PUC Minas
Instituto de Ciências Exatas e Informática

Pontifícia Universidade Católica de Minas Gerais
Bacharelado em Engenharia de Software
Laboratório de Computação III
Prof.: Henrique Batista da Silva
Lista de Exercícios 3.

Árvore B

Nesta prática em laboratório, você deverá codificar e testar a implementação de árvore B.

Você deve implementar no programa principal métodos que permitam imprimir os elementos da Árvore. Além disso, você deve modificar o programa dado para testar a implementação realizada. Seu programa deve criar e inserir, pesquisar e remover uma série de elementos da Árvore B. Por fim, modifique suas implementações de modo a contabilizar para sua classe Arvore o número de comparações realizadas durante as operações de inserção, de pesquisa e de remoção.

```
public interface Item {
  public int compara (Item it);
  public void alteraChave (Object chave);
  public Object recuperaChave ();
}

public class ArvoreB {
  private static class Pagina {
    int n; Item r[]; Pagina p[];
    public Pagina (int mm) {
      this.n = 0; this.r = new Item[mm]; this.p = new Pagina[mm+1];
    }
  }
  private Pagina raiz;
  private int m, mm;
  private void imprime (Pagina p, int nivel) {
    if (p != null) {
      System.out.print ("  Nivel" + nivel + ":");
      for (int i = 0; i < p.n; i++)
        System.out.print (" "+p.r[i].toString());
      System.out.println ();
      for (int i = 0; i <= p.n; i++) {
```

```
        if (p.p[i] != null)
          if (i < p.n) System.out.println ("  Esq: "+
p.r[i].toString());
          else System.out.println ("  Dir: "+ p.r[i-1].toString());
        imprime (p.p[i], nivel + 1);
      }
    }
  }
  private Item pesquisa (Item reg, Pagina ap) {
    if (ap == null) return null; // Registro @{\it n\~ao}@ econtrado
    else {
      int i = 0;
      while ((i < ap.n-1) && (reg.compara (ap.r[i]) > 0)) i++;
      if (reg.compara (ap.r[i]) == 0) return ap.r[i];
      else if (reg.compara (ap.r[i]) < 0) return pesquisa (reg,
ap.p[i]);
      else return pesquisa (reg, ap.p[i+1]);
    }
  }

  private void insereNaPagina (Pagina ap, Item reg, Pagina apDir) {
    int k = ap.n - 1;
    while ((k >= 0) && (reg.compara (ap.r[k]) < 0)) {
      ap.r[k+1] = ap.r[k]; ap.p[k+2] = ap.p[k+1]; k--;
    }
    ap.r[k+1] = reg; ap.p[k+2] = apDir; ap.n++;
  }

  private Pagina insere (Item reg, Pagina ap, Item[] regRetorno,
                         boolean[] cresceu) {
    Pagina apRetorno = null;
    if (ap == null) { cresceu[0] = true; regRetorno[0] = reg; }
    else {
      int i = 0;
      while ((i < ap.n-1) && (reg.compara (ap.r[i]) > 0)) i++;
      if (reg.compara (ap.r[i]) == 0) {
        System.out.println ("Erro: Registro ja existente");
        cresceu[0] = false;
      }
      else {
        if (reg.compara (ap.r[i]) > 0) i++;
        apRetorno = insere (reg, ap.p[i], regRetorno, cresceu);
        if (cresceu[0])
          if (ap.n < this.mm) { // @{\it P\'agina tem espa\c{c}o}@
            this.insereNaPagina (ap, regRetorno[0], apRetorno);
            cresceu[0] = false; apRetorno = ap;
```

```
          }
          else { // Overflow: @{\it P\'agina tem que ser dividida}@
            Pagina apTemp = new Pagina (this.mm); apTemp.p[0] = null;
            if (i <= this.m) {
              this.insereNaPagina (apTemp, ap.r[this.mm-1],
ap.p[this.mm]);
              ap.n--;
              this.insereNaPagina (ap, regRetorno[0], apRetorno);
            } else this.insereNaPagina (apTemp, regRetorno[0],
apRetorno);
            for (int j = this.m+1; j < this.mm; j++) {
              this.insereNaPagina (apTemp, ap.r[j], ap.p[j+1]);
              ap.p[j+1] = null; // @{\it transfere a posse da
mem\'oria}@
            }
            ap.n = this.m; apTemp.p[0] = ap.p[this.m+1];
            regRetorno[0] = ap.r[this.m]; apRetorno = apTemp;
          }
      }
    }
    return (cresceu[0] ? apRetorno : ap);
  }

  private boolean reconstitui (Pagina apPag, Pagina apPai, int posPai)
{
    boolean diminuiu = true;
    if (posPai < apPai.n) { // @{\it aux = P\'agina a direita de
apPag}@
      Pagina aux = apPai.p[posPai+1];
      int dispAux = (aux.n - this.m + 1)/2;
      apPag.r[apPag.n++] = apPai.r[posPai]; apPag.p[apPag.n] =
aux.p[0];
      aux.p[0] = null; // @{\it transfere a posse da mem\'oria}@
      if (dispAux > 0) { // @{\it Existe folga: transfere de aux para
apPag}@
        for (int j = 0; j < dispAux - 1; j++) {
          this.insereNaPagina (apPag, aux.r[j], aux.p[j+1]);
          aux.p[j+1] = null; // @{\it transfere a posse da mem\'oria}@
        }
        apPai.r[posPai] = aux.r[dispAux - 1];
        aux.n = aux.n - dispAux;
        for (int j = 0;  j < aux.n; j++) aux.r[j] = aux.r[j+dispAux];
        for (int j = 0; j <= aux.n; j++) aux.p[j] = aux.p[j+dispAux];
        aux.p[aux.n+dispAux] = null; // @{\it transfere a posse da
mem\'oria}@
        diminuiu = false;
```

```
      }
      else { // @{\it Fus\~ao: intercala aux em apPag e libera aux}@
        for (int j = 0; j < this.m; j++) {
          this.insereNaPagina (apPag, aux.r[j], aux.p[j+1]);
          aux.p[j+1] = null; // @{\it transfere a posse da mem\'oria}@
        }
        aux = apPai.p[posPai+1] = null; /* libera aux */
        for (int j = posPai; j < apPai.n-1; j++) {
          apPai.r[j] = apPai.r[j+1]; apPai.p[j+1] = apPai.p[j+2];
        }
        apPai.p[apPai.n--] = null; // @{\it transfere a posse da
mem\'oria}@
        diminuiu = apPai.n < this.m;
      }
    }
    else { // @{\it aux = P\'agina a esquerda de apPag}@
      Pagina aux = apPai.p[posPai-1];
      int dispAux = (aux.n - this.m + 1)/2;
      for (int j = apPag.n-1; j >= 0; j--) apPag.r[j+1] = apPag.r[j];
      apPag.r[0] = apPai.r[posPai-1];
      for (int j = apPag.n; j >= 0; j--) apPag.p[j+1] = apPag.p[j];
      apPag.n++;
      if (dispAux > 0) { // @{\it Existe folga: transfere de aux para
apPag}@
        for (int j = 0; j < dispAux - 1; j++) {
          this.insereNaPagina (apPag, aux.r[aux.n-j-1], aux.p[aux.n-
j]);
          aux.p[aux.n-j] = null; // @{\it transfere a posse da
mem\'oria}@
        }
        apPag.p[0] = aux.p[aux.n - dispAux + 1];
        aux.p[aux.n - dispAux + 1] = null; // @{\it transfere a posse
da mem\'oria}@
        apPai.r[posPai-1] = aux.r[aux.n - dispAux];
        aux.n = aux.n - dispAux; diminuiu = false;
      } //@\lstcontinue@
      else { // @{\it Fus\~ao: intercala apPag em aux e libera apPag}@
        for (int j = 0; j < this.m; j++) {
          this.insereNaPagina (aux, apPag.r[j], apPag.p[j+1]);
          apPag.p[j+1] = null; // @{\it transfere a posse da
mem\'oria}@
        }
        apPag = null; /* libera apPag */
        apPai.p[apPai.n--] = null; // @{\it transfere a posse da
mem\'oria}@
        diminuiu = apPai.n < this.m;
```

```
      }
    }
    return diminuiu;
  }

  private boolean antecessor (Pagina ap, int ind, Pagina apPai) {
    boolean diminuiu = true;
    if (apPai.p[apPai.n] != null) {
      diminuiu = antecessor (ap, ind, apPai.p[apPai.n]);
      if (diminuiu) diminuiu=reconstitui
(apPai.p[apPai.n],apPai,apPai.n);
    }
    else {
      ap.r[ind] = apPai.r[--apPai.n]; diminuiu = apPai.n < this.m;
    }
    return diminuiu;
  }
  private Pagina retira (Item reg, Pagina ap, boolean[] diminuiu) {
    if (ap == null) {
      System.out.println ("Erro: Registro nao encontrado");
      diminuiu[0] = false;
    }
    else {
      int ind = 0;
      while ((ind < ap.n-1) && (reg.compara (ap.r[ind]) > 0)) ind++;
      if (reg.compara (ap.r[ind]) == 0) { // achou
        if (ap.p[ind] == null) { // @{\it P\'agina folha}@
          ap.n--; diminuiu[0] = ap.n < this.m;
          for (int j = ind; j < ap.n; j++) {
            ap.r[j] = ap.r[j+1]; ap.p[j] = ap.p[j+1];
          }
          ap.p[ap.n] = ap.p[ap.n+1];
          ap.p[ap.n+1] = null; // @{\it transfere a posse da
mem\'oria}@
        }
        else { // @{\it P\'agina n\~ao \'e folha: trocar com
antecessor}@
          diminuiu[0] = antecessor (ap, ind, ap.p[ind]);
          if (diminuiu[0]) diminuiu[0] = reconstitui (ap.p[ind], ap,
ind);
        }
      }
      else { // @{\it n\~ao achou}@
        if (reg.compara (ap.r[ind]) > 0) ind++;
        ap.p[ind] = retira (reg, ap.p[ind], diminuiu);
```

```
        if (diminuiu[0]) diminuiu[0] = reconstitui (ap.p[ind], ap,
ind);
      }
    }
    return ap;
  }
  public ArvoreB (int m) {
    this.raiz = null; this.m = m; this.mm = 2*m;
  }

  public Item pesquisa (Item reg) {
    return this.pesquisa (reg, this.raiz);
  }

  public void insere (Item reg) {
    Item regRetorno[] = new Item[1];
    boolean cresceu[] = new boolean[1];
    Pagina apRetorno = this.insere (reg, this.raiz, regRetorno,
cresceu);
    if (cresceu[0]) {
      Pagina apTemp = new Pagina(this.mm);
      apTemp.r[0] = regRetorno[0];
      apTemp.p[0] = this.raiz;
      apTemp.p[1] = apRetorno;
      this.raiz = apTemp; this.raiz.n++;
    } else this.raiz = apRetorno;
  }

  public void retira (Item reg) {
    boolean diminuiu[] = new boolean[1];
    this.raiz = this.retira (reg, this.raiz, diminuiu);
    if (diminuiu[0] && (this.raiz.n == 0)) { // @{\it \'Arvore diminui
na altura}@
      this.raiz = this.raiz.p[0];
    }
  }

  public void imprime () {
    System.out.println ("ARVORE:");
    this.imprime (this.raiz, 0);
  }
}
```

Referência:

ZIVIANI, Nivio. Projeto de algoritmos: com implementações em Java e C++. São Paulo: Thomson Learning, c2007. 621 p