Trabalho sobre Métodos de Pesquisa e Ordenação em Java

Objetivos

- Avaliar competências e conhecimentos desenvolvidos na disciplina de Pesquisa e Ordenação, por meio de um trabalho experimental.
- Oportunizar o desenvolvimento e escrita de uma atividade nos moldes de um trabalho acadêmico, bem como aperfeiçoar a capacidade de criar, planejar, trabalhar e decidir em grupo de forma cooperativa.
- Aplicar conceitos de POO no desenvolvimento de um programa.
- **Aprimorar-se na programação Java, sendo que os grupos formados apenas por alunos de engenharia podem fazer o trabalho em C.**

Informações gerais

- Data limite para entrega: **a informar** pelo Envio de Trabalho do AVA (o sistema zipado, com todos os arquivos gerados e o texto escrito).
- **O grupo perde 1 ponto por cada aula de atraso na entrega, limitado a duas aulas.**
- Grupos formados por até 4 alunos, sendo que qualquer componente do grupo deve ser capaz de apresentar/defender todas as ideias contidas no trabalho. É fundamental que o trabalho seja feito de forma cooperativa, e não, simplesmente, que cada componente cuide de uma parte do mesmo para depois juntar e formar o todo.
- A parte escrita deverá ser feita de forma bastante sucinta e deverá estar nos padrões adotados pela FAESA para apresentação de relatório de pesquisa.

Critérios avaliativos

- Organização do documento impresso e formatação segundo as normas de trabalho acadêmico.
- Abordagem do tema central de forma clara e precisa.
- Utilização adequada de estruturas de dados e métodos de pesquisa e ordenação.
- **Não usar estruturas prontas do JAVA (não usar arraylist, linkedlist, Hashmap, etc)**
- Programa correto.
- Legibilidade do código e identação.
- Entrevista individual e/ou coletiva.
- Trabalho em equipe e de forma cooperativa.
- Pontualidade na entrega (a cada dia de atraso, o trabalho valerá 1,0 ponto a menos).

Texto nas normas da FAESA	1,0
Descrição dos métodos e conclusões	1,0
Implementação dos métodos	4,0
Apresentação	2,0
TOTAL	8,0

Descrição geral do trabalho

O tema da pesquisa é o estudo de métodos de pesquisa e ordenação num contexto de programação orientada a objetos.

Organização da parte escrita

Introdução:
O capítulo da introdução deve descrever brevemente cada um dos métodos utilizados no trabalho e a metodologia utilizada na implementação destes métodos.

Desenvolvimento:
Implementar os métodos: **HeapSort + Pesquisa Binária**, **QuickSort + Pesquisa Binária**, **Árvore Binária de Busca Balanceada**, **Árvore AVL**, **Hashing com Vetor Encadeado**. Testar todos esses métodos e fazer um quadro de comparação entre eles.
Descrever a máquina em que o programa for rodado. Descrever os arquivos usados para teste e colocar as tabelas com os resultados.

Conclusão:
Apresentar as conclusões geradas a partir da análise dos quadros comparativos de tempo e verificar se elas são compatíveis ou não com a teoria. Se não for compatível, investigue as causas.

Bibliografia:
Listar os títulos que auxiliaram o desenvolvimento do trabalho, segundo as normas da ABNT.

Problema a ser implementado

Serão disponibilizados 15 arquivos do tipo texto, contendo 500, 1.000, 5.000, 10.000 e 50.000 registros, dispostos de forma aleatória, ordenada e invertida.

Os registros representarão um cadastro imobiliário na prefeitura de um município e serão compostos dos campos cpf, inscrição imobiliária, valor do imposto devido e se o proprietário pagou ou não. O formato do arquivo será

cpf;inscrição;valor;pago

por exemplo:

11122233344;12345678;2380.00;true

O trabalho consiste em:

1) Comece a contar o tempo.

2) Carregue o vetor com o arquivo de 500 elementos aleatórios.

3) Use o método **HeapSort** para ordenar os registros pelo CPF. Se tiver CPFs iguais, ordene pelo número de inscrição. Grave o resultado em um arquivo com o nome contendo o método **(HEAP)+tipo(ALEA ou ORD ou INV)+ tamanho.txt.** Por exemplo: **HeapAlea500.txt**)

4) Use **Pesquisa Binária** para pesquisar 400 registros pelo **cpf**. Estes registros estarão em um arquivo que será fornecido. Ao final, deve gerar um outro arquivo onde, para cada cpf que for encontrado será gravado a inscrição dos imóveis, o valor do imposto, se ele foi pago ou não. Some todos os impostos não pagos e mostre no final a soma do imposto não pago. Se o cpf não for encontrado, deve-se gravar uma mensagem de NÃO HÁ NENHUM REGISTRO COM O CPF (colocar o cpf). Veja o exemplo abaixo:

CPF 11122233344:
Inscr: 12346578 Imposto: 2380.00 PAGO
Inscr: 12346579 Imposto: 2500.00 NÃO PAGO
Inscr: 12346582 Imposto: 1200.00 NÃO PAGO
Total Imposto a pagar: 3700.00

CPF 12345612344:
NÃO HÁ NENHUM REGISTRO COM O CPF 12345612344

5) Repita 4 vezes o processo de 2 a 4. **Você deve rodar o processo 5 vezes no total.** Os arquivos gerados podem ser regravados (por cima), ou seja, no final você terá apenas um arquivo para cada tamanho, tipo e método. Por exemplo HeapPesqAlea500.txt.

6) Termine de contar o tempo, faça uma média e armazene este resultado.

7) **Faça os itens de 2 a 6 para cada um dos tamanhos (500, 1000, 5000, 10000 e 50000), para cada tipo de arquivo (aleatório, ordenado e invertido) e para cada método (HeapSort + Pesquisa Binária e QuickSort + Pesquisa Binária)**. Ao todo, o programa rodará 15vezes com Heap + PesqBin e mais 15 vezes com Quick + PesqBin.

Após esse processo, rodaremos as árvores:
Para cada um dos tamanhos e tipos de arquivo (aleatório, ordenado e invertido):

8) Comece a contar o tempo.

9) Carregue cada um dos arquivos de registros imobiliários em uma ABB não balanceada. Balanceie a árvore. **Cuidado com os cpfs iguais.**

10) Faça a pesquisa ABB, usando os 400 CPFs fornecidos pela professora, nos mesmos moldes do item (4). Não esqueça de gravar os resultados em arquivos.

11) Repita 4 vezes os processos 9 e 10

12) Termine de contar o tempo, faça uma média e armazene este resultado.

13) **Faça os itens de 8 a 12 para cada um dos tamanhos (500, 1000, 5000, 10000 e 50000), para cada tipo de arquivo (aleatório, ordenado e invertido) para a ABB e AVL**.

Em seguido, rodaremos o Hashing Encadeado:
Para cada um dos tamanhos e tipos de arquivo (aleatório, ordenado e invertido):

14) Comece a contar o tempo.

15) Carregue cada um dos arquivos de registros imobiliários, tendo como chave o cpf, em um Hashing Encadeado.

16) Faça a pesquisa, usando as 400 CPFs fornecidos pela professora, nos mesmos moldes do item (4). Não esqueça de gravar os resultados em arquivos.

17) Repita 4 vezes os processos 15 e 16

19) Termine de contar o tempo, faça uma média e armazene este resultado. Lembrando que isso deve ser feito em todos os 15 arquivos de registros imobiliários

20) Compare os tempos de todos os algoritmos em cada tamanho e tipo de arquivo e gere conclusões.

Obs.:
1) Para computar o tempo, utilize o método **`System.currentTimeMillis()`** que retorna o tempo da máquina em milissegundos, ou **`System.nanoTime()`** que retorna o tempo da máquina em nanossegundos.

2) Ao rodar o método HeapSort, você deverá gerar e gravar 15 arquivos ordenados, cada um representando a ordenação de um arquivo dado pela professora. Logo depois, ao rodar a Pesquisa Binária, você deverá gerar e gravar mais 15 arquivos com os resultados das pesquisas.

3) Idem ao rodar o QuickSort

4) Para as árvores e o hashing serão gerados somente 15 arquivos com os resultados das pesquisas.

4) Quando for trabalhar com árvores, armazenando chaves secundárias, você deve pensar em um estratégia para armazenar e depois pesquisar todas as ocorrências de chaves iguais.