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.

Informações gerais

- Data limite para entrega: **21/11** (quinta-feira) 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 de um banco e serão compostos dos campos agência, número da conta (conta corrente começa com 001 e poupança começa com 002), CPF do 1º titular e saldo. O formato do arquivo será

agência;número;saldo;cpf do titular

O trabalho consiste em:

Faça os itens de 1 a 6 para cada um dos tamanhos (500, 1000, 5000, 10000 e 50000), para cada tipo de arquivo (aleatório, ordenado e invertido), usando Heapsort + Pesquisa Binária. Ao todo serão gerados 15 arquivos.

1) Comece a contar o tempo.

2) Carregue o vetor com um dos arquivos de registros.

3) Use o método **HeapSort** para ordenar por CPF, se houver mais de um CPF igual, eles serão ordenados pelo número da conta. Grave o resultado em um arquivo com o nome contendo o método **(HEAP)+tipo(ALEA ou ORD ou INV)+ tamanho.txt.**

4) Use **a Pesquisa Binária** para pesquisar 400 registros pelo CPF. Estes registros estarão em um arquivo que será fornecido pela professora. Ao final, o grupo deve gerar um outro arquivo onde, para cada CPF que for encontrado, será gravado a agência, o numero da conta (que deve ser separado e identificado como conta corrente e poupança) e o saldo e, ao final, o saldo total daquele CPF. Se o CPF não for encontrado deve-se gravar uma mensagem de NÃO HÁ CLIENTE COM O CPF (colocar o número do CPF). Veja o exemplo abaixo:

CPF 11122233345:
agencia 123 Conta Corrente 001123123 Saldo: -200,00
agencia 223 Conta Corrente 001223222 Saldo: 500,00
agencia 123 Conta Poupança 00212311 Saldo: 5000,00
Saldo total: 5300,00

CPF 11111222245:
NÃO HÁ CLIENTE COM O CPF 11111222245

5) Repita 4 vezes o processo 2 a 4. **Você deve rodar o processo 5 vezes no total.** Os arquivos gerados podem ser regravados, 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.

Faça os itens de 7 a 12 para cada um dos tamanhos (500, 1000, 5000, 10000 e 50000), para cada tipo de arquivo (aleatório, ordenado e invertido), usando Heapsort + Pesquisa Binária. Ao todo serão gerados 15 arquivos.

7) Comece a contar o tempo.

8) Carregue o vetor com um dos arquivos de registros.

9) Use o método **QuickSort** para ordenar por CPF, se houver mais de um CPF igual, eles serão ordenados pelo número da conta. Grave o resultado em um arquivo com o nome contendo o método **(QUICK)+tipo(ALEA ou ORD ou INV)+ tamanho.txt.**

10) Faça a pesquisa binária, usando os 400 registros fornecidos pela professora, gerando arquivos no mesmo modelo do item (3).

11) Repita 4 vezes os processos 8 a 10

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

Após esse processo, rodaremos as árvores:
Faça os itens de 13 a 17 para cada um dos tamanhos (500, 1000, 5000, 10000 e 50000), para cada tipo de arquivo (aleatório, ordenado e invertido), usando ABB balanceada .

13) Comece a contar o tempo.

14) Carregue uma ABB com os registros de um dos arquivos, tendo como chave o CPF.

Balanceie a árvore. **Cuidado com os CPFs iguais: vocês devem pensar em uma maneira eficiente de armazenar os CPFs iguais**

15) Faça a pesquisa na ABB, usando os 400 registros fornecidos pela professora, gerando arquivos no mesmo modelo do item (3).

16) Repita 4 vezes os processos 14 e 15

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

Faça os itens de 18 a 22 para cada um dos tamanhos (500, 1000, 5000, 10000 e 50000), para cada tipo de arquivo (aleatório, ordenado e invertido), usando AVL.

18) Comece a contar o tempo.

19) Carregue uma AVL com os registros de um dos arquivos, tendo como chave o CPF. Balanceie a árvore. **Cuidado com os CPFs iguais: vocês devem pensar em uma maneira eficiente de armazenar os CPFs iguais**

20) Faça a pesquisa na AVL, usando os 400 registros fornecidos pela professora, gerando arquivos no mesmo modelo do item (3).

21) Repita 4 vezes os processos 19 e 20

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

Faça os itens de 23 a 27 para cada um dos tamanhos (500, 1000, 5000, 10000 e 50000), para cada tipo de arquivo (aleatório, ordenado e invertido), usando Hashing.

23) Comece a contar o tempo.

24) Carregue um dos arquivos de registro, tendo como chave o CPF, em um Hashing Encadeado.

25) Faça a pesquisa, usando as 400 registros fornecidos pela professora, nos mesmos moldes do item (3). Não esqueça de gravar os resultados em arquivos.

26) Repita 4 vezes os processos 24 e 25

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

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 e o QuickSort, você deverá gerar e gravar, para cada um, 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 15 arquivos com os resultados das pesquisas.

3) Ao rodar as árvores e o hashing, você deverá gerar e gravar 15 arquivos apenas 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.