Relatório Projeto 4.2 AED 2020/2021

Nome: Edgar Filipe Ferreira Duarte Nº Estudante:2019216077
TP (inscrição): PL9 *Login* no *Mooshak:AED 2019216077*
Nº de horas de trabalho: 6 *H* *Aulas Práticas de Laboratório:* 2 *H* *Fora de Sala de Aula:* 4 *H*

(A Preencher pelo Docente) CLASSIFICAÇÃO:

Comentários:

Registar os tempos computacionais do QS e das 4 variantes selecionadas do QS+IS 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. Devem apresentar e discutir as regressões para a melhor variante em cada tipo de sequência.

Gráfico para SEQ_ALEATORIA

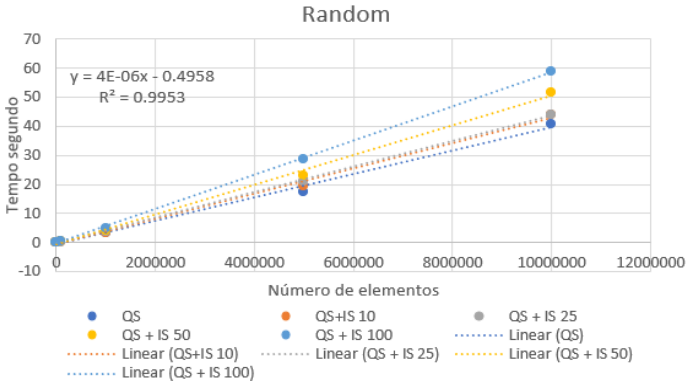


Gráfico para SEQ_ORDENADA_DECRESCENTE

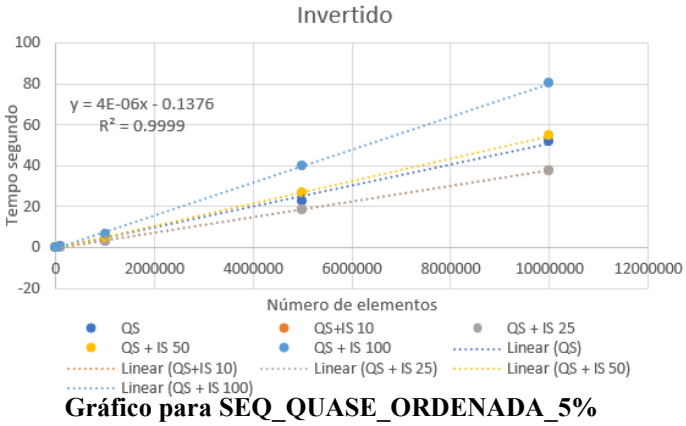


Gráfico para SEQ_QUASE_ORDENADA_1%

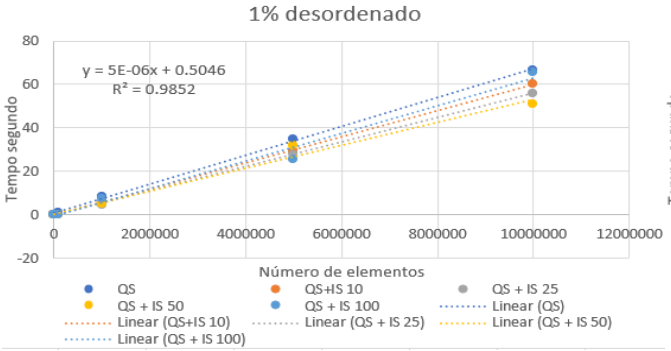
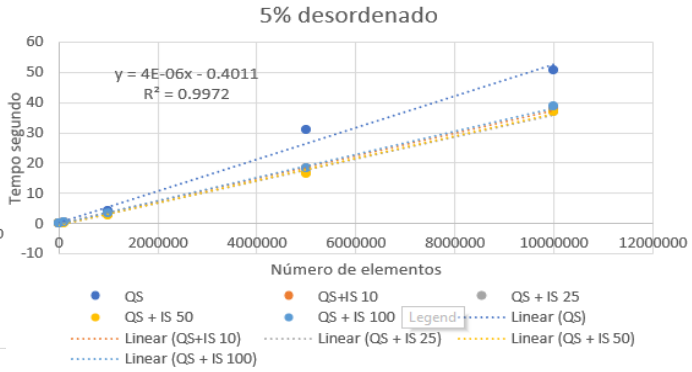


Gráfico para SEQ_QUASE_ORDENADA_5%



Análise dos resultados:

A partir dos resultados obtidos, é possível concluir que os algoritmos QuickSort + Insertion Sort, de um modo geral, não oferecem um ganho de eficiência significativo, relativamente ao QuickSort base. De entre os vários algoritmos desenvolvidos, não existe um vencedor claro. Todos têm complexidade temporal O(n * log(n)). Para uma sequência aleatória o melhor algoritmo foi o QuickSort base, para uma sequência inversa foi o QS+IS 25 e para os restantes foi o QS+IS 50. O algoritmo mais estável é o QS+IS 25 que é ou está perto de ser o melhor algoritmo em todas as situações. Isto significa que, quando as sequências chegam a este

tamanho, o QuickSort já as ordenou para que a utilização do InsertionSort seja vantajosa. Para valores de transição muito superiores a 25 (na tabela o QS + IS 50 e o QS + IS 100) verificou-se uma perda de eficiência (mais notável no QS + IS 100), pois, nestas situações o QuickSort ainda não ordenou muito a sequência inicial, logo o insertionSort tem de fazer muitas trocas, perdendo tempo.