MỤC LỤC

[Bài thực hành số 4: Thực hành sử dụng các cấu trúc dữ liệu cơ bản để giải quyết các bài toán cụ thể 4](#_Toc58862691)

[Phần 1. Bài tập thực hành 4](#_Toc58862692)

[Bài tập 1: Đảo ngược 1 danh sách liên kết đơn Hãy hoàn thiện hàm thao tác trên 1 danh sách liên kết đơn : - Thêm 1 phần tử vào đầu danh sách liên kết - In danh sách - Đảo ngược danh sách liên kết 4](#_Toc58862693)

[Bài tập 2: Tính diện tích tam giác Một điểm trong không gian 2 chiều được biểu diễn bằng pair. Hãy viết hàm double area(Point a, Point b, Point c) tính diện tích tam giác theo tọa độ 3 đỉnh. Trong đó Point là kiểu được định nghĩa sẵn trong trình chấm như sau using Point = pair<double, double>; 6](#_Toc58862694)

[Bài tập 3: Tính tích có hướng của 2 vector Một vector trong không gian 3 chiều được biểu diễn bằng tuple<double, double, double>. Hãy viết hàm Vector cross\_product( Vector a, Vector b) tính tích có hướng của 2 vector. Trong đó kiểu dữ liệu được định nghĩa sẵn trong trình chấm như sau : using Vector = tuple<double, double, double> 8](#_Toc58862695)

[Bài tập 4: Thao tác với vector Cho 2 vector, hãy xóa hết các phần tử chẵn, sắp xếp giảm dần các số trong cả 2 vector trên và trộn lại thành một vector cũng được sắp xếp giảm dần 9](#_Toc58862696)

[Bài tập 5: Viết hàm thực hiện thuật toán DFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề vector< list<int> >. Đồ thị có n đỉnh được đánh số từ 1 đến n. Thuật toán DFS xuất phát từ đỉnh 1. Các đỉnh được thăm theo thứ tự từ trái sang phải theo 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) 13](#_Toc58862697)

[Bài tập 6: Viết hàm thực hiện thuật toán BFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề vector< list<int> >. Đồ thị có n đỉnh được đánh số từ 1 đến n. Thuật toán BFS xuất phát từ đỉnh 1. Các đỉnh được thăm theo thứ tự ưu tiên từ trái sang phải trong danh sách kề. Yêu cầu in ra các đỉnh được thăm (những đỉnh không thể thăm từ đỉnh 1 thì không phải in ra) 15](#_Toc58862698)

[Bài tập 7: Viết các hàm thực hiện phép giao và phép hợp của 2 tập hợp được biểu diễn bằng set 17](#_Toc58862699)

[Bài tập 8: Viết hàm thực hiện phép giao và hợp của 2 tập hợp mờ được biểu diễn bằng map 20](#_Toc58862700)

[Bài tập 9: Cài đặt thuật toán Dijikstra 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> dijikstra(const vector< vector< pari<int, int> > > &adj) nhận đầu vào là danh sách kề chứa các cặp pair<int, int> biểu diễn đỉnh kề và trọng số của cạnh tương ứng. Đồ thị gồm n đỉnh được đánh số từ 0 đến n-1. Hàm cần trả về 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 24](#_Toc58862701)

[Phần 2. Bài tập về nhà 28](#_Toc58862702)

[Bài tập 10: Search Engine Xây dựng 1 máy tìm kiếm đơn giản. Cho N văn bản và Q truy vấn. Với mỗi truy vấn trả về văn bản khớp với truy vấn đó nhất. Sử dụng phương pháp tính điêm TF-IDF: - f(t,d) : là số lần xuất hiện tử từ t trong văn bản d - maxf(d) : là giá trị lớn nhất của f(t,d) với mọi t - df(t) : là số văn bản chứa từ t - TF(t,d) = 0.5 + 0.5 \* f(t,d) / maxf(d) - IDF(t) = log2(N / df(t)) - Điểm số của từ t trong văn bản d là score(t,d) = TF(t,d) \* IDF(t), nếu từ t không xuất hiện trong văn bản d thì score(t,d) = 0 - Điểm số của văn bản d với các truy vấn bằng tổng số điểm của từng từ đối với văn bản đó. Ta coi văn bản càng cao thì càng khớp với truy vấn 28](#_Toc58862703)

[Bài tập 11: Bảo vệ lâu đài Bức tường bao quanh 1 lâu đài nọ được cấu thành từ n đoạn tường được đánh số từ 1 đến n. Quân địch 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 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 dược đoạn tường này. Yêu cầu hãy viết chương trình phân bố lính canh ở các đoạn tường sao cho số tên địch lọt vào là ít nhất. 35](#_Toc58862704)

[Bài tập 12: Lược đồ 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 39](#_Toc58862705)

[Bài tập 13: Đếm xâu con Cho 1 xâu nhị phân độ dài n. Hãy viết chương trình đếm số lượng xâu con chứa số kí tự 0 và 1 bằng nhau 42](#_Toc58862706)

DANH MỤC HÌNH ẢNH

[Hình 1 : Đảo ngược danh sách liên kết 6](#_Toc58863029)

[Hình 2 : Tính diện tích tam giác 7](#_Toc58863030)

[Hình 3 : Tính tích có hướng của 2 vector 9](#_Toc58863031)

[Hình 4 : Thao tác với vector 13](#_Toc58863032)

[Hình 5 : DFS 15](#_Toc58863033)

[Hình 6 : BFS 17](#_Toc58863034)

[Hình 7 : Phép giao và hợp của 2 tập hợp biểu diễn bằng set 20](#_Toc58863035)

[Hình 8 : Phép giao và hợp của 2 tập hợp mờ được biêu diễn bằng map 23](#_Toc58863036)

[Hình 9 : Thuật toán Dijikstra 27](#_Toc58863037)

[Hình 10 : Máy tìm kiếm đơn giản 35](#_Toc58863038)

[Hình 11 : Bảo vệ lâu đài 39](#_Toc58863039)

[Hình 12 : Hình chữ nhật có diện tích lớn nhất 42](#_Toc58863040)

[Hình 13 : Đếm xâu con 45](#_Toc58863041)

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

Phần 1. Bài tập thực hành

Bài tập 1: Đảo ngược 1 danh sách liên kết đơn  
Hãy hoàn thiện hàm thao tác trên 1 danh sách liên kết đơn :   
- Thêm 1 phần tử vào đầu danh sách liên kết  
- In danh sách   
- Đảo ngược danh sách liên kết

**Tên file: 20183554-NguyenQuangHuy\_Bai 4\_1.cpp**

#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) {

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

Ho va ten : Nguyen Quang Huy

MSSV : 20183554

\*/

Node\* new\_node = new Node(data);

new\_node->next = head;

head = new\_node;

return head;

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

}

// print the list content on a line

void print(Node\* head) {

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

Ho va ten : Nguyen Quang Huy

MSSV : 20183554

\*/

Node \*tmp = head;

while(tmp != NULL){

printf("%d ",tmp->data);

tmp = tmp->next;

}

cout << endl;

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

}

// return the new head of the reversed list

Node\* reverse(Node\* head) {

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

Node\* current = head->next;

Node\* prev = head; prev->next = NULL;

Node\* next = NULL;

while(current != NULL){

next = current->next;

current->next = prev;

prev = current;

current = next;

}

head = prev;

return head;

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

}

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);

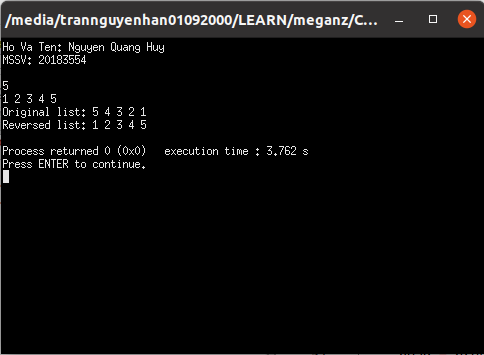
cout << "Reversed list: ";

print(head);

return 0;

}

Kết quả:



Hình : Đảo ngược danh sách liên kết

Bài tập 2: Tính diện tích tam giác  
Một điểm trong không gian 2 chiều được biểu diễn bằng pair. Hãy viết hàm double area(Point a, Point b, Point c) tính diện tích tam giác theo tọa độ 3 đỉnh. Trong đó Point là kiểu được định nghĩa sẵn trong trình chấm như sau using Point = pair<double, double>;

### Tên file: 20183554-NguyenQuangHuy\_Bai 4\_2.cpp

### #include <iostream>

#include <cmath>

#include <iomanip>

#include <utility>

using namespace std;

using Point = pair<double, double>;

double distance(Point a, Point b){

return sqrt((a.first - b.first)\*(a.first - b.first) + (a.second - b.second)\*(a.second - b.second));

}

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

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

Ho va ten : Nguyen Quang Huy

MSSV: 20183554

\*/

double ab = distance(a,b);

double bc = distance(b,c);

double ca = distance(c,a);

double p = (ab + bc + ca) /2;

return sqrt(p \* (p-ab) \* (p-bc) \* (p-ca));

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

}

int main() {

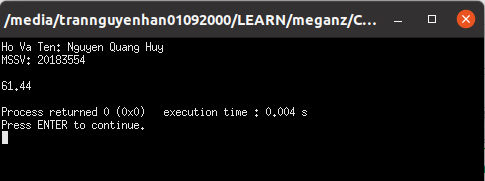
cout << setprecision(2) << fixed;

cout << area({1, 2}, {2.5, 10}, {15, -5.25}) << endl;

return 0;

}

### Kết quả:



Hình : Tính diện tích tam giác

Bài tập 3: Tính tích có hướng của 2 vector  
Một vector trong không gian 3 chiều được biểu diễn bằng tuple<double, double, double>. Hãy viết hàm Vector cross\_product( Vector a, Vector b) tính tích có hướng của 2 vector. Trong đó kiểu dữ liệu được định nghĩa sẵn trong trình chấm như sau : using Vector = tuple<double, double, double>

### Tên file: 20183554-NguyenQuangHuy\_Bai 4\_3.cpp

### #include <iostream>

#include <cmath>

#include <iomanip>

using namespace std;

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

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

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

Ho va ten : Nguyen Quang Huy

MSSV : 20183554

\*/

double x = (get<1>(a)\*get<2>(b) - get<1>(b)\*get<2>(a));

double y = -(get<2>(a)\*get<0>(b) - get<0>(a)\*get<2>(b));

double z = (get<0>(a)\*get<1>(b) - get<0>(b)\*get<1>(a));

return {x,y,z};

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

}

int main() {

cout << setprecision(2) << fixed;

Vector a {1.2, 4, -0.5};

Vector b {1.5, -2, 2.5};

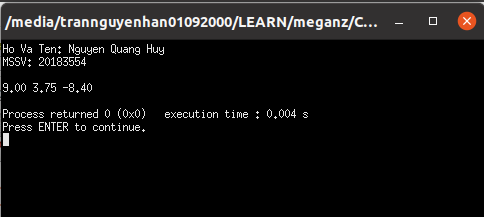
Vector c = cross\_product(a, b);

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

return 0;

}

### Kết quả:



Hình : Tính tích có hướng của 2 vector

Bài tập 4: Thao tác với vector  
Cho 2 vector, hãy xóa hết các phần tử chẵn, sắp xếp giảm dần các số trong cả 2 vector trên và trộn lại thành một vector cũng được sắp xếp giảm dần

### Tên file: 20183554-NguyenQuangHuy\_Bai 4\_4.cpp

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

void print\_vector(const vector<int> &a) {

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

cout << endl;

}

bool compare(int a, int b){

return a > b;

}

vector<int> delete\_even(vector<int> &a) {

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

Ho va ten : Nguyen Quang Huy

MSSV : 20183554

\*/

vector<int> tmp;

for(int v : a){

if(v % 2 != 0) tmp.push\_back(v);

}

return tmp;

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

}

void sort\_decrease(vector<int> &a) {

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

Ho va ten : Nguyen Quang Huy

MSSV : 20183554

\*/

sort(a.begin(), a.end(), compare);

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

}

vector<int> merge\_vectors(const vector<int> &a, const vector<int> &b) {

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

vector<int> sum;

for(int v : a){

sum.push\_back(v);

}

for(int v : b){

sum.push\_back(v);

}

sort(sum.begin(), sum.end(), compare);

return sum;

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

}

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);

}

a = delete\_even(a);

cout << "Odd elements of a: ";

print\_vector(a);

b = 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);

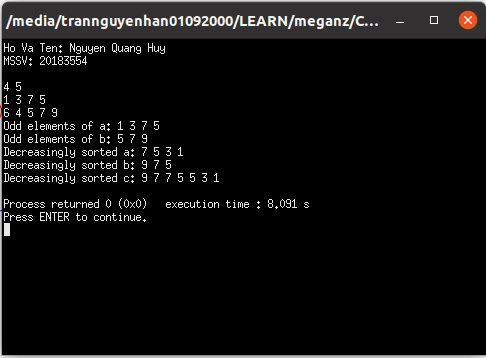
cout << "Decreasingly sorted c: ";

print\_vector(c);

return 0;

}

Kết quả :



Hình : Thao tác với vector

Bài tập 5: Viết hàm thực hiện thuật toán DFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề vector< list<int> >. Đồ thị có n đỉnh được đánh số từ 1 đến n. Thuật toán DFS xuất phát từ đỉnh 1. Các đỉnh được thăm theo thứ tự từ trái sang phải theo 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)

### Tên file: 20183554-NguyenQuangHuy\_Bai 4\_5.cpp

#include<bits/stdc++.h>

using namespace std;

void dfs(std::vector< std::list<int> > adj) {

std::stack<int> S;

std::vector<bool> visited(adj.size());

S.push(1); // Bắt đầu từ đỉnh số 1

while (!S.empty()) {

int u=S.top();

if (!visited[u]){

visited[u] = true;

std::cout<< u << std::endl;

}

if (!adj[u].empty()){

int v=adj[u].front(); adj[u].pop\_front();

if(!visited[v]){

S.push(v);

}

}else { S.pop();}

}

}

int main(){

int n = 7;

vector< list<int> > adj;

adj.resize(n + 1);

adj[1].push\_back(2);

adj[2].push\_back(4);

adj[1].push\_back(3);

adj[3].push\_back(4);

adj[3].push\_back(5);

adj[5].push\_back(2);

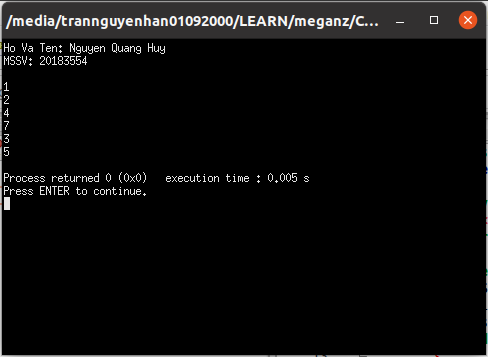
adj[2].push\_back(7);

adj[6].push\_back(7);

dfs(adj);

}

Kết quả:



Hình : DFS

Bài tập 6: Viết hàm thực hiện thuật toán BFS không sử dụng đệ quy trên đồ thị biểu diễn bằng danh sách kề vector< list<int> >. Đồ thị có n đỉnh được đánh số từ 1 đến n. Thuật toán BFS xuất phát từ đỉnh 1. Các đỉnh được thăm theo thứ tự ưu tiên từ trái sang phải trong danh sách kề. Yêu cầu in ra các đỉnh được thăm (những đỉnh không thể thăm từ đỉnh 1 thì không phải in ra)

### Tên file: 20183554-NguyenQuangHuy\_Bai 4\_6.cpp

#include<bits/stdc++.h>

using namespace std;

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

queue<int> Q;

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

Q.push(1); // Bắt đầu từ đỉnh số 1

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

Ho va ten : Nguyen Quang Huy

MSSV : 20183554

\*/

while (!Q.empty()) {

int u=Q.front();

if (!visited[u]){

visited[u] = true;

std::cout<< u << std::endl;

}

if (!adj[u].empty()){

int v=adj[u].front(); adj[u].pop\_front();

if(!visited[v]){

Q.push(v);

}

}else { Q.pop();}

}

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

}

int main(){

int n = 7;

vector< list<int> > adj;

adj.resize(n + 1);

adj[1].push\_back(2);

adj[2].push\_back(4);

adj[1].push\_back(3);

adj[3].push\_back(4);

adj[3].push\_back(5);

adj[5].push\_back(2);

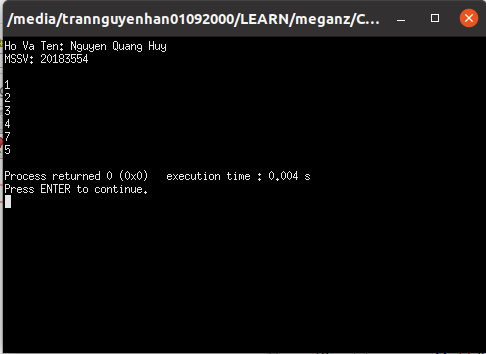
adj[2].push\_back(7);

adj[6].push\_back(7);

bfs(adj);

}

Đáp án:



Hình : BFS

Bài tập 7: Viết các hàm thực hiện phép giao và phép hợp của 2 tập hợp được biểu diễn bằng set

**Tên file: 20183554-NguyenQuangHuy\_Bai 4\_7.cpp**

#include <iostream>

#include <set>

using namespace std;

template<class T>

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

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

Ho va ten : Nguyen Quang Huy

MSSV : 20183554

\*/

set<T> c;

for(T v : a){

c.insert(v);

}

for(T v : b){

c.insert(v);

}

return c;

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

}

template<class T>

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

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

set<T> c;

for(T v : a){

int index = 0;

for(T u : b){

if(v == u) index++;

}

if(index != 0) c.insert(v);

}

return c;

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

}

template<class T>

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

for (const T &x : a) {

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

}

std::cout << std::endl;

}

int main() {

std::set<int> a = {1, 2, 3, 5, 7};

std::set<int> b = {2, 4, 5, 6, 9};

std::set<int> c = set\_union(a, b);

std::set<int> d = set\_intersection(a, b);

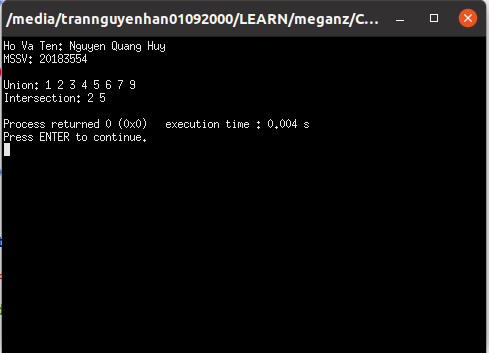
std::cout << "Union: "; print\_set(c);

std::cout << "Intersection: "; print\_set(d);

return 0;

}

Đáp án:



Hình : Phép giao và hợp của 2 tập hợp biểu diễn bằng set

Bài tập 8: Viết hàm thực hiện phép giao và hợp của 2 tập hợp mờ được biểu diễn bằng map

**Tên file: 20183554-NguyenQuangHuy\_Bai 4\_8.cpp**

#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) {

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

Ho va ten : Nguyen Quang Huy

MSSV : 20183554

\*/

map<T, double> c;

for(auto v : a){

int index = 0;

for(auto u : b){

if(v.first == u.first){

c.insert({v.first, max(v.second, u.second)});

index++;

break;

}

}

if(index == 0) c.insert({v.first, v.second});

}

for(auto v : b){

int index = 0;

for(auto u : a){

if(v.first == u.first){

c.insert({v.first, max(v.second, u.second)});

index++;

break;

}

}

if(index == 0) c.insert({v.first, v.second});

}

return c;

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

}

template<class T>

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

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

map<T, double> c;

for(auto v : a){

for(auto u : b){

if(v.first == u.first){

c.insert({v.first, min(v.second, u.second)});

break;

}

}

}

return c;

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

}

template<class T>

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

cout << "{ ";

for (const auto &x : a) {

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

}

cout << "}";

std::cout << std::endl;

}

int main() {

std::map<int, double> a = {{1, 0.2}, {2, 0.5}, {3, 1}, {4, 0.6}, {5, 0.7}};

std::map<int, double> b = {{1, 0.5}, {2, 0.4}, {4, 0.9}, {5, 0.4}, {6, 1}};

std::cout << "A = "; print\_fuzzy\_set(a);

std::cout << "B = "; print\_fuzzy\_set(b);

std::map<int, double> c = fuzzy\_set\_union(a, b);

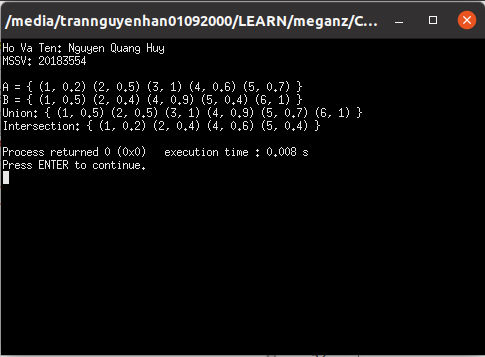
std::map<int, double> d = fuzzy\_set\_intersection(a, b);

std::cout << "Union: "; print\_fuzzy\_set(c);

std::cout << "Intersection: "; print\_fuzzy\_set(d);

}

Kết quả:



Hình : Phép giao và hợp của 2 tập hợp mờ được biêu diễn bằng map

Bài tập 9: Cài đặt thuật toán Dijikstra 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> dijikstra(const vector< vector< pari<int, int> > > &adj) nhận đầu vào là danh sách kề chứa các cặp pair<int, int> biểu diễn đỉnh kề và trọng số của cạnh tương ứng. Đồ thị gồm n đỉnh được đánh số từ 0 đến n-1. Hàm cần trả về 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

**Tên file: 20183554-NguyenQuangHuy\_Bai 4\_9.cpp**

#include <iostream>

#include <queue>

#include <climits>

using namespace std;

struct compare{

bool operator() (pair<int, int> a, pair<int, int> b){

return a.second > b.second;

}

};

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

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

Ho va ten : Nguyen Quang Huy

MSSV : 20183554

\*/

priority\_queue<pair<int, int>, vector< pair<int, int> >, compare> q;

vector<int> dist(adj.size());

int dist\_size = dist.size();

for(int i=0; i<dist\_size; i++){

dist[i] = INT\_MAX;

}

dist[0] = 0; // khoang cach tu 0 -> 0 = 0

// khoi tao gia tri cho hang doi uu tien

for(int i=0; i<dist\_size; i++){

q.push({i, dist[i]});

}

while(!q.empty()){

pair<int, int> u\_pair = q.top(); q.pop();

int u = u\_pair.first; // lay ra dinh co thu tu uu tien nhat trong hang doi

for(auto v\_pair : adj[u]){

int v = v\_pair.first;

int weight = v\_pair.second;

if(dist[v] > dist[u] + weight){

dist[v] = dist[u] + weight;

q.push({v, dist[v]});

}

}

}

return dist;

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

}

int main() {

int n = 9;

vector< vector< pair<int, int> > > adj(n);

auto add\_edge = [&adj] (int u, int v, int w) {

adj[u].push\_back({v, w});

adj[v].push\_back({u, w});

};

add\_edge(0, 1, 4);

add\_edge(0, 7, 8);

add\_edge(1, 7, 11);

add\_edge(1, 2, 8);

add\_edge(2, 3, 7);

add\_edge(2, 8, 2);

add\_edge(3, 4, 9);

add\_edge(3, 5, 14);

add\_edge(4, 5, 10);

add\_edge(5, 6, 2);

add\_edge(6, 7, 1);

add\_edge(6, 8, 6);

add\_edge(7, 8, 7);

vector<int> distance = dijkstra(adj);

for (int i = 0; i < distance.size(); ++i) {

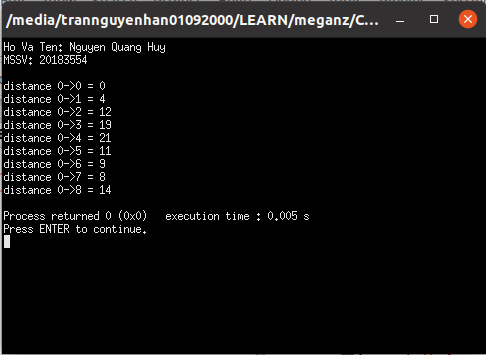
cout << "distance " << 0 << "->" << i << " = " << distance[i] << endl;

}

return 0;

}

Kết quả:



Hình : Thuật toán Dijikstra

Phần 2. Bài tập về nhà

Bài tập 10: Search Engine  
Xây dựng 1 máy tìm kiếm đơn giản.  
Cho N văn bản và Q truy vấn. Với mỗi truy vấn trả về văn bản khớp với truy vấn đó nhất. Sử dụng phương pháp tính điêm TF-IDF:  
- f(t,d) : là số lần xuất hiện tử từ t trong văn bản d  
- maxf(d) : là giá trị lớn nhất của f(t,d) với mọi t  
- df(t) : là số văn bản chứa từ t  
- TF(t,d) = 0.5 + 0.5 \* f(t,d) / maxf(d)  
- IDF(t) = log2(N / df(t))  
- Điểm số của từ t trong văn bản d là score(t,d) = TF(t,d) \* IDF(t), nếu từ t không xuất hiện trong văn bản d thì score(t,d) = 0  
- Điểm số của văn bản d với các truy vấn bằng tổng số điểm của từng từ đối với văn bản đó.  
Ta coi văn bản càng cao thì càng khớp với truy vấn

**Tên file: 20183554-NguyenQuangHuy\_Bai 4\_10.cpp**

#include<bits/stdc++.h>

using namespace std;

int n, q;

vector< vector<string> > vector\_train;

vector< vector<string> > vector\_test;

vector<int> f\_max; // tan suat xuat hien tu xuat hien nhieu nhat trong van ban i

map<string, int> df; // tinh xem tu word nam trong bao nhieu van ban

map<pair<string, int>, int> fe; // tinh xem tu word xuat hien bao nhieu lan trong van ban i <word, i> = int

// split string

vector<string> split\_string(string str){

vector<string> vt;

while (!str.empty()){

string tmp = str.substr(0, str.find(","));

int pos = tmp.find(" ");

if(pos > tmp.size())

vt.push\_back(tmp);

else {

while(pos <= tmp.size()){

tmp.erase(pos,1);

pos = tmp.find(" ");

}

vt.push\_back(tmp);

}

if (str.find(",") > str.size()){

break;

} else {

str.erase(0, str.find(",") + 1);

}

}

return vt;

}

// input

void input(){

vector<string> document\_train;

vector<string> document\_test;

cin >> n;

string str\_tmp;

getline(cin, str\_tmp);

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

string str\_tmp;

getline(cin, str\_tmp);

document\_train.push\_back(str\_tmp);

}

cin >> q;

getline(cin, str\_tmp);

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

string str\_tmp;

getline(cin, str\_tmp);

document\_test.push\_back(str\_tmp);

}

for(string v : document\_train){

vector<string> element = split\_string(v);

vector\_train.push\_back(element);

}

for(string v : document\_test){

vector<string> element = split\_string(v);

vector\_test.push\_back(element);

}

}

// preprocessing

void pre\_processing(){

// tinh tan suat tu xuat hien nhieu nhat trong van ban i

for(vector<string> str\_tmp : vector\_train){

map<string, int> m;

// thiet lap tu dien mini m voi chi so : [sotu] [tansuatxuathien]

int max\_f = 0;

for(string word\_tmp : str\_tmp){

map<string, int>::iterator ite = m.find(word\_tmp);

if(ite == m.end()){ // neu tu nay chua co trong tu dien mini

m.insert({word\_tmp, 1});

} else {

ite->second += 1;

}

max\_f = max(m[word\_tmp], max\_f);

}

f\_max.push\_back(max\_f);

}

}

// tinh xem word xuat hien bao nhieu lan trong van ban i

int frequence\_word\_int\_document\_i(string word, int i){

if(fe.find({word, i}) != fe.end()){ // neu da co trong kho luu tru thi lay ra va tra ve

return fe[{word, i}];

}

int index = 0;

vector<string> str\_tmp = vector\_train[i];

for(string v : str\_tmp){

if(word == v) index++;

}

fe.insert({{word, i}, index});

return index;

}

// tinh xem tu word nam trong bao nhieu van ban

int count\_document\_contain\_word(string word){ // neu da co trong kho luu tru thi lay ket qua va tra ve

if(df.find(word) != df.end()){

return df[word];

}

int index = 0;

for(vector<string> str\_tmp : vector\_train){

vector<string>::iterator ite = find(str\_tmp.begin(), str\_tmp.end(), word);

if(ite != str\_tmp.end()){

index++;

}

}

df.insert({word,index});

return index;

}

// du doan van ban

int search\_engine(vector<string> list\_word){

double score\_max = -1000;

int predict\_label = -1;

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

vector<string> list\_word\_train\_doc = vector\_train[i];

double score = 0;

for(string word : list\_word){

if(find(list\_word\_train\_doc.begin(), list\_word\_train\_doc.end(), word) == list\_word\_train\_doc.end()){ // tu nay khong xuat hien trong van ban

continue;

} else {

int ftd = frequence\_word\_int\_document\_i(word, i);

int dft = count\_document\_contain\_word(word);

int maxfd = f\_max[i];

double tf\_word = 0.5 + 0.5 \* ((double) ftd / maxfd);

double idf\_word = log2((double) n / dft);

score += tf\_word \* idf\_word;

}

}

if(score > score\_max) {

predict\_label = i;

score\_max = score;

}

}

return predict\_label + 1;

}

int main(){

ios\_base::sync\_with\_stdio(false); cin.tie(NULL);

input();

pre\_processing();

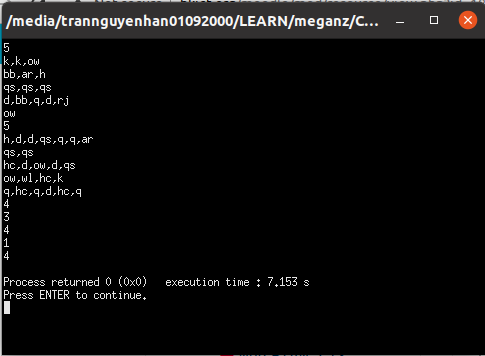
for(int i=0; i<q; i++)

cout << search\_engine(vector\_test[i]) << endl;

return 0;

}

Kết quả:



Hình : Máy tìm kiếm đơn giản

Bài tập 11: Bảo vệ lâu đài  
Bức tường bao quanh 1 lâu đài nọ được cấu thành từ n đoạn tường được đánh số từ 1 đến n. Quân địch 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 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 dược đoạn tường này. Yêu cầu hãy viết chương trình phân bố lính canh ở các đoạn tường sao cho số tên địch lọt vào là ít nhất.

**Tên file: 20183554-NguyenQuangHuy\_Bai 4\_11.cpp**

#include<bits/stdc++.h>

using namespace std;

struct data {

int ai;

int ki;

data(int ai, int ki){

this->ai = ai;

this->ki = ki;

}

};

struct compare{

bool operator() (data a, data b){

int ra, rb;

if(a.ai <= a.ki) ra = a.ai;

else ra = a.ki;

if(b.ai <= b.ki) rb = b.ai;

else rb = b.ki;

return ra < rb;

}

};

int n, s, kill\_enemy, total\_enemy;

priority\_queue<data, vector<data>, compare> p\_q;

void input(){

cin >> n >> s;

kill\_enemy = 0;

total\_enemy = 0;

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

int tmp1, tmp2;

cin >> tmp1 >> tmp2;

total\_enemy += tmp1;

p\_q.push(data(tmp1, tmp2));

}

}

void solve(){

while(!p\_q.empty() && s>0){

data inf = p\_q.top(); p\_q.pop();

if(inf.ai <= inf.ki){

kill\_enemy += inf.ai;

} else {

int nenemy = inf.ai - inf.ki;

p\_q.push(data(nenemy, inf.ki));

kill\_enemy += inf.ki;

}

s -= 1;

}

}

int calc\_enemy(){

/\*

int sum = 0;

while(!p\_q.empty()){

data v = p\_q.top(); p\_q.pop();

sum += v.ai;

}

return sum;

\*/

return total\_enemy - kill\_enemy;

}

int main(){

input();

solve();

cout << calc\_enemy();

/\*

while(!p\_q.empty()){

auto v = p\_q.top(); p\_q.pop();

cout << v.ai << " " << v.ki << endl;

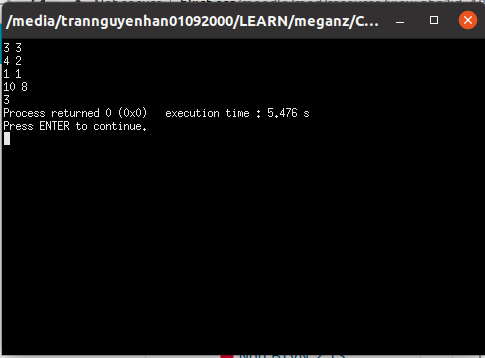
}

\*/

return 0;

}

Kết quả :



Hình : Bảo vệ lâu đài

Bài tập 12: Lược đồ  
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

**Tên file: 20183554-NguyenQuangHuy\_Bai 4\_12.cpp**

#include<bits/stdc++.h>

using namespace std;

long long n;

vector<long long> vt;

vector<long long> L; // mang trai luu tru gia tri ben trai gan nhat nho hon gia tri dang xet

vector<long long> R; // mang phai luu tru gia tri ben phai gan nhat nho hon gia tri dang xet

void inputData(){

cin >> n;

vt.push\_back(-1); // Dat linh canh o 2 dau mang

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

long long tmp;

cin >> tmp;

vt.push\_back(tmp);

}

vt.push\_back(-1); // Dat linh canh o 2 dau mang

}

void solve(){

while(true){

inputData();

if(n==0) return;

// Xu ly o day

stack<long long> st; // 1 stack de thuc hien thuat toan ghi ra cac gia tri L, R

L.resize(n+2);

R.resize(n+2);

for(long long i=1; i<=n+1; i++){

while(!st.empty() && vt[i] < vt[st.top()]){

L[st.top()] = i;

st.pop();

}

st.push(i);

}

for(long long i=n; i>=0; i--){

while(!st.empty() && vt[i] < vt[st.top()]){

R[st.top()] = i;

st.pop();

}

st.push(i);

}

long long maxh = 0;

for(long long i=1; i<=n; i++){

long long h = (L[i] - R[i] -1)\*vt[i];

if(h > maxh) maxh = h;

}

cout << maxh << endl; break;

// Xoa mang truoc khi bat dau vong lap moi ( mang vt, mang L, mang R)

vt.erase(vt.begin(),vt.end());

L.erase(L.begin(),L.end());

R.erase(R.begin(),R.end());

}

}

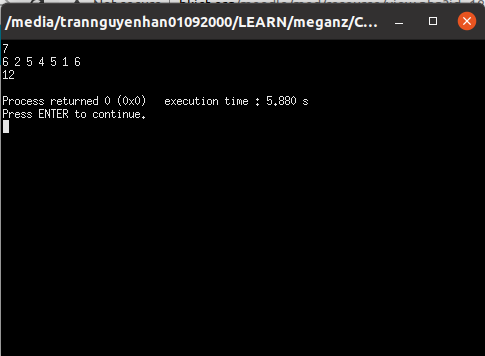
int main(){

solve();

return 0;

}

Kết quả :



Hình : Hình chữ nhật có diện tích lớn nhất

Bài tập 13: Đếm xâu con  
Cho 1 xâu nhị phân độ dài n. Hãy viết chương trình đếm số lượng xâu con chứa số kí tự 0 và 1 bằng nhau

**Tên file: 20183554-NguyenQuangHuy\_Bai 4\_13.cpp**

#include<bits/stdc++.h>

using namespace std;

const int MAX = 1000000;

string str;

int n, cnt;

int slg0[MAX]; // slg0[i] la dem so luong so 0 trong khoang 0 toi i

int slg1[MAX]; // slg0[i] la dem so luong so 1 trong khoang 0 toi i

void input(){

cnt = 0;

cin >> str;

n = str.length();

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

if(i==0){

if(str[i] == '0'){

slg0[i] = 1;

slg1[i] = 0;

} else {

slg0[i] = 0;

slg1[i] = 1;

}

continue;

}

if(str[i] == '0'){

slg0[i] = slg0[i-1] + 1;

slg1[i] = slg1[i-1];

} else {

slg0[i] = slg0[i-1];

slg1[i] = slg1[i-1] + 1;

}

}

}

void solve(){

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

for(int j=i+1; j<n; j = j + 2){

if(i==0){

if(slg0[j] == slg1[j]) cnt++;

} else {

int c0 = slg0[j] - slg0[i-1];

int c1 = slg1[j] - slg1[i-1];

if(c0 == c1) cnt++;

}

}

}

}

int main(){

ios\_base::sync\_with\_stdio(false); cin.tie(NULL);

input();

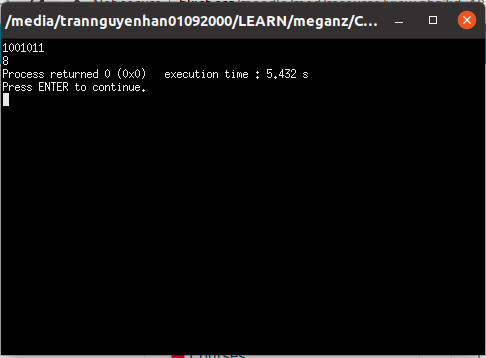
solve();

cout << cnt;

return 0;

}

Kết quả :



Hình : Đếm xâu con