Laboratório de Ordenação

Nome: Matheus Crivellari Bueno Jorge

N de Matrícula: 636178

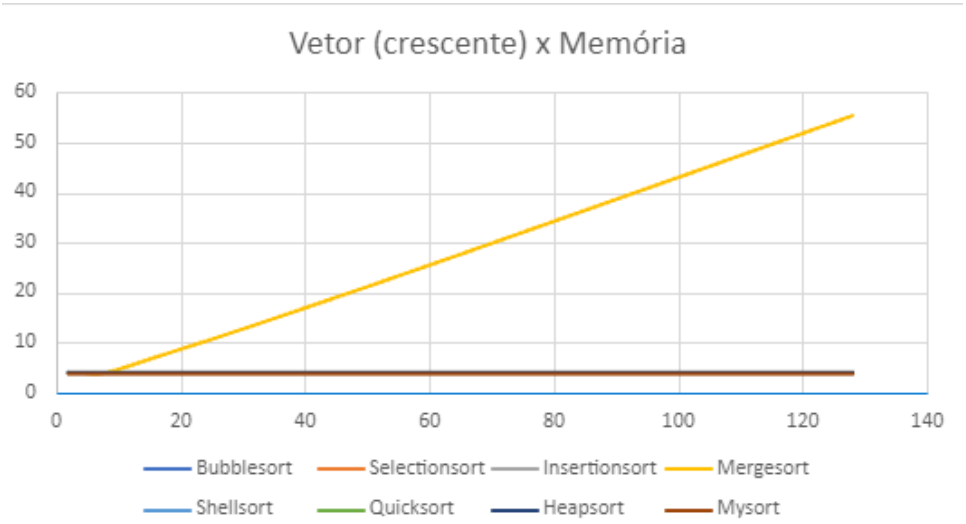
Foram usados os Algoritmos Bubblesort, Selectionsort, Insertionsort, Mergesort, Shellsort, Quicksort, Heapsort, Countingsort e o Mysort (Quicksort + Bubblesort) para ordenar vetores que vão de n=2k até n=128k elementos. Os vetores foram dispostos em 3 ordens (crescente, decrescente e aleatório). Abaixo estão os dados do gasto (memória e tempo) que cada algoritmo apresentou:

1) Memória em KB de cada Algoritmo:

a) Vetores de ordem crescente de 2K a 128k elementos:

Vetor (n)	Bubblesort	Selectionsort	Insertionsort	Mergesort	Shellsort
2	4,02	4,03	4,03	4,02	4,03
4	4,02	4,03	4,03	4,02	4,03
8	4,02	4,03	4,03	4,02	4,03
16	4,02	4,03	4,03	7,14	4,03
32	4,02	4,03	4,03	13,59	4,03
64	4,02	4,03	4,03	27,25	4,03
128	4,02	4,03	4,03	55,39	4,03

Vetor (n)	Quicksort	Heapsort	Mysort	Countingsort
2	4,03	4,03	3,8	4,03
4	4,03	4,03	3,8	4,03
8	4,03	4,03	3,8	4,03
16	4,03	4,03	3,8	4,03
32	4,03	4,03	3,8	4,03
64	4,03	4,03	3,8	4,03
128	4,03	4,03	3,8	4,03



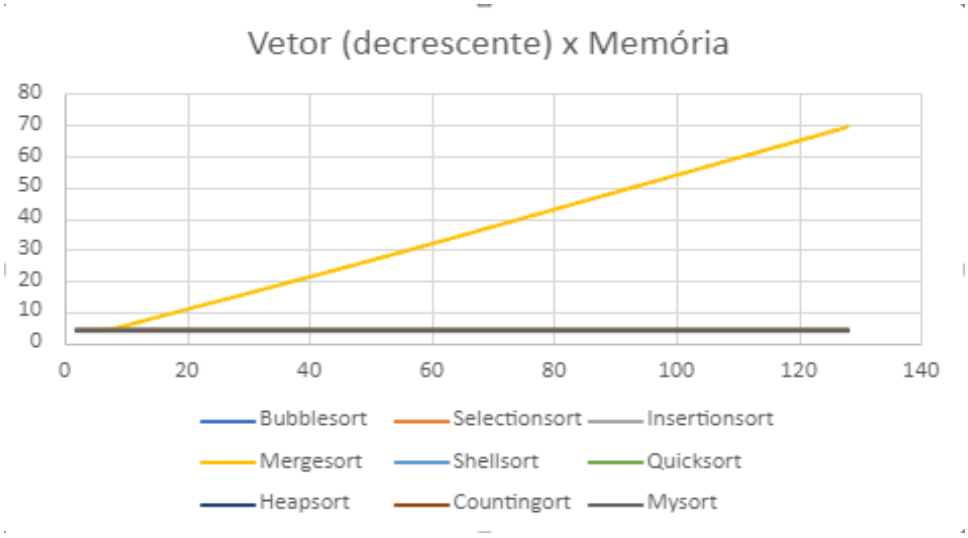
b) Vetores de ordem decrescente de 2K a 128k elementos:

Tabela:

Vetor (n)	Bubblesort	Selectionsort	Insertionsort	Mergesort	Shellsort
2	4,02	4,03	4,03	4,02	4,03
4	4,02	4,03	4,03	4,02	4,03
8	4,02	4,03	4,03	4,57	4,03
16	4,02	4,03	4,03	8,69	4,03
32	4,02	4,03	4,03	16,94	4,03
64	4,02	4,03	4,03	33,99	4,03
128	4,02	4,03	4,03	69,37	4,03

Vetor (n)	Quicksort	Heapsort	Mysort	Countingsort
2	4,03	4,03	3,8	4,03
4	4,03	4,03	3,8	4,03
8	4,03	4,03	3,8	4,03
16	4,03	4,03	3,8	4,03
32	4,03	4,03	3,8	4,03
64	4,03	4,03	3,8	4,03
128	4,03	4,03	3,8	4,03

Gráfico:



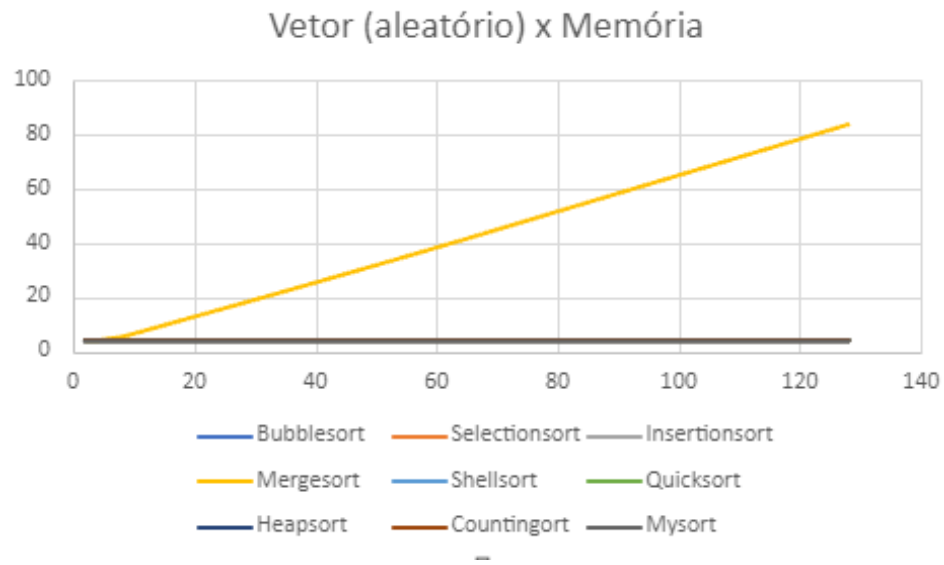
c) Vetores de ordem aleatória de 2K a 128k elementos:

Tabela:

Vetor (n)	Bubblesort	Selectionsort	Insertionsort	Mergesort	Shellsort
2	4,02	4,03	4,03	4,02	4,03
4	4,02	4,03	4,03	4,02	4,03
8	4,02	4,03	4,03	5,34	4,03
16	4,02	4,03	4,03	10,24	4,03
32	4,02	4,03	4,03	20,29	4,03
64	4,02	4,03	4,03	40,95	4,03
128	4,02	4,03	4,03	83,81	4,03

Vetor (n)	Quicksort	Heapsort	Mysort	Countingsort
2	4,03	4,03	3,8	4,03
4	4,03	4,03	3,8	4,03
8	4,03	4,03	3,8	4,03
16	4,03	4,03	3,8	4,03
32	4,03	4,03	3,8	4,03
64	4,03	4,03	3,8	4,03
128	4,03	4,03	3,8	4,03

Gráfico:



2) Tempo gasto em segundos por cada algoritmo:

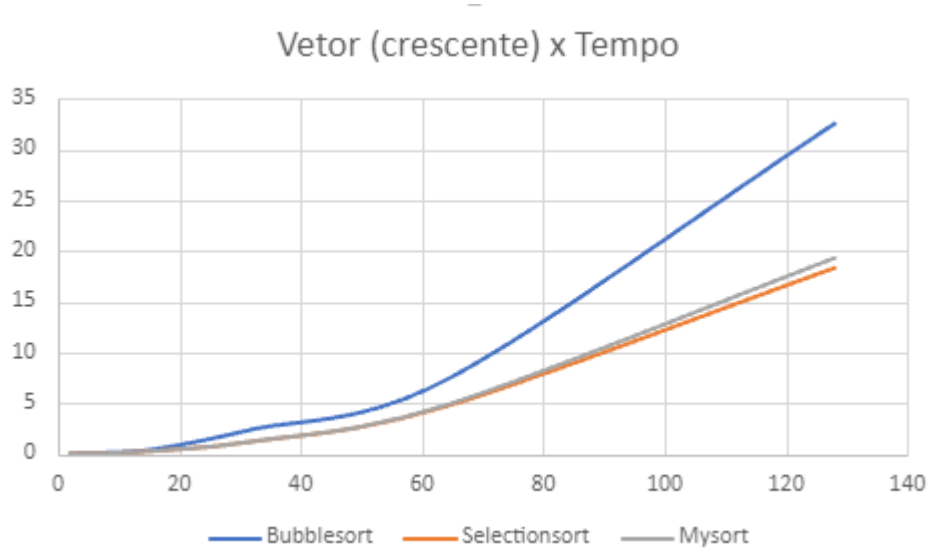
A) Vetores de ordem crescente de 2K a 128k elementos:

I) Algoritmos que gastam mais de segundo para ordenar um vetor ou mais:

Tabela:

Vetor	Bubblesort	Selectionsort	Mysort
2	0,00812	0,004892	0,006129
4	0,047355	0,016836	0,020788
8	0,133795	0,086342	0,08349
16	0,465394	0,298475	0,303249
32	2,410058	1,209995	1,221809
64	7,301923	4,693698	4,80811
128	32,58261	18,336963	19,31737

Gráfico:

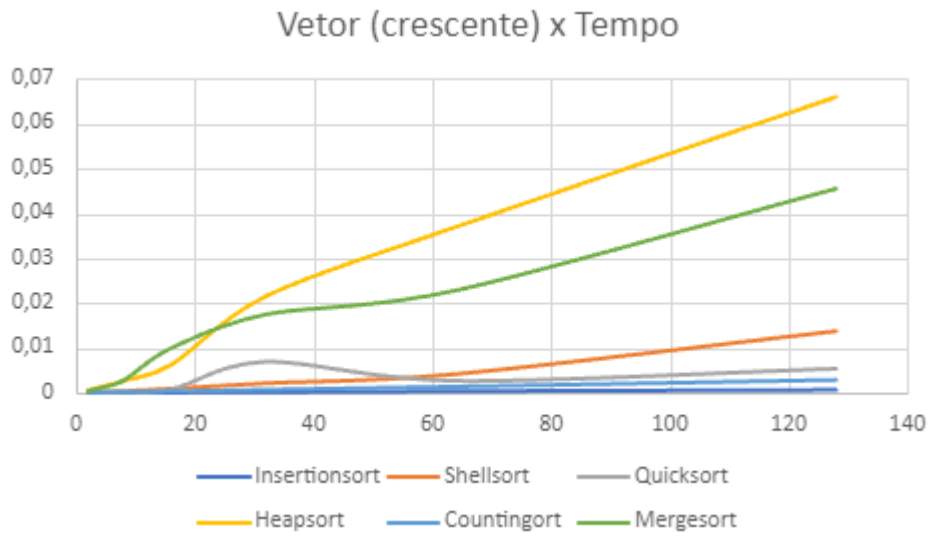


II) Algoritmos que gastam menos de 1 segundo para ordenar um vetor ou mais:

Tabela:

Vetor	Insertionsort	Shellsort	Quicksort	Heapsort	Countingort	Mergesort
2	0,000014	0,000106	0,000086	0,000586	0,000046	0,000402
4	0,000058	0,000213	0,000112	0,001223	0,000092	0,001072
8	0,000031	0,000372	0,000241	0,002665	0,000175	0,002739
16	0,000073	0,000932	0,000722	0,006233	0,00033	0,009957
32	0,000151	0,002139	0,00694	0,021573	0,0007	0,017499
64	0,000366	0,004187	0,002661	0,037031	0,001376	0,022842
128	0,000664	0,013754	0,00538	0,066004	0,002889	0,045534

Gráfico:



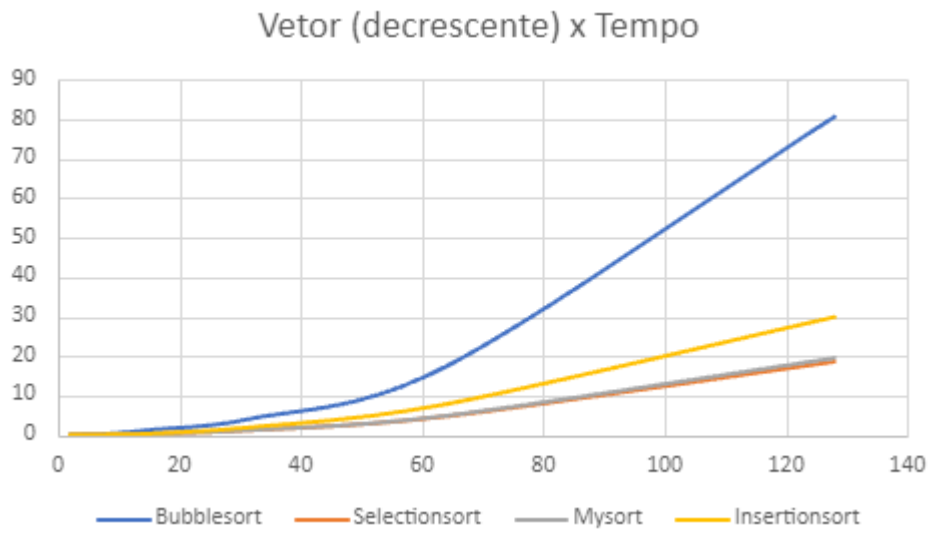
B) Vetores de ordem decrescente de 2K a 128k elementos:

I) Algoritmos que gastam mais de segundo para ordenar um vetor ou mais:

Tabela:

Vetor	Bubblesort	Selectionsort	Mysort	Insertionsort
2	0,019944	0,004628	0,00581	0,0055
4	0,098085	0,017948	0,020484	0,021208
8	0,272617	0,079204	0,079256	0,099774
16	1,37405	0,305736	0,305147	0,376209
32	4,158557	1,225201	1,236545	1,969901
64	17,37807	4,705709	4,856194	7,807221
128	80,704794	18,581577	19,465674	29,884952

Gráfico:

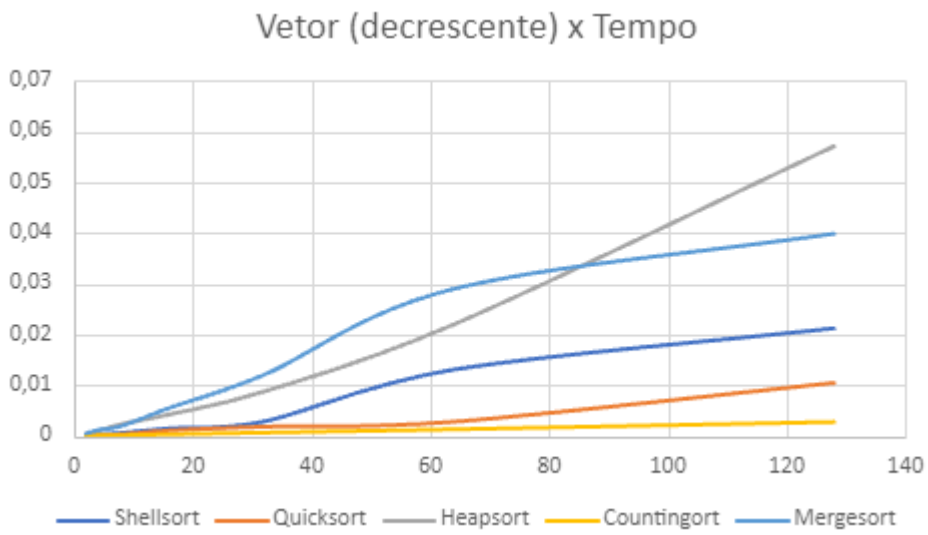


II) Algoritmos que gastam menos de segundo para ordenar um vetor:

Tabela:

Vetor	Shellsort	Quicksort	Heapsort	Countingort	Mergesort
2	0,000136	0,000068	0,000423	0,000043	0,000446
4	0,00028	0,000117	0,000913	0,000081	0,001152
8	0,000596	0,000295	0,002344	0,000152	0,00195
16	0,001482	0,001075	0,004223	0,000324	0,005499
32	0,002849	0,001816	0,008754	0,000651	0,012057
64	0,013091	0,002792	0,022009	0,001297	0,029024
128	0,021228	0,010455	0,057145	0,002753	0,03989

Gráfico:



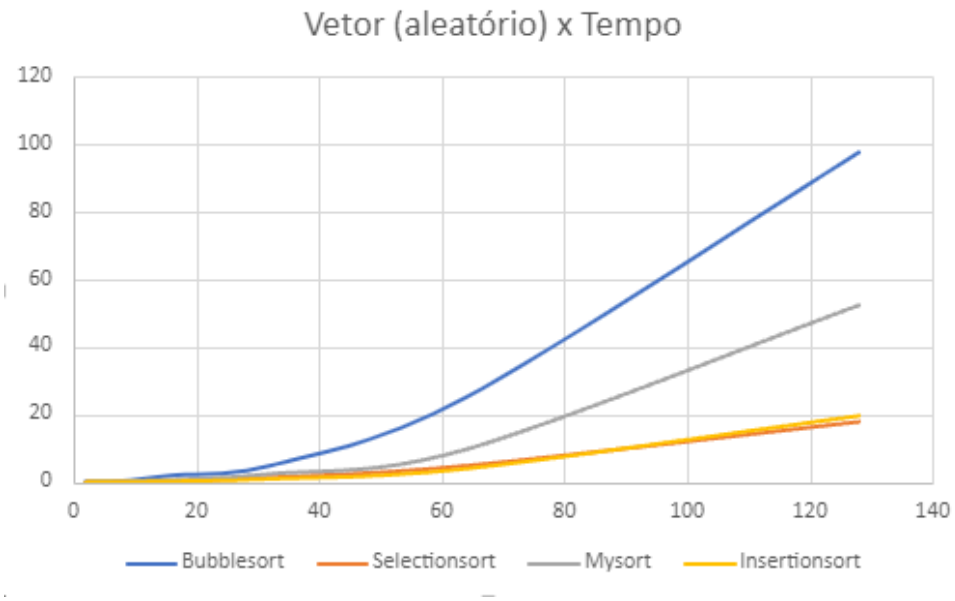
C) Vetores de ordem aleatória de 2K a 128k elementos:

I) Algoritmos que gastam mais de segundo para ordenar um vetor ou mais:

Tabela:

Vetor	Bubblesort	Selectionsort	Mysort	Insertionsort
2	0,012838	0,005342	0,007506	0,002641
4	0,080834	0,021905	0,39052	0,010034
8	0,270531	0,079769	0,211893	0,047707
16	1,934926	0,279566	0,636986	0,21133
32	4,775996	1,289525	2,34906	0,87474
64	25,086737	4,788412	9,631192	3,908829
128	97,629943	17,796473	52,362244	19,578648

Gráfico:

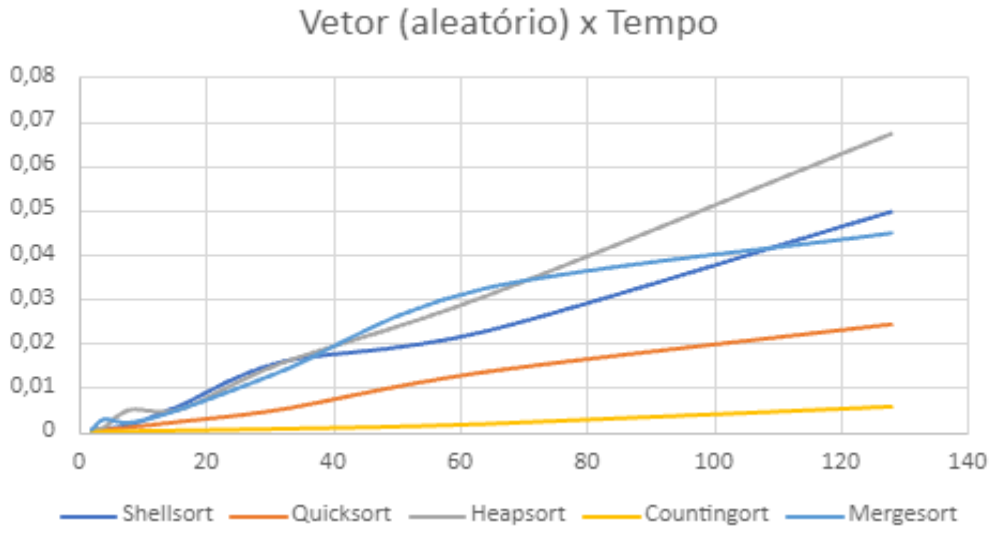


II) Algoritmos que gastam menos de segundo para ordenar um vetor:

Tabela:

Vetor	Shellsort	Quicksort	Heapsort	Countingort	Mergesort
2	0,00034	0,000242	0,000539	0,000039	0,000526
4	0,000706	0,000443	0,001096	0,00008	0,002867
8	0,001527	0,001018	0,004991	0,000171	0,00205
16	0,005983	0,002322	0,005134	0,000329	0,005126
32	0,015796	0,005083	0,01553	0,000688	0,013717
64	0,022657	0,013503	0,030567	0,001791	0,032369
128	0,049627	0,024259	0,0672	0,005697	0,0448

Gráfico:



Análise:

Gasto de memória:

Se tratando da memória gasta em KB, os algoritmos em geral, com exceção do Mergesort, gastaram entre 3,8 e 4,3 kb para todos os tamanhos e para todas as ordens (crescente, decrescente e aleatória) dos vetores. Vale portanto, destacar o comportamento do Mergesort que apresenta um crescimento de memória em todas as ordens de vetores, proporcional ao crescimento de elementos dos arrays. O Mergesort é um algoritmo que ordena através da divisão de um vetor em subvetores, sendo assim, exigindo do hardware mais memória para quanto mais vetores precisarem ser criados.

Gasto de tempo:

Para o gasto de tempo com vetores de ordem crescente, o Bubblesort apresentou o maior gasto, já que sua complexidade é de O(n^2) e apesar de estar em seu melhor caso para a troca de posições, o gasto comparativo ainda é quadrático. O Mysort mesmo realizando a primeira mudança no vetor através do Quicksort, que tem uma maior eficiência, acaba ficando com um alto gasto ao utilizar o Bubblesort O(n^2) para terminar a ordenação. O Selectionsort segue em terceiro lugar no gasto de tempo com um O(n^2). O resto dos vetores também apresentaram um gasto temporal correlacionado a sua complexidade: O Insertionsort cai no seu melhor caso: O(n) e assim apresenta o menor gasto temporal dentre todos os algoritmos. O Heapsort é o com maior gasto dentre os algoritmos que gastam menos de 1 segundo para ordenar algum vetor, com complexidade: O(n log n), sendo seguido nesse grupo dos inferiores de 1 segundo no gasto por: Mergesort O(n log n), Shellsort no qual tem uma análise de complexidade não tão bem definida matematicamente, sendo analisado majoritariamente de forma empírica, com uma complexidade de O (n^1,25) e Countingsort O (n +k).
Para o gasto de tempo com vetores em ordem decrescente e aleatória, a mesma análise de vetores em ordem crescente se mantém com algumas diferenças em certos algoritmos. O Bubblesort agora está em seu pior caso assim como o Selectionsort mantendo a complexidade de O(n^2). O Insertionsort está em seu pior caso e agora não é mais O(n) e sim O (n^2) sendo o segundo algoritmo de maior gasto, perdendo apenas para o Bubblesort e sendo seguido pelo Selectionsort e MySort com a mesma complexidade. O resto dos algoritmos apresentam o mesmo comportamento e complexidade da ordenação de vetores com elementos crescentes.

Conclusão:

Com a avaliação do gasto de cada algoritmo de ordenação (tempo e memória) correlacionado a complexidade de cada um especificamente, é possível perceber que algoritmos com a complexidade maior, exemplo os que apresentam O (n^2) independente do caso, sendo melhor, pior ou médio, tem sempre o maior gasto de tempo quando se vai aumentando o número de elementos no vetor. Estes como Bubblesort e Selectionsort não são úteis sob nenhuma pespectiva de eficiência temporal, pois caem neste caso. O Insertionsort é um que com o vetor quase ordenado cai perto da complexidade de O (n) sendo apenas neste caso útil. O Mergesort tem uma boa complexidade, que rende um bom gasto temporal, mas pode ser problemático ao se notar que é um dos algoritmos que aumenta progressivamente seu gasto de memória. O Quicksort que é um dos algoritmos mais utilizados no mundo, dentro das grandes empresas para o desenvolvimento dos mais diversos tipos de software com a complexidade O(n log n) em melhor e caso médio, nos testes não apresentou tanta diferença para os outros com a mesma complexidade. O Mysort foi um algoritmo criado para se testar a junção de um algoritmo pouco eficiente em questão de gasto de tempo (Bubblesort) com o um muito eficiente (Quicksort), o resultado dos testes foi como o esperado já que utilizando o Quicksort na primeira parte da ordenação, o gasto quadrático feito pelo Bubblesort ficou otimizado e consequentemente mais rápido, apesar de ainda manter a complexidade O(n^2). Vale citar que certos algoritmos testados podem ser bons em alguns casos: Countingsort eficiente para ordenar inteiros e Shellsort para arquivos de tamanho moderado.