TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN MÔN**

**CẤU TRÚC DỮ LIỆU & GIẢI THUẬT 2**

**Xây dựng cấu trúc dữ liệu cây nhị phân tìm kiếm (BST) và cây cân bằng (AVL)**

**Hiện thực trực quan hóa (winform) cho các giải thuật**

*Người hướng dẫn*: **TS NGUYỄN DUY HÀM**

*Người thực hiện*: **ĐỖ DUY THỊNH – 51703188**

Lớp **: 17050303**

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 MÔN**

**CẤU TRÚC DỮ LIỆU & GIẢI THUẬT 2**

**Xây dựng cấu trúc dữ liệu cây nhị phân tìm kiếm (BST) và cây cân bằng (AVL)**

**Hiện thực trực quan hóa (winform) cho các giải thuật**

Người hướng dẫn: **TS NGUYỄN DUY HÀM**

Người thực hiện: **ĐỖ DUY THỊNH**

Lớp **: 17050303**

Khoá  **: 21**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2019**

LỜI CẢM ƠN

Em xin cảm ơn chân thành tới Tiến sĩ Nguyễn Duy Hàm người đã tận tình chỉ dẫn em trong suốt quá trình hoàn thành đề tài.

Em cũng xin gửi lời cảm ơn tới Tiến sĩ Nguyễn Thị Hải Bình đã giúp em giải đáp các thắc mắc về đề tài này.

Với điều kiện và vốn kiến thức còn ít, nên đề tài khi thực hiện có thể còn nhiều thiếu sót. Em rất mong nhận được sự chỉ bảo của các thầy cô để em có thể nâng cao kiến thức cho bản thân khi thực hiện các đề tài sau này.

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm bài tập lớn của riêng tôi và được sự hướng dẫn của TS Nguyễn Duy Hàm và TS Nguyễn Thị Hải Bình;. 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 trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án 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 tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Đỗ Duy Thịnh*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

Cây nhị phân tìm kiếm (BST) và cây nhị phân cân bằng (AVL) thường được sử dụng cho nhiều mục đích khác nhau và việc sử dụng cây nhị phân để lưu trữ thông tin và tìm kiếm thông tin vẫn được sử dụng rộng rãi.

MỤC LỤC

[LỜI CẢM ƠN i](#_Toc6573009)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN iii](#_Toc6573010)

[TÓM TẮT iv](#_Toc6573011)

[MỤC LỤC 1](#_Toc6573012)

[DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ 4](#_Toc6573013)

[CHƯƠNG 1 – CÂY NHỊ PHÂN TÌM KIẾM BST 5](#_Toc6573014)

[1.1 Khái niệm 5](#_Toc6573015)

[1.2 Cấu trúc Node của BST 5](#_Toc6573016)

[1.3 Các giải thuật và thuộc tính trong BST: 5](#_Toc6573017)

[1.3.1 Thuộc tính trong BST: 6](#_Toc6573018)

[1.3.2 Giải thuật Search (get()): 6](#_Toc6573019)

[1.3.3 Giải thuật Insert (put()): 6](#_Toc6573020)

[1.3.4 Giải thuật remove (delete()): 7](#_Toc6573021)

[1.3.5 Giải thuật dựng cây BST: 8](#_Toc6573022)

[1.3.5.1 Giải thuật dựng cây rỗng: 8](#_Toc6573023)

[1.3.5.2 Giải thuật dựng cây ngẫu nhiên: 8](#_Toc6573024)

[1.3.5.3 Giải thuật dụng cây Lệch Trái 9](#_Toc6573025)

[1.3.5.4 Giải thuật dựng cây lệch Phải: 10](#_Toc6573026)

[1.3.6 Giải thuật duyệt cây LNR, LRN, NLR, RNL, NRL, RLN: 11](#_Toc6573027)

[1.3.7 Giải thuật tìm kiếm 13](#_Toc6573028)

[1.3.7.1 Giải thuật tìm kiếm theo tên: 13](#_Toc6573029)

[1.3.7.2 Giải thuật tìm kiếm theo ngày sinh 13](#_Toc6573030)

[1.3.7.3 Giải thuật tìm kiếm theo điểm trung bình 14](#_Toc6573031)

[1.3.7.4 Giải thuật tìm kiếm theo tín chỉ tích lũy 14](#_Toc6573032)

[1.3.7.5 Giải thuật tìm kiếm Node lớn nhất, nhỏ nhất 15](#_Toc6573033)

[1.3.8 Giải thuật predecessor và successor: 15](#_Toc6573034)

[1.3.9 Giải thuật cập nhật thông tin cho Node: 17](#_Toc6573035)

[CHƯƠNG 2: CÂY NHỊ PHÂN TÌM KIẾM CÂN BẰNG (AVL) 19](#_Toc6573036)

[2.1 Khái niệm: 19](#_Toc6573037)

[2.2 Cấu trúc Node cây cân bằng 19](#_Toc6573038)

[2.3 Các thuộc tính và giải thuật trong cây AVL: 20](#_Toc6573039)

[2.4 Giải thuật trong cây AVL 20](#_Toc6573040)

[2.4.1 Giải thuật kiểm tra độ cân bằng cây: 20](#_Toc6573041)

[2.4.2 Giải thuật balance: 21](#_Toc6573042)

[2.4.3 Giải thuật xoay trái: 21](#_Toc6573043)

[2.4.4 Giải thuật xoay phải 22](#_Toc6573044)

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**DANH MỤC HÌNH**

[Hình 1.1: Cấu trúc của lớp Node 5](#_Toc6573046)

[Hình 1.2 Node parent của BST 6](#_Toc6573047)

[Hình 1.3: Phương thức get() BST 6](#_Toc6573048)

[Hình 1.4: Phương thức put() BST 7](#_Toc6573049)

[Hình 1.5: Phương thức delete() trong BST 8](#_Toc6573050)

[Hình 1.6: Phương thức tạo cây BST Ngẫu Nhiên 9](#_Toc6573051)

[Hình 1.7: Giải thuật dựng cây lệch trái 10](#_Toc6573052)

[Hình 1.8: Giải thuật dựng cây lệch phải 11](#_Toc6573053)

[Hình 1.9: Giải thuật duyệt cây LNR 12](#_Toc6573054)

[Hình 1.10: Giải thuật duyệt cây LRN 12](#_Toc6573055)

[Hình 1.11: Giải thuật duyệt cây NLR 12](#_Toc6573056)

[Hình 1.12: Giải thuật duyệt cây NRL 13](#_Toc6573057)

[Hình 1.13: Giải thuật tìm kiếm theo tên 13](#_Toc6573058)

[Hình 1.14: Giải thuật tìm kiếm theo ngày sinh 14](#_Toc6573059)

[Hình 1.15: Giải thuật tìm kiếm theo điểm trung bình 14](#_Toc6573060)

[Hình 1.16: Giải thuật tìm kiếm theo tín chỉ 15](#_Toc6573061)

[Hình 1.17: Giải thuật tìm kiếm min, max 15](#_Toc6573062)

[Hình 1.18: Giải thuật tìm kiếm khóa liền trước 16](#_Toc6573063)

[Hình 1.19: Giải thuật tìm kiếm khóa liền sau 17](#_Toc6573064)

[Hình 1.20: Giải thuật cập nhật tên sinh viên 18](#_Toc6573065)

[Hình 1.21: Giải thuật cập nhật ngày sinh sinh viên 18](#_Toc6573066)

[Hình 1.22: Giải thuật cập nhật điểm trung bình tích lũy sinh viên 18](#_Toc6573067)

[Hình 1.23: Giải thuật cập nhật tín chỉ tích lũy cho sinh viên 19](#_Toc6573068)

[Hình 2.24: Cấu trúc Node của cây cân bằng 20](#_Toc6573069)

[Hình 2.25: Phương thức kiểm tra độ lệch của cây AVL 21](#_Toc6573070)

[Hình 2.26 Giải thuật cân bằng trong cây AVL 21](#_Toc6573071)

[Hình 2.27: Giải thuật xoay trái cây AVL 22](#_Toc6573072)

[Hình 2.28: Giải thuật xoay phải 22](#_Toc6573073)

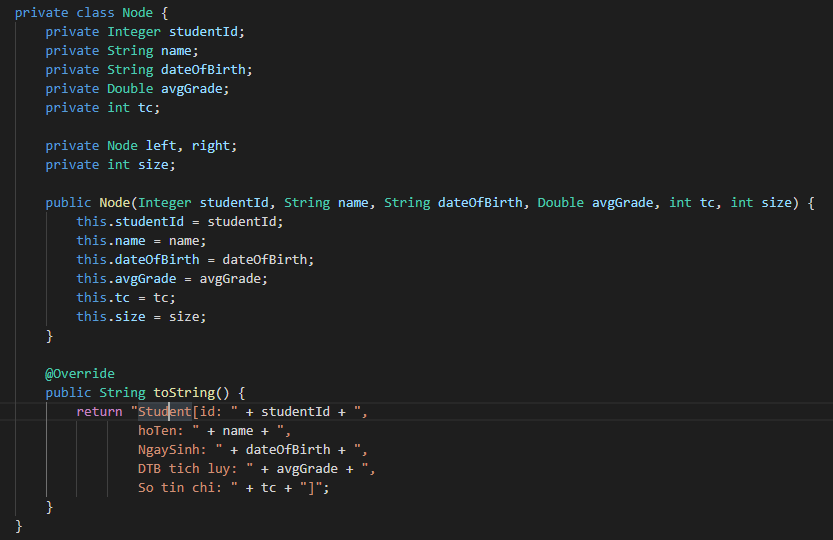
CHƯƠNG 1 – CÂY NHỊ PHÂN TÌM KIẾM BST

1.1 Khái niệm

Cây nhị phân tìm kiếm (Binary Search Tree – BST) là một cấu trúc dữ liệu mở rộng của cây nhị phân thỏa tính chất quan trọng: giá trị được lưu ở node hiện tại lớn hơn giá trị lưu ở node bên trái và phải nhỏ hơn giá trị lưu ở node bên phải.

1.2 Cấu trúc Node của BST

Một lớp Node (private) được tạo bên trong lớp Cây nhị phân tìm kiếm (BST) chứa thông tin của một sinh viên bao gồm: mã số sinh viên (studentId) kiểu Integer gồm 3 chữ số được xem là key của cây nhị phân. Họ tên (name) kiểu String, ngày sinh (dateOfBirth) kiểu String, điểm trung bình tích lũy (avgGrade) kiểu Double và số tín chỉ tích lũy (tc) kiểu int. Có thành phần Left, Right.



Hình 1.1: Cấu trúc của lớp Node

1.3 Các giải thuật và thuộc tính trong BST:

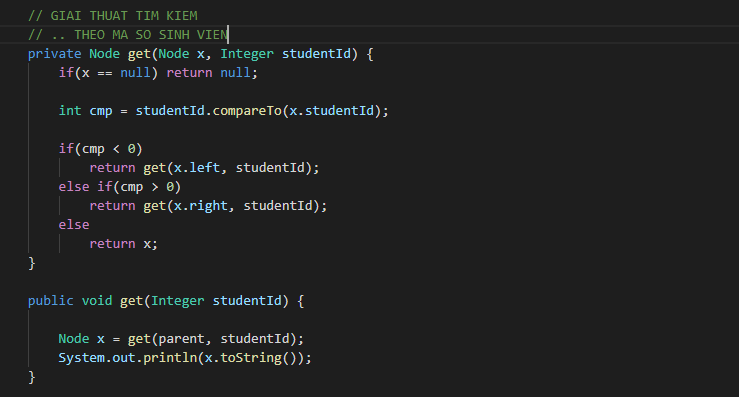
1.3.1 Thuộc tính trong BST:

Node parent: chứa gốc (root) của cây nhị phân.

Hình 1.2 Node parent của BST

1.3.2 Giải thuật Search (get()):

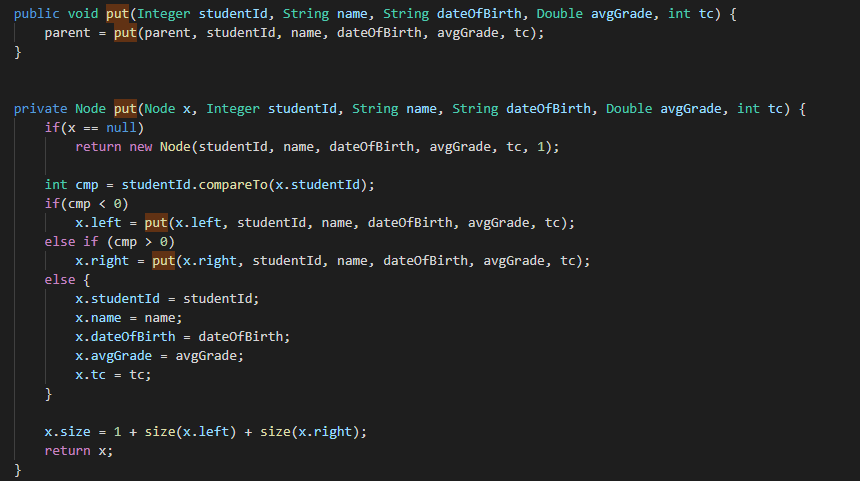
Bằng cách sử dụng đệ qui để tìm một Node trong cây dựa vào khóa (studentId). Nếu root trống thì Node trả về là null. Ngược lại, nếu khóa bé hơn giá trị của Node hiện tại thuật toán sẽ duyệt về nhánh phía trái để tìm, ngược lại nếu khóa lớn hơn giá trị của Node hiện tại thuật toán sẽ duyệt về nhánh phía bên phải để tìm.



Hình 1.3: Phương thức get() BST

1.3.3 Giải thuật Insert (put()):

Thuật toán cũng tương tự như search như trên. Phương thức put() chỉ khác khi thực hiện thuật toán nếu parent là null (cây rỗng) thì cây sẽ tự tạo một Node mới và thêm vào cây nhị phân tìm kiếm. Nếu khóa thêm vào có giá trị bé hơn giá trị của Node hiện tại thì Node mới sẽ được thêm vào nhánh bên trái, ngược lại nếu giá trị của Node thêm vào lớn hơn giá trị của Node hiện tại thì Node mới sẽ được thêm vào nhánh bên phải. Sau khi hoàn tất thực hiện đệ qui thì cây sẽ cập nhật lại size (số lượng Node trong cây).



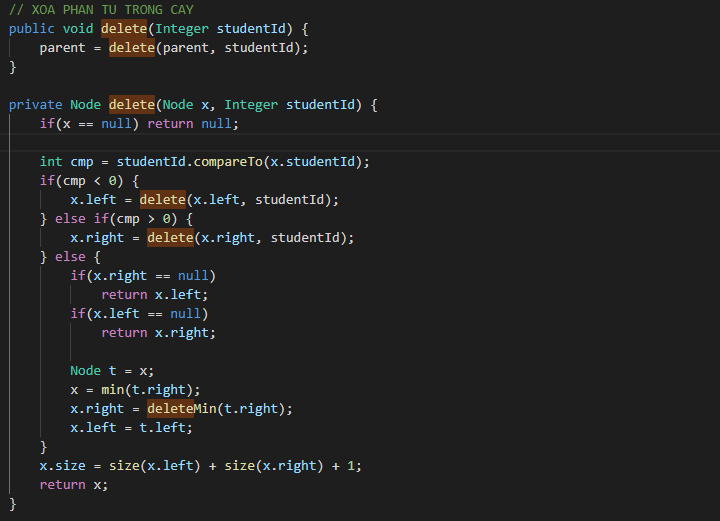
Hình 1.4: Phương thức put() BST

1.3.4 Giải thuật remove (delete()):

Thuật toán để tìm một node để xóa khỏi cây nhị phân cũng áp dụng đệ qui. Nếu giá trị của Node đang tìm bé hơn Node hiện tại, thuật toán sẽ duyệt nhánh bên trái cây và ngược lại. Sau đó nếu đã tìm được Node phù hợp, thuật toán sẽ tiến hành các bước như sau để xóa:

* Trường hợp 1: Nếu Node cần xóa (x) có nhánh trái là null, return về nhánh phải.
* Trường hợp 2: Nếu Node cần xóa (x) có nhánh phải là null, return về nhánh trái.
* Trường hợp 3: Nếu Node cần xóa (x) có 2 nhánh con. Ta sẽ tạo một Node tạm thời (t) cho Node x. Gán Node x là giá trị min của nhánh bên phải Node t, sau đó gọi hàm xóa Node nhỏ nhất bên nhánh phải của Node x, và gán Node trái của x là Node trái của t.

Khi đã xóa Node xong, ta sẽ cập nhật lại size cho cây nhị phân tìm kiếm



Hình 1.5: Phương thức delete() trong BST

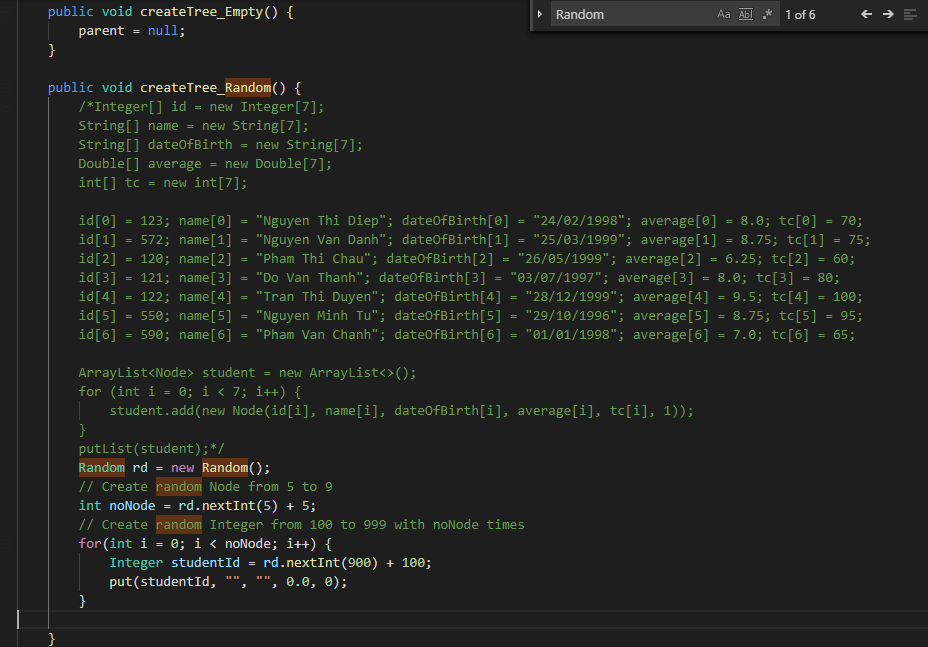
1.3.5 Giải thuật dựng cây BST:

1.3.5.1 Giải thuật dựng cây rỗng:

Tạo một constructor trong lớp BST. Khi gọi constructor, một cây nhị phân tìm kiếm sẽ tạo ra có root của cây là null.

1.3.5.2 Giải thuật dựng cây ngẫu nhiên:

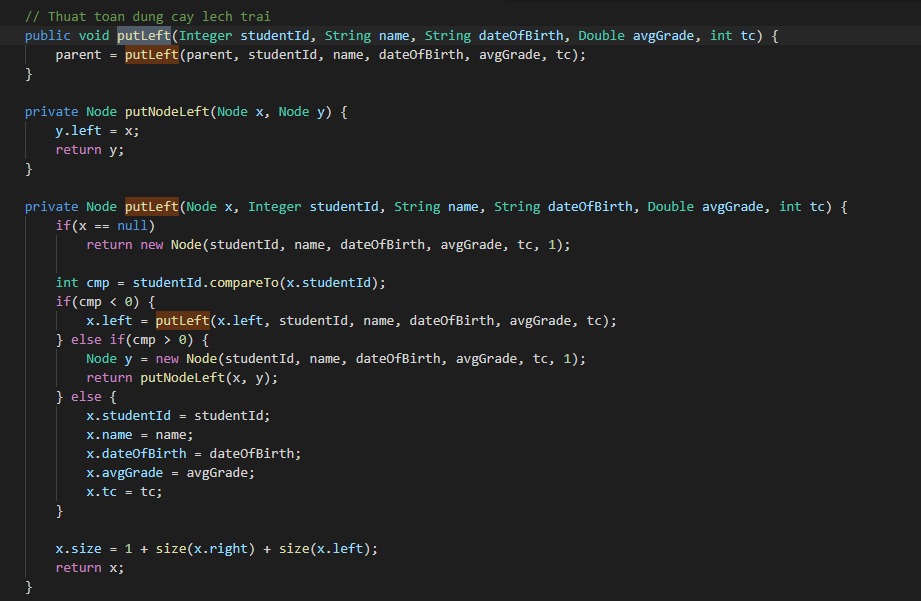
Tạo một đối tượng rd (lớp Random) để thực hiện chọn số ngẫu nhiên cho số Node trong cây (từ 10 đến 15 Node), và chọn ngẫu nhiên mã số sinh viên (giá trị từ 100 đến 999). Sau đó dùng giải thuật put() để đưa Node vào cây.



Hình 1.6: Phương thức tạo cây BST Ngẫu Nhiên

1.3.5.3 Giải thuật dụng cây Lệch Trái

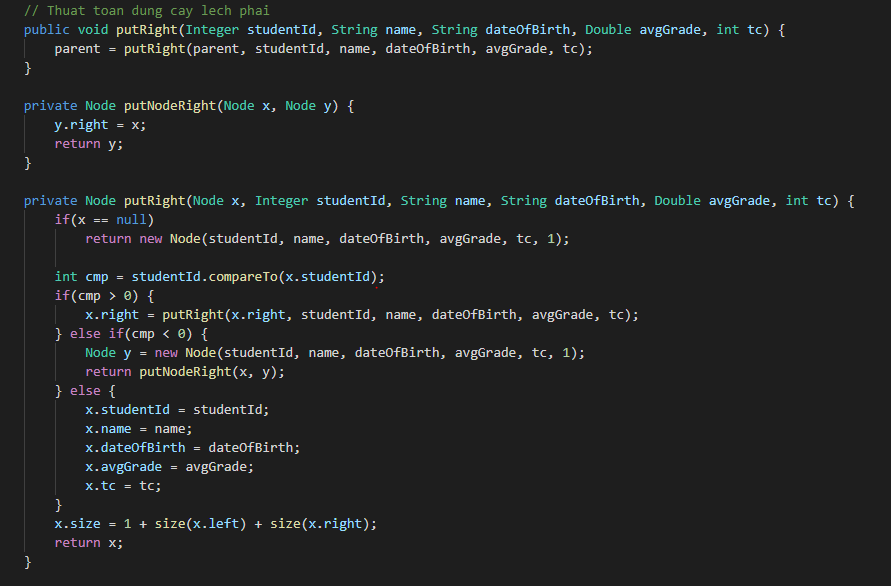
Giải thuật dựng cây lệch trái cũng tương tự như giải thuật put(). Điểm khác nhau là khi put() một Node có giá trị lớn hơn Node x đang xét, thuật toán sẽ gán Node x là Node y (Node y đang giữ nhánh trái của x). Lúc này cây sẽ lệch trái.



Hình 1.7: Giải thuật dựng cây lệch trái

1.3.5.4 Giải thuật dựng cây lệch Phải:

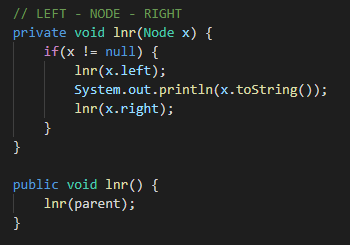
Giải thuật dựng cây lệch phải tương tự như cây lệch trái. Khi thêm một Node có giá trị bé hơn Node x đang xét, thuật toán sẽ gán Node x là Node y (Node y đang giữ nhánh phải của x). Lúc này cây sẽ lệch phải.

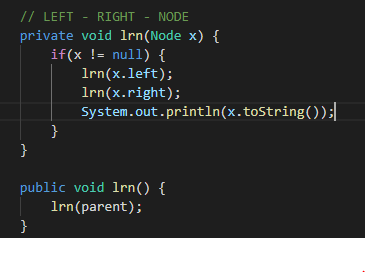


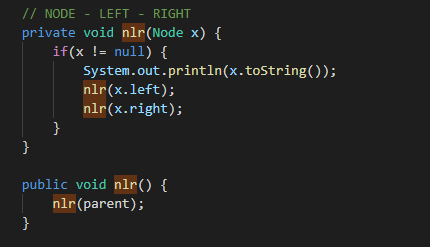
Hình 1.8: Giải thuật dựng cây lệch phải

1.3.6 Giải thuật duyệt cây LNR, LRN, NLR, RNL, NRL, RLN:

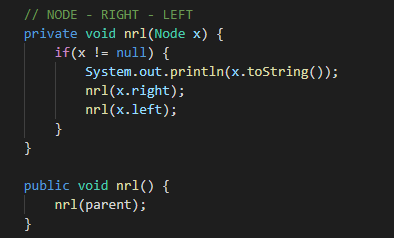
Giải thuật sử dụng đệ qui để duyệt toàn bộ cây nhị phần và in thông tin của từng Node ra màn hình.



Hình 1.9: Giải thuật duyệt cây LNR

Hình 1.10: Giải thuật duyệt cây LRN

Hình 1.11: Giải thuật duyệt cây NLR

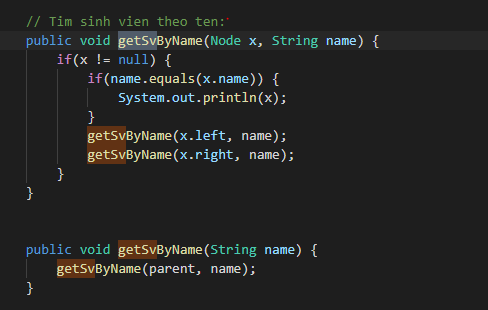


Hình 1.12: Giải thuật duyệt cây NRL

1.3.7 Giải thuật tìm kiếm

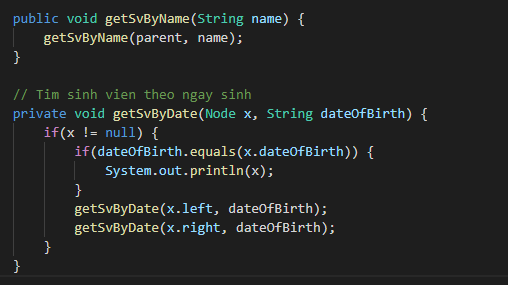
1.3.7.1 Giải thuật tìm kiếm theo tên:

Giải thuật sử dụng đệ qui để duyệt toàn bộ cây. Node nào có tên trùng với tên cần tìm sẽ in toàn bộ thông tin của Node đó.



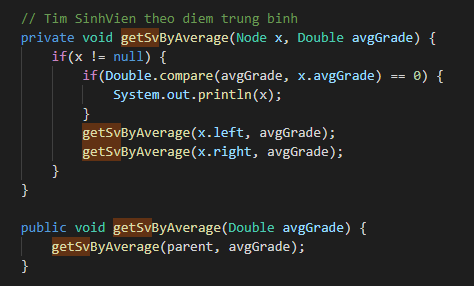
Hình 1.13: Giải thuật tìm kiếm theo tên

1.3.7.2 Giải thuật tìm kiếm theo ngày sinh



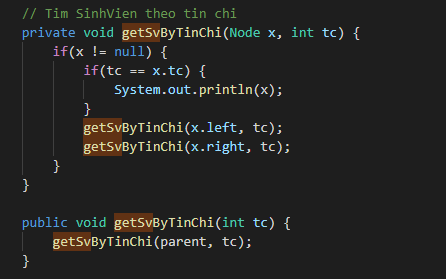
Hình 1.14: Giải thuật tìm kiếm theo ngày sinh

1.3.7.3 Giải thuật tìm kiếm theo điểm trung bình



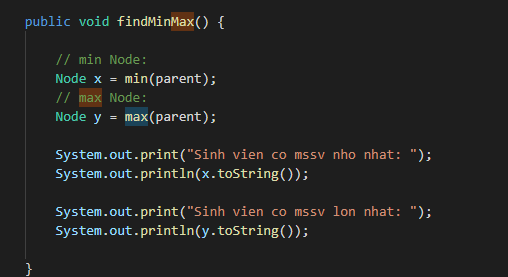
Hình 1.15: Giải thuật tìm kiếm theo điểm trung bình

1.3.7.4 Giải thuật tìm kiếm theo tín chỉ tích lũy



Hình 1.16: Giải thuật tìm kiếm theo tín chỉ

1.3.7.5 Giải thuật tìm kiếm Node lớn nhất, nhỏ nhất

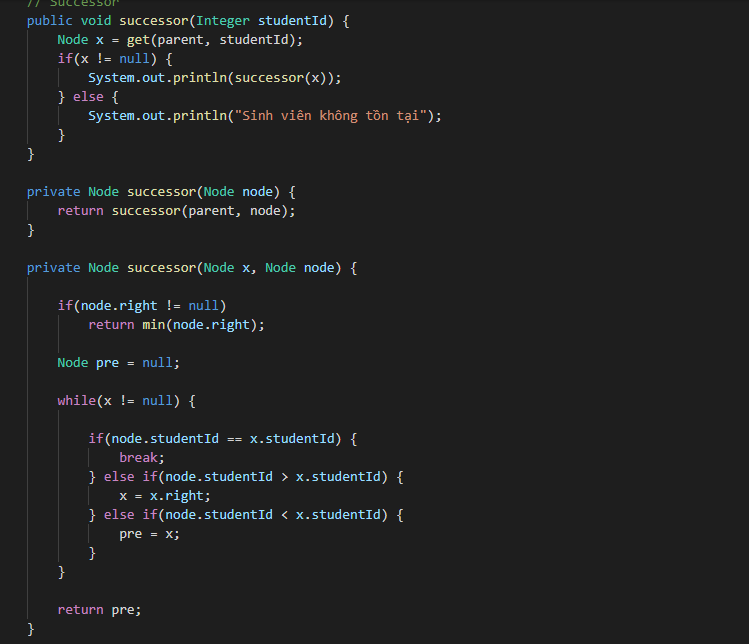


Hình 1.17: Giải thuật tìm kiếm min, max

1.3.8 Giải thuật predecessor và successor:



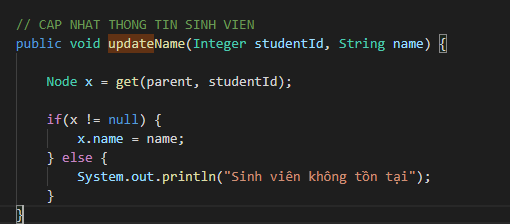
Hình 1.18: Giải thuật tìm kiếm khóa liền trước



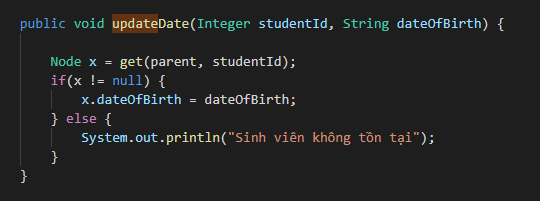
Hình 1.19: Giải thuật tìm kiếm khóa liền sau

1.3.9 Giải thuật cập nhật thông tin cho Node:

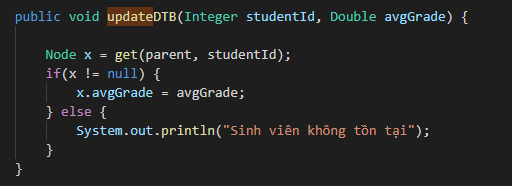
Giải thuật cập nhật thông tin cho từng giá trị riêng biệt của Node theo mã số sinh viên như: tên, ngày sinh, điểm trung bình tích lũy và tín chỉ tích lũy.



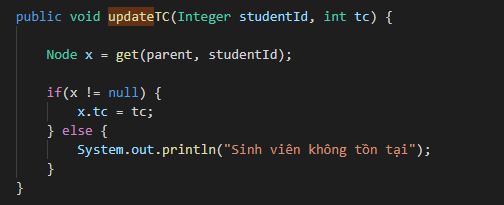
Hình 1.20: Giải thuật cập nhật tên sinh viên



Hình 1.21: Giải thuật cập nhật ngày sinh sinh viên



Hình 1.22: Giải thuật cập nhật điểm trung bình tích lũy sinh viên



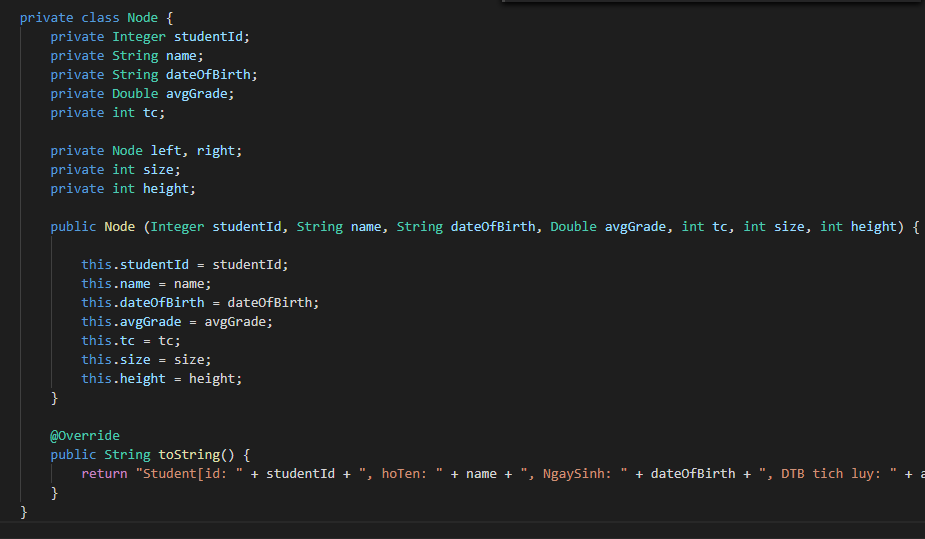
Hình 1.23: Giải thuật cập nhật tín chỉ tích lũy cho sinh viên

CHƯƠNG 2: CÂY NHỊ PHÂN TÌM KIẾM CÂN BẰNG (AVL)

2.1 Khái niệm:

Cấu trúc của cây AVL cũng tương tự như cấu trúc của cây BST. Tuy nhiên, cây BST sẽ có thể bị lệch trái hoặc lệch phải làm cho việc tìm kiếm trở nên tốn thời gian. Giải thuật tìm kiếm trên cây cân bằng sẽ thêm một thuộc tính height trong cấu trúc Node của cây.

2.2 Cấu trúc Node cây cân bằng



Hình 2.24: Cấu trúc Node của cây cân bằng

2.3 Các thuộc tính và giải thuật trong cây AVL:

Cây AVL thêm một thuộc tính height cho cây AVL để tính toán chiều cao của cây. Từ đây ta có thể biết được cây đã cân bằng hay chưa.

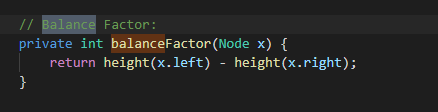
Các giải thuật trong cây AVL cũng tương tự như trong BST nhưng sẽ có thêm các hàm kiểm tra để cân bằng các cây nếu cây bị lệch trái hoặc lệch phải.

2.4 Giải thuật trong cây AVL

Dưới đây là các giải thuật cho cây AVL:

2.4.1 Giải thuật kiểm tra độ cân bằng cây:

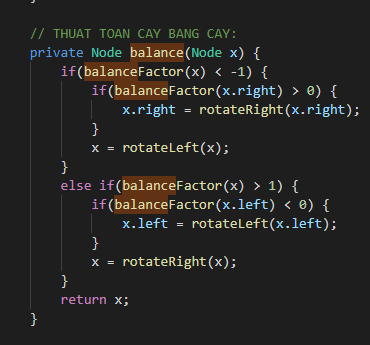
BalanceFactor() là phương thức dùng để kiểm tra xem độ lệch của cây nhị phân. Cây gọi là bị lệch nếu hàm trả về lớn hơn 1.



Hình 2.25: Phương thức kiểm tra độ lệch của cây AVL

2.4.2 Giải thuật balance:

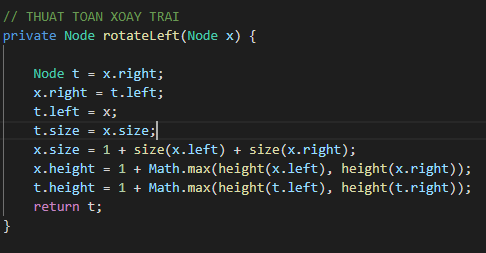
Hàm này được gọi mỗi khi thêm Node vào cây hay xóa Node trong cây. Giải thuật balance sẽ kiểm tra độ lệch của cây (balanceFactor), nếu độ lệch quá 1 thì sẽ bắt đầu thực hiện việc cân bằng.



Hình 2.26 Giải thuật cân bằng trong cây AVL

2.4.3 Giải thuật xoay trái:

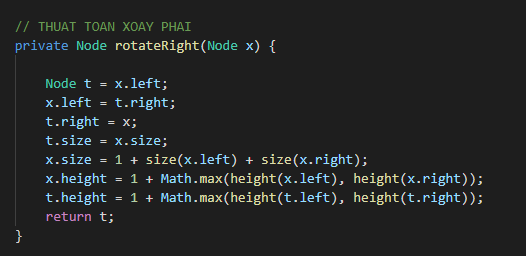
Giải thuật xoay trái sẽ được gọi trong các trường hợp cây bị lệch phải, hoặc lệch trái phải.



Hình 2.27: Giải thuật xoay trái cây AVL

2.4.4 Giải thuật xoay phải

Giải thuật xoay phải sẽ được gọi trong các trường hợp cây bị lệch trái, hoặc lệch phải trái.



Hình 2.28: Giải thuật xoay phải

**TÀI LIỆU THAM KHẢO**

1. Robert Sedgewick & Kevin Wayne (2011), “Algorithms 4th edition”, nhà xuất bản Addison-Wesley Professional

**PHỤ LỤC**

Phần này bao gồm những nội dung cần thiết nhằm minh họa hoặc hỗ trợ cho nội dung luận văn như số liệu, biểu mẫu, tranh ảnh. . . . nếu sử dụng những câu trả lời cho một *bảng câu hỏi thì bảng câu hỏi mẫu này phải được đưa vào phần Phụ lục ở dạng nguyên bản* đã dùng để điều tra, thăm dò ý kiến; **không được tóm tắt hoặc sửa đổi**. Các tính toán mẫu trình bày tóm tắt trong các biểu mẫu cũng cần nêu trong Phụ lục của luận văn. Phụ lục không được dày hơn phần chính của luận văn