Mục lục nội dung

Bài thực hành 4: Thực hành sử dụng các cấu trúc dữ liệu cơ bản để giải quyết các bài toán cụ thể

Phần 1: Bài tập thực hành

Bài tập 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))

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>;

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>;

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.

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).

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).

Bài tập 7:

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

Bài tập 8:

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

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

Phần 2: Bài tập về nhà

Bài tập 10: Search Engine

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

Cho NN văn bản và QQ truy vấn. Với mỗi truy vấn, cần trả về văn bản khớp với truy vấn đó nhất.

Bài tập 11: Lịch trình chụp ảnh

Superior là một hòn đảo tuyệt đẹp với nn địa điểm chụp ảnh và các đường một chiều nối các điểm chụp ảnh với nhau. Đoàn khách tham quan có rr người với sở thích chụp ảnh khác nhau. Theo đó, mỗi người sẽ đưa ra danh sách các địa điểm mà họ muốn chụp. Bạn cần giúp mỗi người trong đoàn lập lịch di chuyển sao cho đi qua các điểm họ yêu cầu đúng một lần, không đi qua điểm nào khác, bắt đầu tại điểm đầu tiên và kết thúc tại điểm cuối cùng trong danh sách mà họ đưa ra, và có tổng khoảng cách đi lại là nhỏ nhất.

Bài tập 12: Đếm đường đi

Cho đồ thị vô hướng GG, hãy đếm số đường đi đi qua kk cạnh và không đi qua đỉnh nào quá một lần.

Mục lục hình ảnh

Bài thực hành số 4 – Tuần 15

BÀI TẬP TRÊN LAP

Bài 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))

For example:

Input	Result
10 -1 4 5 7 2 4 6 7 12 50	Original list: 50 12 7 6 4 2 7 5 4 -1 Reversed list: -1 4 5 7 2 4 6 7 12 50

My Courses ▾ English (en) ▾

```
41      while(p2 != NULL) {
42          Node* tmp = p2->next;  // Lưu trữ node tiếp theo
43          p2->next = p1;  // Đảo ngược liên kết
44          p1 = p2;
45          p2 = tmp;
46      }
47      head = p1;  // Cập nhật head mới
48      return head;  // Trả về head mới
49  }
50
51  // Họ và tên: Nguyễn Viêtt Anh
52  // MSSV: 20215307
53
54  int main() {
55      int n, u;
56      cin >> n;
57      Node* head = NULL;
58      for (int i = 0; i < n; ++i) {
59          cin >> u;
60          head = prepend(head, u);  // Thêm phần tử vào đầu danh sách
61      }
62
```

Precheck | Check

	Input	Expected	Got
✔	10 -1 4 5 7 2 4 6 7 12 50	Original list: 50 12 7 6 4 2 7 5 4 -1 Reversed list: -1 4 5 7 2 4 6 7 12 50	Ori Rev
✔	1 6	Original list: 6 Reversed list: 6	Ori Rev
✔	15 2 3 -1 4 6 -7 12 5 7 12 4 76 2 5 54	Original list: 54 5 2 76 4 12 7 5 12 -7 6 4 -1 3 2 Reversed list: 2 3 -1 4 6 -7 12 5 7 12 4 76 2 5 54	Ori Rev

Hình 1 Bài 4.1: Đảo ngược danh sách liên kết đơn

```
#include <iostream>
using namespace std;

struct Node {
    int data;
    Node* next;

    Node(int data) {
        this->data = data;  // Khởi tạo giá trị data của node
        next = NULL;  // Khởi tạo next là NULL
    }
};

// Thêm một phần tử mới vào đầu danh sách
Node* prepend(Node* head, int data) {
    Node* q = new Node(data);  // Tạo một node mới với giá trị data
    if(head == NULL) return q;
    q->next = head;  // Liên kết node mới với head hiện tại
    head = q;
    return head;
}
```

```
// In nội dung của danh sách trên một dòng
void print(Node* head) {
    Node* p = head;
    if(head == NULL) return;  // Nếu danh sách rỗng, thoát khỏi hàm
    while(p != NULL) {
        cout << p->data << ' ';  // In giá trị của node hiện tại
        p = p->next;  // Di chuyển tới node tiếp theo
    }
    return;
}

// Trả về head mới của danh sách đã đảo ngược
Node* reverse(Node* head) {
    if(head == NULL) return NULL;  // Nếu danh sách rỗng, trả về NULL
    if(head->next == NULL) return head;  // Nếu danh sách chỉ có một phần tử, trả về head
    Node* p1 = head;  // Khởi tạo p1 là head
    Node* p2 = head->next;  // Khởi tạo p2 là node thứ hai
    p1->next = NULL;  // Ngắt liên kết của node đầu tiên
    while(p2 != NULL) {
        Node* tmp = p2->next;  // Lưu trữ node tiếp theo
        p2->next = p1;  // Đảo ngược liên kết
        p1 = p2;
```

```
        p2 = tmp;
    }
    head = p1;  // Cập nhật head mới
    return head;  // Trả về head mới
}

// Họ và tên: Nguyễn Viettt Anh
// MSSV: 20215307

int main() {
    cout << "Ho va ten: Nguyen Viet Anh\n";
    cout << "MSSV: 20215307\n";
    int n, u;
    cin >> n;
    Node* head = NULL;
    for (int i = 0; i < n; ++i) {
        cin >> u;
        head = prepend(head, u);  // Thêm phần tử vào đầu danh sách
    }

    cout << "Original list: ";
    print(head);  // In danh sách ban đầu
```

```
    head = reverse(head);  // Đảo ngược danh sách
    cout << endl;
    cout << "Reversed list: ";
    print(head);  // In danh sách đã đảo ngược

    return 0;
}
```

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>;

For example:

Test	Result
cout << setprecision(2) << fixed; cout << area({1, 2}, {2.5, 10}, {15, -5.25}) << endl;	61.44

Answer: (penalty regime: 10, 20, ... %)

```
#include <iomanip>
#include <utility>
using namespace std;
using Point = pair<double, double>;

double distance(Point a, Point b) {
    // Hàm tính khoảng cách giữa hai điểm a và b
    return sqrt((a.first - b.first)*(a.first - b.first) + (a.second - b.second)*(a.second -
}

double area(Point a, Point b, Point c) {
    // Hàm tính diện tích tam giác được tạo bởi ba điểm a, b, c
    double ab = distance(a, b);  // Tính độ dài cạnh AB
    double bc = distance(b, c);  // Tính độ dài cạnh BC
    double ca = distance(c, a);  // Tính độ dài cạnh CA
    double p = (ab + bc + ca) / 2;  // Tính nửa chu vi tam giác
    double area = sqrt(p * (p - ab) * (p - bc) * (p - ca));  // Sử dụng công thức Heron
    return area;
}
// Ho va ten: Nguyen Viet Anh
// MSSV: 20215307
```

Precheck Check

	Test	Expected	Got	
✔	cout << setprecision(2) << fixed; cout << area({1, 2}, {2.5, 10}, {15, -5.25}) << endl;	61.44	61.44	✔
✔	cout << setprecision(2) << fixed; cout << area({1, 2.5}, {2.5, 15}, {-5.2, -5.75}) << endl;	32.56	32.56	✔

Passed all tests! ✔

Hình 2 Bài 4.2: Viết hàm tính diện tích tam giác

```
#include <iostream>
#include <cmath>
#include <iomanip>
#include <utility>
using namespace std;
using Point = pair<double, double>;

double distance(Point a, Point b) {
    // Hàm tính khoảng cách giữa hai điểm a và b
    return sqrt((a.first - b.first)*(a.first - b.first) + (a.second - b.second)*(a.second - b.second));
}

double area(Point a, Point b, Point c) {
    // Hàm tính diện tích tam giác được tạo bởi ba điểm a, b, c
    double ab = distance(a, b);  // Tính độ dài cạnh AB
    double bc = distance(b, c);  // Tính độ dài cạnh BC
    double ca = distance(c, a);  // Tính độ dài cạnh CA
    double p = (ab + bc + ca) / 2;  // Tính nửa chu vi tam giác
    double area = sqrt(p * (p - ab) * (p - bc) * (p - ca));  // Sử dụng công thức Heron
    return area;
}

// Ho va ten: Nguyen Viet Anh
// MSSV: 20215307

int main() {
```

```
    cout << "Ho va ten: Nguyen Viet Anh\n";
    cout << "MSSV: 20215307\n";
    cout << setprecision(2) << fixed;
    cout << area({1, 2}, {2.5, 10}, {15, -5.25}) << endl;
    return 0;
}
```

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

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>;

For example:

Test	Result
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;	9.00 -3.75 -8.40

Answer: (penalty regime: 10, 20, ... %)

```cpp
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
using Vector = tuple<double, double, double>;

Vector cross_product(Vector a, Vector b) {
    // Hàm tính tích chéo của hai vector a và b
    double x, y, z;
    z = get<0>(a) * get<1>(b) - get<0>(b) * get<1>(a);  // Thành phần z của tích chéo
    x = get<1>(a) * get<2>(b) - get<1>(b) * get<2>(a);  // Thành phần x của tích chéo
    y = get<2>(a) * get<0>(b) - get<2>(b) * get<0>(a);  // Thành phần y của tích chéo
    return Vector(x, y, z);  // Trả về vector kết quả
}
//Ho va ten: Nguyen Viet Anh
//MSSV: 20215307
```

Check

	Test	Expected	Got	
✔	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;	9.00 -3.75 -8.40	9.00 -3.75 -8.40	✔
✔	cout << setprecision(2) << fixed; Vector a {-2.2, 4.5, -1.5}; Vector b {3.5, -7, 7.5}; Vector c = cross_product(a, b); cout << get<0>(c) << ' ' << get<1>(c) << ' ' << get<2>(c) << endl;	23.25 11.25 -0.35	23.25 11.25 -0.35	✔

Hình 3 Bài 4.3 Viết hàm tính tích có hướng của 2 vector

```
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
using Vector = tuple<double, double, double>;

Vector cross_product(Vector a, Vector b) {
    // Hàm tính tích chéo của hai vector a và b
    double x, y, z;
    z = get<0>(a) * get<1>(b) - get<0>(b) * get<1>(a);  // Thành phần z của tích chéo
    x = get<1>(a) * get<2>(b) - get<1>(b) * get<2>(a);  // Thành phần x của tích chéo
    y = get<2>(a) * get<0>(b) - get<2>(b) * get<0>(a);  // Thành phần y của tích chéo
    return Vector(x, y, z);  // Trả về vector kết quả
}
//Ho va ten: Nguyen Viet Anh
//MSSV: 20215307
int main() {
    cout << "Ho va ten: Nguyen Viet Anh\n";
    cout << "MSSV: 20215307\n";
    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;
    return 0;
}
```

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.

For example:

Input	Result
5 6 2 3 6 7 -5 13 5 2 4 9 35	Odd elements of a: 3 7 -5 Odd elements of b: 13 5 9 35 Decreasingly sorted a: 7 3 -5 Decreasingly sorted b: 35 13 9 5 Decreasingly sorted c: 35 13 9 7 5 3 -5

Answer: (penalty regime: 10, 20, ... %)

```cpp
#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) {
    // Hàm xóa các phần tử chẵn khỏi vector a
    a.erase(remove_if(a.begin(), a.end(), [](int a) {
        return (a % 2 == 0);  // Kiểm tra phần tử có phải là số chẵn không
    }), a.end());  // Xóa các phần tử chẵn
}

void sort_decrease(vector<int> &a) {
    // Hàm sắp xếp vector a giảm dần
    sort(a.rbegin(), a.rend());
}

```

Precheck Check

	Input	Expected	Got
✔	5 6 2 3 6 7 -5 13 5 2 4 9 35	Odd elements of a: 3 7 -5 Odd elements of b: 13 5 9 35 Decreasingly sorted a: 7 3 -5 Decreasingly sorted b: 35 13 9 5 Decreasingly sorted c: 35 13 9 7 5 3 -5	Odd e Odd e Decre Decre Decre
✔	10 15 2 4 -7 2 5 7 13 9 43 55 12 3 65 32 2 4 675 76 21 57 87 321 54 76 -100	Odd elements of a: -7 5 7 13 9 43 55 Odd elements of b: 3 65 675 21 57 87 321 Decreasingly sorted a: 55 43 13 9 7 5 -7 Decreasingly sorted b: 675 321 87 65 57 21 3 Decreasingly sorted c: 675 321 87 65 57 55 43 21 13 9 7 5 3 -7	Odd e Odd e Decre Decre Decre

Passed all tests! ✔

Hình 4 Bài 4.4 Xóa phần tử, sắp xếp vector

```
#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) {
    // Hàm xóa các phần tử chẵn khỏi vector a
    a.erase(remove_if(a.begin(), a.end(), [](int a) {
        return (a % 2 == 0);  // Kiểm tra phần tử có phải là số chẵn không
    }), a.end());  // Xóa các phần tử chẵn
}

void sort_decrease(vector<int> &a) {
    // Hàm sắp xếp vector a giảm dần
    sort(a.rbegin(), a.rend());
}

//Ho va ten: Nguyen Viet Anh
//MSSV: 20215307

vector<int> merge_vectors(const vector<int> &a, const vector<int> &b) {
    // Hàm hợp nhất hai vector a và b, sau đó sắp xếp theo thứ tự giảm dần
    vector<int> sum;  // Vector kết quả
```

```
    for(int v : a) {
        sum.push_back(v);  // Thêm các phần tử của a vào sum
    }
    for(int v : b) {
        sum.push_back(v);  // Thêm các phần tử của b vào sum
    }
    sort(sum.rbegin(), sum.rend());  // Sắp xếp vector sum theo thứ tự giảm dần
    return sum;
}

int main() {
    cout << "Ho va ten: Nguyen Viet Anh\n";
    cout << "MSSV: 20215307\n";
    int m, n, 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).

For example:

Test	Result

`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);`
	`1`
	`2`
	`4`
	`7`
	`3`
	`5`

Answer: (penalty regime: 10, 20, ... %)

```cpp
#include <bits/stdc++.h>
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
//Ho va ten: Nguyen Viet Anh
//MSSV: 20215307
    while (!S.empty()) {
        int u = S.top();  // Lấy đỉnh đầu ngăn xếp
        if (!visited[u]) {
            visited[u] = true;  // Đánh dấu đỉnh u đã được thăm
            cout << u << endl;  // In đỉnh u
        }
        if (!adj[u].empty()) {
            int v = adj[u].front();  // Lấy đỉnh kề đầu tiên của u
            adj[u].pop_front();  // Xóa đỉnh kề đầu tiên của u khỏi danh sách
            if (!visited[v]) {
                S.push(v);  // Nếu đỉnh v chưa được thăm, thêm nó vào ngăn xếp
            }
        } else {
```

Precheck Check

	Test	Expected	Got	

✔	`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);`
		`1`
		`2`
		`4`
		`7`
		`3`
		`5`
			`1`
			`2`
			`4`
			`7`
			`3`
			`5`
				✔

✔	`int n = 10;`
	`vector< list<int> > adj;`
	`adj.resize(n + 1);`
		`1`
		`2`
		`7`
			`1`
			`2`
			`7`
				✔

Hình 5 Bài 4.5 Hàm trả thứ tự các đỉnh được thăm theo DFS

```
#include <bits/stdc++.h>
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
//Ho va ten: Nguyen Viet Anh
//MSSV: 20215307
    while (!S.empty()) {
        int u = S.top();  // Lấy đỉnh đầu ngăn xếp
        if (!visited[u]) {
            visited[u] = true;  // Đánh dấu đỉnh u đã được thăm
            cout << u << endl;  // In đỉnh u
        }
        if (!adj[u].empty()) {
            int v = adj[u].front();  // Lấy đỉnh kề đầu tiên của u
            adj[u].pop_front();  // Xóa đỉnh kề đầu tiên của u khỏi danh sách
            if (!visited[v]) {
                S.push(v);  // Nếu đỉnh v chưa được thăm, thêm nó vào ngăn xếp
            }
        } else {
            S.pop();  // Nếu tất cả các đỉnh kề của u đã được thăm, loại bỏ u khỏi ngăn xếp
        }
    }
}
```

```
int main() {
    cout << "Ho va ten: Nguyen Viet Anh\n";
    cout << "MSSV:20215307\n";
    // Chưa có mã cụ thể trong hàm main
    return 0;
}
```

Bài 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).

For example:

Test	Result

`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);`
	1
	2
	3
	4
	7
	5

Answer: (penalty regime: 10, 20, ... %)

```
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
//Ho va ten: Nguyen Viet Anh
//MSSV: 20215307
    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();}
    }
}
```

Precheck Check

	Test	Expected	Got	

✔	`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);`
		1
		2
		3
		4
		7
		5
			1
			2
			3
			4
			7
			5
				✔

✔	`int n = 10;`	1	1	✔

Hình 6 Bài 4.6 Hàm trả thứ tự các đỉnh được thăm theo BFS

```
#include <bits/stdc++.h>
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
//Ho va ten:Nguyen Viet Anh
//MSSV: 20215307
    while (!Q.empty()) {
        int u = Q.front();  // Lấy đỉnh đầu hàng đợi
        if (!visited[u]) {
            visited[u] = true;  // Đánh dấu đỉnh u đã được thăm
            cout << u << endl;
        }
        if (!adj[u].empty()) {
            int v = adj[u].front();  // Lấy đỉnh kề đầu tiên của u
            adj[u].pop_front();  // Xóa đỉnh kề đầu tiên của u khỏi danh sách
            if (!visited[v]) {
                Q.push(v);  // Nếu đỉnh v chưa được thăm, thêm nó vào hàng đợi
            }
        } else {
            Q.pop();  // Nếu tất cả đỉnh kề của u đã được thăm, bỏ u khỏi hàng đợi
        }
    }
}
```

```
int main() {
    cout << "Ho va ten: Nguyen Viet Anh\n";
    cout << "MSSV: 20215307\n";
    // Chưa có mã cụ thể trong hàm main
    return 0;
}
```

Bài 4.7. 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 set<T> &a, const set<T> &b) {
    /*****************
    # YOUR CODE HERE #
    *****************/
}
template<class T>
set<T> set_intersection(const set<T> &a, const set<T> &b) {
    /*****************
    # YOUR CODE HERE #
    *****************/
}
```

Lưu ý: Trong ví dụ dưới đây, hàm print_set() là hàm được định nghĩa sẵn như sau:

```
template<class T>
void print_set(const std::set<T> &a) {
    for (const T &x : a) {
        std::cout << x << ' ';
    }
    std::cout << std::endl;
}
```

For example:

Test	Result

`set<int> a = {1, 2, 3, 5, 7};`
`set<int> b = {2, 4, 5, 6, 9};`
`set<int> c = set_union(a, b);`
`set<int> d = set_intersection(a, b);`
 	
`cout << "Union: "; print_set(c);`
`cout << "Intersection: "; print_set(d);`
	`Union: 1 2 3 4 5 6 7 9`
	`Intersection: 2 5`

For example:

Test	Result

set<int> a = {1, 2, 3, 5, 7};
set<int> b = {2, 4, 5, 6, 9};
set<int> c = set_union(a, b);
set<int> d = set_intersection(a, b);
 	
cout << "Union: "; print_set(c);
cout << "Intersection: "; print_set(d);
	Union: 1 2 3 4 5 6 7 9
	Intersection: 2 5

Answer: (penalty regime: 10, 20, ... %)

```
#include <iostream>
#include <set>
using namespace std;

template<class T>
set<T> set_union(const set<T> &a, const set<T> &b) {
    // Hàm tính hợp của hai tập hợp a và b
    set<T> union_set;  // Tập hợp kết quả
    for (T tmp : a) {
        union_set.insert(tmp);  // Thêm các phần tử của tập hợp a vào tập hợp kết quả
    }
    for (T tmp : b) {
        union_set.insert(tmp);  // Thêm các phần tử của tập hợp b vào tập hợp kết quả
    }
    return union_set;  // Trả về tập hợp kết quả
}
//Ho va ten: Nguyen Viet Anh
//MSSV: 20215307
template<class T>
set<T> set_intersection(const set<T> &a, const set<T> &b) {
    // Hàm tính giao của hai tập hợp a và b
```

Precheck Check

	Test	Expected	Got

✔	set<int> a = {1, 2, 3, 5, 7};
	set<int> b = {2, 4, 5, 6, 9};
	set<int> c = set_union(a, b);
	set<int> d = set_intersection(a, b);
	
	cout << "Union: "; print_set(c);
	cout << "Intersection: "; print_set(d);
		Union: 1 2 3 4 5 6 7 9
		Intersection: 2 5
			Union: 1 2 3 4 5 6
			Intersection: 2 5

✔	std::set<int> a = {1, 9, 10, 6, 17, 8}; std::set<int> b = {2, 10, 5, 6, 9, -5, 12, 4, 15, 21};	Union: -5 1 2 4 5 6 8 9 10 12 15 17 21 Intersection: 6 9 10	Union: -5 1 2 4 5 6 Intersection: 6 9 1

Hình 7 Bài 4.7 Viết hàm thực hiện các phép giao và hợp của hai tập hợp 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) {
    // Hàm tính hợp của hai tập hợp a và b
    set<T> union_set;  // Tập hợp kết quả
    for (T tmp : a) {
        union_set.insert(tmp);  // Thêm các phần tử của tập hợp a vào tập hợp kết quả
    }
    for (T tmp : b) {
        union_set.insert(tmp);  // Thêm các phần tử của tập hợp b vào tập hợp kết quả
    }
    return union_set;  // Trả về tập hợp kết quả
}
//Ho va ten: Nguyen Viet Anh
//MSSV: 20215307
template<class T>
set<T> set_intersection(const set<T> &a, const set<T> &b) {
    // Hàm tính giao của hai tập hợp a và b
    set<T> c;  // Tập hợp kết quả
    for (T v : a) {
        int index = 0;
        for (T u : b) {
            if (v == u) {
                index++;  // Nếu phần tử v có trong cả hai tập hợp, tăng biến đếm
                break;
            }
```

```
        }
        if (index != 0) c.insert(v);  // Nếu phần tử v có trong cả hai tập hợp, thêm vào tập hợp kết quả
    }
    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() {
    cout << "Ho va ten: Nguyen Viet Anh\n";
    cout << "MSSV:20215307\n";
    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 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.

```
template<class T>
map<T, double> fuzzy_set_union(const map<T, double> &a, const map<T, double> &b) {
    /*****************
    # YOUR CODE HERE #
    *****************/
}
template<class T>
map<T, double> fuzzy_set_intersection(const map<T, double> &a, const map<T, double> &b) {
    /*****************
    # YOUR CODE HERE #
    *****************/
}
```

Lưu ý: Trong ví dụ dưới đây, hàm print_fuzzy_set() được định nghĩa sẵn như sau:

```
template<class T>
void print_fuzzy_set(const map<T, double> &a) {
    cout << "{ ";
    for (const auto &x : a) {
        cout << "(" << x.first << ", " << x.second << ") ";
    }
    cout << "}";
    cout << endl;
}
```

For example:

Test	Result

`map<int, double> a = {{1, 0.2}, {2, 0.5}, {3, 1}, {4, 0.6}, {5, 0.7}};`
`map<int, double> b = {{1, 0.5}, {2, 0.4}, {4, 0.9}, {5, 0.4}, {6, 1}};`
`cout << "A = "; print_fuzzy_set(a);`
`cout << "B = "; print_fuzzy_set(b);`
`map<int, double> c = fuzzy_set_union(a, b);`
`map<int, double> d = fuzzy_set_intersection(a, b);`
`cout << "Union: "; print_fuzzy_set(c);`
`cout << "Intersection: "; print_fuzzy_set(d);`
	`A = { (1, 0.2) (2, 0.5) (3, 1) (4, 0.6) (5, 0.7) }`
	`B = { (1, 0.5) (2, 0.4) (4, 0.9) (5, 0.4) (6, 1) }`
	`Union: { (1, 0.5) (2, 0.5) (3, 1) (4, 0.9) (5, 0.7) (`
	`Intersection: { (1, 0.2) (2, 0.4) (4, 0.6) (5, 0.4) }`

Test	Result

map<int, double> a = {{1, 0.2}, {2, 0.5}, {3, 1}, {4, 0.6}, {5, 0.7}};
map<int, double> b = {{1, 0.5}, {2, 0.4}, {4, 0.9}, {5, 0.4}, {6, 1}};
cout << "A = "; print_fuzzy_set(a);
cout << "B = "; print_fuzzy_set(b);
map<int, double> c = fuzzy_set_union(a, b);
map<int, double> d = fuzzy_set_intersection(a, b);
cout << "Union: "; print_fuzzy_set(c);
cout << "Intersection: "; print_fuzzy_set(d);
	A = { (1, 0.2) (2, 0.5) (3, 1) (4, 0.6) (5, 0.7) }
	B = { (1, 0.5) (2, 0.4) (4, 0.9) (5, 0.4) (6, 1) }
	Union: { (1, 0.5) (2, 0.5) (3, 1) (4, 0.9) (5, 0.7) (
	Intersection: { (1, 0.2) (2, 0.4) (4, 0.6) (5, 0.4) }

Answer: (penalty regime: 10, 20, ... %)

```
#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) {
    // Hàm tính hợp của hai tập a và b
    map<T, double> c = a;  // Khởi tạo tập hợp kết quả bằng tập hợp a
    for (const auto &e : b) {
        if (c.count(e.first)) {
            c[e.first] = max(e.second, c[e.first]);  // Nếu phần tử e có trong cả hai tập hợp, lấy g
        } else {
            c.insert(e);  // Nếu phần tử e chỉ có trong b, thêm nó vào tập hợp kết quả
        }
    }
    return c;
}
//Ho va ten: Nguyen Viet Anh
//MSSV: 20215307
template<class T>
map<T, double> fuzzy_set_intersection(const map<T, double> &a, const map<T, double> &b) {

```

Precheck Check

	Test	Expected

✔	map<int, double> a = {{1, 0.2}, {2, 0.5}, {3, 1}, {4, 0.6}, {5, 0.7}};
	map<int, double> b = {{1, 0.5}, {2, 0.4}, {4, 0.9}, {5, 0.4}, {6, 1}};
	cout << "A = "; print_fuzzy_set(a);
	cout << "B = "; print_fuzzy_set(b);
	map<int, double> c = fuzzy_set_union(a, b);
	map<int, double> d = fuzzy_set_intersection(a, b);
	cout << "Union: "; print_fuzzy_set(c);
	cout << "Intersection: "; print_fuzzy_set(d);
		A = { (1, 0.2) (2, 0.5)
		B = { (1, 0.5) (2, 0.4)
		Union: { (1, 0.5) (2, 0
		Intersection: { (1, 0.2

✔	map<int, double> a = {{-1, 0.2}, {2, 0.65}, {3, 1}, {4, 0.6}, {5, 0.75}, {1, 0.7}, {10, 0.1}};
	map<int, double> b = {{1, 0.15}, {2, 0.14}, {4, 0.9}, {5, 0.41}, {6, 1}};
	cout << "A = "; print_fuzzy_set(a);
	cout << "B = "; print_fuzzy_set(b);
	map<int, double> c = fuzzy_set_union(a, b);
	map<int, double> d = fuzzy_set_intersection(a, b);
	cout << "Union: "; print_fuzzy_set(c);
	cout << "Intersection: "; print_fuzzy_set(d);
		A = { (-1, 0.2) (1, 0.7
		B = { (1, 0.15) (2, 0.1
		Union: { (-1, 0.2) (1,
		Intersection: { (1, 0.1

Passed all tests! ✔

Hình 8 Bài 4.8 Viết hàm thực hiện các phép giao và hợp của hai tập hợp biểu diễn bằng map

```
#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)
{
    // Hàm tính hợp của hai tập a và b
    map<T, double> c = a;  // Khởi tạo tập hợp kết quả bằng tập hợp a
    for (const auto &e : b) {
        if (c.count(e.first)) {
            c[e.first] = max(e.second, c[e.first]);  // Nếu phần tử e có trong cả hai tập hợp, lấy giá trị lớn hơn
        } else {
            c.insert(e);  // Nếu phần tử e chỉ có trong b, thêm nó vào tập hợp kết quả
        }
    }
    return c;
}
//Ho va ten: Nguyen Viet Anh
//MSSV: 20215307
template<class T>
map<T, double> fuzzy_set_intersection(const map<T, double> &a, const map<T, double> &b) {
    // Hàm tính giao của hai tập a và b
    map<T, double> c;  // Khởi tạo tập hợp kết quả
    for (const auto &x : a) {
        const auto it = b.find(x.first);  // Tìm phần tử x trong b
```

```
        if (it != b.end()) {
            c[x.first] = min(x.second, it->second);  // Nếu phần tử x có trong cả hai tập hợp, lấy giá trị nhỏ hơn
        }
    }
    return c;  // Trả về tập hợp kết quả
}

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 << ") ";  // In từng phần tử của tập mờ a
    }
    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);
```

```
    return 0;
}
```

Nguyễn Việt Anh - 20215307

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

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.

```
vector<int> dijkstra(const vector< vector< pair<int, int> > >&adj) {
    /*****************
    # YOUR CODE HERE #
    *****************/
}
```

For example:

Test	Result

`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 (unsigned int i = 0; i < distance.size(); ++i) {`
`  cout << "distance " << 0 << "->" << i << " = " << distance[i] << endl;`
`}`
	`distance 0->0 = 0`
	`distance 0->1 = 4`
	`distance 0->2 = 12`
	`distance 0->3 = 19`
	`distance 0->4 = 21`
	`distance 0->5 = 11`
	`distance 0->6 = 9`
	`distance 0->7 = 8`
	`distance 0->8 = 14`

Answer: (penalty regime: 10, 20, ... %)

```cpp
#include <iostream>
#include <queue>
#include <climits>
using namespace std;

//Ho va ten: Nguyen Viet Anh
//MSSV: 20215307
vector<int> dijkstra(const vector<vector<pair<int, int>>>& adj) {
    // Hàm tính đường đi ngắn nhất từ đỉnh 0 đến các đỉnh khác trong đồ thị bằng thuật toán Dijkstra
    priority_queue<pair<int, int>> S;  // Khởi tạo hàng đợi ưu tiên để lưu các đỉnh cần duyệt
    vector<int> d(adj.size(), INT_MAX);  // Khởi tạo vector khoảng cách, ban đầu đặt tất cả khoảng c
    d[0] = 0;  // Khoảng cách từ đỉnh 0 đến chính nó là 0
    S.push({0, 0});  // Đẩy đỉnh 0 vào hàng đợi với khoảng cách 0

    while (!S.empty()) {
        int du = -S.top().first;  // Lấy khoảng cách âm để chuyển thành khoảng cách dương
        int u = S.top().second;  // Lấy đỉnh đầu hàng đợi
        S.pop();  // Loại bỏ đỉnh đầu hàng đợi

        if (du != d[u]) continue;  // Bỏ qua nếu khoảng cách hiện tại không còn tối ưu

```

Buổi 3-Tuần 13 - IT3040 – 2023.2 – 738922

Precheck Check

	Test	Expected	Got	

✔	`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 (unsigned int i = 0; i < distance.size(); ++i) {`
	`  cout << "distance " << 0 << "->" << i << " = " << distance[i] << endl;`
	`}`
		`distance 0->0 = 0`
		`distance 0->1 = 4`
		`distance 0->2 = 12`
		`distance 0->3 = 19`
		`distance 0->4 = 21`
		`distance 0->5 = 11`
		`distance 0->6 = 9`
		`distance 0->7 = 8`
		`distance 0->8 = 14`
			`distance 0->0 = 0`
			`distance 0->1 = 4`
			`distance 0->2 = 12`
			`distance 0->3 = 19`
			`distance 0->4 = 21`
			`distance 0->5 = 11`
			`distance 0->6 = 9`
			`distance 0->7 = 8`
			`distance 0->8 = 14`
				✔

✔	`int n = 10;`
	`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, 2);`
	`add_edge(0, 7, 3);`
	`add_edge(1, 7, 15);`
	`add_edge(1, 2, 8);`
	`add_edge(1, 8, 38);`
	`add_edge(2, 3, 2);`
	`add_edge(2, 8, 12);`
	`add_edge(3, 4, 9);`
	`add_edge(3, 5, 4);`
	`add_edge(4, 5, 7);`
	`add_edge(5, 6, 2);`
	`add_edge(5, 9, 2);`
	`add_edge(6, 7, 1);`
	`add_edge(6, 8, 6);`
	`add_edge(7, 8, 7);`
	`add_edge(7, 9, 71);`
	`add_edge(7, 5, 17);`
	
	`vector<int> distance = dijkstra(adj);`
	`for (unsigned int i = 0; i < distance.size(); ++i) {`
	`  cout << "distance " << 0 << "->" << i << " = " << distance[i] << endl;`
	`}`
		`distance 0->0 = 0`
		`distance 0->1 = 2`
		`distance 0->2 = 10`
		`distance 0->3 = 10`
		`distance 0->4 = 13`
		`distance 0->5 = 6`
		`distance 0->6 = 4`
		`distance 0->7 = 3`
		`distance 0->8 = 10`
		`distance 0->9 = 8`
			`distance 0->0 = 0`
			`distance 0->1 = 2`
			`distance 0->2 = 10`
			`distance 0->3 = 10`
			`distance 0->4 = 13`
			`distance 0->5 = 6`
			`distance 0->6 = 4`
			`distance 0->7 = 3`
			`distance 0->8 = 10`
			`distance 0->9 = 8`
				✔

Passed all tests! ✔

```
#include <iostream>
#include <queue>
#include <climits>
using namespace std;

//Ho va ten: Nguyen Viet Anh
//MSSV: 20215307
vector<int> dijkstra(const vector<vector<pair<int, int>>>& adj) {
    // Hàm tính đường đi ngắn nhất từ đỉnh 0 đến các đỉnh khác trong đồ thị bằng thuật toán Dijkstra
    priority_queue<pair<int, int>> S;  // Khởi tạo hàng đợi ưu tiên để lưu các đỉnh cần duyệt
    vector<int> d(adj.size(), INT_MAX);  // Khởi tạo vector khoảng cách, ban đầu đặt tất cả khoảng cách là vô cùng
    d[0] = 0;  // Khoảng cách từ đỉnh 0 đến chính nó là 0
    S.push({0, 0});  // Đẩy đỉnh 0 vào hàng đợi với khoảng cách 0

    while (!S.empty()) {
        int du = -S.top().first;  // Lấy khoảng cách âm để chuyển thành khoảng cách dương
        int u = S.top().second;  // Lấy đỉnh đầu hàng đợi
        S.pop();  // Loại bỏ đỉnh đầu hàng đợi

        if (du != d[u]) continue;  // Bỏ qua nếu khoảng cách hiện tại không còn tối ưu

        for (auto e : adj[u]) {  // Duyệt tất cả các đỉnh kề của u
            int v = e.first;  // Đỉnh kề
            int c = e.second;  // Trọng số cạnh
            if (d[v] > d[u] + c) {  // Nếu tìm được đường đi ngắn hơn
                d[v] = d[u] + c;  // Cập nhật khoảng cách
```

```
                S.push({-d[v], v});  // Đẩy đỉnh v vào hàng đợi với khoảng cách mới
            }
        }
    }
    return d;  // Trả về vector khoảng cách từ đỉnh 0 đến các đỉnh khác
}
```

BÀI TẬP VỀ NHÀ

Chụp ảnh kết quả của tất cả các test.

Bài tập 10: Search Engine

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

Cho NN văn bản và QQ 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:

- f(t,d)f(t,d) là số lần xuất hiện của từ tt trong văn bản dd
- maxf(d)maxf(d) là giá trị lớn nhất của f(t,d)f(t,d) với mọi tt
- df(t)df(t) là số văn bản chứa từ tt
- TF(t,d)=0.5+0.5·f(t,d)maxf(t,d)TF(t,d)=0.5+0.5·f(t,d)maxf(t,d)
- IDF(t)=log2(Ndf(t))IDF(t)=log2(Ndf(t))
- Điểm số của từ tt trong văn

  bản dd là score(t,d)=TF(t,d)·IDF(t)score(t,d)=TF(t,d)·IDF(t), nếu từ tt không xuất hiện trong văn bản dd thì score(t,d)=0score(t,d)=0.
- Điểm số của văn bản dd đối với truy vấn gồm các từ (có thể trùng

  nhau) t1,t2,...,tqt1,t2,...,tq là ∑qi=1score(ti,d)∑i=1qscore(ti,d)

Ta coi văn bản có điểm số càng cao thì càng khớp với truy vấn.

Input:

- Dòng đầu tiên chứa số NN
- Dòng thứ ii trong NN dòng tiếp theo thể hiện văn bản ii, mỗi dòng là một dãy các từ ngăn cách nhau bởi dấu phẩy
- Dòng tiếp theo chứa số QQ
- Dòng thứ ii trong QQ dòng tiếp theo thể hiện truy vấn thứ ii, mỗi dòng là một dãy các từ ngăn cách nhau bởi dấu phẩy

Output: Gồm QQ dòng, dòng thứ ii là chỉ số của văn bản khớp với truy vấn thứ ii nhất. Nếu có nhiều văn bản có điểm số bằng nhau, in ra văn bản có chỉ số nhỏ nhất.

Ví dụ:

Input:

5

k,k,ow

bb,ar,h

qs,qs,qs

d,bb,q,d,rj

ow

5

h,d,d,qs,q,q,ar

qs,qs

hc,d,ow,d,qs

ow,wl,hc,k

q,hc,q,d,hc,q

Output:

4

3

4

1

4

Giới hạn:

- N≤1000N≤1000
- Q≤1000Q≤1000
- Số từ trong mỗi văn bản không quá 10001000
- Số từ trong mỗi truy vấn không quá 1010
- Độ dài mỗi từ không quá 1010

Tham khảo:

- https://en.wikipedia.org/wiki/Tf%E2%80%93idf

Desktop > lab4-4.10(HW).cpp > ...

```
19  void input() {
39      }
40
41      // Tách chuỗi tài liệu và từ cần tìm thành các từ đơn
42      for(string d : doc_raw) {
43          doc.push_back(split_string(d));
44      }
45      for(string w : word_raw) {
46          word.push_back(split_string(w));
47      }
48  }
49  //Ho va ten: Nguyen Viet Anh
50  //MSSV: 20215307
51
52  // Hàm tách chuỗi thành các từ đơn
53  vector<string> split_string(string str) {
54      vector<string> split_stringg;
55
56      while (!str.empty()) {
57          string tmp = str.substr(0, str.find(','')); // Tách chuỗi đến dấu phẩy
58          int pos = str.find(' '); // Tìm vị trí dấu cách
59          int position = str.find(' ');
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Ho va ten: Nguyen Viet Anh
MSSV: 20215307
5
k,k,ow
bb,ar,h
qs,qs,qs
d,bb,q,d,rj
ow
5
h,d,d,qs,q,q,ar
qs,qs
hc,d,ow,d,qs
ow,wl,hc,k
q,hc,q,d,hc,q
4
3
4
1
4
PS D:\Desktop\output>
```

```
#include<bits/stdc++.h>
using namespace std;

// Biến toàn cục
int n, q;
vector<vector<string>> doc; // Lưu trữ danh sách tài liệu
vector<vector<string>> word; // Lưu trữ danh sách từ cần tìm kiếm
vector<int> f_max; // Lưu trữ tần số từ lớn nhất trong từng tài liệu
map<string, int> df; // Lưu trữ số tài liệu chứa từ khóa
map<pair<string, int>, int> fe; // Lưu trữ tần số xuất hiện của từ trong tài liệu

// Hàm tách chuỗi thành các từ
vector<string> split_string(string str);

// Hàm xử lý tiền xử lý dữ liệu
void pre();

// Hàm nhập dữ liệu
void input() {
    vector<string> doc_raw; // Lưu trữ tạm thời các tài liệu
    vector<string> word_raw; // Lưu trữ tạm thời các từ cần tìm

    cin >> n; // Nhập số lượng tài liệu
    string tmp;
    getline(cin, tmp); // Đọc bỏ dòng trống

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

```
        getline(cin, tmp); // Đọc từng tài liệu
        doc_raw.push_back(tmp);
    }

    cin >> q; // Nhập số lượng từ cần tìm
    getline(cin, tmp); // Đọc bỏ dòng trống
    for(int i = 0; i < q; i++) {
        string tmp;
        getline(cin, tmp); // Đọc từng từ cần tìm
        word_raw.push_back(tmp);
    }

    // Tách chuỗi tài liệu và từ cần tìm thành các từ đơn
    for(string d : doc_raw) {
        doc.push_back(split_string(d));
    }
    for(string w : word_raw) {
        word.push_back(split_string(w));
    }
}
//Ho va ten: Nguyen Viet Anh
//MSSV: 20215307

// Hàm tách chuỗi thành các từ đơn
vector<string> split_string(string str) {
    vector<string> split_stringg;

    while (!str.empty()) {
```

```
        string tmp = str.substr(0, str.find(',')); // Tách chuỗi đến dấu phẩy
        int pos = str.find(' '); // Tìm vị trí dấu cách
        int position = str.find(',');
        if(pos > tmp.size())
            split_stringg.push_back(tmp); // Thêm từ vào vector
        else {
            while(pos <= tmp.size()) {
                tmp.erase(pos, 1); // Xóa dấu cách
                pos = tmp.find(" ");
            }
        }
        if(position > str.size()) {
            break;
        } else {
            str.erase(0, position + 1); // Xóa phần đã tách khỏi chuỗi gốc
        }
    }

    return split_stringg;
}

// Hàm tính tần số xuất hiện của từ trong tài liệu
int fre(string word_s, int i) {
    if(fe.find({word_s, i}) != fe.end()) {
        return fe[{word_s, i}];
    }

    int index = 0;
```

```
    vector<string> str_tmp = doc[i];

    for(string v : str_tmp) {
        if(word_s == v) index++;
    }

    fe.insert({{word_s, i}, index});
    return index;
}

// Hàm đếm số lượng tài liệu chứa từ
int count(string word_use) {
    if(df.find(word_use) != df.end()) {
        return df[word_use];
    }
    int index = 0;
    for(vector<string> str_tmp : doc) {
        vector<string>::iterator ite = find(str_tmp.begin(), str_tmp.end(), word_use);
        if(ite != str_tmp.end()) {
            index++;
        }
    }

    df.insert({word_use, index});
    return index;
}

// Hàm tiền xử lý dữ liệu
```

```
void pre() {
    for(vector<string> word_str : doc) {
        map<string, int> m;
        int max_f = 0;
        for(string word_tmp : word_str) {
            map<string, int>::iterator ite = m.find(word_tmp);
            if(ite == m.end()) {
                m.insert({word_tmp, 1});
            } else {
                ite->second += 1;
            }
            max_f = max(m[word_tmp], max_f);
        }
        f_max.push_back(max_f); // Thêm tần số lớn nhất vào vector
    }
}

// Hàm tìm kiếm tài liệu phù hợp nhất với danh sách từ cần tìm
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 = doc[i];
        double score = 0;
        for(string word_tmp : list_word) {
            if(find(list_word_train_doc.begin(), list_word_train_doc.end(), word_tmp) ==
list_word_train_doc.end()) { // từ này không xuất hiện trong văn bản
                continue;
```

```
            } else {
                int ftd = fre(word_tmp, i);
                int dft = count(word_tmp);
                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() {
    cout << "Ho va ten: Nguyen Viet Anh\n";
    cout << "MSSV: 20215307\n";
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    input(); // Nhập dữ liệu
    pre(); // Tiền xử lý dữ liệu
    int i = 0;
    while(i < q) {
        cout << search_engine(word[i]) << endl; // In kết quả tìm kiếm cho từng từ
        i++;
```

```
    }
    return 0;
}
```

Case 1

Desktop > lab4-4.10(HW).cpp > ...

```
131    int search_engine(vector<string> list_word) {
152            }
153        }
154        return predict_label + 1;
155    }
156
157    int main() {
158        cout << "Ho va ten: Nguyen Viet Anh\n";
159        cout << "MSSV: 20215307\n";
160        ios_base::sync_with_stdio(false);
161        cin.tie(NULL);
162        input(); // Nhập dữ liệu
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
86
13
53
9
38
41
41
35
32
57
62
70
43
52
35
94
7
32
62
92
47
16
20
40
62
93
82
42
62
PS D:\Desktop\output>
```

Case 2

Desktop > lab4-4.10(HW).cpp > ...

```
131    int search_engine(vector<string> list_word) {
152            }
153        }
154        return predict_label + 1;
155    }
156
157    int main() {
158        cout << "Ho va ten: Nguyen Viet Anh\n";
159        cout << "MSSV: 20215307\n";
160        ios_base::sync_with_stdio(false);
161        cin.tie(NULL);
162        input(); // Nhập dữ liệu
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
110
114
179
48
195
179
71
120
61
22
19
143
110
11
101
51
184
7
185
187
171
87
73
35
19
57
56
106
38
PS D:\Desktop\output>
```

Case 3

Desktop > lab4-4.10(HW).cpp > ...

```
131  int search_engine(vector<string> list_word) {
152          }
153      }
154      return predict_label + 1;
155  }
156
157  int main() {
158      cout << "Ho va ten: Nguyen Viet Anh\n";
159      cout << "MSSV: 20215307\n";
160      ios_base::sync_with_stdio(false);
161      cin.tie(NULL);
162      input(); // Nhập dữ liệu
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
272
213
144
162
42
38
103
300
280
150
47
149
272
38
84
129
2
239
171
50
24
171
13
275
21
268
102
57
263
PS D:\Desktop\output>
```

Case 4

Desktop > lab4-4.10(HW).cpp > ...

```
131    int search_engine(vector<string> list_word) {
134        for(int i = 0; i < n; i++) {
135            vector<string> list_word_train_doc = doc[i];
136            double score = 0;
137            for(string word_tmp : list_word) {
138                if(find(list_word_train_doc.begin(), list_word_train_doc.end(),
139                    continue;
140                } else {
141                    int ftd = fre(word_tmp, i);
142                    int dft = count(word_tmp);
143                    int maxfd = f_max[i];
144                    double tf_word = 0.5 + 0.5 * ((double) ftd / maxfd);
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
81
13
52
64
98
26
3
189
293
11
345
272
11
116
4
191
213
252
187
267
73
203
15
83
29
216
183
193
297
PS D:\Desktop\output>
```

Case 5

Desktop > lab4-4.10(HW).cpp > ...

```
131    int search_engine(vector<string> list_word) {
134        for(int i = 0; i < n; i++) {
135            vector<string> list_word_train_doc = doc[i];
136            double score = 0;
137            for(string word_tmp : list_word) {
138                if(find(list_word_train_doc.begin(), list_word_train_doc.end(), word_
139                    continue;
140                } else {
141                    int ftd = fre(word_tmp, i);
142                    int dft = count(word_tmp);
143                    int maxfd = f_max[i];
144                    double tf_word = 0.5 + 0.5 * ((double) ftd / maxfd);
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
123
401
339
54
175
214
433
393
470
75
37
391
98
421
136
210
185
367
84
20
416
84
115
17
165
100
168
50
220
PS D:\Desktop\output>
```

Bài tập 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 a_i 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 k_i tên giặc. Giả sử đoạn tường thứ i có x_i lính. Khi đó nếu số tên giặc không vượt quá x_i thì không có tên giặc nào lọt vào được qua đoạn tường này. Ngược lại sẽ có $a_i - x_i * k_i$ 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.

Dữ liệu vào:

Dòng thứ nhất chứa các số nguyên n và s ($1 \leq n \leq 100000; 1 \leq s \leq 10^9$).

n dòng tiếp theo chứa hai số nguyên ai và ki lần lượt là số tên giặc tấn công đoạn tường thứ i và khả năng chống trả của một lính ở đoạn tường thứ i ($1 \leq a_i, k_i \leq 10^9$).

Kết quả:

Ghi ra một số nguyên duy nhất là số lượng tên giặc tối thiểu có thể lọt vào lâu đài.

Ví dụ:

Dữ liệu vào	**Kết quả**
3 3 4 2 1 1 10 8	3
1 10 8 1	0

Desktop > lab4-4.11(HW).cpp > ...

```cpp
// Cấu trúc so sánh để sắp xếp tường trong hàng đợi
struct compare {
    bool operator() (wall a, wall b) {
        int ra, rb;

        // Tính giá trị ưu tiên của tường a
        if (a.ai <= a.ki) ra = a.ai;
        else ra = a.ki;

        // Tính giá trị ưu tiên của tường b
        if (b.ai <= b.ki) rb = b.ai;
        else rb = b.ki;

        return ra < rb; // So sánh giá trị ưu tiên để sắp xếp
    }
};

// Ho va ten: Nguyen Viet Anh
// MSSV: 20215307
int n, s;
priority_queue<wall, vector<wall>, compare> now; // Hàng đợi ưu tiên chứa các tường
int tong_dich = 0, dich_da_giet = 0; // Tổng số địch và số địch đã bị tiêu diệt

// Hàm thực hiện thuật toán xử lý tường
void algo() {
    while (!now.empty() && s > 0) {
        wall a = now.top(); // Lấy tường có độ ưu tiên cao nhất
        now.pop(); // Loại bỏ tường này khỏi hàng đợi ưu tiên
        if (a.ai <= a.ki) {
            // Nếu số lượng địch ít hơn hoặc bằng khả năng tiêu diệt của tường
            dich_da_giet += a.ai; // Toàn bộ địch bị tiêu diệt
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

```
PS D:\Desktop> cd 'd:\Desktop\output'
PS D:\Desktop\output> & .\'lab4-4.11(HW).exe'
Ho va ten: Nguyen Viet Anh
MSSV: 20215307
3 3
4 2
1 1
10 8
3
PS D:\Desktop\output>
```

```
#include <iostream>
#include <queue>
#include <vector>
#include <map>
using namespace std;

// Định nghĩa cấu trúc lưu trữ thông tin của một wall
struct wall {
    int ai; // Số lượng địch
    int ki; // Khả năng tiêu diệt của tường

    wall(int ai, int ki) {
        this->ai = ai;
        this->ki = ki;
    }
};

// Cấu trúc so sánh để sắp xếp tường trong hàng đợi
struct compare {
    bool operator() (wall a, wall b) {
        int ra, rb;

        // Tính giá trị ưu tiên của tường a
        if (a.ai <= a.ki) ra = a.ai;
        else ra = a.ki;

        // Tính giá trị ưu tiên của tường b
        if (b.ai <= b.ki) rb = b.ai;
```

```
        else rb = b.ki;

        return ra < rb; // So sánh giá trị ưu tiên để sắp xếp
    }
};

// Ho va ten: Nguyen Viet Anh
// MSSV: 20215307
int n, s;
priority_queue<wall, vector<wall>, compare> now; // Hàng đợi ưu tiên chứa các tường
int tong_dich = 0, dich_da_giet = 0; // Tổng số địch và số địch đã bị tiêu diệt

// Hàm thực hiện thuật toán xử lý tường
void algo() {
    while (!now.empty() && s > 0) {
        wall a = now.top(); // Lấy tường có độ ưu tiên cao nhất
        now.pop(); // Loại bỏ tường này khỏi hàng đợi ưu tiên
        if (a.ai <= a.ki) {
            // Nếu số lượng địch ít hơn hoặc bằng khả năng tiêu diệt của tường
            dich_da_giet += a.ai; // Toàn bộ địch bị tiêu diệt
        } else {
            // Nếu số lượng địch lớn hơn khả năng tiêu diệt của tường
            int now_enemy = a.ai - a.ki; // Số địch còn lại sau khi tiêu diệt
            now.push(wall(now_enemy, a.ki)); // Đẩy lại số địch còn lại vào hàng đợi với tường có cùng khả năng tiêu diệt
            dich_da_giet += a.ki; // Cập nhật số địch bị tiêu diệt
        }
        s -= 1; // Giảm số lượng lượt tấn công còn lại
```

```
    }
}

// Hàm nhập dữ liệu
void input() {
    cin >> n >> s; // Nhập số lượng tường và số lượt tấn công
    int i = 0;
    while (i < n) {
        int ai, ki;
        cin >> ai >> ki; // Nhập số địch và khả năng tiêu diệt của từng tường
        now.push(wall(ai, ki)); // Đẩy tường vào hàng đợi ưu tiên
        tong_dich += ai; // Cập nhật tổng số địch
        i++;
    }
}

int main() {
    cout << "Ho va ten: Nguyen Viet Anh\n";
    cout << "MSSV: 20215307\n";
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    input(); // Nhập dữ liệu
    algo(); // Thực hiện thuật toán
    cout << tong_dich - dich_da_giet; // In ra số địch còn lại chưa bị tiêu diệt
}
```

Case 1

Desktop > lab4-4.11(HW).cpp > main()

```
19  struct compare {
20      bool operator() (wall a, wall b) {
30
31          return ra < rb; // So sánh giá trị ưu tiên để sắp xếp
32      }
33  };
34
35  // Ho va ten: Nguyen Viet Anh
36  // MSSV: 20215307
37  int n, s;
38  priority_queue<wall, vector<wall>, compare> now; // Hàng đợi ưu tiên chứa các tường
39  int tong_dich = 0, dich_da_giet = 0; // Tổng số địch và số địch đã bị tiêu diệt
40
41  // Hàm thực hiện thuật toán xử lý tường
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Desktop\output> & .\'lab4-4.11(HW).exe'
Ho va ten: Nguyen Viet Anh
MSSV: 20215307
23 2
10 46
13 26
5 56
20 97
1 22
17 63
13 42
7 75
15 87
6 48
7 16
19 40
11 47
19 14
15 67
10 6
23 86
10 36
20 23
7 12
13 14
22 13
2 75
242
PS D:\Desktop\output>
```

Case 2

Desktop > lab4-4.11(HW).cpp > main()

```
19  struct compare {
20      bool operator() (wall a, wall b) {
30
31          return ra < rb; // So sánh giá trị ưu tiên để sắp xếp
32      }
33  };
34
35  // Ho va ten: Nguyen Viet Anh
36  // MSSV: 20215307
37  int n, s;
38  priority_queue<wall, vector<wall>, compare> now; // Hàng đợi ưu tiên chứa các tường
39  int tong_dich = 0, dich_da_giet = 0; // Tổng số địch và số địch đã bị tiêu diệt
40
41  // Hàm thực hiện thuật toán xử lý tường
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
25 100
54 25
4 14
48 41
55 5
57 57
42 44
52 98
3 87
42 79
7 13
35 34
38 87
36 38
36 52
7 67
3 59
44 42
40 5
26 16
51 68
49 67
22 76
17 85
21 38
14 35
24 52
506
PS D:\Desktop\output>
```

Case 3

Desktop > lab4-4.11(HW).cpp > main()

```
19  struct compare {
20      bool operator() (wall a, wall b) {
30
31          return ra < rb; // So sánh giá trị ưu tiên để sắp xếp
32      }
33  };
34
35  // Ho va ten: Nguyen Viet Anh
36  // MSSV: 20215307
37  int n, s;
38  priority_queue<wall, vector<wall>, compare> now; // Hàng đợi ưu tiên chứa các tường
39  int tong_dich = 0, dich_da_giet = 0; // Tổng số địch và số địch đã bị tiêu diệt
40
41  // Hàm thực hiện thuật toán xử lý tường
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
28 97
32 52
56 96
82 7
88 42
40 89
62 96
61 54
76 75
74 58
7 59
40 82
17 37
45 67
18 1
79 36
75 85
95 65
82 92
63 7
16 55
33 84
3 79
22 81
88 61
63 99
62 77
0
PS D:\Desktop\output>
```

Case 4

Desktop > lab4-4.11(HW).cpp > main()

```
19  struct compare {
20      bool operator() (wall a, wall b) {
30
31          return ra < rb; // So sánh giá trị ưu tiên để sắp xếp
32      }
33  };
34
35  // Ho va ten: Nguyen Viet Anh
36  // MSSV: 20215307
37  int n, s;
38  priority_queue<wall, vector<wall>, compare> now; // Hàng đợi ưu tiên chứa các tường
39  int tong_dich = 0, dich_da_giet = 0; // Tổng số địch và số địch đã bị tiêu diệt
40
41  // Hàm thực hiện thuật toán xử lý tường
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
11 1
12 1
46 2
91 1
72 1
18 2
68 1
71 1
62 1
52 2
92 2
100 1
84 2
96 2
63 2
70 2
26 1
38 1
60 2
28 2
45 2
43 1
18 2
14 1
54 1
97 1
62 2
1476
PS D:\Desktop\output>
```

Case 5

Desktop > lab4-4.11(HW).cpp > main()

```
19  struct compare {
20      bool operator() (wall a, wall b) {
30
31          return ra < rb; // So sánh giá trị ưu tiên để sắp xếp
32      }
33  };
34
35  // Ho va ten: Nguyen Viet Anh
36  // MSSV: 20215307
37  int n, s;
38  priority_queue<wall, vector<wall>, compare> now; // Hàng đợi ưu tiên chứa các tường
39  int tong_dich = 0, dich_da_giet = 0; // Tổng số địch và số địch đã bị tiêu diệt
40
41  // Hàm thực hiện thuật toán xử lý tường
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
100 1
100 1
100 1
100 1
100 1
100 1
100 1
100 1
100 1
100 1
100 1
100 1
100 1
100 1
100 1
100 1
100 1
100 1
100 1
100 1
100 1
100 1
100 1
100 1
100 1
0
PS D:\Desktop\output>
```

Bài tập 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 $h_1,h_2,\ldots,h_n$. Hãy xác định hình chữ nhật có diện tích lớn nhất có thể tạo thành từ các cột liên tiếp.

Dữ liệu vào:
Dòng thứ nhất chứa số nguyên dương n ($1 \le n \le 10^6$). Dòng thứ hai chứa n số nguyên không âm $h_1,h_2,\ldots,h_n$ cách nhau bởi dấu cách ($0 \le h_i \le 10^9$).

Kết quả: In ra số nguyên duy nhất là diện tích hình chữ nhật lớn nhất có thể tạo thành từ các cột liên tiếp của lược đồ.

Ví dụ:

Dữ liệu vào	Kết quả
7 6 2 5 4 5 1 6	12

Desktop > lab4-4.12(HW).cpp > ...

```cpp
//Ho va ten: Nguyen Viet Anh
//MSSV: 20215307
// Hàm tính diện tích lớn nhất của hình chữ nhật trong histogram
int dienTichLonNhatTrongHistogram(vector<int> histogram) {
    stack<int> s;
    int dienTichLonNhat = 0;
    int dienTich = 0;
    int i = 0;
    int size = histogram.size();

    // Duyệt từng phần tử để tính cận trái và phải
    while (i < size) {
        if (s.empty() || histogram[s.top()] <= histogram[i]) {
            // Thêm cận trái của phần tử tiếp theo
            s.push(i);
            // Tăng chỉ số để xét phần tử tiếp theo
            i++;
        } else {
            // Tìm cận phải của phần tử thứ i-1 trong stack
            int top = s.top();
            s.pop();
            // Tính số ô theo 2 cận trái và phải
            dienTich = histogram[top] * (s.empty() ? i : (i - s.top() - 1));
            // Cập nhật diện tích lớn nhất nếu tìm được giá trị tốt hơn
            if (dienTich > dienTichLonNhat) {
                dienTichLonNhat = dienTich;
            }
        }
    }

    // Xử lý các phần tử còn lại trong stack
    while (!s.empty()) {
        int top = s.top();
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Desktop> cd 'd:\Desktop\output'
PS D:\Desktop\output> & .\'lab4-4.12(HW).exe'
Ho va ten: Nguyen Viet Anh
MSSV:20215307
7
6 2 5 4 5 1 6
12
PS D:\Desktop\output>
```

```
#include <bits/stdc++.h>
using namespace std;
//Ho va ten: Nguyen Viet Anh
//MSSV: 20215307
// Hàm tính diện tích lớn nhất của hình chữ nhật trong histogram
int dienTichLonNhatTrongHistogram(vector<int> histogram) {
    stack<int> s;
    int dienTichLonNhat = 0;
    int dienTich = 0;
    int i = 0;
    int size = histogram.size();

    // Duyệt từng phần tử để tính cận trái và phải
    while (i < size) {
        if (s.empty() || histogram[s.top()] <= histogram[i]) {
            // Thêm cận trái của phần tử tiếp theo
            s.push(i);
            // Tăng chỉ số để xét phần tử tiếp theo
            i++;
        } else {
            // Tìm cận phải của phần tử thứ i-1 trong stack
            int top = s.top();
            s.pop();
            // Tính số ô theo 2 cận trái và phải
            dienTich = histogram[top] * (s.empty() ? i : (i - s.top() - 1));
```

```
            // Cập nhật diện tích lớn nhất nếu tìm được giá trị tốt hơn
            if (dienTich > dienTichLonNhat) {
                dienTichLonNhat = dienTich;
            }
        }
    }

    // Xử lý các phần tử còn lại trong stack
    while (!s.empty()) {
        int top = s.top();
        s.pop();
        dienTich = histogram[top] * (s.empty() ? i : (i - s.top() - 1));
        if (dienTich > dienTichLonNhat) {
            dienTichLonNhat = dienTich;
        }
    }

    // Trả về diện tích lớn nhất
    return dienTichLonNhat;
}
//Ho va ten: Nguyen Viet Anh
//MSSV: 20215307
int main() {
    cout << "Ho va ten: Nguyen Viet Anh\n";
    cout << "MSSV:20215307\n";
    int n, in;
    cin >> n;
    vector<int> chieuCao;
```

```
    // Nhập danh sách chiều cao của các cột trong histogram
    for (int i = 0; i < n; ++i) {
        cin >> in;
        chieuCao.push_back(in);
    }

    // In ra diện tích lớn nhất của hình chữ nhật trong histogram
    cout << dienTichLonNhatTrongHistogram(chieuCao) << endl;
    return 0;
}
```

Case 1

Desktop > lab4-4.12(HW).cpp > dienTichLonNhatTrongHistogram(vector<int>)

```cpp
//Ho va ten: Nguyen Viet Anh
//MSSV: 20215307
// Hàm tính diện tích lớn nhất của hình chữ nhật trong histogram
int dienTichLonNhatTrongHistogram(vector<int> histogram) {
    stack<int> s;
    int dienTichLonNhat = 0;
    int dienTich = 0;
    int i = 0;
    int size = histogram.size();

    // Duyệt từng phần tử để tính cận trái và phải
    while (i < size) {
        if (s.empty() || histogram[s.top()] <= histogram[i]) {
            // Thêm cận trái của phần tử tiếp theo
            s.push(i);
            // Tăng chỉ số để xét phần tử tiếp theo
            i++;
        } else {
            // Tìm cận phải của phần tử thứ i-1 trong stack
            int top = s.top();
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Ho va ten: Nguyen Viet Anh
MSSV:20215307
7
6 2 5 4 5 1 6
12
PS D:\Desktop\output> cd 'd:\Desktop\output'
PS D:\Desktop\output> & .\'lab4-4.12(HW).exe'
Ho va ten: Nguyen Viet Anh
MSSV:20215307
10000
3368 3433 2649 2068 2548 3618 779 231 1185 4459 331 1938 4244 724 287 1048 3033 1
21 3570 3771 508 3059 316 4644 3882 1019 4108 1540 4840 391 3833 3477 1328 2587 1
 2249 242 4589 3135 2898 990 2883 2823 3542 2459 2053 1292 798 1523 264 2807 1888
20 4194 409 1211 3808 3971 2665 3528 3627 2783 4615 149 2293 150 4134 1323 4902 5
4177 2822 1547 35 2186 4474 159 963 473 1070 3678 1456 232 2230 2178 1883 967 209
 2091 1989 4845 1672 2355 1768 3435 2215 425 4859 4311 1478 2278 961 77 600 2501
0 2009 2435 3149 1661 27 4247 254 4551 593 902 2657 4098 3570 3503 836 61 2847 47
1380 4548 3960 3150 1172 3094 1286 4839 4571 630 2703 4159 2432 3990 2503 4531 74
35 3274 4069 3173 3703 3237 400 3268 3093 2523 2849 1335 162 2475 2741 2742 2334
31 4075 1000 4177 702 3545 669 2759 4103 4548 1347 3138 43 2530 2972 3845 2677 31
8 205 2579 3011 988 171 4538 624 2714 3134 2129 4097 723 921 1857 3870 2693 4524
70 309 3506 1842 4717 446 2670 4413 1986 1490 2580 3163 1494 3733 2052 4322 1192
```

Case 2

lab4-4.12(HW).cpp

Desktop > lab4-4.12(HW).cpp > ...

```cpp
#include <bits/stdc++.h>
using namespace std;
//Ho va ten: Nguyen Viet Anh
//MSSV: 20215307
// Hàm tính diện tích lớn nhất của hình chữ nhật trong histogram
int dienTichLonNhatTrongHistogram(vector<int> histogram) {
    stack<int> s;
    int dienTichLonNhat = 0;
    int dienTich = 0;
    int i = 0;
    int size = histogram.size();

    // Duyệt từng phần tử để tính cận trái và phải
    while (i < size) {
        if (s.empty() || histogram[s.top()] <= histogram[i]) {
            // Thêm cận trái của phần tử tiếp theo
            s.push(i);
            // Tăng chỉ số để xét phần tử tiếp theo
            i++;
        } else {
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
706 2977 3737 864 4106 1142 929 107 1810 629 1623 4440 4541 2131 2567 3614 2103 1504 2054 384
333 2792 1053 3744 1539 459 753 964 2851 4144 1601 4921 1876 2619 516 3963 525 57 494 1621 191
3236 1614 1274 1420 2216 55 1530 3643 1772 445 4113 1069 924 2309 3488 1406 1226 2884 3277 131
1000 4046 3912 3329 335 2482 1122 4764 2135 3868 74 3719 3021 1927 3601 4390 4802 2122 4568 11
942 2166 123 3508 2091 3924 3434 3543 3677 1293 430 3549 3385 3809 3818 4073 4319 1542 3323 13
727 926 14 4998 1857 4854 1108 181 4155 4343 3379 894 2286 4790 4743 917 2819 57 1641 4221 230
 1331 3354 849 3623 2964 1942 4884 200 1280 1753 4571 981 2604 4526 1717 3386 4685 602 4651 14
4310 4153 2233 713 2941 1539 972 2388 161 4639 1539 2475 567 3709 282 2395 2447 625 2642 186 1
 2293 2056 2166 3623 325 3551 2923 1360 3266 927 2043 268 3343 4123 2850 1787 1672 475 1676 43
7 1431 4147 352 2753 3729 1031 1670 2355 1132 3325 3943 1240 3846 2636 4205 368 3111 1050 162
3772 1560 56 594 860 4883 2291 25 4229 4482 4126 4529 4977 2578 3123 4508 4639 1733 2188 1030
1947 4807 2448 3000 4733 1987 4794 3262 2763 475 1312 2472 1425 2858 3291 354 1532 453 2360 16
425 2821 373 2046 4716 4588 3163 2236 684 2833 4576 1937 4128 3991 3803 2955 1813 4740 1890 46
9 4242 169 1451 2083 801 356 1620 1644 4162 964 366 3785 1358 4578 1711 259 1421 4402 2764 344
 1513 279 3759 1489 1140 2368 4926 1681 4700 3091 3268 1081 906 367 2429 60 4948 1701 4385 301
430 1027 1259 2775 147 2441 2499 1851 1821 2095 1550 191 2184 2483 2901 412 2152 3887 2039 686
1994 3153 2690 4004 746 1147 2504 1580 1028 1307 847 4918 610 2091 835 1997 316 4379 2051 2286
2468 3844 1233 1433 2122 1775 1754 3651 2810 3698 790 3760 188 3284 4764 1229 430 108 1395 368
2247 1937 2632 1519 3664 3086 2355 1942 2833 4859 193 982 3313 1165 2113 3864 2347 2815 3833 1
1 795 728 1148 2636 2611 308 694 3771 2596 2757 4121 1243 1199 1160 1687 936 2161 1504 2156 28
8 3416 4123 2847 4839 2506 1617 1346 4842 1549 3898 2587 4026 2356 3162 226 3051 4286 1310 424
41684
```

Bài 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.

Dữ liệu vào:
Một dòng duy nhất chứa một xâu nhị phân độ dài n ($1 \leq n \leq 10^6$).

Kết quả:
Ghi ra một số nguyên duy nhất là số lượng xâu con có số ký tự 0 và số ký tự 1 bằng nhau.

Ví dụ:

Dữ liệu vào	Kết quả
1001011	8

Desktop > lab4-4.13(HW).cpp > main()

```cpp
//Ho va ten: Nguyen Viet Anh
//MSSV: 20215307
// Hàm đếm số lượng xâu con có số lượng '0' và '1' bằng nhau
int dem_xau_con(const string& s) {
    int n = s.length();
    unordered_map<int, int> dem_chenhlech;
    dem_chenhlech[0] = 1; // Khởi tạo cho trường hợp chênh lệch bằng 0

    int chenhlech = 0; // Chênh lệch giữa số lượng '0' và '1'
    int result = 0; // Kết quả: số lượng xâu con hợp lệ

    // Duyệt qua từng ký tự trong xâu nhị phân
    for (int i = 0; i < n; ++i) {
        if (s[i] == '0') {
            chenhlech--; // Gặp '0', giảm chênh lệch
        } else {
            chenhlech++; // Gặp '1', tăng chênh lệch
        }

        // Kiểm tra xem giá trị chênh lệch đã xuất hiện trước đó bao nhiêu lần
        if (dem_chenhlech.find(chenhlech) != dem_chenhlech.end()) {
            result += dem_chenhlech[chenhlech]; // Cộng số lần xuất hiện vào kết quả
        }

        dem_chenhlech[chenhlech]++; // Cập nhật số lần xuất hiện của giá trị chênh lệ
    }

    return result; // Trả về kết quả
}

int main() {
    cout << "Ho va ten: Nguyen Viet Anh\n";
    cout << "MSSV: 20215307\n";
    string s;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Desktop\output> & .\'lab4-4.13(HW).exe'
Ho va ten: Nguyen Viet Anh
MSSV: 20215307
1001011
8
PS D:\Desktop\output>
```

```
#include <iostream>
#include <unordered_map>
#include <vector>
using namespace std;

//Ho va ten: Nguyen Viet Anh
//MSSV: 20215307
// Hàm đếm số lượng xâu con có số lượng '0' và '1' bằng nhau
int dem_xau_con(const string& s) {
    int n = s.length();
    unordered_map<int, int> dem_chenhlech;
    dem_chenhlech[0] = 1; // Khởi tạo cho trường hợp chênh lệch bằng 0

    int chenhlech = 0; // Chênh lệch giữa số lượng '0' và '1'
    int result = 0; // Kết quả: số lượng xâu con hợp lệ

    // Duyệt qua từng ký tự trong xâu nhị phân
    for (int i = 0; i < n; ++i) {
        if (s[i] == '0') {
            chenhlech--; // Gặp '0', giảm chênh lệch
        } else {
            chenhlech++; // Gặp '1', tăng chênh lệch
        }

        // Kiểm tra xem giá trị chênh lệch đã xuất hiện trước đó bao nhiêu lần
        if (dem_chenhlech.find(chenhlech) != dem_chenhlech.end()) {
```

```
            result += dem_chenhlech[chenhlech]; // Cộng số lần xuất hiện vào kết quả
        }

        dem_chenhlech[chenhlech]++; // Cập nhật số lần xuất hiện của giá trị chênh lệch hiện tại
    }

    return result; // Trả về kết quả
}

int main() {
    cout << "Ho va ten: Nguyen Viet Anh\n";
    cout << "MSSV: 20215307\n";
    string s;
    cin >> s; // Nhập xâu nhị phân từ người dùng

    cout << dem_xau_con(s) << endl; // In ra số lượng xâu con hợp lệ

    return 0;
}
```

Case 1

Desktop > lab4-4.13(HW).cpp > main()

```cpp
//Ho va ten: Nguyen Viet Anh
//MSSV: 20215307
// Hàm đếm số lượng xâu con có số lượng '0' và '1' bằng nhau
int dem_xau_con(const string& s) {
    int n = s.length();
    unordered_map<int, int> dem_chenhlech;
    dem_chenhlech[0] = 1; // Khởi tạo cho trường hợp chênh lệch bằng 0

    int chenhlech = 0; // Chênh lệch giữa số lượng '0' và '1'
    int result = 0; // Kết quả: số lượng xâu con hợp lệ

    // Duyệt qua từng ký tự trong xâu nhị phân
    for (int i = 0; i < n; ++i) {
        if (s[i] == '0') {
            chenhlech--; // Gặp '0', giảm chênh lệch
        } else {
            chenhlech++; // Gặp '1', tăng chênh lệch
        }

        // Kiểm tra xem giá trị chênh lệch đã xuất hiện trước đó bao nhiêu lần
        if (dem_chenhlech.find(chenhlech) != dem_chenhlech.end()) {
            result += dem_chenhlech[chenhlech]; // Cộng số lần xuất hiện vào kết quả
        }

        dem_chenhlech[chenhlech]++; // Cập nhật số lần xuất hiện của giá trị chênh lệch
    }

    return result; // Trả về kết quả
}

int main() {
    cout << "Ho va ten: Nguyen Viet Anh\n";
    cout << "MSSV: 20215307\n";
    string s;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Desktop> cd 'd:\Desktop\output'
PS D:\Desktop\output> & .\'lab4-4.13(HW).exe'
Ho va ten: Nguyen Viet Anh
MSSV: 20215307
10011001101011010101000111111111100001010001101010001001001010011111010011001010101011011110111110111
583
```

Case 2

Desktop > lab4-4.13(HW).cpp > main()

```cpp
//Ho va ten: Nguyen Viet Anh
//MSSV: 20215307
// Hàm đếm số lượng xâu con có số lượng '0' và '1' bằng nhau
int dem_xau_con(const string& s) {
    int n = s.length();
    unordered_map<int, int> dem_chenhlech;
    dem_chenhlech[0] = 1; // Khởi tạo cho trường hợp chênh lệch bằng 0

    int chenhlech = 0; // Chênh lệch giữa số lượng '0' và '1'
    int result = 0; // Kết quả: số lượng xâu con hợp lệ

    // Duyệt qua từng ký tự trong xâu nhị phân
    for (int i = 0; i < n; ++i) {
        if (s[i] == '0') {
            chenhlech--; // Gặp '0', giảm chênh lệch
        } else {
            chenhlech++; // Gặp '1', tăng chênh lệch
        }

        // Kiểm tra xem giá trị chênh lệch đã xuất hiện trước đó bao nhiêu lần
        if (dem_chenhlech.find(chenhlech) != dem_chenhlech.end()) {
            result += dem_chenhlech[chenhlech]; // Cộng số lần xuất hiện vào kế
        }

        dem_chenhlech[chenhlech]++; // Cập nhật số lần xuất hiện của giá trị ch
    }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Desktop> cd 'd:\Desktop\output'
PS D:\Desktop\output> & .\'lab4-4.13(HW).exe'
Ho va ten: Nguyen Viet Anh
MSSV: 20215307
11110101100100011000000100011111001111000010001100000011000001110100110111101100000011110111
11000000001100000110101110101100101111101111011000000101010001100011001001110011000010100111
10101110100001100000111010011110100000100011010000010100001110001000101000101101111111011010
01011011011000001000000001000001011010010111011110000011011101111111100100111011010111110111
10111011100000001100001111011111001000111110011101010100011110111101011000001110000000101001
10010110001110111100001110001010110110100110100010001100110101100110100100000001101010011010
00101001110101100000010010110111111100000
14342
```

Case 3

Desktop > lab4-4.13(HW).cpp > ...

```cpp
//Ho va ten: Nguyen Viet Anh
//MSSV: 20215307
// Hàm đếm số lượng xâu con có số lượng '0' và '1' bằng nhau
int dem_xau_con(const string& s) {
    int n = s.length();
    unordered_map<int, int> dem_chenhlech;
    dem_chenhlech[0] = 1; // Khởi tạo cho trường hợp chênh lệch bằng 0

    int chenhlech = 0; // Chênh lệch giữa số lượng '0' và '1'
    int result = 0; // Kết quả: số lượng xâu con hợp lệ

    // Duyệt qua từng ký tự trong xâu nhị phân
    for (int i = 0; i < n; ++i) {
        if (s[i] == '0') {
            chenhlech--; // Gặp '0', giảm chênh lệch
        } else {
            chenhlech++; // Gặp '1', tăng chênh lệch
        }

        // Kiểm tra xem giá trị chênh lệch đã xuất hiện trước đó bao nhiêu lần
        if (dem_chenhlech.find(chenhlech) != dem_chenhlech.end()) {
            result += dem_chenhlech[chenhlech]; // Cộng số lần xuất hiện vào ké
        }

        dem_chenhlech[chenhlech]++; // Cập nhật số lần xuất hiện của giá trị ch
    }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Desktop> cd 'd:\Desktop\output'
PS D:\Desktop\output> & .\'lab4-4.13(HW).exe'
Ho va ten: Nguyen Viet Anh
MSSV: 20215307
```

Case 4

Desktop > lab4-4.13(HW).cpp > main()

```cpp
#include <iostream>
#include <unordered_map>
#include <vector>
using namespace std;

//Ho va ten: Nguyen Viet Anh
//MSSV: 20215307
// Hàm đếm số lượng xâu con có số lượng '0' và '1' bằng nhau
int dem_xau_con(const string& s) {
    int n = s.length();
    unordered_map<int, int> dem_chenhlech;
    dem_chenhlech[0] = 1; // Khởi tạo cho trường hợp chênh lệch bằng 0

    int chenhlech = 0; // Chênh lệch giữa số lượng '0' và '1'
    int result = 0; // Kết quả: số lượng xâu con hợp lệ

    // Duyệt qua từng ký tự trong xâu nhị phân
    for (int i = 0; i < n; ++i) {
        if (s[i] == '0') {
            chenhlech--; // Gặp '0', giảm chênh lệch
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
1101010111011111111111011000001001010001110011001110000100001111010000010111100111010110101111
1001111101000100101011110011001011111011011001100111010100001100000110110100000011111010001110
0101111101001101100010100100100000100111111011101111010100101000101111001000100010011000110101
1010101111110000100101101001111000111111111011111100100101111110111110110011001000101010001111
0010010010000101100101001110000010000111111110011011100000011101101111100110011001000101010111
1100011011010110111111100111111011111101000011010110001010010111011111101101011001101100101011
1011100100110000101101010011001000000000010011110001100110110011000100011001010111011010001111
0011011110000110100010110100000010101110110001111111111010011001111010000100000100011111000001
1101101001010100001000101001100110010011111010010110110001110101011100011100111100001010001111
0110001110101101101001100001101100000111011000001010100111100110101010100110110101011101010010
1011111001110001011011000010100001100100011110001010110101110000010100101000011100000101110110
1001100011111110110011000010110000110000000100101101111101101010000101001010100111100010100000
0010101001000100111011111011000100110110100100000000100010100110100011011111011000111100000010
1001101010011110011110100101010100100010001101110001010011111010010000010100000100110010001111
0100110011001000000110110101110011001111110000111010010110011000100011100110100011001111110100
1101110010100001110000101101011010100010000001011011011000010111100000100001010011101111111000
1100010010010000011011111011100101001111001000011110110101111000100000001011010110110110000101
0111001111110101010000101110011011000100100010100111000110110011111111110011010011101010100010
0100110101001010110110001100001111001111110111001101011111100010010011001000001111010110110011
0011011001001001101000001111111000101000110100100010111001110110000011100001110001010010011011
124022
PS D:\Desktop\output> 281665459
```