CEFET-MG

Algoritmos e Estrutura de Dados II — 14/10/2022

Prova 1

Professor: Gustavo Henrique Borges Martins

Aluno: _____ Matrícula: _____

Instruções para a prova:

- Preencha o cabeçalho e todas as folhas desta prova com seu nome e sua matrícula.
- Leia atentamente a todas as questões antes de resolvê-las.
- As questões desta prova foram planejadas para serem resolvidas em linguagem Java.
- Não deixe de responder nenhuma questão.
- Deixe comentários sobre as questões, eles podem ser considerados para a resolução da questão.

1. Sobre as árvores Trie e Patricia, considere o poema "Quadrilha", de Carlos Drummond de Andrade, e a tabela que indexa o texto:

   "João amava Teresa que amava Raimundo que amava Maria que amava Joaquim que amava Lill que não amava ninguém."

Chave	Código	Chave	Código	Chave	Código
João	$0100\ 1011_2$	amava	$0001\ 1101_2$	Teresa	$1110\ 1011_2$
que	$1010\ 0101_2$	Raimundo	$1101\ 1010_2$	Maria	$0110\ 0101_2$
Joaquim	$0010\ 1110_2$	Lili	$0101\ 0011_2$	não	$1001\ 1100_2$
ninguém	$1011\ 0010_2$				

   (a) (1 ponto) Desenhe a Trie que resulta da inserção sucessiva das chaves do poema em uma árvore inicialmente vazia.

   (b) (1 ponto) Qual a altura h desta árvore?

   (c) (1 ponto) Desenhe a Patricia que resulta da inserção sucessiva das chaves do poema em uma árvore inicialmente vazia.

   (d) (1 ponto) Qual a altura h desta árvore?

   (e) (1 ponto) Faça uma pesquisa pelas chaves "amava", "ninguém"e "Lill". Mostre o caminho percorrido para cada pesquisa.

2. Sobre as árvores SBBs:
   (a) (1 ponto) Desenhe a SBB que resulta da inserção sucessiva das chaves [77 57 82 65 37 28 18 98 70 91 41 14 67 69 13] em uma árvore inicialmente vazia.
   (b) (1 ponto) Qual a altura vertical h desta árvore?
   (c) (1 ponto) Qual a altura máxima de caminhada k desta árvore?
   (d) (1 ponto) Quais os custos de se procurar os elementos "89", "10", "62"e "8"?
   (e) (1 ponto) Desenhe a árvore resultante da retirada dos elementos "69", "70", "67"e "13".

3. Analise o seguinte trecho de código e responda:
   (a) (3 pontos) Parte teórica:
      i. Quais as vantagens das SBBs sobre as árvores binárias de busca?
      ii. Quais as diferenças entre as SBBs e as árvores binárias de busca balanceadas?
      iii. Em função do número de trocas realizadas no balanceamento das árvores, quais os custos necessários para manter uma árvore binária de busca balanceada e uma SBB? Indique o melhor e o pior caso usando a notação Θ do número de elementos n.
   (b) (3 pontos) Análise de código:
      i. Para que servem os métodos **ee**, **ed**, **dd** e **de** (os métodos **dd** e **de** não estão escritos no código)?
      ii. Para que servem os métodos **esqCurto** e **dirCurto** (este último não está escrito no código)?
      iii. A remoção de um elemento que possui duas subárvores, utiliza de qual lado para fazer a retirada do elemento?
   (c) (4 pontos) Implementação de código: (Escolha duas opções para resolver)
      i. Escreva o código necessário para fazer impressão ordenada dos elementos da árvore.
      ii. Escreva o código complementar para a transformação horizontal direita - horizontal direita.
      iii. Escreva o código complementar para a transformação horizontal direita - horizontal esquerda.
      iv. Escreva o código complementar para a retirada de um nó curto à direita.

Questões	1	2	3	Total
Total de pontos	5	5	10	20
Pontos obtidos				

```
1  public class ArvoreSBB {
2      private static class No {
3          Item reg;
4          No esq, dir;
5          byte incE, incD;
6      }
7      private static final byte Horizontal = 0;
8      private static final byte Vertical = 1;
9      private No raiz;
10     private boolean propSBB;
11
12     private No ee (No ap) {
13         No tmp = ap.esq;
14         ap.esq = tmp.dir;
15         tmp.dir = ap;
16         tmp.incE = Vertical;
17         ap.incE = Vertical;
18         ap = tmp;
19         return ap;
20     }
21     private No ed (No ap) {
22         No tmpe = ap.esq;
23         No tmpd = tmpe.dir;
24         tmpe.incD = Vertical;
25         ap.incE = Vertical;
26         tmpe.dir = tmpd.esq;
27         tmpd.esq = tmpe;
28         ap.esq = tmpd.dir;
29         tmpd.dir = ap;
30         ap = tmpd;
31         return ap;
32     }
33     private No esqCurto (No ap) {
34         if (ap.incE == Horizontal) {
35             ap.incE = Vertical;
36             this.propSBB = true;
37         } else if (ap.incD == Horizontal) {
38             No tmp = ap.dir;
39             ap.dir = tmp.esq;
40             tmp.esq = ap;
41             ap = tmp;
42             if (ap.esq.dir.incE == Horizontal) {
43                 ap.esq = this.de (ap.esq);
44                 ap.incE = Horizontal;
45             } else if (ap.esq.dir.incD == Horizontal) {
46                 ap.esq = this.dd (ap.esq);
47                 ap.incE = Horizontal;
48             }
49             this.propSBB = true;
50         } else {
51             ap.incD = Horizontal;
52             if (ap.dir.incE == Horizontal) {
53                 ap = this.de (ap);
54                 this.propSBB = true;
```

```
55              } else if (ap.dir.incD == Horizontal) {
56                  ap = this.dd (ap);
57                  this.propSBB = true;
58              }
59          }
60          return ap;
61      }
62      private No antecessor (No q, No r) {
63          if (r.dir != null) {
64              r.dir = antecessor (q, r.dir);
65              if (!this.propSBB) r = this.dirCurto (r);
66          } else {
67              q.reg = r.reg;
68              r = r.esq;
69              if (r != null) this.propSBB = true;
70          }
71          return r;
72      }
73      private No retira (Item reg, No ap) {
74          if (ap == null) {
75              System.out.println ("Erro: Registro nao encontrado");
76              this.propSBB = true;
77          } else if (reg.compara (ap.reg) < 0) {
78              ap.esq = retira (reg, ap.esq);
79              if (!this.propSBB) ap = this.esqCurto (ap);
80          } else if (reg.compara (ap.reg) > 0) {
81              ap.dir = retira (reg, ap.dir);
82              if (!this.propSBB) ap = this.dirCurto (ap);
83          } else {
84              this.propSBB = false;
85              if (ap.dir == null) {
86                  ap = ap.esq;
87                  if (ap != null) this.propSBB = true;
88              } else if (ap.esq == null) {
89                  ap = ap.dir;
90                  if (ap != null) this.propSBB = true;
91              } else {
92                  ap.esq = antecessor (ap, ap.esq);
93                  if (!this.propSBB) ap = this.esqCurto (ap);
94              }
95          }
96          return ap;
97      }
98      ...
99  }
```