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: **13/06** (segunda-feira) NO HORÁRIO DA AULA. Não serão aceitos trabalhos fora do horário de aula.
- **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 entrega do trabalho deve ser feita por e-mail (programa – cinthia.cristina@faesa.br) e papel (conclusões).
- 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).

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 em site de busca de emprego e serão compostos dos campos nome do cargo, nome da empresa e sigla do Estado em que o cargo está sendo oferecido. O formato do arquivo será

cargo;empresa;sigla

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 cargo. Se tiver cargos iguais, ordene pelo Estado e depois pela empresa. Grave o resultado da ordenação em um arquivo. (O nome do arquivo pode ser HeapAlea500.txt)

4) Use **Pesquisa Binária** para pesquisar 400 registros contendo **o cargo e o estado**. Estes registros estarão em um arquivo que será fornecido. Ao final, deve gerar um outro arquivo onde, para cada cargo no estado específico, que for encontrado, será gravado o

nome das empresas que ofertaram esse cargo (essas empresas devem estar ordenadas). Se o cargo não for encontrado no estado indicado, deve-se gravar uma mensagem de NÃO HÁ NENHUM REGISTRO COM O CARGO (colocar o nome do cargo) NO ESTADO DO (colocar o nome do estado). Veja o exemplo abaixo:

Professor no Espirito Santo:
Colegio Faesa
IFES
UFES

Contador no Para:
NÃO HÁ NENHUM REGISTRO COM O CARGO Contador NO ESTADO DO Para

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, 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 30 vezes.

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 o arquivo de cargos, tendo como chaves o cargo e o estado, em uma ABB não balanceada. Balanceie a árvore. **Cuidado com os cargos e estados iguais.**

10) Faça a pesquisa ABB, usando os 400 registros 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 AVL**.

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

14) Comece a contar o tempo.

15) Carregue o arquivo de cargos, tendo como chave o cargo e o estado, em um Hashing Encadeado.

16) Faça a pesquisa, usando as 400 datas fornecidas 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.

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 15 arquivos com os resultados das pesquisas.

3) Idem ao rodar o QuickSort, as árvores e hashing.

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.