Universidade Presbiteriana Mackenzie

ESTRUTURA DE DADOS II

Aplicação 2 (Apl2) - Árvores BST e AVL
Atividade (grupos de no máximo 4 pessoas)

Objetivo

Desenvolver um programa para mapear os dados de um arquivo externo (memória secundária) para árvores BST e AVL (memória principal), realizar operações com os dados mapeados na memória principal e armazenar o conteúdo de volta para a memória secundária.

Instruções

- A atividade deve ser resolvida usando a linguagem Java.
- As árvores usadas na sua solução devem ser implementações de sua autoria (pode usar como base o conteúdo visto em aula), isto é, o projeto não deve usar estruturas relacionadas a árvores que são oferecidas pela linguagem Java nem bibliotecas de terceiros (soluções que usarem tais estruturas/bibliotecas serão desconsideradas – zero).
- Caso necessário, sua solução pode usar as estruturas de pilha, fila e/ou lista encadeada oferecidas pela linguagem Java.
- Inclua a identificação do grupo (nome completo e RA de cada integrante) no início de cada arquivo de código, como comentário.
- Inclua todas as referências (livros, artigos, sites, vídeos, entre outros) consultadas para solucionar a atividade como comentário no arquivo `.java` que contém a `main()`.

ESTRUTURA DE DADOS II

Problema

A empresa TC – Teóricos da Computação *criou um formato de arquivo texto chamado* ED2*, com o objetivo de usar esse novo formato em diversas aplicações.*

Quando os dados estão armazenados permanentemente em disco, a empresa quer usar o novo formato ED2 *para a representação dos dados.*

Na memória principal do computador, a empresa quer avaliar a viabilidade de se usar uma árvore BST e uma árvore AVL para representar os dados de um arquivo ED2*.*

Para isso, a empresa resolveu criar um hackaton: equipes formadas por até 3 pessoas desenvolverão um programa de computador em Java que demonstra, na prática, o uso do novo formato ED2 *e seu mapeamento para árvores BST e AVL, com base nos requisitos descritos neste documento.*

A partir dos critérios de avaliação que também se encontram neste documento, a empresa espera ter várias opções para selecionar a melhor forma de se implementar o mapeamento ED2 => *árvore e avaliar o uso do formato de arquivo* ED2 *com árvores BST e AVL.*

Para que as equipes entendam o novo formato de arquivo, a empresa elaborou um documento com a descrição do formato ED2 *(arquivo* `ed2_file_format.txt` *disponibilizado junto com este documento). É de extrema importância que este documento adicional seja consultado, para que o formato* ED2 *seja lido e validado corretamente.*

A empresa também disponibilizou um arquivo `lexer_parser.zip` *que contém o código de um projeto Java que implementa um tokenizer (lexer) e parser de exemplo. O código de exemplo não é a única solução possível para escrever um parser, mas pode ser um ponto de partida para as equipes realizarem a leitura e a validação de arquivos no formato* ED2*.*

Descrição do programa

O **funcionamento/comportamento do programa** deve atender aos seguintes requisitos:

0. O programa deve apresentar um menu de opções contendo 9 opções:

1. Carregar dados de um arquivo ED2
2. Buscar uma chave/escopo na árvore
3. Inserir uma chave/escopo na árvore
4. Alterar uma chave da árvore
5. Remover uma chave da árvore
6. Salvar dados para um arquivo
7. Exibir o conteúdo e as propriedades da árvore BST
8. Exibir o conteúdo e as propriedades da árvore AVL
9. Encerrar o programa

1. Carregar dados de um arquivo ED2

A pessoa usuária do programa informa o local e nome de um arquivo no formato ED2 que será lido e validado pelo programa.

O programa deve analisar o conteúdo do arquivo e, caso o arquivo seja um arquivo ED2 válido:
1. Os dados do arquivo devem ser mapeados para uma árvore BST.
2. Os dados do arquivo devem ser mapeados para uma árvore AVL.

Caso o arquivo informado não seja um arquivo ED2 válido, o programa deve exibir uma mensagem apropriada e nenhuma árvore deve ser construída.

Observação: As opções 2-8 só podem ser executadas quando um arquivo ED2 for carregado na memória. Esse estado deve ser verificado pelo programa e uma mensagem apropriada deve ser exibida quando as opções 2-8 não puderem ser executadas.

2. Buscar uma chave/escopo na árvore

A pessoa usuária do programa informa uma string que é usada como termo de busca para verificar se uma chave ou um escopo existe no arquivo ED2 indicado e carregado na opção 1.

Caso uma chave ou um escopo exista, o programa deve exibir todos os dados da chave/escopo e indicar: 1) Quantas comparações foram realizadas na árvore BST e quantas comparações foram realizadas na árvore AVL até encontrar a chave/escopo informada; 2) O nível em que a chave/escopo se encontra em cada árvore.

Caso não exista, o programa deve exibir uma mensagem apropriada.

Observação: "Todos os dados da chave/escopo" deve incluir, pelo menos: 1) O nome da chave/escopo; 2) O valor atribuído à chave (somente no caso de ser chave); 3) O tipo de dado (chave ou escopo); 4) O escopo onde a chave/escopo está localizada.

Observação 2: Como o formato ED2 permite identificadores de mesmo nome, a busca deve informar todos os identificadores encontrados nas árvores e, para cada identificador encontrado, deve-se exibir as informações indicadas anteriormente (quantidade de comparações, nível e dados).

Por exemplo, considerando um arquivo ED2 com o seguinte conteúdo:

```
xyz=123
xyz (
  xyz=abc
)
```

Podemos observar que no escopo `global` existe uma chave `xyz` e um escopo `xyz`. No escopo `xyz`, existe uma chave `xyz`. Assim, se a pessoa decide buscar pelo termo `xyz`, o programa deve apresentar três resultados da busca.

3. Inserir uma chave/escopo da árvore

A pessoa usuária do programa informa se quer inserir uma nova chave na árvore ou um novo escopo.

Em seguida, a pessoa informa todos os dados da nova chave/escopo, incluindo em qual escopo a nova chave/escopo deve ser inserida.

A nova chave/escopo só será inserida na árvore se a chave/escopo não existir (não é possível ter duplicatas na árvore).

O programa deve exibir uma mensagem apropriada (chave/escopo inserida com sucesso ou inserção não realizada).

Observação: Em um mesmo escopo, é possível ter uma chave e um escopo de mesmo nome.

4. Alterar uma chave da árvore

A pessoa usuária do programa informa uma string que é usada como termo de busca para acessar uma chave da árvore.

Caso a chave exista, a pessoa informa qual é o novo valor da chave e a chave é atualizada.

Caso não exista, o programa deve exibir uma mensagem apropriada.

Observação: Essa opção permite alterar *apenas* o valor de uma chave. Não é permitido alterar o nome/escopo de uma chave nem o nome de um escopo.

5. Remover uma chave da árvore

A pessoa usuária do programa informa uma string que é usada como termo de busca para remover uma chave da árvore.

Caso a chave exista, a chave é removida da árvore.

Caso não exista, o programa deve exibir uma mensagem apropriada.

Observação: Essa opção permite remover *apenas* chaves da árvore.

6. Salvar dados para um arquivo

A pessoa usuária do programa informa o nome do arquivo onde os dados devem ser salvos.

O programa deve lidar com os possíveis cenários de escrita do arquivo (por exemplo, acesso não permitido, sobrescrita de arquivo já existente, nome de arquivo inválido, etc.) e exibir mensagens apropriadas em cada cenário.

Observação: O arquivo salvo deve seguir o formato ED2.

7. Exibir o conteúdo e as propriedades da árvore BST

O programa exibe a árvore BST em pré-ordem, em ordem e em pós-ordem.

Observação: "Exibir a árvore" significa exibir todos os dados da árvore (grau, altura e quantidade de nós) e todos os dados dos nós (se é raiz/folha, grau, nível, altura e todos os dados das chaves/escopos).

8. Exibir o conteúdo e as propriedades da árvore AVL

O programa exibe a árvore AVL em pré-ordem, em ordem e em pós-ordem.

Observação: "Exibir a árvore" significa exibir todos os dados da árvore (grau, altura e quantidade de nós) e todos os dados dos nós (se é raiz/folha, grau, nível, altura e todos os dados das chaves/escopos).

9. Encerrar o programa

O programa é encerrado quando a pessoa usuária do programa escolher a opção 9.

10. Arquivos ED2 de teste

O grupo deve criar arquivos ED2 para testar o funcionamento do programa. Devem existir tanto arquivos ED2 válidos quanto inválidos (para testar a correta leitura e validação de arquivos ED2).

Entrega

Código:

Compacte o código-fonte (somente arquivos `*.java`) e os arquivos `*.ed2` de teste criados pelo grupo, no **formato `zip`**.

Atenção: O arquivo `zip` não deve conter arquivos intermediários e/ou pastas geradas pelo compilador/IDE (ex. arquivos `*.class`, etc.).

Prazo de entrega: **via link do Moodle até 28/05/2024 07:30**.

Questionário:

A explicação sobre a implementação e o funcionamento do programa será feita por meio de um questionário, a ser respondido **individualmente no dia 28/05/2024 em horário de aula**, sendo permitido consultar apenas o código-fonte do programa entregue via Moodle.

ESTRUTURA DE DADOS II

Critérios de avaliação

A nota da atividade é calculada de acordo com os critérios da tabela a seguir.

ITEM AVALIADO	PONTUAÇÃO MÁXIMA
1. Carregar dados de um arquivo ED2.	2,0
2. Buscar uma chave/escopo na árvore.	0,75
3. Inserir uma chave/escopo na árvore.	0,5
4. Alterar uma chave da árvore.	0,25
5. Remover uma chave da árvore.	0,5
6. Salvar dados para um arquivo.	1,0
7. Exibir o conteúdo e as propriedades da árvore BST.	0,25
8. Exibir o conteúdo e as propriedades da árvore AVL.	0,25
9. Arquivos ED2 de teste (pelo menos um arquivo bem formado e um com problemas).	0,5
10. Questionário individual.	4,0

Tabela 1 - Critérios de avaliação.

A tabela a seguir contém critérios de avaliação que podem **reduzir** a nota final da atividade.

ITEM INDESEJÁVEL	REDUÇÃO DE NOTA
O projeto é cópia de outro projeto.	Projeto é zerado
O projeto usa estruturas relacionadas a árvores que são oferecidas pela linguagem Java ou bibliotecas de terceiros.	Projeto é zerado
Há erros de compilação e/ou o programa trava/"quebra" durante a execução[1].	-1,0
Não há identificação do grupo (código-fonte). Não há indicação de referências (código-fonte). Arquivos enviados em formatos incorretos. Arquivos e/ou pastas intermediárias que são criadas no processo de compilação ou pela IDE foram enviadas junto com o código-fonte.	-1,0

Tabela 2 - Critérios de avaliação (redução de nota).

O código-fonte será compilado com o compilador `javac` (21.0.2) na plataforma Windows da seguinte forma:

```
> javac *.java -encoding utf8
```

O código compilado será executado com `java` (21.0.2) na plataforma Windows da seguinte forma:

```
> java <Classe>
```

Sendo que `<Classe>` deve ser substituído pelo nome da classe que contém o método `public static void main(String[] args)`.

[1] Sobre erros de compilação: considere apenas erros. Não há problema se o projeto tiver *warnings* (embora *warnings* podem avisar sobre possíveis travamentos em tempo de execução, como loop infinito, divisão por zero, etc.).