Universidade de Brasília

Departamento de Ciência da Computação

Disciplina: Projeto e Análise de Algoritmos 1/2016

Código da Disciplina: 117536

Douglas Shiro Yokoyama

13/0024902

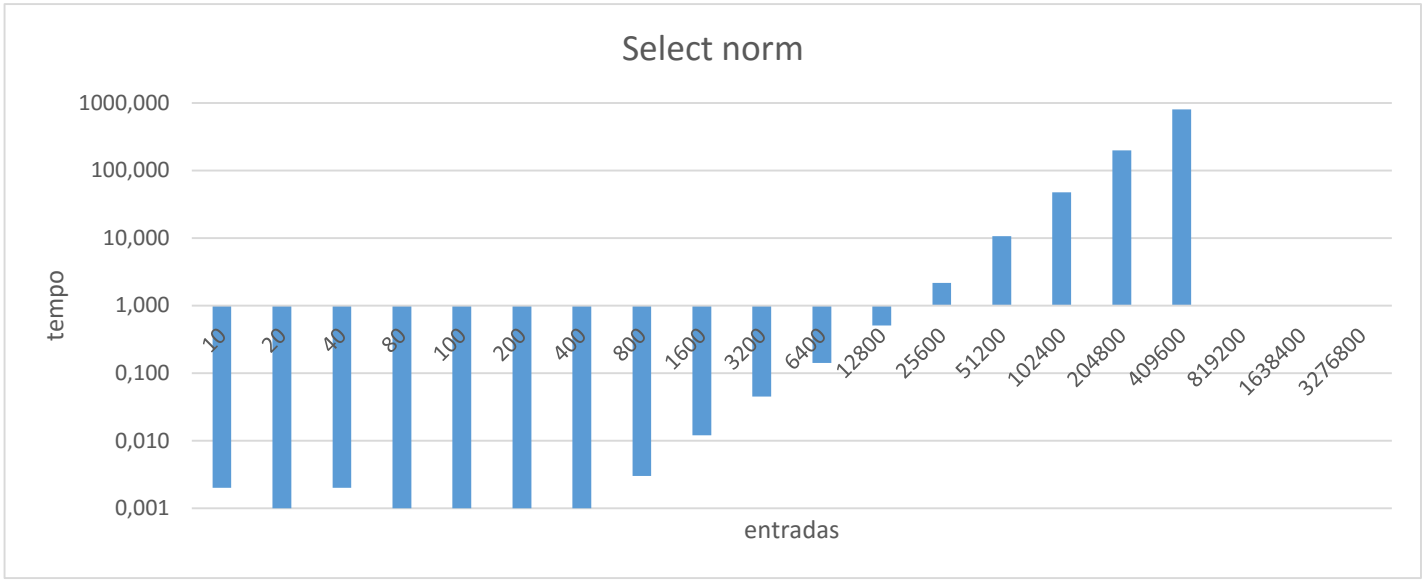
TRABALHO 2: Algoritmos de ordenação

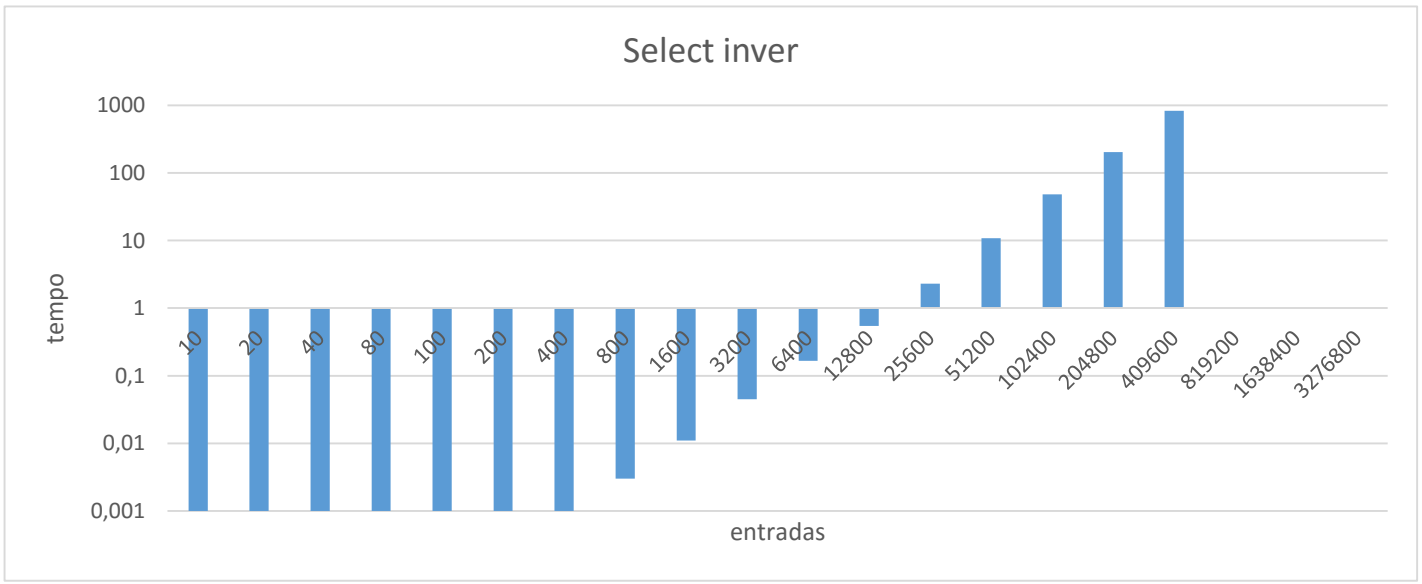
Brasília

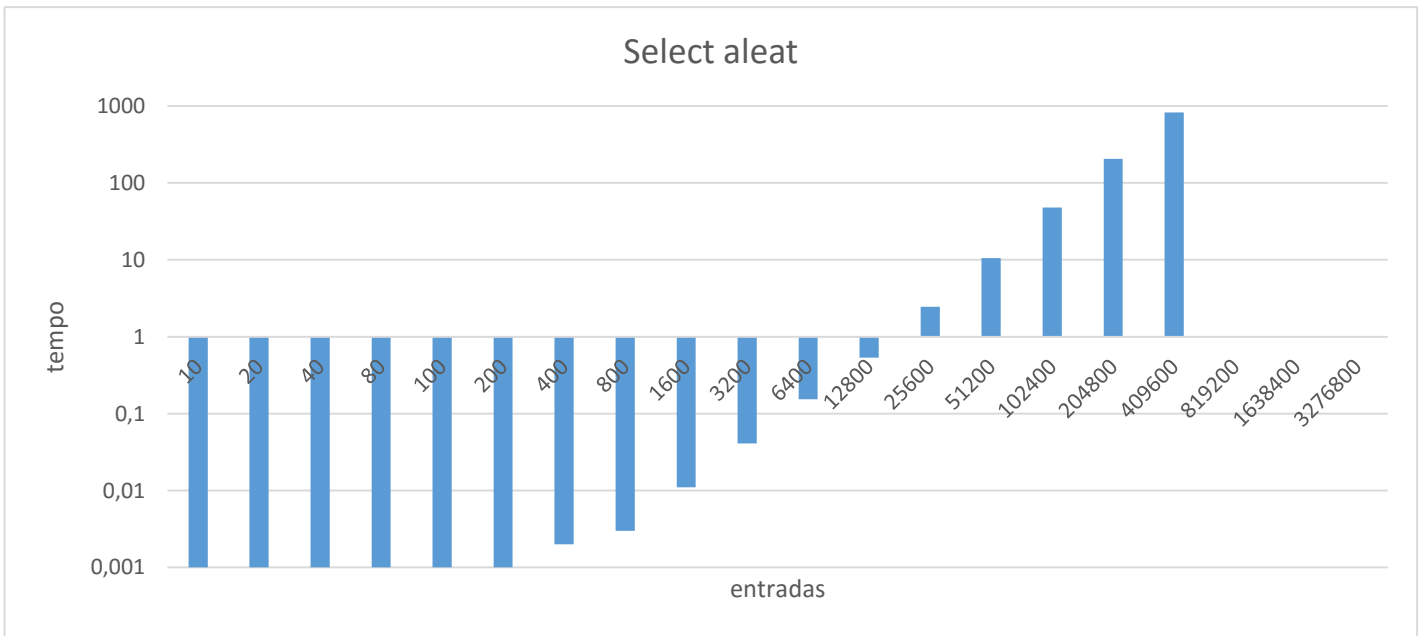
ABRIL/2016

2. Algoritmo Selection sort

Complexidade: melhor: $O(n^2)$, medio: $O(n^2)$, pior: $O(n^2)$

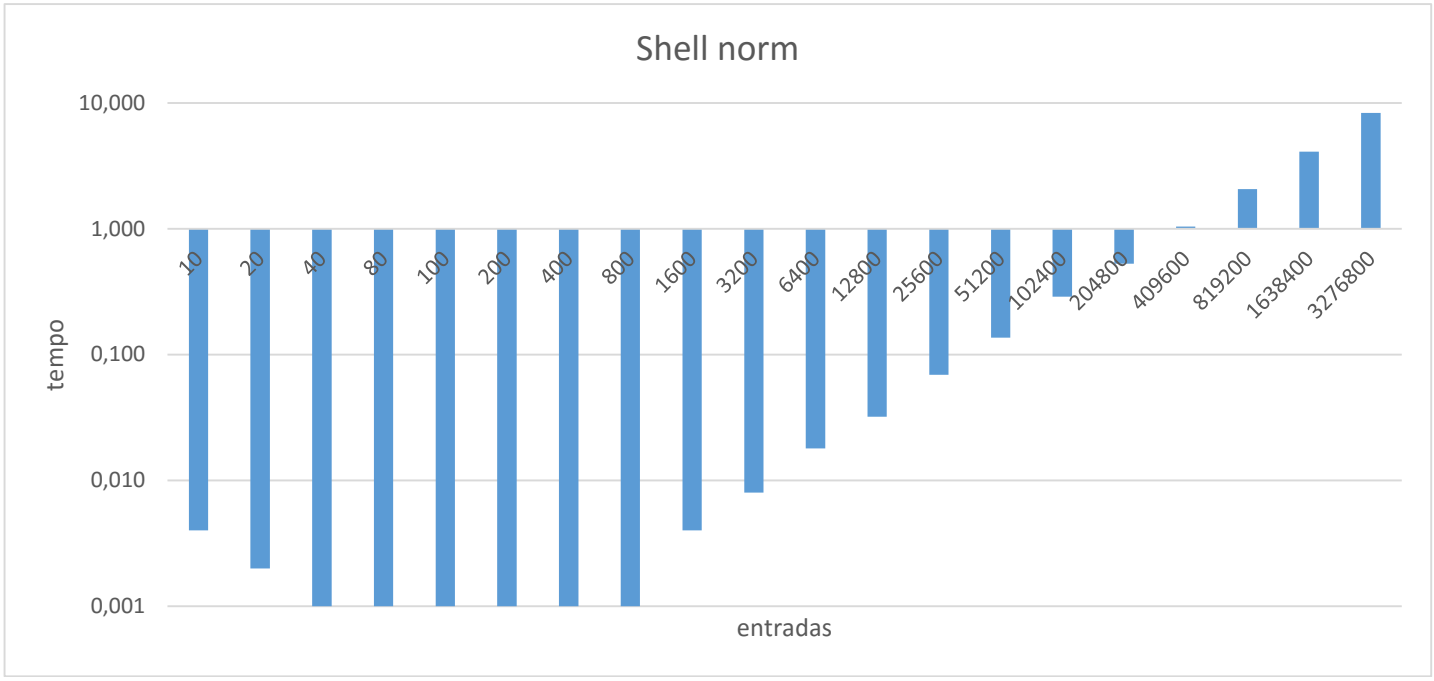


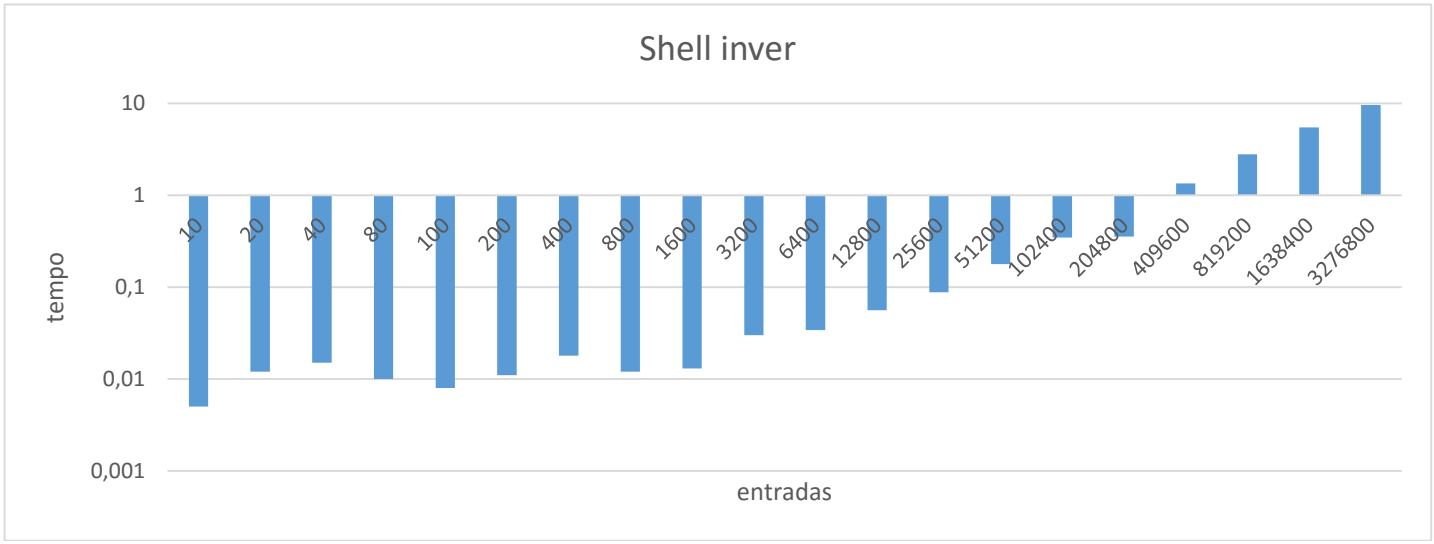


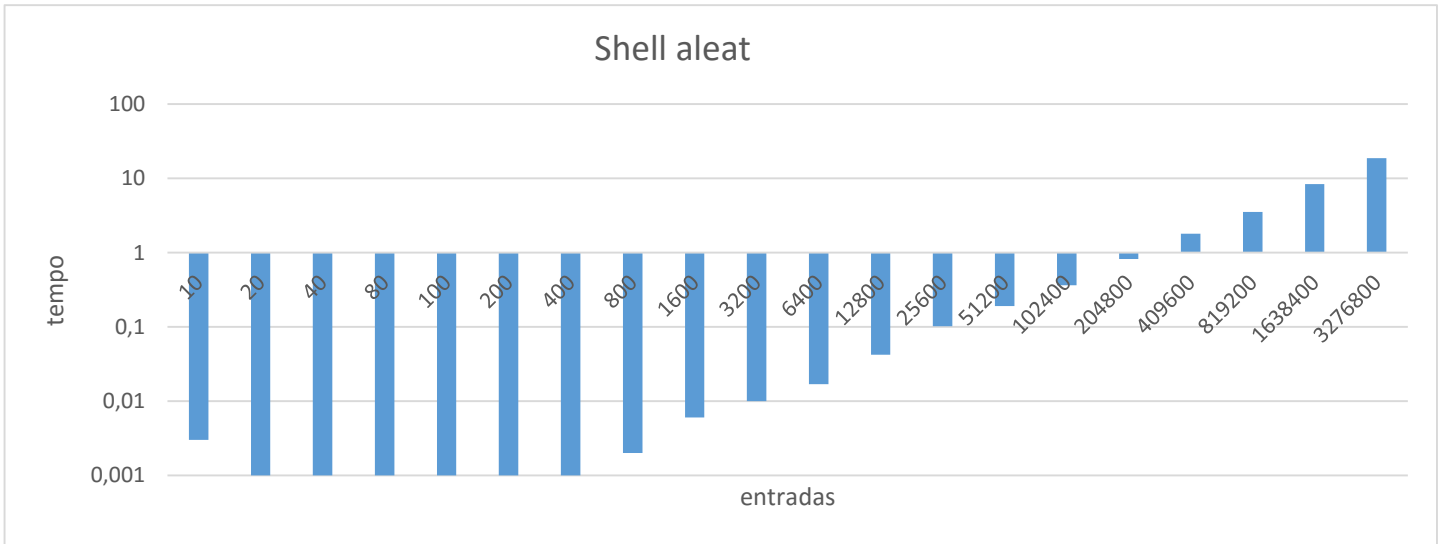


3. Algoritmo Shell sort

Complexidade: melhor: O(n), pior: O(nlog²n)

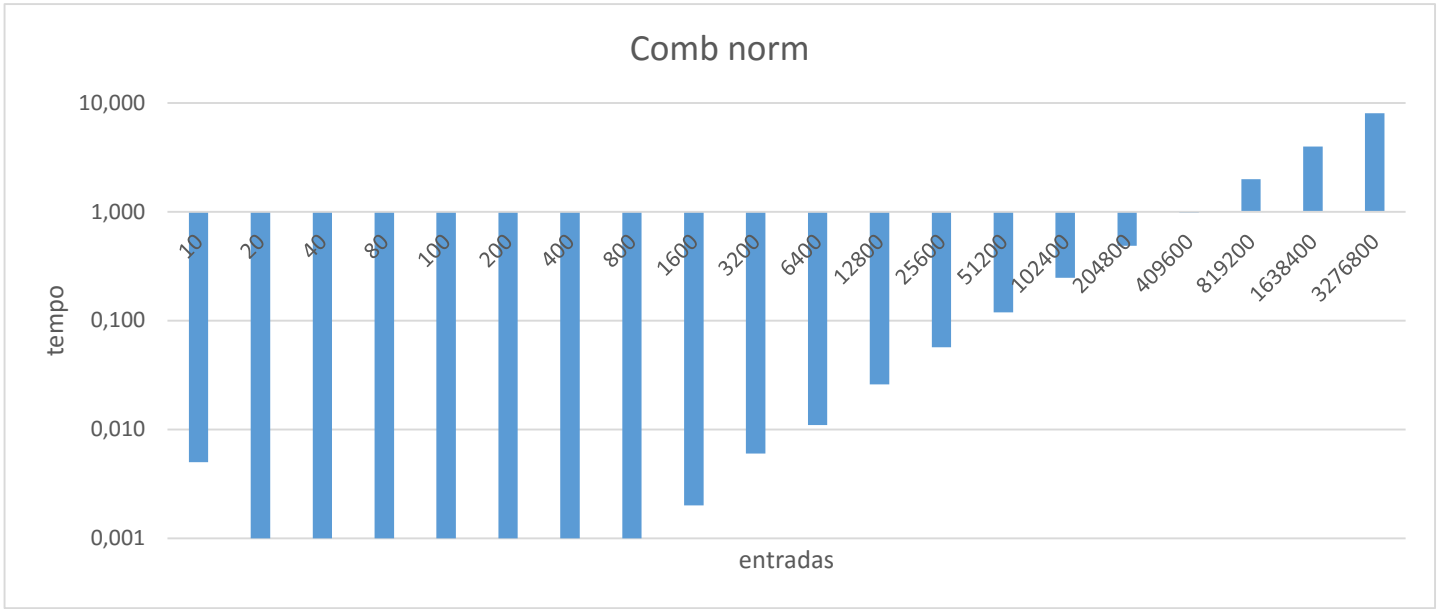


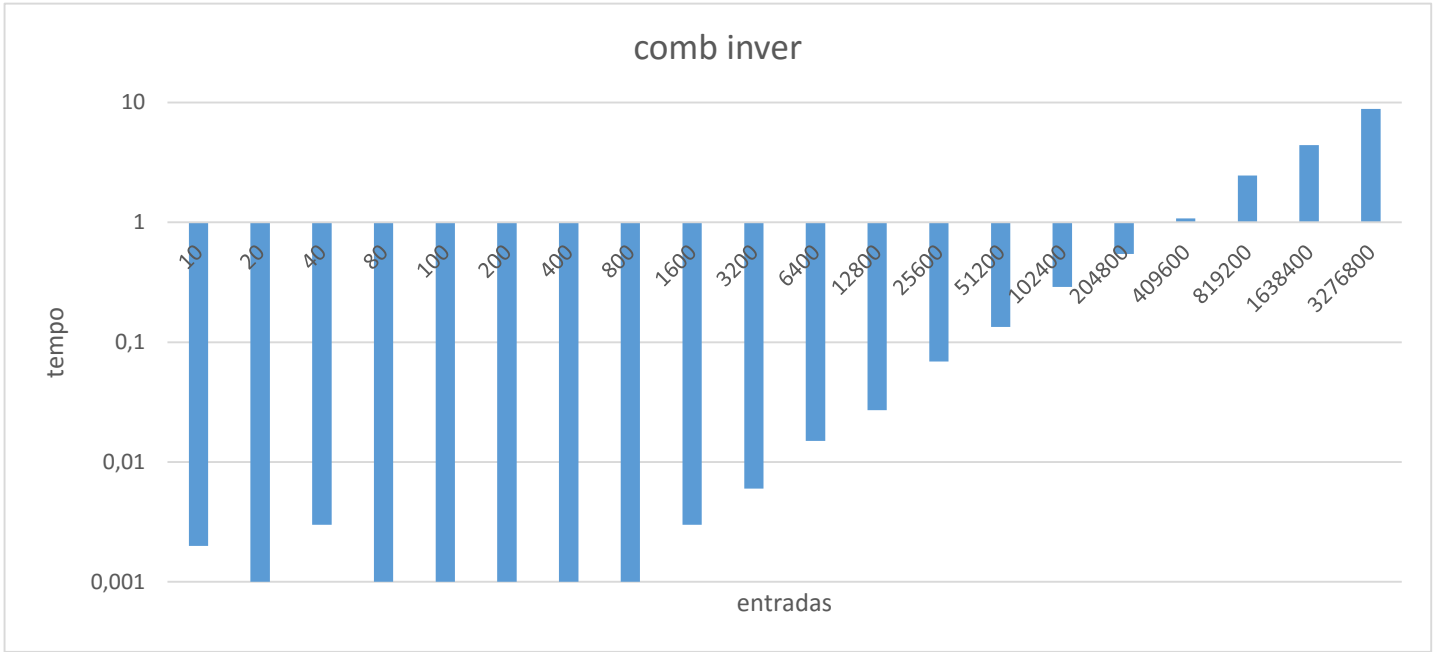


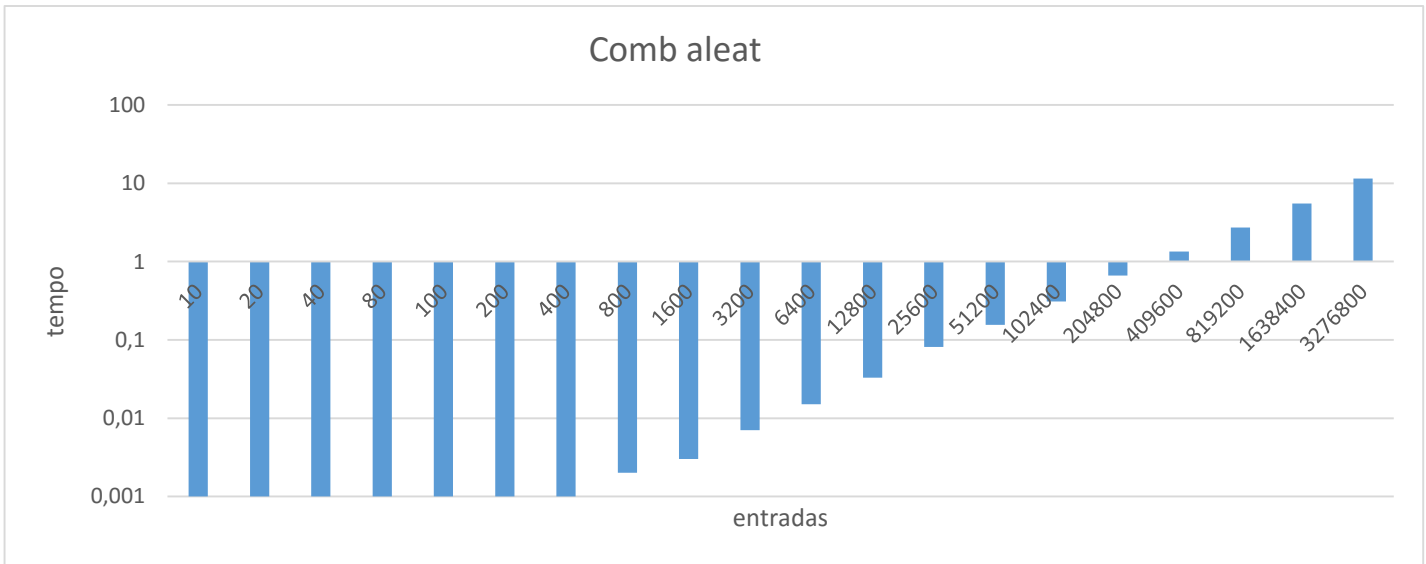


4. Algoritmo Comb sort

Complexidade: pior: $O(n^2)$

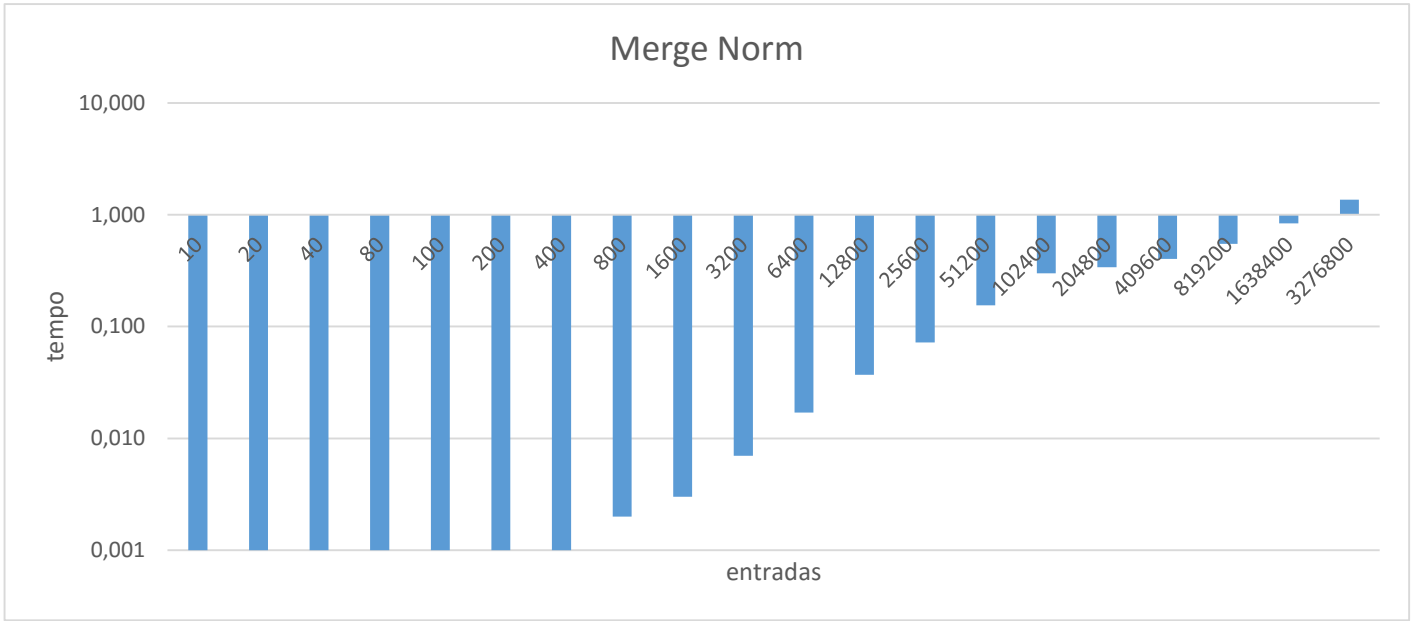


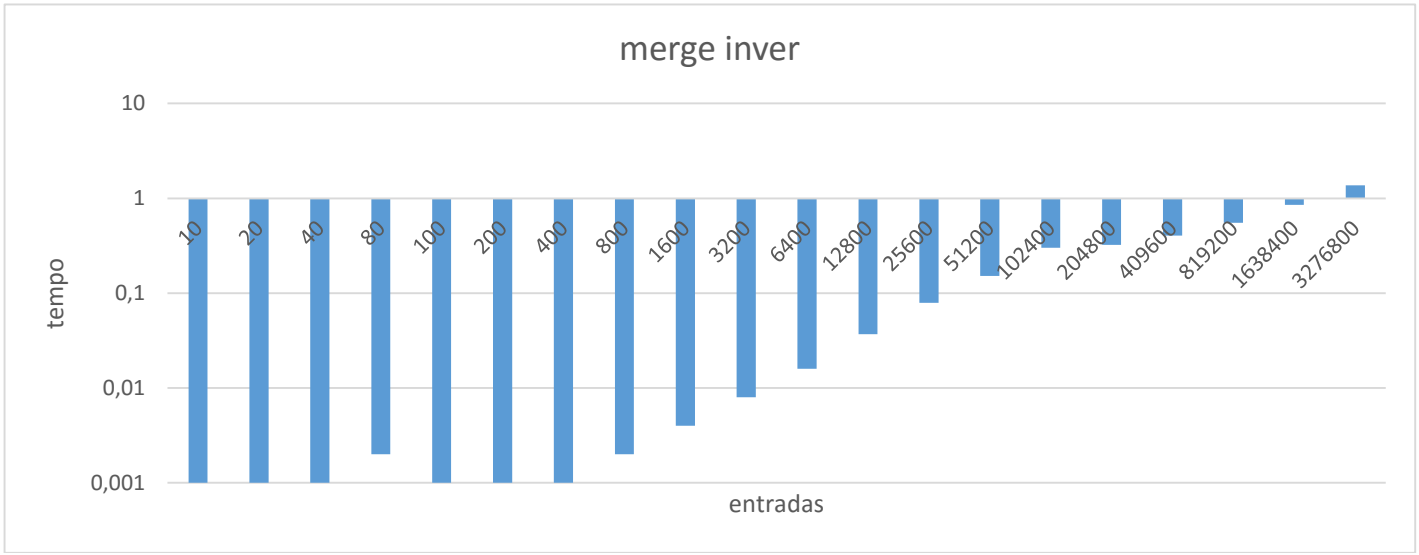


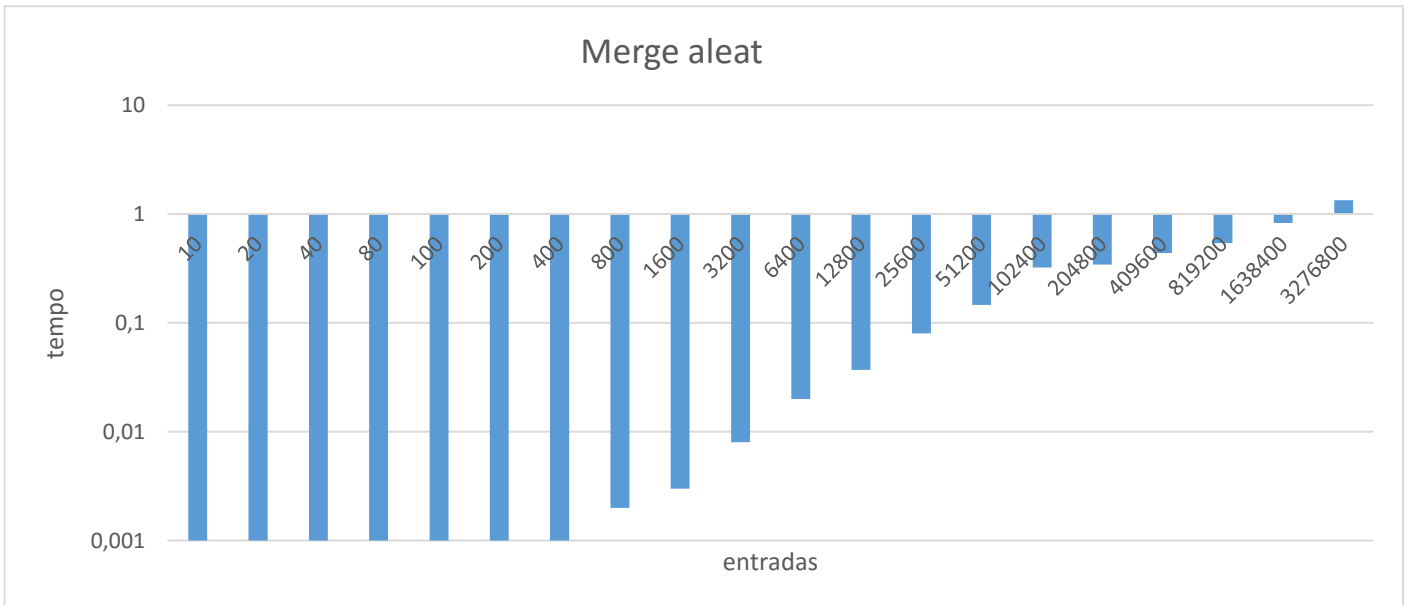


5. Algoritmo Merge sort

Complexidade:melhor: Θ(nlogn), medio: Θ(nlogn), pior: Θ(nlog2n)

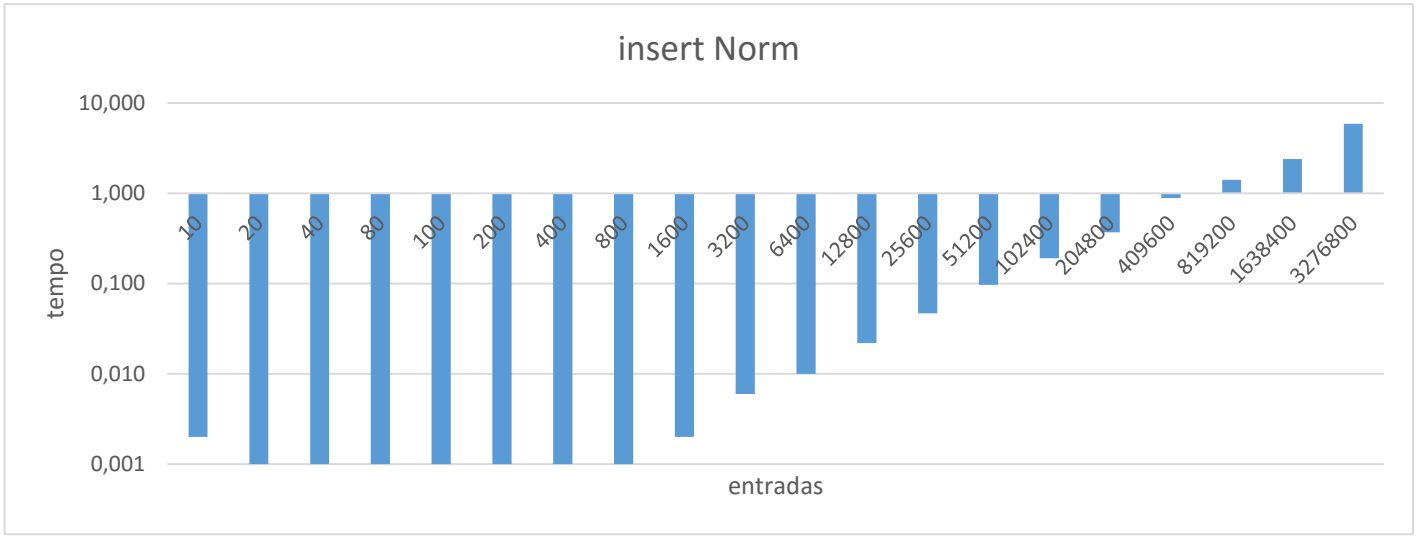


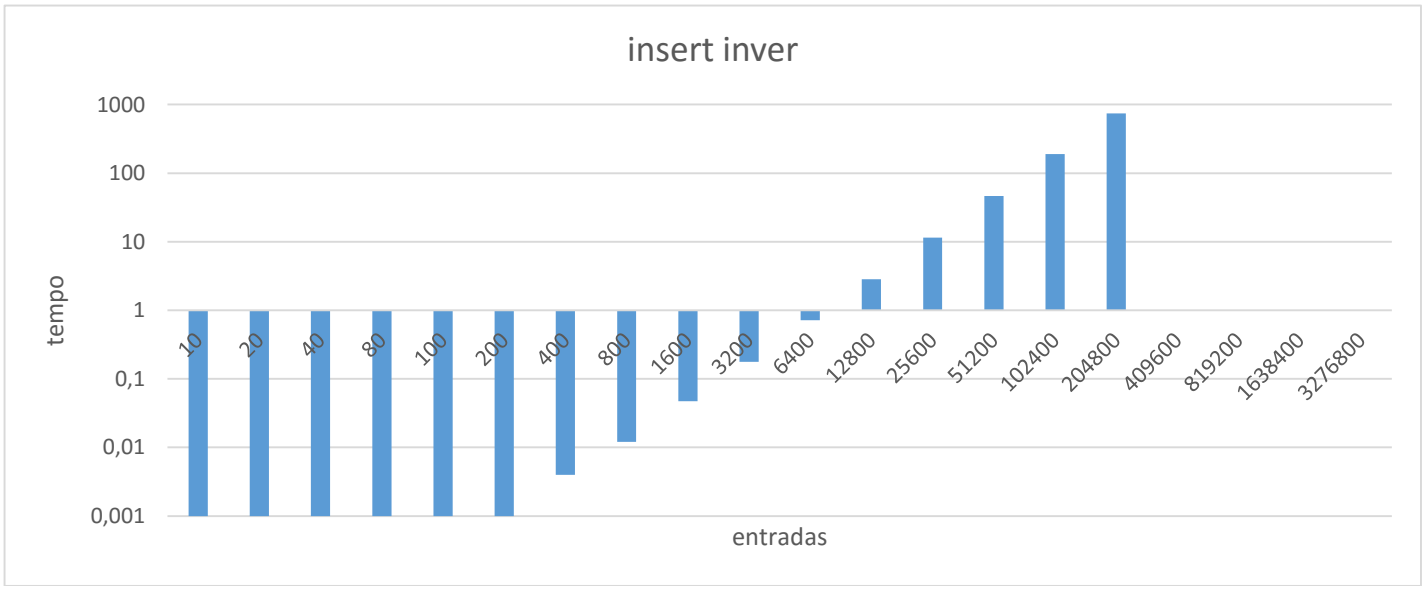


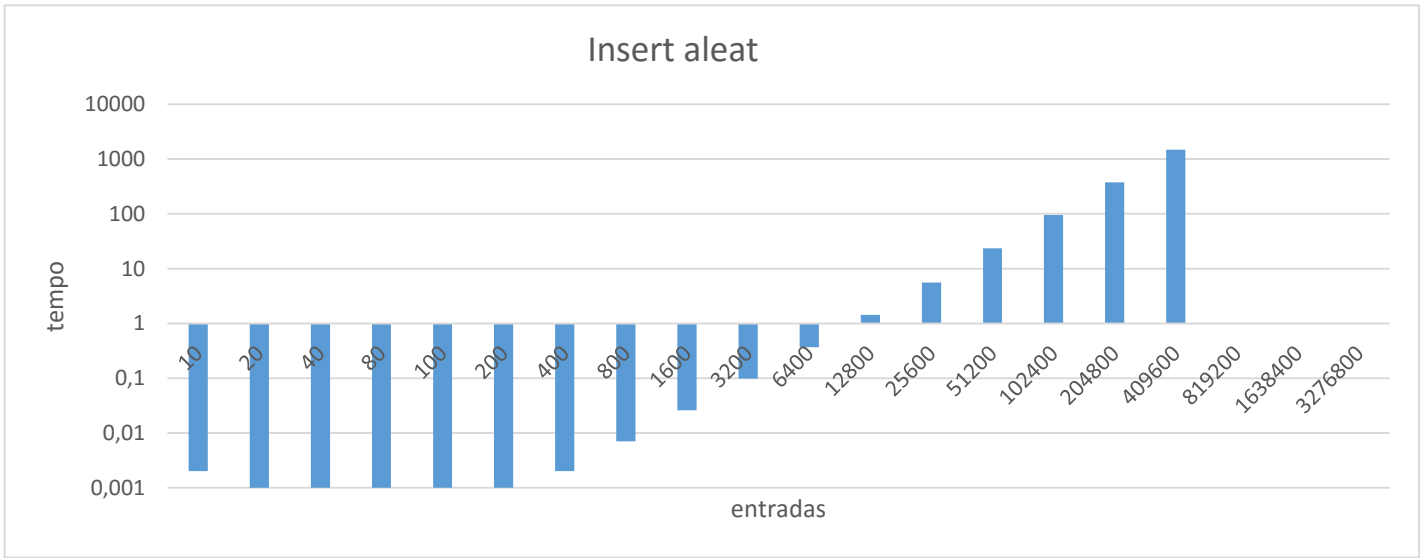


6. Algoritmo Insertion sort

Complexidade: melhor: O(n), medio: O(n²), pior: O(n²)

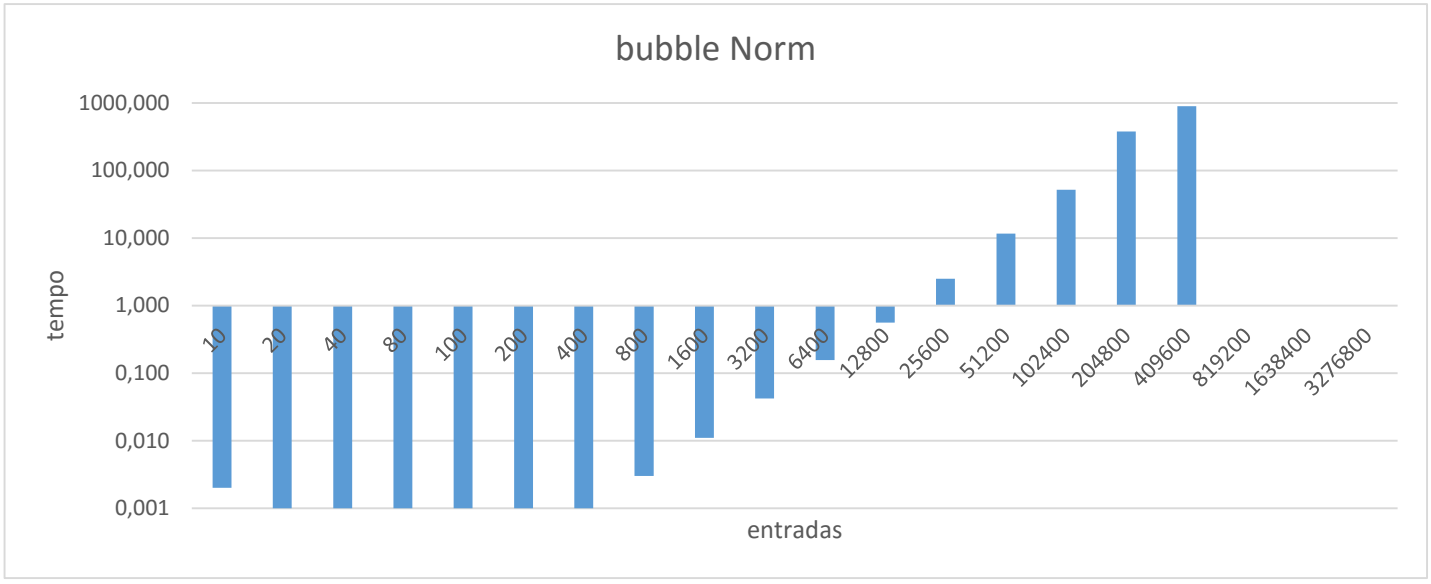


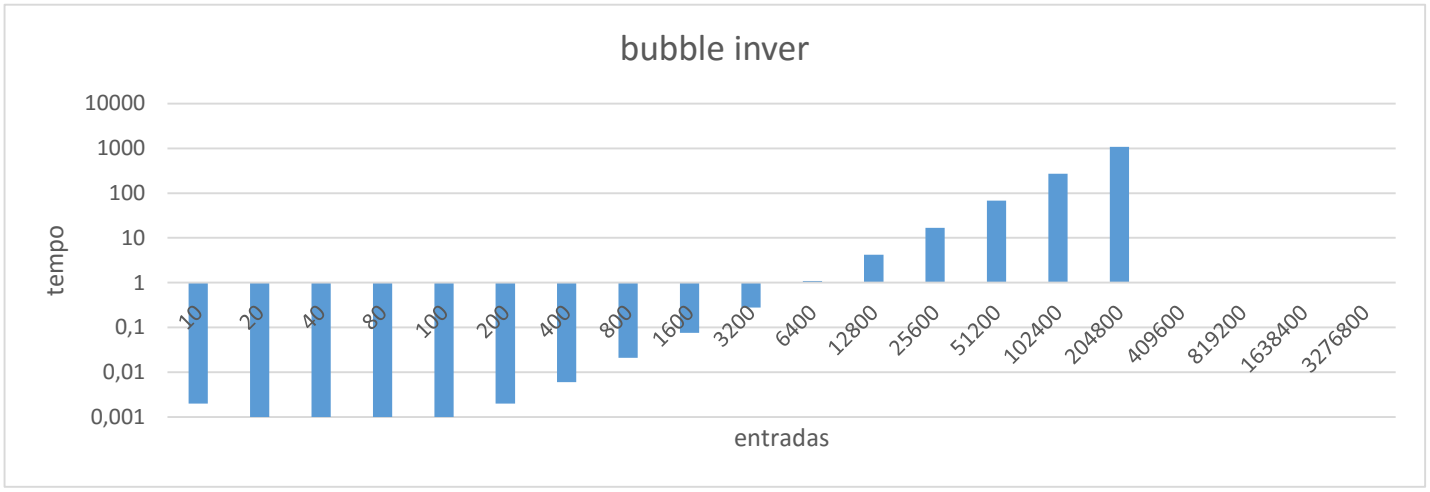


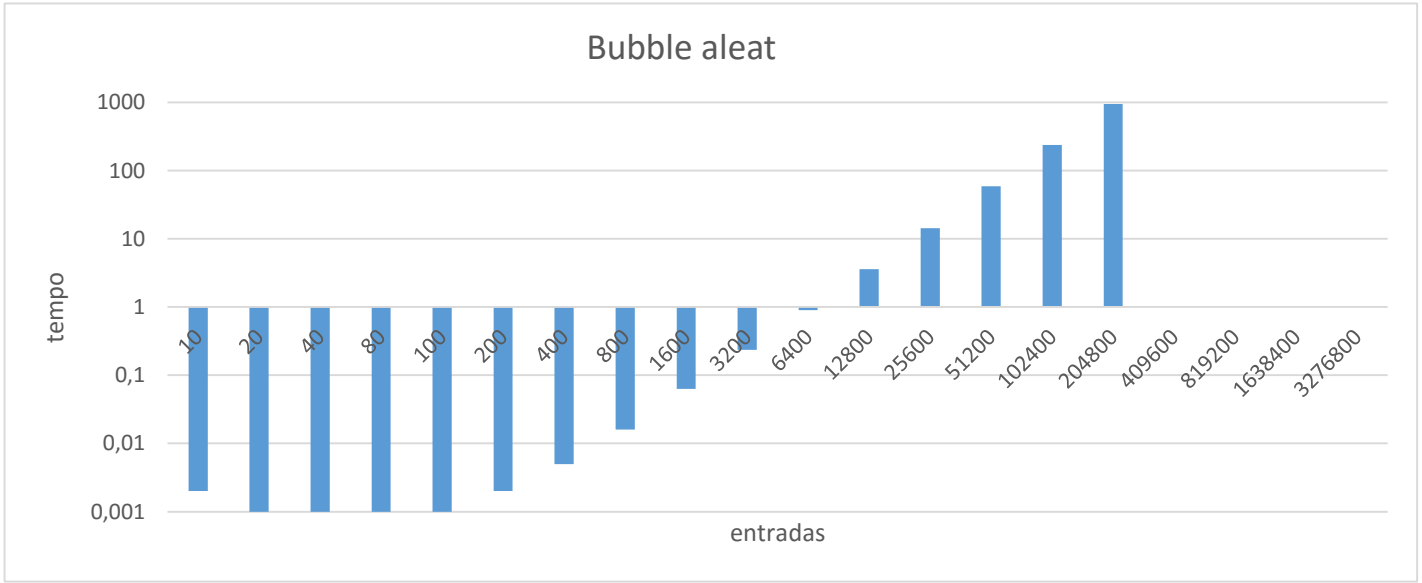


7. Algoritmo Bubble sort

Complexidade: melhor: O(n), medio: O(n²), pior: O(n²)



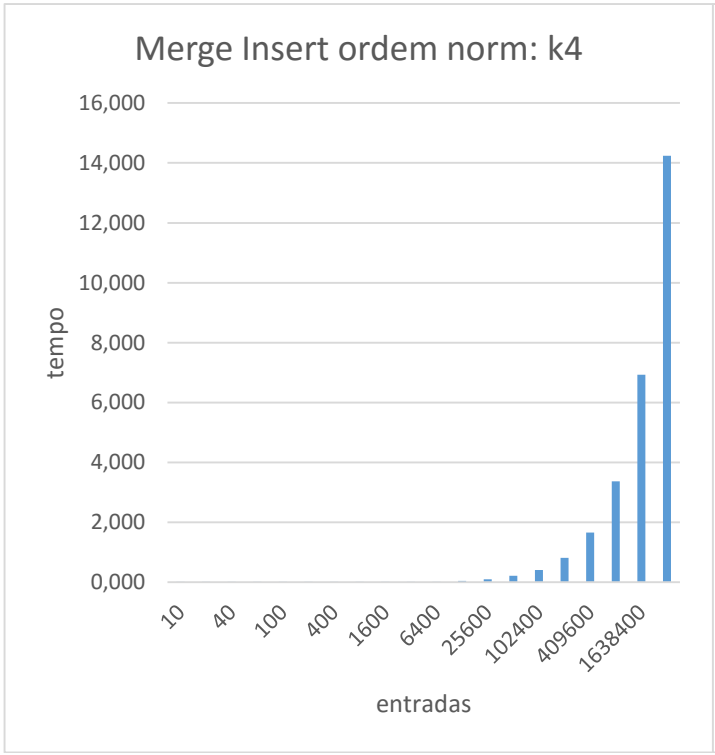


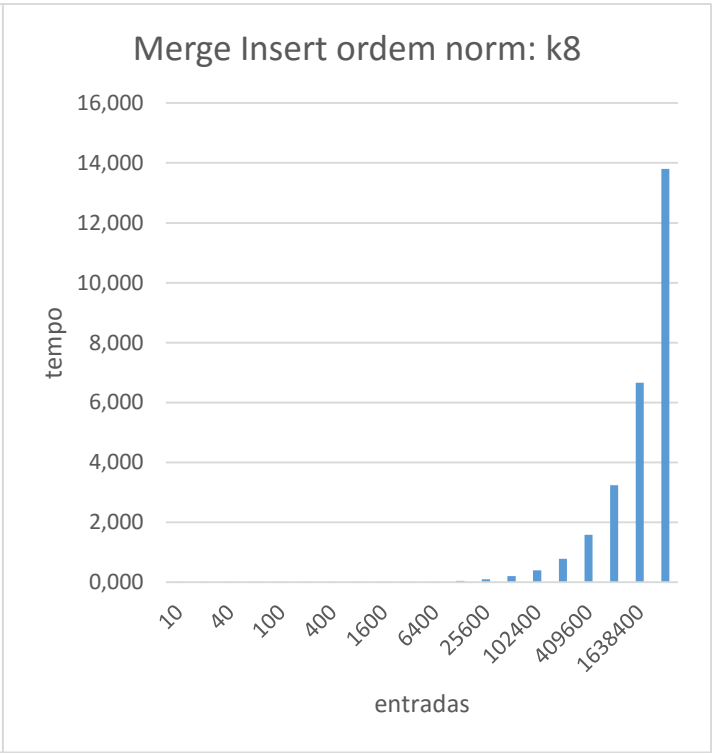


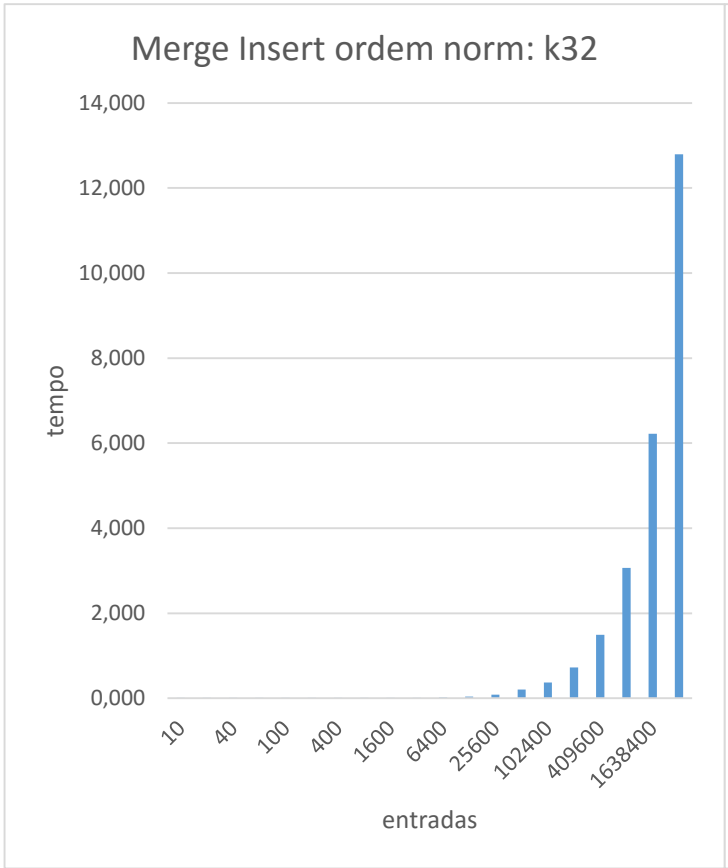
8. Algoritmo Merge_Insert sort

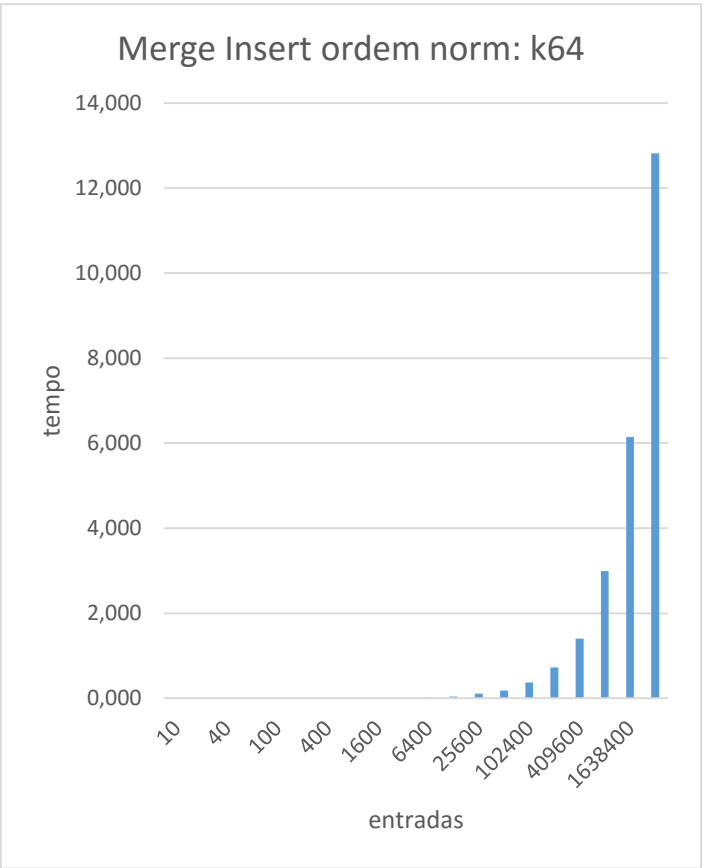
Complexidade:

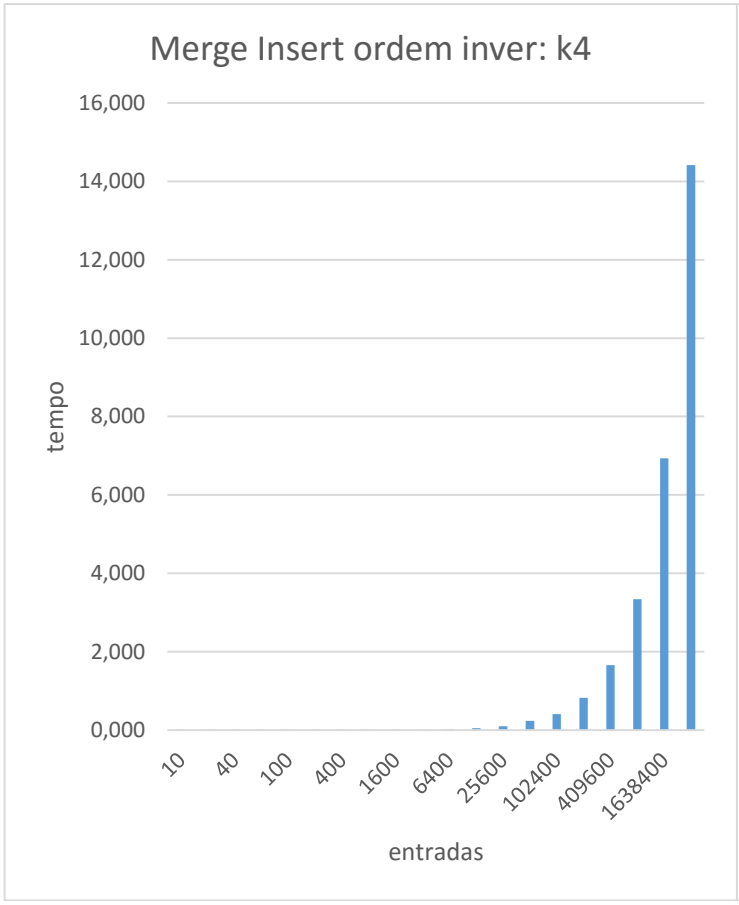
Gráficos comparativos para as ordens e os parâmetros k.

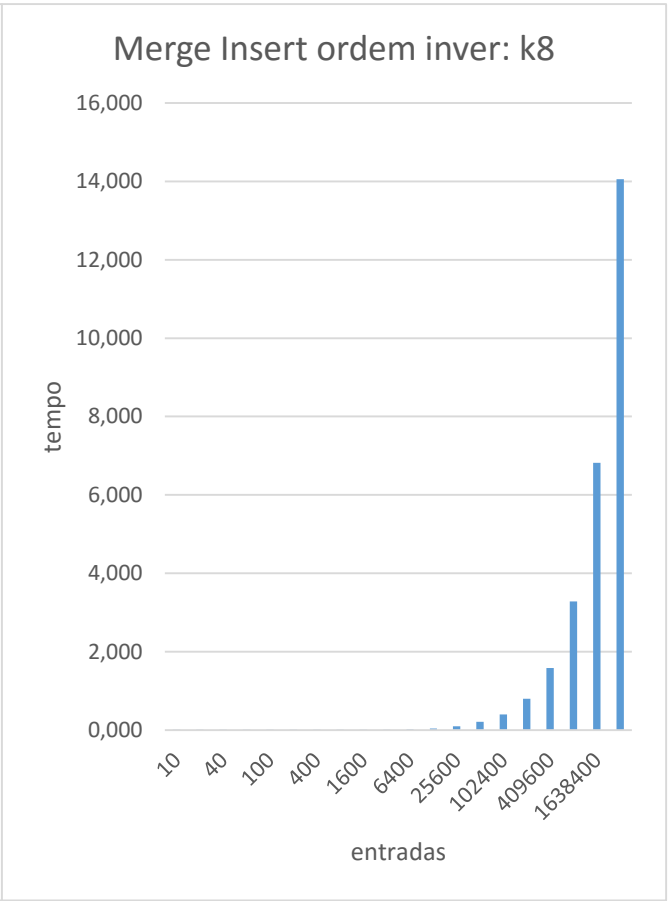


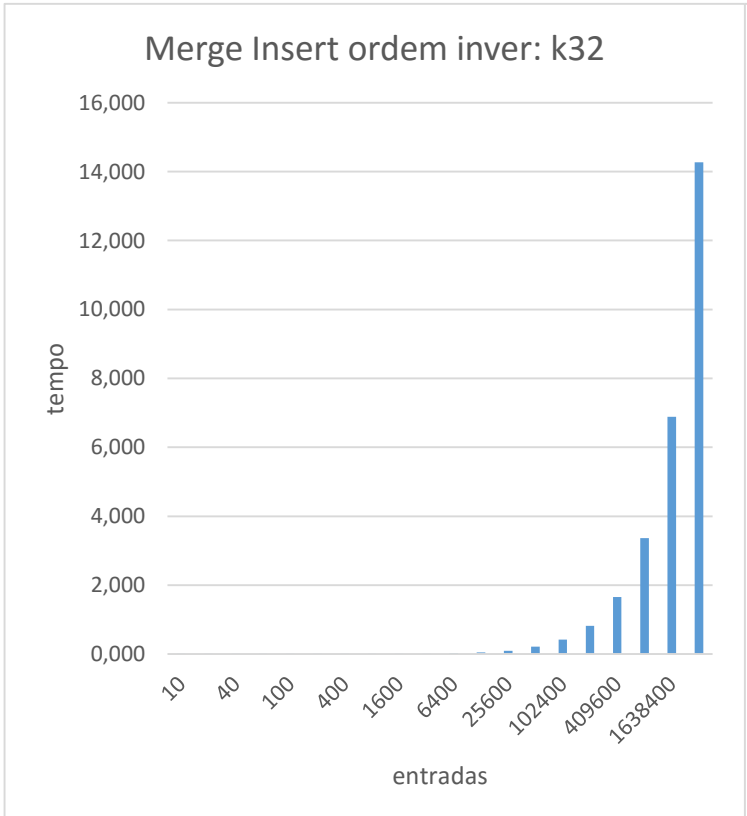


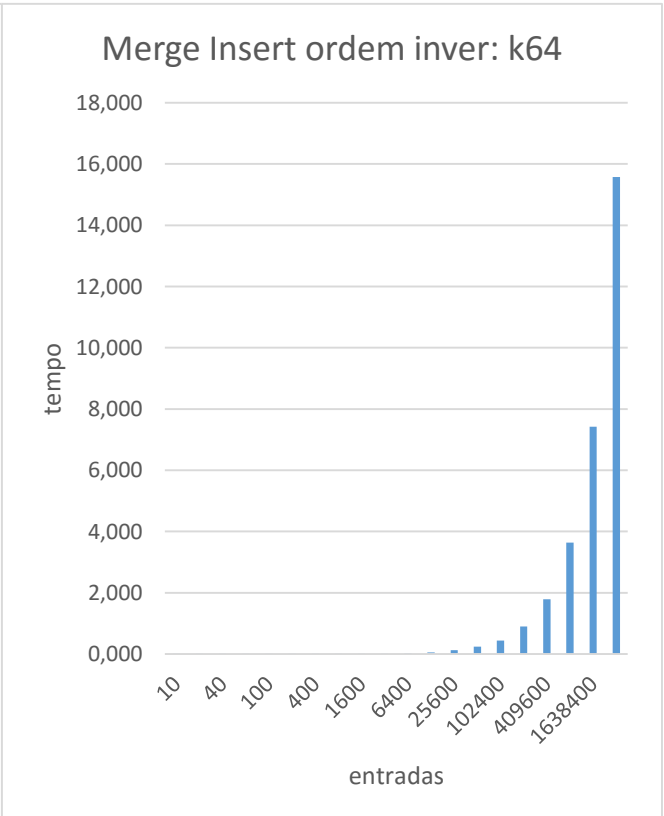


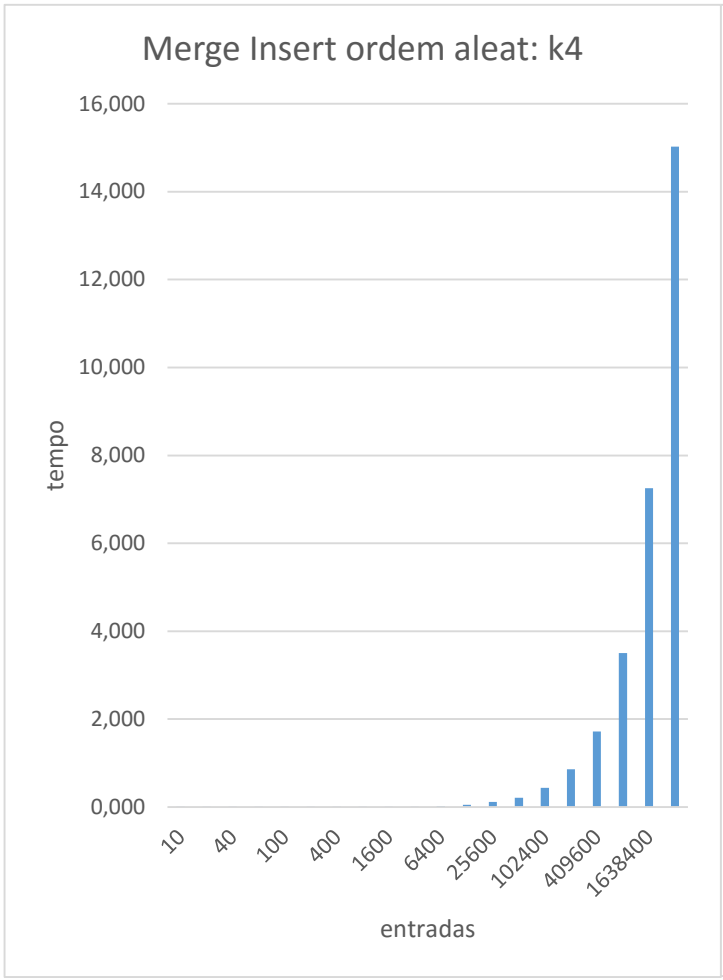


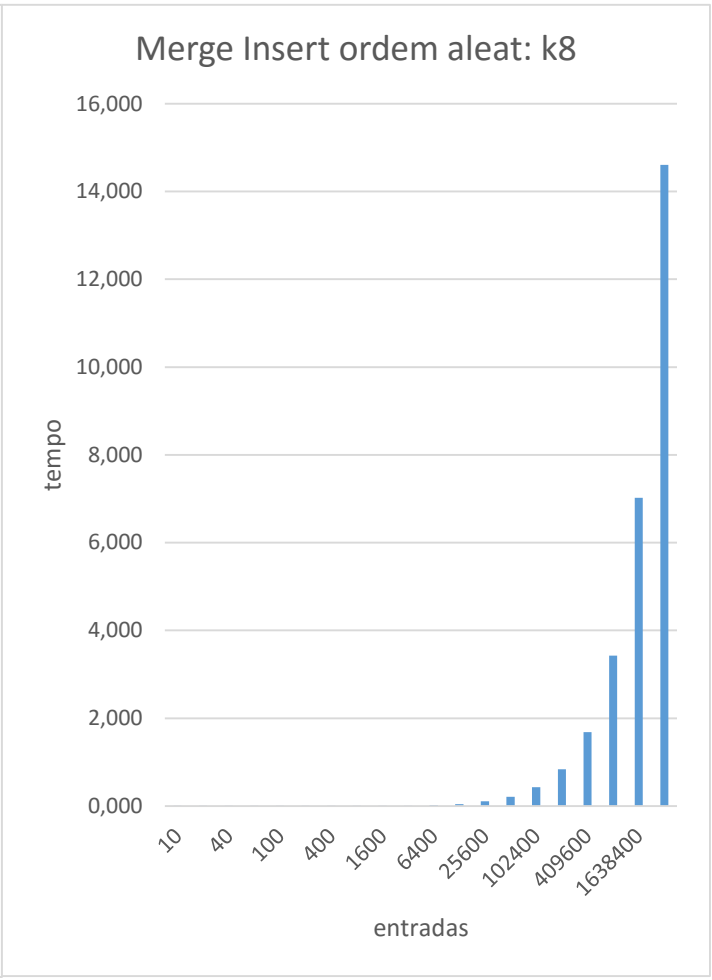

Merge Insert ordem inver: k4
16,000
14,000
12,000
10,000
8,000
6,000
4,000
2,000
0,000
tempo
10
40
100
400
1600
6400
25600
102400
409600
1638400
entradas

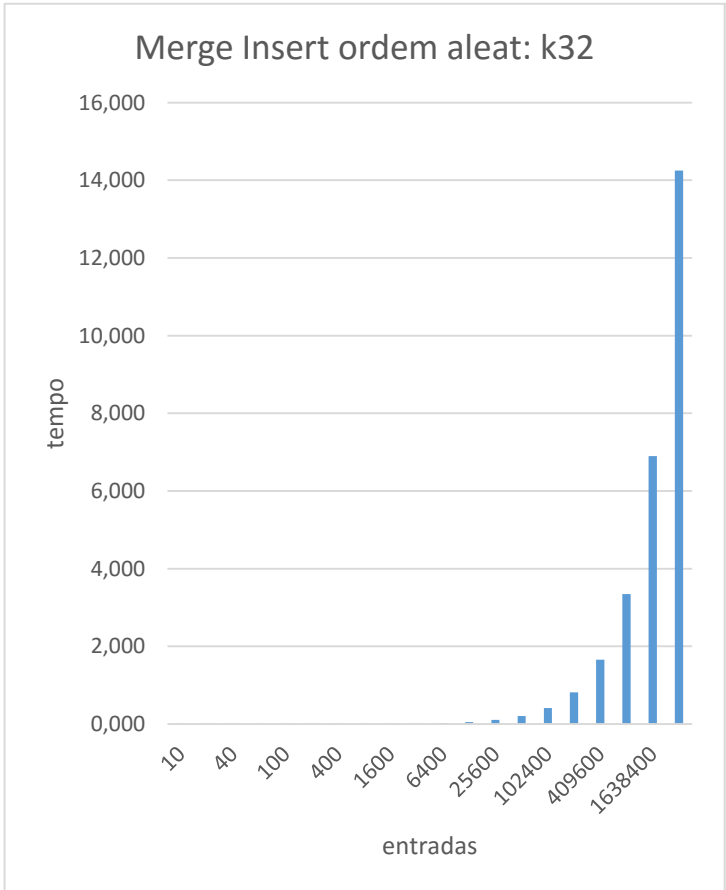

Merge Insert ordem inver: k8
16,000
14,000
12,000
10,000
8,000
6,000
4,000
2,000
0,000
tempo
10
40
100
400
1600
6400
25600
102400
409600
1638400
entradas

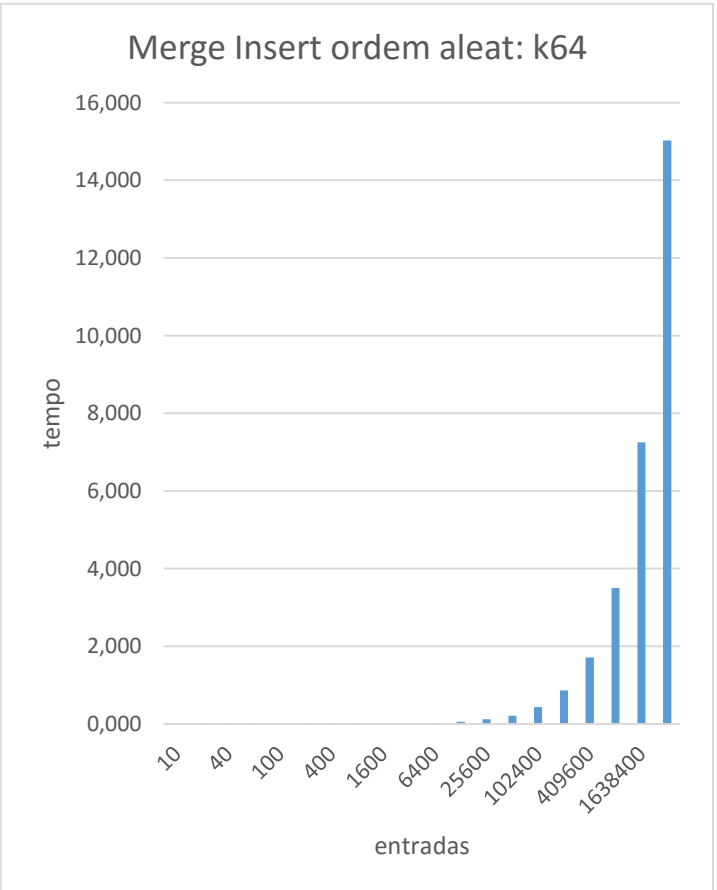

Merge Insert ordem inver: k32
16,000
14,000
12,000
10,000
8,000
6,000
4,000
2,000
0,000
tempo
10
40
100
400
1600
6400
25600
102400
409600
1638400
entradas


Merge Insert ordem inver: k64
18,000
16,000
14,000
12,000
10,000
8,000
6,000
4,000
2,000
0,000
tempo
10
40
100
400
1600
6400
25600
102400
409600
1638400
entradas


Merge Insert ordem aleat: k4
16,000
14,000
12,000
10,000
8,000
6,000
4,000
2,000
0,000
tempo
10
40
100
400
1600
6400
25600
102400
409600
1638400
entradas


Merge Insert ordem aleat: k8
16,000
14,000
12,000
10,000
8,000
6,000
4,000
2,000
0,000
tempo
10
40
100
400
1600
6400
25600
102400
409600
1638400
entradas


Merge Insert ordem aleat: k32
16,000
14,000
12,000
10,000
8,000
6,000
4,000
2,000
0,000
tempo
10
40
100
400
1600
6400
25600
102400
409600
1638400
entradas


Merge Insert ordem aleat: k64
16,000
14,000
12,000
10,000
8,000
6,000
4,000
2,000
0,000
tempo
10
40
100
400
1600
6400
25600
102400
409600
1638400
entradas

9. Análise comparativa de todos dos algoritmos de ordenação. Cada um deles deve ter uma média de 10 execuções.

Para o caso do merge_insert, todos os casos devem ser rodados para k=4, k=8, k=32 e k=64.

10.

a) Tabela e gráfico comparativa todos algoritmos ordem normal

Entradas	Select	Shell	Comb	Merge	Insert	Bubble	MI:k4	MI:k8	MI:k32	MI:k64
10	0.002	0.004	0.005	0.001	0.002	0.002	0.001	0.002	0.001	0,001
20	0.001	0.002	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0,001
40	0.002	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0,001
80	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0,001
100	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0,001
200	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0,001
400	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0,001
800	0.003	0.001	0.001	0.002	0.001	0.003	0.002	0.002	0.002	0,002
1600	0.012	0.004	0.002	0.003	0.002	0.011	0.004	0.004	0.003	0,003
3200	0.045	0.008	0.006	0.007	0.006	0.042	0.009	0.009	0.008	0,007
6400	0.141	0.018	0.011	0.017	0.010	0.156	0.020	0.020	0.018	0,018
12800	0.510	0.032	0.026	0.037	0.022	0.562	0.048	0.046	0.039	0,043
25600	2.158	0.069	0.057	0.072	0.047	2.490	0.099	0.102	0.086	0,105
51200	10.663	0.136	0.119	0.155	0.097	11.591	0.216	0.204	0.207	0,183
102400	47.663	0.289	0.247	0.300	0.191	51.911	0.414	0.396	0.370	0,370
204800	198.06	0.529	0.486	0.339	0.368	37.690	0.820	0.782	0.722	0,727
409600	801.22	1.038	0.978	0.404	0.882	896.65	1.662	1.586	1.495	1,406
819200	--------	2.067	1.987	0.549	1.410	--------	3.373	3.243	3.069	2,989
1638400	--------	4.102	3.977	0.836	2.396	--------	6.935	6.667	6.225	6,149
3276800	--------	8.301	8.053	1.359	5.879	--------	14.238	13.802	12.797	12,819

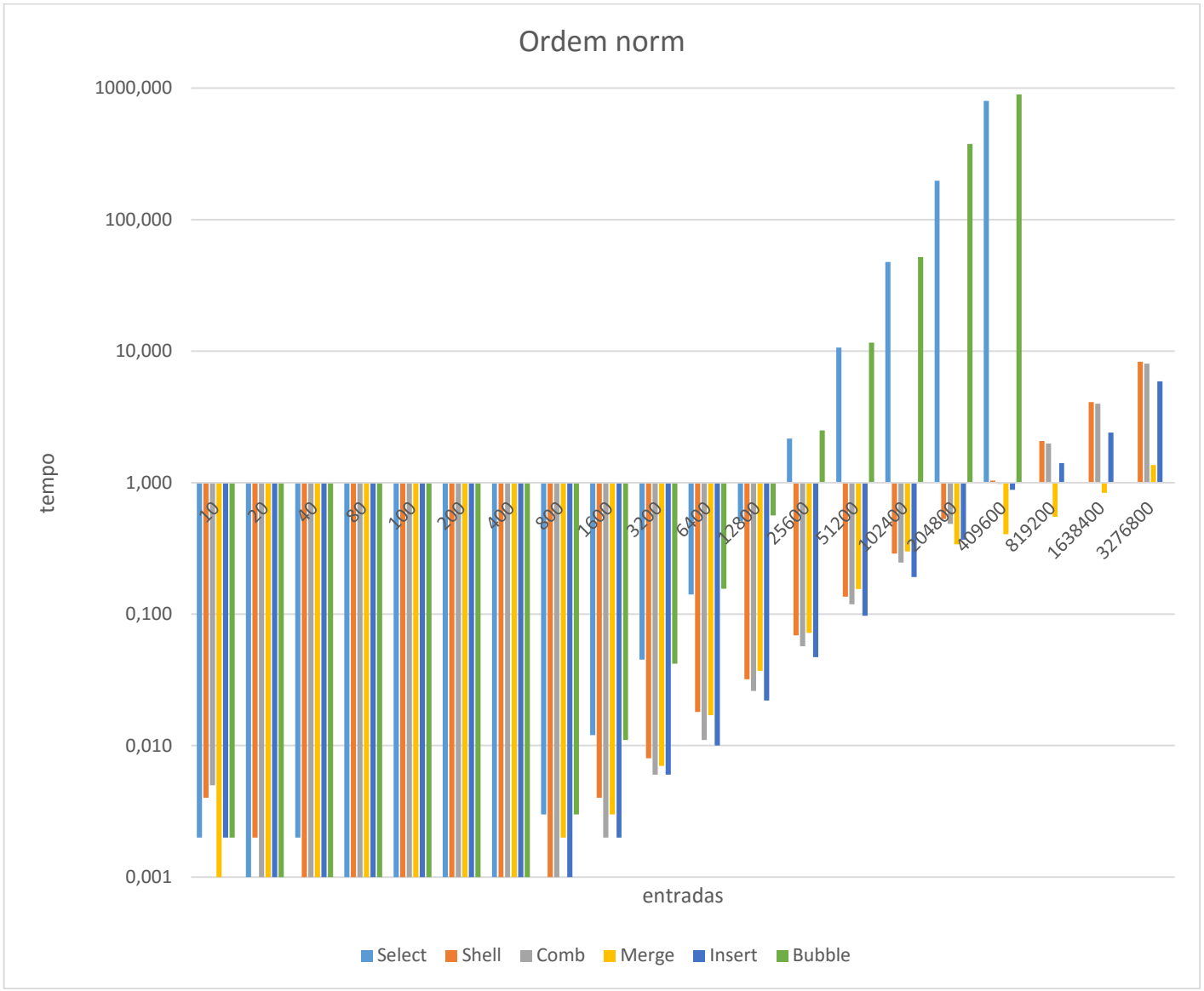


Figura 1- gráfico comparativo dos algoritmos. ordem norm. tempo em segundos em escala logarítmica para melhor visualização

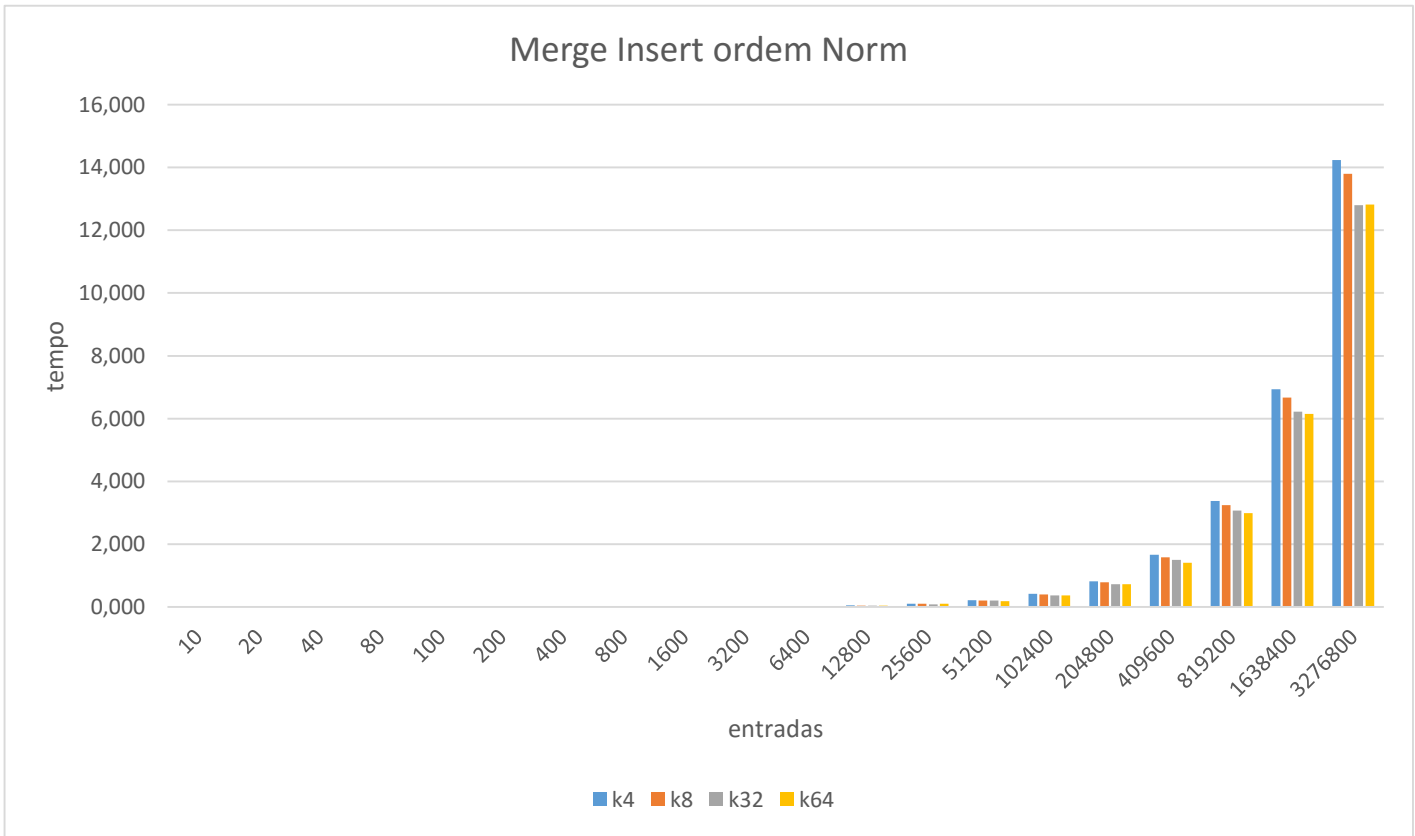


Figura 2 - gráfico comparativo dos algoritmos de merge-inser. Ordem norm.

b) Tabela e gráficos dos algoritmos. média de seus tempos comparados na ordem inversa para as mesmas entradas do item a)

Entradas	Select	Shell	Comb	Merge	Insert	Bubble	MI:k4	MI:k8	MI:k32	MI:k64
10	0.001	0.005	0.002	0.001	0.001	0.002	0.001	0.001	0.001	0.001
20	0.001	0.012	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
40	0.001	0.015	0.003	0.001	0.001	0.001	0.001	0.001	0.001	0.001
80	0.001	0.010	0.001	0.002	0.001	0.001	0.001	0.001	0.001	0.001
100	0.001	0.008	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
200	0.001	0.011	0.001	0.001	0.001	0.002	0.001	0.001	0.001	0.001
400	0.001	0.018	0.001	0.001	0.004	0.006	0.002	0.001	0.001	0.002
800	0.003	0.012	0.001	0.002	0.012	0.021	0.002	0.002	0.002	0.002
1600	0.011	0.013	0.003	0.004	0.047	0.076	0.004	0.004	0.004	0.004
3200	0.045	0.030	0.006	0.008	0.178	0.278	0.009	0.009	0.009	0.010
6400	0.166	0.034	0.015	0.016	0.718	1.083	0.020	0.021	0.025	0.019
12800	0.543	0.056	0.027	0.037	2.827	4.209	0.048	0.047	0.047	0.048
25600	2.300	0.088	0.069	0.079	11.466	16.778	0.102	0.104	0.097	0.124
51200	10.795	0.179	0.134	0.152	46.617	67.644	0.236	0.217	0.212	0.240
102400	48.314	0.347	0.289	0.302	190.09	271.07	0.412	0.402	0.421	0.443
204800	204.27	0.357	0.546	0.324	741.25	1076.7	0.823	0.805	0.819	0.897
409600	832.63	1.349	1.078	0.406	--------	--------	1.662	1.587	1.653	1.787
819200	--------	2.792	2.454	0.553	--------	--------	3.344	3.285	3.365	3.639
1638400	--------	5.491	4.410	0.853	--------	--------	6.935	6.816	6.883	7.421
3276800	--------	9.594	8.814	1.373	--------	--------	14.415	14.057	14.272	15.576

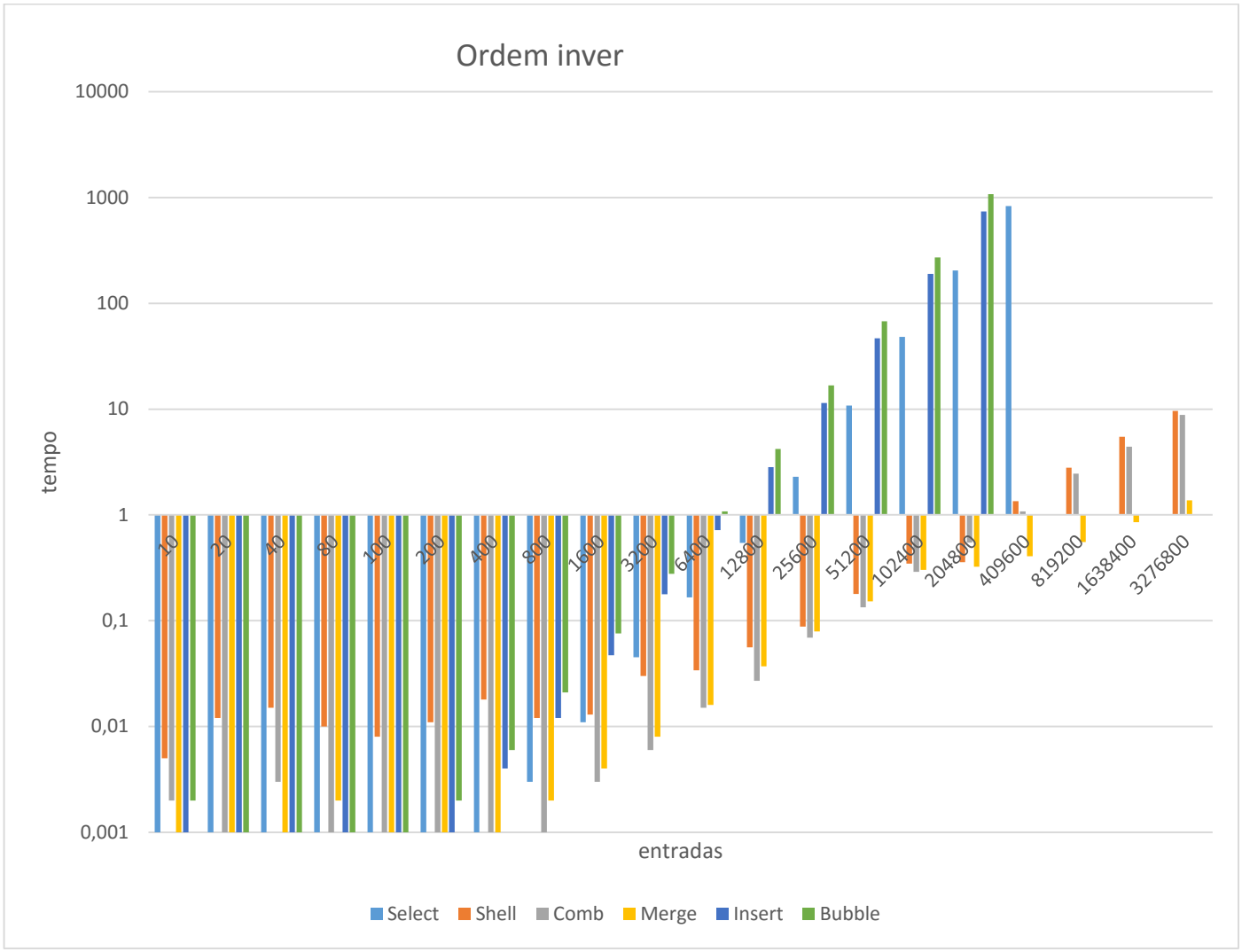


Figura 3-gráfico comparativo dos algoritmos. ordem inver. tempo em segundos em escala logarítmica para melhor visualização

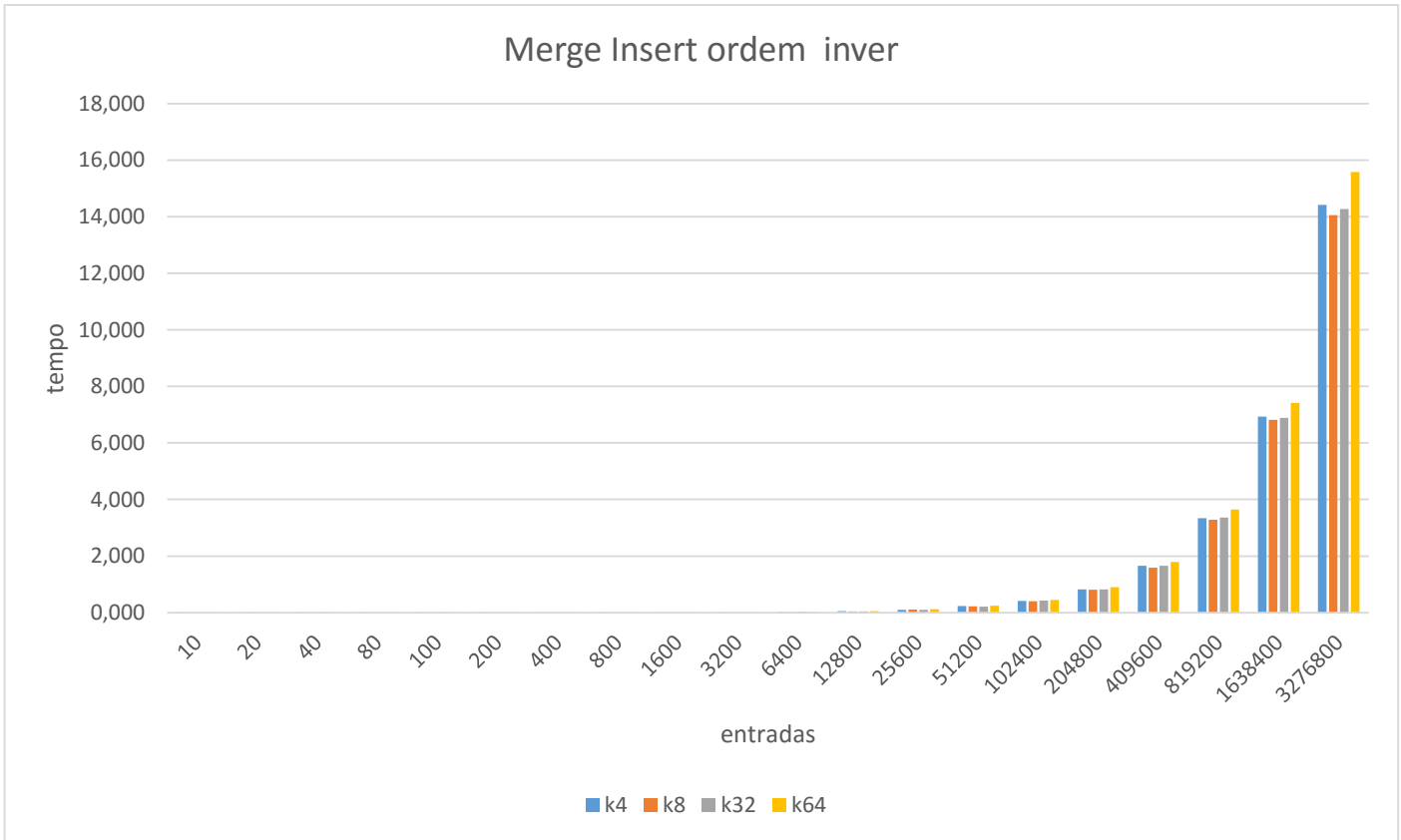


Figura 4 - gráfico comparativo dos algoritmos de merge-insert.. Ordem norm.

c) Tabela e gráficos. média dos algoritmos ordem aleatória

Entradas	Select	Shell	Comb	Merge	Insert	Bubble	MI:k4	MI:k8	MI:k32	MI:k64
10	0.001	0.003	0.001	0.001	0.002	0.002	0.001	0.001	0.002	0.001
20	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
40	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
80	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
100	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
200	0.001	0.001	0.001	0.001	0.001	0.002	0.001	0.001	0.001	0.001
400	0.002	0.001	0.001	0.001	0.002	0.005	0.001	0.001	0.001	0.001
800	0.003	0.002	0.002	0.002	0.007	0.016	0.002	0.002	0.002	0.002
1600	0.011	0.006	0.003	0.003	0.026	0.063	0.005	0.004	0.004	0.004
3200	0.041	0.010	0.007	0.008	0.098	0.235	0.010	0.009	0.009	0.009
6400	0.154	0.017	0.015	0.020	0.369	0.904	0.023	0.020	0.022	0.023
12800	0.533	0.042	0.033	0.037	1.432	3.580	0.052	0.045	0.054	0.052
25600	2.453	0.102	0.081	0.080	5.593	14.295	0.120	0.106	0.104	0.120
51200	10.570	0.190	0.155	0.146	23.477	58.585	0.210	0.208	0.210	0.210
102400	47.768	0.365	0.309	0.323	95.962	235.99	0.436	0.429	0.415	0.436
204800	206.59	0.820	0.667	0.343	377.96	950.43	0.863	0.837	0.819	0.863
409600	822.29	1.792	1.339	0.439	1476.7	--------	1.716	1.686	1.656	1.716
819200	--------	3.510	2.729	0.542	--------	--------	3.505	3.425	3.350	3.505
1638400	--------	8.380	5.543	0.826	--------	--------	7.254	7.020	6.900	7.254
3276800	--------	18.725	11.539	1.343	--------	--------	15.024	14.604	14.255	15.024

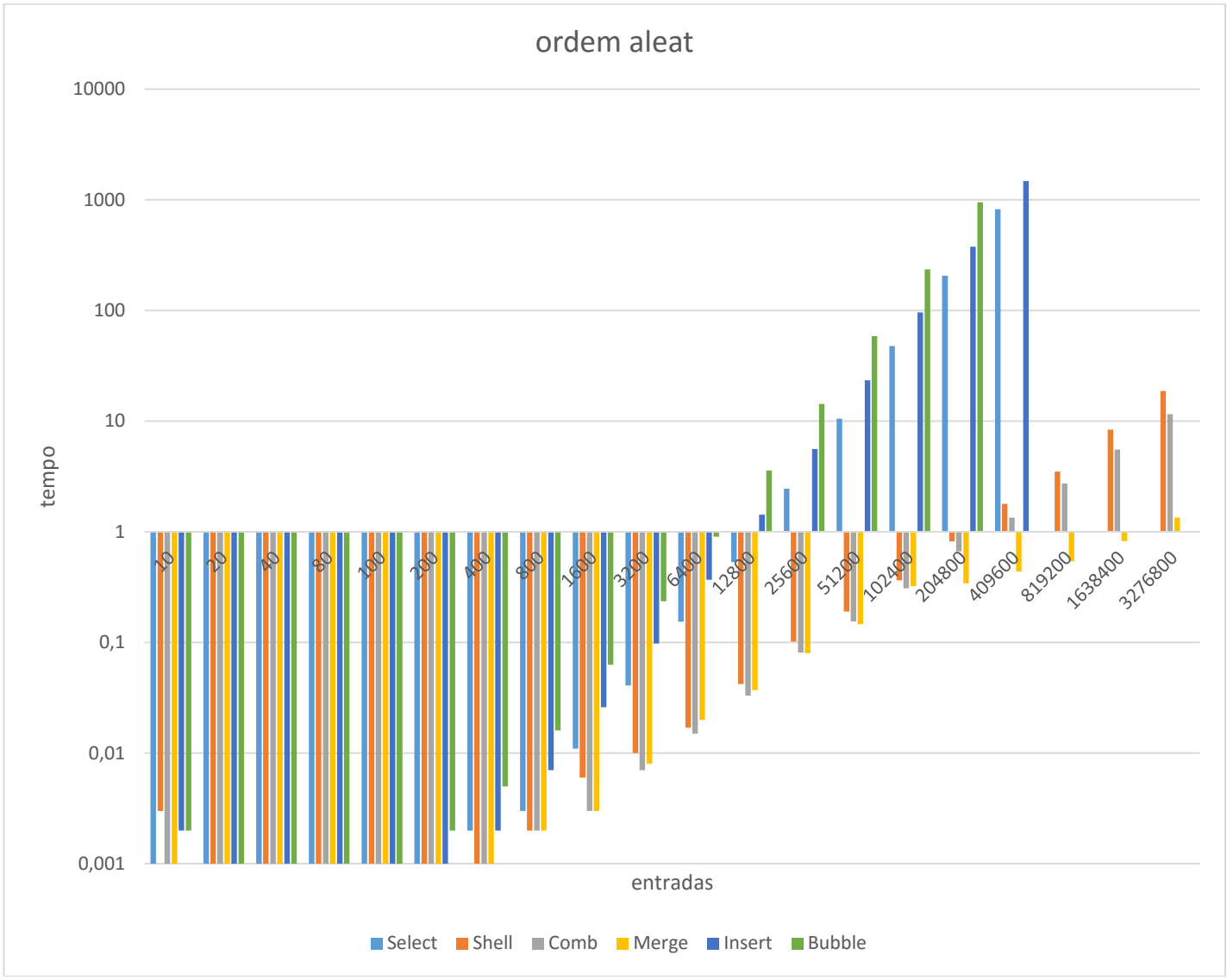


Figura 5-gráfico comparativo dos algoritmos. ordem aleat. tempo em segundos em escala logarítmica para melhor visualização

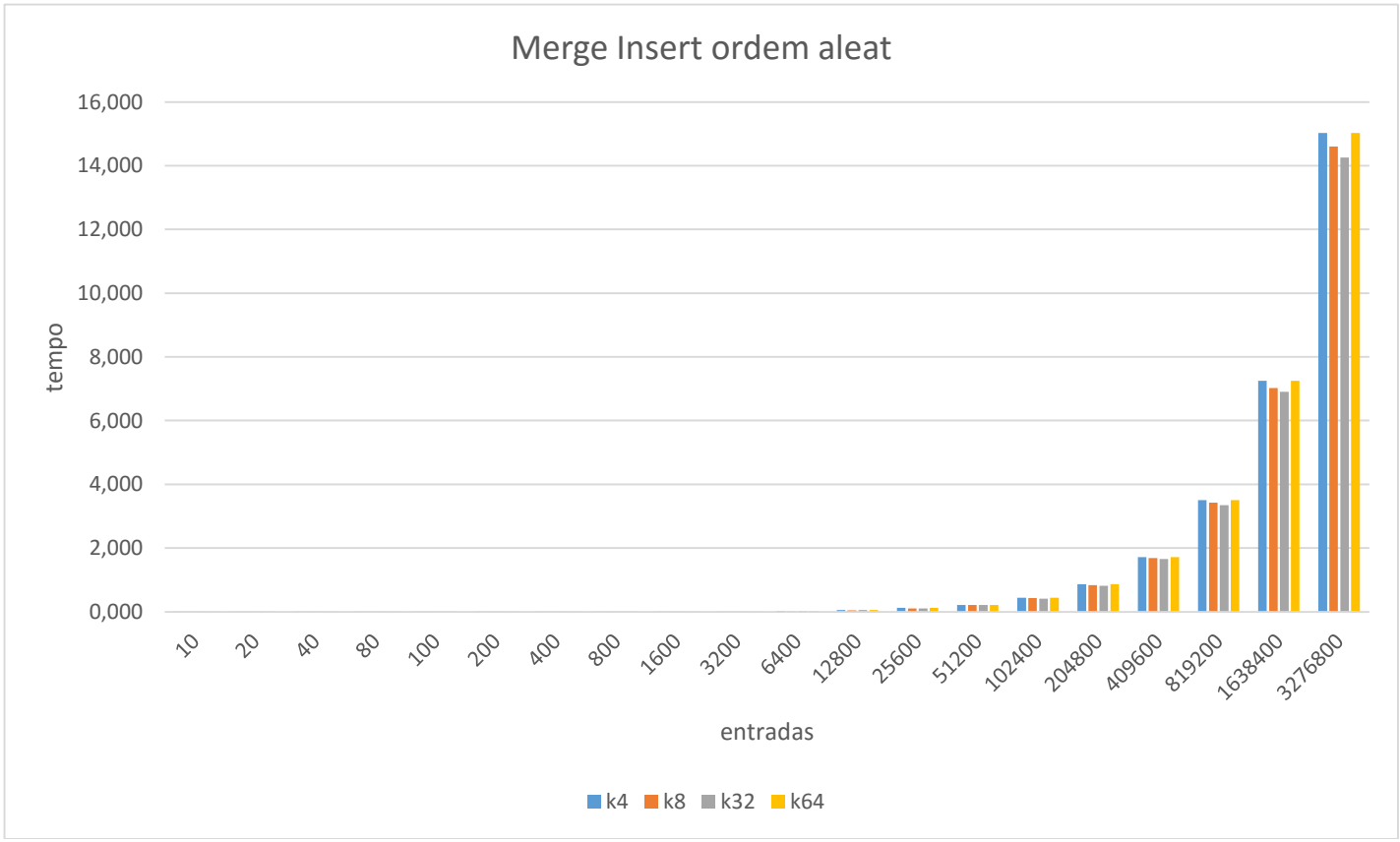


Figura 6- gráfico comparativo dos algoritmos de merge-insert. Ordem aleat.

d)Qual a relação entre o tempo de execução dos algoritmos e o número de inversões no arquivo a ser ordenado? Justifique sua resposta. Qual a complexidade do programa conta_inversão? Justifique sua resposta. Usando os tempos de execução e o número de inversões. faça um gráfico.

A relação entre o tempo de execução e o número de inversões no arquivo a ser ordenado é diretamente proporcional, cada vez que é necessário ser realizada mais inversões, mais processamento é necessário para ordenar as strings.

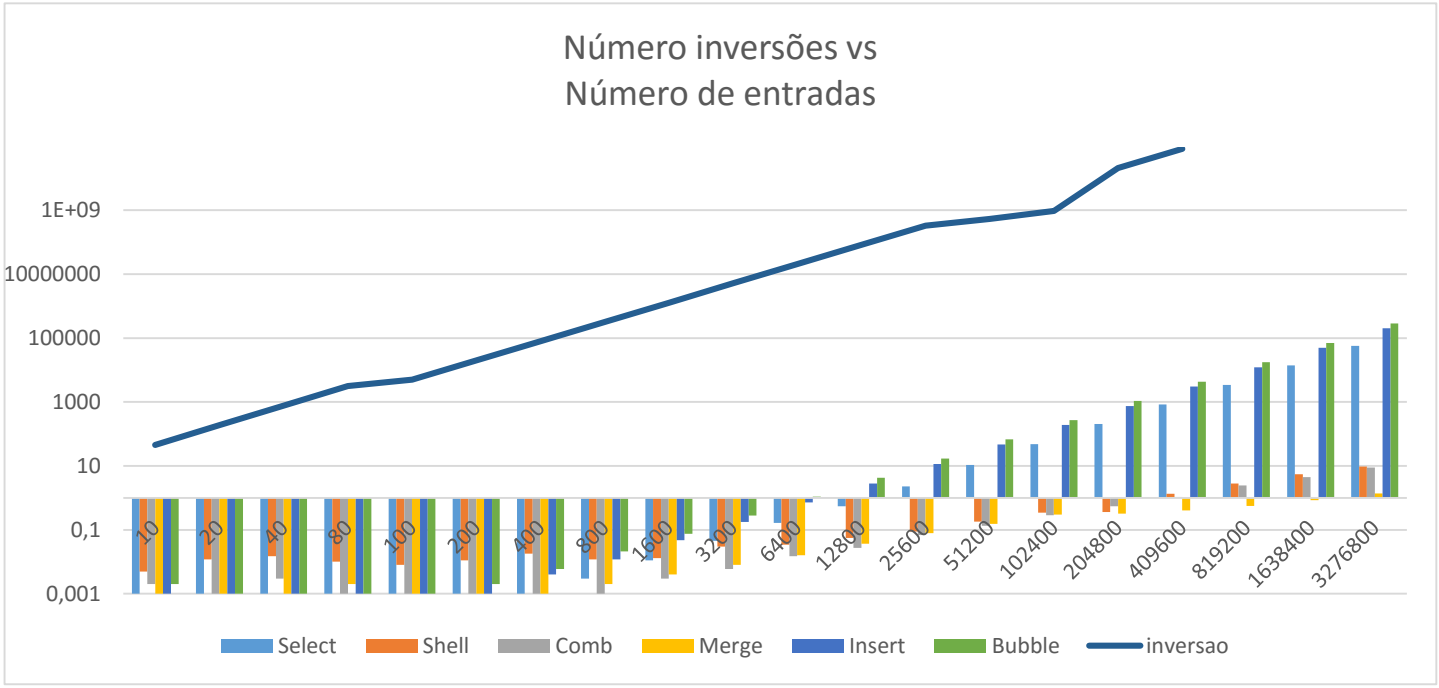


Figura 7- tempo em escala logarítmica para melhor visualização do crescimento da curva

e) Compare o impacto dos valores de k para o merge_insert. Faça um gráfico que mostre a influencia de k no tempo total. Quais valores de k obtiveram os melhores resultados? Justifique.

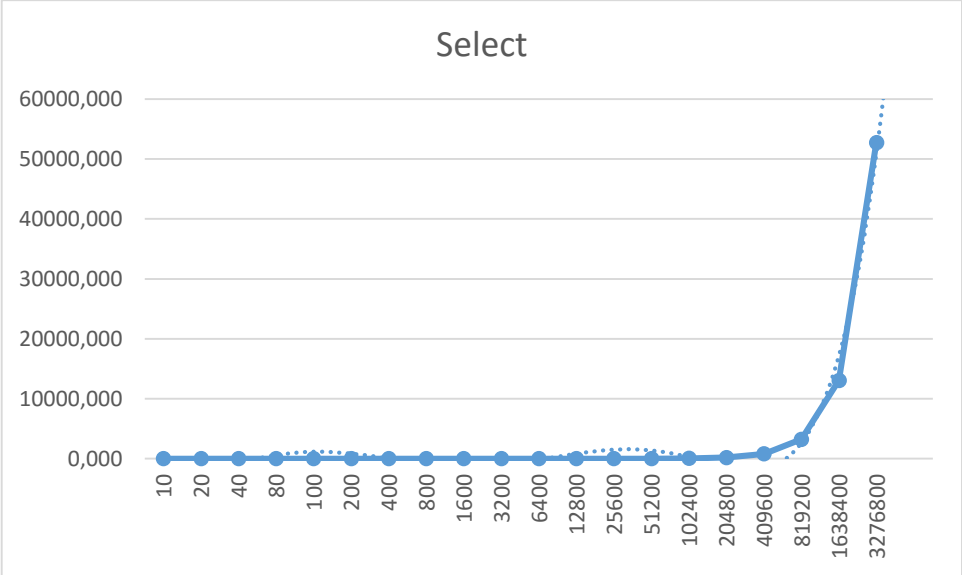
A execução do algoritmo de merge_insert obteve melhores resultados para k = 8 e k = 32. Esses foram os melhores resultados pois é a faixa onde o algoritmo de insertion sort é mais rápido para ordenar do que o merge sort e o tempo é maior em k = 4 e k = 64 pois para k = 4 o número de ordenações feita pelo insertion sort é menor e em k = 64 pois o número de elementos a serem ordenados já está começando a ficar grande e lento para ser ordenado pelo insertion sort.

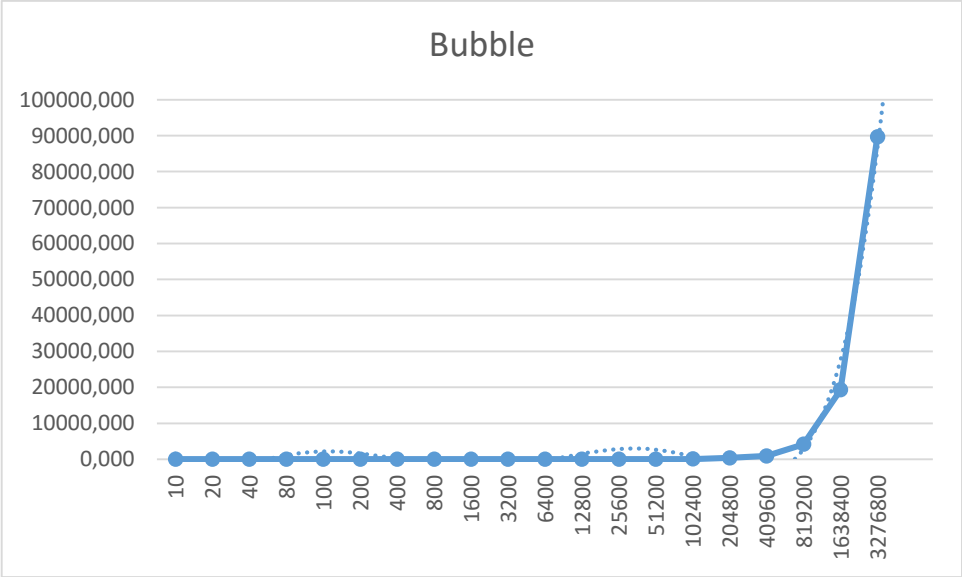
Para a ordem norm k = 32 e k = 64 possuem tempos melhores, com isso podemos concluir que na ordem norm o algoritmo de insertion sort é mais rápido que o de merge nesses valores de k. E pelo gráfico abaixo comparativo dos algoritmos mais rápidos, podemos observar que o insertion sort é mais veloz que o de merge até n = 204800 após isso n = 409600 o merge sort é mais veloz.

f) Para os casos de algoritmos (norm. inver e aleat) que não puderam rodar devido ao estouro do tempo. estime baseado na complexidade do algoritmo e nos tempos que foram medidos qual seria o tempo necessário para ordenar um vetor de tamanho n=3276800.

Norm:

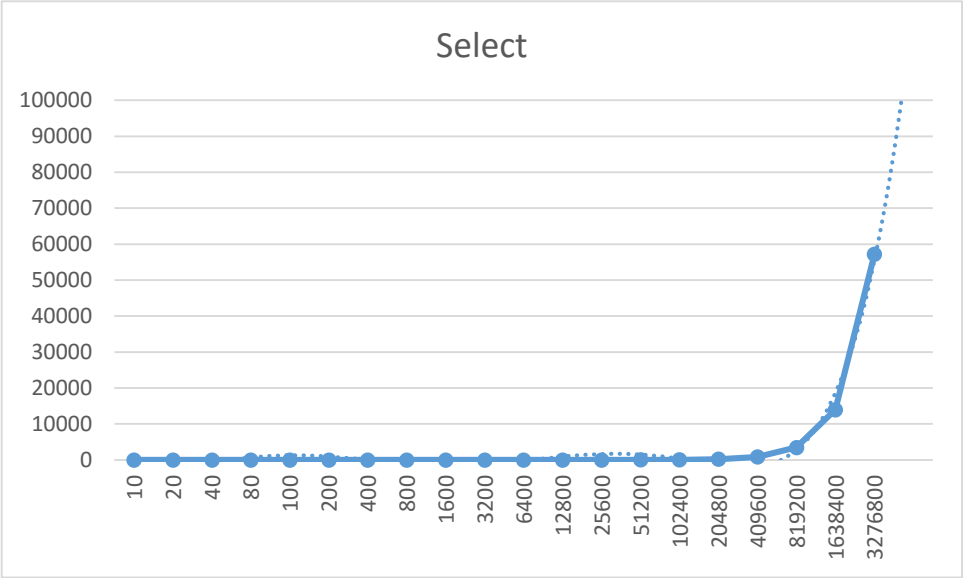
n	Selection sort	Bubble sort
3276800	~52750 s	~89600 s

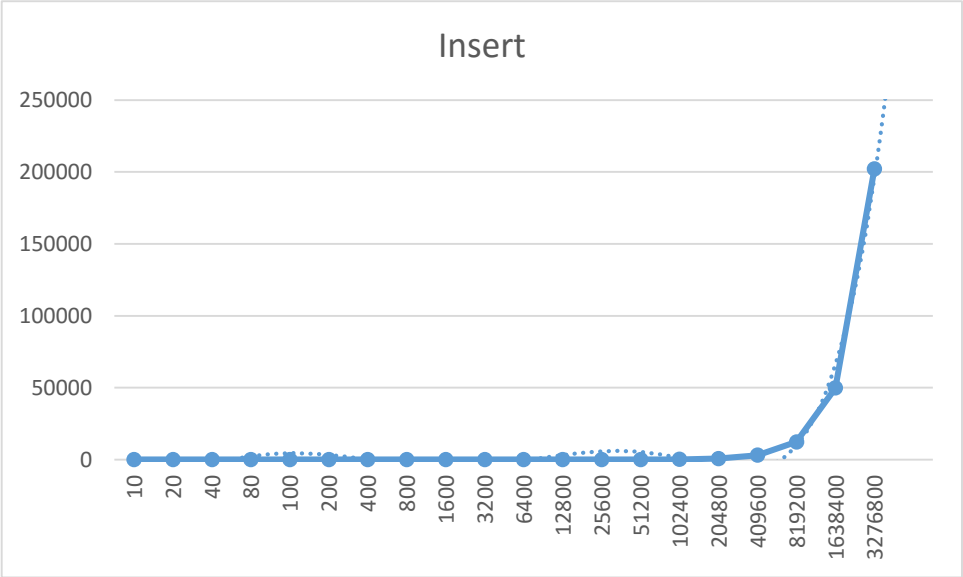


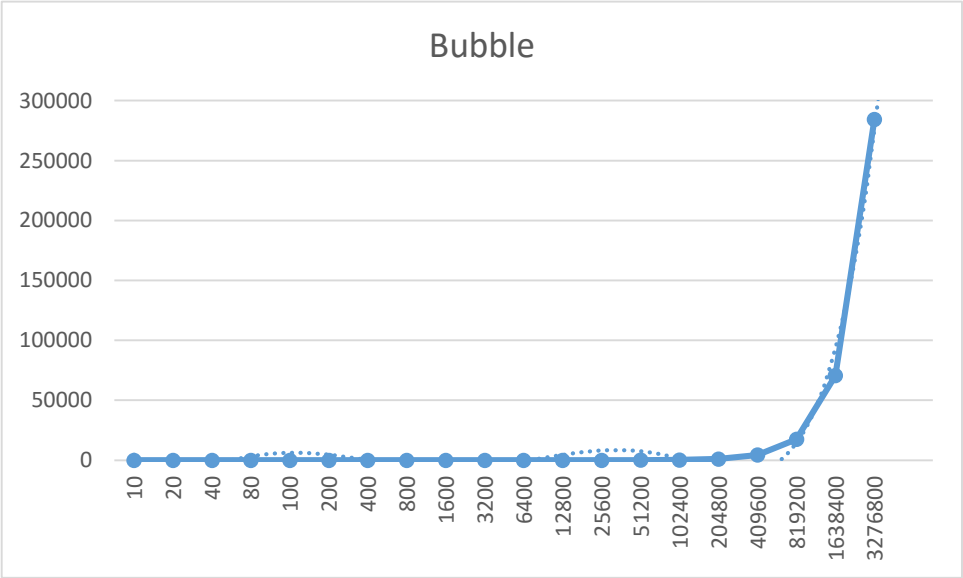


Inver:

n	Selection sort	Insertion sort	Bubble sort
3276800	~60000 s	~200000 s	~285000 s

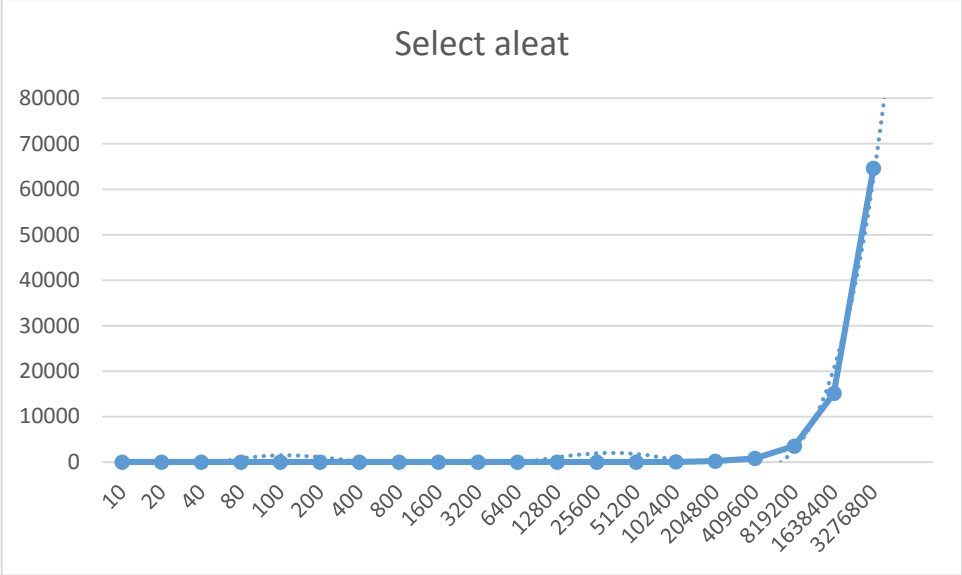


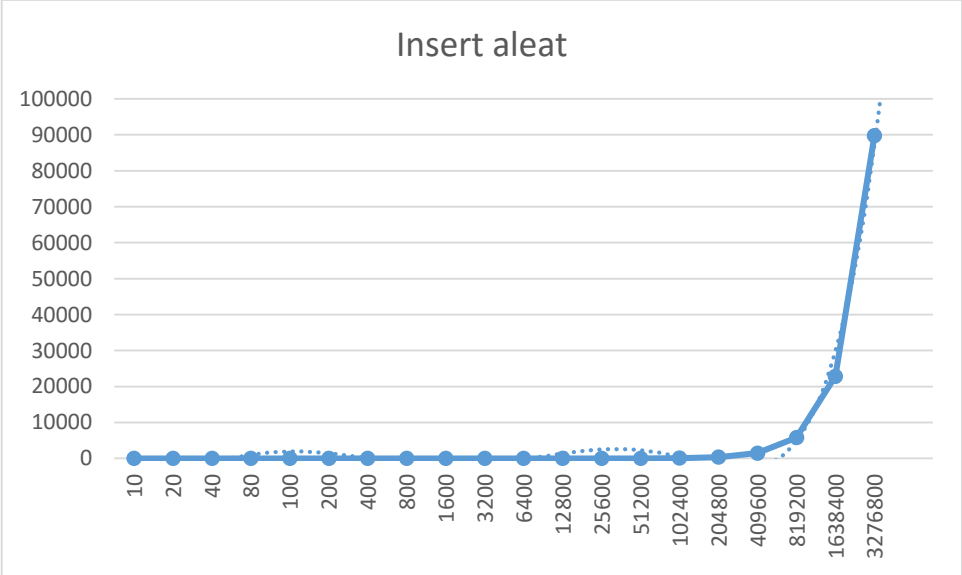


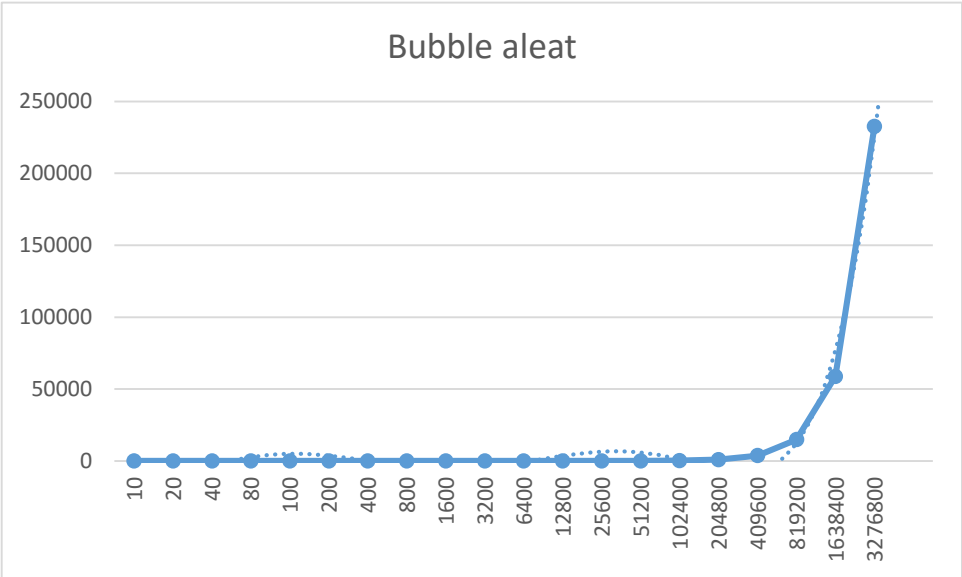


Aleat:

n	Selection sort	Insertion sort	Bubble sort
3276800	~ 8000 s	~ 89733,94 s	~ 232511,843 s







g) Para cada um dos casos (norm. inver e aleat) e para cada um dos tamanhos de n (10 - 3276800) faça uma tabela que indique qual algoritmo foi mais rápido. Os resultados estão de acordo com a complexidade do algoritmo? Justifique.

Entradas	Norm	Inver	Aleat
10	Merge	Select. Merge. Insert	Select. Comb. Merge
20	Select. Comb. Merge. Insert. Bubble	Select. Comb. Merge. Insert. Bubble	Select. Shell. Comb. Merge. Insert. Bubble
40	Shell. Comb. Merge. Insert. Bubble	Select. Merge. Insert. Bubble	Select. Shell. Comb. Merge. Insert. Bubble
80	Select. Shell. Comb. Merge. Insert. Bubble	Select. Comb. Insert. Bubble	Select. Shell. Comb. Merge. Insert. Bubble
100	Select. Shell. Comb. Merge. Insert. Bubble	Select. Comb. Merge. Insert. Bubble	Select. Shell. Comb. Merge. Insert. Bubble
200	Select. Shell. Comb. Merge. Insert. Bubble	Select. Comb. Merge	Select. Shell. Comb. Merge. Insert
400	Select. Shell. Comb. Merge. Insert. Bubble	Select. Comb. Merge	Shell. Comb. Merge
800	Shell. Comb. Insert	Comb	Shell. Comb. Merge
1600	Comb. Insert	Comb	Comb. Merge
3200	Comb. Insert	Comb	Comb
6400	Insert	Comb	Comb
12800	Insert	Comb	Comb
25600	Insert	Comb	Merge
51200	Insert	Comb	Merge
102400	Insert	Comb	Comb
204800	Merge	Merge	Merge
409600	Merge	Merge	Merge
819200	Merge	Merge	Merge
1638400	Merge	Merge	Merge
3276800	Merge	Merge	Merge