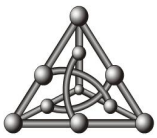

UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
Faculdade de Computação

Implementação Algorítmica

Atividade 2 — Ordenação

1 Descrição

Ordenação é uma operação básica em Computação. Diversos algoritmos de ordenação têm sido propostos ao longo da história, mesmo quando computadores ainda não existiam. A partir de meados do século passado, os algoritmos começaram a ser transformados em programas e executados em computadores. Esta atividade pede que você implemente alguns dos algoritmos de ordenação mais conhecidos e realize experimentos com os mesmos.

Em particular, você deve implementar os algoritmos da tabela abaixo, de acordo com as possibilidades de escolhas correspondentes.

Algoritmos	**Escolhas**
INSERTIONSORT, BUBBLESORT, SELECTIONSORT	2
SHELLSORT	1
MERGESORT, HEAPSORT, QUICKSORT, RANDOMIZED-QUICKSORT	2
RADIXSORT, BUCKETSORT	1
Total	6

Dessa forma, você deve implementar 6 algoritmos de ordenação listados na tabela acima, com alguma liberdade. Na primeira linha e primeira coluna da tabela estão listados três nomes de algoritmos e na segunda coluna da mesma linha, a quantidade de algoritmos desse conjunto que você deve implementar. Por exemplo, você pode escolher implementar o SELECTIONSORT e o INSERTIONSORT desse conjunto. O mesmo vale para a terceira linha. Observe que a segunda linha mostra que você não tem escolha e você deve implementar o SHELLSORT.

2 Programa, entrada e saída

Você deve implementar os algoritmos conforme as regras descritas na seção 1, desenvolvendo um ou mais programas. Você deve construir conjuntos de dados de entrada com aleatorização. Cada conjunto de entrada A deve conter números inteiros não negativos escolhidos (pseudo)aleatoriamente. Um tal conjunto de números A deve ter n números. Os valores de n variam, para cada conjunto construído, de acordo com os seguintes parâmetros:

- **`inc`** é o valor inicial;
- **`max`** é o valor final; e
- **`stp`** é o intervalo entre dois tamanhos.

Por exemplo, se **inc** $= 100$, **max** $= 1000$ e **stp** $= 10$, então os tamanhos dos vetores de entrada A que devem ser construídos são $n = 100, 110, 120, \ldots, 990, 1000$. O valor máximo possível depende de diversos aspectos. Nesta descrição, o valor máximo é 20000, mas esta não é uma exigência. Na verdade, quanto maior esse valor máximo, melhor para os experimentos. Você deve informar ao usuário do seu programa qual o valor máximo que ele(a) pode usar na execução de seu programa.

Um conjunto A com n elementos assim gerado é chamado de **caso de teste**. Para cada caso de teste, você deve executar os seis algoritmos mencionados na Seção 1 para ordenar esse conjunto A de n números inteiros. Para cada n, você deve repetir o processo acima um determinado número **rpt** de vezes, obtendo então a média dos tempos medidos.

A saída consiste de uma primeira linha contendo os rótulos da quantidade de elementos n e a identificação de cada um dos algoritmos. Cada linha a seguir contém o valor da quantidade de elementos n de um caso de teste e as médias dos tempos gastos da execução de cada algoritmo.

2.1 Exemplo de entrada e saída

Um exemplo, para casos de teste com n variando de acordo com os parâmetros **inc** $= 1000$, **max** $= 20000$ e **stp** $= 1000$, é mostrado a seguir. O parâmetro relativo ao número de repetições foi adotado como **rpt** $= 10$. As médias dos tempos de execução dos algoritmos são mostradas em segundos.

n	Bubble	Insertion	Shell	Merge	Quick	Bucket
1000	0.003574	0.001138	0.000137	0.000155	0.000153	0.000159
2000	0.011531	0.003079	0.000262	0.000276	0.000277	0.000250
3000	0.027635	0.007076	0.000425	0.000439	0.000445	0.000420
4000	0.050160	0.012447	0.000590	0.000574	0.000588	0.000531
5000	0.084190	0.019866	0.000788	0.000758	0.000770	0.000721
6000	0.123435	0.028035	0.000963	0.000915	0.000928	0.000928
7000	0.168734	0.037639	0.001133	0.001096	0.001094	0.000994
8000	0.228222	0.050131	0.001347	0.001252	0.001270	0.001209
9000	0.289802	0.063790	0.001572	0.001431	0.001466	0.001424
10000	0.359711	0.077567	0.001740	0.001584	0.001621	0.001603
11000	0.433337	0.092772	0.001917	0.001742	0.001805	0.001693
12000	0.520124	0.109963	0.002129	0.001898	0.001933	0.001755
13000	0.612092	0.129613	0.002319	0.002086	0.002132	0.001947
14000	0.715013	0.149777	0.002550	0.002264	0.002309	0.002091
15000	0.824394	0.172034	0.002833	0.002444	0.002506	0.002258
16000	0.941937	0.196605	0.003006	0.002620	0.002704	0.002416
17000	1.082374	0.223746	0.003246	0.002828	0.002852	0.002939
18000	1.200358	0.247536	0.003437	0.002984	0.003005	0.002695
19000	1.342269	0.277014	0.003641	0.003152	0.003213	0.002910
20000	1.486634	0.304419	0.003899	0.003351	0.003414	0.003116

3 Entrega

Instruções para entrega da sua atividade:

1. **O que entregar?**

   O arquivo a ser entregue deve conter o seguinte:

   - programa(s) desenvolvidos (e um arquivo **`Makefile`**, se for o caso), e
   - um gráfico no formato (**`.pdf`**) gerado a partir da tabela dos tempos de execução dos algoritmos. Um exemplo, usando o software **`gnuplot`**, é mostrado na Figura 1.

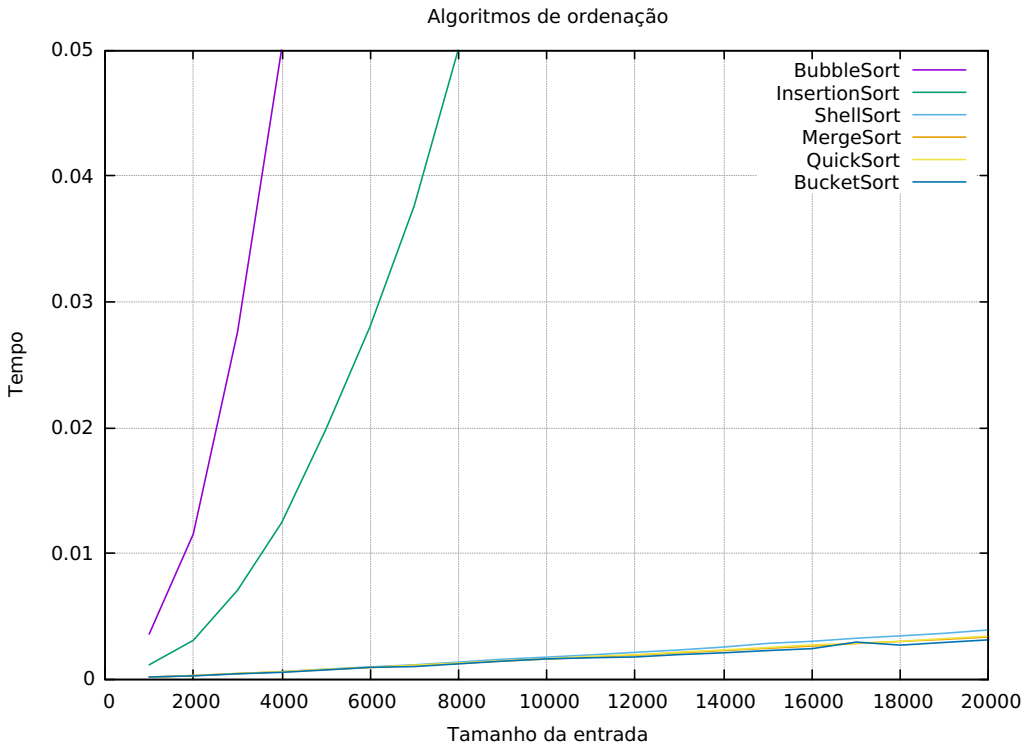

Figura 1: Execução dos algoritmos para os casos de teste especificados nesta atividade.

Compacte todos esses arquivos com o compactador de sua preferência e entregue um único arquivo (com extensão **`.tgz`**, **`.bz2`**, **`.zip`**, **`.rar`**, . . .).

2. **Forma de entrega**

   A entrega será realizada diretamente no Sistema (AVA/UFMS), na disciplina Implementação Algorítmica – T01. Após abrir uma sessão digitando seu *login* e sua senha, vá até a sessão "Atividades" e escolha o tópico "Atividade 2 – Ordenação", onde você poderá entregar sua atividade. Um fórum de discussão deste trabalho já se encontra aberto, além de outras informações adicionais. Você pode entregar o trabalho quantas vezes quiser até às **23 horas e 59 minutos** do dia **23 de setembro de 2021**. A última versão entregue é aquela que será corrigida. Encerrado o prazo, não serão mais aceitos trabalhos.

3. **Atrasos**

   Trabalhos atrasados não serão aceitos. Não deixe para entregar seu trabalho na última hora. Para prevenir imprevistos como queda de energia, problemas com o sistema, e/ou falha de conexão com a internet, sugerimos que a entrega do trabalho seja feita pelo menos um dia antes do prazo determinado.

4. **Erros**

   Trabalhos com erros de compilação/interpretação receberão nota ZERO. Faça todos os testes necessários para garantir que seu programa está livre de erros de compilação.

5. **Linguagem de programação e arquivo com o(s) programa(s)-fonte**

   Você pode escolher a sua linguagem de programação preferida para implementar esta atividade. Adicionalmente, fique atento(a) para que seu(s) arquivo(s) contendo o(s) programa(s)-fonte esteja(m) bem organizado(s). Um programa tem de ser muito bem compreendido por uma pessoa. Verifique se seu programa tem a indentação adequada, se não tem linhas muito longas, se tem variáveis com nomes significativos, entre outros. Não esqueça que um programa bem descrito e bem organizado é a chave de seu sucesso. E não esqueça da documentação de seu programa.

6. **Conduta Ética**

   O trabalho deve ser feito INDIVIDUALMENTE por cada grupo. Cada grupo tem responsabilidade sobre cópias de seu trabalho, mesmo que parciais. Não compartilhe seu programa ou trechos de seu programa. Você pode consultar seu(sua) colega de dupla para esclarecer dúvidas e discutir idéias sobre o trabalho, pode consultar o professor em uma chamada virtual ou no fórum de discussão da disciplina, mas NÃO copie a atividade!

   Trabalhos considerados plagiados terão nota ZERO.