	48990	75706
8000	8808341	9225757	105906	174882
16000	34806482	38983120	228107	389089
32000	138786289	156043452	488204	849689
64000	552220569	622724749	1040397	1830932
128000	2173761626	2496474822	2208782	3926303
256000	8368024736	10358179472	4665757	8364484
512000	muito tempo	4803587425	13666313	17683662
1024000		10603942752	28742649	37037660

Gráfico de comparações:



Movimentações (aleatório):

Tamanho	insertionSort	quicksort	mergesort	heapsort
250	16402	1055	1673	1768
500	60811	2643	3859	3992
1000	250113	6439	8703	9027
2000	996238	14522	19434	20043
4000	3942606	30442	42791	44084
8000	15816441	73412	93537	96009
16000	64300795	129316	203276	208369
32000	255827079	301316	438564	449327
64000	1025048650	640466	940968	963542
128000	4094667031	1362297	2010082	2055354
256000	16365429736	3089054	4276504	4362657
512000	muito tempo	1361521300	9063174	9239653
1024000		5290215133	19149018	19503916

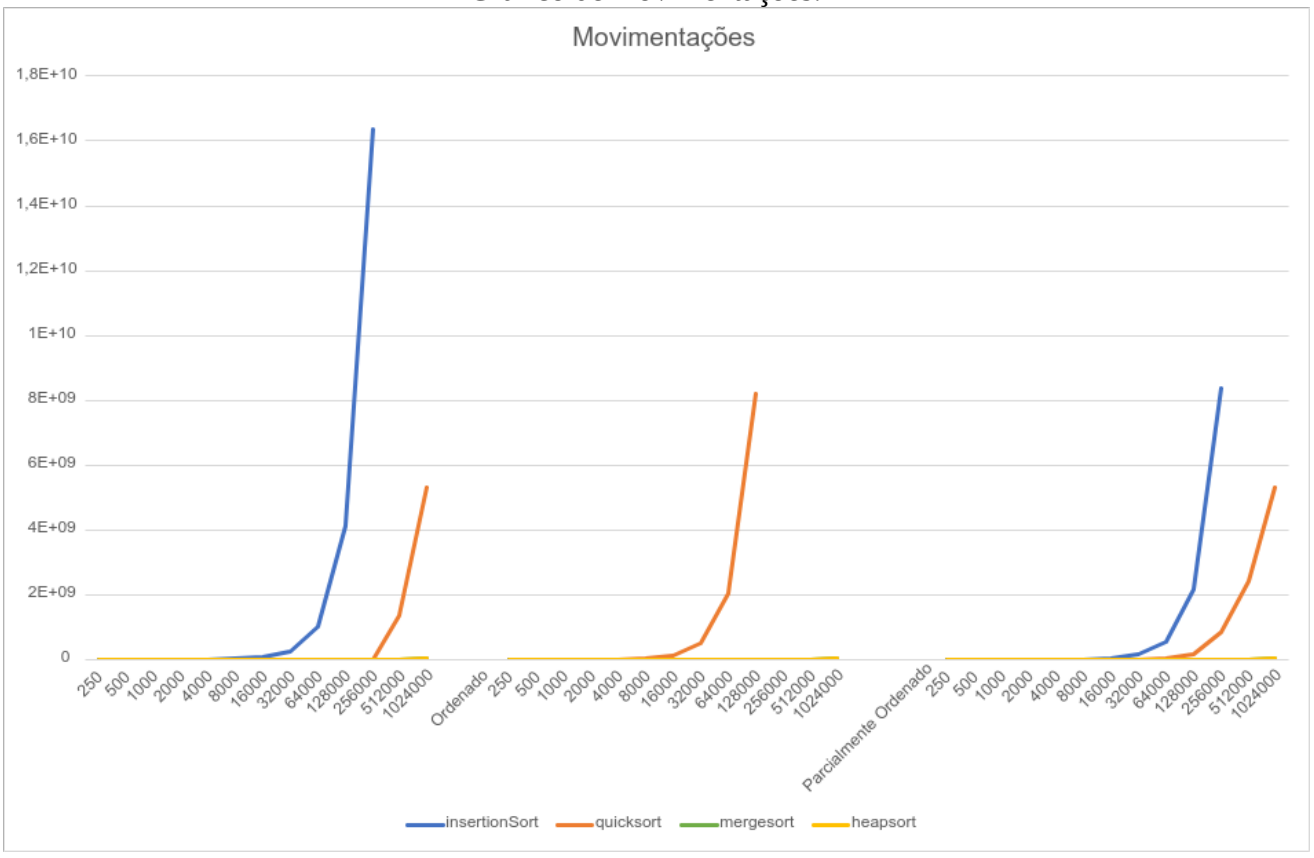
Movimentações (ordenado):

Tamanho	insertionSort	quicksort	mergesort	heapsort
250	249	31374	1011	249
500	499	125249	2272	499
1000	999	500499	5044	1917
2000	1999	2000999	11088	12263
4000	3999	8001999	24176	35414
8000	7999	32003999	52352	85554
16000	15999	128007999	112704	194007
32000	31999	512015999	241408	423437
64000	63999	2048031999	514816	907679
128000	127999	8192063999	1093632	1934023
256000	255999	segmentation fault	2315264	4127199
512000	511999		4886528	8694454
1024000	1023999		10285056	18210418

Movimentações (parcialmente ordenado):

Tamanho	insertionSort	quicksort	mergesort	heapsort
250	9743	845	1090	861
500	38288	2721	2428	1792
1000	145353	9821	5350	4479
2000	579374	37370	11677	15395
4000	2239158	140497	25327	39474
8000	8808341	573538	54519	91771
16000	34806482	2298313	117547	204349
32000	138786289	9107927	251084	445567
64000	552220569	36739328	534157	957387
128000	2173761626	145146962	1132302	2049240
256000	8368024736	849929365	2384797	4358244
512000	muito tempo	2399245409	8998159	9201244
1024000		5296768402	18836260	19270061

Gráfico de movimentações:



Tempo (aleatório):

Tempo	insertionSort	quicksort	mergesort	heapsort
250	0m 0,008s	0m 0,002s	0m 0,005s	0m 0,003s
500	0m 0,003s	0m 0,004s	0m 0,001s	0m 0,003s
1000	0m 0,003s	0m 0,001s	0m 0,002s	0m 0,004s
2000	0m 0,011s	0m 0,002s	0m 0,004s	0m 0,005s
4000	0m 0,031s	0m 0,002s	0m 0,006s	0m 0,008s
8000	0m 0,131s	0m 0,005s	0m 0,014s	0m 0,013s
16000	0m 0,521s	0m 0,008s	0m 0,039s	0m 0,022s
32000	0m 2,362s	0m 0,015s	0m 0,047s	0m 0,021s
64000	0m 13,610s	0m 0,035s	0m 0,101s	0m 0,043s
128000	2m 19,461s	0m 0,077s	0m 0,185s	0m 0,091s
256000	15m 59,692s	0m 0,278s	0m 0,537s	0m 0,233s
512000	muito tempo	0m 12,932s	0m 0,652s	0m 0,587s
1024000		1m 13,318s	0m 1,382s	0m 1,408s

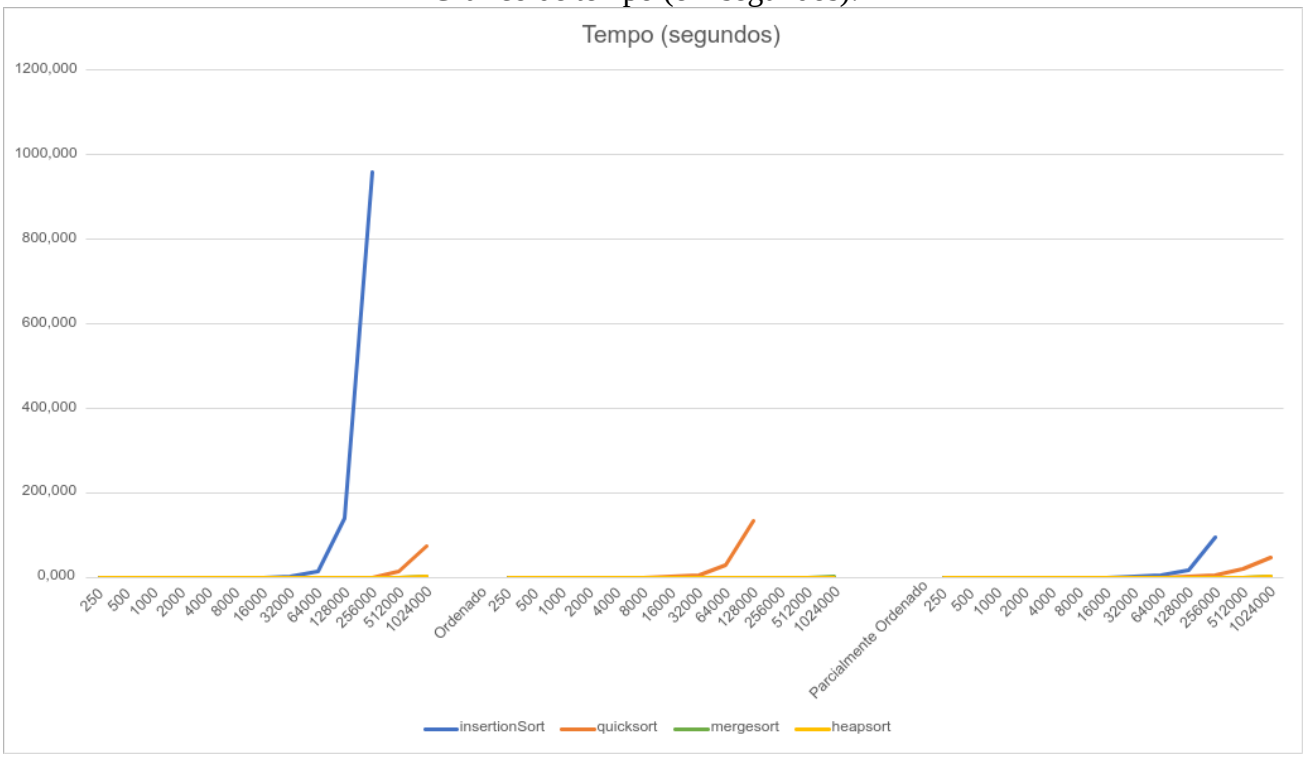
Tempo (ordenado):

Tempo	insertionSort	quicksort	mergesort	heapsort
250	0m0,001s	0m0,002s	0m0,004s	0m0,001s
500	0m0,002s	0m0,006s	0m0,002s	0m0,001s
1000	0m0,002s	0m0,006s	0m0,005s	0m0,001s
2000	0m0,002s	0m0,017s	0m0,003s	0m0,001s
4000	0m0,003s	0m0,082s	0m0,009s	0m0,002s
8000	0m0,004s	0m0,271s	0m0,014s	0m0,004s
16000	0m0,004s	0m 0,962s	0m0,023s	0m0,007s
32000	0m0,007s	0m 5,071s	0m0,052s	0m0,016s
64000	0m0,012s	0m 30,162s	0m0,098s	0m0,031s
128000	0m0,020s	2m 14,218s	0m0,191s	0m0,067s
256000	0m0,032s	segmentation fault	0m0,284s	0m0,141s
512000	0m0,060s		0m 0,568s	0m 0,293s
1024000	0m 0,011s		0m 1,098s	0m 0,618s

Tempo (parcialmente ordenado):

Tempo	insertionSort	quicksort	mergesort	heapsort
250	0m 0,001s	0m 0,001s	0m 0,002s	0m 0,001s
500	0m 0,001s	0m 0,002s	0m 0,001s	0m 0,001s
1000	0m 0,002s	0m 0,002s	0m 0,002s	0m 0,001s
2000	0m 0,005s	0m 0,004s	0m 0,002s	0m 0,002s
4000	0m 0,015s	0m 0,013s	0m 0,004s	0m 0,002s
8000	0m 0,058s	0m 0,047s	0m 0,009s	0m 0,004s
16000	0m 0,231s	0m 0,183s	0m 0,016s	0m 0,008s
32000	0m 1,081s	0m 0,698s	0m 0,033s	0m 0,018s
64000	0m 4,331s	0m 2,867s	0m 0,065s	0m 0,038s
128000	0m 17,483s	0m 13,381s	0m 0,136s	0m 0,082s
256000	1m 34,949s	1m 4,104s	0m 0,288s	0m 0,202s
512000	muito tempo	0m 18,958s	0m 0,612s	0m 0,467s
1024000		0m 47,819s	0m 1,278s	0m 0,965s

Gráfico de tempo (em segundos):



Conclusão:

Analisando os gráficos e tabelas produzidas, é possível afirmar que são compatíveis com as complexidades calculadas para os algoritmos:

Para o insertionSort, verifica-se a complexidade de $O(n^2)$ para entradas aleatórias, $O(n)$ para entradas ordenadas, e um meio termo dessas complexidades para entradas parcialmente ordenadas.

Para o quicksort, também é possível verificar a complexidade $O(n \log n)$ em casos gerais e $O(n^2)$ para entradas ordenadas, além de um meio termo entre essas complexidades para entradas parcialmente ordenadas. Além disso, vemos que para a entrada de 512000 palavras, a pilha de recursão estoura.

Para o mergesort e o heapsort, ambos mantém a complexidade $O(n \log n)$ para todas as entradas verificadas. Não é possível ver muito bem nos gráficos, mas pelas tabelas os tempos desses sorts são bem parecidos, com o heap sendo um pouco mais rápido. Em questão de comparações, os algoritmos começam com um número semelhante, porém com tamanhos maiores as comparações do heap crescem mais rapidamente. Para as movimentações, eles também são bem parecidos, com o heap tendo menor número em entradas ordenadas/parcialmente ordenadas e pequenas.