TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

KHOA CÔNG NGHỆ THÔNG TIN

---------------o0o---------------

**Bài tập lớn môn học**

**CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

Giảng viên hướng dẫn: TS. Hoàng Văn Thông

Sinh viên thực hiện: Lường Văn Sơn

Lớp: CNTT 3 – K63

**Hà Nội tháng 11 năm 2023**

Mục Lục

[I. (BÀI SỐ 09 TRONG DANH SÁCH BÀI TẬP) 3](#_Toc149658743)

[1.1. Đề bài 3](#_Toc149658744)

[1.2. Phân tích bài toán. 3](#_Toc149658745)

[1.3. Cài đặt các lớp và hàm main bằng C++ 6](#_Toc149658746)

[1.4. Phân tích thời gian chạy của từng phương thức có trong các lớp 6](#_Toc149658747)

[II. (BÀI SỐ 33 TRONG DANH SÁCH BÀI TẬP) 16](#_Toc149658748)

[2.1. Đề bài. 16](#_Toc149658749)

[2.2. Phân tích bài toán 17](#_Toc149658750)

[2.3. Cài đặt các lớp và hàm main bằng C++ 17](#_Toc149658751)

[2.4. Phân tích thời gian chạy của từng phương thức có trong các lớp 21](#_Toc149658752)

[III. DANH SÁCH TÀI LIỆU THAM KHẢO 40](#_Toc149658753)

# I. (BÀI SỐ 09 TRONG DANH SÁCH BÀI TẬP)

## **1.1. Đề bài**

Xây dựng một lớp quản lý danh sách sinh viên bằng danh sách liên kết kép, mỗi sinh viên là một cấu trúc gồm:

+ Mã số sinh viên

+ Họ đệm, tên sinh viên

+ Điểm trung bình

+ Tên lớp

Lớp có các phương thức:

a. Tạo danh sách sinh viên: Quá trình nhập danh sách sẽ dừng lại khi nhập mã sv <= 0.

b. Thêm 1 sinh viên vào danh sách, vị trí sinh viên thêm vào do người dùng chọn, nếu không chọn thì thêm vào cuối.

c. Liệt kê danh sách sinh viên trên màn hình theo thứ tự giảm dần của tổng điểm.

d. Căn cứ vào ĐTB của sinh viên, liệt kê tất cả sinh viên xếp loại giỏi (có ĐTB >= 8.0)

e. Nhập vào tên lớp, in ra màn hình danh sách sinh viên của lớp này.

f. Nhập vào mã số sinh viên, sau đó in ra thông tin và vị trí của sinh viên trong danh sách.

g. Sắp xếp danh sách sinh viên theo thứ tự alphabet.

h. Xoá toàn bộ danh sách những sinh viên có ĐTB < 5.0

Xây dựng hàm main để kiểm tra các chức năng của lớp.

## **1.2. Phân tích bài toán.**

* **Yêu cầu của bài toán:**

-Tao danh sách liên kết kép.

-Mỗi Node trong danh sách liên kết chưa thông tin của một sinh viên.

-Viết chương trình để tạo ra các phương thức như yêu cầu của đề bài.

* **Mô tả các phương thức và chức năng của từng phương thức**

Và để tiện cho việc xây dưng và fix lỗi được dễ dàng và thuận tiện hơn cho người code cũng như cho người đọc.Tôi đã tạo ra một folder chứa 3 file.h bao gồm:

-SinhVien.h:trong file này tôi đã khởi tạo một class SinhVien gồm.

* Các thuộc tính.

-long long msv;// ma sinh vien

-string hodem; // ho dem

-string ten; //ten sinh vien

-float DiemTB; // diem trung binh

-string tenlop; //ten lop

* Các phương thức và chức năng của từng phương thức.

-long long getmsv():dùng để lấy mã sinh viên từ trong class SinhVien.

-void setmsv(long long msv):dùng để đặt lại giá trị cho mã sinh viên trong class SinhVien

-string gethodem():dùng để lấy họ đệm từ trong class SinhVien

-void sethodem(string hodem):dùng để đặt lại giá trị cho họ đệm

-string getten():dùng để lấy tên từ trong class SinhVien

-void setten(string ten):dùng để đặt lại giá trị cho tên trong class SinhVien

-float getDiemTB():dùng để lấy điểm trung bình từ trong class sinh viên

-void setDiemTB(float DTB):dùng để đặt lại giá trị của điểm trung bình trong class SinhVien

-string gettenlop():dùng để lấy tên lớp từ trong class SinhVien

-void settenlop(string tenlop):dùng để đặt lại tên lớp từ trong class SinhVien

-istream& operator>>(istream& input, sinhvien &a):dùng để nhập các thuộc tính của class SinhVien.

-ostream& operator<<(ostream& output,sinhvien a):dùng để in các thuộc tính của class SinhVien

-NODE.h: trong file này tôi đã tạo ra một class Node.

* Các thuộc tính.

-sinhvien sv

-Node \*next

-Node \*prev

* Các phương thức và chức năng của từng phương thức.

-Node &operator=(Node \*a):dùng để gán giá trị giữa các Node

-Node():dùng gán giá trị các con trỏ bằng NULL khi khởi tạo

-~Node():dùng gán giá trị các con trỏ bằng NULL khi kết thúc

-sinhvien getsv():dùng để lấy sv từ bên trong class Node

-void setsv(sinhvien a):dùng để đặt lại giá trị của sv

-Node \*getnext():dùng để lấy địa chỉ của con trỏ next

-void setnext(Node \*a):dùng để đặt lại địa chỉ của con trỏ next

-Node \*getprev():dùng để lấy địa chỉ của con trỏ prev

-void setprev(Node \*a):dùng để đặt lại địa chỉ của con trỏ prev

-LinkList.h:trong file này em đã tạo ra một class LinkList

* Các thuộc tính

-Node \*head

* Các phương thước và chức năng của từng phương thức

- int sz():dùng để tính số phần tử bên trong danh sách liên kết

-void insert(sinhvien a, int k):dùng để thêm một Node vào vị trị bất kì trong danh sách liên kết

-void SinhVienXepLoaiGioi():dùng để in các sinh viên có điểm trung bình >=8 ra màn hình

-void FindFromClassName():dùng để tìm kiếm sinh viên theo lớp

-void FindFromId():dùng để tìm kiếm sinh viên theo Id

-void SapXepGiamDan():dùng để sắp xếp các sinh viên theo thứ tự giảm dần của tổng điẻm

- void SapXepTheoAlphabet():dùng để sắp xếp các sinh viên theo tên

- void XoaSinhVienDiemKem():dùng để xóa tất cả các sinh viên có điểm trung binh <=5 ra khỏi danh sách liên kết đôi

Tất cả các file này sẽ được để chung vào folder Source để giúp dễ dàng quản lí hơn đối với code của mình.

Ngoài ra còn có một folder có tên là Main.trong đó có một file main.cpp để tạo ra một menu giúp cho người dùng dễ dàng thao tác hơn với danh sách liên kết kép.

## 1.3. Cài đặt các lớp và hàm main bằng C++

Source on: <https://github.com/LuongSon28112004/DuAnCTDL-GT.git>

## 1.4. Phân tích thời gian chạy của từng phương thức có trong các lớp

-Trong các dự án thì thời gian chạy của chương trình là rất quan trong,và dưới đây là độ phức tạp của các class trong danh sách liên kết kép.

+SinhVien.h:trong file này sẽ chứa một class sinhvien.h.

|  |
| --- |
| long long getmsv()  {  return msv; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| void setmsv(long long msv)  {  this->msv=msv; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| string gethodem()  {  return hodem; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| void sethodem(string hodem)  {  this->hodem=hodem; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| void setten(string ten)  {  this->ten=ten; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| float getDiemTB(){  return DiemTB; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| void setDiemTB(float DTB)  {  this->DiemTB=DTB; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| string gettenlop()  {  return tenlop; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm O(1) |

|  |
| --- |
| void settenlop(string tenlop)  {  this->tenlop=tenlop; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm O(1) |

|  |
| --- |
| istream& operator>>(istream& input, sinhvien &a)  {  cout<<"Nhap ma sinh vien:"; input>>a.msv; độ phức tạp O(1)  input.ignore(); độ phức tạp O(1)  if(a.msv<0)độ phức tạp O(1)  {  return input; độ phức tạp O(1)  }  cout<<"Nhap ho dem:";getline(input,a.hodem); độ phức tạp O(n) với n là độ dài chuỗi  cout<<"Nhap ten:";getline(input,a.ten); độ phức tạp O(n) với n là độ dài chuỗi  cout<<"Nhap Diem Trung Binh:";input>>a.DiemTB; độ phức tạp O(1)  input.ignore(); độ phức tạp O(1)  cout<<"Nhap Ten Lop:";getline(input,a.tenlop); độ phức tạp O(n) với n là độ dài chuỗi  return input; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

|  |
| --- |
| ostream& operator<<(ostream& output,sinhvien a)  {  output<<a.msv<<"\t"<<a.hodem<<"\t"<<a.ten<<"\t"<<a.DiemTB<<"\t"<<a.tenlop<<"\n"; độ phức tạp O(n+n+n+1) với n là độ dài chuỗi  return output; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

+Node.h:file này chứa một class Node là dữ liệu của một Node trong danh sách liên kết đôi.

|  |
| --- |
| Node()  {  next = NULL; độ phức tạp O(1)  prev = NULL; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| ~Node()  {  next = NULL; độ phức tạp O(1)  prev = NULL; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| sinhvien getsv()  {  return sv; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm O(1) |

|  |
| --- |
| void setsv(sinhvien a)  {  sv = a; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| Node \*getnext()  {  return next; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| void setnext(Node \*a)  {  next = a; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| void setprev(Node \*a)  {  prev = a; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

+LinkList.h: trong file này chứa một class LinkList và dữ liệu có nó là một con trỏ chứa địa chỉ Node đầu trong danh sách liên kết kép.

|  |
| --- |
| LinkList()  {  head = NULL; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| ~LinkList()  {  head = NULL; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| Node \*gethead()  {  return head; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| void LinkList::SapXepTheoAlphabet()  {  if (head != NULL) độ phức tạp O(1)  {  for (Node \*p = head; p->next != NULL; p = p->next) độ phức tạp 2\*O(n)  {  Node \*min = p; độ phức tạp O(1)  for (Node \*q = p->next; q != NULL; q = q->next) độ phức tạp O(2\*(n\*(n+1)/2))  {  if (q->getsv().getten() < min->getsv().getten()) độ phức tạp O(n\*(n+1)/2)  {  min = q; độ phức tạp O(n\*(n+1)/2)  }  }  sinhvien tmp = min->getsv(); độ phức tạp O(n)  min->setsv(p->getsv()); độ phức tạp O(n)  p->setsv(tmp);độ phức tạp O(n)  }  }  }  🡺Phương thức trên có độ phức tạp tiệm cận O(n^2) |

|  |
| --- |
| void LinkList::SapXepGiamDan()  {  if (head != NULL) độ phức tạp O(1)  {  for (Node \*p = head; p->next != NULL; p = p->next) độ phức tạp 2\*O(n)  {  Node \*max = p; độ phức tạp O(n)  for (Node \*q = p->next; q != NULL; q = q->next) độ phức tạp O(2\*(n\*(n+1)/2))  {  if (q->getsv().getDiemTB() > max->getsv().getDiemTB())  {  max = q; độ phức tạp O(n\*(n+1)/2))  }  }  sinhvien tmp = max->getsv(); độ phức tạp O(n)  max->setsv(p->getsv()); độ phức tạp O(n)  p->setsv(tmp);độ phức tạp O(n)  }  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n^2) |

|  |
| --- |
| void LinkList::XoaSinhVienDiemKem()  {  if (head == NULL) độ phức tạp O(1)  {  cout << " Danh Sach Lien Ket Rong!\n"; độ phức tạp O(1)  return;độ phức tạp O(1)  }  else  {  Node \*p = head; độ phức tạp O(1)  while (p != NULL) độ phức tạp O(n)  {  if (p->sv.getDiemTB() <= 5) độ phức tạp O(n)  {  Node \*temp = p; độ phức tạp O(n)  if (p->prev == NULL) độ phức tạp O(n)  {  head = p->next; độ phức tạp O(n)  if(head!=NULL) độ phức tạp O(n)  {  head->prev=NULL; độ phức tạp O(n)  }  p = p->next; độ phức tạp O(n)  }  else if (p->next == NULL) độ phức tạp O(n)  {  p->prev->next = NULL; độ phức tạp O(n)  p=NULL; độ phức tạp O(n)  }  else // neu la node giua  {  p->prev->next = p->next; độ phức tạp O(n)  p->next->prev = p->prev; độ phức tạp O(n)  p = p->next; độ phức tạp O(n)  }  delete temp; độ phức tạp O(n)  }  else // neu khong roi vao diem trung binh <=5  {  p = p->next; độ phức tạp O(n)  }  }  delete p; độ phức tạp O(n)  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

|  |
| --- |
| void LinkList::FindFromId()  {  if (head == NULL) độ phức tạp O(1)  {  cout << "Danh Sach Lien Ket Rong!\n"; độ phức tạp O(1)  return; độ phức tạp O(1)  }  else  {  long long id; độ phức tạp O(1)  int cnt = 0; độ phức tạp O(1)  cout << "Nhap Ten Id Can tim:"; độ phức tạp O(1)  cin >> id; độ phức tạp O(1)  cin.ignore();độ phức tạp O(1)  Node \*p = head; độ phức tạp O(1)  bool flag = false; độ phức tạp O(1)  while (p != NULL) ) độ phức tạp o(n)  {  cnt++;độ phức tạp O(2\*n)  if (p->sv.getmsv() == id) độ phức tạp O(n)  {  flag = true; độ phức tạp O(n)  cout << p->sv; độ phức tạp O(n)  cout << "Vi Tri:" << cnt << "\n";độ phức tạp O(n)  }  p = p->next; độ phức tạp O(n)  }  if (!flag) độ phức tạp O(1)  {  cout << "Khong tim thay sinh vien co id:" << id << "\n";độ phức tạp O(1)  }  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

|  |
| --- |
| void LinkList::FindFromClassName()  {  if (head == NULL) độ phức tạp O(1)  {  cout << "Danh Sach Lien Ket Rong!\n";độ phức tạp O(1)  return; độ phức tạp O(1)  }  else  {  string Class\_Name; độ phức tạp O(1)  cout << "Nhap Ten Class Can tim:"; độ phức tạp O(1)  getline(cin, Class\_Name); độ phức tạp O(n) với n là độ dài chuỗi  Node \*p = head; độ phức tạp O(1)  bool flag = false; độ phức tạp O(1)  while (p != NULL) độ phức tạp O(n)  {  if (p->sv.gettenlop() == Class\_Name) độ phức tạp O(n)  {  flag = true; độ phức tạp O(n)  cout << p->sv;  }  p = p->next; độ phức tạp O(n)  }  if (!flag) độ phức tạp O(1)  {  cout << "Khong tim thay sinh vien co lop:" << Class\_Name << "\n"; độ phức tạp O(1)  }  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

|  |
| --- |
| void LinkList::SinhVienXepLoaiGioi()  {  if (head == NULL) độ phức tạp O(1)  {  cout << "Danh Sach Lien Ket Rong!\n"; độ phức tạp O(1)  return; độ phức tạp O(1)  }  else  {  cout << "Danh Sach Sinh Vien Co Diem Xep Loai Gioi La:\n"; độ phức tạp O(1)  Node \*p = head; độ phức tạp O(1)  bool flag = false; độ phức tạp O(1)  while (p != NULL) độ phức tạp O(n)  {  if (p->sv.getDiemTB() >= 8) độ phức tạp O(n)  {  flag = true; độ phức tạp O(n)  cout << p->sv;  }  p = p->next; độ phức tạp O(n)  }  if (!flag) độ phức tạp O(1)  {  cout << "Khong co sinh vien nao co diem trung binh >=8\n";  }  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

|  |
| --- |
| void LinkList::insert(sinhvien a, int k)  {  Node \*tmp = new Node; độ phức tạp O(1)  tmp->sv = a; độ phức tạp O(1)  tmp->next = NULL; độ phức tạp O(1)  tmp->prev = NULL; độ phức tạp O(1)  if (k == 1) độ phức tạp O(1)  {  if (head == NULL) độ phức tạp O(1)  {  head = tmp; độ phức tạp O(1)  }  else  {  tmp->next = head; độ phức tạp O(1)  head->prev = tmp; độ phức tạp O(1)  head = tmp; độ phức tạp O(1)  }  }  else if (k >= sz() + 1 || k == 0) độ phức tạp O(2)  {  if (head == NULL) độ phức tạp O(1)  {  head = tmp; độ phức tạp O(1)  }  else  {  Node \*p = head; độ phức tạp O(1)  while (p->next != NULL) độ phức tạp O(n)  {  p = p->next;độ phức tạp O(n)  }  p->next = tmp; độ phức tạp O(1)  tmp->prev = p; độ phức tạp O(1)  }  }  else if (k > 1 || k <= sz())độ phức tạp O(1)  {  if (head == NULL) độ phức tạp O(1)  {  head = tmp; độ phức tạp O(1)  }  else  {  Node \*p = head; độ phức tạp O(1)  for (int i = 0; i < k - 1; i++) độ phức tạp O(k)  {  p = p->next; độ phức tạp O(k-1)  }  tmp->next = p; độ phức tạp O(1)  p->prev->next = tmp; độ phức tạp O(1)  tmp->prev = p->prev; độ phức tạp O(1)  p->prev = tmp; độ phức tạp O(1)  }  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

|  |
| --- |
| int LinkList::sz()  {  int cnt = 0; độ phức tạp O(1)  Node \*p = head; độ phức tạp O(1)  while (p != NULL) độ phức tạp O(n)  {  p = p->next; độ phức tạp O(n)  cnt++; độ phức tạp O(2\*n)  }  return cnt; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

# II. (BÀI SỐ 33 TRONG DANH SÁCH BÀI TẬP)

## 2.1. Đề bài.

- Xây dựng lớp cây tìm kiếm nhị phân.

- Viết chương trình quản lý lịch công tác trong tháng đơn giản: cho phép nhập vào nội dung công việc cần làm theo ngày, theo giờ. Trong một ngày có thể có nhiều công việc, mỗi công việc có giờ bắt đầu, tên công việc, nội dung công việc, tính chất công việc {rất quan trọng, quan trọng, bình thường, không cần thiết}, …

-Chương trình có các chức năng chính như sau:

+ Nhập nội dung công việc cần làm theo ngày, theo giờ.

+ Xem lịch công tác theo ngày yêu cầu.

+ Xem các công việc theo tính chất: rất quan trọng, quan trọng, …

+ Xem các công việc đã hoàn tất.

+ Xem các công việc chưa thực hiện.

+ Xem các công việc từ ngày a đến ngày b.

+ Xóa hay điều chỉnh lịch công tác. Nếu sau khi điều chỉnh, ngày nào không còn việc phải làm sẽ xóa khỏi lịch công tác.

-Yêu cầu: Cấu trúc dữ liệu được sử dụng là cây nhị phân tìm kiếm (BST), trong đó:

+ Mỗi nút trên cây BST là một ngày của lịch công tác.

+ Trong mỗi nút ngày trên cây lại chứa một danh sách liên kết lưu thông tin các công việc.

+ Khi thêm một công việc vào một ngày đã tồn tại trên cây thì công việc này sẽ được đưa vào danh sách liên kết chứa các công việc theo thứ tự tăng dần của giờ bắt đầu.

-Xây dựng ham main để kiểm tra các chức năng của lớp

## 2.2. Phân tích bài toán.

* **Yêu cầu của bài toán**

**-**Xây dựng lớp cây tìm kiếm nhị phân

-Mỗi Node là một ngày của lịch công tác

-Mỗi Node trên cây lại chứa một danh sách liên kết lưu thông tin các công việc trong ngày

-Viết chương trình để kiểm tra sự hoạt động của cây và làm các phương thức như yêu cầu của đề bài

* **Mô tả các phương thức và chức năng của từng phương thức**

Và để tiện cho việc xây dưng và fix lỗi được dễ dàng và thuận tiện hơn cho người code cũng như cho người đọc.em đã tạo ra một folder có tên Source chứa 5 file.h bao gồm:

-CongViec.h: trong file này em se tạo một class CongViec.

* Các thuộc tính.

-string time\_start

-string ten\_cong\_viec

-string noi\_dung

-string tinh\_chat\_cong\_viec

-bool check\_Hoan\_Thanh

* Các phương thức

-bool getcheck\_Hoan\_thanh():giúp lấy ra được kết quả true hoặc false để kiểm tra xem công việc đã hoàn thành hay chưa

-void setcheck\_Hoan\_Thanh(bool check):dùng để đặt lại giá trị của check\_Hoan\_Thanh bên trong class

-string gettime\_start():dùng để lấy ra thời gian bắt đầu công việc

-void settime\_start(string time):dùng để đặt lại thời gian bắt đầu công việc

-string getten\_cong\_viec():dùng để lấy ra tên công việc

-void setten\_cong\_viec(string cong\_viec):dùng để đặt lại tên công việc

-string gettinh\_chat\_cong\_viec():dùng để lấy ra tính chất công việc

-void settinh\_chat\_cong\_viec(string tinh\_chat):dùng để đặt lại tính chất cônh việc

-string getnoi\_dung():dùng để lấy ra nội dụng

-void setnoi\_dung(string noidung):dùng để đặt lại nội dung

-friend istream& operator>>(istream&input,CongViec &a):dùng để nhập các công việc

-friend ostream& operator<<(ostream&output,const CongViec &a):dùng để xuất thông tin công việc

-LinkList.h: trong file này chứa một cấu trúc struct là một danh sách liên kết đơn.

* Các thuộc tính.

-CongViec data

-Node \*next

* Các phương thức.

-node makeNode(CongViec x):tạo một node mới có dữ liệu là một công việc x

-bool empty(node a):check xem danh sách có rỗng hay không

-int Size(node a):dùng để kiểm tra số phần tử của danh sách liên kết

-void TickCongViecHoanThanh(node a,string giobatdau,string tencv):dùng để dánh dấu các công việc đã hoàn thành

-int findPos(node a):dùng để tìm vị trí công việc đã được hoàn thành trong danh sách liên kết

-void insertFirst(node &a, CongViec x):thêm một phần tử vào đầu danh sách liên kết

-void insertLast(node &a, CongViec x):thêm một phần tử vào cuối danh sách liên kết

-void insertMiddle(node &a, CongViec x, int pos):thêm một phần từ vào giữa danh sách liên kết

-void deleteDau(node &a):xóa phần tử ở đầu danh sách liên kết

-void deleteCuoi(node &a):xóa phần tử ở cuối danh sách liên kết

-void deleteGiua(node &a, int pos):xóa phần tử ở giữa danh sách liên kết

-void in(node a):in danh sách liên kết

-void in\_Theo\_Tinh\_Chat\_Cong\_Viec(node a,string tinh\_chat):in danh sách theo tính chất công việc

-void in\_Theo\_Cong\_Viec\_Da\_Hoan\_Thanh(node a):in các công việc đã hoàn thành trong danh sách liên kết

-void in\_Theo\_Cong\_Viec\_Chua\_Hoan\_Thanh(node a):in các công việc chưa hoàn thành trong danh sách liên kết

-void sapxep(node &a):dùng để sắp xếp danh sách liên kết theo thứ tự tăng dần của giờ bắt đầu.

-CongViecTrongNgay.h: trong file này sẽ chứa một class CongViecTrongNgay giúp quản lí được các công việc trong ngày

* Các thuộc tính.

-int ten\_ngay

-node head

* Các phương thức và chức năng của từng phương thức.

-CongViecTrongNgay():khởi tạo các giá trị đầu cho các thuộc tính

-int getten\_ngay():dùng để lấy tên ngày

-void setten\_ngay(int ngay):dùng để đặt lại tên ngay

-node gethead():dùng để lấy địa chỉ của con trỏ head

-void sethead(node a):dùng để đặt lại địa chỉ của con trỏ head

-CongViecTrongNgay &operator=(const CongViecTrongNgay &a):dùng để gán giá trị cho nhau giữa các công việc trong ngay

-void insert():dùng để nhập một danh sách các công việc và thêm vào con trỏ head

-void inDS():dùng để in danh sách tất cả các công việc

-void inDS\_Theo\_Tinh\_Chat\_Cong\_Viec(string tinhchat):dùng để in danh sách theo tính chất công việc

-void inDS\_Theo\_Cong\_Viec\_Da\_Hoan\_Thanh():dùng để in các công việc đã hoàn thành

-void inDS\_Theo\_Cong\_Viec\_Chua\_Hoan\_Thanh():dùng để in các công việc chưa hoàn thành

-NodeBST.h: trong file này em sẽ tạo một class NodeBST.tương ứng là một Node trong cây tìm kiếm nhị phân và giúp quản lí một node trong cây nhị phân.

* Các thuộc tính.

-CongViecTrongNgay cv

-NodeBST\* left

-NodeBST\* right

* Các phương thức và chức năng của từng phương thức.

-NodeBST():dùng để khởi tạo các thuộc tính.

~NodeBST():dùng để hủy các giá trị của các thuộc tính.

CongViecTrongNgay getcv():dùng để lấy giá trị của công việc trong ngày.

-void setcv(const CongViecTrongNgay &a):dùng để đặt lại giá trị của công việc trong ngày

-friend NodeBST \* makenode(CongViecTrongNgay a):dùng để tạo ra một node mới.

-CayBST.h:file này chứa class CayBST.class này chứa Node root của cây và trong class này sẽ sử lí các yêu cầu của đề bài.

* Các thuộc tính.

-NodeBST \*headBST

-int cnt

* Các phương thức và chức năng của từng phương thức.

-CayBST():dùng để đặt giá trị cho các thuộc tính trong class CayBST.

-~CayBST():dùng để hủy các giá trị cho các thuộc tính trong class CayBST.

-NodeBST \*getheadBST():dùng để lấy địa chỉ của headBST

-void setheadBST(NodeBST \*head):dùng để đặt lại địa chỉ của headBST.

-int getcnt():dùng để lấy số phần tử của cây.

-void setcnt(int count):dùng để đặt lại số phần tử của cây.

-bool is\_empty():dùng để kiểm tra xem cây có rỗng hay. không.

-void insertNodeBST(NodeBST \*&a, CongViecTrongNgay x):dùng để thêm một Node vào trong cây.

-void XemLichCongTacTheoNgayYeuCau(NodeBST \*a, int ngay):dùng để xem lịch công tác theo ngày yêu cầu.

-void XemLichCongTacTheoTinhChatYeuCau(NodeBST \*a, string tinh\_chat):dùng để xem lịch công tác theo tính chất công việc

-void DanhDauCacCongViecDaHoanThanh(NodeBST \*a, int ngay, string giobatdau, string tencv):dùng để đánh dấu các công việc đã hoàn thành

-void XemCacCongViecDaHoanhThanh(NodeBST \*a):dùng để xem các công việc đã hoàn thanh.

-void XemCacCongViecChuaHoanhThanh(NodeBST \*a):dùng để xem các công việc chưa hoàn thành.

-void XemCacCongViecTuNgayaDenNgayb(NodeBST \*a, int x, int y):dùng để xem các công việc từ ngày a đến ngày b

-void DeleteCacCongViecDaHoanThanh(NodeBST \*a):dùng để xóa các công việc đã hoàn thành

-void DieuChinhLichCongTac(NodeBST \*a, NodeBST \*&b):dùng để xóa các ngày mà không còn công việc

-void xuat(NodeBST \*a):dùng để in tất cả các ngày và các công việc trong cây.

Ngoài ra em còn tạo ra một folder có tên Main.trong đó chứa một file main.cpp để tạo một menu cho người dùng thao tác được dễ dàng hơn.

2.3. Cài đặt các lớp và hàm main bằng C++

Source on: <https://github.com/LuongSon28112004/DuAnCTDL-GT.git>

## 2.4. Phân tích thời gian chạy của từng phương thức có trong các lớp

-Trong các dự án thì thời gian chạy của chương trình là rất quan trong,và dưới đây là độ phức tạp của các class trong danh sách liên kết đôi. vì khi nhập hoặc in một chuỗi thì độ phức tạp sẽ là độ dài của chuỗi.vì vậy ta sẽ gọi độ dài của chuỗi lớn nhất là n.

+CongViec.h:trong file này sẽ chứa một class CongViec.

|  |
| --- |
| CongViec(string time\_start="",string ten\_cong\_viec="",string noi\_dung="",string tinh\_chat\_cong\_viec="")  {  this->time\_start=time\_start;độ phức tạp O(1)  this->ten\_cong\_viec=ten\_cong\_viec; độ phức tạp O(1)  this->noi\_dung=noi\_dung; độ phức tạp O(1)  this->tinh\_chat\_cong\_viec=tinh\_chat\_cong\_viec; độ phức tạp O(1)  check\_Hoan\_Thanh=false; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| bool getcheck\_Hoan\_thanh() const{  return check\_Hoan\_Thanh ; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| void setcheck\_Hoan\_Thanh(bool check)  {  check\_Hoan\_Thanh=check; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| string gettime\_start() const{  return time\_start; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| void settime\_start(string time)  {  time\_start=time; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| string getten\_cong\_viec() const{  return ten\_cong\_viec; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| void setten\_cong\_viec(string cong\_viec)  {  ten\_cong\_viec=cong\_viec; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| string gettinh\_chat\_cong\_viec() const{  return tinh\_chat\_cong\_viec; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| void settinh\_chat\_cong\_viec(string tinh\_chat)  {  tinh\_chat\_cong\_viec=tinh\_chat; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| string getnoi\_dung() const{  return noi\_dung; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| void setnoi\_dung(string noidung)  {  noi\_dung=noidung; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| istream& operator>>(istream&input,CongViec &a)  {  cout<<"Nhap Gio Bat Dau:";getline(input,a.time\_start); độ phức tạp O(n)  cout<<"Nhap Ten Cong Viec:";getline(input,a.ten\_cong\_viec); độ phức tạp O(n)  cout<<"Nhap Noi Dung:";getline(input,a.noi\_dung); độ phức tạp O(n)  cout<<"Nhap Tinh Chat Cong Viec:";getline(input,a.tinh\_chat\_cong\_viec); độ phức tạp O(n)  return input;  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) với n là chiều dài của chuỗi. |

|  |
| --- |
| ostream& operator<<(ostream&output,const CongViec &a)  {  output<<a.time\_start<<" "<<a.ten\_cong\_viec<<" "<<a.noi\_dung<<" "<<a.tinh\_chat\_cong\_viec<<"\n";độ phức tạp O(n+n+n+n)  return output; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

+LinkList.h:trong file này là một struct của danh sách liên kết và trong file có chứa các hàm để thao tác với danh sách liên kết.

|  |
| --- |
| node makeNode(CongViec x){  node tmp = new Node();độ phức tạp O(1)  tmp->data = x; độ phức tạp O(1)  tmp->next = NULL; độ phức tạp O(1)  return tmp; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| bool empty(node a){  return a == NULL; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| int Size(node a){  int cnt = 0; độ phức tạp O(1)  while(a != NULL){ độ phức tạp O(n)  ++cnt; độ phức tạp O(2\*n)  a = a->next; độ phức tạp O(n)  }  return cnt; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

|  |
| --- |
| void TickCongViecHoanThanh(node a,string giobatdau,string tencv)  {  while(a!=NULL) độ phức tạp O(n)  {  if(a->data.gettime\_start()==giobatdau && a->data.getten\_cong\_viec()==tencv) độ phức tạp O(2\*n)  {  a->data.setcheck\_Hoan\_Thanh(true); độ phức tạp O(n)  }  a=a->next; độ phức tạp O(n)  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

|  |
| --- |
| int findPos(node a)  {  int cnt=1; độ phức tạp O(1)  while(a!=NULL) độ phức tạp O(n)  {  if(a->data.getcheck\_Hoan\_thanh())độ phức tạp O(n)  {  return cnt; độ phức tạp O(1)  }  cnt++;độ phức tạp O(2\*n)  a=a->next; độ phức tạp O(n)  }  return 0; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

|  |
| --- |
| void insertFirst(node &a, CongViec x){  node tmp = makeNode(x); độ phức tạp O(1)  if(a == NULL){ độ phức tạp O(1)  a = tmp; độ phức tạp O(1)  }  else{  tmp->next = a; độ phức tạp O(1)  a = tmp; độ phức tạp O(1)  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| void insertLast(node &a, CongViec x){  node tmp = makeNode(x); độ phức tạp O(1)  if(a == NULL){ độ phức tạp O(1)  a = tmp; độ phức tạp O(1)  }  else{  node p = a; độ phức tạp O(1)  while(p->next != NULL){ độ phức tạp O(n)  p = p->next; độ phức tạp O(n)  }  p->next = tmp; độ phức tạp O(1)  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

|  |
| --- |
| void insertMiddle(node &a, CongViec x, int pos){  int n = Size(a); độ phức tạp O(n) ví phương thức Size(a) có độ phức tạp n.  if(pos <= 0 || pos > n + 1) độ phức tạp O(2)  {  cout << "Vi tri chen khong hop le !\n";  }  if(pos == 1){  insertFirst(a, x); return;độ phức tạp O(1)  }  else if(pos == n +1 ){  insertLast(a, x); return;độ phức tạp O(1)  }  node p = a;  for(int i = 1; i < pos - 1; i++){độ phức tạp O(2\*pos)  p = p->next;độ phức tạo O(pos-1)  }  Các phương thức gán có độ phức tạp độ phức tạp O(1)  node tmp = makeNode(x);  tmp->next = p->next;  p->next = tmp;  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

|  |
| --- |
| void deleteDau(node &a){  if(a == NULL){  return; độ phức tạp O(1)  }  a = a->next; độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| void deleteCuoi(node &a){  if(a == NULL) return;  các phương thức gán có độ phức tạp O(1)  node truoc = NULL;độ phức tạp O(1)  sau = a;độ phức tạp O(1)  while(sau->next != NULL){ độ phức tạp O(n)  truoc = sau;độ phức tạp O(n)  sau = sau->next;độ phức tạp O(n)  }  if(truoc == NULL){độ phức tạp O(1)  a = NULL; độ phức tạp O(1)  }  else{  truoc->next = NULL; độ phức tạp O(1)  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

|  |
| --- |
| void deleteGiua(node &a, int pos){  if(pos <=0 || pos > Size(a)) return;độ phức tạp O(1)  node truoc = NULL;độ phức tạp O(1)  sau = a;độ phức tạp O(1)  for(int i = 1; i < pos; i++){độ phức tạp O(2\*(pos+1))  truoc = sau;độ phức tạp O(pos)  sau = sau->next;độ phức tạp O(pos)  }  if(truoc == NULL){độ phức tạp O(1)  a = a->next; độ phức tạp O(1)  }  else{  truoc->next = sau->next; độ phức tạp O(1)  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

|  |
| --- |
| void in(node a){  while(a != NULL){ độ phức tạp O(n) với n là số phần tử của danh sách liên kết  cout << a->data;độ phức tạp O(1)  a = a->next;độ phức tạp O(1)  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

|  |
| --- |
| void in\_Theo\_Tinh\_Chat\_Cong\_Viec(node a,string tinh\_chat)  {  while(a != NULL) độ phức tạp O(n) với n là số phần tử của danh sách liên kết.  {  if(a->data.gettinh\_chat\_cong\_viec()==tinh\_chat)  {  cout<<a->data;độ phức tạp O(n)  }  a = a->next;độ phức tạp O(n)  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

|  |
| --- |
| void in\_Theo\_Cong\_Viec\_Da\_Hoan\_Thanh(node a)  {  Các phép gán có độ phức tạp O(1)  while(a!=NULL) độ phức tạp O(n) với n là số phần tử của danh sách liên kết.  {  if(a->data.getcheck\_Hoan\_thanh())độ phức tạp O(n)  {  cout<<a->data;độ phức tạp O(n)  }  a = a->next;độ phức tạp O(n)  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

|  |
| --- |
| void in\_Theo\_Cong\_Viec\_Chua\_Hoan\_Thanh(node a)  {  while(a!=NULL) độ phức tạp O(n)  {  if(!(a->data.getcheck\_Hoan\_thanh()))độ phức tạp O(n)  {  cout<<a->data;độ phức tạp O(n)  }  a = a->next;độ phức tạp O(n)  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

|  |
| --- |
| void sapxep(node &a){  for(node p = a; p->next != NULL; p = p->next){ độ phức tạp O(2\*n)  node tmp = p; độ phức tạp O(n)  for(node q = p->next; q != NULL; q = q->next){ độ phức tạp O(2\*n\*(n+1)/2)  if(stoi(q->data.gettime\_start()) < stoi(tmp->data.gettime\_start())){  tmp = q; độ phức tạp O(n\*(n+1)/2)  }  }  CongViec swap = tmp->data; độ phức tạp O(n)  tmp->data = p->data; độ phức tạp O(n)  p->data = swap; độ phức tạp O(n)  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n^2) |

+CongViecTrongNgay.h:file này chứa một class công việc trọng ngày giúp người dùng quản lí danh sách các công việc trong một ngày bất kì.

|  |
| --- |
| CongViecTrongNgay()  {  Các biểu thức gán này có độ phức tạp O(1)  ten\_ngay = 0;  head = NULL;  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| int getten\_ngay() const  {  return ten\_ngay;độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| void setten\_ngay(int ngay)  {  ten\_ngay = ngay;độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| node gethead()  {  return head;độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| void sethead(node a)  {  head = a;độ phức tạp O(1)  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| CongViecTrongNgay &operator=(const CongViecTrongNgay &a)  {  if (this == &a)độ phức tạp O(1)  {  return \*this; độ phức tạp O(1)  }  Các phương thức gán này đều có độ phức tạp là O(1)  this->ten\_ngay = a.ten\_ngay;  this->head = a.head;  return \*this;  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| void insert()  {  cout << "Nhap Ten Ngay:";độ phức tạp O(1)  cin >> ten\_ngay;độ phức tạp O(1)  cin.ignore();độ phức tạp O(1)  while (true)  {  CongViec a;  cout << "Nhap Thong Tin Cong Viec:\n";độ phức tạp O(n)  cin >> a;độ phức tạp O(n)  insertLast(head, a);độ phức tạp O(n^2)  cout << "Ban Muon Nhap Tiep Cong Viec khong Dong y(1) or Thoat(0)\n";  cout << "Nhap lua chon:\n";  sapxep(head);  int lc;  cin >> lc;  cin.ignore();  if (lc!=1)  {  break;  }  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n^2) |

|  |
| --- |
| void inDS()  {  cout << "cong viec trong ngay " << ten\_ngay << ":\n";  in(head);độ phức tạp O(n)  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

|  |
| --- |
| void inDS\_Theo\_Tinh\_Chat\_Cong\_Viec(string tinhchat)  {  cout << "cong viec trong ngay " << ten\_ngay << ":\n";  in\_Theo\_Tinh\_Chat\_Cong\_Viec(head, tinhchat);  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

|  |
| --- |
| void inDS\_Theo\_Cong\_Viec\_Da\_Hoan\_Thanh()  {  cout << "cong viec trong ngay " << ten\_ngay << ":\n";  in\_Theo\_Cong\_Viec\_Da\_Hoan\_Thanh(head);  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

|  |
| --- |
| void inDS\_Theo\_Cong\_Viec\_Chua\_Hoan\_Thanh()  {  cout << "cong viec trong ngay " << ten\_ngay << ":\n";  in\_Theo\_Cong\_Viec\_Chua\_Hoan\_Thanh(head);  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

+NodeBST.h:file này chứa một class NodeBST là dữ liệu của một Node trong cây tìm kiếm nhị phân.

|  |
| --- |
| NodeBST()  {  left=NULL;  right=NULL;  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| ~NodeBST()  {  left=NULL;  right=NULL;  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| CongViecTrongNgay getcv(){  return cv;  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| void setcv(const CongViecTrongNgay &a)  {  cv=a;  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| NodeBST \*makenode(CongViecTrongNgay a)  {  NodeBST \*tmp = new NodeBST;  tmp->cv = a;  tmp->left = NULL;  tmp->right = NULL;  return tmp;  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

+CayBST.h:file này lưu trữ địa chỉ Node root của cây và các thao tác với cây mà đề bài yêu cầu.

|  |
| --- |
| CayBST()  {  Các phép gán có độ phức tạp O(1)  headBST = NULL;  cnt = 0;  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| ~CayBST()  {  Các phép gán có độ phức tạp O(1)  headBST = NULL;  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| NodeBST \*getheadBST()  {  Các phép gán có độ phức tạp O(1)  return headBST;  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| void setheadBST(NodeBST \*head)  {  headBST = head;  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| int getcnt() const  {  return cnt;  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| void setcnt(int count)  {  cnt=count;  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| bool is\_empty()  {  if (headBST == NULL)  {  return true;  }  return false;  }  🡺phương thức trên có độ phức tạp tiệm cận O(1) |

|  |
| --- |
| void insertNodeBST(NodeBST \*&a, CongViecTrongNgay x)  {  NodeBST \*tmp = new NodeBST;độ phức tạp O(1)  tmp = makenode(x);độ phức tạp O(1)  if (a == NULL)độ phức tạp O(1)  {  cnt++;độ phức tạp O(2)  a = tmp;độ phức tạp O(1)  if (is\_empty())độ phức tạp O(1)  {  headBST = a;độ phức tạp O(1)  }  return;độ phức tạp O(1)  }  else if (tmp->cv.getten\_ngay() < a->cv.getten\_ngay())  {  insertNodeBST(a->left, x);  }  else if (tmp->cv.getten\_ngay() > a->cv.getten\_ngay())  {  insertNodeBST(a->right, x);  }  else if (tmp->cv.getten\_ngay() == a->cv.getten\_ngay())  {  if (a->cv.gethead() == NULL)  {  node p = tmp->cv.gethead();độ phức tạp O(1)  a->cv.sethead(p);độ phức tạp O(1)  }  else  {  int n = Size(tmp->cv.gethead());độ phức tạp O(n)  node node1\_ = a->cv.gethead();độ phức tạp O(n)  node node2\_ = tmp->cv.gethead();độ phức tạp O(n)  for (int i = 0; i < n; i++)độ phức tạp O(2\*(n+1))  {  CongViec p = node2\_->data;độ phức tạp O(n)  insertLast(node1\_, p);độ phức tạp O(n^2)  node2\_ = node2\_->next;độ phức tạp O(n)  }  sapxep(node1\_);độ phức tạp O(n^2)  }  return;độ phức tạp O(1)  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n^2) |

|  |
| --- |
| void XemLichCongTacTheoNgayYeuCau(NodeBST \*a, int ngay)  {  if (a != NULL)độ phức tạp O(1)  {  if (a->cv.getten\_ngay() == ngay)độ phức tạp O(1)  {  a->cv.inDS();độ phức tạp O(n)  return;độ phức tạp O(1)  }  else if (a->cv.getten\_ngay() < ngay)độ phức tạp O(1)  {  XemLichCongTacTheoNgayYeuCau(a->right, ngay);  }  else if (a->cv.getten\_ngay() > ngay)độ phức tạp O(1)  {  XemLichCongTacTheoNgayYeuCau(a->left, ngay);  }  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

|  |
| --- |
| void XemLichCongTacTheoTinhChatYeuCau(NodeBST \*a, string tinh\_chat)  {  if (a != NULL)  {  a->cv.inDS\_Theo\_Tinh\_Chat\_Cong\_Viec(tinh\_chat);  XemLichCongTacTheoTinhChatYeuCau(a->right, tinh\_chat);  XemLichCongTacTheoTinhChatYeuCau(a->left, tinh\_chat);  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n^2) |

|  |
| --- |
| void DanhDauCacCongViecDaHoanThanh(NodeBST \*a, int ngay, string giobatdau, string tencv)  {  if (a != NULL)  {  if (a->cv.getten\_ngay() == ngay)  {  node p = a->cv.gethead();  TickCongViecHoanThanh(p, giobatdau, tencv);  Return;  }  DanhDauCacCongViecDaHoanThanh(a->right, ngay, giobatdau, tencv);  DanhDauCacCongViecDaHoanThanh(a->left, ngay, giobatdau, tencv);  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n^2) |

|  |
| --- |
| void XemCacCongViecDaHoanhThanh(NodeBST \*a)  {  if (a != NULL)  {  a->cv.inDS\_Theo\_Cong\_Viec\_Da\_Hoan\_Thanh();  XemCacCongViecDaHoanhThanh(a->right);  XemCacCongViecDaHoanhThanh(a->left);  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n^2) |

|  |
| --- |
| void XemCacCongViecChuaHoanhThanh(NodeBST \*a)  {  if (a != NULL)  {  a->cv.inDS\_Theo\_Cong\_Viec\_Chua\_Hoan\_Thanh();  XemCacCongViecChuaHoanhThanh(a->right);  XemCacCongViecChuaHoanhThanh(a->left);  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n^2) |

|  |
| --- |
| void XemCacCongViecTuNgayaDenNgayb(NodeBST \*a, int x, int y)  {  if (a != NULL)  {  if (a->cv.getten\_ngay() >= x && a->cv.getten\_ngay() <= y)  {  a->cv.inDS();  }  XemCacCongViecTuNgayaDenNgayb(a->right, x, y);  XemCacCongViecTuNgayaDenNgayb(a->left, x, y);  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n^2) |

|  |
| --- |
| void DeleteCacCongViecDaHoanThanh(NodeBST \*a)  {  if (a != NULL)  {  node tmp = a->cv.gethead();  while (findPos(tmp) != 0)  {  if (findPos(tmp) == 1)  {  deleteDau(tmp);  a->cv.sethead(tmp); // dieu chinh lai gia tri cua node dau vi khi xoa dau thi dia chi thay doi  }  else if (findPos(tmp) == Size(tmp))  {  deleteCuoi(tmp);  }  else  {  deleteGiua(tmp, findPos(tmp));  }  }  DeleteCacCongViecDaHoanThanh(a->right);  DeleteCacCongViecDaHoanThanh(a->left);  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n^2) |

|  |
| --- |
| void FindNodeTheMang(NodeBST \*&x, NodeBST \*&y)  {  if (y->left != NULL)  {  FindNodeTheMang(x, y->left);  }  else  {  x->cv = y->cv;  x = y;  y = y->right;  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(log(n)) |

|  |
| --- |
| void DeleteNode(NodeBST \*&a, int ten\_ngay)  {  if (a == NULL)  {  return;  }  if (a->cv.getten\_ngay() == ten\_ngay)  {  NodeBST \*p = a;  if (a->left == NULL)  {  if(a==headBST)  {  headBST=a->right;  --cnt;  }  else  {  a = a->right;  --cnt;  }  }  else if (a->right == NULL)  {  if(a==headBST)  {  headBST=a->left;  --cnt;  }  else  {  a = a->left;  --cnt;  }  }  else if (a->left != NULL && a->right != NULL) // ca cay con trai va cay con phai deu khác NULL  {  NodeBST \*q = a;  FindNodeTheMang(p, q->right);  --cnt;  }  delete p; // xoa node  return;  }  else if (a->cv.getten\_ngay() < ten\_ngay)  {  DeleteNode(a->right, ten\_ngay);  }  else if (a->cv.getten\_ngay() > ten\_ngay)  {  DeleteNode(a->left, ten\_ngay);  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(log(n)) |

|  |
| --- |
| void DieuChinhLichCongTac(NodeBST \*a, NodeBST \*&b)  {  if (a != NULL)  {  if (a->cv.gethead() == NULL)  {  DeleteNode(b, a->cv.getten\_ngay());  }  DieuChinhLichCongTac(a->left, b);  DieuChinhLichCongTac(a->right, b);  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n^2) |

|  |
| --- |
| void xuat(NodeBST \*a)  {  if (a != NULL)  {  xuat(a->left); // duyet trai  a->cv.inDS();  xuat(a->right); // duyet phai  }  }  🡺phương thức trên có độ phức tạp tiệm cận O(n) |

# III. DANH SÁCH TÀI LIỆU THAM KHẢO

Tài Liệu Tham Khảo:

+[lms.utc.edu.vn](https://lms.utc.edu.vn/course/view.php?id=355)

+[Wikipedia](https://vi.wikipedia.org/wiki/C%C3%A2y_t%C3%ACm_ki%E1%BA%BFm_nh%E1%BB%8B_ph%C3%A2n)