UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CAMPUS CHAPECÓ
CURSO DE CIÊNCIA DA COMPUTAÇÃO

JOÃO VICTOR WINDERFELD BUSSOLOTTO
GUSTAVO DE SOUZA ZAMPIERON

COMPARAÇÃO DE ALGORITMOS DE ORDENAÇÃO:

DESEMPENHO DE ALGORITMOS DE ORDENAÇÃO QUADRÁTICOS.

CHAPECÓ
2022

JOÃO VICTOR WINDERFELD BUSSOLOTTO
GUSTAVO DE SOUZA ZAMPIERON

COMPARAÇÃO DE ALGORITMOS DE ORDENAÇÃO:

DESEMPENHO DE ALGORITMOS DE ORDENAÇÃO QUADRÁTICOS.

Trabalho apresentado ao Curso Ciência da Computação da Universidade Federal da Fronteira Sul (UFFS) como requisito parcial para aprovação na disciplina Pesquisa e Ordenação de Dados.

Professor: Geomar Andre Schreiner

CHAPECÓ

2022

1 INTRODUÇÃO

O presente trabalho apresenta uma análise do desempenho dos seguintes algoritmos de ordenação quadráticos: Bubble Sort, Insertion Sort e Selection Sort.

2 METODOLOGIA

Para analisar o desempenho, foram gerados três vetores: em ordem crescente, ordem decrescente e ordem aleatória. Ainda, o experimento foi repetido para vetores de 3 tamanhos: 10000, 50000, 100000. No intuito de testar o algoritmo sempre com os mesmos valores, os vetores foram gerados (com o arquivo arrayGenerator.c) e armazenados em arquivos de texto, para possibilitar uma comparação mais fidedigna. Os vetores estão anexados no trabalho e possuem o título no formato: tipo-tamanho. Ex.: aleatorio-10000.

Os algoritmos foram implementados em linguagem C. Ao executar o executável, deve-se informar o vetor. O código irá imprimir, no formato CSV, os dados de tempo de execução, número de trocas e número de comparações. Essas informações foram utilizadas para construir os gráficos e tabelas aqui presentes. Para fazer os gráficos, foi utilizado uma ferramenta online: rapidtables.com.

Para executar os algoritmos, foram utilizados os computadores dos dois integrantes do grupo: I5 da sétima geração com 8GB de RAM e um AMD Ryzen 2200g e 8GB de RAM. Percebeu-se que o desempenho era praticamente o mesmo em ambas as máquinas.

3 RESULTADOS

Tendo em vista uma melhor visualização, será apresentado os dados em tabelas e gráficos.

3.1 TABELAS

Tabela 1 - Tabela representando os valores achados com o Bubble Sort.

Bubble Sort				
Tipo do Vetor	Tamanho	Tempo	Comparações	Trocas
Aleatório	10000	0.440032	49995000	24905373
Aleatório	50000	11.820884	1249975000	626412981
Aleatório	100000	48.447628	4999950000	2494692214
Crescente	10000	0.000000	9999	0
Crescente	50000	0.000001	49999	0
Crescente	100000	0.000010	99999	0
Decrescente	10000	0.491038	49995000	49995000
Decrescente	50000	12.307922	1249975000	1249975000
Decrescente	100000	51.646172	4999950000	4999950000

Fonte: os autores.

Em primeiro momento, nota-se o bom desempenho para vetores ordenados crescentemente. Isso se deve a maneira que foi implementado, com a utilização de uma flag para evitar comparações desnecessárias. No entanto, esse método se mostrou um tempo elevado de execução para os outros casos, conforme o aumento do tamanho do vetor.

Tabela 2 - Tabela representando os valores achados com o Insertion Sort

Insertion Sort				
Tipo do Vetor	Tamanho	Tempo	Comparações	Trocas
Aleatório	10000	0.256529	49995000	24905373
Aleatório	50000	6.191462	1249975000	626412981
Aleatório	100000	24.280814	4999950000	2494692214
Crescente	10000	0.000000	9999	0
Crescente	50000	0.000999	49999	0
Crescente	100000	0.001002	99999	0
Decrescente	10000	0.462034	49995000	49995000
Decrescente	50000	12.118908	1249975000	1249975000
Decrescente	100000	47.587068	4999950000	4999950000

Fonte: os autores.

O Insertion Sort, como esperado, desempenha muito bem quando o vetor já está ordenado, ou parcialmente ordenado. Apresenta desempenho parecido com o Bubble Sort.

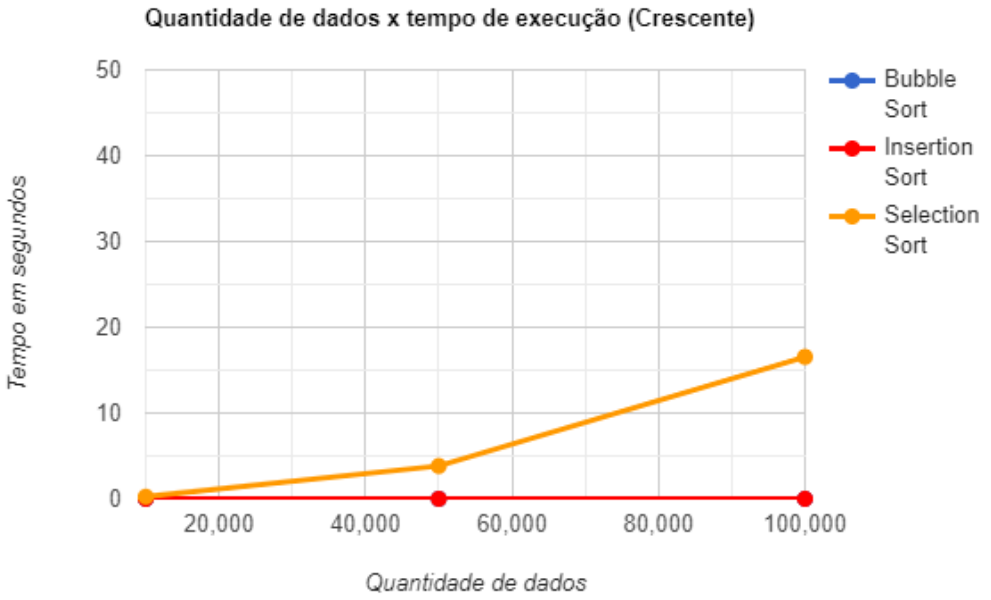
Tabela 3 - Tabela representando os valores achados com o Selection Sort

Selection Sort				
Tipo do Vetor	Tamanho	Tempo	Comparações	Trocas
Aleatório	10000	0.179013	49995000	9993
Aleatório	50000	4.112307	1249975000	49990
Aleatório	100000	15.756180	4999950000	99988
Crescente	10000	0.128008	49995000	0
Crescente	50000	4.055305	1249975000	0
Crescente	100000	16.147714	4999950000	0
Decrescente	10000	0.150010	49995000	5000
Decrescente	50000	4.093306	1249975000	25000
Decrescente	100000	16.207213	4999950000	50000

O selection sort mostrou ser o algoritmo de maior desempenho. É o mais eficiente dos três para todos os casos, exceto, quando o vetor está ordenado de maneira crescente. Nesse caso, ele realiza mais comparações que os demais, levando mais tempo.

3.2 GRÁFICOS

Figura 1 - Gráfico de linha.



Fonte:os autores.

Percebe-se pelo gráfico, que, quando o vetor já está ordenado de maneira crescente, o Insertion Sort e o Bubble Sort têm desempenho melhor que o Selection Sort para todos os tamanhos de vetor analisados.

Figura 2 - Gráfico de barra. Vetor decrescente. Quantidade de dados x tempo (em segundos).

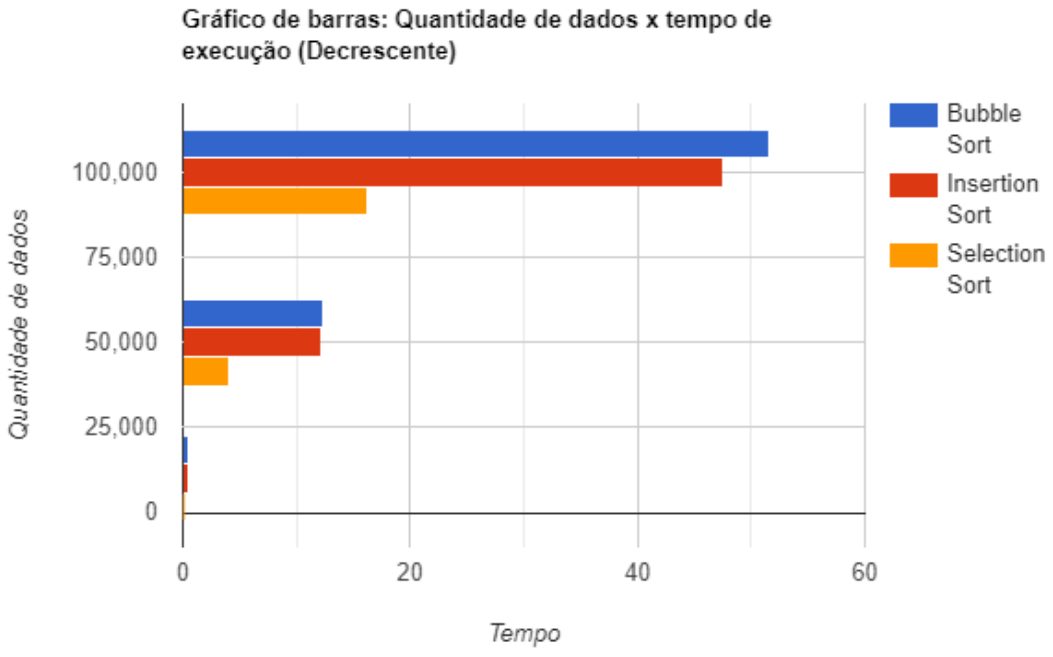
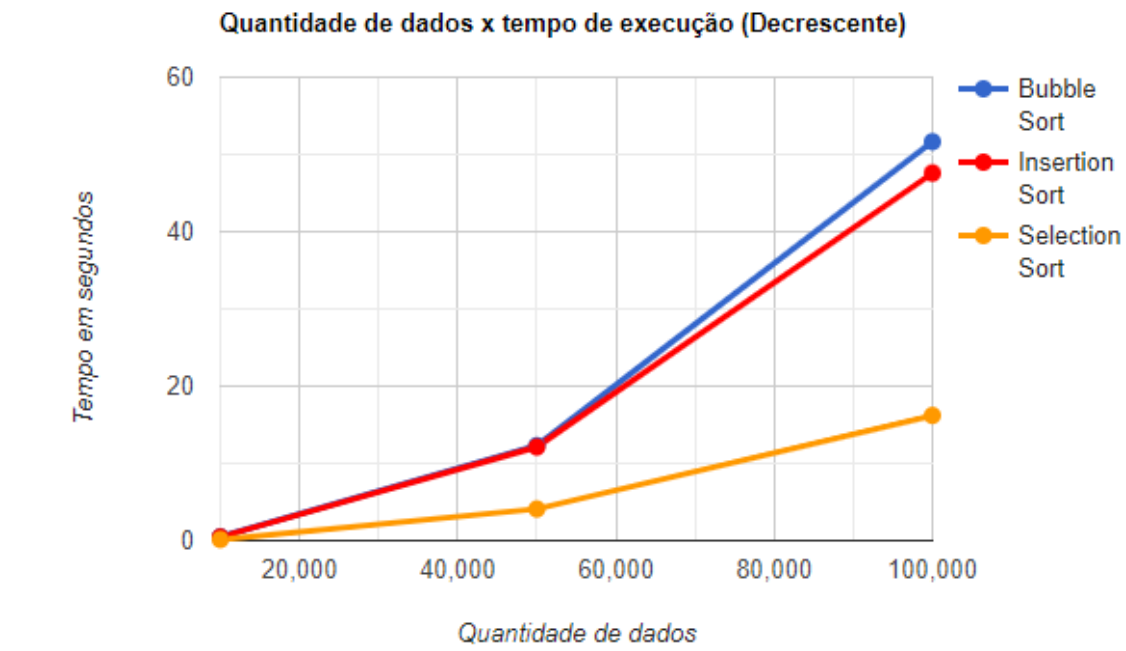


Figura 3 - Gráfico de barra. Vetor decrescente. Quantidade de dados x tempo (em segundos).



Fonte: os autores.

Nota-se, pelas figuras 2 e 3, que no caso do vetor estar ordenado de maneira decrescente(pior caso), o selection sort desempenha muito melhor no quesito tempo de execução. Ainda, bubble e insertion sort têm desempenhos praticamente iguais.

Figura 4 - Gráfico de barra. Vetor aleatório. Quantidade de dados x tempo (em segundos).

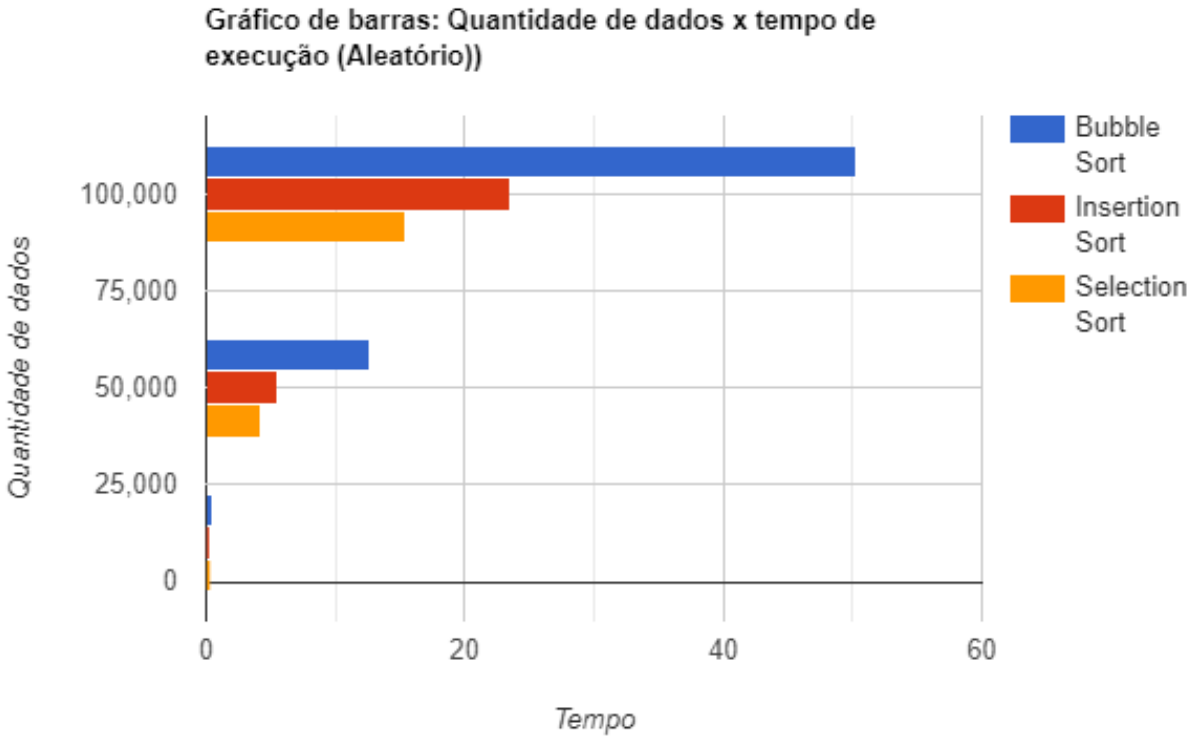
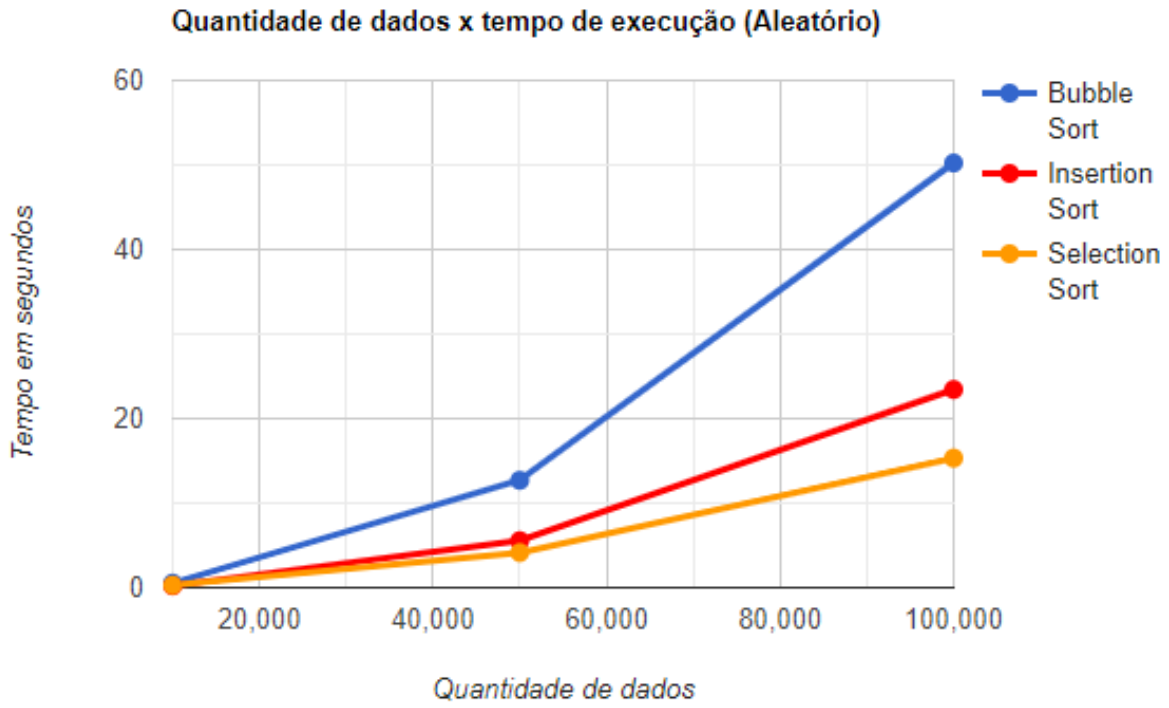


Figura 5 - Gráfico de linha. Vetor aleatório. Quantidade de dados x tempo (em segundos).



Nota-se através das figuras 4 e 5, que analisando de uma maneira que os vetores inseridos são aleatórios, o Selection Sort acaba por se sobressair em questão de velocidade em relação ao Insertion e Bubble Sortings. Em pequenas quantidades de dados ainda há possibilidade de uso do Insertion e o Bubble, porém para grandes quantidades há uma queda significativa de desempenho em relação ao Selection.

4 DIFICULDADES ENCONTRADAS

Durante o desenvolvimento, foi encontrado dificuldades para realizar a contagem de comparações. No entanto, o grupo acredita ter encontrado a solução mais adequada. Além disso, tivemos dificuldade para confeccionar os gráficos.

5 CONCLUSÃO

Em primeiro momento, notou-se que o desempenho dos algoritmos em relação ao tempo de execução é fortemente dependente do uso de CPU e memória

RAM. Em situações em que os testes foram executados com processos em segundo plano, o tempo de execução foi muito elevado para todos os algoritmos.

No tangente aos métodos de ordenação, fica claro que, o Selection Sort é o que melhor performa para o pior caso (vetor decrescente) e o vetor aleatório. A única situação em que o Selection Sort foi inferior, foi quando o vetor já se encontrava ordenado. Como, em situações reais, isso é de pouca probabilidade, faz mais sentido utilizar do Selection Sort para vetores de todos os tamanhos. Não obstante, a utilização do Bubble Sort para vetores de tamanhos muito grandes não é recomendada, visto a sua execução muito lenta.