

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

Nguyễn Việt Anh - 20215307

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

Hình 1 Bài 4.1 Đảo ngược danh sách liên kết đơn.....	3
Hình 2 Bài 4.2 Viết hàm tính diện tích tam giác	5
Hình 3 Bài 4.3 Viết hàm tính tích có hướng của 2 vector	13
Hình 4 Bài 4.4 Viết hàm xóa phần tử, sắp xếp vector	9
Hình 5 Bài 4.5 Hàm trả thứ tự các đỉnh được thăm theo DFS	11
Hình 6 Bài 4.6 Hàm trả thứ tự các đỉnh được thăm theo BFS	15
Hình 7 Bài 4.7 Viết hàm thực hiện các phép giao và hợp của hai tập hợp được biểu diễn bằng set	126
Hình 8 Bài 4.8 Viết hàm thực hiện các phép giao và hợp của hai tập hợp được biểu diễn bằng map.....	21

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 Việt 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	Original list: 50 12 7 6 4 2 7 5 4 -1 Reversed list: -1 4 5 7 2 4 6 7 12 50
✓	1 6	Original list: 6 Reversed list: 6	Original list: 6 Reversed list: 6
✓	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	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

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 Việt 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;  
  
}
```

Nguyễn Việt Anh - 20215307

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

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

```
3 #include <iomanip>
4 #include <utility>
5 using namespace std;
6 using Point = pair<double, double>;
7
8 double distance(Point a, Point b) {
9     // Hàm tính khoảng cách giữa hai điểm a và b
10    return sqrt((a.first - b.first)*(a.first - b.first) + (a.second - b.second)*(a.second -
11    });
12
13 double area(Point a, Point b, Point c) {
14     // Hàm tính diện tích tam giác được tạo bởi ba điểm a, b, c
15     double ab = distance(a, b); // Tính độ dài cạnh AB
16     double bc = distance(b, c); // Tính độ dài cạnh BC
17     double ca = distance(c, a); // Tính độ dài cạnh CA
18     double p = (ab + bc + ca) / 2; // Tính nửa chu vi tam giác
19     double area = sqrt(p * (p - ab) * (p - bc) * (p - ca)); // Sử dụng công thức Heron
20     return area;
21 }
22 // Ho va ten: Nguyen Viet Anh
23 // MSSV: 20215307
```

Precheck

Check

	Test	Expected	Got	
✓	<pre>cout << setprecision(2) << fixed; cout << area({1, 2}, {2.5, 10}, {15, -5.25}) << endl;</pre>	61.44	61.44	✓
✓	<pre>cout << setprecision(2) << fixed; cout << area({1, 2.5}, {2.5, 15}, {-5.2, -5.75}) << endl;</pre>	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;  
}
```

Nguyễn Việt Anh - 20215307

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>;

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, ... %)

```
1 #include <iostream>
2 #include <cmath>
3 #include <iomanip>
4 using namespace std;
5 using Vector = tuple<double, double, double>;
6
7 Vector cross_product(Vector a, Vector b) {
8     // Hàm tính tích chéo của hai vector a và b
9     double x, y, z;
10    z = get<0>(a) * get<1>(b) - get<0>(b) * get<1>(a); // Thành phần z của tích chéo
11    x = get<1>(a) * get<2>(b) - get<1>(b) * get<2>(a); // Thành phần x của tích chéo
12    y = get<2>(a) * get<0>(b) - get<2>(b) * get<0>(a); // Thành phần y của tích chéo
13    return Vector(x, y, z); // Trả về vector kết quả
14 }
15 //Ho va ten: Nguyen Viet Anh
16 //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	Odd elements of a: 3 7 -5
2 3 6 7 -5	Odd elements of b: 13 5 9 35
13 5 2 4 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, ... %)

```

1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 using namespace std;
5
6 void print_vector(const vector<int> &a) {
7     for (int v : a) cout << v << ' ';
8     cout << endl;
9 }
10
11 void delete_even(vector<int> &a) {
12     // Hàm xóa các phần tử chẵn khỏi vector a
13     a.erase(remove_if(a.begin(), a.end(), [](int a) {
14         return (a % 2 == 0); // Kiểm tra phần tử có phải là số chẵn không
15     }), a.end()); // Xóa các phần tử chẵn
16 }
17
18 void sort_decrease(vector<int> &a) {
19     // Hàm sắp xếp vector a giảm dần
20     sort(a.rbegin(), a.rend());
21 }
22

```

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, ... %)

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 void dfs(vector<list<int>> adj) {
5     stack<int> S;
6     vector<bool> visited(adj.size());
7     S.push(1); // Bắt đầu từ đỉnh số 1
8     //Ho va ten: Nguyen Viet Anh
9     //MSSV: 20215307
10    while (!S.empty()) {
11        int u = S.top(); // Lấy đỉnh đầu ngăn xếp
12        if (!visited[u]) {
13            visited[u] = true; // Đánh dấu đỉnh u đã được thăm
14            cout << u << endl; // In đỉnh u
15        }
16        if (!adj[u].empty()) {
17            int v = adj[u].front(); // Lấy đỉnh kề đầu tiên của u
18            adj[u].pop_front(); // Xóa đỉnh kề đầu tiên của u khỏi danh sách
19            if (!visited[v]) {
20                S.push(v); // Nếu đỉnh v chưa được thăm, thêm nó vào ngăn xếp
21            }
22        } else {
23            S.pop();
24        }
25    }
26 }

```

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
<pre>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);</pre>	<pre>1 2 3 4 7 5</pre>

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

```
2 using namespace std;
3
4 void bfs(vector< list<int> > adj) {
5     queue<int> Q;
6     vector<bool> visited(adj.size());
7     Q.push(1); // Bắt đầu từ đỉnh số 1
8     //Ho va ten: Nguyen Viet Anh
9     //MSSV: 20215307
10    while (!Q.empty()) {
11        int u=Q.front();
12        if (!visited[u]){
13            visited[u] = true;
14            std::cout<< u << std::endl;
15        }
16        if (!adj[u].empty()){
17            int v=adj[u].front(); adj[u].pop_front();
18            if(!visited[v]){
19                Q.push(v);
20            }
21        }else { Q.pop();}
22    }
23 }
```

Precheck

Check

	Test	Expected	Got	
✓	<pre>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);</pre>	<pre>1 2 3 4 7 5</pre>	<pre>1 2 3 4 7 5</pre>	✓
✓	<pre>int n = 10; vector< list<int> > adj;</pre>	<pre>1 2</pre>	<pre>1 2</pre>	✓

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
<pre>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);</pre>	<pre>Union: 1 2 3 4 5 6 7 9 Intersection: 2 5</pre>

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

```

1 #include <iostream>
2 #include <set>
3 using namespace std;
4
5 template<class T>
6 set<T> set_union(const set<T> &a, const set<T> &b) {
7     // Hàm tính hợp của hai tập hợp a và b
8     set<T> union_set; // Tập hợp kết quả
9     for (T tmp : a) {
10         union_set.insert(tmp); // Thêm các phần tử của tập hợp a vào tập hợp kết quả
11     }
12     for (T tmp : b) {
13         union_set.insert(tmp); // Thêm các phần tử của tập hợp b vào tập hợp kết quả
14     }
15     return union_set; // Trả về tập hợp kết quả
16 }
17 //Ho va ten: Nguyen Viet Anh
18 //MSSV: 20215307
19 template<class T>
20 set<T> set_intersection(const set<T> &a, const set<T> &b) {
21     // Hàm tính giao của hai tập hợp a và b
22

```

Precheck

Check

	Test	Expected	Got
✓	<pre>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);</pre>	<pre>Union: 1 2 3 4 5 6 7 9 Intersection: 2 5</pre>	<pre>Union: 1 2 3 4 5 6 Intersection: 2 5</pre>
✓	<pre>std::set<int> a = {1, 9, 10, 6, 17, 8}; std::set<int> b = {2, 10, 5, 6, 9, -5, 12, 4, 15, 21};</pre>	<pre>Union: -5 1 2 4 5 6 8 9 10 12 15 17 21 Intersection: 6 9 10</pre>	<pre>Union: -5 1 2 4 5 6 Intersection: 6 9 10</pre>

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
<pre>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);</pre>	<pre>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) (6, 1) } Intersection: { (1, 0.2) (2, 0.4) (4, 0.6) (5, 0.4) }</pre>



Test	Result
<pre>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);</pre>	<pre>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) (6, 1) } Intersection: { (1, 0.2) (2, 0.4) (4, 0.6) (5, 0.4) }</pre>

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

```
1 #include <iostream>
2 #include <map>
3 using namespace std;
4
5 template<class T>
6 map<T, double> fuzzy_set_union(const map<T, double> &a, const map<T, double> &b) {
7     // Hàm tính hợp của hai tập a và b
8     map<T, double> c = a; // Khởi tạo tập hợp kết quả bằng tập hợp a
9     for (const auto &e : b) {
10         if (c.count(e.first)) {
11             c[e.first] = max(e.second, c[e.first]); // Nếu phần tử e có trong cả hai tập hợp, lấy giá trị lớn hơn
12         } else {
13             c.insert(e); // Nếu phần tử e chỉ có trong b, thêm nó vào tập hợp kết quả
14         }
15     }
16     return c;
17 }
18 //Ho va ten: Nguyen Viet Anh
19 //MSSV: 20215307
20 template<class T>
21 map<T, double> fuzzy_set_intersection(const map<T, double> &a, const map<T, double> &b) {
22     // Hàm tính giao của hai tập a và b
23     map<T, double> c;
```

Precheck

Check

	Test	Expected
✓	<pre>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);</pre>	<pre>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) (6, 1) } Intersection: { (1, 0.2) (2, 0.4) (4, 0.6) (5, 0.4) }</pre>
✓	<pre>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);</pre>	<pre>A = { (-1, 0.2) (1, 0.7) (2, 0.65) (3, 1) (4, 0.6) (5, 0.75) (6, 1) (10, 0.1) } B = { (1, 0.15) (2, 0.14) (4, 0.9) (5, 0.41) (6, 1) } Union: { (-1, 0.2) (1, 0.7) (2, 0.65) (3, 1) (4, 0.9) (5, 0.75) (6, 1) (10, 0.1) } Intersection: { (1, 0.15) (2, 0.14) }</pre>

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;  
}
```

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
<pre>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 " << i << "->" << i << " = " << distance[i] << endl; }</pre>	<pre>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</pre>

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

```

1  #include <iostream>
2  #include <queue>
3  #include <climits>
4  using namespace std;
5
6  //Ho va ten: Nguyen Viet Anh
7  //MSSV: 20215307
8  vector<int> dijkstra(const vector<vector<pair<int, int>>>& adj) {
9      // Hàm tính đường đi ngắn nhất từ đỉnh 0 đến các đỉnh khác trong đồ thị bằng thuật toán Dijkstra
10     priority_queue<pair<int, int>> S; // Khởi tạo hàng đợi ưu tiên để lưu các đỉnh cần duyệt
11     vector<int> d(adj.size(), INT_MAX); // Khởi tạo vector khoảng cách, ban đầu đặt tất cả khoảng c
12     d[0] = 0; // Khoảng cách từ đỉnh 0 đến chính nó là 0
13     S.push({0, 0}); // Đẩy đỉnh 0 vào hàng đợi với khoảng cách 0
14
15     while (!S.empty()) {
16         int du = -S.top().first; // Lấy khoảng cách âm để chuyển thành khoảng cách dương
17         int u = S.top().second; // Lấy đỉnh đầu hàng đợi
18         S.pop(); // Loại bỏ đỉnh đầu hàng đợi
19
20         if (du != d[u]) continue; // Bỏ qua nếu khoảng cách hiện tại không còn tối ưu
21
22         for (int v : adj[u]) {
23             int w = adj[u][v].second;
24             if (d[v] > du + w) {
25                 d[v] = du + w;
26                 S.push({-d[v], v});
27             }
28         }
29     }
30     return d;
31 }
```

Precheck

Check

Test	Expected	Got	
<div>✓</div> <pre> 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 " << i << "->" << i << " = " << distance[i] << endl; } </pre>	<pre> 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 </pre>	<pre> 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 </pre>	<div>✓</div>
<div>✓</div> <pre> 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 " << i << "->" << i << " = " << distance[i] << endl; } </pre>	<pre> 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 </pre>	<pre> 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 </pre>	<div>✓</div>

Passed all tests! ✓

```
#include <iostream>
#include <queue>
#include <limits>
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:

- $f(t,d)$ là số lần xuất hiện của từ t trong văn bản d
- $\max_d f(t,d)$ là giá trị lớn nhất của $f(t,d)$ với mọi t
- $df(t)$ là số văn bản chứa từ t
- $TF(t,d) = 0.5 + 0.5 \cdot f(t,d) / \max_d f(t,d)$
- $IDF(t) = \log_2(N / df(t))$
- Điểm số của từ t trong văn

bản d là $score(t,d) = TF(t,d) \cdot IDF(t)$, nếu từ t không xuất hiện trong văn bản d thì $score(t,d) = 0$.

- Điểm số của văn bản d đối với truy vấn gồm các từ (có thể trùng nhau) t_1, t_2, \dots, t_q là $\sum_{i=1}^q score(t_i, d)$

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

Input:

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

Output: Gồm QQ dòng, dòng thứ i là chỉ số của văn bản khớp với truy vấn thứ i 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:

- $N \leq 1000$
- $Q \leq 1000$
- Số từ trong mỗi văn bản không quá 1000
- 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 > G+ 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('\n');

```

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 > G+ 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
163     ... // ...

```

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 > G+ 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
163     output(); // Xuất 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 > G+ 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
163     // ...

```

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 > G+ lab4-4.10(HW).cpp > ...
131 int search_engine(vector<string> list_word) {
132     int n = list_word.size();
133     for(int i = 0; i < n; i++) {
134         vector<string> list_word_train_doc = doc[i];
135         double score = 0;
136         for(string word_tmp : list_word) {
137             if(find(list_word_train_doc.begin(), list_word_train_doc.end(),
138                 word_tmp) != 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);
145             }
146         }
147     }
148 }

```

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 > G+ lab4-4.10(HW).cpp > ...
131 int search_engine(vector<string> list_word) {
132     int n = list_word.size();
133     for(int i = 0; i < n; i++) {
134         vector<string> list_word_train_doc = doc[i];
135         double score = 0;
136         for(string word_tmp : list_word) {
137             if(find(list_word_train_doc.begin(), list_word_train_doc.end(), word_tmp) != list_word_train_doc.end())
138                 continue;
139             } else {
140                 int ftd = fre(word_tmp, i);
141                 int dft = count(word_tmp);
142                 int maxfd = f_max[i];
143                 double tf_word = 0.5 + 0.5 * ((double) ftd / maxfd);
144             }
145         }
146     }
147 }
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2
```

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 a_i và k_i 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 > G+ lab4-4.11(HW).cpp > ...
18 // Cấu trúc so sánh để sắp xếp tường trong hàng đợi
19 struct compare {
20     bool operator() (wall a, wall b) {
21         int ra, rb;
22
23         // Tính giá trị ưu tiên của tường a
24         if (a.ai <= a.ki) ra = a.ai;
25         else ra = a.ki;
26
27         // Tính giá trị ưu tiên của tường b
28         if (b.ai <= b.ki) rb = b.ai;
29         else rb = b.ki;
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
42 void algo() {
43     while (!now.empty() && s > 0) {
44         wall a = now.top(); // Lấy tường có độ ưu tiên cao nhất
45         now.pop(); // Loại bỏ tường này khỏi hàng đợi ưu tiên
46         if (a.ai <= a.ki) {
47             // Nếu số lượng địch ít hơn hoặc bằng khả năng tiêu diệt của tường
48             dich_da_giet += a.ai; // Toàn bộ địch bị tiêu diệt
49         }
50     }
51 }
52
53 int main() {
54     // Đọc dữ liệu
55     cin >> n;
56     for (int i = 0; i < n; i++) {
57         wall w;
58         cin >> w.ai >> w.ki;
59         now.push(w);
60     }
61     cin >> s;
62     algo();
63     cout << tong_dich << endl;
64     return 0;
65 }

```

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			
147	6			

```

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, \dots, 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 ($1 \leq n \leq 10^6$). Dòng thứ hai chứa n số nguyên không âm h_1, h_2, \dots, h_n cách nhau bởi dấu cách ($0 \leq h_i \leq 10^9$).

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 > G+ lab4-4.12(HW).cpp > ...
3 //Ho va ten: Nguyen Viet Anh
4 //MSSV: 20215307
5 // Hàm tính diện tích lớn nhất của hình chữ nhật trong histogram
6 int dienTichLonNhatTrongHistogram(vector<int> histogram) {
7     stack<int> s;
8     int dienTichLonNhat = 0;
9     int dienTich = 0;
10    int i = 0;
11    int size = histogram.size();
12
13    // Duyệt từng phần tử để tính cận trái và phải
14    while (i < size) {
15        if (s.empty() || histogram[s.top()] <= histogram[i]) {
16            // Thêm cận trái của phần tử tiếp theo
17            s.push(i);
18            // Tăng chỉ số để xét phần tử tiếp theo
19            i++;
20        } else {
21            // Tìm cận phải của phần tử thứ i-1 trong stack
22            int top = s.top();
23            s.pop();
24            // Tính số ô theo 2 cận trái và phải
25            dienTich = histogram[top] * (s.empty() ? i : (i - s.top() - 1));
26            // Cập nhật diện tích lớn nhất nếu tìm được giá trị tốt hơn
27            if (dienTich > dienTichLonNhat) {
28                dienTichLonNhat = dienTich;
29            }
30        }
31    }
32
33    // Xử lý các phần tử còn lại trong stack
34    while (!s.empty()) {
35        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;
}

//Họ và tên: Nguyen Viet Anh
//MSSV: 20215307

int main() {
    cout << "Họ và tên: 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 > G+ lab4-4.12(HW).cpp > dienTichLonNhatTrongHistogram(vector<int>)
3 //Ho va ten: Nguyen Viet Anh
4 //MSSV: 20215307
5 // Hàm tính diện tích lớn nhất của hình chữ nhật trong histogram
6 int dienTichLonNhatTrongHistogram(vector<int> histogram) {
7     stack<int> s;
8     int dienTichLonNhat = 0;
9     int dienTich = 0;
10    int i = 0;
11    int size = histogram.size();
12
13    // Duyệt từng phần tử để tính cận trái và phải
14    while (i < size) {
15        if (s.empty() || histogram[s.top()] <= histogram[i]) {
16            // Thêm cận trái của phần tử tiếp theo
17            s.push(i);
18            // Tăng chỉ số để xét phần tử tiếp theo
19            i++;
20        } else {
21            // Tìm cận phải của phần tử thứ i-1 trong stack
22            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 X
Desktop > lab4-4.12(HW).cpp > ...
1  #include <bits/stdc++.h>
2  using namespace std;
3  //Ho va ten: Nguyen Viet Anh
4  //MSSV: 20215307
5  // Hàm tính diện tích lớn nhất của hình chữ nhật trong histogram
6  int dienTichLonNhatTrongHistogram(vector<int> histogram) {
7      stack<int> s;
8      int dienTichLonNhat = 0;
9      int dienTich = 0;
10     int i = 0;
11     int size = histogram.size();
12
13     // Duyệt từng phần tử để tính cận trái và phải
14     while (i < size) {
15         if (s.empty() || histogram[s.top()] <= histogram[i]) {
16             // Thêm cận trái của phần tử tiếp theo
17             s.push(i);
18             // Tăng chỉ số để xét phần tử tiếp theo
19             i++;
20         } else {
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()
6 //Ho va ten: Nguyen Viet Anh
7 //MSSV: 20215307
8 // Hàm đếm số lượng xâu con có số lượng '0' và '1' bằng nhau
9 int dem_xau_con(const string& s) {
10     int n = s.length();
11     unordered_map<int, int> dem_chenhlech;
12     dem_chenhlech[0] = 1; // Khởi tạo cho trường hợp chênh lệch bằng 0
13
14     int chenhlech = 0; // Chênh lệch giữa số lượng '0' và '1'
15     int result = 0; // Kết quả: số lượng xâu con hợp lệ
16
17     // Duyệt qua từng ký tự trong xâu nhị phân
18     for (int i = 0; i < n; ++i) {
19         if (s[i] == '0') {
20             chenhlech--; // Gặp '0', giảm chênh lệch
21         } else {
22             chenhlech++; // Gặp '1', tăng chênh lệch
23         }
24
25         // Kiểm tra xem giá trị chênh lệch đã xuất hiện trước đó bao nhiêu lần
26         if (dem_chenhlech.find(chenhlech) != dem_chenhlech.end()) {
27             result += dem_chenhlech[chenhlech]; // Cộng số lần xuất hiện vào kết quả
28         }
29
30         dem_chenhlech[chenhlech]++; // Cập nhật số lần xuất hiện của giá trị chênh lệch
31     }
32
33     return result; // Trả về kết quả
34 }
35
36 int main() {
37     cout << "Ho va ten: Nguyen Viet Anh\n";
38     cout << "MSSV: 20215307\n";
39     string s;
40     // Nhập xâu nhị phân
41     cin >> s;
42
43     // Gọi hàm đếm số lượng xâu con có số lượng '0' và '1' bằng nhau
44     int result = dem_xau_con(s);
45
46     // In kết quả
47     cout << "Số lượng xâu con có số lượng '0' và '1' bằng nhau là: " << result << endl;
48
49     return 0;
50 }

```

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()) {
```


Nguyễn Việt Anh - 20215307

```
    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()

6 //Ho va ten: Nguyen Viet Anh
7 //MSSV: 20215307
8 // Hàm đếm số lượng xâu con có số lượng '0' và '1' bằng nhau
9 int dem_xau_con(const string& s) {
10     int n = s.length();
11     unordered_map<int, int> dem_chenhlech;
12     dem_chenhlech[0] = 1; // Khởi tạo cho trường hợp chênh lệch bằng 0
13
14     int chenhlech = 0; // Chênh lệch giữa số lượng '0' và '1'
15     int result = 0; // Kết quả: số lượng xâu con hợp lệ
16
17     // Duyệt qua từng ký tự trong xâu nhị phân
18     for (int i = 0; i < n; ++i) {
19         if (s[i] == '0') {
20             chenhlech--; // Gặp '0', giảm chênh lệch
21         } else {
22             chenhlech++; // Gặp '1', tăng chênh lệch
23         }
24
25         // Kiểm tra xem giá trị chênh lệch đã xuất hiện trước đó bao nhiêu lần
26         if (dem_chenhlech.find(chenhlech) != dem_chenhlech.end()) {
27             result += dem_chenhlech[chenhlech]; // Cộng số lần xuất hiện vào kết quả
28         }
29
30         dem_chenhlech[chenhlech]++; // Cập nhật số lần xuất hiện của giá trị chênh lệch
31     }
32
33     return result; // Trả về kết quả
34 }
35
36 int main() {
37     cout << "Ho va ten: Nguyen Viet Anh\n";
38     cout << "MSSV: 20215307\n";
39     string s;
40
41     // Đọc xâu nhị phân từ stdin
42     while (true) {
43         char c;
44         while ((c = getchar()) != '\n' && c != '\0') {}
45         if (c == '\0') break;
46         s += c;
47     }
48
49     int result = dem_xau_con(s);
50     cout << "Số lượng xâu con có số lượng '0' và '1' bằng nhau: " << result << endl;
51 }

```

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
100110011010110101000111111111000010100011010100010010010100111101001100101010101101111011110111
583
PS D:\Desktop\output>

```

Case 2

```

Desktop > lab4-4.13(HW).cpp > main()
6 //Ho va ten: Nguyen Viet Anh
7 //MSSV: 20215307
8 // Hàm đếm số lượng xâu con có số lượng '0' và '1' bằng nhau
9 int dem_xau_con(const string& s) {
10     int n = s.length();
11     unordered_map<int, int> dem_chenhlech;
12     dem_chenhlech[0] = 1; // Khởi tạo cho trường hợp chênh lệch bằng 0
13
14     int chenhlech = 0; // Chênh lệch giữa số lượng '0' và '1'
15     int result = 0; // Kết quả: số lượng xâu con hợp lệ
16
17     // Duyệt qua từng ký tự trong xâu nhị phân
18     for (int i = 0; i < n; ++i) {
19         if (s[i] == '0') {
20             chenhlech--; // Gặp '0', giảm chênh lệch
21         } else {
22             chenhlech++; // Gặp '1', tăng chênh lệch
23         }
24
25         // Kiểm tra xem giá trị chênh lệch đã xuất hiện trước đó bao nhiêu lần
26         if (dem_chenhlech.find(chenhlech) != dem_chenhlech.end()) {
27             result += dem_chenhlech[chenhlech]; // Cộng số lần xuất hiện vào kết quả
28         }
29
30         dem_chenhlech[chenhlech]++; // Cập nhật số lần xuất hiện của giá trị chênh lệch
31     }
}

```

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
1111010110010001100000010001111100111100001000110000001100000111010011011110110000001111011
1100000000110000011010111010110010111101111011000000101010001100011001001110011000001010011
101011101000011000001110100111101000001000110100000101000011100010001010001011011111101101
010110110110000010000000010000010110100101111011110000110111011111100100111011010111110111
1011101110000000110000111101111100100011111001110101010001111011110101100000111000000101001
100101100011101111000011100010101101101001101000100011001101011001101001000000110101001101
0010100111010110000001001011011111100000
14342

```

Case 3

```

Desktop > lab4-4.13(HW).cpp > ...
6 //Ho va ten: Nguyen Viet Anh
7 //MSSV: 20215307
8 // Hàm đếm số lượng xâu con có số lượng '0' và '1' bằng nhau
9 int dem_xau_con(const string& s) {
10     int n = s.length();
11     unordered_map<int, int> dem_chenhlech;
12     dem_chenhlech[0] = 1; // Khởi tạo cho trường hợp chênh lệch bằng 0
13
14     int chenhlech = 0; // Chênh lệch giữa số lượng '0' và '1'
15     int result = 0; // Kết quả: số lượng xâu con hợp lệ
16
17     // Duyệt qua từng ký tự trong xâu nhị phân
18     for (int i = 0; i < n; ++i) {
19         if (s[i] == '0') {
20             chenhlech--; // Gặp '0', giảm chênh lệch
21         } else {
22             chenhlech++; // Gặp '1', tăng chênh lệch
23         }
24
25         // Kiểm tra xem giá trị chênh lệch đã xuất hiện trước đó bao nhiêu lần
26         if (dem_chenhlech.find(chenhlech) != dem_chenhlech.end()) {
27             result += dem_chenhlech[chenhlech]; // Cộng số lần xuất hiện vào kết quả
28         }
29
30         dem_chenhlech[chenhlech]++; // Cập nhật số lần xuất hiện của giá trị chênh lệch
31     }
32 }

```

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
1011000001111101010101100101101111010111010000011010111101000001101001111000001101111100001
011111101000000110111111111100110100110101010010110110011000011101000000110011111100111000
01001000001111000001110011011111111000001100000000111100101110110100101001011000000001101
1111101110100110000111110010011011110111010000101100100100010110001111010000000011001110100
01101100001011011000100110000111110111001100010101110110000110101100110001010011100110111
1001100000111000110000011010110001011101000000011000101111100011010011101000001001001101000
0011110001000011100000010001010000010001111001000100011101011101111000101111101101011001001
0100010001000011110111010111110100101101110100000001111110011010001100010100011100001110100
1000000101010101110100100011011000100111011110001000000101100010000011001011110010100100111
100111101011100100001001001100110000100101111100011011100010011101011110010000100110100
110001111111011010000100001111111101110000000011011010110000110100110101110111010001000111

```

Case 4

```

Desktop > lab4-4.13(HW).cpp > main()
1  #include <iostream>
2  #include <unordered_map>
3  #include <vector>
4  using namespace std;
5
6  //Ho va ten: Nguyen Viet Anh
7  //MSSV: 20215307
8  // Hàm đếm số lượng xâu con có số lượng '0' và '1' bằng nhau
9  int dem_xau_con(const string& s) {
10     int n = s.length();
11     unordered_map<int, int> dem_chenhlech;
12     dem_chenhlech[0] = 1; // Khởi tạo cho trường hợp chênh lệch bằng 0
13
14     int chenhlech = 0; // Chênh lệch giữa số lượng '0' và '1'
15     int result = 0; // Kết quả: số lượng xâu con hợp lệ
16
17     // Duyệt qua từng ký tự trong xâu nhị phân
18     for (int i = 0; i < n; ++i) {
19         if (s[i] == '0') {
20             chenhlech--; // Gặp '0', giảm chênh lệch

```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

```

1101010111011111111111110110000010010100011100110011100001000011110100000101111001110101101011111
1001111101000100101011110011001011110110110011001110101000011000001101101000000111110100011101
0101111101001101100010100100100000100111111011110111101010010100010111100100010001001100011010
1010101111110000100101101001111000111111101111100100101111101111101100110010001010100011111
00100100100001011001010011100000100001111111001101110000011101101111100110011001000101010110
11000110110101101111111001111110111110100011010110001010010111011111101101011001101100101011
101110010011000010110101001100100000000010011110001100110110011000100011001010111011010001111
00110111100001101000101101000000101011101100011111110100110011110100001000001000111110000011
1101101001010100001000101001100110010011111010010110110001110101011100011100111100001010001111
011000111010110110100110000110110000011110110000010101001111001101010101001101101010111010001
1011111001110001011011000010100001100100011110001010110101110000010100101000011100000101110110
100110001111111011001100001011000011000000010010110111110110101010001010010101011110001010000
0010101001000100110111110110001001101101001000000010001010011010001101111011000111100000010
1001101010011110011110100101010100100010001101110001010011111010010000010100000100110010001111
0100110011001000000110110101110011001111110000110100101100110001000111001101000110011111101000
110111001010000111000010110101101010001000000101101101100010111100000100001010011101111111000
11000100100100000110111110111001010011110010000111101101111000100000001011010110110000101
011100111111010101000010111001101100010010001010011100011011001111111100110100111010101000100
01001101010010101101100011000011110011111011100110101111100010010011001000001111010110110011
0011011001001001101000001111111000101000110100100010111001110110000011100001110001010010011011
124022
PS D:\Desktop\output> 281665459

```