

Exercício-Programa 3

Árvores Binárias de Busca

1º Semestre 2003

1 Introdução

Árvores binárias de busca fornecem uma maneira eficiente de se localizar dados.

Cada um destes é associado a uma chave (uma informação que permita realizar uma ordenação), e então é inserido na árvore, de modo que para cada nó na árvore, à sua esquerda sempre está um elemento de chave inferior, e à sua direita está um elemento de chave superior.

As operações de inserção e remoção na árvore binária de busca são ambas $O(n)$ no pior caso e $O(\log n)$ no caso médio.

2 Classes e descrições

Arquivos componentes:

- ArvoreBinaria.java
- NoBinario.java
- ResultadoBusca.java
- TestaAlgoritmos.java

2.1 Classe ArvoreBinaria

Classe gerenciadora da árvore binária, composta de elementos do tipo `NoBinario`.

A árvore implementa as operações de busca de nós, percurso simétrico/*in-ordem* e determinação de altura.

2.2 Classe NoBinario

O nó binário é o componente da árvore binária.

É uma estrutura que possui um campo chave, um campo para informações (implementado como `String`) e duas referências para objetos de mesmo tipo, sendo estes os nós à esquerda e à direita do mesmo.

2.3 Classe ResultadoBusca

Encapsula o resultado de uma busca pela árvore.

Possui um indicador do resultado da busca, que podem ser:

- 0: árvore binária de busca é vazia.
- 1: chave existe e o resultado está contido na instância.
- 2: chave não está na árvore e deve ser inserida à esquerda do último nó pesquisado, contido na instância.
- 3: chave não está na árvore e deve ser inserida à direita do último nó pesquisado, contido na instância.

2.4 Classe TestaAlgoritmos

Fornece métodos para teste das estruturas de dados e algoritmos implementados.

Carrega dados da entrada-padrão, qualquer tipo de informação pode ser inserido na árvore binária. Uma informação qualquer terá associada a ela uma chave calculada aleatoriamente (no caso, um código de *hash*) para ser inserida na árvore. As inserções são mostradas passo-a-passo, em seguida é mostrado o número de níveis da árvore e o resultado do percurso simétrico.

Para utilizar, basta:

```
java TestaAlgoritmos < arquivo.txt
```

ou:

```
programa | java TestaAlgoritmos (pode-se usar a saída do ls, rwho, etc. por exemplo).
```

3 Códigos-fonte

Estão apresentados a seguir os códigos-fonte das classes relacionadas.

3.1 ArvoreBinariaBusca.java

```

1  /* ArvoreBinariaBusca.java
2     Copyright (C) 2004 Rodolpho Iemini Atoji
3
4     This program is free software; you can redistribute it and/or modify
5     it under the terms of the GNU General Public License as published by
6     the Free Software Foundation; either version 2 of the License, or
7     (at your option) any later version.
8
9     This program is distributed in the hope that it will be useful,
10    but WITHOUT ANY WARRANTY; without even the implied warranty of
11    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
12    GNU General Public License for more details.
13
14    You should have received a copy of the GNU General Public License
15    along with this program; if not, write to the Free Software
16    Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
17  */
```

```

18
19 // *****
20 // **   MAC323 - Estruturas de Dados           **
21 // **   IME-USP - Primeiro Semestre de 2004   **
22 // **   Turma 45 - Professor Siang Wun Song    **
23 // **                                           **
24 // **   Exercício-Programa 3 -- Árvores Binárias de Busca **
25 // **                                           **
26 // **   Arquivo:ArvoreBinariaBusca.java       **
27 // **                                           **
28 // **   Rodolpho Iemini Atoji           4894631 **
29 // **                                           **
30 // **   Data de entrega: 30/06/2004         **
31 // *****
32
33 /**
34  * Representante de uma árvore binária de busca, utilizando elementos
35  * 'NoBinario'.
36  *
37  * @author Rodolpho Iemini Atoji
38  * @see NoBinario
39  * @version 0.1
40  */
41 public class ArvoreBinariaBusca {
42
43     private NoBinario raiz;
44
45     /**
46      * Construtor-padrão.
47      * Instancia uma árvore binária com raiz nula.
48      */
49     public ArvoreBinariaBusca() {
50         this(null);
51     }
52
53     /**
54      * Construtor inicializador.
55      *
56      * @param r Nó para inicializar a árvore com alguma raiz.
57      */
58     public ArvoreBinariaBusca(NoBinario r) {
59         this.raiz = r;
60     }
61
62     /**
63      * Insere um nó com uma dada chave na árvore binária.
64      *

```

```
65     * @param chave Chave do nó a ser criado e inserido na árvore.
66     * @param info Informação do nó a ser criado e inserido na árvore.
67     */
68     public void insereNo(int chave, String info) {
69         ResultadoBusca res = buscaChave(chave, raiz);
70
71         /* Verifica se a chave já existe na árvore. */
72         if (res.status() == 1) {
73             System.out.println("Tentativa de reinserção da chave " +chave +
74                 " bloqueada.");
75             return;
76         }
77
78         /* Caso contrário, criar um novo nó. */
79         else {
80             NoBinario novo = new NoBinario(chave, info, null, null);
81
82             /* Insere na raiz se a árvore for vazia. */
83             if (res.status() == 0)
84                 raiz = novo;
85
86             /*
87              * Sabendo que o nó não existe na árvore, já é determinado
88              * previamente pelo 'buscaArvore' que o mesmo deve ser inserido
89              * do lado esquerdo (status) do último nó pesquisado.
90              */
91             else if (res.status() == 2)
92                 res.devolveResultado().mudaNoEsquerdo(novo);
93
94             /*
95              * Sabendo que o nó não existe na árvore, já é determinado
96              * previamente pelo 'buscaArvore' que o mesmo deve ser inserido
97              * do lado direito (status) do último nó pesquisado.
98              */
99             else
100                 res.devolveResultado().mudaNoDireito(novo);
101         }
102     } // insereNo
103
104     /**
105     * Busca na árvore binária por um nó de chave ch.
106     * Assume-se que as informações dos nós sejam identificadas unicamente
107     * por uma chave.
108     *
109     * @param ch Chave do nó a ser pesquisado.
110     * @param no Referência de início de busca.
111     * @return
```

```
112     */
113     public ResultadoBusca buscaChave(int ch, NoBinario no) {
114         NoBinario temp = no;
115         ResultadoBusca resultado = new ResultadoBusca();
116
117         /* Verifica se a árvore está vazia. */
118         if (temp == null) {
119             resultado.mudaStatus((byte)0);
120
121             return resultado;
122         }
123
124         /* Se não estiver, buscar recursivamente o nó desejado. */
125         else {
126
127             /* Encontra de primeira a chave. */
128             if (ch == temp.chave()) {
129                 resultado.mudaStatus((byte)1);
130                 resultado.mudaResultado(temp);
131
132                 return resultado;
133             }
134
135             /*
136              * Verifica se a chave procurada pode estar nas subárvores à
137              * esquerda.
138              */
139             else if (ch < temp.chave()) {
140                 if (temp.noEsquerdo() == null) {
141                     resultado.mudaStatus((byte)2);
142                     resultado.mudaResultado(temp);
143
144                     return resultado;
145                 } else
146                     return buscaChave(ch, temp.noEsquerdo());
147             }
148
149             /*
150              * Na busca do nó pelas subárvores à direita, verifica-se
151              * se foi atingido o limite da árvore, significando que o nó não
152              * está na árvore.
153              */
154             else if (temp.noDireito() == null) {
155                 resultado.mudaStatus((byte)3);
156                 resultado.mudaResultado(temp);
157
158                 return resultado;
```

```
159         }
160
161         /*
162         * Verifica se a chave procurada pode estar nas subárvores à
163         * direita.
164         */
165         else
166             return buscaChave(ch, temp.noDireito());
167     }
168
169 } // buscaChave
170
171 /**
172  * Percorre em ordem simétrica (In-Ordem) uma árvore/subárvore binária,
173  * imprimindo na tela as chaves visitadas.
174  *
175  * @param no Nó de partida para percorrer.
176  */
177 public void percorreInOrdem(NoBinario no) {
178     if (no == null)
179         return;
180
181     percorreInOrdem(no.noEsquerdo());
182     visitaNo(no);
183     percorreInOrdem(no.noDireito());
184 }
185
186 /**
187  * Visita o nó, imprimindo sua chave na tela.
188  *
189  * @param no Nó a ser visitado.
190  */
191 public void visitaNo(NoBinario no) {
192     System.out.print(no.chave() + " ");
193 }
194
195 /**
196  * Determina a altura de uma árvore binária.
197  *
198  * @param no Nó da árvore ou subárvore do qual se parte a busca.
199  * @return Altura da árvore binária.
200  */
201 public int alturaArvore(NoBinario no) {
202     if (no == null)
203         return 0;
204     else {
205         int alturaEsquerda = alturaArvore(no.noEsquerdo());
```

```

206         int alturaDireita = alturaArvore(no.noDireito());
207         if (alturaEsquerda > alturaDireita)
208             return (alturaEsquerda + 1);
209         else
210             return (alturaDireita + 1);
211     }
212 }
213
214 /**
215  * Devolve referência para a raiz da árvore binária.
216  *
217  * @return Referência para a raiz da árvore binária.
218  */
219 public NoBinario devolveRaiz() {
220     return raiz;
221 }
222
223 } // class ArvoreBinariaBusca

```

3.2 NoBinario.java

```

1  /* NoBinario.java
2     Copyright (C) 2004 Rodolpho Iemini Atoji
3
4     This program is free software; you can redistribute it and/or modify
5     it under the terms of the GNU General Public License as published by
6     the Free Software Foundation; either version 2 of the License, or
7     (at your option) any later version.
8
9     This program is distributed in the hope that it will be useful,
10    but WITHOUT ANY WARRANTY; without even the implied warranty of
11    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
12    GNU General Public License for more details.
13
14    You should have received a copy of the GNU General Public License
15    along with this program; if not, write to the Free Software
16    Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
17 */
18
19 // *****
20 // **  MAC323 - Estruturas de Dados                **
21 // **  IME-USP - Primeiro Semestre de 2004        **
22 // **  Turma 45 - Professor Siang Wun Song        **
23 // **                                             **
24 // **  Exercício-Programa 3 -- Árvores Binárias de Busca **
25 // **                                             **
26 // **  Arquivo:NoBinario.java                    **
27 // **                                             **

```

```

28 // ** Rodolpho Iemini Atoji          4894631          **
29 // **                                **
30 // ** Data de entrega: 30/06/2004      **
31 // *****
32
33 /**
34  * Representante de um nó de uma árvore binária.
35  * Cada nó armazena uma informação-objeto do tipo String, que pode ser
36  * convertida para determinados tipos numéricos, se desejado.
37  *
38  * @author Rodolpho Iemini Atoji
39  * @see ArvoreBinariaBusca
40  * @version 0.1
41  */
42 public class NoBinario {
43
44     private int chave;
45     private String info;
46     private NoBinario esq, dir;
47
48     /**
49     * Construtor-padrão.
50     */
51     public NoBinario() {
52         this(0, "Nó não inicializado!", null, null);
53     }
54
55     /**
56     * Construtor inicializador.
57     *
58     * @param c Chave identificadora do nó.
59     * @param i Informação contida no nó.
60     * @param e Referência para nó à esquerda.
61     * @param d Referência para nó à direita.
62     */
63     public NoBinario(int c, String i, NoBinario e, NoBinario d) {
64         this.chave = c;
65         this.info = i;
66         this.esq = e;
67         this.dir = d;
68     }
69
70     /**
71     * Altera a referência para o nó esquerdo.
72     *
73     * @param no Nó a ser apontado.
74     */

```

```
75     public void mudaNoEsquerdo(NoBinario no) {
76         this.esq = no;
77     }
78
79     /**
80      * Altera a referência para o nó direito.
81      *
82      * @param no Nó a ser apontado.
83      */
84     public void mudaNoDireito(NoBinario no) {
85         this.dir = no;
86     }
87
88     /**
89      * Altera informação do nó.
90      *
91      * @param inf Informação a substituir a informação atual do nó.
92      */
93     public void mudaInfo(String inf) {
94         this.info = inf;
95     }
96
97     /**
98      * Altera valor da chave do nó.
99      * Não é recomendável utilizar este método se o nó já estiver inserido
100     * em uma árvore binária de busca.
101     *
102     * @param n Novo valor da chave do nó.
103     */
104     public void mudaValorNo(int n) {
105         this.chave = n;
106     }
107
108     /**
109     * Devolve referência para o nó à esquerda.
110     *
111     * @return Referência para o nó à esquerda.
112     */
113     public NoBinario noEsquerdo() {
114         return esq;
115     }
116
117     /**
118     * Devolve referência para o nó à direita.
119     *
120     * @return Referência para o nó à esquerda.
121     */
```

```

122     public NoBinario noDireito() {
123         return dir;
124     }
125
126     /**
127      * Devolve valor da chave do nó.
128      *
129      * @return Valor da chave do nó.
130      */
131     public int chave() {
132         return chave;
133     }
134
135     /**
136      * Devolve informação contida no nó.
137      *
138      * @return Informação contida no nó.
139      */
140     public String devolveInfo() {
141         return info;
142     }
143
144 } // class NoBinario

```

3.3 ResultadoBusca.java

```

1  /* ResultadoBusca.java
2     Copyright (C) 2004 Rodolpho Iemini Atoji
3
4     This program is free software; you can redistribute it and/or modify
5     it under the terms of the GNU General Public License as published by
6     the Free Software Foundation; either version 2 of the License, or
7     (at your option) any later version.
8
9     This program is distributed in the hope that it will be useful,
10    but WITHOUT ANY WARRANTY; without even the implied warranty of
11    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
12    GNU General Public License for more details.
13
14    You should have received a copy of the GNU General Public License
15    along with this program; if not, write to the Free Software
16    Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
17 */
18
19 // *****
20 // **  MAC323 - Estruturas de Dados                **
21 // **  IME-USP - Primeiro Semestre de 2004        **
22 // **  Turma 45 - Professor Siang Wun Song        **

```

```
23 // ** **
24 // ** Exercício-Programa 3 -- Árvores Binárias de Busca **
25 // ** **
26 // ** Arquivo:ResultadoBusca.java **
27 // ** **
28 // ** Rodolpho Iemini Atoji 4894631 **
29 // ** **
30 // ** Data de entrega: 30/06/2004 **
31 // *****
32
33 /**
34  * Encapsula o resultado de uma busca pela árvore binária.
35  *
36  * @author Rodolpho Iemini Atoji
37  * @see ArvoreBinaria
38  * @version 0.1
39  */
40 public class ResultadoBusca {
41
42     private byte status;
43     private NoBinario encontrado;
44
45     /**
46     * Construtor padrão.
47     * Inicializa o resultado com o status em 9, pois não é utilizado
48     * nas verificações da árvore binária.
49     */
50     public ResultadoBusca() {
51         this((byte)9, null);
52     }
53
54     /**
55     * Construtor inicializador.
56     *
57     * @param st Status do resultado:
58     *     0: árvore binária de busca é vazia.
59     *     1: chave existe e o resultado está contido na instância.
60     *     2: chave não está na árvore e deve ser inserida à esquerda
61     *     do último nó pesquisado, contido na instância.
62     *     3: chave não está na árvore e deve ser inserida à direita
63     *     do último nó pesquisado, contido na instância.
64     * @param ref Referência para resultado encontrado, se houver.
65     */
66     public ResultadoBusca(byte st, NoBinario ref) {
67         this.status = st;
68         this.encontrado = ref;
69     }
}
```

```
70
71     /**
72      * Indica o status do resultado.
73      *
74      * @return Status da busca.
75      */
76     public byte status() {
77         return status;
78     }
79
80     /**
81      * Muda o status do resultado.
82      *
83      * @param st Novo status do resultado.
84      */
85     public void mudaStatus(byte st) {
86         status = st;
87     }
88
89     /**
90      * Atribui uma referência ao resultado encontrado.
91      *
92      * @param ref Referência para o resultado encontrado.
93      */
94     public void mudaResultado(NoBinario ref) {
95         encontrado = ref;
96     }
97
98     /**
99      * Devolve a referência para o resultado encontrado.
100     *
101     * @return Referência para o resultado encontrado.
102     */
103     public NoBinario devolveResultado() {
104         return encontrado;
105     }
106
107 } // class ResultadoBusca
```

3.4 TestaAlgoritmos.java

```
1  /* TestaAlgoritmos.java
2     Copyright (C) 2004 Rodolpho Iemini Atoji
3
4     This program is free software; you can redistribute it and/or modify
5     it under the terms of the GNU General Public License as published by
6     the Free Software Foundation; either version 2 of the License, or
7     (at your option) any later version.
```

```

8
9     This program is distributed in the hope that it will be useful,
10    but WITHOUT ANY WARRANTY; without even the implied warranty of
11    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
12    GNU General Public License for more details.
13
14    You should have received a copy of the GNU General Public License
15    along with this program; if not, write to the Free Software
16    Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
17 */
18
19 // *****
20 // **  MAC323 - Estruturas de Dados                **
21 // **  IME-USP - Primeiro Semestre de 2004        **
22 // **  Turma 45 - Professor Siang Wun Song        **
23 // **                                              **
24 // **  Exercício-Programa 3 -- Árvores Binárias de Busca **
25 // **                                              **
26 // **  Arquivo:TestaAlgoritmos.java              **
27 // **                                              **
28 // **  Rodolpho Iemini Atoji          4894631     **
29 // **                                              **
30 // **  Data de entrega: 30/06/2004                **
31 // *****
32
33 import java.io.*;
34
35 /**
36  * Classe de teste para a árvore binária e seus algoritmos:
37  *
38  * - Inserção;
39  * - Busca de chaves;
40  * - Determinação de altura;
41  * - Percurso em ordem simétrica/in-ordem.
42  *
43  * @author Rodolpho Iemini Atoji
44  * @version 0.1
45  */
46 public class TestaAlgoritmos {
47
48     private ArvoreBinariaBusca arvore;
49
50     /**
51      * Conduz os testes listados acima.
52      */
53     public static void main(String[] args) {
54         TestaAlgoritmos t = new TestaAlgoritmos();

```

```
55     t.carregaDados();
56     t.determinaAltura();
57     t.percorreArvore();
58
59     System.exit(0);
60 }
61
62 /**
63  * Construtor: prepara uma árvore binária de busca para os testes.
64  */
65 public TestaAlgoritmos() {
66     arvore = new ArvoreBinariaBusca();
67 }
68
69 /**
70  * Carrega os dados da entrada-padrão para iniciar os testes.
71  * As chaves das informações são determinadas por um "hash code"
72  * (código de espalhamento), apenas pelo caráter aleatório, de modo
73  * que se possa testar o bom funcionamento da inserção de chaves na
74  * árvore.
75  */
76 private void carregaDados() {
77     BufferedReader leitor;
78
79     try {
80         leitor = new BufferedReader(new InputStreamReader(System.in));
81         String info;
82
83         System.out.println("\nInserindo informações...");
84         while ((info = leitor.readLine()) != null) {
85             arvore.insereNo(info.hashCode(), info);
86             System.out.println("Chave: " +info.hashCode()
87                 +" Informação: " +info);
88         }
89     } catch (IOException io) {
90         System.err.println("Problemas ao ler da entrada-padrão.");
91         System.exit(1);
92     }
93 }
94
95 /**
96  * Imprime a altura da árvore criada.
97  */
98 private void determinaAltura() {
99     System.out.println("Altura da árvore: "
100         +arvore.alturaArvore(arvore.devolveRaiz()));
101 }
```

```
102
103     /**
104     * Imprime um percurso em ordem simétrica pela árvore binária.
105     */
106     private void percorreArvore() {
107         System.out.println("Percorrendo a árvore em ordem simétrica:");
108         arvore.percorreInOrdem(arvore.devolveRaiz());
109         System.out.println("\n");
110     }
111
112 } // class TestaAlgoritmos
```

4 Testes

A validade dos algoritmos implementados pode ser verificada através do percurso em ordem simétrica, que deve apresentar as chaves ordenadas.

4.1 Teste 1

Teste realizado com o seguinte arquivo de entrada:

```
1
10
9
7
4
3
5
6
8
10
18
20
```

Saída obtida:

```
Inserindo informações...
Chave: 49 Informação: 1
Chave: 1567 Informação: 10
Chave: 57 Informação: 9
Chave: 55 Informação: 7
Chave: 52 Informação: 4
Chave: 51 Informação: 3
Chave: 53 Informação: 5
Chave: 54 Informação: 6
Chave: 56 Informação: 8
Tentativa de reinserção da chave 1567 bloqueada.
```

```
Chave: 1567 Informação: 10
Chave: 1575 Informação: 18
Chave: 1598 Informação: 20
Altura da árvore: 7
Percorrendo a árvore em ordem simétrica:
49 51 52 53 54 55 56 57 1567 1575 1598
```

4.2 Teste 2

Redirecionamento do comando `ls` para o programa.

Saída do `ls`, utilizada como entrada para o programa:

```
backup
bin
boot
dev
etc
freebsd
home
initrd
lib
lost+found
misc
mnt
opt
proc
root
sbin
tftpboot
tmp
usr
var
win
```

Saída obtida:

```
Inserindo informações...
Chave: -1396673086 Informação: backup
Chave: 97543 Informação: bin
Chave: 3029746 Informação: boot
Chave: 99349 Informação: dev
Chave: 100756 Informação: etc
Chave: -603799481 Informação: freebsd
Chave: 3208415 Informação: home
Chave: -1184075966 Informação: initrd
```

```

Chave: 107141 Informação: lib
Chave: 1247008347 Informação: lost+found
Chave: 3351788 Informação: misc
Chave: 108275 Informação: mnt
Chave: 110259 Informação: opt
Chave: 3449686 Informação: proc
Chave: 3506402 Informação: root
Chave: 3523508 Informação: sbin
Chave: -233923488 Informação: tftpboot
Chave: 114967 Informação: tmp
Chave: 116116 Informação: usr
Chave: 116519 Informação: var
Chave: 117724 Informação: win
Altura da árvore: 12
Percorrendo a árvore em ordem simétrica:
-1396673086 -1184075966 -603799481 -233923488 97543 99349 100756 107141 108275 1
10259 114967 116116 116519 117724 3029746 3208415 3351788 3449686 3506402 352350
8 1247008347

```

4.3 Teste 3

Redirecionamento do comando rwho para o programa.

Saída do rwho | awk '{print \$1}', utilizada como entrada para o programa:

```

adal      alanus::0      Jun 25 18:24
adal      alanus:pts/0   Jun 25 18:25 :07
adal      alanus:pts/1   Jun 25 18:25 :07
adal      alanus:pts/2   Jun 25 18:25 :07
adal      alanus:ttyp0   Jun 25 18:24
andrea    japura::0      Jun 25 18:24
andy      amazonas:pts/1 Jun 25 16:56 :30
andy      amazonas:pts/3 Jun 25 18:01 :05
andy      negro::0       Jun 25 16:25
andy      negro:pts/1    Jun 25 16:28 :05
andy      negro:pts/2    Jun 25 16:45 :30
apereira  epicurus:pts/5 Jun 25 14:38
blgama    tapajos::0     Jun 25 17:31
blgama    tapajos:ttyp0  Jun 25 17:31 :07
btco      sauna::0       Jun 25 17:58
cagoto    spencer::0     Jun 25 17:58
cagoto    spencer:pts/0  Jun 25 17:59 :31
cagoto    spencer:pts/1  Jun 25 18:00 :25
canettas  schiller:pts/2 Jun 25 17:12
carloshf  epicurus:pts/4 Jun 25 17:55 :34
ediniel   tiquie::0      Jun 25 16:37
eiji      prometeu::0    Jun 25 18:31
eiji      prometeu:pts/0 Jun 25 18:31

```

fjrf	tiete::0	Jun 25 13:51
fsobral	hiperion::0	Jun 25 18:25
gabsa	capibaribe::0	Jun 25 18:15
ghsilva	tamanduatei::0	Jun 25 14:56
guardian	peseta::0	Jun 25 16:57
hiroe	platon::0	Jun 25 16:03
lazo	amazonas:pts/0	Jun 25 18:05 :02
lazo	titania::0	Jun 25 18:04
lazo	titania:pts/0	Jun 25 18:05 :03
lobato	sartre::0	Jun 25 13:36
lobato	sartre:pts/0	Jun 25 14:43 :10
lobato	sartre:pts/2	Jun 25 18:13 :01
ludias	sauna:pts/4	Jun 25 16:06
magal	epicurus:pts/0	Jun 25 14:40 :27
marcel	democrito::0	Jun 25 18:04
marcel	democrito:pts/0	Jun 25 18:04 :06
marco	pascal::0	Jun 25 13:51
marco	pascal:ttyp0	Jun 25 13:51 :07
mauhcs	leibniz::0	Jun 25 14:29
mdiez	tiete:pts/1	Jun 25 13:29
mendonca	libra::0	Jun 25 18:24
milton	schiller::0	Jun 25 18:04
mmattos	euro::0	Jun 25 17:59
mmattos	euro:pts/0	Jun 25 17:59 :07
myn	dante::0	Jun 25 18:17
myn	dante:pts/0	Jun 25 18:17 :16
patty	jacuzzi::0	Jun 25 14:24
paulorrr	elara::0	Jun 25 14:01
portuga	amazonas::0	Jun 25 18:11
prnunes	oberon::0	Jun 25 13:59
prnunes	oberon:pts/0	Jun 25 13:59 :08
prnunes	oberon:pts/1	Jun 25 13:59 :44
psergio	aristoteles::0	Jun 25 13:04
queensa	hegel::0	Jun 25 17:41
queensa	hegel:pts/0	Jun 25 18:23 :09
ratoji	rousseau::0	Jun 25 14:59
ratoji	rousseau:pts/0	Jun 25 14:59
ratoji	rousseau:pts/2	Jun 25 16:03 :19
ratoji	sauna:pts/3	Jun 25 16:04 :18
rgimenes	solimoes::0	Jun 25 15:44
rizia	xingu::0	Jun 25 15:38
root	coruscant:tty3	Jun 25 18:33
sato	berkeley::0	Jun 25 18:16
seiti	pandora::0	Jun 25 13:49
seiti	pandora:pts/0	Jun 25 13:49 :02
sol	iene::0	Jun 25 15:58
sol	iene:pts/1	Jun 25 17:14 :05

sone	real::0	Jun 25 17:25
tigod	madeira::0	Jun 25 17:40
tigod	madeira:pts/0	Jun 25 17:40
tigod	madeira:pts/1	Jun 25 18:12 :17
trevisan	parana::0	Jun 25 13:47
trevisan	parana:pts/0	Jun 25 17:29 :11
trevisan	parana:pts/1	Jun 25 13:50
van	marco::0	Jun 25 15:15
vanessas	euro:pts/0	Jun 25 14:30 :07
vrafael	socrates::0	Jun 25 17:10
vvecchi	madeira:pts/0	Jun 25 17:20
vvecchi	madeira:pts/1	Jun 25 17:20 :17
willian	amazonas:pts/0	Jun 25 17:17 :02
willian	amazonas:pts/2	Jun 25 17:17
yatch	araguaia::0	Jun 25 16:05
zamlutti	epicurus:pts/1	Jun 25 17:41
zanella	purus::0	Jun 25 15:00

Saída obtida:

```
Inserindo informações...
Chave: 2988942 Informação: adal
Tentativa de reinserção da chave 2988942 bloqueada.
Chave: 2988942 Informação: adal
Tentativa de reinserção da chave 2988942 bloqueada.
Chave: 2988942 Informação: adal
Tentativa de reinserção da chave 2988942 bloqueada.
Chave: 2988942 Informação: adal
Tentativa de reinserção da chave 2988942 bloqueada.
Chave: 2988942 Informação: adal
Chave: -1413260457 Informação: andrea
Chave: 2998658 Informação: andy
Tentativa de reinserção da chave 2998658 bloqueada.
Chave: 2998658 Informação: andy
Tentativa de reinserção da chave 2998658 bloqueada.
Chave: 2998658 Informação: andy
Tentativa de reinserção da chave 2998658 bloqueada.
Chave: 2998658 Informação: andy
Tentativa de reinserção da chave 2998658 bloqueada.
Chave: 2998658 Informação: andy
Chave: 870014191 Informação: apereira
Chave: -1386405064 Informação: blgama
Tentativa de reinserção da chave -1386405064 bloqueada.
Chave: -1386405064 Informação: blgama
Chave: 3034174 Informação: btco
Chave: -1367920959 Informação: cagoto
Tentativa de reinserção da chave -1367920959 bloqueada.
```

Chave: -1367920959 Informação: cagoto
Tentativa de reinserção da chave -1367920959 bloqueada.
Chave: -1367920959 Informação: cagoto
Chave: -120872249 Informação: canettas
Chave: -40710 Informação: carloshf
Chave: -1888142772 Informação: ediniel
Chave: 3113187 Informação: eiji
Tentativa de reinserção da chave 3113187 bloqueada.
Chave: 3113187 Informação: eiji
Chave: 3144184 Informação: fjrf
Chave: -566009667 Informação: fsobral
Chave: 98110230 Informação: gabsa
Chave: 10470848 Informação: ghsilva
Chave: -1530021487 Informação: guardian
Chave: 99287335 Informação: hiroe
Chave: 3314538 Informação: lazo
Tentativa de reinserção da chave 3314538 bloqueada.
Chave: 3314538 Informação: lazo
Tentativa de reinserção da chave 3314538 bloqueada.
Chave: 3314538 Informação: lazo
Chave: -1097491715 Informação: lobato
Tentativa de reinserção da chave -1097491715 bloqueada.
Chave: -1097491715 Informação: lobato
Tentativa de reinserção da chave -1097491715 bloqueada.
Chave: -1097491715 Informação: lobato
Chave: -1091883904 Informação: ludias
Chave: 103655614 Informação: magal
Chave: -1081313748 Informação: marcel
Tentativa de reinserção da chave -1081313748 bloqueada.
Chave: -1081313748 Informação: marcel
Chave: 103666250 Informação: marco
Tentativa de reinserção da chave 103666250 bloqueada.
Chave: 103666250 Informação: marco
Chave: -1081219625 Informação: mauhcs
Chave: 103747047 Informação: mdiez
Chave: -618668853 Informação: mendonca
Chave: -1074087677 Informação: milton
Chave: 1167383205 Informação: mmattos
Tentativa de reinserção da chave 1167383205 bloqueada.
Chave: 1167383205 Informação: mmattos
Chave: 108610 Informação: myn
Tentativa de reinserção da chave 108610 bloqueada.
Chave: 108610 Informação: myn
Chave: 106439272 Informação: patty
Chave: 1267543979 Informação: paulorrr
Chave: -392114194 Informação: portuga
Chave: -309897805 Informação: prnunes

Tentativa de reinserção da chave -309897805 bloqueada.
Chave: -309897805 Informação: prnunes
Tentativa de reinserção da chave -309897805 bloqueada.
Chave: -309897805 Informação: prnunes
Chave: -289676323 Informação: psergio
Chave: 654705400 Informação: queensa
Tentativa de reinserção da chave 654705400 bloqueada.
Chave: 654705400 Informação: queensa
Chave: -938096727 Informação: ratoji
Tentativa de reinserção da chave -938096727 bloqueada.
Chave: -938096727 Informação: ratoji
Tentativa de reinserção da chave -938096727 bloqueada.
Chave: -938096727 Informação: ratoji
Tentativa de reinserção da chave -938096727 bloqueada.
Chave: -938096727 Informação: ratoji
Chave: 1145323696 Informação: rgimenes
Chave: 108530043 Informação: rizia
Chave: 3506402 Informação: root
Chave: 3522889 Informação: sato
Chave: 109318412 Informação: seiti
Tentativa de reinserção da chave 109318412 bloqueada.
Chave: 109318412 Informação: seiti
Chave: 114064 Informação: sol
Tentativa de reinserção da chave 114064 bloqueada.
Chave: 114064 Informação: sol
Chave: 3536147 Informação: sone
Chave: 110359015 Informação: tigod
Tentativa de reinserção da chave 110359015 bloqueada.
Chave: 110359015 Informação: tigod
Tentativa de reinserção da chave 110359015 bloqueada.
Chave: 110359015 Informação: tigod
Chave: 1402501894 Informação: trevisan
Tentativa de reinserção da chave 1402501894 bloqueada.
Chave: 1402501894 Informação: trevisan
Tentativa de reinserção da chave 1402501894 bloqueada.
Chave: 1402501894 Informação: trevisan
Chave: 116515 Informação: van
Chave: -1367245004 Informação: vanessas
Chave: 707691847 Informação: vrafael
Chave: 825815174 Informação: vvecchi
Tentativa de reinserção da chave 825815174 bloqueada.
Chave: 825815174 Informação: vvecchi
Chave: 1347878212 Informação: willian
Tentativa de reinserção da chave 1347878212 bloqueada.
Chave: 1347878212 Informação: willian
Chave: 114750417 Informação: yatch
Chave: 1286925658 Informação: zamlutti

Chave: -511969533 Informação: zanella

Altura da árvore: 17

Percorrendo a árvore em ordem simétrica:

-1888142772 -1530021487 -1413260457 -1386405064 -1367920959 -1367245004 -1097491
715 -1091883904 -1081313748 -1081219625 -1074087677 -938096727 -618668853 -56600
9667 -511969533 -392114194 -309897805 -289676323 -120872249 -40710 108610 114064
116515 2988942 2998658 3034174 3113187 3144184 3314538 3506402 3522889 3536147
10470848 98110230 99287335 103655614 103666250 103747047 106439272 108530043 109
318412 110359015 114750417 654705400 707691847 825815174 870014191 1145323696 11
67383205 1267543979 1286925658 1347878212 1402501894