**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**Logo, company name

Description automatically generated**

**BÀI TẬP LỚN**

**TÊN MÔN HỌC: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

**TÊN ĐỀ TÀI: QUẢN LÝ DANH SÁCH SINH VIÊN**

**Giảng viên hướng dẫn: ThS. Trần Quý Nam**

**Sinh viên thực hiện: Nguyễn Chí Công**

**Hà Nội, 2025**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**Logo, company name

Description automatically generated**

**BÀI TẬP LỚN**

**TÊN MÔN HỌC: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

**TÊN ĐỀ TÀI: QUẢN LÝ DANH SÁCH SINH VIÊN**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Mã Sinh Viên** | **Họ và Tên** | **Ngày Sinh** | **Điểm** | |
| **Bằng Số** | **Bằng Chữ** |
| **1** | **1871040005** | **Nguyễn Chí Công** | **07/12/2006** |  |  |

**CÁN BỘ CHẤM THI 1 CÁN BỘ CHẤM THI 2**

**Hà Nội, 2024**

MỤC LỤC

[1. GIỚI THIỆU 3](#_Toc193006735)

[2. CẤU TRÚC DỮ LIỆU 4](#_Toc193006736)

[3. CÂY NHỊ PHÂN TÌM KIẾM (BST) 4](#_Toc193006737)

[3.1. Định nghĩa 4](#_Toc193006738)

[3.2 Chức năng chính 5](#_Toc193006739)

[3.3. Tìm kiếm sinh viên 5](#_Toc193006740)

[3.4. Xóa sinh viên 6](#_Toc193006741)

[3.5. Hiển thị danh sách sinh viên 7](#_Toc193006742)

[4. CÂY AVL 8](#_Toc193006743)

[5. KẾT LUẬN 9](#_Toc193006744)

[DANH MỤC TÀI LIỆU THAM KHẢO 10](#_Toc193006745)

# 1. GIỚI THIỆU

Chương trình quản lý danh sách sinh viên sử dụng Cây Nhị Phân Tìm Kiếm (BST). Hệ thống hỗ trợ thêm, xóa, tìm kiếm và hiển thị danh sách sinh viên.

# 2. CẤU TRÚC DỮ LIỆU

Mỗi sinh viên được lưu trữ trong một nút của BST với thông tin:

Mã sinh viên (maSV)

Họ và tên (hoTen)

Điểm (diem)

Con trái (left) và con phải (right)

Khai báo cấu trúc

struct SinhVien {

string maSV;

string hoTen;

float diem;

SinhVien\* left;

SinhVien\* right;

};

# 3. CÂY NHỊ PHÂN TÌM KIẾM (BST)

## 3.1. Định nghĩa

Cây Nhị Phân Tìm Kiếm (BST - Binary Search Tree) là một loại cây nhị phân, trong đó:

Giá trị của nút con trái nhỏ hơn giá trị của nút cha

Giá trị của nút con phải lớn hơn giá trị của nút cha

BST hỗ trợ các thao tác tìm kiếm, chèn, xóa với độ phức tạp trung bình là O(log n) nếu cây cân bằng. Trong trường hợp xấu nhất (cây bị lệch hoàn toàn), độ phức tạp có thể lên đến O(n).

## 3.2 Chức năng chính

Thêm sinh viên

Hàm thêm sinh viên vào BST dựa theo mã sinh viên:

SinhVien\* themSinhVien(SinhVien\* root, string maSV, string hoTen, float diem) {

if (root == NULL) {

return new SinhVien{maSV, hoTen, diem, NULL, NULL};

}

if (maSV < root->maSV)

root->left = themSinhVien(root->left, maSV, hoTen, diem);

else

root->right = themSinhVien(root->right, maSV, hoTen, diem);

return root;

}

## 3.3. Tìm kiếm sinh viên

Tìm sinh viên theo mã sinh viên:

SinhVien\* timKiem(SinhVien\* root, string maSV) {

if (root == NULL || root->maSV == maSV)

return root;

if (maSV < root->maSV)

return timKiem(root->left, maSV);

return timKiem(root->right, maSV);

}

## 3.4. Xóa sinh viên

Xóa sinh viên theo mã số:

SinhVien\* timMin(SinhVien\* root) {

while (root->left != NULL)

root = root->left;

return root;

}

SinhVien\* xoaSinhVien(SinhVien\* root, string maSV) {

if (root == NULL) return root;

if (maSV < root->maSV)

root->left = xoaSinhVien(root->left, maSV);

else if (maSV > root->maSV)

root->right = xoaSinhVien(root->right, maSV);

else {

if (root->left == NULL) {

SinhVien\* temp = root->right;

delete root;

return temp;

} else if (root->right == NULL) {

SinhVien\* temp = root->left;

delete root;

return temp;

}

SinhVien\* temp = timMin(root->right);

root->maSV = temp->maSV;

root->hoTen = temp->hoTen;

root->diem = temp->diem;

root->right = xoaSinhVien(root->right, temp->maSV);

}

return root;

}

## 3.5. Hiển thị danh sách sinh viên

In danh sách sinh viên theo tự nhiên:

void inDanhSach(SinhVien\* root) {

if (root != NULL) {

inDanhSach(root->left);

cout << "MSSV: " << root->maSV << " | Ten: " << root->hoTen << " | Diem: " << root->diem << endl;

inDanhSach(root->right);

}

}

# 4. CÂY AVL

Cây AVL là một dạng mở rộng của **Cây Nhị Phân Tìm Kiếm (BST)**, trong đó mỗi nút có một **hệ số cân bằng (balance factor)** bằng **chiều cao của cây con trái trừ đi chiều cao của cây con phải**. Cây AVL luôn đảm bảo rằng hệ số cân bằng của mọi nút chỉ có thể là -1, 0 hoặc 1.

**Ưu điểm của cây AVL**

Luôn duy trì cây ở trạng thái cân bằng, giúp giảm độ sâu của cây.

Thao tác **tìm kiếm, thêm, xóa** đều có độ phức tạp **O(log n)** ngay cả trong trường hợp xấu nhất.

Phù hợp với các ứng dụng yêu cầu **truy vấn dữ liệu nhanh**.

**Cân bằng cây AVL**

Khi một nút mới được chèn hoặc xóa, nếu cây mất cân bằng, ta cần thực hiện các phép quay để đưa cây về trạng thái cân bằng:

Quay phải (Right Rotation - LL Rotation)

Quay trái (Left Rotation - RR Rotation)

Quay trái-phải (Left-Right Rotation - LR Rotation)

Quay phải-trái (Right-Left Rotation - RL Rotation)

# 5. KẾT LUẬN

Chương trình quản lý sinh viên sử dụng Cây Nhị Phân Tìm Kiếm (Binary Search Tree) giúp tối ưu tìm kiếm, thêm, xóa sinh viên với độ phức tạp trung bình **O(log n)**. Nếu cây mất cân bằng, ta có thể sử dụng **cây AVL** để cải thiện hiệu suất.

# DANH MỤC TÀI LIỆU THAM KHẢO

Giáo trình Cấu trúc dữ liệu và Giải thuật – Nguyễn Văn Linh

Algorithms in C++ – Robert Sedgewick

Data Structures and Algorithm Analysis – Mark Allen Weiss

Trang web GeeksforGeeks (https://www.geeksforgeeks.org/)

Trang web Programiz (<https://www.programiz.com/>)

**Website :**

https://github.com/nguyenchicong-star/Quanlydanhsachsinhvien.git