1. Training 1
   1. ADD

Cho hai số nguyên a và b. Yêu cầu viết chương trình tính tổng c=a+b bằng ngôn ngữ C/C++. Lưu ý giới hạn: a,b<1019 dẫn đến c có thể vượt quá khai báo long long.

Input

Gồm hai dòng, mỗi dòng ghi một số nguyên

Output

Một dòng chứa số nguyên là kết quả bài toán

Scoring

0≤a,b≤9×1018 Có 50% test với a,b≤109

-Code

#include<iostream>

using namespace std;

string Sum(string a, string b){

while (a.length() < b.length())

{

a = "0" + a;

}

while (b.length() < a.length())

{

b = "0" + b;

}

string tong;

int sum, nho = 0;

int sokytu = a.length();

for (int i = sokytu - 1; i >= 0; i--)

{

sum = (a[i] - '0') + (b[i] - '0') + nho;

nho = sum/10;

tong = char(sum%10 + '0') + tong;

}

if(nho == 1) tong = "1" + tong;

return tong;

}

main(){

string a, b;

cin >> a >> b;

cout << Sum(a,b);

}

* 1. SUBSEQMAX

Cho mảng s=(a1,...,an)

Một đoạn s(i,j)=(ai,...,aj),1≤i≤j≤n

Trọng số w(i,j)=ai+ai+1...+aj

Hãy tìm một đoạn trong mảng có trọng số lớn nhất, nghĩa là tổng các số trong đoạn là lớn nhất.

Input

Dòng thứ nhất chứa một số nguyên n≤106 .

Dòng thứ hai chứa n số nguyên.

Output

Ghi ra duy nhất một số nguyên là trọng số lớn nhất tìm được.

-Code

#include<iostream>

using namespace std;

int a[1000000];

main(){

int n, x, y;

cin >> n;

// int \*a = new int[n];

int max = - INTPTR\_MAX;

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

{

cin >> a[i];

}

x = a[0];

y = a[0];

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

{

x += a[i];

if (x < a[i])

{

x = a[i];

}

if (x > y)

{

y = x;

}

}

cout << "\n" << y;

// delete a;

}

* 1. ADDMOD

(a + b) mod (10 ^ 9 + 7) a và b thuộc loại dài

Đầu vào

chứa a và b

Đầu ra

(a + b) mod (10 ^ 9 + 7)

-Code

#include<bits/stdc++.h>

using namespace std;

int main(){

int p = 1e9 + 7;

long long a, b;

cin >> a >> b;

a = a%p;

b = b%p;

cout << (a + b )%p;

}

* 1. EXPMOD

Cho hai số nguyên a và b. Tính toán a^b mod (10^9+ 7)

Đầu vào

Một dòng chứa hai số nguyên a và b

Đầu ra

Giá trị a^b mod (10^9+ 7)

-Code

#include<bits/stdc++.h>

using namespace std;

int main(){

int p = 1e9 + 7;

long long a, b;

long ret = 1;

cin >> a >> b;

a %= p;

b %= p - 1;

while (b > 0)

{

if (b % 2 > 0)

ret = ret \* a % p;

a = a \* a % p;

b /= 2;

}

cout << ret;

}

* 1. SUMSEQ

Đưa ra một chuỗi a1,a2,..an. Tính toánS= =a1+a2+ ... +an.

n ≤106, 0 ≤mộtTôi≤109

Đầu vào

Dòng 1: số n của các yếu tố

Dòng 2: n số nguyên a1,a2, ... ,an

Đầu ra

Giá trị độc nhất S mod (109+ 7)

-Code

#include<bits/stdc++.h>

using namespace std;

int main(){

int p = 1e9 + 7;

int n;

cin >> n;

int a[n];

int S = 0;

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

{

cin >> a[i];

a[i] = a[i] % p;

S = (S + a[i]) %p;

}

cout << S;

}

1. Training 2
   1. HIST

Input

Chứa một hoặc nhiều test. Mỗi test mô tả một đa giác bắt đầu bằng số nguyên n (1≤n≤1000000) là số lượng hình chữ nhật nhỏ cấu thành đa giác. Tiếp theo sau là n số nguyên l1,l2,…,ln với 0≤li≤100000000 lần lượt từ trái sang phải biểu thị chiều dài của các hình chữ nhật. Chiều rộng của các hình chữ nhật bằng nhau và bằng 1. File kết thúc với dòng ghi duy nhất một số 0.

Output

Với mỗi test ghi trên một dòng diện tích của hình chữ nhật nằm trong đa giác thỏa mãn điều kiện đề bài.

#include<iostream>

#include<stack>

using namespace std;

long long calculate(int a[], int n){

stack<int> st;

long long max = 0;

int top;

long long area\_top;

int i = 0;

while (i < n)

{

if (st.empty() || a[st.top()] <= a[i])

{

st.push(i++);

}else

{

top = st.top();

st.pop();

area\_top =(long long) a[top] \* (st.empty() ? i : i - st.top() - 1);

if (max < area\_top)

{

max = area\_top;

}

}

}

while (st.empty() == false)

{

top = st.top();

st.pop();

area\_top = (long long) a[top] \* (st.empty() ? i : i - st.top() - 1);

if (max < area\_top)

{

max = area\_top;

}

}

return max;

}

main(){

int testCase;

cin >> testCase;

while (testCase)

{

int a[testCase];

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

{

cin >> a[i];

}

cout << calculate(a, testCase) << endl;

cin >> testCase;

}

}

* 1. POSTMAN

Chuyển phát hàng là một công việc quan trọng trong thương mại điện tử là lĩnh vực phát triển bùng nổ trong thời gian hiện nay. Ta xét công việc của một nhân viên giao hàng của Công ty XYZ chuyên bán hàng trên mạng. Nhân viên giao hàng cần phát các kiện hàng (được đóng gói trong các hộp cùng kích thước) đến các khách hàng có địa chỉ trên một đại lộ có dạng một đường thẳng.

Nhân viên giao hàng sẽ nhận các kiện hàng tại trụ sở công ty ở vị trí x=0, và cần chuyển phát hàng đến n khách hàng, được đánh số từ 1 đến n. Biết xi và mi là vị trí của khách hàng i và số lượng kiện hàng cần chuyển cho khách hàng này. Do các kiện hàng là khá cồng kềnh nên mỗi lần đi chuyển phát nhân viên giao hàng chỉ có thể mang theo không quá k kiện hàng.

Nhân viên giao hàng xuất phát từ trụ sở, nhận một số (không quá k) kiện hàng và di chuyển theo đại lộ để chuyển phát cho một số khách hàng. Khi giao hết các kiện hàng mang theo, nhân viên giao hàng lại quay trở về trụ sở và lặp lại công việc nói trên cho đến khi chuyển phát được tất cả các kiện hàng cho khách hàng. Sau khi kết thúc công việc chuyển phát, nhân viên phải quay trở lại trụ sở công ty để nộp cho phòng kế toán tất cả các hóa đơn giao nhận có ký nhận của khách hàng. Giả thiết là: tốc độ di chuyển của nhân viên là 1 đơn vị khoảng cách trên một đơn vị thời gian. Thời gian nhận hàng ở trụ sở công ty và thời gian bàn giao hàng cho khách hàng được coi là bằng 0.

Yêu cầu: Giả sử thời điểm mà nhân viên giao hàng bắt đầu công việc là 0. Hãy giúp nhân viên giao hàng tìm cách hoàn thành công việc đã mô tả ở trên tại thời điểm sớm nhất.

Input

Dòng đầu tiên chứa hai số nguyên dương được ghi cách nhau bởi dấu cách n và k (n≤1000;k≤107).

Dòng thứ i trong số n dòng tiếp theo chứa hai số nguyên được ghi cách nhau bởi dấu cách xi (|xi|≤107) và mi(1≤mi≤107)

Output

Ghi ra một số nguyên là thời điểm sớm nhất mà người giao hàng có thể hoàn thành nhiệm vụ của mì

#include<bits/stdc++.h>

using namespace std;

#define ii pair<int, int>

vector<ii> listDuong;

vector<ii> listAm;

bool compare(ii a, ii b) {

return a.first < b.first;

}

long long delivery(int k, vector<ii> a) {

if (a.size() == 0)

{

return 0;

}

long long sum = 0;

int last = a.size() - 1;

while (last >= 0)

{

int chuaHet = 1, soHang = k, times, check = a[last].second;

if(check % k == 0) {

times = check / k;

chuaHet = 0;

}

else

{

times = check / k +1;

soHang = k - check % k;

}

sum += (long long) a[last].first \* times \* 2;

last--;

while(chuaHet == 1 && last >=0)

{

int temp = a[last].second;

if(temp < soHang) {

soHang -= temp;

last--;

}else if(temp == soHang)

{

last--;

chuaHet = 0;

}else

{

a[last].second -= soHang;

chuaHet = 0;

}

if (last < 0)

{

break;

}

}

}

return sum;

}

int main() {

int n,k,x,m;

cin >> n >> k;

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

{

cin >> x >> m;

if(x > 0) listDuong.push\_back(ii(x,m));

if(x < 0) listAm.push\_back(ii(abs(x),m));

}

if(listAm.size() > 0) sort(listAm.begin(), listAm.end(), compare);

if(listDuong.size() > 0) sort(listDuong.begin(), listDuong.end(), compare);

cout << delivery(k, listDuong) + delivery(k, listAm);

}

* 1. LOCATE

Hùng là tướng quân của một đội quân lớn, bộ phận thông tin đã chỉ ra rằng kẻ địch đã khởi động một loạt những máy bay chiến đấu nhỏ tiến sát tấn công quân của Hùng. Chỉ còn rất ít thời gian để phát hiện ra vị trí chính xác của các chiến đấu cơ này để mà tiêu diệt chúng. Hùng đã điều chỉnh lên tối đa độ nhạy của rada để định vị các chiến đấu cơ này dù rằng kích thước của chúng rất nhỏ, tuy nhiên điều đó cũng định vị luôn cả những chú chim đang bay gần đó. Tuy vậy Hùng biết rằng tất cả các chiến đấu cơ dịch chuyển chính xác theo cùng một cách, điều đó hi vọng giúp Hùng định vị chính xác vị trí của đàn chiến đấu cơ. Cho biết mô tả của 2 hình ảnh mà rada thu được cách nhau 1 phút, Hùng muốn định vị tập lớn nhất các điểm trên ảnh 1, mà có thể tìm thấy nó trên ảnh 2 sau một khoảng di chuyển xác định. Biết rằng dàn chiến đấu cơ có thể không xuất hiện ở trên cả hai ảnh rada.

Input

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

Dòng đầu tiên chứa 2 số nguyên L và C tương ứng là số dòng và số cột của 2 ảnh (1≤L,C≤1000). 2∗L dòng tiếp theo mỗi dòng chứa C số nguyên 0 hoặc 1 cách nhau bởi dấu cách, trong đó L dòng đầu mô tả ảnh 1, L 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.

Output

Với mỗi test tương ứng, ghi duy nhất một số nguyên N là số điểm lớn nhất các đối tượng dịch chuyển theo cùng một cách thức.

#include <bits/stdc++.h>

using namespace std;

typedef pair<int, int> ii;

int a[2000][2000];

int main()

{

int testcase;

int L, C;

int point = 0;

cin >> testcase;

while(testcase--)

{

int size1 = 0;

cin >> L;

cin >> C;

ii pic1[10000];

int MAX = 0;

ii object;

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

{

for (int j = 0; j < C; j++)

{

cin >> point;

if (point)

pic1[size1++] = make\_pair(i, j);

}

}

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

{

for (int j = 0; j < C; j++)

{

cin >> point;

if (point){

for (int p = 0; p < size1; ++p){

a[i - pic1[p].first+L][j - pic1[p].second+C]++;

}

}

}

}

for (int i = 0; i < 2 \* L; i++)

{

for (int j = 0; j < 2 \* C; j++)

{

MAX = max(a[i][j],MAX);

}

}

cout << MAX << endl;

memset(a,0,sizeof(a));

}

return 0;

}

* 1. WATERJUG BFS]

Có hai bình, a-lít bình và b-lít bình (a , blà các số nguyên dương). Có một máy bơm với nước không giới hạn. Cho một số nguyên dươngc, làm thế nào để có được chính xác c lít.

Đầu vào

Dòng duy nhất chứa số nguyên dương a , b , c (1 ≤ a , b , c ≤ 900).

Đầu ra

Dòng chứa số bước tối thiểu để có được c lít hoặc -1 nếu không tìm thấy giải pháp.

#include <iostream>

#include <algorithm>

using namespace std;

int ansFun(int m,int n,int s){

int ans=1;

int Tm = m;

int Tn =0;

while(Tn != s && Tm !=s){

int temp = min(Tm,n-Tn);

Tn+=temp;

Tm-=temp;

ans++;

if(Tm == s || Tn == s){

break;

}

if(Tm==0){

ans++;

Tm = m;

}

if(Tn == n){

ans++;

Tn=0;

}

}

return ans;

}

int fun(int m,int n,int s){

// if( s>n){

// return -1;

// }

if(m==s || n==s){

return 1;

}

int g = \_\_gcd(m,n);

if(s%g !=0){

return -1;

}

return min(ansFun(m,n,s),ansFun(n,m,s));

}

int main()

{

int b,c,d;

cin>>b>>c>>d;

cout<<fun(b,c,d)<<endl;

return 0;

}

* 1. REROAD

Đường vành đai III của thành phố Naho nổi tiếng về chất lượng mặt đường tồi tệ. Lí do là đội ngũ sửa chữa đường của thành phố này quá tùy tiện. Đường vành đai III được chia nhỏ thành N đoạn kế tiếp nhau có cùng chiều dài đơn vị. Mỗi lần sửa chữa đường họ tiến hành như sau: Một nhóm công nhân sẽ lựa chọn một đoạn đường nào đó và thay thế toàn bộ lớp nhựa phủ đường trên đoạn đó. Loại nhựa đường được thay thế trên đoạn này có thể khác hẳn với loại nhựa đường trên các đoạn khác làm gây khó khăn cho việc đi lại trên đường. Là cư dân thành phố Naho và là một lập trình viên giỏi, Hải quyết định sử dụng hiểu biết của mình để giúp ích cho xã hội và làm thuận tiện cho cuộc sống của người dân thành phố khi phải đi qua đường vành đai III. Cụ thể là Hải quyết định tạo trang web chứa thông tin về độ gập ghềnh của đường. Hải đánh số các đoạn đường từ 1 đến N và thu thập thông tin về loại nhựa đường trên từng đoạn t1,t2,…,tN (ti là mã loại nhựa đường phủ trên đoạn đường thứ i ). Hải định nghĩa một phần đường là một dãy liên tục các đoạn đường được phủ cùng loại nhựa phủ tk và bên trái và bên phải phần đường đó là các đoạn đường (nếu tồn tại) được phủ loại nhựa khác. Cuối cùng, Hải xác định độ gập ghềnh của đường bằng tổng số lượng phần đường trên đường vành đai III. Ví dụ đường phố chứa các đoạn đường được phủ bởi loại nhựa có mã lần lượt tương ứng với 1,1,0,1,1,1 sẽ có độ gập ghềnh bằng 3 vì nó chứa đúng 3 phần đường 11, 0 và 111. Đường phố chứa các đoạn đường được phủ bởi loại nhựa có mã lần lượt tương ứng với 2,2,2,2 là lý tưởng vì nó chỉ chứa 1 phần đường và có độ gập ghềnh đúng bằng 1.

Dân chúng sẽ hài lòng nếu Hải luôn có thể tính toán và cung cấp trên trang web độ gập ghềnh của đường tại thời điểm hiện tại. Đáng tiếc là mặt đường được thay đổi khá thường xuyên và Hải không muốn mỗi lần như vậy lại phải ra đường thu thập dữ liệu. Vì vậy Hải yêu cầu đội ngũ sửa chữa đường mỗi lần sửa đường phải gửi một thông báo cho Hải. Mỗi thông báo bao gồm 2 số là số thứ tự đoạn đường được sửa và mã loại nhựa được phủ mới. Nhiệm vụ của Hải là phải cập nhật độ gập ghềnh thực tế của đường sau mỗi thông báo như vậy.

Input

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

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

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

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

Output

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

#include<bits/stdc++.h>

using namespace std;

int rugged(int a[], int N, int p, int c, int test, int &count){

if(test == c) return count;

a[p-1] = c;

int a1 = a[p-2], a3 = a[p];

if (p>1 && p< N)

{

if (a1 == test && a3 == test)

{

count +=2;

return count;

}else if (a1 == test)

{

if (a3 == c)

{

return count;

}

count++;

return count;

}else if (a3 == test)

{

if (a1 == c)

{

return count;

}

count++;

return count;

}else

{

count = count - ((a1-c)?0:1) - ((a3-c)?0:1);

return count;

}

}else if (p == 1)

{

if (c == a3)

{

count--;

return count;

}

if (a3 == test)

{

count++;

return count;

}

return count;

}else

{

if (c == a1)

{

count--;

return count;

}

if (a1 == test)

{

count++;

return count;

}

return count;

}

}

main(){

int N, Q;

cin >> N;

int count = 1;

int road[N];

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

{

cin >> road[i];

if(i != 0 && road[i] != road[i - 1]) count++;

}

cin >> Q;

int p, c;

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

{

cin >> p >> c;

cout << rugged(road, N, p, c, road[p-1], count) << endl;

}

}

* 1. SIGNAL

Một thiết bị cảm biến có nhiệm vụ thu nhận dữ liệu về các đối tượng trong 1 khu vực để truyền về cho trung tâm xử lý. Mỗi đối tượng sẽ được biểu diễn bởi 1 dãy số nguyên dương. Như vậy, cảm biến sẽ truyền các dãy số về cho trung tâm xử lý. Tuy nhiên, do các đối tượng ở gần nhau và có tín hiệu nhiễu nên một dãy số gửi về cho trung tâm có thể là dữ liệu của 2 đối tượng. Dãy a1,...,an sẽ là dữ liệu của 2 đối tượng nếu có 1 vị trí i (1<i<n) sao cho max{a1,..,ai−1}−ai≥b và max{ai+1,..,an}−ai≥b với b là hằng số cho trước (xem minh hoạ trong Hình 1). Khi phát hiện một dãy số a1,...,an là dữ liệu của 2 đối tượng thì cần phải tiến hành cắt dãy số đó thành 2 dãy, mỗi dãy là dữ liệu của một đối tượng. Khi đó vị trí cắt sẽ là vị trí i sao cho max{a1,..,ai−1}−ai+max{ai+1,..,an}−ai đạt giá trị lớn nhất (giá trị đó gọi là độ đo cắt tín hiệu).

Ví dụ: với giá trị b = 5 thì dãy số 3, 5, 4, 7, 2, 5, 4, 6, 9, 8 là dữ liệu của 2 đối tượng vì tìm thấy vị trí i=5 tại đó max{3,5,4,7}−2≥5 và max{5,4,6,9,8}−2≥5 và vị trí i=5 cũng chính là vị trí cắt.

Yêu cầu: cho trước giá trị n,b và dãy số nguyên dương a1,...,an. Hãy lập trình kiểm tra xem dãy số a1,...,an có phải là dữ liệu biểu diễn 2 đối tượng hay không và tính độ đo cắt tín hiệu nếu câu trả lời là có.

Input

Dữ liệu đầu vào bao gồm các dòng sau:

Dòng thứ nhất chứa 2 số nguyên dương n và b (3≤n≤200000,1≤b≤50)

Dòng thứ 2 chứa n tự nhiên a1,...,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.

#include<bits/stdc++.h>

using namespace std;

int\* leftMax(int a[], int size){

int max = a[0];

int \*b = new int[size - 2];

for (int i = 1; i < size-1; i++)

{

b[i-1] = max - a[i];

if (max < a[i])

{

max = a[i];

}

}

return b;

}

int\* rightMax(int a[], int size){

int max = a[size - 1];

int \*b= new int[size - 2];

for (int i = size - 2; i > 0; i--)

{

b[i-1] = max - a[i];

if (max < a[i])

{

max = a[i];

}

}

return b;

}

int calculate(int a[], int b, int c){

int \*left;

left = leftMax(a, b);

int \*right;

right = rightMax(a, b);

int max = -1;

for (int i = 0; i < b - 2; i++)

{

int templeft = left[i];

int tempright = right[i];

if (templeft >= c && tempright >= c)

{

if (max < tempright + templeft)

{

max = tempright + templeft;

}

}

}

delete[] left;

delete[] right;

return max;

};

main(){

int n, b;

cin >> n >> b;

int number[n];

int max = 0, min = \_\_INT16\_MAX\_\_;

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

{

cin >> number[i];

}

cout << calculate(number, n, b);

}

* 1. SORT INT

Cho mảng các số nguyên a. Hãy sắp xếp mảng A theo thứ tự tăng dần

Input

Dòng đầu chứa một số nguyên là số phần tử của mảng n≤106.

Dòng thứ hai chứa n số nguyên.

Output

In ra trên một dòng mảng a theo thứ tự tăng dần.

#include<bits/stdc++.h>

using namespace std;

void swap(int &a, int &b) {

int temp;

temp = a;

a = b;

b = temp;

};

int mark(int A[], int Left, int Right) {

int i = Left, j = Right+1, p = A[Left];

while (i < j) {

i++;

while((i <= Right ) && (A[i] < p)) i++;

j--;

while((j >= Left ) && (A[j] > p)) j--;

swap (A[i], A[j]);

}

swap(A[i], A[j]);

swap(A[j], A[Left]);

return j;

};

void quickSort(int A[], int first, int last) {

if (first < last) {

int partition = mark(A, first, last);

if (first < partition) quickSort(A, first, partition-1);

if (last > partition) quickSort(A, partition+1, last);

}

};

int main(){

int p = 1e9 + 7;

int n;

cin >> n;

int a[n];

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

{

cin >> a[i];

}

quickSort(a, 0, n - 1);

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

{

cout << a[i] << " ";

}

}

* 1. Cd and pwd commands

Vasya đang viết một shell hệ điều hành và nó sẽ có các lệnh để làm việc với các thư mục. Để bắt đầu, anh quyết định thực hiện chỉ với hai lệnh: cd (thay đổi thư mục hiện tại) và pwd (hiển thị thư mục hiện tại).

Các thư mục trong hệ điều hành của Vasya tạo thành một cấu trúc cây phân cấp truyền thống. Có một thư mục gốc duy nhất, được biểu thị bằng ký tự gạch chéo " / ". Mỗi thư mục khác có một tên - một chuỗi không trống bao gồm các chữ cái Latinh viết thường. Mỗi thư mục (ngoại trừ thư mục gốc) có một thư mục mẹ - thư mục chứa thư mục đã cho. Nó được ký hiệu là " .. ".

Lệnh cd nhận một tham số duy nhất, đó là một đường dẫn trong hệ thống tệp. Lệnh thay đổi thư mục hiện tại thành thư mục được chỉ định bởi đường dẫn. Đường dẫn bao gồm tên của các thư mục được phân tách bằng dấu gạch chéo. Tên của thư mục có thể là " .. ", có nghĩa là một bước lên thư mục mẹ. « .. » Có thể được sử dụng ở bất kỳ nơi nào trên đường dẫn, có thể nhiều lần. Nếu đường dẫn bắt đầu bằng dấu gạch chéo, nó được coi là đường dẫn tuyệt đối, nghĩa là thư mục thay đổi thành đường dẫn được chỉ định, bắt đầu từ thư mục gốc. Nếu tham số bắt đầu bằng tên thư mục (hoặc " .."), Nó được coi là một đường dẫn tương đối, nghĩa là thư mục thay đổi thành thư mục được chỉ định, bắt đầu từ thư mục hiện tại.

Lệnh pwd sẽ hiển thị đường dẫn tuyệt đối đến thư mục hiện tại. Đường dẫn này không được chứa " .. ".

Ban đầu, thư mục hiện tại là root. Tất cả các thư mục được đề cập rõ ràng hoặc được truyền gián tiếp trong bất kỳ lệnh cd nào đều được coi là tồn tại. Nó được đảm bảo rằng không có sự cố gắng chuyển sang thư mục mẹ của thư mục gốc.

Đầu vào

Dòng đầu tiên của dữ liệu đầu vào chứa số nguyên n ( 1 ≤  n  50 ) - số lượng lệnh.

Sau đó theo n dòng, mỗi dòng chứa một lệnh. Mỗi dòng này chứa lệnh pwd hoặc lệnh cd , theo sau là một tham số không trống được phân tách bằng dấu cách.

Tham số lệnh cd chỉ chứa các chữ cái Latinh chữ thường, dấu gạch chéo và dấu chấm, hai dấu gạch chéo không thể đi liên tiếp, dấu chấm chỉ xuất hiện dưới dạng tên của thư mục giả cha mẹ. Tham số lệnh cd không kết thúc bằng dấu gạch chéo, ngoại trừ khi đó là ký hiệu duy nhất trỏ đến thư mục gốc. Tham số lệnh có độ dài từ 1 đến 200 ký tự, bao gồm.

Các thư mục trong hệ thống tập tin có thể có cùng tên.

Đầu ra

Đối với mỗi lệnh pwd, bạn nên in đường dẫn tuyệt đối đầy đủ của thư mục đã cho, kết thúc bằng dấu gạch chéo. Nó nên bắt đầu bằng dấu gạch chéo và chứa danh sách các thư mục được phân tách bằng dấu gạch chéo theo thứ tự được lồng từ thư mục gốc đến thư mục hiện tại. Nó không chứa dấu chấm.

#include<bits/stdc++.h>

using namespace std;

deque<string> path;

string current = "/";

stack<string> st;

void store(string a){

int i = 0;

while (i < a.length())

{

int j = i;

while (j++ < a.length())

{

if (a[j] == '/')

{

break;

}

}

if (j == a.length())

{

current = current + a.substr(i, j);

break;

}

string temp = a.substr(i,j - i);

i = j + 1;

/\* tra ve:

../

ahklh/

\*/

if (temp == "/")

{

current = "/";

continue;

}

if (temp[0] == '.')

{

int k = current.length() - 2;

while (current[k] != '/')

{

k--;

}

current = current.substr(0, k);

current = current + '/';

}

else if (temp[0] == '/')

{

current = temp + '/';

}else

{

current = current + temp + '/';

}

}

}

int main(){

int n;

// freopen("input.txt", "r", stdin);

cin >> n;

string b;

getline(cin, b);

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

{

string a;

getline(cin, a);

if (a[0] == 'c')

{

int tmp = a.length() - 1;

a = a.substr(3, tmp);// loai bo "cd "

store(a);

}else

{

path.push\_back(current);

}

}

while (!path.empty())

{

cout << path.front() << endl;

path . pop\_front();

}

}

* 1. Array

Vitaly có một mảng gồm n số nguyên khác nhau. Vitaly muốn chia mảng này thành ba bộ không trống để các điều kiện sau được giữ:

Tích của tất cả các số trong tập đầu tiên nhỏ hơn 0 (<0) .

Tích của tất cả các số trong tập thứ hai lớn hơn 0 (> 0) .

Tích của tất cả các số trong tập thứ ba bằng 0.

Mỗi số từ mảng ban đầu phải xuất hiện trong đúng một bộ.

Giúp Vitaly. Chia mảng đã cho.

Đầu vào

Dòng đầu tiên của đầu vào chứa số nguyên n (3  n  100) . Dòng thứ hai chứa n số nguyên khác biệt được phân tách bằng dấu cách a 1 ,  a 2 , ...,  a n (| a i | 10 3 ) - các phần tử mảng.

Đầu ra

Trong dòng đầu tiên in số nguyên n 1 ( n 1  > 0) - số phần tử trong tập đầu tiên. Sau đó in n 1 số - các phần tử có được tập đầu tiên.

Trong dòng tiếp theo in số nguyên n 2 ( n 2  > 0) - số phần tử trong tập thứ hai. Sau đó in n 2 số - các phần tử có được tập thứ hai.

Trong dòng tiếp theo in số nguyên n 3 ( n 3  > 0) - số phần tử trong tập thứ ba. Sau đó in n 3 số - các phần tử đã đến tập thứ ba.

Các bộ in phải đáp ứng các điều kiện được mô tả. Nó được đảm bảo rằng các giải pháp tồn tại. Nếu có một vài giải pháp, bạn được phép in bất kỳ giải pháp nào.

#include<bits/stdc++.h>

using namespace std;

int main(){

int n;

cin >> n;

int nega;

int a[n];

int ai = 0, bi = 0;

int b[n];

int check = 0;

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

{

int temp;

cin >> temp;

if (temp > 0)

{

a[ai++] = temp;

}else if(temp < 0)

{

b[bi++] = temp;

}

}

if(ai > 0){

cout << 1 << " " << b[0] << endl;

cout << 1 << " " << a[0] << endl;

cout << ai - 1 + bi << " " << 0 << " ";

for (int i = 1; i < ai; i++)

{

cout << a[i] << " ";

}

for (int i = 1; i < bi; i++)

{

cout << b[i] << " ";

}

}else

{

cout << 1 << " " << b[0] << endl;

cout << 2 << " " << b[1] << " " << b[2] << endl;

cout << bi - 2 << " " << 0 << " ";

for (int i = 3; i < bi; i++)

{

cout << b[i] << " ";

}

}

}

1. Training 3
   1. TSP

Một người du lịch xuất phát từ thành phố thứ nhất muốn đi thăm quan tất cả n−1 thành phố khác. mỗi thành phố đúng một lần, rồi quay trở lại thành phố xuất phát.

Yêu cầu: Cho biết chi phí đi lại giữa các thành phố, hãy giúp người du lịch tìm hành trình với tổng chi phí là nhỏ nhất.

Input

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

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

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

Output

Ghi ra duy nhất một số là tổng chi phí hành trình nhỏ nhất tìm được.

* 1. KNAPSAC

Một nhà thám hiểm cần đem theo một cái túi có trọng lượng không quá b. Có n đồ vật có thể đem theo. Đồ vật thứ j có trọng lượng aj và giá trị sử dụng cj. Hỏi nhà thám hiểm cần đem theo những đồ vật nào để cho tổng giá trị sử dụng là lớn nhất mà tổng trọng lượng đồ vật mang theo cái túi không vượt quá b?

Input

Dòng đầu tiên chứa hai số nguyên dương n,b (n≤30,b≤106).

Dòng thứ j trong số n dòng tiếp theo mỗi dòng ghi ra hai số nguyên dương aj,cj≤106.

Output

Ghi ra duy nhất một số là tổng giá trị lớn nhất tìm được của các đồ vật cho vào túi.

#include<iostream>

#include<algorithm>

using namespace std;

#define inf 1000000

int n, b;

int a[33], c[33]; // a - trong luong; c - gia tri

int bag(int n, int b) {

int m[n+2][b+2]; // m[i][j]: trong luong lon nhat khong vuot qua j trong i do vat

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

for (int w = 0; w <= b; w++) {

if (i == 0 || w == 0) m[i][w] = 0;

else if (a[i] <= w) // neu do i duoc chon

m[i][w] = max(c[i] + m[i-1][w-a[i]], m[i-1][w]);

else m[i][w] = m[i-1][w]; // neu do i khong duoc chon

}

}

return m[n][b];

}

int main() {

cin >> n >> b;

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

cin >> a[j] >> c[j];

}

cout << bag(n, b);

}

* 1. BCA

Vào đầu học kỳ, trưởng phòng khoa học máy tính D phải phân công các khóa học cho giáo viên một cách cân bằng. Khoa D có m giáo viên.T= { 1 , 2 , . . . , m } và n khóa học C= { 1 , 2 , . . . , n }. Mỗi giáo viênt ∈ Tcó một danh sách ưu tiên là danh sách các khóa học mà anh ấy / cô ấy có thể dạy tùy thuộc vào chuyên môn của anh ấy / cô ấy. Chúng tôi đã biết một danh sách các cặp xung đột hai khóa học không thể được chỉ định cho cùng một giáo viên vì các khóa học này đã được lên lịch trong cùng một thời gian biểu. Tải trọng của một giáo viên là số lượng các khóa học được giao cho cô ấy / anh ấy. Cách gánn các khóa học để m giáo viên sao cho mỗi khóa học được chỉ định cho một giáo viên nằm trong danh sách ưu tiên của anh ấy / cô ấy, không có hai khóa học xung đột nào được gán cho cùng một giáo viên và tải tối đa là tối thiểu.

Đầu vào

Đầu vào bao gồm các dòng sau

Dòng 1: chứa hai số nguyên m và n (1 ≤ m ≤ 10, 1 ≤ n ≤ 30)

Hàng Tôi+1: chứa số nguyên dương k và k số nguyên dương chỉ ra các khóa học mà giáo viên Tôi có thể dạy (∀ i = 1 , ... , m)

Hàng m + 2: chứa một số nguyên k

Hàng i + m + 2: chứa hai số nguyên Tôi và j chỉ ra hai khóa học mâu thuẫn (∀ i = 1 , ... , k)

Đầu ra

Đầu ra chứa một số duy nhất là tải tối đa của các giáo viên trong giải pháp được tìm thấy và giá trị -1 nếu không tìm thấy giải pháp.

#include <bits/stdc++.h>

using namespace std;

int n, m;

int c[100][100], a[100][100];

int kq, x[100], k, dem[100];

void duyet(int i){

if (i > m){

kq = k;

return;

}

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

bool ok = true;

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

if (a[i][t] && x[t]==j) ok = false;

}

if (ok && c[j][i] && dem[j] + 1 < kq){

int p\_k = k;

dem[j]++;

k = max(k, dem[j]);

x[i] = j;

duyet(i+1);

dem[j]--;

k = p\_k;

}

}

}

main(){

cin >> n >> m;

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

int k;

cin >> k;

while (k--){

int j;

cin >> j;

c[i][j] = 1;

}

}

int t;

cin >> t;

while(t--){

int i, j;

cin >> i >> j;

a[i][j] = a[j][i] = 1;

}

kq = m;

duyet(1);

cout << kq;

}

* 1. CBUS

Có n hành khách 1 , 2 , ... , n. Hành khách Tôi muốn đi du lịch từ điểm Tôi chỉ i + n ( i = 1 , 2 , ... , n ). Có một chiếc xe buýt nằm ở điểm0 và có knơi để vận chuyển hành khách (có nghĩa là bất cứ lúc nào, có nhiều nhất k hành khách trên xe buýt). Bạn được cung cấp ma trận khoảng cách c trong đóc ( i , j ) là khoảng cách di chuyển từ điểm Tôi chỉ j ( i , j = 0 , 1 , ... , 2 n ). Tính toán tuyến đường ngắn nhất cho xe buýt, phục vụn hành khách và quay trở lại điểm 0 không truy cập bất kỳ điểm nào nhiều hơn một lần (ngoại trừ điểm 0).

Đầu vào

Hàng 1 chứa đựng n và k ( 1 ≤ n ≤ 11 , 1 ≤ k ≤ 10 ). Hàngi + 1 ( i = 1 , 2 , ... , 2 n + 1 ) chứa ( tôi - 1)t h dòng của ma trận c (hàng và cột được lập chỉ mục từ 0 , 1 , 2 , . . , 2 n).

Đầu ra

Dòng duy nhất chứa chiều dài của tuyến đường ngắn nhất.

#include<bits/stdc++.h>

using namespace std;

int a[30][30];

int mark[30];

int shortest = INT32\_MAX;

int temproute = 0;

int n, k, cmin=INT32\_MAX;

int dem =0;

int route[30];

int checkvisited(int t){

if(t <= n) return (!mark[t] && dem < k);

return (mark[t-n] && !mark[t]);

}

void Try(int i){

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

{

if (checkvisited(j))

{

if (j<=n)

{

dem++;

}else dem--;

mark[j] = 1;

route[i] = j;

temproute += a[route[i-1]][j];

if (i == 2\*n)

{

shortest = min(shortest,temproute + a[route[i]][0]);

}else

{

int g=temproute+(2\*n+1-i)\*cmin;

if(shortest > g) Try(i+1);

}

temproute -= a[route[i-1]][j];

mark[j] = 0;

route[i] = 0;

if (j<=n)

{

dem--;

}else dem++;

}

}

}

int main(){

// freopen("input.txt", "r", stdin);

cin >> n >> k;

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

{

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

{

cin >> a[i][j];

if (a[i][j])

{

cmin = min(a[i][j], cmin);

}

}

}

Try(1);

cout << shortest;

}

* 1. BACP

BACP là thiết kế một chương trình giảng dạy học thuật cân bằng bằng cách gán thời gian cho các khóa học theo cách mà khối lượng học tập của từng thời kỳ được cân bằng. CóN các khóa học 1 , 2 , ... , N phải được chỉ định cho M Chu kỳ 1 , 2 , ĐÁ , M. Mỗi khóa học Tôi có tín dụng cTôivà có một số khóa học là điều kiện tiên quyết. Tải trọng của một giai đoạn được xác định là tổng số tín chỉ của các khóa học được chỉ định cho giai đoạn đó. Thông tin tiên quyết được thể hiện bằng ma trậnMộtN× N trong đó Mộttôi , j = 1 chỉ ra rằng các khóa học Tôi phải được chỉ định cho một khoảng thời gian trước thời gian mà khóa học j được gán. Tính toán bài tập sao cho tải tối đa cho tất cả các giai đoạn là tối thiểu.

Đầu vào

Dòng 1 chứa N và M (2 ≤ N≤ 16 , 2 ≤ M≤ 5)

Dòng 2 chứa c1,c2, ... ,cN

Hàng i + 2 ( i = 1 , Nhìn , N) chứa Tôit h dòng của ma trận Một

Đầu ra

Dòng duy nhất chứa tải tối đa đó cho tất cả các giai đoạn của giải pháp được tìm thấy

#include<bits/stdc++.h>

using namespace std;

int n,m;

int credits[100];

// int a[100][100];

#define ii pair<int, int>

vector<ii> a;

int assign\_periods[100];

int tong[100];

int ans = INT32\_MAX;

int check(int i, int j){

for (int k = 0; k < a.size(); k++)

{

if (a[k].first == j && (assign\_periods[a[k].second] >= i || assign\_periods[a[k].second] == 0)) return 0;

}

return 1;

}

void Try(int i, int maxsum){

if (i > n)

{

ans = min(ans,maxsum);

return;

}

for (int j = 1; j <= m; j++)

{

if (check(j, i))

{

assign\_periods[i] = j;

tong[j] += credits[i];

int temp = max(tong[j], maxsum);

if (temp < ans)

{

Try(i+1,temp);

}

tong[j] -= credits[i];

}

}

}

int main(){

cin >> n >> m;

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

{

cin >> credits[i];

}

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

{

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

{

int temp;

cin >> temp;

if(temp) a.push\_back(ii(i,j));

}

}

Try(1,0);

cout << ans;

}

* 1. TAXI

Có n hành khách 1 , 2 , ... , n. Hành khách Tôi muốn đi du lịch từ điểm Tôi chỉ i + n ( i = 1 , 2 , ... , n ). Có một chiếc taxi nằm ở điểm0để vận chuyển hành khách. Cho ma trận khoảng cách c ( 2 n + 1 ) \* ( 2 n + 1 ) trong đó c ( i , j ) là khoảng cách di chuyển từ điểm Tôi chỉ j ( i , j = 0 , 1 , ... , 2 n ) Tính toán tuyến đường ngắn nhất cho taxi, phục vụ n hành khách và quay trở lại điểm 0 đến nỗi tại bất kỳ thời điểm nào, không có nhiều hơn một hành khách trên taxi và không có điểm nào được truy cập nhiều hơn một lần (ngoại trừ điểm 0, có thể được truy cập lên đến hai lần).

Đầu vào

Hàng 1 chứa đựng n ( 1 ≤ n ≤ 11 ). Hàngi + 1 ( i = 1 , 2 , ... , 2 n + 1 ) chứa tôi t h dòng của ma trận c.

Đầu ra

Dòng duy nhất chứa chiều dài của tuyến đường ngắn nhất.

#include<bits/stdc++.h>

using namespace std;

int a[30][30];

int mark[30];

int shortest = INT32\_MAX;

int temproute = 0;

int n;

void Try(int i, int src){

if (i == n)

{

temproute += a[src][0];

shortest = shortest < temproute ? shortest: temproute;

return;

}

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

{

if (!mark[j])

{

mark[j] = 1;

int k = temproute;

temproute += a[src][j] + a[j][j+n];

if (temproute < shortest) Try(i+1, j + n);

mark[j] = 0;

temproute = k;

}

}

}

int main(){

// freopen("input.txt", "r", stdin);

cin >> n;

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

{

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

{

cin >> a[i][j];

}

}

Try(0, 0);

cout << shortest;

}

* 1. CVRPCOUNT

Một đội K xe tải giống hệt nhau có công suất Q cần được lên lịch để giao các gói pepsi từ kho trung tâm 0 cho khách hàng 1 , 2 , ... , n. Mỗi khách hàngTôi yêu cầu d[ i ] gói.

Mỗi khách hàng được truy cập chính xác bởi một tuyến đường

Tổng số gói theo yêu cầu của khách hàng của mỗi xe tải không thể vượt quá khả năng của nó

Mỗi xe tải phải ghé thăm ít nhất một khách hàng

Mục tiêu

Số tính toán R giải pháp

Lưu ý rằng: các đơn đặt hàng của khách hàng trong một tuyến rất quan trọng, ví dụ: các tuyến 0 -> 1 -> 2 -> 3 -> 0 và 0 -> 3-> 2 -> 1 -> 0 là khác nhau.

Đầu vào

Dòng 1: n , K, Q (2 ≤ n ≤ 10 , 1 ≤ K≤ 5 , 1 ≤ Q ≤ 20)

Dòng 2: d[ 1 ] , . . . , d[ n ] (1 ≤ d[ I ] ≤ 10)

Đầu ra

R mod 10^9+ 7

#include<bits/stdc++.h>

using namespace std;

#define Fr(i,a,b) for(int i=a;i<=b;i++)

#define For(i,a,b) for(int i=a;i<b;i++)

#define dembit1(x) \_\_builtin\_popcountll(x)

#define fi first

#define se second

typedef long long ll;

typedef pair<int,int> ii;

typedef pair<int,ii> iii;

typedef pair<ll,int> lii;

const int mod=1e9+7;

const int oo = INT\_MAX-1;

const int N = 101;

int n,q,k,d[N],w[N],mark[N], L[N];

int ans, giaithuaa;

int giaithua(int a){

if(a == 0) return 1;

return a \* giaithua(a-1);

}

void dfs(int u)

{

if(u>n)

{

bool ok=true;

Fr(i,1,k) if(w[i]==0)

{

ok=false;

break;

}

if(ok) {

int an = 1;

Fr(i, 1, k) {

an \*= giaithua(L[i]);

}

ans += an;

}

return;

}

Fr(i,1,k)

{

if(w[i]+d[u]<=q)

{

w[i]+=d[u];

L[i]++;

dfs(u+1);//xet nguoi tiep theo

w[i]-=d[u];

L[i]--;

}

}

}

int main()

{

scanf("%d %d %d",&n,&k,&q);

giaithuaa = giaithua(k);

Fr(i,1,n) scanf("%d",&d[i]);

ans=0;

dfs(1);

ans /= giaithuaa;

printf("%d",ans % mod);

return 0;

}

* 1. CVRP OPT

Một đội K xe tải giống hệt nhau có công suất Q cần được lên lịch để giao các gói pepsi từ kho trung tâm 0 cho khách hàng 1 , 2 , ... , n. Mỗi khách hàng i yêu cầu d[ i ]gói. Khoảng cách từ vị tríTôi đến vị trí j Là c [ i , j ], 0 ≤ i , j ≤ n .. Giải pháp phân phối là một tập hợp các tuyến đường: mỗi xe tải được liên kết với một tuyến đường, bắt đầu từ kho, ghé thăm một số khách hàng và quay lại kho để giao các gói pepsi được yêu cầu như sau:

Mỗi khách hàng được truy cập chính xác bởi một tuyến đường

Tổng số gói theo yêu cầu của khách hàng của mỗi xe tải không thể vượt quá khả năng của nó

Mỗi xe tải phải ghé thăm ít nhất một khách hàng

Mục tiêu

Tìm một giải pháp có tổng khoảng cách di chuyển tối thiểu

Lưu ý rằng: các đơn đặt hàng của khách hàng trong một tuyến rất quan trọng, ví dụ: các tuyến 0 -> 1 -> 2 -> 3 -> 0 và 0 -> 3-> 2 -> 1 -> 0 là khác nhau.

Đầu vào

Dòng 1: n , K, Q (2 ≤ n ≤ 12 , 1 ≤ K≤ 5 , 1 ≤ Q ≤ 50)

Dòng 2: d[ 1 ] , . . . , d[ n ] (1 ≤ d[ I ] ≤ 10)

Hàng i + 3: các hang thứ i của ma trận khoảng cách c ( i = 0 , ... , n ) (1 ≤ c [ i , j ] ≤ 30)

Đầu ra

Tổng khoảng cách di chuyển tối thiểu

#include<bits/stdc++.h>

using namespace std;

#define Fr(i,a,b) for(int i=a;i<=b;i++)

#define For(i,a,b) for(int i=a;i<b;i++)

#define dembit1(x) \_\_builtin\_popcountll(x)

#define fi first

#define se second

typedef long long ll;

typedef pair<int,int> ii;

typedef pair<int,ii> iii;

typedef pair<ll,int> lii;

const int mod=1e9+7;

const int oo = INT\_MAX-1;

const int N = 101;

int n,q,k,d[N],w[N],mark[N], L[N];

int ans, giaithuaa;

int giaithua(int a){

if(a == 0) return 1;

return a \* giaithua(a-1);

}

void dfs(int u)

{

if(u>n)

{

bool ok=true;

Fr(i,1,k) if(w[i]==0)

{

ok=false;

break;

}

if(ok) {

int an = 1;

Fr(i, 1, k) {

an \*= giaithua(L[i]);

}

ans += an;

}

return;

}

Fr(i,1,k)

{

if(w[i]+d[u]<=q)

{

w[i]+=d[u];

L[i]++;

dfs(u+1);//xet nguoi tiep theo

w[i]-=d[u];

L[i]--;

}

}

}

int main()

{

scanf("%d %d %d",&n,&k,&q);

giaithuaa = giaithua(k);

Fr(i,1,n) scanf("%d",&d[i]);

ans=0;

dfs(1);

ans /= giaithuaa;

printf("%d",ans % mod);

return 0;

}

* 1. CNK

Cho các số nguyên dương k,n (k<n). Hãy tính giá trị nCk module 10^9+7

Input

Một dòng chứa 2 số nguyên dương k và n (1≤k<n≤1000)

Output

Ghi ra giá trị Ckn module 10^9+7

#include<bits/stdc++.h>

using namespace std;

const unsigned int M = 1e9 + 7;

long long tohop[1000][1000];

int main() {

int n, k;

cin >> k >> n;

for (int i = 0; i <= n; i++) tohop[i][0] = 1;

for (int i = 1; i <= k; i++) tohop[0][i] = 0;

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

for (int j = 1; j <= k; j++) {

tohop[i][j] = ((tohop[i-1][j] % M) + (tohop[i-1][j-1] % M)) % M;

}

}

cout << tohop[n][k];

}

* 1. DIGITS

Viết chương trình C đọc giá trị nguyên N từ stdin, in đến stdout số Q cách gán giá trị 1, 2, …, 9 cho các ký tự I, C, T, H, U, S, K (các ký tự được gán với các giá trị khác nhau) sao cho: ICT- K62 + HUST= N

Đầu vào

Dòng duy nhất chứa một số nguyên N (1 ≤ N≤105)

Đầu ra

Viết giá trị Q

#include<bits/stdc++.h>

using namespace std;

//ICT - K62 + HUST = N =2062

//t = 6; sc = 23, 97; i-k+u =10 || 9; h = 1

//iuk = 594, 785, 897|| 584, 685

int I, C, T, K, H, U, S;

int Q = 0;

int check[10];

//tinh hang ngan

void cal10000(int n, int miss){

for (int i = 1; i <= 9; i++)

{

if (check[i])

{

continue;

}

if (i + miss == n / 1000)

{

// cout << miss << endl;

// cout << I << C << T << "-" << K << "00+" << i << U << S << T << "-" << n << endl;

Q++;

}

}

}

// tinh hang tram

void cal1000(int n, int miss){

for (int i = 1; i <= 9; i++)

{

if (check[i])

{

continue;

}

I = i;

check[i] = 1;

for (int j = 1; j <= 9; j++)

{

if (check[j])

{

continue;

}

U= j;

check[j] = 1;

for (int k = 1; k <= 9; k++)

{

if (check[k])

{

continue;

}

K = k;

check[k] = 1;

int temp = i + j + miss - k;

int temp1 = (n % 1000 - n % 100) / 100;

int miss1 = temp >= 10 ? 1 : 0;

if (temp % 10 == temp1)

{

cal10000(n, miss1);

}

check[k] = 0;

}

check[j] = 0;

}

check[i] = 0;

}

}

// tinh hang chuc2062

void cal100(int n, int miss){

for (int i = 1; i <= 9; i++)

{

if (check[i])

{

continue;

}

S = i;

// cout << "i=" <<miss << "miss" << i;

check[i] = 1;

for (int j = 1; j <= 9; j++)

{

if (check[j])

{

continue;

}

// cout << j;

C = j;

check[j] = 1;

int temp = i + j + miss;

int temp1 = (n % 100 - n % 10) / 10;

int miss1 = temp >= 10 ? 1 : 0;

if (temp % 10 == temp1)

{

cal1000(n, miss1);

}

check[j] = 0;

}

check[i] = 0;

}

}

// tinh hang don vi

void cal10(int n){

for (int i = 1; i <= 9; i++)

{

if (2 \* i % 10 == n % 10)

{

T = i;

int miss = i > 4 ? 1 : 0;

check[i] = 1;

cal100(n, miss);

check[i] = 0;

}

}

}

main(){

int N;

// freopen("ducloz.txt", "r", stdin);

cin >> N;

N += 62;

if (N % 2 != 0)

{

Q = 0;

}else

{

for (int i = 1; i <= 9; i++)

{

check[i] = 0;

}

cal10(N);

}

cout << Q;

}

* 1. PERMUTATIONGEN

Cho 1 một hoán vị H độ dài n với các thành phần nằm trong tập {1,2,…,n}. Tìm hoán vị kế tiếp của hoán vị H trong thứ tự từ điển.

Input

Dòng đầu ghi 1 số nguyên dương n≤104.

Dòng thứ 2 ghi n số nguyên dương ≤n cách nhau bởi dấu cách là hoán vị H.

Output

Ghi ra hoán vị H trên một dòng duy nhất, các thành phần cách nhau bởi dấu cách. Nếu không tồn tại thì ghi ra -1.

#include<bits/stdc++.h>

using namespace std;

int n;

int a[10010];

int mark;

int ans =0 ;

void cal(){

//Tìm từ phải qua trái : j là chỉ số lớn nhất thoả mãn a[j] < a[j+1]

for (int i = n-1; i >= 1; i--)

{

if(a[i] < a[i+1]) {

ans = 1;

mark = i;

break;

}

}

//Tìm a[k] là số nhỏ nhất còn lớn hơn a[j] trong các số bên phải a[j]

if (ans)

{

int k;

for (int i = n; i >= mark; i--)

{

if(a[i]>a[mark]) {

k =i;

break;

}

}

swap(a[mark], a[k]);

int r = n, s = mark+1;

while (r>s)

{

swap(a[r], a[s]);

r--;s++;

}

}

}

int main(){

cin >> n;

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

{

cin >> a[i];

}

cal();

if(ans) for (int i = 1; i <= n; i++)

{

cout << a[i] << " ";

}else

{

cout << -1;

}

}

* 1. COMBINATIONGEN

Cho 1 một chuỗi tổ hợp C độ dài m với các thành phần nằm trong tập {1,2,…,n}. Tìm chuỗi tổ hợp kế tiếp của chuỗi C trong thứ tự từ điển.

Input

Dòng đầu 2 số nguyên dương n,m≤104.

Dòng thứ 2 ghi m số nguyên dương ≤n cách nhau bởi dấu cách.

Output

Ghi ra chuỗi C trên một dòng duy nhất, các thành phần cách nhau bởi dấu cách. Nếu không tồn tại thì ghi ra -1.

#include <iostream>

using namespace std;

int a[100000] ;

int m , n ;

void printResult() { // hàm dùng để in một cấu hình ra ngoài

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

cout<<a[i]<<" " ;

}

cout<<endl ;

}

int main()

{

cin>>n >>m;

int check = 0;

for (int i = 1; i <= m; i++)

{

cin >> a[i];

}

for (int i = m; i >= 1; i--)

{

if (a[i] != n - m + i)

{

check = 1;

a[i]++;

for (int j = i +1; j <= m; j++)

{

a[j] = a[i] + j -i;

}

break;

}

}

if (check)

{

printResult();

}else

{

cout << -1;

}

return 0;

}

* 1. BINARYGEN

Cho 1 một xâu nhị phân S độ dài n. Tìm xâu nhị phân kế tiếp của xâu S trong thứ tự từ điển.

Input

Dòng đầu 1 số nguyên dương n≤104.

Dòng thứ 2 ghi n số 0 hoặc 1 liên tiếp nhau.

#include<bits/stdc++.h>

using namespace std;

int main (){

int n;

cin >> n;

string bin;

cin >> bin;

int check = 0;

for (int i = n - 1; i >= 0; i--)

{

if (bin[i] == '1')

{

bin[i] = '0';

}else

{

bin[i] = '1';

check = 1;

break;

}

}

if (check)

{

cout << bin;

}

else

{

cout << "-1";

}

}

* 1. NURSE

Giám đốc bệnh viện muốn sắp xếp kế hoạch làm việc cho y tá trong một khoảng thời gian nhất định N ngày liên tiếp 1, ..., N. Do chính sách của bệnh viện, mỗi y tá không thể làm việc trong tất cả các ngày 1, ...,N. Thay vào đó, phải có những ngày nghỉ trong đó y tá cần nghỉ ngơi. Một kế hoạch làm việc là một chuỗi các giai đoạn làm việc rời rạc. Một thời gian làm việc của một y tá được định nghĩa là một chuỗi các ngày liên tiếp mà y tá phải làm việc và thời gian làm việc là số ngày liên tiếp của thời gian làm việc đó. Bệnh viện áp đặt hai hạn chế:

Mỗi y tá có thể nghỉ ngơi chỉ một ngày giữa hai giai đoạn làm việc liên tiếp. điều đó có nghĩa là nếu y tá nghỉ ngơi hôm nay, thì cô ấy phải làm việc vào ngày mai (1)

Độ dài của mỗi giai đoạn làm việc phải lớn hơn hoặc bằng K1 và nhỏ hơn hoặc bằng K2 (2)

Giám đốc bệnh viện muốn biết có bao nhiêu kế hoạch làm việc có thể thỏa mãn những hạn chế trên?

Đầu vào

Đầu vào bao gồm một dòng chứa 3 số nguyên dương N,K1,K2( N≤ 1000 ,K1<K2≤ 400 )

Đầu ra

Đầu ra chỉ bao gồm một số nguyên duy nhất M modulo 109+ 7 Ở đâu M là tổng số kế hoạch làm việc thỏa mãn các ràng buộc trên.

* 1. ROUTING

Đảo Superior là một hòn đảo đẹp như tranh vẽ và chỉ được phép đi xe đạp trên đảo. Do đó, có nhiều con đường xe đạp một chiều kết nối các điểm chụp ảnh tốt nhất khác nhau trên đảo. Để giúp du khách lên kế hoạch cho chuyến đi đến đảo, ủy ban du lịch muốn chỉ định các tuyến đường xe đạp khác nhau đi qua một số điểm chụp ảnh đẹp nhất trên đảo. Đưa ra một bản đồ của tất cả các con đường xe đạp trên đảo và một danh sách các điểm chụp ảnh tốt nhất được bao gồm trên mỗi trong ba tuyến đường theo kế hoạch (không được bao gồm các điểm không được liệt kê trong tuyến đường), vui lòng viết chương trình cho lập kế hoạch cho mỗi tuyến đường r sao cho khoảng cách trên mỗi tuyến là tối thiểu. Lưu ý rằng mỗi điểm chụp ảnh tốt nhất chỉ có thể xuất hiện tối đa một lần trên tuyến.

Đầu vào

Có hai phần đầu vào. Phần đầu tiên của đầu vào cung cấp thông tin về các con đường xe đạp trên đảo. Dòng đầu tiên chứa hai số nguyênn và r , n ≤ 100 và r ≤ 10 , chỉ ra rằng có n điểm chụp ảnh tốt nhất trên đảo và có rcác tuyến đường được lên kế hoạch. Tiếp theon dòng (dòng 2 đến dòng n + 1) chứa đựng n × n số nguyên (n dòng với n số nguyên trên mỗi dòng), trong đó j số nguyên -th trên dòng Tôi biểu thị khoảng cách từ điểm chụp ảnh tốt nhất tôi - 1 đến điểm chụp ảnh đẹp nhất j; khoảng cách là từ 0 đến 10, trong đó 0 chỉ ra rằng không có đường một chiều đi từ điểm chụp ảnh tốt nhấttôi - 1 phát hiện j.

Phần thứ hai của đầu vào có r dòng, biểu thị r tuyến đường tham quan được lên kế hoạch. Mỗi dòng liệt kê các điểm dừng chụp ảnh tốt nhất được đưa vào tuyến đường đó. Các số nguyên trên mỗi dòng biểu thị các điểm dừng chụp ảnh được đề xuất trên tuyến tham quan cụ thể đó. Số nguyên đầu tiên trên dòng là điểm bắt đầu của tuyến và số nguyên cuối cùng là điểm dừng cuối cùng trên tuyến. Tuy nhiên, các điểm dừng ở giữa có thể được truy cập theo bất kỳ thứ tự nào.

Đầu ra

Đầu ra r số nguyên trên r các dòng (một số nguyên trên mỗi dòng) cho biết khoảng cách của mỗi rtuyến đường quy hoạch. Nếu một tuyến đường là không thể, đầu ra '0'.

#include<bits/stdc++.h>

using namespace std;

int k;

int n, r;

int a[1000][1000];

int last, best, kq, mark[1000], routing[1000];

void Try(int i, int L, int sum){

if (L > k)

{

best = sum;

return;

}

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

{

if (a[i][j] && !mark[j] && (j != last || L == k))

{

mark[j] = 1;

if (sum + a[i][j] < best)

{

Try(j, L+1, sum + a[i][j]);

}

mark[j] = 0;

}

}

}

int main(){

freopen("input.txt", "r", stdin);

cin >> n >> r;

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

{

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

{

cin >> a[i][j];

}

}

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

{

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

{

mark[j] = 1;

}

int first = -1;

k = 0;

while (1)

{

int temp;

cin >> temp;

if(k > 0) mark[temp] = 0;

++k;

last = temp;

if (first == -1)

{

first = temp;

}

if(getchar() == '\n') break;

}

best = INT32\_MAX;

Try(first, 2, 0);

if (best > 1e8)

{

best = 0;

}

routing[i] = best;

}

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

{

cout << routing[i] << endl;

}

}

1. Training 4
   1. PIE

Sinh nhật của tôi sắp đến và theo truyền thống tôi đang phục vụ bánh. Không chỉ một chiếc bánh, không, tôi có một số N trong số chúng, với nhiều khẩu vị khác nhau và có kích cỡ khác nhau. F trong số những người bạn của tôi đang đến bữa tiệc của tôi và mỗi người trong số họ đều có một miếng bánh. Đây phải là một miếng của một chiếc bánh, không phải là một vài miếng nhỏ vì nó trông lộn xộn. Mảnh này có thể là một toàn bộ chiếc bánh mặc dù. Bạn bè của tôi rất khó chịu và nếu một trong số họ nhận được một phần lớn hơn những người khác, họ bắt đầu phàn nàn. Do đó, tất cả chúng nên có các miếng có kích thước bằng nhau (nhưng không nhất thiết phải có hình dạng bằng nhau), ngay cả khi điều này dẫn đến một số chiếc bánh bị hỏng (tốt hơn là làm hỏng bữa tiệc). Tất nhiên, tôi cũng muốn một miếng bánh cho riêng mình, và miếng đó cũng phải có cùng kích cỡ. Kích thước mảnh lớn nhất có thể tất cả chúng ta có thể nhận được là gì? Tất cả các bánh có hình trụ và chúng đều có cùng chiều cao 1, nhưng bán kính của bánh có thể khác nhau.

Đầu vào

Một dòng có số nguyên dương: số lượng test. Sau đó cho từng trường hợp thử nghiệm:

Một dòng có hai số nguyên N và F với 1 ≤ N, F≤ 10 000: số lượng bánh và số lượng bạn bè. 25 % các bài kiểm tra có N, F≤ 10 và 25 % kiểm tra có giải pháp ≤ 0,1.

Một dòng với N số nguyên rTôi với 1 ≤rTôi≤ 10 000: bán kính của bánh nướng.

Đầu ra

Đối với mỗi trường hợp thử nghiệm, hãy xuất một dòng có âm lượng lớn nhất có thể V sao cho tôi và bạn bè của tôi đều có thể có được một miếng bánh có kích thước V. Câu trả lời phải được đưa ra dưới dạng số dấu phẩy làm tròn thành 6 chữ số sau dấu phẩy động (lỗi chấp nhận ≤10- 6).

#include<bits/stdc++.h>

using namespace std;

int N, F;

double pi = 3.141592;

int a[10021];

int V = 0;

double ans[100];

int P(double v){

int t = 0;

for (int i = 1; i <= N; i++)

{

t += floor(pi \* a[i] \*a[i] / v);

}

return t >= F+1;

}

int main(){

int testcase;

// freopen("input.txt", "r", stdin);

cin >> testcase;

for(int j = 0; j < testcase;j++)

{

cin >> N >> F;

for (int i = 1; i <= N; i++)

{

cin >> a[i];

}

double L = 0, H = INT32\_MAX;

while(H - L > 1e-7){

double M = (H+L) / 2;

if(P(M)) L = M;

else

{

H = M;

}

}

ans[j] = H;

}

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

{

printf("%.6f\n", ans[i]);

}

}

* 1. EKO

Lumberjack Mirko cần phải chặt xuống Mmét gỗ. Đó là một công việc dễ dàng đối với anh ta vì anh ta có một máy cắt gỗ mới tiện lợi có thể hạ gục những khu rừng như lửa rừng. Tuy nhiên, Mirko chỉ được phép chặt một hàng cây.

Máy của Mirko hoạt động như sau: Mirko đặt tham số chiều cao H (tính bằng mét), và cỗ máy nâng một lưỡi cưa khổng lồ lên độ cao đó và cắt đứt tất cả các bộ phận của cây cao hơn H (tất nhiên, cây không cao hơn Hmét vẫn còn nguyên). Mirko sau đó lấy những phần bị cắt đi. Ví dụ: nếu hàng cây chứa các cây có chiều cao 20, 15, 10 và 17 mét và Mirko nâng lưỡi cưa của mình lên 15 mét, chiều cao cây còn lại sau khi cắt sẽ lần lượt là 15, 15, 10 và 15 mét , trong khi Mirko sẽ lấy 5 mét khỏi cây thứ nhất và 2 mét khỏi cây thứ tư (tổng cộng 7 mét gỗ).

Mirko có đầu óc sinh thái, vì vậy anh không muốn chặt gỗ nhiều hơn mức cần thiết. Đó là lý do tại sao anh ta muốn đặt lưỡi cưa của mình càng cao càng tốt. Giúp Mirko tìm chiều cao nguyên tối đa của lưỡi cưa mà vẫn cho phép anh ta cắt ít nhấtM mét gỗ.

Đầu vào

Dòng đầu tiên chứa hai số nguyên dương được phân tách bằng dấu cách, N (số lượng cây, 1 ≤ N≤ 1 000 000) và M (Số lượng gỗ cần thiết của Mirko, 1 ≤ M≤ 2 000 000 000).

Dòng đầu vào thứ hai chứa Nsố nguyên dương cách nhau không gian nhỏ hơn 1 000 000 000, chiều cao của mỗi cây (tính bằng mét). Tổng của tất cả các độ cao sẽ vượt quáM, do đó Mirko sẽ luôn có thể có được số lượng gỗ cần thiết.

Đầu ra

Dòng đầu ra đầu tiên và duy nhất phải chứa cài đặt chiều cao yêu cầu

#include<bits/stdc++.h>

using namespace std;

long long int N, M;

vector<long long int> tree;

int main(){

// freopen("input.txt", "r", stdin);

ios\_base::sync\_with\_stdio(0);

cin >> N >> M;

long long int lonnhat = 0, h = 0;

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

{

long long int temp;

cin >> temp;

tree.push\_back(temp);

lonnhat = max(lonnhat,temp);

}

long long int beg, end, mid;

long long int cutted;

beg = 0;

end = lonnhat;

while ( beg <= end ) {

mid = ( beg + end ) / 2;

cutted = 0;

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

if ( tree[ i ] - mid > 0 ) {

cutted += tree[ i ] - mid;

}

}

if ( cutted > M ) {

beg = mid + 1;

if ( mid > h ) {

h = mid;

}

}

else if ( cutted < M ) {

end = mid - 1;

}

else {

h = mid;

break;

}

}

cout << h;

return 0;

}

* 1. FIBWORDS

The Fibonacci word sequence of bit strings is defined as:

F(n) =







0 if n = 0

1 if n = 1

F(n − 1) + F(n − 2) if n ≥ 2

Here denotes concatenation of strings. The first few elements are:

n F(n)

0 0

1 1

2 10

3 101

4 10110

5 10110101

6 1011010110110

7 101101011011010110101

8 1011010110110101101011011010110110

9 1011010110110101101011011010110110101101011011010110101

Given a bit pattern p and a number n, how often does p occur in F(n)?

Input

The first line of each test case contains the integer n (0 ≤ n ≤ 100). The second line contains the bit

pattern p. The pattern p is nonempty and has a length of at most 100 000 characters.

Output

For each test case, display its case number followed by the number of occurrences of the bit pattern p in

F(n). Occurrences may overlap. The number of occurrences will be less than

2^63

#include <bits/stdc++.h>

using namespace std;

vector<string> f;

vector<string> g, h;

string p;

long long saveCal[101];

long long cal(int n){

if (saveCal[n] != -1) return saveCal[n];

if (n <= 1) return p == f[n];

long long ans = cal(n-1) + cal(n-2);

string s = g[n-1] + h[n-2];

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

int ok = 1;

for (int j = 0; j < p.size(); ++j){

if (p[j] != s[i+j]){

ok = false;

break;

}

}

ans += ok;

}

saveCal[n] = ans;

return ans;

}

main(){

f.push\_back("0");

f.push\_back("1");

for (int i = 2; f.back().size() < 100000; ++i){

f.push\_back(f.back() + f[f.size() - 2]);

}

int n, t = 0;

while (cin >> n >> p){

g.resize(n+1), h.resize(n+1);

//tinh g va h

int k = p.size()-1;

for (int i = 0; i < f.size() && i <= n; ++i){

g[i] = (f[i].size() <= k) ? f[i] : string(f[i].end()-k, f[i].end());

}

for (int i = f.size(); i <= n; ++i) g[i] = g[i-2];

for (int i = 0; i < f.size() && i <= n; ++i){

h[i] = (f[i].size() <= k) ? f[i] : string(f[i].begin(), f[i].begin()+k);

}

for (int i = f.size(); i <= n; ++i) h[i] = h[i-2];

for (int i = 0; i <= n; ++i) saveCal[i] = -1;

printf("Case %d: %lld\n", ++t, cal(n));

}

}

1. Training 5
   1. MARBLE

Phong là một nhà điêu khắc, ông có một tấm đá cẩm thạch hình chữ nhật kích thước W×H. Ông ta muốn cắt tấm đá thành các miếng hình chữ nhật kích thước W1×H1,W2×H2,…,WN×HN. Ông ta muốn cắt đến tối đa các mẫu kích thước có thể. Tấm đá có những vân đá cho nên không thể xoay khi sử dụng, có nghĩa là không thể cắt ra miếng B×A thay cho miếng A×B trừ khi A=B. Các miếng phải được cắt tại các điểm nguyên trên hàng cột và mỗi nhát cắt phải cắt đến hết hàng hoặc hết cột. Sau khi cắt sẽ còn lại những mẩu đá còn thừa bỏ đi, nghĩa là những mẩu đá không thể cắt thành miếng kích thước cho trước nào.

Yêu cầu: Hãy tìm cách cắt sao cho còn ít nhất diện tích đá thừa bỏ đi.

Hình dưới minh họa cách cắt các phiến đá trong ví dụ với diện tích thừa ít nhất tìm được là 10.

Famous sculptor Phong is making preparations to build a marvelous monument. For this purpose he needs rectangular marble plates of sizes W1×H1,W2×H2,…,WN×HN.

Recently, Phong has received a large rectangular marble slab. He wants to cut the slab to obtain plates of the desired sizes. Any piece of marble (the slab or the plates cut from it) can be cut either horizontally or vertically into two rectangular plates with integral widths and heights, cutting completely through that piece. This is the only way to cut pieces and pieces cannot be joined together. Since the marble has a pattern on it, the plates cannot be rotated: if Phong cuts a plate of size A×B then it cannot be used as a plate of size B×A unless A=B. He can make zero or more plates of each desired size. A marble plate is wasted if it is not of any of the desired sizes after all cuts are completed. Phong wonders how to cut the initial slab so that as little of it as possible will be wasted.

As an example, assume that in the figure below the width of the original slab is 21 and the height of the original slab is 11, and the desired plate sizes are 10×4,6×2,7×5, and 15×10. The minimum possible area wasted is 10, and the figure shows one sequence of cuts with total waste area of size 10.

Your task is to write a program that, given the size of the original slab and the desired plate sizes, calculates the minimum total area of the original slab that must be wasted.

Input

Dòng đầu tiên chứa hai số nguyên: W và H.

Dòng thứ hai chứa một số nguyên N. N dòng tiếp theo mỗi dòng chứa hai số nguyên Wi và Hi.

The first line of input contains two integers: first W, the width of the original slab, and then H, the height of the original slab;

The second line contains one integer N: the number of desired plate sizes. The following N lines contain the desired plate sizes. Each of these lines contains two integers: first the width Wi and then the height Hi of that desired plate size (1≤i≤N).

Output

Kết quả ghi ra duy nhất một số nguyên là tổng diện tích nhỏ nhất các miếng thừa bỏ đi.

For each dataset, write in one line a single integer: the minimum total area of the original slab that must be wasted.

Scoring

1≤W≤600,1≤H≤600,0<N≤200,1≤Wi≤W, and 1≤Hi≤H.

Có 50% số test ứng với W≤20,H≤20 và N≤5.

#include<bits/stdc++.h>

using namespace std;

int W,H,N;

int f[1000][1000], mark[1000][1000];

int main(){

cin >> W >> H >> N;

for (int i = 1; i <= N; i++)

{

int a,b;

cin >> a >> b;

mark[a][b] = 1;

}

for (int i = 1; i <= W; i++)

{

for (int j = 1; j <= H; j++)

{

f[i][j] = i \* j;

if (mark[i][j])

{

f[i][j] = 0;

}

for (int k = 1; k < i; k++)

{

f[i][j] = min(f[i][j], f[k][j]+ f[i-k][j]);

}

for (int k = 1; k < j; k++)

{

f[i][j] = min(f[i][j], f[i][k]+ f[i][j-k]);

}

}

}

cout << f[W][H];

}

* 1. GOLD MINING

Vương quốc ALPHA có n kho vàng nằm trên một đường thẳng và được đánh số 1 , 2 , . . . , n. Kho i có số lượng ai (ai là số nguyên không âm) và nằm ở tọa độ i (i = 1 , . . . , n). Vua ALPHA mở ra một cuộc cạnh tranh cho các thợ săn chịu trách nhiệm tìm một tập hợp các kho vàng có tổng lượng vàng lớn nhất với điều kiện khoảng cách giữa hai kho được chọn phải lớn hơn hoặc bằngL1 và nhỏ hơn hoặc bằng L2.

Đầu vào

Dòng 1 chứa n, L1và L2 (1 ≤ n ≤ 100000 , 1 ≤L1≤L2≤ n)

Dòng 2 chứa n số nguyên a1,a2, ... ,an

Đầu ra

Chỉ chứa một số nguyên duy nhất biểu thị tổng số vàng của các kho được chọn.

#include<bits/stdc++.h>

using namespace std;

int n, l1, l2;

#define ii pair<int, int>

int a[100010];

int ans =0 ;

int main(){

cin >> n>>l1 >> l2;

vector<int> gold(n);

for (int &a: gold)

{

cin >> a;

}

vector<long long> f(n);

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

{

f[i] = gold[i];

for (int j = max(0,i - l2); j <= min(i -l1, n-1); j++)

{

f[i] = max(f[i], gold[i] + f[j]);//12

}

}

ans = \*max\_element(f.begin(),f.end());

cout << ans;

}

* 1. MACHINE

Một kỹ sư cần lên lịch để máy chạy trong một số khoảng thời gian nhất định 1 , ... , n sản xuất một sản phẩm hóa học C. Mỗi thời kỳTôi được đại diện bởi một điểm thời gian bắt đầu Si và chấm dứt thời điểm tTôi (Si<ti). Do hạn chế về kỹ thuật, máy phải chạy đúng hai giai đoạn không trùng nhau (hai thời kỳ Tôi và j không trùng nhau nếu ti<Sj hoặc là tj<Si). Nếu máy được chạy trong khoảng thời gian Tôi, sau đó số lượng C nó sẽ sản xuất bằng với thời lượng của thời kỳ i (bằng với ti-Si). Giúp kỹ sư chọn hai khoảng thời gian không chồng lấp để chạy máy sao cho số lượng C sản xuất là tối đa.

Đầu vào

Dòng 1: chứa số nguyên dương n (2 ≤ n ≤ 2 ⋅106)

Hàng tôi + 1: chứa hai số nguyên dương STôi và tTôi (1 ≤Si<ti≤ 2 ⋅106)

Đầu ra

Đầu ra chỉ bao gồm một số nguyên duy nhất là số lượng sản phẩm CMáy sẽ sản xuất trong hai giai đoạn được chọn. Trong trường hợp không có giải pháp (không tồn tại hai khoảng thời gian không trùng nhau), đầu ra chứa giá trị -1.

#include<bits/stdc++.h>

using namespace std;

#define ii pair<int,int>

int n;

int main(){

cin >> n;

vector<int> right(2000001,-1e9), left(2000001,-1e9);

while (n--)

{

int a,b;

cin >> a >> b;

right[b] = max(right[b], b-a);

left[a] = max(left[a], b-a);

}

for(int i = 1; i < right.size(); i++) right[i] = max(right[i],right[i-1]);

for(int i = left.size() - 1; i >= 0; i--) left[i] = max(left[i],left[i+1]);

int ans = -1;

for (int i = 1; i < right.size()-1; i++)

{

ans = max(ans,right[i]+left[i+1]);

}

cout << ans;

}

* 1. INCREASE SUBSEQ

Cho một dãy số nguyên a1, ... ,an. Tính toán thứ tự dài nhất (các phần tử không nhất thiết phải liên tục) trong đó các phần tử theo thứ tự tăng dần.

#include<bits/stdc++.h>

using namespace std;

int main(){

int n;

cin >> n;

vector<int> seq(n);

int dem = 0;

for(int &a: seq) cin >> a;

vector<int> subseq(n);

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

{

subseq[i] = 1;

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

{

if (seq[i] > seq[j])

{

subseq[i] = max(subseq[i], subseq[j] + 1);

}

}

}

cout << \*max\_element(subseq.begin(),subseq.end());

}

* 1. LONGEST COMMON SUBSEQUENCE

Một chuỗi các chuỗi đã cho S= =S1, ... ,Sn có được bằng cách loại bỏ một số yếu tố từ S. Cho hai dãy số nguyênX= (X1, ... ,Xn) và Y= (Y1, ... ,Ym). Tìm chuỗi con chung dài nhất của X và Y.

Đầu vào

Dòng 1 chứa n và m (1 ≤ n , m ≤104)

Dòng 2 chứa X1, ... ,Xn

Dòng 3 chứa Y1, ... ,Ym

Đầu ra

Viết độ dài của chuỗi dài nhất của chuỗi đã cho

#include<bits/stdc++.h>

using namespace std;

int n,m;

int a[10010][10010];

int main(){

ios\_base::sync\_with\_stdio(0);

cin >> n >> m;

vector<int> x(n),y(m);

for(int &t: x) cin >> t;

for(int &t: y) cin >> t;

if (x[0] == y[0])

{

a[0][0] = 1;

}

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

{

if(x[0] == y[i]) a[0][i] = 1;

else

{

a[0][i] = a[0][i-1];

}

}

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

{

if(x[i] == y[0]) a[i][0] = 1;

else

{

a[i][0] = a[i-1][0];

}

}

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

{

for (int j = 1; j < m; j++)

{

a[i][j] = max(a[i - 1][j], a[i][j - 1]);

if (x[i] == y[j])

{

a[i][j] = max(a[i][j], 1 + a[i - 1][j - 1]);

}

}

}

printf("%d", a[n - 1][m - 1]);

}

* 1. TOWER

Có lẽ bạn đã nghe nói về truyền thuyết về Tháp Babylon. Ngày nay nhiều chi tiết của câu chuyện này đã bị lãng quên. Vì vậy, bây giờ, phù hợp với tính chất giáo dục của cuộc thi này, chúng tôi sẽ kể cho bạn toàn bộ câu chuyện:

Người babylon có n loại khối, và nguồn cung cấp không giới hạn của từng loại. Mỗi loại-i khối là một khối rắn hình chữ nhật với kích thước tuyến tính (xi,yi,zi). Một khối có thể được định hướng lại để bất kỳ hai trong ba chiều của nó xác định kích thước của cơ sở và kích thước khác là chiều cao. Họ muốn xây dựng tòa tháp cao nhất có thể bằng cách xếp chồng các khối. Vấn đề là, trong việc xây dựng một tòa tháp, một khối chỉ có thể được đặt trên đỉnh của một khối khác miễn là hai kích thước cơ sở của khối trên đều nhỏ hơn so với kích thước cơ sở tương ứng của khối dưới. Điều này có nghĩa là, ví dụ, các khối được định hướng để có các cơ sở có kích thước bằng nhau không thể được xếp chồng lên nhau.

Công việc của bạn là viết một chương trình xác định chiều cao của tòa tháp cao nhất mà người babylon có thể xây dựng với một tập hợp các khối nhất định.

Đầu vào

Tệp đầu vào sẽ chứa một hoặc nhiều trường hợp thử nghiệm. Dòng đầu tiên của mỗi trường hợp kiểm tra chứa một số nguyênn, biểu thị số lượng khối khác nhau trong tập dữ liệu sau. Giá trị tối đa chon là 30. Mỗi n dòng tiếp theo chứa ba số nguyên biểu thị các giá trị xi,yi và zi.

Đầu vào được kết thúc bằng giá trị 0 (0) cho n.

Đầu ra

Đối với mỗi trường hợp thử nghiệm, hãy in một dòng chứa số trường hợp (chúng được đánh số tuần tự bắt đầu từ 1) và chiều cao của tháp cao nhất có thể theo định dạng "Trường hợp trường hợp : chiều cao tối đa = chiều cao"

#include<bits/stdc++.h>

using namespace std;

#define ii pair<int, int>

#define iii pair<ii, int>

#define h second

#define width first.first

#define length first.second

int n;

int ans[100];

vector<iii> gach;

void cal(vector<int> f, int dem){

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

{

f[i] = gach[i].h;

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

{

if (gach[j].width > gach[i].width && gach[j].length > gach[i].length)

{

f[i] = max(f[i], f[j] + gach[i].h);

}

}

}

ans[dem] = \*max\_element(f.begin(), f.end());

};

int main(){

int dem = 1;

cin >> n;

while (n)

{

gach.clear();

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

{

int m,n,p;

cin >> m >> n >> p;

gach.push\_back(iii(ii(min(n,p),max(n,p)),m));

gach.push\_back(iii(ii(min(m,p),max(m,p)),n));

gach.push\_back(iii(ii(min(n,m),max(n,m)),p));

}

sort(gach.begin(),gach.end());

reverse(gach.begin(), gach.end());

vector<int> f(gach.size());

cal(f,dem++);

cin >> n;

}

for (int i = 1; i < dem; i++)

{

cout << "Case " << i << ": maximum height = " << ans[i] << endl;

}

}

1. Training 6
   1. CONNECTEDCOMPONENTS
   2. BIPARTIE GRAPH
   3. STRONGLY CONNECTED COMPONENT