Relatório Projeto 4.4 AED 2020/2021

Nome: Pedro Afonso Ferreira Lopes Martins Nº Estudante: 2019216826

TP (inscrição): PL8 Login no *Mooshak*: 2019216826

Nº de horas de trabalho: 06*H* *Aulas Práticas de Laboratório: 02H* *Fora de Sala de Aula: 04H*

(A Preencher pelo Docente) CLASSIFICAÇÃO:

Comentários:

Registar os tempos computacionais das variantes em consideração para os diferentes tipos de sequências. O tamanho das sequências (N) deve ser crescente e terminar em 10,000,000. Só deve ser contabilizado o tempo de ordenamento. Exclui-se o tempo de leitura do input e de impressão dos resultados.

Gráfico para SEQ_ALEATORIA

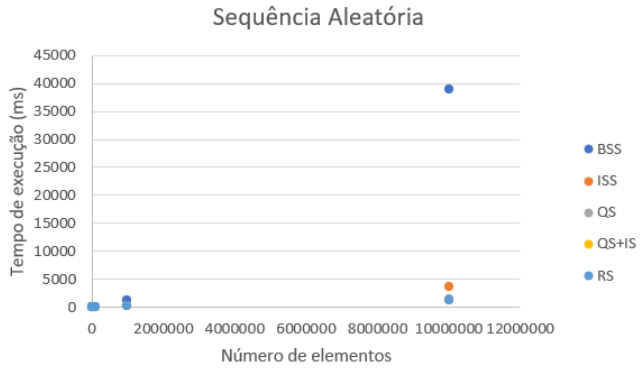


Gráfico para SEQ_ORDENADA_DECRESCENTE

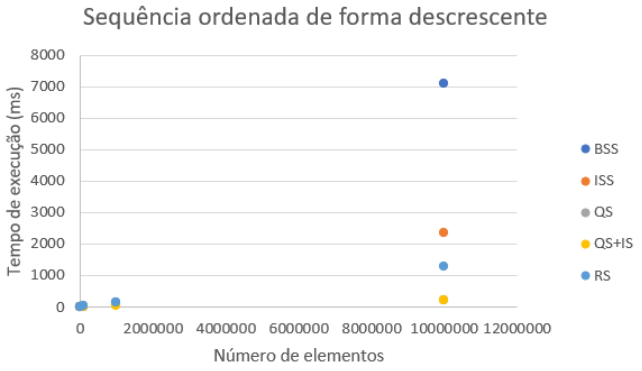


Gráfico para SEQ_QUASE_ORDENADA_1%

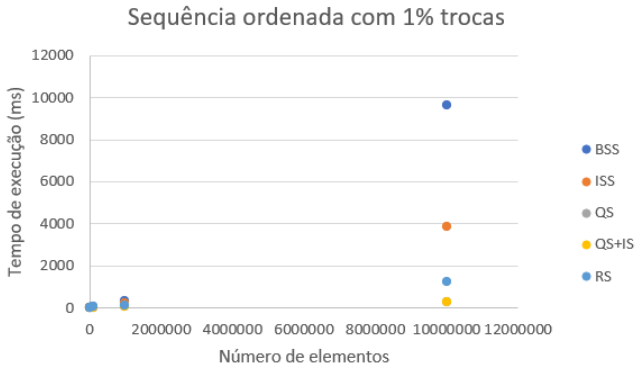
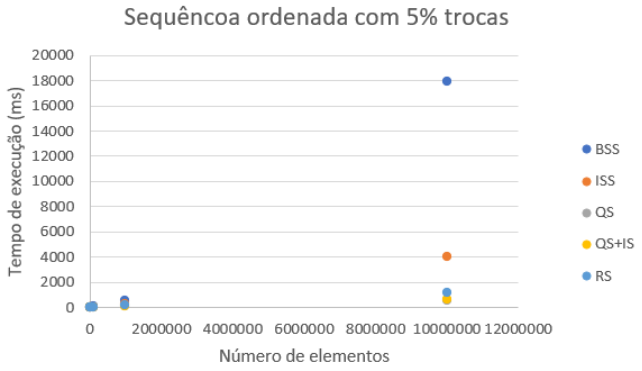


Gráfico para SEQ_QUASE_ORDENADA_5%



Sequência de incremento ou regra de incremento do I-SS para cada tipo de sequência:

I-SS (A108870): (1/5 * (9 * (9/4)^(k-1) - 4)).

Análise dos resultados considerando também a complexidade espacial do algoritmo:

Através da análise dos resultados obtidos podemos concluir que ambos os Shell Sort têm um desempenho bastante mau quando comparados com os outros algoritmos. Isto deve-se em parte à complexidade do BSS (O(n^2)) e do ISS (O(n^(4/3))), complexidades piores relativamente ao outros algoritmos de ordenação. De seguida o Radix Sort aparece como o algoritmo intermédio em termos de desempenho, o que é justificado pelos tamanhos de arrays usados aquando do estudo (inferiores a 10 milhões); este algoritmo apresenta grandes melhorias para arrays com um tamanho grande (superior a 100 milhões), passando assim a ser o algoritmo mais eficiente. Em termos de complexidade o Radix Sort tem complexidade linear, enquanto ambos os Quick Sorts (QS e QS+IS) tem complexidade n x log(n) no caso médio. Comparados entre si, os Quick Sorts apresentam resultados bastante semelhantes.