Escola de Artes, Ciências e Humanidades – EACH-USP

ACH2002 – Introdução à Ciência da Computação II
EACH – Segundo Semestre de 2008

Segundo Exercício-Programa — Data de entrega: até 10 de novembro de 2008.

Professor: Delano Medeiros Beder

1 Introdução

Neste segundo exercício-programa do semestre (EP2) o objetivo é criar um conjunto de classes para avaliar empiricamente algoritmos de ordenação para diferentes tipos de arranjos e de diferentes tamanhos. O EP2 é composto de três partes, sendo as duas primeiras obrigatórias e a terceira opcional.
Um detalhe importante do EP2 é que ele não requer a implementação dos algoritmos de ordenação. Os algoritmos *InsertionSort*, *SelectionSort*, *BubbleSort*, *MergeSort*, *QuickSort*, *QuickSort Aleatório* e *HeapSort* já estão implementados (arquivo `Ordenacao.java` dado).

1.1 Primeira Parte — Estrutura de Classes

O programa deve gerar arranjos de inteiros que serão ordenados pelos algoritmos de ordenação acima.

Classe GeradorDeArranjos

Esta classe deverá conter métodos para gerar arranjos de diferentes tipos, a saber:

1. Arranjos de elementos aleatórios: todos os elementos A[i], $0 \leq i < A.length$, do arranjo A gerado contêm valores aleatórios.
2. Arranjos de elementos *misturados crescentemente*: o arranjo gerado deve possuir elementos organizados da seguinte forma: os elementos pares A[0], A[2], A[4]... possuem valores tais que $A[0] \leq A[2] \leq A[4]$...; os elementos ímpares A[1], A[3], A[5],... são aleatórios.
3. Arranjos de elementos *misturados decrescentemente*: o arranjo gerado deve possuir elementos organizados da seguinte forma: os elementos pares A[0], A[2], A[4]... possuem valores tais que $A[0] \geq A[2] \geq A[4]$...; os elementos ímpares A[1], A[3], A[5],... são aleatórios.
4. Arranjos de elementos *misturados*: o arranjo gerado deve possuir elementos organizados da seguinte forma: os elementos pares A[0], A[2], A[4]... possuem valores tais que $A[0] \geq A[2] \geq A[4]$...; os elementos ímpares A[1], A[3], A[5],... possuem valores tais que $A[1] \leq A[3] \leq A[5]$...
5. Arranjos de elementos ordenados crescentemente: o arranjo A gerado é tal que os valores dos elementos A[i] estão ordenados crescentemente.
6. Arranjos de elementos ordenados decrescentemente: o arranjo A gerado é tal que os valores dos elementos A[i] estão ordenados decrescentemente.
7. Arranjos com elementos iguais: O arranjo A gerado é tal que os valores dos elementos A[i] são todos iguais.

Os elementos dos arranjos A gerados são tais que $0 \leq A[i] < 30.000$ onde $0 \leq i < A.length$. Os métodos que geram os diferentes arranjos descritos acima devem receber como parâmetro o tamanho do arranjo a ser gerado e retornar uma referência para o arranjo A gerado.

Classe Avaliação

A classe Avaliação é a responsável pela análise empírica dos algoritmos de ordenação. Cada objeto desta classe representa a avaliação de determinados algoritmos de ordenação, utilizando tamanhos de arranjos pré-estabelecidos. Os métodos desta classe são os seguintes:

int ajustaAvaliacaoAlgoritmoXXX() — indica que a avaliação a ser realizada irá utilizar o algoritmo XXX que pode ser: *InsertionSort*, *SelectionSort*, *BubbleSort*, *MergeSort*, *QuickSort*, *QuickSort Aleatório* e *HeapSort*. Retorna o número de algoritmos já ajustados.

int ajustaArranjoTipoYYY() — indica que a avaliação a ser realizada utilizará um arranjo do tipo YYY (aleatório, misturado crescentemente, misturado crescentemente, ordenado crescentemente, ordenado decrescentemente, iguais). Retorna o número de tipos já ajustados.

int ajustaTamanhoArranjos(int tamanho) — este método permite que se defina o tamanho dos arranjos a serem utilizados durante a avaliação. É possível ajustar até 10 tamanhos diferentes. Retorna o número de tamanhos já ajustados.

void executaAvaliacao() — este método é responsável pela execução dos algoritmos selecionados sobre os arranjos estabelecidos e salva os resultados (tempo de execução em milissegundos).

void imprimeResultadoTabela() — imprime em formato tabela os tempos em milissegundos dos testes utilizando os tipos de arranjos, os algoritmos e os tamanhos dos arranjos ajustados previamente.

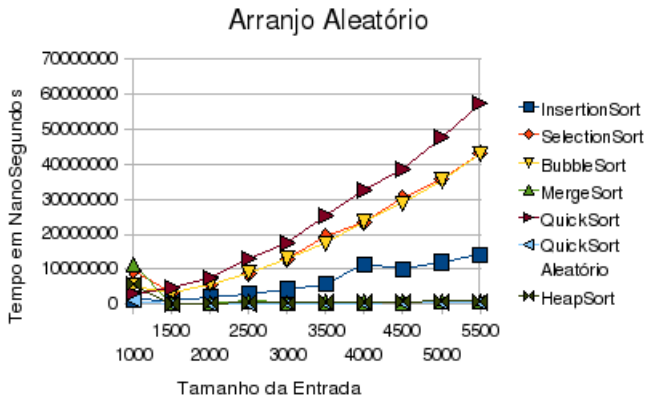
Segunda Parte — Interface e análise

Crie, utilizando um método `main`, uma interface textual (menu de opções) que permite ao usuário selecionar os tipos de arranjos, os tipos de algoritmos e os tamanhos de arranjos para as avaliações desejadas. Além da interface, você deverá fazer experimentos com seu programa e elaborar um relatório.
Neste relatório você deverá analisar criticamente os diversos algoritmos de ordenação apresentados em classe em função do tipo dos arranjos utilizados e também em função dos tamanhos desses arranjos. Crie uma tabela relacionando tipos de arranjos, algoritmos, tamanhos dos arranjos e tempos obtidos (em milissegundos ou nanossegundos). Esta tabela pode ser gerada utilizando o OpenOffice ou o Office da MicroSoft.

Arranjo Aleatório

	Tamanho dos Arranjos									
Algoritmo	1000	1500	2000	2500	3000	3500	4000	4500	5000	5500
InsertionSort	1506415	1124311	1951375	2958493	4386056	5863766	11398159	10019484	11983083	14181698
SelectionSort	9628231	3271943	5803423	8956145	12836408	19359479	23488797	30550767	35633144	43337657
BubbleSort	5225483	3199098	5871239	9064260	12866999	17409290	23385850	28819112	35450927	42924264
MergeSort	11431333	294033	308630	880633	479882	570327	647152	742766	853046	916322
QuickSort	3077922	4858186	7647871	13124086	17505113	25317950	32693928	38593452	47690398	57390566
QuickSort Al.	1205468	151836	193950	259951	305767	356961	404453	464866	516340	573470
HeapSort	5860903	233691	321621	573610	500066	595749	692201	782785	878817	978831

Os tempos em milissegundos (ou nanosegundos) dos testes utilizando os tipos de arranjos, os algoritmos, os tamanhos dos arranjos ajustados previamente devem ser plotados, seguidos de uma discussão crítica.



Observação: Caso algum algoritmo não possa ser executado com algum valor (por exemplo, por estouro de memória), indique que o algoritmo não processou.
Dicas:

- Utilize seus conhecimentos de TADI para elaborar a tabela ou veja o livro do Ziviani [2] (páginas 115 a 118) para ter algumas idéias.
- Para obter números pseudoaleatórios utilize o método java.lang.Math.random().
- Para obter os tempos de execução verifique os métodos nanoTime() e currentTimeMillis() da classe System (http://java.sun.com/j2se/1.5.0/docs/api/java/lang/System.html).

Terceira Parte — Visualização Gráfica (totalmente opcional)

Se você já terminou a segunda parte e está sedento por um desafio, pode tentar implementar esta parte para conseguir um pontinho extra (o EP2 passa então a valer 11 pontos).
A idéia aqui é, ao invés de implementar a interface e mostrar os resultados em uma interface textual, utilizar uma interface gráfica. Seu programa deverá abrir uma nova janela na qual o usuário poderá fazer os ajustes da avaliação desejada e o resultado deve ser gerado em outra janela.
ATENÇÃO: *Para fazer esta terceira parte, você terá que usar várias classes de Java que não são ensinadas em ACH2002. Você terá que aprendê-las sozinho, por sua conta e risco!*
Para criar a interface gráfica, você deverá usar um biblioteca de Java chamada *Swing*. Algumas fontes sobre o *Swing*:

1. livro do Camarão [1], no Capítulo 11, há uma introdução ao *Swing*;
2. tutorial sobre *Swing* em `http://java.sun.com/docs/books/tutorial/uiswing/learn`.

Critérios importantes

Sobre a elaboração:

- Este exercício-programa deve ser elaborado individualmente.
- Guarde uma cópia do seu programa e relatório entregues.

Sobre a avaliação:

- Não serão toleradas cópias! Exercícios copiados (com ou sem eventuais disfarces) receberão nota ZERO. O exercício do aluno alvo da cópia também receberá nota ZERO.
- Exercícios com erros de sintaxe (ou seja, erros de compilação) receberão nota ZERO. Exercícios sem método `main` para a execução em linha de comando receberão nota ZERO.
- É muito importante que seu programa tenha comentários e esteja bem identado, ou seja, elaborado de maneira a ressaltar a estrutura de subordinação dos comandos do programa (conforme visto em aula). A qualidade do seu trabalho sob esse ponto de vista influenciará sua nota!
- As informações impressas pelo seu programa na tela devem aparecer da forma mais clara possível. Este aspecto também será levado em consideração no cômputo de sua nota.
- Uma regra básica é a seguinte: do ponto de vista do monitor responsável pela correção dos trabalhos, quanto mais convenientemente apresentado estiver o seu programa, melhor será a disposição dele em atribuir-lhe uma nota generosa.

Sobre a entrega:

- O prazo de entrega é o dia 10 de novembro 2008 às 24h.
- No início do arquivo, acrescente um cabeçalho bem informativo, como o seguinte:

```
/*****************************************************************/
/**   ACH2002  - Introdução à Computação II                     **/
/**   EACH-USP - Segundo Semestre de 2008                       **/
/**   <turma> - <nome do professor>                             **/
/**                                                             **/
/**   Segundo Exercício-Programa  --  Avaliação de Algoritmos   **/
/**   Arquivo: <nome do arquivo>                                **/
/**                                                             **/
/**   <nome do(a) aluno(a)>               <número USP>          **/
/**                                                             **/
/**   <data de entrega>                                         **/
/*****************************************************************/
```

  Não é obrigatório que o cabeçalho seja idêntico a esse, apenas que contenha pelo menos estas informações.

- A entrega será feita unicamente pelo CoL. Não serão aceitos trabalhos enviados por email. Fiquem atentos, pois agora está ativada no CoL a funcionalidade "horário limite para entrega de trabalhos".
- O seu EP2 (Exercício Programa) deve estar todo contido em único arquivo .java. e o relatório em um arquivo .pdf. Os nomes dos arquivos a ser entregue deve possuir o seguinte formato: avaliacao<Número USP>.java e avaliacao<Número USP>.pdf, ambos compactados em um arquivo avaliacao<Número USP>.zip. Por exemplo: avaliacao56777333.zip.

Referências

[1] Carlos Camarão & Lucília Figueiredo. *Programação de Computadores em Java*. LTC Editora, Rio de Janeiro, 2003.

[2] Nívio Ziviani. *Projeto de Algoritmos - com implementação em C e Pascal*. Thomson Editora, 2a edição, 2004.