Contents

[Bài thực hành số 4 – Tuần 38 2](#_Toc138875727)

[**Bài tập 4.1.**Đảo ngược một danh sách liên kết đơn 2](#_Toc138875728)

[**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](#_Toc138875729)

[**Bài tập 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. 9](#_Toc138875730)

[**Bài tập 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 12](#_Toc138875731)

[**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). 19](#_Toc138875732)

[**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). 23](#_Toc138875733)

[**Bài tập 4.7.**Viết các hàm thực hiện các phép giao và hợp của hai tập hợp được biểu diễn bằng set 27](#_Toc138875734)

[**Bài tập 4.8.**Viết các hàm thực hiện các phép giao và hợp của hai tập hợp mờ được biểu diễn bằng map. 31](#_Toc138875735)

[**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\_queue 36](#_Toc138875736)

[**Bài tập 4.10.** Xây dựng một máy tìm kiếm (search engine) đơn giản. 40](#_Toc138875737)

[**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 . 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. 54](#_Toc138875738)

[**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. 63](#_Toc138875739)

[**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. 74](#_Toc138875740)

# Bài thực hành số 4 – Tuần 38

## **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, Phông chữ, đại số

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

Ảnh có chứa văn bản, Phông chữ, số, hàng

Mô tả được tạo tự động

// Nguyen Van Duy - 20215334

/\*

Bài 4.1. 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))

\*/

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

Node(int data) {

this->data = data;

next = NULL;

}

};

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

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

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

# YOUR CODE HERE #

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

Node\* newNode = new Node(data); // new node is new head

newNode->next = head; // point to head

return newNode; // return new node

}

// print the list content on a line

void print(Node\* head) {

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

# YOUR CODE HERE #

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

Node\* temp = head; // temp node

while (temp != NULL) { // loop through

cout << temp->data << " "; // print

temp = temp->next; // next node

}

cout << endl; // end of

}

// return the new head of the reversed list

Node\* reverse(Node\* head) {

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

# YOUR CODE HERE #

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

Node\* prev = NULL; // previous

Node\* curr = head; // current

Node\* next = NULL; // next

while (curr != NULL) { // loop through

next = curr->next; // save next node

curr->next = prev; // move the cursor to previous node

prev = curr; // move to next

curr = next; // move to next

}

return prev;

}

int main() {

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;

}

// Nguyen Van Duy - 20215334

## **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.

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

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

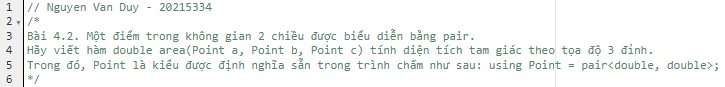
# YOUR CODE HERE #

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

}

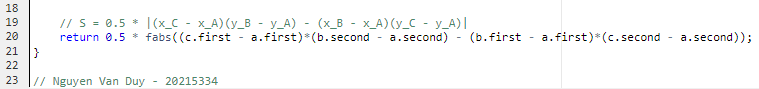
trong đó, Point là kiểu được định nghĩa trước trong trình chấm như sau:

using Point = pair<double, double>;



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

Mô tả được tạo tự động



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

Mô tả được tạo tự động

// Nguyen Van Duy - 20215334

/\*

Bài 4.2. 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>;

\*/

#include <iostream>

#include <cmath>

#include <iomanip>

#include <utility>

using namespace std;

using Point = pair<double, double>;

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

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

# YOUR CODE HERE #

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

// S = 0.5 \* |(x\_C - x\_A)(y\_B - y\_A) - (x\_B - x\_A)(y\_C - y\_A)|

return 0.5 \* fabs((c.first - a.first)\*(b.second - a.second) - (b.first - a.first)\*(c.second - a.second));

}

// Nguyen Van Duy - 20215334

## **Bài tập 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.

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

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

# YOUR CODE HERE #

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

}

trong đó Vector là kiểu được định nghĩa sẵn trong trình chấm như sau:

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

Ảnh có chứa văn bản, hàng, Phông chữ, số

Mô tả được tạo tự động

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

Mô tả được tạo tự động

Ảnh có chứa văn bản, số, Phông chữ, hàng

Mô tả được tạo tự động

// Nguyen Van Duy - 20215334

/\*

Bài 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>;

\*/

#include <iostream>

#include <cmath>

#include <iomanip>

using namespace std;

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

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

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

# YOUR CODE HERE #

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

// res\_X = a\_Y \* b\_Z - a\_Z \* b\_Y

// res\_Y = a\_Z \* b\_X - a\_X \* b\_Z

// res\_Z = a\_X \* b\_Y - a\_Y \* b\_X

return make\_tuple(get<1>(a) \* get<2>(b) - get<2>(a) \* get<1>(b),

get<2>(a) \* get<0>(b) - get<0>(a) \* get<2>(b),

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

}

// Nguyen Van Duy - 20215334

## **Bài tập 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

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, số

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

Ảnh có chứa văn bản, Phông chữ, số, hàng

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

// Nguyen Van Duy - 20215334

/\*

Bài 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.

\*/

#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) {

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

# YOUR CODE HERE #

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

for (size\_t i = 0; i < a.size(); i++) { // loop

if (a[i] % 2 == 0) { // check even numbers

a.erase(a.begin() + i); // remove

i--; // decrement to keep index

}

}

}

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

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

# YOUR CODE HERE #

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

sort(a.begin(), a.end(), [] (int a, int b) -> bool {

return a >= b; // stable sort by ascending order

});

}

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

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

# YOUR CODE HERE #

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

// merge 2 sorted vectors, return sorted vector

vector<int> c; // result

size\_t i = 0, j = 0; // index

while (i < a.size() && j < b.size()) { // loop until a ends or b ends

// add smaller element

if (a[i] > b[j]) {

c.push\_back(a[i]);

i++; // increment index of vector a

} else {

c.push\_back(b[j]);

j++; // increment index of vector b

}

}

while (i < a.size()) {

c.push\_back(a[i]); // add the remaining elements of the vector a

i++;

}

while (j < b.size()) {

c.push\_back(b[j]); // add the remaining elements of the vector b

j++;

}

return c; // return

}

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;

}

// Nguyen Van Duy - 20215334

## **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, Phông chữ, số

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

// Nguyen Van Duy - 20215334

/\*

Bài 4.5. 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).

\*/

#include <iostream>

#include <vector>

#include <list>

#include <stack>

using namespace std;

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

stack<int> S;

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

S.push(1); // Bắt đầu từ đỉnh số 1

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

# YOUR CODE HERE #

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

while (!S.empty()) { // loop

int u = S.top(); // get top

S.pop(); // pop from the top of the stack

if (!visited[u]) { // check if u is not visited

visited[u] = true; // visit u

cout << u << endl; // print

list<int>::iterator it; // iterator

for (it = adj[u].end(); it-- != adj[u].begin(); ) { // loop through adjacent of u

if (!visited[\*it]) { // check if it is not visited

S.push(\*it); // push it to stack to visit later

}

}

}

}

}

// Nguyen Van Duy - 20215334

## **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, Phông chữ, số

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

// Nguyen Van Duy - 20215334

/\*

Bài 4.6. 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).

\*/

#include <iostream>

#include <vector>

#include <list>

#include <queue>

using namespace std;

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

queue<int> Q;

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

Q.push(1); // Bắt đầu từ đỉnh số 1

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

# YOUR CODE HERE #

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

visited[1] = true; // mark 1

while (!Q.empty()) { // loop

int u = Q.front(); // get first

Q.pop(); // remove an element from the front of the queue

cout << u << endl; // print

for (auto v : adj[u]) { // loop through adjacent of u

if (!visited[v]) { // check if it is not visited

Q.push(v); // push it to queue to visit later

visited[v] = true; // visit v

}

}

}

}

// Nguyen Van Duy - 20215334

## **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, Phông chữ, số

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

Ảnh có chứa văn bản, Phông chữ, hàng, số

Mô tả được tạo tự động

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

Mô tả được tạo tự động

// Nguyen Van Duy - 20215334

/\*

Bài 4.7. Viết các hàm thực hiện các phép giao

và hợp của hai tập hợp được biểu diễn bằng set.

\*/

#include <iostream>

#include <set>

using namespace std;

template<class T>

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

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

# YOUR CODE HERE #

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

set<T> c; // result

for (auto x : a) { // for each element in the set<T> a

c.insert(x); // insert to result

}

for (auto x : b) { // for each element in the set<T> b

c.insert(x); // insert to result

}

return c; // return result

}

template<class T>

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

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

# YOUR CODE HERE #

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

set<T> c; // result

for (auto x : a) { // for each element in the set<T> a

if (b.find(x) != b.end()) { // check if the element is already in the set<T> b

c.insert(x); // add it to the set<T> c (result)

}

}

return c; // return result

}

template<class T>

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

for (const T &x : a) {

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

}

std::cout << std::endl;

}

// Nguyen Van Duy - 20215334

## **Bài tập 4.8.**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, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, số

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

Ảnh có chứa văn bản, đồ điện tử, ảnh chụp màn hình, màn hình

Mô tả được tạo tự động

// Nguyen Van Duy - 20215334

/\*

Bài 4.8. 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.

\*/

#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) {

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

# YOUR CODE HERE #

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

map<T, double> c; // result

for (auto it = a.begin(); it != a.end(); ++it) { // loop through elements in the set a

c[it->first] = it->second; // add element to result

}

for (auto it = b.begin(); it != b.end(); ++it) { // loop through elements in the set b

if (c.find(it->first) == c.end()) { // check if element is not yet in the set

c[it->first] = it->second; // add element to result

} else { // if element is already in the set

c[it->first] = max(c[it->first], it->second); // max equals element

}

}

return c; // return result

}

template<class T>

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

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

# YOUR CODE HERE #

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

map<T, double> c; // result

for (auto it = a.begin(); it != a.end(); ++it) { // loop through elements of in the set a

if (b.find(it->first) != b.end()) { // check if element is already in the set b

c[it->first] = min(it->second, b.at(it->first)); // min equals element

}

}

return c; // return result

}

template<class T>

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

cout << "{ ";

for (const auto &x : a) {

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

}

cout << "}";

std::cout << std::endl;

}

// Nguyen Van Duy - 20215334

## **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\_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, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

// Nguyen Van Duy - 20215334

/\*

Bài 4.9. 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.

\*/

#include <iostream>

#include <queue>

#include <climits>

using namespace std;

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

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

# YOUR CODE HERE #

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

vector<int> dist(adj.size(), INT\_MAX); // result

dist[0] = 0; // update distance

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

pq.push({0, 0}); // start with 0

while (!pq.empty()) { // loop

int u = pq.top().second; // get vertex (u)

pq.pop(); // remove from top of priority queue

for (size\_t i = 0; i < adj[u].size(); i++) { // loop through adjacent vertices of current vertex (u)

int v = adj[u][i].first; // get vertex (v)

int w = adj[u][i].second; // get weight to the vertex (v)

if (dist[v] > dist[u] + w) { // check if old distance is greater than new distance

dist[v] = dist[u] + w; // update distance

pq.push({dist[v], v}); // recalculate distance from vertex (v)

}

}

}

return dist;

}

// Nguyen Van Duy - 20215334

## **Bài tập 4.10.** 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, Phông chữ, số

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, đại số

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, đại số

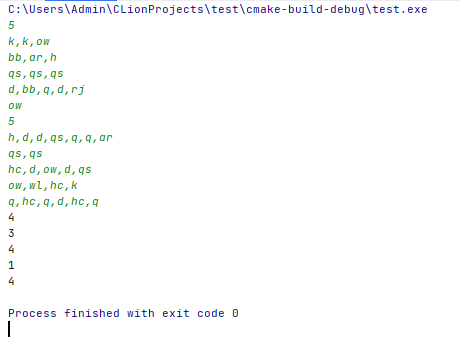
Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động



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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, số

Mô tả được tạo tự động

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

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, số

Mô tả được tạo tự động

// Nguyen Van Duy - 20215334

/\*

Bài 4.10. 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

\*/

#include <iostream>

#include <vector>

#include <map>

#include <cmath>

using namespace std;

int n, q;

vector<vector<string>> text\_vector; // n texts

vector<vector<string>> query\_vector; // q queries

map<pair<string, int>, int> f; // frequency of occurrence of the word @t in the text[i] (pair<string, int>)

vector<int> max\_f; // maximum frequency of occurrence in the text[i]

map<string, int> df; // number of text containing the word t

// split string by ','

inline vector<string> split\_string(const string &s) {

vector<string> res; // result

string t;

for (char i: s) {

if (i != ',') {

t.push\_back(i); // add to t or t += i

} else {

res.push\_back(t); // add t to vector res

t.clear(); // t = ""

}

}

res.push\_back(t); //

return res; // return result

}

inline void input() {

cin >> n; // input n

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

string temp; // allocate

cin >> temp; // get string

text\_vector.push\_back(split\_string(temp)); // add split string

}

cin >> q; // input q

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

string temp; // allocate

cin >> temp; // get string

query\_vector.push\_back(split\_string(temp)); // add split string

}

}

inline void calc\_f() {

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

int maximum = 0; // max f

for (auto &t : text\_vector[i]) {

pair<string, int> key = make\_pair(t, i); // key : word @t in text[i]

if (f.find(key) != f.end()) { // check if key exists

++f[key]; // increase f

} else {

f[key] = 1; // set f

}

maximum = max(maximum, f[key]); // check and update maximum

}

max\_f.push\_back(maximum); // push\_back max f to vector

}

}

inline int calc\_df(const string& t) {

if (df.find(t) != df.end()) { // check if t exists

return df[t]; // return calculated result

}

int res = 0; // result

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

for (string &s : text\_vector[i]) {

if (t == s) {

++res; // increase 1

break;

}

}

}

df[t] = res; // update result

return res; // return result

}

inline double score(const string& t, int index\_text) {

if (f[{t, index\_text}] == 0) return 0; // score = 0

// score = tf \* idf

// trong do :

// tf = 0.5 + 0.5 \* f / max\_f

// idf = log2 (N / df)

return (0.5 + 0.5 \* f[{t, index\_text}] / max\_f[index\_text]) \* log2(n / calc\_df(t));

}

inline int search\_engine(vector<string> &queries) {

int res = -1; // result

double max\_score = -1; // maximum score

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

double score\_cur = 0; // current score

for (auto &t : queries) {

score\_cur += score(t, i); // calculate and increase

}

if (score\_cur > max\_score) { // check if max score

max\_score = score\_cur; // update max score

res = i; // update index (result)

}

}

return res + 1; // return

}

int main() {

input();

calc\_f(); // pre-calc

for (auto &query : query\_vector) {

cout << search\_engine(query) << endl;

}

return 0;

}

// Nguyen Van Duy - 20215334

## **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 . 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.

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, Phông chữ, số

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp 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

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, phần mềm, số

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

// Nguyen Van Duy - 20215334

/\*

Bài 4.11. 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.

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.

\*/

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

class Wall {

private:

int a; // there are @a members in Army

int k; // each soldier can repel the attack of @k enemies

public:

// constructor with no arguments

Wall() : a(0), k(0) {}

// constructor with arguments : @a, @k

Wall(int a, int k) : a(a), k(k) {}

[[nodiscard]] int passed() const { return a; }

[[nodiscard]] int getK() const { return k; }

void send\_soldier(int numbers = 1) {

a = max(0, a - numbers \* k); // recalculate number of enemies entering

k = min(k, a); // update @k

}

friend bool operator<(const Wall &w\_1, const Wall &w\_2) { return w\_1.k < w\_2.k; }

friend bool operator>(const Wall &w\_1, const Wall &w\_2) { return w\_1.k > w\_2.k; }

friend bool operator<=(const Wall &w\_1, const Wall &w\_2) { return w\_1.k <= w\_2.k; }

friend bool operator>=(const Wall &w\_1, const Wall &w\_2) { return w\_1.k >= w\_2.k; }

friend istream &operator>>(istream &is, Wall &w) {

is >> w.a >> w.k; // input @a, @k

return is; // return stream

}

friend ostream &operator<<(ostream &os, const Wall w) {

os << "{a = " << w.a << "; k = " << w.k << "}"; // output

return os; // return stream

}

};

// sort\_walls : descending order

inline void sort\_walls(vector<Wall> &v) { sort(v.begin(), v.end(), greater<>()); }

inline int number\_of\_enemies\_entering(vector<Wall> &v) {

int count = 0; // counter

for (auto &w: v) { // loop over each wall

count += w.passed(); // increment counter

}

return count; // return result

}

int main() {

int n, s; // number of wall, and number of soldier

cin >> n >> s; // enter @n, @s

vector<Wall> walls(n); // wall list

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

cin >> walls[i]; // enter @a, @k of the wall

}

while (s--) { // send soldiers one by one to the wall

sort\_walls(walls); // descending order : sort from largest to smallest

walls[0].send\_soldier(); // send a soldier to

}

cout << number\_of\_enemies\_entering(walls) << endl; // output

return 0;

}

// Nguyen Van Duy – 20215334

## **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, Phông chữ, ảnh chụp màn hình

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

// Nguyen Van Duy - 20215334

/\*

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.

\*/

#include <iostream>

#include <vector>

#include <queue>

using namespace std;

struct rectangle {

int height;

int width;

rectangle() : height(0), width(0) {}

rectangle(int height, int width) : height(height), width(width) {}

rectangle increment(bool increment\_height = false, bool increment\_width = true) {

if (increment\_height) { // check if increment height or not

height++; // increment height

}

if (increment\_width) { // check if increment width or not

width++; // increment width

}

return \*this; // return rectangle

}

rectangle decrement(bool decrement\_height = false, bool decrement\_width = true) {

if (decrement\_height) { // check if decrement height or not

height--; // decrement height

}

if (decrement\_width) { // check if decrement width or not

width--; // decrement width

}

return \*this; // return rectangle

}

friend rectangle operator++(rectangle &rect) { return rect.increment(); }

friend const rectangle operator++(rectangle &rect, int) {

rectangle temp = rect; // copy old rectangle

rect.increment(); // increment rectangle

return temp; // return old rectangle

}

friend rectangle operator--(rectangle &rect) { return rect.decrement(); }

friend rectangle operator--(rectangle &rect, int) {

rectangle temp = rect; // copy old rectangle

rect.decrement(); // decrement rectangle

return temp; // return old rectangle

}

friend rectangle operator+(rectangle &rect, int v) {

return {rect.height, rect.width + v};

}

friend void operator+=(rectangle &rect, int v) {

rect.width += v;

}

friend ostream &operator<<(ostream &os, rectangle rect) {

// output : (height x width)

os << "(" << rect.height << " x " << rect.width << ")";

return os; // return stream

}

friend istream &operator>>(istream &is, rectangle &rect) {

is >> rect.height; // input : height

rect.width = 0; // set width = 0

return is; // return stream

}

[[nodiscard]] int superficie() const {

return height \* width; // area(superficie) = height \* width

}

};

int main() {

int n; // number of columns

int res = INT\_MIN; // reslut : largest area(superficie), set to INT\_MIN

cin >> n; // input n

vector<rectangle> v(n); // rectangles, whose height corresponds to height of columns

queue<rectangle> q, old\_queue, new\_queue;

// q : contains original rectangles

// old\_queue : (current\_queue) contains current rectangles

// new\_queue : (next\_queue) contains next rectangles

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

cin >> v[i]; // input rectangles

q.push(v[i]); // push into q (original rectangles)

}

while (!q.empty()) {

rectangle rect = q.front(); // get 1 of original rectangles

q.pop(); // remove from q (original rectangles)

while (!old\_queue.empty()) { // loop

rectangle old\_rect = old\_queue.front(); // get from old queue

old\_queue.pop(); // remove an element from old queue

if (old\_rect.height < rect.height) { // check if

new\_queue.push(++old\_rect); // rectangle extension and push into next queue

} else {

rect.width = max(rect.width, old\_rect.width); // update for original rectangle

res = max(res, old\_rect.superficie()); // update largest rectangle

}

}

old\_queue.swap(new\_queue); // switch to next queue

old\_queue.push(++rect); // push original rectangle into current queue

}

while (!old\_queue.empty()) { // loop remaining rectangle

rectangle old\_rect = old\_queue.front(); // get rectangle

old\_queue.pop(); // remove from queue

res = max(res, old\_rect.superficie()); // update remaining rectangle

}

cout << res; // print result

return 0;

}

// Nguyen Van Duy - 20215334

## **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, số, Phông chữ

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, tài liệu

Mô tả được tạo tự động

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

Mô tả được tạo tự động

// Nguyen Van Duy - 20215334

/\*

Bài 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.

\*/

#include <iostream>

#include <map>

using namespace std;

inline int calc(int v) {

// voi so lan xuat hien chenh lech la v, thi so xau dem duoc la:

// result = 1 + 2 + 3 + ... + (v-1) = v \* (v-1) / 2

return v \* (v-1) / 2;

}

int main() {

string str;

cin >> str;

map<int, int> m; // <chenh lech : so lan>

map<int, int>::iterator it;

int chenh\_lech = 0; // chenh lech = so luong 1 - so luong 0

for (char c : str) {

if (c == '0') chenh\_lech--; // giam 1 neu xuat hien 0

else chenh\_lech++; // tang 1 neu xuat hien 1

it = m.find(chenh\_lech);

if (it != m.end()) { // neu do chenh lech da ton tai

it->second += 1; // tang so lan xuat hien cua chenh lech

} else {

m.insert({chenh\_lech, 1});

}

}

int res = 0; // result

for (it = m.begin(); it != m.end() ; ++it) {

res += calc(it->second);

if (it->first == 0) res += it->second; // neu chenh lech = 0

}

cout << res; // print result

return 0;

}

// Nguyen Van Duy - 20215334