**MỤC LỤC**

[**Bài thực hành số 4 - tuần 39** 3](#_Toc139492713)

[**Bài tập 1:** Đảo ngược một danh sách liên kết đơn 3](#_Toc139492714)

[**Bài tập 2:** Tính diện tích tam giác 7](#_Toc139492715)

[**Bài tập 3:** Tính tích có hướng của 2 vector 8](#_Toc139492716)

[**Bài tập 4:** Thao tác với vector 10](#_Toc139492717)

[**Bài tập 5:**Viết hàm thực hiện thuật toán DFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề vector< list<int> > . 14](#_Toc139492718)

[**Bài tập 6:**Viết hàm thực hiện thuật toán BFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề vector< list<int> > . 17](#_Toc139492719)

[**Bài tập 7:** Viết các hàm thực hiện các phép giao và hợp của hai tập hợp được biểu diễn bằng set. 19](#_Toc139492720)

[**Bài tập 8:**Viết các hàm thực hiện các phép giao và hợp của hai tập hợp mờ được biểu diễn bằng map. 21](#_Toc139492721)

[**Bài tập 9:**Cài đặt thuật toán Dijkstra trên đồ thị vô hướng được biểu diễn bằng danh sách kề sử dụng priority\_queue. 24](#_Toc139492722)

[**Bài tập 10:** Search Engine 27](#_Toc139492723)

[**Bài tập 11:** Bảo vệ lâu đài 31](#_Toc139492724)

[**Bài tập 12.** Lược đồ 35](#_Toc139492725)

[**Bài tập 13:** Đếm xâu con 36](#_Toc139492726)

**Mục lục hình ảnh**

[Hình ảnh 1: Code bài 4.1 3](#_Toc139492749)

[Hình ảnh 2: Kết quả bài 4.1 4](#_Toc139492750)

[Hình ảnh 3: Code bài 4.2 7](#_Toc139492751)

[Hình ảnh 4: Kết quả bài 4.2 7](#_Toc139492752)

[Hình ảnh 5: Code bài 4.3 8](#_Toc139492753)

[Hình ảnh 6: Kết quả bài 4.3 9](#_Toc139492754)

[Hình ảnh 7:Code bài 4.4 10](#_Toc139492755)

[Hình ảnh 8: Kết quả bài 4.4 11](#_Toc139492756)

[Hình ảnh 9: Code bài 4.5 14](#_Toc139492757)

[Hình ảnh 10: Kết quả bài 4.5 15](#_Toc139492758)

[Hình ảnh 11:Code bài 4.6 17](#_Toc139492759)

[Hình ảnh 12:Kết quả bài 4.6 17](#_Toc139492760)

[Hình ảnh 13: Code bài 4.7 19](#_Toc139492761)

[Hình ảnh 14: Kết quả bài 4.7 20](#_Toc139492762)

[Hình ảnh 15:Code bài 4.8 21](#_Toc139492763)

[Hình ảnh 16: Kết quả bài 4.8 22](#_Toc139492764)

[Hình ảnh 17: Code bài 4.9 24](#_Toc139492765)

[Hình ảnh 18: Kết quả bài 4.9 25](#_Toc139492766)

[Hình ảnh 19: Code bài 4.10 29](#_Toc139492767)

[Hình ảnh 20: Kết quả bài 4.10 31](#_Toc139492768)

[Hình ảnh 21: Code bài 4.11 32](#_Toc139492769)

[Hình ảnh 22: Kết quả bài 4.11 35](#_Toc139492770)

[Hình ảnh 23: Code bài 4.12 36](#_Toc139492771)

[Hình ảnh 24: Kết quả bài 4.12 36](#_Toc139492772)

[Hình ảnh 25: Code bài 4.13 37](#_Toc139492773)

[Hình ảnh 26: Kết quả bài 4.13 38](#_Toc139492774)

# **Bài thực hành số 4 - tuần 39**

## **Bài tập 1:** Đảo ngược một danh sách liên kết đơn

Ảnh có chứa văn bản, phần mềm, ảnh chụp màn hình, Biểu tượng máy tính

Mô tả được tạo tự động

Ảnh có chứa văn bản, phần mềm, Biểu tượng máy tính, Hệ điều hành

Mô tả được tạo tự động

Hình ảnh 1: Code bài 4.1

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Trang web

Mô tả được tạo tự động

Hình ảnh 2: Kết quả bài 4.1

//Vu Van Hao 20215572

/\*

Bài tập 1: Đảo ngược một danh sách liên kết đơn

\*/

#include <iostream>

using namespace std;

//cấu trúc của node trong danh sách liên kết

struct Node {

int data;

Node\* next;

Node(int data) { //constructor

this->data = data;

next = NULL;

}

};

// push a new element to the beginning of the list

Node\* prepend(Node\* head, int data) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Vu Van Hao 20215572

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Node\* tmp = new Node(data); //khởi tạo node với dữ liệu data

tmp->next = head; //cho next trỏ tới phàn tử đầu cũ

head = tmp; //chuyển head về phần tử đầu

return head;

}

// print the list content on a line

void print(Node\* head) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Vu Van Hao - 20215572

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

while(head != NULL){

cout << head->data << " ";

head = head->next;

}

cout<< endl;

}

// return the new head of the reversed list

Node\* reverse(Node\* head) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Vu Van Hao 20215572

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Node\* next = NULL;

Node\* prev = NULL;

while(head){

next = head->next; //luwulaij phần tử tiếp theo

head->next = prev; //đảo ngược lại

prev = head; //cho prev di chuyển đến phần tử hiện tại

head = next; //head chuyển đến phần tử tiếp theo

}

return prev;

}

int main() {

int n, u;

cin >> n; //số phần tử của linklist

Node\* head = NULL;

for (int i = 0; i < n; ++i){

//nhập vào linklist

cin >> u;

head = prepend(head, u);

}

cout << "Original list: ";

print(head);

head = reverse(head); //đảo ngược danh sách

cout << "Reversed list: ";

print(head);

return 0;

}

//Vu Van Hao 20215572

## **Bài tập 2:** Tính diện tích tam giác

Một điểm trong không gian 2 chiều được biểu diễn bằng pair. Hãy viết hàm double area(Point a, Point b, Point c) tính diện tích tam giác theo tọa độ 3 đỉnh. Trong đó, Point là kiểu được định nghĩa sẵn trong trình chấm như sau: using Point = pair<double, double>;

Ảnh có chứa văn bản, ảnh chụp màn hình, thiết kế

Mô tả được tạo tự động

Hình ảnh 3: Code bài 4.2

Ảnh có chứa văn bản, phần mềm, Hệ điều hành, Biểu tượng máy tính

Mô tả được tạo tự động

Hình ảnh 4: Kết quả bài 4.2

//Vu Van Hao 20215572

/\*

Bài tập 2: Tính diện tích tam giác

Một điểm trong không gian 2 chiều được biểu diễn bằng pair. Hãy viết hàm double area(Point a, Point b, Point c) tính diện tích tam giác theo tọa độ 3 đỉnh.

Trong đó, Point là kiểu được định nghĩa sẵn trong trình chấm như sau: using Point = pair<double, double>;

\*/

using Point = pair<double, double>;

double area(Point a, Point b, Point c) {

//S = 1/2|(xC-xA)(yB-yC) - (xB-xA)(yC-yA)|

return 1.0/2 \* fabs((c.first - a.first)\*(b.second - a.second) - (b.first - a.first)\*(c.second - a.second));

}

//Vu Van Hao 20215572

## **Bài tập 3:** Tính tích có hướng của 2 vector

Một vector trong không gian 3 chiều được biểu diễn bằng tuple<double, double, double>. Hãy viết hàm Vector cross\_product(Vector a, Vector b) tính tích có hướng của 2 vector. Trong đó Vector là kiểu dữ liệu được định nghĩa sẵn trong trình chấm như sau: using Vector = tuple<double, double, double>;

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Trang web

Mô tả được tạo tự động

Hình ảnh 5: Code bài 4.3

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động

Hình ảnh 6: Kết quả bài 4.3

//Vu Van Hao 20215572

/\*

Bài tập 3: Tính tích có hướng của 2 vector¶

Một vector trong không gian 3 chiều được biểu diễn bằng tuple<double, double, double>. Hãy viết hàm Vector cross\_product(Vector a, Vector b)

tính tích có hướng của 2 vector. Trong đó Vector là kiểu dữ liệu được định nghĩa sẵn trong trình chấm như sau: using Vector = tuple<double, double, double>;

\*/

using Vector = tuple<double, double, double>;

Vector cross\_product(Vector a, Vector b) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Vu Van Hao 20215572

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Vector rs;

get<0>(rs) = get<1>(a)\*get<2>(b) - get<1>(b)\*get<2>(a);

get<1>(rs) = get<2>(a)\*get<0>(b) - get<2>(b)\*get<0>(a);

get<2>(rs) = get<0>(a)\*get<1>(b) - get<0>(b)\*get<1>(a);

return rs;

}

//Vu Van Hao 20215572

## **Bài tập 4:** Thao tác với vector

Cho hai vector, hãy xóa hết các phần tử chẵn, sắp xếp giảm dần các số trong cả 2 vector và trộn lại thành một vector cũng được sắp xếp giảm dần.

Ảnh có chứa văn bản, phần mềm, Biểu tượng máy tính, Trang web

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Biểu tượng máy tính

Mô tả được tạo tự động

Hình ảnh 7:Code bài 4.4

Ảnh có chứa văn bản, phần mềm, Biểu tượng máy tính, Trang web

Mô tả được tạo tự động

Hình ảnh 8: Kết quả bài 4.4

//Vu Van Hao 20215572

/\*

Bài 4.4. Cho hai std::vector, hãy xóa hết các phần tử chẵn, sắp xếp giảm dần các số trong cả 2 vector và trộn lại thành một

vector cũng được sắp xếp giảm dần.

\*/

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

void print\_vector(const vector<int> &a) {

for (int v : a) cout << v << ' ';

cout << endl;

}

void delete\_even(vector<int> &a) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Vu Van Hao 20215572

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

for(unsigned int i = 0; i < a.size(); i++){

if(a[i]%2 == 0){

a.erase(a.begin() + i); //delete a[i] if a[i] is even

i--;

}

}

}

void sort\_decrease(vector<int> &a) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Vu Van Hao 20215572

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

std::sort(a.begin(), a.end(), [](int a, int b){

return a > b;

});

}

vector<int> merge\_vectors(const vector<int> &a, const vector<int> &b) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Vu Van Hao 20215572

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

vector<int> rs(a.size() + b.size());

std::merge(a.begin(), a.end(), b.begin(), b.end(), rs.begin(), [](int a, int b){

return a > b;

});

return rs;

}

int main() {

int m, n, u;

std::vector<int> a, b;

std::cin >> m >> n;

for(int i = 0; i < m; i++){

std:: cin >> u;

a.push\_back(u);

}

for(int i = 0; i < n; i++){

std:: cin >> u;

b.push\_back(u);

}

delete\_even(a);

cout << "Odd elements of a: ";

print\_vector(a);

delete\_even(b);

cout << "Odd elements of b: ";

print\_vector(b);

sort\_decrease(a);

cout << "Decreasingly sorted a: ";

print\_vector(a);

sort\_decrease(b);

cout << "Decreasingly sorted b: ";

print\_vector(b);

vector<int> c = merge\_vectors(a, b);

cout << "Decreasingly sorted c: ";

print\_vector(c);

return 0;

}

//Vu Van Hao 20215572

## **Bài tập 5:**Viết hàm thực hiện thuật toán DFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề vector< list<int> > .

Đồ thị có n đỉnh được đánh số từ 1 đến n. Thuật toán DFS xuất phát từ đỉnh 1. Các đỉnh được thăm theo thứ tự ưu tiên từ trái sang phải trong danh sách kề. Yêu cầu hàm trả ra thứ tự các đỉnh được thăm (những đỉnh không thể thăm từ đỉnh 1 thì không phải in ra)

Ảnh có chứa văn bản, phần mềm, Biểu tượng máy tính, Hệ điều hành

Mô tả được tạo tự động

Hình ảnh 9: Code bài 4.5

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, thiết kế

Mô tả được tạo tự động

Hình ảnh 10: Kết quả bài 4.5

//Vu Van Hao 20215572

/\*

Bài tập 5:

Viết hàm thực hiện thuật toán DFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề vector< list<int> >.

Đồ thị có n đỉnh được đánh số từ 1 đến n. Thuật toán DFS xuất phát từ đỉnh 1. Các đỉnh được thăm theo thứ tự

ưu tiên từ trái sang phải trong danh sách kề. Yêu cầu hàm trả ra thứ tự các đỉnh được thăm

(những đỉnh không thể thăm từ đỉnh 1 thì không phải in ra).

\*/

void dfs(vector< list<int> > adj) {

stack<int> S;

vector<bool> visited(adj.size());

S.push(1); // Bắt đầu từ đỉnh số 1

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Vu Van Hao 20215572

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

while(!S.empty()){

int v = S.top();

if( !visited[v] ){

//neu dinh v chua tham thi in ra v va danh dau v da duoc tham

visited[v] = true;

cout << v << endl;

}

if( !adj[v].empty() ){

int tmp = adj[v].front();

//lay ra dinh ke cua v

adj[v].pop\_front();

if(!visited[tmp]){ //neu dinh ke do chua dc tham thi push vao stack de xet(goi de quy tham dinh ke cua v)

S.push(tmp);

}

}else{ //v khong con dinh ke nua

S.pop(); //loai bo v ra khoi stack

}

}

}

//Vu Van Hao 20215572

## **Bài tập 6:**Viết hàm thực hiện thuật toán BFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề vector< list<int> > .

Đồ thị có n đỉnh được đánh số từ 1 đến n. Thuật toán BFS xuất phát từ đỉnh 1. Các đỉnh được thăm theo thứ tự ưu tiên từ trái sang phải trong danh sách kề. Yêu cầu hàm trả ra thứ tự các đỉnh được thăm (những đỉnh không thể thăm từ đỉnh 1 thì không phải in ra)

Ảnh có chứa văn bản, phần mềm, Trang web, Biểu tượng máy tính

Mô tả được tạo tự động

Hình ảnh 11:Code bài 4.6

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, thiết kế

Mô tả được tạo tự động

Hình ảnh 12:Kết quả bài 4.6

//Vu Van Hao 20215572

/\*

Bài tập 6:

Viết hàm thực hiện thuật toán BFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề vector< list<int> >.

Đồ thị có n đỉnh được đánh số từ 1 đến n. Thuật toán BFS xuất phát từ đỉnh 1. Các đỉnh được thăm theo thứ tự ưu tiên

từ trái sang phải trong danh sách kề. Yêu cầu hàm trả ra thứ tự các đỉnh được thăm

(những đỉnh không thể thăm từ đỉnh 1 thì không phải in ra).

\*/

void bfs(vector< list<int> > adj) {

queue<int> Q; //hang doi cac dinh can duoc tham

vector<bool> visited(adj.size()); //vector danh dau xem dinh do duoc day vao hang doi chua

Q.push(1); // Bắt đầu từ đỉnh số 1

visited[1] = true; //danh dau dinh 1 da duoc day vao hang doi de cho duoc tham

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Vu Van Hao 20215572

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

while(!Q.empty()){

int v = Q.front(); //lay ra dinh dau tien cua hang doi

cout << v << endl;

Q.pop(); //tham dinh do va loai bo ra khoi hang doi

if(!adj[v].empty()){ //kiem tra xem v co canh ke khong

for(int u : adj[v]){

//duyet cac dinh ke voi v

if(!visited[u]){

//neu dinh nay chua duoc day vao queue thi day vao queue va danh dau da duoc day vao queue roi

Q.push(u);

visited[u] = true;

}

}

}

}

}

//Vu Van Hao 20215572

## **Bài tập 7:** Viết các hàm thực hiện các phép giao và hợp của hai tập hợp được biểu diễn bằng set.

Ảnh có chứa văn bản, phần mềm, Biểu tượng máy tính, Trang web

Mô tả được tạo tự động

Hình ảnh 13: Code bài 4.7

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Biểu tượng máy tính

Mô tả được tạo tự động

Hình ảnh 14: Kết quả bài 4.7

//Vu Van Hao 20215572

/\*

Bài tập 7: Viết các hàm thực hiện các phép giao và hợp của hai tập hợp được biểu diễn bằng set.

\*/

template<class T>

set<T> set\_union(const std::set<T> &a, const std::set<T> &b) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Vu Van Hao 20215572

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

set<T> rs = a; //copy set a vao set result

rs.insert(b.begin(), b.end()); //copy set b vao set result

return rs;

}

template<class T>

set<T> set\_intersection(const std::set<T> &a, const std::set<T> &b) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Vu Van Hao 20215572

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

set<T> rs;

for(auto x : a){

if(b.count(x)) //neu trong set b co phan tu x trong set a thi insert x vao set rs

rs.insert(x);

}

return rs;

}

//Vu Van Hao 20215572

## **Bài tập 8:**Viết các hàm thực hiện các phép giao và hợp của hai tập hợp mờ được biểu diễn bằng map.

Trong đó mỗi phần tử được gán cho một số thực trong đoạn [0..1] biểu thị độ thuộc của phần tử trong tập hợp, với độ thuộc bằng 1 nghĩa là phần tử chắc chắn thuộc vào tập hợp và ngược lại độ thuộc bằng 0 nghĩa là phần tử chắc chắn không thuộc trong tập hợp.

Phép giao và hợp của 2 tập hợp được thực hiện trên các cặp phần tử bằng nhau của 2 tập hợp, với độ thuộc mới được tính bằng phép toán min và max của hai độ thuộc.

Ảnh có chứa văn bản, phần mềm, Trang web, Website

Mô tả được tạo tự động

Hình ảnh 15:Code bài 4.8

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Trang web

Mô tả được tạo tự động

Hình ảnh 16: Kết quả bài 4.8

//Vu Van Hao 20215572

/\*

Bài tập 8:

Viết các hàm thực hiện các phép giao và hợp của hai tập hợp mờ được biểu diễn bằng map.

Trong đó mỗi phần tử được gán cho một số thực trong đoạn [0..1] biểu thị độ thuộc của phần tử trong tập hợp,

với độ thuộc bằng 1 nghĩa là phần tử chắc chắn thuộc vào tập hợp và ngược lại độ thuộc bằng 0 nghĩa là phần tử

chắc chắn không thuộc trong tập hợp.

Phép giao và hợp của 2 tập hợp được thực hiện trên các cặp phần tử bằng nhau của 2 tập hợp,

với độ thuộc mới được tính bằng phép toán min và max của hai độ thuộc.

\*/

template<class T>

map<T, double> fuzzy\_set\_union(const map<T, double> &a, const map<T, double> &b){

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Vu Van Hao 20215572

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

map<T, double> rs = a; //coppy cac phan tu trong map a vao map result

for(const auto x : b){

if(rs.count(x.first)){ /\*kiem tra tung phan tu trong map b

neu phan tu do co key da xuat hien trong result thi cap nhat do thuoc\*/

rs[x.first] = rs[x.first] < x.second ? x.second : rs[x.first]; //voi hop thi do thuoc la max

}else{

//neu phan tu chua xuat hien trong result thi insert vao result

rs.insert(x);

}

}

return rs;

}

template<class T>

map<T, double> fuzzy\_set\_intersection(const map<T, double> &a, const map<T, double> &b){

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Vu Van Hao 20215572

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

map<T, double> rs;

for(const auto x : a){ /\*duyet qua cac phan tu trong map a neu phan tu do co key cung xuat hien trong map b

thi them phan tu do vao trong result voi do thuoc min\*/

if(b.count(x.first))

rs[x.first] = x.second < (b.find(x.first)) -> second ? x.second : (b.find(x.first)) -> second;

//khong dung dc b[x.first] do b khai bao sonst ma [] khong phai phuong thuc const

//can su dung find tra ve iterator tro den phan tu do

}

return rs;

}

//Vu Van Hao 20215572

## **Bài tập 9:**Cài đặt thuật toán Dijkstra trên đồ thị vô hướng được biểu diễn bằng danh sách kề sử dụng priority\_queue.

Cụ thể, bạn cần cài đặt hàm vector<int> dijkstra(const vector< vector< pair<int, int> > >&adj) nhận đầu vào là danh sách kề chứa các cặp pair<int, int> biểu diễn đỉnh kề và trọng số tương ứng của cạnh. Đồ thị gồm n đỉnh được đánh số từ 0 tới n-1. Hàm cần trả vector<int> chứa n phần tử lần lượt là khoảng cách đường đi ngắn nhất từ đỉnh 0 tới các đỉnh 0, 1, 2, ..., n-1.

Ảnh có chứa văn bản, phần mềm, Hệ điều hành, máy tính

Mô tả được tạo tự động

Hình ảnh 17: Code bài 4.9

Ảnh có chứa văn bản, phần mềm, máy tính, Hệ điều hành

Mô tả được tạo tự độngẢnh có chứa văn bản, phần mềm, ảnh chụp màn hình, thiết kế

Mô tả được tạo tự động

Hình ảnh 18: Kết quả bài 4.9

//Vu Van Hao 20215572

/\*

Bài tập 9:

Cài đặt thuật toán Dijkstra trên đồ thị vô hướng được biểu diễn bằng danh sách kề sử dụng priority\_queue Cụ thể,

bạn cần cài đặt hàm vector<int> dijkstra(const vector< vector< pair<int, int> > >&adj) nhận đầu vào là danh

dách kề chứa các cặp pair<int, int> biểu diễn đỉnh kề và trọng số tương ứng của cạnh.

Đồ thị gồm n đỉnh được đánh số từ 0 tới n-1. Hàm cần trả vector<int> chứa n phần tử lần lượt là khoảng cách

đường đi ngắn nhất từ đỉnh 0 tới các đỉnh 0, 1, 2, ..., n-1

\*/

vector<int> dijkstra(const vector< vector< pair<int, int> > >&adj) { //adj[u] : danh sach dinh ke voi dinh u

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Vu Van Hao 20215572

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//mang luu do dai duong di tu dinhh 0 den cac dinh khac

vector<int> distance(adj.size(), 1e9); //ban dau do dai duong di la infinitive

priority\_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> Q; //greater mac dinh se so sanh gia tri cua first trong hai pair

distance[0]= 0;

Q.push({0, 0});

while(!Q.empty()){

pair<int, int> top = Q.top(); //lay ra dinh co do dai duong di nho nhat trong cac dinh chua tham

Q.pop();

int u = top.second; //u la dinh dang set

int du = top.first; //du la do dai duong di tu dinh 0 -> u hien tai

if(du > distance[u]) continue; //neu distance[u] nho hon co nghia no co duong di ngan hon tu dinh khac va duoc scap nhat roi

for(pair<int, int> adj\_u : adj[u]){ //xet cac dinh ke voi dinh u

//adj\_u la dinh ke voi dinh u dang xet

int v = adj\_u.first; //v dinh ke voi u

int w = adj\_u.second; //trong so canh tu u->v

//cap nhhat lai do dai duong di toi v

if(distance[v] > (du + w)){ //neu duong di den v tu u nho hon thi update duong toi v di tu u

distance[v] = du + w;

Q.push({distance[v], v}); //push v vao hang doi de xet cac lai v sau do

}

}

}

return distance; //tra ve mang do dai duong di toi cac dinh tu dinh 0

}

//Vu Van Hao 20215572

## **Bài tập 10:** Search Engine

Xây dựng một máy tìm kiếm (search engine) đơn giản.Cho N văn bản và Q truy vấn. Với mỗi truy vấn, cần trả về văn bản khớp với truy vấn đó nhất. Sử dụng phương pháp tính điểm TF-IDF:

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Mô tả được tạo tự động

Ảnh có chứa ảnh chụp màn hình, văn bản, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

Ảnh có chứa ảnh chụp màn hình, văn bản, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

Ảnh có chứa ảnh chụp màn hình, văn bản, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

Ảnh có chứa văn bản, Phần mềm đa phương tiện, phần mềm, Phần mềm đồ họa

Mô tả được tạo tự động

Hình ảnh 19: Code bài 4.10

Ảnh có chứa ảnh chụp màn hình, văn bản, Phần mềm đa phương tiện, phần mềm

Mô tả được tạo tự động

Ảnh có chứa ảnh chụp màn hình, Phần mềm đa phương tiện, phần mềm, Phần mềm đồ họa

Mô tả được tạo tự động

Ảnh có chứa ảnh chụp màn hình, văn bản, Phần mềm đa phương tiện, phần mềm

Mô tả được tạo tự động

*Test Case 1*

Hình ảnh 20: Kết quả bài 4.10

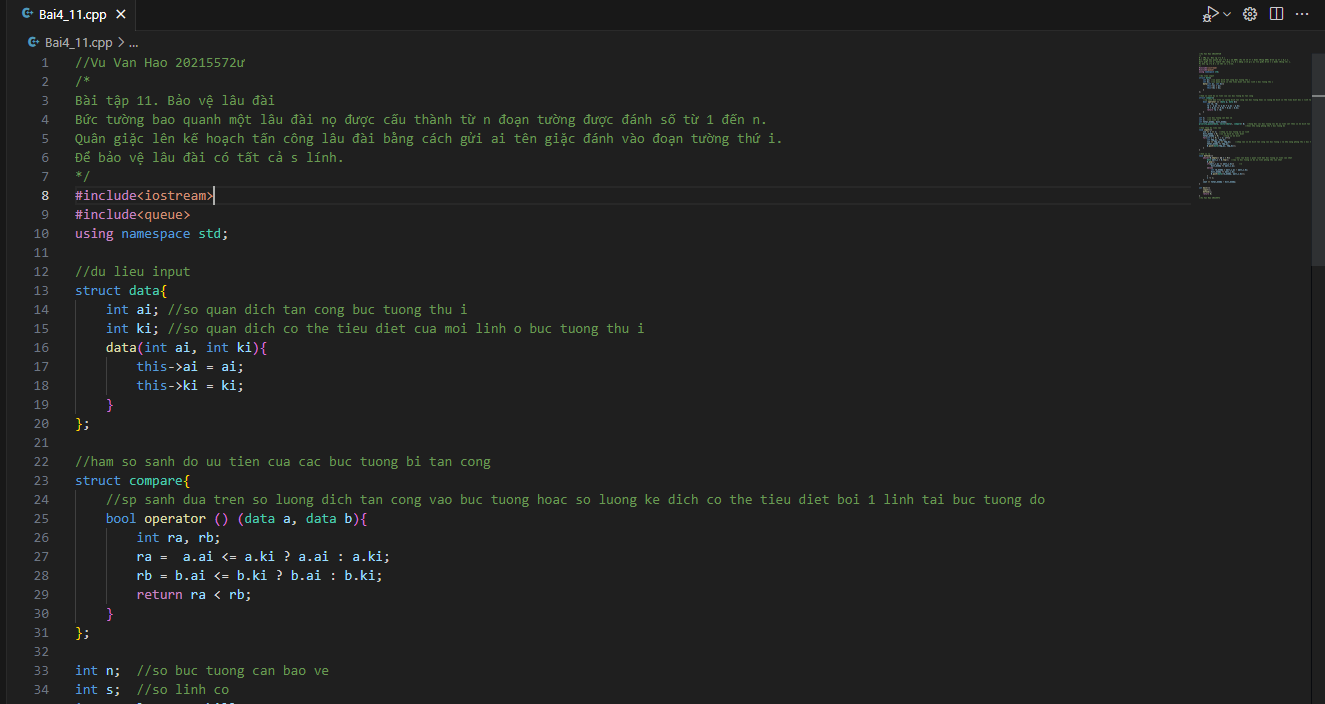
## **Bài tập 11:** Bảo vệ lâu đài

Bức tường bao quanh một lâu đài nọ được cấu thành từ n đoạn tường được đánh số từ 1 đến n. Quân giặc lên kế hoạch tấn công lâu đài bằng cách gửi ai tên giặc đánh vào đoạn tường thứ i. Để bảo vệ lâu đài có tất cả s lính.

Do các đoạn tường có chất lượng khác nhau nên khả năng bảo vệ tại các đoạn tường cũng khác nhau. Cụ thể tại đoạn tường thứ i, mỗi lính có thể đẩy lùi tấn công của ki tên giặc.

Giả sử đoạn tường thứ i có xi lính. Khi đó nếu số tên giặc không vượt quá xi×ki thì không có tên giặc nào lọt vào được qua đoạn tường này. Ngược lại sẽ có ai−xi×ki tên giặc lọt vào lâu đài qua đoạn tường này.

Yêu cầu hãy viết chương trình phân bố lính đứng ở các đoạn tường sao cho tổng số lính là s và tổng số lượng tên giặc lọt vào lâu đài là nhỏ nhất.



Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Mô tả được tạo tự động

Ảnh có chứa Phần mềm đa phương tiện, phần mềm, Phần mềm đồ họa, ảnh chụp màn hình

Mô tả được tạo tự động

Hình ảnh 21: Code bài 4.11

Ảnh có chứa ảnh chụp màn hình, văn bản, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

*Test case 1*

Ảnh có chứa ảnh chụp màn hình, văn bản, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

*Test case 2*

Ảnh có chứa ảnh chụp màn hình, văn bản

Mô tả được tạo tự động

*Test case 3*

Ảnh có chứa ảnh chụp màn hình, văn bản

Mô tả được tạo tự động

*Test case 4*

Ảnh có chứa ảnh chụp màn hình, văn bản

Mô tả được tạo tự động

*Test case 5*

Hình ảnh 22: Kết quả bài 4.11

## **Bài tập 12.** Lược đồ

Cho một lược đồ gồm n cột chữ nhật liên tiếp nhau có chiều rộng bằng 1 và chiều cao lần lượt là các số nguyên không âm h1,h2,…,hnℎ1,ℎ2,…,ℎ. Hãy xác định hình chữ nhật có diện tích lớn nhất có thể tạo thành từ các cột liên tiếp.

Ảnh có chứa ảnh chụp màn hình, văn bản, phần mềm, màn hình

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Mô tả được tạo tự động

Hình ảnh 23: Code bài 4.12

Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động

Hình ảnh 24: Kết quả bài 4.12

## **Bài tập 13:** Đếm xâu con

Cho một xâu nhị phân độ dài n. Hãy viết chương trình đếm số lượng xâu con chứa số ký tự 0 và số ký tự 1 bằng nhau.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Mô tả được tạo tự động

Hình ảnh 25: Code bài 4.13

Ảnh có chứa văn bản, phần mềm, Phần mềm đa phương tiện, ảnh chụp màn hình

Mô tả được tạo tự động

*Test case 1*

Ảnh có chứa văn bản, phần mềm, Phông chữ, ảnh chụp màn hình

Mô tả được tạo tự động

*Test case 2*

Hình ảnh 26: Kết quả bài 4.13