Mục lục nội dung

[Bài thực hành số 04 – Tuần 38 2](#_Toc168079354)

[BÀI TẬP TRÊN LAP 2](#_Toc168079355)

[**Bài tập 4.1.**Đảo ngược một danh sách liên kết đơn 2](#_Toc168079356)

[**Bài tập 4.2.** Tính diện tích tam giác 5](#_Toc168079357)

[**Bài tập 4.3.** Tính tích có hướng hai vector. 7](#_Toc168079358)

[**Bài tập 4.4.** Thao tác với vector. 7](#_Toc168079359)

[**Bài tập 3.5.** DFS không đệ quy 11](#_Toc168079360)

[**Bài tập 4.6.** BFS không đệ quy. 15](#_Toc168079361)

[**Bài tập 4.7.** Giao và hợp hai tập hợp (set) 17](#_Toc168079362)

[**Bài tập 4.8.** Giao và hợp hai tập hợp (map) 19](#_Toc168079363)

[**Bài tập 4.9.** Dijkstra trên đồ thị vô hướng. 22](#_Toc168079364)

[**BÀI TẬP VỀ NHÀ** 25](#_Toc168079365)

[**Bài tập 4.10.** Search Engine 25](#_Toc168079366)

[**Bài tập 4.11.** Bảo vệ lâu đài 28](#_Toc168079367)

[**Bài tập 4.12.** Lược đồ 35](#_Toc168079368)

[**Bài tập 4.13.** Đếm xâu con 39](#_Toc168079369)

Mục lục hình ảnh

[Hình 1 Bài 4.1 3](#_Toc168083556)

[Hình 2 Bài 4.2 6](#_Toc168083557)

[Hình 3 Bài 4.3 7](#_Toc168083558)

[Hình 4 Bài 4.4 8](#_Toc168083559)

[Hình 5 Bài 4.5 13](#_Toc168083560)

[Hình 6 Bài 4.6 16](#_Toc168083561)

[Hình 7 Bài 4.7 18](#_Toc168083562)

[Hình 8 Bài 4.8 20](#_Toc168083563)

[Hình 9 Bài 4.9 23](#_Toc168083564)

[Hình 10 Bài 4.10 26](#_Toc168083565)

[Hình 11 Bài 4.11 33](#_Toc168083566)

[Hình 12 Bài 4.12 38](#_Toc168083567)

[Hình 13 Bài 4.13 44](#_Toc168083568)

# Bài thực hành số 04 – Tuần 38

# BÀI TẬP TRÊN LAP

## **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))

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Hình 1 Bài 4.1

|  |
| --- |
| #include <iostream>  using namespace std;  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Ho va ten: Nguyen Khong Duy Hoang  MSSV : 20225130  \*/  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 #  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  if ( head == NULL ){  head = new Node(data);  return head;  }  else{  Node \*newNode = new Node(data);  newNode->next = head;  return newNode;  }  }  // print the list content on a line  void print(Node\* head)  {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  # YOUR CODE HERE #  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  Node \*tmp = head;  while ( tmp!=NULL ){  cout << tmp->data << ' ';  tmp = tmp->next;  }  cout << endl;  }  // return the new head of the reversed list  Node\* reverse(Node\* head)  {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  # YOUR CODE HERE #  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  if ( head == NULL )  return NULL;  if ( head->next == NULL )  return head;  Node \*oldN = NULL;  Node \*crrN = head;  while ( crrN != NULL ){  Node \*tmp = crrN->next;  crrN->next = oldN;  oldN = crrN;  crrN = tmp;  }  return oldN;  }  int main()  {  #define taskname "hello123"  if ( fopen(taskname".inp","r" ) ){  freopen(taskname".inp","r",stdin);  freopen(taskname".out","w",stdout);  }  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;  } |

## **Bài tập 4.2.** 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>;

A screen shot of a computer code

Description automatically generated

A screenshot of a computer code

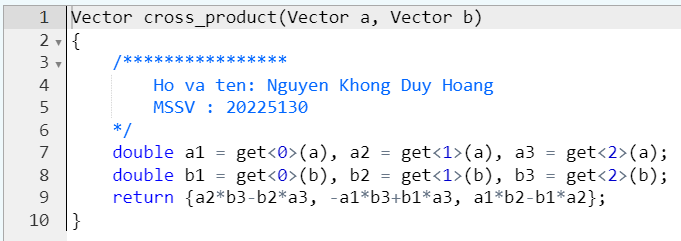
Description automatically generated

Hình 2 Bài 4.2

|  |
| --- |
| double area(Point a, Point b, Point c)  {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Ho va ten: Nguyen Khong Duy Hoang  MSSV : 20225130  \*/  Point ab = {b.first - a.first, b.second-a.second};  Point ac = {c.first - a.first, c.second-a.second};  return (double) (0.5\*abs(ab.first\*ac.second-ab.second\*ac.first) );  } |

## **Bài tập 4.3.** Tính tích có hướng hai vector.

Một vector trong không gian 3 chiều được biểu diễn bằng tuple. 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;



A screenshot of a computer program

Description automatically generated

Hình 3 Bài 4.3

|  |
| --- |
| Vector cross\_product(Vector a, Vector b)  {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Ho va ten: Nguyen Khong Duy Hoang  MSSV : 20225130  \*/  double a1 = get<0>(a), a2 = get<1>(a), a3 = get<2>(a);  double b1 = get<0>(b), b2 = get<1>(b), b3 = get<2>(b);  return {a2\*b3-b2\*a3, -a1\*b3+b1\*a3, a1\*b2-b1\*a2};  } |

## **Bài tập 4.4.** Thao tác với vector.

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

A screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generated

Hình 4 Bài 4.4

|  |
| --- |
| #include <iostream>  #include <vector>  #include <algorithm>  using namespace std;  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Ho va ten: Nguyen Khong Duy Hoang  MSSV : 20225130  \*/  bool cmp(int a, int b){  return a>b;  }  bool isEven(int x) { return x%2==0;}  void print\_vector(vector<int> &a)  {  for (int v : a) cout << v << ' ';  cout << endl;  }  void delete\_even(vector<int> &a)  {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  # YOUR CODE HERE #  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  for (vector<int>::iterator it = a.begin(); it != a.end(); ) {  if (\*it % 2 == 0) {  it = a.erase(it);  } else {  ++it;  }  }  }  void sort\_decrease(vector<int> &a)  {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  # YOUR CODE HERE #  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  sort(a.begin(), a.end(), cmp);  }  vector<int> merge\_vectors(const vector<int> &a, const vector<int> &b)  {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  # YOUR CODE HERE #  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  vector<int> ans;  for ( auto v : a )  ans.push\_back(v);  for ( auto v : b )  ans.push\_back(v);  sort(ans.begin(),ans.end(),cmp);  return ans;  }  int main()  {  #define taskname "hello123"  if ( fopen(taskname".inp","r" ) ){  freopen(taskname".inp","r",stdin);  freopen(taskname".out","w",stdout);  }  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;  } |

## **Bài tập 3.5.** DFS không đệ quy

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).

A screenshot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

Hình 5 Bài 4.5

|  |
| --- |
| void dfs(vector< list<int> > adj)  {  stack<int> S;  vector<bool> visited(adj.size());  S.push(1); // Bat dau tu dinh so 1  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Ho va ten: Nguyen Khong Duy Hoang  MSSV : 20225130  \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  # YOUR CODE HERE #  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  while ( !S.empty() ){  int top = S.top();  if ( !visited[top] ){  visited[top] = true;  cout << top << '\n';  }  if ( !adj[top].empty() ){  int nextTop = adj[top].front();  adj[top].pop\_front();  if ( !visited[nextTop] )  S.push(nextTop);  } else { S.pop(); }  }  } |

## **Bài tập 4.6.** BFS không đệ quy.

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).

A screenshot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

Hình 6 Bài 4.6

|  |
| --- |
| void bfs(vector< list<int> > adj)  {  queue<int> Q;  vector<bool> visited(adj.size());  Q.push(1); // Bat dau tu dinh 1  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Ho va ten: Nguyen Khong Duy Hoang  MSSV : 20225130  \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  # YOUR CODE HERE #  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  while ( !Q.empty() ){  int t = Q.front();  if ( !visited[t] ){  visited[t] = true;  cout << t << '\n';  }  Q.pop();  for ( auto v : adj[t] ){  if ( !visited[v] ){  //visited[v] = true;  Q.push(v);  }  }  }  } |

## **Bài tập 4.7.** Giao và hợp hai tập hợp (set)

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

A screenshot of a computer

Description automatically generated

Hình 7 Bài 4.7

|  |
| --- |
| template<class T>  set<T> set\_union(const set<T> &a, const set<T> &b)  {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Ho va ten: Nguyen Khong Duy Hoang  MSSV : 20225130  \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  # YOUR CODE HERE #  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  set<T> ans;  for ( auto x : a )  ans.insert(x);  for( auto x : b )  ans.insert(x);  return ans;  }  template<class T>  set<T> set\_intersection(const set<T> &a, const set<T> &b)  {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Ho va ten: Nguyen Khong Duy Hoang  MSSV : 20225130  \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  # YOUR CODE HERE #  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  set<T> ans;  for ( auto x : a ){  int check = 0;  for ( auto y : b ){  if ( x==y ){  check=1;  break;  }  }  if ( check == 1 )  ans.insert(x);  }  return ans;  } |

## **Bài tập 4.8.** Giao và hợp hai tập hợp (map)

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.

A screenshot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

Hình 8 Bài 4.8

|  |
| --- |
| template<class T>  map<T, double> fuzzy\_set\_union(const map<T, double> &a, const map<T, double> &b)  {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  # YOUR CODE HERE #  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Ho va ten: Nguyen Khong Duy Hoang  MSSV : 20225130  \*/  map<T, double> ans;  for ( auto x : a ){  int cnt = 0;  for ( auto y : b ){  if ( x.first == y.first ){  ans.insert({x.first, max(x.second, y.second)});  cnt++;  break;  }  }  if ( cnt==0 )  ans.insert(x);  }  for ( auto x : b ){  int cnt = 0;  for ( auto y : a ){  if ( x.first == y.first ){  ans.insert({x.first, max(x.second, y.second)});  cnt++;  break;  }  }  if ( cnt==0 )  ans.insert(x);  }  return ans;  }  template<class T>  map<T, double> fuzzy\_set\_intersection(const map<T, double> &a, const map<T, double > &b)  {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  # YOUR CODE HERE #  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Ho va ten: Nguyen Khong Duy Hoang  MSSV : 20225130  \*/  map<T, double> ans;  for ( auto x : a ){  for ( auto y : b ){  if ( x.first == y.first ){  ans.insert({x.first, min(x.second, y.second)});  break;  }  }  }  return ans;  } |

## **Bài tập 4.9.** Dijkstra trên đồ thị vô hướng.

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.

A screenshot of a computer code

Description automatically generated A screenshot of a computer

Description automatically generated

Hình 9 Bài 4.9

|  |
| --- |
| #define pII pair<int, int>  vector<int> dijkstra(const vector< vector< pair<int, int> > >&adj)  {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Ho va ten: Nguyen Khong Duy Hoang  MSSV : 20225130  \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  # YOUR CODE HERE #  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  vector<int> ans(adj.size());  int l = ans.size();  for ( int i=0 ; i<l ; i++ )  ans[i] = INT\_MAX;  ans[0] = 0;  priority\_queue<pII, vector<pII>, greater<pII>> Q;  for ( int i=0 ; i<l ; i++ )  Q.push({i,ans[i]});  while ( !Q.empty() )  {  /// Chon ra dinh tu 0 nho nhat  pII top = Q.top();  Q.pop();  int u = top.first;  int kc = top.second;  if ( kc > ans[u] ) continue;  for ( auto it : adj[u] )  {  int v = it.first;  int w = it.second;  if ( ans[v] > ans[u]+w )  {  ans[v] = ans[u]+w;  Q.push({v, ans[v]});  }  }  }  return ans;  } |

# **BÀI TẬP VỀ NHÀ**

Chụp ảnh kết quả của tất cả các test.

## **Bài tập 4.10.** Search Engine

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Hình 10 Bài 4.10

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Ho va ten: Nguyen Khong Duy Hoang  MSSV : 20225130  \*/  #define msi map<string, int>  int n, q;  string s,t;  /// So lan xuat hien cua tu t trong van ban thu d la f[d][t]  vector<msi> f;  /// maxf[d] la so lan xuat hien nhieu nhat cua f[d][t], ans luu ket qua  vector<int> maxf, ans;  /// df[t] la so van ban chua tu t  msi df;  /// score(t, d) = TF(t, d) \* IDF(t)  double scoreCalculate(string t, int d){  if(f[d].count(t) == 0)  return 0;/// neu khong tim thay tu t trong van ban d thi diem = 0  double tf = 0.5 + 0.5 \* (double)f[d][t]/maxf[d];  double idf = log2((double)n / df[t]);  return tf \* idf;  }  int main(){  #define taskname "hello123"  if ( fopen(taskname".inp","r" ) ){  freopen(taskname".inp","r",stdin);  freopen(taskname".out","w",stdout);  }  ios\_base::sync\_with\_stdio(false);  cin.tie(NULL);  cout.tie(NULL);  cin >> n;  f.resize(n);  maxf.resize(n);  fill(maxf.begin(), maxf.end(), 0);/// cac phan tu trong maxf ban dau = 0  cin.ignore();  for(int i = 0; i < n; i++){  getline(cin, s);  istringstream str(s);  set<string> check;  while(getline(str, t, ',')){  f[i][t]++;  maxf[i] = max(maxf[i], f[i][t]);  /// check luu lai chuoi t de kiem tra  if(check.find(t) == check.end()){  check.insert(t);  df[t]++;  }  }  }  cin >> q;  cin.ignore();  for(int i = 0; i < q; i++){  int res = 0;  getline(cin, s);  istringstream str(s);  vector<double> score(n, 0);  while(getline(str, t, ','))  for(int j = 0; j < n; j++)  score[j] += scoreCalculate(t, j);/// tinh diem cua t voi van ban j dang xet  for ( int k = 0; k < n; k++)  if( score[k] > score[res] )  res = k;  cout << res+1 << endl;  }  return 0;  } |

## **Bài tập 4.11.** Bảo vệ lâu đài

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 n và s (1≤n≤100000;1≤s≤109).

n dòng tiếp theo chứa hai số nguyên ai và ki lần lượt là số tên giặc tấn công đoạn tường thứ i và khả năng chống trả của một lính ở đoạn tường thứ i (1≤ai,ki≤109).

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generated

Hình 11 Bài 4.11

|  |
| --- |
| #include <iostream>  #include <queue>  using namespace std;  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Ho va ten: Nguyen Khong Duy Hoang  MSSV : 20225130  \*/  #define pII pair<int, int>  int n,linh;  // dinh nghia mot functor cho ham so sanh  struct cmp {  bool operator()(pII a, pII b) {  return min(a.first, a.second) < min(b.first, b.second);  }  };  priority\_queue<pII, vector<pII>, cmp> Q;  void duyet(priority\_queue<pII, vector<pII>, cmp> Q){  while ( !Q.empty() ){  pII tmp = Q.top(); Q.pop();  cout << tmp.first << ' ' << tmp.second << endl;  }  }  int killcnt = 0;  void solve(){  while ( !Q.empty() && linh !=0 ){  pII tmp = Q.top(); Q.pop();  if ( tmp.first <= tmp.second )  killcnt += tmp.first;  else{  killcnt += tmp.second;  Q.push({tmp.first-tmp.second,tmp.second});  }  linh--;  }  }  int main()  {  #define taskname "hello123"  if ( fopen(taskname".inp","r" ) ){  freopen(taskname".inp","r",stdin);  freopen(taskname".out","w",stdout);  }  ios\_base::sync\_with\_stdio(false);  cin.tie(nullptr);  cout.tie(nullptr);  int a,b,total = 0;  cin >> n >> linh;  for ( int i=0 ; i<n ; i++ ){  cin >> a >> b;  total += a;  Q.push({a,b});  }  /// duyet(Q);  solve();  cout << total-killcnt;  return 0;  } |

## **Bài tập 4.12.** 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.

**Dữ liệu vào:**  
Dòng thứ nhất chứa số nguyên dương n (1≤n≤106).  
Dòng thứ hai chứa n số nguyên không âm h1,h2,…,hn cách nhau bởi dấu cách (0≤hi≤109).

**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.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated A screenshot of a computer program

Description automatically generated A screenshot of a computer program

Description automatically generated

Hình 12 Bài 4.12

|  |
| --- |
| #include <iostream>  #include <stack>  #include <vector>  using namespace std;  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Ho va ten: Nguyen Khong Duy Hoang  MSSV : 20225130  \*/  int main()  {  #define taskname "hello123"  if ( fopen(taskname".inp","r" ) ){  freopen(taskname".inp","r",stdin);  freopen(taskname".out","w",stdout);  }  ios\_base::sync\_with\_stdio(false);  cin.tie(nullptr);  cout.tie(nullptr);  int n;  stack<int> st;  vector<int> h(1000005, 0);  ///vector<int> s(1000005, 0);  cin >> n;  for ( int i=1 ; i<=n ; i++ )  cin >> h[i];  h[n+1] = 0;  /\*\*  for ( int i=1 ; i<=n+1 ; i++ )  cout << h[i] << ' ';  cout << endl;\*/  int maxx = 0;  for ( int i=1; i<=n+1 ; i++ ){  if ( st.empty() || h[st.top()] <= h[i] );  else{  int tmp;  while ( !st.empty() && h[st.top()] > h[i] ){  tmp = st.top();  st.pop();  if ( st.empty() )  tmp = h[tmp]\*(i-1);  else  tmp = h[tmp]\*(i-st.top()-1);  maxx = max(maxx, tmp);  }  }  st.push(i);  }  cout << maxx << endl;  return 0;  } |

## **Bài tập 4.13.** Đếm xâu con

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 n𝑛 (1≤n≤106).

**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.

A screenshot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generated A screenshot of a computer program

Description automatically generated A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generated

Hình 13 Bài 4.13

|  |
| --- |
| #include <iostream>  #include <map>  #include <string>  using namespace std;  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Ho va ten: Nguyen Khong Duy Hoang  MSSV : 20225130  \*/  int cnt(string s){  map<int, int> mp;  int dif = 0;  for ( int i=0 ; i<(int)s.size() ; i++ ){  if ( s[i] == '0' )  dif++;  else if ( s[i] == '1' )  dif--;  mp[dif]++;  }  dif = 0;  for ( auto v : mp ){  dif += v.second\*(v.second-1)/2;  }  return dif+mp[0];  }  int main() {  #define taskname "hello123"  if ( fopen(taskname".inp","r" ) ){  freopen(taskname".inp","r",stdin);  freopen(taskname".out","w",stdout);  }  ios\_base::sync\_with\_stdio(false);  cin.tie(nullptr);  cout.tie(nullptr);  string s;  cin >> s;  cout << cnt(s) << endl;  return 0;  } |