TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN: DỰNG CÂY BST VÀ AVL

TÌM HIỂU NODE JS

Người hướng dẫn: **G.V Trần Lương Quốc Đại**
Người thực hiện: **Trần Quốc Lĩnh - 51703124**
Lớp: **17050301**
Khoá: **21**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2019

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN: DỰNG CÂY BST VÀ AVL

TÌM HIỂU NODE JS

Người hướng dẫn: **G.V Trần Lương Quốc Đại**
Người thực hiện: **Trần Quốc Lĩnh - 51703124**
Lớp: **17050301**
Khoá: **21**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2019

LỜI CẢM ƠN

Cảm ơn thầy đã dạy môn cấu trúc dữ liệu và giải thuật 2 hết sức nhiệt tình và tâm huyết, cảm ơn thầy đã định hướng và tạo cơ sở để em có thể tự tìm hiểu và học thêm về các giải thuật.

Còn đây là bài tập lớn, một nội dung trong trương trình giảng dạy mà thầy đã giao cho em. Trong quá trình làm bài tập lớn này, em vẫn còn nhiều thiếu sót. Em rất mong nhận được sự đánh giá và chỉ bảo từ thầy!

BÀI TẬP LỚN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Em xin cam đoan đây là sản phẩm bài tập lớn của riêng em. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu ,hình ảnh được chính em thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong bài tiểu luận còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào em xin hoàn toàn chịu trách nhiệm về nội dung bài tập lớn của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do em gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 31 tháng 03 năm 2019

Tác giả

(Đã ký)

Trần Quốc Lĩnh

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

Tp. Hồ Chí Minh, ngày 31 tháng 03 năm 2019
(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

Tp. Hồ Chí Minh, ngày 31 tháng 03 năm 2019
(kí và ghi họ tên)

TÓM TẮT

Đây là phần trình bày về cách thực thi các thuật toán và mức độ hoàn thành yêu cầu của để bài.

Gồm có các nội dung chính:

- Xây dựng cấu trúc dữ liệu của cây nhị phân tìm kiếm (BST) và cây nhị phân tìm kiếm cân bằng (AVL).
- Xây dựng các giải thuật trên hai cấu trúc dữ liệu BST và AVL.
- Cách gọi và chạy các phương thức.
- Hiện thực trực quan hoá.

MỤC LỤC

LỜI CẢM ƠN 3
CAM KẾT 4
ĐÁNH GIÁ CỦA GIÁO VIÊN 5
TÓM TẮT 6
MỤC LỤC 7
DANH MỤC CHÚ THÍCH CÁC THUẬT NGỮ VÀ HÌNH ẢNH 8
MÔ TẢ VÀ MỨC ĐỘ HOÀN THÀNH BÀI TẬP LỚN 9
1. Xây dựng cấu trúc dữ liệu 9
2. Xây dựng các giải thuật trên hai cấu trúc dữ liệu BST và AVL 10
3. Cách gọi và chạy các phương thức 16
4. Hiện thực trực quan hóa 18
TÀI LIỆU THAM KHẢO 19

DANH MỤC CHÚ THÍCH CÁC THUẬT NGỮ VÀ HÌNH ẢNH

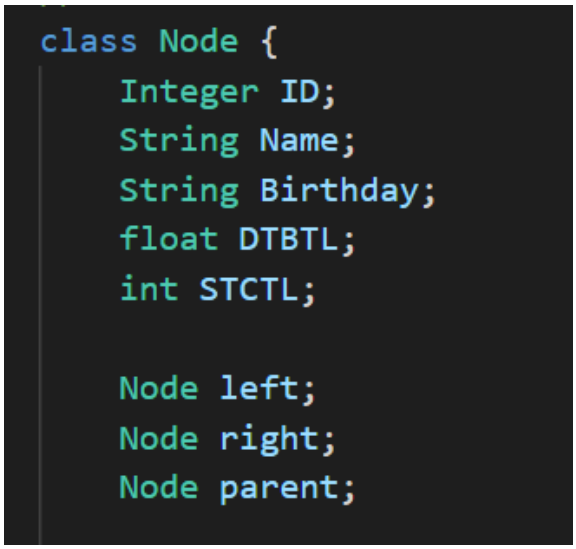
Thuật ngữ

:

Hình ảnh

MÔ TẢ VÀ MỨC ĐỘ HOÀN THÀNH BÀI TẬP LỚN

1. Xây dựng cấu trúc dữ liệu. (Hoàn thành 100%)
Mỗi node trong cây có các thành phần:



Hình 1: Cấu trúc dữ liệu của một node

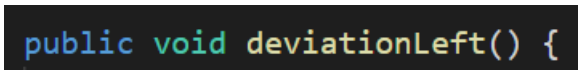
Trong đó:
- ID là mã số sinh viên, thuộc kiểu số nguyên, đóng vai trò là khóa chính của node.
- Name là họ tên của sinh viên đó, thuộc kiểu chuỗi.
- Birthday là ngày tháng năm sinh của sinh viên, thuộc kiểu chuỗi.
- DTBTL là điểm trung bình tích lũy, thuộc kiểu số thực (float).
- STCTL là số tính chỉ tích lũy, thuộc kiểu số nguyên.
- Node left, right, parent là các thành phần khác theo yêu cầu của đề bài.

2. Các giải thuật trên cây. (Hoàn thành 100%)
a) Cây BST.
o) Giải thuật dựng cây: Dựng cây rỗng, dựng cây với các thông tin ngẫu nhiên, dựng cây lệch trái, dựng cây lệch phải

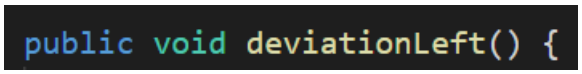
Hình 2: Dựng cây rỗng

Hình 3: Dựng cây với thông tin ngẫu nhiên

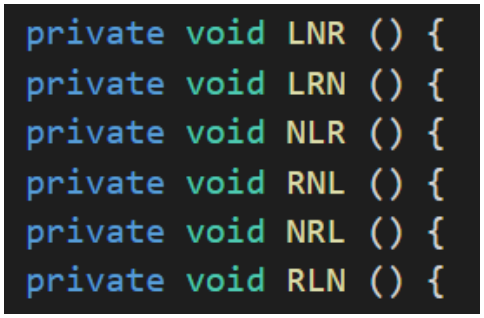
Lệch trái



Lệch phải



o Giải thuật duyệt cây theo: LNR, LRN, NLR, RNL, NRL, RLN. Hiển thị đầy đủ thông tin của một node (bao gồm: mã số sinh viên, họ tên, ngày sinh, điểm trung bình, số tín chỉ tích luỹ).



```java
private void Display (Node x)   {
    System.out.println("ID: " + x.ID);
    System.out.println("Name: " + x.Name);
    System.out.println("Birthday: " + x.Birthday);
    System.out.println("DTBTL: " + x.DTBTL);
    System.out.println("STCTL: " + x.STCTL);
    System.out.println("------------------");
}
```

Hình 4: Hiển thị thông tin

o Giải thuật tìm kiếm: theo mã số sinh viên, theo họ tên, theo ngày sinh, theo điểm trung bình, theo số tín chỉ tích luỹ, tìm phần tử lớn nhất – nhỏ nhất của cây.

```java
public Node getElementByID(Integer ID) {    return getElementByID(root, ID);    }
public void getElementByName(String Name) { Result.clear(); getElementByName(root, Name);   }
public void getElementByBirthday(String Birthday) { Result.clear(); getElementByBirthday(root, Birthday);   }
public void getElementByDTBTL(float DTBTL) {    Result.clear(); getElementByDTBTL(root, DTBTL); }
public void getElementBySTCTL(Integer STCTL) {  Result.clear(); getElementBySTCTL(root, STCTL); }
public Node getMin()    {   return getMin(root);    }
public Node getMax()    {   return getMax(root);    }
```

* Bởi vì khi tìm kiếm theo họ tên, theo ngày sinh, theo điểm trung bình tích lũy, theo số tính chỉ tích lũy không trả về giá trị duy nhất nên kết quả sẽ được lưu vào vector result.

o Giải thuật thêm phần tử vào cây: Thêm tuần tự từng node, thêm danh sách các node.

- Thêm từng node

```java
private Node Put(Node x, Integer ID, String Name, String Birthday, float DTBTL, int STCTL, Node src)    {
    if (x == null)  {
        Node child = new Node(ID, Name, Birthday, DTBTL, STCTL);
        child.parent = src;
        return child;
    }
    int cmp = ID.compareTo(x.ID);
    if (cmp < 0)
        x.left = Put(x.left, ID, Name, Birthday, DTBTL, STCTL, x);
    else if (cmp > 0)
        x.right = Put(x.right, ID, Name, Birthday, DTBTL, STCTL, x);
    else; // This value has exist, so do nothing
    return x;
}

public void Put(Node add) {
    root = Put(root, add.ID, add.Name, add.Birthday, add.DTBTL, add.STCTL, root);
}
//~~ end Put - in normal case
```

- Thêm danh sách các node

```
private void addListNode (Vector<Node> x)   {
    for (int i = 0; i < x.size(); i++)
        Put(x.get(i));
}
```

o Giải thuật xoá phần tử ra khỏi cây: Xoá tuần tự từng node, xoá danh sách node.

- Xóa từng node

```
private Node removeNode (Node r, Node x) {
    if (r == null)
        return null;
    int cmp = x.ID.compareTo(r.ID);
    if (cmp < 0)
        r.left = removeNode(r.left, x);
    else if (cmp > 0)
        r.right = removeNode(r.right, x);
    else {
        if (r.left == null)
            return r.right;
        if (r.right == null)
            return r.left;
        Node t = r;
        r = getMin(t.right);
        r.right = removeNode(t.right, getMin(t.right));
        r.left = t.left;
    }
    return r;
}
```

- Xóa danh sách các node

```
private void removeListNode (Vector<Node> x) {
    for (int i = 0; i < x.size(); i ++)
        removeNode(x.get(i));
}
```

o Giải thuật xác định phần tử liền trước (Predecessor), phần tử liền sau (Successor).

- Tìm liền trước

```
public Node getPredecessor(Node x) {
    if (x.left != null)
        return getMax(x.left);
    Node p = x.parent;
    Node T = x;
    while (p != null && T == p.left)    {
        T = p;
        p = T.parent;
    }
    return p;
}
```

- Tìm liền sau

```
public Node getSuccessor(Node x) {
    if (x.right != null)
        return getMin(x.right);
    Node p = x.parent;
    Node T = x;
    while (p != null && T == p.right)   {
        T = p;
        p = T.parent;
    }
    return p;
}
```

o Giải thuật cập nhật thông tin trên Node: Cho phép cập nhật họ tên, ngày sinh, điểm trung bình, số tín chỉ dựa vào mã số sinh viên.

```java
private void Update(Integer ID) {
    Scanner sc = new Scanner(System.in);

    Node x = getElementByID(ID);
    if (x == null)
        System.out.print("This tree do not contain ID: " + ID);
    else {
        System.out.print("New name: ");
        x.Name = sc.nextLine();
        System.out.print("New birthday: ");
        x.Birthday = sc.nextLine();
        System.out.print("New DTBTL: ");
        x.DTBTL = sc.nextFloat();
        System.out.print("New STCTL: ");
        x.STCTL = sc.nextInt();
        System.out.println("After update: ");
        Display(x);
    }
}
```

b) Cây AVL (tương tự cây BST) - Tuy nhiên cây AVL không chứa cách dựng cây lệch trái, lệch phải. Ngoài ra cây AVL còn chứa các phương thức kiểm tra và tạo ra cây cân bằng, như:

```java
public int height(Node x)   {
    return x==null?-1:Math.max(height(x.left), height(x.right)) + 1;
}

public int checkBalance(Node x) {
    return height(x.left) - height(x.right);
}

private Node rotateLeft(Node x) {
    Node y = x.right;
    x.right = y.left;
    y.left = x;
    return y;
}

private Node rotateRight(Node x) {
    Node y = x.left;
    x.left = y.right;
    y.right = x;
    return y;
}
```

```
private Node balance(Node x)    {
    if (checkBalance(x) < -1)   {
        if (checkBalance(x.right) > 0)
            x.right = rotateRight(x.right);
        x = rotateLeft(x);
    }
    else if (checkBalance(x) > 1) {
        if (checkBalance(x.left) < 0)
            x.left = rotateLeft(x.left);
        x = rotateRight(x);
    }
    return x;
}
```

3. Cách gọi và chạy các phương thức
- Để chạy hàm main, chương trình, dùng lệnh dưới đây trên cửa sổ cmd. Cây BST và cây AVL thực hiện tương tự.

```
javac BST.java && java BST
```

Mặc định đọc dữ liệu từ file input.txt (tập tin đi kèm).

- Kiểm tra các phương thức trong hàm main.
+ Tạo một obj BST để gọi các phương thức (+ Tạo cây rỗng).

```
BST bst = new BST();
```

+ Đọc dữ liệu từ file input.txt

```
bst.fileToListNode("input.txt");
```

+ Tạo cây ngẫu nhiên.

```
bst.randomBST();
```

+ Tạo cây lệch trái.

```
bst.deviationLeft();
```

+ Tạo cây lệch phải.

```
bst.deviationRight();
```

+ Các hàm duyệt cây.

```
bst.LNR();
bst.LRN();
bst.RNL();
bst.RLN();
bst.NLR();
bst.NRL();
```

+ Tìm kiếm dựa vào ID. Trong trường hợp này, ID là 10.

```
bst.getElementByID(10);
```

+ Tìm kiếm dựa vào Tên.

```
bst.getElementByID(10);
```

+ Tìm kiếm dựa vào birthday

```
bst.getElementByBirthday("4-7-2001");
```

+ Tìm kiếm dựa vào Điểm trung bình tích lũy.

```
bst.getElementByDTBTL((float)6.5);
```

+ Tìm kiếm dựa vào số tính chỉ tích lũy

```
bst.getElementBySTCTL(100);
```

+ Tìm min và max

```
System.out.println(bst.getMin().Name);
System.out.println(bst.getMax().Name);
```

+ Chèn một danh sách các node

```
bst.addListNode(bst.ListNode);
```

+ Xóa một node, trong trường hợp này là node có ID = 9

```
bst.removeNode(bst.getElementByID(9));
```

+ Xóa danh sách các node

```
bst.removeListNode(removeList);
```

+ Tìm liền trước và liền sau, trong trường hợp này là liền trước và liền sau của phần tử có ID = 59

```
System.out.print(bst.getSuccessor(bst.getElementByID(59)).ID);
System.out.print(bst.getPredecessor(bst.getElementByID(59)).ID);
```

+ Cập nhật 1 phần từ, trong trường hợp này là phần tử có ID = 10

```
bst.Update(10);
```

4. Hiện thực trực quan hóa (Chưa hoàn thành)

TÀI LIỆU THAM KHẢO

Tiếng Việt

Tiếng Anh