**BÁO CÁO BUỔI THỰC HÀNH SỐ 4**

Bộ Môn Kỹ Thuật Lập Trình

A blue logo with black background

Description automatically generated

A red and white logo

Description automatically generated

**Sinh viên:** Nguyễn Duy Khánh

**Mã số sinh viên:** 20225019

**Giảng viên hướng dẫn:** ThS. Lê Thị Hoa

**HTGD:** Đoàn Sỹ Nguyên

**MỤC LỤC**

[**DANH MỤC HÌNH ẢNH** 4](#_Toc185390590)

[**Bài thực hành số 4 – Tuần 14** 5](#_Toc185390591)

[**Bài 4.1:**Đảo ngược một danh sách liên kết đơn 5](#_Toc185390592)

[**Bài 4.2:** Một điểm trong không gian 2 chiều được biểu diễn bằng pair. Hãy viết hàm tính diện tích tam giác theo tọa độ 3 đỉnh. 8](#_Toc185390593)

[**Bài 4.3**: Một vector trong không gian 3 chiều được biểu diễn bằng tuple<double, double, double>. Hãy viết hàm tính tích có hướng của 2 vector. 10](#_Toc185390594)

[**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. 11](#_Toc185390595)

[**Bài 4.5.** Viết hàm void dfs(vector< list<int> > adj) thực hiện thuật toán DFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề. 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). 15](#_Toc185390596)

[**Bài 4.6**. đồ thị biểu diễn bằng danh sách kề. Đồ thị có n đỉnh được đánh số từ 1 đến n. Thuật 6. Viết hàm void bfs(vector< list<int> > adj) thực hiện thuật toán BFS không sử dụng đệ quy trên 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). 19](#_Toc185390597)

[**Bài 4.7**. Viết các hàm thực hiện các phép giao và hợp của hai tập hợp được biểu diễn bằng set 22](#_Toc185390598)

[**Bài 4.8.** Viết các hàm thực hiện các phép giao và hợp của hai tập hợp mờ được biểu diễn bằng map. 26](#_Toc185390599)

[**Bài 4.9**. Cài đặt thuật toán Dijkstra trên đồ thị vô hướng được biểu diễn bằng danh sách kề sử dụng std::priority\_queue 29](#_Toc185390600)

[**Bài 4.10.** Xây dựng một máy tìm kiếm (search engine) đơn giản. 33](#_Toc185390601)

[**Bài 4.11.** Bảo vệ lâu đài 43](#_Toc185390602)

[**Bài 4.12.** 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,…,hn . Hãy xác định hình chữ nhật có diện tích lớn nhất có thể tạo thành từ các cột liên tiếp. 48](#_Toc185390603)

[**Bài 4.13**. Đếm xâu con 52](#_Toc185390604)

# **DANH MỤC HÌNH ẢNH**

[Hình 1.1. Code bài 1 – Ảnh 1 5](#_Toc185386131)

[Hình 1.2. Code bài 1 – Ảnh 2 6](#_Toc185386132)

[Hình 1.3. Testcase bài 1 6](#_Toc185386133)

[Hình 2.1. Code bài 2 9](#_Toc185386134)

[Hình 2.2. Testcase bài 2 9](#_Toc185386135)

[Hình 3.1. Code bài 3 11](#_Toc185386136)

[Hình 3.2. Code bài 3 11](#_Toc185386137)

[Hình 4.1. Code bài 4 - Ảnh 1 12](#_Toc185386138)

[Hình 4.2. Code bài 4 - Ảnh 2 12](#_Toc185386139)

[Hình 4.3. Code bài 4 - Ảnh 3 13](#_Toc185386140)

[Hình 4.4. Testcase bài 4 13](#_Toc185386141)

[Hình 5.1. Code bài 5 - Ảnh 1 16](#_Toc185386142)

[Hình 5.2. Code bài 5 - Ảnh 2 17](#_Toc185386143)

[Hình 5.3. Testcase bài 5 17](#_Toc185386144)

[Hình 6.1. Code bài 6 - Ảnh 1 20](#_Toc185386145)

[Hình 6.2. Code bài 6 - Ảnh 2 20](#_Toc185386146)

[Hình 6.3. Testcase bài 6 21](#_Toc185386147)

[Hình 7.1. Code bài 7 - Ảnh 1 23](#_Toc185386148)

[Hình 7.2. Code bài 7 - Ảnh 2 24](#_Toc185386149)

[Hình 7.3. Testcase bài 7 24](#_Toc185386150)

[Hình 8.1. Code bài 8 - Ảnh 1 26](#_Toc185386151)

[Hình 8.2. Code bài 8 - Ảnh 2 27](#_Toc185386152)

[Hình 8.3. Testcase bài 8 27](#_Toc185386153)

[Hình 9.1. Code bài 9 - Ảnh 1 30](#_Toc185386154)

[Hình 9.2. Code bài 9 - Ảnh 2 30](#_Toc185386155)

[Hình 9.3. Testcase bài 9 - Ảnh 1 31](#_Toc185386156)

[Hình 9.4. Testcase bài 9 - Ảnh 2 31](#_Toc185386157)

[Hình 10.1. Code bài 10 – Ảnh 1 34](#_Toc185386158)

[Hình 10.2. Code bài 10 – Ảnh 2 34](#_Toc185386159)

[Hình 10.3. Code bài 10 – Ảnh 3 35](#_Toc185386160)

[Hình 10.4. Code bài 10 – Ảnh 4 35](#_Toc185386161)

[Hình 10.5. Testcase 1 – Bài 10 36](#_Toc185386162)

[Hình 10.6. Testcase 2 – Bài 10 36](#_Toc185386163)

[Hình 10.7. Testcase 3 – Bài 10 37](#_Toc185386164)

[Hình 10.8. Testcase 4 – Bài 10 37](#_Toc185386165)

[Hình 10.9. Testcase 5 – Bài 10 38](#_Toc185386166)

# **Bài thực hành số 4 – Tuần 14**

## **Bài 4.1:**Đảo ngược một danh sách liên kết đơn

A screenshot of a computer program

Description automatically generated

Hình 1.1. Code bài 1 – Ảnh 1

A screenshot of a computer

Description automatically generated

Hình 1.2. Code bài 1 – Ảnh 2

A screenshot of a computer

Description automatically generated

Hình 1.3. Testcase bài 1

/\*

Bai 4.1 - Tuan 15

Nguyen Duy Khanh - 20225019 - 744469 - 20241

\*/

#include<iostream>

using namespace std;

struct Node {

int data;

Node \*next;

Node(int data){

this->data = data;

next = NULL;

}

};

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

Node \*newNode = new Node(data);

newNode->next = head;

head = newNode;

return head;

}

void print(Node \*head){

Node \*tmp = head;

while (tmp != NULL){

cout << tmp->data << " ";

tmp = tmp->next;

}

cout << endl;

}

Node \*reverse(Node \*head){

Node \*prev = NULL;

Node \*current = head;

Node \*next = NULL;

while (current != NULL){

next = current->next;

current->next = prev;

prev = current;

current = next;

}

head = prev;

return head;

}

int main(){

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

Nguyen Duy Khanh - 20225019

Ma lop TH: 744469

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

int n, u;

cin >> n;

Node \*head = NULL;

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

cin >> u;

head = prepend(head, u);

}

cout << "Original list: ";

print(head);

head = reverse(head);

cout << "Reversed list: ";

print(head);

return 0;

}

## **Bài 4.2:** Một điểm trong không gian 2 chiều được biểu diễn bằng pair. Hãy viết hàm tính diện tích tam giác theo tọa độ 3 đỉnh.

A screenshot of a computer program

Description automatically generated

Hình 2.1. Code bài 2

A screenshot of a computer

Description automatically generated

Hình 2.2. Testcase bài 2

/\*

Bai 4.2 - Tuan 15

Nguyen Duy Khanh - 20225019 - 744469 - 20241

\*/

#include<iostream>

#include<vector>

#include<cmath>

#include<iomanip>

#include<utility>

using namespace std;

using Point = pair<double, double>;

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

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

}

int main(){

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

Nguyen Duy Khanh - 20225019

Ma lop TH: 744469

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

Point p1 = {1, 2};

Point p2 = {2.5, 10};

Point p3 = {15, -5.25};

cout << setprecision(2) << fixed;

cout << area(p1, p2, p3) << endl;

return 0;

}

## **Bài 4.3**: Một vector trong không gian 3 chiều được biểu diễn bằng tuple<double, double, double>. Hãy viết hàm tính tích có hướng của 2 vector.

A screenshot of a computer

Description automatically generated

Hình 3.1. Code bài 3

A screenshot of a computer

Description automatically generated

Hình 3.2. Code bài 3

## **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.

A screenshot of a computer program

Description automatically generated

Hình 4.1. Code bài 4 - Ảnh 1

A screenshot of a computer program

Description automatically generated

Hình 4.2. Code bài 4 - Ảnh 2

A screenshot of a computer

Description automatically generated

Hình 4.3. Code bài 4 - Ảnh 3

A screenshot of a computer

Description automatically generated

Hình 4.4. Testcase bài 4

/\*

Bai 4.4 - Tuan 15

Nguyen Duy Khanh - 20225019 - 744469 - 20241

\*/

#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 ){

a.erase(remove\_if(a.begin(), a.end(), [](int x){ return x % 2 == 0; }), a.end());

}

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

sort(a.begin(), a.end(), greater<int>());

}

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

vector<int> c;

merge(a.begin(), a.end(), b.begin(), b.end(), back\_inserter(c), greater<int>());

return c;

}

bool cmp(int a, int b){

return a > b;

}

int main(){

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

Nguyen Duy Khanh - 20225019

Ma lop TH: 744469

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

int n, m, u;

vector<int> a, b;

cin >> m >> n;

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

cin >> u;

a.push\_back(u);

}

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

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;

}

## **Bài 4.5.** Viết hàm void dfs(vector< list<int> > adj) thực hiện thuật toán DFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề. 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).

A screenshot of a computer program

Description automatically generated

Hình 5.1. Code bài 5 - Ảnh 1

A screenshot of a computer code

Description automatically generated

Hình 5.2. Code bài 5 - Ảnh 2

A screenshot of a computer

Description automatically generated

Hình 5.3. Testcase bài 5

/\*

Bai 4.5 - Tuan 15

Nguyen Duy Khanh - 20225019 - 744469 - 20241

\*/

#include<iostream>

#include<vector>

#include<stack>

#include<list>

#include<algorithm>

using namespace std;

vector<list<int>> adj;

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

stack<int> S;

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

S.push(1);

while (!S.empty()){

int u = S.top();

if (!visited[u]){

visited[u] = true;

cout << u << endl;

}

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

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

adj[u].pop\_front();

if (!visited[v]){

S.push(v);

}

} else {

S.pop();

}

}

}

int main(int argc, const char \*\*argv){

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

Nguyen Duy Khanh - 20225019

Ma lop TH: 744469

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

int n = 7;

vector<list<int>> adj;

adj.resize(n + 1);

adj[1].push\_back(2);

adj[2].push\_back(4);

adj[1].push\_back(3);

adj[3].push\_back(4);

adj[3].push\_back(5);

adj[5].push\_back(2);

adj[2].push\_back(7);

adj[6].push\_back(7);

dfs(adj);

return 0;

}

## **Bài 4.6**. đồ thị biểu diễn bằng danh sách kề. Đồ thị có n đỉnh được đánh số từ 1 đến n. Thuật 6. Viết hàm void bfs(vector< list<int> > adj) thực hiện thuật toán BFS không sử dụng đệ quy trên 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).

A screenshot of a computer

Description automatically generated

Hình 6.1. Code bài 6 - Ảnh 1

A screenshot of a computer program

Description automatically generated

Hình 6.2. Code bài 6 - Ảnh 2

A screenshot of a computer

Description automatically generated

Hình 6.3. Testcase bài 6

/\*

Bai 4.6 - Tuan 15

Nguyen Duy Khanh - 20225019 - 744469 - 20241

\*/

#include<iostream>

#include<stack>

#include<queue>

#include<list>

#include<vector>

#include<algorithm>

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

queue<int> Q;

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

Q.push(1);

while (!Q.empty()){

int u = Q.front();

if (!visited[u]){

visited[u] = true;

cout << u << endl;

}

if (adj[u].size()){

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

adj[u].pop\_front();

if (!visited[v]){

Q.push(v);

}

} else {

Q.pop();

}

}

}

int main(int argc, const char \*\*argv){

int n = 7;

vector<list<int>> adj;

adj.resize(n + 1);

adj[1].push\_back(2);

adj[2].push\_back(4);

adj[1].push\_back(3);

adj[3].push\_back(4);

adj[3].push\_back(5);

adj[5].push\_back(2);

adj[2].push\_back(7);

adj[6].push\_back(7);

bfs(adj);

return 0;

}

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

Nguyen Duy Khanh - 20225019

Ma lop TH: 744469

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

## **Bài 4.7**. Viết các hàm thực hiện các phép giao và hợp của hai tập hợp được biểu diễn bằng set

A screenshot of a computer program

Description automatically generated

Hình 7.1. Code bài 7 - Ảnh 1

A screenshot of a computer

Description automatically generated

Hình 7.2. Code bài 7 - Ảnh 2

A screenshot of a computer program

Description automatically generated

Hình 7.3. Testcase bài 7

/\*

Bai 4.7 - Tuan 15

Nguyen Duy Khanh - 20225019 - 744469 - 20241

\*/

#include <iostream>

#include <set>

using namespace std;

template<class T>

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

set<T> c;

for (const T &x : a) {

c.insert(x);

}

for (const T &x : b) {

c.insert(x);

}

return c;

}

template<class T>

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

set<T> c;

for (const T &x : a) {

if (b.find(x) != b.end()) {

c.insert(x);

}

}

return c;

}

template<class T>

void print\_set(const std::set<T> &a) {

for (const T &x : a) {

std::cout << x << ' ';

}

std::cout << std::endl;

}

int main() {

std::set<int> a = {1, 2, 3, 5, 7};

std::set<int> b = {2, 4, 5, 6, 9};

std::set<int> c = set\_union(a, b);

std::set<int> d = set\_intersection(a, b);

std::cout << "Union: "; print\_set(c);

std::cout << "Intersection: "; print\_set(d);

return 0;

}

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

Nguyen Duy Khanh - 20225019

Ma lop TH: 744469

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

## **Bài 4.8.** Viết các hàm thực hiện các phép giao và hợp của hai tập hợp mờ được biểu diễn bằng map.

A screenshot of a computer code

Description automatically generated

Hình 8.1. Code bài 8 - Ảnh 1

A screenshot of a computer code

Description automatically generated

Hình 8.2. Code bài 8 - Ảnh 2

A screenshot of a computer program

Description automatically generated

Hình 8.3. Testcase bài 8

/\*

Bai 4.8 - Tuan 15

Nguyen Duy Khanh - 20225019 - 744469 - 20241

\*/

#include <iostream>

#include <map>

using namespace std;

template<class T>

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

map<T, double> result;

for (const auto &x : a) {

result[x.first] = x.second;

}

for (const auto &x : b) {

if (result.find(x.first) == result.end()) {

result[x.first] = x.second;

} else {

result[x.first] = std::max(x.second, result.at(x.first));

}

}

return result;

}

template<class T>

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

map<T, double> result;

for (const auto &x : a) {

if (b.find(x.first) != b.end()) {

result[x.first] = std::min(x.second, b.at(x.first));

}

}

return result;

}

template<class T>

void print\_fuzzy\_set(const std::map<T, double> &a) {

cout << "{ ";

for (const auto &x : a) {

std::cout << "(" << x.first << ", " << x.second << ") ";

}

cout << "}";

std::cout << std::endl;

}

int main() {

std::map<int, double> a = {{1, 0.2}, {2, 0.5}, {3, 1}, {4, 0.6}, {5, 0.7}};

std::map<int, double> b = {{1, 0.5}, {2, 0.4}, {4, 0.9}, {5, 0.4}, {6, 1}};

std::cout << "A = "; print\_fuzzy\_set(a);

std::cout << "B = "; print\_fuzzy\_set(b);

std::map<int, double> c = fuzzy\_set\_union(a, b);

std::map<int, double> d = fuzzy\_set\_intersection(a, b);

std::cout << "Union: "; print\_fuzzy\_set(c);

std::cout << "Intersection: "; print\_fuzzy\_set(d);

}

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

Nguyen Duy Khanh - 20225019

Ma lop TH: 744469

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

## **Bài 4.9**. Cài đặt thuật toán Dijkstra trên đồ thị vô hướng được biểu diễn bằng danh sách kề sử dụng std::priority\_queue

A screenshot of a computer program

Description automatically generated

Hình 9.1. Code bài 9 - Ảnh 1

A screenshot of a computer program

Description automatically generated

Hình 9.2. Code bài 9 - Ảnh 2

A screenshot of a computer program

Description automatically generated

Hình 9.3. Testcase bài 9 - Ảnh 1

A screenshot of a computer program

Description automatically generated

Hình 9.4. Testcase bài 9 - Ảnh 2

/\*

Bai 4.9 - Tuan 15

Nguyen Duy Khanh - 20225019 - 744469 - 20241

\*/

#include <iostream>

#include <queue>

#include <climits>

using namespace std;

const int INF = 1e9;

vector<int> dijkstra(const vector< vector< pair<int, int> > >&adj) {

int n = adj.size();

vector<int> distance(n, INF);

distance[0] = 0;

priority\_queue< pair<int, int>, vector< pair<int, int> >, greater< pair<int, int> > > pq;

pq.push({0, 0});

while (!pq.empty()) {

int u = pq.top().second;

int du = pq.top().first;

pq.pop();

if (du != distance[u]) continue;

for (const auto &edge : adj[u]) {

int v = edge.first;

int uv = edge.second;

if (distance[v] > du + uv) {

distance[v] = du + uv;

pq.push({distance[v], v});

}

}

}

return distance;

}

int main() {

int n = 9;

vector< vector< pair<int, int> > > adj(n);

auto add\_edge = [&adj] (int u, int v, int w) {

adj[u].push\_back({v, w});

adj[v].push\_back({u, w});

};

add\_edge(0, 1, 4);

add\_edge(0, 7, 8);

add\_edge(1, 7, 11);

add\_edge(1, 2, 8);

add\_edge(2, 3, 7);

add\_edge(2, 8, 2);

add\_edge(3, 4, 9);

add\_edge(3, 5, 14);

add\_edge(4, 5, 10);

add\_edge(5, 6, 2);

add\_edge(6, 7, 1);

add\_edge(6, 8, 6);

add\_edge(7, 8, 7);

vector<int> distance = dijkstra(adj);

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

cout << "distance " << 0 << "->" << i << " = " << distance[i] << endl;

}

return 0;

}

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

Nguyen Duy Khanh - 20225019

Ma lop TH: 744469

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

**BÀI TẬP VỀ NHÀ**

## **Bài 4.10.** Xây dựng một máy tìm kiếm (search engine) đơn giản.

A screenshot of a computer

Description automatically generated

Hình 10.1. Code bài 10 – Ảnh 1

A white background with black and red text

Description automatically generated

Hình 10.2. Code bài 10 – Ảnh 2

A screenshot of a computer

Description automatically generated

Hình 10.3. Code bài 10 – Ảnh 3

A screenshot of a computer program

Description automatically generated

Hình 10.4. Code bài 10 – Ảnh 4

A screenshot of a computer

Description automatically generated

Hình 10.5. Testcase 1 – Bài 10

A screenshot of a computer

Description automatically generated

Hình 10.6. Testcase 2 – Bài 10

A computer screen shot of a black screen

Description automatically generated

Hình 10.7. Testcase 3 – Bài 10

A computer screen with a black screen

Description automatically generated

Hình 10.8. Testcase 4 – Bài 10

A computer screen shot of a black screen

Description automatically generated

Hình 10.9. Testcase 5 – Bài 10

/\*

Bai 4.10 - Tuan 15

Nguyen Duy Khanh - 20225019 - 744469 - 20241

\*/

#include<bits/stdc++.h>

using namespace std;

int n, q;

vector< vector<string> > vector\_train;

vector< vector<string> > vector\_test;

vector<int> f\_max; // tan suat xuat hien tu xuat hien nhieu nhat trong van ban i

map<string, int> df; // tinh xem tu word nam trong bao nhieu van ban

map<pair<string, int>, int> fe; // tinh xem tu word xuat hien bao nhieu lan trong van ban i <word, i> = int

vector<string> split\_string(string str){

vector<string> vt;

while (!str.empty()){

string tmp = str.substr(0, str.find(","));

int pos = tmp.find(" ");

if(pos > tmp.size())

vt.push\_back(tmp);

else {

while(pos <= tmp.size()){

tmp.erase(pos,1);

pos = tmp.find(" ");

}

vt.push\_back(tmp);

}

if (str.find(",") > str.size()){

break;

} else {

str.erase(0, str.find(",") + 1);

}

}

return vt;

}

void input(){

vector<string> document\_train;

vector<string> document\_test;

cin >> n;

string str\_tmp;

getline(cin, str\_tmp);

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

string str\_tmp;

getline(cin, str\_tmp);

document\_train.push\_back(str\_tmp);

}

cin >> q;

getline(cin, str\_tmp);

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

string str\_tmp;

getline(cin, str\_tmp);

document\_test.push\_back(str\_tmp);

}

for(string v : document\_train){

vector<string> element = split\_string(v);

vector\_train.push\_back(element);

}

for(string v : document\_test){

vector<string> element = split\_string(v);

vector\_test.push\_back(element);

}

}

void pre\_processing(){

// tinh tan suat tu xuat hien nhieu nhat trong van ban i

for(vector<string> str\_tmp : vector\_train){

map<string, int> m;

int max\_f = 0;

for(string word\_tmp : str\_tmp){

map<string, int>::iterator ite = m.find(word\_tmp);

if(ite == m.end()){ // neu tu nay chua co trong tu dien mini

m.insert({word\_tmp, 1});

} else {

ite->second += 1;

}

max\_f = max(m[word\_tmp], max\_f);

}

f\_max.push\_back(max\_f);

}

}

int frequence\_word\_int\_document\_i(string word, int i){

if(fe.find({word, i}) != fe.end()){ // neu da co trong kho luu tru thi lay ra va tra ve

return fe[{word, i}];

}

int index = 0;

vector<string> str\_tmp = vector\_train[i];

for(string v : str\_tmp){

if(word == v) index++;

}

fe.insert({{word, i}, index});

return index;

}

/\* Nguyen Duy Khanh - 20225019 \*/

int count\_document\_contain\_word(string word){

if(df.find(word) != df.end()){

return df[word];

}

int index = 0;

for(vector<string> str\_tmp : vector\_train){

vector<string>::iterator ite = find(str\_tmp.begin(), str\_tmp.end(), word);

if(ite != str\_tmp.end()){

index++;

}

}

df.insert({word,index});

return index;

}

int search\_engine(vector<string> list\_word){

double score\_max = -1000;

int predict\_label = -1;

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

vector<string> list\_word\_train\_doc = vector\_train[i];

double score = 0;

for(string word : list\_word){

if(find(list\_word\_train\_doc.begin(), list\_word\_train\_doc.end(), word) == list\_word\_train\_doc.end()){ // tu nay khong xuat hien trong van ban

continue;

} else {

int ftd = frequence\_word\_int\_document\_i(word, i);

int dft = count\_document\_contain\_word(word);

int maxfd = f\_max[i];

double tf\_word = 0.5 + 0.5 \* ((double) ftd / maxfd);

double idf\_word = log2((double) n / dft);

score += tf\_word \* idf\_word;

}

}

if(score > score\_max) {

predict\_label = i;

score\_max = score;

}

}

return predict\_label + 1;

}

int main(){

ios\_base::sync\_with\_stdio(false); cin.tie(NULL);

input();

pre\_processing();

cout<<endl;

for(int i=0; i<q; i++)

cout << search\_engine(vector\_test[i]) << endl;

return 0;

}

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

Nguyen Duy Khanh - 20225019

Ma lop TH: 744469

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

## **Bài 4.11.** Bảo vệ lâu đài

A white background with blue lines

Description automatically generated

Hình 11.1. Code bài 11 - Ảnh 1

A white background with blue and red lines

Description automatically generated

Hình 11.2. Code bài 11 - Ảnh 2

A computer screen shot of a black screen

Description automatically generated

Hình 11.3. Test case 1 – Bài 11

A computer screen with a black screen

Description automatically generated

Hình 11.4. Test case 2 – Bài 11

A computer screen with a black screen

Description automatically generated

Hình 11.5. Test case 3 – Bài 11

A screenshot of a computer

Description automatically generated

Hình 11.6. Test case 4 – Bài 11

A computer screen with a black screen

Description automatically generated

Hình 11.7. Test case 5 – Bài 11

/\*

Bai 4.11 - Tuan 15

Nguyen Duy Khanh - 20225019 - 744469 - 20241

\*/

#include<bits/stdc++.h>

using namespace std;

int n, s;

struct couple{

int a;

int k;

};

int sum = 0;

int died = 0;

couple wall[100001];

stack<int> ST;

priority\_queue<int, vector<int>> PQ;

bool compare(couple x, couple y){

return x.k > y.k;

}

void process(){

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

while (!PQ.empty() && PQ.top() > wall[i].k && s>0) {

died += PQ.top();

PQ.pop();

s--;

}

int thuong = wall[i].a / wall[i].k;

int du = wall[i].a % wall[i].k;

if (du != 0) PQ.push(du);

if (thuong <= s) {

s -= thuong;

died += thuong\*wall[i].k;

} else {

died += s\*wall[i].k;

s = 0;

break;

}

}

while (s>0 && !PQ.empty()){

died += PQ.top();

s--;

PQ.pop();

}

}

int main(){

cin>>n>>s;

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

{

cin>>wall[i].a>>wall[i].k;

sum += wall[i].a;

}

sort(wall, wall+n, compare); //Sap xep giam dan theo kha nang bao ve k

process();

cout<<endl;

cout<<sum-died;

return 0;

}

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

Nguyen Duy Khanh - 20225019

Ma lop TH: 744469

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

## **Bài 4.12.** 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,…,hn . Hãy xác định hình chữ nhật có diện tích lớn nhất có thể tạo thành từ các cột liên tiếp.

A screenshot of a computer

Description automatically generated

Hình 12.1. Code bài 12 - Ảnh 1

A screenshot of a computer

Description automatically generated

Hình 12.2. Code bài 12 - Ảnh 2

A screenshot of a computer

Description automatically generated

Hình 12.3. Testcase 1 – Bài 12

A screenshot of a computer

Description automatically generated

Hình 12.4. Testcase 2 – Bài 12

/\*

Bai 4.12 - Tuan 15

Nguyen Duy Khanh - 20225019 - 744469 - 20241

\*/

#include<iostream>

#include<stack>

using namespace std;

int h[1000001];

int left\_position[1000001]; // vi tri phan tu khong nho hon o ben trai nhat

int right\_position[1000001]; // vi tri phan tu không nho hon o ben phai nhat

stack<pair<int, int>> S\_right; // luu gia tri và vi tri

stack<pair<int, int>> S\_left; // luu gia tri và vi tri

int main(){

int n;

cin>>n;

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

cin>>h[i];

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

left\_position[i] = i;

while (!S\_left.empty()){

pair<int, int> top = S\_left.top();

if (top.first >= h[i]){

left\_position[i] = top.second;

S\_left.pop();

} else break;

}

S\_left.push({h[i], left\_position[i]});

}

for (int i=n-1; i>=0; i--){

right\_position[i] = i;

while (!S\_right.empty()){

pair<int, int> top = S\_right.top();

if (top.first >= h[i]){

right\_position[i] = top.second;

S\_right.pop();

} else break;

}

S\_right.push({h[i], right\_position[i]});

}

int max\_s=0;

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

int current\_s = h[i]\*(right\_position[i] - left\_position[i] + 1);

max\_s = max(max\_s, current\_s);

}

cout<<max\_s;

return 0;

}

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

Nguyen Duy Khanh - 20225019

Ma lop TH: 744469

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

## **Bài 4.13**. Đếm xâu con

A screenshot of a computer

Description automatically generated

Hình 13.1. Code bài 13

A computer screen shot of a black screen

Description automatically generated

Hình 13.2. Testcase 1 – Bài 13

A screenshot of a computer

Description automatically generated

Hình 13.3. Testcase 2 – Bài 13

A screenshot of a computer

Description automatically generated

Hình 13.4. Testcase 3 – Bài 13

/\*

Bai 4.13 - Tuan 15

Nguyen Duy Khanh - 20225019 - 744469 - 20241

\*/

#include<iostream>

#include<bits/stdc++.h>

using namespace std;

map<int, int> Map;

int sum = 0;

int count\_chain = 0;

int main(){

ios\_base::sync\_with\_stdio(false); cin.tie(NULL);

string s;

cin>>s;

Map.insert({0, 1});

int l = s.length();

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

if (s[i] == '0') sum += -1;

else sum += 1;

count\_chain += Map[sum];

Map[sum]++;

}

cout<<endl;

cout<<count\_chain;

return 0;

}

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

Nguyen Duy Khanh - 20225019

Ma lop TH: 744469

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

A screenshot of a computer

Description automatically generated

Hình 13.5. Testcase 4 – Bài 13

A screenshot of a computer

Description automatically generated

Hình 13.6. Testcase 5 – Bài 13