

Mục lục nội dung

Bài thực hành số 04 – Tuần 38	3
BÀI TẬP TRÊN LAP	3
Bài tập 1. Đảo ngược một danh sách liên kết đơn.	3
Bài tập 2. Tính diện tích tam giác theo tọa độ 3 đỉnh.	7
Bài tập 3. Tính tích có hướng của 2 vector.	9
Bài tập 4. Xóa hết các phần tử chẵn, sắp xếp giảm dần các số trong cả 2 vector và trộn lại thành một vector cũng được sắp xếp giảm dần.....	11
Bài tập 5. DFS.	12
Bài tập 6. BFS.	15
Bài tập 7. Các hàm thực hiện các phép giao và hợp của hai tập hợp được biểu diễn bằng set.....	18
Bài tập 1.8. Viết các hàm thực hiện các phép giao và hợp của hai tập hợp mờ được biểu diễn bằng map.	21
Bài tập 9. Cài đặt thuật toán Dijkstra trên đồ thị vô hướng được biểu diễn bằng danh sách kề sử dụng std::priority_queue.....	25
BÀI TẬP VỀ NHÀ	29
Bài tập 10. Search Engine	29
Bài tập 11.	37
Bài tập 12	42
Bài tập 13	47

Hoàng Thê Anh - 20215301

Mục lục hình ảnh

Hình 1 Bài 1 Đảo ngược danh sách liên kết đơn.....	3
Hình 2 Bài 2 tính diện tích tam giác khi biết tọa độ 3 đỉnh.....	7
Hình 3 Bài 3 Tính tích vô hướng của 2 vector	9
Hình 4 Bài 4 Thao tác 2 vector.....	11
Hình 5 Bài 5 DFS	12
Hình 6 Bài 6 BFS.....	15
Hình 7 Bài 7 Các hàm thực hiện các phép giao và hợp của hai tập hợp được biểu diễn bằng set.....	18
Hình 8 Bài 8 các hàm thực hiện các phép giao và hợp của hai tập hợp mờ được biểu diễn bằng map.....	21

Bài thực hành số 04 – Tuần 38

BÀI TẬP TRÊN LAP

Bài tập 1. Đảo ngược một danh sách liên kết đơn.

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

Passed all tests! ✓

```

27     while (head != nullptr) {
28         // In giá trị của nút đang xét
29         cout << head->data << " ";
30         // dịch con trỏ head trả sang phần tử liên kế nó để in tiếp
31         head = head->next;
32     }
33 }
34
35 // return the new head of the reversed list
36 Node *reverse(Node *head) {
37     // con trỏ prev là con trỏ trước của con trỏ hiện tại (tương ứng với phần tử liên
38     // Node *prev = nullptr;
39     // con trỏ cur đại diện cho vị trí con trỏ head
40     Node *cur = head;
41     // con trỏ next là con trỏ trả đến phần tử kế tiếp trong danh sách liên kết tương
42     Node *next = nullptr;
43     while (cur != nullptr) {
44         // gán con trỏ next là giá trị kế tiếp của con trỏ cur để lưu giá trị sau khi đảo ngược
45         next = cur->next;
46         // đảo ngược chiều liên kết của con trỏ hiện tại bằng cách trả đến con trỏ đã được lưu trước đó
47         cur->next = prev;
48         // dịch con trỏ prev sang phần tử tiếp theo để chuẩn bị việc đảo ngược
49         prev = cur;
50         // dịch con trỏ hiện tại sang phần tử kế tiếp để chuẩn bị đảo ngược khi chạy lại vòng lặp
51         cur = next;
52     }
53     // sau khi chạy xong vòng lặp thì cả con trỏ next và cur đều trả đến phần cuối cùng của dãy trước khi đảo ngược là null
54     // con trỏ prev sẽ trả đến phần tử cuối khác null tương ứng là phần tử đầu sau khi đảo ngược nên sẽ return prev
55     return prev;
56 }
57
58 main

```

Hình 1 Bài 1 Đảo ngược danh sách liên kết đơn

#include <bits/stdc++.h>

using namespace std;

struct Node {

```
int data;  
Node *next;  
  
Node(int data) {  
    this->data = data;  
    next = nullptr;  
}  
};  
  
Node *prepend(Node *head, int data) {  
    // Tạo 1 nút mới có có data được cho trước  
    Node *newNode = new Node(data);  
    // cho nút mới trỏ đến đầu danh sách hiện tại  
    newNode->next = head;  
    // trả về con trỏ mới là nút đầu mới  
    return newNode;  
}  
  
// print the list content on a line  
void print(Node *head) {  
    // duyệt đến khi đến phần tử cuối cùng là có next là null  
    while (head != nullptr) {  
        // In giá trị của nút đang xét  
        cout << head->data << " ";  
        // dịch con trỏ head trỏ sang phân tử liên kè nó để in tiếp  
        head = head->next;  
    }  
}
```

```
// return the new head of the reversed list

Node *reverse(Node *head) {
    // con trỏ prev là con trỏ trước của con trỏ hiện tại (tương ứng với phần tử liền kề kế tiếp sau khi đảo ngược danh sách liên kết)

    Node *prev = nullptr;

    // con trỏ cur đại diện cho vị trí con trỏ head

    Node *cur = head;

    // con trỏ next là con trỏ trỏ đến phần tử kế tiếp trong danh sách liên kết tương ứng với con trỏ sẽ trỏ đến con trỏ cur sau khi đảo ngược

    Node *next = nullptr;

    while (cur != nullptr) {
        // gán con trỏ next là giá trị kế tiếp của con trỏ cur để lưu giá trị sau khi đảo ngược

        next = cur->next;

        // đảo ngược chiều liên kết của con trỏ hiện tại bằng cách trỏ đến con trỏ đã được lưu trước đó

        cur->next = prev;

        // dịch con trỏ prev sang phần tử tiếp theo để chuẩn bị việc đảo ngược

        prev = cur;

        // dịch con trỏ hiện tại sang phần tử kế tiếp để chuẩn bị đảo ngược khi chạy lại vòng lặp

        cur = next;

    }

    // sau khi chạy xong vòng lặp thì cả con trỏ next và cur đều trỏ đến phần cuối cùng của dãy trước khi đảo ngược là null

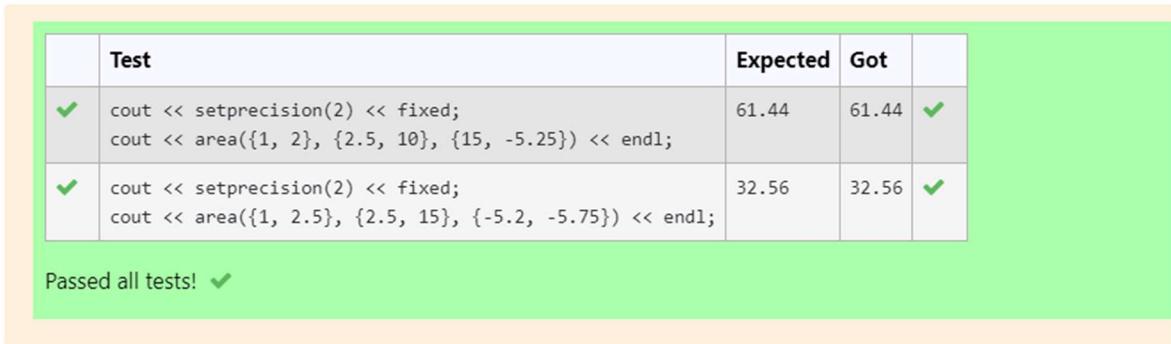
    // con trỏ prev sẽ trỏ đến phần tử cuối khác null tương ứng là phần tử đầu sau khi đảo ngược nên sẽ return prev

    return prev;
}
```

```
int main() {
    int n, u;
    cin >> n;
    Node *head = nullptr;
    for (int i = 0; i < n; ++i) {
        cin >> u;
        head = prepend(head, u);
    }

    cout << "Original list: ";
    print(head);
    head = reverse(head);
    cout << endl;
    cout << "Reversed list: ";
    print(head);
    cout << endl;
    cout << "*****" << endl;
    cout << "Hoang The Anh - 20215301" << endl;
    cout << "*****" << endl;
    return 0;
}
```

Bài tập 2. Tính diện tích tam giác theo tọa độ 3 đỉnh.



```

1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 #define Point pair<int, int>
6
7 double area(Point a, Point b, Point c) {
8     // tính độ dài cạnh ab bằng công thức tính khoảng cách giữa 2 điểm trong hệ trục tọa độ
9     double ab = sqrt((a.first - b.first) * (a.first - b.first) + (a.second - b.second) * (a.second - b.second));
10    // tính độ dài cạnh ac bằng công thức tính khoảng cách giữa 2 điểm trong hệ trục tọa độ
11    double ac = sqrt((a.first - c.first) * (a.first - c.first) + (a.second - c.second) * (a.second - c.second));
12    // tính độ dài cạnh bc bằng công thức tính khoảng cách giữa 2 điểm trong hệ trục tọa độ
13    double bc = sqrt((b.first - c.first) * (b.first - c.first) + (b.second - c.second) * (b.second - c.second));
14    // tính nửa chu vi của tam giác
15    double p = 0.5 * (ab + ac + bc);
16    // áp dụng công thức Heron để tính diện tích tam giác
17    return sqrt(p * (p - ab) * (p - ac) * (p - bc));
18 }
19
20 int main() {
21     cout << setprecision(2) << fixed;
22     cout << area({x: 1, y: 2}, {x: 2.5, y: 10}, {x: 15, y: -5.25}) << endl;
23
24     cout << "*****" << endl;
25     cout << "Hoang The Anh - 20215301" << endl;
26     cout << "*****" << endl;
27
28     return 0;
29 }

```

Hình 2 Bài 2 tính diện tích tam giác khi biết tọa độ 3 đỉnh

#include <bits/stdc++.h>

using namespace std;

#define Point pair<int, int>

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

// tính độ dài cạnh ab bằng công thức tính khoảng cách giữa 2 điểm trong hệ trục tọa độ

Hoàng Thé Anh - 20215301

```
double ab = sqrt((a.first - b.first) * (a.first - b.first) + (a.second - b.second) * (a.second - b.second));  
// tính độ dài cạnh ac bằng công thức tính khoảng cách giữa 2 điểm trong hệ trực toạ độ  
double ac = sqrt((a.first - c.first) * (a.first - c.first) + (a.second - c.second) * (a.second - c.second));  
// tính độ dài cạnh bc bằng công thức tính khoảng cách giữa 2 điểm trong hệ trực toạ độ  
double bc = sqrt((b.first - c.first) * (b.first - c.first) + (b.second - c.second) * (b.second - c.second));  
// tính nửa chu vi của tam giác  
double p = 0.5 * (ab + ac + bc);  
// áp dụng công thức Heron để tính diện tích tam giác  
return sqrt(p * (p - ab) * (p - ac) * (p - bc));  
}
```

```
int main() {  
    cout << setprecision(2) << fixed;  
    cout << area({1, 2}, {2.5, 10}, {15, -5.25}) << endl;  
  
    cout << "*****" << endl;  
    cout << "Hoang The Anh - 20215301" << endl;  
    cout << "*****" << endl;  
  
    return 0;  
}
```

Hoàng Thé Anh - 20215301

Bài tập 3. Tính tích có hướng của 2 vector.

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	✓

Passed all tests! ✓

```
#define Vector tuple<double, double, double>

Vector cross_product(Vector a, Vector b) {
    // lấy giá trị của x, y và z của vector a
    double a1 = get<0>(&a);
    double a2 = get<1>(&a);
    double a3 = get<2>(&a);
    // lấy giá trị của x, y và z của vector b
    double b1 = get<0>(&b);
    double b2 = get<1>(&b);
    double b3 = get<2>(&b);
    // tính x, y và z của vector tính có hướng
    double x = a2 * b3 - a3 * b2;
    double y = a3 * b1 - a1 * b3;
    double z = a1 * b2 - a2 * b1;
    // trả về vector đã tìm được
    return make_tuple(x, y, z);
}

int main() {
    cout << setprecision( n: 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;

    cout << "*****" << endl;
    cout << "Hoang The Anh - 20215301" << endl;
    cout << "*****" << endl;

    return 0;
}
```

Hình 3 Bài 3 Tính tích vô hướng của 2 vector

```
#include <bits/stdc++.h>
```

using namespace std;

```
#define Vector tuple<double, double, double>
```

```
Vector cross_product(Vector a, Vector b) {
```

```
    // lấy giá trị của x, y và z của vector a
```

```
    double a1 = get<0>(a);
```

```
    double a2 = get<1>(a);
```

```
    double a3 = get<2>(a);
```

```
    // lấy giá trị của x, y và z của vector b
```

```
    double b1 = get<0>(b);
```

```
    double b2 = get<1>(b);
```

```
    double b3 = get<2>(b);
```

```
    // tính x, y và z của vector tính có hướng
```

```
    double x = a2 * b3 - a3 * b2;
```

```
    double y = a3 * b1 - a1 * b3;
```

```
    double z = a1 * b2 - a2 * b1;
```

```
    // trả về vector đã tìm được
```

```
    return make_tuple(x, y, z);
```

```
}
```

```
int main() {
```

```
    cout << setprecision(2) << fixed;
```

```
    Vector a{1.2, 4, -0.5};
```

```
    Vector b{1.5, -2, 2.5};
```

```
    Vector c = cross_product(a, b);
```

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

```

cout << "*****" << endl;
cout << "Hoang The Anh - 20215301" << endl;
cout << "*****" << endl;

return 0;
}

```

Bài tập 4. 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.

	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 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 Decreasingly sorted c: 35 13 9
✓	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 elements of a: -7 5 7 13 9 Odd elements of b: 3 65 675 21 Decreasingly sorted a: 55 43 13 Decreasingly sorted b: 675 321 Decreasingly sorted c: 675 321
Passed all tests! ✓			

```

B3.cpp      B4.cpp
5 void print_vector(const vector<int> &a) {
6     for (int v: a) cout << v << ' ';
7     cout << endl;
8 }
9 // Hàm kiểm tra xem phần tử hiện tại là phần tử chẵn hay không
10 bool isEven(int i) {
11     return i % 2 == 0;
12 }
13
14 void delete_even(vector<int> &a) {
15     // hàm remove_if tạo ra 1 range mới là các phần tử lẻ lên đầu và trả về iterator phần
16     // tử chẵn
17     // dùng hàm erase để xoá các phần tử chẵn
18     a.erase( first: remove_if( first: a.begin(), last: a.end(), pred: isEven), last: a.end());
19 }
20 // định nghĩa hàm so sánh được sử dụng cho việc sắp xếp các phần tử giảm dần
21 bool compare(int a, int b) {
22     return a > b;
23 }
24
25 void sort_decrease(vector<int> &a) {
26     // sắp xếp các phần tử theo hàm compare
27     sort( first: a.begin(), last: a.end(), comp: compare);
28 }
29
30 vector<int> merge_vectors(const vector<int> &a, const vector<int> &b) {
31     // khởi tạo1 vector là hợp của 2 vector đầu vào
32     vector<int> v( n: a.size() + b.size());
33     // hợp 2 vector với thứ tự là hàm compare đã được định nghĩa
34     merge( first: a.begin(), last: a.end(), first2: b.begin(), last2: b.end(), result: v.begin(), comp: compare);
35     // trả về vector tìm được
36     return v;
37 }

```

Hình 4 Bài 4 Thao tác 2 vector

Bài tập 5. DFS.

```

int n = 10;
vector< list<int> > adj;
adj.resize(n + 1);
adj[1].push_back(2);
adj[1].push_back(3);
adj[1].push_back(6);
adj[2].push_back(7);
adj[2].push_back(4);
adj[2].push_back(8);
adj[3].push_back(10);
adj[3].push_back(9);
adj[4].push_back(1);
adj[4].push_back(8);
adj[5].push_back(2);
adj[5].push_back(4);
adj[6].push_back(7);
adj[6].push_back(9);
adj[7].push_back(3);
adj[7].push_back(9);
adj[7].push_back(10);
adj[8].push_back(9);
adj[8].push_back(2);
adj[9].push_back(3);
dfs(adj);

```

Passed all tests! ✓

```

#include <bits/stdc++.h>

using namespace std;

void dfs(vector<list<int> > adj) {
    // khởi tạo ngăn xếp duyệt cạnh con của cạnh trước
    stack<int> S;
    // mảng đánh dấu các đỉnh đã thăm chưa
    vector<bool> visited( n: adj.size());
    // thêm phần tử gốc vào ngăn xếp
    S.push( x: 1); // Bắt đầu từ đỉnh số 1
    while( !S.empty()) {
        // lấy phần tử đầu tiên của ngăn xếp
        int top = S.top();
        // bỏ khỏi ngăn xếp
        S.pop();
        // bỏ qua nếu đỉnh đã được xét
        if (visited[top]) continue;
        // in đỉnh ra màn hình
        cout << top << endl;
        // đánh dấu đã thăm đối với đỉnh chưa thăm
        visited[top] = true;
        // duyệt các đỉnh kế của đỉnh hiện tại theo thứ tự ngược để khi xét trong ngăn xếp, phần tử đầu tiên sẽ được xét trước
        for(auto it : reverse_iterator<list<int> > = adj[top].rbegin(); it != adj[top].rend(); ++it){
            // chỉ xét các đỉnh chưa được xét
            if(!visited[*it]){
                // thêm đỉnh chưa được xét vào ngăn xếp
                S.push( x: *it);
            }
        }
    }
}

```

Hình 5 Bài 5 DFS

```
#include <bits/stdc++.h>
```

```
using namespace std;
```

Hoàng Thé Anh - 20215301

```
void dfs(vector<list<int>> adj) {
    // khởi tạo ngăn xếp duyệt cạnh con của cạnh trước
    stack<int> S;
    // mảng đánh dấu các đỉnh đã được thăm chưa
    vector<bool> visited(adj.size());
    // thêm phần tử gốc vào ngăn xếp
    S.push(1); // Bắt đầu từ đỉnh số 1
    while (!S.empty()) {
        // lấy phần tử đầu tiên của ngăn xếp
        int top = S.top();
        // bỏ khỏi ngăn xếp
        S.pop();
        // bỏ qua nếu đỉnh đã được xét
        if (visited[top]) continue;
        // in đỉnh ra màn hình
        cout << top << endl;
        // đánh dấu đã thăm đối với đỉnh chưa thăm
        visited[top] = true;
        // duyệt các đỉnh kề của đỉnh hiện tại theo thứ tự ngược để khi xét trong ngăn xếp,
        // phần tử đầu tiên sẽ được xét trước
        for(auto it = adj[top].rbegin(); it != adj[top].rend(); ++it){
            // chỉ xét các đỉnh chưa được xét
            if(!visited[*it]){
                // thêm đỉnh chưa được xét vào ngăn xếp
                S.push(*it);
            }
        }
    }
}
```

}

```
int main() {  
    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);  
    cout << endl;  
  
    cout << "*****" << endl;  
    cout << "Hoang The Anh - 20215301" << endl;  
    cout << "*****" << endl;  
  
    return 0;  
}
```

Bài tập 6. BFS.

```

bfs(adj);
✓
int n = 10;
vector<list<int> > adj;
adj.resize(n + 1);
adj[1].push_back(2);
adj[1].push_back(3);
adj[1].push_back(6);
adj[2].push_back(7);
adj[2].push_back(4);
adj[2].push_back(8);
adj[2].push_back(10);
adj[3].push_back(9);
adj[4].push_back(1);
adj[4].push_back(8);
adj[5].push_back(2);
adj[5].push_back(4);
adj[6].push_back(7);
adj[6].push_back(9);
adj[7].push_back(3);
adj[7].push_back(9);
adj[7].push_back(10);
adj[8].push_back(9);
adj[8].push_back(2);
adj[9].push_back(3);
bfs(adj);

```

Passed all tests! ✓

```

void bfs(vector<list<int> > adj) {
    queue<int> Q;
    vector<bool> visited( n: adj.size());
    Q.push( x: 1); // Bắt đầu từ đỉnh số 1
    // đánh dấu đỉnh đầu đã được duyệt
    visited[1] = true;
    while (!Q.empty()) {
        // lấy đỉnh đầu của hàng đợi và xoá
        int front = Q.front();
        Q.pop();
        cout << front << endl;

        // xét tất cả các đỉnh kề của đỉnh đang xét
        for (int vertex: adj[front]) {
            // tại trường hợp đỉnh kề chưa được thăm
            if (!visited[vertex]) {
                // đánh dấu là đã thăm
                visited[vertex] = true;
                // thêm đỉnh chưa xét vào hàng đợi
                Q.push( x: vertex);
            }
        }
    }
}

```

Hình 6 Bài 6 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  
// đánh dấu đỉnh đầu đã được duyệt  
visited[1] = true;  
while (!Q.empty()) {  
    // lấy điểm đầu của hàng đợi và xoá  
    int front = Q.front();  
    Q.pop();  
    cout << front << endl;  
  
    // xét tập các đỉnh kề của đỉnh đang xét  
    for (int vertex: adj[front]) {  
        // tại trường hợp đỉnh kề chưa được thăm  
        if (!visited[vertex]) {  
            // đánh dấu là đã thăm  
            visited[vertex] = true;  
            // thêm điểm chưa xét vào hàng đợi  
            Q.push(vertex);  
        }  
    }  
}  
  
int main() {  
    int n = 7;  
    vector<list<int> > adj;  
    adj.resize(n + 1);
```

Hoàng Thê Anh - 20215301

```
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);
cout << endl;

cout << "*****" << endl;
cout << "Hoang The Anh - 20215301" << endl;
cout << "*****" << endl;

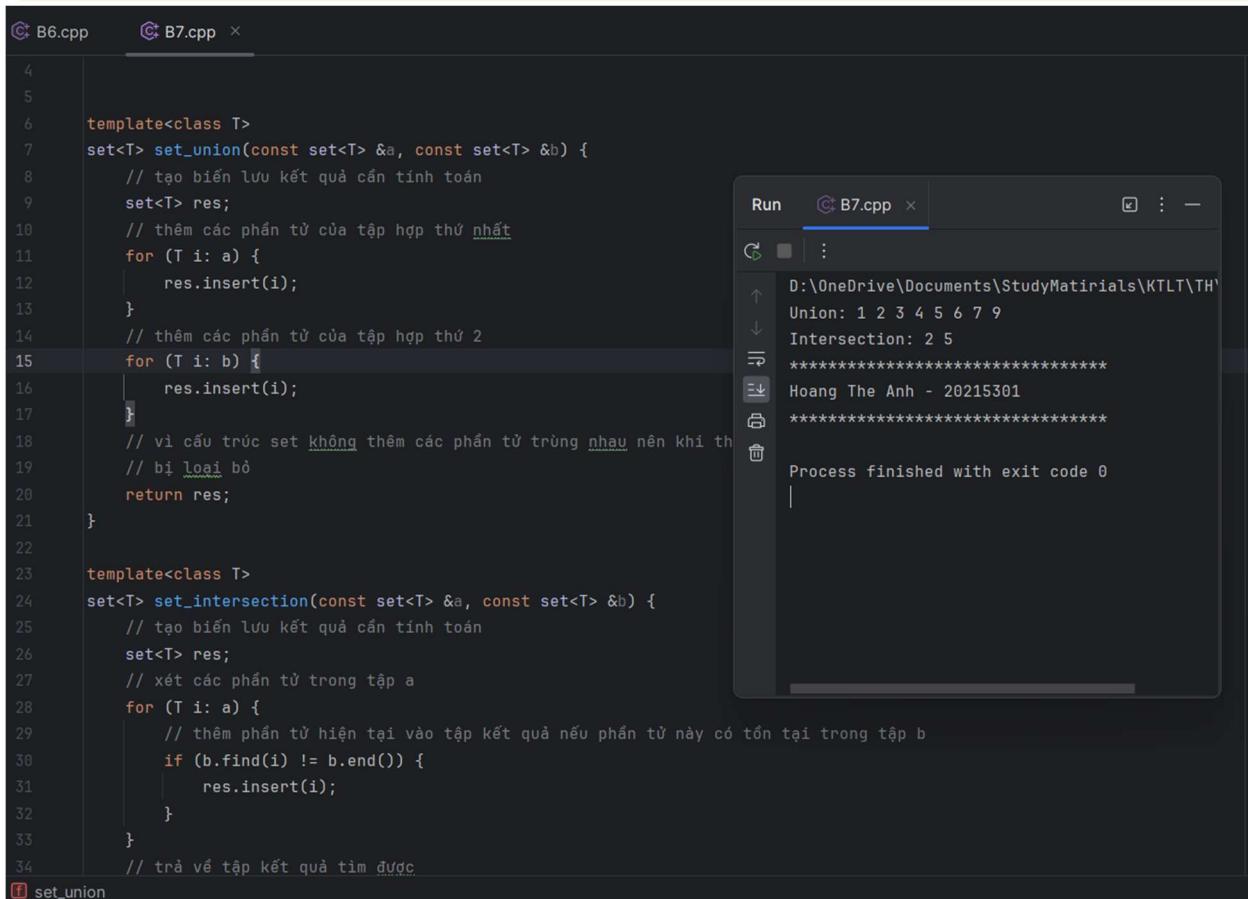
return 0;
}
```

Hoàng Thé Anh - 20215301

Bài tập 7. Các hàm thực hiện các phép giao và hợp của hai tập hợp được biểu diễn bằng set.

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>	Union: 1 2 3 4 5 6 7 9 Intersection: 2 5	Union: 1 2 3 4 5 6 7 9 Intersection: 2 5	✓
<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}; 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);</pre>	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 8 9 10 12 15 17 21 Intersection: 6 9 10	✓

Passed all tests! ✓



```
4  
5  
6     template<class T>  
7     set<T> set_union(const set<T> &a, const set<T> &b) {  
8         // tạo biến lưu kết quả cần tính toán  
9         set<T> res;  
10        // thêm các phần tử của tập hợp thứ nhất  
11        for (T i: a) {  
12            res.insert(i);  
13        }  
14        // thêm các phần tử của tập hợp thứ 2  
15        for (T i: b) {  
16            res.insert(i);  
17        }  
18        // vì cấu trúc set không thêm các phần tử trùng nhau nên khi th  
19        // bị loại bỏ  
20        return res;  
21    }  
22  
23    template<class T>  
24    set<T> set_intersection(const set<T> &a, const set<T> &b) {  
25        // tạo biến lưu kết quả cần tính toán  
26        set<T> res;  
27        // xét các phần tử trong tập a  
28        for (T i: a) {  
29            // thêm phần tử hiện tại vào tập kết quả nếu phần tử này có tồn tại trong tập b  
30            if (b.find(i) != b.end()) {  
31                res.insert(i);  
32            }  
33        }  
34        // trả về tập kết quả tìm được  
35    set_union
```

Hình 7 Bài 7 Các hàm thực hiện các phép giao và hợp của hai tập hợp được biểu diễn bằng set

```
#include <bits/stdc++.h>
```

```
using namespace std;
```

```
template<class T>
set<T> set_union(const set<T> &a, const set<T> &b) {
    // tạo biến lưu kết quả cần tính toán
    set<T> res;
    // thêm các phần tử của tập hợp thứ nhất
    for (T i: a) {
        res.insert(i);
    }
    // thêm các phần tử của tập hợp thứ 2
    for (T i: b) {
        res.insert(i);
    }
    // vì cấu trúc set không thêm các phần tử trùng nhau nên khi thêm 2 tập hợp vào res thì
    // các phần tử trùng nhau sẽ
    // bị loại bỏ
    return res;
}
```

```
template<class T>
set<T> set_intersection(const set<T> &a, const set<T> &b) {
    // tạo biến lưu kết quả cần tính toán
    set<T> res;
    // xét các phần tử trong tập a
    for (T i: a) {
        // thêm phần tử hiện tại vào tập kết quả nếu phần tử này có tồn tại trong tập b
        if (b.find(i) != b.end()) {
            res.insert(i);
        }
    }
}
```

```
    }  
}  
// trả về tập kết quả tìm được  
return res;  
}
```

```
template<class T>  
void print_set(const std::set<T> &a) {  
    for (const T &x: a) {  
        std::cout << x << ' ';  
    }  
    std::cout << std::endl;  
}  
  
int main() {  
    std::set<int> a = {1, 2, 3, 5, 7};  
    std::set<int> b = {2, 4, 5, 6, 9};  
    std::set<int> c = set_union(a, b);  
    std::set<int> d = set_intersection(a, b);  
  
    std::cout << "Union: ";  
    print_set(c);  
    std::cout << "Intersection: ";  
    print_set(d);  
  
    cout << "*****" << endl;  
    cout << "Hoang The Anh - 20215301" << endl;  
    cout << "*****" << endl;
```

return 0;

}

Bài tập 1.8. Viết các hàm thực hiện các phép giao và hợp của hai tập hợp mờ được biểu diễn bằng map.

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, 0.4) (6, 1) Intersection: { (1, 0.2) (2, 0.4) (4, 0.6) (5, 0.7) </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) (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, 0.41) (6, 1) Intersection: { (1, 0.15) (2, 0.14) (4, 0.6) (5, 0.75) </pre>

Passed all tests! ✓

```

4
5     template<class T>
6     map<T, double> fuzzy_set_union(const map<T, double> &a, const map<T, double> &b) {
7         // khởi tạo map lưu kết quả cần tìm
8         map<T, double> res;
9         // duyệt từng phần tử của a
10        for (auto i = a.begin(); i != a.end(); ++i) {
11            // lấy key của phần tử đang xét lưu vào biến key
12            T key = i->first;
13            // lấy độ thuộc của phần tử đang xét và lưu vào biến membership
14            double membership = i->second;
15            // nếu tập b có phần tử trùng giá trị của key với phần tử hiện tại thì trả về max của độ thuộc của 2 phần tử cùng key của 2 tập a, b
16            // nếu tập b không có phần tử nào trùng key với key của phần tử hiện tại thì thêm phần tử hiện tại vào biến kết quả
17            res[key] = b.find(key) != b.end() ? max(membership, b.at(key)) : membership;
18        }
19        // duyệt các phần tử trong tập b
20        for (auto i = b.begin(); i != b.end(); ++i) {
21            // lấy key của phần tử đang xét lưu vào biến key
22            T key = i->first;
23            // lấy độ thuộc của phần tử đang xét và lưu vào biến membership
24            double membership = i->second;
25            // chỉ xét các phần tử chưa có trong biến kết quả
26            if (res.find(key) == res.end()) {
27                // thêm phần tử hiện tại vào biến kết quả
28                res[key] = membership;
29            }
30        }
31        // trả về biến kết quả
32        return res;
33    }
34
35    fuzzy_set_union

```

Run B8.cpp x

D:\OneDrive\Documents\StudyMaterials\KTLT\TH\B8.exe

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) }

Hoang The Anh - 20215301

Process finished with exit code 0

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

#include <bits/stdc++.h>

using namespace std;

```
template<class T>
map<T, double> fuzzy_set_union(const map<T, double> &a, const map<T, double> &b)
{
    // khởi tạo map lưu kết quả cần tìm
    map<T, double> res;
    // duyệt từng phần tử của a
    for (auto i = a.begin(); i != a.end(); ++i) {
        // lấy key của phần tử đang xét lưu vào biến key
        T key = i->first;
        // lấy độ thuộc của phần tử đang xét và lưu vào biến membership
        double membership = i->second;
        // nếu tập b có phần trùng giá trị của key với phần tử hiện tại thì trả về max của độ
        // thuộc của 2 phần tử cùng key của 2 tập a, b
        // nếu tập b không có phần tử nào trùng key với key của phần tử hiện tại thì thêm
        // phần tử hiện tại vào biến kết quả
        res[key] = b.find(key) != b.end() ? max(membership, b.at(key)) : membership;
    }
    // duyệt các phần tử trong tập b
    for (auto i = b.begin(); i != b.end(); ++i) {
        // lấy key của phần tử đang xét lưu vào biến key
        T key = i->first;
        // lấy độ thuộc của phần tử đang xét và lưu vào biến membership
        double membership = i->second;
        // chỉ xét các phần tử chưa có trong biến kết quả (nghĩa là chỉ thuộc b mà không
        // thuộc a)
        if (res.find(key) == res.end()) {
            // thêm phần tử hiện tại vào biến kết quả
            res[key] = membership;
        }
    }
}
```

```
    res[key] = membership;  
}  
}  
// trả về biến kết quả  
return res;  
}  
  
template<class T>  
map<T, double> fuzzy_set_intersection(const map<T, double> &a, const map<T,  
double> &b) {  
    // khởi tạo map lưu kết quả cần tìm  
    map<T, double> res;  
    // duyệt từng phần tử của tập a  
    for (auto i = a.begin(); i != a.end(); ++i) {  
        // lấy key của phần tử đang xét lưu vào biến key  
        T key = i->first;  
        // lấy độ thuộc của phần tử đang xét và lưu vào biến membership  
        double membership = i->second;  
        // chỉ xét những phần tử thuộc cả 2 tập a và b  
        if (b.find(key) != b.end()) {  
            // thêm phần tử có độ thuộc bé hơn vào biến kết quả  
            res[key] = min(membership, b.at(key));  
        }  
    }  
    // trả về biến kết quả  
    return res;  
}
```

```
template<class T>

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

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

```

print_fuzzy_set(c);
std::cout << "Intersection: ";
print_fuzzy_set(d);

cout << "*****" << endl;
cout << "Hoang The Anh - 20215301" << endl;
cout << "*****" << endl;

return 0;
}

```

Bài tập 9. Cài đặt thuật toán Dijkstra trên đồ thị vô hướng được biểu diễn bằng danh sách kè sử dụng std::priority_queue

	vector<int> distance = dijkstra(adj); for (unsigned int i = 0; i < distance.size(); ++i) { cout << "distance " << 0 << "->" << i << " = " << distance[i] << endl; }			Time left 21:21:46
✓	<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 " << 0 << "->" << 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>	✓	

Passed all tests! ✓

Hoàng Thé Anh - 20215301

The screenshot shows a code editor with a file named B9.cpp. The code implements the Dijkstra algorithm to find the shortest path from vertex 0 to all other vertices in a graph represented by an adjacency list. The output window shows the distance from vertex 0 to each vertex (0, 1, 2, 3, 4, 5, 6, 7, 8) and the author's name.

```
vector<int> dijkstra(const vector<vector<pair<int, int>>> &adj) {
    // Khởi tạo vector lưu giá trị đường đi ngắn nhất của đỉnh ứng với index trong vector
    // mỗi giá trị đường đi của 1 đỉnh được khởi tạo là -1 ứng với việc chưa được xét
    vector<int> res( adj.size(), -1);
    // tạo 1 hàng đợi ưu tiên nhận vào 1 cặp
    // phần tử thứ nhất của cặp đại diện cho quãng đường từ đỉnh 0 đến đỉnh hiện tại
    // phần tử thứ 2 của cặp đại diện cho đỉnh hiện tại đang xét
    // hàng đợi sẽ sắp xếp theo chiều tăng dần của phần tử thứ nhất tương ứng việc xét đỉnh có đường đi bé hơn trước
    priority_queue<pair<int, int>, vector<pair<int, int>>, greater<>> pq;
    // thêm đỉnh bạn đầu là 0 có đường đi đến đỉnh 0 là 0
    pq.push( make_pair( 0, 0));
    while (!pq.empty()) {
        // lấy phần tử đầu tiên ra khỏi hàng đợi ưu tiên
        pair<int, int> top = pq.top();
        pq.pop();
        // Nếu đỉnh hiện tại đã được xét thì sẽ bỏ qua
        // vì hàng đợi ưu tiên sẽ xét đỉnh có đường đi bé nhất
        // thi đường đi chắc chắn lớn giá trị đã xét trước đó
        if (res[top.second] != -1) continue;
        // tại trường hợp đỉnh hiện tại chưa được xét nghĩa là
        // và lưu quãng đường vào biến res
        res[top.second] = top.first;
        // duyệt các đỉnh lân cận để thêm các đường đi khả thi
        for (pair<int, int> v: adj[top.second]) {
            // với trường hợp gặp đỉnh đã tìm được đường đi ngắn
            if (res[v.first] != -1) continue;
            // tính quãng đường từ đỉnh 0 đến đỉnh lân cận
            int weight = top.first + v.second;
            // thêm vào hàng đợi để tiếp tục duyệt
            pq.push( make_pair( weight, v.first));
    }
}
```

Run B9.cpp x
D:\OneDrive\Documents\StudyMaterials\KTLT\TH\B9.exe
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

Hoang The Anh - 20215301

Hình 9 Bài 9 thuật toán Dijkstra

```
#include <bits/stdc++.h>
```

```
using namespace std;
```

```
vector<int> dijkstra(const vector<vector<pair<int, int>>> &adj) {
    // Khởi tạo vector lưu giá trị đường đi ngắn nhất của đỉnh ứng với index trong vector
    // mỗi giá trị đường đi của 1 đỉnh được khởi tạo là -1 ứng với việc chưa được xét
    vector<int> res(adj.size(), -1);
    // tạo 1 hàng đợi ưu tiên nhận vào 1 cặp
    // phần tử thứ nhất của cặp đại diện cho quãng đường từ đỉnh 0 đến đỉnh hiện tại
    // phần tử thứ 2 của cặp đại diện cho đỉnh hiện tại đang xét
    // hàng đợi sẽ sắp xếp theo chiều tăng dần của phần tử thứ nhất tương ứng việc xét
    // đỉnh có đường đi bé hơn trước
    priority_queue<pair<int, int>, vector<pair<int, int>>, greater<>> pq;
    // thêm đỉnh bạn đầu là 0 có đường đi đến đỉnh 0 là 0
```

Hoàng Thé Anh - 20215301

```
    pq.push(make_pair(0, 0));  
    while (!pq.empty()) {  
        // lấy phần tử đầu tiên ra khỏi hàng đợi ưu tiên  
        pair<int, int> top = pq.top();  
        pq.pop();  
        // Nếu điểm hiện tại đã được xét thì sẽ bỏ qua  
        // vì hàng đợi ưu tiên sẽ xét điểm có đường đi bé nhất nên khi gặp 1 điểm lần thứ 2  
        // thì đường đi chắc chắn lớn hơn giá trị đã xét trước đó  
        if (res[top.second] != -1) continue;  
        // tại trường hợp điểm hiện tại chưa được xét nghĩa là đã tìm được đường đi ngắn  
        // nhất từ 0 đến điểm hiện tại  
        // và lưu quãng đường vào biến res  
        res[top.second] = top.first;  
        // duyệt các đỉnh lân cận để thêm các đường đi khả thi  
        for (pair<int, int> v: adj[top.second]) {  
            // với trường hợp gặp đỉnh đã tìm được đường đi ngắn nhất thì sẽ bỏ qua  
            if (res[v.first] != -1) continue;  
            // tính quãng đường từ điểm 0 đến đỉnh lân cận  
            int weight = top.first + v.second;  
            // thêm vào hàng đợi để tiếp tục duyệt  
            pq.push(make_pair(weight, v.first));  
        }  
    }  
    // trả về vector cần tìm  
    return res;  
}  
  
int main() {
```

Hoàng Thé Anh - 20215301

```
int n = 9;  
vector<vector<pair<int, int>> > adj(n);  
auto add_edge = [&adj](int u, int v, int w) {  
    adj[u].push_back({v, w});  
    adj[v].push_back({u, w});  
};  
add_edge(0, 1, 4);  
add_edge(0, 7, 8);  
add_edge(1, 7, 11);  
add_edge(1, 2, 8);  
add_edge(2, 3, 7);  
add_edge(2, 8, 2);  
add_edge(3, 4, 9);  
add_edge(3, 5, 14);  
add_edge(4, 5, 10);  
add_edge(5, 6, 2);  
add_edge(6, 7, 1);  
add_edge(6, 8, 6);  
add_edge(7, 8, 7);  
vector<int> distance = dijkstra(adj);  
for (int i = 0; i < distance.size(); ++i) {  
    cout << "distance " << 0 << "->" << i << " = " << distance[i] << endl;  
}  
  
cout << "*****" << endl;  
cout << "Hoang The Anh - 20215301" << endl;  
cout << "*****" << endl;  
return 0;
```

}

BÀI TẬP VỀ NHÀ

Chụp ảnh kết quả của tất cả các test.

Bài tập 10. Search Engine

```
#include <bits/stdc++.h>
```

```
using namespace std;
```

```
#define MAX 1000
```

```
int N, Q;
```

```
// mảng d để lưu các đoạn văn bản
```

```
string d[MAX], t[MAX];
```

```
map<string, int> df;
```

```
string line;
```

```
// hàm tính số lần xuất hiện của từ word trong văn bản doc
```

```
int calcFrequency(const string &word, const string &doc) {
```

```
    int res = 0;
```

```
    // tách văn bản thành các từ
```

```
    stringstream ss(doc);
```

```
    string substring;
```

```
    while (getline(ss, substring, ',')) {
```

```
        // nếu từ cần tìm trùng với từ hiện tại thì tăng biến tần suất thêm 1
```

```
        if (word == substring) res++;
```

```
}
```

```
// trả về tần suất của từ cần tính đối với văn bản đầu vào
```

```
return res;
```

}

```
// hàm tính maxf của văn bản  
int calcMaxf(const string &doc) {  
    int res = 0;  
    // tách văn bản thành các từ  
    stringstream ss(doc);  
    string substring;  
    while (getline(ss, substring, ',')) {  
        // tính tần suất của từ hiện tại trong văn bản  
        int frequency = calcFrequency(substring, doc);  
        // lấy giá trị lớn hơn của res và tần suất của từ hiện tại  
        res = max(res, frequency);  
    }  
    // trả về giá trị lớn nhất  
    return res;  
}
```

```
// hàm tính số điểm (độ khớp nhau) của văn bản và truy vấn  
double calcScore(const string &document, const string &terms) {  
    // khởi tạo biến lưu số điểm  
    double score = 0;  
    // tính giá trị lớn nhất của tần suất xuất hiện của tất cả các từ có trong văn bản bằng  
    // hàm calcMaxf  
    int maxf = calcMaxf(document);  
    // chuyển chuỗi thành 1 luồng để tách các từ trong 1 câu truy vấn  
    stringstream ss(terms);  
    // khởi tạo biến để lưu trữ từng từ trong 1 truy vấn
```

Hoàng Thê Anh - 20215301

```
string term;  
  
while (getline(ss, term, ',')) {  
    // tính tần suất của từ đang xét trong văn bản  
    int f = calcFrequency(term, document);  
    // khởi tạo biến TF  
    double TF;  
    if (f == 0) {  
        // nếu từ không xuất hiện trong văn bản thì TF = 0  
        TF = 0;  
    } else {  
        // tính TF theo công thức khi f != 0  
        TF = 0.5 + 0.5 * (static_cast<double>(f) / maxf);  
    }  
    // tính IDF của từ hiện tại theo công thức  
    double IDF = log2(static_cast<double>(N + 1) / (df[term] + 1));  
    // cộng vào biến score điểm của từ hiện tại  
    score += TF * IDF;  
}  
// trả về tổng điểm tính được  
return score;  
}  
  
// hàm tính giá trị index của văn bản khớp với truy vấn nhất  
int calcIndexOfBestMatchDoc(const string &terms) {  
    // khởi tạo biến index đại diện cho giá trị index của văn bản  
    int index = 0;  
    // khởi tạo biến maxScore đại diện cho điểm số lớn nhất đến văn bản hiện tại  
    double maxScore = 0;
```

Hoàng Thê Anh - 20215301

```
// duyệt từng văn bản trong tập văn bản đầu vào
for (int i = 0; i < N; ++i) {
    // tính số điểm của văn bản có index i với truy vấn terms bằng hàm calcScore
    double score = calcScore(d[i], terms);
    // tại trường hợp tìm được văn bản có điểm cao hơn (khớp với truy vấn hơn)
    if (score > maxScore) {
        // cập nhật lại biến index(+1 vì giá trị in ra tính thứ tự văn bản từ 1, đoạn mã tính
        // thứ tự văn bản từ 0)
        index = i + 1;
        // cập nhật lại điểm tối đa
        maxScore = score;
    }
}
// trả về index của văn bản có giá trị điểm khớp cao nhất
return index;
}

int main() {
    cin >> N;
    // loại bỏ dòng chứa N
    getline(cin, line);
    for (int i = 0; i < N; ++i) {
        // lấy dòng văn bản đầu vào và lưu vào biến line
        getline(cin, line);
        // gán giá trị lấy được vào mảng lưu các văn bản
        d[i] = line;
    }
}
```

```
cin >> Q;  
// loại bỏ dòng chứa Q  
getline(cin, line);  
for (int i = 0; i < Q; ++i) {  
    // lấy dòng văn bản đầu vào và lưu vào biến line  
    getline(cin, line);  
    // gán giá trị lấy được vào mảng lưu các văn bản  
    t[i] = line;  
}  
// tính df của các từ trong các truy vấn  
// xét từng văn bản  
for (int i = 0; i < N; ++i) {  
    // lấy văn bản dựa vào index  
    string document = d[i];  
    // biến added để lưu các từ trong truy vấn đã xét đối với văn bản hiện tại  
    set<string> added;  
    for (int j = 0; j < Q; ++j) {  
        // lấy truy vấn dựa vào index  
        string terms = t[j];  
        // chuyển chuỗi thành 1 luồng để tách các từ trong 1 câu truy vấn  
        stringstream ss(terms);  
        // tạo 1 biến để lưu các từ trong 1 câu truy vấn  
        string substring;  
        while (getline(ss, substring, ',')) {  
            // tìm từ hiện tại trong set added, nếu chưa xét từ hiện tại thì tiếp tục, xét rồi thì  
            // bỏ qua đối với văn bản hiện tại  
            if (added.find(substring) == added.end()) {  
                // đánh dấu từ hiện tại đã được xét
```

Hoàng Thé Anh - 20215301

```
added.insert(substring);

// tính số lần xuất hiện của từ hiện tại trong văn bản đang xét

// nếu từ hiện tại có trong văn bản thì tăng giá trị của key có giá trị là từ hiện
tại thêm 1 đơn vị

if (calcFrequency(substring, document) != 0) df[substring]++;

}

}

}

}

// xét từng truy vấn

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

// lấy giá trị truy vấn bằng index

string item = t[i];

// tính giá trị index của văn bản khớp nhất đối với truy vấn hiện tại bằng hàm
calcIndexOfBestMatchDoc

// in giá trị tính được ra màn hình

cout << calcIndexOfBestMatchDoc(item) << endl;

}

cout << "*****" << endl;
cout << "Hoang The Anh - 20215301" << endl;
cout << "*****" << endl;

return 0;
}
```

Case 1:

Hoàng Thé Anh - 20215301

```
70     // trả về tổng điểm tính được
71     return score;
72 }
73
74 // hàm tính giá trị index của văn bản khớp với truy vấn nhất
75 int calcIndexOfBestMatchDoc(const string &terms) {
76     // khởi tạo biến index đại diện cho giá trị index của văn bản
77     int index = 0;
78     // khởi tạo biến maxScore đại diện cho điểm số lớn nhất đến văn bản hiện tại
79     double maxScore = 0;
80     // duyệt từng văn bản trong tập văn bản đầu vào
81     for (int i = 0; i < N; ++i) {
82         // tính số điểm của văn bản có index i với truy vấn terms bằng hàm calcScore
83         double score = calcScore( document, d[i], terms);
84         // tại trường hợp tìm được văn bản có điểm cao hơn (khớp với truy vấn hơn)
85         if (score > maxScore) {
86             // cập nhật lại biến index(+1 vì giá trị in ra tinh thứ tự văn bản từ 1, đoạn mã
87             index = i + 1;
88             // cập nhật lại điểm tối đa
89             maxScore = score;
90         }
91     }
92     // trả về index của văn bản có giá trị điểm khớp cao nhất
93     return index;
94 }
95
96 int main() {
97     cin >> N;
98     // loại bỏ dòng chứa N
99     getline( &cin, &line);
100    for (int i = 0; i < N; ++i) {
```

Case 2:

```
70     // trả về tổng điểm tính được
71     return score;
72 }
73
74 // hàm tính giá trị index của văn bản khớp với truy vấn nhất
75 int calcIndexOfBestMatchDoc(const string &terms) {
76     // khởi tạo biến index đại diện cho giá trị index của văn bản
77     int index = 0;
78     // khởi tạo biến maxScore đại diện cho điểm số lớn nhất đến văn bản hiện tại
79     double maxScore = 0;
80     // duyệt từng văn bản trong tập văn bản đầu vào
81     for (int i = 0; i < N; ++i) {
82         // tính số điểm của văn bản có index i với truy vấn terms bằng hàm calcScore
83         double score = calcScore( document, d[i], terms);
84         // tại trường hợp tìm được văn bản có điểm cao hơn (khớp với truy vấn hơn)
85         if (score > maxScore) {
86             // cập nhật lại biến index(+1 vì giá trị in ra tinh thứ tự văn bản từ 1, đoạn mã
87             index = i + 1;
88             // cập nhật lại điểm tối đa
89             maxScore = score;
90         }
91     }
92     // trả về index của văn bản có giá trị điểm khớp cao nhất
93     return index;
94 }
95
96 int main() {
97     cin >> N;
98     // loại bỏ dòng chứa N
99     getline( &cin, &line);
100    for (int i = 0; i < N; ++i) {
```

Hoàng Thé Anh - 20215301

Case 3:

```
70     // trả về tổng điểm tính được
71     return score;
72 }
73
74 // hàm tính giá trị index của văn bản khớp với truy vấn nhất
75 int calcIndexOfBestMatchDoc(const string &terms) {
76     // khởi tạo biến index đại diện cho giá trị index của văn bản
77     int index = 0;
78     // khởi tạo biến maxScore đại diện cho điểm số lớn nhất đến văn bản hiện tại
79     double maxScore = 0;
80     // duyệt từng văn bản trong tập văn bản đầu vào
81     for (int i = 0; i < N; ++i) {
82         // tính số điểm của văn bản có index i với truy vấn terms bằng hàm calcScore
83         double score = calcScore(document: d[i], terms);
84         // tại trường hợp tìm được văn bản có điểm cao hơn (khớp với truy vấn hơn)
85         if (score > maxScore) {
86             // cập nhật lại biến index(+1 vì giá trị in ra tính thứ tự văn bản từ 1, đoạn mã
87             index = i + 1;
88             // cập nhật lại điểm tối đa
89             maxScore = score;
90         }
91     }
92     // trả về index của văn bản có giá trị điểm khớp cao nhất
93     return index;
94 }
95
96 D int main() {
97     cin >> N;
98     // loại bỏ dòng chứa N
99     getline( &cin, &line);
100    for (int i = 0; i < N; ++i) {
101        calcIndexOfBestMatchDoc
102    }
103 }
```

Run TN

- ↑ D:\OneDrive\Documents\StudyMatrials\
- ↓ 21
- ⤒ 277
- ⤓ 132
- ⤔ 82
- ⤖ 84
- ⤘ 278
- ⤙ 141
- ⤚ 222
- ⤛ 268
- ⤜ 275
- ⤝ 88
- ⤞ 142
- ⤟ 287
- ⤠ 177
- ⤡ 211
- ⤢ 105
- ⤣ 121
- ⤤ 124
- ⤥ 171
- ⤦ 149
- ⤧ 3
- ⤨ 192
- ⤩ 283
- ⤪ 238
- ⤫ 74
- ⤬ 21
- ⤭ 242
- ⤮ 3
- ⤯ 239

Case 4:

```
70     // trả về tổng điểm tính được
71     return score;
72 }
73
74 // hàm tính giá trị index của văn bản khớp với truy vấn nhất
75 int calcIndexOfBestMatchDoc(const string &terms) {
76     // khởi tạo biến index đại diện cho giá trị index của văn bản
77     int index = 0;
78     // khởi tạo biến maxScore đại diện cho điểm số lớn nhất đến văn bản hiện tại
79     double maxScore = 0;
80     // duyệt từng văn bản trong tập văn bản đầu vào
81     for (int i = 0; i < N; ++i) {
82         // tính số điểm của văn bản có index i với truy vấn terms bằng hàm calcScore
83         double score = calcScore(document: d[i], terms);
84         // tại trường hợp tìm được văn bản có điểm cao hơn (khớp với truy vấn hơn)
85         if (score > maxScore) {
86             // cập nhật lại biến index(+1 vì giá trị in ra tính thứ tự văn bản từ 1, đoạn mã
87             index = i + 1;
88             // cập nhật lại điểm tối đa
89             maxScore = score;
90         }
91     }
92     // trả về index của văn bản có giá trị điểm khớp cao nhất
93     return index;
94 }
95
96 D int main() {
97     cin >> N;
98     // loại bỏ dòng chứa N
99     getline( &cin, &line);
100    for (int i = 0; i < N; ++i) {
101        calcIndexOfBestMatchDoc
102    }
103 }
```

Run TN

- ↑ D:\OneDrive\Documents\StudyMatrials\
- ↓ 380
- ⤒ 22
- ⤓ 271
- ⤔ 260
- ⤖ 106
- ⤘ 295
- ⤙ 226
- ⤚ 193
- ⤛ 272
- ⤜ 223
- ⤝ 183
- ⤞ 69
- ⤨ 248
- ⤩ 237
- ⤪ 126
- ⤪ 34
- ⤪ 43
- ⤪ 212
- ⤪ 38
- ⤪ 22
- ⤪ 280
- ⤪ 192
- ⤪ 91
- ⤪ 69
- ⤪ 175
- ⤪ 171
- ⤪ 68
- ⤪ 70
- ⤪ 143

Case 5:

```

    // trả về tổng điểm tính được
    return score;
}

// hàm tính giá trị index của văn bản khớp với truy vấn nhất
int calcIndexOfBestMatchDoc(const string &terms) {
    // khởi tạo biến index đại diện cho giá trị index của văn bản
    int index = 0;
    // khởi tạo biến maxScore đại diện cho điểm số lớn nhất đến văn bản hiện tại
    double maxScore = 0;
    // duyệt từng văn bản trong tập văn bản đầu vào
    for (int i = 0; i < N; ++i) {
        // tính số điểm của văn bản có index i với truy vấn terms bằng hàm calcScore
        double score = calcScore(document, d[i], terms);
        // tại trường hợp tìm được văn bản có điểm cao hơn (khớp với truy vấn hơn)
        if (score > maxScore) {
            // cập nhật lại biến index(+1 vì giá trị in ra tinh thứ tự văn bản từ 1, đoạn m
            index = i + 1;
            // cập nhật lại điểm tối đa
            maxScore = score;
        }
    }
    // trả về index của văn bản có giá trị điểm khớp cao nhất
    return index;
}

int main() {
    cin >> N;
    // loại bỏ dòng chứa N
    getline( &cin, &line );
    for (int i = 0; i < N; ++i) {

```

File: main.cpp

Path: D:\OneDrive\Documents\StudyMaterials\

Line numbers (right side):

- 348
- 122
- 333
- 220
- 307
- 104
- 54
- 29
- 153
- 7
- 336
- 130
- 124
- 100
- 123
- 353
- 397
- 4
- 151
- 208
- 123
- 397
- 6
- 177
- 321
- 198
- 252
- 32
- 202

Time: 80:52 LF UTF-8

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 ai 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 tên giặc. Giả sử đoạn tường thứ i có xi lính. Khi đó nếu số tên giặc không vượt quá xi thì không có tên giặc nào lọt vào được qua đoạn tường này. Ngược lại sẽ có ai - xi * ki tên giặc lọt vào lâu đài qua đoạn tường này.

Yêu cầu hãy viết chương trình phân bổ lính đứng ở các đoạn tường sao cho tổng số lính là s và tổng số lượng tên giặc lọt vào lâu đài là nhỏ nhất.

```
#include <bits/stdc++.h>
```

```
using namespace std;
```

```
#define pii pair<int, int>

// n là số đoạn tường
// s là số người phòng thủ
// ai là số người tấn công tường
// ki là khả năng trống trả
int n, s, ai, ki, numOfDefenders = 0;
// res là số lượng người tấn công có thể lọt vào đến thời điểm hiện tại
long long int res = 0;
priority_queue<pii> pq;

int main() {
    cin >> n >> s;
    for (int i = 0; i < n; ++i) {
        cin >> ai >> ki;
        // công tất các người tấn công để tính tổng số người tấn công
        res += ai;
        // thêm vào hàng đợi ưu tiên theo thứ tự khả năng chống trả -> số người tấn công
        // ưu tiên xét những tường có khả năng chống trả lớn để chống trả được nhiều tên
        // giặc nhất có thể
        pq.push(make_pair(ki, ai));
    }

    while (!pq.empty()) {
        // lấy bức tường chống trả tốt nhất để xét
        pii top = pq.top();
        pq.pop();
        // kiểm tra xem có bao nhiêu tên giặc bị tiêu diệt
        if (top.second > res) {
            res = top.second;
        }
    }
}
```

Hoàng Thé Anh - 20215301

// nếu khả năng chống trả lớn hơn số lượng người tấn công thì đánh lại thứ tự ưu tiên cho bức tường

```
if (top.first > top.second) {  
    // thứ tự ưu tiên bây giờ chỉ là số lượng người tấn công  
    pq.push(make_pair(top.second, top.second));  
    continue;  
}
```

// tại trường hợp khả năng chống trả nhỏ hơn số người tấn công thì sẽ thêm 1 người phòng thủ vào tường hiện tại

```
// trừ biến res đi số lượng người tấn công vì có thêm 1 người phòng thủ ở đây  
res -= top.first;
```

```
// tăng biến đếm số người phòng thủ lên 1  
numOfDefenders++;
```

```
// hết người để phòng thủ thì kết thúc vòng lặp  
if (numOfDefenders == s) break;
```

```
// tính số người tấn công còn lại của bức tường đang xét  
int remainingAttacker = top.second - top.first;
```

```
// nếu không còn ai tấn công tường này thì không cần thêm vào hàng đợi ưu tiên  
if (remainingAttacker == 0) continue;
```

```
if (remainingAttacker < top.first) {
```

// tại trường hợp số người tấn công còn lại nhỏ hơn khả năng chống trả thì sẽ thêm phần tử mới với độ ưu tiên phù hợp

```
    pq.push(make_pair(remainingAttacker, remainingAttacker));
```

```
} else {
```

// tại trường hợp số người tấn công còn lại lớn hơn khả năng chống trả thì sẽ thêm phần tử mới bình thường

```
    pq.push(make_pair(top.first, remainingAttacker));
```

```
}
```

```

    }

// in ra kết quả tính được

cout << res << endl;

return 0;
}

```

Case1:

```

B11.cpp  main.cpp  input.txt

24     pq.push( make_pair( &k1, &ai));
25 }
26
27 while (!pq.empty()) {
28     // lấy bức tường chống trả tốt nhất để xét
29     pii top = pq.top();
30     pq.pop();
31
32     // nếu khả năng chống trả lớn hơn số lượng người tấn công thì đánh lại thù
33     if (top.first > top.second) {
34         // thù tự ưu tiên bấy giờ chỉ là số lượng người tấn công
35         pq.push( make_pair( &top.second, &top.second));
36         continue;
37     }
38     // tại trường hợp khả năng chống trả nhỏ hơn số người tấn công thì sẽ thay
39     // trừ biến res đi số lượng người tấn công vì có thêm 1 người phòng thủ ở
40     res -= top.first;
41     // tăng biến đếm số người phòng thủ lên 1
42     numOfDefenders++;
43     // hết người để phòng thủ thì kết thúc vòng lặp
44     if (numOfDefenders == s) break;
45     // tìm số người tấn công còn lại của bức tường đang xét
46     int remainingAttacker = top.second - top.first;
47     // nếu không còn ai tấn công tường này thi không cần thêm vào hàng đợi ư
48     if (remainingAttacker == 0) continue;
49
50     if (remainingAttacker < top.first) {
51         // tại trường hợp số người tấn công còn lại nhỏ hơn khả năng chống trả
52         pq.push( make_pair( &remainingAttacker, &remainingAttacker));
53     } else {
54         // tại trường hợp số người tấn công còn lại lớn hơn khả năng chống trả
55     }
56 }

main
H > input.txt

```

Run TN x

D:\OneDrive\Documents\StudyMaterials\KTLT\TH\
242

Hoang The Anh - 20215301

Process finished with exit code 0

Case2:

Hoàng Thé Anh - 20215301

```
B11.cpp main.cpp input.txt
1 57 37
2 25 61
3 32 66
4 31 74
5 38 88
6 49 62
7 37 9
8 34 83
9 48 24
10 6 15
11 26 82
12 50 15
13 26 88
14 34 81
15 1 4
16 40 63
17 50 41
18 51 73
19 45 15
20 47 59
21 15 52
22 7 72
23 42 65
24 38 56
25 48 1
26 24 71
27 56 65
28 29 49
29 44 50
30 4 92
31 51 11
32 25 100
main
```

Case3:

```
B11.cpp main.cpp input.txt
72 43 100
73 42 60
74 21 57
75 28 97
76 32 52
77 56 96
78 82 7
79 88 42
80 40 89
81 62 96
82 61 54
83 76 75
84 74 58
85 7 59
86 40 82
87 17 37
88 45 67
89 18 1
90 79 36
91 75 85
92 95 65
93 82 92
94 63 7
95 16 55
96 33 84
97 3 79
98 22 81
99 88 61
100 63 99
101 62 77
102
main
```

Case4:

Hoàng Thế Anh - 20215301

```
B11.cpp main.cpp input.txt
1 100 2569
2 90 1
3 57 2
4 40 2
5 40 2
6 27 1
7 50 1
8 74 2
9 93 2
10 6 1
11 39 1
12 42 1
13 71 2
14 20 1
15 21 1
16 42 2
17 42 1
18 62 1
19 85 1
20 18 2
21 88 1
22 73 1
23 37 1
24 48 2
25 93 2
26 29 2
27 49 2
28 75 1
29 67 1
30 36 2
31 77 1
32 91 2
```

Case5:

```
B11.cpp main.cpp input.txt
1 100 10000
2 100 1
3 100 1
4 100 1
5 100 1
6 100 1
7 100 1
8 100 1
9 100 1
10 100 1
11 100 1
12 100 1
13 100 1
14 100 1
15 100 1
16 100 1
17 100 1
18 100 1
19 100 1
20 100 1
21 100 1
22 100 1
23 100 1
24 100 1
25 100 1
26 100 1
27 100 1
28 100 1
29 100 1
30 100 1
31 100 1
32 100 1
```

Bài tập 12

Cho một lược đồ gồm n cột chữ nhật liên tiếp nhau có chiều rộng bằng 1 và chiều cao là lát lượt là các số nguyên không âm h1,h2,...,hn . Hãy xác định hình chữ nhật có diện tích lớn nhất có thể tạo thành từ các cột liên tiếp.

```
#include <bits/stdc++.h>

using namespace std;

inline int largestRectangleInHistogram(vector<int> histogram) {

    stack<int> s;
    int maxArea = 0;
    int area = 0;
    int i = 0;
    int size = histogram.size();

    // duyệt từng phần tử để tính cận trái và phải
    while (i < size) {
        // tại s.empty() là trường hợp đặc biệt: xét phần tử đầu tiên
        // tại trường hợp phần tử bên phải lớn hoặc bằng phần tử hiện tại lưu index của
        // phần tử hiện tại đại diện cho cận trái của phần tử tiếp theo
        if (s.empty() || histogram[s.top()] <= histogram[i]) {
            // thêm cận trái của phần tử tiếp theo
            s.push(i);
            // tăng giá trị của index để xét tiếp
            i++;
        } else {
            // tại trường hợp lọt vào else là tìm thấy cận phải của phần tử thứ i-1 của stack
            // đỉnh của stack đại diện cho cận trái
            int top = s.top();
            s.pop();
            area = top * histogram[i];
            if (area > maxArea)
                maxArea = area;
        }
    }
}
```

Hoàng Thé Anh - 20215301

// tính số ô theo 2 cận trái phải. tại trường hợp đặc biệt là ko có cận trái nghĩa là stack rỗng thì i sẽ là cận số cột

// tại trường hợp bình thường ta sẽ tính số ô với 2 cận là i và s.top(), 2 cận này không tính vào số ô

// tương đương với việc số ô tính từ s.top()+1 đến i-1 nên ta có công thức i-s.top()-1

// diện tích bằng chiều cao * số ô chiếm chỗ

area = histogram[top] * (s.empty() ? i : (i - s.top() - 1));

// cập nhật giá trị diện tích tối đa nếu tìm được giá trị tốt hơn

if (area > maxArea) {

 maxArea = area;

}

}

// sau khi làm vòng lặp trên ta được (nếu có thể) các chiều cao tăng dần vì tăng dần nên không có cận phải của phần tử cuối

// vì là phần tử cuối cùng nên t sẽ cho cận phải là i vì đã +1 trong vòng lặp

while (!s.empty()) {

 // lấy index để tính cận trái và chiều cao

 int top = s.top();

 s.pop();

 // áp dụng công thức như trên vì 2 cận đã xác định được

 area = histogram[top] * (s.empty() ? i : (i - s.top() - 1));

 // cập nhật lại giá trị nếu tìm được giá trị tốt hơn

 if (area > maxArea) {

 maxArea = area;

 }

}

Hoàng Thê Anh - 20215301

```
// trả về giá trị tốt nhất tìm được  
return maxArea;  
}
```

```
int main() {  
    int n, in;  
    cin >> n;  
  
    vector<int> inPut;  
    // lấy danh sách chiều cao của các cột trong mảng đầu vào  
    for (int i = 0; i < n; ++i) {  
        cin >> in;  
        inPut.push_back(in);  
    }  
    // in ra giá trị diện tích tối đa  
    cout << largestRectangleInHistogram(inPut) << endl;  
  
    cout << "*****" << endl;  
    cout << "Hoang The Anh - 20215301" << endl;  
    cout << "*****" << endl;  
  
    return 0;  
}
```

CASE1:

Hoàng Thé Anh - 20215301

```
main.cpp x : input.txt x
6     stack<int> s; ▲ 2 ✘ 54 ▾
7     int maxArea = 0;
8     int area = 0;
9     int i = 0;
10    int size = histogram.size();
11
12    // duyệt từng phần tử để tính cận trái và phải
13    while (i < size) {
14        // tại s.empty() là trường hợp đặc biệt: xét phần tử đầu tiên
15        // tại trường hợp phần tử bên phải lớn hoặc bằng phần tử hiện tại lùi
16        if (s.empty() || histogram[s.top()] <= histogram[i]) {
17            // thêm cận trái của phần tử tiếp theo
18            s.push(x[i]);
19            // tăng giá trị của index để xét tiếp
20            i++;
21        } else {
22            // tại trường hợp lọt vào else là tìm thấy cận phải của phần tử thứ i
23            // định của stack đại diện cho cận trái
24            int top = s.top();
25            s.pop();
26            // tính số ô theo 2 cận trái phải, tại trường hợp đặc biệt là ko có cả
27            // tại trường hợp bình thường ta sẽ tính số ô với 2 cận là i và s.top()
28            // tương đương với việc số ô tính từ s.top()+1 đến i-1 nên ta có công
29            // diện tích bằng chiều cao * số ô chiếm chỗ
30            area = histogram[top] * (s.empty() ? i : (i - s.top() - 1));
31            // cập nhật giá trị diện tích tối đa nếu tìm được giá trị tốt hơn
32            if (area > maxArea) {
33                maxArea = area;
34            }
35        }
36    }
largestRectangleInHistogram
```

D:\OneDrive\Documents\StudyMaterials\KTLT\TH\45954

Hoang The Anh - 20215301

Process finished with exit code 0

Case 2:

```
main.cpp x : input.txt x
6     stack<int> s; ▲ 2 ✘ 54 ▾
7     int maxArea = 0;
8     int area = 0;
9     int i = 0;
10    int size = histogram.size();
11
12    // duyệt từng phần tử để tính cận trái và phải
13    while (i < size) {
14        // tại s.empty() là trường hợp đặc biệt: xét phần tử đầu tiên
15        // tại trường hợp phần tử bên phải lớn hoặc bằng phần tử hiện tại lùi
16        if (s.empty() || histogram[s.top()] <= histogram[i]) {
17            // thêm cận trái của phần tử tiếp theo
18            s.push(x[i]);
19            // tăng giá trị của index để xét tiếp
20            i++;
21        } else {
22            // tại trường hợp lọt vào else là tìm thấy cận phải của phần tử thứ i
23            // định của stack đại diện cho cận trái
24            int top = s.top();
25            s.pop();
26            // tính số ô theo 2 cận trái phải, tại trường hợp đặc biệt là ko có cả
27            // tại trường hợp bình thường ta sẽ tính số ô với 2 cận là i và s.top()
28            // tương đương với việc số ô tính từ s.top()+1 đến i-1 nên ta có công
29            // diện tích bằng chiều cao * số ô chiếm chỗ
30            area = histogram[top] * (s.empty() ? i : (i - s.top() - 1));
31            // cập nhật giá trị diện tích tối đa nếu tìm được giá trị tốt hơn
32            if (area > maxArea) {
33                maxArea = area;
34            }
35        }
36    }
largestRectangleInHistogram
```

D:\OneDrive\Documents\StudyMaterials\KTLT\TH\41684

Hoang The Anh - 20215301

Process finished with exit code 0

Case 3:

```

6   stack<int> s;
7   int maxArea = 0;
8   int area = 0;
9   int i = 0;
10  int size = histogram.size();
11
12  // duyệt từng phần tử để tính cận trái và phải
13  while (i < size) {
14      // tại s.empty() là trường hợp đặc biệt: xét phần tử đầu tiên
15      // tại trường hợp phần tử bên phải lớn hoặc bằng phần tử hiện tại lứa
16      if (s.empty() || histogram[s.top()] <= histogram[i]) {
17          // thêm cận trái của phần tử tiếp theo
18          s.push(i);
19          // tăng giá trị của index để xét tiếp
20          i++;
21      } else {
22          // tại trường hợp lọt vào else là tìm thấy cận phải của phần tử thứ i
23          // định của stack đại diện cho cận trái
24          int top = s.top();
25          s.pop();
26          // tính số ô theo 2 cận trái phải. tại trường hợp đặc biệt là ko có cả
27          // tại trường hợp bình thường ta sẽ tính số ô với 2 cận là i và s.top()
28          // tương đương với việc số ô tính từ s.top() + 1 đến i - 1 nên ta có công
29          // diện tích bằng chiều cao * số ô chiếm chỗ
30          area = histogram[top] * (s.empty() ? i : (i - s.top() - 1));
31          // cập nhật giá trị diện tích tối đa nếu tìm được giá trị tốt hơn
32          if (area > maxArea) {
33              maxArea = area;
34          }
35      }
36  }
37
38  largestRectangleInHistogram

```

The output window shows the execution of the program with input file "input.txt". The output consists of two parts: the run configuration (TN) and the process output. The run configuration shows the path to the input file and the command executed. The process output shows the contents of the input file, which is a single line of 10000 '1's, followed by the message "Process finished with exit code 0".

Case 4:

```

6   stack<int> s;
7   int maxArea = 0;
8   int area = 0;
9   int i = 0;
10  int size = histogram.size();
11
12  // duyệt từng phần tử để tính cận trái và phải
13  while (i < size) {
14      // tại s.empty() là trường hợp đặc biệt: xét phần tử đầu tiên
15      // tại trường hợp phần tử bên phải lớn hoặc bằng phần tử hiện tại lứa
16      if (s.empty() || histogram[s.top()] <= histogram[i]) {
17          // thêm cận trái của phần tử tiếp theo
18          s.push(i);
19          // tăng giá trị của index để xét tiếp
20          i++;
21      } else {
22          // tại trường hợp lọt vào else là tìm thấy cận phải của phần tử thứ i
23          // định của stack đại diện cho cận trái
24          int top = s.top();
25          s.pop();
26          // tính số ô theo 2 cận trái phải. tại trường hợp đặc biệt là ko có cả
27          // tại trường hợp bình thường ta sẽ tính số ô với 2 cận là i và s.top()
28          // tương đương với việc số ô tính từ s.top() + 1 đến i - 1 nên ta có công
29          // diện tích bằng chiều cao * số ô chiếm chỗ
30          area = histogram[top] * (s.empty() ? i : (i - s.top() - 1));
31          // cập nhật giá trị diện tích tối đa nếu tìm được giá trị tốt hơn
32          if (area > maxArea) {
33              maxArea = area;
34          }
35      }
36  }
37
38  largestRectangleInHistogram

```

The output window shows the execution of the program with input file "input.txt". The run configuration shows the path to the input file and the command executed. The process output shows the contents of the input file, followed by the message "Process finished with exit code 0".

Bài tập 13

#include <bits/stdc++.h>

using namespace std;

string input;

```
// hàm kiểm tra số lượng bit 0 và 1
bool check(const string &substring) {
    // khởi tạo 2 biến đếm số lượng bit 0 và 1
    int numOf1 = 0, numOf0 = 0;

    // tìm bit 1 bằng phương thức find
    size_t found1 = substring.find('1');

    // khi tìm được bit 1
    while (found1 != string::npos) {
        // tăng biến đếm lên 1
        numOf1++;

        // tìm bit 1 tiếp theo
        found1 = substring.find('1', found1 + 1);
    }

    // tìm bit 0 bằng phương thức find
    size_t found0 = substring.find('0');

    // khi tìm được bit 0
    while (found0 != string::npos) {
        // tăng biến đếm lên 1
        numOf0++;

        // tìm bit 0 kế tiếp
        found0 = substring.find('0', found0 + 1);
    }
}
```

Hoàng Thê Anh - 20215301

```
// trả về so sánh số lượng bit 0 và 1  
return numOf0 == numOf1;  
}  
  
int main() {  
    int res = 0;  
    cin >> input;  
    // duyệt các dãy con có số phần tử là chẵn  
    // tính vùng duyệt của biến i sao cho 2*i số lượng phần tử của dãy con  
    int range = input.length() / 2;  
    // duyệt từng giá trị của i ứng với dãy con có 2 phần tử đến số phần tử lớn nhất là chẵn  
    for (int i = 1; i <= range; ++i) {  
        // duyệt j từ 0 là điểm bắt đầu của dãy con  
        for (int j = 0; j + 2 * i <= input.length(); ++j) {  
            // lấy dãy con bắt đầu từ j và có 2*i phần tử  
            // dùng hàm check kiểm tra xem dãy con có số lượng phần tử 0 và 1 bằng nhau  
            // hay không  
            if (check(input.substr(j, 2 * i))) {  
                // nếu bằng nhau thì tăng biến đếm lên 1  
                res++;  
            }  
        }  
    }  
    // in ra kết quả tính được  
    cout << res << endl;  
  
    cout << "*****" << endl;  
    cout << "Hoang The Anh - 20215301" << endl;
```

```
cout << "*****" << endl;
```

```
return 0;
```

```
}
```

Case 1:

The screenshot shows a code editor with two tabs: `main.cpp` and `input.txt`. The `main.cpp` tab displays C++ code for counting the number of 0s and 1s in a string. The `input.txt` tab shows binary data. The run output window shows the program's execution path, the student's name, and the exit code.

```
main.cpp
1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 string input;
6
7 // hàm kiểm tra số lượng bit 0 và 1
8 bool check(const string &substring) {
9     // khởi tạo 2 biến đếm số lượng bit 0 và 1
10    int numOf1 = 0, numOf0 = 0;
11
12    // tìm bit 1 bằng phương thức find
13    size_t found1 = substring.find('1');
14    // khi tìm được bit 1
15    while (found1 != string::npos) {
16        // tăng biến đếm lên 1
17        numOf1++;
18        // tìm bit 1 tiếp theo
19        found1 = substring.find('1', pos: found1 + 1);
20    }
21
22    // tìm bit 0 bằng phương thức find
23    size_t found0 = substring.find('0');
24    // khi tìm được bit 0
25    while (found0 != string::npos) {
26        // tăng biến đếm lên 1
27        numOf0++;
28        // tìm bit 0 kế tiếp
29        found0 = substring.find('0', pos: found0 + 1);
30    }
31    // trả về so sánh số lượng bit 0 và 1
}

```

input.txt

```
1 100110011010101010101000111111111000010100011010100010010
\$01011011101011110111
```

Run TN

```
D:\OneDrive\Documents\StudyMaterials\KTLT\TH\
583
*****
Hoang The Anh - 20215301
*****
Process finished with exit code 0
```

Case 2:

Hoàng Thé Anh - 20215301

The screenshot shows a code editor with a C++ file named `main.cpp` and a terminal window. The code in `main.cpp` is as follows:

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 string input;
6
7 // hàm kiểm tra số lượng bit 0 và 1
8 bool check(const string &substring) {
9     // khởi tạo 2 biến đếm số lượng bit 0 và 1
10    int numOf1 = 0, numOf0 = 0;
11
12    // tìm bit 1 bằng phương thức find
13    size_t found1 = substring.find('1');
14    // khi tìm được bit 1
15    while (found1 != string::npos) {
16        // tăng biến đếm lên 1
17        numOf1++;
18        // tìm bit 1 tiếp theo
19        found1 = substring.find('1', pos: found1 + 1);
20    }
21
22    // tìm bit 0 bằng phương thức find
23    size_t found0 = substring.find('0');
24    // khi tìm được bit 0
25    while (found0 != string::npos) {
26        // tăng biến đếm lên 1
27        numOf0++;
28        // tìm bit 0 kế tiếp
29        found0 = substring.find('0', pos: found0 + 1);
30    }
31    // trả về so sánh số lượng bit 0 và 1
32 }
```

The terminal window shows the output of the program. It reads from a file named `input.txt` containing binary data. The output shows the results of the `check` function for different substrings, including the name "Hoàng Thé Anh - 20215301". The process finished with exit code 0.

Case 3: