# 第一部分 ZOJ

## zoj 1119 SPF

Background

Consider the two networks shown below. Assuming that data moves around these networks only between directly connected nodes on a peer-to-peer basis, a failure of a single node, 3, in the network on the left would prevent some of the still available nodes from communicating with each other. Nodes 1 and 2 could still communicate with each other as could nodes 4 and 5, but communication between any other pairs of nodes would no longer be possible.

Node 3 is therefore a Single Point of Failure (SPF) for this network. Strictly, an SPF will be defined as any node that, if unavailable, would prevent at least one pair of available nodes from being able to communicate on what was previously a fully connected network. Note that the network on the right has no such node; there is no SPF in the network. At least two machines must fail before there are any pairs of available nodes which cannot communicate.

Input

The input will contain the description of several networks. A network description will consist of pairs of integers, one pair per line, that identify connected nodes. Ordering of the pairs is irrelevant; 1 2 and 2 1 specify the same connection. All node numbers will range from 1 to 1000. A line containing a single zero ends the list of connected nodes. An empty network description flags the end of the input. Blank lines in the input file should be ignored.

Output

For each network in the input, you will output its number in the file, followed by a list of any SPF nodes that exist.

The first network in the file should be identified as ��Network #1��, the second as ��Network #2��, etc. For each SPF node, output a line, formatted as shown in the examples below, that identifies the node and the number of fully connected subnets that remain when that node fails. If the network has no SPF nodes, simply output the text ��No SPF nodes�� instead of a list of SPF nodes.

Example

Input

1 2

5 4

3 1

3 2

3 4

3 5

0

1 2

2 3

3 4

4 5

5 1

0

1 2

2 3

3 4

4 6

6 3

2 5

5 1

0

0

Output

Network #1

SPF node 3 leaves 2 subnets

Network #2

No SPF nodes

Network #3

SPF node 2 leaves 2 subnets

SPF node 3 leaves 2 subnets

参考答案

#include <iostream>

#include <algorithm>

#include <queue>

#include <math.h>

#include <stdio.h>

#include <string.h>

using namespace std;

#define MOD 1000000007

#define maxn 1010

bool G[maxn][maxn];

int low[maxn],pre[maxn];

int ans[maxn];

int dfst;

int node;

int child;

void init(){

dfst=0;

node=0;

child=0;

memset(G,0,sizeof(G));

memset(ans,0,sizeof(ans));

memset(pre,0,sizeof(pre));

}

int dfs(int u){

int lowu;

lowu=pre[u]=++dfst;

int v;

for(v=1;v<=node;v++){

if(G[u][v]){

if(!pre[v]){

int lowv=dfs(v);

lowu=min(lowu,lowv);

if(lowv>=pre[u]){

if(u!=1) ans[u]++;

else child++;

}

}

else lowu=min(lowu,pre[v]);

}

}

return lowu;

}

int main()

{

int u,v;

int Case=0;

while(scanf("%d",&u),u){

init();

scanf("%d",&v);

node=max(u,v);

G[u][v]=G[v][u]=1;

while(scanf("%d",&u),u){

scanf("%d",&v);

node=max(node,v);

node=max(node,u);

G[u][v]=G[v][u]=1;

}

dfs(1);

int found=0;

if(child>1) ans[1]=child-1;

if(Case) printf("\n");

printf("Network #%d\n",++Case);

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

if(ans[i]){

found=1;

printf(" SPF node %d leaves %d subnets\n",i,ans[i]+1);

}

if(!found)

printf(" No SPF nodes\n");

}

return 0;

}

## zoj 1179 Finding Rectangles

Consider the point sets in figures 1a, 2a, and 3a. Using only those points as vertices, figures 1b, 2b, and 3b show all the rectangles that can be formed with horizontal and vertical sides. No rectangles can be formed from the points in figure 4.

Your task is to write a program that can find all rectangles that can be formed from a given set of points. The example input and output given below correspond to the figures above.

Input

The input contains one or more point sets, followed by a line containing the number 0 that signals the end of the file. Each point set begins with a line containing n, the number of points, and is followed by n lines that describe the points. Each point description contains a capital letter that is the label of the point, then a space, the horizontal coordinate, a space, and the vertical coordinate. Within each set, points labels occur in alphabetical order.

Note that since each point is labelled with a capital letter there can be at most 26 points. All coordinates are nonnegative integers less than 50. Points within a set are unique.

Output

The output for each point set starts with ``Point set ", followed by the number of the point set and a colon. If there are no rectangles, `` No rectangles" appears after the colon. If there are rectangles, they are listed starting on the next line. A blank precedes each rectangle. Each rectangle is given by its vertex labels, in clockwise order from the upper left, so the order is upper left, upper right, lower right, lower left. The rectangles are listed ten per line, except for the last line, where there may be as few as one. The rectangles are listed in alphabetical order.

Sample Input

7

A 1 1

B 2 1

C 3 1

D 2 3

E 3 3

F 1 4

G 3 4

8

B 1 1

D 2 1

F 4 1

J 4 4

L 2 4

M 2 3

N 4 3

P 1 2

12

A 1 5

B 2 5

C 1 4

D 2 4

E 1 3

F 2 3

G 1 2

H 2 2

I 1 1

J 2 1

K 1 0

L 2 0

5

B 1 1

D 2 1

L 2 4

N 2 3

P 1 2

0

Sample Output

Point set 1:

DECB FGCA

Point set 2:

LJFD LJNM MNFD

Point set 3:

ABDC ABFE ABHG ABJI ABLK CDFE CDHG CDJI CDLK EFHG

EFJI EFLK GHJI GHLK IJLK

Point set 4: No rectangles

参考答案

#include<cstdio>

#include<string>

#include<vector>

#include<algorithm>

#include<iostream>

using namespace std;

namespace

{

struct Point

{

char c;

int x, y;

};

bool cmp(const Point \*p1, const Point \*p2)

{

if (p1->x != p2->x)

return p1->x < p2->x;

else

return p1->y < p2->y;

}

bool valid(Point \*p1, Point \*p2, Point \*p3, Point \*p4)

{

return p1->x == p2->x && p3->x == p4->x && p1->y == p3->y

&& p2->y == p4->y;

}

}

int main()

{

int n, x, y, count = 0;

vector<Point\*> V;

vector<string> res;

char s[2];

while (scanf("%d", &n), n)

{

getchar();

V.clear();

res.clear();

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

{

scanf("%s %d %d", s, &x, &y);

Point \*p = new Point();

p->c = s[0];

p->x = x;

p->y = y;

V.push\_back(p);

}

sort(V.begin(), V.end(), cmp);

for (size\_t i = 0; i < V.size(); i++)

for (size\_t j = i + 1; j < V.size(); j++)

for (size\_t k = j + 1; k < V.size(); k++)

for (size\_t l = k + 1; l < V.size(); l++)

if (valid(V[i], V[j], V[k], V[l]))

{

string str;

str.push\_back(V[j]->c);

str.push\_back(V[l]->c);

str.push\_back(V[k]->c);

str.push\_back(V[i]->c);

res.push\_back(str);

}

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

printf("Point set %d:", ++count);

if (res.size())

{

for (size\_t i = 0; i < res.size(); i++)

{

if (i % 10 == 0)

putchar('\n');

cout << ' ' << res[i];

}

putchar('\n');

}

else

puts(" No rectangles");

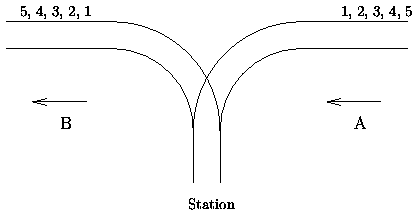
}

return 0;

}

## zoj 1259 Rails

There is a famous railway station in PopPush City. Country there is incredibly hilly. The station was built in last century. Unfortunately, funds were extremely limited that time. It was possible to establish only a surface track. Moreover, it turned out that the station could be only a dead-end one (see picture) and due to lack of available space it could have only one track.



The local tradition is that every train arriving from the direction A continues in the direction B with coaches reorganized in some way. Assume that the train arriving from the direction A has N <= 1000 coaches numbered in increasing order 1, 2, ..., N. The chief for train reorganizations must know whether it is possible to marshal coaches continuing in the direction B so that their order will be a1, a2, ..., aN. Help him and write a program that decides whether it is possible to get the required order of coaches. You can assume that single coaches can be disconnected from the train before they enter the station and that they can move themselves until they are on the track in the direction B. You can also suppose that at any time there can be located as many coaches as necessary in the station. But once a coach has entered the station it cannot return to the track in the direction A and also once it has left the station in the direction B it cannot return back to the station.

Input

The input consists of blocks of lines. Each block except the last describes one train and possibly more requirements for its reorganization. In the first line of the block there is the integer N described above. In each of the next lines of the block there is a permutation of 1, 2, ..., N. The last line of the block contains just 0.

The last block consists of just one line containing 0.

Output

The output contains the lines corresponding to the lines with permutations in the input. A line of the output contains Yes if it is possible to marshal the coaches in the order required on the corresponding line of the input. Otherwise it contains No. In addition, there is one empty line after the lines corresponding to one block of the input. There is no line in the output corresponding to the last ``null'' block of the input.

Sample Input

5

1 2 3 4 5

5 4 1 2 3

0

6

6 5 4 3 2 1

0

0

Sample Output

Yes

No

Yes

参考答案

#include<iostream>

#include<stack>

using namespace std;

int main()

{

int n,i,count;

int target[1000];

stack<int> sk;

while(cin>>n&&n!=0)

{

while(cin>>target[0]&&target[0])

{

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

cin>>target[i];

count=1;

i=0;

while(i<n)

{

if(count==target[i])

{

count++;

i++;

}

else

{

if(!sk.empty()&&sk.top()==target[i])

{

sk.pop();

i++;

}

else

{

if(count<=n)

sk.push(count++);

else

break;

}

}

}

if(i==n)

cout<<"Yes"<<endl;

else

cout<<"No"<<endl;

while(!sk.empty())

sk.pop();

}

cout<<endl;

}

return 0;

}

## zoj 1344 A Mazing Problem

In a maze of size M by N, there is a savage hound named "Fatdog". Fatdog keeps moving around, and will have a good bite at his victim as soon as anyone moves into his sightline. You are supposed to compute the minimum time taken to cross this maze alive, and Fatdog must be for sure avoided.

In every time unit, you may move one block to one of the upper, lower, left and right neighboring blocks, or you may stay where you are. Of course you must not move into the wall, neither must you walk right into Fatdog, and you must not fall into Fatdog's sightline as well. The entrance given is guaranteed to avoid Fatdog's sight.

At the mean time, Fatdog is not as smart as you are -- in every time unit, he either moves forward by one block, or stays and turns 90 degrees to the left or to the right.

Assume that in every time unit you and Fatdog are taking actions simultaneously.

Input

There are several test cases. For each test case:

The first line of input contains 2 integers M and N (0 < M,N <= 50) which indicating the size of a maze, provided that the origin is at the upper-left corner.

The second line contains 7 numbers: x1,y1,x2,y2,x3,y3,D where x1 and y1 are the coordinates of your starting position, x2 and y2 are your exit coordinates, x3 and y3 are the coordinates of the initial position of Fatdog (assuming that he is facing North at every beginning)

The next line contains D(0 < D <= 50) characters, representing Fatdog's actions. When he finishes these D actions, he will be back to his starting position and facing North, and then he will repeat these actions again and again forever (poor Fatdog...). These characters are:

'G' means moving one block forward

'L' means turning 90 degrees to the left

'R' means turning 90 degrees to the right

Then there will be M lines followed, with N characters in each line, describing the maze:

'.' represents an empty block, meaning there is a path.

'\*' represents a block of wall.

The input is finished by a pair of 0's as M and N.

Output

For each test case, the first line of output must be in the form "Case d:" where d is the case number (start counting from 1). The second line must be either "Minimal time is: d" where d is the minimum crossing time, or "No way out" meaning that it is impossible to cross the maze alive.

Two test cases must be separated by a blank line.

Sample Input

5 3

3 1 3 3 4 2 8

GGRRGGRR

\*\*\*

\*.\*

...

\*.\*

\*\*\*

5 3

3 1 3 3 2 2 2

LR

\*\*\*

\*.\*

...

\*.\*

\*\*\*

3 3

2 1 2 3 2 2 4

RRRR

\*\*\*

...

\*\*\*

0 0

Sample Output

Case 1:

Minimal time is: 3

Case 2:

Minimal time is: 2

Case 3:

No way out

参考答案

#include<iostream>

#include<queue>

#include<cstdio>

#include<cctype>

#include<string.h>

using namespace std;

enum {

SIZ = 54,

};

struct Pnt{

unsigned x,y;

int m;

};

int move[5][2] = {

{-1,0},{0,1},{1,0},{0,-1},{0,0}

};

int M, N, D;

int stat[SIZ][SIZ][SIZ];

int mat[SIZ][SIZ];

Pnt st, ex, dog;

int fun(){

Pnt one, nex;

queue<Pnt> q;

st.m = 0;

stat[0][st.x][st.y] = 1;

q.push(st);

while(!q.empty()){

one = q.front(); q.pop();

if(one.x == ex.x && one.y == ex.y){

return stat[one.m][one.x][one.y]-1;

}

nex.m = (one.m + 1)%D ;

for(int i=0; i<5; i++){

nex.x = one.x + move[i][0];

nex.y = one.y + move[i][1];

if(nex.x>=M||nex.y>=N

||mat[nex.x][nex.y]

||stat[nex.m][nex.x][nex.y] != 0)

continue;

stat[nex.m][nex.x][nex.y] = stat[one.m][one.x][one.y] + 1;

q.push(nex);

}

}

return -1;

}

void forbid(int s, const Pnt &p){

unsigned x, y, d;

x = p.x; y = p.y; d = p.m;

while(x<M && y<N && mat[x][y] ==0){

stat[s][x][y] = -1;

x += move[d][0];

y += move[d][1];

}

}

int readIn(){

static char buf[SIZ];

scanf("%d%d",&M,&N);

if(M + N ==0) return 0;

memset(stat, 0, sizeof(stat));

int x, y;

scanf("%d%d",&x,&y);

st.x = x-1, st.y = y-1;

scanf("%d%d",&x,&y);

ex.x = x-1, ex.y = y-1;

scanf("%d%d",&x,&y);

dog.x = x-1, dog.y = y-1; dog.m = 0;

scanf("%d %s ", &D, buf);

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

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

scanf(" %c ", &mat[i][j]);

mat[i][j] = (mat[i][j]=='\*');

}

}

x = dog.x, y = dog.y;

forbid(0, dog);

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

switch(buf[i]){

case 'G':

dog.x += move[dog.m][0];

dog.y += move[dog.m][1];

break;

case 'L':

dog.m = (dog.m + 3) % 4;

break;

case 'R':

dog.m = (dog.m + 1) % 4;

break;

}

forbid(i+1, dog);

}

if(x == dog.x && y==dog.y && dog.m == 0){

;

} else {

D++;

}

return 1;

}

int main(){

int tst = 0, t;

while(readIn() > 0){

if(tst++) printf("\n");

printf("Case %d:\n", tst);

t = fun();

if(t < 0){

printf("No way out\n");

} else {

printf("Minimal time is: %d\n", t);

}

}

return 0;

}

## zoj 1418 Lazy Math Instructor

A math instructor is too lazy to grade a question in the exam papers in which students are supposed to produce a complicated formula for the question asked. Students may write correct answers in different forms which makes grading very hard. So, the instructor needs help from computer programmers and you can help.

You are to write a program to read different formulas and determine whether or not they are arithmetically equivalent.

Input

The first line of the input contains an integer N (1 <= N <= 20) that is the number of test cases. Following the first line, there are two lines for each test case. A test case consists of two arithmetic expressions, each on a separate line with at most 80 characters. There is no blank line in the input. An expression contains one or more of the following:

Single letter variables (case insensitive).

Single digit numbers.

Matched left and right parentheses.

Binary operators +, - and \* which are used for addition, subtraction and multiplication respectively.

Arbitrary number of blank or tab characters between above tokens.

Note: Expressions are syntactically correct and evaluated from left to right with equal precedence (priority) for all operators. The coefficients and exponents of the variables are guaranteed to fit in 16-bit integers.

Output

Your program must produce one line for each test case. If input expressions for each test data are arithmetically equivalent, "YES", otherwise "NO" must be printed as the output of the program. Output should be all in upper-case characters.

Sample Input

3

(a+b-c)\*2

(a+a)+(b\*2)-(3\*c)+c

a\*2-(a+c)+((a+c+e)\*2)

3\*a+c+(2\*e)

(a-b)\*(a-b)

(a\*a)-(2\*a\*b)-(b\*b)

Sample Output

YES

YES

NO

参考答案

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

int count1,count2;

int opnd[200];

char optr[200];

char opp[9] = {'+','-','\*','/','(',')','#','\0'};

int h[7][7] = {{1,1,-1,-1,-1,1,1},{1,1,-1,-1,-1,1,1},{1,1,1,1,-1,1,1},

{1,1,1,1,-1,1,1},{-1,-1,-1,-1,-1,0,-2},{1,1,1,1,-2,1,1},{-1,-1,-1,-1,-1,-2,0}};

void push1(int x)

{

opnd[count1++] = x;

}

void push2(int x)

{

optr[count2++] = x;

}

int pop1()

{

count1--;

return opnd[count1];

}

char pop2()

{

count2--;

return optr[count2];

}

int computer(int x,int y, char ch)

{

switch(ch)

{

case '+':return x+y;

case '-':return x-y;

case '\*':return x\*y;

case '/':return x/y;

}

}

int record(char a,char b)

{

int ta,tb;

int i;

for(i=0; i<7; i++)

{

if( a == opp[i] )

ta = i;

if( b == opp[i] )

tb = i;

}

return h[ta][tb];

}

int result(char fir[],int plus)

{

char c,ch;

int x,y,i = 0;

count1 = 0;

count2 = 0;

push2('#');

while( fir[i]!='#' || optr[count2-1]!='#')

{

c = fir[i];

if( isdigit(c) )

{

push1(c-'0');

i++;

continue;

}

if( isalpha(c) )

{

push1(c-'a'+plus);

i++;

continue;

}

if(c == '+' || c== '-'||c=='\*' || c=='('||c==')'||c=='#')

{

switch( record(optr[count2-1],c) )

{

case -1: push2(c); i++; break;

case 0 : pop2(); i++; break;

case 1 : ch = pop2();

y = pop1();

x = pop1();

push1(computer(x,y,ch));

break;

}

}

else

i++;

}

return opnd[count1-1];

}

int main(void)

{

char c,ch;

int length,i,x,y,plus,result1,result2,ncases;

int len1,len2,flag;

char fir[90],sec[90];

scanf("%d",&ncases);

getchar();

while(ncases--)

{

gets(fir);

gets(sec);

len1 = strlen( fir );

len2 = strlen( sec );

fir[len1] = '#';

sec[len2] = '#';

plus = 10;

result1= result(fir,plus);

result2= result(sec,plus);

if( result1 != result2 )

printf("NO\n");

else

printf("YES\n");

}

return 0;

}

## zoj 1507 Crazy Search

Many people like to solve hard puzzles some of which may lead them to madness. One such puzzle could be finding a hidden prime number in a given text. Such number could be the number of different substrings of a given size that exist in the text. As you soon will discover, you really need the help of a computer and a good algorithm to solve such a puzzle.

Your task is to write a program that given the size, N, of the substring, the number of different characters that may occur in the text, NC, and the text itself, determines the number of different substrings of size N that appear in the text.

As an example, consider N=3, NC=4 and the text "daababac". The different substrings of size 3 that can be found in this text are: "daa", "aab", "aba", "bab", "bac". Therefore, the answer should be 5.

Input

The first line of input consists of two numbers, N and NC, separated by exactly one space. This is followed by the text where the search takes place. You may assume that the maximum number of substrings formed by the possible set of characters does not exceed 16 Millions.

Output

The program should output just an integer corresponding to the number of different substrings of size N found in the given text.

This problem contains multiple test cases!

The first line of a multiple input is an integer N, then a blank line followed by N input blocks. Each input block is in the format indicated in the problem description. There is a blank line between input blocks.

The output format consists of N output blocks. There is a blank line between output blocks.

Sample Input

1

3 4

daababac

Sample Output

5

代码：

#include <iostream>

#include <stdio.h>

#include <cstring>

using namespace std;

const int N=16000005;

const int NUM=300;

bool hash[N];

int m[NUM];

char str[1000000];

int main()

{

int tt;

cin>>tt;

while(tt--)

{

int n,nc,i,j,sum,seed=0,ans=0;

memset(hash,false,sizeof(hash));

memset(m,0,sizeof(m));

memset(str,'\0',sizeof(str));

cin>>n>>nc>>str;

for(i=0; '\0' != str[i]; ++i)

{

if(!m[str[i]])

m[str[i]]=++seed;

if(seed == nc)

break;

}

int len=strlen(str);

for(i=0; i<=len-n; ++i)

{

sum=0;

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

sum=sum\*nc+m[str[i+j]]-1;

if(!hash[sum])

{

hash[sum]=true;

++ans;

}

}

if(tt)

cout<<ans<<endl<<endl;

else

cout<<ans;

}

return 0;

}

## zoj 1563 Pearls

In Pearlania everybody is fond of pearls.

在皮尔兰尼亚，每个人都喜欢珍珠。

One company, called The Royal Pearl, produces a lot of jewelry with pearls in it.

一家名为“皇家珍珠”的公司生产了许多带有珍珠的珠宝。

The Royal Pearl has its name because it delivers to the royal family of Pearlania.

皇家珍珠得名于它的名字，因为它提供给皮尔兰尼亚的皇室。

But it also produces bracelets and necklaces for ordinary people.

但它也为普通人生产手镯和项链。

Of course the quality of the pearls for these people is much lower then the quality of pearls for the royal family.

当然，对于这些人来说，珍珠的质量要比皇室的珍珠质量低得多。

In Pearlania pearls are separated into 100 different quality classes.

在皮尔兰尼亚，珍珠被分成了上百种不同的质量等级。

A quality class is identified by the price for one single pearl in that quality class.

在质量等级中，一个质量等级是由一个单一的珍珠价格来决定的。

This price is unique for that quality class and the price is always higher then the price for a pearl in a lower quality class.

这个价格对于质量等级来说是独一无二的，而且价格总是高于质量较低等级的珍珠的价格。

Every month the stock manager of The Royal Pearl prepares a list with the number of pearls needed in each quality class.

每个月，皇家珍珠公司的股票经理都会准备一份清单，上面列出了每一个质量等级所需的珍珠数量。

The pearls are bought on the local pearl market.

这些珍珠是在当地的珍珠市场上买的。

Each quality class has its own price per pearl, but for every complete deal in a certain quality class one has to pay an extra amount of money equal to ten pearls in that class.

每一个质量等级都有它自己的每颗珍珠的价格，但是对于每一个在质量等级上的完整的交易，你必须支付额外的一笔钱，相当于这门课上的10颗珍珠。

This is to prevent tourists from buying just one pearl.

这是为了防止游客只买一颗珍珠。

Also The Royal Pearl is suffering from the slow-down of the global economy.

此外，皇家珍珠也正遭受着全球经济放缓的影响。

Therefore the company needs to be more efficient.

因此，公司需要提高效率。

The CFO (chief financial officer) has discovered that he can sometimes save money by buying pearls in a higher quality class than is actually needed.

首席财务官（首席财务官）发现，他有时可以通过购买比实际需要更高质量的珍珠来省钱。

No customer will blame The Royal Pearl for putting better pearls in the bracelets, as long as the prices remain the same.

只要价格保持不变，没有顾客会责怪皇家珍珠把更好的珍珠放在手镯上。

For example 5 pearls are needed in the 10 Euro category and 100 pearls are needed in the 20 Euro category. That will normally cost: (5+10)\*10 + (100+10)\*20 = 2350 Euro.

例如，在10欧元的类别中需要5颗珍珠，在20欧元的类别中需要100颗珍珠。这通常要花费：（5+10）10+（100+10）20=2350欧元。

Buying all 105 pearls in the 20 Euro category only costs: (5+100+10)\*20 = 2300 Euro.

在20欧元的类别中购买所有105颗珍珠，只需要花费：（5+100+10）20=2300欧元。

The problem is that it requires a lot of computing work before the CFO knows how many pearls can best be bought in a higher quality class.

问题是，在首席财务官知道有多少珍珠可以在更高质量的课堂上买到，这需要大量的计算工作。

You are asked to help The Royal Pearl with a computer program.

你被要求用电脑程序帮助皇家珍珠。

Given a list with the number of pearls and the price per pearl in different quality classes, give the lowest possible price needed to buy everything on the list. Pearls can be bought in the requested, or in a higher quality class, but not in a lower one.

给定一份清单，上面有珍珠的数量和不同质量等级的珍珠价格，给出了购买清单上所有商品所需的最低价格。珍珠可以在被要求的地方买到，或者在更高质量的等级中，但不是在较低的等级。

Input

The first line of the input contains the number of test cases. Each test case starts with a line containing the number of categories c (1 <= c <= 100). Then, c lines follow, each with two numbers ai and pi. The first of these numbers is the number of pearls ai needed in a class (1 <= ai <= 1000). The second number is the price per pearl pi in that class (1 <= pi <= 1000). The qualities of the classes (and so the prices) are given in ascending order. All numbers in the input are integers.

输入的第一行包含测试用例的数量。每个测试用例都以一个包含c类（1<=c<=100）的行开始。然后，c行跟随，每一行有两个数字ai和pi。第一个数字是一个类需要的珍珠数量（1<=ai<=1000）。第二个数字是这个类中每一个珍珠pi的价格（1<=pi<=1000）。类的质量（以及价格）是按升序给出的。输入中的所有数字都是整数。

Output

For each test case a single line containing a single number: the lowest possible price needed to buy everything on the list.

对于每个测试用例，一个包含单个数字的单行：在列表中购买所有东西所需的最低价格。

Sample Input

2

2

100 1

100 2

3

1 10

1 11

100 12

Sample Output

330

1344

## zoj 1810 The Gourmet Club

The gourmet club of ACM City has 16 members. They contracted the proprietor of the local French restaurant Chateau Java to arrange dinner parties for 5 consecutive evenings. They asked to be seated around 4 tables, 4 persons per table. They also stipulated that during the 5 evenings, every member of the club will share a table exactly once with each member of the club. Mr. I.B. Emm, the restaurateur, assigned his Maitre D' the task of scheduling the seating for the 5 evenings. On the first evening, the Maitre D' seated the members as they arrived and recorded their seating. Each subsequent evening, he carefully planned the seating to match the requirement that no member will be dining twice with some other member. Unfortunately, the Maitre D' disappeared on the morning of the fourth evening. Mr. Emm was left only with his notes which included the recorded seating arrangements during the previous 3 evenings. As he was trying to schedule the seating for the remaining evenings, it dawned on him that this task may not be that easy. He is asking for your help to try and see whether the remaining two evenings can be scheduled. The following is a sample of the Maitre D's seating arrangements during the first 3 evenings:

ABCD EFGH IJKL MNOP

AEIM BFJN CGKO DHLP

AFKP BGLM CHIN DEJO

The members of the gourmet club were identified by the letters A,B,C,...,P.

Each line represents one evening of seating with each set of four letters a single table. Thus on the first evening A dines with B, C and D etc. Write a program that will read from the input file the seating arrangement of the first three evenings and will either complete the schedule or determine that the Maitre D' screwed up.

Input

Each data set will be 3 lines. Each line will consist of four blocks, each 4 letters long. All letters will be in upper case. Blocks will be separated by "white space". Data sets will be separated by blank lines.

Output

For a successful schedule, echo the input and add two lines showing the successful schedule. If it is not possible to complete the schedule, do not echo the input, but print "It is not possible to complete this schedule." Separate output for each data set with a blank line.

Sample Input

ABCD EFGH IJKL MNOP

AEIM BFJN CGKO DHLP

AFKP BGLM CHIN DEJO

Sample Output

It is not possible to complete this schedule.

参考答案

#include <stdio.h>

#include <string.h>

#include <iostream>

using namespace std;

const int N = 25;

char str[N][N];

int g[N][N], sn, tmp[N], tn;

void build(int i) {

for (int j = 0; j < 4; j++)

for (int k = 0; k < 4; k++)

g[str[i][j] - 'A'][str[i][k] - 'A'] = g[str[i][k] - 'A'][str[i][j] - 'A'] = 1;

}

bool check1(int i, int j) {

int num = 0;

if (g[j][i]) return false;

for (int k = 0; k < 16; k++)

if (g[i][k] + g[j][k] == 0) {

str[12][num + 2] = k + 'A';

num++;

}

return num == 2;

}

bool check2(int num) {

tn = 0;

for (int i = 0; i < 16; i++) {

if (g[num][i]) continue;

tmp[tn++] = i;

}

return tn == 3;

}

bool solve() {

int i;

for (i = 1; i < 16; i++) {

if (g[0][i]) continue;

if (!check1(0, i)) return false;

str[12][0] = 'A'; str[12][1] = i + 'A';

build(sn); sn++;

break;

}

for (i = 0; i < 4; i++) {

if (!check2(str[12][i] - 'A')) return false;

str[16 + i][0] = str[12][i];

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

str[16 + i][1 + j] = tmp[j] + 'A';

build(16 + i);

sn++;

}

for (i = 1; i < 4; i++) {

if (!check2(str[16][i] - 'A')) return false;

str[12 + i][0] = str[16][i];

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

str[12 + i][1 + j] = tmp[j] + 'A';

build(12 + i);

sn++;

}

return true;

}

int main() {

int bo = 0, i;

memset(str, 0, sizeof(str));

while (~scanf("%s", str[0])) {

memset(g, 0, sizeof(g));

for (i = 0; i < 16; i++)

g[i][i] = 1;

build(0); sn = 12;

for (i = 1; i < 12; i++) {

scanf("%s", str[i]);

build(i);

}

if (!solve()) printf("It is not possible to complete this schedule.\n");

else {

for (int i = 0; i < 20; i++) {

for (int j = 0; j < 4; j++)

printf("%c", str[i][j]);

if ((i + 1) % 4 == 0)

printf("\n");

else printf("\t");

}

}

printf("\n");

}

return 0;

}

## zoj 1861 Gas Station Numbers

Many gas stations use plastic digits on an illuminated sign to indicate prices. When there is an insufficient quantity of a particular digit, the attendant may substitute another one upside down.

The digit "6" looks much like "9" upside down. The digits "0", "1" and "8" look like themselves. The digit "2" looks a bit like a "5" upside down (well, at least enough so that gas stations use it).

Due to rapidly increasing prices, a certain gas station has used all of its available digits to display the current price. Fortunately, this shortage of digits need not prevent the attendant from raising prices. She can simply rearrange the digits, possibly reversing some of them as described above.

Your job is to compute, given the current price of gas, the next highest price that can be displayed using exactly the same digits.

Input

The input consists of several lines, each containing between 2 and 30 digits (to account for future prices) and a decimal point immediately before the last digit. There are no useless leading zeroes; that is, there is a leading zero only if the price is less than 1.

Output

You are to compute the next highest price that can be displayed using the same digits and the same format rules. An input line containing a decimal point alone terminates the input. If the price cannot be raised, print "The price cannot be raised."

Sample Input

65.2

76.7

77.7

.

Sample Output

65.5

77.6

The price cannot be raised.

代码：

#include<iostream>

#include<vector>

#include<string.h>

#include<stdio.h>

#include<algorithm>

#include<cmath>

using namespace std;

enum {

SIZ = 40,

};

char rep[10] = {0, 1, 5, 3, 4, 2, 9, 7, 8, 6};

char dig[SIZ];

int len;

int fun(){

int i, t = dig[len], hi;

if(rep[t] > t){

dig[len] = rep[t];

return 1;

}

hi = t;

for(i=len-1; i>=0; i--){

t = dig[i];

if(t < hi || rep[t] > t){

int k=i;

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

t = dig[j];

if(t > dig[i] && t <= hi){

hi = t;

k = j;

}

if(rep[t] > dig[i] && rep[t] <= hi){

hi = rep[t];

dig[j] = rep[t];

k = j;

}

}

t = dig[i];

if(rep[t] > t && (hi < t || hi > rep[t])){

dig[i] = rep[t];

} else {

swap(dig[i], dig[k]);

}

for(k= i+1; k<=len; k++){

dig[k] = min(dig[k], rep[dig[k]]);

}

sort(dig+i+1, dig+len+1);

return 1;

}

hi = max(hi, t);

}

return 0;

}

void trim(){

len = 0;

for(int i=0; dig[i]; i++){

if(dig[i]!='.'){

dig[len++] = dig[i] - '0';

}

}

dig[len--] = 0;

}

void output(int s){

if(s <=0){

printf("The price cannot be raised.\n");

return ;

}

for(s=0; s<len; s++){

printf("%d", dig[s]);

}

printf(".%d\n", dig[len]);

}

int main(){

scanf("%s", dig);

while(dig[0]!='.'){

trim();

output(fun());

scanf("%s", dig);

}

return 0;

}

## zoj 1898 Discrete Logging

Given a prime P, 2 <= P < 2^31, an integer B, 2 <= B < P, and an integer N, 2 <= N < P, compute the discrete logarithm of N, base B, modulo P. That is, find an integer L such that

B^L == N (mod P)

Input

Read several lines of input, each containing P,B,N separated by a space, and for each line print the logarithm on a separate line.

Output

If there are several, print the smallest; if there is none, print "no solution".

The solution to this problem requires a well known result in number theory that is probably expected of you for Putnam but not ACM competitions. It is Fermat's theorem that states

B^(P-1) == 1 (mod P)

for any prime P and some other (fairly rare) numbers known as base-B pseudoprimes. A rarer subset of the base-B pseudoprimes, known as Carmichael numbers, are pseudoprimes for every base between 2 and P-1. A corollary to Fermat's theorem is that for any m

B^(-m) == B^(P-1-m) (mod P) .

Sample Input

5 2 1

5 2 2

5 2 3

5 2 4

5 3 1

5 3 2

5 3 3

5 3 4

5 4 1

5 4 2

5 4 3

5 4 4

12345701 2 1111111

1111111121 65537 1111111111

Sample Output

0

1

3

2

0

3

1

2

0

no solution

no solution

1

9584351

462803587

参考答案

#include<iostream>

#include<cstdio>

#include<algorithm>

#include<cstring>

#include<cmath>

#include<vector>

#include<queue>

#include<map>

#include<set>

#include<time.h>

#include<string>

#define cl(a,b) memset(a,b,sizeof(a))

#define max(x,y) ((x)>(y)?(x):(y))

#define min(x,y) ((x)<(y)?(x):(y))

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

#define REP1(i,a,b) for(int i=a;i<=b;++i)

#define REP2(i,a,b) for(int i=a;i>=b;--i)

#define MP make\_pair

#define LL long long

#define ULL unsigned long long

#define X first

#define Y second

#define MAXN 100050

using namespace std;

int p,b,n;

struct node

{

int a,id;

}q[MAXN];

bool cmp(node a,node b)

{

return a.a<b.a||(a.a==b.a&&a.id<b.id);

}

int qmod(int a,int b,int c)

{

int res=1;

while(b)

{

if(b&1)

res=((LL)res\*a)%c;

a=((LL)a\*a)%c;

b>>=1;

}

return res;

}

int main()

{

while(scanf("%d%d%d",&p,&b,&n)!=EOF){

int m=sqrt(p\*1.0);

while(m\*m<p)m++;

int res=1;

int tail=0;

int a=qmod(b,m,p);

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

{

q[tail].id=i;

q[tail++].a=res;

res=((LL)res\*b)%p;

}

sort(q,q+tail,cmp);

int h=0;

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

{

if(i==0||q[i].a!=q[i-1].a)

{

q[h++]=q[i];

}

}

int flag=0,ans;

res=n;

int ra=qmod(a,p-2,p);

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

{

int l=0,r=h-1;

while(r-l>1)

{

int mid=(l+r)>>1;

if(q[mid].a>res)r=mid;

else l=mid;

}

int pos=-1;

if(q[l].a==res)pos=l;

if(q[r].a==res)pos=r;

if(pos!=-1){

ans=i\*m+q[pos].id;

flag=1;

break;

}

res=((LL)res\*ra)%p;

}

if(flag==0)puts("no solution");

else

printf("%d\n",ans);

}

}

## zoj 2011 Secret Code

The Sarcophagus itself is locked by a secret numerical code. When somebody wants to open it, he must know the code and set it exactly on the top of the Sarcophagus. A very intricate mechanism then opens the cover. If an incorrect code is entered, the tickets inside would catch fire immediately and they would have been lost forever. The code (consisting of up to 100 integers) was hidden in the Alexandrian Library but unfortunately, as you probably know, the library burned down completely.

But an almost unknown archaeologist has obtained a copy of the code something during the 18th century. He was afraid that the code could get to the "wrong people" so he has encoded the numbers in a very special way. He took a random complex number B that was greater (in absolute value) than any of the encoded numbers. Then he counted the numbers as the digits of the system with basis B. That means the sequence of numbers an, an-1, ..., a1, a0 was encoded as the number X = a0 + a1B + a2B2 + ...+ anBn.

Your goal is to decrypt the secret code, i.e. to express a given number X in the number system to the base B. In other words, given the numbers X and Byou are to determine the "digit" a0 through an.

Input

The input consists of T test cases. The number of them (T) is given on the first line of the input. Each test case consists of one single line containing four integer numbers Xr, Xi, Br, Bi (|Xr|,|Xi| <= 1000000, |Br|,|Bi| <= 16). These numbers indicate the real and complex components of numbers X and B, i.e. X = Xr + i.Xi, B = Br + i.Bi. B is the basis of the system (|B| > 1), X is the number you have to express.

Output

Your program must output a single line for each test case. The line should contain the "digits" an, an-1, ..., a1, a0, separated by commas. The following conditions must be satisfied:

for all i in {0, 1, 2, ...n}: 0 <= ai < |B|

X = a0 + a1B + a2B2 + ...+ anBn

if n > 0 then an <> 0

n <= 100

If there are no numbers meeting these criteria, output the sentence "The code cannot be decrypted.". If there are more possibilities, print any of them.

Sample Input

4

-935 2475 -11 -15

1 0 -3 -2

93 16 3 2

191 -192 11 -12

Sample Output

8,11,18

1

The code cannot be decrypted.

16,15

参考答案

#include <cmath>

#include <stack>

#include <cstdio>

using namespace std;

int main()

{

int re;

int r, i, xr, xi, br, bi, bb, b, a;

scanf("%d", &re);

while (re--) {

scanf("%d%d%d%d", &xr, &xi, &br, &bi);

bb = br \* br + bi \* bi;

b = (int)(sqrt((double)bb) - 1e-6);

stack<int> ans;

do {

for (a = 0; a <= b; a++) {

r = (xr - a) \* br - xi \* (-bi);

i = (xr - a) \* (-bi) + xi \* br;

if (r % bb == 0 && i % bb == 0) {

xr = r / bb;

xi = i / bb;

ans.push(a);

break;

}

}

if (a > b || ans.size() > 100) {

break;

}

} while (xr != 0 || xi != 0);

if (xr != 0 || xi != 0) {

puts("The code cannot be decrypted.");

}

else {

while(!ans.empty()) {

printf("%d", ans.top());

ans.pop();

putchar(ans.empty() ? '\n' : ',');

}

}

}

return 0;

}

## zoj 2151 The Highest Profits

This program will not be part of the KOKOD��H collection, but it may be more important than the games themselves. It will serve to the marketing staff to find suitable procedures for promoting and selling the collection. So try to do your best.

Extensive marketing case study tried to prove that the total income from selling a product is polynomial function of number of satisfied customers. Experiments showed that the real results are not exactly the same which you can obtain using the function. However, this method is widely used. The reason is probably nonexistence of better solution. Let's denote the number of satisfied customers <var>y</var>, than we can express profits (denoted <var>x</var>) as

<var>x = P(y) = a0 + a1.y + a2.y2 + ... + am.ym</var>

The number of satisfied customers depends on the price of a product. Again, there is hypothese that this dependence is polynomial. If we denote the price <var>z</var>, we can write

<var>y = Q(x) = b0 + b1.z + b2.z2 + ... + bn.zn</var>

Coeficients <var>ai</var> and <var>bi</var> strongly depend on the season of the year, the moon phase, the purchasing power of customers, inflation rate and hunderds of other parameters. Besides on the kind of product and its quality, of course. In the past there was lot of effort put into the reserarch of these parameters. For various combinations of input parameters, the coeficients are stated in Pyshwejc's marketing tables. It is not thus difficult to find out their values. But the degree of polynoms is usualy very high. It is very difficult to substitute one polynom into the other and to compute the dependency of the profit on the price. This dependency is usually crucial for us to set the right price.

Your goal is to write the program which can substitute the polynom <var>Q</var> into the polynom <var>P</var> and determine the restulting polynom <var>R</var> indicating dependency of the profit on the price:

<var>x = R(z) = c0 + c1.z + c2.z2 + ... + cp.zp</var>

Input Specification

The input consists of <var>Z</var> assignments. The number of them is given by the single positive integer <var>Z</var> appearing on the first line of input. Each assingement constist of three lines. On the first line there are two integers <var>m</var> and <var>n</var> (<var>0 <= n,m <= 100</var>) separated by space. These numbers give the degree of polynoms <var>Q</var> and <var>P</var>. On the second line there are <var>m+1</var> integers <var>a0</var> ... <var>am</var>. These numbers are coeficients of polynom <var>P</var>. On the third line there are <var>n+1</var> integers <var>b0</var> ... <var>bn</var>. Always <var>am <> 0</var> and <var>bn <> 0</var>. Coeficients <var>ai</var> and <var>bi</var> are separated by space and they are chosen in order to each resulting coeficient could fit into the standard type integer.

Output Specification

The program prints exactly one line for each assignement. On this line, there will be <var>p+1</var> numbers. These numbers are coeficients <var>c0</var> ... <var>cp</var> of resulting polynom. The coeficients are separated by space and the line does not consist any redundant spaces. The coeficient in the highest degree should not be zero.

Sample Input

3

0 0

7

-2

1 1

6 6

9 -6

3 3

-3 6 -5 1

0 3 -3 1

Output for the Sample Input

7

60 -36

-3 18 -63 123 -156 138 -86 36 -9 1

参考答案

#include <iostream>

#include <cstdlib>

#include <algorithm>

#include <vector>

#include <string.h>

#define N 10005

using namespace std;

void cheng(vector<int> &a,vector<int> &b)

{

int i,j,k,c[N],m;

k = a.size()+b.size()-1;

m = b.size();

for ( i = 0 ; i < k ; i++)

{

if( i >= m ) j = i-m+1;

else j = 0 ;

for ( c[i] = 0 ; j <= i ; j++)

{

if( j < a.size() && (i-j) < b.size() )

c[i] += a[j]\*b[i-j];

}

}

a.clear();

for( i = 0 ; i < k ; i++) a.push\_back(c[i]);

}

int main()

{

int i,j,n,m,num,x;

int res[N];

cin>>num;

while (num--)

{

vector<int> a,b,c;

cin>>m>>n;

for ( i = 0 ; i <= m ; i++)

{

cin>>x;

a.push\_back(x);

}

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

{

cin>>x;

b.push\_back(x);

c.push\_back(x);

}

memset(res,0,sizeof(res));

for ( i = 1 , res[0] = a[0] ; i <= m ; i++)

{

for ( j = 0 ; j < c.size() ; j++)

res[j] += c[j]\*a[i];

cheng(c,b);

}

for ( i = 0 ; i <= n\*m ; i++)

{

if( !(i == n\*m && res[i] == 0 && i == 0 ))

{

if( i != 0 )

cout<<" ";

cout<<res[i];

}

else cout<<0;

}

cout<<endl;

}

return 0;

}

## zoj 2247 Magic Trick

A magician invented a new card trick and presented it in the prestigious American Conference of Magicians (ACM). The trick was so nice it received the "Best Magic Award" at the conference. The trick requires three participants: the magician himself, a spectator and an assistant. During the trick the spectator is asked to shuffle a deck of 52 cards and pick randomly 5 cards out of the deck. The five cards are given to the assistant (without the magician seeing the cards) who looks at them and shows four of the five cards one by one to the magician. After seeing the four cards the magician magically guesses the missing fifth card!

The trick works because once the assistant has the five cards he can always choose four of them and use those to `code' information about the fifth one. The code is based on an ordering of the cards. Cards are ordered first by their suits and then by their face value. We will use the following order:

1. H 2. 1

Assume the spectator chose the cards JD, 8S, 7H, 8C, QH (Jack of Diamonds, 8 of Spades, 7 of Hearts, 8 of Clubs and Queen of Hearts). The strategy for the assistant is the following:

1. Find a suit s which appears at least twice in the set of chosen cards (Hearts in the example). If more than one suit appears two times, choose the one with lowest order.

2. Hide the card x with suit s that is at most six positions ahead in the cyclic order 1 3. Show y to the magician. At this point the magician knows the suit of the hidden card, and also knows that the face value of the hidden card x is at most six positions in front of the face value of y.

4. With the three cards the assistant has left, he must code a number between 1 and 6. That can be done as follows. Say the three cards z1 , z2 , z3 are in the order z1 -- z1, z2, z3 means 1,

-- z1, z3, z2 means 2,

-- z2, z1, z3 means 3,

-- z2, z3, z1 means 4,

-- z3, z1, z2 means 5,

-- z3, z2, z1 means 6.

In this way, once the magician is shown the four cards one by one, he has enough information to "magically" guess the fifth one!

Your job is to develop a program that, given the four cards shown by the assistant, informs the magician which is the hidden card.

Input

The input contains several test cases. The first line in the input contains an integer N specifying the number of test cases (1 N

Output

For each test case in the input your program must produce one line of output, containing the description of the hidden card.

Sample Input

2

7H 8S 8C JD

TC 2D 1S 5H

Sample Input

QH

1C

参考答案

#include<iostream>

#include<string.h>

#include<stdio.h>

#include<map>

using namespace std;

enum {

SIZ = 3,

};

struct Node {

int v;

int t;

bool operator<(const Node&rhs){

if(t!=rhs.t){

if(t < rhs.t)

return true;

return false;

}

if(v < rhs.v)

return true;

return false;

}

};

Node tree[SIZ];

map<int,int> tab;

static char face[] = {'T','J','Q', 'K'};

static char suit[] = {'H','C','D', 'S'};

inline char v2f(char c){

c += 12;

c %= 13;

c += 1;

if(c<=9)

return c +'0';

c -= 10;

return face[c];

}

inline char f2v(char c){

if(c <= '9'){

return c-'0';

}

for(int i=0;i<sizeof(face);i++){

if(face[i] == c){

return i + 10;

}

}

return 0;

}

void convert(Node &one, string &s){

int i;

for(i=0;i<sizeof(suit);i++){

if(s[1]== suit[i]){

one.t = i;

break;

}

}

one.v = f2v(s[0]);

}

inline void insert(int a, int b, int v){

int key = (a<<4)+b;

tab[key] = v;

}

void init(){

insert(0,2,1);

insert(0,1,2);

insert(1,2,3);

insert(2,1,4);

insert(1,0,5);

insert(2,0,6);

}

char type, val;

void readIn(){

string s;

cin>>s;

type = s[1];

val = f2v(s[0]);

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

cin>>s;

convert(tree[i], s);

}

}

int getVal(){

int a,b;

if(tree[0] < tree[1]){

a = 1; b = 0;

} else {

a = 0, b= 1;

}

if(tree[a] < tree[2]){

a = 2;

} else if(tree[2] < tree[b]){

b = 2;

}

int key = (b<<4) + a;

return tab[key];

}

int fun(){

int t=getVal();

val += t;

val --;

val %= 13;

val ++;

val = v2f(val);

cout<<val<<type<<endl;

}

int main(){

init();

int tstcase;

cin>>tstcase;

while(tstcase --){

readIn();

fun();

}

return 0;

}

## zoj 2561 Order-Preserving Codes

Binary code is a mapping of characters of some alphabet to the set of finite length bit sequences. For example, standard ASCII code is a fixed length code, where each character is encoded using 8 bits.

Variable length codes are often used to compress texts taking into account the frequencies of occurence of different characters. Characters that occur more often get shorter codes, while characters occuring less often -- longer ones.

To ensure unique decoding of variable length codes so called prefix codes are usually used. In a prefix code no code sequence is a proper prefix of another sequence. Prefix code can be easily decoded scanning the encoded sequence from left to right, since no code is the prefix of another, one always knows where the code for the current character ends and the new character starts.

Among prefix codes, the optimal code is known, so called Huffman code. It provides the shortest possible length of the text among all prefix codes that separatly encode each character with an integer number of bits.

However, as many other codes, Huffman code does not preserve character order. That is, Huffman codes for lexicographically ordered characters are not necessarily lexicographicaly ordered.

In this problem you are asked to develop a prefix code that would be optimal for the given text among all order-preserving prefix codes. Code is called order-preserving if for any two characters the code sequence for the character that goes earlier in the alphabet is lexicographically smaller.

Since text itself is not essential for finding the code, only the number of occurences of each character is important, only this data is given.

Input:

The input consists of several test cases

For each test case, the first line contains n -- the number of characters in the alphabet (2 9). Characters are described in the alphabetical order.

Output:

For each test case, Output n bit sequences, one on a line -- the optimal order-preserving prefix code for the described text.

Sample Input:

5

1 8 2 3 1

Sample Output:

00

01

10

110

111

代码：

#include<iostream>

#include<sstream>

#include<vector>

#include<list>

#include<deque>

#include<queue>

#include<stack>

#include<map>

#include<set>

#include<bitset>

#include<algorithm>

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<cctype>

#include<cmath>

#include<ctime>

using namespace std;

const double eps(1e-8);

typedef long long lint;

#define clr(x) memset( x , 0 , sizeof(x) )

#define sz(v) ((int)(v).size())

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

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

#define repd(i, a, b) for (int i = (a); i >= (b); --i)

#define clrs( x , y ) memset( x , y , sizeof(x) )

const int Maxn = 2010;

int n;

lint f[Maxn][Maxn], sum[Maxn], x[Maxn];

int g[Maxn][Maxn];

bool v[Maxn][Maxn];

vector <int> ret[Maxn];

vector <int> tmp;

void Dfs(int l, int r) {

if (v[l][r]) return;

v[l][r] = true;

if (l + 1 == r) {

f[l][r] = sum[r] - sum[l - 1];

g[l][r] = l;

return;

}

if (l == r) {

g[l][r] = 0;

f[l][r] = 0;

return;

}

Dfs(l, r - 1);

Dfs(l + 1, r);

f[l][r] = -1;

for (int i = g[l][r - 1]; i <= g[l + 1][r]; i ++) {

if (i < l) continue;

if (i > r - 1) continue;

Dfs(l, i);

Dfs(i + 1, r);

if (f[l][i] + f[i + 1][r] + sum[r] - sum[l - 1] < f[l][r] || f[l][r] == -1) {

f[l][r] = f[l][i] + f[i + 1][r] + sum[r] - sum[l - 1];

g[l][r] = i;

}

}

}

void Search(int l, int r) {

if (l == r) {

ret[l] = tmp;

return;

}

tmp.push\_back(0);

Search(l, g[l][r]);

tmp.pop\_back();

tmp.push\_back(1);

Search(g[l][r] + 1, r);

tmp.pop\_back();

}

int main() {

while (scanf("%d", &n) == 1) {

sum[0] = 0;

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

scanf("%lld", &x[i]);

sum[i] = sum[i - 1] + x[i];

}

memset(v, 0, sizeof(v));

Dfs(1, n);

tmp.clear();

Search(1, n);

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

for (int j = 0; j < (int)ret[i].size(); j ++)

printf("%d", ret[i][j]);

printf("\n");

}

}

return 0;

}

## zoj 2814 Surprising Strings

The <dfn>D-pairs</dfn> of a string of letters are the ordered pairs of letters that are distance D from each other. A string is <dfn>D-unique</dfn> if all of its D-pairs are different. A string is <dfn>surprising</dfn> if it is D-unique for every possible distance D.

Consider the string ZGBG. Its 0-pairs are ZG, GB, and BG. Since these three pairs are all different, ZGBG is 0-unique. Similarly, the 1-pairs of ZGBG are ZB and GG, and since these two pairs are different, ZGBG is 1-unique. Finally, the only 2-pair of ZGBG is ZG, so ZGBG is 2-unique. Thus ZGBG is surprising. (Note that the fact that ZG is both a 0-pair and a 2-pair of ZGBG is irrelevant, because 0 and 2 are different distances.)

Input: The input consists of one or more nonempty strings of at most 79 uppercase letters, each string on a line by itself, followed by a line containing only an asterisk that signals the end of the input.

Output: For each string of letters, output whether or not it is surprising using the exact output format shown below.

Acknowledgement: This problem is inspired by the "Puzzling Adventures" column in the December 2003 issue of Scientific American.

<thead></thead>

Example input: Example output:

ZGBG

X

EE

AAB

AABA

AABB

BCBABCC

\*

ZGBG is surprising.

X is surprising.

EE is surprising.

AAB is surprising.

AABA is surprising.

AABB is NOT surprising.

BCBABCC is NOT surprising.

代码：

#include <iostream>

#include <stdio.h>

#include <string.h>

using namespace std;

int main()

{

char s[80];

while(scanf("%s",s))

{

if(strcmp(s,"\*")==0)

return 0;

else

{

int flag;

for(int d = 1 ; d < strlen(s)-1; d++)

{

char \*\*a = new char\*[strlen(s)-d];

for(int i = 0 ; i < strlen(s)-d; i++)

{

a[i] = new char[2];

a[i][0] = s[i];

a[i][1] = s[i+d];

}

flag = 0;

for( int j = 0 ; j < strlen(s)-d; j++)

{

for(int k = j+1; k < strlen(s)-d; k++)

{

if(strcmp(a[j],a[k])==0)

{

flag = 1;

break;

}

}

if(flag == 1) break;

}

if(flag == 1) break;

}

if(flag == 1)

cout<<s<<" is NOT surprising."<<endl;

else

cout<<s<<" is surprising."<<endl;

}

}

return 0;

}

## zoj 2990 Decoding

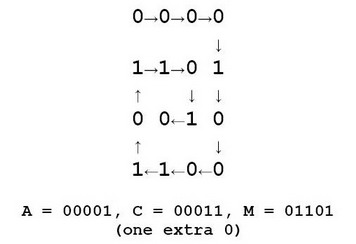
Chip and Dale have devised an encryption method to hide their (written) text messages. They first agree secretly on two numbers that will be used as the number of rows (R) and columns (C) in a matrix. The sender encodes an intermediate format using the following rules:

1.

The text is formed with uppercase letters [A-Z] and <space>.

2.

Each text character will be represented by decimal values as follows:



<space> = 0, A = 1, B = 2, C = 3, ..., Y = 25, Z = 26 The sender enters the 5 digit binary representation of the characters' values in a spiral pattern along the matrix as shown below. The matrix is padded out with zeroes (0) to fill the matrix completely. For example, if the text to encode is: "ACM" and R=4 and C=4, the matrix would be filled in as follows:

A = 00001, C = 00011, M = 01101(one extra 0) The bits in the matrix are then concatenated together in row major order and sent to the receiver. The example above would be encoded as: 0000110100101100

Input

The first line of input contains a single integer N, (1 <= N <= 1000) which is the number of datasets that follow.

Each dataset consists of a single line of input containing R (1 <= R <= 21), a space, C (1 <= C <= 21), a space, and a string of binary digits that represents the contents of the matrix (R \* C binary digits). The binary digits are in row major order.

Output

For each dataset, you should generate one line of output with the following values: The dataset number as a decimal integer (start counting at one), a space, and the decoded text message. You should throw away any trailing spaces and/or partial characters found while decoding.

Sample Input

4

4 4 0000110100101100

5 2 0110000010

2 6 010000001001

5 5 0100001000011010110000010

Sample Output

1 ACM

2 HI

3 HI

4 HI HO

参考答案

#include <iostream>

#include <string.h>

#include <stdio.h>

#include <algorithm>

using namespace std;

int n,m;

int dir[4][2]={0,1,1,0,0,-1,-1,0};

char str[1010];

int bo[22][22];

int vis[22][22];

int check(int x,int y)

{

if(x<0||y<0||x>=n||y>=m||vis[x][y])

return 0;

return 1;

}

int sum=0,limit;

int ans[1010];

void dfs(int x,int y,int d,int num,int po)

{

vis[x][y]=1;

int xx=x+dir[d][0],yy=y+dir[d][1];

limit--;

if(po==4)

{

ans[sum++]=num\*2+bo[x][y];

po=0;

if(limit<=0)

return;

if(check(xx,yy))

{

dfs(xx,yy,d,0,0);

}

else

{

d=(d+1)%4;

xx=x+dir[d][0],yy=y+dir[d][1];

dfs(xx,yy,d,0,0);

}

}

else

{

if(check(xx,yy))

{

dfs(xx,yy,d,num\*2+bo[x][y],po+1);

}

else

{

d=(d+1)%4;

xx=x+dir[d][0],yy=y+dir[d][1];

dfs(xx,yy,d,num\*2+bo[x][y],po+1);

}

}

}

int main()

{

int ncase,time=0;

scanf("%d",&ncase);

while(ncase--)

{

scanf("%d%d",&n,&m);

printf("%d ",++time);

scanf("%s",str);

sum=0;

int len=n\*m,i,j;

limit=len/5\*5;

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

{

for(j=0;j<m;j++)

bo[i][j]=str[i\*m+j]-'0';

}

memset(vis,0,sizeof(vis));

dfs(0,0,0,0,0);

while(ans[sum-1]==0)sum--;

for(i=0;i<sum;i++)

{

if(ans[i]==0)

printf(" ");

else

printf("%c",ans[i]-1+'A');

}

printf("\n");

}

return 0;

}

## zoj 3019 Puzzle

For sequences of integers a and b, if you can make the two sequences the same by deleting some elements in a and b, we call the remaining sequence "the common sub sequence". And we call the longest one the LCS.

Now you are given two sequences of integers a and b . You can arrange elements in a and b in any order. You are to calculate the max length of the LCS of each arrangement of a and b .

Input

Input will consist of multiple test cases. The first line of each case is two integers N(0 < N < 10000), M(0 < M < 10000) indicating the length of a and b . The second line is N 32-bit signed integers in a . The third line is M 32-bit signed integers in b .

Output

Each case one line. The max length of the LCS of each arrangement of a and b .

Sample Input

5 4

1 2 3 2 1

1 4 2 1

Sample Output

3

代码：

#include<iostream>

#include<cstdio>

#include<vector>

#include<queue>

#include<cstring>

#include<cmath>

#include<algorithm>

using namespace std;

#define N 100005

#define ll long long

#define pi acos(-1.0)

int a[N],b[N];

int main()

{

int n,m,ans;

while(scanf("%d%d",&n,&m)!=EOF)

{

for(int i=1;i<=n;i++) scanf("%d",a+i);

for(int i=1;i<=m;i++) scanf("%d",b+i);

sort(a+1,a+n+1);

sort(b+1,b+m+1);

ans = 0;

int ii = 1,jj = 1;

while(ii<=n&&jj<=m)

{

if(a[ii]==b[jj])

{

ii++;

jj++;

ans++;

}

while(ii<=n&&a[ii]<b[jj]) ii++;

while(jj<=m&&b[jj]<a[ii]) jj++;

}

printf("%d\n",ans);

}

}

## zoj 3109 Decode Message

Being in charge of the computer department of the Agency of International Espionage, you are asked to write a program that will allow a spy to decode their messages.

You can assume a spy's message is at most 400 characters long, and it includes all the uppercase letters of the English alphabet plus the space, any digit, and any of the following characters: ! , . : ; ? - The algorithm that the department will use to encode all their messages will be following:

They first encode the message with a simple code key. This simple code key is a one for one character substitution based upon a single arithmetic manipulation of the printable portion of the ASCII character set (0-126). This single arithmetic manipulation is the same for each character of the message. (We assume the code is 7, please see sample for more detail.)

After, they agree secretly on two numbers that will be used as the number of rows (R) and columns (C) in a matrix.

The department then enters the letters of the message encode in spiral pattern along the matrix.

For example, if the message is:

-CDC IS THE TRADEMARK OF THE CONTROL DATA CORPORATION.

and there R=9 and C=6, the department would write down.

Message encode:

4JKJ'PZ'[OL'[YHKLTHYR'VM'[OL'JVU[YVS'KH[H'JVYWVYH[PVU5

The matrix would be filled in as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4 | J | K | J | ' | P |
| [ | O | L | ' | J | Z |
| ' | V | Y | W | V | ' |
| M | J | 5 | V | U | [ |
| V | ' | U | Y | [ | O |
| ' | H | V | H | Y | L |
| R | [ | P | [ | V | ' |
| Y | H | K | ' | S | [ |
| H | T | L | K | H | Y |

**Input**

There will be multiple datasets. Each dataset consist of two lines. The first line contain R (2<=R<=20), a space, C (2<=C<=20). The next line is a string of characters that represent the contents of the matrix (R X C characters). The characters are in row major order. The last input set is followed by a line containing two zeros (0 0). This line should not be processed.

**Output**

For each dataset, you should one line of output, giving the decoded message.

**Sample Input**

9 6

4JKJ'P[OL'JZ'VYWV'MJ5VU[V'UY[O'HVHYLR[P[V'YHK'S[HTLKHY

0 0

**Sample Output**

-CDC IS THE TRADEMARK OF THE CONTROL DATA CORPORATION.

代码：

#include<stdio.h>

int main()

{

int r, c;

char a[20][20];

while (scanf("%d %d", &r, &c), r && c)

{

getchar();

int i, j, index = 0;

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

for (j = 0; j < c; j++)

a[i][j] = getchar();

getchar();

int isRow = 1, isOrder = 1, rb = 0, re = c - 1, cb = 0, ce = r - 1;

while (1)

{

if (isRow)

{

if (rb > re)

break;

if (isOrder)

{

for (i = rb; i <= re; i++)

putchar(a[index][i] - 7);

index = re;

cb++;

}

else

{

for (i = re; i >= rb; i--)

putchar(a[index][i] - 7);

index = rb;

ce--;

}

isRow = !isRow;

}

else

{

if (cb > ce)

break;

if (isOrder)

{

for (i = cb; i <= ce; i++)

putchar(a[i][index] - 7);

index = ce;

re--;

}

else

{

for (i = ce; i >= cb; i--)

putchar(a[i][index] - 7);

index = cb;

rb++;

}

isOrder = !isOrder;

isRow = !isRow;

}

}

putchar('\n');

}

return 0;

}

## zoj 3114 Double Queue

The new founded Balkan Investment Group Bank (BIG-Bank) opened a new office in Bucharest, equipped with a modern computing environment provided by IBM Romania, and using modern information technologies. As usual, each client of the bank is identified by a positive integer K and, upon arriving to the bank for some services, he or she receives a positive integer priority P. One of the inventions of the young managers of the bank shocked the software engineer of the serving system. They proposed to break the tradition by sometimes calling the serving desk with the lowest priority instead of that with the highest priority. Thus, the system will receive the following types of request:

0 The system needs to stop serving

1 K P Add client K to the waiting list with priority P

2 Serve the client with the highest priority and drop him or her from the waiting list

3 Serve the client with the lowest priority and drop him or her from the waiting list

Your task is to help the software engineer of the bank by writing a program to implement the requested serving policy.

Input

Each line of the input contains one of the possible requests; only the last line contains the stop-request (code 0). You may assume that when there is a request to include a new client in the list (code 1), there is no other request in the list of the same client or with the same priority. An identifier K is always less than 106, and a priority P is less than 107. The client may arrive for being served multiple times, and each time may obtain a different priority.

Output

For each request with code 2 or 3, the program has to print, in a separate line of the standard output, the identifier of the served client. If the request arrives when the waiting list is empty, then the program prints zero (0) to the output.

Sample Input

2

1 20 14

1 30 3

2

1 10 99

3

2

2

0

Sample Output

0

20

30

10

0

参考答案

#include <cmath>

#include <string>

#include <cstdio>

#include <cstring>

#include <vector>

#include <set>

#include <string>

#include <algorithm>

#include <functional>

using namespace std;

typedef long long LL;

struct Client {

int id, val;

Client() { }

Client(int x, int y) : id(x), val(y) { }

};

struct great {

bool operator ()(const Client& a, const Client& b) {

return a.val > b.val;

}

};

struct small {

bool operator ()(const Client& a, const Client& b) {

return a.val < b.val;

}

};

set<Client, great> s1;

set<Client, small> s2;

int main() {

int q;

while (~scanf("%d", &q), q) {

if (q == 1) {

int p, q;

scanf("%d%d", &p, &q);

s1.insert(Client(p, q));

s2.insert(Client(p, q));

}

if (q == 2) {

if (s1.empty()) {

printf("0\n");

} else {

Client ans = \*(s1.begin());

printf("%d\n", ans.id);

s1.erase(ans);

s2.erase(ans);

}

}

if (q == 3) {

if (s2.empty()) {

printf("0\n");

} else {

Client ans = \*(s2.begin());

printf("%d\n", ans.id);

s1.erase(ans);

s2.erase(ans);

}

}

}

return 0;

}

## zoj 3235 Prototype

Prototype

Time Limit: 1 Second Memory Limit: 32768 KB

Prototype is a 3D game which allow you to control a person named Alex with much super ability to finish missions with gut along.

原型是一个3D游戏，它允许你控制一个叫做Alex的人物，他有足够的能力完成任务。

Alex has the abilitiy to glide in the sky.

Alex有能力在天空中滑翔。

What's more, he can make at most 3-level glide, which means before he lands at the ground, he has two chances to adjust and perform another glide.

更重要的是，他能够最多滑翔3级，这意味着在他着陆之前有两次机会调整执行另一次滑翔。

We assume that each time he perform a glide, his vertical speed become zero and glide forward with a new speed.

我们假设每次他进行滑翔时，他的垂直速度会变成零，并以一个新的速度向前滑行。

And the orbit will be a parabola due to the gravity.

轨道会因为重力而变成抛物线。

To make the problem easier, we now only consider at most 2-level glide.

为了使这个问题变得简单，我们现在只考虑最多2级滑翔。

The binomial coefficient of the mathematical equation of the fist glide will be given as -a and the second will be -b, which means the formulations are (y - y0) = -ax2 and (y - y0) = -b(x - x0)2.

As the picture above, Alex perform a glide from the top of Building1, make a 1-level or a 2-level glide and lands exactly at point B. What's more, there is Building2 standing between Building1 and point B. Alex has to avoid crashing onto it.

Input

There are no more than 15 cases. Proceed till the end of file.

Each case contains only one line of six real number h1, h2, d1, d2, a, b. h1 is the height of Building1, h2 is the height of Building2, d1 is the X-distance between Building1 and Building2, d2 is the X-distance between point B and Building1. These four numbers are in [0, 1000] , and satisfies d1 < d2. And a and b are in (0, 1000].

Output

If it is possible for Alex to land exactly on point B, print Yes, otherwise print No.

Sample Input

25 1 6 7 1 1

4 3 1 2 1 1

Sample Output

Yes

Yes

HINT

In case 2, Alex just glide over the building2 and do not crash onto it.

Author: ZHUANG, Junyuan

Source: ZOJ Monthly, August 2009

代码：

#include <cstdio>

#include <cmath>

const double eps=1e-9;

double h1,h2,d1,d2,a,b;

bool check(double x0){

double y;

if(x0<d1){

double y0=h1-a\*x0\*x0;

y=y0-b\*(d1-x0)\*(d1-x0);

}

else if(x0-eps>d2) return false;

else y=h1-a\*d1\*d1;

return y+eps>=h2;

}

int main(){

while(scanf("%lf %lf %lf %lf %lf %lf",&h1,&h2,&d1,&d2,&a,&b)==6){

double A=a+b;

double B=-2.\*b\*d2;

double C=b\*d2\*d2-h1;

double det=B\*B-4\*A\*C;

bool ans=false;

if(det+eps>=0){

det=sqrt(fabs(det));

double x1=(-B-det)/(2\*A);

double x2=(-B+det)/(2\*A);

if(x1+eps>=0.) ans=ans||check(x1);

if(x2+eps>=0.) ans=ans||check(x2);

}

puts(ans?"Yes":"No");

}

return 0;

}

## zoj 3393 Routing

You work as an engineer for the Inane Collaboration for Performance Computing, where you are in charge of designing an intercommunication network for their computers. The network is arranged as a rectangular array of 2n - 1 rows, each having 2 n-1 switches. A switch is a device with two input wires, X and Y, and two output wires, X' and Y'. If the switch is off, data from input X will be relayed to output X' , and data from Y to Y'. If it is on, X will be connected to Y' and Y to X' . Additionally, there are 2 n computers in the topmost and bottommost rows, and messages need to be sent between pairs of them. Notice that data from two different sources cannot share a wire but, of course, both pieces of data can be routed through the same switch on different inputs.

You have come to the conclusion that the network that best suits your purposes has the Benes topology. A 1-Benes network is just a switch. For n > 1, a n-Benes network can be constructed recursively as follows:

In the first (top) row there are 2 n-1 switches such that switch j (0 ≤ j < 2 n-1) has data inputs from computers 2j and 2j + 1 (we label the computers in the topmost and bottommost rows with integers between 0 and 2 n - 1, inclusive, from left to right).

Then a perfect shuffle permutation is applied to the output wires between the first and the second rows of switches, meaning that output number j in a row is connected to input number j' in the next row, where j' is obtained by rotating the n-bit pattern representing j in binary one bit to the right (again, inputs and outputs are numbered from left to right).

If n > 2, the next rows of switches, up to (and including) the last-but-one, form two (n - 1)-Benes subnetworks, one on the left side and the other on the right side.

Finally, the inverse shuffle permutation is applied to the outputs and a last row of switches is added.

\

Figure (a)                      Figure (b)

For example, Figure (a) shows the Benes network for n = 3 (squares represent switches; computers in the top and bottom rows are not drawn, but assigned with integers from 0 to 7). Figure (b) shows a possible state of the switches; squares where two of the lines cross are switches that have been turned on. You may verify that this state allows us to simultaneously establish communication paths from computers 0, 1, 2, 3, 4, 5, 6, 7 at the bottom to 3, 7, 4, 0, 2, 6, 1, 5 at the top, respectively.

You are given a set of pairs (a, b) of computers to connect simultaneously (where a is a computer in the bottom row and b a computer in the top row) by means of wire-disjoint paths, and you are to find how to select the state of all switches so that this can be accomplished.

Input

The first line of each test case is an integer n(1 ≤ n ≤ 13), meaning that you have 2 n pairs of computers to connect, as described above. A line with n = 0 marks the end of the input and should not be processed.

Each line with n > 0 will be followed by another line containing 2 n integers. The i-th integer(0 ≤ i < 2 n ) will be the computer in the topmost row that the i-th computer in the bottommost row needs to communicate with.

Output

The output for each case should have 2n-1 lines, each containing a binary string of length 2 n-1 indicating, for each switch, whether it must be turned on (1) or off (0).

The input given will always have at least one solution. In case of several solutions, return the lexicographically smallest one. That is, the string in the top row must be lexicographically smallest; in case of a tie, the string in the second row must be lexicographically smallest, and so on.

Outputs for different test cases should be separated by a blank line.

Sample Input

2

3 2 1 0

3

3 7 4 0 2 6 1 5

0

Sample Output

00

11

11

0011

0000

0110

1111

1101

代码：

#include <map>

#include <queue>

#include <cstdio>

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

void dump(const vector<int>& v) {

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

printf("%d ", v[i]);

}

puts("");

}

vector<int> P[16], Q[16];

map<int, int> minv(int n, const vector<int>& p) {

map<int, int> ret;

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

ret[p[i]] = i;

}

return ret;

}

vector<int> inv(int n, const vector<int>& p) {

vector<int> ret(n);

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

ret[p[i]] = i;

}

return ret;

}

vector<int> per(int n, const vector<int>& s, const vector<int>& t) {

map<int, int> rt = minv(n, t);

vector<int> ret(n);

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

ret[i] = rt[s[i]];

}

return inv(ret.size(), ret);

}

vector<string> gao(int n, vector<int> p) {

if (n == 1) {

if (p[0] == 0) {

return vector<string>(1, "0");

} else {

return vector<string>(1, "1");

}

} else {

int s = 1 << n;

int m = 1 << (n - 1);

vector<int> rp = inv(s, p);

vector<int> d(s, 0);

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

if (d[i] != 0) {

continue;

}

queue<int> q;

d[i] = -1;

q.push(i);

while (!q.empty()) {

int k = q.front();

q.pop();

if (d[k ^ 1] == 0) {

d[k ^ 1] = -d[k];

q.push(k ^ 1);

}

if (d[p[rp[k] ^ 1]] == 0) {

d[p[rp[k] ^ 1]] = -d[k];

q.push(p[rp[k] ^ 1]);

}

}

}

vector<int> xs, xt, ys, yt;

string x, y;

for (int i = 0; i < s; i += 2) {

if (d[i] == -1) {

x += '0';

xs.push\_back(i);

ys.push\_back(i ^ 1);

} else {

x += '1';

ys.push\_back(i);

xs.push\_back(i ^ 1);

}

}

for (int i = 0; i < s; i += 2) {

if (d[p[i]] == -1) {

y += '0';

xt.push\_back(p[i]);

yt.push\_back(p[i ^ 1]);

} else {

y += '1';

yt.push\_back(p[i]);

xt.push\_back(p[i ^ 1]);

}

}

vector<string> ret, left, right;

ret.push\_back(x);

left = gao(n - 1, per(m, xs, xt));

right = gao(n - 1, per(m, ys, yt));

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

ret.push\_back(left[i] + right[i]);

}

ret.push\_back(y);

return ret;

}

}

void init() {

for (int n = 1; n <= 13; ++n) {

P[n].resize(1 << n);

for (int j = 0; j < (int)P[n].size(); ++j) {

P[n][j] = (j >> 1) ^ ((j & 1) << (n - 1));

}

Q[n] = inv(P[n].size(), P[n]);

}

}

int main() {

bool blank = false;

int n;

vector<int> p;

vector<string> s;

init();

while (scanf("%d", &n) != EOF && n > 0) {

if (blank) {

puts("");

} else {

blank = true;

}

p.resize(1 << n);

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

scanf("%d", &p[i]);

}

s = gao(n, p);

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

puts(s[i].c\_str());

}

}

return 0;

}

## zoj 3502 Contest

Lazyman is going to join the programming contest! There are n problems in the contest. Because Lazyman is so lazy, he just tried each problem once and only once. However each problem in the contest may be not independent, it may have some relationship with other problems. So here is a propbility P n\*n matrix. The jth element of the ith row is denoted by Pij . Suppose he have tried m problems, a1 ,a2 ,...am (No matter the problems are solved or not.) (1 ≤ a1 ,a2 ,...am ≤ n) If he tries the rth problems next, (r!=a1 ,a2 ,...am ) the propbility that he can solve it is the maximum value of the m + 1 values, Prr , Pa1r %, Pa2r %, ... Pamr %.

Help him to decide the order he tries the problems to make the expected number of the solved problems maximum.

Input

There are multiple test cases. The first line of input is an integer T (0 < T ≤ 100), indicating the number of test cases. Then T test cases follow. The first line of each test case is a positive integer n (0 < n ≤ 10). Then comes n lines. The jth element of the ith line is an integer Pij . (0 ≤ Pij ≤100 1 ≤ i,j ≤ n)

Output

For each test case, output 2 lines. The first line is the maxmum expected number of problems, accurated to the nearest 0.01. The second the order he should try each problems to achieve this expected number of problems. If there are several sequeces achieve the same maximum expected number, choose the lexicographically smallest one. Use capital letters to denote each problems please. (Start from A)

Sample Input

2

1

0

2

2 3

5 7

Sample Output

0.00

A

0.12

BA

Hint

For the 2nd sample, if the order is AB:

The probility of the result that he solves problem A is 0.02\*(1-0.07)=0.0186

The probility of the result that he solves problem B is (1-0.02)\*0.07=0.0686

The probility of the result that he solves problem A and B is 0.02\*0.07=0.0014

So the expected number of the problem sovled is 1\*0.0186+1\*0.0686+2\*0.0014=0.09

For the 2nd sample, if the order is BA:

The probility of the result that he solves problem A is (1-0.07)\*0.05=0.0465

The probility of the result that he solves problem B is 0.07\*(1-0.05)=0.0665

The probility of the result that he solves problem A and B is 0.07\*0.05=0.0035

So the expected number of the problem sovled is 1\*0.0465+1\*0.0665+2\*0.0035=0.12

参考答案

#include <algorithm>

#include <iostream>

#include <cstring>

#include <sstream>

#include <vector>

#include <cmath>

#include <set>

#include <map>

#include <iomanip>

using namespace std;

#define int long long

#define ff first

#define ss second

#define mp make\_pair

#define sqr(x) ((x)\*(x))

typedef long long ll;

typedef pair <int, int> pie;

typedef pair <int, int> pii;

int n;

int mat[20][20];

pii dp[1<<15];

inline void main2(){

cin >> n;

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

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

cin >> mat[i][j];

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

dp[i] = pii(-1,0);

dp[0] = pii(0,0);

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

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

int maxi = 0;

for (int j=0; j<n; j++) if (mask & (1<<j))

maxi = max(maxi, mat[j][i]);

dp[mask] = max(dp[mask], pii(dp[mask ^ (1<<i)].first + maxi, dp[mask ^ (1<<i)].second \* 10 - i));

}

}

cout << fixed << setprecision(2) << dp[(1<<n)-1].first/100.0 << endl;

int cur = -dp[(1<<n)-1].second;

string ret(n, 'A');

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

ret[n-1-i] = char('A' + cur%10);

cur/=10;

}

cout << ret << endl;

}

main() {

ios::sync\_with\_stdio (false);

int testCase; cin >> testCase;

for (int o=1; o<=testCase; o++){

main2();

}

return 0;

}

## zoj 3531 Alice Madness Return

Recently there is a new game called <<Alice Madness Return>>. In this game, Alice was trapped in the wonderland by some evil spirits and she also lost her memory. The player need to help Alice find her memory and escape from wonderland.

The chess board

Now Alice is facing a big door which is locked. If she wants to open the door, she must pass a strange maze on a chess board with size m\*n. There is a pawn controlled by Alice, and there is a mirror pawn indirectly controlled by Alice. In each turn, Alice can move her own pawn 1 step to one of the four direction (north, south, west, east). At the same time, the mirror pawn will also move 1 step in opposite direction. Since there are 2 exits, Alice must find a way to make her pawn and the mirror pawn reach both the 2 exits simultaneously so that she can open the door.

This game is not easy because there are 2 kinds of chess pieces on the chess board which are listed below:

Rook: You can consider the Rook as a heavy stone so that nobody can step into the grid occupied by a Rook. The Rook itself will never move.

Knight: There is only one Knight on the chess board. Since the Knight is very dangerous, both Alice's pawn and the mirror pawn cannot be caught by the Knight (it means they meet in the same grid), or they will be killed. At first the Knight stands in a grid, facing one of the four direction. In each turn, after Alice and the mirror pawn has moved, the Knight will also move 1 step to the direction he is facing. If he is blocked by a Rook or he is on the boundary of the board, he will turn around (i.e turn to north if he is facing south) instead of moving.

Besides, there are some memory pieces of Alice on the chess board. Alice must control her own pawn to collect all the memory pieces before she leave the maze. There are at most 5 memory pieces on the chess board.

Now it's your job to help Alice find a way to collect all her memory and open the door as soon as possible.

Notice:

Any two chess pieces cannot occupy the same grid but the Knight can move into grid occupied by Alice or the mirror and in that situation he will kill them. You can consider the exit as empty grid so that Knight can pass the exit.

In each turn, Alice must move her own pawn and cannot move out of the chess board. If she cannot move, she will die.

After Alice's move, the mirror will move, but if the mirror is blocked by any chess piece(i.e the mirror is blocked by a Rook) or it is forced to move out of the chess board, then the mirror will stay unmoved.

Once Alice and the mirror reach the 2 exits, the door will open and the Knight cannot catch them after that. Of course, if the door has been open, Alice cannot get back to the chessboard to collect her memory any more.

Input

The input contains multiple test cases.

In each test case, fisrt there are two integers, <var>m</var> and <var>n</var> ( 1 <= <var>m</var>, <var>n</var> <=10 ) , which is the size of the chess board.

Then there are <var>m</var> lines each containing <var>n</var> characters. In these characters, '\*' means empty grid, 'R' means Rook, 'K' means Kight, 'A' means Alice's pawn, 'B' means the mirror pawn, 'E' means exit, 'M' means memory piece.

Finally there is a line containing one character describing the initial direction the Knight is facing. 'N' means north, 'S' means south, 'W' means west, 'E' means east.

We guarantee that there are exactly 1 Knight, 2 exits, no more than 5 memory pieces on the chess board.

Output

For each test case, output one line with an integer which is the minimum number of turns Alice needed to collect all her memory pieces and open the door. If she cannot finish the task, output -1 instead.

Sample Input

6 6

\*BR\*E\*

\*\*\*\*\*\*

\*\*R\*\*\*

\*\*\*K\*\*

\*\*R\*\*\*

E\*\*R\*A

E

Sample Output

8

参考答案

#include<stdio.h>

#include<string.h>

#include<math.h>

#include<iostream>

#include<algorithm>

#include<map>

#include<set>

#include<vector>

#include<queue>

using namespace std;

typedef long long lld;

#define pb push\_back

#define mp make\_pair

#define X first

#define Y second

#define inf 0xfffffff

int n,m;

char g[110][110];

bool inside(int x,int y)

{

if(x < 0 || x >= n || y < 0 || y >= m)

return false;

return true;

}

struct Node

{

int a,b,c,mask;

Node(){}

Node(int a0,int b0,int c0,int mask0)

{

a=a0;

b=b0;

c=c0;

mask=mask0;

}

};

char str[110];

queue<Node>Q;

int dis[10][10][10][10][2][10][1<<5];

int tx[4]={1,-1,0,0};

int ty[4]={0,0,1,-1};

int af,sf;

int id[110][110];

int T;

int bfs()

{

while(!Q.empty())

{

Node o=Q.front();

Q.pop();

int ax,ay,bx,by,cx,cy,ck,mask;

ax=o.a/10;

ay=o.a%10;

bx=o.b/10;

by=o.b%10;

if(af == 0)

{

cx=o.c%10;

cy=sf;

}

else

{

cx=sf;

cy=o.c%10;

}

int c01=o.c/10;

int cat=o.c%10;

mask=o.mask;

ck=o.c/10+af;

for(int k=0;k<4;k++)

{

int x1=ax+tx[k];

int y1=ay+ty[k];

if(!inside(x1,y1) || g[x1][y1] == 'R' ||

(x1 == bx && y1 == by) || (x1 == cx && y1 == cy))

continue;

int tk=k^1;

int x2=bx+tx[tk];

int y2=by+ty[tk];

if(!inside(x2,y2) || g[x2][y2] == 'R' ||

(x2 == x1 && y2 == y1) || (x2 == cx && y2 == cy))

{

x2=bx;

y2=by;

}

int next=mask;

if(id[x1][y1] != -1)

next|=(1<<id[x1][y1]);

if(g[x1][y1] == 'E' && g[x2][y2] == 'E' && (x1 != x2 || y1 != y2))

{

if(next == ((1<<T)-1))

return dis[ax][ay][bx][by][c01][cat][mask]+1;

continue;

}

int x3=cx+tx[ck];

int y3=cy+ty[ck];

int nc01=c01;

if(!inside(x3,y3) || g[x3][y3] == 'R')

{

x3=cx;

y3=cy;

nc01^=1;

}

else

{

if((x3 == x1 && y3 == y1) || (x3 == x2 && y3 == y2))

continue;

}

int ncat;

if(af == 0)

ncat=x3;

else

ncat=y3;

if(dis[x1][y1][x2][y2][nc01][ncat][next] == -1)

{

dis[x1][y1][x2][y2][nc01][ncat][next]=dis[ax][ay][bx][by][c01][cat][mask]+1;

Q.push(Node(x1\*10+y1,x2\*10+y2,nc01\*10+ncat,next));

}

}

}

return -1;

}

int main()

{

while(scanf("%d %d",&n,&m)!=EOF)

{

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

scanf("%s",g[i]);

int ax,ay,bx,by,c,cx,cy,f;

T=0;

memset(id,-1,sizeof(id));

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

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

{

if(g[i][j] == 'A')

{

ax=i;

ay=j;

}

if(g[i][j] == 'B')

{

bx=i;

by=j;

}

if(g[i][j] == 'K')

{

cx=i;

cy=j;

}

if(g[i][j] == 'M')

id[i][j]=T++;

}

scanf("%s",str);

if(str[0] == 'N' || str[0] == 'S')

c=cx,sf=cy;

else

c=cy,sf=cx;

if(str[0] == 'S')

f=0,af=0;

if(str[0] == 'N')

f=1,af=0;

if(str[0] == 'E')

f=0,af=2;

if(str[0] == 'W')

f=1,af=2;

memset(dis,-1,sizeof(dis));

dis[ax][ay][bx][by][f][c][0]=0;

while(!Q.empty())

Q.pop();

Q.push(Node(ax\*10+ay,bx\*10+by,f\*10+c,0));

printf("%d\n",bfs());

}

return 0;

}

## zoj 3590 -3+1

ZOJ is 10 years old! For celebrating, we are offering the easiest problem in the world to you.

Recently we received a long sequence. We can modify the sequence once by the following two steps.

Choose any element in the sequence, say x(satisfying x ≥ 3), and subtract 3 from x.

Choose any element in the sequence, say x, and add 1 to x.

Now, we want to know how many times at most the sequence can be modified.

Input

The input contains multiple test cases. For each case, the first line contains an integer <var>n</var>(1 ≤ <var>n</var> ≤ 20000). The second line contains <var>n</var> integers describing the sequence. All the numbers in the sequence are non-negative and not greater than 1000000.

Output

Output number of times at most the sequence can be modified, one line per case.

Sample Input

1

10

2

10 11

Sample Output

4

10

代码：

#include<iostream>

#include<cstdlib>

#include<cstring>

#include<cstdio>

using namespace std;

long long x,ans,sum,cn2,cn1;

int n;

int main(){

while(scanf("%d",&n)==1)

{

ans=0;

cn2=0;

cn1=0;

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

{

scanf("%I64d",&x);

ans+=x/3;

if(x%3==2) cn2++;

if(x%3==1) cn1++;

}

sum=ans;

if(ans==0) {printf("0\n");continue;}

ans+=cn2;

if(sum<=cn1) ans+=(sum-1);

else

{

ans+=cn1;

ans+=(sum-cn1-1)/2;

}

printf("%lld\n",ans);

}

return 0;

}

## zoj 3770 Ranking System

Few weeks ago, a famous software company has upgraded its instant messaging software. A ranking system was released for user groups. Each member of a group has a level placed near his nickname. The level shows the degree of activity of a member in the group.

Each member has a score based his behaviors in the group. The level is determined by this method:

Level Percentage The number of members in this level

LV1 / All members whose score is zero

LV2 / All members who can not reach level 3 or higher but has a positive score

LV3 30% ⌊(The number of members with a positive score) \* 30%⌋

LV4 20% ⌊(The number of members with a positive score) \* 20%⌋

LV5 7% ⌊(The number of members with a positive score) \* 7%⌋

LV6 3% ⌊(The number of members with a positive score) \* 3%⌋

⌊<var>x</var>⌋ is the maximum integer which is less than or equal to <var>x</var>.

The member with the higher score will get the higher level. If two members have the same score, the earlier one who joined the group will get the higher level. If there is still a tie, the user with smaller ID will get the higher level.

Please write a program to calculate the level for each member in a group.

Input

There are multiple test cases. The first line of input is an integer <var>T</var> indicating the number of test cases. For each test case:

The first line contains an integer <var>N</var> (1 <= <var>N</var> <= 2000) indicating the number of members in a group.

The next <var>N</var> lines, each line contains three parts (separated by a space):

The ID of the i-th member <var>Ai</var> (0 <= <var>Ai</var> <= 1000000000). The ID of each member is unique.

The date of the i-th member joined the group, in the format of YYYY/MM/DD. The date will be in the range of [1900/01/01, 2014/04/06].

The score <var>Si</var> (0 <= <var>Si</var> <= 9999) of the i-th member.

Output

For each test case, output <var>N</var> lines. Each line contains a string represents the level of the i-th member.

Sample Input

1

5

123456 2011/03/11 308

123457 2011/03/12 308

333333 2012/03/18 4

555555 2014/02/11 0

278999 2011/03/18 308

Sample Output

LV3

LV2

LV2

LV1

LV2

代码：

#include <iostream>

#include <map>

#include <stdio.h>

#include <algorithm>

using namespace std;

const int MAX\_N = 2000 + 100;

int n, T, cnt;

struct Rank

{

int ymd;

int score;

int id;

string lv;

};

int y, m, d, id, score;

Rank r[MAX\_N];

int a[MAX\_N];

int cmp(int m, int n)

{

if(r[m].score != r[n].score)

return r[m].score > r[n].score;

else if(r[m].score == r[n].score)

{

if(r[m].ymd != r[n].ymd)

return r[m].ymd < r[n].ymd;

else if(r[m].ymd == r[n].ymd)

return r[m].id < r[n].id;

}

}

int lv[6];

int main()

{

scanf("%d", &T);

while(T--)

{

scanf("%d", &n);

int cnt = 0;

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

{

scanf("%d %d/%d/%d %d", &r[i].id, &y, &m, &d, &r[i].score);

r[i].ymd = y \* 10000 + m \* 100 + d;

a[i] = i;

if(r[i].score != 0)

cnt++;

}

sort(a, a + n, cmp);

lv[6] = (int)(cnt \* 0.03);

lv[5] = (int)(cnt \* 0.07);

lv[4] = (int)(cnt \* 0.2);

lv[3] = (int)(cnt \* 0.3);

lv[2] = cnt - lv[6] - lv[5] - lv[4] - lv[3];

lv[1] = n - cnt;

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

{

if(lv[6] != 0)

{

r[a[i]].lv = "LV6";

lv[6]--;

}

else if(lv[5] != 0)

{

r[a[i]].lv = "LV5";

lv[5]--;

}

else if(lv[4] != 0)

{

r[a[i]].lv = "LV4";

lv[4]--;

}

else if(lv[3] != 0)

{

r[a[i]].lv = "LV3";

lv[3]--;

}

else if(lv[2] != 0)

{

r[a[i]].lv = "LV2";

lv[2]--;

}

else

r[a[i]].lv = "LV1";

}

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

cout << r[i].lv << endl;

}

return 0;

}

## zoj 3836 Circulation pipe

Darkgy is a transport pipe master. One day, due to some strange redstone signal, an Iron pipe changed its direction and make a part of the pipe system become a circulation pipe.

The circulation pipe consists of L unit pipe numbered from 0 to <var>L</var>-1. Every <var>K</var> ticks, an item will input into pipe 0, and it will be transported in pipes with 1 unit pipe length per tick speed from pipe 0 to pipe <var>L</var>-1. When it was transported into pipe <var>L</var>-1, its direction will reversed and will be transported from <var>L</var>-1 to 0. When it reached pipe 0, its direction will be reversed again.

This process will repeat until the moment when there are more than <var>C</var> items in one of the <var>L</var> pipes, <var>C</var> is the capacity of each pipe.

For example, if <var>L</var>=5, <var>K</var>=3, <var>C</var>=1.

In tick 0, the first item will input into pipe 0.

In tick 3, it will be transported into pipe 3 and the second item will input into pipe 0.

In tick 4, the first item reached pipe 4 and its direction reversed, at the same time, the second item moved into pipe 1.

In tick 6, the third item appeared in pipe 0, the first item moved into pipe 2 while the second item was in pipe 3. Though the first item and the second item crossed, but......it does not matter XD.

In tick 7, the first item and the third item meet in pipe 1, and pipe 1 blast.

Darkgy want to know in which tick, the circulation pipe will blast.

Input

There are large amount of test cases, for each test case, there will be only one line with 3 integers 1 ≤ <var>L</var>, <var>K</var>, <var>C</var> ≤ 104 which was mentioned in the description.

Output

For each test case, you should output only one line with an integer <var>T</var> which means the tick when circulation pipe was blast.

Sample Input

5 3 1

1 1 1

1 1 2

1 1 3

Sample Output

7

1

2

3

参考答案

#include<cstdio>

#include<iostream>

#include<cstring>

#define ll long long

using namespace std;

ll a,b,x,y,q;

ll l,k,c;

ll myabs(ll x){

return x>0?x:-x;

}

void extend\_gcd(ll a,ll b,ll &x,ll &y,ll &q){

if(b==0){x=1;y=0;q=a;}

else{

extend\_gcd(b,a%b,x,y,q);

ll tmp=x;x=y;y=tmp-a/b\*y;

}

}

ll forward(int i){

ll ans=0;

a=2\*(l-1),b=-k;

extend\_gcd(a,b,x,y,q);

ll t=k/myabs(q)\*a;

if(i==0||i==l-1) return (c-1)\*t+i;

if(2\*i%q) ans=(c-1)\*t+i;

else{

x\*=-2\*i/q;

ll tmp=x/(k/q),res;

res=x-k/q\*tmp;

if(res<0) res+=myabs(k/q);

if(c&1) ans=c/2\*t+i;

else ans=(c-1)/2\*t+res\*a+2\*i+a-i;

}

return ans;

}

ll backward(int i){

ll ans=0;

a=2\*(l-1),b=-k;

extend\_gcd(a,b,x,y,q);

ll t=k/myabs(q)\*a;

if(i==0||i==l-1) return (c-1)\*t+i;

if(2\*(l-i-1)%q) ans=(c-1)\*t+a-i;

else{

x\*=-2\*(l-i-1)/q;

ll tmp=x/(k/q),res;

res=x-k/q\*tmp;

if(res<0) res+=myabs(k/q);

if(c&1) ans=c/2\*t+a-i;

else ans=(c-1)/2\*t+res\*a+2\*(l-i-1)+i;

}

return ans;

}

int main()

{

ll res;

while(~scanf("%lld%lld%lld",&l,&k,&c)){

if(l==1) {printf("%lld\n",c\*k);continue;}

c+=1;

res=min(forward(0),backward(l-1));

for(int i=1;i<l-1;i++){

res=min(res,forward(i));

res=min(res,backward(i));

}

printf("%lld\n",res);

}

return 0;

}

## zoj 3890 Wumpus

One day Leon finds a very classic game called Wumpus.The game is as follow.

Once an agent fell into a cave. The legend said that in this cave lived a kind of monster called Wumpus, and there were horrible pits which could lead to death everywhere. However, there were also a huge amount of gold in the cave. The agent must be careful and sensitive so that he could grab all of the gold and climb out of the cave safely.

The cave can be regarded as a n\*n board. In each square there could be a Wumpus, a pit, a brick of gold, or nothing. The agent would be at position (0,0) at first and headed right.(As the picture below)

Wumpus1

For each step, there are six possible movements including going forward, turning left, turning right, shooting, grabbing the gold, and climbing out of the cave. If the agent steps into a square containing a pit or Wumpus, he will die. When the agent shoots, the Wumpus in front of him will die. The goal of the agent is to grab all of the gold and return to the starting position and climb out(it's OK if any Wumpus is still living).When a brick of gold is grabbed successfully, you will gain 1000 points. For each step you take, you will lose 10 points.

Your job is to help him compute the highest point he can possibly get.

For the purpose of simplification, we suppose that there is only one brick of gold and the agent cannot shoot the Wumpus.

If there is a pit at (0, 0), the agent dies immediately. There will not be a Wumpus at (0, 0).

Input

There are multiple cases. The first line will contain one integer <var>k</var> that indicates the number of cases.

For each case:

The first line will contain one integer <var>n</var> (n <= 20).

The following lines will contain three integers, each line shows a position of an object. The first one indicates the type of the object. 1 for Wumpus, 2 for pit and 3 for gold. Then the next two integers show the x and y coordinates of the object.

The input end with -1 -1 -1. (It is guaranteed that no two things appear in one position.)

Output

The output contains one line with one integer, which is the highest point Leon could possibly get. If he cannot finish the game with a non-negative score, print "-1".

Sample Input

2

3

1 1 1

2 2 0

3 2 2

-1 -1 -1

3

1 1 1

3 2 2

-1 -1 -1

Sample Output

850

870

Hint

For the sample 1, the following steps are taken:

turn left, forward, forward, turn right, forward, forward, grab, turn left, turn left, forward, forward, turn left, forward, forward, climb.

There are in all 15 steps, so the final score is 840. For the sample 2 , the path is as follow:

Wumpus2

参考答案

#include<cstdio>

#include<cmath>

#include<queue>

#include<vector>

#include<stack>

#include<map>

#include<string>

#include<cstring>

#include<algorithm>

#include<iostream>

using namespace std;

typedef long long ll;

const ll maxn = 25;

int T, n, flag, x, y;

int mp[maxn][maxn];

int f[maxn][maxn][4][2];

struct point

{

int x, y, d, v;

point(int x, int y, int d, int v) :x(x), y(y), d(d), v(v){}

};

int bfs()

{

queue<point> p;

p.push(point(0, 0, 0, 0));

if (mp[0][0] == 1) return -1;

f[0][0][0][0] = 0;

int x, y, d, v;

while (!p.empty())

{

point q = p.front(); p.pop();

if (f[q.x][q.y][q.d][q.v] > 98) continue;

x = q.x; y = q.y; v = q.v;

if (mp[x][y] == 3 && f[x][y][q.d][1] > f[x][y][q.d][0])

{

f[x][y][q.d][1] = f[x][y][q.d][0];

p.push(point(x, y, q.d, 1));

}

d = (q.d + 1) % 4;

if (f[x][y][d][v] > f[x][y][q.d][v] + 1)

{

f[x][y][d][v] = f[x][y][q.d][v] + 1;

p.push(point(x, y, d, v));

}

d = (q.d + 3) % 4;

if (f[x][y][d][v] > f[x][y][q.d][v] + 1)

{

f[x][y][d][v] = f[x][y][q.d][v] + 1;

p.push(point(x, y, d, v));

}

d = q.d;

if (d == 0) ++x;

if (d == 1) --y;

if (d == 2) --x;

if (d == 3) ++y;

if (x >= 0 && x < n&&y >= 0 && y<n)

if (mp[x][y] != 1 && mp[x][y] != 2)

if (f[x][y][d][v] > f[q.x][q.y][d][v] + 1)

{

f[x][y][d][v] = f[q.x][q.y][d][v] + 1;

p.push(point(x, y, d, v));

}

}

int ans = f[0][0][0][1];

for (int i = 1; i < 4; i++) ans = min(ans, f[0][0][i][1]);

ans = 980 - ans \* 10;

if (ans < 0) return -1;

return ans;

}

int main()

{

scanf("%d", &T);

while (T--)

{

scanf("%d", &n);

memset(mp, 0, sizeof(mp));

while (scanf("%d%d%d", &flag, &x, &y) == 3)

{

if (x == -1 && y == -1 && flag == -1) break;

mp[x][y] = flag;

}

memset(f, 1, sizeof(f));

printf("%d\n", bfs());

}

return 0;

}

# 第二部分 HDU

## HDU 1062 Text Reverse

Ignatius likes to write words in reverse way. Given a single line of text which is written by Ignatius, you should reverse all the words and then output them.

输入描述:

The input contains several test cases. The first line of the input is a single integer T which is the number of test cases. T test cases follow.

Each test case contains a single line with several words. There will be at most 1000 characters in a line.

输出描述:

For each test case, you should output the text which is processed.

参考答案

#include<iostream>

#include<string.h>

#include<stdio.h>

using namespace std;

const int Max = 1000 + 10;

char str[Max];

char w[Max];

int main()

{

int cases;

cin >> cases;

char c = getchar();

while(cases--)

{

gets(str);

int pos = 0;

int len = strlen(str);

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

{

if(str[i] == ' ')

{

w[pos++] = '\0';

strrev(w);

printf("%s ", w);

pos = 0;

}

else if(str[i] == '\0')

{

w[pos++] = '\0';

strrev(w);

printf("%s", w);

pos = 0;

}

else

{

w[pos++] = str[i];

}

}

printf("\n");

}

return 0;

}

## HDU 1214 圆桌会议

HDU ACM集训队的队员在暑假集训时经常要讨论自己在做题中遇到的问题.每当面临自己解决不了的问题时,他们就会围坐在一张圆形的桌子旁进行交流,经过大家的讨论后一般没有解决不了的问题,这也只有HDU ACM集训队特有的圆桌会议,有一天你也可以进来体会一下哦:),在一天在讨论的时候,Eddy想出了一个极为古怪的想法,如果他们在每一分钟内,一对相邻的两个ACM队员交换一下位子,那么要多少时间才能得到与原始状态相反的座位顺序呢?(即对于每个队员，原先在他左面的队员后来在他右面，原先在他右面的队员在他左面),这当然难不倒其他的聪明的其他队友们,马上就把这个古怪的问题给解决了,你知道是怎么解决的吗?

输入描述:

对于给定数目N(1<=N<=32767)，表示有N个人,求要多少时间才能得到与原始状态相反的座位顺序(reverse)即对于每个人，原先在他左面的人后来在他右面，原先在他右面的人在他左面。

输出描述:

对每个数据输出一行，表示需要的时间(以分钟为单位)

参考答案

#include<stdio.h>

int main()

{

int n,a;

while(~scanf("%d",&n))

{

a=n/2;

n-=a;

printf("%d\n",a\*(a-1)/2+n\*(n-1)/2);

}

return 0;

}

## HDU 1269 迷宫城堡

为了训练小希的方向感，Gardon建立了一座大城堡，里面有N个房间(N<=10000)和M条通道(M<=100000)，每个通道都是单向的，就是说若称某通道连通了A房间和B房间，只说明可以通过这个通道由A房间到达B房间，但并不说明通过它可以由B房间到达A房间。Gardon需要请你写个程序确认一下是否任意两个房间都是相互连通的，即：对于任意的i和j，至少存在一条路径可以从房间i到房间j，也存在一条路径可以从房间j到房间i。

输入描述:

输入包含多组数据，输入的第一行有两个数：N和M，接下来的M行每行有两个数a和b，表示了一条通道可以从A房间来到B房间。文件最后以两个0结束。

输出描述:

对于输入的每组数据，如果任意两个房间都是相互连接的，输出"Yes"，否则输出"No"。

代码：

#include<stdio.h>

#include<string.h>

int pre[2][100010];

int n,m;

int find(int a,int i)

{

int r=a;

while(r!=pre[i][r])

{

r=pre[i][r];

}

return r;

}

void fun(int a,int b)

{

if(a!=n)

{

int fa=find(a,0),fb=find(b,0);

if(fa!=fb)

pre[0][a]=b;

}

if(b!=n)

{

int fa=find(a,1),fb=find(b,1);

if(fa!=fb)

pre[1][b]=a;

}

}

int main()

{

while(scanf("%d%d",&n,&m)!=EOF,n||m)

{

int i,w=1;

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

{

pre[0][i]=pre[1][i]=i;

}

while(m--)

{

int a,b;

scanf("%d%d",&a,&b);

fun(a,b);

}

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

{

if(find(i,0)!=n||find(i,1)!=n)

{

w=0;

break;

}

}

if(w)

printf("Yes\n");

else

printf("No\n");

}

}

## HDU 1438 钥匙计数之一

一把锁匙有N个槽，槽深为1，2，3，4。每锁匙至少有3个不同的深度且至少有1对相连的槽其深度之差为3。求这样的锁匙的总数。

输入描述:

本题无输入

输出描述:

对N>=2且N<=31，输出满足要求的锁匙的总数。

代码：

#include<iostream>

#include<cmath>

using namespace std;

int main(){

int i;

long long a[32] = { 0, 0, 0 }, c = 0, b = 2,t;

for (i = 3; i < 32; ++i)

{

a[i] = a[i - 1] \* 4 + (long long)(pow(2, i - 1) - 2) \* 2 + (c + b) \* 2;

t = c + b\*2;

b = (c + b)\*2 + 2;

c = t;

}

for (i = 2; i < 32; ++i)

cout << "N=" << i <<": " << a[i] << endl;

}

## HDU 1527 取石子游戏

有两堆石子，数量任意，可以不同。游戏开始由两个人轮流取石子。游戏规定，每次有两种不同的取法，一是可以在任意的一堆中取走任意多的石子；二是可以在两堆中同时取走相同数量的石子。最后把石子全部取完者为胜者。现在给出初始的两堆石子的数目，如果轮到你先取，假设双方都采取最好的策略，问最后你是胜者还是败者。

输入描述:

输入包含若干行，表示若干种石子的初始情况，其中每一行包含两个非负整数a和b，表示两堆石子的数目，a和b都不大于1,000,000,000。

输出描述:

输出对应也有若干行，每行包含一个数字1或0，如果最后你是胜者，则为1，反之，则为0。

代码：

#include<stdio.h>

#include<math.h>

int main()

{

int a,b;

while(scanf("%d%d",&a,&b)==2)

{

double j,k,r,R;

int t;

if(a>b)

{

t=a;

a=b;

b=t;

}

r=(sqrt(5.0)-1)/2;

R=1.0/r;

j=(int)(r\*a);

if(a!=(int)(j\*R))

j+=1;

printf("%d\n",b!=(int)(j\*R)+j);

}

return 0;

}

## HDU 1622 Trees on the level

Trees are fundamental in many branches of computer science. Current state-of-the art parallel computers such as Thinking Machines' CM-5 are based on fat trees. Quad- and octal-trees are fundamental to many algorithms in computer graphics.

This problem involves building and traversing binary trees.

Given a sequence of binary trees, you are to write a program that prints a level-order traversal of each tree. In this problem each node of a binary tree contains a positive integer and all binary trees have have fewer than 256 nodes.

In a level-order traversal of a tree, the data in all nodes at a given level are printed in left-to-right order and all nodes at level k are printed before all nodes at level k+1.

For example, a level order traversal of the tree

is: 5, 4, 8, 11, 13, 4, 7, 2, 1.

In this problem a binary tree is specified by a sequence of pairs (n,s) where n is the value at the node whose path from the root is given by the string s. A path is given be a sequence of L's and R's where L indicates a left branch and R indicates a right branch. In the tree diagrammed above, the node containing 13 is specified by (13,RL), and the node containing 2 is specified by (2,LLR). The root node is specified by (5,) where the empty string indicates the path from the root to itself. A binary tree is considered to be completely specified if every node on all root-to-node paths in the tree is given a value exactly once.

输入描述:

The input is a sequence of binary trees specified as described above. Each tree in a sequence consists of several pairs (n,s) as described above separated by whitespace. The last entry in each tree is (). No whitespace appears between left and right parentheses.

All nodes contain a positive integer. Every tree in the input will consist of at least one node and no more than 256 nodes. Input is terminated by end-of-file.

输出描述:

For each completely specified binary tree in the input file, the level order traversal of that tree should be printed. If a tree is not completely specified, i.e., some node in the tree is NOT given a value or a node is given a value more than once, then the string ``not complete'' should be printed

参考答案

#include<algorithm>

#include<iostream>

#include<string.h>

#include<cstdlib>

#include<cstdio>

#include<vector>

#include<queue>

using std::queue;

using std::vector;

const int Max\_N = 260;

struct Node {

int v, vis;

Node \*ch[2];

inline void set(int \_v, Node \*p) {

vis = 0, v = \_v;

ch[0] = ch[1] = p;

}

};

struct BinTree {

int fail;

char buf[Max\_N];

Node \*tail, \*root, stack[Max\_N];

void init() {

fail = 0;

tail = &stack[0];

}

inline Node \*newNode(int v = 0) {

Node \*p = tail++;

p->set(v, NULL);

return p;

}

inline void insert(const char \*src, const int v) {

int n = strlen(src);

Node \*x = root;

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

if (src[i] == 'L') {

if (!x->ch[0]) x->ch[0] = newNode();

x = x->ch[0];

} else if (src[i] == 'R') {

if (!x->ch[1]) x->ch[1] = newNode();

x = x->ch[1];

}

}

if (x->vis) fail = 1;

x->v = v;

x->vis = 1;

}

inline void bfs() {

vector<int> ans;

queue<Node \*> que;

que.push(root);

while (!que.empty()) {

Node \*u = que.front(); que.pop();

if (!u->vis) {

fail = 1;

break;

}

ans.push\_back(u->v);

if (u->ch[0]) que.push(u->ch[0]);

if (u->ch[1]) que.push(u->ch[1]);

}

if (fail) {

puts("not complete");

return;

}

int n = ans.size();

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

printf("%d%c", ans[i], i < n - 1 ? ' ' : '\n');

}

}

inline int gogo() {

init();

int v = 0;

root = newNode();

for (;;) {

if (scanf("%s", buf) != 1) return 0;

if (!strcmp(buf, "()")) break;

sscanf(&buf[1], "%d", &v);

insert(strchr(buf, ',') + 1, v);

}

bfs();

return 1;

}

}tree;

int main() {

while (tree.gogo());

return 0;

}

## HDU 1969 Pie

My birthday is coming up and traditionally I'm serving pie. Not just one pie, no, I have a number N of them, of various tastes and of various sizes. F of my friends are coming to my party and each of them gets a piece of pie. This should be one piece of one pie, not several small pieces since that looks messy. This piece can be one whole pie though.

My friends are very annoying and if one of them gets a bigger piece than the others, they start complaining. Therefore all of them should get equally sized (but not necessarily equally shaped) pieces, even if this leads to some pie getting spoiled (which is better than spoiling the party). Of course, I want a piece of pie for myself too, and that piece should also be of the same size.

What is the largest possible piece size all of us can get? All the pies are cylindrical in shape and they all have the same height 1, but the radii of the pies can be different.

输入描述:

One line with a positive integer: the number of test cases. Then for each test case:

---One line with two integers N and F with 1 <= N, F <= 10 000: the number of pies and the number of friends.

---One line with N integers ri with 1 <= ri <= 10 000: the radii of the pies.

输出描述:

For each test case, output one line with the largest possible volume V such that me and my friends can all get a pie piece of size V. The answer should be given as a floating point number with an absolute error of at most 10^(-3).

代码：

#include<stdio.h>

#include<string.h>

#include<math.h>

#define pi acos(-1.0)

#define eps 1e-7

int main()

{

int n,t,f,i,r,pep;

double s[10005];

double max,min,mid;

scanf("%d",&t);

while(t--)

{

min=0;max=0;

scanf("%d%d",&n,&f);

f++;

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

{

scanf("%d",&r);

s[i]=pi\*r\*r;

if(max<s[i]) max=s[i];

}

while(max-min>eps)

{

mid=(max+min)/2;

pep=0;

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

{

pep+=(int)(s[i]/mid);

}

if(pep>=f)

min=mid;

else

max=mid;

}

printf("%.4lf\n",mid);

}

return 0;

}

## HDU 2156 分数矩阵

我们定义如下矩阵:

1/1 1/2 1/3

1/2 1/1 1/2

1/3 1/2 1/1

矩阵对角线上的元素始终是1/1，对角线两边分数的分母逐个递增。

请求出这个矩阵的总和。

输入描述:

每行给定整数N (N<50000)，表示矩阵为 N\*N.当N为0时，输入结束。

输出描述:

输出答案，保留2位小数。

参考答案

#include<stdio.h>

int main(){

double m,j,i;

while(scanf("%lf",&m)!=EOF&&m!=0){

j=0;int k=1;

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

j=j+1.00/i\*k++;

printf("%.2f\n",2\*j-m);}

return 0;

}

## HDU 2212 DFS

A DFS(digital factorial sum) number is found by summing the factorial of every digit of a positive integer.

For example ,consider the positive integer 145 = 1!+4!+5!, so it's a DFS number.

Now you should find out all the DFS numbers in the range of int( [1, 2147483647] ).

There is no input for this problem. Output all the DFS numbers in increasing order. The first 2 lines of the output are shown below.

输入描述:

no input

输出描述:

Output all the DFS number in increasing order.

代码：

#include <stdio.h>

int jieceng(int a)

{

int s,i;

s=1;

for(i=2;i<=a;i++)

s\*=i;

return s;

}

int weishu(int a)

{

int sum=0;

int b;

b=a;

while(a)

{

sum+=jieceng(a%10);

a/=10;

}

if(sum==b)

return 1;

return 0;

}

int main()

{

int n,i;

for(i=1;i<=50000;i++)

{

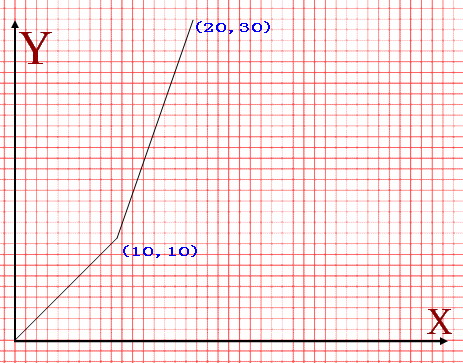
if(weishu(i))

printf("%d\n",i);

}

}

## HDU 2241 考研路茫茫――早起看书

考研并不是说说就可以了，要付诸于行动。   
  
对于Lele来说，最痛苦的事莫过于早起看书了，不过为了考研，也就豁出去了。由于早起看书会对看书效率产生影响，所以对于要什么时候起床看书，还是有必要考虑的。   
  
经过周密的调查，Lele发现早起的时间会对上午和下午的看书效率都产生影响，具体如下：   
  
他把早起的程度标记为一个非负有理数X，X数值越大，表示越早起。   
  
1.对上午的影响F：符合 F = N / (X^2) 。其中N是一个参数。即越早起床，对上午的效率影响越少。   
  
2.对下午的影响Y：一般越早起，对下午的效率影响越大。不过Y和X的关系比较复杂，并且在不同时候关系也是不同的，于是Lele把它绘制成为函数图形了。在某天，函数图形如下。   
  
<center> </center>  
  
X轴的值表示早起的程度，Y轴的值表示对下午看书效率的影响。函数图像为折线上升的。   
  
不过由于N值和Y-X的图像并不确定，所以Lele每次都要进行大量工作，来确保对整天的看书效率影响最小(F+Y的值最小)，现在就请你帮帮他吧。   
  
记住早起时间的取值X一定要在折线包含的范围之内。(对于上面这个图象，X一定要在[0,20]之内)。 

输入描述:

本题目包含多组输入，请处理到文件结束。

每组测试第一行包含两个整数M和N(1<M<10000,0<=N<=2^31)。其中M表示X-Y图像中顶点的数目。N含义见题目描述。

接下来有M行整数，分别表示这M个点在图像中的坐标Xi和Yi，Xi和Yi范围在[0,2^30]之内。

注意，第一个坐标一定为(0,0)，并且X坐标和Y坐标是不降的，即对于任意 i<j Xi<Xj 且 Yi<=Yj。

而Lele早起的时间一定在[0,Xm-1]这个范围之内。

输出描述:

对于每组数据，请在一行内输出可能取到的对全天效率(Y+F)影响的最小值。

结果保留三位小数

代码：

#include <iostream>

#include <math.h>

using namespace std;

#define N 10005

#define EPS 1e-10

double MIN;

double n;

double x[N],y[N];

double yyy(double xx,int id)

{

double yy,k;

k=(y[id+1]-y[id])/(x[id+1]-x[id]);

yy=y[id]+k\*(xx-x[id]);

return yy;

}

void san\_fen(int m)

{

int i;

double x1,y1,x2,y2,midx1,midx2,d,elem1,elem2;

for(i=0;i<m-1;i++)

{

x1=x[i];y1=y[i];

x2=x[i+1];y2=y[i+1];

d=(y2-y1)/(x2-x1);

do

{

midx1=(x1+x2)/2;

midx2=(midx1+x2)/2;

elem1=n/(midx1\*midx1)+yyy(midx1,i);

elem2=n/(midx2\*midx2)+yyy(midx2,i);

if(elem1>elem2)

x1=midx1;

else if(elem1<elem2)

x2=midx2;

}

while(fabs(elem1-elem2)>EPS);

if(MIN>elem1)

MIN=elem1;

}

}

int main()

{

int m,i;

while(scanf("%d%lf",&m,&n)!=EOF)

{

MIN=999999.0;

for(i=0;i<m;i++)

scanf("%lf%lf",&x[i],&y[i]);

san\_fen(m);

printf("%.3lf\n",MIN);

}

return 0;

}

## HDU 2306 Kingdom

King Kong is the feared but fair ruler of Transylvania. The kingdom consists of two cities and N < 150 towns, with nonintersecting roads between some of them. The roads are bidirectional, and it takes the same amount of time to travel them in both directions. Kong has G < 353535 soldiers.

Due to increased smuggling of goat cheese between the two cities, Kong has to place his soldiers on some of the roads in such a way that it is impossible to go from one city to the other without passing a soldier. The soldiers must not be placed inside a town, but may be placed on a road, as close as Kong wishes, to any town. Any number of soldiers may be placed on the same road. However, should any of the two cities be attacked by a foreign army, the king must be able to move all his soldiers fast to the attacked city. Help him place the soldiers in such a way that this mobilizing time is minimized.

Note that the soldiers cannot be placed in any of the cities or towns. The cities have ZIP-codes 95050 and 104729, whereas the towns have ZIPcodes from 0 to N - 1. There will be at most one road between any given pair of towns or cities.

输入描述:

The input contains several test cases. The first line of each test case is N, G and E, where N and G are as defined above and E < 5000 is the number of roads. Then follow E lines, each of which contains three integers: A and B, the ZIP codes of the endpoints, and φ, the time required to travel the road,φ < 1000. The last line of the input is a line containing a single 0.

输出描述:

For each test case in the input, print the best mobilizing time possible, with one decimal. If the given number of soldiers is not enough to stop the goat cheese, print "Impossible" instead.

代码：

#include<iostream>

#include<string.h>

#define INF 1000000000

#define lINF 400000

const int pN=200,eN=20000;

int f\_min(int x,int y){

return x<y?x:y;

}

int f\_max(int x,int y){

return x>y?x:y;

}

int f\_min(int a,int b,int c,int d){

return f\_min(f\_min(a,b),f\_min(c,d));

}

struct Edge{

int u,v,next;

int w,l;

};

Edge edge[eN];

int en,head[pN],cur[pN],ps[pN],dep[pN];

void insert(int u,int v,int l){

edge[en].u=u;

edge[en].v=v;

edge[en].l=l;

edge[en].next=head[u];

head[u]=en++;

edge[en].u=v;

edge[en].v=u;

edge[en].l=l;

edge[en].next=head[v];

head[v]=en++;

}

int max\_flow(int n,int s,int t){

int tr,res=0;

int i,j,k,f,r,top;

while(1){

memset(dep,-1,n\*sizeof(int));

for(f=dep[ps[0]=s]=0,r=1;f!=r;)

for(i=ps[f++],j=head[i];j!=-1;j=edge[j].next){

if(edge[j].w&&-1==dep[k=edge[j].v]){

dep[k]=dep[i]+1;

ps[r++]=k;

if(k==t){

f=r;

break;

}

}

}

if(-1==dep[t])break;

memcpy(cur,head,n\*sizeof(int));

for(i=s,top=0;;){

if(i==t){

for(k=0,tr=INF;k<top;++k)

if(edge[ps[k]].w<tr)

tr=edge[ps[f=k]].w;

for(k=0;k<top;++k)

edge[ps[k]].w-=tr,edge[ps[k]^1].w+=tr;

res+=tr;

i=edge[ps[top=f]].u;

}

for(j=cur[i];cur[i]!=-1;j=cur[i]=edge[cur[i]].next)

if(edge[j].w&&dep[i]+1==dep[edge[j].v])break;

if(cur[i]!=-1){

ps[top++]=cur[i];

i=edge[cur[i]].v;

}else{

if(0==top)break;

dep[i]=-1;

i=edge[ps[--top]].u;

}

}

}

return res;

}

int N,K,M;

int tran(int x){

if(x==95050)return N-2;

if(x==104729)return N-1;

return x;

}

int diss[200],dise[200];

void get\_data(){

scanf("%d%d",&K,&M);

int u,v,l;

memset(head,-1,sizeof(head));en=0;

N+=2;

while(M--){

scanf("%d%d%d",&u,&v,&l);

u=tran(u);

v=tran(v);

insert(u,v,l);

}

}

int que[100000],qhead,qtail;

bool inque[200];

void spfa(int x[200],int s){

int u,v,i;

for(i=0;i<N;i++)x[i]=INF;

qhead=qtail=0;

memset(inque,0,sizeof(inque));

inque[s]=1;x[s]=0;

que[qtail++]=s;

while(qtail>qhead){

u=que[qhead++];inque[u]=0;

for(i=head[u];i!=-1;i=edge[i].next){

v=edge[i].v;

if(x[v]>x[u]+edge[i].l){

x[v]=x[u]+edge[i].l;

if(!inque[v]){

inque[v]=1;

que[qtail++]=v;

}

}

}

}

}

void f\_swap(int &a,int &b){

int t=a;

a=b;

b=t;

}

int cal(int l1,int l2,int v){

if(l1>l2)f\_swap(l1,l2);

if(l1+v<=l2)return l2\*2;

return l1+v+l2;

}

int get\_w(int i){

int s1,s2,e1,e2,temp;

s1=diss[edge[i].u];s2=diss[edge[i].v];

e1=dise[edge[i].u];e2=dise[edge[i].v];

temp=f\_min(2\*f\_max(s1,e1),2\*f\_max(s2,e2));

temp=f\_min(temp,cal(s1,e2,edge[i].l));

temp=f\_min(temp,cal(s2,e1,edge[i].l));

return temp;

}

void get\_dis\_w(){

spfa(diss,N-2);

spfa(dise,N-1);

int i;

for(i=0;i<en;i+=2){

edge[i].l=get\_w(i);

edge[i^1].l=edge[i].l;

}

}

bool ok(int lim){

int i;

for(i=0;i<en;i+=2){

if(edge[i].l<=lim){

edge[i].w=edge[i^1].w=1;

}else{

edge[i].w=edge[i^1].w=lINF;

}

}

int temp=max\_flow(N,N-2,N-1);

if(temp>K)return 0;

return 1;

}

void run(){

get\_dis\_w();

int h,l,mid;

h=200000;l=0;

while(h>l){

mid=(h+l)>>1;

if(ok(mid))h=mid;

else l=mid+1;

}

if(h==200000)printf("Impossible\n");

else printf("%.1lf\n",h\*0.5);

}

int main(){

while(scanf("%d",&N),N){

get\_data();

run();

}

return 0;

}

## HDU 2399 GPA

Each course grade is one of the following five letters: A, B, C, D, and F. (Note that there is no grade E.) The grade A indicates superior achievement , whereas F stands for failure. In order to calculate the GPA, the letter grades A, B, C, D, and F are assigned the following grade points, respectively: 4, 3, 2, 1, and 0.

输入描述:

The input file will contain data for one or more test cases, one test case per line. On each line there will be one or more upper case letters, separated by blank spaces.

输出描述:

Each line of input will result in exactly one line of output. If all upper case letters on a particular line of input came from the set {A, B, C, D, F} then the output will consist of the GPA, displayed with a precision of two decimal places. Otherwise, the message "Unknown letter grade in input" will be printed.

参考答案

#include<stdio.h>

#include<string.h>

int main()

{

char s[1000];

int i,num,flag;

double t;

while(gets(s))

{

t=0;

num=0;

i=0;

flag=0;

while(s[i])

{

switch(s[i++])

{

case 'A':

t+=4.0;

num++;

continue;

case 'B':

t+=3.0;

num++;

continue;

case 'C':

t+=2.0;

num++;

continue;

case 'D':

t+=1.0;

num++;

continue;

case 'F':

num++;

continue;

case ' ':

continue;

default:

flag=1;

break;

}

}

if(flag==1)

printf("Unknown letter grade in input\n");

else

printf("%.2lf\n",t/num);

}

return 0;

}

## HDU 2498 Digits



A googol written out in decimal has 101 digits. A googolplex has one plus a googol digits. That's a lot of digits!

Given any number x0 , define a sequence using the following recurrence:

<center>xi+1 = the number of digits in the decimal representation of xi </center>

Your task is to determine the smallest positive i such that xi = xi-1 .

输入描述:

Input consists of several lines. Each line contains a value of x0. Every value of x0 is non-negative and has no more than one million digits. The last line of input contains the word END.

输出描述:

For each value of x0 given in the input, output one line containing the smallest positive i such that xi =

xi-1.

参考答案

#include<iostream>

using namespace std;

int f(int x)

{

int sum=0;

while(x)

{

x=x/10;

sum++;

}

return sum;

}

int main()

{

string a;

while(cin>>a)

{

if(a[0]=='E'){break;}

int l=a.size();

if(l==1&&a[0]=='1'){cout<<1<<endl;continue;}

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

{

if(l==f(l))

{

cout<<i+1<<endl;break;

}

else

{

l=f(l);

}

}

}

return 0;

}

## HDU 2509 Be the Winner

Let's consider m apples divided into n groups. Each group contains no more than 100 apples, arranged in a line. You can take any number of consecutive apples at one time.

For example "@@@" can be turned into "@@" or "@" or "@ @"(two piles). two people get apples one after another and the one who takes the last is

the loser. Fra wants to know in which situations he can win by playing strategies (that is, no matter what action the rival takes, fra will win).

输入描述:

You will be given several cases. Each test case begins with a single number n (1 <= n <= 100), followed by a line with n numbers, the number of apples in each pile. There is a blank line between cases.

输出描述:

If a winning strategies can be found, print a single line with "Yes", otherwise print "No".

参考答案

#include<iostream>

using namespace std;

int main()

{

int N,x,i,s,a;

while(cin>>N)

{

s=a=0;

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

{

cin>>x;

s^=x;

if(x>1)

a=1;

}

if(a==0)

{

if(N%2)

cout<<"No"<<endl;

else

cout<<"Yes"<<endl;

}

else

{

if(s==0)

cout<<"No"<<endl;

else

cout<<"Yes"<<endl;

}

}

return 0;

}

## HDU 2568 前进

轻松通过墓碑，进入古墓后，才发现里面别有洞天。

突然，Yifenfei发现自己周围是黑压压的一群蝙蝠，个个扇动翅膀正准备一起向他发起进攻！

形势十分危急！

好在此时的yifenfei已经不是以前那个经常被lemon抢走MM的菜鸟了！面对众多蝙蝠的嗜血狂攻，只见yifenfei使出轻灵的剑法，刷，刷，刷，瞬间搞定……

现已知yifenfei使用了2招（剑招A和剑招B）：剑招A，一招能杀死一半的蝙蝠。但是如果当前的蝙蝠数为奇数，那么就必须先出一招剑招B杀死其中任意一个，使蝙蝠数为偶数，再出剑招A。

现在请问：杀死n只蝙蝠需要使出多少招剑招B？



输入描述:

输入数据首先给出一个整数C，表示测试组数。

然后是C组数据，每组包含一个正整数n (n<2^31)。

输出描述:

对应每组数据，请输出一个整数，表示yifenfei使用的剑招B的数目，每组输出占一行。

参考答案

#include<stdio.h>

int main()

{

int C;

scanf("%d",&C);

while(C--)

{

int n,k=0;

scanf("%d",&n);

while(n!=0)

{

while(n%2==0)

{

n=n/2;

}

while(n%2!=0)

{

n--;

k++;

}

}

printf("%d\n",k);

}

return 0;

}

## HDU 2719 The Seven Percent Solution

Uniform Resource Identifiers (or URIs) are strings like http://icpc.baylor.edu/icpc/, mailto:foo@bar.org, ftp://127.0.0.1/pub/linux, or even just readme.txt that are used to identify a resource, usually on the Internet or a local computer. Certain characters are reserved within URIs, and if a reserved character is part of an identifier then it must be percent-encoded by replacing it with a percent sign followed by two hexadecimal digits representing the ASCII code of the character. A table of seven reserved characters and their encodings is shown below. Your job is to write a program that can percent-encode a string of characters.

Character Encoding

" " (space) %20

"!" (exclamation point) %21

"$" (dollar sign) %24

"%" (percent sign) %25

"(" (left parenthesis) %28

")" (right parenthesis) %29

"\*" (asterisk) %2a

输入描述:

The input consists of one or more strings, each 1�79 characters long and on a line by itself, followed by a line containing only "#" that signals the end of the input. The character "#" is used only as an end-of-input marker and will not appear anywhere else in the input. A string may contain spaces, but not at the beginning or end of the string, and there will never be two or more consecutive spaces.

输出描述:

For each input string, replace every occurrence of a reserved character in the table above by its percent-encoding, exactly as shown, and output the resulting string on a line by itself. Note that the percent-encoding for an asterisk is %2a (with a lowercase "a") rather than %2A (with an uppercase "A").

代码：

#include <stdio.h>

#include <string.h>

int main()

{

int i;

char s[85];

while(gets(s)&&s[0]!='#')

{

int l=strlen(s);

for(i=0;i<l;i++)

{

if(s[i]==' ')

s[i]=1;

if(s[i]=='!')

s[i]=2;

if(s[i]=='$')

s[i]=3;

if(s[i]=='%')

s[i]=4;

if(s[i]=='(')

s[i]=5;

if(s[i]==')')

s[i]=6;

if(s[i]=='\*')

s[i]=7;

}

for(i=0;i<l;i++)

{

if(s[i]==1)

printf("%%20");

if(s[i]==2)

printf("%%21");

if(s[i]==3)

printf("%%24");

if(s[i]==4)

printf("%%25");

if(s[i]==5)

printf("%%28");

if(s[i]==6)

printf("%%29");

if(s[i]==7)

printf("%%2a");

if(s[i]!=1 &&s[i]!=2 &&s[i]!=3 &&s[i]!=4&&s[i]!=5 &&s[i]!=6 &&s[i]!=7)

printf("%c",s[i]);

}

printf("\n");

}

return 0;

}

## HDU 2840 Self-Replicating Numbers

Sherlock is fond of playing with numbers. Two days ago he discovered that 9376^2 = 87909376 - the last four digits constitute 9376 again. He called such numbers self-replicating.

More precisely, an n-digit number is called self-replicating if it is equal to the number formed by the last n digits of its square. Now Sherlock often asks Xay to help him to find new such numbers. To make the things worse, Sherlock already knows what the scales of notation are, so he asks Xay to find, for example, hexadecimal or binary self-replicating numbers.

Xay wants to help Sherlock, but unfortunately he is very busy now: he is seriously preparing and training for the next ACM Regional Contest. So he asked you to write a program that for a given base b and length n will find all n-digit self-replicating numbers in the scale of notation with base b.

输入描述:

there are multiple test cases. one line of each test case contains two integer numbers b and n separated by a single space, the base b of the scale of notation (2 ≤ b ≤ 36) and the required length n (1 ≤ n ≤ 2000).

输出描述:

For eacho test case, the first line contains K - the total number of self-replicating numbers of length n in base b. Next K lines contain one n-digit number in base b each. Uppercase Latin letters from A to Z must be used to represent digits from 10 to 35. The self-replicating numbers should be output int lexicographic order.

参考答案

#include<iostream>

#include<cstdio>

#include<list>

#include<algorithm>

#include<cstring>

#include<string>

#include<stack>

#include<map>

#include<vector>

#include<cmath>

#include<memory.h>

#include<set>

#include<cctype>

#define ll long long

#define LL \_\_int64

#define eps 1e-8

#define inf 0xfffffff

using namespace std;

typedef struct Node {

char s[2000 + 10];

};

Node sasd[2000 + 10];

int b,n;

int num[2000 + 10];

int tot;

void clear() {

memset(num,0,sizeof(num));

tot = 0;

}

void dfs(int pos,int e) {

if(pos == n) {

if(n != 1 && num[pos - 1] == 0)return;

for(int i=pos - 1;i>=0;i--) {

if(num[i] >= 10)

sasd[tot].s[n - i - 1] = 'A' + num[i] - 10;

else

sasd[tot].s[n - i - 1] = '0' + num[i];

}

sasd[tot++].s[n] = 0;

return;

}

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

num[pos] = i;

int ans = 0;

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

ans += num[j] \* num[pos - j];

if((ans + e)%b == i)

dfs(pos + 1,(ans + e)/b);

}

}

bool cmp(Node x,Node y) {

return strcmp(x.s,y.s) < 0;

}

int main() {

while(scanf("%d %d",&b,&n) == 2) {

clear();

dfs(0,0);

sort(sasd,sasd + tot,cmp);

printf("%d\n",tot);

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

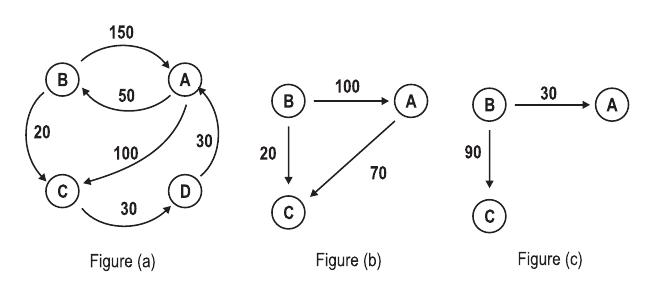
printf("%s\n",sasd[i].s);

}

return 0;

}

## HDU 2924 Think I&rsquo;ll Buy Me a Football Team

Falling Stocks. Bankrupted companies. Banks with no Cash. Seems like the best time to invest: ``Think I'll buy me a football team!"   
  
No seriously, I think I have the solution to at least the problem of cash in banks. Banks nowadays are all owing each other great amounts of money and no bank has enough cash to pay other banks' debts even though, on paper at least, they should have enough money to do so. Take for example the inter-bank loans shown in figure (a). The graph shows the amounts owed between four banks (A ...D). For example, A owes B 50M while, at the same time, B owes A 150M. (It is quite common for two banks to owe each other at the same time.) A total amount of 380M in cash is needed to settle all debts between the banks.   
<center> </center>  
  
In an attempt to decrease the need for cash, and after studying the example carefully, I concluded that there's a lot of cash being transferred unnecessarily. Take a look:   
  
1. C owes D the same amount as D owes A, so we can say that C owes A an amount of 30M and get D out of the picture.   
2. But since A already owes C 100M, we can say that A owes C an amount of 70M.   
3. Similarly, B owes A 100M only, (since A already owes B 50M.) This reduces the above graph to the one shown in figure (b) which reduces the needed cash amount to 190M (A reduction of 200M, or 53%.)   
4. I can still do better. Rather than B paying A 100M and A paying 70M to C, B can pay 70M (out of A's 100M) directly to C. This reduces the graph to the one shown in figure (c). Banks can settle all their debts with only 120M in cash. A total reduction of 260M or 68%. Amazing!   
  
I have data about inter-bank debts but I can't seem to be able to process it to obtain the minimum amount of cash needed to settle all the debts. Could you please write a program to do that? 

输入描述:

Your program will be tested on one or more test cases. Each test case is specified on N + 1 lines where N < 1, 000 is the number of banks and is specified on the first line. The remaining N lines specifies the inter-bank debts using an N×N adjacency matrix (with zero diagonal) specified in row-major order. The ith row specifies the amounts owed by the ith bank. Amounts are separated by one or more spaces. All amounts are less than 1000. The last line of the input file has a single 0.

输出描述:

For each test case, print the result using the following format:

k . B A   
  
  
where k is the test case number (starting at 1,) is a space character, B is the amount of cash needed before reduction and A is the amount of cash after reduction.

代码：

#include<iostream>

#include<cstdio>

using namespace std;

const int M = 1010;

int Map[M][M];

int C[M];

int R[M];

int main()

{

int n;

int k = 0;

while(scanf("%d",&n),n)

{

memset(C,0,sizeof(C));

memset(R,0,sizeof(R));

int ans = 0;

int res = 0;

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

{

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

{

cin>>Map[i][j];

ans += Map[i][j];

C[i] += Map[i][j];

}

}

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

{

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

{

R[i] += Map[j][i];

}

int cnt = C[i] - R[i];

if(cnt > 0)

{

res += cnt;

}

}

printf("%d. %d %d\n",++k,ans,res);

}

return 0;

}

## HDU 3065 病毒侵袭持续中

小t非常感谢大家帮忙解决了他的上一个问题。然而病毒侵袭持续中。在小t的不懈努力下，他发现了网路中的“万恶之源”。这是一个庞大的病毒网站，他有着好多好多的病毒，但是这个网站包含的病毒很奇怪，这些病毒的特征码很短，而且只包含“英文大写字符”。当然小t好想好想为民除害，但是小t从来不打没有准备的战争。知己知彼，百战不殆，小t首先要做的是知道这个病毒网站特征：包含多少不同的病毒，每种病毒出现了多少次。大家能再帮帮他吗？

输入描述:

第一行，一个整数N（1<=N<=1000），表示病毒特征码的个数。

接下来N行，每行表示一个病毒特征码，特征码字符串长度在1―50之间，并且只包含“英文大写字符”。任意两个病毒特征码，不会完全相同。

在这之后一行，表示“万恶之源”网站源码，源码字符串长度在2000000之内。字符串中字符都是ASCII码可见字符（不包括回车）。

输出描述:

按以下格式每行一个，输出每个病毒出现次数。未出现的病毒不需要输出。

病毒特征码: 出现次数

冒号后有一个空格，按病毒特征码的输入顺序进行输出。

代码：

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <queue>

using namespace std;

char ss[1005][55], T[2000005];

int cnt[1005];

struct Node

{

int flag;

Node \*ch[26], \*fail;

};

void getstr( char \*s )

{

int p = 0;

char c;

while( c = getchar(), c == '\n' ) ;

s[p++] = c;

while( c = getchar(), c != '\n' )

s[p++] = c;

s[p] = '\0';

}

Node \*init( )

{

Node \*p = new Node;

p->flag = -1;

p->fail = NULL;

memset( p->ch, 0, sizeof( p->ch ) );

return p;

}

void BuildTree( Node \*p, char \*in, int No )

{

int dx = \*in - 'A';

if( \*in )

{

if( !p->ch[dx] )

p->ch[dx] = init();

BuildTree( p->ch[dx], in + 1, No );

}

else

p->flag = No;

}

void BuildFailPoint( Node \*r )

{

queue<Node \*>q;

q.push( r );

while( !q.empty() )

{

Node \*pos = q.front();

q.pop();

for( int i = 0; i <26; ++i )

{

Node \*f = pos->fail;

if( pos->ch[i] )

{

while( f )

{

if( f->ch[i] )

{

pos->ch[i]->fail = f->ch[i];

break;

}

f = f->fail;

}

if( !f )

{

pos->ch[i]->fail = r;

}

q.push( pos->ch[i] );

}

}

}

}

void AcAuto( Node \*r, char \*text )

{

int len = strlen( text );

Node \*f = r;

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

{

int dx = text[i] - 'A';

if( dx < 0 || dx >= 26 )

{

f = r;

continue;

}

while( f )

{

if( f->ch[dx] )

{

f = f->ch[dx];

break;

}

f = f->fail;

}

if( !f )

f = r;

else

{

Node \*t = f;

while( t )

{

if( t->flag!= -1 )

{

cnt[t->flag]++;

}

t = t->fail;

}

}

}

}

void \_free( Node \*p )

{

for( int i = 0; i < 26; ++i )

{

if( p->ch[i] )

\_free( p->ch[i] );

}

free( p );

}

int main()

{

int N;

while( scanf( "%d", &N ) != EOF )

{

Node \*r = init();

memset( cnt, 0, sizeof( cnt ) );

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

{

getstr( ss[i] );

BuildTree( r, ss[i], i );

}

BuildFailPoint( r );

getstr( T );

AcAuto( r, T );

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

{

if( cnt[i] )

printf( "%s: %d\n", ss[i], cnt[i] );

}

\_free( r );

}

return 0;

}

## HDU 3138 Coconuts

A group of n castle guards are voting to determine whether African swallows can carry coconuts. While each guard has his own personal opinion on the matter, a guard will often vote contrary to his beliefs in order to avoid disagreeing with the votes of his friends.

You are given a list of guards who either do or do not believe in the coconut-carrying capacity of African swallows, and a list of all pairs of guards who are friends. Your task is to determine how each guard must vote in order to minimize the sum of the total number of disagreements between friends and the total number of guards who must vote against their own beliefs.

输入描述:

The input to this problem will contain multiple test cases. Each test case begins with a single line containing an integer n (where 2 <=

n <= 300), the number of guards, and an integer m (where 1 <=

m <=

n(

n-1)/2), the number of pairs of guards who are friends. The second line of the test case contains n integers, where the ith integer is 1 if the ith guard believes in the ability of African swallows to carry coconuts, and 0 otherwise. Finally, the next m lines of the test case each contain two distinct integers i and j (where 1 <=

i,

j <=

n), indicating that guards i and j are friends. Guards within each pair of friends may be listed in any order, but no pair of guards will be repeated. The input is terminated by an invalid test case with n =

m = 0, which should not be processed.

输出描述:

For each input test case, print a single line containing the minimum possible sum of the total number of disagreements between all friends plus the total number of guards who must vote against their own beliefs.

参考答案

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <string>

#include <iostream>

#include <algorithm>

#include <sstream>

#include <cmath>

using namespace std;

#include <queue>

#include <stack>

#include <vector>

#include <deque>

#include <map>

#define cler(arr, val) memset(arr, val, sizeof(arr))

typedef long long LL;

const int MAXN = 543;

const int MAXM = 260110;

const int INF = 0x3f3f3f3f;

const int mod = 1000000007;

struct Edge

{

int to,next,cap,flow;

}edge[MAXM];

int tol;

int head[MAXN];

int gap[MAXN],dep[MAXN],pre[MAXN],cur[MAXN];

void init()

{

tol = 0;

memset(head,-1,sizeof (head));

}

void addedge (int u,int v,int w,int rw=0)

{

edge[tol].to = v;edge[tol].cap = w;edge[tol].next = head[u];

edge[tol].flow = 0;head[u] = tol++;

edge[tol].to = u;edge[tol].cap = rw;edge[tol]. next = head[v];

edge[tol].flow = 0;head[v]=tol++;

}

int sap(int start,int end, int N)

{

memset(gap,0,sizeof(gap));

memset(dep,0,sizeof(dep));

memcpy(cur,head,sizeof(head));

int u = start;

pre[u] = -1;

gap[0] = N;

int ans = 0;

int i;

while(dep[start] < N)

{

if(u == end)

{

int Min = INF;

for( i = pre[u];i != -1; i = pre[edge[i^1]. to])

{

if(Min > edge[i].cap - edge[i]. flow)

Min = edge[i].cap - edge[i].flow;

}

for( i = pre[u];i != -1; i = pre[edge[i^1]. to])

{

edge[i].flow += Min;

edge[i^1].flow -= Min;

}

u = start;

ans += Min;

continue;

}

bool flag = false;

int v;

for( i = cur[u]; i != -1;i = edge[i].next)

{

v = edge[i]. to;

if(edge[i].cap - edge[i].flow && dep[v]+1 == dep[u])

{

flag = true;

cur[u] = pre[v] = i;

break;

}

}

if(flag)

{

u = v;

continue;

}

int Min = N;

for( i = head[u]; i != -1;i = edge[i]. next)

{

if(edge[i].cap - edge[i].flow && dep[edge[i].to] < Min)

{

Min = dep[edge[i].to];

cur[u] = i;

}

}

gap[dep[u]]--;

if(!gap[dep[u]]) return ans;

dep[u] = Min+1;

gap[dep[u]]++;

if(u != start) u = edge[pre[u]^1].to;

}

return ans;

}

int n,m,s,e;

void build()

{

init();

s=n+1,e=n+2;

int u,v,w;

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

{

scanf("%d",&w);

if(w==0)

addedge(s,i,1);

else

addedge(i,e,1);

}

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

{

scanf("%d%d",&u,&v);

addedge(u,v,1);

addedge(v,u,1);

}

}

int main()

{

while(scanf("%d%d",&n,&m),n+m)

{

build();

printf("%d\n",sap(s,e,n+2));

}

return 0;

}

## HDU 3191 How Many Paths Are There

oooccc1 is a Software Engineer who has to ride to the work place every Monday through Friday. For a long period, he went to office with the shortest path because he loves to sleep late…Time goes by, he find that he should have some changes as you could see, always riding with the same path is boring.

One day, oooccc1 got an idea! Why could I take another path? Tired at all the tasks he got, he got no time to carry it out. As a best friend of his, you’re going to help him!

Since oooccc1 is now getting up earlier, he is glad to take those paths, which are a little longer than the shortest one. To be precisely, you are going to find all the second shortest paths.

You would be given a directed graph G, together with the start point S which stands for oooccc’1 his house and target point E presents his office. And there is no cycle in the graph. Your task is to tell him how long are these paths and how many there are.

输入描述:

There are some cases. Proceed till the end of file.

The first line of each case is three integers N, M, S, E (3 <= N <= 50, 0 <= S , E <N)

N stands for the nodes in that graph, M stands for the number of edges, S stands for the start point, and E stands for the end point.

Then M lines follows to describe the edges: x y w. x stands for the start point, and y stands for another point, w stands for the length between x and y.

All the nodes are marked from 0 to N-1.

输出描述:

For each case,please output the length and count for those second shortest paths in one line. Separate them with a single space.

参考答案

#include<iostream>

#include<string.h>

#include<algorithm>

#define N 55

#define M 10000

#define inf 0x7fffffff

using namespace std;

int cnt[N][2],dis[N][2];

bool h[N][2];

int NE,head[N];

int n,m;

struct node{

int next,v,w;

node(){};

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

next=a;v=b;w=c;

}

}E[M];

void init(){

NE=0;

memset(head,-1,sizeof(head));

}

void insert(int u,int v,int w){

E[NE]=node(head[u],v,w);

head[u]=NE++;

}

void dijkstra(int beg,int end){

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

dis[i][0]=dis[i][1]=inf;

cnt[i][0]=cnt[i][1]=0;

}

memset(h,0,sizeof(h));

dis[beg][0]=0;

cnt[beg][0]=1;

while(true){

int u,flag;

int Min=inf;

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

if(!h[i][0]&&dis[i][0]<Min){

Min=dis[i][0];

u=i;flag=0;

}

else if(!h[i][1]&&dis[i][1]<Min){

Min=dis[i][1];

u=i;flag=1;

}

}

if(u==end&&flag==1) break;

if(Min==inf) break;

h[u][flag]=1;

for(int i=head[u];i!=-1;i=E[i].next){

int v=E[i].v;

int w=dis[u][flag]+E[i].w;

if(dis[v][0]>w){

if(dis[v][0]!=inf){

dis[v][1]=dis[v][0];

cnt[v][1]=cnt[v][0];

}

dis[v][0]=w;

cnt[v][0]=cnt[u][flag];

}

else if(dis[v][0]==w)

cnt[v][0]+=cnt[u][flag];

else if(dis[v][1]>w){

dis[v][1]=w;

cnt[v][1]=cnt[u][flag];

}

else if(dis[v][1]==w)

cnt[v][1]+=cnt[u][flag];

}

}

printf("%d %d\n",dis[end][1],cnt[end][1]);

}

int main(void){

int s,t;

while(~scanf("%d%d%d%d",&n,&m,&s,&t)){

init();

while(m--){

int x,y,w;

scanf("%d%d%d",&x,&y,&w);

insert(x,y,w);

}

dijkstra(s,t);

}

}

## HDU 3223 Decrypt Messages

In the game BioHazard 4, the president's daughter has been abducted by some crazy villagers. Leon S. Kennedy, the secret agent of White House, goes to rescue her. He keeps in contact with Hunnigan, the president's secretary.

But the time in their contact log has been encrypted, using the following method:

Count the number of seconds from 2000.01.01 00:00:00 to that time, assume this number is x. Then calculate x q , modulo it by a prime number p. The remainder a is the encrypted number.

Your task is to help Leon write a program to decrypt the contact log, and tell him all the possible original time.

1. Remember that if the year can be divided evenly by 4 but can't be divided evenly by 100, or it can be divided evenly by 400, this year is a leap year. The February of a leap year has 29 days, while the February of other years has 28 days.

2. In this problem, if the year modulo 10 is 5 or 8, at the end of this year, there is one “leap second”, i.e., the second after 2005.12.31 23:59:59 is 2005.12.31 23:59:60, and after that second, it's 2006.01.01 00:00:00.

You may assume that from 2000.01.01 00:00:00 till that time, less than p seconds have passed.

输入描述:

There are multiple test cases.

The first line of the input contains an integer T, meaning the number of the test cases.

For each test case, a single line of three integers: p, q, and a. (2<p≤1000000007, 1<q≤10, 0≤a<p, p is always a prime.)

输出描述:

The time. If there are multiple solutions, you must output all possible solutions in chronological order.

If the solution doesn't exist, output Transmission error instead.

See the sample output for further details.

参考答案

#include<cstdio>

#include<vector>

#include<cmath>

#include<map>

#include<iostream>

#include<algorithm>

using namespace std;

typedef long long ll;

ll pow(ll a,ll b,ll mod){

ll as=1;

while(b){

if(b&1)as=as\*a%mod;

b>>=1; a=a\*a%mod;

}

return as;

}

vector<ll>as,f;

ll gcd(ll a,ll b,ll& x,ll& y){

if(b==0){

x=1;y=0;

return a;

}

else{

ll r=gcd(b,a%b,y,x);

y-=a/b\*x;

return r;

}

}

void moxianxing(ll a,ll b,ll n){

ll x,y;

ll d=gcd(a,n,x,y);

as.clear();

if(b%d==0){

x%=n;x+=n;x%=n;

as.push\_back(x\*(b/d)%(n/d));

for(ll i=1;i<d;i++)

as.push\_back((as[0]+i\*n/d)%n);

}

}

bool gtest(ll g,ll p){

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

if(pow(g,(p-1)/f[i],p)==1)

return 0;

return 1;

}

ll gen(ll p){

ll tmp=p-1;

f.clear();

for(ll i=2;i<=tmp/i;i++)

if(tmp%i==0){

f.push\_back(i);

while(tmp%i==0)

tmp/=i;

}

if(tmp!=1)f.push\_back(tmp);

ll g=0;

while(++g)if(gtest(g,p))return g;

}

map<ll,int>rec;

ll lisanshu(ll x,ll n,ll m){

ll s=(ll)(sqrt((double)m)+0.5);

ll cur=1;rec.clear();

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

rec[cur]=i;

cur=cur\*x%m;

}

ll mul=cur;cur=1;

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

ll more=n\*pow(cur,m-2,m)%m;

if(rec.count(more))return i\*s+rec[more];

cur=cur\*mul%m;

}

return -1;

}

int dd[33],hh[30],mi[80];

int mm[20]={0,31,28,31,30,31,30,31,31,30,31,30,31};

void solve(ll x){

int y,mo,d,h,min,s,i;

for(y=2000;1;y++){

s=365\*24\*60\*60;

if(y%4==0&&y%100||y%400==0)s+=24\*60\*60;

if(y%10==5||y%10==8)s++;

if(x-s<0)break;

x-=s;

}

if(y%4==0&&y%100||y%400==0)

for(i=2;i<13;i++)

mm[i]+=24\*60\*60;

if(y%10==5||y%10==8){

mm[12]++;dd[31]++;hh[24]++;mi[60]++;

}

for(i=1;i<13;i++)if(x-mm[i]<0)break;x-=mm[i-1];mo=i;

for(i=1;i<33;i++)if(x-dd[i]<0)break;x-=dd[i-1];d=i;

for(i=1;i<25;i++)if(x-hh[i]<0)break;x-=hh[i-1];h=i-1;

for(i=1;i<61;i++)if(x-mi[i]<0)break;x-=mi[i-1];min=i-1;

if(y%4==0&&y%100||y%400==0)

for(i=2;i<13;i++)

mm[i]-=24\*60\*60;

if(y%10==5||y%10==8){

mm[12]--;dd[31]--;hh[24]--;mi[60]--;

}

printf("%d.%02d.%02d %02d:%02d:%02lld\n",y,mo,d,h,min,x);

}

int main(){

ll a,p,q,g,u,v;

int cs,csi;

for(int i=1;i<15;i++)mm[i]=mm[i]\*24\*60\*60+mm[i-1];

for(int i=1;i<33;i++)dd[i]=i\*24\*60\*60;

for(int i=1;i<30;i++)hh[i]=i\*60\*60;

for(int i=1;i<80;i++)mi[i]=i\*60;

scanf("%d",&cs);

for(csi=1;csi<=cs;csi++){

scanf("%lld%lld%lld",&p,&q,&a);

printf("Case #%d:\n",csi);

if(a==0){puts("2000.01.01 00:00:00");continue;}

as.clear();

g=gen(p);

v=lisanshu(g,a,p);

if(v==-1){puts("Transmission error");continue;}

moxianxing(q,v,p-1);

if((int)as.size()==0){puts("Transmission error");continue;}

for(int i=0;i<as.size();i++)as[i]=pow(g,as[i],p);

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

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

solve(as[i]);

}

return 0;

}

## HDU 3269 P2P File Sharing System

Peer-to-peer(P2P) computing technology has been widely used on the Internet to exchange data. A lot of P2P file sharing systems exist and gain much popularity in nowadays.

Let's consider a simplified model of P2P file sharing system: There are many computers in the system, and they can receive data from or send data to others. To simplify the problem, let's assume that there is just ONE large file which is of our concern in the system. Some computers already have the whole file (we call them "servers" and some don't(we call them "clients". Every client needs to download the file from servers. When a client gets the WHOLE file, it becomes a server.

These computers are not always online. An online client will down load the file from all online servers. Different servers send data of different parts of the file to a client, so the client can download the file faster.

Now given the transfer speed between each pair of computers, what time is every computer online or offline, and which computers are servers at the beginning, please analyze the running of the system in a period of time.

输入描述:

The first line contains an integer indicating the number of test cases.

For each test case:

Line 1: It contains two positive integers: n and T(n<=20, T<=1000) meaning that there are n computers in the system numbered from 1 to n, and you task is to figure out that how many percentage of the file does every computer gets after T seconds from the beginning.

Line 2: It contains two positive integers: k and S (k<=n, S<=220) meaning that at the beginning there are k servers, and the size of the file of our concern is S (KB).

Line 3: It contains k integers. It's a list of all servers'No.

Line 4 ~ n+3: These n lines form a matrix of size n\*n. The j-th integer in the i-th row of the matrix represents the transfer speed (in KB/s, no more than 210) between computer i and computer j (i and j start from 1). The matrix is symmetrical and the integers on the diagonal are meaningless.

Line n+4 ~ 2n+3: Each line contains an online/offline pattern for a computer in the following format( These lines are in the ascending order of computer No.) :

t online\_time 1 offline\_time 1 online\_time 1 offline\_time 2 ... online\_time t offline\_time t

t is an integer no more than 10 and the time given are all non-negative integers and in ascending order. During the time between online\_timei and offline\_timei, the computer is online and can download data from other computers or send data to other computers.

Line 2n+4: It contains one positive integer m, representing the number of download actions in the system.

Line 2n+5 ~ 2n+m+4: Each line contains two integers representing a download action in the following format:

download\_time i computer\_id i

At time download\_time i, the computer computer\_idi starts to download the file. 0<= download\_time i <=T, 1<= computer\_id i <=n. These lines are given in non-descending order of time. It's guaranteed that servers never try to download the file. It's ensured that at time download\_time i the computer computer\_idi is online (Though it's possible that it instantly go offline after issuing a download command).

When a client starts to download, it will try to connect to all servers and download data simultaneously from online servers. The client's download speed is the sum of all connections. We assume the construction of a connection to be instant and cost no time. Only data transfer is time consuming.

When a client goes offline, unfinished download task will be saved and continued when it's online next time. If a server goes online, all computers that are currently downloading will connect to it and download data from it. What's more, when a client becomes a server, it begins to send data to clients immediately. NOTE: To simplify the problem, time used to download a file should be rounded up to integer(If the file size is 6KB and download speed is 5KB/s, the download task will cost 2 seconds instead of 1.2 seconds ----- 5KB for the first second and 1KB for the next second).

Please note that all times given above are in seconds.

输出描述:

For each test case, the output should contain n lines, each for a computer.

The i-th line contains a percentage indicating the amount of data the i-th computer gets after T seconds from the beginning, in the format: "percentage%". The percentage should be rounded down to integer.

参考答案

#include <stdio.h>

#include <iostream>

#include <string.h>

#include <algorithm>

using namespace std;

#define N 32

#define cls(a) memset(a, 0, sizeof(a))

#define cls1(a,i,j) memset(a+i, 1, (j-i+1)\*sizeof(bool))

#define cls0(a, i) memset(a, 0, i\*sizeof(bool))

struct computer{

bool able[1005];

int pcent;

}cpt[N];

bool isSver[N];

int Sver[N];

int Clnt[N];

int rate[N][N];

int main()

{

int d, n, t, k, s, i, j, l, st, ed, m, c;

scanf("%d", &d);

while(d--)

{

cls(isSver);

cls(Clnt);

cls(Sver);

cls(cpt);

cls(rate);

scanf("%d%d", &n, &t);

scanf("%d%d", &s, &k);

for(i=0; i<s; i++)

{

scanf("%d", Sver+i);

isSver[Sver[i]] = 1;

cpt[Sver[i]].pcent = k;

}

for(c=0, i=1; i<=n; i++)

{

if(!isSver[i]) Clnt[c++] = i;

}

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

{

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

{

scanf("%d", rate[i]+j);

}

}

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

{

scanf("%d", &m);

for(j=0; j<m; j++)

{

scanf("%d%d", &st, &ed);

if(st > t) continue;

if(ed > t) ed = t;

for(l=st; l<ed; l++) cpt[i].able[l] = 1;

}

}

scanf("%d", &m);

for(i=0; i<m; i++)

{

scanf("%d%d", &ed, &st);

cls0(cpt[st].able, ed);

}

for(i=0; i<=t; i++)

{

for(j=0; j<c; j++)

{

if(cpt[Clnt[j]].able[i])

{

for(l=0; l<s; l++)

{

if(cpt[Sver[l]].able[i])

{

cpt[Clnt[j]].pcent += rate[Clnt[j]][Sver[l]];

}

}

}

}

for(j=0; j<c; j++)

if(cpt[Clnt[j]].pcent >= k)

{

isSver[Clnt[j]] = 1;

Sver[s++] = Clnt[j];

for(c--, l=j; l<c; l++)

Clnt[l] = Clnt[l+1];

}

}

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

{

printf("%d%%\n", cpt[i].pcent>=k ? 100 : (cpt[i].pcent\*100/k));

}

}

return 0;

}

## HDU 3314 Trouble with Election!

Taman and his friends are going to arrange an election for their club. The rule of this election is simple. Every member of the club can cast a single vote for one person. Even one can vote for himself too. If A casts his vote for B and B casts his vote for C then it means that C gets both the votes of A and B. It means if a man casts his vote for someone then the votes he got will also be added to the person he votes for. Now there is a problem if A’s vote goes for B and B’s vote for A. In this case, both of them have the same number of votes and it is a tie! So a tiebreaker will be needed. In this situation, all the votes of A and B and their supporters will be cancelled. If one’s vote goes for another then one is considered as the supporter of another.

Now you are elected as the election commissioner for the election of the club. It is your duty to publish the result of the election. If you find no possible winner or if two or more members have same number of votes then you should declare the situation as “Trouble”, print the name of the winner otherwise. Instead of name, a unique number for each member identifies the members of this club and the number should always be non-negative and less than the total number of members of the club.

输入描述:

There will be multiple test cases. Every test case starts with a single integer N on a line. 0<N<= 100000. N denotes the number of members of the club. N lines following. Each I’th line will be consisting of a single integer J; denoting I’th member casts his vote for J’th member. 0<=I, J<N.

输出描述:

The output for every case consists of a single integer, which denotes that the M’th member is the winner or a single word, “Trouble”, if a unique winner can not be determined. Here quotes are for clarity.

代码：

#include<iostream>

#include<string.h>

using namespace std;

const int maxn=100000+10;

int p[maxn],r[maxn],cnt[maxn];

void make()

{

memset(r,0,sizeof(r));

memset(p,255,sizeof(p));

for(int i=0;i<maxn;i++) cnt[i]=1;

}

int find(int x)

{

int px,t;

for(px=x;p[px]>=0;px=p[px]);

while (x!=px) {

t=p[x];

p[x]=px;

x=t;

}

return px;

}

int unio(int x,int y)

{

x=find(x),y=find(y);

if(x==y) return -1;

if(r[x]>r[y])

{

p[y]=x;

cnt[y]+=cnt[x];

cnt[x]=cnt[y];

return x;

}

else

{

p[x]=y;

cnt[x]+=cnt[y];

cnt[y]=cnt[x];

if(r[x]==r[y]) r[y]++;

return y;

}

}

int d[maxn];

int main()

{

int n;

while (scanf("%d",&n)!=EOF)

{

make();

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

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

unio(i,d[i]);

}

int tmp\_max=-1,k=-1;

bool flag=false;

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

{

if(i==d[i])

{

if(tmp\_max==cnt[find(i)])

{

flag=false;

}

if(cnt[find(i)]>tmp\_max)

{

flag=true;

tmp\_max=cnt[find(i)];

k=i;

}

}

}

if(!flag)

{

printf("Trouble\n");

}

else

{

printf("%d\n",k);

}

}

return 0;

}

## HDU 3369 Robot

An intelligent robot decided to learn a new language which has many words. In the i-th day the robot could memorize i^k (the k-th power of i) words. The exception is, on Saturday and Sunday, the robot will do some sports instead of learning the boring language. For example, if the first day is Wednesday and k = 2, the number of words the robot memorized each day is 1, 4, 9, 0, 0, 36, 49...

And n days passed. Do you know how many words the robot has memorized? The answer may be huge, just output the remainder after divided by 1000000007.

输入描述:

The first line contains an integer T (T<=20) indicating the number of test cases.

T\*2 lines follows. the first line of each test case contains a string(Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday) indicating the first day after the robot decided to learn a new language.

The second line contains two integers n (1<=n<=1000000000) and k (1<=k<=10).

输出描述:

For each test case, print the case number and the answer in a single line.

Please follow the format of the sample output.

参考答案

#include <utility>

#include <algorithm>

#include <string>

#include <cstring>

#include <cstdio>

#include <iostream>

#include <iomanip>

#include <set>

#include <vector>

#include <cmath>

#include <queue>

#include <bitset>

#include <map>

#include <iterator>

using namespace std;

#define clr(a,v) memset(a,v,sizeof(a))

#define lson l,m,rt<<1

#define rson m+1,r,rt<<1|1

const int INF = 0x7f7f7f7f;

const int maxn = 211111;

const double pi = acos(-1.0);

const double eps = 1e-10;

const int mod = 1000000007;

typedef long long LL;

typedef pair<int, int> pii;

typedef vector<LL> VI;

typedef vector<VI> VVI;

typedef vector<VVI> VVVI;

VVI E, I, A;

int size;

LL f[20][20];

LL P[20];

LL num[20][20];

LL arr[20];

int convert(char \*s) {

if (strcmp(s, "Monday") == 0)

return 1;

if (strcmp(s, "Tuesday") == 0)

return 2;

if (strcmp(s, "Wednesday") == 0)

return 3;

if (strcmp(s, "Thursday") == 0)

return 4;

if (strcmp(s, "Friday") == 0)

return 5;

if (strcmp(s, "Saturday") == 0)

return 6;

if (strcmp(s, "Sunday") == 0)

return 7;

return -1;

}

LL POW(LL a, LL b) {

LL res = 1;

while (b) {

if (b & 1)

res = res \* a % mod;

a = a \* a % mod;

b >>= 1;

}

return res;

}

VVI multi(const VVI& A, const VVI& B) {

VVI res = VVI(size, VI(size, 0));

int i, j, k;

for (i = 0; i < size; ++i) {

for (j = 0; j < size; ++j) {

for (k = 0; k < size; ++k) {

res[i][j] += A[i][k] \* B[k][j];

if (res[i][j] >= mod)

res[i][j] %= mod;

}

}

}

return res;

}

VVI cal(const VVI& A, LL k) {

VVI res = E, t = A;

while (k) {

if (k & 1)

res = multi(res, t);

t = multi(t, t);

k >>= 1;

}

return res;

}

void output(const VVI &arr) {

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

cout.width(5);

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

cout << arr[i][j] << " ";

}

cout << endl;

}

}

LL go(int n, int k, int s) {

size = k + 2;

int i, j, t;

int a = (7 - s % 7) % 7, b = (8 - s % 7) % 7;

LL ans = 0;

if (n <= 7) {

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

t = i % 7;

if (t == a || t == b)

continue;

ans += POW(i, k);

ans = ans >= mod ? ans % mod : ans;

}

return ans;

}

E = VVI(size, VI(size, 0));

for (i = 0; i < size; ++i)

E[i][i] = 1;

I = VVI(size, VI(size, 0));

I[0][0] = 1;

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

I[0][i] = f[k][i - 1] \* P[i - 1] % mod;

for (i = 1; i < size; ++i) {

for (j = i; j < size; ++j) {

I[i][j] = f[k + 1 - i][j - i] \* P[j - i] % mod;

}

}

LL p = n / 7, tmp;

VVI res = E, ta, tb;

int r = n % 7;

ta = cal(I, p - 1);

tb = multi(ta, I);

for (i = 1; i <= 7; ++i) {

if (i % 7 == a || i % 7 == b)

continue;

arr[0] = num[i][k];

tmp = 0;

for (j = 1; j < size; ++j)

arr[j] = num[i][k + 1 - j];

if (i <= r) {

for (j = 0; j < size; ++j) {

tmp += tb[0][j] \* arr[j];

tmp %= mod;

}

} else {

for (j = 0; j < size; ++j) {

tmp += ta[0][j] \* arr[j];

tmp %= mod;

}

}

ans += tmp;

ans %= mod;

}

return ans;

}

int main() {

ios::sync\_with\_stdio(false);

char s[14];

int n, k, i, j;

P[0] = 1;

for (i = 1; i < 20; ++i)

P[i] = P[i - 1] \* 7 % mod;

for (i = 0; i < 20; ++i)

f[i][0] = f[i][i] = 1;

for (i = 1; i < 20; ++i) {

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

f[i][j] = f[i - 1][j - 1] + f[i - 1][j];

f[i][j] %= mod;

}

}

for (i = 0; i < 20; ++i)

num[i][0] = 1;

for (i = 1; i < 20; ++i) {

for (j = 1; j < 20; ++j) {

num[i][j] = num[i][j - 1] \* i % mod;

}

}

int T, cas;

scanf("%d", &T);

for (cas = 1; cas <= T; ++cas) {

scanf("%s", s);

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

printf("Case %d: %I64d\n", cas, go(n, k, convert(s)));

}

return 0;

}

## HDU 3443 Shift Number

If a number is the sum of an integer’s several shifting forms, it is called the Shift Number.

For example, by shifting 123 four times and adding the four numbers together, we get 136653, which is a Shift Number.

However, a shift number may be the sum of more than one integer’s shifting forms. Such as, 45177 is a Shift Number, which could be generated from both 407 and 4107.

123+1230+12300+123000=136653

407+4070+40700=45177

4107+41070=45177

Given a Shift Number x , would you please help us find the least integer which could generate x by shifting.

输入描述:

A shift number x.

Input ends with x=0.

输出描述:

the least integer which could generate x by shifting.

代码：

#include <cstdio>

#define LL \_\_int64

int main()

{

LL x,k;

while(scanf("%I64d",&x)&&x){

k=1;

while(k<x)k=k\*10+1;

k/=10;

while(x%k!=0)k/=10;

printf("%I64d\n",x/k);

}

return 0;

}

## HDU 3458 Rectangles Too!

A rectangle in the Cartesian plane is speci ed by a pair of coordinates (x1 , y1) and (x2 , y2) indicating its lower-left and upper-right corners, respectively (where x1 ≤ x2 and y1 ≤ y2). Given a pair of rectangles,A = ((x A 1 , y A 1 ), (x A 2 ,y A 2 )) and B = ((x B 1 , y B 1 ), (x B 2 , y B 2 )), we write A ≤ B (i.e., A "precedes" B), if x A 2 < x B 1 and y A 2 < y B 1 :In this problem, you are given a collection of rectangles located in the two-dimension Euclidean plane. Find the length L of the longest sequence of rectangles (A 1 ,A 2 ,…,A L ) from this collection such that A 1 ≤ A 2 ≤ … ≤ A L .

输入描述:

The input fi le will contain multiple test cases. Each test case will begin with a line containing a single integer n (where 1 ≤ n ≤ 100000), indicating the number of input rectangles. The next n lines each contain four integers x i1 ,y i1 ,x i2 ,y i2 (where -1000000 ≤ x i1 ≤ x i2 ≤ 1000000, -1000000 ≤ y i1 ≤ y i2 ≤ 1000000, and 1 ≤ i ≤ n), indicating the lower left and upper right corners of a rectangle. The end-of-file is denoted by asingle line containing the integer 0.

输出描述:

For each input test case, print a single integer indicating the length of the longest chain.

参考答案

#include<iostream>

#include<algorithm>

using namespace std;

#define N 200020

struct Node {

int x,y,lt,id ;

}point[N];

int n,len,maxlp[N],by[N];

int cmp(Node a,Node b) {

if(a.x != b.x)

return a.x < b.x ;

if(a.y == b.y)

return a.lt > b.lt ;

return a.y < b.y ;

}

int find(int len,int \_y) {

int mid, left = 1, right = len ;

while(left <= right) {

mid = (left+right) >> 1 ;

if(by[mid] >= \_y)

right = mid-1 ;

else left = mid+1 ;

}

return left ;

}

int main() {

while(scanf("%d",&n) && n) {

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

scanf("%d%d",&point[2\*i].x,&point[2\*i].y);

point[2\*i].lt = 1 ;

scanf("%d%d",&point[2\*i+1].x,&point[2\*i+1].y);

point[2\*i+1].lt = 0 ;

point[2\*i].id = point[2\*i+1].id = i ;

}

n \*= 2 ;

sort(point,point+n,cmp) ;

len = 0 ;

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

if(point[i].lt == 1) maxlp[point[i].id] = find(len,point[i].y) ;

else {

int tmp = maxlp[point[i].id] ;

if(tmp > len) by[++len] = point[i].y ;

else by[tmp] = min(by[tmp],point[i].y) ;

}

}

printf("%d\n",len);

}

return 0;

}

## HDU 3468 Treasure Hunting

Do you like treasure hunting? Today, with one of his friend, iSea is on a venture trip again. As most movie said, they find so many gold hiding in their trip.

Now iSea’s clever friend has already got the map of the place they are going to hunt, simplify the map, there are three ground types:

● '.' means blank ground, they can get through it

● '#' means block, they can’t get through it

● '\*' means gold hiding under ground, also they can just get through it (but you won’t, right?)

What makes iSea very delighted is the friend with him is extraordinary justice, he would not take away things which doesn’t belong to him, so all the treasure belong to iSea oneself!

But his friend has a request, he will set up a number of rally points on the map, namely 'A', 'B' ... 'Z', 'a', 'b' ... 'z' (in that order, but may be less than 52), they start in 'A', each time friend reaches to the next rally point in the shortest way, they have to meet here (i.e. iSea reaches there earlier than or same as his friend), then start together, but you can choose different paths. Initially, iSea’s speed is the same with his friend, but to grab treasures, he save one time unit among each part of road, he use the only one unit to get a treasure, after being picked, the treasure’s point change into blank ground.

Under the premise of his friend’s rule, how much treasure iSea can get at most?

输入描述:

There are several test cases in the input.

Each test case begin with two integers R, C (2 ≤ R, C ≤ 100), indicating the row number and the column number.

Then R strings follow, each string has C characters (must be ‘A’ � ‘Z’ or ‘a’ � ‘z’ or ‘.’ or ‘#’ or ‘\*’), indicating the type in the coordinate.

The input terminates by end of file marker.

输出描述:

For each test case, output one integer, indicating maximum gold number iSea can get, if they can’t meet at one or more rally points, just output -1.

参考答案

#include <cstring>

#include <cstdio>

#include <queue>

#include <cstdlib>

#include <cctype>

#define MAXN 105\*105\*2

#define MAXM 105\*105\*100

#define inf 0x3f3f3f3f

using namespace std;

struct node

{

int u,v,f;

};

node e[MAXM];

int first[MAXN],ne[MAXM];

int gap[MAXN],d[MAXN],curedge[MAXN],pre[MAXN];

int cc, n,m;

inline void add\_edge(int u,int v,int f)

{

e[cc].u=u;

e[cc].v=v;

e[cc].f=f;

ne[cc]=first[u];

first[u]=cc;

cc++;

e[cc].u=v;

e[cc].v=u;

e[cc].f=0;

ne[cc]=first[v];

first[v]=cc;

cc++;

}

int ISAP(int s,int t,int n)

{

int cur\_flow,flow\_ans=0,u,tmp,neck,i,v;

memset(d,0,sizeof(d));

memset(gap,0,sizeof(gap));

memset(pre,-1,sizeof(pre));

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

curedge[i]=first[i];

gap[0]=n+1;

u=s;

while(d[s]<=n)

{

if(u==t)

{

cur\_flow=inf;

for(i=s;i!=t;i=e[curedge[i]].v)

{

if(cur\_flow>e[curedge[i]].f)

{

neck=i;

cur\_flow=e[curedge[i]].f;

}

}

for(i=s;i!=t;i=e[curedge[i]].v)

{

tmp=curedge[i];

e[tmp].f-=cur\_flow;

e[tmp^1].f+=cur\_flow;

}

flow\_ans+=cur\_flow;

u=neck;

}

for(i=curedge[u];i!=-1;i=ne[i])

{

v=e[i].v;

if(e[i].f&&d[u]==d[v]+1)

break;

}

if(i!=-1)

{

curedge[u]=i;

pre[v]=u;

u=v;

}

else

{

if(0==--gap[d[u]])

break;

curedge[u]=first[u];

for(tmp=n+5,i=first[u];i!=-1;i=ne[i])

if(e[i].f)

tmp=min(tmp,d[e[i].v]);

d[u]=tmp+1;

++gap[d[u]];

if(u!=s)

u=pre[u];

}

}

return flow\_ans;

}

char map[105][105];

int rally[105\*105];

int gold[105\*105];

int pos[105\*105];

int dist[55][105\*105];

int vis[105][105];

int xx[4]={0,1,0,-1};

int yy[4]={1,0,-1,0};

int trans(char word)

{

if(word>='A'&&word<='Z')

return word-'A';

else

return word-'a'+26;

}

void bfs(int s)

{

queue<int> q;

memset(vis,0,sizeof(vis));

memset(dist[s],inf,sizeof(dist[s]));

int t=pos[s];

vis[t/m][t%m]=1;

q.push(t);

dist[s][t]=0;

while(!q.empty())

{

int u=q.front();

q.pop();

int x=u/m;

int y=u%m;

int i;

for(i=0;i<4;i++)

{

int nx=x+xx[i];

int ny=y+yy[i];

if(nx<0||nx>=n||ny<0||ny>=m)

continue;

if(vis[nx][ny]||map[nx][ny]=='#')

continue;

vis[nx][ny]=1;

dist[s][nx\*m+ny]=dist[s][x\*m+y]+1;

q.push(nx\*m+ny);

}

}

}

int main()

{

while(scanf("%d%d",&n,&m)!=EOF)

{

memset(first,-1,sizeof(first));

memset(ne,-1,sizeof(ne));

cc=0;

memset(pos,0,sizeof(pos));

int i,j;

int cnt1=0;

int cnt2=0;

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

{

scanf("%s",map[i]);

for(j=0;j<m;j++)

{

if(isalpha(map[i][j]))

{

pos[trans(map[i][j])]=i\*m+j;

cnt1++;

}

else if(map[i][j]=='\*')

{

gold[cnt2]=i\*m+j;

cnt2++;

}

}

}

for(i=0;i<cnt1;i++)

bfs(i);

int flag=1;

for(i=1;i<=cnt1-1;i++)

{

for(j=0;j<cnt2;j++)

{

if(dist[i][gold[j]]+dist[i-1][gold[j]]==dist[i-1][pos[i]])

{

add\_edge(i,cnt1+j,1);

}

if(dist[i-1][pos[i]]==inf)

{

flag=0;

}

}

}

if(!flag)

{

printf("-1\n");

continue;

}

int s=0,t=cnt1+cnt2;

for(i=1;i<=cnt1-1;i++)

add\_edge(s,i,1);

for(i=0;i<cnt2;i++)

add\_edge(cnt1+i,t,1);

int res=ISAP(s,t,t);

printf("%d\n",res);

}

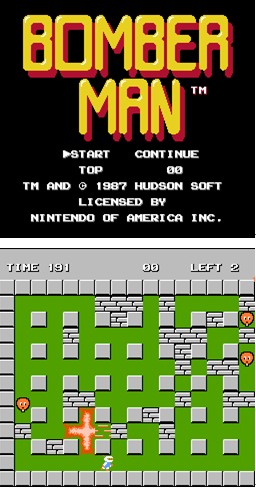
return 0;

}

## HDU 3529 Bomberman - Just Search!

Bomberman has been a very popular game ever since it was released. As you can see above, the game is played in an N\*M rectangular room. Bomberman can go around the room and place bombs. Bombs explode in 4 directions with radius r. To finish a stage, bomberman has to defeat all the foes with his bombs and find an exit behind one of the walls.

Since time is limited, bomberman has to do this job quite efficiently. Now he has successfully defeated all the foes, and is searching for the exit. It's really troublesome to destroy the walls one by one, so he's asking for your help to calculate the minimal number of bombs he has to place in order to destroy all the walls, thus he can surely find the exit.



输入描述:

The input contains several cases. Each case begins with two integers: N and M(4 <= N, M <= 15). N lines follow, each contains M characters, describing the room. A '\*' means a concrete wall which can never be destroyed, a '#' is an ordinary wall that can be destroyed by a single bomb, a '.' is an empty space where bombs can only be placed. There're at most 30 ordinary walls. The borders of the room is always surrounded by concrete walls, as you can see from the samples. You may assume that the explosion radius r is infinite, so that an explosion can reach as far as possible until it meets a wall and destroys it if it's an ordinary one. Proceed until the end of file.

输出描述:

For each case, output the minimal number of bombs that should be placed to destroy all the ordinary walls. Note that two bombs can't be placed at the same position and all bombs explode simultaneously, which makes the result for the second sample to be 3 instead of 2. You may assume that there's always a solution.

参考答案

# include<stdio.h>

# include<string.h>

# define N 200

# define V 900

int n,m,size,ak;

int U[V],D[V];

int L[V],R[V];

int C[V];

int H[N],S[N];

int visit[17][17];

int rr,cc;

char map[17][17];

void Link(int r,int c)

{

S[c]++;C[size]=c;

U[size]=U[c];D[U[c]]=size;

D[size]=c;U[c]=size;

if(H[r]==-1) H[r]=L[size]=R[size]=size;

else

{

L[size]=L[H[r]];R[L[H[r]]]=size;

R[size]=H[r];L[H[r]]=size;

}

size++;

}

void remove(int c)

{

int i;

for(i=D[c];i!=c;i=D[i])

L[R[i]]=L[i],R[L[i]]=R[i];

}

void resume(int c)

{

int i;

for(i=U[c];i!=c;i=U[i])

L[R[i]]=R[L[i]]=i;

}

int h()

{

int i,j,k,count=0;

bool hash[N];

memset(hash,0,sizeof(hash));

for(i=R[0];i;i=R[i])

{

if(hash[i]) continue;

hash[i]=1;

count++;

for(j=D[i];j!=i;j=D[j])

for(k=R[j];k!=j;k=R[k])

hash[C[k]]=1;

}

return count;

}

void Dance(int k)

{

int i,j,Min,c;

if(h()+k>=ak) return;

if(!R[0])

{

if(k<ak) ak=k;

return;

}

for(Min=N,i=R[0];i;i=R[i])

if(Min>S[i]) Min=S[i],c=i;

for(i=D[c];i!=c;i=D[i])

{

remove(i);

for(j=R[i];j!=i;j=R[j])

remove(j);

Dance(k+1);

for(j=L[i];j!=i;j=L[j])

resume(j);

resume(i);

}

}

int main()

{

int i,j,k1,k2;

while(scanf("%d%d",&n,&m)!=EOF)

{

cc=0;

memset(visit,0,sizeof(visit));

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

{

scanf("%s",map[i]);

for(j=0;map[i][j];j++)

if(map[i][j]=='#') visit[i][j]=++cc;

}

for(i=0;i<=cc;i++)

{

S[i]=0;

D[i]=U[i]=i;

L[i+1]=i;R[i]=i+1;

}R[cc]=0;

size=cc+1;

memset(H,-1,sizeof(H));

rr=0;

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

{

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

if(map[i][j]=='.')

{

rr++;

k1=i-1;k2=j;

while(map[k1][k2]=='.') k1--;

if(map[k1][k2]=='#') Link(rr,visit[k1][k2]);

k1=i+1;k2=j;

while(map[k1][k2]=='.') k1++;

if(map[k1][k2]=='#') Link(rr,visit[k1][k2]);

k1=i;k2=j-1;

while(map[k1][k2]=='.') k2--;

if(map[k1][k2]=='#') Link(rr,visit[k1][k2]);

k1=i;k2=j+1;

while(map[k1][k2]=='.') k2++;

if(map[k1][k2]=='#') Link(rr,visit[k1][k2]);

}

}

ak=N;

Dance(0);

printf("%d\n",ak);

}

return 0;

}

## poj 3635 Full Tank?

After going through the receipts from your car trip through Europe this summer, you realised that the gas prices varied between the cities you visited. Maybe you could have saved some money if you were a bit more clever about where you filled your fuel?

To help other tourists (and save money yourself next time), you want to write a program for finding the cheapest way to travel between cities, filling your tank on the way. We assume that all cars use one unit of fuel per unit of distance, and start with an empty gas tank.

输入描述

The first line of input gives 1 ≤ n ≤ 1000 and 0 ≤ m ≤ 10000, the number of cities and roads. Then follows a line with n integers 1 ≤ pi ≤ 100, where pi is the fuel price in the ith city. Then follow m lines with three integers 0 ≤ u, v < n and 1 ≤ d ≤ 100, telling that there is a road between u and v with length d. Then comes a line with the number 1 ≤ q ≤ 100, giving the number of queries, and q lines with three integers 1 ≤ c ≤ 100, s and e, where c is the fuel capacity of the vehicle, s is the starting city, and e is the goal.

输出描述

For each query, output the price of the cheapest trip from s to e using a car with the given capacity, or "impossible" if there is no way of getting from s to e with the given car.

输入例子

5 5

10 10 20 12 13

0 1 9

0 2 8

1 2 1

1 3 11

2 3 7

2

10 0 3

20 1 4

输出例子

170

impossible

参考答案

#include <iostream>

#include <cstring>

#include <cstdlib>

#include <cstdio>

#include <queue>

#define MAXN 1005

#define MAXM 100005

#define INF 1000000000

using namespace std;

struct Edge

{

int v, w, next;

}edge[MAXM];

struct Node

{

int v, cost, f;

bool operator <(const Node &a) const{

return a.cost < cost;

}

};

int head[MAXN], e, n, m, cap;

int dp[MAXN][105], used[MAXN][105], p[MAXN];

int s, t, ask;

priority\_queue<Node>q;

void init()

{

memset(head, -1, sizeof(head));

e = 0;

}

void ready()

{

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

for(int j = 0; j <= 100; j++)

dp[i][j] = INF;

dp[s][0] = 0;

memset(used, 0, sizeof(used));

while(!q.empty()) q.pop();

}

void insert(int x, int y, int w)

{

edge[e].v = y;

edge[e].w = w;

edge[e].next = head[x];

head[x] = e++;

}

int bfs()

{

Node a, b;

a.v = s, a.cost = 0, a.f = 0;

q.push(a);

while(!q.empty())

{

a = q.top();

q.pop();

int u = a.v;

int cost = a.cost;

int f = a.f;

used[u][f] = 1;

if(u == t) return cost;

if(f + 1 <= cap && !used[u][f + 1] && dp[u][f + 1] > dp[u][f] + p[u])

{

dp[u][f + 1] = dp[u][f] + p[u];

b.v = u;

b.f = f + 1;

b.cost = dp[u][f + 1];

q.push(b);

}

for(int i = head[u]; i != -1; i = edge[i].next)

{

int v = edge[i].v;

int w = edge[i].w;

if(f >= w && !used[v][f - w] && dp[v][f - w] > cost)

{

dp[v][f - w] = cost;

b.v = v;

b.f = f - w;

b.cost = dp[v][f - w];

q.push(b);

}

}

}

return -1;

}

int main()

{

int x, y, w;

scanf("%d%d", &n, &m);

init();

for(int i = 0; i < n; i++) scanf("%d", &p[i]);

while(m--)

{

scanf("%d%d%d", &x, &y, &w);

insert(x, y, w);

insert(y, x, w);

}

scanf("%d", &ask);

while(ask--)

{

scanf("%d%d%d", &cap, &s, &t);

ready();

int ans = bfs();

if(ans != -1) printf("%d\n", ans);

else printf("impossible\n");

}

return 0;

}

## HDU 3639 Hawk-and-Chicken

Kids in kindergarten enjoy playing a game called Hawk-and-Chicken. But there always exists a big problem: every kid in this game want to play the role of Hawk.

So the teacher came up with an idea: Vote. Every child have some nice handkerchiefs, and if he/she think someone is suitable for the role of Hawk, he/she gives a handkerchief to this kid, which means this kid who is given the handkerchief win the support. Note the support can be transmitted. Kids who get the most supports win in the vote and able to play the role of Hawk.(A note:if A can win

support from B(A != B) A can win only one support from B in any case the number of the supports transmitted from B to A are many. And A can't win the support from himself in any case.

If two or more kids own the same number of support from others, we treat all of them as winner.

Here's a sample: 3 kids A, B and C, A gives a handkerchief to B, B gives a handkerchief to C, so C wins 2 supports and he is choosen to be the Hawk.

输入描述:

There are several test cases. First is a integer T(T <= 50), means the number of test cases.

Each test case start with two integer n, m in a line (2 <= n <= 5000, 0 <m <= 30000). n means there are n children(numbered from 0 to n - 1). Each of the following m lines contains two integers A and B(A != B) denoting that the child numbered A give a handkerchief to B.

输出描述:

For each test case, the output should first contain one line with "Case x:", here x means the case number start from 1. Followed by one number which is the total supports the winner(s) get.

Then follow a line contain all the Hawks' number. The numbers must be listed in increasing order and separated by single spaces.

参考答案

#include <cstdio>

#include <algorithm>

#include <cstring>

#include <iostream>

#include <stack>

#include <vector>

using namespace std;

const int N=5006;

stack <int> s;

vector <int> adj[N], arc[N], com[N], ans;

int n, m, t, ind, T, sum;

int dfn[N], low[N], id[N], in[N];

bool vs[N];

void tarjan(int u)

{

dfn[u]=low[u]=T++;

s.push(u), vs[u]=1;

int len=(int)adj[u].size();

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

{

int v=adj[u][i];

if(dfn[v]==-1)

{

tarjan(v);

if(low[u]>low[v]) low[u]=low[v];

}

else if(vs[v] && low[u]>dfn[v]) low[u]=dfn[v];

}

if(low[u]==dfn[u])

{

for(int v; 1; )

{

v=s.top();

s.pop(), vs[v]=0;

id[v]=ind, com[ind].push\_back(v);

if(v==u) break;

}

ind++;

}

}

void DFS(int u)

{

vs[u]=1;

int len=(int)arc[u].size();

sum+=(int)com[u].size();

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

{

int v=arc[u][i];

if(vs[v]) continue;

DFS(v);

}

}

int main()

{

scanf("%d", &t);

for(int ca=1; ca<=t; ca++)

{

scanf("%d%d", &n, &m);

for(int i=0; i<n; i++) adj[i].clear();

for(int i=0, a, b; i<m; i++)

{

scanf("%d%d", &a, &b);

adj[a].push\_back(b);

}

for(int i=0; i<n; i++) vs[i]=0, dfn[i]=-1, com[i].clear();

ind=T=0;

while(!s.empty()) s.pop();

for(int i=0; i<n; i++) if(dfn[i]==-1) tarjan(i);

for(int i=0; i<ind; i++) arc[i].clear(), in[i]=0, dfn[i]=0;

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

{

int len=(int)adj[i].size();

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

{

int v=id[adj[i][j]];

int u=id[i];

if(u!=v) arc[v].push\_back(u), in[u]++;

}

}

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

{

if(in[i]!=0) continue;

for(int j=0; j<ind; j++) vs[j]=0;

sum=0;

DFS(i);

dfn[i]=sum;

}

int Max=-0x3fffffff;

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

if(Max<dfn[i]) Max=dfn[i];

ans.clear();

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

{

if(dfn[i]!=Max) continue;

int len=(int)com[i].size();

for(int j=0; j<len; j++) ans.push\_back(com[i][j]);

}

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

int len=(int)ans.size();

printf("Case %d: %d\n", ca, Max-1);

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

{

printf("%d", ans[i]);

if(i==len-1) printf("\n");

else printf(" ");

}

}

return 0;

}

## HDU 3734 How Many W

Given a sequence (S\_1, S\_2, ..., S\_N) (1 <= S\_i <= 1,000,000,000, S\_i != S\_j), count the number of W, which can be regarded as a quintuple (a, b, c, d, e) satisfying a < b < c < d < e, S\_b < S\_c < S\_a, S\_d < S\_c < S\_e.

输入描述:

The beginning of the input is an integer T, which is the number of test cases. T cases are followed. The first line of each test case is an integer N (1 <= N <= 100,000), which denotes the length of the sequence. The second line is N integer S\_1, S\_2, ..., S\_N, which denote the sequence.

输出描述:

For each test case, print a line contains a single integer which is the number of W, modulo 1,000,000,007.

代码：

#include<cstdio>

#include<cmath>

#include<cstring>

#include<algorithm>

#include<iostream>

using namespace std;

const int Mn = 100010;

const int Mod = 1e9 + 7;

int a[Mn], b[Mn], n;

int s[4][Mn];

pair<int, int >c[Mn];

inline int query(int k, int num){

int res(0);

for(;k;k -= k &-k)

res = (res+s[num][k]) %Mod;

return res;

}

inline void update(int k, int val, int num){

for(; k <= n; k += k & -k){

s[num][k] = (s[num][k] + val) % Mod;

}

}

int main()

{

int T;

scanf("%d", &T);

while(T--){

memset(s, 0, sizeof(s));

scanf("%d", &n);

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

scanf("%d", &a[i]),b[i] = a[i];

sort(b+1, b+n+1);

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

a[i] = lower\_bound(b+1, b+n+1,a[i]) - b;

c[i] = make\_pair(a[i],i);

}

sort(c+1, c+n+1);

long long ans(0);

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

long long t0 = query(c[i].second, 0);

long long t1 = query(c[i].second, 1);

long long t2 = query(n-c[i].second+1,2);

long long t3 = query(n-c[i].second+1,3);

ans = (ans+(t0-(1+t1) \* t1 /2) % Mod \*(t2-(1+t3)\*t3/2)%Mod) %Mod;

update(c[i].second, c[i].second, 0);

update(c[i].second, 1, 1);

update(n-c[i].second+1, n-c[i].second+1,2);

update(n-c[i].second+1,1,3);

}

printf("%d\n", (int)(ans%Mod+Mod) %Mod);

}

return 0;

}

## HDU 3892 Common Roots

We have many polynomials modulo p (p is a prime number). An interesting issue would be to determine whether they have some roots in common. Notice roots we mention here are integers in modulo p system (0 <= root < p). Moreover, if the given polynomial is of order r, we will guarantee that it has r roots.

For example, we have

x^2 + 13x + 36 (mod 37)

x^3 + 14x^2 + 49x + 36 (mod 37)

If x = 33 or x = 28, both of them would give the value of 0. So 33 and 28 are the roots in common.

输入描述:

There are many test cases (less than1000).

In each case, the integer in the first line is n (the number of polynomials in this case). Then n lines followed. Each of them starts with an integer r (order of polynomials, r <= 50), and r + 1 integers (a(r), a(r-1) ,..., a(0)), which means the polynomial goes like:

a(r) \* x^r + a(r-1) \* x^(r-1) + … +a(1) \* x + a(0) (mod 999983).

To make it easier, p is set to be 999983, as you see.

输出描述:

For each case, just output “YES” if they have common roots, otherwise “NO” in a single line.

参考答案

#include <iostream>

#include <string.h>

#include <algorithm>

#include <stdio.h>

#include <vector>

using namespace std;

typedef long long LL;

const LL MOD = 999983;

vector<LL> p[505];

int T;

LL quick\_mod(LL a,LL b,LL m)

{

LL ans = 1;

a %= m;

while(b)

{

if(b&1)

{

ans = ans\*a%m;

b--;

}

b>>=1;

a=a\*a%m;

}

return ans;

}

vector<LL> poly\_gcd(vector<LL> a,vector<LL> b)

{

if(b.size() == 0) return a;

int t = a.size() - b.size();

vector<LL> c;

for(LL i=0; i<=t; i++)

{

LL tmp = a[i] \* quick\_mod(b[0],MOD-2,MOD) % MOD;

for(LL j=0; j<b.size(); j++)

a[i+j] = (a[i+j] - tmp \* b[j] % MOD + MOD) % MOD;

}

LL p = -1;

for(LL i=0; i<a.size(); i++)

{

if(a[i] != 0)

{

p=i;

break;

}

}

if(p >= 0)

for(LL i=p; i<a.size(); i++)

c.push\_back(a[i]);

return poly\_gcd(b,c);

}

bool Import()

{

LL n,t;

if(scanf("%d",&T) == 1)

{

for(LL i=0;i<T;i++)

{

p[i].clear();

scanf("%I64d",&n);

for(LL j=0;j<=n;j++)

{

scanf("%I64d",&t);

p[i].push\_back(t);

}

}

return true;

}

return false;

}

void Work()

{

if(T==1)

{

if(p[0].size() > 1) puts("YES");

else puts("NO");

return;

}

vector<LL> v = poly\_gcd(p[0],p[1]);

LL i = 2;

while(i < T && v.size() > 1)

{

v = poly\_gcd(v,p[i]);

i++;

}

if(v.size() > 1) puts("YES");

else puts("NO");

}

int main()

{

while(Import())

Work();

return 0;

}

## HDU 3959 Board Game Dice

hh is a Board Game hobbyist, he often plays Board Game such as **Catan** , **Carcassonne** , **The Werewolves** , **A song of ice and fire**with friends.   
To play the games, we need some dices, and these dices are very unusual. Maybe with eight or twelve sides.   
<center> </center>  
hh plays with N friends today(including himself). They'll choose one person to be the judge. But the problem is: there is only a M-sided dice. How to pick a judge with the dice, so that everyone has equal probability of being chosen (the probability should always be 1/N)?  
hh has an idea here:   
1)Get x   
Decide rolling the dice x times. x is the smallest integer to make M x larger than or equal to N.   
  
2)Players choose sequences   
Each player chooses a sequence with x elements (1~M).   
For example, a 6-sides dice and x equal to 3, hh will gets sequence [5 4 6]. Players' sequences should be different from each other.(such as [6 4 5] is different from [5 4 6])   
  
3)Pick the judge   
Roll the dice for x times, we can get a result sequence, if someone has the same sequence as the result, he will be the judge; otherwise, repeat 1)-3), until the judge is chosen.   
  
It's a bigger project, hh wants know the expected number of times we will need to throw dice to determine the judge.

输入描述:

The first line is a number T(1<=T<=100), which represents the number of cases. The next T blocks following are the cases.

Each case contains two integer N , M(2<=N<=10 9, 2<=M<=20)

输出描述:

For each case, output the number of case and expected number of rolling as an irreducible fraction in the following form: "a/b" (as shown in the sample output)

参考答案

#include<stdio.h>

long long gcd(long long a,long long b)

{

if(b==0) return a;

return gcd(b,a%b);

}

int main()

{

int s,m,n,i,j,count=1;

long long sum,t;

scanf("%d",&s);

while(s--)

{

sum=1;

scanf("%d %d",&n,&m);

for(i=1;i<=110;i++)

{

sum=sum\*m;

if(sum>=n) break;

}

sum=sum\*i;

t=gcd(sum,n);

sum=sum/t;n=n/t;

printf("Case %d: %lld/%d\n",count++,sum,n);

}

return 0;

}

## HDU 4027 Can you answer these queries?

A lot of battleships of evil are arranged in a line before the battle. Our commander decides to use our secret weapon to eliminate the battleships. Each of the battleships can be marked a value of endurance. For every attack of our secret weapon, it could decrease the endurance of a consecutive part of battleships by make their endurance to the square root of it original value of endurance. During the series of attack of our secret weapon, the commander wants to evaluate the effect of the weapon, so he asks you for help.

You are asked to answer the queries that the sum of the endurance of a consecutive part of the battleship line.

Notice that the square root operation should be rounded down to integer.

输入描述:

The input contains several test cases, terminated by EOF.

For each test case, the first line contains a single integer N, denoting there are N battleships of evil in a line. (1 <= N <= 100000)

The second line contains N integers Ei, indicating the endurance value of each battleship from the beginning of the line to the end. You can assume that the sum of all endurance value is less than 2 63.

The next line contains an integer M, denoting the number of actions and queries. (1 <= M <= 100000)

For the following M lines, each line contains three integers T, X and Y. The T=0 denoting the action of the secret weapon, which will decrease the endurance value of the battleships between the X-th and Y-th battleship, inclusive. The T=1 denoting the query of the commander which ask for the sum of the endurance value of the battleship between X-th and Y-th, inclusive.

输出描述:

For each test case, print the case number at the first line. Then print one line for each query. And remember follow a blank line after each test case.

代码：

#include <queue>

#include <stack>

#include <cmath>

#include <cstdio>

#include <stdlib.h>

#include <iostream>

#include <limits.h>

#include <string>

#include <cstring>

#include <algorithm>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <math.h>

#include <algorithm>

#define LL \_\_int64

using namespace std;

const int N=100010;

struct Node

{

int L,R;

LL len;

LL sum;

} t[N\*4];

LL data[N];

int n,q;

void up(int fa){

t[fa].sum=t[fa<<1].sum+t[fa<<1|1].sum;

}

void down(int l,int r,int fa)

{

if(t[fa].L==t[fa].R)

{

t[fa].sum=(LL)(sqrt(t[fa].sum));

return;

}

int ls=fa<<1;

int rs=fa<<1|1;

int mid=(t[fa].L+t[fa].R)/2;

down(l,mid,ls);

down(mid+1,r,rs);

up(fa);

}

void built(int l,int r,int fa)

{

t[fa].L=l;

t[fa].R=r;

t[fa].len=r-l+1;

t[fa].sum=0LL;

if(l==r)

{

t[fa].sum=data[l];

return;

}

int mid=(l+r)/2;

built(l,mid,fa<<1);

built(mid+1,r,fa<<1|1);

up(fa);

}

void update(int l,int r,int fa)

{

int ls=fa<<1;

int rs=fa<<1|1;

int mid=(t[fa].L+t[fa].R)/2;

if(t[fa].L==l&&t[fa].R==r)

{

if(t[fa].sum==t[fa].len)

return ;

down(l,r,fa);

return ;

}

if(r<=mid)

update(l,r,ls);

else if(l>mid)

update(l,r,rs);

else

{

update(l,mid,ls);

update(mid+1,r,rs);

}

up(fa);

}

LL query(int l,int r,int fa)

{

int ls=fa<<1;

int rs=fa<<1|1;

int mid=(t[fa].L+t[fa].R)/2;

if(t[fa].L>=l&&t[fa].R<=r)

return t[fa].sum;

if(r<=mid)

return query(l,r,ls);

else if(l>mid)

return query(l,r,rs);

else

return query(l,mid,ls)+query(mid+1,r,rs);

}

int main()

{

int t=1;

while(scanf("%d",&n)!=EOF)

{

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

scanf("%I64d",&data[i]);

built(1,n,1);

int a,b,c;

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

cin>>q;

while(q--)

{

scanf("%d%d%d",&a,&b,&c);

if(b>c)swap(b,c);

if(a)

cout<<query(b,c,1)<<endl;

else

update(b,c,1);

}

cout<<endl;

}

return 0;

}

## HDU 4276 The Ghost Blows Light

My name is Hu Bayi, robing an ancient tomb in Tibet. The tomb consists of N rooms (numbered from 1 to N) which are connected by some roads (pass each road should cost some time). There is exactly one route between any two rooms, and each room contains some treasures. Now I am located at the 1st room and the exit is located at the Nth room.

Suddenly, alert occurred! The tomb will topple down in T minutes, and I should reach exit room in T minutes. Human beings die in pursuit of wealth, and birds die in pursuit of food! Although it is life-threatening time, I also want to get treasure out as much as possible. Now I wonder the maximum number of treasures I can take out in T minutes.

输入描述:

There are multiple test cases.

The first line contains two integer N and T. (1 <= n <= 100, 0 <= T <= 500)

Each of the next N - 1 lines contains three integers a, b, and t indicating there is a road between a and b which costs t minutes. (1<=a<=n, 1<=b<=n, a!=b, 0 <= t <= 100)

The last line contains N integers, which Ai indicating the number of treasure in the ith room. (0 <= Ai <= 100)

输出描述:

For each test case, output an integer indicating the maximum number of treasures I can take out in T minutes; if I cannot get out of the tomb, please output "Human beings die in pursuit of wealth, and birds die in pursuit of food!".

参考答案

#include <iostream>

#include <cstdio>

#include <cstring>

#include <queue>

using namespace std;

const int N=105;

int head[N],tot;

int dp[N][N\*5];

struct Edge

{

int from,to,cost,pre;

Edge(){}

Edge(int a,int b,int c,int d){from=a;to=b;cost=c;pre=d;}

}edge[N\*3];

void addEdge(int u,int v,int cost)

{

edge[tot]=Edge(u,v,cost,head[u]);

head[u]=tot++;

}

int spfa(int st,int ed)

{

queue<int> que;

int vis[N],dis[N],pre[N];

memset(vis,0,sizeof(vis));

memset(dis,0x7f,sizeof(dis));

memset(pre,-1,sizeof(pre));

que.push(st);

vis[st]=1; dis[st]=0;

while(!que.empty())

{

int u=que.front();

que.pop(); vis[u]=0;

for(int i=head[u];i!=-1;i=edge[i].pre)

{

int v=edge[i].to,cost=edge[i].cost;

if(dis[v]>dis[u]+cost)

{

dis[v]=dis[u]+cost;

pre[v]=i;

if(!vis[v])

{

vis[v]=1;

que.push(v);

}

}

}

}

for(int i=ed;i!=st;i=edge[pre[i]].from)

{

edge[pre[i]].cost=0;

edge[pre[i]^1].cost=0;

}

return dis[ed];

}

void dfs(int u,int fa,int coin)

{

for(int e=head[u];e!=-1;e=edge[e].pre)

{

int v=edge[e].to,cost=edge[e].cost\*2;

if(v==fa||coin<cost) continue;

dfs(v,u,coin-cost);

for(int i=coin;i>=cost;i--)

{

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

{

if(dp[u][i-j-cost]!=-1&&dp[v][j]!=-1)

dp[u][i]=max(dp[u][i],dp[u][i-j-cost]+dp[v][j]);

}

}

}

}

int main()

{

int n,t;

while(scanf("%d%d",&n,&t)!=EOF)

{

tot=0;

memset(dp,-1,sizeof(dp));

memset(head,-1,sizeof(head));

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

{

int a,b,c;

scanf("%d%d%d",&a,&b,&c);

addEdge(a,b,c);

addEdge(b,a,c);

}

for(int i=1;i<=n;i++) scanf("%d",&dp[i][0]);

t-=spfa(1,n);

if(t<0) puts("Human beings die in pursuit of wealth, and birds die in pursuit of food!");

else

{

dfs(1,0,t);

int res=0;

for(int i=0;i<=t;i++) res=max(res,dp[1][i]);

printf("%d\n",res);

}

}

return 0;

}

## HDU 4423 Simple Function

Knowing that x can be any real number that x 2 + Dx + E ≠ 0. Now, given the following function:

http://acm.hdu.edu.cn/data/images/4423-1.png

What is the range of y?

输入描述:

The first line contains a single integer T (T ≤ 10000), indicating that there are T cases below.

Each case contains five integers in a single line which are values of A, B, C, D and E (-100 ≤ A, B, C, D, E ≤ 100).

输出描述:

For each case, output the range of y in the form of standard interval expression like in a single line.

The expression is made up by one interval or union of several disjoint intervals.

Each interval is one of the following four forms: "(a, b)", "(a, b]", "[a, b)", "[a, b]"(there is a single space between ',' and 'b'), where a, b are real numbers rounded to 4 decimal places, or "-INF" or "INF" if the value is negative infinity or positive infinity.

If the expression is made up by several disjoint intervals, put the letter 'U' between adjacent intervals. There should be a single space between 'U' and nearby intervals.

In order to make the expression unique, the expression should contain as minimum of intervals as possible and intervals should be listed in increasing order.

See sample output for more detail.

## HDU 4043 FXTZ II

Cirno is playing a fighting game called "FXTZ" with Sanae.

Sanae is a ChuShou(master) of the game while Cirno is a ShaBao(noob). Since Cirno is a ShaBao, she just presses a random key on the keyboard for every 0.5 second, expecting to make a BiShaJi.

The battle begins. Having tried exactly 9 times, she finally makes a BiShaJi! She find herself summoned N iceballs!!! Then Sanae's HP decreases to 0 immediately....It should have been like that. But Cirno is too simple, always navie. She doesn't know how to handle the iceballs, so she starts to press the keyboard at random, again.

Let's see how the iceball damages. Each iceball has a fixed energy: the first ball's energy is 2^0, the second ball's energy is 2^1,..., and the N-th ball's energy is 2^(N-1). The damage caused by an iceball is equal to its energy. Cirno will shoot N times. Since Cirno is pressing the keyboard at random, each time Cirno will choose exactly one iceball with equal possibility to shoot out. Once shot out, the iceball can't be chosen again. And even worse, the target may be either her opponent or herself, with equal possibility(50%). What a big ShaBao she is. =\_=

During shooting, once Cirno's HP is less than Sanae's, she will lose the game. Otherwise, she wins.

You may assume Sanae did nothing while Cirno's shooting(all damages are caused by Cirno's iceball), and their original HP are both 2^N (No one will die in the middle of the battle unless Cirno's HP is less than Sanae's).

Here comes the question: Can you calculate the possibility of Cirno's victory?

输入描述:

The first line an integer C (C<=30), the number of test cases.

For each case, the only line contains one integer N(0<N<=500), indicating the number of iceballs.

输出描述:

For each case output a fraction, the possibility of Cirno's victory. The fraction must be reduced.

代码：

#include<stdio.h>

#include<string.h>

#define N 2010

#define K 510

char mol[K][N],den[K][N];

char virt[N];

int gcd(int a,int b)

{

return b?gcd(b,a%b):a;

}

int divide(char \*str,int n,char \*s)

{

int i,k,len;

len=strlen(str);

s[len]=0;

for(k=0,i=len-1;i>=0;i--)

{

k=k\*10+str[i]-'0';

s[i]=k/n+'0';

k%=n;

}

for(i=len-1;i>0;i--)

{

if(s[i]!='0')

break;

s[i]=0;

}

return k;

}

void mult(char \*str,int n,char \*x)

{

int i,s[N];

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

s[i]=0;

for(i=0;str[i];i++)

s[i]=(str[i]-'0')\*n;

for(i=0;i<N-1;i++)

{

s[i+1]+=s[i]/10;

s[i]%=10;

}

for(i=N-1;i>0 && s[i]==0;i--);

x[i+1]=0;

for(;i>=0;i--)x[i]=s[i]+'0';

}

void init()

{

int i,j,k;

strcpy(mol[1],"1");

strcpy(den[1],"2");

for(i=2;i<K;i++)

{

j=2\*i;

k=gcd(j,divide(mol[i-1],j,virt));

divide(mol[i-1],k,mol[i]);

mult(den[i-1],j/k,den[i]);

j--;

k=gcd(j,divide(den[i],j,virt));

if(k>1)

divide(den[i],k,den[i]);

mult(mol[i],j/k,mol[i]);

strrev(mol[i-1]);

strrev(den[i-1]);

}

}

int main()

{

int n,T;

init();

scanf("%d",&T);

while(T--)

{

scanf("%d",&n);

printf("%s/%s\n",mol[n],den[n]);

}

return 0;

}

## HDU 4061 A Card Game

There are N cards on the table, whose front side is writen one integer number from 1 to M. We call one card "a type k card" if its number is k. The quantity of type i cards is a\_i.

Let's play a game with these cards. We divide these cards into M piles by random with the only constrains that the quantity of cards in i-th (indexed from 1) pile must exactly be a\_i. The possbility of each card appears in i-th pile is directly proportional to the size of this pile. That is to say, if the size of a pile is A, the possibility for each card appears in this pile is A/N assuming that N is the amount of all cards. We choose pile 1 to start the game. Assuming the we now play this game at pile k, we randomly choose a card from pile k with the same possibility for all cards in it, remember the number written on this card and throw it away. If the number on the chosen card is j, we continue this game at pile j on next round. The game terminates when we are going to get a card from an empty pile.

Now the question is, when the game ends, what is the possibility that all piles are empty?

输入描述:

There is only one input file. The first line is the number of test cases T. T cases follow, each of which contains two lines. The first line is an integer M (1 <= M <= 100), the number of type of cards (and the number of piles, they are exactly the same). The second line contains M positive integers not greater than 1000, the i-th number of which is a\_i.

输出描述:

For each test case, output the possibility you are required to calculate. Answers are rounded to 6 numbers after the decimal point.(as shown in the sample output)

代码：

#include<stdio.h>

#include<cstring>

#include<algorithm>

#define I(x) scanf("%d",&x)

using namespace std;

int main(){

int n,a,sum,t,b,ca=0;

I(t);

while(t--){

I(n);

sum=0;

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

I(a);

sum+=a;

if(i==0) b=a;

}

printf("Case %d: %.6lf\n",++ca,1.0\*b/sum);

}

return 0;

}

## HDU 4079 Happy Telephones

In the land of Eden, all phone conversations are happy ones. People complaining on the phone are immediately put in jail. To enforce this law, the police taps all phone conversations.   
The police wants to hire the approriate number of operators to listen to all conversations in a given period of time. Unfortunately, each of their operators can listen to one conversation only before needing a really long break to rest from the effort.   
As a contractor of the police department, you have been asked to provide a program capable of determining the required number of operators. If the program does not work correctly, you will be put in jail as well, along with all the unhappy complainers. Do you really want to end up there?   
<center> </center> 

输入描述:

Each test case starts with two integers denoting the number of phone calls N (1 <= N < 10 000) and the number of intervals M (1 <= M < 100). This is followed by N lines describing the telephone calls, each one consisting of four integers Source, Destination, Start and Duration. Source and Destination identify the pair of telephone numbers establishing the connection (0 <= Source, Destination <= 10 000 000). Start and Duration are the start time and duration of the call in seconds (1 <= Duration <= 10 000 and Start >= 0). You can safely assume that the sum of Start and Duration fits into a 32-bit signed integer.

Afterwards follow M lines containing the time intervals the police are interested in, each on described by two integers Start and Duration, in the same format and with the same meaning and constraints as those in the telephone calls. The last test case is represented by N = M = 0 and must not be processed.

输出描述:

For each of the M intervals of each test case, print the number of calls that are active during at least one second of the interval.

代码：

#include"stdio.h"

#include"string.h"

#define N 10002

struct node

{

int x,y;

}A[N];

int main()

{

int i,j,k;

int n,m;

int s,d,x,y;

while(scanf("%d%d",&n,&m),n||m)

{

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

{

scanf("%d%d",&s,&d);

scanf("%d%d",&x,&y);

A[i].x=x;

A[i].y=x+y;

}

for(i=0;i<m;i++)

{

scanf("%d%d",&x,&y);

y=x+y;

for(k=0,j=0;j<n;j++)

{

if(A[j].y>x&&A[j].x<y)

k++;

}

printf("%d\n",k);

}

}

return 0;

}

## HDU 4182 Help-or-else

A penal colony for finance professionals will soon be holding its annual community service activity with some rules that are considered suitable for a penal colony. Every inmate is assigned a set P of N people to help with their finances and a limit of K minutes. In addition to the circumstances of the jth person, 1 <= j <= N, a time penalty of ej for choosing not to give advice and the time duration of dj minutes allotted to provide the advice are also made clear to the inmate. An inmate starts his community service at time T equal to zero. If the inmate started working with the jth person at time T, then he must terminate his work no later than T+dj. Regardless of the validity of the advice and time of completion, a value of C j ( = T+ d j ) is deducted from the inmate's alloted minutes. Also the inmate is not permitted to work with another person until the time T+ d j . If S is the set of people helped by an inmate, then the total number of used minutes is calculated as

Your task is to write a program to calculate the maximum number of persons that can be helped by an inmate without exceeding his K minutes limit.

输入描述:

Input consists of sets for many inmates. The description for each inmate begins with two integers N and K, separated by a single space on a line by themselves, that represent the number of people and the maximum allowed minutes. 0 < N <= 200 and 0 < K <= 6000. Each of the following N lines contains two integers, separated by a single space, which represent the penalty and time duration one person to be assisted. All integers have values between 0 and 10000, inclusive. Input terminates with two zeros on a line by themselves.

输出描述:

For each inmate, the output consists of a single line that contains the maximum number of persons to be helped within the given time limit using the format shown. “Mission Impossible” is entered where not exceeding the given time limit is not possible.

代码：

#include<iostream>

#include<cstdio>

#include<cstring>

#include<algorithm>

#include<queue>

using namespace std ;

#define maxn 201

#define INF 10000000

struct node

{

int e , d ;

}qe[maxn] ;

int dp[maxn][maxn] , k ;

int cmp( node a , node b )

{

return a.d > b.d ;

}

bool check( int n , int m )

{

int i , j ;

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

for( j = 0 ; j <= m ;j++ )

dp[i][j] = INF ;

dp[0][0] = 0 ;

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

{

dp[i][0] = dp[i-1][0]+qe[i].e ;

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

{

dp[i][j] = dp[i-1][j]+qe[i].e ;

dp[i][j] = min( dp[i][j],dp[i-1][j-1]+j\*qe[i].d) ;

}

}

return dp[n][m] <= k ;

}

int main()

{

int i , j , n , m ,ans ,case1 = 0 , tot ;

while( scanf("%d%d",&n , &k ) != EOF)

{

if( n == k && k == 0 ) break ;

tot = 0 ;

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

scanf("%d%d" , &qe[i].e,&qe[i].d) ;

tot += qe[i].e ;

}

sort(qe+1,qe+1+n,cmp) ;

ans = -1 ;

if( tot <= k ) ans = 0 ;

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

{

if(check(n,i))

{

ans = i ;

break ;

}

}

if( ans == -1 )printf("%d: Mission Impossible\n",++case1) ;

else printf("%d: %d\n",++case1,ans) ;

}

}

## HDU 4423 Simple Function

Knowing that x can be any real number that x 2 + Dx + E ≠ 0. Now, given the following function:

http://acm.hdu.edu.cn/data/images/4423-1.png

What is the range of y?

输入描述:

The first line contains a single integer T (T ≤ 10000), indicating that there are T cases below.

Each case contains five integers in a single line which are values of A, B, C, D and E (-100 ≤ A, B, C, D, E ≤ 100).

输出描述:

For each case, output the range of y in the form of standard interval expression like in a single line.

The expression is made up by one interval or union of several disjoint intervals.

Each interval is one of the following four forms: "(a, b)", "(a, b]", "[a, b)", "[a, b]"(there is a single space between ',' and 'b'), where a, b are real numbers rounded to 4 decimal places, or "-INF" or "INF" if the value is negative infinity or positive infinity.

If the expression is made up by several disjoint intervals, put the letter 'U' between adjacent intervals. There should be a single space between 'U' and nearby intervals.

In order to make the expression unique, the expression should contain as minimum of intervals as possible and intervals should be listed in increasing order.

See sample output for more detail.

## HDU 4652 Dice

You have a dice with m faces, each face contains a distinct number. We assume when we tossing the dice, each face will occur randomly and uniformly. Now you have T query to answer, each query has one of the following form:

0 m n: ask for the expected number of tosses until the last n times results are all same.

1 m n: ask for the expected number of tosses until the last n consecutive results are pairwise different.

输入描述:

The first line contains a number T.(1≤T≤100) The next T line each line contains a query as we mentioned above. (1≤m,n≤10 6) For second kind query, we guarantee n≤m. And in order to avoid potential precision issue, we guarantee the result for our query will not exceeding 10 9 in this problem.

输出描述:

For each query, output the corresponding result. The answer will be considered correct if the absolute or relative error doesn't exceed 10-6.

代码：

#include <cstdio>

#include <iostream>

#include <cstring>

#include <algorithm>

#include <cmath>

using namespace std;

double solve0(int m,int n)

{

double ans=0;

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

ans+=pow(1.0\*m,i);

return ans;

}

double solve1(int m,int n)

{

double ans=0;

double tmp=1;

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

{

ans+=tmp;

tmp\*=(m+0.0)/(m-i);

}

return ans;

}

int main()

{

int t;

while(scanf("%d",&t)!=EOF)

{

while(t--)

{

int n,m,op;

scanf("%d%d%d",&op,&m,&n);

if(op==0)

printf("%.9lf\n",solve0(m,n));

else

printf("%.9lf\n",solve1(m,n));

}

}

return 0;

}

## HDU 4714 Tree2cycle

A tree with N nodes and N-1 edges is given. To connect or disconnect one edge, we need 1 unit of cost respectively. The nodes are labeled from 1 to N. Your job is to transform the tree to a cycle(without superfluous edges) using minimal cost.

A cycle of n nodes is defined as follows: (1)a graph with n nodes and n edges (2)the degree of every node is 2 (3) each node can reach every other node with these N edges.

输入描述:

The first line contains the number of test cases T( T<=10 ). Following lines are the scenarios of each test case.

In the first line of each test case, there is a single integer N( 3<=N<=1000000 ) - the number of nodes in the tree. The following N-1 lines describe the N-1 edges of the tree. Each line has a pair of integer U, V ( 1<=U,V<=N ), describing a bidirectional edge (U, V).

输出描述:

For each test case, please output one integer representing minimal cost to transform the tree to a cycle.

参考答案

#include<iostream>

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<string>

#include<queue>

#include<algorithm>

#include<map>

#include<iomanip>

#define INF 99999999

using namespace std;

#pragma comment(linker,"/STACK:102400000,102400000")

const int MAX=1000000+10;

int head[MAX],size,sum=0;

struct Edge{

int v,next;

Edge(){}

Edge(int V,int NEXT):v(V),next(NEXT){}

}edge[MAX\*2];

void Init(int num){

memset(head,-1,sizeof(int)\*(num+1));

size=sum=0;

}

void InsertEdge(int u,int v){

edge[size]=Edge(v,head[u]);

head[u]=size++;

}

int dfs(int u,int father){

int num=0,v;

for(int i=head[u];i != -1;i=edge[i].next){

int v=edge[i].v;

if(v == father)continue;

num+=dfs(v,u);

}

if(num>=2){

if(u == 1)sum+=num-2;

else sum+=num-1;

return 0;

}else return 1;

}

int main(){

int t,n,u,v;

scanf("%d",&t);

while(t--){

scanf("%d",&n);

Init(n);

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

scanf("%d%d",&u,&v);

InsertEdge(u,v);

InsertEdge(v,u);

}

dfs(1,0);

printf("%d\n",sum\*2+1);

}

return 0;

}

## HDU 4760 Good Firewall

Professor X is an expert in network security. These days, X is planning to build a powerful network firewall, which is called Good Firewall (a.k.a., GFW). Network flows enter in the GFW will be forwarded or dropped according to pre-established forwarding policies.

Basically, a forwarding policy P is a list of IP subnets, {ip\_subnet\_1, …, ip\_subnet\_n}. If P is enabled in GFW, a network flow F with source and destination IP address both located in P can be accepted and forwarded by GFW, otherwise F will be dropped by GFW.

You may know that, an IP address is a 32-bit identifier in the Internet, and can be written as four 0~255 decimals. For example, IP address 01111011.00101101.00000110.01001110 can be expressed as 123.45.6.78 . An IP subnet is a block of adjacent IP address with the same binary prefix, and can be written as the first IP address in its address block together with the length of common bit prefix. For example, IP subnet 01111011.00101101.00000100.00000000/22 (123.45.4.0/22) is an IP subnet containing 1024 IP addresses, starting from 123.45.4.0 to 123.45.7.255. If an IP address is in the range of an IP subnet, we say that the IP address is located in the IP subnet. And if an IP address is located in any IP subnet(s) in a policy P, we say that the IP address is located in the policy P.

How will you design the GFW, if you take charge of the plan?

输入描述:

The input file contains no more than 32768 lines. Each line is in one of the following three formats:

E id n ip\_subnet\_1 ip\_subnet\_2 … ip\_subnet\_n

D id

F ip\_src ip\_dst

The first line means that a network policy P id (1<=id<=1024) is enabled in GFW, and there are n (1<=n <=15) IP subnets in P id. The second line means that policy P id (which is already enabled at least once) is disabled in GFW. The last line means that a network flow with source and destination IP address is entered in GFW, and you need to figure out whether GFW is going to forward (F) or drop (D) this flow:

1. If the source and destination IP address both are located in one of enabled policy group P id, GFW will forward this flow.

2. Otherwise GFW will drop this flow. That is, if the source or destination IP address is not located in any of enabled policy group, or they are only located in different enabled policy group(s), GFW will drop it.

IP subnets can be overlapped. An IP address may or may not be located in any policy group, and can also be located in multiple policy groups.

In the global routing table, most of the IP subnets have at least 2^8 IP addresses, and at most 2^24 IP addresses. In our dataset, every IP subnet has a prefix length between 8 and 24.

输出描述:

For each ‘F’ operation, output a single ‘F’ (forward) or ‘D’ (drop) in a single line. Just see the sample output for more detail.

代码：

#include <iostream>

#include <cstdio>

#include <cstring>

#include <cmath>

#include <algorithm>

#include <vector>

#define pii pair<int,LL>

using namespace std;

typedef long long LL;

const int N = 100005;

struct Trie {

Trie \*next[2];

vector<pii> a;

}\*root , s[N << 2];

char ope[10];

int tot = 0 , can[N] , ok[1025] , idx = 0;

LL ip (int a , int b , int c , int d) {

LL ret = 0;

ret |= (LL)a << 24;

ret |= (LL)b << 16;

ret |= (LL)c << 8;

ret |= d;

return ret;

}

Trie \*newnode () {

Trie \*t = &s[tot ++];

t -> next[0] = t -> next[1] = NULL;

t -> a.clear();

return t;

}

void insert (Trie \*p , LL ip , int l , int id) {

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

int c = (ip >> (31 - i)) & 1;

if (p -> next[c] == NULL)

p -> next[c] = newnode ();

p = p -> next[c];

}

LL num = ip & ((1 << (32 - l)) - 1);

p -> a.push\_back (make\_pair (id , num));

}

void down (LL ip) {

Trie \*p = root;

for (int i = 31 ; i >= 0 ; i --) {

int c = (ip >> i) & 1;

if (p -> next[c] == NULL) return ;

p = p -> next[c];

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

if (can[p -> a[j].first] && p -> a[j].second <= (ip & ((1 << (i - 1)) - 1)))

ok[p -> a[j].first] = idx;

}

}

}

bool up (LL ip) {

Trie \*p = root;

for (int i = 31 ; i >= 0 ; i --) {

int c = (ip >> i) & 1;

if (p -> next[c] == NULL) return false;

p = p -> next[c];

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

if (can[p -> a[j].first] && p -> a[j].second <= (ip & ((1 << (i - 1)) - 1)) && ok[p -> a[j].first] == idx)

return true;

}

}

return false;

}

int main () {

root = newnode ();

while (scanf ("%s" , ope) != EOF) {

if (ope[0] == 'E') {

int id , k , a , b , c , d , u;

scanf ("%d %d" , &id , &k);

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

scanf ("%d.%d.%d.%d/%d" , &a , &b , &c , &d , &u);

insert (root , ip (a , b , c , d) , u , id);

}

can[id] = 1;

}

else if (ope[0] == 'D') {

int id;scanf ("%d" , &id);

can[id] = 0;

}

else {

idx ++;

int a , b , c , d;

scanf ("%d.%d.%d.%d" , &a , &b , &c , &d);

down (ip (a , b , c , d));

scanf ("%d.%d.%d.%d" , &a , &b , &c , &d);

puts (up (ip (a , b , c , d)) ? "F" : "D");

}

}

return 0;

}

## HDU 4780 Candy Factory

A new candy factory opens in pku-town. The factory import M machines to produce high quality candies. These machines are numbered from 1 to M.

There are N candies need to be produced. These candies are also numbered from 1 to N. For each candy i , it can be produced in any machine j. It also has a producing time(s i ,t i ) , meaning that candy i must start producing at time s i and will finish at t i . Otherwise if the start time is p i (s i < p i < t i ) then candy will still finish at t i but need additional K\*(p i - s i ) cost. The candy can’t be produced if p i is greater than or equal to t i . Of course one machine can only produce at most one candy at a time and can’t stop once start producing.

On the other hand, at time 0 all the machines are in their initial state and need to be “set up” or changed before starting producing. To set up Machine j from its initial state to the state which is suitable for producing candiy i, the time required is C ij and cost is D ij . To change a machine from the state suitable for candy i 1 into the state suitable for candy i 2 , time required is E i1i2 and cost is F i1i2 .

As the manager of the factory you have to make a plan to produce all the N candies. While the sum of producing cost should be minimized.

输入描述:

　　There are multiple test cases.

　　For each case, the first line contains three integers N(1<=N<=100), M(1<=M<=100), K(1<=K<=100) . The meaning is described above.

　　Then N lines follow, each line contains 2 integers s i and t i(0 <= s i < t i <100000).

　　Then N lines follow, each line contains M integers, the j-th integer of the i-th line indicating C ij(1<=C ij<=100000) .

　　Then N lines follow, each line contains M integers, the j-th integer of the i-th line indicating D ij(1<=D ij<=100000) .

　　Then N lines follow, each line contains N integers, the i 2-th integer of the i 1-th line indicating E i1i2(1<=E i1j2<=100000) .

　　Then N lines follow, each line contains N integers, the i 2-th integer of the i 1-th line indicating F i1i2(1 <= F i1j2<=100000) .

　　Since the same candy will only be produced once, E ii and F ii are meaningless and will always be -1.

　　The input ends by N=0 M=0 K=0. Cases are separated with a blank line.

输出描述:

For each test case, if all of M candies can be produced, output the sum of minimum producing cost in a single line. Otherwise output -1.

代码：

#include <cstdio>

#include <cstring>

#include <algorithm>

#include <queue>

using namespace std;

const int N = 510;

const int M = 1000010;

const int INF = 0x7f7f7f7f;

struct Edge {

int v, cap, cost, next;

Edge() {}

Edge(int a, int b, int c, int d)

:v(a), cap(b), cost(c), next(d) {}

}e[M];

int head[N], sz;

int dis[N], pre[N], cur[N];

bool inq[N];

queue<int> q;

void graphinit() {

memset(head, -1, sizeof(head));

sz = 0;

}

void addedge(int u, int v, int cp, int ct) {

e[sz] = Edge(v, cp, ct, head[u]);

head[u] = sz++;

e[sz] = Edge(u, 0, -ct, head[v]);

head[v] = sz++;

}

pair<int, int> mcmf(int s, int t) {

int mc = 0, mf = 0;

while (true) {

memset(pre, -1, sizeof(pre));

memset(inq, 0, sizeof(inq));

memset(dis, 0x7f, sizeof(dis));

dis[s] = 0;

q.push(s);

while (!q.empty()) {

int u = q.front(); q.pop();

inq[u] = false;

for (int i = head[u]; i != -1; i = e[i].next) {

int v = e[i].v;

if (e[i].cap > 0 && dis[v] > dis[u] + e[i].cost) {

dis[v] = dis[u] + e[i].cost;

pre[v] = u; cur[v] = i;

if (!inq[v]) { inq[v] = true; q.push(v); }

}

}

}

if (pre[t] == -1) break;

int aug = INF;

for (int i = t; i != s; i = pre[i])

aug = min(aug, e[cur[i]].cap);

mf += aug;

mc += dis[t] \* aug;

for (int i = t; i != s; i = pre[i]) {

e[cur[i]].cap -= aug;

e[cur[i] ^ 1].cap += aug;

}

}

return make\_pair(mf, mc);

}

const int MAXN = 110;

int n, m, k;

int candy\_s[MAXN], candy\_t[MAXN];

int start\_time[MAXN][MAXN], start\_cost[MAXN][MAXN];

int change\_time[MAXN][MAXN], change\_cost[MAXN][MAXN];

void read\_matrix(int a[MAXN][MAXN], int n, int m) {

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

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

scanf("%d", &a[i][j]);

}

void work() {

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

scanf("%d%d", candy\_s + i, candy\_t + i);

read\_matrix(start\_time, n, m);

read\_matrix(start\_cost, n, m);

read\_matrix(change\_time, n, n);

read\_matrix(change\_cost, n, n);

graphinit();

int ss = 2 \* n + m, tt = ss + 1;

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

addedge(ss, i, 1, 0);

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

addedge(i + n, tt, 1, 0);

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

addedge(i + n + m, tt, 1, 0);

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

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

if (start\_time[i][j] >= candy\_t[i]) continue;

int cost = start\_cost[i][j];

if (start\_time[i][j] > candy\_s[i])

cost += k \* (start\_time[i][j] - candy\_s[i]);

addedge(i, j + n, 1, cost);

}

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

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

int dt = candy\_t[i] + change\_time[i][j];

if (dt >= candy\_t[j]) continue;

int cost = change\_cost[i][j];

dt -= candy\_s[j];

if (dt > 0)

cost += k \* dt;

addedge(j, i + n + m, 1, cost);

}

pair<int, int> ans = mcmf(ss, tt);

if (ans.first < n) puts("-1");

else printf("%d\n", ans.second);

}

int main() {

while (scanf("%d%d%d", &n, &m, &k), n || m || k) {

work();

}

return 0;

}

## HDU 4810 Wall Painting

Ms.Fang loves painting very much. She paints GFW(Great Funny Wall) every day. Every day before painting, she produces a wonderful color of pigments by mixing water and some bags of pigments. On the K-th day, she will select K specific bags of pigments and mix them to get a color of pigments which she will use that day. When she mixes a bag of pigments with color A and a bag of pigments with color B, she will get pigments with color A xor B.

When she mixes two bags of pigments with the same color, she will get color zero for some strange reasons. Now, her husband Mr.Fang has no idea about which K bags of pigments Ms.Fang will select on the K-th day. He wonders the sum of the colors Ms.Fang will get with different plans.

For example, assume n = 3, K = 2 and three bags of pigments with color 2, 1, 2. She can get color 3, 3, 0 with 3 different plans. In this instance, the answer Mr.Fang wants to get on the second day is 3 + 3 + 0 = 6.

Mr.Fang is so busy that he doesn’t want to spend too much time on it. Can you help him?

You should tell Mr.Fang the answer from the first day to the n-th day.

输入描述:

There are several test cases, please process till EOF.

For each test case, the first line contains a single integer N(1 <= N <= 10 3).The second line contains N integers. The i-th integer represents the color of the pigments in the i-th bag.

输出描述:

For each test case, output N integers in a line representing the answers(mod 10 6 +3) from the first day to the n-th day.

参考答案

#include<stdio.h>

#include<string.h>

#include<map>

#include<algorithm>

using namespace std;

#define mod 1000003

int n;

int a[1001];

long long ans[1001];

int sum;

long long c[1001][1001];

int num[33];

inline void init()

{

memset(c,0,sizeof(c));

c[0][0]=1;

for(int i=1;i<=1000;i++) c[i][0]=1,c[i][i]=1;

for(int i=2;i<=1000;i++)

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

c[i][j]=(c[i-1][j-1]+c[i-1][j])%mod;

}

inline void chuli(int t)

{

int val=t;

int s=1;

while(val)

{

if(val%2)

num[s]++;

val/=2;

s++;

}

}

int main()

{

init();

while(~scanf("%d",&n))

{

memset(ans,0,sizeof(ans));

memset(num,0,sizeof(num));

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

{

scanf("%d",&a[i]);

chuli(a[i]);

}

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

{

for(int j=1;j<=32;j++)

{

for(int k=1;k<=i;k+=2)

ans[i]+=(c[num[j]][k]\*(c[n-num[j]][i-k])%mod)\*((1\*1LL)<<(j-1) %mod),ans[i]%=mod;

}

}

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

{

printf("%I64d",ans[i]%mod);

if(i==n)

printf("\n");

else

printf(" ");

}

}

}

## HDU 4833 Best Financing

小A想通过合理投资银行理财产品达到收益最大化。已知小A在未来一段时间中的收入情况，描述为两个长度为n的整数数组dates和earnings，表示在第dates[i]天小A收入earnings[i]元（0<=i<n）。银行推出的理财产品均为周期和收益确定的，可描述为长度为m的三个整数数组start、finish和interest\_rates, 若购买理财产品i（0<=i<m），需要在第start[i]天投入本金，在第finish[i]天可取回本金和收益，在这期间本金和收益都无法取回，收益为本金\*interest\_rates[i]/100.0。当天取得的收入或理财产品到期取回的本金当天即可购买理财产品（注意：不考虑复利，即购买理财产品获得的收益不能用于购买后续的理财产品）。假定闲置的钱没有其他收益，如活期收益等，所有收益只能通过购买这些理财产品获得。求小A可以获得的最大收益。

限制条件:

1<=n<=2500

1<=m<=2500

对于任意i（0<=i<n）,1<=dates[i]<=100000,1<=earnings[i]<=100000, dates中无重复元素。

对于任意i（0<=i<m）,1<=start[i]<finish[i]<=100000, 1<=interest\_rates[i]<=100。

输入描述:

第一行为T (T<=200)，表示输入数据组数。

每组数据格式如下：

第一行是n m

之后连续n行，每行为两个以空格分隔的整数，依次为date和earning

之后连续m行，每行为三个以空格分隔的整数，依次为start, finish和interest\_rate

输出描述:

对第i组数据，i从1开始计，输出

Case #i:

收益数值，保留小数点后两位，四舍五入。

代码：

#include <stdio.h>

#include <string.h>

#include <iostream>

#include <algorithm>

#include <vector>

#include <queue>

#include <set>

#include <map>

#include <string>

#include <math.h>

#include <stdlib.h>

#include <time.h>

using namespace std;

struct NN1

{

int d,e;

void input()

{

scanf("%d%d",&d,&e);

}

}node1[3000];

struct NN2

{

int start,finish;

int r;

void input()

{

scanf("%d%d%d",&start,&finish,&r);

}

}node2[3000];

int a[5010];

long long f[100010];

long long f2[5010];

int dp[5010];

vector<int>vec[5010];

vector<int>vec2[5010];

int main()

{

int T;

int iCase = 0;

int n,m;

scanf("%d",&T);

while(T--)

{

iCase++;

printf("Case #%d:\n",iCase);

scanf("%d%d",&n,&m);

int cnt = 0;

memset(f,0,sizeof(f));

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

{

node1[i].input();

f[node1[i].d] += node1[i].e;

}

for(int i = 1;i <= 100000;i++)

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

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

{

node2[i].input();

a[cnt++] = node2[i].start;

a[cnt++] = node2[i].finish;

}

sort(a,a+cnt);

cnt = unique(a,a+cnt) - a;

map<int,int>mp;

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

mp[a[i]] = i;

f2[0] = f[a[0]];

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

f2[i] = f[a[i]] - f[a[i-1]];

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

{

vec[i].clear();

vec2[i].clear();

}

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

{

node2[i].start = mp[node2[i].start];

node2[i].finish = mp[node2[i].finish];

vec[node2[i].start].push\_back(node2[i].finish);

vec2[node2[i].start].push\_back(node2[i].r);

}

memset(dp,0,sizeof(dp));

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

{

dp[i] = dp[i+1];

int sz = vec[i].size();

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

dp[i] = max(dp[i],dp[vec[i][j]] + vec2[i][j]);

}

long long ans ;

ans = 0;

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

{

ans += (long long)dp[i]\*f2[i];

}

printf("%.2lf\n",(double)ans/100);

}

return 0;

}

## HDU 5199 Gunner

Long long ago, there is a gunner whose name is Jack. He likes to go hunting very much. One day he go to the grove. There are nn birds and nn trees. The i−thi−th bird stands on the top of the i−thi−th tree. The trees stand in straight line from left to the right. Every tree has its height. Jack stands on the left side of the left most tree. When Jack shots a bullet in height H to the right, the bird which stands in the tree with height HH will falls.

Jack will shot many times, he wants to know how many birds fall during each shot.

a bullet can hit many birds, as long as they stand on the top of the tree with height of HH.

输入描述:

There are multiple test cases (about 5), every case gives $n,m$ in the first line, $n$ indicates there are $n$ trees and $n$ birds, $m$ means Jack will shot $m$ times.

In the second line, there are $n$ numbers $h[1],h[2],h[3],…,h[n]$ which describes the height of the trees.

In the third line, there are m numbers $q[1],q[2],q[3],…,q[m]$ which describes the height of the Jack’s shots.

Please process to the end of file.

[Technical Specification]

$1 \leq n,m \leq 1000000(10^{6})$

$1 \leq h[i],q[i] \leq 1000000000(10^{9})$

All inputs are integers.

输出描述:

For each $q[i]$, output an integer in a single line indicates the number of birds Jack shot down.

代码：

#include<cstdio>

#include <cstring>

#include <algorithm>

#include <queue>

#define MAXN 1000010

using namespace std;

struct node{

int id;

int h;

}q[MAXN];

int ans[MAXN];

priority\_queue<int,vector<int>,greater<int> > pq;

bool cmp(node a,node b){

if(a.h==b.h) return a.id<b.id;

return a.h<b.h;

}

int n,m;

int main(){

while(scanf("%d%d",&n,&m)==2){

while(!pq.empty()) pq.pop();

int tm;

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

{

scanf("%d",&tm);

pq.push(tm);

}

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

scanf("%d",&q[i].h);

q[i].id=i;

}

sort(q,q+m,cmp);

memset(ans,0,sizeof(ans));

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

int tans=0;

while(!pq.empty()&&pq.top()<q[i].h) pq.pop();

while(!pq.empty()&&i<m&&pq.top()==q[i].h){

tans++;

pq.pop();

}

ans[q[i].id]=tans;

if(pq.empty())

break;

}

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

printf("%d\n",ans[i]);

}

return 0;

}

## HDU 5202 Rikka with string

As we know, Rikka is poor at math. Yuta is worrying about this situation, so he gives Rikka some math tasks to practice. There is one of them:

One day, Yuta got a string which contains n&#8201;letters but Rikka lost it in accident. Now they want to recover the string. Yuta remembers that the string only contains lowercase letters and it is not a palindrome string. Unfortunately he cannot remember some letters. Can you help him recover the string?

It is too difficult for Rikka. Can you help her?

输入描述:

This problem has multi test cases (no more than $20$). For each test case, The first line contains a number $n(1 \leq n \leq 1000)$. The next line contains an n-length string which only contains lowercase letters and ‘?’ � the place which Yuta is not sure.

输出描述:

For each test cases print a n-length string � the string you come up with. In the case where more than one string exists, print the lexicographically &#64257;rst one. In the case where no such string exists, output “QwQ”.

代码：

#include <stdio.h>

#include <string.h>

const int N = 1005;

char str[N];

int n, pos[N], num, len, cnt;

bool judge() {

for (int i = 0, j = len - 1; i < len; i++, j--)

if (str[i] != str[j])

return false;

return true;

}

int main() {

while (scanf("%d", &n) == 1) {

scanf("%s", str);

num = 0;

len = strlen(str);

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

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

str[i] = 'a';

pos[num++] = i;

}

}

if (num == 0 && !judge()) {

printf("%s\n", str);

continue;

}

if (num == 0 && judge() || len == 1) {

printf("QwQ\n");

continue;

}

if (len & 1) {

if (!judge()) {

printf("%s\n", str);

continue;

}

bool flag = judge();

if (flag && pos[num - 1] != len / 2) {

str[pos[num - 1]] = 'b';

printf("%s\n", str);

}

else if (flag && pos[num - 1] == len / 2 && num == 1)

printf("QwQ\n");

else if (flag && pos[num - 1] == len / 2 && num > 1) {

str[pos[num - 2]] = 'b';

printf("%s\n", str);

}

}

else {

if (judge())

str[pos[num - 1]] = 'b';

printf("%s\n", str);

}

}

return 0;

}

## HDU 5309 JRY is Fighting

Long long ago, there is a hero fighting against the emperor JRY. At the very beginning, the hero has mm HPs(health-points). There points represent his health - if ever they fall below or equal to zero, the hero will die. In the following nn seconds, he will be hurt by XXY. At the ii seconds, his HP will reduce by hihi. If hi<0hi<0, it means his HP will increase by |hi||hi|.

The hero has a magic bottle which can store HPs. At first, the bottle is empty. Each time after the hero got hurt, the bottle can get kk more HPs, and the hero can decide whether he will release the HPs in the bottle. If he does, he will gain the HPs in the bottle and the bottle will be empty.

We define the hero's operating sequence as ss, representing that he used the magic bottle at the sisi-th seconds. |s||s| represent the times he used, as well as the length of the sequence.

Now, you should maximize the mininum time interval between two adjacent operation. In other words, let T=max{min{si−si−1}(1<i≤|s|)}T=max{min{si−si−1}(1<i≤|s|)}, you should find the value of TT. We can easily find that if |s|≤1|s|≤1, T=+∞T=+∞.

You should give him a plan as an operating sequence ss which is right for the hero to survive successfully. The hero is so strict that you should find the lexicographically smallest one.

Sequence u1,u2,⋯,unu1,u2,⋯,un is lexicographically smaller than sequence v1,v2,⋯,vmv1,v2,⋯,vm, if

n<mn<m and u1=v1,u2=v2,⋯,un=vnu1=v1,u2=v2,⋯,un=vn, or

there exists an integer k(1≤k≤min(n,m))k(1≤k≤min(n,m)) where u1=v1,u2=v2,⋯,uk−1=vk−1u1=v1,u2=v2,⋯,uk−1=vk−1 and uk<vkuk<vk all hold.

输入描述:

There are multiple testcases, the sum of $n$ is less then $10^6$.

The first line contains three space-separated integers each, $n(1\leq n\leq 500000)$, $m(1\leq m \leq 10^6)$, $k(1\leq k \leq 100)$.

The second line contains $n$ space-separated integers, $a\_i(0\leq |a\_i| \leq 100)$.

输出描述:

If the hero can't survive, print "Poor Hero!".

If $T=+\infty$, print "Poor JRY!".

Otherwise, print three lines:

The first line, an integer, representing the value of $T$.

The second line, an integer, $|s|$.

The third line, $|s|$ space-separated intergers, $s\_i$.

参考答案

#include<iostream>

#include<cstdio>

#include<cmath>

#include<queue>

#include<cstring>

#include<algorithm>

using namespace std;

#define sf scanf

#define mx 100000000

struct node{

int id,v;

//friend bool operator<(node a,node b){return (a.v==b.v)?(a.id>b.id):(a.v>b.v);}

bool operator < (const node &a)const {

if(a.v==v)return id>a.id;

return v<a.v;

}

};

priority\_queue<node> que1,que2;

queue<int> que;

int n,m,k;

int g[1000000],sum[1000000],a[1000000],r[1000000];

int main()

{

while(~scanf("%d%d%d",&n,&m,&k))

{

while(!que1.empty()) que1.pop();

while(!que2.empty())que2.pop();

for(int i = 1;i<=n;i++) {sf("%d",a+i);sum[i]=sum[i-1]+a[i];}

int t = 0;

for(int i = 0;i<=n;i++){while(t<=n&&sum[t]<i\*k + m) t++;r[i] = t-1;}

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

{

g[i] = 0;

if(r[i] == n) g[i] = mx + i;

else

{

while(!que1.empty()){

node tmp = que1.top();

if(tmp.id>r[i])que1.pop();

else if(tmp.v\*2>=g[tmp.id] + i)

{

que2.push(node{tmp.id,g[tmp.id]-tmp.id});

que1.pop();

}

else break;

}

while(!que2.empty())

{

node tmp = que2.top();

if(tmp.id>r[i])que2.pop();

else break;

}

if(!que1.empty())g[i] = max(g[i],que1.top().v);

if(!que2.empty())g[i] = max(g[i],que2.top().v+i);

}

if(g[i]>i) que1.push((node){i,i});

else que2.push((node){i,g[i]-i});

}

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

g[0] = 0;

for(int i = 1;i<=r[0];i++) g[0] = max(g[0],g[i]);

if(g[0] > n) puts("Poor JRY!");

else if(!g[0]) puts("Poor Hero!");

else

{

printf("%d\n",g[0]);

int i = 1;

while(g[i]!=g[0]) i++;

que.push(i);

while(g[i]!=mx)

{

int j=i+g[0];

while(g[j]<g[0]) {

j++;

}

i = j;

que.push(i);

}

printf("%d\n",que.size());

while(!que.empty())

{

int tmp = que.front();

printf("%d",tmp);

que.pop();

if(que.size())printf(" ");

else printf("\n");

}

}

}

return 0;

}

## HDU 5326 Work



It’s an interesting experience to move from ICPC to work, end my college life and start a brand new journey in company.

As is known to all, every stuff in a company has a title, everyone except the boss has a direct leader, and all the relationship forms a tree. If A’s title is higher than B(A is the direct or indirect leader of B), we call it A manages B.

Now, give you the relation of a company, can you calculate how many people manage k people.

输入描述:

There are multiple test cases.

Each test case begins with two integers n and k, n indicates the number of stuff of the company.

Each of the following n-1 lines has two integers A and B, means A is the direct leader of B.

1 <= n <= 100 , 0 <= k < n

1 <= A, B <= n

输出描述:

For each test case, output the answer as described above.

参考答案

#include <iostream>

#include <stdio.h>

#include <memory.h>

using namespace std;

const int MAX=100+5;

int a[MAX][MAX];

int b[MAX];

void dfs(int j,int jj,int num,int n)

{

for(int k=jj;k<=n;k++)

{

if(a[j][ k ])

{

b[num]++;

dfs(k,1,num,n);

}

}

}

int main()

{

int n,k;

while(~scanf("%d %d",&n,&k))

{

int ans=0;

int A,B;

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

memset(b,0,sizeof(b));

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

{

scanf("%d %d",&A,&B);

a[A][B]=1;

}

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

{

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

{

if(a[i][j])

{

b[i]++;

dfs(j,1,i,n);

}

}

}

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

{

if(b[i]==k)

{

ans++;

}

}

printf("%d\n",ans);

}

return 0;

}

## HDU 5406 CRB and Apple

In Codeland there are many apple trees.

One day CRB and his girlfriend decided to eat all apples of one tree.

Each apple on the tree has height and deliciousness.

They decided to gather all apples from top to bottom, so an apple can be gathered only when it has equal or less height than one just gathered before.

When an apple is gathered, they do one of the following actions.

1. CRB eats the apple.

2. His girlfriend eats the apple.

3. Throw the apple away.

CRB(or his girlfriend) can eat the apple only when it has equal or greater deliciousness than one he(she) just ate before.

CRB wants to know the maximum total number of apples they can eat.

Can you help him?

输入描述:

There are multiple test cases. The first line of input contains an integer $T$, indicating the number of test cases. For each test case:

The first line contains a single integer $N$ denoting the number of apples in a tree.

Then $N$ lines follow, $i$-th of them contains two integers $H\_{i}$ and $D\_{i}$ indicating the height and deliciousness of $i$-th apple.

1 ≤ $T$ ≤ 48

1 ≤ $N$ ≤ 1000

1 ≤ $H\_{i}$, $D\_{i}$ ≤ $10^{9}$

输出描述:

For each test case, output the maximum total number of apples they can eat.

参考答案

#include <cstdio>

#include <algorithm>

#include <iostream>

#include <cmath>

#include <vector>

#include <cstring>

#include <map>

#include <set>

#include <unordered\_map>

#include <queue>

#include <sstream>

using namespace std;

const int MAXN = 1005;

struct A{

int H, D;

bool operator<(const A rhs)const{

if(H != rhs.H)

return H > rhs.H;

else

return D < rhs.D;

}

}arr[MAXN];

int N;

int dval[MAXN], dvn;

#define MAXP 2010

#define MAXQ 1000005

#define MAXI 0x7fffffff

#define MAXC 0x0fffffff

#define BIGM MAXC

struct EDGE{

int st, ed;

int flow;

int low;

int upf;

int cost;

int hnext;

int tnext;

int hprev;

int tprev;

char set;

}edge[MAXQ+MAXP];

int ecnt;

int ecnt2;

int sece;

bool search\_method;

struct NODE{

int head;

int tail;

int d;

int pi;

int pred;

int depth;

}node[MAXP];

int ncnt;

struct LOOP{

int e;

int prev;

int next;

}loop[MAXP+3];

int lcnt;

int addedge(int a, int b, int low, int upf, int cost, int &cnt)

{

edge[cnt].st = a;

edge[cnt].ed = b;

edge[cnt].low = low;

edge[cnt].upf = upf;

edge[cnt].cost = cost;

edge[cnt].hnext = -1;

edge[cnt].tnext = -1;

edge[cnt].hprev = -1;

edge[cnt].tprev = -1;

cnt++;

return (cnt - 1);

}

void insert\_tree(int j)

{

int a, b;

a = edge[j].st;

b = edge[j].ed;

edge[j].hprev = -1;

edge[j].hnext = node[a].head;

if(node[a].head >= 0)

edge[node[a].head].hprev = j;

node[a].head = j;

edge[j].tprev = -1;

edge[j].tnext = node[b].tail;

if(node[b].tail >= 0)

edge[node[b].tail].tprev = j;

node[b].tail = j;

}

void delete\_tree(int j)

{

int a, b;

a = edge[j].st;

b = edge[j].ed;

if(edge[j].hprev >= 0)

{

edge[edge[j].hprev].hnext = edge[j].hnext;

}else{

node[a].head = edge[j].hnext;

}

if(edge[j].hnext >= 0)

{

edge[edge[j].hnext].hprev = edge[j].hprev;

}

edge[j].hprev = -1;

edge[j].hnext = -1;

if(edge[j].tprev >= 0)

{

edge[edge[j].tprev].tnext = edge[j].tnext;

}else{

node[b].tail = edge[j].tnext;

}

if(edge[j].tnext >= 0)

{

edge[edge[j].tnext].tprev = edge[j].tprev;

}

edge[j].tprev = -1;

edge[j].tnext = -1;

}

void M\_init()

{

int i, j;

node[0].d = 0;

node[0].head = -1;

node[0].tail = -1;

for(i=0;i<ecnt;i++)

{

edge[i].flow = edge[i].low;

edge[i].set = 'L';

node[edge[i].st].d -= edge[i].low;

node[edge[i].ed].d += edge[i].low;

}

ecnt2 = ecnt;

for(i=1;i<=ncnt;i++)

{

if(node[i].d > 0)

{

j = addedge(i, 0, 0, MAXC, BIGM, ecnt2);

edge[j].flow = node[i].d;

}else{

j = addedge(0, i, 0, MAXC, BIGM, ecnt2);

edge[j].flow = -node[i].d;

}

edge[j].set = 'T';

insert\_tree(j);

}

}

void calc\_pi(int root, int val, int dep, int pred)

{

int j;

node[root].pi = val;

node[root].depth = dep;

node[root].pred = pred;

for(j=node[root].head;j!=-1;j=edge[j].hnext)

{

if(j != pred)

{

calc\_pi(edge[j].ed, val - edge[j].cost, dep + 1, j);

}

}

for(j=node[root].tail;j!=-1;j=edge[j].tnext)

{

if(j != pred)

{

calc\_pi(edge[j].st, val + edge[j].cost, dep + 1, j);

}

}

}

int calc\_dual1()

{

int i, ret = -1, dual0 = -1, dual;

for(i=0;i<ecnt2;i++)

{

dual = edge[i].cost - node[edge[i].st].pi + node[edge[i].ed].pi;

if((edge[i].set == 'L' && dual < 0)||(edge[i].set == 'U' && dual > 0))

{

dual = labs(dual);

if(dual > dual0)

{

ret = i;

dual0 = dual;

}

}

}

return ret;

}

int calc\_dual2()

{

int i, dual;

for(i=sece;i<ecnt2;i++)

{

dual = edge[i].cost - node[edge[i].st].pi + node[edge[i].ed].pi;

if((edge[i].set == 'L' && dual < 0)||(edge[i].set == 'U' && dual > 0))

{

sece = i + 1;

return i;

}

}

for(i=0;i<sece;i++)

{

dual = edge[i].cost - node[edge[i].st].pi + node[edge[i].ed].pi;

if((edge[i].set == 'L' && dual < 0)||(edge[i].set == 'U' && dual > 0))

{

sece = i + 1;

return i;

}

}

return -1;

}

bool augment(int rda)

{

int p, q;

int pl, ql, pp, qp, maxflow;

bool flagp, flagq, flag, ret;

lcnt = 0;

if(edge[rda].set == 'L')

{

p = edge[rda].st;

q = edge[rda].ed;

}else{

q = edge[rda].st;

p = edge[rda].ed;

}

loop[lcnt].e = rda;

loop[lcnt].prev = 1;

loop[lcnt].next = 2;

pl = 1;

ql = 2;

loop[pl].e = -1;

loop[pl].next = 0;

loop[pl].prev = -1;

loop[ql].e = -1;

loop[ql].next = -1;

loop[ql].prev = 0;

lcnt = 3;

maxflow = edge[rda].upf - edge[rda].low;

while(p != q)

{

if(node[p].depth > node[q].depth)

{

flagp = true;

flagq = false;

}else if(node[p].depth < node[q].depth)

{

flagp = false;

flagq = true;

}else{

flagp = true;

flagq = true;

}

if(flagp)

{

pp = node[p].pred;

if(edge[pp].ed == p)

{

maxflow = min(maxflow, edge[pp].upf - edge[pp].flow);

p = edge[pp].st;

}else{

maxflow = min(maxflow, edge[pp].flow - edge[pp].low);

p = edge[pp].ed;

}

loop[pl].e = pp;

loop[pl].prev = lcnt;

loop[lcnt].e = -1;

loop[lcnt].next = pl;

loop[lcnt].prev = -1;

pl = lcnt;

lcnt++;

}

if(flagq)

{

qp = node[q].pred;

if(edge[qp].st == q)

{

maxflow = min(maxflow, edge[qp].upf - edge[qp].flow);

q = edge[qp].ed;

}else{

maxflow = min(maxflow, edge[qp].flow - edge[qp].low);

q = edge[qp].st;

}

loop[ql].e = qp;

loop[ql].next = lcnt;

loop[lcnt].e = -1;

loop[lcnt].next = -1;

loop[lcnt].prev = ql;

ql = lcnt;

lcnt++;

}

}

flag = false;

while(loop[pl].next >= 0)

{

pl = loop[pl].next;

ql = loop[pl].e;

if(ql == -1)

break;

if(ql == rda)

{

ret = (flag == (edge[rda].set == 'L'));

}

if(edge[ql].st == p)

{

edge[ql].flow += maxflow;

if(!flag && (edge[ql].flow == edge[ql].upf))

{

edge[ql].set = 'U';

flag = true;

delete\_tree(ql);

}else{

edge[ql].set = 'T';

}

p = edge[ql].ed;

}else{

edge[ql].flow -= maxflow;

if(!flag && (edge[ql].flow == edge[ql].low))

{

edge[ql].set = 'L';

flag = true;

delete\_tree(ql);

}else{

edge[ql].set = 'T';

}

p = edge[ql].st;

}

}

return ret;

}

bool NSMCalc(int &cost)

{

int rda, i;

M\_init();

calc\_pi(0, 0, 0, -1);

rda = (search\_method ? calc\_dual1() : calc\_dual2());

while(rda >= 0)

{

int j;

insert\_tree(rda);

if(augment(rda))

{

i = edge[rda].st;

j = edge[rda].ed;

calc\_pi(i, node[j].pi + edge[rda].cost, node[j].depth + 1, rda);

}else{

i = edge[rda].ed;

j = edge[rda].st;

calc\_pi(i, node[j].pi - edge[rda].cost, node[j].depth + 1, rda);

}

rda = (search\_method ? calc\_dual1() : calc\_dual2());

}

for(i=ecnt;i<ecnt2;i++)

{

if(edge[i].flow > 0)

return false;

}

cost = 0;

for(i=0;i<ecnt;i++)

{

cost += edge[i].flow \* edge[i].cost;

}

return true;

}

void init\_graph(int pointnum, bool method)

{

int i;

ncnt = pointnum;

ecnt = 0;

sece = 0;

for(i=0;i<=ncnt;i++)

{

node[i].d = 0;

node[i].head = -1;

node[i].tail = -1;

node[i].pred = -1;

node[i].pi = 0;

node[i].depth = 0;

}

search\_method = method;

}

inline char getc(){

static const int BUFLEN = 1 << 15;

static char B[BUFLEN], \*S = B, \*T = B;

if(S == T){

S = B;

T = S + fread(B, 1, BUFLEN, stdin);

}

return (S == T) ? 0 : \*(S ++);

}

int ReadInt(){

char ch;

do ch = getc(); while(!isdigit(ch));

int aa = ch - '0';

for(ch = getc(); isdigit(ch); ch = getc())

aa = aa \* 10 + ch - '0';

return aa;

}

int calc(){

int i, j;

int ret, mxv;

init\_graph(N \* 2 + 3, false);

addedge(0, 1, 0, 2, 0, ecnt);

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

addedge(1, i + 1, 0, 1, 0, ecnt);

addedge(i + N + 1, N \* 2 + 2, 0, 1, 0, ecnt);

addedge(i + 1, i + N + 1, 0, 1, -1, ecnt);

}

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

mxv = 0x3fffffff;

for(j = i + 1; j <= N; j ++){

if(arr[j].D < arr[i].D || arr[j].D >= mxv)

continue;

addedge(i + N + 1, j + 1, 0, 1, 0, ecnt);

addedge(i + 1, j + 1, 0, 1, 0, ecnt);

mxv = arr[j].D;

}

}

node[0].d = 2;

node[N \* 2 + 2].d = -2;

NSMCalc(ret);

return - ret;

}

bool deal1(){

int i;

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

if(arr[i + 1].D < arr[i].D)

return false;

}

printf("%d\n", N);

return true;

}

void domain(){

int i, ans = 0;

N = ReadInt();

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

arr[i].H = ReadInt();

arr[i].D = ReadInt();

}

sort(arr + 1, arr + N + 1);

dvn = 0;

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

dval[dvn ++] = arr[i].D;

}

sort(dval, dval + dvn);

dvn = unique(dval, dval + dvn) - dval;

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

arr[i].D = lower\_bound(dval, dval + dvn, arr[i].D) - dval + 1;

}

ans = calc();

printf("%d\n", ans);

}

int main(){

int T = ReadInt();

while(T --){

domain();

}

return 0;

}

## HDU 5495 LCS

You are given two sequence {a1,a2,...,an}{a1,a2,...,an} and {b1,b2,...,bn}{b1,b2,...,bn}. Both sequences are permutation of {1,2,...,n}{1,2,...,n}. You are going to find another permutation {p1,p2,...,pn}{p1,p2,...,pn} such that the length of LCS (longest common subsequence) of {ap1,ap2,...,apn}{ap1,ap2,...,apn} and {bp1,bp2,...,bpn}{bp1,bp2,...,bpn} is maximum. 

输入描述:

There are multiple test cases. The first line of input contains an integer $T$, indicating the number of test cases. For each test case:

The first line contains an integer $n (1 \le n \le 10^5)$ - the length of the permutation. The second line contains $n$ integers $a\_1,a\_2,...,a\_n$. The third line contains $n$ integers $b\_1,b\_2,...,b\_n$.

The sum of $n$ in the test cases will not exceed $2 \times 10^6$.

输出描述:

For each test case, output the maximum length of LCS.

代码：

#include<iostream>

#include<cstring>

#include<algorithm>

#include<cstdio>

using namespace std;

#define maxn 100007

int a[maxn],b[maxn],vis[maxn];

int dfs(int x){

vis[x] = 1;

if(vis[b[a[x]]]) return 1;

return dfs(b[a[x]])+1;

}

int main(){

int t,n,u;

scanf("%d",&t);

while(t--){

scanf("%d",&n);

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

scanf("%d",&u);

a[u] = i;

}

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

scanf("%d",&b[i]);

memset(vis,0,sizeof(vis));

int ans = n;

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

if(vis[i] == 0) {

if(dfs(i) > 1) ans--;

}

}

printf("%d\n",ans);

}

return 0;

}

# 第三部分 POJ

## poj 1077 Eight

The 15-puzzle has been around for over 100 years; even if you don't know it by that name, you've seen it. It is constructed with 15 sliding tiles, each with a number from 1 to 15 on it, and all packed into a 4 by 4 frame with one tile missing. Let's call the missing tile 'x'; the object of the puzzle is to arrange the tiles so that they are ordered as:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 x

where the only legal operation is to exchange 'x' with one of the tiles with which it shares an edge. As an example, the following sequence of moves solves a slightly scrambled puzzle:

1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4

5 6 7 8 5 6 7 8 5 6 7 8 5 6 7 8

9 x 10 12 9 10 x 12 9 10 11 12 9 10 11 12

13 14 11 15 13 14 11 15 13 14 x 15 13 14 15 x

r-> d-> r->

The letters in the previous row indicate which neighbor of the 'x' tile is swapped with the 'x' tile at each step; legal values are 'r','l','u' and 'd', for right, left, up, and down, respectively.

Not all puzzles can be solved; in 1870, a man named Sam Loyd was famous for distributing an unsolvable version of the puzzle, and

frustrating many people. In fact, all you have to do to make a regular puzzle into an unsolvable one is to swap two tiles (not counting the missing 'x' tile, of course).

In this problem, you will write a program for solving the less well-known 8-puzzle, composed of tiles on a three by three

arrangement.

输入描述

You will receive a description of a configuration of the 8 puzzle. The description is just a list of the tiles in their initial positions, with the rows listed from top to bottom, and the tiles listed from left to right within a row, where the tiles are represented by numbers 1 to 8, plus 'x'. For example, this puzzle

1 2 3

x 4 6

7 5 8

is described by this list:

1 2 3 x 4 6 7 5 8

输出描述

You will print to standard output either the word ``unsolvable'', if the puzzle has no solution, or a string consisting entirely of the letters 'r', 'l', 'u' and 'd' that describes a series of moves that produce a solution. The string should include no spaces and start at the beginning of the line.

输入例子

2 3 4 1 5 x 7 6 8

输出例子

ullddrurdllurdruldr

参考答案

#include <iostream>

#include <stack>

#include <math.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

using namespace std;

const int maxn = 10;

char ans[100];

int tot, dir[4][2] = {{-1,0},{0,1},{1,0},{0,-1}};

struct Node

{

char map[maxn];

int g, move, xpos;

}starts;

void init()

{

for (int i = 0; i < 9; i++)

{

starts.map[i] = ' ';

while (starts.map[i] == ' ')

scanf("%c",&starts.map[i]);

if (starts.map[i] == 'x')

{

starts.map[i] = 9;

starts.xpos = i;

} else

starts.map[i] -= '0';

}

}

int h(Node &a)

{

int x1, x2, y1, y2, i, r = 0;

for (i = 0; i < 9; i++)

{

x1 = i / 3;

y1 = i % 3;

x2 = (a.map[i] - 1) / 3;

y2 = (a.map[i] - 1) % 3;

r += abs(x1 - x2) + abs(y1 - y2);

}

return r;

}

Node getchild(int a, Node ¤ts)

{

int x, y, pos, i;

Node r;

x = currents.xpos / 3 + dir[a][0];

y = currents.xpos % 3 + dir[a][1];

r.xpos = -1;

if (x < 0 || y < 0 || x > 2 || y > 2)

return r;

pos = x \* 3 + y;

r.xpos = pos;

r.g = currents.g + 1;

r.move = a;

for (i = 0; i < 9; i++)

r.map[i] = currents.map[i];

r.map[pos] = 9;

r.map[currents.xpos] = currents.map[pos];

return r;

}

bool ida()

{

int pathlimit, i, temp, next;

bool success = 0;

Node currents, child;

next = h(starts)/2;

stack<Node> stk;

do

{

pathlimit = next;

if (pathlimit > 100)

return false;

tot = 0;

starts.g = 0;

starts.move = -1;

next = 200;

stk.push(starts);

do

{

currents = stk.top();

ans[currents.g] = currents.move;

stk.pop();

temp = h(currents);

if (temp == 0)

{

tot = currents.g;

success = true;

}

else if (pathlimit >= currents.g + temp / 2)

{

for (i = 0; i < 4; i++)

{

child = getchild(i, currents);

if (child.xpos != -1 && abs(child.move - currents.move) != 2)

stk.push(child);

}

}else if (next > currents.g + temp / 2)

next = currents.g + temp / 2;

}while (!success && !stk.empty());

}while (!success);

return true;

}

void print()

{

int i;

for (i = 1; i <= tot; i++)

switch(ans[i])

{

case 0: printf("u"); break;

case 1: printf("r"); break;

case 2: printf("d"); break;

case 3: printf("l"); break;

}

printf("\n");

}

int main()

{

init();

if (ida())

print();

else

printf("unsolvable\n");

return 0;

}

## poj 1090 Chain

Byteland had not always been a democratic country. There were also black pages in its book of history. One lovely day general Bytel − commander of the junta which had power over Byteland −− decided to finish the long−lasting time of war and released imprisoned activists of the opposition. However, he had no intention to let the leader Bytesar free. He decided to chain him to the wall with the bytish chain. It consists of joined rings and the bar fixed to the wall. Although the rings are not joined with the bar, it is hard to take them off.

'General, why have you chained me to the prison walls and did not let rejoice at freedom!' cried Bytesar.

'But Bytesar, you are not chained at all, and I am certain you are able to take off the rings from the bar by yourself.' perfidiously answered general Bytel, and he added 'But deal with that before a clock strikes the cyber hour and do not make a noise at night, otherwise I will be forced to call Civil Cyber Police.'

Help Bytesar! Number the following rings of the chain with integers 1,2,...,n. We may put on and take off these rings according to the following rules:

.only one ring can be put on or taken off from the bar in one move,

.the ring number 1 can be always put on or taken off from the bar,

.if the rings with the numbers 1,...,k−1 (for 1Write a program which:

.reads from std input the description of the bytish chain,

.computes minimal number of moves necessary to take off all rings of the bytish chain from the bar,

.writes the result to std output.

输入描述

In the first line of the input there is written one intege

输出描述

The output should contain exactly one integer equal to the minimal number of moves necessary to take off all the rings of the bytish chain from the bar.

输入例子

4

1 0 1 0

输出例子

6

参考答案

#include<iostream>

#include<string.h>

using namespace std;

int ko[1000];

int cun[1000][350];

int ans[1000];

int main()

{

int length,i,j,sum;

cin>>length;

sum=0;

for(i=0;i<length;i++)

{

cin>>ko[i];

sum=sum+ko[i];

}

for(i=0;i<length;i++)

{

sum=sum-ko[i];

if(sum%2==1)

ko[i]=1-ko[i];

}

memset(cun,0,sizeof(cun));

cun[0][0]=1;

for(i=1;i<length;i++)

{

for(j=0;j<=349;j++)

{

cun[i][j]=cun[i-1][j]\*2;

}

for(j=0;j<=349;j++)

{

if(cun[i][j]>10)

{

cun[i][j+1]=cun[i][j+1]+cun[i][j]/10;

cun[i][j]=cun[i][j]%10;

}

}

}

memset(ans,0,sizeof(ans));

for(i=0;i<length;i++)

{

if(ko[i]==0)

continue;

int carry=0;

for(j=0;j<=349;j++)

{

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

carry=ans[j]/10;

ans[j]=ans[j]%10;

}

}

int flag=-1;

for(i=400;i>=0;i--)

if(ans[i]!=0)

{

flag=i;

break;

}

if(flag==-1)

cout<<0;

for(i=flag;i>=0;i--)

cout<<ans[i];

cout<<endl;

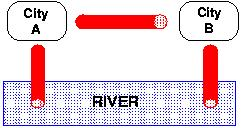
return 0;

}

## poj 1205 Water Treatment Plants

River polution control is a major challenge that authorities face in order to ensure future clean water supply. Sewage treatment plants are used to clean-up the dirty water comming from cities before being discharged into the river.   
  
As part of a coordinated plan, a pipeline is setup in order to connect cities to the sewage treatment plants distributed along the river. It is more efficient to have treatment plants running at maximum capacity and less-used ones switched off for a period. So, each city has its own treatment plant by the river and also a pipe to its neighbouring city upstream and a pipe to the next city downstream along the riverside. At each city's treatment plant there are three choices: 

* either process any water it may receive from one neighbouring city, together with its own dirty water, discharging the cleaned-up water into the river;
* or send its own dirty water, plus any from its downstream neighbour, along to the upstream neighbouring city's treatment plant (provided that city is not already using the pipe to send it's dirty water downstream);
* or send its own dirty water, plus any from the upstream neighbour, to the downstream neighbouring city's plant, if the pipe is not being used.

<center> </center>  
The choices above ensure that:   
  
every city must have its water treated somewhere and   
at least one city must discharge the cleaned water into the river.   
Let's represent a city discharging water into the river as "V" (a downwards flow), passing water onto its neighbours as ">" (to the next city on its right) or else ") and list the cities symbols in order. For example, two cities, A and B, can   
  
each treat their own sewage and each discharges clean water into the river. So A's action is denoted V as is B's and we write "VV" ;   
or else city A can send its sewage along the pipe (to the right) to B for treatment and discharge, denoted ">V";   
or else city B can send its sewage to (the left to) A, which treats it with its own dirty water and discharges (V) the cleaned water into the river. So A discharges (V) and B passes water to the left (We could not have ">   
So we have just 3 possible set-ups that fit the conditions: 

A B A > B A V V V V

RIVER~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~RIVER

"VV" ">V" "V

If we now consider three cities, we can determine 8 possible set-ups.   
Your task is to produce a program that given the number of cities NC (or treatment plants) in the river bank, determines the number of possible set-ups, NS, that can be made according to the rules define above.   
  
You need to be careful with your design as the number of cities can be as large as 100. 

输入描述

The input consists of a sequence of values, one per line, where each value represents the number of cities.

输出描述

Your output should be a sequence of values, one per line, where each value represents the number of possible set-ups for the corresponding number of cities read in the same input line.

输入例子

2

3

20

输出例子

3

8

102334155

代码：

#include<iostream>

#include<string>

#include<algorithm>

using namespace std;

string add(string s1,string s2)

{

string result;

result.reserve(1000);

if(s1.length()<s2.length())

s1.swap(s2);

s2.insert(s2.begin(),s1.length()-s2.length(),'0');

int g=0;

for(int i=s1.length()-1;i>=0;i--)

{

g+=s1[i]+s2[i]-2\*'0';

result+=g%10+'0';

g/=10;

}

if(g>0)

result+=g+'0';

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

return result;

}

string a[101];

void Initial()

{

a[1]="1";

for(int i=2;i<=100;i++)

{

string temp;

temp="1";

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

temp=add(temp,a[k]);

string te=add(a[i-1],a[i-1]);

temp=add(temp,te);

a[i]=temp;

}

}

int main(int argc, char\* argv[])

{

Initial();

int x;

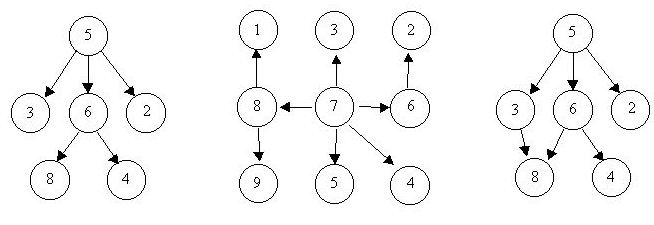
while(cin>>x)

cout<<a[x]<<endl;

return 0;

}

## poj 1308 Is It A Tree?

A tree is a well-known data structure that is either empty (null, void, nothing) or is a set of one or more nodes connected by directed edges between nodes satisfying the following properties.   
  
There is exactly one node, called the root, to which no directed edges point.   
Every node except the root has exactly one edge pointing to it.   
There is a unique sequence of directed edges from the root to each node.   
For example, consider the illustrations below, in which nodes are represented by circles and edges are represented by lines with arrowheads. The first two of these are trees, but the last is not.   
<center> </center>  
In this problem you will be given several descriptions of collections of nodes connected by directed edges. For each of these you are to determine if the collection satisfies the definition of a tree or not.

输入描述

The input will consist of a sequence of descriptions (test cases) followed by a pair of negative integers. Each test case will consist of a sequence of edge descriptions followed by a pair of zeroes Each edge description will consist of a pair of integers; the first integer identifies the node from which the edge begins, and the second integer identifies the node to which the edge is directed. Node numbers will always be greater than zero.

输出描述

For each test case display the line "Case k is a tree." or the line "Case k is not a tree.", where k corresponds to the test case number (they are sequentially numbered starting with 1).

输入例子

6 8 5 3 5 2 6 4

5 6 0 0

8 1 7 3 6 2 8 9 7 5

7 4 7 8 7 6 0 0

3 8 6 8 6 4

5 3 5 6 5 2 0 0

-1 -1

输出例子

Case 1 is a tree.

Case 2 is a tree.

Case 3 is not a tree.

代码：

#include <iostream>

#include <cstdio>

#include <cstdlib>

#include <cstring>

using namespace std;

#define maxn 1006

struct Edge

{

int v, next;

}edge[1000005];

int name[maxn];

int n, ecount, tot, root;

int in[maxn];

int head[maxn];

bool vis[maxn];

bool ok;

void addedge(int a, int b)

{

edge[ecount].next = head[a];

edge[ecount].v = b;

head[a] = ecount;

in[b]++;

ecount++;

}

int find(int a)

{

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

if (name[i] == a)

return i;

name[n++] = a;

return n - 1;

}

void dfs(int root)

{

vis[root] = true;

tot++;

for (int i = head[root]; i != -1; i = edge[i].next)

{

if (vis[edge[i].v])

{

ok = false;

return;

}

if (!ok)

return;

dfs(edge[i].v);

}

}

int main()

{

int t = 0;

while (1)

{

t++;

memset(in, 0, sizeof(in));

memset(head, -1, sizeof(head));

memset(vis, 0, sizeof(vis));

ecount = 0;

n = 0;

tot = 0;

ok = true;

int a, b;

scanf("%d%d", &a, &b);

if (a < 0 && b < 0)

return 0;

if (a == 0 && b == 0)

{

printf("Case %d is a tree.\n", t);

continue;

}

a = find(a);

b = find(b);

addedge(a, b);

while (1)

{

scanf("%d%d", &a, &b);

if (a == 0 && b == 0)

break;

a = find(a);

b = find(b);

addedge(a, b);

}

root = -1;

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

if (in[i] == 0)

root = i;

if (root >= 0)

dfs(root);

if (ok && tot == n && root >= 0)

printf("Case %d is a tree.\n", t);

else

printf("Case %d is not a tree.\n", t);

}

return 0;

}

## poj 1313 Booklet Printing

When printing out a document, normally the first page is printed first, then the second, then the third, and so on until the end. However, when creating a fold-over booklet, the order of printing must be altered. A fold-over booklet has four pages per sheet, with two on the front and two on the back. When you stack all the sheets in order, then fold the booklet in half, the pages appear in the correct order as in a regular book. For example, a 4-page booklet would print on 1 sheet of paper: the front will contain page 4 then page 1, and the back will contain page 2 then page 3.

Front Back

------------- -------------

| | | | | |

| 4 | 1 | | 2 | 3 |

| | | | | |

------------- -------------

Your task is to write a program that takes as input the number of pages to be printed, then generates the printing order.

输入描述

The input contains one or more test cases, followed by a line containing the number 0 that indicates the end of the file. Each test case consists of a positive integer n on a line by itself, where n is the number of pages to be printed; n will not exceed 100.

输出描述

For each test case, output a report indicating which pages should be printed on each sheet, exactly as shown in the example. If the desired number of pages does not completely fill up a sheet, then print the word Blank in place of a number. If the front or back of a sheet is entirely blank, do not generate output for that side of the sheet. Output must be in ascending order by sheet, front first, then back.

输入例子

1

14

4

0

输出例子

Printing order for 1 pages:

Sheet 1, front: Blank, 1

Printing order for 14 pages:

Sheet 1, front: Blank, 1

Sheet 1, back : 2, Blank

Sheet 2, front: 14, 3

Sheet 2, back : 4, 13

Sheet 3, front: 12, 5

Sheet 3, back : 6, 11

Sheet 4, front: 10, 7

Sheet 4, back : 8, 9

Printing order for 4 pages:

Sheet 1, front: 4, 1

Sheet 1, back : 2, 3

参考答案

#include <iostream>

#include <cstdio>

#include <cstdlib>

#include <cstring>

using namespace std;

#define maxn 105

int f[maxn][2];

int n;

void work()

{

int page\_num = (n + 3) / 4;

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

f[i][(i & 1) ^ 1] = i + 1;

int j = page\_num \* 2 + 1;

for (int i = page\_num \* 2 - 1; i >= 0; i--, j++)

f[i][i & 1] = j;

}

void output()

{

int page\_num = (n + 3) / 4;

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

{

if (f[i][0] > n && f[i][1] > n)

continue;

printf("Sheet %d, ", i / 2 + 1);

if (i & 1)

printf("back :");

else

printf("front:");

if (f[i][0] <= n)

printf(" %d", f[i][0]);

else

printf(" Blank");

if (f[i][1] <= n)

printf(", %d", f[i][1]);

else

printf(", Blank");

puts("");

}

}

int main()

{

while (scanf("%d", &n), n)

{

printf("Printing order for %d pages:\n", n);

work();

output();

}

return 0;

}

## poj 1628 Deduction

S is a set of 52 declarations, which is represented by lower case letters ('a', 'b', ..., 'z') and upper case letters ('A', 'B', ..., 'Z'). These declarations are not completely independent with each other, And from some of them, we can deduce some others. We denote this relation as "=>". For instance, a => b means from declaration a, we can deduce declaration b; b => c, means from declaration b, we can deduce declaration c. And from a => b and b =>c, we can also get a => bc, which means from a, we can deduce both b and c. Let's see some further examples: abc => de means from a, b, and c, we can deduce d and e. bc => fgh means from bc, we can deduce fgh. And from abc => de, bc => fhg, we can finally get abc => defgh, even more, we can get abc => abcdefgh.

Now given m deduction relations, which are represented in the form S1 => S2, S1 and S2 are subsets of S, and n queries, each contains a subset Q of S. Your job is to find all declarations can be deduced from Q according to the given relations.

输入描述

Input will contain m + n + 1 lines. The first line contains two positive integers m, n. (m is less than 200, and n is less than 1000). Then m lines follow, each with a deduction relation. The last n lines each contains a query.

You can see that there is no space between declarations in S1, S2 and Q. And there is one space between S1 and "=>" and between "=>" and S2. And it is confirmed that the number of declarations in S1, S2 and Q will not exceed 52.

输出描述

For each query, output the answer for the query. The element for the output should be in the order [a, b, c, ..., y, z, A, B, C, ..., Y, Z].

输入例子

2 1

abc => de

bcb => FGh

abc

输出例子

abcdehFG

参考答案

#include<iostream>

using namespace std;

struct Node {

char s1[200],s2[200];

}decl[250];

bool more,ur[128],visit[250];

int n,m,l,k ;

char s[200],res[200];

int main()

{

while(scanf("%d%d",&n,&m)!=EOF) {

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

scanf("%s => %s",&decl[i].s1,&decl[i].s2);

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

scanf("%s",&s);

l = strlen(s) ;

memset(ur,false,sizeof(ur));

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

ur[s[j]] = true ;

more = true ;

memset(visit,false,sizeof(visit));

while(more) {

more = false ;

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

if(visit[j]) continue ;

l = strlen(decl[j].s1);

for(k=0;k<l;k++)

if(!ur[decl[j].s1[k]]) break ;

if(k==l) {

visit[j] = true ;

more = true ;

l = strlen(decl[j].s2);

for(k=0;k<l;k++)

ur[decl[j].s2[k]] = true ;

}

}

}

for(char c='a';c<='z';c++)

if(ur[c]) printf("%c",c);

for(char c='A';c<='Z';c++)

if(ur[c]) printf("%c",c);

puts("");

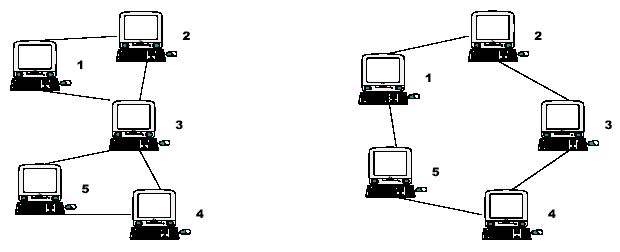
}

}

return 0;

}

## poj 1523 SPF

Consider the two networks shown below. Assuming that data moves around these networks only between directly connected nodes on a peer-to-peer basis, a failure of a single node, 3, in the network on the left would prevent some of the still available nodes from communicating with each other. Nodes 1 and 2 could still communicate with each other as could nodes 4 and 5, but communication between any other pairs of nodes would no longer be possible.   
  
Node 3 is therefore a Single Point of Failure (SPF) for this network. Strictly, an SPF will be defined as any node that, if unavailable, would prevent at least one pair of available nodes from being able to communicate on what was previously a fully connected network. Note that the network on the right has no such node; there is no SPF in the network. At least two machines must fail before there are any pairs of available nodes which cannot communicate.   
<center> </center>

输入描述

The input will contain the description of several networks. A network description will consist of pairs of integers, one pair per line, that identify connected nodes. Ordering of the pairs is irrelevant; 1 2 and 2 1 specify the same connection. All node numbers will range from 1 to 1000. A line containing a single zero ends the list of connected nodes. An empty network description flags the end of the input. Blank lines in the input file should be ignored.

输出描述

For each network in the input, you will output its number in the file, followed by a list of any SPF nodes that exist.   
  
The first network in the file should be identified as "Network #1", the second as "Network #2", etc. For each SPF node, output a line, formatted as shown in the examples below, that identifies the node and the number of fully connected subnets that remain when that node fails. If the network has no SPF nodes, simply output the text "No SPF nodes" instead of a list of SPF nodes.

输入例子

1 2

5 4

3 1

3 2

3 4

3 5

0

1 2

2 3

3 4

4 5

5 1

0

1 2

2 3

3 4

4 6

6 3

2 5

5 1

0

0

输出例子

Network #1

SPF node 3 leaves 2 subnets

Network #2

No SPF nodes

Network #3

SPF node 2 leaves 2 subnets

SPF node 3 leaves 2 subnets

代码：

#include<iostream>

#include<string.h>

using namespace std;

const int size=1001;

struct Node

{

int id; //结点编号

struct Node\* next;

};

class solve

{

public:

solve():cases(0)

{

for(int i=0;i<size;i++) //初始化邻接链表链头

ListTable\_Head[i]=0;

DFS();

}

~solve()

{

delete[] \*ListTable\_Head;

}

int min(int a,int b) const{return a<b?a:b;}

bool Input(void); //返回0: 等待下一组输入。 返回1:程序结束

void Insert(int x,int y); //把无向边(x,y)插入邻接表

void DFS(void); //搜索割点及其对应的连通分量数

void DFS\_SPF(int k,int father,int depth); //搜索割点。k:当前结点。father:k的父亲结点。depth:搜索深度(时间戳)

void DFS\_Subnet(int i); //i为从割点出发的分支,搜索并标记该从该分支出发所能到达的所有node

//（这些node就是一个连通分量）

void Empty(void); //清空邻接表(保留链头并使其指向NULL)

void Del(Node\* p); //删除以结点p为链头的整条链(保留链头并使其指向NULL)

protected:

int cases; //案例数

Node\* ListTable\_Head[size]; //邻接链表链头

int Root; //深搜树的根(搜索起点)

int deep[size]; //结点k的搜索深度deep[k]

int AnceDeep[size]; //结点k及其子孙辈分最高(深度最浅)的祖先的深度AnceDeep[k]

int color[size]; //结点k的颜色color[k]。0:未访问未检查；1:已访问未检查；2:已访问已检查

bool cut[size]; //标记结点是否为割点

bool SPF; //标记网络是否出现割点

int Subnet\_Num; //被割点的划分的连通分量的个数

bool vist[size]; //标记访问过的node

};

bool solve::Input(void)

{

int x,y;

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

{

cin>>x;

if(x==0 && i==0) //程序结束

break;

else if(x==0 && i!=0) //当前case的数据输入完毕

return true;

cin>>y;

Insert(x,y);

}

return false;

}

void solve::Insert(int x,int y)

{

if(!ListTable\_Head[x]) //链头不存在，则创建

{

ListTable\_Head[x]=new Node;

ListTable\_Head[x]->next=0;

}

if(!ListTable\_Head[y])

{

ListTable\_Head[y]=new Node;

ListTable\_Head[y]->next=0;

}

Node\* px=ListTable\_Head[x];

Node\* py=ListTable\_Head[y];

Node\* tmp;

tmp=px->next; //头插入法

px->next=new Node;

px->next->id=y;

px->next->next=tmp;

tmp=py->next; //头插入法

py->next=new Node;

py->next->id=x;

py->next->next=tmp;

return;

}

void solve::DFS(void)

{

while(Input())

{

SPF=false;

memset(deep,0,sizeof(deep));

memset(AnceDeep,0,sizeof(AnceDeep));

memset(color,0,sizeof(color));

memset(cut,false,sizeof(cut));

/\*搜索编号最小的node作为深搜树树的根\*/

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

if(ListTable\_Head[k])

{

Root=k;

break;

}

/\*寻找所有割点\*/

DFS\_SPF(Root,Root,1);

cout<<"Network #"<<++cases<<endl;

if(!SPF)

cout<<" No SPF nodes"<<endl;

else

{

for(int i=Root;i<size;i++)

if(cut[i])

{

Subnet\_Num=0;

memset(vist,false,sizeof(vist));

vist[i]=true;

/\*枚举割点i的所有分支，其中没有访问过的分支则对其逐一深搜\*/

/\*找出被割点i划分的所有连通分量\*/

for(Node\* p=ListTable\_Head[i]->next;p;p=p->next)

{

if(!vist[p->id]) //分支p->id没有被访问

{ //说明当前分支与前面搜索的连通分量 不连通

Subnet\_Num++; //则连通分量数+1

DFS\_Subnet(p->id); //搜索并标记该连通分量下的所有node

}

}

cout<<" SPF node "<<i<<" leaves "<<Subnet\_Num<<" subnets"<<endl;

}

}

cout<<endl;

Empty();

}

return;

}

void solve::DFS\_SPF(int k,int father,int depth)

{

color[k]=1; //染色，结点k已访问未检查

deep[k]=depth; //记录k的搜索深度

AnceDeep[k]=depth; //初始化,k最浅的祖先的深度就是k本身的深度

int son=0; //k的儿子数

for(Node\* p=ListTable\_Head[k]->next;p;p=p->next)

{

int i=p->id;

if(color[i]==0) //未访问未检查的node

{

son++; //k的分支中，所有未访问未检查的node都是其儿子

DFS\_SPF(i,k,depth+1);

AnceDeep[k]=min(AnceDeep[k],AnceDeep[i]); //由于k和其儿子i必然同宗

} //若在i中出现后向边使得i的祖先辈分更高（深度更浅）

//那么k的祖先辈分应该被更新

if(i!=father && color[i]==1) //k的分支中，所有已访问未检查的node都是其祖先

AnceDeep[k]=min(AnceDeep[k],deep[i]); //直接检查祖先辈分（深度）并更新

if((k==Root && son>1) || /\*根结点的儿子数(分支数) >1时，则Root必定是割点\*/

(k!=Root && AnceDeep[i]>=deep[k])) /\*k的儿子i的最大祖先的深度比k的深度要深，则删除k后i与k的father必然断开\*/

{

cut[k]=true;

SPF=true;

}

}

color[k]=2;

return;

}

void solve::DFS\_Subnet(int cp)

{

for(Node\* p=ListTable\_Head[cp]->next;p;p=p->next)

{

int i=p->id;

if(!vist[i])

{

vist[i]=true;

DFS\_Subnet(i);

}

}

return;

}

void solve::Empty(void)

{

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

{

if(ListTable\_Head[i])

Del(ListTable\_Head[i]);

ListTable\_Head[i]=0; //保留链表表头并初始化

}

return;

}

void solve::Del(Node\* p)

{

if(p->next)

Del(p->next);

delete p;

return;

}

int main(void)

{

solve poj1523;

return 0;

}

## poj 1628 Deduction

S is a set of 52 declarations, which is represented by lower case letters ('a', 'b', ..., 'z') and upper case letters ('A', 'B', ..., 'Z'). These declarations are not completely independent with each other, And from some of them, we can deduce some others. We denote this relation as "=>". For instance, a => b means from declaration a, we can deduce declaration b; b => c, means from declaration b, we can deduce declaration c. And from a => b and b =>c, we can also get a => bc, which means from a, we can deduce both b and c. Let's see some further examples: abc => de means from a, b, and c, we can deduce d and e. bc => fgh means from bc, we can deduce fgh. And from abc => de, bc => fhg, we can finally get abc => defgh, even more, we can get abc => abcdefgh.

Now given m deduction relations, which are represented in the form S1 => S2, S1 and S2 are subsets of S, and n queries, each contains a subset Q of S. Your job is to find all declarations can be deduced from Q according to the given relations.

输入描述

Input will contain m + n + 1 lines. The first line contains two positive integers m, n. (m is less than 200, and n is less than 1000). Then m lines follow, each with a deduction relation. The last n lines each contains a query.

You can see that there is no space between declarations in S1, S2 and Q. And there is one space between S1 and "=>" and between "=>" and S2. And it is confirmed that the number of declarations in S1, S2 and Q will not exceed 52.

输出描述

For each query, output the answer for the query. The element for the output should be in the order [a, b, c, ..., y, z, A, B, C, ..., Y, Z].

输入例子

2 1

abc => de

bcb => FGh

abc

输出例子

abcdehFG

代码：

#include<iostream>

using namespace std;

struct Node {

char s1[200],s2[200];

}decl[250];

bool more,ur[128],visit[250];

int n,m,l,k ;

char s[200],res[200];

int main()

{

while(scanf("%d%d",&n,&m)!=EOF) {

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

scanf("%s => %s",&decl[i].s1,&decl[i].s2);

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

scanf("%s",&s);

l = strlen(s) ;

memset(ur,false,sizeof(ur));

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

ur[s[j]] = true ;

more = true ;

memset(visit,false,sizeof(visit));

while(more) {

more = false ;

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

if(visit[j]) continue ;

l = strlen(decl[j].s1);

for(k=0;k<l;k++)

if(!ur[decl[j].s1[k]]) break ;

if(k==l) {

visit[j] = true ;

more = true ;

l = strlen(decl[j].s2);

for(k=0;k<l;k++)

ur[decl[j].s2[k]] = true ;

}

}

}

for(char c='a';c<='z';c++)

if(ur[c]) printf("%c",c);

for(char c='A';c<='Z';c++)

if(ur[c]) printf("%c",c);

puts("");

}

}

return 0;

}

## poj 1664 放苹果

把M个同样的苹果放在N个同样的盘子里，允许有的盘子空着不放，问共有多少种不同的分法？（用K表示）5，1，1和1，5，1 是同一种分法。

输入描述

第一行是测试数据

输出描述

对输入的每组数据M和N，用一行输出相应的K。

输入例子

1

7 3

输出例子

8

代码：

#include<stdio.h>

int fun(int m,int n) //m个苹果放在n个盘子中共有几种方法

{

if(m==0||n==1) //因为我们总是让m>=n来求解的，所以m-n>=0,所以让m=0时候结束，如果改为m=1，

return 1; //则可能出现m-n=0的情况从而不能得到正确解

if(n>m)

return fun(m,m);

else

return fun(m,n-1)+fun(m-n,n);

}

int main()

{

int T,m,n;

scanf("%d",&T);

while(T--)

{

scanf("%d%d",&m,&n);

printf("%d\n",fun(m,n));

}

}

## poj 1953 World Cup Noise

Background

"KO-RE-A, KO-RE-A" shout 54.000 happy football fans after their team has reached the semifinals of the FIFA World Cup in their home country. But although their excitement is real, the Korean people are still very organized by nature. For example, they have organized huge trumpets (that sound like blowing a ship's horn) to support their team playing on the field. The fans want to keep the level of noise constant throughout the match.

The trumpets are operated by compressed gas. However, if you blow the trumpet for 2 seconds without stopping it will break. So when the trumpet makes noise, everything is okay, but in a pause of the trumpet,the fans must chant "KO-RE-A"!

Before the match, a group of fans gathers and decides on a chanting pattern. The pattern is a sequence of 0's and 1's which is interpreted in the following way: If the pattern shows a 1, the trumpet is blown. If it shows a 0, the fans chant "KO-RE-A". To ensure that the trumpet will not break, the pattern is not allowed to have two consecutive 1's in it.

Problem

Given a positive integer n, determine the number of different chanting patterns of this length, i.e., determine the number of n-bit sequences that contain no adjacent 1's. For example, for n = 3 the answer is 5 (sequences 000, 001, 010, 100, 101 are acceptable while 011, 110, 111 are not).

输入描述

The first line contains the number of scenarios.

For each scenario, you are given a single positive integer less than 45 on a line by itself.

输出描述

The output for every scenario begins with a line containing "Scenario #i:", where i is the number of the scenario starting at 1. Then print a single line containing the number of n-bit sequences which have no adjacent 1's. Terminate the output for the scenario with a blank line.

输入例子

2

3

1

输出例子

Scenario #1:

5

Scenario #2:

2

参考答案

#include <iostream>

#include <cstdio>

#include <cstdlib>

#include <cstring>

using namespace std;

#define maxn 45

int f[maxn];

int main()

{

f[1] = 2;

f[2] = 3;

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

f[i] = f[i - 1] + f[i - 2];

int t;

scanf("%d", &t);

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

{

int a;

scanf("%d", &a);

printf("Scenario #%d:\n%d\n\n", i, f[a]);

}

return 0;

}

## poj 1977 Odd Loving Bakers

There is a group of N bakers in the town of Utopia. These bakers hold a monthly celebration in which they award a prize to some of the luckier among themselves. These lucky guys are chosen as follows:

In the beginning there are some chalk marks on some of the bakers' houses. Each baker has a list of his/her favorite bakers. After each celebration each of the winners puts a chalk mark on the house of all the bakers in his/her favorite list. Before each celebration those bakers with an odd number of chalk marks on their house will be chosen as winners. As there will be a great prize for the winners of the tth celebration, you are asked to find the number of its winners.

输入描述

The first line of the input contains a single integer X (1 The first line of each instance contains two integers n (the number of bakers) and t (the number of the celebration we want the winners of).

1 1 The next n lines of the instance each describe a baker. In each of these lines first comes the name of the baker (names are lower case strings of no more than 20 characters without spaces), then comes the number of chalk marks initially on the baker's house, then comes the number of bakers in this baker's favorite list, and after that come the names of the bakers in this baker's list.

输出描述

There should be one line per test case. For each test case write a line containing a single integer, the number of the winners of the t-th celebration.

输入例子

2

3 2

bessie 2 3 bessie linda mary

mary 1 1 linda

linda 0 1 bessie

2 2

siavosh 1 2 siavosh mohammad

mohammad 1 0

输出例子

2

0

代码：

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define MAXN 110

typedef int M[MAXN][MAXN];

struct person

{

char name[25],fav[MAXN][25];

int nfav;

int ori;

};

person P[MAXN];

int n,baker[MAXN],matrix[MAXN][MAXN],O[MAXN][MAXN],cse,t,res;

void copy(M x,M y)

{

int i,j;

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

{

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

{

x[i][j]=y[i][j];

}

}

return ;

}

void mu(M x,M y)

{

int i,j,k,t;

M c;

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

{

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

{

t=0;

for(k=0;k<n;k++)

{

if(x[i][k] && y[k][j])

{

t=(t+x[i][k]\*y[k][j])%2;

}

}

c[i][j]=t;

}

}

copy(x,c);

return ;

}

void Cal(M a,int k)

{

if(k==1)

{

copy(a,O);

return ;

}

Cal(a,k/2);

mu(a,a);

if(k & 1)

{

mu(a,O);

}

}

int main()

{

int i,j,k;

scanf("%d",&cse);

while(cse--)

{

scanf("%d %d",&n,&t);

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

{

getchar();

scanf("%s",P[i].name);

scanf("%d %d",&P[i].ori,&P[i].nfav);

baker[i]=P[i].ori%2;

for(j=0;j<P[i].nfav;j++)

{

getchar();

scanf("%s",&P[i].fav[j]);

}

}

memset(O,0,sizeof(O));

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

{

O[i][i]=1;

}

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

{

for(j=0;j<P[i].nfav;j++)

{

for(k=0;k<n;k++)

{

if(strcmp(P[i].fav[j],P[k].name)==0)

{

break;

}

}

O[i][k]=(O[i][k]+1)%2;

}

}

Cal(matrix,t-1);

res=0;

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

{

int tmp=0 ;

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

{

tmp=(tmp+baker[j]\*matrix[j][i])%2;

}

if(tmp)res++;

}

printf("%d\n",res);

}

return 0;

}

## poj 1989 The Cow Lineup

Farmer John's N cows (1

1 5 3 2 5 1 3 4 4 2 5 1 2 3

Farmer John's acute mathematical mind notices all sorts of properties of number sequences like that above. For instance, he notices that the sequence 3 4 1 3 is a subsequence (not necessarily contiguous) of the sequence of breed IDs above. FJ is curious what is the length of the shortest possible sequence he can construct out of numbers in the range 1..K that is NOT a subsequence of the breed IDs of his cows. Help him solve this problem.

输入描述

\* Line 1: Two integers, N and K

\* Lines 2..N+1: Each line contains a single integer that is the breed ID of a cow. Line 2 describes cow 1; line 3 describes cow 2; and so on.

输出描述

\* Line 1: The length of the shortest sequence that is not a subsequence of the input

输入例子

14 5

1

5

3

2

5

1

3

4

4

2

5

1

2

3

输出例子

3

Hint

All the single digit 'sequences' appear. Each of the 25 two digit sequences also appears. Of the three digit sequences, the sequence 2, 2, 4 does not appear.

代码：

#include<stdio.h>

#include<string.h>

#define clr(x)memset(x,0,sizeof(x))

int v[10005];

int main()

{

int n,k,p,i,res,num;

while(scanf("%d%d",&n,&k)!=EOF)

{

res=num=0;

clr(v);

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

{

scanf("%d",&p);

if(!v[p])

{

v[p]=1;

num++;

if(num==k)

{

res++;

clr(v);

num=0;

}

}

}

printf("%d\n",res+1);

}

return 0;

}

## poj 2323 PERMS

Count the number of permutations that have a specific number of inversions.

Given a permutation a1, a2, a3,..., an of the n integers 1, 2, 3, ..., n, an inversion is a pair (ai, aj) where i aj. The number of inversions in a permutation gives an indication on how "unsorted" a permutation is. If we wish to analyze the average running time of a sorting algorithm, it is often useful to know how many permutations of n objects will have a certain number of inversions.

In this problem you are asked to compute the number of permutations of n values that have exactly k inversions.

For example, if n = 3, there are 6 permutations with the indicated inversions as follows:

123 0 inversions

132 1 inversion (3 > 2)

213 1 inversion (2 > 1)

231 2 inversions (2 > 1, 3 > 1)

312 2 inversions (3 > 1, 3 > 2)

321 3 inversions (3 > 2, 3 > 1, 2 > 1)

Therefore, for the permutations of 3 things

1 of them has 0 inversions

2 of them have 1 inversion

2 of them have 2 inversions

1 of them has 3 inversions

0 of them have 4 inversions

0 of them have 5 inversions

etc.

输入描述

The input consists one or more problems. The input for each problem is specified on a single line, giving the intege

输出描述

For each problem, output the number of permutations of {1, ..., n}with exactly k inversions.

输入例子

3 0

3 1

3 2

3 3

4 2

4 10

13 23

18 80

0 0

输出例子

1

2

2

1

5

0

46936280

184348859235088

参考答案

#include<iostream>

using namespace std;

int main()

{

long long dp[19][200]= {0};

dp[1][0]=1;

for(int i=1; i<18; i++)

for(int j=0; dp[i][j]; j++)

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

dp[i+1][j+k]+=dp[i][j];

int x,y;

while(cin>>x>>y,x)

cout<<dp[x][y]<<endl;

}

## poj 2386 Lake Counting

Due to recent rains, water has pooled in various places in Farmer John's field, which is represented by a rectangle of N x M (1

Given a diagram of Farmer John's field, determine how many ponds he has.

输入描述

\* Line 1: Two space-separated integers: N and M

\* Lines 2..N+1: M characters per line representing one row of Farmer John's field. Each character is either 'W' or '.'. The characters do not have spaces between them.

输出描述

\* Line 1: The number of ponds in Farmer John's field.

输入例子

10 12

W........WW.

.WWW.....WWW

....WW...WW.

.........WW.

.........W..

..W......W..

.W.W.....WW.

W.W.W.....W.

.W.W......W.

..W.......W.

输出例子

3

Hint

OUTPUT DETAILS:

There are three ponds: one in the upper left, one in the lower left,and one along the right side.

代码：

#include<iostream>

#include<queue>

using namespace std;

struct Node

{

int x;

int y;

};

int dir[8][2]={{0,1},{0,-1},{1,0},{-1,0},{1,-1},{-1,1},{1,1},{-1,-1}};

char map[105][105];

int num;

int n,m;

void BFS(int x,int y)

{

queue<Node>q;

int tx,ty;

int i,j;

Node t,p;

t.x=x;

t.y=y;

q.push(t);

while(!q.empty())

{

p=q.front();

q.pop();

for(i=0;i<8;i++)

{

tx=p.x+dir[i][0];

ty=p.y+dir[i][1];

if(tx>=0&&tx<n&&ty>=0&&ty<m&&map[tx][ty]=='W')

{

t.x=tx;

t.y=ty;

q.push(t);

map[tx][ty]='.';

}

}

}

}

int main()

{

int i,j;

while(scanf("%d%d",&n,&m)!=EOF)

{

num=0;

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

{

scanf("%s",map[i]);

}

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

{

for(j=0;j<m;j++)

{

if(map[i][j]=='W')

{

num++;

BFS(i,j);

}

}

}

printf("%d\n",num);

}

return 0;

}

## poj 2417 Discrete Logging

Given a prime P, 2 31, an integer B, 2

BL == N (mod P)

输入描述

Read several lines of input, each containing P,B,N separated by a space.

输出描述

For each line print the logarithm on a separate line. If there are several, print the smallest; if there is none, print "no solution".

输入例子

5 2 1

5 2 2

5 2 3

5 2 4

5 3 1

5 3 2

5 3 3

5 3 4

5 4 1

5 4 2

5 4 3

5 4 4

12345701 2 1111111

1111111121 65537 1111111111

输出例子

0

1

3

2

0

3

1

2

0

no solution

no solution

1

9584351

462803587

Hint

The solution to this problem requires a well known result in number theory that is probably expected of you for Putnam but not ACM competitions. It is Fermat's theorem that states

B(P-1) == 1 (mod P)

for any prime P and some other (fairly rare) numbers known as base-B pseudoprimes. A rarer subset of the base-B pseudoprimes, known as Carmichael numbers, are pseudoprimes for every base between 2 and P-1. A corollary to Fermat's theorem is that for any m

B(-m) == B(P-1-m) (mod P) .

代码：

#include<iostream>

#include<cstdio>

#include<algorithm>

#include<cstring>

#include<cmath>

#include<vector>

#include<queue>

#include<map>

#include<set>

#include<time.h>

#include<string>

#define cl(a,b) memset(a,b,sizeof(a))

#define max(x,y) ((x)>(y)?(x):(y))

#define min(x,y) ((x)<(y)?(x):(y))

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

#define REP1(i,a,b) for(int i=a;i<=b;++i)

#define REP2(i,a,b) for(int i=a;i>=b;--i)

#define MP make\_pair

#define LL long long

#define ULL unsigned long long

#define X first

#define Y second

#define MAXN 100050

using namespace std;

int p,b,n;

struct node

{

int a,id;

}q[MAXN];

bool cmp(node a,node b)

{

return a.a<b.a||(a.a==b.a&&a.id<b.id);

}

int qmod(int a,int b,int c)

{

int res=1;

while(b)

{

if(b&1)

res=((LL)res\*a)%c;

a=((LL)a\*a)%c;

b>>=1;

}

return res;

}

int main()

{

while(scanf("%d%d%d",&p,&b,&n)!=EOF){

int m=sqrt(p\*1.0);

while(m\*m<p)m++;

int res=1;

int tail=0;

int a=qmod(b,m,p);

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

{

q[tail].id=i;

q[tail++].a=res;

res=((LL)res\*b)%p;

}

sort(q,q+tail,cmp);

int h=0;

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

{

if(i==0||q[i].a!=q[i-1].a)

{

q[h++]=q[i];

}

}

int flag=0,ans;

res=n;

int ra=qmod(a,p-2,p);

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

{

int l=0,r=h-1;

while(r-l>1)

{

int mid=(l+r)>>1;

if(q[mid].a>res)r=mid;

else l=mid;

}

int pos=-1;

if(q[l].a==res)pos=l;

if(q[r].a==res)pos=r;

if(pos!=-1){

ans=i\*m+q[pos].id;

flag=1;

break;

}

res=((LL)res\*ra)%p;

}

if(flag==0)puts("no solution");

else

printf("%d\n",ans);

}

}

## poj 2447 RSA

RSA is the best-known public key encryption algorithm. In this algorithm each participant has a private key that is shared with no one else and a public key which is published so everyone knows it. To send a secure message to this participant, you encrypt the message using the widely known public key; the participant then decrypts the messages using his or her private key. Here is the procedure of RSA:

First, choose two different large prime numbers P and Q, and multiply them to get N (= P \* Q).

Second, select a positive integer E (0 Third, compute the decryption key D such that 0

Now the public key is constructed by the pair {E, N}, and the private key is {D, N}. P and Q can be discarded.

Encryption is defined by C = (M ^ E) mod N, and decryption is defined by M = (C ^ D) mod N, here M, which is a non-negative integer and smaller than N, is the plaintext message and C is the resulting ciphertext.

To illustrate this idea, let’s see the following example:

We choose P = 37, Q = 23, So N = P \* Q = 851, and T = 792. If we choose E = 5, D will be 317 ((5 \* 317) mod 792 = 1). So the public key is {5, 851}, and the private key is {317, 851}. For a given plaintext M = 7, we can get the ciphertext C = (7 ^ 5) mod 851 = 638.

As we have known,for properly choosen very large P and Q, it will take thousands of years to break a key, but for small ones, it is another matter.

Now you are given the ciphertext C and public key {E, N}, can you find the plaintext M?

输入描述

The input will contain several test cases. Each test case contains three positive integers C, E

输出描述

Output the plaintext M in a single line.

输入例子

638 5 851

输出例子

7

代码：

#include <cstdio>

#include <cstdlib>

#include <ctime>

using namespace std;

typedef long long LL;

#define maxn 10000

const int S=20;

LL factor[maxn];

int tot;

LL muti\_mod(LL a,LL b,LL c){

a%=c;

b%=c;

LL ret=0;

while (b){

if (b&1){

ret+=a;

if (ret>=c) ret-=c;

}

a<<=1;

if (a>=c) a-=c;

b>>=1;

}

return ret;

}

LL pow\_mod(LL x,LL n,LL mod){

if (n==1) return x%mod;

int bit[64],k=0;

while (n){

bit[k++]=n&1;

n>>=1;

}

LL ret=1;

for (k=k-1;k>=0;k--){

ret=muti\_mod(ret,ret,mod);

if (bit[k]==1) ret=muti\_mod(ret,x,mod);

}

return ret;

}

bool check(LL a,LL n,LL x,LL t){

LL ret=pow\_mod(a,x,n),last=ret;

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

ret=muti\_mod(ret,ret,n);

if (ret==1&& last!=1&& last!=n-1) return 1;

last=ret;

}

if (ret!=1) return 1;

return 0;

}

bool Miller\_Rabin(LL n){

LL x=n-1,t=0;

while ((x&1)==0) x>>=1,t++;

bool flag=1;

if (t>=1&& (x&1)==1){

for (int k=0;k<S;k++){

LL a=rand()%(n-1)+1;

if (check(a,n,x,t)) {flag=1;break;}

flag=0;

}

}

if (!flag || n==2) return 0;

return 1;

}

LL gcd(LL a,LL b){

if (a==0) return 1;

if (a<0) return gcd(-a,b);

while (b){

LL t=a%b; a=b; b=t;

}

return a;

}

LL Pollard\_rho(LL x,LL c){

LL i=1,x0=rand()%x,y=x0,k=2;

while (1){

i++;

x0=(muti\_mod(x0,x0,x)+c)%x;

LL d=gcd(y-x0,x);

if (d!=1&& d!=x){

return d;

}

if (y==x0) return x;

if (i==k){

y=x0;

k+=k;

}

}

}

void findfac(LL n){

if (!Miller\_Rabin(n)){

factor[tot++] = n;

return;

}

LL p=n;

while (p>=n) p=Pollard\_rho(p,rand() % (n-1) +1);

findfac(p);

findfac(n/p);

}

void gcdExtend(long long a,long long b,long long &d,long long &x,long long &y)

{

if(!b) {d=a;x=1;y=0;return;}

gcdExtend(b,a%b,d,y,x);

y-=a/b\*x;

}

int main()

{

LL C, E, N, T, M, D;

LL x, y, d;

while (~scanf("%lld%lld%lld", &C, &E, &N))

{

tot = 0;

findfac(N);

T = (factor[0] - 1) \* (factor[1] - 1);

gcdExtend(E, T, d, x, y);

D = (x % T + T) % T;

M = pow\_mod(C, D, N);

printf("%lld\n", M);

}

return 0;

}

## poj 2452 Sticks Problem

Xuanxuan has n sticks of different length. One day, she puts all her sticks in a line, represented by S1, S2, S3, ...Sn. After measuring the length of each stick Sk (1

Now given the length of S1, S2, S3, …Sn, you are required to find the maximum value j - i.

输入描述

The input contains multiple test cases. Each case contains two lines.

Line 1: a single integer n (n Line 2: n different positive integers (not larger than 100000), indicating the length of each stick in order.

输出描述

Output the maximum value j - i in a single line. If there is no such i and j, just output -1.

输入例子

4

5 4 3 6

4

6 5 4 3

输出例子

1

-1

代码：

#include <cstdio>

#include <cstring>

#include <algorithm>

#define MX 55555

#define lson l,m,n<<1

#define rson m+1,r,n<<1|1

#define lc n<<1

#define rc n<<1|1

using namespace std;

int num[MX], ma[MX << 2], mi[MX << 2];

int L, R, N, ans;

inline void up(int n)

{

ma[n] = num[ma[lc]] > num[ma[rc]] ? ma[lc] : ma[rc];

mi[n] = num[mi[lc]] < num[mi[rc]] ? mi[lc] : mi[rc];

}

void B(int l = 1, int r = N, int n = 1)

{

if (l == r)

{

scanf("%d", &num[l]);

ma[n] = mi[n] = l;

return;

}

int m = l + r >> 1;

B(lson), B(rson), up(n);

}

int QA(int l = 1, int r = N, int n = 1)

{

if (L <= l && r <= R)

return ma[n];

int ans = 0, m = l + r >> 1, t;

if (L <= m)

t = QA(lson), ans = (num[ans] > num[t]) ? ans : t;

if (m < R)

t = QA(rson), ans = (num[ans] > num[t]) ? ans : t;

return ans;

}

int QB(int l = 1, int r = N, int n = 1)

{

if (L <= l && r <= R)

return mi[n];

int ans = N + 1, m = l + r >> 1, t;

if (L <= m)

t = QB(lson), ans = (num[ans] < num[t]) ? ans : t;

if (m < R)

t = QB(rson), ans = (num[ans] < num[t]) ? ans : t;

return ans;

}

int S(int l, int r)

{

if (l >= r) return -1;

L = l, R = r;

int ans = -1, a = QA(), b = QB();

if (a > b)

ans = max(a - b, max(S(l, b), S(a, r)));

else ans = max(S(a + 1, b - 1), max(S(l, a), S(b, r)));

return ans;

}

int main()

{

while (~scanf("%d", &N))

{

B(), num[N + 1] = 111111;

printf("%d\n", S(1, N));

}

return 0;

}

## poj 2482 Stars in Your Window

Fleeting time does not blur my memory of you. Can it really be 4 years since I first saw you? I still remember, vividly, on the beautiful Zhuhai Campus, 4 years ago, from the moment I saw you smile, as you were walking out of the classroom and turned your head back, with the soft sunset glow shining on your rosy cheek, I knew, I knew that I was already drunk on you. Then, after several months’ observation and prying, your grace and your wisdom, your attitude to life and your aspiration for future were all strongly impressed on my memory. You were the glamorous and sunny girl whom I always dream of to share the rest of my life with. Alas, actually you were far beyond my wildest dreams and I had no idea about how to bridge that gulf between you and me. So I schemed nothing but to wait, to wait for an appropriate opportunity. Till now — the arrival of graduation, I realize I am such an idiot that one should create the opportunity and seize it instead of just waiting.

These days, having parted with friends, roommates and classmates one after another, I still cannot believe the fact that after waving hands, these familiar faces will soon vanish from our life and become no more than a memory. I will move out from school tomorrow. And you are planning to fly far far away, to pursue your future and fulfill your dreams. Perhaps we will not meet each other any more if without fate and luck. So tonight, I was wandering around your dormitory building hoping to meet you there by chance. But contradictorily, your appearance must quicken my heartbeat and my clumsy tongue might be not able to belch out a word. I cannot remember how many times I have passed your dormitory building both in Zhuhai and Guangzhou, and each time aspired to see you appear in the balcony or your silhouette that cast on the window. I cannot remember how many times this idea comes to my mind: call her out to have dinner or at least a conversation. But each time, thinking of your excellence and my commonness, the predominance of timidity over courage drove me leave silently.

Graduation, means the end of life in university, the end of these glorious, romantic years. Your lovely smile which is my original incentive to work hard and this unrequited love will be both sealed as a memory in the deep of my heart and my mind. Graduation, also means a start of new life, a footprint on the way to bright prospect. I truly hope you will be happy everyday abroad and everything goes well. Meanwhile, I will try to get out from puerility and become more sophisticated. To pursue my own love and happiness here in reality will be my ideal I never desert.

Farewell, my princess!

If someday, somewhere, we have a chance to gather, even as gray-haired man and woman, at that time, I hope we can be good friends to share this memory proudly to relight the youthful and joyful emotions. If this chance never comes, I wish I were the stars in the sky and twinkling in your window, to bless you far away, as friends, to accompany you every night, sharing the sweet dreams or going through the nightmares together.

<center> </center>

Here comes the problem: Assume the sky is a flat plane. All the stars lie on it with a location (x, y). for each star, there is a grade ranging from 1 to 100, representing its brightness, where 100 is the brightest and 1 is the weakest. The window is a rectangle whose edges are parallel to the x-axis or y-axis. Your task is to tell where I should put the window in order to maximize the sum of the brightness of the stars within the window. Note, the stars which are right on the edge of the window does not count. The window can be translated but rotation is not allowed.

输入描述

There are several test cases in the input. The first line of each case contains 3 integers: n, W, H, indicating the number of stars, the horizontal length and the vertical height of the rectangle-shaped window. Then n lines follow, with 3 integers each: x, y, c, telling the location (x, y) and the brightness of each star. No two stars are on the same point.

There are at least 1 and at most 10000 stars in the sky. 1

输出描述

For each test case, output the maximum brightness in a single line.

输入例子

3 5 4

1 2 3

2 3 2

6 3 1

3 5 4

1 2 3

2 3 2

5 3 1

输出例子

5

6

代码：

#include<iostream>

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<algorithm>

using namespace std;

const int N=40005;

long long r[N\*12];

int n;

long long w,h;

int tot,cnt;

long long mx=0;

long long maxl(long long a,long long b)

{return a>b?a:b;}

struct line

{

long long x,y1,y2,c;

}pos[N\*6];

bool cmp(line a,line b)

{

if(a.x==b.x)return a.c<b.c;

return a.x<b.x;

}

struct sgtr

{

int l,r;

long long dat;

long long add;

}tr[N\*6];

void build(int p,int l,int r)

{

tr[p].l=l,tr[p].r=r,tr[p].dat=tr[p].add=0;

if(l==r)return;

int mid=(l+r)/2;

build(p\*2,l,mid);

build(p\*2+1,mid+1,r);

}

void spread(int p)

{

if(tr[p].l==tr[p].r)return;

if(tr[p].add)

{

tr[p\*2].dat+=tr[p].add;

tr[p\*2+1].dat+=tr[p].add;

tr[p\*2].add+=tr[p].add;

tr[p\*2+1].add+=tr[p].add;

}

tr[p].add=0;

}

void change(int p,int l,int r,long long x)

{

if(l<=tr[p].l&&r>=tr[p].r)

{

tr[p].dat+=x;

tr[p].add+=x;

return;

}

spread(p);

int mid=(tr[p].l+tr[p].r)/2;

if(l<=mid)change(p\*2,l,r,x);

if(r>mid)change(p\*2+1,l,r,x);

tr[p].dat=maxl(tr[p\*2].dat,tr[p\*2+1].dat);

}

void make()

{

sort(r+1,r+tot+1);

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

{

//pos[i].x=lower\_bound(r+1,r+tot+1,pos[i].x)-r;

pos[i].y1=lower\_bound(r+1,r+tot+1,pos[i].y1)-r;

pos[i].y2=lower\_bound(r+1,r+tot+1,pos[i].y2)-r;

mx=maxl(mx,maxl(pos[i].y2,pos[i].y1));

}

sort(pos+1,pos+cnt+1,cmp);

}

void init()

{

memset(&tr,0,sizeof tr);

memset(&pos,0,sizeof pos);

memset(r,0,sizeof r);

mx=tot=cnt=0;

}

int main()

{

while(scanf("%d%I64d%I64d",&n,&w,&h)!=EOF)

{

init();

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

{

long long a,b,c;

scanf("%I64d%I64d%I64d",&a,&b,&c);

pos[++cnt].x=a,pos[cnt].y1=b,pos[cnt].y2=b+h-1,pos[cnt].c=c;

pos[++cnt].x=a+w,pos[cnt].y1=b,pos[cnt].y2=b+h-1,pos[cnt].c=-c;

r[++tot]=b,r[++tot]=b-1,r[++tot]=b+1,r[++tot]=b+h-1,r[++tot]=b+h,r[++tot]=b+h+1;

}

make();

build(1,1,mx+10);

unsigned long long ans=0;

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

{

//cout<<"push "<<pos[i].y1<<" "<<pos[i].y2<<" "<<pos[i].c<<endl;

change(1,pos[i].y1,pos[i].y2,pos[i].c);

ans=maxl(ans,tr[1].dat);

//cout<<ans<<endl;

}

printf("%I64d\n",ans);

}

}

## poj 2506 Tiling

In how many ways can you tile a 2xn rectangle by 2x1 or 2x2 tiles?

Here is a sample tiling of a 2x17 rectangle.

<center> </center>

输入描述

Input is a sequence of lines, each line containing an integer nu

输出描述

For each line of input, output one integer number in a separate line giving the number of possible tilings of a 2xn rectangle.

输入例子

2

8

12

100

200

输出例子

3

171

2731

845100400152152934331135470251

1071292029505993517027974728227441735014801995855195223534251

参考答案

#include <stdio.h>

#include <iostream>

using namespace std;

char ans[][100]={

"1",

"1",

"3",

"5",

"11",

"21",

"43",

"85",

"171",

"341",

"683",

"1365",

"2731",

"5461",

"10923",

"21845",

"43691",

"87381",

"174763",

"349525",

"699051",

"1398101",

"2796203",

"5592405",

"11184811",

"22369621",

"44739243",

"89478485",

"178956971",

"357913941",

"715827883",

"1431655765",

"2863311531",

"5726623061",

"11453246123",

"22906492245",

"45812984491",

"91625968981",

"183251937963",

"366503875925",

"733007751851",

"1466015503701",

"2932031007403",

"5864062014805",

"11728124029611",

"23456248059221",

"46912496118443",

"93824992236885",

"187649984473771",

"375299968947541",

"750599937895083",

"1501199875790165",

"3002399751580331",

"6004799503160661",

"12009599006321323",

"24019198012642645",

"48038396025285291",

"96076792050570581",

"192153584101141163",

"384307168202282325",

"768614336404564651",

"1537228672809129301",

"3074457345618258603",

"6148914691236517205",

"12297829382473034411",

"24595658764946068821",

"49191317529892137643",

"98382635059784275285",

"196765270119568550571",

"393530540239137101141",

"787061080478274202283",

"1574122160956548404565",

"3148244321913096809131",

"6296488643826193618261",

"12592977287652387236523",

"25185954575304774473045",

"50371909150609548946091",

"100743818301219097892181",

"201487636602438195784363",

"402975273204876391568725",

"805950546409752783137451",

"1611901092819505566274901",

"3223802185639011132549803",

"6447604371278022265099605",

"12895208742556044530199211",

"25790417485112089060398421",

"51580834970224178120796843",

"103161669940448356241593685",

"206323339880896712483187371",

"412646679761793424966374741",

"825293359523586849932749483",

"1650586719047173699865498965",

"3301173438094347399730997931",

"6602346876188694799461995861",

"13204693752377389598923991723",

"26409387504754779197847983445",

"52818775009509558395695966891",

"105637550019019116791391933781",

"211275100038038233582783867563",

"422550200076076467165567735125",

"845100400152152934331135470251",

"1690200800304305868662270940501",

"3380401600608611737324541881003",

"6760803201217223474649083762005",

"13521606402434446949298167524011",

"27043212804868893898596335048021",

"54086425609737787797192670096043",

"108172851219475575594385340192085",

"216345702438951151188770680384171",

"432691404877902302377541360768341",

"865382809755804604755082721536683",

"1730765619511609209510165443073365",

"3461531239023218419020330886146731",

"6923062478046436838040661772293461",

"13846124956092873676081323544586923",

"27692249912185747352162647089173845",

"55384499824371494704325294178347691",

"110768999648742989408650588356695381",

"221537999297485978817301176713390763",

"443075998594971957634602353426781525",

"886151997189943915269204706853563051",

"1772303994379887830538409413707126101",

"3544607988759775661076818827414252203",

"7089215977519551322153637654828504405",

"14178431955039102644307275309657008811",

"28356863910078205288614550619314017621",

"56713727820156410577229101238628035243",

"113427455640312821154458202477256070485",

"226854911280625642308916404954512140971",

"453709822561251284617832809909024281941",

"907419645122502569235665619818048563883",

"1814839290245005138471331239636097127765",

"3629678580490010276942662479272194255531",

"7259357160980020553885324958544388511061",

"14518714321960041107770649917088777022123",

"29037428643920082215541299834177554044245",

"58074857287840164431082599668355108088491",

"116149714575680328862165199336710216176981",

"232299429151360657724330398673420432353963",

"464598858302721315448660797346840864707925",

"929197716605442630897321594693681729415851",

"1858395433210885261794643189387363458831701",

"3716790866421770523589286378774726917663403",

"7433581732843541047178572757549453835326805",

"14867163465687082094357145515098907670653611",

"29734326931374164188714291030197815341307221",

"59468653862748328377428582060395630682614443",

"118937307725496656754857164120791261365228885",

"237874615450993313509714328241582522730457771",

"475749230901986627019428656483165045460915541",

"951498461803973254038857312966330090921831083",

"1902996923607946508077714625932660181843662165",

"3805993847215893016155429251865320363687324331",

"7611987694431786032310858503730640727374648661",

"15223975388863572064621717007461281454749297323",

"30447950777727144129243434014922562909498594645",

"60895901555454288258486868029845125818997189291",

"121791803110908576516973736059690251637994378581",

"243583606221817153033947472119380503275988757163",

"487167212443634306067894944238761006551977514325",

"974334424887268612135789888477522013103955028651",

"1948668849774537224271579776955044026207910057301",

"3897337699549074448543159553910088052415820114603",

"7794675399098148897086319107820176104831640229205",

"15589350798196297794172638215640352209663280458411",

"31178701596392595588345276431280704419326560916821",

"62357403192785191176690552862561408838653121833643",

"124714806385570382353381105725122817677306243667285",

"249429612771140764706762211450245635354612487334571",

"498859225542281529413524422900491270709224974669141",

"997718451084563058827048845800982541418449949338283",

"1995436902169126117654097691601965082836899898676565",

"3990873804338252235308195383203930165673799797353131",

"7981747608676504470616390766407860331347599594706261",

"15963495217353008941232781532815720662695199189412523",

"31926990434706017882465563065631441325390398378825045",

"63853980869412035764931126131262882650780796757650091",

"127707961738824071529862252262525765301561593515300181",

"255415923477648143059724504525051530603123187030600363",

"510831846955296286119449009050103061206246374061200725",

"1021663693910592572238898018100206122412492748122401451",

"2043327387821185144477796036200412244824985496244802901",

"4086654775642370288955592072400824489649970992489605803",

"8173309551284740577911184144801648979299941984979211605",

"16346619102569481155822368289603297958599883969958423211",

"32693238205138962311644736579206595917199767939916846421",

"65386476410277924623289473158413191834399535879833692843",

"130772952820555849246578946316826383668799071759667385685",

"261545905641111698493157892633652767337598143519334771371",

"523091811282223396986315785267305534675196287038669542741",

"1046183622564446793972631570534611069350392574077339085483",

"2092367245128893587945263141069222138700785148154678170965",

"4184734490257787175890526282138444277401570296309356341931",

"8369468980515574351781052564276888554803140592618712683861",

"16738937961031148703562105128553777109606281185237425367723",

"33477875922062297407124210257107554219212562370474850735445",

"66955751844124594814248420514215108438425124740949701470891",

"133911503688249189628496841028430216876850249481899402941781",

"267823007376498379256993682056860433753700498963798805883563",

"535646014752996758513987364113720867507400997927597611767125",

"1071292029505993517027974728227441735014801995855195223534251",

"2142584059011987034055949456454883470029603991710390447068501",

"4285168118023974068111898912909766940059207983420780894137003",

"8570336236047948136223797825819533880118415966841561788274005",

"17140672472095896272447595651639067760236831933683123576548011",

"34281344944191792544895191303278135520473663867366247153096021",

"68562689888383585089790382606556271040947327734732494306192043",

"137125379776767170179580765213112542081894655469464988612384085",

"274250759553534340359161530426225084163789310938929977224768171",

"548501519107068680718323060852450168327578621877859954449536341",

"1097003038214137361436646121704900336655157243755719908899072683",

"2194006076428274722873292243409800673310314487511439817798145365",

"4388012152856549445746584486819601346620628975022879635596290731",

"8776024305713098891493168973639202693241257950045759271192581461",

"17552048611426197782986337947278405386482515900091518542385162923",

"35104097222852395565972675894556810772965031800183037084770325845",

"70208194445704791131945351789113621545930063600366074169540651691",

"140416388891409582263890703578227243091860127200732148339081303381",

"280832777782819164527781407156454486183720254401464296678162606763",

"561665555565638329055562814312908972367440508802928593356325213525",

"1123331111131276658111125628625817944734881017605857186712650427051",

"2246662222262553316222251257251635889469762035211714373425300854101",

"4493324444525106632444502514503271778939524070423428746850601708203",

"8986648889050213264889005029006543557879048140846857493701203416405",

"17973297778100426529778010058013087115758096281693714987402406832811",

"35946595556200853059556020116026174231516192563387429974804813665621",

"71893191112401706119112040232052348463032385126774859949609627331243",

"143786382224803412238224080464104696926064770253549719899219254662485",

"287572764449606824476448160928209393852129540507099439798438509324971",

"575145528899213648952896321856418787704259081014198879596877018649941",

"1150291057798427297905792643712837575408518162028397759193754037299883",

"2300582115596854595811585287425675150817036324056795518387508074599765",

"4601164231193709191623170574851350301634072648113591036775016149199531",

"9202328462387418383246341149702700603268145296227182073550032298399061",

"18404656924774836766492682299405401206536290592454364147100064596798123",

"36809313849549673532985364598810802413072581184908728294200129193596245",

"73618627699099347065970729197621604826145162369817456588400258387192491",

"147237255398198694131941458395243209652290324739634913176800516774384981",

"294474510796397388263882916790486419304580649479269826353601033548769963",

"588949021592794776527765833580972838609161298958539652707202067097539925",

"1177898043185589553055531667161945677218322597917079305414404134195079851",

"2355796086371179106111063334323891354436645195834158610828808268390159701",

"4711592172742358212222126668647782708873290391668317221657616536780319403",

"9423184345484716424444253337295565417746580783336634443315233073560638805",

"18846368690969432848888506674591130835493161566673268886630466147121277611",

"37692737381938865697777013349182261670986323133346537773260932294242555221",

"75385474763877731395554026698364523341972646266693075546521864588485110443",

"150770949527755462791108053396729046683945292533386151093043729176970220885",

"301541899055510925582216106793458093367890585066772302186087458353940441771",

"603083798111021851164432213586916186735781170133544604372174916707880883541",

"1206167596222043702328864427173832373471562340267089208744349833415761767083"

};

int main(){

int n;

while(cin>>n)

cout<<ans[n]<<endl;

return 0;

}

## poj 2672 Hotkeys

Tom is designing a menu system for a test editor. The menu consists of some menuitems, and each menuitem may contain some sub-menuitems (a sub-menuitem is also a menuitem) recursively, e.g., menuitem "File" may contain sub-menuitems "Open", "Save", "Save As", "Recent Files", "Send" and "Exit", and "Recent Files" may contain "Setting File", "Send" may contain "Mail" and "Fax". A menuitem may contain several words, and it is confirmed that the first letter in each word is a capital letter, and that the other letters are all small letters.

Each menuitem is associated with a hotkey, the menu is easier to use if the hotkeys are easier to remember. So Tom wants that the hotkey can only be a capital letter in a menuitem, which means the first letter in each word. Take "Save As" for example, the hotkey can only be 'S' or 'A'. It is clear that sub-menuitems of a menuitem can't be associated with the same hotkey, e.g., "Save" and "Save As" can't both use hotkey 'S', because they are both sub-menuitems of menu "File".

Since Tom believes press the same key twice is not convenient, he requests that the hotkeys of a menuitem and its sub-menuitems can't duplicate. In the previous example, "File" and "Recent Files" can't use the same hotkey 'F'. But this rule just affects the menuitem and its direct sub-menuitem, so "File" and "Setting File" can use the same hotkey 'F'.

Given a menu description, Tom wants to know whether it is possible to assign a hotkey for each menuitem or not.

输入描述

The menu is described as follows: a menuitem occupies a single line, and the sub-menuitems of a menuitem is listed just below this menu item. The description of the sub-menuitems starts with a "" after the last sub-menuitem. "" will take a whole line by itself. Since we may consider the menuitems in the top level belong to a global menu, there is always a "" at the end of the input.

Each menuitem may contain words and white spaces separating them. The length of a menuitem name is less than 50 characters.

There are no more than 10000 menuitems. It is confirmed that all the input are legal menuitems described above.

输出描述

A single line contains "Got It!" if a hotkey can be assigned to each menuitem; otherwise, output "No Solution".

输入例子

Exit

>

Edit

>

输出例子

No Solution

代码：

#include <iostream>

#include <cstdio>

#include <cstring>

#include <cmath>

#include <cstdlib>

#include <algorithm>

#include <vector>

using namespace std;

typedef vector<vector<int> > VVI;

VVI vt;

const int maxn = 26;

int match[maxn];

bool vis[maxn+1];

char buf[100];

bool dfs(int v){

if(vis[v]) return false;

vis[v] = true;

for(int i = 0; i < (int)vt[v].size(); i ++){

if(vt[v][i] == 0) continue;

int t = match[i];

if(t == -1 || dfs(t)){

match[i] = v;

return true;

}

}

return false;

}

bool getX(VVI vec, vector<int>& ret){

vt = vec;

ret = vector<int>(26, 0);

if(vec.size() > 26) return false;

memset(match, -1, sizeof(match));

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

memset(vis, false, sizeof(vis));

if(!dfs(i)) return false;

}

for(int i = 0; i < 26; i ++){

if(match[i] == -1) ret[i] = 1;

else{

memset(vis, false, sizeof(vis));

int v = match[i];

match[i] = 26; vis[26] = true;

if(dfs(v)){

ret[i] = 1;

match[i] = -1;

}else match[i] = v;

}

}

return true;

}

vector<int> getV(){

vector<int> vec(26, 0);

for(int i = 0; buf[i]; i ++){

if(buf[i] >= 'A' && buf[i] <= 'Z') vec[buf[i]-'A'] = 1;

}

return vec;

}

void change(vector<int>& vec, vector<int> vec2){

for(int i = 0; i < 26; i ++) vec[i] = vec[i] & vec2[i];

}

int solve(){

vector<VVI> vec;

vec.push\_back(VVI());

if(!gets(buf)) return -1;

vector<int> tmp;

while(1){

gets(buf);

if(strcmp(buf, "<") == 0){

vec.push\_back(VVI());

}else if(strcmp(buf, ">") == 0){

if(!getX(vec.back(), tmp)) return false;

vec.pop\_back();

if(vec.size() == 0) return 1;

change(vec.back().back(), tmp);

}else{

vec.back().push\_back(getV());

}

}

return 1;

}

int main() {

int ans = solve();

if(ans) printf("Got It!\n");

else printf("No Solution\n");

return 0;

}

## poj 2817 WordStack

As editor of a small-town newspaper, you know that a substantial number of your readers enjoy the daily word games that you publish, but that some are getting tired of the conventional crossword puzzles and word jumbles that you have been buying for years. You decide to try your hand at devising a new puzzle of your own.

Given a collection of N words, find an arrangement of the words that divides them among N lines, padding them with leading spaces to maximize the number of non-space characters that are the same as the character immediately above them on the preceding line. Your score for this game is that number.

输入描述

Input data will consist of one or more test sets.

The first line of each set will be an integer N (1

End of input will be indicated by a non-positive value for N .

输出描述

Your program should output a single line containing the maximum possible score for this test case, printed with no leading or trailing spaces.

输入例子

5

abc

bcd

cde

aaa

bfcde

0

输出例子

8

Hint

Note: One possible arrangement yielding this score is:

aaa

abc

bcd

cde

bfcde

代码：

#include<iostream>

#include<cstdio>

#include<algorithm>

#include<cstring>

#include<string>

#include<cmath>

#include<set>

#include<climits>

using namespace std;

const int maxn=200005;

char s[11][11];

int lp[11][11],dp[2][1<<11][11];

int cmp(char \*fir,char \*sec)

{

int lena=strlen(fir);

int lenb=strlen(sec);

int maxn=-1;

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

{

int tem=0;

for(int j=0;j+i<lena&&j<lenb;j++)

{

if(fir[i+j]==sec[j]) tem++;

}

if(tem>maxn) maxn=tem;

}

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

{

int tem=0;

for(int j=0;j+i<lenb&&j<lena;j++)

{

if(sec[i+j]==fir[j]) tem++;

}

if(tem>maxn) maxn=tem;

}

return maxn;

}

int main()

{

int n;

while(scanf("%d",&n)&&n>0)

{

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

{

scanf("%s",s[i]);

}

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

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

{

lp[i][j]=cmp(s[i],s[j]);

lp[j][i]=lp[i][j];

}

memset(dp,-1,sizeof(dp));

int cur=0;

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

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

{

cur^=1;

for(int state=0;state<(1<<n)-1;state++)

{

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

{

if(((1<<k)&state)==0)

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

{

if(nxt!=k&&dp[cur^1][state][nxt]>=0)

dp[cur][state|(1<<k)][k]=max(dp[cur][state|(1<<k)][k],dp[cur^1][state][nxt]+lp[k][nxt]);

}

}

}

}

int finalstate=(1<<n)-1;

int ans=-1;

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

ans=ans>dp[cur][finalstate][i]?ans:dp[cur][finalstate][i];

printf("%d\n",ans);

}

}

## poj 2893 M &times; N Puzzle

The Eight Puzzle, among other sliding-tile puzzles, is one of the famous problems in artificial intelligence. Along with chess, tic-tac-toe and backgammon, it has been used to study search algorithms.

The Eight Puzzle can be generalized into an M × N Puzzle where at least one of M and N is odd. The puzzle is constructed with MN − 1 sliding tiles with each a number from 1 to MN − 1 on it packed into a M by N frame with one tile missing. For example, with M = 4 and N = 3, a puzzle may look like:

1 6 2

4 0 3

7 5 9

10 8 11

Let's call missing tile 0. The only legal operation is to exchange 0 and the tile with which it shares an edge. The goal of the puzzle is to find a sequence of legal operations that makes it look like:

1 2 3

4 5 6

7 8 9

10 11 0

The following steps solve the puzzle given above.

START

1 6 2

4 0 3

7 5 9

10 8 11

DOWN

⇒

1 0 2

4 6 3

7 5 9

10 8 11

LEFT

⇒

1 2 0

4 6 3

7 5 9

10 8 11

UP

⇒

1 2 3

4 6 0

7 5 9

10 8 11

…

RIGHT

⇒

1 2 3

4 0 6

7 5 9

10 8 11

UP

⇒

1 2 3

4 5 6

7 0 9

10 8 11

UP

⇒

1 2 3

4 5 6

7 8 9

10 0 11

LEFT

⇒

1 2 3

4 5 6

7 8 9

10 11 0

GOAL

Given an M × N puzzle, you are to determine whether it can be solved.

输入描述

The input consists of multiple test cases. Each test case starts with a line containing M and N (2 ≤ M, N ≤ 999). This line is followed by M lines containing N numbers each describing an M × N puzzle.

The input ends with a pair of zeroes which should not be processed.

输出描述

Output one line for each test case containing a single word <tt>YES</tt> if the puzzle can be solved and <tt>NO</tt> otherwise.

输入例子

3 3

1 0 3

4 2 5

7 8 6

4 3

1 2 5

4 6 9

11 8 10

3 7 0

0 0

输出例子

YES

NO

参考答案

#include<cstdio>

#include<algorithm>

#include<iostream>

#include<vector>

using namespace std;

int n,m,h0;

int i,j,g,h;

vector<int> map;

//merge sort

int countt;

vector<int> merge\_tmp;

void merge\_sort(vector<int>::iterator l,vector<int>::iterator r){

if(l==r-1)

return;

vector<int>::iterator i,j,mid,now;

mid = l+((r-l)>>1);

merge\_sort(l,mid);

merge\_sort(mid,r);

merge\_tmp.clear();

for(i=l;i!=mid;i++)

merge\_tmp.push\_back(\*i);

for(i=merge\_tmp.begin(),j=mid,now=l;i!=merge\_tmp.end() && j!=r;now++){

if(\*i <= \*j){

\*now = \*i;

i++;

}else{

\*now = \*j;

countt += merge\_tmp.end()-i;

j++;

}

}

while(i!=merge\_tmp.end())

\*(now++)=\*(i++);

while(j!=r)

\*(now++)=\*(j++);

}

int main(){

std::ios\_base::sync\_with\_stdio(false);

while(1){

scanf("%d%d",&m,&n);

//cin >> m >> n;

if(m == 0 && n == 0)

return 0;

map.clear();

for(i=0;i<m;i++)

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

//cin >> g;

scanf("%d",&g);

if(g == 0)

h0=i;

else

map.push\_back(g);

}

countt = 0;

merge\_sort(map.begin(),map.end());

n &= 1;

h0 = (m-h0-1)&1;

//cout << "count" << countt << endl;

countt &= 1;

if(n==1){

if(countt==0)

printf("YES\n");

//cout << "YES" << endl;

else

printf("NO\n");

//cout << "NO" << endl;

}

if(n==0){

if(countt ^ h0 == 0)

printf("YES\n");

//cout << "YES" << endl;

else

printf("NO\n");

//cout << "NO" << endl;

}

}

}

## poj 2918 Tudoku

Tom is a master in several mathematical-theoretical disciplines. He recently founded a research-lab at our university and teaches newcomers like Jim. In the first lesson he explained the game of Tudoku to Jim. Tudoku is a straight-forward variant of Sudoku, because it consists of a board where almost all the numbers are already in place. Such a board is left over when Tom stops solving an ordinary Sudoku because of being too lazy to fill out the last few straight-forward cells. Now, you should help Jim solve all Tudokus Tom left for him.

Sudoku is played on a 9 × 9 board that is divided into nine different 3 × 3 blocks. Initially, it contains only a few numbers and the goal is to fill the remaining cells so that each row, column, and 3 × 3 block contains every number from 1 to 9. This can be quite hard but remember that Tom already filled most cells. A resulting Tudoku board can be solved using the following rule repeatedly: if some row, column or 3 × 3 block contains exactly eight numbers, fill in the remaining one.

In the following example, three cells are still missing. The upper left one cannot be determined directly because neither in its row, column, or block, there are eight numbers present. The missing number for the right cell can be determined using the above rule, however, because its column contains exactly eight numbers. Similarly, the number for the lower-most free cell can be determined by examining its row. Finally, the last free cell can be filled by either looking at its row, column or block.

7 5 3 2 8 4 6 9 1

4 8 2 9 1 6 5 3 7

1 9 6 7 5 3 8 4 2

9 3 1 6 4 2 5

2 7 5 4 9 1 3 8 6

6 4 8 3 2 1 7 9

5 6 7 3 4 9 2 1 8

8 2 4 1 7 5 9 6 3

3 1 9 6 2 8 7 5 4

输入描述

The first line contains the number of scenarios. For each scenario the input contains nine lines of nine digits each. Zeros indicate the cells that have not been filled by Tom and need to be filled by you. Each scenario is terminated by an empty line.

输出描述

The output for every scenario begins with a line containing “<tt>Scenario #</tt>i<tt>:</tt>”, where i is the number of the scenario starting at 1. Then, print the solved Tudoku board in the same format that was used for the input, except that zeroes are replaced with the correct digits. Terminate the output for the scenario with a blank line.

输入例子

2

000000000

817965430

652743190

175439820

308102950

294856370

581697240

903504610

746321580

781654392

962837154

543219786

439182675

158976423

627543918

316728549

895461237

274395861

输出例子

Scenario #1:

439218765

817965432

652743198

175439826

368172954

294856371

581697243

923584617

746321589

Scenario #2:

781654392

962837154

543219786

439182675

158976423

627543918

316728549

895461237

274395861

参考答案

#include<cstdio>

#include<iostream>

#include<cstring>

#include<algorithm>

#include<vector>

#include<cmath>

#include<cstdlib>

#define mem(a,b) memset(a,b,sizeof(a))

using namespace std;

const int maxd=10+5;

typedef long long ll;

int dx[]= {0,0,1,-1};

int dy[]= {1,-1,0,0};

int mz[maxd][maxd],visr[maxd][maxd],visc[maxd][maxd],grid[maxd][maxd];

char m[maxd][maxd];

int row[maxd],col[maxd];

void init()

{

mem(visr,0),mem(visc,0),mem(grid,0);

mem(row,0),mem(col,0);

for(int i=0; i<9; ++i)

{

scanf("%s",m[i]);

for(int j=0; j<9; ++j)

{

mz[i][j]=m[i][j]-'0';

visr[i][mz[i][j]]=1;

visc[j][mz[i][j]]=1;

if(mz[i][j]!=0) row[i]++,col[j]++;

int tmp=i/3\*3+j/3;

grid[tmp][mz[i][j]]=1;

}

}

for(int i=0; i<9; ++i)

if(row[i]==8)

{

int j;

for(j=0; j<9; ++j)

if(mz[i][j]==0) break;

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

if(visr[i][k]==0)

{

mz[i][j]=k;

visr[i][k]=1;

visc[j][k]=1;

int tmp=i/3\*3+j/3;

grid[tmp][k]=1;

break;

}

}

for(int j=0; j<9; ++j)

if(col[j]==8)

{

int i;

for(i=0; i<9; ++i)

if(mz[i][j]==0) break;

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

if(visc[j][k]==0)

{

mz[i][j]=k;

visc[j][k]=1;

visr[i][k]=1;

int tmp=i/3\*3+j/3;

grid[tmp][k]=1;

break;

}

}

}

bool dfs(int x,int y)

{

// if(!setnum(x,y)) return;

if(x==9) return true;

bool flag=false;

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

{

int tmp=x/3\*3+y/3;

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

if(visr[x][i]==0 && visc[y][i]==0 && grid[tmp][i]==0)

{

mz[x][y]=i;

visr[x][i]=1;

visc[y][i]=1;

grid[tmp][i]=1;

if(y==8)

flag=dfs(x+1,0);

else

flag=dfs(x,y+1);

if(flag) return true;

else

{

mz[x][y]=0;

visr[x][i]=0;

visc[y][i]=0;

grid[tmp][i]=0;

}

}

}

else

{

if(y==8)

flag=dfs(x+1,0);

else

flag=dfs(x,y+1);

if(flag) return true;

else return false;

}

return false;

}

void print(int k)

{

printf("Scenario #%d:\n",k);

for(int i=0; i<9; ++i)

{

for(int j=0; j<9; ++j)

printf("%d",mz[i][j]);

printf("\n");

}

printf("\n");

}

int main()

{

int kase;

scanf("%d",&kase);

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

{

init();

dfs(0,0);

print(k);

}

return 0;

}

## poj 3071 Football

Consider a single-elimination football tournament involving 2 n teams, denoted 1, 2, …, 2 n . In each round of the tournament, all teams still in the tournament are placed in a list in order of increasing index. Then, the first team in the list plays the second team, the third team plays the fourth team, etc. The winners of these matches advance to the next round, and the losers are eliminated. After n rounds, only one team remains undefeated; this team is declared the winner.

Given a matrix P = [pij ] such that pij is the probability that team i will beat team j in a match determine which team is most likely to win the tournament.

输入描述

The input test file will contain multiple test cases. Each test case will begin with a single line containing n (1 ≤ n ≤ 7). The next 2 n lines each contain 2 n values; here, the jth value on the ith line represents pij . The matrix P will satisfy the constraints that pij = 1.0 − pji for all i ≠ j, and pii = 0.0 for all i. The end-of-file is denoted by a single line containing the number −1. Note that each of the matrix entries in this problem is given as a floating-point value. To avoid precision problems, make sure that you use either the double data type instead of float.

输出描述

The output file should contain a single line for each test case indicating the number of the team most likely to win. To prevent floating-point precision issues, it is guaranteed that the difference in win probability for the top two teams will be at least 0.01.

输入例子

2

0.0 0.1 0.2 0.3

0.9 0.0 0.4 0.5

0.8 0.6 0.0 0.6

0.7 0.5 0.4 0.0

-1

输出例子

2

Hint

In the test case above, teams 1 and 2 and teams 3 and 4 play against each other in the first round; the winners of each match then play to determine the winner of the tournament. The probability that team 2 wins the tournament in this case is:

P(2 wins) = P(2 beats 1)P(3 beats 4)P(2 beats 3) + P(2 beats 1)P(4 beats 3)P(2 beats 4)

= p 21 p 34 p 23 + p 21 p 43 p 24

= 0.9 · 0.6 · 0.4 + 0.9 · 0.4 · 0.5 = 0.396.

The next most likely team to win is team 3, with a 0.372 probability of winning the tournament.

代码：

#include <iostream>

#include <stdio.h>

#include <algorithm>

#include <string.h>

#include <vector>

#define cls(p) memset(p,0,sizeof(p))

using namespace std;

const int MAX=3e2;

double dp[10][MAX];

double ks[MAX][MAX];

int N,M;

void init()

{

cls(ks);

cls(dp);

for(int i=0;i<M+3;i++)

dp[0][i]=1;

}

char s[MAX];

int main()

{

while(~scanf("%d",&N))

{

if(N==-1) break;

M=1<<N;

init();

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

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

scanf("%lf",&ks[i][j]);

int left,right,mid,L=1;

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

{

for(left=0;left<M;left+=2\*L)

{

mid=left+L;

right=mid+L;

for(int k1=left;k1<mid;k1++)

for(int k2=mid;k2<right;k2++)

{

dp[i][k1]+=dp[i-1][k1]\*dp[i-1][k2]\*ks[k1][k2];

dp[i][k2]+=dp[i-1][k1]\*dp[i-1][k2]\*ks[k2][k1];

}

}

L\*=2;

}

int t;

double ans=0;

for(int i=0;i<M;i++) if(ans<dp[N][i])

{

ans=dp[N][i];

t=i;

}

printf("%d\n",t+1);

}

return 0;

}

## poj 3183 Stump Removal

Always thinking of the cows' grazing experience, FJ has found that he must remove N (1

FJ will use the traditional high explosives to destroy the stumps. These high explosives are formulated to destroy adjacent stumps as long as those adjacent stumps are strictly shorter than the nearest stump being destroyed. The blast can continue past the closest adjacent stump to the next adjacent stump if it is even shorter than the nearest stump just destroyed. As soon as a stump encountered by the blast wave is not shorter, though, no more destruction occurs on that side of the target stump (the other side follows the same rules with whatever stumps might appear there).

Consider a line of nine stumps with these heights:

1 2 5 4 3 3 6 6 2

If FJ blows up the third stump (with height 5), then the second stump will also be destroyed (height 2) and the first stump (height 1) will also be destroyed. Likewise, the fourth stump (height 4) and fifth stump (height 3) will be destroyed since they are successively shorter, leaving the line like this:

\* \* \* \* \* 3 6 6 2

Two more explosives (at stumps 7 and 8) will destroy the rest.

Help FJ determine the minimum number of explosive charges he needs to destroy the stumps.

输入描述

Line 1: A single integer, N

Lines 2..N+1: Line i+1 contains H\_i

输出描述

Lines 1..?: Each line contains one integer which is the index of a stump to blow up. The indices must be listed in increasing order.

输入例子

9

1

2

5

4

3

3

6

6

2

输出例子

3

7

8

参考答案

#include <iostream>

using namespace std;

int h[50050];

int main()

{

int n, i;

cin >> n;

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

cin >> h[i];

}

if (h[1] >= h[2])//边界的处理

cout << 1 << endl;

for (i = 2; i <= n; i++){

if (i == n){

if (h[i] >= h[i-1])

cout << i << endl;

}

else{

if (h[i] >= h[i-1] && h[i] >= h[i+1])

cout << i << endl;

}

}

return 0;

}

## poj 3216 Repairing Company

Lily runs a repairing company that services the Q blocks in the city. One day the company receives M repair tasks, the ith of which occurs in block pi , has a deadline ti on any repairman’s arrival, which is also its starting time, and takes a single repairman di time to finish. Repairmen work alone on all tasks and must finish one task before moving on to another. With a map of the city in hand, Lily want to know the minimum number of repairmen that have to be assign to this day’s tasks.

输入描述

The input contains multiple test cases. Each test case begins with a line containing Q and M (0 < Q ≤ 20, 0 < M ≤ 200). Then follow Q lines each with Q integers, which represent a Q × Q matrix Δ = {δ ij }, where δ ij means a bidirectional road connects the ith and the jth blocks and requires δ ij time to go from one end to another. If δ ij = −1, such a road does not exist. The matrix is symmetric and all its diagonal elements are zeroes. Right below the matrix are M lines describing the repairing tasks. The ith of these lines contains pi , ti and di . Two zeroes on a separate line come after the last test case.

输出描述

For each test case output one line containing the minimum number of repairmen that have to be assigned.

输入例子

1 2

0

1 1 10

1 5 10

0 0

输出例子

2

参考答案

#include <stdio.h>

#include <cstring>

#include <algorithm>

using namespace std;

const int INF = 0X1f1f1f1f;

#define clr(x) memset(x, 0, sizeof(x))

struct Task

{

int p;

int t;

int d;

}task[210];

struct Edge

{

int next;

int to;

}edge[1000000];

int tot;

int head[210];

int link[210];

int vis[210];

int G[25][25], dis[25][25];

int sum, n;

int Q, M;

void floyd()

{

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

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

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

G[i][j] = G[i][j] < G[i][k]+G[k][j]?G[i][j]:G[i][k]+G[k][j];

}

}

void add(int s, int u)

{

edge[tot].to = u;

edge[tot].next = head[s];

head[s] = tot++;

}

int dfs(int x)

{

for(int i = head[x]; i != 0; i = edge[i].next){

int y = edge[i].to;

if(!vis[y]){

vis[y] = true;

if(link[y] == 0 || dfs(link[y])){

link[y] = x;

return true;

}

}

}

return false;

}

void solve()

{

clr(link);

sum = 0;

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

clr(vis);

if(dfs(i))

sum++;

}

}

int main()

{

int i, j, k;

while(scanf("%d %d", &Q, &M) != EOF)

{

if(Q == 0 || M == 0) break;

clr(G), clr(dis), clr(head);

n = M;

tot = 1;

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

for(j = 0; j < Q; ++j) {

scanf("%d", &G[i][j]);

if(G[i][j] == -1) G[i][j] = INF;

}

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

scanf("%d %d %d", &task[i].p, &task[i].t, &task[i].d);

floyd();

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

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

if(i != j) {

if(task[i].d + task[i].t + G[task[i].p-1][task[j].p-1] <= task[j].t)

add(i, j);

}

solve();

printf("%d\n", n-sum);

}

return 0;

}

## poj 3388 Japanese Puzzle

A brand-new Japanese puzzle is coming from the East to strike the world-popular Sudoku game and become an international hit. The rules of this puzzle are kept in secret yet, but the goal is already advertised: given a square grid n × n, where each square contains a block with one of k types of pictures, the player has to rearrange it to get the maximal possible number of equal first rows (two rows are considered equal if both of them are filled with the same pictures in the same order). An unnamed insider of the game production company told the press that the game is about moving blocks of pictures according to some rules, while the overall set of pictures isn’t changed (no pictures removed, no new pictures added). She also mentioned that the puzzle is so exciting because there are thousands of ways to swap two arbitrary pictures on a grid leaving the rest of the grid intact.

Andy works at the puzzles review magazine, and of course he got interest in this Japanese news. He realized that the information known so far is enough to find the number of equal first rows in a puzzle winning position. Now Andy wants to write a computer program for calculating this number for any given starting configuration.

For example, if you are given a puzzle which looks this way:

≡ ≡ +

≡ ∗ ∗

∆ ∗ ∆

one of the optimal rearrangements could look like

≡ ∗ ∆

≡ ∗ ∆

+ ≡ ∗

输入描述

The first line of the input file contains two integers n (1 ≤ n ≤ <nobr>40 000</nobr>) and k (1 ≤ k ≤ <nobr>50 000</nobr>). Each of the next k lines contains the number of blocks with the corresponding type of picture li (li > 0, sum of all li is exactly n 2).

输出描述

Output the maximal possible number of equal first rows at the first line of the output file. The following n lines must contain contents of the row which gives the maximum. Each line shows a single number of picture, in order they must appear. If there are many optimal solutions, any is acceptable.

输入例子

3 4

3

3

2

1

输出例子

2

1

2

3

代码：

#include <stdlib.h>

#include <stdio.h>

#include <vector>

#include <queue>

#include <stack>

#include <string>

#include <set>

#include <map>

#include <algorithm>

#include <cmath>

#include <utility>

#include <set>

#include <functional>

#include <string.h>

#include <iostream>

#include <numeric>

#include <list>

using namespace std;

int main() {

int n, k;

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

vector<int> data(k);

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

scanf("%d", &data[i]);

}

int low = 1;

int high = n + 1;

while (high - low > 1) {

int mid = (high + low) / 2;

int cols = 0;

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

cols += data[i] / mid;

}

if (cols >= n) {

low = mid;

} else {

high = mid;

}

}

printf("%d\n", low);

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

for (int j = data[i] / low; j > 0; --j) {

if (n-- > 0) {

printf("%d\n", i + 1);

} else {

break;

}

}

}

}

## poj 3435 Sudoku Checker

The puzzle game of Sudoku is played on a board of N 2 × N 2 cells. The cells are grouped in N × N squares of N × N cells each. Each cell is either empty or contains a number between 1 and N 2.

The sudoku position is correct when numbers in each row, each column and each square are different. The goal of the game is, starting from some correct position, fill all empty cells so that the final position is still correct.

This game is fairly popular in the Internet, and there are many sites which allow visitors to solve puzzles online. Such sites always have a subroutine to determine a correctness of a given position.

You are to write such a routin.

输入描述

Input file contains integer N, followed by N 4 integers — sudoku position. Empty cells are denoted by zeroes.

Constraints

1 ≤ N ≤ 10.

输出描述

Output file must contain a single string 'CORRECT' or 'INCORRECT'.

输入例子

Sample input 1

2

0 0 0 0

0 0 0 0

0 0 2 0

0 0 0 1

Sample input 2

2

2 1 3 0

3 2 4 0

1 3 2 4

0 0 0 1

输出例子

Sample output 1

CORRECT

Sample output 2

INCORRECT

Hint

Bold texts appearing in the sample sections are informative and do not form part of the actual data.

参考答案

#include<cstdio>

#include<cstring>

#define MAXN 105

int N, arr[MAXN][MAXN];

int x\_used[MAXN][11], y\_used[MAXN][11], n\_used[MAXN][11];

int main() {

while(scanf("%d", &N) != EOF) {

memset(x\_used, 0, sizeof(x\_used));

memset(y\_used, 0, sizeof(y\_used));

memset(n\_used, 0, sizeof(n\_used));

bool flag = true;

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

for(int j = 0; j < N \* N; ++ j) {

scanf("%d", &arr[i][j]);

if(arr[i][j] > 0 && flag) {

if(x\_used[i][arr[i][j]] == 1) {

flag = false;

} else {

x\_used[i][arr[i][j]] = 1;

}

if(y\_used[j][arr[i][j]] == 1) {

flag = false;

} else {

y\_used[j][arr[i][j]] = 1;

}

if(n\_used[i / N \* N + j / N + 1][arr[i][j]] == 1) {

flag = false;

} else {

n\_used[i / N \* N + j / N + 1][arr[i][j]] = 1;

}

}

}

}

if(flag) {

printf("CORRECT\n");

} else {

printf("INCORRECT\n");

}

}

return 0;

}

## poj 3484 Showstopper

Data-mining huge data sets can be a painful and long lasting process if we are not aware of tiny patterns existing within those data sets.

One reputable company has recently discovered a tiny bug in their hardware video processing solution and they are trying to create software workaround. To achieve maximum performance they use their chips in pairs and all data objects in memory should have even number of references. Under certain circumstances this rule became violated and exactly one data object is referred by odd number of references. They are ready to launch product and this is the only showstopper they have. They need **YOU** to help them resolve this critical issue in most efficient way.

Can you help them?

输入描述

Input file consists from multiple data sets separated by one or more empty lines.

Each data set represents a sequence of 32-bit (positive) integers (references) which are stored in compressed way.

Each line of input set consists from three single space separated 32-bit (positive) integers X Y Z and they represent following sequence of references: X, X+Z, X+2\*Z, X+3\*Z, …, X+K\*Z, …(while (X+K\*Z)<=Y).

Your task is to data-mine input data and for each set determine weather data were corrupted, which reference is occurring odd number of times, and count that reference.

输出描述

For each input data set you should print to standard output new line of text with either “no corruption” (low case) or two integers separated by single space (first one is reference that occurs odd number of times and second one is count of that reference).

输入例子

1 10 1

2 10 1

1 10 1

1 10 1

1 10 1

4 4 1

1 5 1

6 10 1

输出例子

1 1

no corruption

4 3

代码：

#include <iostream>

#include <cstring>

#include <cstdlib>

#include <cmath>

#include <vector>

#include <cstdio>

#include <algorithm>

#define N 500005

typedef long long LL;

using namespace std;

int n ;

LL X[N] , Y[N] , Z[N];

char str[55];

LL cal(LL k)

{

LL sum = 0 , x;

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

{

if (k < X[i]) continue;

x = min(k , Y[i]);

sum += (x - X[i]) / Z[i] + 1;

}

return sum;

}

void work()

{

n = 1;

X[n] = 0;

sscanf(str , "%I64d %I64d %I64d" , &X[n] , &Y[n] , &Z[n]);

if (!X[n]) return;

memset(str , 0 , sizeof(str));

while (gets(str) , \*str)

++ n , sscanf(str , "%I64d %I64d %I64d" , &X[n] , &Y[n] , &Z[n]) , memset(str , 0 , sizeof(str));

LL l = 1 , r = 1LL << 33 , m;

while (l < r)

{

m = (l + r) >> 1;

if (cal(m) & 1)

r = m;

else l = m + 1;

}

if (l == 1LL << 33)

puts("no corruption");

else printf("%I64d %I64d\n" , l , (cal(l) - cal(l - 1)));

}

int main()

{

while(gets(str))

work();

return 0;

}

## poj 3495 Bitwise XOR of Arithmetic Progression

Write a program that, given three positive integers x, y and z (x, y, z < 232, x ≤ y), computes the bitwise exclusive disjunction (XOR) of the arithmetic progression x, x + z, x + 2z, …, x + kz, where k is the largest integer such that x + kz ≤ y.

输入描述

The input contains multiple test cases. Each test case consists of three integers x, y, z separated by single spaces on a separate line. There are neither leading or trailing blanks nor empty lines. The input ends once EOF is met.

输出描述

For each test case, output the value of on a separate line. There should be neither leading or trailing spaces nor empty lines.

输入例子

2 173 11

输出例子

48

参考答案

#include <cstdio>

#include <cstring>

#include <iostream>

#include <algorithm>

using namespace std;

long long x,y,z;

long long Calculate(long long a,long long b,long long c,long long n)

{

if(n==0) return 0;

return (b/c)\*n+(a/c)\*n\*(n-1)/2+Calculate(c,(a\*n+b)%c,a%c,(a%c\*n+b%c)/c);

}

int main()

{

int i;

while(cin>>x>>y>>z)

{

long long ans=0;

for(i=0;i<32;i++)

ans|=(Calculate(z,x,1ll<<i,(y-x)/z+1)&1ll)<<i;

cout<<ans<<endl;

}

return 0;

}

## poj 3532 Resistance

H.L. is preparing a circuit for the next coming physical experiment. His circuit consists of N nodes, numbered 1 to N, which are connected by wires with certain resistance. H.L is curious about the equivalent resistance between Node 1 and Node N.

输入描述

The first line contains two positive integers N and M, the number of nodes and wires in the circuit.( N, M ≤ 100)

The next M lines, each describe a wire connection by three integers X, Y, R which indicates that between Node X and Node Y, there is a wire with resistance of R ohm.

输出描述

The equivalent resistance rounded after the second decimal place.

输入例子

2 2

1 2 1

1 2 1

输出例子

0.50

参考答案

#include<cstdio>

#include<cstring>

#include<cmath>

#include<algorithm>

#define INF 1000.0

#define V 105

using namespace std;

int num[V][V],n,N,m,i,x,y,j,p,k,zz;

double R[V][V],map[V][V][V],ans[V],s[V],f[V][V],max,z,A,Max,temp;

int main()

{

while (scanf("%d%d",&n,&m)!=EOF)

{

memset(R,0,sizeof(R));

memset(s,0,sizeof(s));

memset(ans,0,sizeof(ans));

memset(num,0,sizeof(num));

memset(map,0,sizeof(map));

memset(f,0,sizeof(f));

A=0;

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

scanf("%d%d%d",&x,&y,&zz),z=(double)zz,x--,y--,map[x][y][++num[x][y]]=z,map[y][x][++num[y][x]]=z;

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

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

if (num[i][j])

{

for (k=1;k<=num[i][j];k++)

R[i][j]+=(double)1.0/map[i][j][k];

R[i][j]=1.0/R[i][j];

}

ans[0]=INF;ans[n-1]=0;N=n-2;

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

{

if (R[i][0]) f[i][i]-=1.0/R[i][0],s[i]-=ans[0]/R[i][0];

if (R[i][n-1]) f[i][i]-=1.0/R[i][n-1];

for (j=1;j<=N;j++)

if (R[i][j]&&i!=j) f[i][j]=1.0/R[i][j],f[i][i]-=1.0/R[i][j];

}

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

{

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

{

temp=f[j][i]/f[i][i];

for (k=i;k<=N;k++)

f[j][k]-=f[i][k]\*temp;

s[j]-=s[i]\*temp;

}

}

for (i=N;i;i--)

{

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

s[i]-=ans[j]\*f[i][j];

ans[i]=s[i]/f[i][i];

}

for (i=0;i<=N;i++) if (R[i][n-1]) A+=ans[i]/R[i][n-1];

printf("%.2lf\n",(INF/A+1e-4));

}

return 0;

}

## poj 3548 Restoring the digits

Let's consider arithmetic expressions (addition or subtraction) over non-negative decimal integers. The expression syntax is as follows:

the first operand;

the operator sign ('+' or '‑');

the second operand;

the character '=';

the result of the operation (sum or difference, according to the operator).

The operands don't exceed <nobr>999 999 999</nobr>. In case of subtraction the first operand should be greater than or equal to the second one. There are no spaces in the expression.

Upper-case Latin letters are substituted for some digits (possibly including insignificant zeroes) so that identical letters correspond to identical digits and different letters correspond to different digits. It is guaranteed that at least one such substitution is made.

The task is to restore the substituted digits.

输入描述

The input contains only one line with the encoded arithmetic expression.

输出描述

The output consists of several lines. Each line describes one substitution and contains a letter and the corresponding digit. The letter and the digit should be separated by exactly one space. The strings should be sorted in the ascending order of letters. Letters not used in the substitution should not be listed.

输入例子

103K+G0G1=CG36

输出例子

C 1

G 0

K 5

代码：

#include <cstdio>

#include <cstring>

#include <algorithm>

using namespace std;

#define N 20

#define MAX 3628810

char s[3][N],op;

int n[3][N],len[3];

int num;

struct cha

{

char c;

int n;

}a[N];

bool OK;

void init(char \*tmp)

{

bool used[2\*N];

memset(n,0,sizeof(n));

memset(s,0,sizeof(s));

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

memset(used,0,sizeof(used));

num=0;

int i,j,k;

for(i=0,j=0; i<3; i++)

{

len[i]=j;

for(k=0;;j++,k++)

{

if(tmp[j]=='+' || tmp[j]=='-' || tmp[j]=='=' || tmp[j]=='\0')

break;

s[i][k]=tmp[j];

if(tmp[j]>='0' && tmp[j]<='9') n[i][k]=tmp[j]-'0';

else

{

n[i][k]=tmp[j]-'A'+26;

if(!used[tmp[j]-'A'])

{

used[tmp[j]-'A']=true;

a[num++].c=tmp[j];

}

}

}

s[i][k]='\0';

if(i==0) op=tmp[j];

len[i]=j-len[i];

j++;

}

}

int tran(int i)

{

int ans=0;

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

ans=ans\*10+n[i][k];

return ans;

}

void solve(int \*per)

{

sort(per,per+num); //全排列前记得排序

do //对per枚举全排列

{

for(int i=0; i<num; i++) a[i].n=per[i];

for(int i=0; i<3; i++)

for(int j=0; j<len[i]; j++)

if(s[i][j]>='A' && s[i][j]<='Z')

for(int k=0; k<num; k++)

if(s[i][j] == a[k].c)

n[i][j]=a[k].n;

int ss[3];

for(int i=0; i<3; i++)

ss[i]=tran(i);

if(op=='+' && ss[0]+ss[1]==ss[2]) {OK=true; break;}

if(op=='-' && ss[0]-ss[1]==ss[2]) {OK=true; break;}

}

while(next\_permutation(per,per+num));

}

int cmp(struct cha p ,struct cha q)

{

return p.c < q.c;

}

void BF()

{

int per[20];

OK=false;

sort(a,a+num,cmp);

for(int state=0; state<1024; state++)

{

int cc=0;

for(int k=0; k<10; k++) if(state&(1<<k)) per[cc++]=k;

if(cc==num)

solve(per);

if(OK) break;

}

for(int i=0; i<num; i++) printf("%c %d\n",a[i].c,a[i].n);

}

int main()

{

char tmp[3\*N];

while(scanf("%s",tmp)!=EOF)

{

init(tmp);

BF();

}

return 0;

}

## poj 3573 I18n

Internationalization and localization are long words that are usually abbreviated as i18n and l10n. The numbers in between i-n and l-n refer to the number of letters that were omitted. It is a very powerful abbreviation method that can be naturally used for any words.

A word may be abbreviated only when it has previously occurred in a given text. A word is abbreviated by omitting all the letters in the word except for the first and last letter and replacing omitted letters with a number of omitted letters.

Your task is to write a program that expands such abbreviations in a given text whenever possible. Expansion is possible if it is valid and unambiguous.

Expansion is valid if expanded word has previously occurred in the text and its abbreviated form corresponds to the abbreviation that is being expanded. Case is ignored for the purposes of validness. For example, expansion from i18n to internationalization is valid in this problem statement (even as internationalization was previously mentioned only with capital letter I). But expansion of p14n to parameterization is not valid since the word parameterization has never occurred before its abbreviation, and expansion of a11n to abbreviation is not valid, since it is not a correct abbreviation for the word abbreviation (correct one is a10n).

Expansion is unambiguous if there is exactly one valid expansion for it. For example, expansion from l10n to localization is unambiguous in this problem statement, but expansion from p5m cannot be made unambiguously, since both problem and program are abbreviated to p5m.

输入描述

The input file contains at most 1000 lines with at most 80 characters in each line. Each line contains one or more words separated by spaces and special symbols: ‘<tt>-</tt>’, ‘<tt>,</tt>’, ‘<tt>.</tt>’, ‘<tt>"</tt>’, ‘<tt>(</tt>’, ‘<tt>)</tt>’, ‘<tt>:</tt>’, ‘<tt>;</tt>’, ‘<tt>!</tt>’, ‘<tt>?</tt>’. There are no trailing spaces, but other separators are allowed at the end of line.

Words may be either full or abbreviated. Full word consists of one to 32 Latin letters and may be written in one of three capitalization forms: all lowercase, First Capital Letter, or ALL CAPITAL LETTERS. Abbreviated word consists of a Latin letter, followed by a number from 2 to 30 (no leading zero), followed by a Latin letter. Abbreviated words also have three corresponding capitalization forms: all l7e, F3t C5l L4r, or ALL C5L L5S.

输出描述

Write to the output file original text with original separators while expanding abbreviated words into full words whenever possible (see problem statement). Capitalization of the expanded full word shall correspond to the capitalization of the abbreviation that is being expanded.

Invalid or ambiguous abbreviations shall be left in the text as is (abbreviated). Note, that lines in the output file may be longer than 80 characters.

输入例子

The first line of sample input.

The s4d l2e of s4e i3t.

Lone, lone, l4e...

S4e input last l4e!

输出例子

The first line of sample input.

The s4d line of sample input.

Lone, lone, l4e...

Sample input last l4e!

参考答案

#include <iostream>

#include <cstring>

#include <cstdio>

using namespace std;

int a[40][40][40];

char f[40][40][40][40];

char c[100];

char s[1000];

int t[1000];

int g[300];

int main()

{

memset(g,0,sizeof(g));

for (int k=1;k<=26;k++)

{

g['A'+k-1]=k;

g['a'+k-1]=k;

}

for (int k=0;k<=9;k++)

g[k+48]=100;

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

int cnt=0;

while (gets(s))

{

int len=strlen(s);

memset(t,1,sizeof(t));

for (int i=0;s[i];i++)

if (t[i])

{

if (g[s[i]])

{

int num=len;

for (int j=i;s[j];j++)

{

if (g[s[j]]) num=j;

else

break;

}

int m=0;

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

c[m++]=s[j],t[j]=0;

if (m==1)

{

printf("%c",c[0]);

continue;

}

if (c[1]>='0'&&c[1]<='9')

{

int temp=0;

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

temp=temp\*10+c[j]-'0';

if (a[g[c[0]]][g[c[m-1]]][temp]!=1)

{

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

printf("%c",c[j]);

}

else{

int wh0=32;

if (c[m-1]>='A'&&c[m-1]<='Z') wh0=0;

printf("%c",c[0]);

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

printf("%c",

f[g[c[0]]][g[c[m-1]]][temp][j]+wh0);

printf("%c",c[m-1]);

}

}

else

{

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

printf("%c",c[j]);

int temp;

temp=m-2;

int i1=g[c[0]];

int j1=g[c[m-1]];

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

if (c[j]>96&&c[j]<97+26) c[j]-=32;

if (a[i1][j1][temp]==0)

{

a[i1][j1][temp]++;

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

f[i1][j1][temp][j]=c[j];

}

else

if (a[i1][j1][temp]==1)

{

int flag=1;

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

if (f[i1][j1][temp][j]!=c[j])

flag=0;

if (!flag)

a[i1][j1][temp]=2;

}

}

}

else

{

printf("%c",s[i]);

t[i]=0;

}

}

printf("\n");

}

return 0;

}

## poj 3752 字母旋转游戏

给定两个整数M，N，生成一个M\*N的矩阵，矩阵中元素取值为A至Z的26个字母中的一个，A在左上角，其余各数按顺时针方向旋转前进，依次递增放置，当超过26时又从A开始填充。例如，当M=5，N=8时，矩阵中的内容如下：

A B C D E F G H

V W X Y Z A B I

U J K L M N C J

T I H G F E D K

S R Q P O N M L

输入描述

M为行数，N为列数，其中M，N都为大于0的整数。

输出描述

分行输出相应的结果

输入例子

4 9

输出例子

A B C D E F G H I

V W X Y Z A B C J

U J I H G F E D K

T S R Q P O N M L

代码：

#include <iostream>

using namespace std;

char map[10000][10000];

int main()

{

int i, j, m, n, tot;

char ch = 'A';

cin >> m >> n;

for(i = 0; i < m; i++)

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

map[i][j] = '0';

i = j = tot = 0;

map[i][j] = 'A';

while (tot < n\*m-1){//注意这里的判断！

while (j+1 < n && map[i][j+1] == '0'){

if (ch >= 90) ch -= 26;

j = j + 1;

ch = ch + 1;

map[i][j] = ch;

tot = tot + 1;

}

while (i+1 < m && map[i+1][j] == '0'){

if (ch >= 90) ch -= 26;

i = i + 1;

ch = ch + 1;

map[i][j] = ch;

tot = tot + 1;

}

while (j-1 >= 0 && map[i][j-1] == '0'){

if (ch >= 90) ch -= 26;

j = j - 1;

ch = ch + 1;

map[i][j] = ch;

tot = tot + 1;

}

while (i-1 >= 0 && map[i-1][j] == '0'){

if (ch >= 90) ch -= 26;

i = i - 1;

ch = ch + 1;

map[i][j] = ch;

tot = tot + 1;

}

}

for (i = 0; i < m; i++){

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

cout << " " << map[i][j];

cout << endl;

}

}

## poj 3910 GCD Determinant

We say that a set S = {x1, x2, ..., xn} is factor closed if for any xi ∈ S and any divisor d of xi we have d ∈ S. Let’s build a GCD matrix (S) = (sij), where sij = GCD(xi, xj) – the greatest common divisor of xi and xj. Given the factor closed set S, find the value of the determinant:

<center> </center>

输入描述

The input file contains several test cases. Each test case starts with an integer n (0 9. The input data set is correct and ends with an end of file.

输出描述

For each test case find and print the value Dn mod 1000000007.

输入例子

2

1 2

3

1 3 9

4

1 2 3 6

输出例子

1

12

4

代码：

#include<iostream>

using namespace std;

int phi(int n){

int i,ans=n;

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

if(n%i==0){

ans=ans/i\*(i-1);

while(n%i==0) n/=i;

}

}

if(n>1) ans=ans/n\*(n-1);

return ans;

}

int main(){

int n,m;

while(cin>>n){

long long ans=1;

while(n--){

cin>>m;

ans\*=phi(m);

if(ans>=1000000007) ans%=1000000007;

}

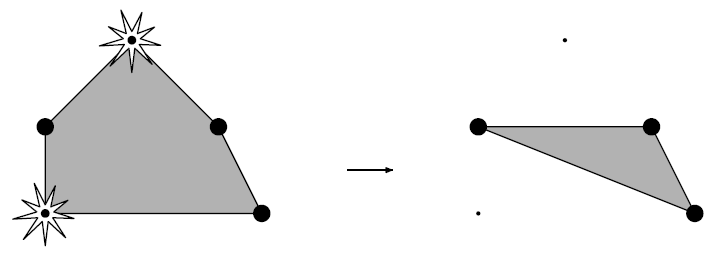
cout<<ans<<endl;

}

return 0;

}

## poj 3968 Jungle Outpost

There is a military base lost deep in the jungle. It is surrounded by n watchtowers with ultrasonic generators. In this problem watchtowers are represented by points on a plane.   
Watchtowers generate ultrasonic field and protect all objects that are strictly inside the towers’ convex hull. There is no tower strictly inside the convex hull and no three towers are on a straight line.   
The enemy can blow up some towers. If this happens, the protected area is reduced to a convex hull of the remaining towers.   
  
<center> </center>  
  
The base commander wants to build headquarters inside the protected area. In order to increase its security, he wants to maximize the number of towers that the enemy needs to blow up to make the headquarters unprotected.

输入描述

The first line of the input file contains a single integer n (3 <= n <= 50 000) — the number of watchtowers. The next n lines of the input file contain the Cartesian coordinates of watchtowers, one pair of coordinates per line. Coordinates are integer and do not exceed 106by absolute value. Towers are listed in the order of traversal of their convex hull in clockwise direction.

输出描述

Write to the output file the number of watchtowers the enemy has to blow up to compromise headquarters protection if the headquarters are placed optimally.

输入例子

|  |  |
| --- | --- |
| **#1** | 3 0 0 50 50 60 10 |
| **#2** | 5 0 0 0 10 10 20 20 10 25 0 |

输出例子

|  |  |
| --- | --- |
| **#1** | 1 |
| **#2** | 2 |

参考答案

#include <cstdio>

#include <cmath>

#include <vector>

#include <algorithm>

using namespace std;

struct Point {

double x, y;

};

typedef Point Vector;

vector<Point> P;

struct Line {

Point P, Q;

Vector v;

};

int dcmp(double x, double y=0)

{

static const double EPS=1e-8;

double d=x-y;

if (d>-EPS && d<EPS) return 0;

return d<0?-1:1;

}

Vector operator-(Vector a, Vector b)

{return (Vector){a.x-b.x,a.y-b.y};}

double Cross(Vector a, Vector b)

{return a.x\*b.y-a.y\*b.x;}

bool Left(Point P, const Line &L)

{return dcmp(Cross(L.P-P,L.Q-P))>0;}

Point Intersection(const Line &a, const Line &b)

{

double t=Cross(b.v,a.P-b.P)/Cross(a.v,b.v);

return (Point){a.P.x+t\*a.v.x, a.P.y+t\*a.v.y};

}

bool Check(int x)

{

int i, j, l, r, n=P.size();

vector<Line> L(n);

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

j = (i+x+1)%n;

L[i].P = P[i];

L[i].Q = P[j];

L[i].v = P[j]-P[i];

}

vector<Point> P(n);

vector<int> Q(n);

Q[l=r=0] = 0;

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

while (r>l && !Left(P[r-1],L[i])) r--;

while (r>l && !Left(P[l],L[i])) l++;

Q[++r] = i;

if (dcmp(Cross(L[Q[r]].v,L[Q[r-1]].v)) == 0) {

r--;

if (Left(L[i].P, L[Q[r]])) Q[r]=i;

}

if (r>l) P[r-1]=Intersection(L[Q[r]],L[Q[r-1]]);

}

while (r>l && !Left(P[r-1],L[Q[l]])) r--;

while (r>l && !Left(P[l],L[Q[r]])) l++;

return r-l+1>=3;

}

int main()

{

int n, i;

while (scanf("%d",&n)==1) {

P.resize(n);

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

scanf("%lf%lf", &P[i].x, &P[i].y);

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

int low=1, high=n-2, mid;

while (low < high) {

mid = (low+high)/2;

if (Check(mid)) low=mid+1;

else high=mid;

}

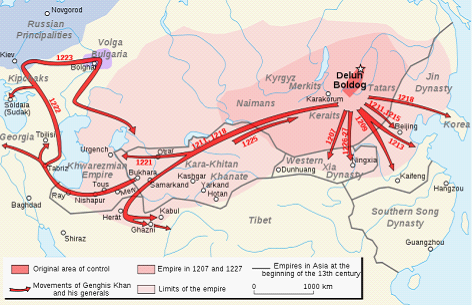
printf("%d\n", low);

}

return 0;

}

## poj 4006 Genghis Khan the Conqueror

Genghis Khan(成吉思汗)(1162-1227), also known by his birth name Temujin(铁木真) and temple name Taizu(元太祖), was the founder of the Mongol Empire and the greatest conqueror in Chinese history. After uniting many of the nomadic tribes on the Mongolian steppe, Genghis Khan founded a strong cavalry equipped by irony discipline, sabers and powder, and he became to the most fearsome conqueror in the history. He stretched the empire that resulted in the conquest of most of Eurasia. The following figure (origin: Wikipedia) shows the territory of Mongol Empire at that time.   
  
<center> </center>  
  
Our story is about Jebei Noyan(哲别), who was one of the most famous generals in Genghis Khan’s cavalry. Once his led the advance troop to invade a country named Pushtuar. The knights rolled up all the cities in Pushtuar rapidly. As Jebei Noyan’s advance troop did not have enough soldiers, the conquest was temporary and vulnerable and he was waiting for the Genghis Khan’s reinforce. At the meantime, Jebei Noyan needed to set up many guarders on the road of the country in order to guarantee that his troop in each city can send and receive messages safely and promptly through those roads.   
  
There were N cities in Pushtuar and there were bidirectional roads connecting cities. If Jebei set up guarders on a road, it was totally safe to deliver messages between the two cities connected by the road. However setting up guarders on different road took different cost based on the distance, road condition and the residual armed power nearby. Jebei had known the cost of setting up guarders on each road. He wanted to guarantee that each two cities can safely deliver messages either directly or indirectly and the total cost was minimal.   
  
Things will always get a little bit harder. As a sophisticated general, Jebei predicted that there would be one uprising happening in the country sooner or later which might increase the cost (setting up guarders) on exactly ONE road. Nevertheless he did not know which road would be affected, but only got the information of some suspicious road cost changes. We assumed that the probability of each suspicious case was the same. Since that after the uprising happened, the plan of guarder setting should be rearranged to achieve the minimal cost, Jebei Noyan wanted to know the new expected minimal total cost immediately based on current information. 

输入描述

There are no more than 20 test cases in the input.   
For each test case, the first line contains two integers N and M (1<=N<=3000, 0<=M<=N×N), demonstrating the number of cities and roads in Pushtuar. Cities are numbered from 0 to N-1. In the each of the following M lines, there are three integers xi , yi and ci (ci 7), showing that there is a bidirectional road between xi and yi , while the cost of setting up guarders on this road is ci . We guarantee that the graph is connected. The total cost of the graph is less or equal to 109 .   
  
The next line contains an integer Q (1<=Q<=10000) representing the number of suspicious road cost changes. In the following Q lines, each line contains three integers Xi, Yi and Ci showing that the cost of road (Xi, Yi) may change to Ci (Ci

输出描述

For each test case, output a real number demonstrating the expected minimal total cost. The result should be rounded to 4 digits after decimal point.

输入例子

3 3

0 1 3

0 2 2

1 2 5

3

0 2 3

1 2 6

0 1 6

0 0

输出例子

6.0000

Hint

The initial minimal cost is 5 by connecting city 0 to 1 and city 0 to 2. In the first suspicious case, the minimal total cost is increased to 6; the second case remains 5; the third case is increased to 7. As the result, the expected cost is (5+6+7)/3 = 6.

代码：

#include<iostream>

#include<cstdio>

#include<string>

#include<cstring>

#include<algorithm>

#include<vector>

using namespace std;

const int N = 3010;

int d[N],f[N];

bool v[N];

int n,m,x,y,z;

int a[N][N];

int dfn[N];

int curr;

int b[N][N];

int prim()

{

int tot = 0;

memset(d,63,sizeof(d));

memset(v,0,sizeof(v));

memset(f,0,sizeof(f));

v[0] = 1; d[0] = 0;

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

{

d[i] = a[0][i];

f[i] = 0;

}

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

{

int k = 0,minn = 1000000000;

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

if (!v[j] && d[j] < minn)

{

minn = d[j];

k = j;

}

tot += d[k];

v[k] = 1;

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

if (!v[j] && d[j] > a[k][j])

{

d[j] = a[k][j];

f[j] = k;

}

}

return tot;

}

void dfs(int x)

{

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

if (f[i] == x && a[x][i] < 1000000000 && a[x][i] != -1)

{

dfs(i);

for (int j = 0; j < n; j++) b[x][j] = min(b[x][j],b[i][j]);

}

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

if (i != f[x]) b[x][i] = min(b[x][i],a[x][i]);

}

int main()

{

while (scanf("%d%d",&n,&m) != EOF)

{

if (n == 0 && m ==0) break;

memset(a,63,sizeof(a));

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

{

scanf("%d%d%d",&x,&y,&z);

if (x != y) a[x][y] =a[y][x] = z;

}

int ans = prim();

memset(b,63,sizeof(b));

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

f[0] = -1;

dfs(0);

int q;

scanf("%d",&q);

long long sum = 0;

//cout<<ans<<endl;

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

{

scanf("%d%d%d",&x,&y,&z);

if (f[x] == y)

{

int minn = 1000000000;

sum = sum + ans;

sum -= a[x][y];

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

if (j != y && b[x][j] != -1) minn = min(minn,b[x][j]);

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

if (f[j] == x && b[j][y] != -1) minn = min(minn,b[j][y]);

minn = min(minn,z);

sum += minn;

}

else if (f[y] == x)

{

int minn = 1000000000;

sum = sum + ans;

sum -= a[y][x];

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

if (j != x && b[y][j] != -1) minn = min(minn,b[y][j]);

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

if (f[j] == y && b[j][x] != -1) minn = min(minn,b[j][x]);

minn = min(minn,z);

sum += minn;

}

else sum += ans;

}

double fin = sum;

fin = fin / q;

printf("%.4f\n",fin);

}

return 0;

}