**Filipe Matos Ferracini**

RGA: 2020.1907.052-6

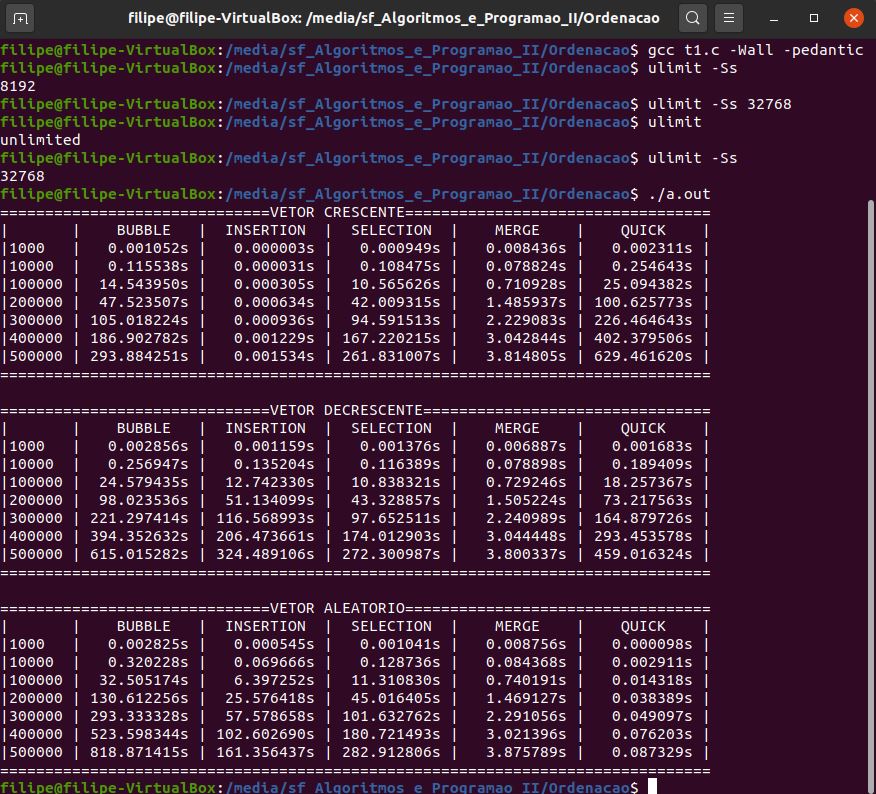
**ALGORITMOS E PROGRAMAÇÃO II – T04:**

**Comparação de algoritmos de ordenação**

Primeiramente, cabe apontar que o objetivo do código anexo é calcular a velocidade de execução de cinco diferentes métodos de ordenação (Bubble Sort – Ordenação Bolha, Insertion Sort – Ordenação por inserção, Selection Sort – Ordenação por inserção, Merge Sort – Ordenação por intercalação e QuickSort), para ordenar na ordem crescente um dado vetor.

Todos os métodos de ordenação foram testados para vetores de ordem crescente, vetores de ordem decrescente e, finalmente, para vetores de ordem aleatória gerados segundo o algoritmo de Fisher-Yates. Quanto aos tamanhos, foram gerados cinco tamanho para cada ordem inicial: 1000, 10000, 100000, 200000, 300000, 400000, 500000.

Os resultados obtidos foram os que seguem:



Cabe aqui, inclusive, ressaltar que para que a execução do programa tenha sucesso, é necessário habilitar um limite maior de alocação de memória, de 8mb para 32mb. Caso o mesmo não seja feito, o programa sofre Segmentation fault, justamente por sofrer Stack overlflow. Esta alteração do limite da shell resolve o problema e permite o sucesso da aplicação.

Primeiro, no método Bolha, o intuito do código é fazer com que o maior valor fique à direita do elemento analisado. Assim, o vetor deve ser percorrido n vezes até a enésima posição. Portanto, é evidente que sua complexidade ser **O(n²)**, porém, neste caso de vetor ordenado, sua complexidade cai para **O(n)**.

De forma similar, o método de inserção se dá por comparação e inserção. Conforme o vetor é percorrido, o algoritmo vai organizando um a um os valores de modo que o menor elemento fique sempre à esquerda do próximo. Assim, apesar de ser um algoritmo mais eficiente, sua complexidade é de **O(n²)**. No melhor caso, ou seja, no vetor crescente, a complexidade cai para **O(n)**.

Primeiramente, analisamos o que seria o melhor caso: Vetor crescente. O vetor já está ordenado, porém o algoritmo continua precisando analisar cada elemento se o vetor está na ordem crescente. Aqui a eficiência do processo começa fica evidente.