Teórico VIII

Lucas Santiago de Oliveira

Maio de 2020

1 Questão 1

```
1 //Funcao que retorna a soma de todos os inteiros de uma arvore
2 public int somarElementos() {
3     return _somarElementos(this.raiz);
4 }
5
6 //Metodo privado que percorre a arvore e soma os elementos
7 private int _somarElementos(No i) {
8     int soma = 0;
9
10    if(i.esq != null) soma += _somarElementos(i.esq);
11    if(i.dir != null) soma += _somarElementos(i.dir);
12
13    soma += i.elemento;
14
15    return soma;
16 }
```

2 Questão 2

```
1 //Funcao que retorna o numero de elementos pares presentes na
      arvore
2 public int numElementosPares() {
```

```
3     return _numElementosPares(this.raiz);
4 }
5
6 //Funcao que percorre a arvore somando os elementos presentes nela
7 private int _numElementosPares(No i) {
8     int numPares = 0;
9
10    if(i.esq != null) numPares += _somarElementos(i.esq);
11    if(i.dir != null) numPares += _somarElementos(i.dir);
12
13    if(i.elemento % 2 == 0) numPares++;
14
15    return numPares;
16 }
```

3 Questão 3

```
1 //Funcao que verifica se duas arvores sao iguais
2 public static boolean arvoresIguais(Arvore A, Arvore B) {
3   return _arvoresIguais(A.raiz, B.raiz);
4 }
5
6 //Funcao para percorrer a arvore e verificar se todos os elementos
      sao iguais
7 private static boolean _arvoresIguais(No i, No j) {
8   boolean iguais = true;
9
10  if(i.esq != null && j.esq != null) iguais = _arvoresIguais(i.esq,
       j.esq);
11  else if(i.esq != null || j.esq != null) iguais = false;  //Se ao
      deloscar para a raiz de uma arvore a outra arvore ainda tiver
      folhas elas nao sao iguais
12  if(i.dir != null && j.dir != null) iguais = _arvoresIguais(i.dir,
       j.dir);
13  else if(i.dir != null || j.dir != null) iguais = false;
14
```

```
15   if(j == null) iguais = false;
16   else if(i.elemento != j.elemento) iguais = false;
17
18   return iguais;
19 }
```

4 Questão 4

```
1 //Funcao que retorna se a arvore tem algum multiplo de 11
2 public boolean multiplo11(Arvore A) {
3   return _multiplo11(A.raiz, false);
4 }
5
6 //Funcao que percorre a arvore verificando se algum elemento eh
      divisivel por 11
7 public boolean _multiplo11(No i, boolean resp) {
8
9   if(i.elemento % 11 == 0) resp = true;
10
11  if(!resp && i.esq != null) resp = _multiplo11(i.esq, resp);
12  if(!resp && i.dir != null) resp = _multiplo11(i.dir, resp);
13
14  return resp;
15 }
```

5 Questão 5

```
1 //Metodo que transforma uma lista simples e um duplamente encadeada
      em uma arvore binaria
2 public No toArvoreBinaria(Celula i, CelulaDupla j) {
3   No cabeca = new Celula(i.elemento);
4   _toArvoreBinaria(cabeca, i.prox, j);
5   return cabeca;
6 }
```

```
7
8 private void _toArvoreBinaria(No A, Celula i, CelulaDupla j) {
9
10   if(j != null) A.inserir(j.elemento);
11   if(i != null) A.inserir(i.elemento);
12
13   if(i.prox != null || j.prox != null) _toArvoreBinaria(A, i.prox,
       j.prox);
14 }
```

6 Questão 6

```
1 //Funcao para inserir na arvore como void
2 public void inserir(int x) {
3   if(this.raiz == null) {
4     this.raiz = new No(x);
5   } else _inserir(x, this.raiz, this.raiz);
6 }
7
8 //Funcao para percorrer a arvore e inserir o elemento
9 private void _inserir(int x, No i, No pai) {
10
11   if(i == null) {
12     if(x < i.elemento) pai.esq = new No(i);
13     else pai.dir = new No(i);
14   } else if(x < i.elemento) _inserir(x, i.esq, i);
15   else if(x > i.elemento) _inserir(x, i.dir, i);
16   else if(x == i.elemento) System.err.println("Elemento nao
       inserido! Elemento ja existente na arvore!");
17 }
```

7 Questão 7

```
1 //Funcao para remover um elemento da arvore
```

```
2 public void remover(int x) throws Exception {
3   No raiz = this.raiz;
4
5   if(raiz == null) throw new Exception("Impossivel remover de
      arvore vazia!");
6   else if(x < raiz.elemento) remover(x, raiz, raiz.esq);
7   else if(x > raiz.elemento) remover(x, raiz, raiz.dir);
8   else if(x == raiz.elemento){
9     if(raiz.esq != null) raiz = raiz.esq;
10    else if(raiz.dir != null) raiz = raiz.dir;
11  } else raiz.esq = antecessor(raiz, raiz.esq);
12
13 }
14
15 //Funcao privada para remover elementos
16 private void remover(int x, No pai, No filho) {
17  if(filho != null) {
18    if(x < filho.elemento) remover(x, filho, filho.esq);
19    else if(x > filho.elemento) remover(x, filho, filho.dir);
20    else{
21      if(filho.esq != null) pai.esq = filho.esq;
22      else if(filho.dir != null) pai.dir = filho.dir;
23      else filho.esq = antecessor(filho, filho.esq);
24    }
25  } else {
26    System.err.println("Elemento nao encontrado na arvore!");
27  }
28 }
```