Avaliação #3

BTree

Matheus Henrique
Vinicius Gasparini
Professor Douglas Dutra
06/11/2018

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

```java
public class BTree {
    static Scanner scan = new Scanner(System.in);
    static NumberFormat formatter = new DecimalFormat("#0.0000");

    public static void main(String[] args) throws Exception {
        String line;
        long cont = 1;
        String conts;

        System.out.println("\n\nDigite o tamanho M maximo de filhos por nodo (M = 2k, k >= 1)\n ->");
        File file = new File("//home//gasp//git//PRA//data.txt");
        Scanner sc = new Scanner(file);

        Tree<String, String> st = new Tree<>(scan.nextInt());

        long startRead = System.currentTimeMillis();

        while (sc.hasNextLine()){
          line = sc.nextLine();
          conts = String.valueOf(cont);
          st.put(conts, line);
          cont+= 69;
        }

        long endRead = System.currentTimeMillis();
        System.out.println("Tempo de inserção:" +formatter.format((endRead - startRead)/1000f)+" s");

        System.out.println("\n\nDigite o indice K do registro a ser procurado (N = 1+69k, k>= 0)\n ->");
        String index = Integer.toString((scan.nextInt())*69+1);
        long startFind = System.currentTimeMillis();
        System.out.println(st.get(index));
        long endFind = System.currentTimeMillis();
        double process2 = (endFind - startFind)/(1000f);
        System.out.println("Tempo de busca:" +formatter.format(process2)+" s");

        System.out.println("\n\nDeseja imprimir a arvore criada?\n1 - Sim\n0 - Não");
        if(scan.nextInt() == 1) System.out.println(st);
        else System.out.println("\nFinalizando...");
    }
}
```

UDESC

```java
public class Tree<Key extends Comparable<Key>, Value>  {
    private static int M;     // max children per B-tree node = M-1 for M even and greater than 2

    private Node root;       // root of the B-tree
    private int height;      // height of the B-tree
    private int n;           // number of key-value pairs in the B-tree

    // B-tree node data type
    private static final class Node {
        private int m;                              // number of children
        private Entry[] children = new Entry[M];    // the array of children

        // create a node with k childrens
        private Node(int k){
            m = k;
        }
    }

    private static class Entry {
        private Comparable key;
        private final Object val;
        private Node next;     // @next to iterate over array entries

        public Entry(Comparable key, Object val, Node next) {
            this.key  = key;
            this.val  = val;
            this.next = next;
        }
    }
```

```java
    private static class Entry {
        private Comparable key;
        private final Object val;
        private Node next;     // @next to iterate over array entries

        public Entry(Comparable key, Object val, Node next) {
            this.key  = key;
            this.val  = val;
            this.next = next;
        }
    }

    /**
     * Initializes an empty B-tree and sets M.
     * @param M
     */
    public Tree(int M) {
        Tree.M = M;
        root = new Node(0);
    }

    /**
     * Returns true if table is empty.
     * @return true if this table is empty
     */
    public boolean isEmpty() {
        return size() == 0;
    }

    /**
     * Returns the number of key-value pairs in this table.
     * @return the number of key-value pairs in this table
     */
    public int size() {
        return n;
    }

    /**
     * Returns the height of this B-tree.
     * @return the height of this B-tree
     */
    public int height() {
        return height;
    }
```

UDESC

```java
/**
 * Returns the value associated with the given key.
 *
 * @param  key the key
 * @return the value associated with the given key if the key is in the table
 */
public Value get(Key key) {
    return search(root, key, height);
}

private Value search(Node x, Key key, int ht) {
    Entry[] children = x.children;

    // external node
    if (ht == 0) {
        for (int j = 0; j < x.m; j++) {
            if (eq(key, children[j].key)) return (Value) children[j].val;
        }
    }

    // internal node
    else {
        for (int j = 0; j < x.m; j++) {
            if (j+1 == x.m || less(key, children[j+1].key))
                return search(children[j].next, key, ht-1);
        }
    }
    return null;
}
```

```java
/**
 * Inserts the key-value pair into the table, overwriting the old value
 * with the new value if the key is already in the table.
 * If the value is null, deletes the key from the table.
 *
 * @param  key the key
 * @param  val the value
 */
public void put(Key key, Value val) {
    Node u = insert(root, key, val, height);
    n++;
    if (u == null) return;

    // split root
    Node t = new Node(2);
    t.children[0] = new Entry(root.children[0].key, null, root);
    t.children[1] = new Entry(u.children[0].key, null, u);
    root = t;
    height++;
}
```

```
private Node insert(Node h, Key key, Value val, int ht) {
    int j;
    Entry t = new Entry(key, val, null);

    // external node
    if (ht == 0) {
        for (j = 0; j < h.m; j++) {
            if (less(key, h.children[j].key)) break;
        }
    }

    // internal node
    else {
        for (j = 0; j < h.m; j++) {
            if ((j+1 == h.m) || less(key, h.children[j+1].key)) {
                Node u = insert(h.children[j++].next, key, val, ht-1);
                if (u == null) return null;
                t.key = u.children[0].key;
                t.next = u;
                break;
            }
        }
    }

    for (int i = h.m; i > j; i--)
        h.children[i] = h.children[i-1];
    h.children[j] = t;
    h.m++;
    if (h.m < M) return null;
    else         return split(h);
}
```

```java
    // split node in half
    private Node split(Node h) {
        Node t = new Node(M/2);
        h.m = M/2;
        for (int j = 0; j < M/2; j++)
            t.children[j] = h.children[M/2+j];
        return t;
    }

    /**
     * Returns representation of this B-tree.
     * @return representation of this B-tree.
     */
    public String toString() {
        return toString(root, height, "") + "\n";
    }

    private String toString(Node h, int ht, String indent) {
        StringBuilder s = new StringBuilder();
        Entry[] children = h.children;

        if (ht == 0) {
            for (int j = 0; j < h.m; j++) {
                s.append(indent + children[j].key + " " + children[j].val + "\n");
            }
        }
        else {
            for (int j = 0; j < h.m; j++) {
                if (j > 0) s.append(indent + "(" + children[j].key + ")\n");
                s.append(toString(children[j].next, ht-1, indent + "     "));
            }
        }
        return s.toString();
    }


    // comparison functions - make Comparable instead of Key to avoid casts
    private boolean less(Comparable k1, Comparable k2) {
        return k1.compareTo(k2) < 0;
    }

    private boolean eq(Comparable k1, Comparable k2) {
        return k1.compareTo(k2) == 0;
    }
}
```

UDESC

Obrigado

matheusheniquecct@hotmail.com
viniciuszeiko@gmail.com

UDESC