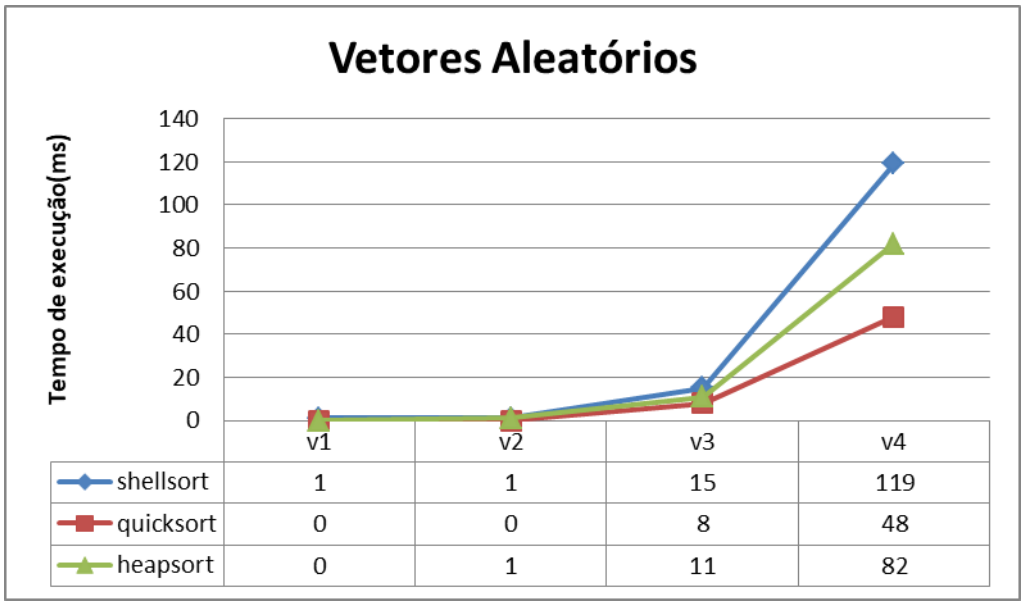
Documentação

O dilema principal deste trabalho prático é o estudo teórico entre alguns dos diversos algoritmos de ordenação existentes, realizando testes experimentais com cada um deles para variados tipos de vetores de tamanhos diferentes. Comparando-os e discutindo sua eficiência para cada caso estudado.

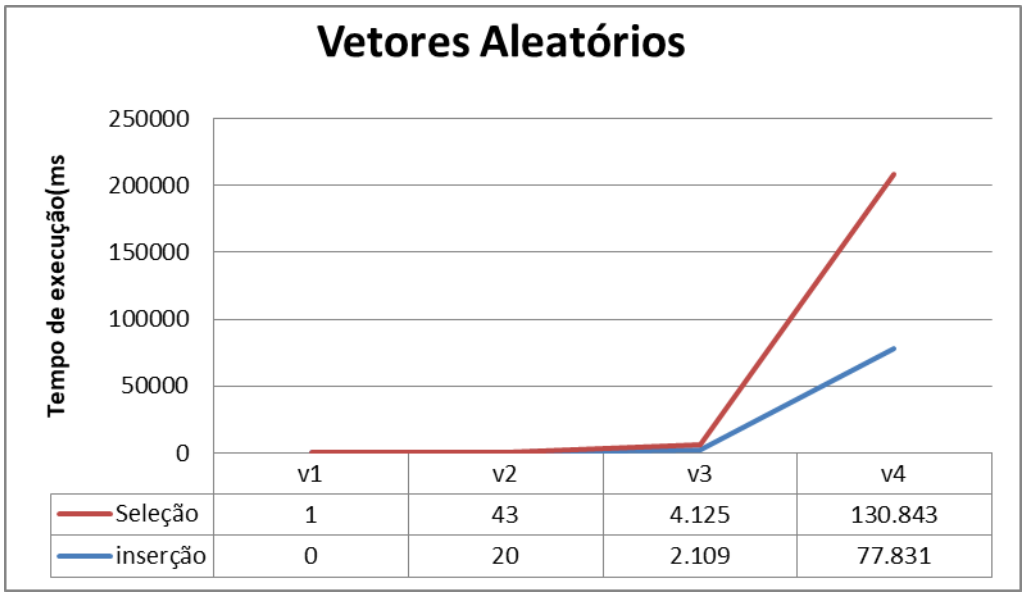
Os objetos de análise são os algoritmos : Inserção, Seleção, Shellsort , Quicksort e Heapsort. Como situação inicial é criado quatro vetores de inteiros para serem discutidos em três casos distintos: de forma aleatória ,crescente e decrescente , denominados V1, V2, V3 e V4 com tamanhos : 500, 5000, 50000 e 300000 respectivamente.

No trabalho foram utilizados os algoritmos de ordenação supracitados , e uma função chamada calctempo() para medir o tempo de execução de cada um, em que recebendo a variável “valor” é feita a distinção entre as funções de ordenação. É criada também uma variável “t” do tipo clock_t e com a função clock() retornar e armazenar o tempo decorrido desde que o programa começou, em seguida é chamado o algoritmo e sequencialmente é feita uma segunda chamada da função clock() para que “t” armazene o intervalo de tempo de execução da ordenação. No código é feito um menu para que possa ser apresentado os resultados de cada objeto estudado.

A ordem de complexidade do código é O($\mathbf{n}^2$) devido ao uso das funções quadráticas de ordenação dentro da função calctempo() .Para a apresentação dos resultados ,consideramos dividi-los em três situações, para os vetores criados de forma aleatória chegamos nos seguintes fatos que podem ser observados graficamente, sempre considerando o tempo de execução em milissegundos , para uma melhor visualização.

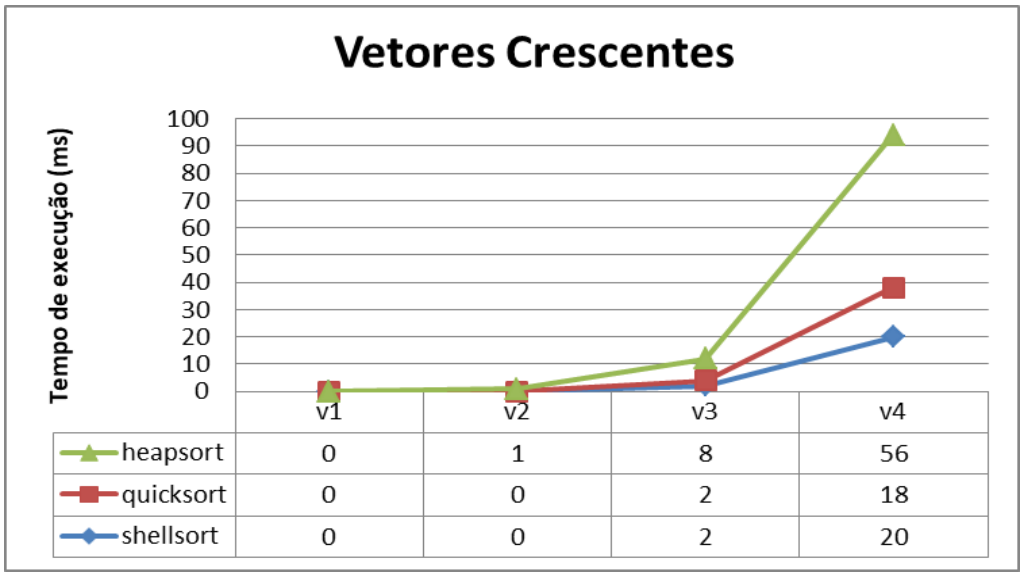


É feita a comparação entre Shellsort, Quicksort e Heapsort , e em seguida Inserção e Seleção devido a disparidade dos resultados obtidos. Nessa primeira comparação podemos observar uma leve diferença exponencial entre os algoritmos por tamanho do vetor , no qual nota-se um maior tempo de execução gasto em V3 e V4.



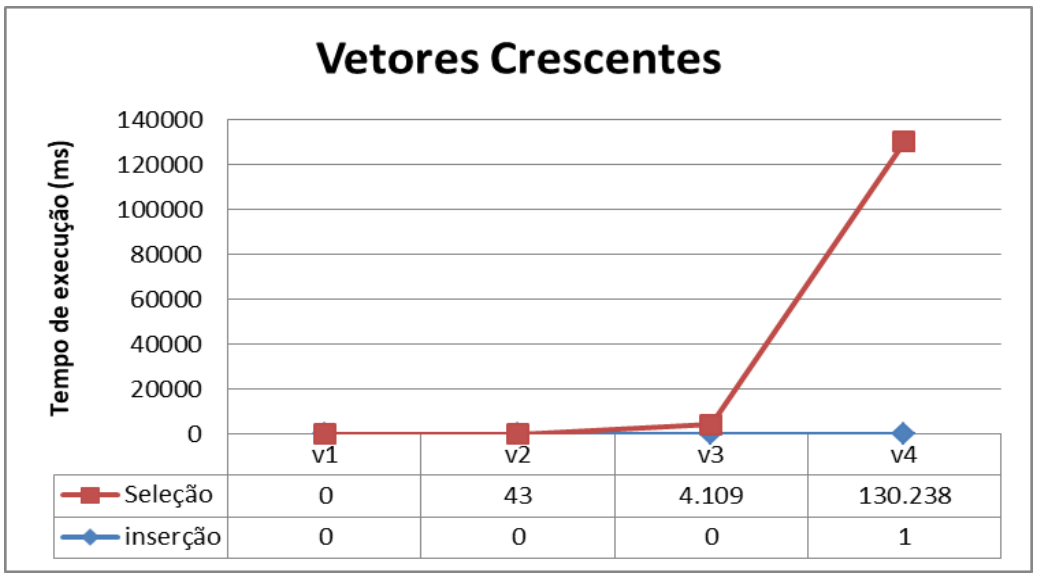
Já nessa segunda comparação é possível constatar uma maior diferença exponencial da eficiência do algoritmo para vetores maiores que V2 , no qual ultrapassa 2000 ms , em comparação com os anteriormente apresentados , que não passa dos 15 ms. E para vetores com tamanhos próximos ao de V4 , essa divergência é ainda mais acentuada, em que o algoritmo de Seleção por exemplo excede os 130.000 ms.

Na segunda situação estudada temos vetores já ordenados , podendo assim destacar a eficiência dos algoritmos para esse tipo de cenário.



	v1	v2	v3	v4
heapsort	0	1	8	56
quicksort	0	0	2	18
shellsort	0	0	2	20

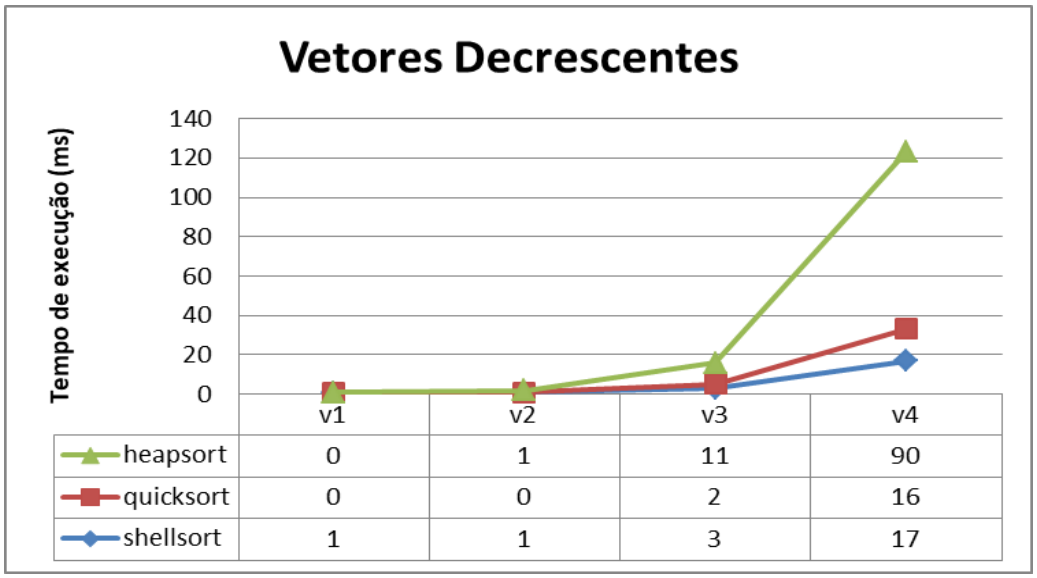
Aqui é percebido que a eficiência dos algoritmos é maior nesse caso do que no anterior , porém não muito grande , e nota-se um crescimento exponencial maior no caso do heapsort a medida que os vetores aumentam seu tamanho.



	v1	v2	v3	v4
Seleção	0	43	4.109	130.238
inserção	0	0	0	1

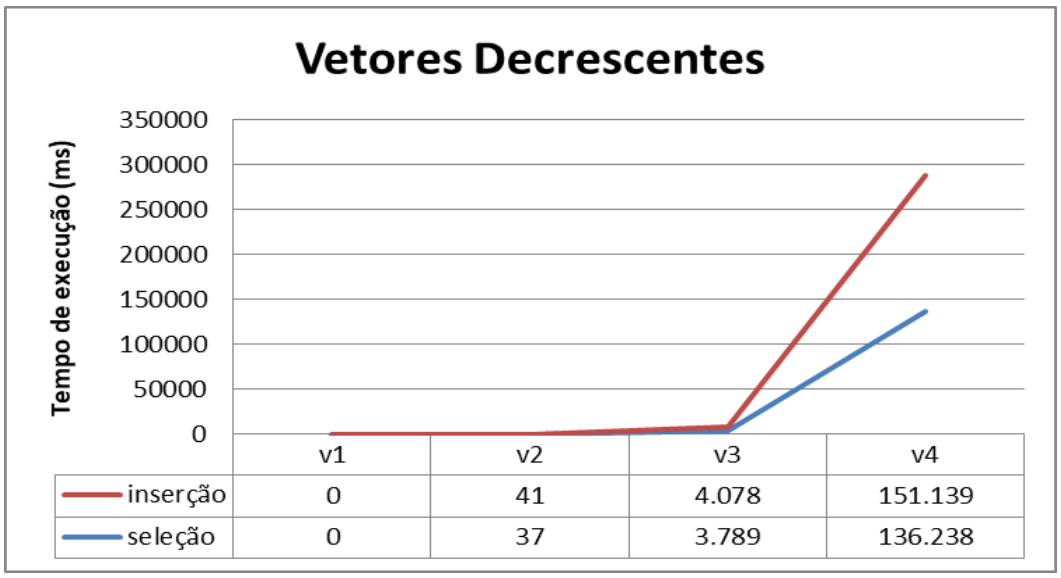
Já nessa comparação o destaque vai para o algoritmo de inserção , no qual sua eficiência é absoluta ao lados de todos os outros, para vetores já ordenados. Em contrapartida o de seleção quase não muda seu desempenho.

No Terceiro cenário temos os vetores em ordem decrescente.



	v1	v2	v3	v4
heapsort	0	1	11	90
quicksort	0	0	2	16
shellsort	1	1	3	17

Nesse caso a performance desses algoritmos se mantem em comparação com os outros , destacando novamente uma curva exponencial mais intensa no caso do heapsort em V4.



	v1	v2	v3	v4
inserção	0	41	4.078	151.139
seleção	0	37	3.789	136.238

Nessa comparação dos algoritmos quadráticos os tempos de execução são bem similares aos da primeira situação , evidenciando seu baixo rendimento para vetores de tamanhos maiores que V3.

Com a análise dos fatos anteriormente citados é possível fazer uma série de novas observações acerca dos objetos estudados.Com isso é possível concluir que por parte dos algoritmos de ordem quadrática , é provado sua eficiência para vetores de tamanhos pequenos, destacando o algoritmo de inserção para vetores já ordenados de qualquer tamanho, e também vale ressaltar a independência do de seleção para com a entrada dos vetores.

Já os algoritmos de ordem logarítmica , o Quicksort é o mais eficiente para uma grande variedade de situações e seu desempenho depende pouco da entrada, salvo pelo seu pior caso que possui baixa probabilidade com vetores aleatórios. Shellsort e Heapsort se mostraram algoritmos de qualidade , e sem ficar muito atrás do primeiro, visto que há pouca diferença entre eles. São vantajosos para vetores de tamanhos moderados. Seu desempenho não é interferido pelos fatores de entrada usando o Heapsort e para vetores parcialmente ordenados Shellsort é uma das melhores opções. E por fim mas não menos importante , é passível de ênfase a desigualdade gigantesca entre os algoritmos quadráticos e os logarítmicos , provados por suas ordens de complexidade.

Diante dos eventos supracitados é possível concluir com sucesso o estudo dos objetos e consolidar o aprendizado envolvido , ligando a analise dos algoritmos com os resultados obtidos.

Referências Bibliográficas

http://www.rafaeltoledo.net/algoritmos-de-ordenacao-4-shell-sort/

https://www.ime.usp.br/~pf/algoritmos/aulas/ordena.html

https://terminaldeinformacao.com/2015/06/15/shell-sort-e-quick-sort/

https://terminaldeinformacao.com/2013/05/10/ordenando-vetores-usando-linguagem-c/