Universidade Federal Fluminense
Programa de Pós-graduação em Ciência da Computação
TIC10002 – Estruturas de Dados e Algoritmos (2017.1)
Aluno: Clerisson dos Santos e Silva

Trabalho de Implementação

Os dados de venda da equipe de vendas de uma empresa estão armazenados em um arquivo txt que obedece ao seguinte formato: filial, ano_mês, cod_vendedor, total_vendido. Faça um programa que carregue os dados do arquivo em estruturas de dados e que permita
responder expressões como:

1) *total de vendas das filiais com códigos entre 10 e 20*

2) *total de vendas das filiais com códigos entre 10 e 20 nos meses de Jan/17 até Jun/17*

3) *total de vendas de todas as filiais nos meses de Ago/17 até Out/17*

As estruturas de dados devem permitir acesso eficiente para todos os tipos de expressões e a conjunção de condições sobre filiais e datas devem ser realizadas por meio de operação de interseção de conjuntos. Cada resumo de venda (linha) do arquivo deve dar origem a somente uma instância (objeto) no programa. Utilize as estruturas de dados mais eficientes para cada tipo de pergunta.

A solução deste trabalho aqui apresentada divide-se em duas partes:

1. Descrição do funcionamento e motivação para a utilização das estruturas de dados apresentadas no algoritmo.
2. Algumas orientações para o funcionamento do programa

1- DESCRIÇÃO DO FUNCIONAMENTO E MOTIVAÇÃO PARA A UTILIZAÇÃO DAS ESTRUTURAS DE DADOS APRESENTADAS NO ALGORITMO:

Dado o enunciado do trabalho acima, as respostas às questões apresentadas foram obtidas utilizando algoritmos baseados em duas estruturas de dados, **Árvores Balanceadas AVL** e **Lista Simplesmente Encadeada**, sendo que foram utilizadas **duas estruturas de Árvore,** uma que ordena as vendas por Filial e outra ordena as vendas por AnoMês, cada árvore por sua vez contém uma lista para armazenar as vendas devidamente ordenadas, estas estruturas foram criadas utilizando a linguagem JAVA, por meio de classes que juntamente com outras auxiliares totalizaram 9 classes, conforme discriminado abaixo:

- **Para a ÁRVORE que ordena os as vendas por FILIAL:**
  - Classes: ArvoreUtil_Filial, Arvore_Filial, NoArvoreFilial
- **Para a ÁRVORE que ordena os as vendas por DATA:**
  - Classes: ArvoreUtilAno_Mes, Arvore_AnoMes, NoArvore_AnoMes
- **Para a LISTA que será utilizada por ambas as ÁRVORES:**
  - Classes: ListaVenda, NoLista
- **Para armazenamento e manipulação das VENDAS:**
  - Classe: Venda
- **Para a execução do programa:**
  - Classe: Main_TrabalhoDeImplementacao

O utilização destas classes será detalhada conforme segue.

Respondendo o item *1) total de vendas das filiais com códigos entre 10 e 20*

Como está expressão requisita o total de vendas no intervalo de duas filiais, optou-se por armazenar a venda de cada FILIAL em uma Estrutura de Dados do tipo LISTA SIMPLESMENTE ENCADEADA (ListaVenda) e por sua vez armazenar esta LISTA em uma Estrutura de Dados do tipo ÁRVORE BALANCEADA AVL(ArvoreUtil_Filial) sendo que cada nó da árvore contém então uma chave COD_FILIAL que permite ordenar esta árvore por FILIAIS e assim otimizar a busca no intervalo de filias conforme requerido pelo problema, assim que uma filial contida no intervalo de busca é encontrada o algoritmo busca internamente na LISTA SIMPLESMENTE ENCADEADAde cada nó da LISTA os totais vendidos de cada venda e retorna assim o Total de vendas de todas as filiais no intervalo requisitado.

Para implementação destas estruturas foram utilizadas as seguintes Classes:

- **ArvoreUtil_Filial** - contém as funções e procedimentos responsáveis por realizar a busca ordenada pelas filiais percorrendo somente filiais contidas no intervalo requisitado
- **Arvore_Filial** - Contem todas as funções e procedimentos responsáveis pela insercção na Arvore AVL e todas as funções e procedimentos responsáveis pelas buscas
- **NoArvoreFilial -** Contem todas as funções e procedimentos responśaveis pela inserção da lista no Nó da árvores e as funções e procedimentos auxiliares a inserção e balanceamento da árvore
- **ListaVenda -** Contem todas as funções e procedimentos responśaveis pela inserção e buscas na lista
- **NoLista -** Contem todas as funções e procedimentos responśaveis pela manipulaçao da venda nos nós da Lista
- **Venda -** Contem todos os atributos, funções e procedimentos responśaveis efetivamente pelos dados das vendas obtidas do arquivoVendas.txt

Motivação

A opção pela utilização das estruturas ÁRVORE BALANCEADA AVL e LISTA SIMPLESMENTE ENCADEADA se deu principalmente por que diante os requisitos de busca apresentados a utilização da Árvore AVL permitiu uma busca otimizada com complexidade próxima de O(log n), visto que, a árvore balanceada fornece uma inserção ordenada e bem distribuída, desta forma a busca se torna mais rápida pois a chave de busca é somente o CÓD_FILIAL, o uso da Estrutura de dados LISTA SIMPLESMENTE ENCADEADA por sua vez se deu para armazenamento das vendas sem compromentimento na “performance” do algoritmo pois no caso dos requisitos de busca apresentados é indiferente a estrutura utilizada pois temos de percorrer todos os Objetos Venda de cada filial armazenada na lista para obter o total vendido por filial. Portanto a busca foi otimizada principalmente pela utilização da estrutura de dados ÁRVORE BALANCEADA AVL.

Respondendo o item *2) total de vendas das filiais com códigos entre 10 e 20 nos meses de Jan/17 até Jun/17*

Para responder esta expressão que nos solicita o *total de vendas no intervalo de duas filiais e no intervalo de dois meses distintos*, optou-se por quetão de praticidade utilizar as mesmas estruturas dados já utilizadas no item ***1) total de vendas das filiais com códigos entre 10 e 20*** para armazenar a vendas na estrutura do tipo LISTA (ListaVenda) com vendas de cada FILIAL em um nó da lista e por sua vez armazenar esta LISTA em uma Estrutura de Dados do tipo ÁRVORE BALANCEADA AVL(ArvoreUtil_Filial), sendo que cada nó da árvore contém então uma chave COD_FILIAL que permite ordenar esta árvore por FILIAIS, podemos então percorrer a lista contendo as filiais e buscar nesta, as filiais que contém vendas no intervalo de ANO_MES requisitado pelo usuário e desta forma responder a requisição com o *total de vendas no intervalo de duas filiais e no intervalo de dois meses distintos.*

Motivação

A opção pela utilização destas estruturas entretanto **não levou em consideração o algoritmo mais eficiente** pois diante os requisitos de busca apresentados a utilização da Árvore AVL nos permitiu uma busca otimizada, entretanto o uso da Estrutura de dados LISTA SIMPLESMENTE ENCADEADA por sua vez não é a melhor opção pois temos de percorrer todas as vendas armazenadas na lista sendo que só desejamos obter o total vendido de alguns meses específicos, há portanto um comprometimento na “performance” do algoritmo diante os requisitos de busca apresentados como temos de percorrer todos os Objetos Venda de cada filial armazenada na lista para obter o total vendido por filial a busca **só** foi otimizada pela utilização da estrutura de dados ÁRVORE BALANCEADA AVL, sendo a escolha da lista afeta um o performance do algoritmo.

Como o algoritmo responsável por responder a esta questão utiliza as mesmas classes apresentadas no item ***1) total de vendas das filiais com códigos entre 10 e 20*** optamos por apresentar só a exceção que é o fato de que nas classes **Arvore_Filial** a principal função é a função **buscarTotalVendidoPorData** que retorna o total vendido tendo como parâmetros um intervalo de filias e meses.

Respondendo o item *3) total de vendas de todas as filiais nos meses de Ago/17 até Out/17*

Como nos é solicitado o total de vendas de todas as filias no intervalo de dois meses distintos, assim como foi feito no item ***1) total de vendas das filiais com códigos entre 10 e 20***, optou-se por armazenar a venda de cada filial em uma estrutura de Dados do tipo LISTA (ListaVenda) sendo que cada LISTA SIMPLESMENTE ENCADEADA desta vez armazena somente vendas com DATAS idênticas, desta forma poderíamos utilizar a árvore balanceada AVL (ArvoreUtilAno_Mes) para guardar de forma ordenada e balanceada uma chave contendo uma Data idêntica a todas as datas de uma dada LISTA e assim realizar a busca de forma otimizada permitindo assim obter o **Total *de vendas de todas as filiais* num dado intervalo de meses**.

Para implementação destas estruturas foram utilizadas as seguintes Classe

- **ArvoreUtilAno_Mes** - contém as funções e procedimentos responsáveis por realizar a busca ordenada pelas filiais percorrendo somente filiais contidas no intervalo requisitado
- **Arvore_AnoMes** - Contem todas as funções e procedimentos responsáveis pela insercção na Arvore AVL e todas as funções e procedimentos responsáveis pelas buscas
- **NoArvore_AnoMes -** Contem todas as funções e procedimentos responśaveis pela inserção da lista no Nó da árvores e as funções e procedimentos auxiliares a inserção e balanceamento da árvore
- **ListaVenda -** Contem todas as funções e procedimentos responśaveis pela inserção e buscas na lista

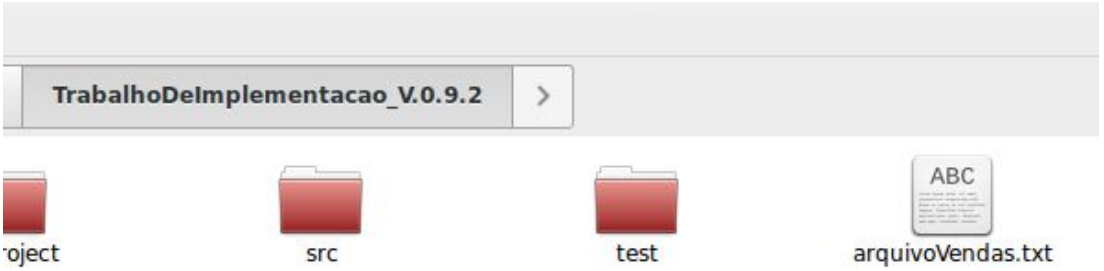
- **NoLista -** Contem todas as funções e procedimentos responsáveis pela manipulaçao da venda nos nós da Lista
- **Venda -** Contem todos os atributos, funções e procedimentos responsáveis efetivamente pelos dados das vendas obtidas do arquivoVendas.txt

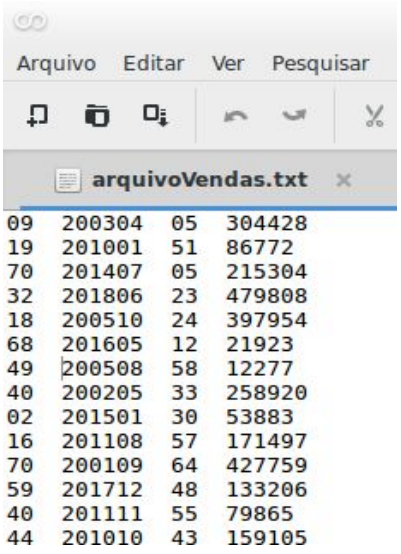
Motivação

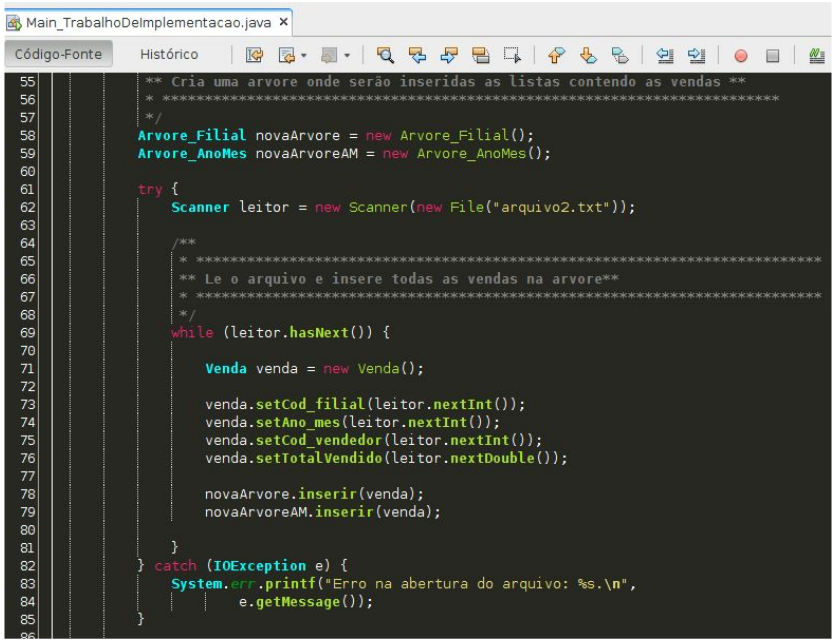
Assim como foi para o item ***1) total de vendas das filiais com códigos entre 10 e 20,*** a opção pela utilização destas estruturas se deu principalmente por que para os requisitos de busca apresentados a utilização da Árvore AVL permitiu também uma busca otimizada com complexidade próxima de O(log n), visto que, a árvore balanceada fornece uma inserção ordenada e bem distribuída, desta forma a busca se torna mais rápida pois a chave de busca é somente o ANO_MES, o uso da Estrutura de dados LISTA SIMPLESMENTE ENCADEADA por sua vez também é escolhido para o armazenamento das vendas sem comprometimento na “performance” do algoritmo pois para o caso dos requisitos de busca apresentados é indiferente a estrutura utilizada pois temos de percorrer todos os Objetos Venda de cada filial armazenada na lista para obter o total vendido por mês. Portanto a busca foi otimizada principalmente pela utilização da estrutura de dados ÁRVORE BALANCEADA AVL.

ALGUMAS ORIENTAÇÕES PARA O FUNCIONAMENTO DO PROGRAMA

O programa carrega as duas Árvores com as vendas contidas no arquivo *arquivoVendas.txt* localizado na raiz da pasta principal *TrabalhoDeImplementacao_V.0.9.3*







Portanto basta substituir na pasta principal o arquivoVenda.txt por outro que tenham o mesmo formato de cod_filial, ano_mes, cod_vendedor, Total_Vendido do arquivo, ou seja filiais com códigos de 0000 a 1000 por exemplo e datas no mesmo formato ano_me, por exemplo 201701 (esta exigência se dá para manter a ordenação por data na arvore de Data) e salvar com o mesmo nome ou alterar pelo nome dado ao arquivo em

```
Scanner leitor = new Scanner(new File("arquivoVendas.txt"));
```

na classe *Main_TrabalhoDeImplementacao.*Como os dados salvos no arquivo original contém filiais gerada aleatoriamente num intervalo de 00 a 100 e aS datas foram geradas num intervalo de 200001 a 201812, fazer buscas nestes intervalos para obter sucesso.

Como acréscimo que ajuda a visualizar o funcionamento do algoritmo foi implementado com ajuda do algoritmo de visualização da árvore disponível *em* https://github.com/lapaesleme/lleme/blob/master/TIC10002/src/main/java/uff/ic/lleme/tic10002/arvoreAVL/ArvoreAVLCorrigida.java
no programa a opção 4 - IMPRIMIR árvores que permite imprimir as árvores BALANCEADAS contendo a chave de ordenação/o fator de balanceamento e a lista de vendas que está armazenada no Nó da árvore conforme exemplo abaixo(O destaque representa a Lista de vendas armazenada no nó da Filial 30).

```
ARVORE ORGANIZADA por FILIAIS

         /----- 30/0 <-[30 201708 50 9000.0] <-[30 201703 50 8000.0]
 /----- 14/0 <-[14 201708 31 10000.0]
 |       \----- 12/0 <-[12 201603 2 3000.0] <-[12 201705 4 2000.0] <-[12 201706 1 1000.0]
7/0 <-[7 201707 3 7000.0]
 \----- 3/1 <-[3 201706 5 6000.0] <-[3 201608 4 5000.0] <-[3 201705 5 4000.0]
         \----- 1/0 <-[1 201709 31 10000.0]
```