

Aluno: Dhruv Babani

Trabalho III - Relatório

4645G-04 - Algoritmos e Estrutura de Dados I Turma 012 - 2022/2 - Prof. Edson Ifarraguirre Moreno

Este relatório consiste em detalhar o problema "A tribo bárbara" e descrever como foi a solução implementada, demonstrando a idealização do algoritmo e da estrutura de dados utilizada no trabalho.

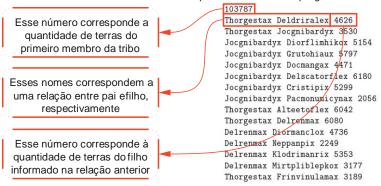
O código foi construído na linguagem Java, respeitando o objetivo apresentado na especificação e seguindo as premissas estruturais e funcionais de forma majoritária.

Inicialmente, durante a etapa de concepção do problema, foi analisado o contexto da enunciação, sendo registrado o seguinte fluxo de raciocínio:

- Ao analisar as primeiras ideias contidas no enunciado do problema, percebi que tratava- se de uma tribo, que por sua vez contém tradições relacionadas à herança de terras entre os membros.
 - a. A primeira tradição diz que as terras de um membro devem ser divididas igualmente entre os filhos;
 - b. A segunda se trata de uma informação adicional da primeira tradição, que além das terras herdadas do pai, o membro também tem a possibilidade de possuir terras de maneira independente;
- Após esse entendimento, verifiquei o objetivo do problema, que consistia em descobrir qual membro teve mais terras e quantas ele possuía, porém deve pertencer a última geração da tribo.
- Com isso em mente, obtive algumas dúvidas imediatas relacionadas ao processo de descobrimento desse membro.
 - a. Deve- se levar em consideração a divisão da herança para os filhos no momento da procura do membro com mais terras?
 - b. A última geração corresponde aos filhos com maior número de gerações precedentes?
- 4. Portanto, na tentativa de compreender o problema com mais detalhes, consegui responder as minhas dúvidas:
 - a. O processo de distribuição das terras aos filhos é uma tradição seguida assim que determinado membro morre, ou seja, o descobrimento daquele

daúltima geração que possui mais terras deve ser realizado prevendo a morte das gerações anteriores e a devida divisão da herança.

Dessa forma, com o raciocínio do problema já estabelecido, dei prosseguimento com a análise dos dados contidos na entrada que será recebida no programa:



Sendo assim, percebi que o padrão contido nos dados de entrada do problema e a imagem do enunciado, representativa da tribo, demonstra ser um indício para implementação e utilização de uma árvore genérica, com cada nó representando um membro da tribo, e um peso, sendo o número de terras.

Além disso, a raiz sendo representaria o primeiro membro da tribo, os galhos corresponderiam as relações entre pai e filho, e por último, as folhas poderiam ser compreendidas como os membros da última geração.

Portanto, iniciei a implementação dessa estrutura de dados, tendo os seguintes atributos e métodos:

Os métodos contidos na classe *TribeTree* são utilizados no decorrer do programa, sendo úteis para as seguintes funcionalidades:

- Descobrir o membro da tribo com mais terras através dos métodos getWarriorWithMoreLands e calculateLandsHelper;
- 2. Inserir um membro através dos métodos insert, findHelper e appendHelper,

Após a implementação dessa classe, iniciei o processo de leitura e recebimento dos dados de entrada do problema, sendo realizado com o seguinte fluxo:

```
public class TribeTree {
   private TreeNode firstWarrior:
   private TreeNode warriorWithMoreLands;
   public TribeTree(String warriorName, double lands) {-
   static class TreeNode {--
   public TreeNode getFirstWarrior() {-
   public String getWarriorWithMoreLands() {-
   private void calculateLandsHelper(TreeNode treeNode, int height) {-
   private Node appendHelper(Node node, TreeNode treeNode) {-
   public TreeNode findHelper(TreeNode treeNode, String warriorName) {-
   public void insert(-
                                                       BufferedReader reader =
                                                          Files.newBufferedReader(path, Charset.defaultCharset())
   Criação do objeto da classe
                                                       String aux ];
                                                       String s = reader.readLine():
        TribeTreeSerializer
implementada exclusivamente
                                                       Integer firstWarriorLands = 0;
     para tratativa dos dados
                                                       if (s = null)
                                                           firstWarriorLands = Integer.parseInt(s);
   recebidos e posteriormente
                                                       String line = null:
 utilizados para constituição da
                                                       TribeTreeSerializer serializer =
                árvore
                                                           new TribeTreeSerializer(firstWarriorLands);
                                                       while ((line = reader.readLine()) # null) {
                                                          aux = line.split(" "):
                                                          String fatherName = aux[0];
                                                          String childName = aux[1];
                                                          Integer childLands = Integer.parseInt(aux[2]);
    Leitura de cada linha dos
                                                           serializer.addLineSerializer(fatherName, childName, childLands);
 arquivos de teste contendo os
         dados de entrada
                                                       return serializer.generateTribeTree();
                                                     catch (IOException e)
                                                       System.err.format("An error ocurred reading the file: ", e);
```

Portanto, após a geração da árvore já contendo os dados da tribo, implementei o fluxo de execução do programa:

```
public static void main(String[] args) {
    String[] testPaths = new String[] {
        "casoMs.txt",
        "casoMs.txt",
```

-Composição de saídas do programa contendo os dados esperados pela enunciação do trabalho

*Os resultados de acordo com os casos de teste apresentados são:

```
Caso de teste:
-> caso04.txt
Guerreiro da última geração da tribo com mais terras (após herança):
-> Kraumifux com 6726.23 terras
Caso de teste:
-> caso05.txt
Guerreiro da última geração da tribo com mais terras (após herança):
-> Gorblepdriverpax com 6880.69 terras
Caso de teste:
-> caso06.txt
Guerreiro da última geração da tribo com mais terras (após herança):
-> Primancribax com 8463.81 terras
Caso de teste:
-> caso07.txt
Guerreiro da última geração da tribo com mais terras (após herança):
-> Nabcraularix com 7673.95 terras
```