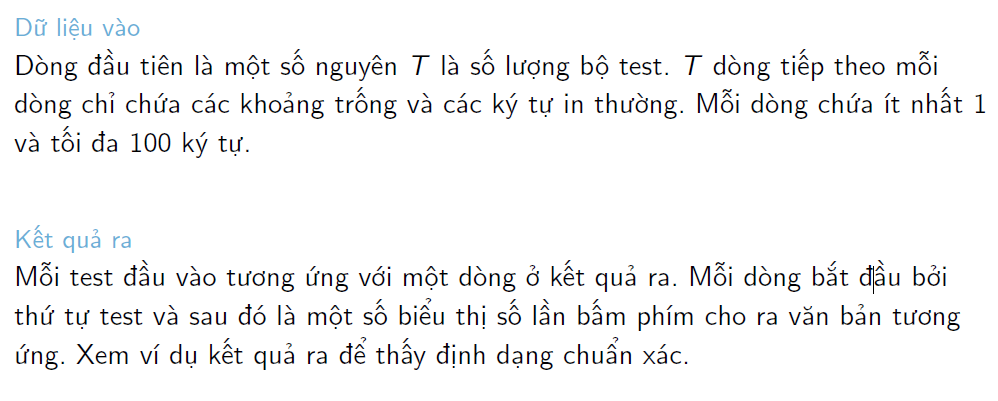
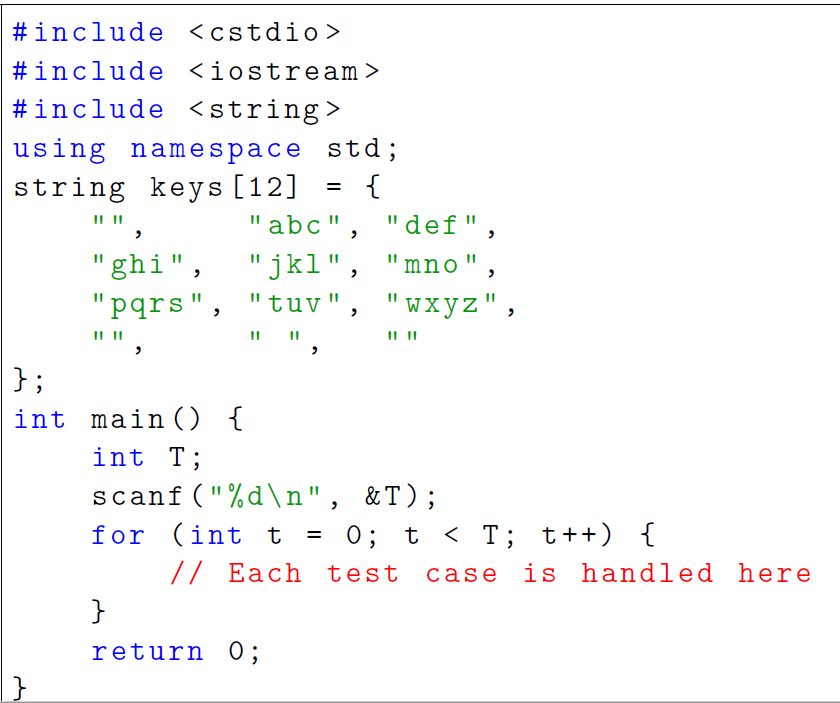
Ôn tập TTUD

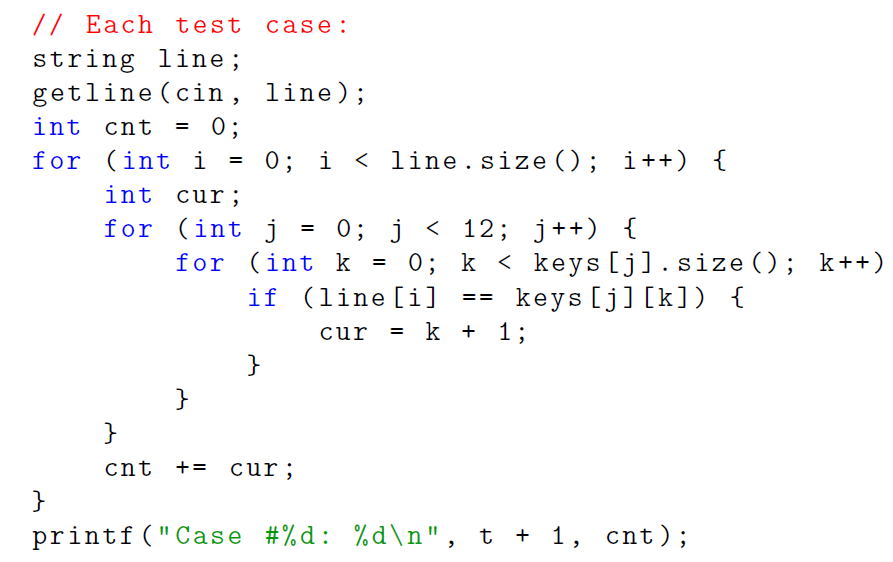
1. Intro

Ước lương xấp xỉ: 2^10 = 10^3

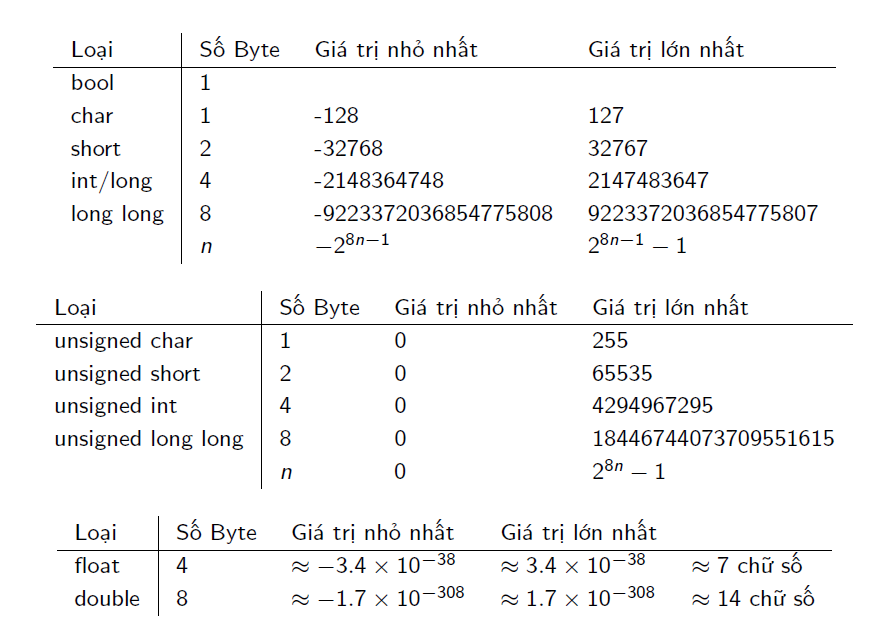
1. Bài toán SMS







1. Cấu trúc dữ liệu
2. Kiểu dữ liệu

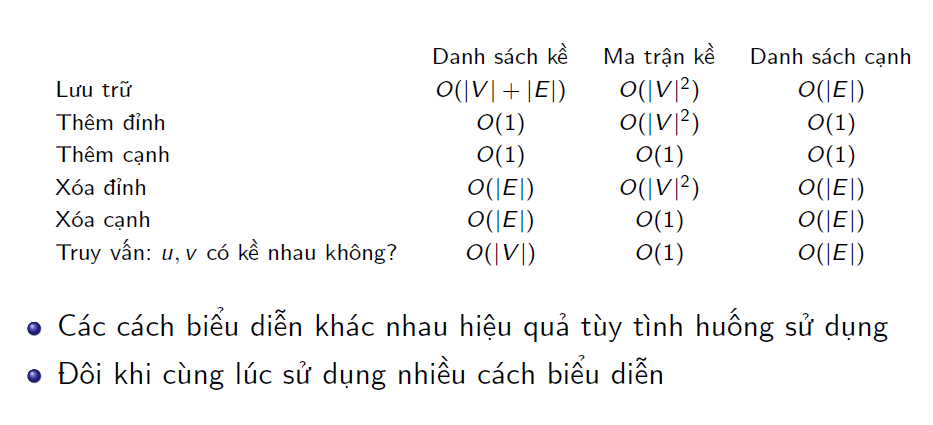


1. Thư viện ctdl và tt: vở
2. Bitmask

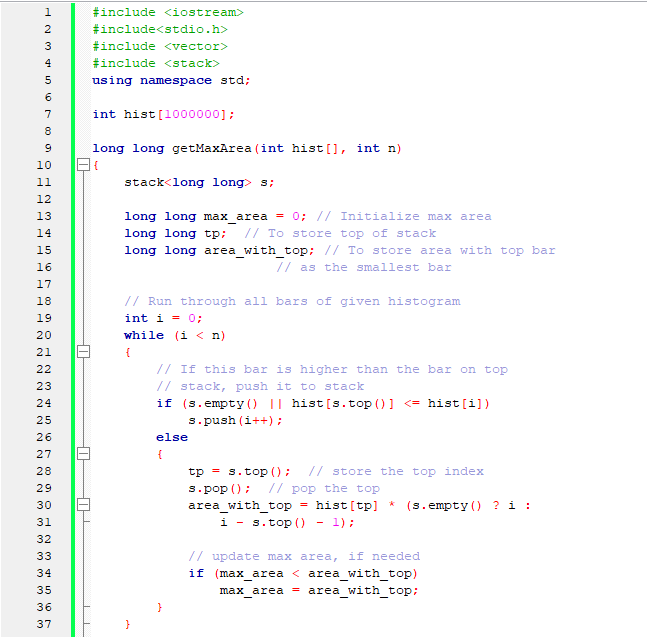
Ktra 1 ptu co trong tập hợp x & (1<<i)

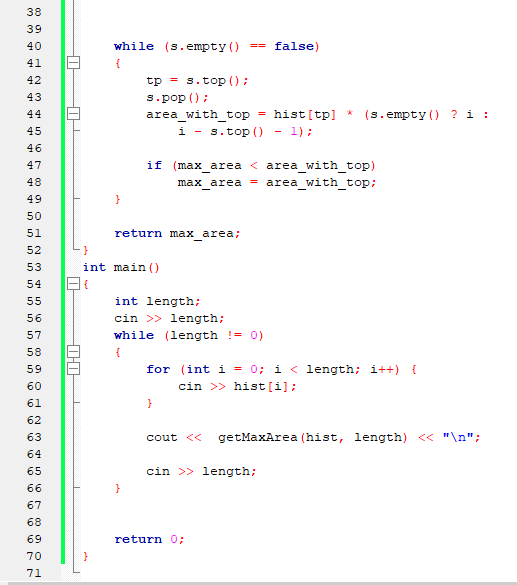
Thêm i=3 vào x||(1<<3)

1. Lí thuyết độ thị ( slide)



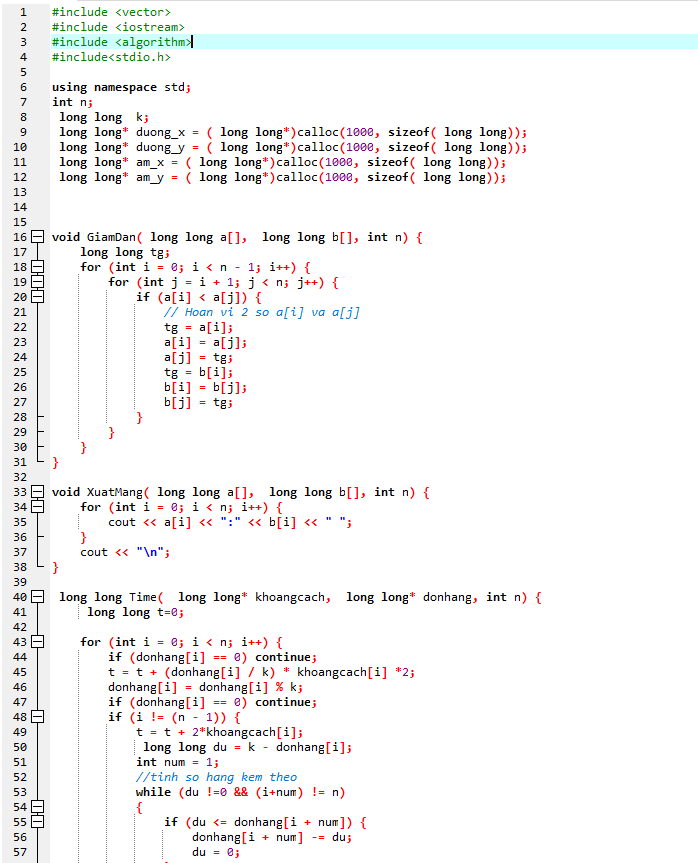
* Các bài tập Code
* A. Hist ( tìm S hình chữ nhật max

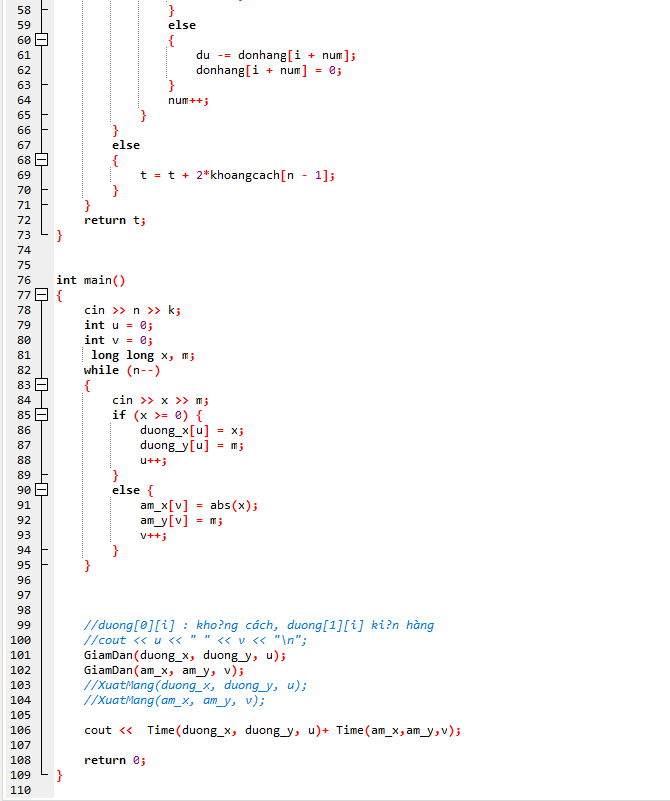




* Postman

Nhân viên đi từ điểm 0 ( mỗi lần đi chỉ đc giao <=k kiện) tới n khách hàng tại n địa điểm, mỗi khách có khoảng cách + số kiện cần giao. Tìm quãng đường giao min





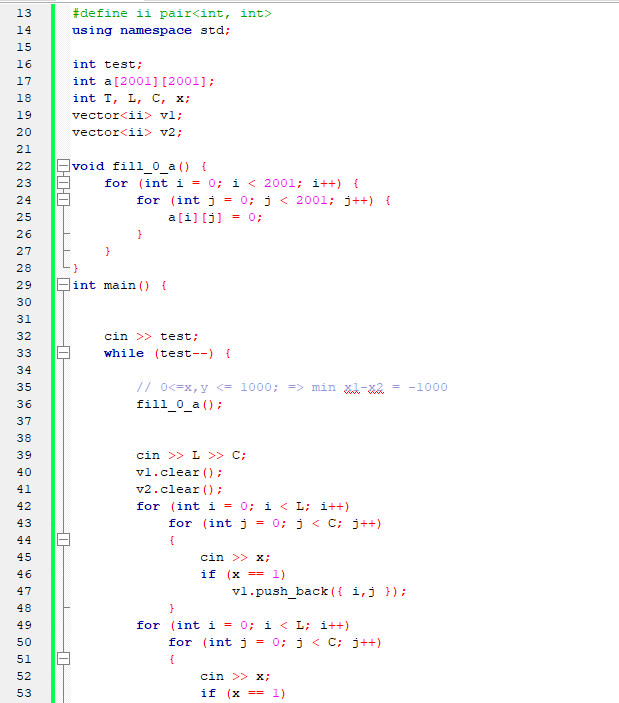
* Locate

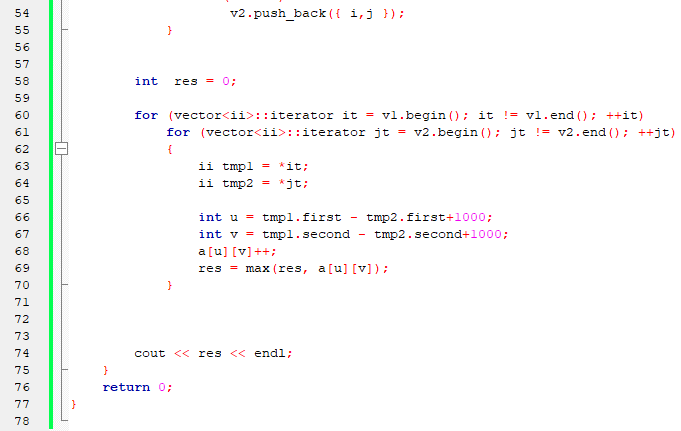
**Input**

Dòng đầu tiên ghi ra một số nguyên TT là số test. Với mỗi test:

Dòng đầu tiên chứa 2 số nguyên LL và CC tương ứng là số dòng và số cột của 2 ảnh (1≤L,C≤10001≤L,C≤1000). 2∗L2∗L dòng tiếp theo mỗi dòng chứa CC số nguyên 0 hoặc 1 cách nhau bởi dấu cách, trong đó LL dòng đầu mô tả ảnh 1, LL dòng sau mô tả ảnh 2. Ở 1 vị trí xác định, 1 tượng trưng cho có chiến đấu cơ hoặc chim ở đó còn 0 tượng trưng cho vị trí đó không có vật thể nào. Biết rằng số lượng số 1 trong một ảnh rada không quá 10000.

2 ma trận bản đồ, Với mỗi test tương ứng, ghi duy nhất một số nguyên NN là số điểm lớn nhất các đối tượng dịch chuyển theo cùng một cách thức.





* WATERJUG BFS

There are two jugs, aa-litres jug and bb-litres jug (a,ba,b are positive integers). There is a pump with unlimited water. Given a positive integer cc, how to get exactly cc litres.

**Input**

Unique line contains positive integers a,b,ca,b,c (1≤a,b,c≤9001≤a,b,c≤900).

**Output**

Line contains the minimal number of steps to get cc litres or -1 if no solution found.

#include<iostream>

#include<algorithm>

using namespace std;

int gcd(int a, int b) {

    if(b == 0)

        return a;

    return gcd(b, a % b);

}

// a: pour from, b: pour to, c: destination

int pour(int a, int b, int c) {

    int from = a;

    int to = 0;

    int step = 1;

    while((from != c) || (to != c)) {

        // find the smaller jug need to fill water

        // fill that water with amount of temp water

        int temp = min(from, b - to);

        to += temp;

        from -= temp;

        step++;

        if((from == c) || (to == c))

            break;

        // case 1: fisrt jug empty

        if(from == 0) {

            from += a;

            step++;

        }

        // case 2: second jug empty

        if(to == b) {

            to = 0;

            step++;

        }

    }

    return step;

}

int steps(int a, int b, int c) {

    if(a > b)

        swap(a, b);

    if(c > b)

        return -1;

    // c is multiple of gcd of a and b

    if(c % gcd(a,b) != 0)

        return -1;

    // find min steps of two cases

    return min(pour(a, b, c), pour(b, a, c));

}

int main() {

    int a, b, c;

    cin >> a;

    cin >> b;

    cin >> c;

    cout << steps(a, b, c);

    return 0;

}

* E. 02.REROAD

Độ gập gềnh: 1,1,0,1,1,1 sẽ có độ gập ghềnh bằng 3

**Input**

Dòng đầu tiên chứa số tự nhiên duy nhất NN là số lượng đoạn đường (1≤1≤N≤105≤105).

Dòng tiếp theo chứa NN số nguyên t1,t2,…,tNt1,t2,…,tN là các loại nhựa đường ban đầu phủ trên các đoạn đường (|ti|≤109|ti|≤109).

Dòng thứ 3 chứa số nguyên duy nhất QQ là số lượng thông báo từ dân chúng về việc sửa chữa mặt đường (1≤Q≤1051≤Q≤105).

Mỗi dòng trong số QQ dòng tiếp theo chứa lần lượt các thông báo. Thông báo thứ ii là cặp hai số nguyên pipi, cici là số thứ tự của đoạn đường được sửa và mã loại nhựa đường mới được phủ lên trên đoạn đường này (1≤pi≤N1≤pi≤N, |ci|≤109|ci|≤109). Đoạn đường được đánh số từ 1 đến NN theo đúng thứ tự ghi loại nhựa đường trong dòng thứ 2 của dữ liệu vào.

**Output**

In ra QQ dòng: dòng thứ ii (1≤i≤Q1≤i≤Q) phải chứa đúng một số nguyên duy nhất là giá trị độ gập ghềnh của đường sau ii thông báo sửa đường đầu tiên.

|  |  |
| --- | --- |
| #include <iostream>  using namespace std;  int a[100000];  /\*d[i] để tính có bao nhiêu độ gập gềnh  - =1 nếu a[i] != a[i-1]  - =0 còn lại  - Từ đó đếm số chỉ d[i] => Độ gập gềnh  - VD : 2 2 2 2 2 2 3 2 2  => 1 0 0 0 (answer=1) 1 0 1 1 0 (answer=3)  \*/  int d(int i) {  if (i == 1) return 1;  return a[i] != a[i - 1];  } | int main()  {  int N;  cin >> N;  for (int i = 1; i <= N; i++) {  cin >> a[i];  }  int answer = 0;  for (int i = 1; i <= N; i++) {  answer = answer + d(i);  }  int Q;  cin >> Q;  int p, c;  while (Q--)  {  cin >> p >> c;  if (p == N) {  answer = answer - d(p);  a[p] = c;  answer = answer + d(p);  }  else {  answer = answer - d(p + 1)-d(p);  a[p] = c;  answer = answer + d(p + 1)+d(p);  }  cout << answer << endl;  }  } |

* F. 02.SIGNAL
* 1 mảng a1,…,an là dữ liệu của 2 đối tượng khi tồn tại vtri I để:max{a1,..,ai−1}−ai≥bmax{a1,..,ai−1}−ai≥b và max{ai+1,..,an}−ai≥b
* Nếu là dl 2 đối tượng => cắt thành 2 dãy
* Vtri cắt (cần tìm )max{a1,..,ai−1}−ai+max{ai+1,..,an}−aimax{a1,..,ai−1}−ai+max{ai+1,..,an}−ai đạt giá trị lớn nhất

**Input**

Dữ liệu đầu vào bao gồm các dòng sau:

* Dòng thứ nhất chứa 2 số nguyên dương nn và bb (3≤n≤200000,1≤b≤503≤n≤200000,1≤b≤50)
* Dòng thứ 2 chứa nn tự nhiên a1,...,ana1,...,an

**Output**

Ghi ra độ đo cắt tín hiệu nếu dãy đầu vào là dữ liệu biểu diễn 2 đối tượng và giá trị -1 nếu ngược lại.

// Train2.cpp : This file contains the 'main' function. Program execution begins and ends there.

//

#include <iostream>

using namespace std;

int max\_prefix[200000];

int max\_suffix[200000];

int input[200000];

int max(int a, int b) {

if (a > b) {

return a;

}

else

{

return b;

}

}

int main()

{

int n, b;

cin >> n >> b;

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

{

cin >> input[i];

}

max\_prefix[0] = input[0];

max\_suffix[n-1] = input[n-1];

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

max\_prefix[i] = max(max\_prefix[i - 1], input[i]);

}

for (int i = n - 2; i >= 0; i--) {

max\_suffix[i] = max(max\_suffix[i + 1], input[i]);

}

int res = -1;

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

int tmp1 = max\_prefix[i - 1] - input[i];

int tmp2 = max\_suffix[i + 1] - input[i];

if (tmp1 >= b && tmp2 >= b) {

res = max(res, tmp1 + tmp2);

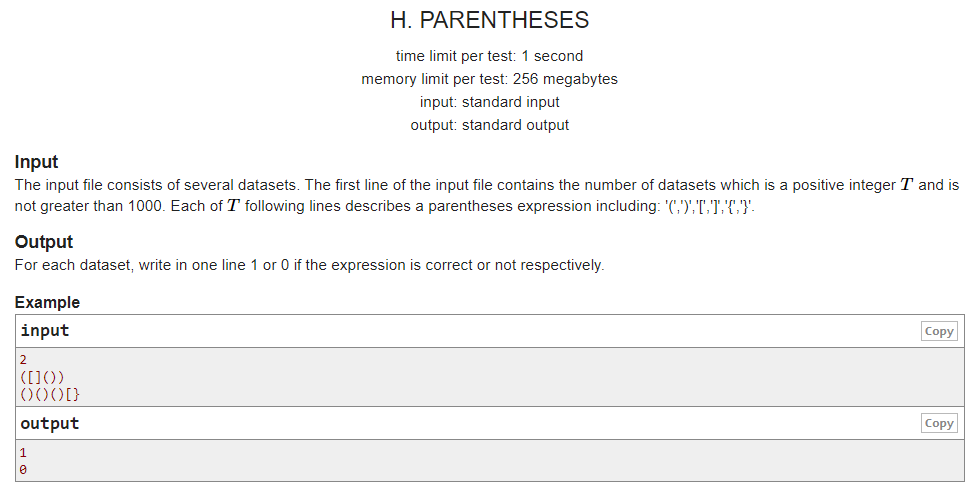
}

}

cout << res;

}

* H. PARENTHESES



#include <iostream>

using namespace std;

#include<stdio.h>

#include <stack>

// CPP program to check for balanced parenthesis.

// function to check if paranthesis are balanced

bool areParanthesisBalanced(string expr)

{

stack<char> s;

char x;

// Traversing the Expression

for (int i = 0; i < expr.length(); i++)

{

if (expr[i] == '(' || expr[i] == '[' || expr[i] == '{')

{

// Push the element in the stack

s.push(expr[i]);

continue;

}

// IF current current character is not opening

// bracket, then it must be closing. So stack

// cannot be empty at this point.

if (s.empty())

return false;

switch (expr[i])

{

case ')':

// Store the top element in a

x = s.top();

s.pop();

if (x == '{' || x == '[')

return false;

break;

case '}':

// Store the top element in b

x = s.top();

s.pop();

if (x == '(' || x == '[')

return false;

break;

case ']':

// Store the top element in c

x = s.top();

s.pop();

if (x == '(' || x == '{')

return false;

break;

}

}

// Check Empty Stack

return (s.empty());

}

// Driver program to test above function

int main()

{

int T;

cin >> T;

string expr;

while (T--)

{

cin >> expr;

if (areParanthesisBalanced(expr))

cout << "1\n";

else

cout << "0\n";

}

return 0;

}

1. Đệ quy và nhánh cận

* Duyệt toàn bộ
* Thuật toán quay lui
* A. 03. TSP
* Đi qua tất cả các thành phố rồi quay về thành phố đầu: Tìm đường đi min

**Input**

Dòng đầu tiên chứa hai số nguyên dương n,mn,m cách nhau bởi dấu cách (n≤20,m<400n≤20,m<400).

mm dòng tiếp theo mỗi dòng chứa ba hai số nguyên dương i,j,ci,j,c (i,j≤n,c≤106i,j≤n,c≤106) biểu thị chi phí đi trực tiếp từ thành phố ii đến thành phố jj là cc.

Lưu ý: nếu từ thành phố ii đến thành phố jj nào không mô tả chi phí đi lại thì có nghĩa là không có đường đi trực tiếp từ ii đến jj.

#include <deque>

#include <iostream>

#include <iterator>

#include <list>

#include <locale>

#include <map>

#include <set>

#include <vector>

using namespace std;

int n, m;

int\*\* c;

int\* x; //luu ds cac dinh di qua theo thu tu

int\* mark;

int  best = INT\_MAX;

int minc = INT\_MAX;

//Cai dat bang de quy nhanh can

void min\_C (){

    for (int i = 0; i <= n; i++) {

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

            if (c[i][j] != 1e7 && c[i][j] < minc) minc = c[i][j];

        }

    }

}

void duyet(int i, int sum) {

    if (i == n) {

        if (sum + c[x[n]][1] < best) best = sum + c[x[n]][1];

        return;

    }

    for (int j = 2; j <= n; j++) {

        if (mark[j] == 0 && c[x[i]][j] != 1e7) {

            if (sum + c[x[i]][j] +minc\*(n-i)<best) {

                mark[j] = 1;

                x[i + 1] = j;

                duyet(i + 1, sum + c[x[i]][j]);

                mark[j] = 0;

            }

        }

    }

}

int main() {

    cin >> n >> m;

    x = (int\*)calloc(n + 1, sizeof(int));

    mark = (int\*)calloc(n + 1, sizeof(int));

    x[1] = 1;

    mark[1] = 1;

    c = (int\*\*)calloc(n + 1, sizeof(int\*));

    for (int i = 0; i < n + 1; i++) c[i] = (int\*)calloc(n + 1, sizeof(int));

    for (int i = 0; i <= n; i++) {

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

            c[i][j]= 1e7;

        }

    }

    while (m--)

    {

        int i, j, k;

        cin >> i >> j >> k;

        c[i][j] = k;

    }

    //thanh pho: 1,2,3...n

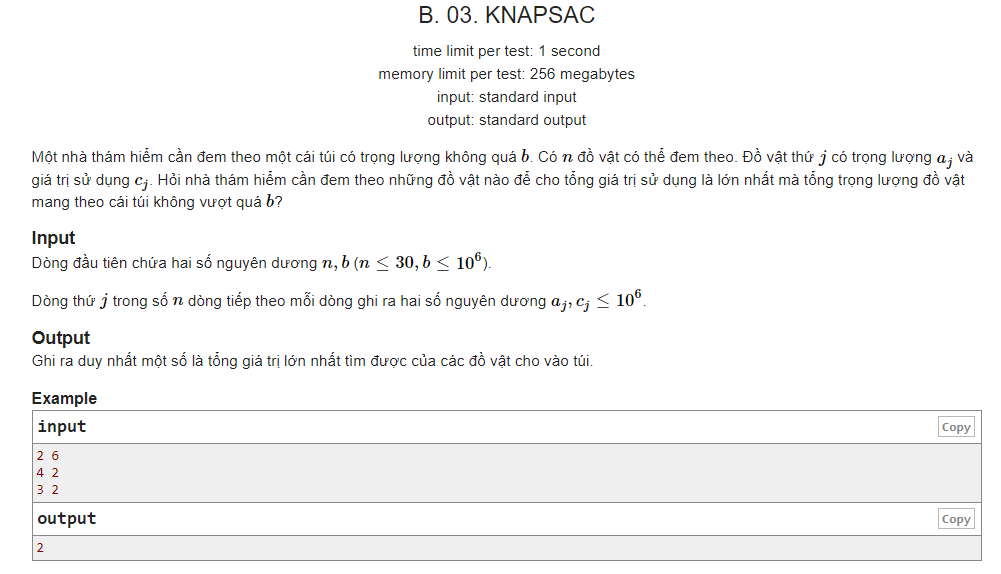
    min\_C();

    duyet(1, 0);

    cout << best;

    return 0;

}



#include <vector>

#include <iostream>

#include <algorithm>

using namespace std;

int n, W;

int\* w;

int\* p;

int\*\* res;

void Fill\_res() {

    //Fill chay dong 1 va 2: res[1]

    //res[0][i] fill 0 res[i][0] fill 0

    //res[1][i]

    for (int i = w[1]; i < W + 1; i++) res[1][i] = p[1];

    //res(i,j) = max{ res(i-1,j), res(i-1,j-w[i]) +p[i])

    // i: 2-> i<n+1

    // j: w[1] -> w< W+1

    for (int i = 2; i < n + 1; i++) {

        for (int j = w[1]; j < W + 1; j++) {

            if (j - w[i] < 0) res[i][j] = res[i - 1][j];

            if (j - w[i] >= 0) {

                res[i][j] = max(res[i - 1][j], res[i - 1][j - w[i]] + p[i]);

            }

        }

    }

}

int main()

{

    cin >> n >> W;

    w = (int\*)calloc(n + 1, sizeof(int));

    p = (int\*)calloc(n + 1, sizeof(int));

    res = (int\*\*)calloc(n+1, sizeof(int\*));

    for (int i = 0; i < n+1; i++) res[i] = (int\*)calloc(W+1, sizeof(int));

    /\*  res:

         0 1 2 3 .... W

w   p  0

       1

       2

      ...

       n

    \*/

    int k = 1;

    int tmp = n;

    while (tmp--)

    {

        cin >> w[k] >> p[k];

        k++;

    }

    //-------------------------------------

    //Sx tang dan w

    for (int i = 0; i < n; i++) {

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

            if (w[i] > w[j]) {

                int tmp = w[i];    int tmp1 = p[i];

                w[i] = w[j];       p[i] = p[j];

                w[j] = tmp;        p[j] = tmp1;

            }

        }

    }

    Fill\_res();

    int kq = 0;

    for (int i = 0; i < n + 1; i++) {

        for (int j = 0; j < W + 1; j++) {

            if (res[i][j] > kq ) kq = res[i][j];

        }

    }

    cout << kq;

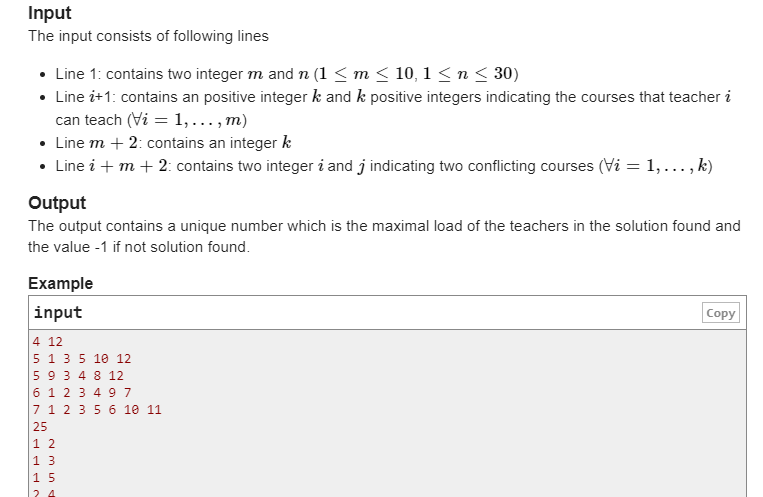
    return 0;

}

* 03. BCA

Có m giáo viên (1,..m) và n khóa học (1,..,n). Mỗi giáo viên có 1 list khóa học và họ có thể dạy + 1 list các khóa học ko thể cùng nhau.

* Tìm cách để giáo viên có tải lớn nhất là nhỏ nhất



#include<stdio.h>

int m,n;

bool Class[11][31] ={0}; // giao vien nao co the day mon nao

bool Congv[11][31]={0}; // giao vien dang day mon gi

bool Subj[31][31]= {0}; //mon hoc trung nhau

int Load[11]={0}; // khoi luong cong viec

int ans =1e7; // Dap an

int a, max;

int min;

int Ktra(int i, int k){ // xem cong viec moi co trung lich cong viec cu ko

    for(int j=1; j<=n; j++){

        if(Congv[i][j]== true && Subj[j][k]== true) return 0;

    }

    return 1;

}

int FMax(){    // tim xem lam nhieu nhat la bao nheiu con viec

    int max=0;

    for(int i=0; i<11; i++){

        if(Load[i]>max) max= Load[i];

    }

    return max;

}

void Try(int k){

    for(int i=1; i<=m; i++){

        if(Class[i][k]== true && Ktra(i, k)){

            Congv[i][k]= true;

            Load[i]++;

            if(k==n){

                max = FMax();

                if(ans>max) ans= max;

            }

            else if(FMax()<ans) Try(k+1);

            Load[i]--;

            Congv[i][k]= 0;

        }

    }

}

main(){

    scanf("%d %d", &m , &n);

    int k, g, j;

    for(int i=1; i<=m; i++){

        scanf("%d", &k);

        for(j=0; j<k; j++){

            scanf("%d", &g);

            Class[i][g]= true;

        }

    }

    scanf("%d", &k);

    for(int i=0; i<k; i++){

        scanf("%d %d", &g, &j);

        Subj[g][j]= true;

        Subj[j][g]= true;

    }

    for(int i=0; i<31; i++){

        Subj[i][i]= true;

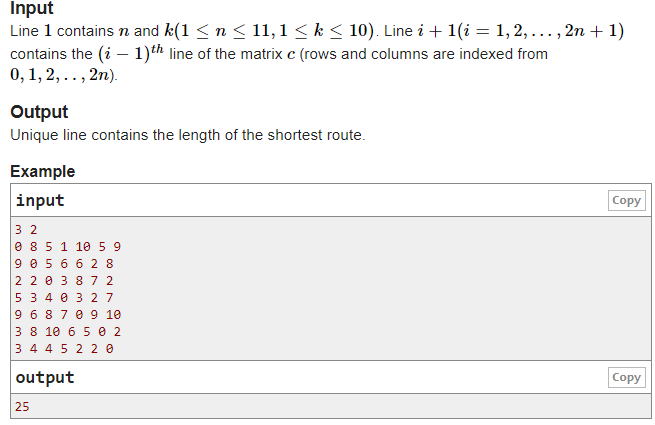
    }

    Try(1);

    printf("%d", ans);

}

* D. 03. CBUS
* Có n hành khách (1,..,n) , hành khách i muốn đi từ điểm I đến i+n
* Xe ở điểm 0 và có k chỗ phục vụ n hành khách và quay về 0
* C(I,j) khoảng cách từ I đến j (0,1..,2n)
* Tìm đường đi ngắn nhất



#include <deque>

#include <iostream>

#include <iterator>

#include <list>

#include <locale>

#include <map>

#include <queue>

#include <set>

#include <vector>

#include <algorithm>

using namespace std;

int n, k, n1, minc; //n1=2\*n+1

int\*\* c;

int save[4194305][11];

bool\* flag = new bool[n1];

void fill\_c() {

    c = (int\*\*)calloc(2 \* n + 1, sizeof(int\*));

    for (int i = 0; i < 2 \* n + 1; i++) c[i] = (int\*)calloc(2 \* n + 1, sizeof(int));

    for (int i = 0; i < 2 \* n + 1; i++) {

        for (int j = 0; j < 2 \* n + 1; j++) {

            cin >> c[i][j];

        }

    }

}

void min\_C() {

    for (int i = 0; i < n1; i++) {

        for (int j = 0; j < n1; j++) {

            if (c[i][j] != 0 && c[i][j] < minc) minc = c[i][j];

        }

    }

}

int fbc(unsigned int data)

{

    data = (data & 0x55555555) + ((data >> 1) & 0x55555555);

    data = (data & 0x33333333) + ((data >> 2) & 0x33333333);

    data = (data & 0x0F0F0F0F) + ((data >> 4) & 0x0F0F0F0F);

    data = (data & 0x00FF00FF) + ((data >> 8) & 0x00FF00FF);

    data = (data & 0x0000FFFF) + ((data >> 16) & 0x0000FFFF);

    return data;

}

/\*

duong di: 0-> 1,2-n...2n ->0

Tap cac dinh X: 0(0) 0(1) 0(2) 0(3) 0(4) 0(5) 0(6)

0 8 5 1 10 5 9

9 0 5 6 6 2 8

2 2 0 3 8 7 2

5 3 4 0 3 2 7

9 6 8 7 0 9 10

3 8 10 6 5 0 2

3 4 4 5 2 2 0

\*/

int tong = 0;

int duyet(int X, int p) {

    int i = fbc(X);

    if (i == n1) return c[p][0];

    if (X < 4194305) {

        if (save[X][p-1] != -1) return save[X][p-1];

    }

    int ans = 2e9;

    for (int s = 1; s < n1; s++) {

        if ((X >> s & 1) == 0 && c[p][s] + minc \* (n1 - i) < ans) {

            if (s <= n) {

                if (tong + 1 > k) continue;

                flag[s] = true;

                tong++;

            }

            else {

                if (flag[s - n] == false) continue;

                tong--;

            }

            ans = min(ans, c[p][s] + duyet(1 << s | X, s));

            if (s <= n) {

                tong--;

                flag[s] = false;

            }

            else {

                tong++;

            }

        }

    }

    if (X < 4194305) {

        save[X][p-1] = ans;

    }

    return ans;

}

int main() {

    cin >> n >> k;

    minc = 1e9;

    fill\_c();

    n1 = 2 \* n + 1;

    for (int i = 0; i < n1; i++) flag[i] = false;

    for (int i = 0; i < 4194305; i++) {

        for (int j = 0; j < 11; j++) save[i][j] = -1;

    }

    min\_C();

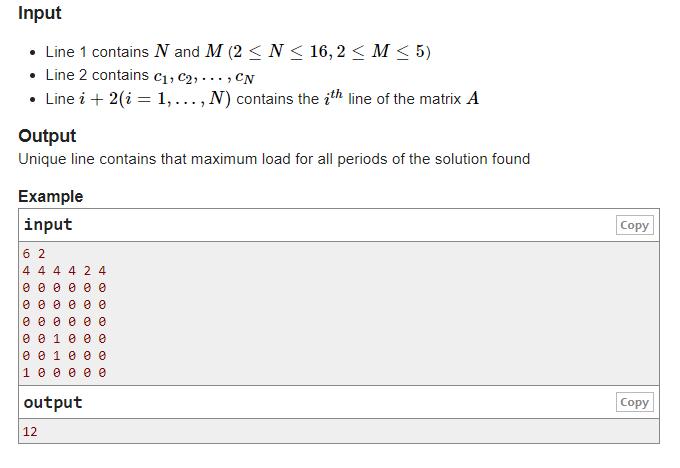
    int res = duyet(1, 0);

    cout << res;

    return 0;

}

* E. 03. BACP
* N khóa học (1,..,N) phải đc đki vào M kì (1,..,M)
* Mối khóa học i có trọng số ci và 1 ma trận Aij =1 nếu môn I phải học trước môn j
* Tìm cách để tải 1 kì lớn nhất là nhỏ nhất



#include<stdio.h>

int m, n, c[17]={0}; // c luu tin chi cua mon hocj i

int clone[17];

int A[17][17]={0}; // xem mon hocj tien quyet

int ans=1e7; // ket qua

bool number[17]={0};

int thutu[6][17] ={0}; // xem ky hoc thi hoc nhung gi

 // ham kiem tra xem mon hocj xuat hien o ky truoc chua

int Test(int i, int j){ // mon hoc j da xuat hien truoc ky hoc i chua

    for(int k=1; k<i; k++){

        if(thutu[k][j]==1) return 0;

    }

    return 1; // 0 la xuat hien roi, 1 la chua xuat hien

}

int tinchi[6] ={0}; // tin chi cho 1ky

int Max(){ // tra ve gia tri tin chi lon nhat

    int a=0;

    for(int i=0; i<6; i++){

        if(tinchi[i]>a) a= tinchi[i];

    }

    return a;

}

int kiemtra(int i, int k){    // i la ky hoc va k la mon hoc thu k

    for(int j=0; j<n; j++){

        if(A[j][k]==1 && Test(i, j)) return 0;

    }

    return 1;

}

void sapxep(){ // sap xep cac mon theo thu tu

    int k =1;

    for(int j=2; j<=m+1; j++){

        for(int i=1; i<=n ; i++){

            if(kiemtra(j,i)==1 && number[i]==0){

                thutu[j][i]=1;

                number[i]=1;

            }

        }

    }

    for(int j=2; j<=m+1; j++){

        for(int i=n; i>=1; i--){

            if(thutu[j][i]==1 ){

                clone[k]=i;

                k++;

        }

    }

}

}

// k la mon hoc

void Try(int k){

    for(int i=2; i<=m+1; i++){

        if(kiemtra(i,clone[k])==1){

            thutu[i][clone[k]]=1;

            tinchi[i]= tinchi[i]+c[clone[k]];

            if(k==n){

                if(Max()<ans) ans =Max();

            }

            else if(Max()<ans) Try(k+1);

            thutu[i][clone[k]]=0;

            tinchi[i]= tinchi[i]-c[clone[k]];

        }

    }

}

int main(){

    scanf("%d %d", &n, &m);

    for(int i=1; i<=n; i++){

        scanf("%d", &(c[i]));

    }

    for(int i=1; i<=n; i++){

        for(int j=1; j<=n; j++){

            scanf("%d", &(A[i][j]));

        }

    }

    sapxep();

    for(int i=0; i<=m; i++){

        for(int j=0; j<=n; j++)

        thutu[i][j]=0;

    }

    Try(1);

    printf("%d", ans);

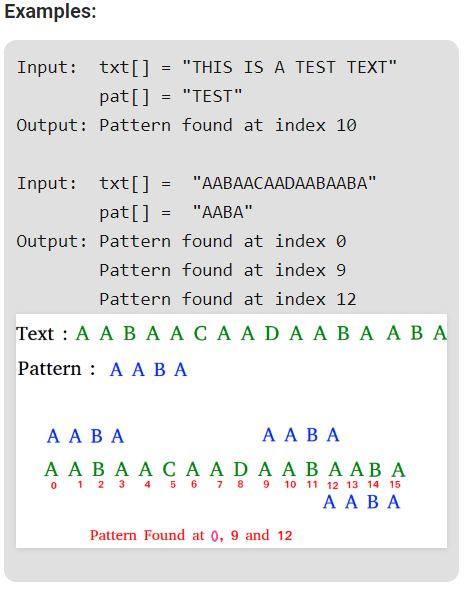
}

1. Chia để chị

* Đếm số lần xâu p xuất hiện trong xâu s

+ đếm từng kí tự -> O(lenp . lens)

+ thuật toán Knuth-Morrí-Fatt (KMP)-> O(lens+lenp): <https://www.geeksforgeeks.org/kmp-algorithm-for-pattern-searching/>



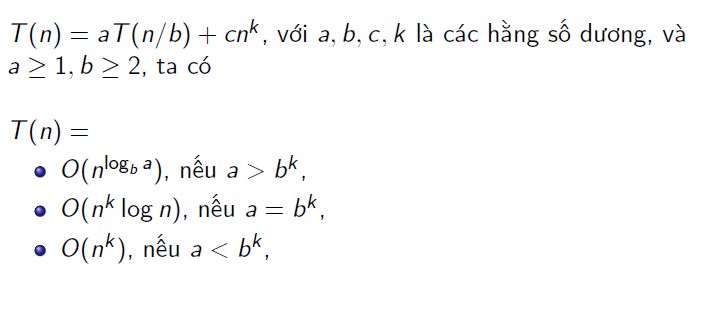
|  |
| --- |
| // C++ program for implementation of KMP pattern searching  // algorithm  #include <bits/stdc++.h>  void computeLPSArray(char\* pat, int M, int\* lps);  // Prints occurrences of txt[] in pat[]  void KMPSearch(char\* pat, char\* txt)  {  int M = strlen(pat);  int N = strlen(txt);  // create lps[] that will hold the longest prefix suffix  // values for pattern  int lps[M];  // Preprocess the pattern (calculate lps[] array)  computeLPSArray(pat, M, lps);  int i = 0; // index for txt[]  int j = 0; // index for pat[]  while (i < N) {  if (pat[j] == txt[i]) {  j++;  i++;  }  if (j == M) {  printf("Found pattern at index %d ", i - j);  j = lps[j - 1];  }  // mismatch after j matches  else if (i < N && pat[j] != txt[i]) {  // Do not match lps[0..lps[j-1]] characters,  // they will match anyway  if (j != 0)  j = lps[j - 1];  else  i = i + 1;  }  }  }  // Fills lps[] for given patttern pat[0..M-1]  void computeLPSArray(char\* pat, int M, int\* lps)  {  // length of the previous longest prefix suffix  int len = 0;  lps[0] = 0; // lps[0] is always 0  // the loop calculates lps[i] for i = 1 to M-1  int i = 1;  while (i < M) {  if (pat[i] == pat[len]) {  len++;  lps[i] = len;  i++;  }  else // (pat[i] != pat[len])  {  // This is tricky. Consider the example.  // AAACAAAA and i = 7. The idea is similar  // to search step.  if (len != 0) {  len = lps[len - 1];  // Also, note that we do not increment  // i here  }  else // if (len == 0)  {  lps[i] = 0;  i++;  }  }  }  }  // Driver program to test above function  int main()  {  char txt[] = "ABABDABACDABABCABAB";  char pat[] = "ABABCABAB";  KMPSearch(pat, txt);  return 0;  } |

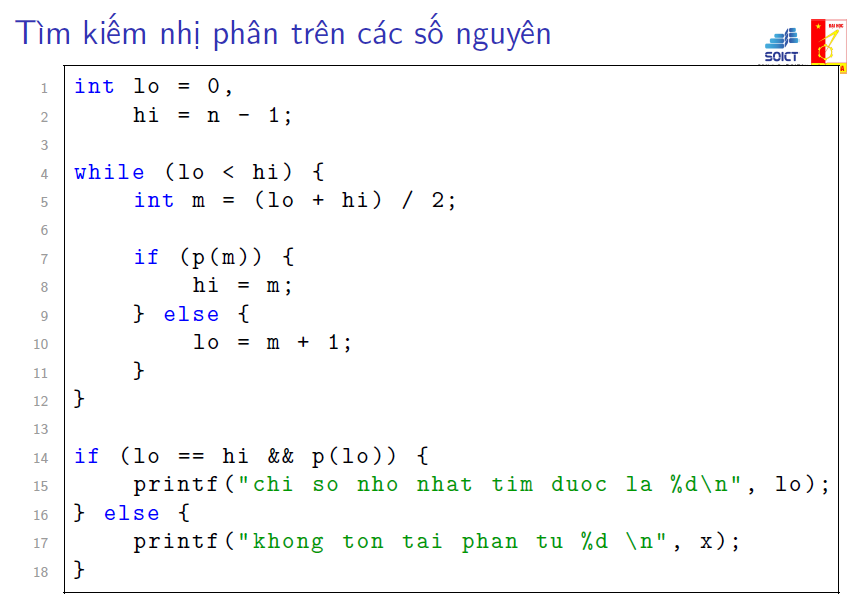
Một số bài toán chia để trị cơ bản

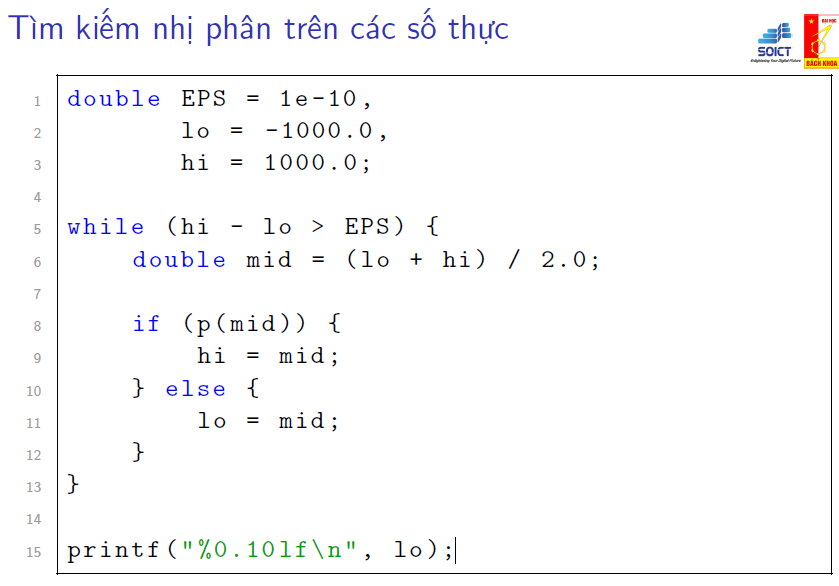
* SX nhanh, sx trộn
* Karatsuba nhân nhanh số lớn: <https://www.geeksforgeeks.org/karatsuba-algorithm-for-fast-multiplication-using-divide-and-conquer-algorithm/> ( đổi só về dạng bit )

|  |
| --- |
| // C++ program to multiply two numbers represented  // as strings.  //Nhan theo kieu String  #include<bits/stdc++.h>  using namespace std;  // Multiplies str1 and str2, and prints result.  string multiply(string num1, string num2)  {  int len1 = num1.size();  int len2 = num2.size();  if (len1 == 0 || len2 == 0)  return "0";  // will keep the result number in vector  // in reverse order  vector<int> result(len1 + len2, 0);  // Below two indexes are used to find positions  // in result.  int i\_n1 = 0;  int i\_n2 = 0;    // Go from right to left in num1  for (int i=len1-1; i>=0; i--)  {  int carry = 0;  int n1 = num1[i] - '0';  // To shift position to left after every  // multiplication of a digit in num2  i\_n2 = 0;    // Go from right to left in num2  for (int j=len2-1; j>=0; j--)  {  // Take current digit of second number  int n2 = num2[j] - '0';  // Multiply with current digit of first number  // and add result to previously stored result  // at current position.  int sum = n1\*n2 + result[i\_n1 + i\_n2] + carry;  // Carry for next iteration  carry = sum/10;  // Store result  result[i\_n1 + i\_n2] = sum % 10;  i\_n2++;  }  // store carry in next cell  if (carry > 0)  result[i\_n1 + i\_n2] += carry;  // To shift position to left after every  // multiplication of a digit in num1.  i\_n1++;  }  // ignore '0's from the right  int i = result.size() - 1;  while (i>=0 && result[i] == 0)  i--;  // If all were '0's - means either both or  // one of num1 or num2 were '0'  if (i == -1)  return "0";  // generate the result string  string s = "";    while (i >= 0)  s += std::to\_string(result[i--]);  return s;  }  // Driver code  int main()  {  string str1 = "1235421415454545454545454544";  string str2 = "1714546546546545454544548544544545";    if((str1.at(0) == '-' || str2.at(0) == '-') &&  (str1.at(0) != '-' || str2.at(0) != '-' ))  cout<<"-";  if(str1.at(0) == '-' && str2.at(0)!='-')  {  str1 = str1.substr(1);  }  else if(str1.at(0) != '-' && str2.at(0) == '-')  {  str2 = str2.substr(1);  }  else if(str1.at(0) == '-' && str2.at(0) == '-')  {  str1 = str1.substr(1);  str2 = str2.substr(1);  }  cout << multiply(str1, str2);  return 0;  } |

Độ phức tạp thuật toán: Sd định lí thợ







hàm pow trong tv #include <math.h> cũng sd chia để trị

int pow( int x, int n) {

2 if (n == 0) return 1;

3 if (n % 2 != 0) return x \* pow(x, n - 1);

4 int st = pow(x, n /2);

5 return st \* st;

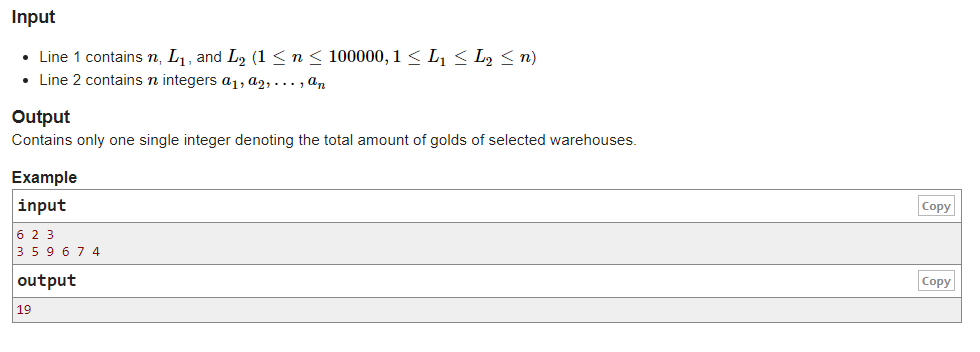
6 }

* Dãy Fibonacci ( số, chuỗi)

1. Tuần 5-6: Quy hoạch động

memset (comp , 0 , sizeof ( comp ));

* A. 05. GOLD MINING
* N ngôi nhà trên vtri 1,..,n. Mỗi nhà có lượng vàng ai
* Tìm ra lượng vàng max sao cho L1<= khoảng cách <=L2



#include <bits/stdc++.h>

//A. 05. GOLD MINING

//Quy hoach vong lap

using namespace std;

#define all(s) s.begin(), s.end()

#define pb push\_back

#define ii pair<int, int>

#define x first

#define y second

#define bit(x, y) ((x >> y) & 1)

int n, L1, L2;

int a[100000];

int mem[100000]; //mem[i]so vang lon nhat ket thuc tại a[i]

//L1<= distance <= L2

int main() {

    ios::sync\_with\_stdio(false); cin.tie(0);

    cout.tie(0);

    cin >> n >> L1 >> L2;

    memset(a, 0, 100000);

    memset(mem, 0, 100000);

    for (int i = 0; i < n; i++) {

        cin >> a[i];

        mem[i] = a[i];

    }

    for (int i = L1; i < n; i++) {

        for (int j = i - L2; j <= i - L1; j++) {

            if (j >= 0) {

                if (mem[i] < mem[j] + a[i]) mem[i] = mem[j] + a[i];

            }

            else {

                continue;

            }

        }

    }

    int res = mem[0];

    for (int i = 0; i < n; i++) {

        res = max(res, mem[i]);

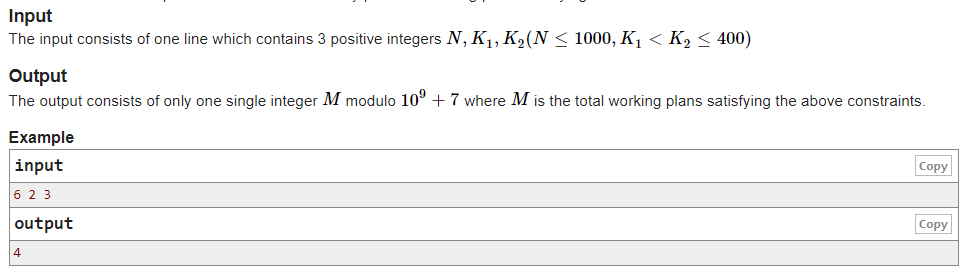
    }

    cout << res;

    return 0;

}

* F. 05. NURSE



#include <bits/stdc++.h>

using namespace std;

typedef unsigned long long ull;

typedef long long ll;

#define forn(i, n) for(int i = 0; i < n; i++)

#define for1(i, n) for(int i = 1; i <= n; i++)

#define max(a, b) ((a) > (b) ? (a) : (b))

#define min(a, b) ((a) < (b) ? (a) : (b))

#define MAXN 1005

int n, k1, k2;

ull ans[MAXN];

int main()

{

    cin >> n >> k1 >> k2;

    ans[0] = ans[1] = 1;

    for (int i = 0; i <= n + 1; i++) {

        if (ans[i] != 0) {

            for (int j = k1 + 1; j <= k2 + 1; j++) {

                if (i + j <= n + 1) {

                    ans[i + j] += ans[i];

                    ans[i + j] = ans[i + j] % 100000007;

                }

            }

        }

    }

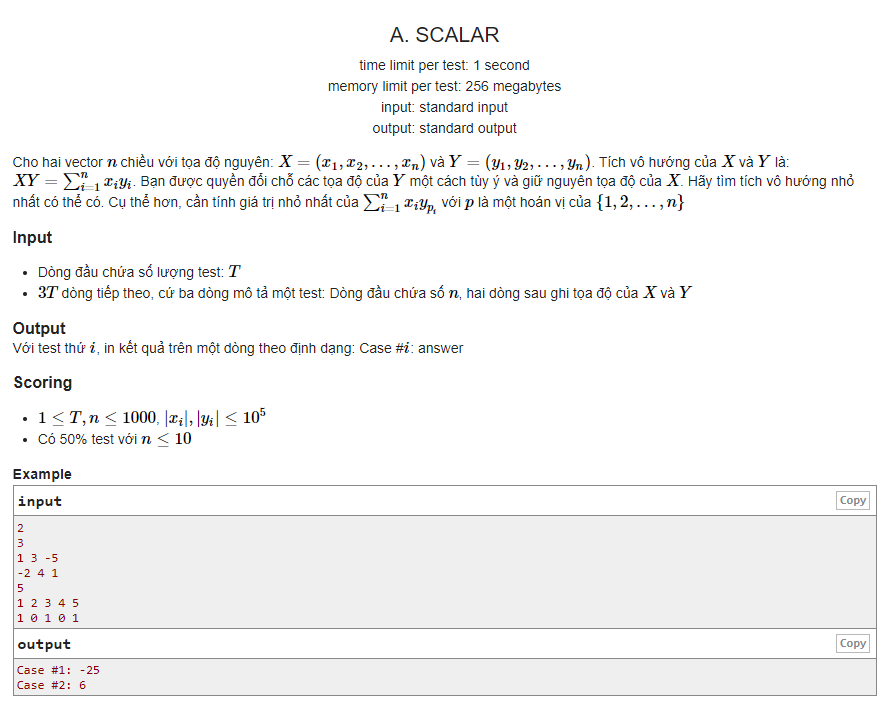
    ull tmp = ans[n] + ans[n + 1];

    tmp %= 100000007;

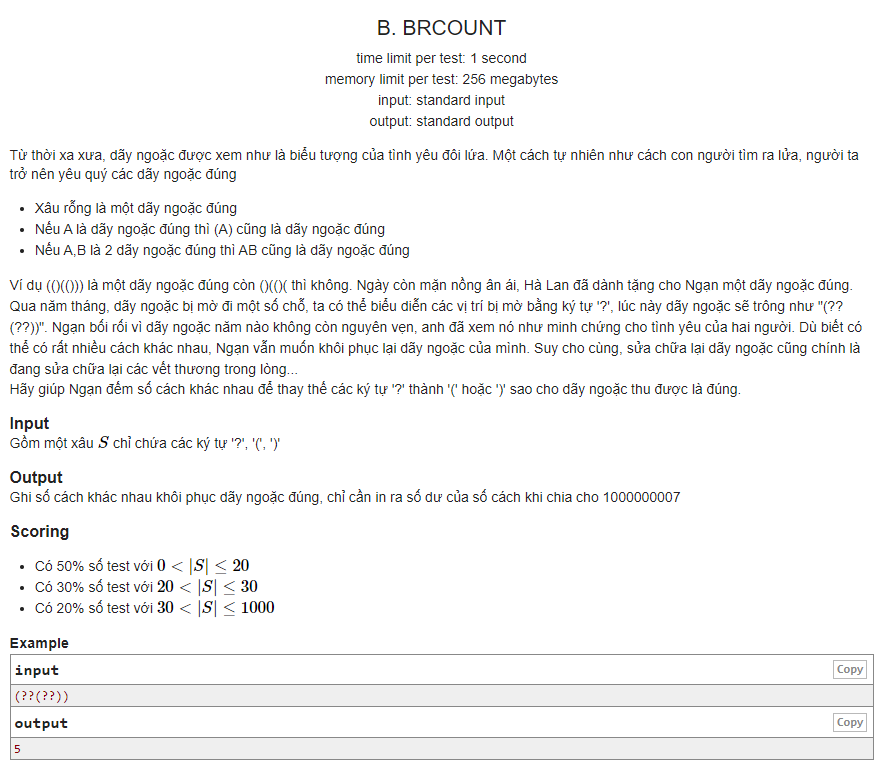
    cout << tmp;

}

**Demo Contest**

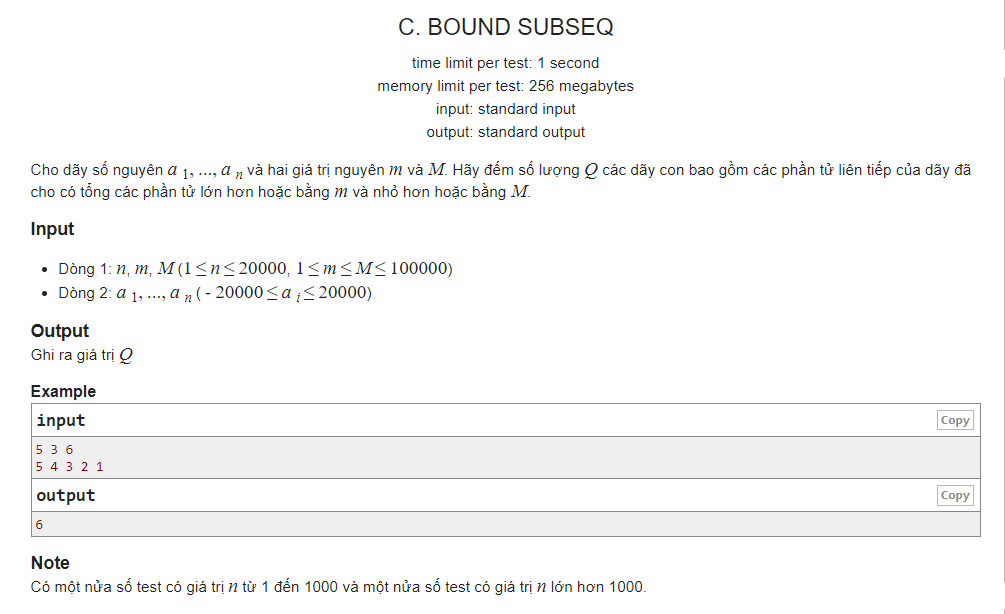


* Sx tăng dần vector X
* Sx giảm dần vector Y
* Lấy X\*Y
* Code:
* **#include** <bits/stdc++.h>
* **using** **namespace** std;
* **#define** all(s) s.begin(), s.end()
* **#define** pb push\_back
* **#define** ii pair<**int**, **int**>
* **#define** x first
* **#define** y second
* **#define** bit(x, y) ((x >> y) & 1) //Ktra bit thứ y của x bằng 1 ?
* **int** main() {
* ios::sync\_with\_stdio(**false**); cin.tie(0);
* cout.tie(0);
* **int** ntest;
* cin >> ntest;
* **for** (**int** test = 1; test <= ntest; test++) {
* **int** n;
* cin >> n;
* **vector<int>** a(n);
* **vector<int>** b(n);
* **for** (**int** &e : a) cin >> e;
* **for** (**int** &e : b) cin >> e;
* sort(all(a));
* sort(all(b));
* reverse(all(b));
* **long** **long** ans = 0;
* **for** (**int** i = 0; i < n; i++) {
* ans += 1LL \* a[i] \* b[i];
* }
* cout << **"Case #"** << test << **": "** << ans << **'\n'**;
* }
* **return** 0;
* }



* Đề bài: Số cách thay dấu ? để xâu thành 1 dãy ngoặc đúng

1. **#include** <bits/stdc++.h>
2. **using** **namespace** std;
4. **#define** all(s) s.begin(), s.end()
5. **#define** pb push\_back
6. **#define** ii pair<**int**, **int**>
7. **#define** x first
8. **#define** y second
9. **#define** bit(x, y) ((x >> y) & 1)
11. **const** **int** MOD = 1000000007;
13. **int** dp[1005][1005];
15. **int** main() {
16. ios::sync\_with\_stdio(**false**); cin.tie(0);
17. cout.tie(0);
18. string s;
19. cin >> s;
20. **int** n = s.size();
21. s = **" "** + s;
22. dp[0][0] = 1;
23. **for** (**int** i = 1; i <= n; i++) {
24. **for** (**int** j = 0; j <= n; j++) {
25. **if** (s[i] == **'('** && j >= 1) dp[i][j] = dp[i - 1][j - 1];
26. **if** (s[i] == **')'**) dp[i][j] = dp[i - 1][j + 1];
27. **if** (s[i] == **'?'**) {
28. dp[i][j] = (dp[i - 1][j + 1] + (j >= 1 ? dp[i - 1][j - 1] : 0)) % MOD;
29. }
30. }
31. }
32. cout << dp[n][0];
33. **return** 0;
34. }



1. **#include** <bits/stdc++.h>
2. **#define** MAXN 20005
3. **using** **namespace** std;
5. **int** a[MAXN], n, m, M;
6. **long** **long** s[MAXN], res;
8. **int** main() {
9. ios::sync\_with\_stdio(0); cin.tie(0);
10. cin >> n >> m >> M;
11. **for**(**int** i = 1; i <= n; i++)
12. cin >> a[i];
13. **for**(**int** i = 1; i <= n; i++)
14. s[i] = s[i - 1] + a[i];
15. **for**(**int** i = 1; i <= n; i++)
16. **for**(**int** j = 1; j <= i; j++) {
17. **long** **long** x = s[i] - s[j - 1];
18. **if** (x >= m && x <= M)
19. res ++;
20. }
21. cout << res;
22. }

#include <bits/stdc++.h>

using namespace std;

#define all(s) s.begin(), s.end()

#define pb push\_back

#define ii pair<int, int>

#define x first

#define y second

#define bit(x, y) ((x >> y) & 1)

#define M\_PI   3.14159265358979323846264338327950288

//04.PIE

/\*

- N cái bánh(htron, bk: r) và F+1 người

- Mỗi người nhận đc 1 miếng bánh (có S bằng nhau) từ 1 chiếc bánh

- Tìm lượng bánh max mỗi người

=> Gọi p[i] là người ăn chiếc bánh thứ i

C1: Tìm kiếm vét cạn, chia người vào từng bánh (lâu)

C2: Tìm kiếm lượng bánh mỗi người nhận đc

  Thử từng kết quả ( S 1 người nhận ) xem nó có thể chia max bn người

    + số người > F+1 => ghi nhận 1 KQ

\*/

int main() {

    ios::sync\_with\_stdio(false); cin.tie(0);

    cout.tie(0);

    int test;

    int N,F;

    int max\_r;

    int r[10000];

    cin >> test;

    while(test--){

        cin >>  N >> F;

        memset(r,0,sizeof(r));

        max\_r=0;

        for(int i=0;i< N;i++){

            int tmp;

            cin >> tmp;

            max\_r =max(max\_r,tmp);

            r[i]=tmp\*tmp;

        }

       sort(r,r+N);

       double low=0,high=M\_PI\*max\_r,mid;

       for(int i=0;i<100;i++){

        mid =(low+high)/2;

        int cont=0;

        for(int i=N-1;i>=0&&cont<=F;i--){

            cont += (int)floor(M\_PI\*r[i] /mid);

        }

        if(cont>F)low=mid;

        else high=mid;

       }

       cout <<fixed<<setprecision(6) << mid <<endl;

    }

    return 0;

}

* FiboWords

#include <deque>

#include <iostream>

#include <iterator>

#include <list>

#include <locale>

#include <map>

#include <queue>

#include <set>

#include <vector>

#include <algorithm>

#include <iostream>

#include <string>

#define ll long long

using namespace std;

string p;

string Fibo[101];

long long c[101];

long long mc[2];

long long fibo\_n = 9; //so Fibo da dc chuan bi

void preprocessing() {

    Fibo[0] = "0";

    Fibo[1] = "1";

    Fibo[2] = "10";

    Fibo[3] = "101";

    Fibo[4] = "10110";

    Fibo[5] = "10110101";

    Fibo[6] = "1011010110110";

    Fibo[7] = "101101011011010110101";

    Fibo[8] = "1011010110110101101011011010110110";

    Fibo[9] = "1011010110110101101011011010110110101101011011010110101";

    while (Fibo[fibo\_n].length()<2000000)

    {

        Fibo[fibo\_n + 1] = Fibo[fibo\_n] + Fibo[fibo\_n - 1];

        fibo\_n++;

        //cout << "Fibo " << fibo\_n << ": " << "\n";

    }

}

long long count(string s, string p) {

    long long count = 0;

    for (long long i = 0; i < s.size() - p.size(); i++) {

        string tmp = s.substr(i, i + p.size());

        if (p == tmp) count++;

    }

    return count;

}

long long count(int n, string p) {

    if (c[n] < 0) {

        c[n] = count(n - 1, p) + count(n - 2, p) + mc[n % 2];

    }

    return c[n];

}

long long solve(int n, string p) {

    int lp = p.size();

    if (n < fibo\_n && Fibo[n].size() < lp) return 0;

    for (int i = 0; i < 101; i++) c[i] = -1;

    int i = 1;

    while (Fibo[i-1].size() < lp) i++; //tim ra fibo dau tien do length >p

    c[i - 1] = count(Fibo[i - 1], p);

    c[i] = count(Fibo[i], p);

    string x = Fibo[i].substr(0, lp - 1);

    string a = Fibo[i - 1].substr(Fibo[i - 1].size() - lp + 1);

    string b = Fibo[i].substr(Fibo[i].size() - lp + 1);

    mc[i % 2] = count(a + x, p);

    mc[(i + 1) % 2] = count(b + x, p);

    return count(n, p);

}

int main()

{

    preprocessing();

    for (int i = 1, n; scanf("%d", &n) != EOF; i++) {

        cin >> p;

        cout << "Case " << i << ": " << solve(n, p) << endl;

    }

    return 0;

}