MỤC LỤC

Bài thực hành số 4 – Tuần 14………………………………………………………………..02

Bài 4.1: Đảo ngược một danh sách liên kết đơn …….…………...………………………….02

Bài 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.…………………….………………...……………………..04

Bài 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..……….…………………..……….06

Bài 4.4: Cho hai std::vector, hãy xóa hết các phần tử chẵn, sắp xếp giảm dần các số trong cả 2 vector và trộn lại thành một vector cũng được sắp xếp giảm dần.…………………………...08

Bài 4.5: Viết hàm void dfs(vector< list<int> > adj) thực hiện thuật toán DFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề. Đồ thị có n đỉnh được đánh số từ 1 đến n. Thuật toán DFS xuất phát từ đỉnh 1. Các đỉnh được thăm theo thứ tự ưu tiên từ trái sang phải trong danh sách kề. Yêu cầu hàm trả ra thứ tự các đỉnh được thăm (những đỉnh không thể thăm từ đỉnh 1 thì không phải in ra).…………………………………………………………10

Bài 4.6: Viết hàm void bfs(vector< list<int> > adj) thực hiện thuật toán BFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề. Đồ thị có n đỉnh được đánh số từ 1 đến n. Thuật toán BFS xuất phát từ đỉnh 1. Các đỉnh được thăm theo thứ tự ưu tiên từ trái sang phải trong danh sách kề. Yêu cầu hàm trả ra thứ tự các đỉnh được thăm (những đỉnh không thể thăm từ đỉnh 1 thì không phải in ra).……………………………………...………………….12

Bài 4.7: Viết các hàm thực hiện các phép giao và hợp của hai tập hợp được biểu diễn bằng set…………………………………………………………………………………………….14

Bài 4.8: Viết các hàm thực hiện các phép giao và hợp của hai tập hợp mờ được biểu diễn bằng map.…………………………………..……………………………………..………….16

Bài 4.9: Cài đặt thuật toán Dijkstra trên đồ thị vô hướng được biểu diễn bằng danh sách kề sử dụng std::priority\_queue.……………………………………………………..……………...19

Bài 4.10: Xây dựng một máy tìm kiếm (search engine) đơn giản..…………….……………21

Bài 4.11: Bảo vệ lâu đài. 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..……………..22

Bài 4.12: Cho một lược đồ gồm n cột chữ nhật liên tiếp nhau có chiều rộng bằng 1 và chiều cao lần lượt là các số nguyên không âm h1,h2,…,hn . Hãy xác định hình chữ nhật có diện tích lớn nhất có thể tạo thành từ các cột liên tiếp.…………………....………………………23

Bài 4.13: Cho một xâu nhị phân độ dài n . Hãy viết chương trình đếm số lượng xâu con chứa số ký tự 0 và số ký tự 1 bằng nhau..…………………....………………………….…………24

BÀI THỰC HÀNH SỐ 4 – TUẦN 14

Bài 4.1: Đảo ngược một danh sách liên kết đơn.

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

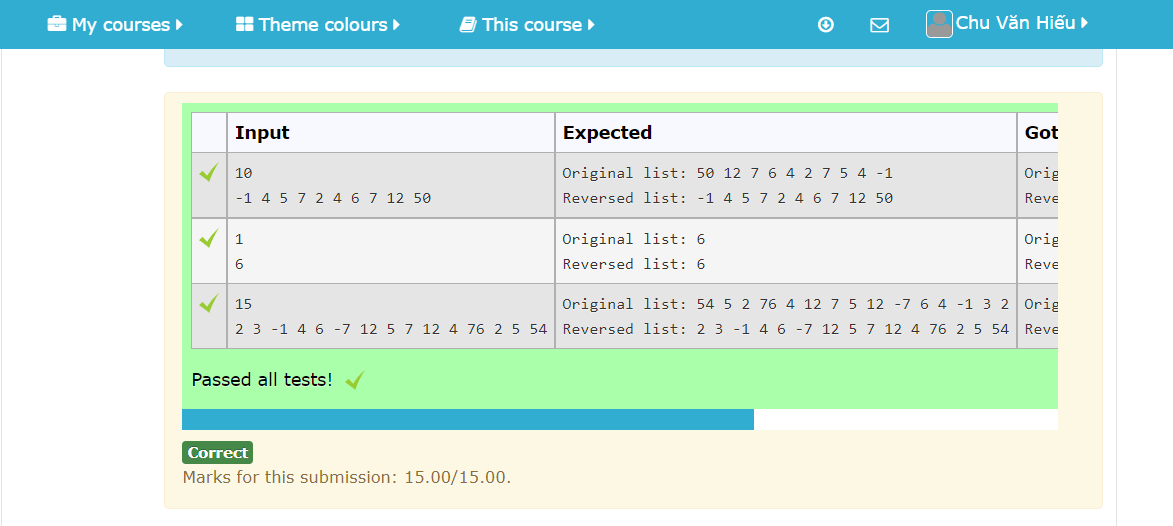
* Thêm một phần tử vào đầu danh sách liên kết
* In danh sách
* Đảo ngược danh sách liên kết (yêu cầu độ phức tạp thời gian O(N) và chi phí bộ nhớ dùng thêm O(1))

Ảnh có chứa văn bản

Mô tả được tạo tự động Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động 

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

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

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

# YOUR CODE HERE #

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

}

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

using Point = pair<double, double>; Ảnh có chứa văn bản

Mô tả được tạo tự động Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

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

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

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

# YOUR CODE HERE #

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

}

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

using Vector = tuple<double, double, double>; Ảnh có chứa văn bản

Mô tả được tạo tự động

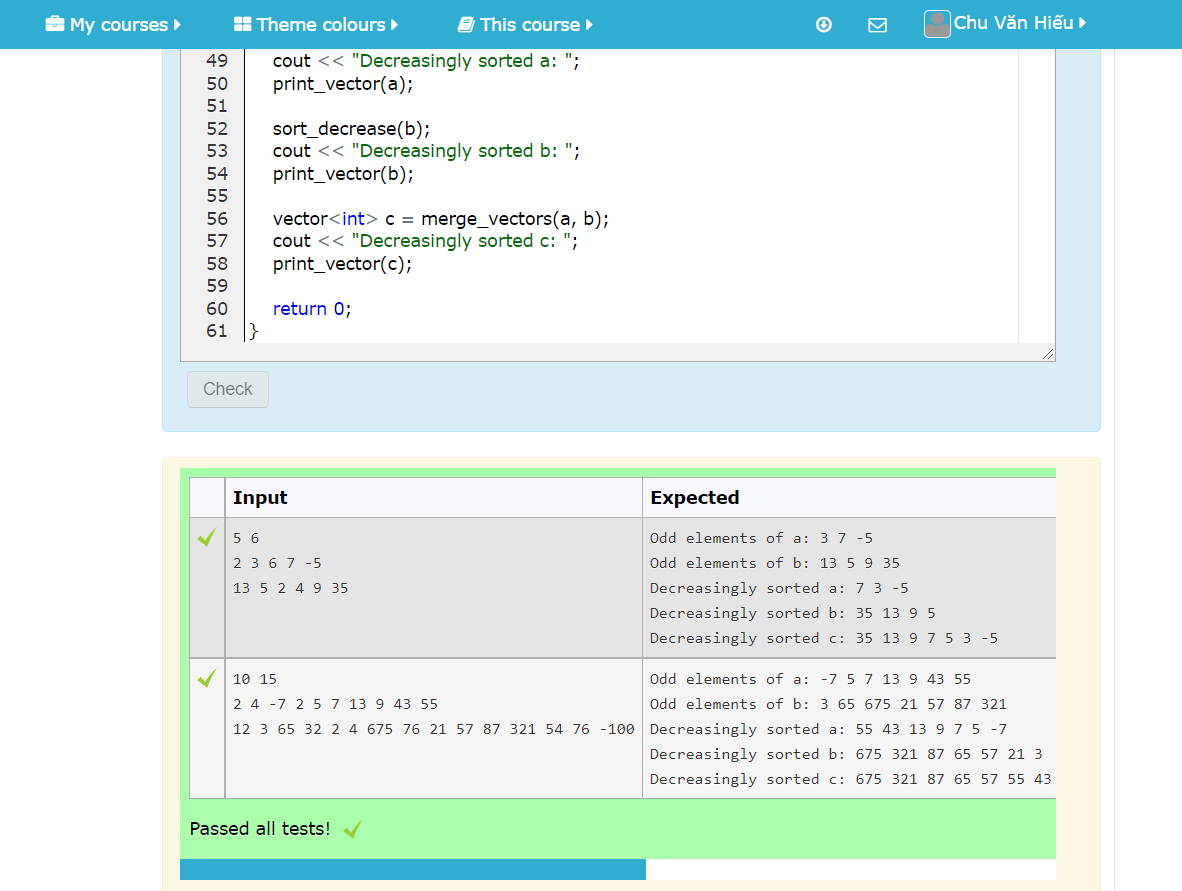
Ảnh có chứa văn bản

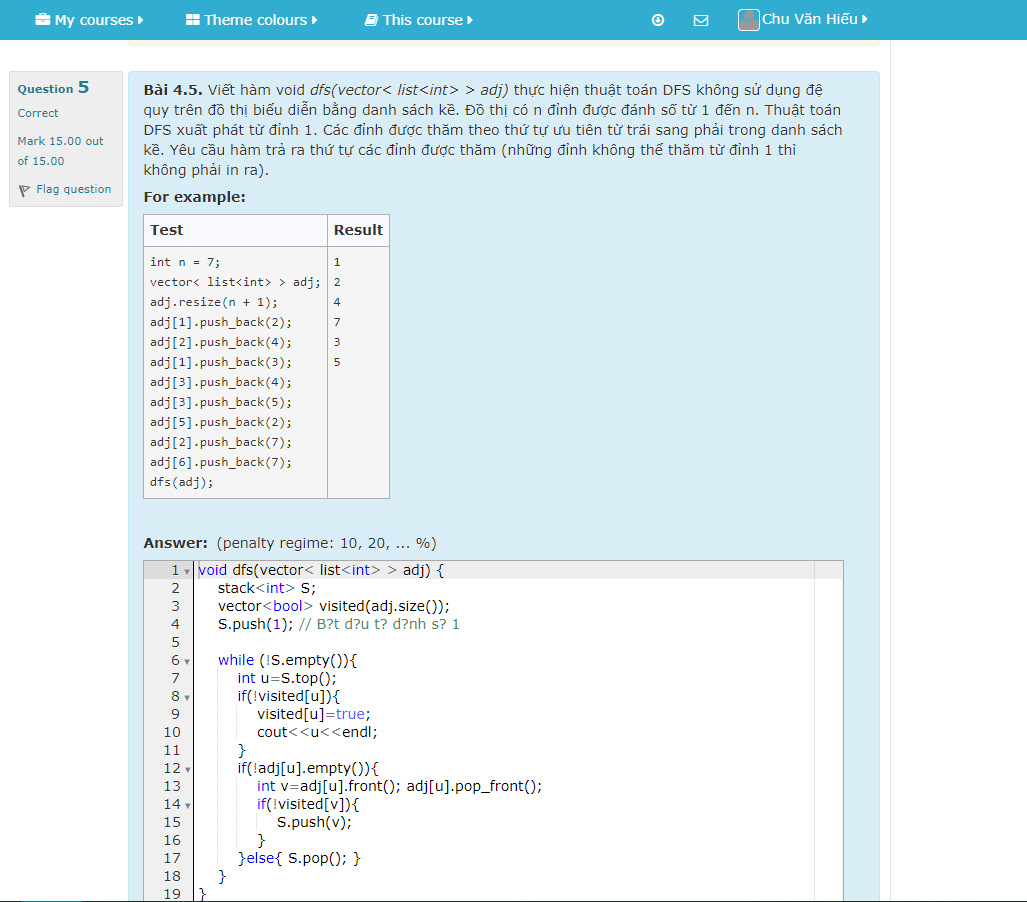
Mô tả được tạo tự động

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

Mô tả được tạo tự động Ảnh có chứa văn bản

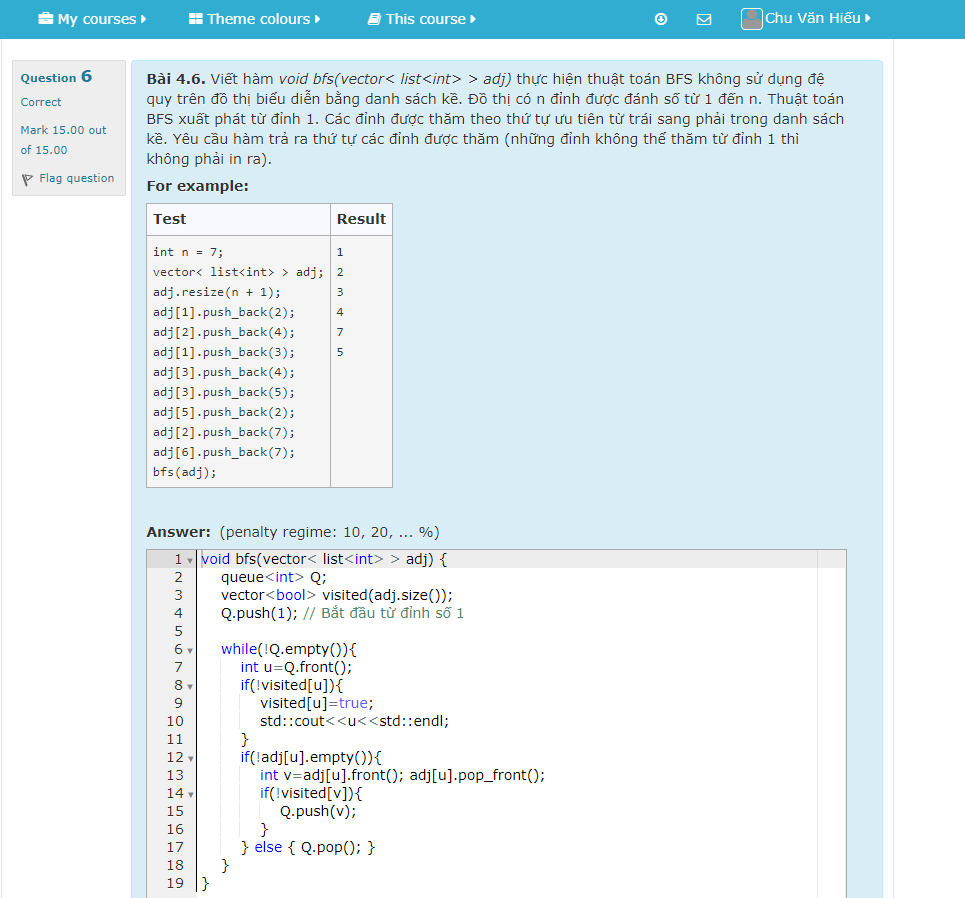
Mô tả được tạo tự động



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

Ảnh có chứa bàn

Mô tả được tạo tự động

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

Ảnh có chứa bàn

Mô tả được tạo tự động

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

template<class T>

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

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

# YOUR CODE HERE #

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

}

template<class T>

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

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

# YOUR CODE HERE #

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

}

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

template<class T>

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

for (const T &x : a) {

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

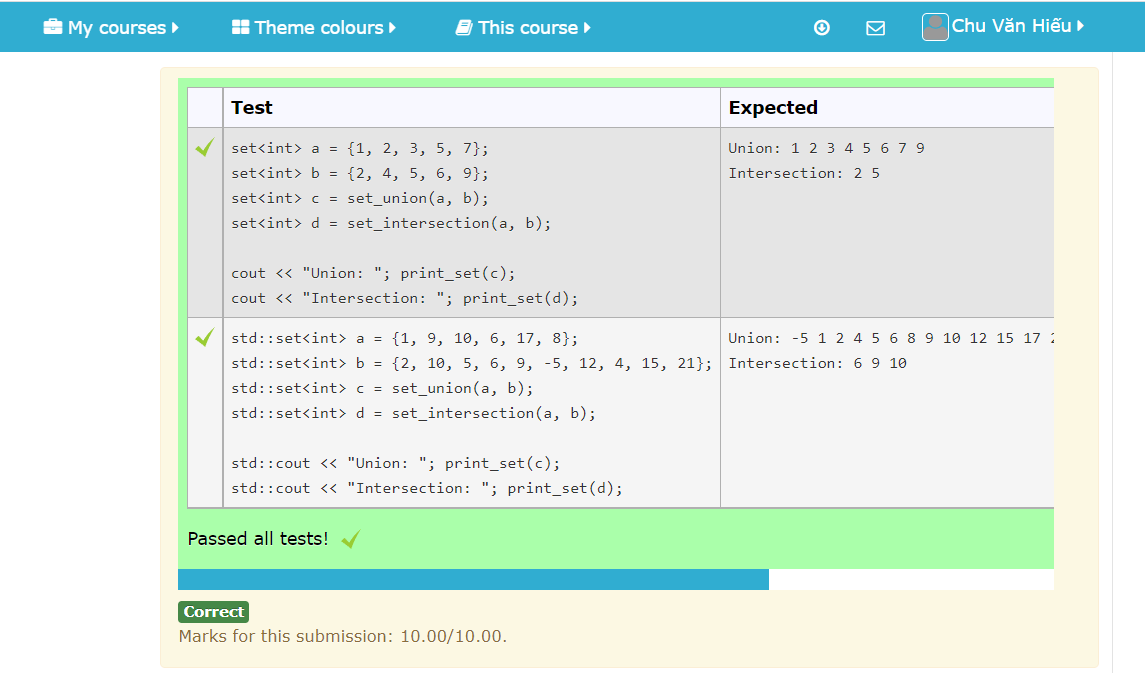
}

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

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động 

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

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

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

template<class T>

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

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

# YOUR CODE HERE #

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

}

template<class T>

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

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

# YOUR CODE HERE #

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

}

Lưu ý: Trong ví dụ dưới đây, hàm print\_fuzzy\_set() được định nghĩa sẵn như sau:

template<class T>

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

cout << "{ ";

for (const auto &x : a) {

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

}

cout << "}";

cout << endl;

}

Ảnh có chứa văn bản

Mô tả được tạo tự động Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

Bài 4.9: Cài đặt thuật toán Dijkstra trên đồ thị vô hướng được biểu diễn bằng danh sách kề sử dụng std::priority\_queue.

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

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

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

# YOUR CODE HERE #

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

}

Ảnh có chứa văn bản

Mô tả được tạo tự động Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

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

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

Sử dụng phương pháp tính điểm TF-IDF:

Ảnh có chứa văn bản

Mô tả được tạo tự động

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

**Input:**

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

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

**Giới hạn:**

* N ≤ 1000
* Q ≤ 1000
* Số từ trong mỗi văn bản không quá 1000
* Số từ trong mỗi truy vấn không quá 10
* Độ dài mỗi từ không quá 10

**Tham khảo:**

* <https://en.wikipedia.org/wiki/Tf%E2%80%93idf>

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.

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 ki tên giặc.

Giả sử đoạn tường thứ i có xi lính. Khi đó nếu số tên giặc không vượt quá xi×ki 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 bố lính đứng ở các đoạn tường sao cho tổng số lính là s và tổng số lượng tên giặc lọt vào lâu đài là nhỏ nhất.

**Dữ liệu vào:**

Dòng thứ nhất chứa các số nguyên nn và ss (1 ≤ n ≤ 100000; 1 ≤ s ≤ 10^9).

nn dòng tiếp theo chứa hai số nguyên aiai và kiki lần lượt là số tên giặc tấn công đoạn tường thứ ii và khả năng chống trả của một lính ở đoạn tường thứ ii (1 ≤ ai,ki ≤ 10^9).

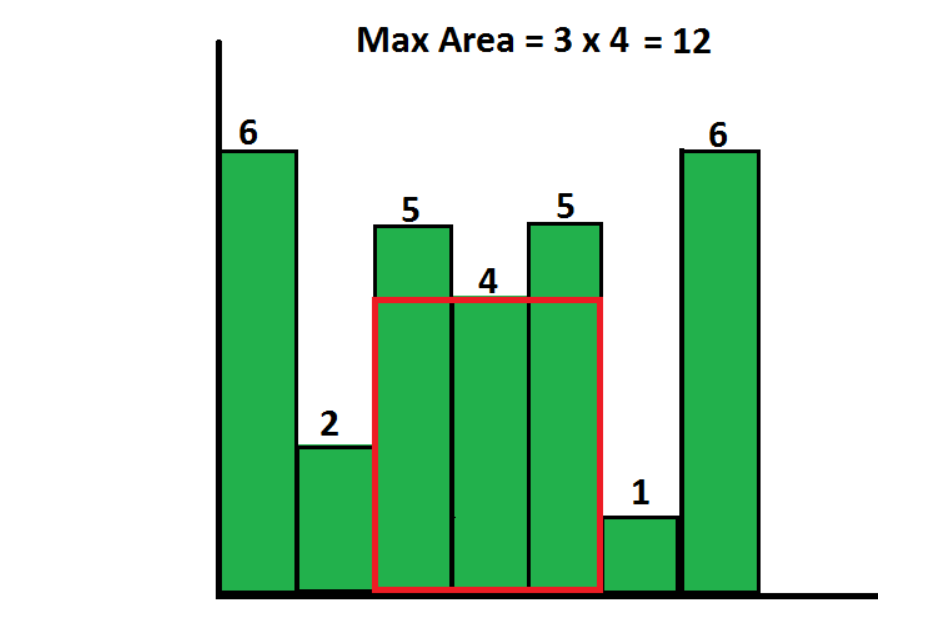
**Kết quả:**

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

Bài 4.12: Cho một lược đồ gồm n cột chữ nhật liên tiếp nhau có chiều rộng bằng 1 và chiều cao lần lượt là các số nguyên không âm h1,h2,…,hn . Hãy xác định hình chữ nhật có diện tích lớn nhất có thể tạo thành từ các cột liên tiếp.

**Dữ liệu vào:**  
Dòng thứ nhất chứa số nguyên dương nn (1 ≤ n ≤ 10^6).  
Dòng thứ hai chứa nn số nguyên không âm h1,h2,…,hnh1,h2,…,hn cách nhau bởi dấu cách (0 ≤ hi ≤ 10^9).

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



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

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

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