

## Contents

Bài thực hành số 4 – Tuần 38.....	2
<b>Bài tập 4.1.</b> Đảo ngược một danh sách liên kết đơn .....	2
<b>Bài tập 4.2.</b> Một điểm trong không gian 2 chiều được biểu diễn bằng pair. Hãy viết hàm tính diện tích tam giác theo tọa độ 3 đỉnh. ....	7
<b>Bài tập 4.3.</b> Một vector trong không gian 3 chiều được biểu diễn bằng tuple<double, double, double>. Hãy viết hàm tính tích có hướng của 2 vector. ....	9
<b>Bài tập 4.4.</b> Cho hai std::vector, hãy xóa hết các phần tử chẵn, sắp xếp giảm dần các số trong cả 2 vector và trộn lại thành một vector cũng được sắp xếp giảm dần.....	12
<b>Bài tập 4.5.</b> Viết hàm void dfs(vector< list<int> > adj) thực hiện thuật toán DFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề. Đồ thị có n đỉnh được đánh số từ 1 đến n. Thuật toán DFS xuất phát từ đỉnh 1. Các đỉnh được thăm theo thứ tự ưu tiên từ trái sang phải trong danh sách kề. Yêu cầu hàm trả ra thứ tự các đỉnh được thăm (những đỉnh không thể thăm từ đỉnh 1 thì không phải in ra). ....	19
<b>Bài tập 4.6.</b> Viết hàm void bfs(vector< list<int> > adj) thực hiện thuật toán BFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề. Đồ thị có n đỉnh được đánh số từ 1 đến n. Thuật toán BFS xuất phát từ đỉnh 1. Các đỉnh được thăm theo thứ tự ưu tiên từ trái sang phải trong danh sách kề. Yêu cầu hàm trả ra thứ tự các đỉnh được thăm (những đỉnh không thể thăm từ đỉnh 1 thì không phải in ra). ....	23
<b>Bài tập 4.7.</b> Viết các hàm thực hiện các phép giao và hợp của hai tập hợp được biểu diễn bằng set... ..	27
<b>Bài tập 4.8.</b> Viết các hàm thực hiện các phép giao và hợp của hai tập hợp mờ được biểu diễn bằng map. ....	31
<b>Bài tập 4.9.</b> Cài đặt thuật toán Dijkstra trên đồ thị vô hướng được biểu diễn bằng danh sách kề sử dụng std::priority_queue .....	36
<b>Bài tập 4.10.</b> Xây dựng một máy tìm kiếm (search engine) đơn giản. ....	40
<b>Bài tập 4.11.</b> Bức tường bao quanh một lâu đài nọ được cấu thành từ n đoạn tường được đánh số từ 1 đến n . Quân giặc lên kế hoạch tấn công lâu đài bằng cách gửi ai tên giặc đánh vào đoạn tường thứ i. Để bảo vệ lâu đài có tất cả s lính. ....	54
<b>Bài tập 4.12.</b> Cho một lược đồ gồm n cột chữ nhật liên tiếp nhau có chiều rộng bằng 1 và chiều cao lần lượt là các số nguyên không âm h1,h2,...,hn.....	63
<b>Bài tập 4.13.</b> Cho một xâu nhị phân độ dài n . Hãy viết chương trình đếm số lượng xâu con chứa số ký tự 0 và số ký tự 1 bằng nhau. ....	74

**Nguyễn Văn Duy – 20215334**  
Bài thực hành số 4 – Tuần 38

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

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

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

```
1 // Nguyen Van Duy - 20215334
2 /*
3 Bài 4.1. Hãy hoàn thiện các hàm thao tác trên một danh sách liên kết:
4 • Thêm một phần tử vào đầu danh sách liên kết
5 • In danh sách
6 • Đảo ngược danh sách liên kết (yêu cầu độ phức tạp thời gian  $O(N)$  và chi phí bộ nhớ dùng thêm  $O(1)$ )
7 */

8 #include <iostream>
9 using namespace std;
10 struct Node {
11     int data;
12     Node* next;
13
14     Node(int data) {
15         this->data = data;
16         next = NULL;
17     }
18 };
19
20 // push a new element to the beginning of the list
21 Node* prepend(Node* head, int data) {
22     /******
23     # YOUR CODE HERE #
24     *****/
25     Node* newNode = new Node(data); // new node is new head
26     newNode->next = head; // point to head
27     return newNode; // return new node
28 }
29
30 // print the list content on a line
31 void print(Node* head) {
32     /******
33     # YOUR CODE HERE #
34     *****/
35     Node* temp = head; // temp node
36     while (temp != NULL) { // loop through
37         cout << temp->data << " "; // print
38         temp = temp->next; // next node
39     }
40     cout << endl; // end of
41 }
```

```

42
43 // return the new head of the reversed list
44 Node* reverse(Node* head) {
45     /*****
46     # YOUR CODE HERE #
47     *****/
48     Node* prev = NULL; // previous
49     Node* curr = head; // current
50     Node* next = NULL; // next
51     while (curr != NULL) { // loop through
52         next = curr->next; // save next node
53         curr->next = prev; // move the cursor to previous node
54         prev = curr; // move to next
55         curr = next; // move to next
56     }
57     return prev;
58 }
59
60 int main() {
61     int n, u;
62     cin >> n;
63     Node* head = NULL;
64     for (int i = 0; i < n; ++i){
65         cin >> u;
66         head = prepend(head, u);
67     }
68
69     cout << "Original list: ";
70     print(head);
71
72     head = reverse(head);
73
74     cout << "Reversed list: ";
75     print(head);
76
77     return 0;
78 }
79 // Nguyen Van Duy - 20215334

```

	Dữ liệu đầu vào	Kết quả đúng	Kết quả chương trình
✓	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

Hoàn thành được tất cả các bộ mẫu ✓

// Nguyen Van Duy - 20215334

/\*

Bài 4.1. Hãy hoàn thiện các hàm thao tác trên một danh sách liên kết:

- Thêm một phần tử vào đầu danh sách liên kết
- In danh sách

- Đảo ngược danh sách liên kết (yêu cầu độ phức tạp thời gian  $O(N)$  và chi phí bộ nhớ dùng thêm  $O(1)$ )

```
*/  
  
#include <iostream>  
  
using namespace std;  
  
struct Node {  
    int data;  
    Node* next;  
  
    Node(int data) {  
        this->data = data;  
        next = NULL;  
    }  
};  
  
// push a new element to the beginning of the list  
Node* prepend(Node* head, int data) {  
    /*****  
    # YOUR CODE HERE #  
    *****/  
  
    Node* newNode = new Node(data); // new node is new head  
    newNode->next = head; // point to head  
    return newNode; // return new node  
}  
  
// print the list content on a line  
void print(Node* head) {  
    /*****
```

```
# YOUR CODE HERE #

*****/

Node* temp = head; // temp node
while (temp != NULL) { // loop through
    cout << temp->data << " "; // print
    temp = temp->next; // next node
}
cout << endl; // end of
}

// return the new head of the reversed list
Node* reverse(Node* head) {
    /*****
    # YOUR CODE HERE #
    *****/

    Node* prev = NULL; // previous
    Node* curr = head; // current
    Node* next = NULL; // next
    while (curr != NULL) { // loop through
        next = curr->next; // save next node
        curr->next = prev; // move the cursor to previous node
        prev = curr; // move to next
        curr = next; // move to next
    }
    return prev;
}
```

```
int main() {  
    int n, u;  
    cin >> n;  
    Node* head = NULL;  
    for (int i = 0; i < n; ++i){  
        cin >> u;  
        head = prepend(head, u);  
    }  
  
    cout << "Original list: ";  
    print(head);  
  
    head = reverse(head);  
  
    cout << "Reversed list: ";  
    print(head);  
  
    return 0;  
}  
// Nguyen Van Duy - 20215334
```

**Bài tập 4.2.** Một điểm trong không gian 2 chiều được biểu diễn bằng pair. Hãy viết hàm tính diện tích tam giác theo tọa độ 3 đỉnh.

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

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

# YOUR CODE HERE #

\*\*\*\*\*/

}

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

using Point = pair<double, double>;

```
1 // Nguyen Van Duy - 20215334
2 /*
3 Bài 4.2. Một điểm trong không gian 2 chiều được biểu diễn bằng pair.
4 Hãy viết hàm double area(Point a, Point b, Point c) tính diện tích tam giác theo tọa độ 3 đỉnh.
5 Trong đó, Point là kiểu được định nghĩa sẵn trong trình chấm như sau: using Point = pair<double, double>;
6 */
7 #include <iostream>
8 #include <cmath>
9 #include <iomanip>
10 #include <utility>
11 using namespace std;
12 using Point = pair<double, double>;
13
14 double area(Point a, Point b, Point c) {
15     /******
16     # YOUR CODE HERE #
17     *****/
18
19     // S = 0.5 * |(x_C - x_A)(y_B - y_A) - (x_B - x_A)(y_C - y_A)|
20     return 0.5 * fabs((c.first - a.first)*(b.second - a.second) - (b.first - a.first)*(c.second - a.second));
21 }
22
23 // Nguyen Van Duy - 20215334
```

	Test	Kết quả đúng	Kết quả chương trình	
✓	cout << setprecision(2) << fixed; cout << area({1, 2}, {2.5, 10}, {15, -5.25}) << endl;	61.44	61.44	✓
✓	cout << setprecision(2) << fixed; cout << area({1, 2.5}, {2.5, 15}, {-5.2, -5.75}) << endl;	32.56	32.56	✓

Hoàn thành được tất cả các bộ mẫu ✓

// Nguyen Van Duy - 20215334

/\*

Bài 4.2. Một điểm trong không gian 2 chiều được biểu diễn bằng pair.

Hãy viết hàm double area(Point a, Point b, Point c) tính diện tích tam giác theo tọa độ 3 đỉnh.

Trong đó, Point là kiểu được định nghĩa sẵn trong trình chấm như sau: using Point = pair<double, double>;

\*/

```
#include <iostream>
```

```
#include <cmath>
```

```
#include <iomanip>
```

```
#include <utility>
```

```
using namespace std;
```

```
using Point = pair<double, double>;
```

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

```
    /*****
```

```
    # YOUR CODE HERE #
```

```
    *****/
```

```
    //  $S = 0.5 * |(x_C - x_A)(y_B - y_A) - (x_B - x_A)(y_C - y_A)|$ 
```

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

```
}
```

```
// Nguyen Van Duy - 20215334
```



**Bài tập 4.3.** Một vector trong không gian 3 chiều được biểu diễn bằng tuple<double, double, double>. Hãy viết hàm tính tích có hướng của 2 vector.

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

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

# YOUR CODE HERE #

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

}

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

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

```

1 // Nguyen Van Duy - 20215334
2 /*
3 Bài 4.3. Một vector trong không gian 3 chiều được biểu diễn bằng tuple<double, double, double>.
4 Hãy viết hàm Vector cross_product(Vector a, Vector b) tính tích có hướng của 2 vector.
5 Trong đó Vector là kiểu dữ liệu được định nghĩa sẵn trong trình chấm như sau:
6 using Vector = tuple<double, double, double>;
7 */
8 #include <iostream>
9 #include <cmath>
10 #include <iomanip>
11 using namespace std;
12 using Vector = tuple<double, double, double>;
13
14 Vector cross_product(Vector a, Vector b) {
15     /*
16     # YOUR CODE HERE #
17     */
18
19     // res_X = a_Y * b_Z - a_Z * b_Y
20     // res_Y = a_Z * b_X - a_X * b_Z
21     // res_Z = a_X * b_Y - a_Y * b_X
22     return make_tuple(get<1>(a) * get<2>(b) - get<2>(a) * get<1>(b),
23                       get<2>(a) * get<0>(b) - get<0>(a) * get<2>(b),
24                       get<0>(a) * get<1>(b) - get<1>(a) * get<0>(b));
25 }
26
27 // Nguyen Van Duy - 20215334

```

**Nguyễn Văn Duy – 20215334**

	Test	Kết quả đúng	Kết quả chương trình	
✓	<pre>cout &lt;&lt; setprecision(2) &lt;&lt; fixed; Vector a {1.2, 4, -0.5}; Vector b {1.5, -2, 2.5}; Vector c = cross_product(a, b); cout &lt;&lt; get&lt;0&gt;(c) &lt;&lt; ' ' &lt;&lt; get&lt;1&gt;(c) &lt;&lt; ' ' &lt;&lt; get&lt;2&gt;(c) &lt;&lt; endl;</pre>	9.00 -3.75 -8.40	9.00 -3.75 -8.40	✓
✓	<pre>cout &lt;&lt; setprecision(2) &lt;&lt; fixed; Vector a {-2.2, 4.5, -1.5}; Vector b {3.5, -7, 7.5}; Vector c = cross_product(a, b); cout &lt;&lt; get&lt;0&gt;(c) &lt;&lt; ' ' &lt;&lt; get&lt;1&gt;(c) &lt;&lt; ' ' &lt;&lt; get&lt;2&gt;(c) &lt;&lt; endl;</pre>	23.25 11.25 -0.35	23.25 11.25 -0.35	✓

Hoàn thành được tất cả các bộ mẫu ✓

// Nguyen Van Duy - 20215334

/\*

Bài 4.3. Một vector trong không gian 3 chiều được biểu diễn bằng tuple<double, double, double>.

Hãy viết hàm Vector cross\_product(Vector a, Vector b) tính tích có hướng của 2 vector.

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

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

\*/

#include <iostream>

#include <cmath>

#include <iomanip>

using namespace std;

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

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

/\*\*\*\*\*

# YOUR CODE HERE #

\*\*\*\*\*/

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

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

**Nguyễn Văn Duy – 20215334**

```
// res_Z = a_X * b_Y - a_Y * b_X  
return make_tuple(get<1>(a) * get<2>(b) - get<2>(a) * get<1>(b),  
                  get<2>(a) * get<0>(b) - get<0>(a) * get<2>(b),  
                  get<0>(a) * get<1>(b) - get<1>(a) * get<0>(b));  
}
```

// Nguyen Van Duy - 20215334

**Bài tập 4.4.** Cho hai `std::vector`, hãy xóa hết các phần tử chẵn, sắp xếp giảm dần các số trong cả 2 vector và trộn lại thành một vector cũng được sắp xếp giảm dần

```

1 // Nguyen Van Duy - 20215334
2 /*
3 Bài 4.4. Cho hai vector, hãy xóa hết các phần tử chẵn,
4 sắp xếp giảm dần các số trong cả 2 vector và
5 trộn lại thành một vector cũng được sắp xếp giảm dần.
6 */
7 #include <iostream>
8 #include <vector>
9 #include <algorithm>
10 using namespace std;
11 void print_vector(const vector<int> &a) {
12     for (int v : a) cout << v << ' ';
13     cout << endl;
14 }
15
16 void delete_even(vector<int> &a) {
17     /*****
18      # YOUR CODE HERE #
19      *****/
20
21     for (size_t i = 0; i < a.size(); i++) { // loop
22         if (a[i] % 2 == 0) { // check even numbers
23             a.erase(a.begin() + i); // remove
24             i--; // decrement to keep index
25         }
26     }
27 }
28
29 void sort_decrease(vector<int> &a) {
30     /*****
31      # YOUR CODE HERE #
32      *****/
33
34     sort(a.begin(), a.end(), [] (int a, int b) -> bool {
35         return a >= b; // stable sort by ascending order
36     });
37 }
38

```

```

39 vector<int> merge_vectors(const vector<int> &a, const vector<int> &b) {
40     /*****
41     # YOUR CODE HERE #
42     *****/
43
44     // merge 2 sorted vectors, return sorted vector
45     vector<int> c; // result
46     size_t i = 0, j = 0; // index
47     while (i < a.size() && j < b.size()) { // loop until a ends or b ends
48         // add smaller element
49         if (a[i] > b[j]) {
50             c.push_back(a[i]);
51             i++; // increment index of vector a
52         } else {
53             c.push_back(b[j]);
54             j++; // increment index of vector b
55         }
56     }
57
58     while (i < a.size()) {
59         c.push_back(a[i]); // add the remaining elements of the vector a
60         i++;
61     }
62     while (j < b.size()) {
63         c.push_back(b[j]); // add the remaining elements of the vector b
64         j++;
65     }
66     return c; // return
67 }

```

```

68 int main() {
69     int m, n, u;
70     std::vector<int> a, b;
71
72     std::cin >> m >> n;
73     for(int i = 0; i < m; i++){
74         std::cin >> u;
75         a.push_back(u);
76     }
77     for(int i = 0; i < n; i++){
78         std::cin >> u;
79         b.push_back(u);
80     }
81
82     delete_even(a);
83     cout << "Odd elements of a: ";
84     print_vector(a);
85
86     delete_even(b);
87     cout << "Odd elements of b: ";
88     print_vector(b);
89
90     sort_decrease(a);
91     cout << "Decreasingly sorted a: ";
92     print_vector(a);
93

```

```

94     sort_decrease(b);
95     cout << "Decreasingly sorted b: ";
96     print_vector(b);
97
98     vector<int> c = merge_vectors(a, b);
99     cout << "Decreasingly sorted c: ";
100    print_vector(c);
101
102    return 0;
103 }
104 // Nguyen Van Duy - 20215334

```

	Dữ liệu đầu vào	Kết quả đúng	
✓	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	O O D D D
✓	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	O O D D D
Hoàn thành được tất cả các bộ mẫu ✓			

// Nguyen Van Duy - 20215334

/\*

Bài 4.4. Cho hai vector, hãy xóa hết các phần tử chẵn,  
sắp xếp giảm dần các số trong cả 2 vector và  
trộn lại thành một vector cũng được sắp xếp giảm dần.

\*/

```
#include <iostream>
```

```
#include <vector>
```

```
#include <algorithm>
```

```
using namespace std;
```

```
void print_vector(const vector<int> &a) {
```

```
    for (int v : a) cout << v << ' ';
```

```
    cout << endl;
```

```
}
```

```
void delete_even(vector<int> &a) {
    /***/
    # YOUR CODE HERE #
    /***/

    for (size_t i = 0; i < a.size(); i++) { // loop
        if (a[i] % 2 == 0) { // check even numbers
            a.erase(a.begin() + i); // remove
            i--; // decrement to keep index
        }
    }
}

void sort_decrease(vector<int> &a) {
    /***/
    # YOUR CODE HERE #
    /***/

    sort(a.begin(), a.end(), [] (int a, int b) -> bool {
        return a >= b; // stable sort by ascending order
    });
}

vector<int> merge_vectors(const vector<int> &a, const vector<int> &b) {
    /***/
    # YOUR CODE HERE #
}
```

\*\*\*\*\*/

```
// merge 2 sorted vectors, return sorted vector
vector<int> c; // result
size_t i = 0, j = 0; // index
while (i < a.size() && j < b.size()) { // loop until a ends or b ends
    // add smaller element
    if (a[i] > b[j]) {
        c.push_back(a[i]);
        i++; // increment index of vector a
    } else {
        c.push_back(b[j]);
        j++; // increment index of vector b
    }
}
while (i < a.size()) {
    c.push_back(a[i]); // add the remaining elements of the vector a
    i++;
}
while (j < b.size()) {
    c.push_back(b[j]); // add the remaining elements of the vector b
    j++;
}
return c; // return
}

int main() {
```



```
int m, n, u;
std::vector<int> a, b;

std::cin >> m >> n;
for(int i = 0; i < m; i++){
    std::cin >> u;
    a.push_back(u);
}
for(int i = 0; i < n; i++){
    std::cin >> u;
    b.push_back(u);
}

delete_even(a);
cout << "Odd elements of a: ";
print_vector(a);

delete_even(b);
cout << "Odd elements of b: ";
print_vector(b);

sort_decrease(a);
cout << "Decreasingly sorted a: ";
print_vector(a);

sort_decrease(b);
cout << "Decreasingly sorted b: ";
```

```
print_vector(b);

vector<int> c = merge_vectors(a, b);
cout << "Decreasingly sorted c: ";
print_vector(c);

return 0;
}
// Nguyen Van Duy - 20215334
```

**Bài tập 4.5.** Viết hàm void dfs(vector< list<int> > adj) thực hiện thuật toán DFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề. Đồ thị có n đỉnh được đánh số từ 1 đến n. Thuật toán DFS xuất phát từ đỉnh 1. Các đỉnh được thăm theo thứ tự ưu tiên từ trái sang phải trong danh sách kề. Yêu cầu hàm trả ra thứ tự các đỉnh được thăm (những đỉnh không thể thăm từ đỉnh 1 thì không phải in ra).

```

1 // Nguyen Van Duy - 20215334
2 /*
3 Bài 4.5. Viết hàm thực hiện thuật toán DFS không sử dụng đệ quy
4 trên đồ thị biểu diễn bằng danh sách kề vector< list<int> > .
5 Đồ thị có n đỉnh được đánh số từ 1 đến n.
6 Thuật toán DFS xuất phát từ đỉnh 1. Các đỉnh được thăm theo thứ tự ưu tiên
7 từ trái sang phải trong danh sách kề.
8 Yêu cầu hàm trả ra thứ tự các đỉnh được thăm
9 (những đỉnh không thể thăm từ đỉnh 1 thì không phải in ra).
10 */
11 #include <iostream>
12 #include <vector>
13 #include <list>
14 #include <stack>
15
16 using namespace std;
17
18 void dfs(vector< list<int> > adj) {
19     stack<int> S;
20     vector<bool> visited(adj.size());
21     S.push(1); // Bắt đầu từ đỉnh số 1
22
23     /*****
24     # YOUR CODE HERE #
25     *****/
26
27     while (!S.empty()) { // loop
28         int u = S.top(); // get top
29         S.pop(); // pop from the top of the stack
30         if (!visited[u]) { // check if u is not visited
31             visited[u] = true; // visit u
32             cout << u << endl; // print
33             list<int>::iterator it; // iterator
34             for (it = adj[u].end(); it-- != adj[u].begin(); ) { // loop through adjacent of u
35                 if (!visited[*it]) { // check if it is not visited
36                     S.push(*it); // push it to stack to visit later
37                 }
38             }
39         }
40     }
41 }
42
43 // Nguyen Van Duy - 20215334

```

	Test	Kết quả đúng	Kết quả chương trình	
✓	<pre>int n = 7; vector&lt; list&lt;int&gt; &gt; 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);</pre>	<pre>1 2 4 7 3 5</pre>	<pre>1 2 4 7 3 5</pre>	✓
✓	<pre>int n = 10; vector&lt; list&lt;int&gt; &gt; 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);</pre>	<pre>1 2 7 3 10 9 4 8 6</pre>	<pre>1 2 7 3 10 9 4 8 6</pre>	✓

Hoàn thành được tất cả các bộ mẫu ✓

// Nguyen Van Duy - 20215334

/\*

Bài 4.5. Viết hàm thực hiện thuật toán DFS không sử dụng đệ quy

trên đồ thị biểu diễn bằng danh sách kề `vector< list<int> >` .

Đồ thị có n đỉnh được đánh số từ 1 đến n.

Thuật toán DFS xuất phát từ đỉnh 1. Các đỉnh được thăm theo thứ tự ưu tiên

từ trái sang phải trong danh sách kề.

Yêu cầu hàm trả ra thứ tự các đỉnh được thăm

(những đỉnh không thể thăm từ đỉnh 1 thì không phải in ra).

```
*/
```

```
#include <iostream>
```

```
#include <vector>
```

```
#include <list>
```

```
#include <stack>
```

```
using namespace std;
```

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

```
    stack<int> S;
```

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

```
    S.push(1); // Bắt đầu từ đỉnh số 1
```

```
    /*****
```

```
    # YOUR CODE HERE #
```

```
    *****/
```

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

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

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

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

```
        visited[u] = true; // visit u
```

```
        cout << u << endl; // print
```

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

**Nguyễn Văn Duy – 20215334**

```
for (it = adj[u].end(); it-- != adj[u].begin(); ) { // loop through adjacent of u
    if (!visited[*it]) { // check if it is not visited
        S.push(*it); // push it to stack to visit later
    }
}
}
```

// Nguyen Van Duy - 20215334

**Bài tập 4.6.** Viết hàm void bfs(vector< list<int> > adj) thực hiện thuật toán BFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề. Đồ thị có n đỉnh được đánh số từ 1 đến n. Thuật toán BFS xuất phát từ đỉnh 1. Các đỉnh được thăm theo thứ tự ưu tiên từ trái sang phải trong danh sách kề. Yêu cầu hàm trả ra thứ tự các đỉnh được thăm (những đỉnh không thể thăm từ đỉnh 1 thì không phải in ra).

```

1 // Nguyen Van Duy - 20215334
2 /*
3 Bài 4.6. Viết hàm thực hiện thuật toán BFS không sử dụng đệ quy
4 trên đồ thị biểu diễn bằng danh sách kề vector< list<int> > .
5 Đồ thị có n đỉnh được đánh số từ 1 đến n.
6 Thuật toán BFS xuất phát từ đỉnh 1. Các đỉnh được thăm theo thứ tự ưu tiên
7 từ trái sang phải trong danh sách kề.
8 Yêu cầu hàm trả ra thứ tự các đỉnh được thăm
9 (những đỉnh không thể thăm từ đỉnh 1 thì không phải in ra).
10 */
11 #include <iostream>
12 #include <vector>
13 #include <list>
14 #include <queue>
15
16 using namespace std;
17

```

```

1 // Nguyen Van Duy - 20215334
2 /*
3 Bài 4.6. Viết hàm thực hiện thuật toán BFS không sử dụng đệ quy
4 trên đồ thị biểu diễn bằng danh sách kề vector< list<int> > .
5 Đồ thị có n đỉnh được đánh số từ 1 đến n.
6 Thuật toán BFS xuất phát từ đỉnh 1. Các đỉnh được thăm theo thứ tự ưu tiên
7 từ trái sang phải trong danh sách kề.
8 Yêu cầu hàm trả ra thứ tự các đỉnh được thăm
9 (những đỉnh không thể thăm từ đỉnh 1 thì không phải in ra).
10 */
11 #include <iostream>
12 #include <vector>
13 #include <list>
14 #include <queue>
15
16 using namespace std;
17

```

```

27     visited[1] = true; // mark 1
28     while (!Q.empty()) { // loop
29         int u = Q.front(); // get first
30         Q.pop(); // remove an element from the front of the queue
31         cout << u << endl; // print
32         for (auto v : adj[u]) { // loop through adjacent of u
33             if (!visited[v]) { // check if it is not visited
34                 Q.push(v); // push it to queue to visit later
35                 visited[v] = true; // visit v
36             }
37         }
38     }
39 }
40
41 // Nguyen Van Duy - 20215334

```

	Test	Kết quả đúng	Kết quả chương trình	
✓	<pre>int n = 7; vector&lt; list&lt;int&gt; &gt; 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&lt; list&lt;int&gt; &gt; 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); bfs(adj);</pre>	<pre>1 2 3 6 7 4 8 10 9</pre>	<pre>1 2 3 6 7 4 8 10 9</pre>	✓

Hoàn thành được tất cả các bộ mẫu ✓

// Nguyen Van Duy - 20215334

/\*

Bài 4.6. Viết hàm thực hiện thuật toán BFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề `vector< list<int> >` .

Đồ thị có n đỉnh được đánh số từ 1 đến n.



Thuật toán BFS xuất phát từ đỉnh 1. Các đỉnh được thăm theo thứ tự ưu tiên từ trái sang phải trong danh sách kề.

Yêu cầu hàm trả ra thứ tự các đỉnh được thăm

(những đỉnh không thể thăm từ đỉnh 1 thì không phải in ra).

\*/

```
#include <iostream>
```

```
#include <vector>
```

```
#include <list>
```

```
#include <queue>
```

```
using namespace std;
```

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

```
    queue<int> Q;
```

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

```
    Q.push(1); // Bắt đầu từ đỉnh số 1
```

```
    /*****
```

```
    # YOUR CODE HERE #
```

```
    *****/
```

```
    visited[1] = true; // mark 1
```

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

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

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

```
        cout << u << endl; // print
```

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

```
if (!visited[v]) { // check if it is not visited
    Q.push(v); // push it to queue to visit later
    visited[v] = true; // visit v
}
}
}
}
```

// Nguyen Van Duy - 20215334

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

```

1 // Nguyen Van Duy - 20215334
2 /*
3 Bài 4.7. Viết các hàm thực hiện các phép giao
4 và hợp của hai tập hợp được biểu diễn bằng set.
5 */
6 #include <iostream>
7 #include <set>
8
9 using namespace std;
10
11 template<class T>
12 set<T> set_union(const set<T> &a, const set<T> &b) {
13     /*****
14      # YOUR CODE HERE #
15      *****/
16
17     set<T> c; // result
18     for (auto x : a) { // for each element in the set<T> a
19         c.insert(x); // insert to result
20     }
21     for (auto x : b) { // for each element in the set<T> b
22         c.insert(x); // insert to result
23     }
24     return c; // return result
25 }
26
27 template<class T>
28 set<T> set_intersection(const set<T> &a, const set<T> &b) {
29     /*****
30      # YOUR CODE HERE #
31      *****/
32
33     set<T> c; // result
34     for (auto x : a) { // for each element in the set<T> a
35         if (b.find(x) != b.end()) { // check if the element is already in the set<T> b
36             c.insert(x); // add it to the set<T> c (result)
37         }
38     }
39     return c; // return result
40 }
41
42 template<class T>
43 void print_set(std::set<T> &a) {
44     for (const T &x : a) {
45         std::cout << x << ' ';
46     }
47     std::cout << std::endl;
48 }
49
50 // Nguyen Van Duy - 20215334

```

	Test	Kết quả đúng	Kết quả chương trình
✓	<pre>set&lt;int&gt; a = {1, 2, 3, 5, 7}; set&lt;int&gt; b = {2, 4, 5, 6, 9}; set&lt;int&gt; c = set_union(a, b); set&lt;int&gt; d = set_intersection(a, b);  cout &lt;&lt; "Union: "; print_set(c); cout &lt;&lt; "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 Intersection: 2 5</pre>
✓	<pre>std::set&lt;int&gt; a = {1, 9, 10, 6, 17, 8}; std::set&lt;int&gt; b = {2, 10, 5, 6, 9, -5, 12, 4, 15, 21}; std::set&lt;int&gt; c = set_union(a, b); std::set&lt;int&gt; d = set_intersection(a, b);  std::cout &lt;&lt; "Union: "; print_set(c); std::cout &lt;&lt; "Intersection: "; print_set(d);</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 Intersection: 6 9</pre>

Hoàn thành được tất cả các bộ mẫu ✓

// Nguyen Van Duy - 20215334

/\*

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

\*/

#include <iostream>

#include <set>

using namespace std;

template<class T>

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

    /\*\*\*\*\*

    # YOUR CODE HERE #

    \*\*\*\*\*/

    set<T> c; // result

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

```

        c.insert(x); // insert to result
    }
    for (auto x : b) { // for each element in the set<T> b
        c.insert(x); // insert to result
    }
    return c; // return result
}

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

    set<T> c; // result
    for (auto x : a) { // for each element in the set<T> a
        if (b.find(x) != b.end()) { // check if the element is already in the set<T> b
            c.insert(x); // add it to the set<T> c (result)
        }
    }
    return c; // return result
}

```

```

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

```

```
}  
std::cout << std::endl;  
}
```

```
// Nguyen Van Duy - 20215334
```

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

Trong đó mỗi phần tử được gán cho một số thực trong đoạn  $[0..1]$  biểu thị độ thuộc của phần tử trong tập hợp, với độ thuộc bằng 1 nghĩa là phần tử chắc chắn thuộc vào tập hợp và ngược lại độ thuộc bằng 0 nghĩa là phần tử chắc chắn không thuộc trong tập hợp.

Phép giao và hợp của 2 tập hợp được thực hiện trên các cặp phần tử bằng nhau của 2 tập hợp, với độ thuộc mới được tính bằng phép toán min và max của hai độ thuộc.

```

1 // Nguyen Van Duy - 20215334
2 /*
3 Bài 4.8. Viết các hàm thực hiện các phép giao và hợp của hai tập hợp mờ được biểu diễn bằng map.
4
5 Trong đó mỗi phần tử được gán cho một số thực trong đoạn [0..1]
6 biểu thị độ thuộc của phần tử trong tập hợp,
7 với độ thuộc bằng 1 nghĩa là phần tử chắc chắn thuộc vào tập hợp
8 và ngược lại độ thuộc bằng 0 nghĩa là phần tử chắc chắn không thuộc trong tập hợp.
9
10 Phép giao và hợp của 2 tập hợp được thực hiện trên các cặp phần tử bằng nhau của 2 tập hợp,
11 với độ thuộc mới được tính bằng phép toán min và max của hai độ thuộc.
12 */
13
14 #include <iostream>
15 #include <map>
16
17 using namespace std;
18
19 template<class T>
20 map<T, double> fuzzy_set_union(const map<T, double> &a, const map<T, double> &b) {
21     /*****
22      # YOUR CODE HERE #
23      *****/
24
25     map<T, double> c; // result
26     for (auto it = a.begin(); it != a.end(); ++it) { // loop through elements in the set a
27         c[it->first] = it->second; // add element to result
28     }
29     for (auto it = b.begin(); it != b.end(); ++it) { // loop through elements in the set b
30         if (c.find(it->first) == c.end()) { // check if element is not yet in the set
31             c[it->first] = it->second; // add element to result
32         } else { // if element is already in the set
33             c[it->first] = max(c[it->first], it->second); // max equals element
34         }
35     }
36     return c; // return result
37 }
38

```

```

39 template<class T>
40 map<T, double> fuzzy_set_intersection(const map<T, double> &a, const map<T, double> &b) {
41     /*****
42     # YOUR CODE HERE #
43     *****/
44
45     map<T, double> c; // result
46     for (auto it = a.begin(); it != a.end(); ++it) { // loop through elements of in the set a
47         if (b.find(it->first) != b.end()) { // check if element is already in the set b
48             c[it->first] = min(it->second, b.at(it->first)); // min equals element
49         }
50     }
51     return c; // return result
52 }
53
54 template<class T>
55 void print_fuzzy_set(std::map<T, double> &a) {
56     cout << "{ ";
57     for (const auto &x : a) {
58         std::cout << "(" << x.first << ", " << x.second << ") ";
59     }
60     cout << "}";
61     std::cout << std::endl;
62 }
63
64 // Nguyen Van Duy - 20215334

```

	Test	Kết quả đúng
✓	<pre> map&lt;int, double&gt; a = {{1, 0.2}, {2, 0.5}, {3, 1}, {4, 0.6}, {5, 0.7}}; map&lt;int, double&gt; b = {{1, 0.5}, {2, 0.4}, {4, 0.9}, {5, 0.4}, {6, 1}}; cout &lt;&lt; "A = "; print_fuzzy_set(a); cout &lt;&lt; "B = "; print_fuzzy_set(b); map&lt;int, double&gt; c = fuzzy_set_union(a, b); map&lt;int, double&gt; d = fuzzy_set_intersection(a, b); cout &lt;&lt; "Union: "; print_fuzzy_set(c); cout &lt;&lt; "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, 0.4) (3, 1) (4, 0.6) (5, 0.4) (6, 1) } Intersection: { (1, 0.2) (2, 0.4) (4, 0.6) (5, 0.4) } </pre>
✓	<pre> map&lt;int, double&gt; a = {{-1, 0.2}, {2, 0.65}, {3, 1}, {4, 0.6}, {5, 0.75}, {1, 0.7}, {10, 0.1}}; map&lt;int, double&gt; b = {{1, 0.15}, {2, 0.14}, {4, 0.9}, {5, 0.41}, {6, 1}}; cout &lt;&lt; "A = "; print_fuzzy_set(a); cout &lt;&lt; "B = "; print_fuzzy_set(b); map&lt;int, double&gt; c = fuzzy_set_union(a, b); map&lt;int, double&gt; d = fuzzy_set_intersection(a, b); cout &lt;&lt; "Union: "; print_fuzzy_set(c); cout &lt;&lt; "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, 0.6) (5, 0.75) (10, 0.1) (1, 0.15) (2, 0.14) (4, 0.9) (5, 0.41) (6, 1) } Intersection: { (1, 0.15) (2, 0.14) (4, 0.6) (5, 0.41) } </pre>

Hoàn thành được tất cả các bộ mẫu ✓

// Nguyen Van Duy - 20215334

/\*

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



Trong đó mỗi phần tử được gán cho một số thực trong đoạn  $[0..1]$  biểu thị độ thuộc của phần tử trong tập hợp, với độ thuộc bằng 1 nghĩa là phần tử chắc chắn thuộc vào tập hợp và ngược lại độ thuộc bằng 0 nghĩa là phần tử chắc chắn không thuộc trong tập hợp.

Phép giao và hợp của 2 tập hợp được thực hiện trên các cặp phần tử bằng nhau của 2 tập hợp,

với độ thuộc mới được tính bằng phép toán min và max của hai độ thuộc.

\*/

```
#include <iostream>
```

```
#include <map>
```

```
using namespace std;
```

```
template<class T>
```

```
map<T, double> fuzzy_set_union(const map<T, double> &a, const map<T, double> &b)
{
```

```
    /***/
```

```
    # YOUR CODE HERE #
```

```
    *****/
```

```
    map<T, double> c; // result
```

```
    for (auto it = a.begin(); it != a.end(); ++it) { // loop through elements in the set a
        c[it->first] = it->second; // add element to result
    }
```

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

```
if (c.find(it->first) == c.end()) { // check if element is not yet in the set
    c[it->first] = it->second; // add element to result
} else { // if element is already in the set
    c[it->first] = max(c[it->first], it->second); // max equals element
}
}
return c; // return result
}
```

```
template<class T>
```

```
map<T, double> fuzzy_set_intersection(const map<T, double> &a, const map<T,
double> &b) {
```

```
    /*****
```

```
    # YOUR CODE HERE #
```

```
    *****/
```

```
    map<T, double> c; // result
```

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

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

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

```
        }
```

```
    }
```

```
    return c; // return result
```

```
}
```

```
template<class T>
```

```
void print_fuzzy_set(std::map<T, double> &a) {
```

```
    cout << "{ ";
```

```
for (const auto &x : a) {  
    std::cout << "(" << x.first << ", " << x.second << ") ";  
}  
cout << " }";  
std::cout << std::endl;  
}
```

```
// Nguyen Van Duy - 20215334
```

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

Cụ thể, bạn cần cài đặt hàm `vector<int> dijkstra(const vector< vector< pair<int, int> > >&adj)` nhận đầu vào là danh sách kề chứa các cặp `pair<int, int>` biểu diễn đỉnh kề và trọng số tương ứng của cạnh. Đồ thị gồm  $n$  đỉnh được đánh số từ 0 tới  $n-1$ . Hàm cần trả `vector<int>` chứa  $n$  phần tử lần lượt là khoảng cách đường đi ngắn nhất từ đỉnh 0 tới các đỉnh 0, 1, 2, ...,  $n-1$ .

```

1 // Nguyen Van Duy - 20215334
2 /*
3 Bài 4.9. Cài đặt thuật toán Dijkstra trên đồ thị vô hướng
4 được biểu diễn bằng danh sách kề sử dụng priority_queue
5 Cụ thể, bạn cần cài đặt hàm
6 vector<int> dijkstra(const vector< vector< pair<int, int> > >&adj)
7 nhận đầu vào là danh sách kề chứa các cặp pair<int, int>
8 biểu diễn đỉnh kề và trọng số tương ứng của cạnh.
9 Đồ thị gồm n đỉnh được đánh số từ 0 tới n-1.
10 Hàm cần trả vector<int> chứa n phần tử lần lượt là
11 khoảng cách đường đi ngắn nhất từ đỉnh 0 tới các đỉnh 0, 1, 2, ..., n-1.
12 */
13
14 #include <iostream>
15 #include <queue>
16 #include <climits>
17 using namespace std;
18
19 vector<int> dijkstra(const vector< vector< pair<int, int> > >&adj) {
20     /*****
21      # YOUR CODE HERE #
22      *****/
23
24     vector<int> dist(adj.size(), INT_MAX); // result
25     dist[0] = 0; // update distance
26     priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq; // priority queue
27     pq.push({0, 0}); // start with 0
28     while (!pq.empty()) { // loop
29         int u = pq.top().second; // get vertex (u)
30         pq.pop(); // remove from top of priority queue
31         for (size_t i = 0; i < adj[u].size(); i++) { // loop through adjacent vertices of current vertex (u)
32             int v = adj[u][i].first; // get vertex (v)
33             int w = adj[u][i].second; // get weight to the vertex (v)
34             if (dist[v] > dist[u] + w) { // check if old distance is greater than new distance
35                 dist[v] = dist[u] + w; // update distance
36                 pq.push({dist[v], v}); // recalculate distance from vertex (v)
37             }
38         }
39     }
40     return dist;
41 }
42
43 // Nguyen Van Duy - 20215334

```

Test	Kết quả đúng	Kết quả chương trình	
<div>✓</div> <pre> int n = 9; vector&lt; vector&lt; pair&lt;int, int&gt; &gt; &gt; adj(n); auto add_edge = [&amp;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&lt;int&gt; distance = dijkstra(adj); for (unsigned int i = 0; i &lt; distance.size(); ++i) {     cout &lt;&lt; "distance " &lt;&lt; 0 &lt;&lt; "-&gt;" &lt;&lt; i &lt;&lt; " = " &lt;&lt; distance[i] &lt;&lt; endl; } </pre>	<pre> distance 0-&gt;0 = 0 distance 0-&gt;1 = 4 distance 0-&gt;2 = 12 distance 0-&gt;3 = 19 distance 0-&gt;4 = 21 distance 0-&gt;5 = 11 distance 0-&gt;6 = 9 distance 0-&gt;7 = 8 distance 0-&gt;8 = 14 </pre>	<pre> distance 0-&gt;0 = 0 distance 0-&gt;1 = 4 distance 0-&gt;2 = 12 distance 0-&gt;3 = 19 distance 0-&gt;4 = 21 distance 0-&gt;5 = 11 distance 0-&gt;6 = 9 distance 0-&gt;7 = 8 distance 0-&gt;8 = 14 </pre>	<div>✓</div>
<div>✓</div> <pre> int n = 10; vector&lt; vector&lt; pair&lt;int, int&gt; &gt; &gt; adj(n); auto add_edge = [&amp;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&lt;int&gt; distance = dijkstra(adj); for (unsigned int i = 0; i &lt; distance.size(); ++i) {     cout &lt;&lt; "distance " &lt;&lt; 0 &lt;&lt; "-&gt;" &lt;&lt; i &lt;&lt; " = " &lt;&lt; distance[i] &lt;&lt; endl; } </pre>	<pre> distance 0-&gt;0 = 0 distance 0-&gt;1 = 2 distance 0-&gt;2 = 10 distance 0-&gt;3 = 10 distance 0-&gt;4 = 13 distance 0-&gt;5 = 6 distance 0-&gt;6 = 4 distance 0-&gt;7 = 3 distance 0-&gt;8 = 10 distance 0-&gt;9 = 8 </pre>	<pre> distance 0-&gt;0 = 0 distance 0-&gt;1 = 2 distance 0-&gt;2 = 10 distance 0-&gt;3 = 10 distance 0-&gt;4 = 13 distance 0-&gt;5 = 6 distance 0-&gt;6 = 4 distance 0-&gt;7 = 3 distance 0-&gt;8 = 10 distance 0-&gt;9 = 8 </pre>	<div>✓</div>

Hoàn thành được tất cả các bộ mẫu ✓

// Nguyen Van Duy - 20215334

/\*

Bài 4.9. Cài đặt thuật toán Dijkstra trên đồ thị vô hướng

được biểu diễn bằng danh sách kề sử dụng priority\_queue

Cụ thể, bạn cần cài đặt hàm

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

nhận đầu vào là danh sách kề chứa các cặp pair<int, int>

biểu diễn đỉnh kề và trọng số tương ứng của cạnh.

Đồ thị gồm n đỉnh được đánh số từ 0 tới n-1.

Hàm cần trả vector<int> chứa n phần tử lần lượt là

khoảng cách đường đi ngắn nhất từ đỉnh 0 tới các đỉnh 0, 1, 2, ..., n-1.

\*/

```
#include <iostream>
```

```
#include <queue>
```

```
#include <climits>
```

```
using namespace std;
```

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

```
    /*****
```

```
    # YOUR CODE HERE #
```

```
    *****/
```

```
    vector<int> dist(adj.size(), INT_MAX); // result
```

```
    dist[0] = 0; // update distance
```

```
    priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq; //  
priority queue
```

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

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

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

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

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

    int v = adj[u][i].first; // get vertex (v)
    int w = adj[u][i].second; // get weight to the vertex (v)
    if (dist[v] > dist[u] + w) { // check if old distance is greater than new distance
        dist[v] = dist[u] + w; // update distance
        pq.push({dist[v], v}); // recalculate distance from vertex (v)
    }
}

return dist;
}
```

// Nguyen Van Duy - 20215334

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

Cho  $N$  văn bản và  $Q$  truy vấn.

Với mỗi truy vấn, cần trả về văn bản khớp với truy vấn đó nhất.

Sử dụng phương pháp tính điểm TF-IDF

```
1 // Nguyen Van Duy - 20215334
2 /*
3  Bài 4.10. Xây dựng một máy tìm kiếm (search engine) đơn giản.
4  Cho  $N$  văn bản và  $Q$  truy vấn.
5  Với mỗi truy vấn, cần trả về văn bản khớp với truy vấn đó nhất.
6
7  Sử dụng phương pháp tính điểm TF-IDF
8  */
9
10 #include <iostream>
11 #include <vector>
12 #include <map>
13 #include <cmath>
14
15 using namespace std;
16
17 int n, q;
18 vector<vector<string>> text_vector; // n texts
19 vector<vector<string>> query_vector; // q queries
20 map<pair<string, int>, int> f; // frequency of occurrence of the word @t in the text[i] (pair<string, int>)
21 vector<int> max_f; // maximum frequency of occurrence in the text[i]
22 map<string, int> df; // number of text containing the word t
23
24 // split string by ','
25 inline vector<string> split_string(const string &s) {
26     vector<string> res; // result
27     string t;
28     for (char i: s) {
29         if (i != ',') {
30             t.push_back(i); // add to t or t += i
31         } else {
32             res.push_back(t); // add t to vector res
33             t.clear(); // t = ""
34         }
35     }
36     res.push_back(t); //
37     return res; // return result
38 }
39
```



```

40  ✓ inline void input() {
41      cin >> n; // input n
42  ✓  for (int i = 0; i < n; ++i) {
43      |     string temp; // allocate
44      |     cin >> temp; // get string
45      |     text_vector.push_back(split_string(s: temp)); // add split string
46      | }
47      cin >> q; // input q
48  ✓  for (int i = 0; i < q; ++i) {
49      |     string temp; // allocate
50      |     cin >> temp; // get string
51      |     query_vector.push_back(split_string(s: temp)); // add split string
52      | }
53  }
54
55  ✓ inline void calc_f() {
56  ✓  for (int i = 0; i < n; ++i) {
57      |     int maximum = 0; // max f
58  ✓  |     for (auto &t : string& : text_vector[i]) {
59      |         pair<string, int> key = make_pair(&t, &i); // key : word @t in text[i]
60  ✓  |         if (f.find(x: key) != f.end()) { // check if key exists
61      |             ++f[key]; // increase f
62      |         } else {
63      |             f[key] = 1; // set f
64      |         }
65      |         maximum = max(maximum, f[key]); // check and update maximum
66      |     }
67      |     max_f.push_back(maximum); // push_back max f to vector
68  }
69  }
70
71  ✓ inline int calc_df(const string& t) {
72  ✓  |     if (df.find(x: t) != df.end()) { // check if t exists
73      |         return df[t]; // return calculated result
74      |     }
75
76      |     int res = 0; // result
77  ✓  |     for (int i = 0; i < n; ++i) {
78  ✓  |         for (string &s : text_vector[i]) {
79  ✓  |             |     if (t == s) {
80      |             |         ++res; // increase 1
81      |             |         break;
82      |             |     }
83      |         }
84      |     }
85      |     df[t] = res; // update result
86      |     return res; // return result
87  }
88

```

```

89  ✓ inline double score(const string& t, int index_text) {
90      if (f[{ x: t, &: index_text}] == 0) return 0; // score = 0
91
92      // score = tf * idf
93      // trong do :
94      //   tf = 0.5 + 0.5 * f / max_f
95      //   idf = log2 (N / df)
96      return (0.5 + 0.5 * f[{ x: t, &: index_text}] / max_f[index_text]) * log2( x: n / calc_df(t));
97  }
98
99  ✓ inline int search_engine(vector<string> &queries) {
100      int res = -1; // result
101      double max_score = -1; // maximum score
102      ✓ for (int i = 0; i < n; ++i) {
103          double score_cur = 0; // current score
104          ✓ for (auto &t : string& : queries) {
105              score_cur += score(t, index_text: i); // calculate and increase
106          }
107
108          ✓ if (score_cur > max_score) { // check if max score
109              max_score = score_cur; // update max score
110              res = i; // update index (result)
111          }
112      }
113      return res + 1; // return
114  }
115
116  ► ✓ int main() {
117      input();
118      calc_f(); // pre-calc
119      ✓ for (auto &query : vector<string> & : query_vector) {
120          cout << search_engine( &: query) << endl;
121      }
122
123      return 0;
124  }
125
126      // Nguyen Van Duy - 20215334
127

```

C:\Users\Admin\CLionProjects\test\cmake-build-debug\test.exe

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

Process finished with exit code 0

|

main.cpp
 input.txt

```

1 100
2 tnckle,bhwco,yvvburc,cfu,rumbmem,iohord,gpnkaz,jfcky,frkml,oqntc,tax,blns,gfrteo
3 zamuc,iulea,men,ewd,exfbv,dntd,atzz,oay,sgsp,kejhz,wctd,lbstno,igzn,qtu,bzt,vbs
4 hvokiia,rageltk,qdvadiw,dgc,sgsp,sgd,gvlc,tatbz,iohord,xnjc,gvw,ljg,ehioxhh,snsn
5 jrxhrdy,wphapj,ahkgccn,fdimn,ikss,jigskmv,hepdif,deu,ukw,inhwilk,zdrc,mjcxo,yggx
6 evslqp,dsgcfoh,deyqcrq,ahkgccn,qdvadiw,jbq,ozjqp,frpjb,gyetyye,vnxu,ehhsv,bcvkz,
7 qlsz,baecplp,mmsbaop,nllu,hoq,wphapj,wdt,zowkkyh,scczybn,lvcnvb,imq,qrpvy,snfwz
8 rqopub,okbfug,qan,hoq,bhcda,nsp,fhic,zgs,gxrx,ujquo,vnxu,pbegvsu,ppleq,fdpax,ine
9 jizy,pzz,vtnu,hdmxk,tuse,frtzlqs,fjma,mkpvgr, lkzivde,kdmeo,hdmxk,mro,kqzx,eettw
10 lzbknh,wjuga,rxxv,clt,adgkhuf,vtr,oxle,tut,nzlg,nqx,rjmow,tax,dbxjmi,okcyea,bhcd
11 ozry,kbljo,tyjri,kbljo,dmfhaop,ikss,nes,moi,pycf,clt,jrxhrdy,bqr,wkvu,qinuqni,xi
12 ameybn,tfiu,pnim,wphapj,zgs,mzhlvih,bapriu,jrxhrdy,wlrbbm,sysh,timi,riicfs,tatbz
13 kqz,etc,kwabhsl,sgd,uzhyv,iohord,vtq,ntndx,nqx,rdsy
14 pycf,lyskrhu,pqzeqv,wmjobm,ktx,rref,fkyhs,hkicq,baecplp,rtdgyr,oqmsbo,fjma,ddqs
15 yoes,lunm,mixf
16 pyyeq,hepdif,nov,yve,ehhsv,yggxpw,wjuga,nltc,mbaqw,julkfu,wdtxjbm,fdupl,gtqvvhb,
17 elhh,ddqnqdo,qmlfotp,ghlc,euyga,wctgt,vnxu,cfu,jcpwsdq,frqqw,bzt,asi,ifz,deyqcrq
18 dipfzws,bhwco,gyetyye,eettwdn,gsieeh,mws,gkmlh,opkmco,xjecw
19

```

Run

test

C:\Users\Admin\CLionProjects\test\cmake-build-debug\test.exe

↑
 ↓
 ↺
 ↻
 🖨
 🗑

71  
50  
64  
42  
47  
98  
43  
71  
69  
94  
90  
84  
23  
85  
84  
52  
57

main.cpp input.txt x

```

1 200
2 bie,ffke,sxtl,ghf,xnbqs,ikdzpbe,ssnkxuh,wpbn,pcxxbcm,cavggy,iusqi
3 sxwkyu
4 vspjmuu,zqja,xwbwogf,beb,jlnl,fmp,jol,hofffhl,rdmqfn,sjkdoxc,qgschf,fdgl,jahtswe,cavggy,kml
5 krgube,nwq,rinukm,wbxlvtl,txie,gsxx,jkmiqnd,rbt,ltwhb,txkq,kwr,dhkijf,azkwqgo,qduba,ffyl,
6 dnzdmc,srzup,eavo,uvpvq,tkpityy,wdnpape,znxuot,ndk,ccmkl,yzt,sft,dggfn,zkr,ftv,czoycem,gatv
7 ymgsmc,ruqd,ngaw,siyxaeu,etjnhx,psm,zogf,adny,vpne,onnmk,eckitk,yzqcin,xnsovo,lsti,igjrm,al
8 vgqx,ctccby,qgschf,hdamnnw,vbgho,lifmu,ffke,ovkhi,npayxim,wraxexq,hmtynkq,msrfzl,riahl,rzj:
9 xcvhjkr,gsdixze,djizayy,tcah,rrvu,uouvn,enwkif,slmf,sxtl,gsxx,ionstau,eckitk,oyuba,bnff,i:
10 ykm,fypepde,fwixu,hzawn,vzf,jelj,myug,ukus,wsr,nmvltf,ytpupxk,xdjc,ymgsmc,msrfzl,ptcwjql,q:
11 ehsyy,wwcu,koj,qgxzl,vfyg,tbiidyg,msj,syhabx,yxkg,grmdvw,swdbexy,umczbnj,fwixu,jxhvn,demsi,
12 refq,gybi,sjxuq,izrkhpc,vdcnabz,udzye,jol,bgzs,rmgeex,kyswidv,rveck,kdq,ipzev,bud,ikfear,bu
13 zxawvbj,lpwf,grethp,knuyzq,ucucb,hxqbyq,yxkg,gvnbtul,pjluhdi,mrtbok,pyph,fncfukw
14 ioba,qhseu,zhsls,jfidea,yccr,smigtwj,qckvnzv,vzf,zmzb,vtsiui,xfyr,amibtik,xpvqzu,oychif,vb:
15 hff,qugap,fofvyi,zwqvfa,ohifqi,nyxwci,hqnjili,dosub,svslurs,uth,twjin,zetk,mpz,dmlrbuc,b:
16 oyichif,viy,dxxjy,smowdy,jcxaw,mwfynii,tvd,uencf,bzigf,clalqcg,izn,dcp,etiu,edac,gjbbcu,jol,
17 eiaqyu,eioj,zqal,gjtsflt,dnuo,uxi,wwmxbn,rqmhbok,ize,vfyg,jwog,iaxesvm,scx,rwckh,okqgxk,:
18 ncg,dknkkw,pphqxxg,kli,jzrl,frjmsf,zdszbg,foj,brg,jhuoc,oqv,sqiz,epgoci,fqwupo,rphqbg,yccr,
19

```

Run test x

C:\Users\Admin\CLionProjects\test\cmake-build-debug\test.exe

```

5
6
200
46
5
6
67
24
78
104
105
18
25
13
123
180
138

```

est > input.txt

main.cpp
 

input.txt
 ×

```

1 300
2 dizc,dnzohyz,bbekdtr,aiw,nqrz,kif,zuuz,dhyhv,ykyxv,wulzlds,vzkxe,aoaxyt,gnfvcb,
3 fbsdjcn,ubvypc,nmxv,lvo,nbxlcng,qbpewzj,lxcd,ljqskul,hokpxsj,pnkiu,btsek,jekj
4 bkdvyof,pjjxr,pbawphc,wzolz,ghcajps,zicxpdu,tlykqga,yncqv,hweinl,okhoix,lyk,py
5 ghtrtr,cksmh,hzngn,wlgypn,khpbnl,yvwwhp,xzstn,xfbuoi,bjsqi,sdwg,vwtwtr,otbgad,
6 cbjhc,qawonqq,abgnng,hbqj,dgsi,wqrlho,yzsoqu,ywus,vzkxe,hapoc,retyla,bxnwsh,zk
7 upjwrf,wnhheom,ncaze,fenvk,eqbcx,qgogwgd,howhdos,aen,kcrao,kcfacm,zgwhulx,kmft
8 wqrbkx,lyuiz,hmalkh,gdsydz,pyrg,qqrietr,olpxamo,pctc,gwqtzz,oclkhrw,qgu,xow,c
9 hwhbqq,redfgc,iizt,byq,khpbnl,cximpvz,kplawyb,ilg,ixhdkd,bhx,katu,kjxryaj,hep
10 vwgp,cfmkb,ooifns,dmvro,bgmd,jxvwjks,evlpw,cutyua,hgfax,ffhmemf,ymy,sgzjigw,zl
11 iqpj,eah,ooa,ooemwk,llvu,xbmq,jzhwdbg,hinm,jeiu,uze,lzjfne,ysh,xotdgpv,fuj,mmv
12 nngvo,ruwwj,dpihfy,soo,lvzrq,nkxowa,fyt,webv,vfjx,grohg,eyccta,zez,zfcazzu,r
13 ilg,dhv,puzm,ynpn,zwzt,nls,ybyo,fijcbhr,wgm,izoql,btmflb,kbemny,dogomdf,exsyt
14 vvtbjq,tgd,xwzrp,kcblgsc,hub,wigsm,bkdvyof,ttnn,lqpxexr,pnp,yfa,nnyan,rzxwu,ol
15 ndt,qlexh,wjuzhjh,mff,gmvxnj,byym,loet,ornh,kmmrhuh,ymok,oyqms,zcszp,wtwr,bdm
16 udf,xow,oanl,kmdcbf,llbpw,okypj,bktd,jaqpzun,lxjt,lihj,jvaxldz,baalqgo,pevb,kh
17 uujzgm,jwaumbn,mbhy,ojm,colypfg,yurxn,ccgtdw,gqs,ipq,bnpekm,ucj,yehnbrh,udxfip,
18 slu,ucxlx,dtdoj,ynjmfq,sopzreo,rlua,vrlx,cijyg,hdzg,tzjbgn,adtnsi,lrcn,ownsdpc
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
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

```

Run

test ×

C:\Users\Admin\CLionProjects\test\cmake-build-debug\test.exe

↑

↓

≡

↺

↻

🖨

🗑

21  
277  
132  
82  
84  
278  
141  
222  
268  
275  
88  
142  
287  
177  
211  
105  
121

test > 

input.txt

IT3040 – 2022.2 – Mã lớp TH: 727638

main.cpp ×
 input.txt ×

```

1 400
2 ddt,yed,shch,hpfs,yumfmij,rflud,mzu,oyjta,mmjhjr,yeqyccp,qakd,axx,ers,nnntnd,:
3 njaeep,inp,ialpfi,wcm,rwgox,jwgyy,iocgpb,bautmb,aagl,ttuwgso,rav,yadrq,bw
4 kes,kztjscw,gnwmb,faphcg,hvver,wagyvb,cqa,xyey,rbfx,nbdi,xhc,bbxsp,gyiaew,d
5 ozay,rsskwz,bucohv,zjisgx,uhm,sifg,tsvgd,rjeo,nqbd,palg,fab,guj,agmrrwj,qrwk
6 qcpsq,kgnukv,eezavwj,ywyutoy,mqmq,emwct,grrpiim,araxyl,lxmtuzn,hknlroy,ltgcl
7 bucdua,pmkzhe,cze,gzsevj,frnt,pmcy,humovco,fuc,sgsp,mdrln,xbfcci,mcdjo,jhxvn
8 wslc,ppply,tjyhf,yicoew,wcrjhei,nts,pwusd,fpwqugj,drd,yrncwe,dkzm,qcjjv,aqrr
9 doc,doc,xguev,trxgiag,ykfgn,pyjuah,pjdxupd,mdeydf,cuwu,tzs,ofv,aqz,vxw,jgheh
10 whgpi,ngkvsb,jjhalw,wbkilq,faf,owq,bvj,cwser,wrq,lbyz,kxods,echdx,rii,ihkhg
11 rdqiz,ijre,lqjrlne,lzopga,jxgbdf,vyr,zadz,fcqez,sjakqm,rwgox,urdk,itot,tuxoc:
12 mcomusn,kwgcwjj,wayi,mbuk,rrw,bygjjo,wgcfr,y,rrkk,hbfng,pkmvmm,xzcb,vxro,sjrh
13 wmlko,lvnded,ecrqcsg,afgnk,bqowcs,qkmusq,cym,jnxmpg,gzsevj,lex,yxd,zgnf,nxl,i
14 dlbr,siy,vfiieh,pwcz,zpg,takrkyv,vubpg,tpvneun,jzmo,ikntvwu,wwz,mqrlef,hbci
15 dirdl,wkkq,lvnded,pxwwq,wnr,wjx,kbzetbb,jilqdx,bdzpgkd,wuw,jpuj,oorvdcz,nav:
16 qasdkx,hpb,grrpiim,fnigi,udfnly,ntfhyr,uvaog,iaj,lcibh,vdzng,ttwfii,vgidbc,g
17 cboo,soiafic,zadz,tmt,fruauol,zfus,htknmbc,vteqst,ekw,qaks,mzks,nzfam,hbozqjl
18 fnl,msy,kmlspsc,tzgeb,opxrh,bclcf,mxeaafh,jhjf,ritip,tawdeac,afklf,hdxlggh,u
19
20

```

Run

test ×

C:\Users\Admin\CLionProjects\test\cmake-build-debug\test.exe

380  
22  
271  
260  
106  
295  
226  
193  
272  
223  
183  
69  
248  
237  
126  
34  
43

test > input.txt

IT3040 – 2022.2 – Mã lớp TH: 727638

```

1 500
2 zvesjqw, smyy, vjf, mhrpzo, odlzzpu, hhbrwa, qzqixw, hfa, lnbvzzz, nxevekz, vrv, vzkm,
3 adaxfa, zftkjh, bzn, ngk, vfun, wxxftwq, wejuaw, srywkbe, zzzimwe, dtjifyx, odcotec, k
4 uul, bfoak, mjjs, azglbjh, rbyfiz, roc, imom, bfwqeu, rjuirr, kyadvp, gzja, ywvbqd, oet
5 smcr, xab, dcpscf, prakx, wplxod, egkplh, jaukjb, kysg, jlnjn, netf, txxt, wyinei, qvcp
6 hftxm, jrqbwrw, uczyg, vxiog, zqy, mridok, tiizru, edj, qpfrzz, wwtb, hcuutwh, zxoo, da
7 fscmj, nkfkgec, kvxk, octn, ripssr, ihnukpu, pvwkk, goyiald, wajyr, cubshn, ueidlw, bi
8 abx, ycqr, apbd, mwicccff, nfb, uxw, rgjzn, zjs, svcg, wvonz, dbt, yklolb, mscv, tfqflb, h
9 imjb, hxl, vqtpgah, sovu, nzjbie, nlyfy, febxuc, jajd, zgqz, wnasihw, bdfiox, alw, grv,
10 zbn, snb, xug, jwlt, brcyq, kvu, ogl, hmuhrah, tgaba, mxu, miap, epwa, lqt, behgpo, mejak
11 hkucw, kivrqvn, betn, uhyk, tfrxnah, xflrp, sghl, hvazjn, lec, ankol, byfwey, ihp, xho,
12 xyv, erqzsl, uars, bfulcnq, cdlopn, upohl, jamxl, vujf, stu, xedu, ztqe, vglgaa, jytesx
13 cyt, xgkn, mjga, sgau, gdtclvv, lmr, trfs, fsmcrp, cxabcx, qknkzsi, jgihh, ujc, lyj, bne
14 uatxcs, vbrfstt, xel, bebb, muv, echww, kqjh, gocvk, vhqqqv, fyskuc, rsx, tjlnw, icantw
15 esun, eotgs, gqq, fimoo, avb, qknqnnz, nprzz, fqpy, pslme, chj, jjjdy, bwbai, usy, rpfi
16 kul, xrlf, vogvir, web, yzm, nlc, twoym, fpeac, wbz, gklvk, jppzaas, zxiejgl, uzwf, nnp
17 yotx, uhssf, dgdcg, wvkkwet, osamtfs, opleau, ganiicm, atjjw, cue, njkc, mwicccff, rqod
18 weh, dhwuid, fvkrrj, gluyj, owusx, qxwlmr, cbafolb, pxyg, wuoce, tdckqsu, xcv, imtc, rdb
19
20

```

Run test

C:\Users\Admin\CLionProjects\test\cmake-build-debug\test.exe

```

348
122
333
220
307
104
54
29
452
7
336
130
124
100
123
353
397

```

test > input.txt

// Nguyen Van Duy - 20215334

/\*

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

Cho N văn bản và Q truy vấn.

Với mỗi truy vấn, cần trả về văn bản khớp với truy vấn đó nhất.



Sử dụng phương pháp tính điểm TF-IDF

\*/

```
#include <iostream>
```

```
#include <vector>
```

```
#include <map>
```

```
#include <cmath>
```

```
using namespace std;
```

```
int n, q;
```

```
vector<vector<string>> text_vector; // n texts
```

```
vector<vector<string>> query_vector; // q queries
```

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

```
vector<int> max_f; // maximum frequency of occurrence in the text[i]
```

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

```
// split string by ','
```

```
inline vector<string> split_string(const string &s) {
```

```
    vector<string> res; // result
```

```
    string t;
```

```
    for (char i: s) {
```

```
        if (i != ',') {
```

```
            t.push_back(i); // add to t or t += i
```

```
        } else {
```

```
res.push_back(t); // add t to vector res

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

}

}

res.push_back(t); //

return res; // return result

}

inline void input() {
    cin >> n; // input n
    for (int i = 0; i < n; ++i) {
        string temp; // allocate
        cin >> temp; // get string
        text_vector.push_back(split_string(temp)); // add split string
    }
    cin >> q; // input q
    for (int i = 0; i < q; ++i) {
        string temp; // allocate
        cin >> temp; // get string
        query_vector.push_back(split_string(temp)); // add split string
    }
}

inline void calc_f() {
    for (int i = 0; i < n; ++i) {
        int maximum = 0; // max f
        for (auto &t : text_vector[i]) {
```

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

if (f.find(key) != f.end()) { // check if key exists
    ++f[key]; // increase f
} else {
    f[key] = 1; // set f
}

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

max_f.push_back(maximum); // push_back max f to vector
}
}
```

```
inline int calc_df(const string& t) {
    if (df.find(t) != df.end()) { // check if t exists
        return df[t]; // return calculated result
    }
}
```

```
int res = 0; // result
for (int i = 0; i < n; ++i) {
    for (string &s : text_vector[i]) {
        if (t == s) {
            ++res; // increase 1
            break;
        }
    }
}

df[t] = res; // update result
```

```
return res; // return result
}

inline double score(const string& t, int index_text) {
    if (f[{t, index_text}] == 0) return 0; // score = 0

    // score = tf * idf
    // trong đó :
    // tf = 0.5 + 0.5 * f / max_f
    // idf = log2 (N / df)
    return (0.5 + 0.5 * f[{t, index_text}] / max_f[index_text]) * log2(n / calc_df(t));
}

inline int search_engine(vector<string> &queries) {
    int res = -1; // result
    double max_score = -1; // maximum score
    for (int i = 0; i < n; ++i) {
        double score_cur = 0; // current score
        for (auto &t : queries) {
            score_cur += score(t, i); // calculate and increase
        }

        if (score_cur > max_score) { // check if max score
            max_score = score_cur; // update max score
            res = i; // update index (result)
        }
    }
}
```

```
    return res + 1; // return
}

int main() {
    input();
    calc_f(); // pre-calc
    for (auto &query : query_vector) {
        cout << search_engine(query) << endl;
    }

    return 0;
}

// Nguyen Van Duy - 20215334
```

**Bài tập 4.11.** Bức tường bao quanh một lâu đài nọ được cấu thành từ  $n$  đoạn tường được đánh số từ 1 đến  $n$ . Quân giặc lên kế hoạch tấn công lâu đài bằng cách gửi  $a_i$  tên giặc đánh vào đoạn tường thứ  $i$ . Để bảo vệ lâu đài có tất cả  $s$  lính.

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

```
1 // Nguyen Van Duy - 20215334
2 /*
3  Bài 4.11. Bức tường bao quanh một lâu đài nọ được cấu thành
4  từ  $n$  đoạn tường được đánh số từ 1 đến  $n$ .
5  Quân giặc lên kế hoạch tấn công lâu đài bằng cách gửi  $a_i$ 
6  tên giặc đánh vào đoạn tường thứ  $i$ .
7  Để bảo vệ lâu đài có tất cả  $s$  lính.
8
9  Yêu cầu hãy viết chương trình phân bố lính đứng ở các đoạn
10  tường sao cho tổng số lính là  $s$  và tổng số lượng tên giặc
11  lọt vào lâu đài là nhỏ nhất.
12 */
13
14 #include <iostream>
15 #include <vector>
16 #include <algorithm>
17
18 using namespace std;
19
20 class Wall {
21 private:
22     int a; // there are @a members in Army
23     int k; // each soldier can repel the attack of @k enemies
24
25 public:
26     // constructor with no arguments
27     Wall() : a(0), k(0) {}
28
29     // constructor with arguments : @a, @k
30     Wall(int a, int k) : a(a), k(k) {}
31
32     [[nodiscard]] int passed() const { return a; }
33
34     [[nodiscard]] int getK() const { return k; }
35
36     void send_soldier(int numbers = 1) {
37         a = max(0, a - numbers * k); // recalculate number of enemies entering
38         k = min(k, a); // update @k
39     }
40 }
```

```

41     friend bool operator<(const Wall &w_1, const Wall &w_2) { return w_1.k < w_2.k; }
42
43     friend bool operator>(const Wall &w_1, const Wall &w_2) { return w_1.k > w_2.k; }
44
45     friend bool operator<=(const Wall &w_1, const Wall &w_2) { return w_1.k <= w_2.k; }
46
47     friend bool operator>=(const Wall &w_1, const Wall &w_2) { return w_1.k >= w_2.k; }
48
49
50     friend istream &operator>>(istream &is, Wall &w) {
51         is >> w.a >> w.k; // input @a, @k
52         w.k = min(w.k, w.a); // update @k
53         return is; // return stream
54     }
55
56     friend ostream &operator<<(ostream &os, const Wall w) {
57         os << "{a = " << w.a << "; k = " << w.k << "}"; // output
58         return os; // return stream
59     };
60
61 // sort_walls : descending order
62 inline void sort_walls(vector<Wall> &v) { sort( first: v.begin(), last: v.end(), comp: greater<>()); }
63
64 inline int number_of_enemies_entering(vector<Wall> &v) {
65     int count = 0; // counter
66     for (auto &w : Wall& : v) { // loop over each wall
67         count += w.passed(); // increment counter
68     }
69     return count; // return result
70 }
71
72 ► int main() {
73     int n, s; // number of wall, and number of soldier
74     cin >> n >> s; // enter @n, @s
75
76     vector<Wall> walls(n); // wall list
77     for (int i = 0; i < n; i++) {
78         cin >> walls[i]; // enter @a, @k of the wall
79     }
80
81     while (s--) { // send soldiers one by one to the wall
82         sort_walls( & walls); // descending order : sort from largest to smallest
83         walls[0].send_soldier(); // send a soldier to
84     }
85
86     cout << number_of_enemies_entering( & walls) << endl; // output
87
88     return 0;
89 }
90
91 // Nguyen Van Duy - 20215334

```

```
C:\Users\Admin\CLionProjects\test\cmake-build-debug\test.exe
```

```
3 3  
4 2  
1 1  
10 8  
3
```

```
Process finished with exit code 0
```

```
C:\Users\Admin\CLionProjects\test\cmake-build-debug\test.exe
```

```
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
```

```
Process finished with exit code 0
```



main.cpp input.txt x

```

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

```

Run test x

C:\Users\Admin\CLionProjects\test\cmake-build-debug\test.exe  
506

Process finished with exit code 0

main.cpp input.txt x

```

1 100 322
2 84 9
3 21 48
4 35 47
5 69 33
6 55 2
7 84 74
8 64 86
9 9 97
10 49 30
11 53 90
12 43 83
13 22 57
14 81 89
15 66 66

```

Run test x

C:\Users\Admin\CLionProjects\test\cmake-build-debug\test.exe  
0

Process finished with exit code 0

```

main.cpp  input.txt x
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

Run test x
C:\Users\Admin\CLionProjects\test\cmake-build-debug\test.exe
1476
Process finished with exit code 0

main.cpp  input.txt x
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

Run test x
C:\Users\Admin\CLionProjects\test\cmake-build-debug\test.exe
0
Process finished with exit code 0

```

// Nguyen Van Duy - 20215334

/\*

Bài 4.11. Bức tường bao quanh một lâu đài nọ được cấu thành từ  $n$  đoạn tường được đánh số từ 1 đến  $n$ .

Quân giặc lên kế hoạch tấn công lâu đài bằng cách gửi ai tên giặc đánh vào đoạn tường thứ  $i$ .

Để bảo vệ lâu đài có tất cả  $s$  lính.

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

\*/

```
#include <iostream>
```

```
#include <vector>
```

```
#include <algorithm>
```

```
using namespace std;
```

```
class Wall {
```

```
private:
```

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

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

public:

```
// constructor with no arguments
```

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

```
// constructor with arguments : @a, @k
```

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

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

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

```
void send_soldier(int numbers = 1) {
```

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

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

```
}
```

```
friend bool operator<(const Wall &w_1, const Wall &w_2) { return w_1.k < w_2.k; }
```

```
friend bool operator>(const Wall &w_1, const Wall &w_2) { return w_1.k > w_2.k; }
```

```
friend bool operator<=(const Wall &w_1, const Wall &w_2) { return w_1.k <= w_2.k; }
```

```
friend bool operator>=(const Wall &w_1, const Wall &w_2) { return w_1.k >= w_2.k; }
```

```
friend istream &operator>>(istream &is, Wall &w) {  
    is >> w.a >> w.k; // input @a, @k  
    return is; // return stream  
}  
  
friend ostream &operator<<(ostream &os, const Wall w) {  
    os << "{a = " << w.a << "; k = " << w.k << "}"; // output  
    return os; // return stream  
}  
};  
  
// sort_walls : descending order  
inline void sort_walls(vector<Wall> &v) { sort(v.begin(), v.end(), greater<>()); }  
  
inline int number_of_enemies_entering(vector<Wall> &v) {  
    int count = 0; // counter  
    for (auto &w: v) { // loop over each wall  
        count += w.passed(); // increment counter  
    }  
    return count; // return result  
}
```

```
int main() {

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

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


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

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

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

    }


    while (s--> 0) { // send soldiers one by one to the wall

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

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

    }


    cout << number_of_enemies_entering(walls) << endl; // output


    return 0;

}
```

// Nguyen Van Duy – 20215334

**Bài tập 4.12.** Cho một lược đồ gồm  $n$  cột chữ nhật liên tiếp nhau có chiều rộng bằng 1 và chiều cao lần lượt là các số nguyên không âm  $h_1, h_2, \dots, h_n$ .

Hãy xác định hình chữ nhật có diện tích lớn nhất có thể tạo thành từ các cột liên tiếp.

```
1 // Nguyen Van Duy - 20215334
2 /*
3  Bài 4.12. Cho một lược đồ gồm  $n$  cột chữ nhật liên tiếp nhau
4  có chiều rộng bằng 1 và chiều cao lần lượt là các số nguyên
5  không âm  $h_1, h_2, \dots, h_n$ .
6  Hãy xác định hình chữ nhật có diện tích lớn nhất có thể tạo thành từ các cột liên tiếp.
7  */
8
9 #include <iostream>
10 #include <vector>
11 #include <queue>
12
13 using namespace std;
14
15 struct rectangle {
16     int height;
17     int width;
18
19     rectangle() : height(0), width(0) {}
20
21     rectangle(int height, int width) : height(height), width(width) {}
22
23     rectangle increment(bool increment_height = false, bool increment_width = true) {
24         if (increment_height) { // check if increment height or not
25             height++; // increment height
26         }
27         if (increment_width) { // check if increment width or not
28             width++; // increment width
29         }
30         return *this; // return rectangle
31     }
32
33     rectangle decrement(bool decrement_height = false, bool decrement_width = true) {
34         if (decrement_height) { // check if decrement height or not
35             height--; // decrement height
36         }
37         if (decrement_width) { // check if decrement width or not
38             width--; // decrement width
39         }
40         return *this; // return rectangle
41     }
42 }
```

```

43 friend rectangle operator++(rectangle &rect) { return rect.increment(); }
44
45 friend const rectangle operator++(rectangle &rect, int) {
46     rectangle temp = rect; // copy old rectangle
47     rect.increment(); // increment rectangle
48     return temp; // return old rectangle
49 }
50
51 friend rectangle operator--(rectangle &rect) { return rect.decrement(); }
52
53 friend rectangle operator--(rectangle &rect, int) {
54     rectangle temp = rect; // copy old rectangle
55     rect.decrement(); // decrement rectangle
56     return temp; // return old rectangle
57 }
58
59 friend rectangle operator+(rectangle &rect, int v) {
60     return {rect.height, width: rect.width + v};
61 }
62
63 friend void operator+=(rectangle &rect, int v) {
64     rect.width += v;
65 }
66
67 friend ostream &operator<<(ostream &os, rectangle rect) {
68     // output : (height x width)
69     os << "(" << rect.height << " x " << rect.width << ")";
70     return os; // return stream
71 }
72
73 friend istream &operator>>(istream &is, rectangle &rect) {
74     is >> rect.height; // input : height
75     rect.width = 0; // set width = 0
76     return is; // return stream
77 }
78
79 [[nodiscard]] int superficie() const {
80     return height * width; // area(superficie) = height * width
81 }
82 };
83

```



```

84 ► int main() {
85     int n; // number of columns
86     int res = INT_MIN; // result : largest area(superficie), set to INT_MIN
87     cin >> n; // input n
88
89     vector<rectangle> v(n); // rectangles, whose height corresponds to height of columns
90     queue<rectangle> q, old_queue, new_queue;
91     // q : contains original rectangles
92     // old_queue : (current_queue) contains current rectangles
93     // new_queue : (next_queue) contains next rectangles
94
95     for (int i = 0; i < n; i++) {
96         cin >> v[i]; // input rectangles
97         q.push(v[i]); // push into q (original rectangles)
98     }
99
100    while (!q.empty()) {
101        rectangle rect = q.front(); // get 1 of original rectangles
102        q.pop(); // remove from q (original rectangles)
103
104        while (!old_queue.empty()) { // loop
105            rectangle old_rect = old_queue.front(); // get from old queue
106            old_queue.pop(); // remove an element from old queue
107
108            if (old_rect.height < rect.height) { // check if
109                new_queue.push(++old_rect); // rectangle extension and push into next queue
110            } else {
111                rect.width = max(rect.width, old_rect.width); // update for original rectangle
112                res = max(res, old_rect.superficie()); // update largest rectangle
113            }
114        }
115
116        old_queue.swap(new_queue); // switch to next queue
117        old_queue.push(++rect); // push original rectangle into current queue
118    }
119
120    while (!old_queue.empty()) { // loop remaining rectangle
121        rectangle old_rect = old_queue.front(); // get rectangle
122        old_queue.pop(); // remove from queue
123        res = max(res, old_rect.superficie()); // update remaining rectangle
124    }
125
126    cout << res; // print result
127    return 0;
128 }
129
130 // Nguyen Van Duy - 20215334

```

C:\Users\Admin\CLionProjects\test\cmake-build-debug\test.exe

7

6 2 5 4 5 1 6

12

Process finished with exit code 0

C:\Users\Admin\CLionProjects\test\cmake-build-debug\test.exe

10000

3368 3433 2649 2068 2548 3618 779 231 1185 4459 331 1938 4244 724 287 1048 3033 1205 4  
45954

```
Process finished with exit code 0
```

C:\Users\Admin\CLionProjects\test\cmake-build-debug\test.exe

20000

2440 2581 350 308 1893 1361 250 3202 2531 3089 4937 3205 3609 890 3217 1885 3094 4409 1  
41684

```
Process finished with exit code 0
```



```

main.cpp  input.txt x
1 500000
2 14722 957 2717 1081 159 1932 5428 129 4544 17120 5880 11894 19770 8069
  17810 9245 14121 4936 6965 6084 7011 655 11744 9823 9721 8409 9988 187
  18680 2313 8381 7288 16658 6703 4104 11199 12066 13711 12636 19374 527
  3955 2468 15776 3473 7583 12607 8820 4997 9169 10082 12679 15648 8843
  2489 19459 697 17449 10194 18908 11515 2266 243 9690 321 18882 13327 1
  11024 15084 5936 1797 468 7564 8966 8102 12656 5856 3796 1499 11338 10
  8074 10734 8688 10093 1081 18011 1872 2033 8827 6088 6581 7367 7508 4
  6850 9081 945 11556 13579 6627 6146 16094 13012 13859 16790 1612 19819
  11230 13114 8207 290 1416 10087 4994 12510 15282 2990 19911 6145 5265
  2490 3539 14670 7034 9261 9529 9978 3901 5809 8848 12143 9134 16493 4
  13311 6821 6640 13417 17855 8526 6521 9597 9396 14392 6494 10024 11306
  5363 15821 3858 9627 15172 17803 15652 7259 4512 6921 7957 7081 19572
  6194 314 5274 579 1910 11907 15934 18363 7465 12701 18296 6578 15534 1
  414 18480 17742 11814 6904 18077 9902 143 3873 15997 3902 8750 277 688
  17143 14397 3261 13908 14809 3515 2068 1026 14649 8830 15746 4271 8781

Run test x
C:\Users\Admin\CLionProjects\test\cmake-build-debug\test.exe
48932400
Process finished with exit code 0

```

// Nguyen Van Duy - 20215334

/\*

Bài 4.12. Cho một lược đồ gồm  $n$  cột chữ nhật liên tiếp nhau

có chiều rộng bằng 1 và chiều cao lần lượt là các số nguyên

không âm  $h_1, h_2, \dots, h_n$ .

Hãy xác định hình chữ nhật có diện tích lớn nhất có thể tạo thành từ các cột liên tiếp.

\*/

#include <iostream>

#include <vector>

#include <queue>

```
using namespace std;
```

```
struct rectangle {
```

```
    int height;
```

```
    int width;
```

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

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

```
    rectangle increment(bool increment_height = false, bool increment_width = true) {
```

```
        if (increment_height) { // check if increment height or not
```

```
            height++; // increment height
```

```
        }
```

```
        if (increment_width) { // check if increment width or not
```

```
            width++; // increment width
```

```
        }
```

```
        return *this; // return rectangle
```

```
    }
```

```
    rectangle decrement(bool decrement_height = false, bool decrement_width = true) {
```

```
        if (decrement_height) { // check if decrement height or not
```

```
            height--; // decrement height
```

```
}

if (decrement_width) { // check if decrement width or not

    width--; // decrement width

}

return *this; // return rectangle

}

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

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

    rectangle temp = rect; // copy old rectangle

    rect.increment(); // increment rectangle

    return temp; // return old rectangle

}

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

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

    rectangle temp = rect; // copy old rectangle

    rect.decrement(); // decrement rectangle

    return temp; // return old rectangle

}
```

```
friend rectangle operator+(rectangle &rect, int v) {  
  
    return {rect.height, rect.width + v};  
  
}  
  
friend void operator+=(rectangle &rect, int v) {  
  
    rect.width += v;  
  
}  
  
friend ostream &operator<<(ostream &os, rectangle rect) {  
  
    // output : (height x width)  
  
    os << "(" << rect.height << " x " << rect.width << " )";  
  
    return os; // return stream  
  
}  
  
friend istream &operator>>(istream &is, rectangle &rect) {  
  
    is >> rect.height; // input : height  
  
    rect.width = 0; // set width = 0  
  
    return is; // return stream  
  
}  
  
[[nodiscard]] int superficie() const {  
  
    return height * width; // area(superficie) = height * width  
  
}
```

```
};
```

```
int main() {
```

```
    int n; // number of columns
```

```
    int res = INT_MIN; // result : largest area(superficie), set to INT_MIN
```

```
    cin >> n; // input n
```

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

```
    queue<rectangle> q, old_queue, new_queue;
```

```
    // q : contains original rectangles
```

```
    // old_queue : (current_queue) contains current rectangles
```

```
    // new_queue : (next_queue) contains next rectangles
```

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

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

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

```
    }
```

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

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

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

```
        while (!old_queue.empty()) { // loop
```

```
rectangle old_rect = old_queue.front(); // get from old queue

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


if (old_rect.height < rect.height) { // check if

    new_queue.push(++old_rect); // rectangle extension and push into next queue

} else {

    rect.width = max(rect.width, old_rect.width); // update for original rectangle

    res = max(res, old_rect.superficie()); // update largest rectangle

}

}

old_queue.swap(new_queue); // switch to next queue

old_queue.push(++rect); // push original rectangle into current queue

}


while (!old_queue.empty()) { // loop remaining rectangle

    rectangle old_rect = old_queue.front(); // get rectangle

    old_queue.pop(); // remove from queue

    res = max(res, old_rect.superficie()); // update remaining rectangle

}


cout << res; // print result

return 0;
```



}

// Nguyen Van Duy - 20215334

**Bài tập 4.13.** Cho một xâu nhị phân độ dài  $n$ . Hãy viết chương trình đếm số lượng xâu con chứa số ký tự 0 và số ký tự 1 bằng nhau.

```

1 // Nguyen Van Duy - 20215334
2 /*
3 Bài 4.13. Cho một xâu nhị phân độ dài  $n$ .
4 Hãy viết chương trình đếm số lượng xâu con chứa số ký tự 0 và số ký tự 1 bằng nhau.
5 */
6
7 #include <iostream>
8 #include <map>
9
10 using namespace std;
11
12 inline int calc(int v) {
13     // voi so lan xuat hien chenh lech la v, thi so xau dem duoc la:
14     // result = 1 + 2 + 3 + ... + (v-1) = v * (v-1) / 2
15     return v * (v-1) / 2;
16 }
17
18 int main() {
19     string str;
20     cin >> str;
21
22     map<int, int> m; // <chenh lech : so lan>
23     map<int, int>::iterator it;
24     int chenh_lech = 0; // chenh lech = so luong 1 - so luong 0
25
26     for (char c : str) {
27         if (c == '0') chenh_lech--; // giam 1 neu xuat hien 0
28         else chenh_lech++; // tang 1 neu xuat hien 1
29
30         it = m.find(x: chenh_lech);
31         if (it != m.end()) { // neu do chenh lech da ton tai
32             it->second += 1; // tang so lan xuat hien cua chenh lech
33         } else {
34             m.insert(x: { &: chenh_lech, y: 1});
35         }
36     }
37
38     int res = 0; // result
39     for (it = m.begin(); it != m.end() ; ++it) {
40         res += calc(v: it->second);
41         if (it->first == 0) res += it->second; // neu chenh lech = 0
42     }
43
44     cout << res; // print result
45     return 0;
46 }
47
48 // Nguyen Van Duy - 20215334
49

```

```
C:\Users\Admin\CLionProjects\test\cmake-build-debug\test.exe
```

```
1001011
```

```
8
```

```
Process finished with exit code 0
```

```
C:\Users\Admin\CLionProjects\test\cmake-build-debug\test.exe
```

```
10011001101011010101000111111111000010100011010100010010010100111101001100101010101101111011110111
```

```
583
```

```
Process finished with exit code 0
```

```
C:\Users\Admin\CLionProjects\test\cmake-build-debug\test.exe
```

```
11110101100100011000000100011111001111000010001100000011000001110100110111101100000011
```

```
14342
```

```
Process finished with exit code 0
```

```
C:\Users\Admin\CLionProjects\test\cmake-build-debug\test.exe
```

```
10110000011111010101011001011011110101110100000110101111010000011010011110000011011111
```

```
10354354
```

```
Process finished with exit code 0
```

```
1111011000000011000011000001000000110001011100000010011001110016  
000001001110100100010110001011110000010110000111011110000110106  
0001010111110110010100111011100110001000000010100001010010001106  
001010111011010000000001000111000011111001000101110101010111016  
010111101101001001001010010100010101010011101011110011110010106  
1110000110001100011011110000001010010010000100010101111001111106  
11101000111110001011011001011011110010000101101110110110110006  
0001010001101001010110100110100001111011011011010001001110110016  
001010000110110101010001111100101001000111111011100001111011106  
0111110001101100110111001110010100000010110011100101011101101001  
001010011100000101111010010100100111100000000000100000101000106  
1101111111101110010000000101000000100010111100000100100000101001  
0110110111000100101100110110000111001011011010101000001100110011  
110110010010000000111100001111000110001011100011001000010110101  
000000000110101011001010
```

```
281665459
```

```
Process finished with exit code 0
```

```
01000101010001011000110101001101111110011010000010001001101110100010000100010001001
0010111101010011111110011101101110100101111111100110100100000010101010111010010001
11101011010001101010101101000000110001110100111101011100100001001001100110000100101
1011000100010011100011011111000111111101101000010000111111101110000000011011010110
10011010000111101010110001100010110100111111010101110101010001011010011101111001101
100011101101000011011110100001011100011101101000000111010001010000100000111011100
01111110001111001111000000110110101001101110000001010000110001010001000100110010
0111110010001001010100010010001111110100100110010101000001111010000011011100000010
01110101111010111000000011100110100000010010101100011101110000010111100101111100
0011101100111011110100110010001100101001110000100101110011111111001110110101110001
0001111011010001001101101001011101011110000000110
```

```
Process finished with exit code 0
```

 $\diagup^*$ 

Hãy viết chương trình đếm số lượng xâu con chứa số ký tự 0 và số ký tự 1 bằng nhau.

 $\ast/$ 

```
#include <map>
```

```
inline int calc(int v) {
```

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

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

}

```
int main() {  
  
    string str;  
  
    cin >> str;  
  
  
    map<int, int> m; // <chenh lech : so lan>  
  
    map<int, int>::iterator it;  
  
    int chenh_lech = 0; // chenh lech = so luong 1 - so luong 0  
  
  
    for (char c : str) {  
  
        if (c == '0') chenh_lech--; // giam 1 neu xuat hien 0  
  
        else chenh_lech++; // tang 1 neu xuat hien 1  
  
  
        it = m.find(chenh_lech);  
  
        if (it != m.end()) { // neu do chenh lech da ton tai  
  
            it->second += 1; // tang so lan xuat hien cua chenh lech  
  
        } else {  
  
            m.insert({chenh_lech, 1});  
  
        }  
    }  
  
  
    int res = 0; // result  
  
    for (it = m.begin(); it != m.end() ; ++it) {  
  
        res += calc(it->second);  
  
    }  
}
```

**Nguyễn Văn Duy – 20215334**

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

```
}
```

```
cout << res; // print result
```

```
return 0;
```

```
}
```

```
// Nguyen Van Duy - 20215334
```