Mục lục nội dung

[**BÀI THỰC HÀNH SỐ 04 – TUẦN 39** 3](#_Toc168645092)

[BÀI TẬP TRÊN LAP 3](#_Toc168645093)

**Bài tập 4.1.** Đảo ngược danh sách liên kết đơn…………………………………………3

[**Bài tập 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. 7](#_Toc168645094)

## **[Bài tập 4.3.](#_Toc168645095)** [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>;](#_Toc168645095)

[8](#_Toc168645095)

[**Bài tập 4.4.** 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. 9](#_Toc168645096)

[**Bài tập 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). 14](#_Toc168645097)

[**Bài tập 4.6.** 16](#_Toc168645098)

[**Bài tập 4.7.** 20](#_Toc168645099)

**Bài tập 4.8**……………………………………………………………………………24

[**Bài tập về nhà:** 30](#_Toc168645100)

[**Bài tập 4.11:** Bức tường bao quanh một lâu đài nọ được cấu thành từ n đoạn tường được đánh số từ 1 đến n 37](#_Toc168645101)

[**Bài tập 4.13:** 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. 43](#_Toc168645102)

# **BÀI THỰC HÀNH SỐ 04 – TUẦN 39**

# BÀI TẬP TRÊN LAP

**Bài tập 4.1.** Đảo ngược một danh sách liên kết đơn

Hãy hoàn thiện các hàm thao tác trên một danh sách liên kết:

* Thêm một phần tử vào đầu danh sách liên kết
* In danh sách
* Đảo ngược danh sách liên kết (yêu cầu độ phức tạp thời gian O(N) và chi phí bộ nhớ dùng thêm O(1))

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

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#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 \*temp = new Node(data);

temp->next = head;

head = temp;

return head;

}

void print(Node \*head){

Node \*tmp = head;

while (tmp != NULL){

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

tmp = tmp->next;

}

std::cout<<std::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: 738919

\*/

int n, u;

std::cin>>n;

Node \*head = NULL;

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

std::cin>>u;

head = prepend(head, u);

}

std::cout<<"Original list: ";

print(head);

head = reverse(head);

std::cout<<"Reversed list: ";

print(head);

return 0;

}

## **Bài tập 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.

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

Mô tả được tạo tự động

#include <iostream>

#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: 738919

\*/

cout << setprecision(2) << fixed;

return 0;

}

## **Bài tập 4.3.** 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, màn hình, phần mềm

Mô tả được tạo tự động

#include<bits/stdc++.h>

using namespace std;

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

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

double x1 = get<0>(a);

double y1 = get<1>(a);

double z1 = get<2>(a);

double x2 = get<0>(b);

double y2 = get<1>(b);

double z2 = get<2>(b);

double x = y1\*z2 - y2\*z1;

double y = z1\*x2 - z2\*x1;

double z = x1\*y2 - x2\*y1;

return make\_tuple(x,y,z);

}

int main(){

cout<<setprecision(2)<<fixed;

Vector a{1.2, 4, -0.5};

Vector b{1.5, -2, 2.5};

Vector c = cross\_product(a, b);

cout<<get<0>(c)<<" "<<get<1>(c)<<" "<<get<2>(c)<<endl;

}

## **Bài tập 4.4.** 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, đồ điện tử, ảnh chụp màn hình, phần mềm

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

Mô tả được tạo tự động

#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(std::vector<int> &a) {

a.erase(std::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;

}

int main() {

/\*

Nguyen Duy Khanh - 20225019

Ma lop: 738919

\*/

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;

}

## 

## **Bài tập 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).Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm Mô tả được tạo tự động

#include <iostream>

#include <vector>

#include <list>

#include <algorithm>

#include <stack>

using namespace std;

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

stack<int> S;

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

S.push(1);

while (!S.empty()) {

int u = S.top();

if (!visited[u]){

visited[u] = true;

std::cout<< u << std::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()

// {

/\*

Nguyen Duy Khanh – 20225019

Ma lop: 738919

\*/

// 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);

// }

**Bài tập 4.6.** Viết hàm *void bfs(vector< list<int> > adj)* thực hiện thuật toán BFS 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 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, ảnh chụp màn hình, màn hình, phần mềm

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

Mô tả được tạo tự động

#include <iostream>

#include <vector>

#include <list>

#include <algorithm>

#include <queue>

using namespace std;

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

queue<int> Q;

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

Q.push(1);

while (!Q.empty()) {

int u = Q.front();

if (!visited[u]){

visited[u] = true;

std::cout<< u << std::endl;

}

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

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

adj[u].pop\_front();

if(!visited[v]){

Q.push(v);

}

}else {

Q.pop();

}

}

}

// int main()

// {

/\*

Nguyen Duy Khanh – 20225019

Ma lop: 738919

\*/

// 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);

// }

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

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

Mô tả được tạo tự động

#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;

}

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

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

Mô tả được tạo tự độngẢnh có chứa văn bản, đồ điện tử, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động

#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() {

/\*

Nguyen Duy Khanh – 20225019

Ma lop: 738919

\*/

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);

}

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

#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() {

/\*

Nguyen Duy Khanh - 20225019

Ma lop: 738919

\*/

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;

}

## **Bài tập về nhà:**

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

Ảnh có chứa văn bản, Phông chữ, phần mềm, số

Mô tả được tạo tự động Ảnh có chứa văn bản, hàng, số, ảnh chụp màn hình

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động

#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;

}

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(){

/\*

Nguyen Duy Khanh - 20225019

Ma lop: 738919

\*/

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;

}

## **Bài tập 4.11:** Bức tường bao quanh một lâu đài nọ được cấu thành từ n đoạn tường được đánh số từ 1 đến n

Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động Ảnh có chứa văn bản, phần mềm, hàng, ảnh chụp màn hình

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, màu đen

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, màu đen

Mô tả được tạo tự động

#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(){

/\*

Nguyen Duy Khanh - 20225019

Ma lop: 738919

\*/

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;

}

**Bài tập 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.

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

Mô tả được tạo tự động



#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(){

/\*

Nguyen Duy Khanh - 20225019

Ma Lop: 738919

\*/

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;

}

## **Bài tập 4.13:** 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, phần mềm, hàng, Phông chữ

Mô tả được tạo tự động



#include<iostream>

#include<bits/stdc++.h>

using namespace std;

map<int, int> Map;

int sum = 0;

int count\_chain = 0;

int main(){

/\*

Nguyen Duy Khanh - 20225019

Ma lop: 738919

\*/

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;

}