CEFET-MG
Centro Federal de Educação Tecnológica de Minas Gerais

Relatório Trabalho Prático 1

Árvore Patricia

Bruna Gomes Camilo

Erick Henrique Dutra de Souza

Lucas Cota Dornelas

Julho de 2021

Laboratório de Algoritmo e Estrutura de Dados II

1. Descrição

O presente trabalho consiste em implementar um sistema onde se busca indexar as palavras de um determinado texto utilizando-se a estrutura de dados conhecida como Árvore Patricia.

A Árvore Patricia é uma representação compacta de uma Trie, que é uma estrutura de dados do tipo Árvore Ordenada, que pode ser usada para indexar algumas chaves que normalmente é formado por uma cadeia de caracteres. Uma Trie usa cada uma das partes de uma chave por vez para determinar uma sub-árvore, enquanto a Árvore Patricia escolhe um elemento da chave, que armazena sua posição, para determinar a sub-árvore. Esse fato, remove a necessidade de nós com apenas um descendente.

Sendo assim, Patricia é um algoritmo para realização de buscas em árvores com as chaves de nós representadas em binário, sem armazenar as chaves nos nós.

2. Implementação

Para implementarmos o algoritmo de inserção é busca em uma Árvore Patricia, utilizamos previamente as classes `ExtraiPalavra` e `ArvorePatricia`, contidas no livro 'Projeto de Algoritmos com implementações em Java e C++', do autor Nivio Ziviani, como base para a implementação do algoritmo. A classe `ExtraiPalavra` é responsável por extrair de um texto as palavras que serão indexadas na Árvore Patricia, enquanto a classe `ArvorePatricia` é responsável por inserir é buscar chaves contidas em sua estrutura.

Além disso, temos a classe `Item` que cria uma nova chave para inserir na Árvore e a classe `Main` que nos permite implementar todo o algoritmo desenvolvido.

Sendo assim, temos:

ExtraiPalavra.java > ...

You, seconds ago | 2 authors (Lucas Cota and others)

```java
import java.util.StringTokenizer;
import java.io.*;
```

You, seconds ago | 2 authors (Lucas Cota and others)

```java
public class ExtraiPalavra {
    private BufferedReader arqDelim, arqTxt;
    private StringTokenizer work;
    private String delimitadores;
    public int linhaMatriz;
    public int colunaMatriz;

    public ExtraiPalavra (String nomeArqDelim, String nomeArqTxt) throws Exception {

        this.arqDelim = new BufferedReader (new FileReader (nomeArqDelim));
        this.arqTxt = new BufferedReader (new FileReader (nomeArqTxt));

        // os delimitadores estão juntos em uma única linha do arquivo delim.txt
        int crlf = 0x0D0A;
        this.delimitadores = arqDelim.readLine() + '\r' + '\n' + (char)crlf;

        this.work = null;
        this.linhaMatriz=0;
        this.colunaMatriz=0;
    }

    public String nextWord() throws Exception{
        if (work == null || !work.hasMoreTokens()) {
            String linha = arqTxt.readLine(); linhaMatriz++; colunaMatriz=0;

            if (linha == null) {
                return null;
            }
            this.work = new StringTokenizer (linha, this.delimitadores);

            if (!work.hasMoreTokens()) {
                return "";
            }
        }
        colunaMatriz++;

        return this.work.nextToken();
    }

    public void fecharArquivos () throws Exception {

        this.arqDelim.close();
        this.arqTxt.close();
    }
}
```

ArvorePatricia.java

```
You, a minute ago | 2 authors (You and others)
1  import java.util.ArrayList;
2
   You, a minute ago | 2 authors (You and others)
3  public class ArvorePatricia {
4
5      // variavel 'global' dos numeros de bits de um char padrao
6      public static final int nBitsChar = 8;
       You, 2 days ago | 1 author (You)
7      private static abstract class PatNo {
8      }
       Lucas Cota, 3 days ago | 1 author (Lucas Cota)
9      private static class PatNoInt extends PatNo {
10         int index; PatNo esq, dir;
11     }
       You, 2 days ago | 2 authors (You and others)
12     private static class PatNoExt extends PatNo {
13         // O tipo da chave depende da aplicacao
14         String chave;
15         ArrayList<int[]> instancias;
16
17         public PatNoExt() {
18             instancias = new ArrayList<int[]>();
19         }
20
21         public void addInstacia(int linha, int coluna)
22         {
23             int[] array = {linha,coluna};
24             boolean b = false;
25             for(int i = 0; i < this.instancias.size(); i++) {
26                 int[] aux = this.instancias.get(i);
27                 if(aux[0] == linha && aux[1] == coluna) {
28                     b = true;
29                     break;
30                 }
31             }
32             if(!b)
33                 this.instancias.add(array);
34         }
35     }
36
37     private PatNo raiz;
38     private int nbitsChave;
39
```

ArvorePatricia.java

```java
    // Retorna o i-esimo bit da chave k a partir da esquerda
    private int bit (int index, String str) {
        if(index != 0) {
            index--;
            int chave = (int)(str.charAt((int)(index / nBitsChar)));
            for (int j = 1; j <= nBitsChar - index % nBitsChar; j++)
                chave = chave/2;
            return chave % 2;
        }

        else
            return 0;
    }

    // Verifica a classe de p
    private boolean isClass(PatNo p) {
        Class classe = p.getClass ();
        return classe.getName().equals(PatNoExt.class.getName());
    }

    // cria um no interno na arvore
    private PatNo criaNoInt(int index, PatNo esq, PatNo dir) {
        PatNoInt p = new PatNoInt ();
        p.index = index;
        p.esq = esq;
        p.dir = dir;
        return p;
    }

    // cria um no externo na arvore
    private PatNo criaNoExt (Item it) {
        PatNoExt p = new PatNoExt ();
        p.chave = it.chave;
        p.addInstacia(it.linha,it.coluna);
        return p;
    }
```

ArvorePatricia.java

```java
    }

    // metodo que realiza a pesquisa da string em um no
    private void pesquisa(String str, PatNo t) {
        if (!this.isClass (t)) {
            PatNoInt aux = (PatNoInt)t;
            if (this.bit (aux.index, str) != 0)
                pesquisa (str, aux.dir);
            else
                pesquisa (str, aux.esq);
        }
        else {
            PatNoExt aux = (PatNoExt)t;
            // se a string estiver na chave
            if (!aux.chave.equals(str))
                System.out.println ("Elemento nao encontrado");
            else {
                System.out.println ("Elemento encontrado");
                // imprimindo as linhas de ocorrencias
                for(int i=0; i<aux.instancias.size(); i++)
                    System.out.println("Linha "+aux.instancias.get(i)[0]+" / Coluna "+aux.instancias.get(i)[1]);
            }
        }
    }
```

ArvorePatricia.java

```java
    // metodo que insere um item
    private PatNo insereEntre(Item it, PatNo t, int index) {
        PatNoInt aux = null;
        if (!this.isClass (t))
            aux = (PatNoInt)t;

        // Cria um novo no externo
        if (this.isClass (t) || (index < aux.index)) {
            PatNo p = this.criaNoExt (it);
            if (this.bit (index, it.chave) != 1)
                return this.criaNoInt (index, p, t);
            else
                return this.criaNoInt (index, t, p);
        }
        else {
            if (this.bit (aux.index, it.chave) != 1)
                aux.esq = this.insereEntre (it, aux.esq, index);
            else
                aux.dir = this.insereEntre (it, aux.dir, index);
            return aux;
        }
    }
```

ArvorePatricia.java

```
122
123     private PatNo insere(Item it, PatNo t) {
124         if (t != null)  {
125             PatNo p = t;
126             while (!this.isClass (p)) {
127                 PatNoInt aux = (PatNoInt)p;
128                 if (this.bit (aux.index, it.chave) == 1)
129                     p = aux.dir;
130                 else
131                     p = aux.esq;
132             }
133             PatNoExt aux = (PatNoExt)p;
134             // acha o primeiro bit diferente
135             int i = 1;
136             while ((i <= this.nbitsChave) && (this.bit (i, it.chave) == this.bit (i, aux.chave)))
137                 i++;
138             if (i <= this.nbitsChave)
139                 return this.insereEntre (it, t, i);
140             else {
141                 System.out.println ("Erro: chave ja esta na arvore");
142                 if(this.isClass(aux))
143                     aux.addInstacia(it.linha,it.coluna);
144                 return t;
145             }
146         }
147         else
148             return this.criaNoExt (it);
149     }
150
```

ArvorePatricia.java

```
149     }
150
151     private void desmembrar(PatNo pai, PatNo filho, String str) {
152         if (filho != null) {
153             if (this.isClass(filho)) {
154                 PatNoExt aux = (PatNoExt)filho;
155                 if (pai == null)
156                     System.out.println ("Pai: "+ pai + " " + str+ " Ext: " + aux.chave);
157                 else
158                     System.out.println ("Pai: "+ ((PatNoInt)pai).index + " " + str+ " Ext: " + aux.chave);
159             } else {
160                 PatNoInt aux = (PatNoInt)filho;
161                 desmembrar(filho, aux.esq, "Esq");
162                 if (pai == null)
163                     System.out.println ("Pai: "+ pai + " " + str+ " Int: " + aux.index);
164                 else
165                     System.out.println ("Pai: "+ ((PatNoInt)pai).index + " " + str+ " Int: " + aux.index);
166
167                     desmembrar(filho, aux.dir, "Dir");
168             }
169         }
170     }
171
172     public void imprime () {
173         this.desmembrar (null, this.raiz, "Raiz");
174     }
175
176     public ArvorePatricia(int nbitsChave) {
177         this.raiz = null; this.nbitsChave = nbitsChave;
178     }
179
180     public void pesquisa(String k) {
181         this.pesquisa (k, this.raiz);
182     }
183
184     public void insere(Item k) {
185         this.raiz = this.insere (k, this.raiz);
186     }
187 }
188
```

Item.java > ...

You, 3 minutes ago | 2 authors (Lucas Cota and others)

```java
public class Item {
    public String chave;
    public static int size;
    public int linha;
    public int coluna;

    public Item (String chave,int i,int j) {
        this.chave = chave;
        this.linha = i;
        this.coluna = j;
        Item.size = this.chave.length();
    }
}
```

Main.java > Main > insereArvore(ArvorePatricia, ExtraiPalavra)

```java
import java.io.*;
import java.util.Scanner;

public class Main {
    public static final int nCharPalavra = 16;
    public static final int nBitsChar = 8;
    ...
    private static class Str2bin{
        public static String str16(String str){

            // Garantir 16 bits
            if(str.length() > nCharPalavra)
                str = str.substring(0,nCharPalavra-1);
            else
                while (str.length() < nCharPalavra)
                    str = str.concat(" ");
            return str;
        }
    }

    public static ArvorePatricia insereArvore(ArvorePatricia dicionario, ExtraiPalavra input) {
        // Insere cada chave na arvore
        for (int i = 0;; i++) {
            String str;
            try{
                str = input.nextWord().toLowerCase();

                //verificando se a string e vazia
                if(str.compareTo(" ") == (-1))
                    str = input.nextWord().toLowerCase();

                //garantir 16 bits na palavra
                str = Str2bin.str16(str);

            }catch (Exception e){
                break;
            }

            // inserindo o item na arvore
            dicionario.insere(new Item(str,input.linhaMatriz,input.colunaMatriz));

            System.out.println ("Inseriu chave "+ i + ": " + str + str.charAt(0) + " - Linha "
                + input.linhaMatriz + " / Coluna " + input.colunaMatriz);
        }
        return dicionario;
    }
```

Main.java > Main > insereArvore(ArvorePatricia, ExtraiPalavra)

```
45      }
46
47      public static Scanner pesquisaArvore(ArvorePatricia dicionario, Scanner search_input) {
48          // Pesquisa cada chave(contida no arquivo pesquisa.txt) na arvore
49          File file = new File("pesquisa.txt");
50          try{
51              search_input = new Scanner(file);
52          } catch (Exception e){
53              System.out.println("Erro: arquivo de pesquisa não aberto");
54              return null;
55          }
56
57          int index = 0;
58          while(search_input.hasNext()) {
59              String c = search_input.nextLine();
60              System.out.println ("Pesquisando chave" + index + ": " + c);
61              c = Str2bin.str16(c);
62              dicionario.pesquisa(c);
63              index++;
64          }
65          return search_input;
66      }
67
68
        Run | Debug
69      public static void main (String[] args) {
70          ArvorePatricia dicionario = new ArvorePatricia(nCharPalavra*nBitsChar);
71
72          if(args[0].length() == 0) {
73              System.out.println("Error: nenhum arquivo passado por parametro.");
74          }
75
76          // Abre o arquivo
77          ExtraiPalavra input;
78          try{
79              input = new ExtraiPalavra("delim.txt", args[0]);
80          }catch (Exception e){
81              System.out.println("Erro: arquivo não aberto");
82              return;
83          }
84
85          // inserindo elementos na arvore
86          dicionario = insereArvore(dicionario, input);
87
88          System.out.println();
89
90          // Imprime arvore
91          dicionario.imprime();
92
```

Main.java > Main > insereArvore(ArvorePatricia, ExtraiPalavra)

```
89
90          // Imprime arvore
91          dicionario.imprime();
92
93          System.out.println();
94
95          Scanner search_input = null;
96
97          // pesquisando elementos na arvore
98          search_input = pesquisaArvore(dicionario, search_input);
99
100         try{
101             input.fecharArquivos();
102         } catch (Exception e){
103             System.out.println("Erro: não foi possível fechar arquivo");
104         }
105
106         search_input.close();
107     }
108 }
```

3. Resultados

Para testar o nosso algoritmo, testamos a busca de algumas palavras em dois arquivos listados a seguir:

Arquivo			
Exemplo1.txt		**Exemplo2.txt**	
Palavra Buscada	**Resultado**	**Palavra Buscada**	**Resultado**
trabalho	Elemento encontrado Linha 22 / Coluna 2 Linha 24 / Coluna 2	sociedade	Elemento encontrado Linha 6 / Coluna 14 Linha 7 / Coluna 29 Linha 16 / Coluna 6 Linha 33 / Coluna 4 Linha 33 / Coluna 22 Linha 36 / Coluna 4 Linha 38 / Coluna 29

			Linha 39 / Coluna 44 Linha 62 / Coluna 2 Linha 74 / Coluna 6
computacao	Elemento encontrado Linha 1 / Coluna 3 Linha 5 / Coluna 8 Linha 8 / Coluna 2 Linha 9 / Coluna 7 Linha 14 / Coluna 6 Linha 21 / Coluna 6	software	Elemento encontrado Linha 8 / Coluna 26 Linha 8 / Coluna 54 Linha 22 / Coluna 6
governo	Elemento encontrado Linha 6 / Coluna 9	ideia	Elemento encontrado Linha 8 / Coluna 5 Linha 8 / Coluna 57
educacao	Elemento encontrado Linha 4 / Coluna 10 Linha 27 / Coluna 2 Linha 28 / Coluna 9	pessoa	Elemento encontrado Linha 6 / Coluna 54 Linha 7 / Coluna 23 Linha 7 / Coluna 49 Linha 14 / Coluna 15 Linha 17 / Coluna 13 Linha 18 / Coluna 12 Linha 19 / Coluna 10 Linha 24 / Coluna 12 Linha 28 / Coluna 70
tecnologia	Elemento encontrado Linha 20 / Coluna 4 Linha 30 / Coluna 10 Linha 33 / Coluna 8	informatica	Elemento encontrado Linha 6 / Coluna 12 Linha 14 / Coluna 23 Linha 16 / Coluna 19 Linha 27 / Coluna 56 Linha 28 / Coluna 8 Linha 28 / Coluna 66 Linha 47 / Coluna 63 Linha 60 / Coluna 15 Linha 62 / Coluna 13

			Linha 74 / Coluna 21 Linha 74 / Coluna 66
formacao	Elemento encontrado Linha 1 / Coluna 6 Linha 22 / Coluna 10	etica	Elemento encontrado Linha 1 / Coluna 2 Linha 4 / Coluna 5 Linha 6 / Coluna 22 Linha 6 / Coluna 30 Linha 6 / Coluna 59 Linha 7 / Coluna 2 Linha 7 / Coluna 77 Linha 9 / Coluna 2 Linha 12 / Coluna 3 Linha 14 / Coluna 8 Linha 14 / Coluna 70 Linha 15 / Coluna 43 Linha 16 / Coluna 46 Linha 16 / Coluna 91 Linha 16 / Coluna 109 Linha 27 / Coluna 37 Linha 28 / Coluna 46 Linha 30 / Coluna 6 Linha 32 / Coluna 4 Linha 40 / Coluna 4 Linha 41 / Coluna 7 Linha 42 / Coluna 9 Linha 42 / Coluna 36 Linha 46 / Coluna 5 Linha 48 / Coluna 29 Linha 51 / Coluna 2 Linha 63 / Coluna 7 Linha 74 / Coluna 64
desenvolvimento	Elemento encontrado Linha 26 / Coluna 4 Linha 27 / Coluna 6	muito	Elemento encontrado Linha 7 / Coluna 5 Linha 48 / Coluna 7
que	Elemento encontrado	ciencia	Elemento encontrado

	Linha 4 / Coluna 4 Linha 5 / Coluna 4 Linha 8 / Coluna 8 Linha 14 / Coluna 10 Linha 16 / Coluna 5 Linha 24 / Coluna 6 Linha 26 / Coluna 5 Linha 30 / Coluna 6 Linha 30 / Coluna 14 Linha 34 / Coluna 5 Linha 34 / Coluna 9		Linha 6 / Coluna 62 Linha 7 / Coluna 40 Linha 7 / Coluna 94
informatica	Elemento encontrado Linha 16 / Coluna 2 Linha 27 / Coluna 10	computacao	Elemento encontrado Linha 1 / Coluna 4 Linha 6 / Coluna 24 Linha 14 / Coluna 10 Linha 16 / Coluna 9 Linha 16 / Coluna 93 Linha 18 / Coluna 9 Linha 27 / Coluna 10 Linha 27 / Coluna 94 Linha 46 / Coluna 25 Linha 55 / Coluna 8 Linha 74 / Coluna 34
em	Elemento encontrado Linha 10 / Coluna 8 Linha 14 / Coluna 1 Linha 14 / Coluna 3 Linha 15 / Coluna 1 Linha 19 / Coluna 8 Linha 28 / Coluna 8	que	Elemento encontrado Linha 4 / Coluna 3 Linha 6 / Coluna 63 Linha 7 / Coluna 56 Linha 7 / Coluna 72 Linha 8 / Coluna 29 Linha 8 / Coluna 37 Linha 8 / Coluna 51 Linha 8 / Coluna 69 Linha 8 / Coluna 79 Linha 8 / Coluna 89 Linha 8 / Coluna 111 Linha 9 / Coluna 16

			Linha 14 / Coluna 12 Linha 14 / Coluna 26 Linha 14 / Coluna 39 Linha 14 / Coluna 52 Linha 14 / Coluna 57 Linha 14 / Coluna 62 Linha 14 / Coluna 93 Linha 15 / Coluna 13 Linha 15 / Coluna 26 Linha 15 / Coluna 38 Linha 16 / Coluna 10 Linha 16 / Coluna 22 Linha 16 / Coluna 73 Linha 16 / Coluna 100 Linha 25 / Coluna 11 Linha 26 / Coluna 9 Linha 27 / Coluna 98 Linha 28 / Coluna 2 Linha 28 / Coluna 71 Linha 30 / Coluna 2 Linha 37 / Coluna 7 Linha 38 / Coluna 20 Linha 39 / Coluna 16 Linha 40 / Coluna 12 Linha 40 / Coluna 18 Linha 40 / Coluna 59 Linha 42 / Coluna 50 Linha 42 / Coluna 70 Linha 42 / Coluna 86 Linha 46 / Coluna 36 Linha 47 / Coluna 24 Linha 59 / Coluna 18 Linha 60 / Coluna 7 Linha 69 / Coluna 19 Linha 69 / Coluna 28 Linha 73 / Coluna 8 Linha 73 / Coluna 32 Linha 74 / Coluna 26 Linha 74 / Coluna 59

crise	Elemento encontrado Linha 31 / Coluna 6	area	Linha 14 / Coluna 18 Linha 14 / Coluna 89 Linha 16 / Coluna 17 Linha 16 / Coluna 29 Linha 28 / Coluna 6 Linha 28 / Coluna 79 Linha 34 / Coluna 12 Linha 38 / Coluna 17 Linha 46 / Coluna 23 Linha 54 / Coluna 6 Linha 56 / Coluna 13
		moral	Linha 6 / Coluna 41 Linha 6 / Coluna 42 Linha 7 / Coluna 14 Linha 8 / Coluna 28 Linha 8 / Coluna 68 Linha 42 / Coluna 44

4. Desafios encontrados

O algoritmo de inserção e busca em uma Árvore Patricia exigiu o conhecimento teórico de construção da Árvore Patricia, para a implementação do algoritmo. A classe `ExtraiPalavra` identificou todos caracteres contidos no arquivo, e como desejávamos que apenas as palavras fossem inseridas na Árvore, tivemos que implementar alguns delimitadores de leitura, como por exemplo '*', '/' '(' ')' ',', dentre outros, para garantir que apenas as palavras fossem inseridas na Árvore. Logo, todos os caracteres que não constituem como sendo letras e algarismos, não são inseridos na Árvore.

Sendo assim, tivemos que nos atentar em relação ao filtro de que a sequência de caracteres não deve conter símbolos, como por exemplo a String " `verdade`," ao ser inserida na Árvore Patricia deverá ser inserida como "`verdade`". Essa implementação foi necessária a fim de que o próximo passo de execução, a busca, seja realizada corretamente.