TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI HÀ NỘI

KHOA CÔNG NGHỆ THÔNG TIN

---------- ℘ 🕮 ℘ ----------

BÀI TẬP LỚN

MÔN: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Giáo viên hướng dẫn : TS. Hoàng Văn Thông

Sinh viên thực hiện: Trương Văn Minh

Lớp: CNTT4 – K63

Hà Nội, 2023

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI HÀ NỘI

KHOA CÔNG NGHỆ THÔNG TIN

---------- ℘ 🕮 ℘ ----------

BÀI TẬP LỚN

MÔN: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Giáo viên hướng dẫn : TS. Hoàng Văn Thông

Sinh viên thực hiện: Trương Văn Minh

Lớp: CNTT4 – K63

Hà Nội, 2023

Mục lục

Lời mở đầu

Đầu tiên em xin gửi lời chào trân trọng và lòng biết ơn sâu sắc đến thầy Hoàng Văn Thông về sự hướng dẫn và cung cấp kiến thức quý báu trong môn học Cấu Trúc Dữ Liệu và Giải Thuật. Bài báo cáo này là kết quả của sự nỗ lực không ngừng nghỉ của em trong quá trình thực hiện bài tập lớn, và em xin được trình bày nó với lòng tôn trọng và hy vọng sẽ đánh giá cao sự hỗ trợ của thầy.

Bài tập lớn này đã giúp em hiểu rõ hơn về quy trình thiết kế cấu trúc dữ liệu và giải thuật, cùng với khả năng áp dụng kiến thức vào thực tế. Nó cũng giúp em phát triển kỹ năng, tư duy logic và khả năng giải quyết vấn đề.

Một lần nữa em xin gửi lời cảm ơn đặc biệt đến thầy vì sự hỗ trợ, sự hướng dẫn và những lời khuyên quý báu trong suốt thời gian qua. Không có được những sự chỉ dẫn của thầy, em không thể hoàn thành bài tập này một cách hiệu quả.

Bài báo cáo này sẽ tập trung trình bày về nhiệm vụ của em, quá trình thiết kế và triển khai cấu trúc dữ liệu và giải thuật, cũng như kết quả đạt được. Em rất mong nhận được sự đánh giá và phản hồi từ thầy Hoàng Văn Thông để cải thiện kiến thức và kỹ năng của bản thân mình.

Bài 1. (Bài số 10) Xây dựng lớp quản lý nhân sự bằng danh sách liên kết đơn.

I. Đề bài

1. Xây dựng lớp quản lý nhân sự bằng danh sách liên kết đơn, mỗi cán bộ là một cấu trúc gồm:

- Mã cán bộ
- Họ đệm, tên cán bộ
- Phòng ban
- Chức vụ
- Hệ số lương

Với các phương thức sau:

a. Tạo danh sách cán bộ: Quá trình nhập danh sách sẽ dừng lại khi nhập mã cán bộ <= 0.

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

c. Tính lương cho nhân viên, biết rằng: Lương = Hệ số lương * 1350000

d. In lên màn hình tất cả cán bộ có hệ số lương >= 4.4

e. Tìm và in danh sách cán bộ theo Chức vụ

f. Tìm và in danh sách cán bộ theo hệ số lương và phòng ban (nghĩa là nhập vào hệ số lương, tên phòng ban cần tìm, sau đó in danh sách những cán bộ thỏa mãn cả 2 điều kiện này).

g. Sắp xếp danh sách cán bộ theo thứ tự của tên.

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

II. Phân tích bài toán

1. Xác định các lớp, các thuộc tính, các phương thức của lớp và mô tả chức năng của từng lớp, từng phương thức.

Xây dựng lớp node: tạo biến data để lưu giá trị và con trỏ next để trỏ đến thành phần kế tiếp của danh sách.

```
class node
{
   T data;
   node *next;
};
```

Xây dựng các phương thức khác của node như khởi tạo, phương thức gán hay thiết lập giá trị cho data, next, truy xuất giá trị của data, next.

```
node() {}
node(T data) {}
node &operator = (const node&x){ }
 void setE(T x) { }
 void setNext(node<T> *x) }
 T &getE() { }
 node<T> *&getNext()}
 ~node(){ }
```

Xây dựng lớp iter: tạo biến a là một con trỏ node để có thể duyệt qua danh sách liên kết.

```
class iter{
   node<T> *a;
}
```

Lớp iter gồm các phương thức khởi tạo, phương thức gán, so sánh, tăng tiền tố , hậu tố, lấy giá trị, truy xuất node:

```
iter() { }
iter(node<T> *x) {; }
node<T> *getnode() { }
```

iter<*T*> &operator=(iter<*T*> *x*) {}

bool operator!=(iter<*T*> *x*) { }

iter<*T*> operator++(){}

iter<*T*> operator++(int) {}

T &operator*() {}

Xây dựng lớp Don_list với chức năng là một danh sách liên kết đơn: tạo hai con trỏ node để trỏ tới đầu và cuối danh sách, biến num để lưu trữ số node có trong danh sách:

```
class Don_list{
   node<T> *head, *end;
   int num;
}
```

Lớp Don_list gồm các phương thức khởi tạo, truy xuất số node, truy xuất giá trị node đầu cuối, phương thức gán, …

Don_list(){}

int size() { }

bool empty() { }

T &front() { }

T &back() { }

Don_list &operator = (const *Don_list*&*x*){}

void push_front(*T x*) {}

void insert(node<*T*> **p*, *T x*) {}

void push_back(*T x*){}

void erase_head(){}

void erase(node<*T*> **p*) {}

void pop_front(){}

void pop_back(){}

iter<*T*> dau() { }

iter<*T*> cuoi() { }

Xây dựng lớp can_bo: gồm các thuộc tính id, ho, ten, ban, cvu, luong và phương thức tinh_luong() để quản lý thông tin cán bộ:

```
class can_bo
{
   int id;
   string ho, ten, ban, cvu;
   float hsl;
   long long luong;
   void tinh_luong(){}
}
```

Lớp cán bộ gồm các phương thức khởi tạo, gán, nhập, xuất, truy xuất mã cán bộ, họ tên cán bộ, chức vụ, phòng ban, lương, hệ số lương:

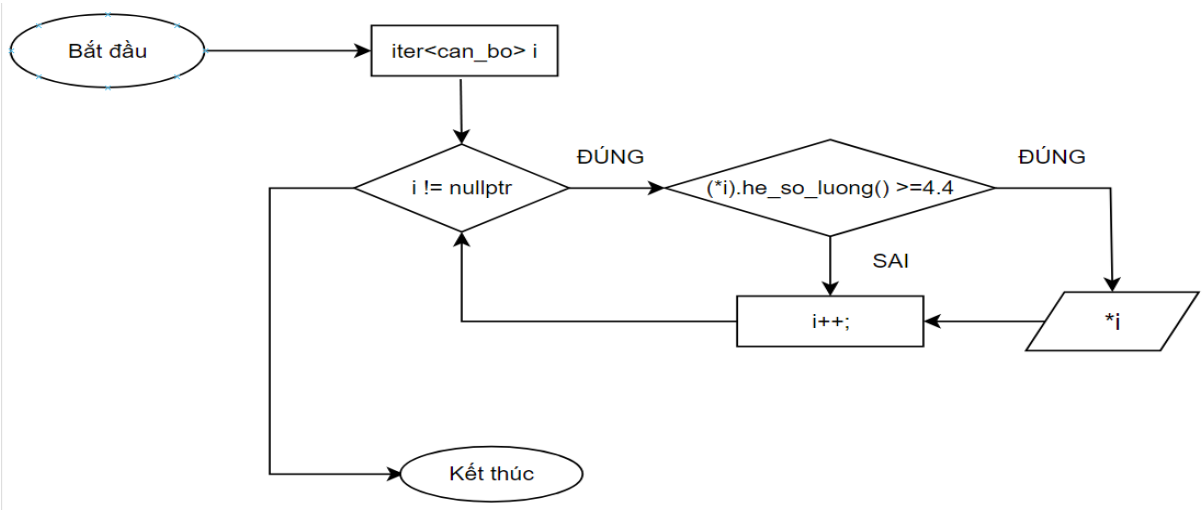
```
can_bo(int id = 0, string ho = "", string ten = "", string ban = "", string cvu = "", float hsl = 0) {}
can_bo(const can_bo &x) {}
can_bo &operator=(const can_bo &x) {}
bool input(){}
friend istream &operator>>(istream &cin, can_bo &x) {}
friend ostream &operator<<(ostream &cout, const can_bo &x) {}
int ma_can_bo() { }
string ho_ten(){}
string Ho() { }
string Ten() { }
string phong_ban() { }
string chuc_vu() { }
float he_so_luong() { }
long long Luong() { }
```

2. Cài đặt cấu trúc Don_list với kiểu dữ liệu can_bo.

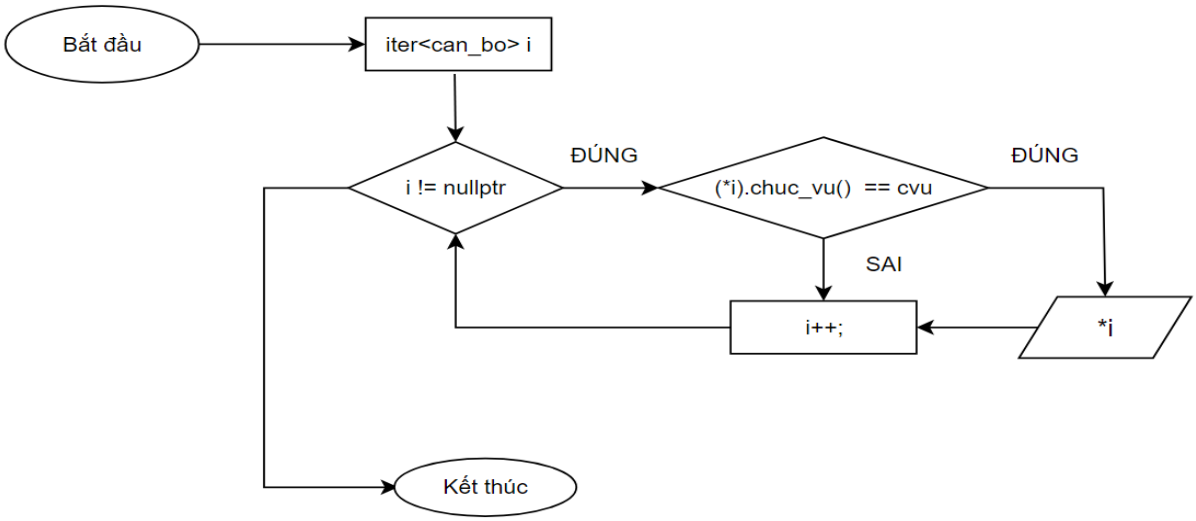
Cấu trúc danh sách liên kết đơn gồm hai phần, một là dữ liệu được lưu trong node (can_bo) và địa chỉ của node tiếp sau nó.

Lớp cán bộ chứa các thuộc tính để xây dựng node chứa thông tin cán bộ.

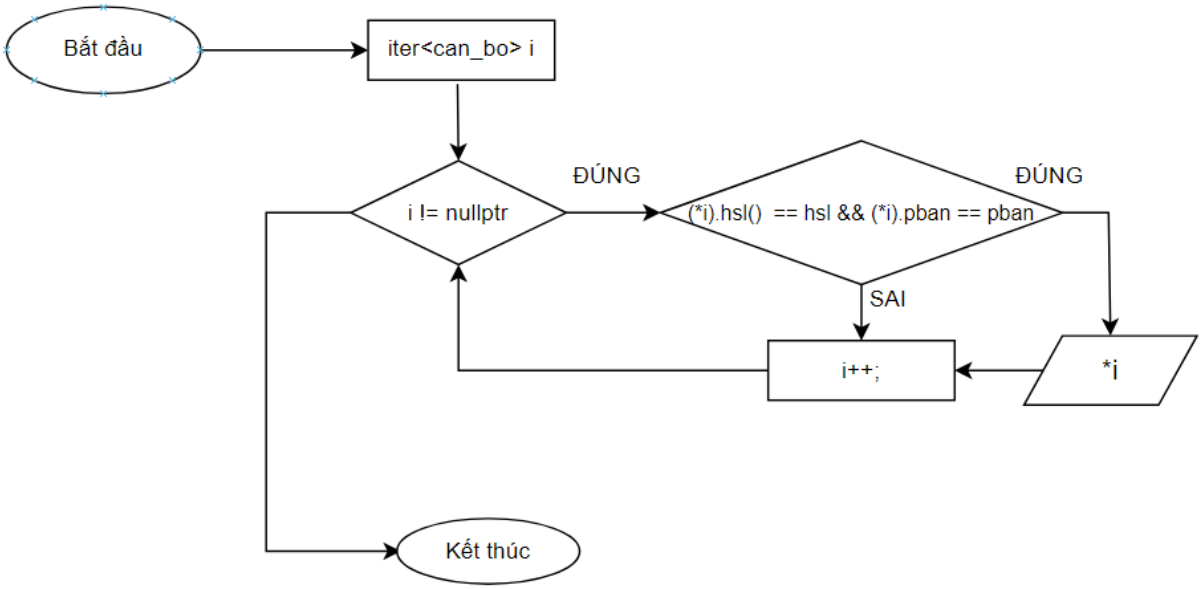
- Phương thức nhập vào dữ liệu của từng thông tin từ bàn phím:
    - Phương thức tạo mới danh sách cán bộ:
        - Nhập vào thông tin của từng cán bộ cho đến khi mã cán bộ nhỏ hơn 1.
    - Phương thức thêm 1 cán bộ:
        - Nhập vào từng cán bộ theo vị trí người dùng chọn.
        - Nếu không chọn thì sẽ được thêm vào cuối danh sách.
- Phương thức sắp xếp danh sách cán bộ: sắp xếp danh sách cán bộ bằng thuật toán Bubble sort.
- Phương thức hiển thị danh sách cán bộ.
- Phương thức hiển thị danh sách cán bộ có hệ số lương cao hơn 4.4. Thuật toán được mô tả theo sơ đồ:



- Phương thức tìm cán bộ theo chức vụ:



- Phương thức tìm cán bộ theo hệ số lương và phòng ban:



- Hàm main gọi tới các phương thức đã được xử lý.

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

1. Flie class node

```
#include <bits/stdc++.h>
#ifndef _node_CPP
#define _node_CPP
using namespace std;
template <class T>
class node
{
   T data;
   node *next;

public:
   node() {}
   node(T data)
   {
      next = new node;
      next = nullptr;
      this->data = data;
   }
   void setE(T x) { data = x; }
   void setNext(node<T> *x) { next = x; }
   T &getE() { return data; }
   node<T> *&getNext() { return next; }
   ~node()
   {
      delete next;
   }
```

```
};
#endif
```

2. Flie class iter

```
#include "node.cpp"
#ifndef _iterator_CPP
#define _iterator_CPP
using namespace std;
template <class T>
class iter
{
   node<T> *a;

public:
   iter() { a = nullptr; }
   iter(node<T> *x) { a = x; }
   node<T> *getnode() { return a; }
   iter<T> &operator=(iter<T> x)
   {
      this->a = x.getnode();
      return *this;
   }
   bool operator!=(iter<T> x) { return a != x.getnode(); }
   iter<T> operator++()
   {
      a->getNext();
      return a;
   }
   iter<T> operator++(int)
   {
      iter<T> temp = *this;
      a = a->getNext();
      return temp;
   }
   T &operator*()
   {
      return a->getE();
   }
};
#endif
```

3. Flie class Don_list

```
#include "node_iterator.cpp"
#ifndef _linked_list_CPP
#define _linked_list_CPP
```

```
using namespace std;
template <class T>
class Don_list
{
   node<T> *head, *end;
   int num;

public:
   Don_list()
   {
      head = end = nullptr;
      num = 0;
   }
   int size() { return num; }
   bool empty() { return num == 0; }
   T &front() { return head->getE(); }
   T &back() { return end->getE(); }
   iter<T> dau() { return head; }
   iter<T> cuoi() { return end; }
   void push_front(T x)
   {
      node<T> *temp = new node<T>;
      temp->setNext(head);
      temp->setE(x);
      head = temp;
      if (!num)
         end = temp;
      num++;
   }
   void insert(node<T> *p, T x)
   {
      if (!num)
         push_front(x);
      else
      {
         node<T> *temp = new node<T>;
         temp->setNext(p->getNext());
         p->setNext(temp);
         temp->setE(x);
         if (end == p)
            end = temp;
         num++;
      }
   }
   void push_back(T x)
   {
      insert(cuoi().getnode(),x);
   }
```

```
        void erase_head()
        {
            if (!num)
                return;
            if (num == 1)
            {
                head = end = nullptr;
                return;
            }
            node<T> *temp = new node<T>;
            temp = head->getNext();
            head = temp;
            num--;
        }
        void erase(node<T> *p)
        {
            node<T> *temp = p->getNext();
            p = temp->getNext();
            num--;
        }
        void pop_front()
        {
            erase_head();
        }
        void pop_back()
        {
            if (num < 2)
                erase_head();
            else
            {
                node<T> *temp = new node<T>;
                while (temp->getNext() != end)
                {
                    temp++;
                }
                erase(temp);
            }
        }
    };
#endif
```

4. File class can_bo

```
#include <bits/stdc++.h>
#ifndef _can_bo_cpp
#define _can_bo_cpp
using namespace std;
class can_bo
{
   int id;
   string ho, ten, ban, cvu;
   float hsl;
   long long luong;
   void tinh_luong()
   {
      luong = hsl * 1350000;
   }

public:
   can_bo(int id = 0, string ho = "", string ten = "", string ban = "", string cvu =
"", float hsl = 0)
   {
      this->id = id;
      this->ho = ho;
      this->ten = ten;
      this->ban = ban;
      this->cvu = cvu;
      this->hsl = hsl;
      tinh_luong();
   }
   can_bo(const can_bo &x)
   {
      this->id = x.id;
      this->ho = x.ho;
      this->ten = x.ten;
      this->ban = x.ban;
      this->cvu = x.cvu;
      this->hsl = x.hsl;
      tinh_luong();
   }
   can_bo &operator=(const can_bo &x)
   {
      this->id = x.id;
      this->ho = x.ho;
      this->ten = x.ten;
      this->ban = x.ban;
      this->cvu = x.cvu;
      this->hsl = x.hsl;
      tinh_luong();
      return *this;
```

```
}
bool input()
{
   cout << "\tNhap ma can bo: ";
   cin >> id;
   if (id <= 0)
      return false;
   cin.ignore(1);
   cout << "\tNhap ho ten can bo: ";
   string temp;
   getline(cin, temp);
   bool test = false;
   for (int i = temp.length() - 1; i >= 0; i--)
   {
      if (temp[i] == ' ')
      {
         ho = temp.substr(0, i);
         ten = temp.substr(i + 1, temp.length() - i);
         test = true;
         break;
      }
      if (!test)
      {
         ten = temp;
         ho = "";
      }
   }
   cout << "\tPhong ban: ";
   getline(cin, ban);
   cout << "\tChuc vu: ";
   getline(cin, cvu);
   cout << "\tHe so luong: ";
   cin >> hsl;
   tinh_luong();
   return true;
}
friend istream &operator>>(istream &cin, can_bo &x)
{
   cout << "\tNhap ma can bo: ";
   cin >> x.id;
   cin.ignore(1);
   cout << "\tNhap ho ten can bo: ";
   string temp;
   getline(cin, temp);
   bool test = false;
   for (int i = temp.length() - 1; i >= 0; i--)
   {
      if (temp[i] == ' ')
```

```
            {
                x.ho = temp.substr(0, i);
                x.ten = temp.substr(i + 1, temp.length() - i);
                test = true;
                break;
            }
            if (!test)
            {
                x.ten = temp;
                x.ho = "";
            }
        }
        cout << "\tPhong ban: ";
        getline(cin, x.ban);
        cout << "\tChuc vu: ";
        getline(cin, x.cvu);
        cout << "\tHe so luong: ";
        cin >> x.hsl;
        x.tinh_luong();
        return cin;
    }
    friend ostream &operator<<(ostream &cout, const can_bo &x)
    {
        cout << "\tMa can bo: " << x.id << endl;
        cout << "\tHo ten can bo: " << x.ho << " " << x.ten << endl;
        cout << "\tPhong ban: " << x.ban << endl;
        cout << "\tChuc vu: " << x.cvu << endl;
        cout << "\tHe so luong: " << x.hsl << endl;
        cout << "\tLuong: " << x.luong << endl;
        return cout;
    }
    int ma_can_bo() { return id; }
    string ho_ten()
    {
        string temp = ho + " " + ten;
        return temp;
    }
    string Ho() { return ho; }
    string Ten() { return ten; }
    string phong_ban() { return ban; }
    string chuc_vu() { return cvu; }
    float he_so_luong() { return hsl; }
    long long Luong() { return luong; }
};
#endif
```

5. File class list_cb

```
#include "can_bo.cpp"
#include "linked_list.cpp"
#ifndef _danh_sach_cpp
#define _danh_sach_cpp
using namespace std;
bool cnt = true;
class list_cb
{
   Don_list<can_bo> a;

public:
   list_cb() {}
   list_cb(Don_list<can_bo> a)
   {
      this->a = a;
   }
   bool create_list()
   {
      cout << "\nTao danh sach can bo\nLuu y nhap ma can bo nho hon 1 de
dung!\n\n";
      int i = 0;
      bool x;
      while (1)
      {
         cout << "Nhap thong tin can bo thu " << ++i << endl;
         can_bo temp;
         x = temp.input();
         if (!x)
            break;
         a.push_back(temp);
      }
      if (i == 1 && !x)
         return true;
      else
         return false;
   }
   void add()
   {
      cout << "Hien tai danh sach co " << a.size() << " can bo.\n";
      cout << "Ban muon chon vi tri se them?\n";
      cout << "\t1. Chon vi tri.\n";
      cout << "\t2. Khong chon\n";
      int t;
      cin >> t;
      switch (t)
      {
      case 1:
```

```
            {
                cout << "\tChon vi tri muon them: ";
                int n;
                cin >> n;
                while (n < 1 || n > a.size() + 1)
                {
                    cout << "Khong hop le, vui long nhap lai: ";
                    cin >> n;
                }
                can_bo temp;
                cin >> temp;
                if (n == 1)
                    a.push_front(temp);
                else if (n == a.size() + 1)
                    a.push_back(temp);
                else
                {
                    int j = 0;
                    for (iter<can_bo> i = a.dau(); i != nullptr; i++, j++)
                        if (j == n - 1)
                            a.insert(i.getnode(), temp);
                }
                break;
            }
            case 2:
            {
                can_bo temp;
                cin >> temp;
                a.push_back(temp);
                break;
            }
            default:
                cout << "\tThao tac khong hop le!\n";
                break;
            }
        }
        void sort()
        {
            for (iter<can_bo> i = a.dau(); i != nullptr; i++)
            {
                for (iter<can_bo> j = i; j != nullptr; j++)
                {
                    can_bo X = *i;
                    can_bo Y = *j;
                    if (X.Ten() > Y.Ten() || (X.Ten() == Y.Ten() && X.Ho() > Y.Ho()))
                    {
                        j.getnode()->setE(X);
                        i.getnode()->setE(Y);
```

```
            }
        }
    }
}
void display()
{
    int j = 0;
    for (iter<can_bo> i = a.dau(); i != nullptr; i++)
    {
        if ((*i).he_so_luong() >= 4.4)
        {
            if (!j)
                cout << "Danh sach cac can bo co he so luong tren 4.4 la:\n";
            cout << ++j << ". " << *i << endl;
        }
    }
    if (!j)
        cout << "Khong co can bo co he so luong tren 4.4\n";
    return;
}
void _display()
{
    int j = 0;
    for (iter<can_bo> i = a.dau(); i != nullptr; i++)
    {
        cout << ++j << ". " << *i << endl;
    }
    return;
}
void search(string cvu)
{
    int j = 0;
    for (iter<can_bo> i = a.dau(); i != nullptr; i++)
    {
        if ((*i).chuc_vu() == cvu)
        {
            if (!j)
                cout << "Danh sach cac can bo co chuc vu " << cvu << " la" << endl;
            cout << ++j << ". " << *i << endl;
        }
    }
    if (!j)
        cout << "Khong co can bo giu chuc vu " << cvu << endl;
    return;
}
void search(float hsl, string phong)
{
    int j = 0;
```

```
        for (iter<can_bo> i = a.dau(); i != nullptr; i++)
        {
            if ((*i).he_so_luong() == hsl && (*i).phong_ban() == phong)
            {
                if (!j)
                    cout << "Danh sach cac can bo co he so luong tren 4.4 o phong ban "
<< phong << "la:\n";
                cout << ++j << ". " << *i << endl;
            }
        }
        if (!j)
            cout << "Khong co can bo co he so luong " << hsl << " o phong " <<
phong << endl;
        return;
    }
    void home()
    {
        if (cnt)
        {
            cout << "\t-----------------^-^-----------------\n";
            cout << "\t|\tDanh sach can bo trong      |\n";
            cout << "\t-------------------------------------\n";
            cout << "\t|\t1. Tao moi danh sach        |\n";
            cout << "\t|\t2. Thoat chuong trinh.      |\n";
            cout << "\t-------------------------------------\n";
            int n;
            cout << "Chon tac vu mong muon: ";
            cin >> n;
            switch (n)
            {
            case 1:
            {
                cout << "\n-------------------------^-^-----------------------------\n";
                cnt = create_list();
                cout << "\nTao moi hoan tat!\n\n";
                cout << "-------------------------^-^-----------------------------\n";
                cout << "Nhap 'y' de tro ve man hinh chinh hoac 'n' de thoat: ";
                char x;
                cin >> x;
                if (x == 'y')
                    home();
                else if (x == 'n')
                    return;
                else
                    while (x != 'y' && x != 'n')
                    {
                        cout << "\nKhong hop le, vui long nhap lai: ";
                        cin >> x;
```

```
                }
            if (x == 'y')
                home();
            else if (x == 'n')
                return;
            break;
        }
        case 2:
            return;
        default:
        {
            cout << "Thao tac khong hop le!\nBan co muon tiep tuc.\n'y' de tiep tuc
hoac 'n' de tu choi: ";
            char x;
            cin >> x;
            if (x == 'y')
                home();
            else
                return;
            break;
        }
        }
    }
    else
    {
        cout << "\t-------------------------^-^-----------------------------\n";
        cout << "\t|\t1. Tao danh sach moi.                          |\n";
        cout << "\t|\t2. Them mot can bo moi.                        |\n";
        cout << "\t|\t3. Xem danh sach can bo.                       |\n";
        cout << "\t|\t4. Thong ke can bo co he so luong tren 4.4.    |\n";
        cout << "\t|\t5. Loc theo chuc vu.                           |\n";
        cout << "\t|\t6. Loc theo he so luong va phong ban.          |\n";
        cout << "\t|\t7. Sap xep danh sach.                          |\n";
        cout << "\t|\t8. Thoat chuong trinh.                         |\n";
        cout << "\t-------------------------------------------------------\n";
        int n;
        cout << "Chon tac vu mong muon: ";
        cin >> n;
        switch (n)
        {
        case 1:
        {
            cout << "\n-------------------------^-^-----------------------------\n";
            cout << "Luu y! Viec tao moi se xoa di danh sach dang co.\n";
            cout << "Nhap 'y' de xac nhan hoac 'n' de quay lai: ";
            cout << endl;
            char x;
            cin >> x;
```

```
            if (x == 'y')
            {
                cnt = create_list();
                cout << "\nTao moi hoan tat!\n";
                cout << "-------------------------^-^-----------------------------\n";
                cout << "Nhap 'y' de tro ve man hinh chinh hoac 'n' de thoat: ";
                cin >> x;
                if (x == 'y')
                    home();
                else
                    return;
            }
            else
                home();
            break;
        }
        case 2:
        {
            cout << "\n-------------------------^-^-----------------------------\n";
            add();
            cout << "\n-------------------------^-^-----------------------------\n";
            cout << "Nhap 'y' de tro ve man hinh chinh hoac 'n' de thoat: ";
            char x;
            cin >> x;
            if (x == 'y')
                home();
            else if (x == 'n')
                return;
            else
                while (x != 'y' && x != 'n')
                {
                    cout << "\nKhong hop le, vui long nhap lai: ";
                    cin >> x;
                }
            if (x == 'y')
                home();
            else if (x == 'n')
                return;
            break;
        }
        case 3:
        {
            cout << "\n-------------------------^-^-----------------------------\n";
            _display();
            cout << "\n-------------------------^-^-----------------------------\n";
            cout << "Nhap 'y' de tro ve man hinh chinh hoac 'n' de thoat: ";
            char x;
            cin >> x;
```

```
            if (x == 'y')
                home();
            else if (x == 'n')
                return;
            else
                while (x != 'y' && x != 'n')
                {
                    cout << "\nKhong hop le, vui long nhap lai: ";
                    cin >> x;
                }
            if (x == 'y')
                home();
            else if (x == 'n')
                return;
            break;
        }
        case 4:
        {
            cout << "\n-------------------------^-^-----------------------------\n";
            display();
            cout << "\n-------------------------^-^-----------------------------\n";
            cout << "Nhap 'y' de tro ve man hinh chinh hoac 'n' de thoat: ";
            char x;
            cin >> x;
            if (x == 'y')
                home();
            else if (x == 'n')
                return;
            else
                while (x != 'y' && x != 'n')
                {
                    cout << "\nKhong hop le, vui long nhap lai: ";
                    cin >> x;
                }
            if (x == 'y')
                home();
            else if (x == 'n')
                return;
            break;
        }
        case 5:
        {
            cout << "\n-------------------------^-^-----------------------------\n";
            cout << "Nhap chuc vu can tim: ";
            string y;
            cin.ignore(1);
            getline(cin, y);
            search(y);
```

```
            cout << "\n\t------------------------^-^----------------------------\n";
            cout << "Nhap 'y' de tro ve man hinh chinh hoac 'n' de thoat: ";
            char x;
            cin >> x;
            if (x == 'y')
               home();
            else if (x == 'n')
               return;
            else
               while (x != 'y' && x != 'n')
               {
                  cout << "\nKhong hop le, vui long nhap lai: ";
                  cin >> x;
               }
            if (x == 'y')
               home();
            else if (x == 'n')
               return;
            break;
         }
         case 6:
         {
            cout << "\n-------------------------^-^----------------------------\n";
            cout << "Nhap he so luong va phong ban can can tim: ";
            string z;
            float y;
            cin >> y;
            cin.ignore(1);
            getline(cin, z);
            search(y, z);
            cout << "\n-------------------------^-^----------------------------\n";
            cout << "Nhap 'y' de tro ve man hinh chinh hoac 'n' de thoat: ";
            char x;
            cin >> x;
            if (x == 'y')
               home();
            else if (x == 'n')
               return;
            else
            {
               while (x != 'y' && x != 'n')
               {
                  cout << "\nKhong hop le, vui long nhap lai: ";
                  cin >> x;
               }
               if (x == 'y')
                  home();
               else if (x == 'n')
```

```
                return;
            if (x == 'y')
                home();
            else if (x == 'n')
                return;
        }
        break;
    }
    case 7:
    {
        cout << "\n------------------------^-^----------------------------\n";
        cout << "Sap xep thanh cong!\n";
        sort();
        _display();
        cout << "\nNhap 'y' de tro ve man hinh chinh hoac 'n' de thoat: ";
        char x;
        cin >> x;
        if (x == 'y')
            home();
        else if (x == 'n')
            return;
        else
            while (x != 'y' && x != 'n')
            {
                cout << "\nKhong hop le, vui long nhap lai: ";
                cin >> x;
            }
        if (x == 'y')
            home();
        else if (x == 'n')
            return;
        break;
    }
    case 8:
        return;
    default:
    {
        cout << "Thao tac khong hop le!\nBan co muon tiep tuc.\n'y' de tiep tuc
hoac 'n' de tu choi: ";
        char x;
        cin >> x;
        if (x == 'y')
            home();
        else
            return;
        break;
    }
    }
```

```
            }
        }
        void end_task()
        {
            cout << "\t\t    __        __" << endl;
            cout << "\t\t ********  ********" << endl;
            cout << "\t\t****_____________****" << endl;
            cout << "\t\t****| Goodbye! |***" << endl;
            cout << "\t\t ***------------**" << endl;
            cout << "\t\t   **************" << endl;
            cout << "\t\t      *******" << endl;
            cout << "\t\t         *" << endl;
        }
    };
#endif
```

6. File main

```
    #include "danh_sach_can_bo.cpp"
    using namespace std;
    int main()
    {
        list_cb a;
        a.home();
        a.end_task();
        return 0;
}
```

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

1. Class node

Phương thức khởi tạo, setE, setNext, getE, getNext: O(1).

2. Class iter

Phương thức khởi tạo, getNode, gán, toán tử so sánh, toán tử tăng, toán tử *: O(1)

3. Class Don_list

Phương thức khởi tạo, size, empty, front, back, toán tử gán, push_front, push_back, insert, erase_head, erase, pop_front, pop, dau, cuoi: O(1).

Phương thức pop_back: O(n).

4. Class can_bo

Phương thức khởi tạo, gán, copy, toán tử xuất, ho_ten, Ho, Ten, phong_ban, chuc_vu, he_so_luong, luong, ma_can_bo: O(1).

5. Class list_cb

Phương thức khởi tạo: O(1)

Phương thức create_list, add, display, _display, search: O(n).

Phương thức sort: $O(n^2)$.

Bài 2: (bài số 21) Ứng dụng danh sách liên kết đơn giải quyết bài toán đồ thị.

I. Đề bài

1. Ứng dụng danh sách liên kết đơn để giải quyết bài toán đồ thị

- Xây lớp đồ thị (Graph) vô hướng có trọng số, đồ thị được mô tả bằng danh sách kề, có các phương thức:

a. Đọc đồ thị từ file

b. Ghi đồ thị ra file

c. Xây dựng các đồ thị con của nó, mỗi đồ thị con là một thành phần liên thông, Phương thức trả lại danh sách đồ thị con của đồ thị

d. Duyệt đồ thị theo chiều rộng (BFS) nếu đồ thị liên thông

e. Duyệt đồ thị theo chiều sâu (DFS) nếu đồ thị liên thông

f. Tìm đường đi ngắn nhất giữa 2 đỉnh bất kỳ nếu đồ thị liên thông

2. Viết hàm main, thực hiện các chức năng của lớp Graph

II. Phân tích bài toán

1. Xác định các lớp, các thuộc tính, các phương thức của lớp và mô tả chức năng của từng lớp, từng phương thức.
   Lớp node, iter, Don_list tương tự Bài 1.
   Lớp Graph để biểu diễn đồ thị bằng danh sách kề

```
class Graph{
   int n, m, _n;
   Don_list<int> *a;
   int *trongso;
}
```

Biến n lưu chỉ số lớn nhất của danh sách kề, m để lưu số cạnh, _n để lưu số đỉnh, trongso để lưu trọng số, Don_list để sử dụng như một danh sách kề.

Xây dựng các phương thức của lớp Graph:

- Khởi tạo: thực hiện cấp phát động để khởi tạo danh sách kề ban đầu cho đồ thị:

```
Graph(){}
Graph(int n, int _n, int m){
   this->_n = _n;
   this->m = m;
   this->n = n;
   this->a = new Don_list<int>[n];
   this->trongso = new int[n * n];
   int j = 0;
   for (int i = 0; i < n; i++){
     a[i].push_back(i);
     trongso[i * n + j++] = 0;
   }
}
```

- Phương thức hủy: ~Graph(){}
- Toán tử gán: *Graph* &operator=(const *Graph* &*x*) { }
- Phương thức thêm một cạnh vào đồ thị: void add(int *s*, int *d*, int *tso*){}
- Toán tử đọc đồ thị từ file:

  friend *ifstream* &operator>>(*ifstream* &*cin*, *Graph* &*x*) {}
- Toán tử đọc đồ thị từ bàn phím:

  friend *istream* &operator>>(*ifstream* &*cin*, *Graph* &*x*) {}
- Toán tử xuất đồ thị ra file:

  friend *ofstream* &operator<<(*ofstream* &*cout*, const *Graph* &*x*) {}
- Toán tử xuất đồ thị ra màn hình:

  friend *ofstream* &operator<<(*ofstream* &*cout*, const *Graph* &*x*) {}
- Phương thức tìm kiếm theo chiều sâu:

  void _dfs(int *u*, int *t*[]){}

  void DFS(int *u*, int *t*[]){}
- Phương tìm tìm kiếm theo chiều rộng:

  void _bfs(int *u*, int *t*[], int *_t*[], int &*j*, *ofstream* &*output*){}

void BFS(){}

- Phương thức đếm số thành phần liên thông:

  int _dem_lien_thong(){}

- Phương thức xây dựng đồ thị con:

  void con(*ofstream* &*output*) {}

- Phương thức tìm đường đi ngắn nhất (Dijkstra):

  Void dijkstra(){}

2. Cài đặt cấu trúc danh sách kề bằng Don_list với kiểu dữ liệu số nguyên

Cấu trúc Don_list như bài 1.

Lớp Graph chứa các thuộc tính để chưa thông tin về cạnh, số đỉnh, trọng số giữa các cạnh của đồ thị.

- Phương thức nhập vào đồ thị:
  - Đọc vào số đỉnh, số cạnh của đồ thị, các đỉnh, trọng số, sau đó tìm đỉnh có chỉ số lớn nhất.
  - Cấp phát động cho danh sách kề:

    x.a = **new** Don_list<int>[*x*.n];
    x.trongso = **new** int[*x*.n * *x*.n];

  - Hình thành danh sách kề ban đầu gồm các đỉnh từ 0 đến đỉnh có chỉ số lớn nhất.
  - Thêm lần lượt các cạnh cùng trọng số của các cạnh đã nhập vào danh sách kề.
- Phương thức xuất đồ thị:
  - Khởi tạo một ma trận tmp để đánh số các cạnh đã xuất để tránh các cạnh trùng lặp.
  - Duyệt lần lượt từ đầu danh sách kề, đến mỗi đỉnh sẽ duyệt danh sách liên kết của đỉnh đó và đánh dấu cặp đỉnh đã xuất.
- Phương thức đếm thành phần liên thông:

  Xây dựng phương thức _dem_lien_thong() sử dụng thuật toán DFS để phục vụ cho việc đếm thành phần liên thông bằng hàm _dfs().

- Phương thức tìm kiếm chiều sâu:
    - Xây dựng phương thức _DFS() sử dụng phương thức đếm thành phần liên thống để kiểm tra tính liên thông của đồ thị.
    - Nếu đồ thị liên thông sẽ duyệt bằng phương thức DFS().
    - Nếu đồ thị không liên thông, hủy phương thức tìm kiếm.
- Phương thức tìm kiếm chiều rộng
    - Xây dựng phương thức BFS() sử dụng phương thức đếm thành phần liên thông để kiểm tra tính liên thông của đồ thị.
    - Nếu đồ thị liên thông, khai báo mảng t[n+1] để đánh dấu đỉnh đã duyệt qua và queue để phục vụ cho việc duyệt BFS:
        - Push một đỉnh bất kỳ, sau đó duyệt các đỉnh con nằm trong danh sách liên kết của đỉnh đó.
        - Duyệt vòng lặp đến khi queue rỗng, trong mỗi vòng lặp, sẽ lấy ra giá trị đầu tiên của queue đang có.
        - Kiểm tra đỉnh đã được duyệt hay chưa, nếu chưa sẽ in ra và đánh dấu.
        - Tiếp tục push các đỉnh con của đỉnh vừa kiểm tra
    - Nếu đồ thị không liên thông, hủy phương thức tìm kiếm.
- Phương thức xây dựng đồ thị con:
    - Khai báo ma trận _t[n*n+1] để kiểm tra cặp đỉnh, mảng t[n+1] để đánh dấu.
    - Duyệt lần lượt các đỉnh trong đồ thị mẹ, kiểm tra xem đỉnh đã được duyệt hay chưa.
    - Nếu đỉnh chưa được duyệt sẽ dùng phương thức _bfs để xây dựng hồ thị con chứa đỉnh.
    - Trong phương thức _bfs khai báo queue để chưa các đỉnh phục vụ cho việc duyệt BFS.
    - Sử dụng vòng lặp while đến khi queue rỗng, trong mỗi vòng lặp kiểm tra xem t[v] và _t[k * n + v] với k là giá trị ở đầu queue để biết đã duyệt đến cặp đỉnh k – v chưa.
    - Nếu chưa sẽ in cạnh ra, push đỉnh v vào queue và đánh dấu cặp đỉnh là đã duyệt.

- Phương thức tìm đường đi ngắn nhất giữa hai đỉnh:
    - Sử dụng phương thức đếm thành phần liên thông để kiểm tra tính liên thông của đồ thị, nếu đồ thị không liên thông sẽ hủy phương thức tìm kiếm.
    - Nếu đồ thị liên thông, yêu cầu người dùng nhập vào cặp đỉnh cần tìm đường đi.
    - Khởi tạo vector d có n phần tử để lưu khoảng cách nhỏ nhất để đi từ đỉnh đầu đến mỗi đỉnh, vector res để lưu đỉnh có đường đi ngắn nhất từ đỉnh đầu.
    - Khởi tạo hàng đợi ưu tiên priority_queue với cặp giá trị là khoảng cách ngắn nhất đi đến mỗi đỉnh và đỉnh đó. Ưu tiên từ bé đến lớn
    - Sử dụng vòng lặp while đến khi hàng đợi rỗng. Trong mỗi vòng lặp sẽ lấy giá trị ở đầu hàng đợi, kiểm tra xem khoảng cách ngắn nhất ở các đỉnh kề hiện tại có ngắn hơn khoảng cách từ nó đi đến hay không. Nếu có đường đi khác ngắn hơn sẽ cập nhật và push vào hàng đợi. Cập nhật lại giá trị trong vector res để lưu lại đỉnh kề có đường đi tới ngắn nhất.
    - Khởi tạo vector _res để lưu đường đi ngắn nhất. Push đỉnh cuối vào, dùng vòng lặp while để truy xuất đường đi đến mỗi đỉnh.

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

1. File Don_list như bài 1.

2. File class Graph

```
#include "linked_list.cpp"
#ifndef _adjacency_list
#define _adjacency_list
using namespace std;
class Graph {
   int n, m, _n;
   Don_list<int> *a;
   float *trongso;

public:
```

```
Graph(){
    this->m = 0;
    this->n = 0;
    this->a = nullptr;
    this->trongso = nullptr;
}
Graph(int n, int _n, int m){
    this->_n = _n;
    this->m = m;
    this->n = n;
    this->a = new Don_list<int>[n];
    this->trongso = new float[n * n];
    int j = 0;
    for (int i = 0; i < n; i++)
    {
        a[i].push_back(i);
        trongso[i * n + j++] = 0;
    }
}
Graph &operator=(const Graph &x){
    delete[] this->a;
    delete[] this->trongso;
    this->_n = x._n;
    this->m = x.m;
    this->n = x.n;
    this->a = new Don_list<int>[x.n];
    this->trongso = new float[x.n * x.n];
    for (int i = 0; i < x.n; i++){
        for (int j = 0; j < x.n; j++){
            this->trongso[i * x.n + j] = x.trongso[i * x.n + j];
        }
        iter<int> temp = x.a[i].dau();
```

```
            while (temp != nullptr) {
                this->a[i].push_back(*temp);
                temp++;
            }
        }
        return *this;
    }
    ~Graph(){
        delete[] a;
        delete[] trongso;
    }
    void add(int s, int d, int tso) {
        a[s].push_back(d);
        a[d].push_back(s);
        trongso[s * n + d] = trongso[d * n + s] = tso;
    }
    friend ifstream &operator>>(ifstream &cin, Graph &x) {
        cin >> x._n >> x.m;
        int tmp[x.m][2], tso[x.m];
        for (int i = 0; i < x.m; i++){
            cin >> tmp[i][0] >> tmp[i][1] >> tso[i];
            x.n = max(tmp[i][0], max(tmp[i][1], x.n));
        }
        x.n++;
        int j = 0;
        x.a = new Don_list<int>[x.n];
        x.trongso = new float[x.n * x.n];
        for (int i = 0; i < x.n; i++){
            x.a[i].push_back(i);
            x.trongso[i * x.n + j++] = 0;
        }
        for (int i = 0; i < x.m; i++){
```

```
            x.add(tmp[i][0], tmp[i][1], tso[i]);
        }
        cout << "Doc thanh cong!\n";
        return cin;
    }
    friend istream &operator>>(istream &cin, Graph &x) {
        cin >> x._n >> x.m;
        int tmp[x.m][2], tso[x.m];
        for (int i = 0; i < x.m; i++){
            cin >> tmp[i][0] >> tmp[i][1] >> tso[i];
            x.n = max(tmp[i][0], max(tmp[i][1], x.n));
        }
        x.n++;
        int j = 0;
        x.a = new Don_list<int>[x.n];
        x.trongso = new float[x.n * x.n];
        for (int i = 0; i < x.n; i++){
            x.a[i].push_back(i);
            x.trongso[i * x.n + j++] = 0;
        }
        for (int i = 0; i < x.m; i++){
            x.add(tmp[i][0], tmp[i][1], tso[i]);
        }
        cout << "Doc thanh cong!\n";
        return cin;
    }
    friend ofstream &operator<<(ofstream &cout, const Graph &x){
        int tmp[x.n * x.n] = {0};
        cout << "Số đỉnh: " << x._n << endl;
        for (int i = 0; i < x.n; i++){
            iter<int> temp = x.a[i].dau();
            if (temp != nullptr) {
```

```
            cout << "Đỉnh " << x.a[i].front() << " kết nối với:\n";
            int j = x.a[i].front();
            while (temp != nullptr){
                int k = *temp;
                if (!tmp[j * x.n + k]) {
                    tmp[j * x.n + k] = 1;
                    cout << "\tĐỉnh " << k << " có trọng số "
                                                << x.trongso[j * x.n + k] << endl;
                }
                temp++;
            }
            cout << endl;
        }
    }
    return cout;
}
friend ostream &operator<<(ostream &cout, const Graph &x){
    int tmp[x.n * x.n] = {0};
    cout << "So dinh: " << x.n << endl;
    for (int i = 0; i < x.n; i++){
        iter<int> temp = x.a[i].dau();
        if (temp != nullptr){
            cout << "Dinh " << x.a[i].front() << " ket noi voi:\n";
            int j = x.a[i].front();
            while (temp != nullptr){
                int k = *temp;
                if (!tmp[j * x.n + k]) {
                    tmp[j * x.n + k] = 1;
                    cout << "\tDinh " << k << " co trong so " << x.trongso[j * x.n + k]
                                                                << endl;
                }
                temp++;
```

```
            }
            cout << endl;
        }
    }
    return cout;
}
void _dfs(int u, int t[]){
    t[u] = 1;
    iter<int> temp = a[u].dau();
    while (temp != nullptr) {
        int v = *temp;
        if (!t[v])
            _dfs(v, t);
        temp++;
    }
}
void DFS(int u, int t[]){
    cout << u << " ";
    t[u] = 1;
    iter<int> temp = a[u].dau();
    while (temp != nullptr) {
        int v = *temp;
        if (!t[v])
            DFS(v, t);
        temp++;
    }
}
void _bfs(int u, int t[], int _t[], int &j, ofstream &output){
    bool cnt = true;
    queue<int> c;
    c.push(u);
    while (!c.empty()){
```

```
            int k = c.front();
            c.pop();
            t[k] = 1;
            iter<int> temp = a[k].dau();
            while (temp != nullptr){
                int v = *temp;
                if (!t[v] && !_t[k * n + v]) {
                    if (cnt) {
                        output << "Do thi con thu " << j << endl;
                        cout << "Do thi con thu " << j++ << endl;
                        cnt = false;
                    }
                    cout << "\t" << k << " " << v << endl;
                    output << "\t" << k << " " << v << endl;
                    c.push(v);
                    _t[k * n + v] = 1;
                }
                temp++;
            }
        }
    }
    void BFS(){
        if (_dem_lien_thong() == 1) {
            int t[n + 1] = {0};
            queue<int> c;
            c.push(1);
            while (!c.empty()){
                int k = c.front();
                c.pop();
                if (!t[k])
                    cout << k << " ";
                t[k] = 1;
```

```
            iter<int> temp = a[k].dau();
            while (temp != nullptr) {
                int v = *temp;
                if (!t[v])
                    c.push(v);
                temp++;
            }
        }
    }
    else
        cout << "Do thi khong lien thong.\nHuy lenh BFS!\n";
}
void _DFS()
{
    if (_dem_lien_thong() == 1) {
        int t[n + 1] = {0};
        DFS(1, t);
    }
    else
        cout << "Do thi khong lien thong.\nHuy lenh DFS!\n";
}
int _dem_lien_thong(){
    int t[n + 1] = {0}, res = 0;
    for (int i = 1; i < n; i++){
        iter<int> temp = a[i].dau().getnode()->getNext();
        if (!t[i] && temp != nullptr) {
            res++;
            _dfs(i, t);
            cout << endl;
        }
    }
    return res;
```

```
}
void con(ofstream &output){
    output << "\nDanh sach do thi con\n";
    cout << "\nDanh sach do thi con\n";
    int _t[n * n + 1] = {0}, t[n + 1] = {0}, j = 1;
    for (int i = 0; i < n; i++)
        if (!t[i])
            _bfs(i, t, _t, j, output);
}
void dijkstra(){
    if (_dem_lien_thong() == 1) {
        cout << "Nhap dinh dau va cuoi: ";
        int dau, cuoi;
        cin >> dau >> cuoi;
        vector<float> d(n, INT_MAX), res(n);
        d[dau] = 0;
        priority_queue<pair<float, int>, vector<pair<int, int>>,
                                                    greater<pair<int, int>>> q;
        q.push({0, dau});
        while (!q.empty()){
            pair<float, int> top = q.top();
            q.pop();
            int u = top.second;
            float kc = top.first;
            iter<int> temp = a[u].dau();
            while (temp != nullptr){
                int v = *temp;
                float w = trongso[u * n + v];
                if (d[v] > d[u] + w) {
                    res[v] = u;
                    d[v] = d[u] + w;
                    q.push({d[v], v});
```

```
                }
                temp++;
            }
        }
        cout << "Duong di ngan nhat tu dinh " << dau << " den dinh " << cuoi
                                                                << " la: ";
        vector<int> _res;
        _res.push_back(cuoi);
        int k = res[cuoi];
        while (k != dau) {
            _res.push_back(k);
            k = res[k];
        }
        _res.push_back(dau);
        for (int i = _res.size() - 1; i >= 0; i--)
            cout << _res[i] << " ";
        cout << "\nKhoang cach giua hai dinh la: " << d[cuoi] << endl;
    }
    else
        cout << "Do thi khong lien thong.\nHuy lenh tim kiem!\n";
  }
};
#endif
```

3. File main

```
#include "danhsachke.cpp"
using namespace std;

int main()

{

   ifstream input("input.txt");

   Graph a;

   input >> a;

   ofstream output("output.txt");

   output << a;

   a.dem_lien_thong();

   a.BFS();

   a.con(output);

   a.dijkstra();

   cout << "end";

}
```

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

1. Class node, iter, Don_list tương tự bài 1.
2. Class Graph

Phương thức Graph, add: O(1).

Phương thức Grap(int n, int _n, int m): O(n).

Phương thức copy: $O(n^2)$.

Phương thức toán tử nhập, _dfs, DFS, _DFS, _bfs, BFS, _dem_lien_thong,

con: O(m+n).

Phương thức toán tử xuất: O(m*n).

Phương thức dijksta: O((n+m)*log n).

Danh sách tài liệu tham khảo

1. Slide bài giảng Cấu trúc dữ liệu và Giải thuật.
2. Code mẫu trong bài giảng Cấu trúc dữ liệu và Giải thuật.
3. Diễn đàn tin học, thuật toán VNOI: https://vnoi.info/