**1x2小矩形填nxm矩形dfs**

#include<iostream>

#include<cstdio>

#include<cstdlib>

#include<cstring>

using namespace std;

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

int n,m,now,maxm;

long long f[2][1<<11];

void dfs(int pos,int s,int j){

if (pos>m){

f[now][s]+=f[1-now][j];

return;

}

if ((j>>(pos-1))&1)dfs(pos+1,s,j);//上面竖着铺了一个下来，这格不用管s

else {

if (pos<m&&!((j>>pos)&1))dfs(pos+2,s,j);//左边横着铺了一个过来，这格不用管

dfs(pos+1,s|(1<<pos-1),j);//这格自己打竖铺

}

}

int main(){

while (scanf("%d%d",&n,&m),n>0){

if ((n\*m)%2){printf("0\n");continue;}

memset(f[0],0,sizeof(f[0]));

f[0][0]=1;now=0;

maxm=(1<<m)-1;

rep(i,1,n){

now=1-now;memset(f[now],0,sizeof(f[now]));

rep(j,0,maxm)

if (f[1-now][j])dfs(1,0,j);

}

cout<<f[now][0]<<endl;

}

return 0;

}

**1x2小矩形填nxm矩形dp**

#include<iostream>

#include<cstring>

#include<cstdio>

using namespace std;

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

int f[4\*30][16];

void solve(int T,int n,int m) {

memset(f,0,sizeof(f));

f[0][(1<<m)-2]=1;//假设网格的上面还有一行且已经填满

for (int i=1;i<n\*m;i++)//枚举格子

for (int k=0;k<(1<<m);k++)//枚举二进制状态

if (k&1) {//如果第i格被覆盖

if (i/m>0)f[i][k]=f[i-1][k>>1];//如果有上一行，就可以竖着放一个

if(i%m>0&&(k&2))

f[i][k]+=f[i-1][((k>>1)^1)|(1<<m>>1)];

//如果左边一格空着，就可以横着放一个

}

else f[i][k]=f[i-1][(k>>1)|(1<<m>>1)];

//如果第i格没有被覆盖，则第i-m个格子一定是1，其他格子状态相同

cout<<T<<" "<<f[n\*m-1][(1<<m)-1]<<endl;

}

int main() {

int T,n,m;

cin>>T;

rep(t,1,T){

cin>>n;

solve(t,n,4);

}

return 0;

}

**高精度：**

#include<cstdio>

#include<cstring>

#include<cstdlib>

#include<iostream>

using namespace std;

struct BigNum{

const static int Maxn=610;

int v[Maxn],flag;//flag=0正数 flag=1负数

BigNum(int n=0 ){//数字转化

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

if (n<0)n=-n,flag=1;

else flag=0;

do{

v[++v[0]]=n%10;

n/=10;

}while(n>0);

}

BigNum(string str){//字符串转化

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

if (str[0]=='-')flag=1,str.erase(0,1);

else flag=0;

v[0]=str.size();

for(int i=1;i<=v[0];i++)v[i]=str[v[0]-i]-48;

}

bool operator <= (const BigNum & y){

BigNum & x = \* this;

if(x.v[0]!=y.v[0]) return x.v[0]<y.v[0];

for(int i=x.v[0];i>0;i--)

if(x.v[i]<y.v[i])return true;

else if(x.v[i]>y.v[i]) return false;

return true;

}

bool operator == (const BigNum & y){

BigNum & x = \* this;

if(x.v[0]!=y.v[0]) return false;

for(int i=x.v[0];i>0;i--)

if(x.v[i]!=y.v[i])return false;

return true;

}

BigNum operator + (const BigNum & y){//高精度+

BigNum & x = \* this;

BigNum c;

int i;

c.v[0]=max(x.v[0],y.v[0]);

for(i=1;i<=c.v[0];i++){

c.v[i]+=x.v[i]+y.v[i];

c.v[i+1]=c.v[i]/10;

c.v[i]%=10;

}

for(;c.v[i];i++)c.v[i+1]=c.v[i]/10,c.v[i]%=10;

c.v[0]=i-1;

return c;

}

BigNum operator - (const BigNum & y){//高精度-

BigNum & x = \* this;

BigNum c;

for(int i=1;i<=x.v[0];i++){

c.v[i]+=x.v[i]-y.v[i];

if(c.v[i]<0) c.v[i+1]--,c.v[i]+=10;

}

c.v[0]=x.v[0];

while((c.v[c.v[0]]==0)&&(c.v[0]>1)) c.v[0]--;

return c;

}

BigNum operator \* (const int & y){//高精\*单精

BigNum x = \* this ;

BigNum c;

int i,tt=0;

if(y<0)tt=1;

for(i=1;i<=x.v[0];i++){

c.v[i]+=x.v[i]\*y;

c.v[i+1]=c.v[i]/10;

c.v[i]%=10;

}

for(;c.v[i];i++)c.v[i+1]=c.v[i]/10,c.v[i]%=10;

while((c.v[c.v[0]]==0)&&(c.v[0]>1)) c.v[0]--;

c.flag=x.flag^tt;

return c;

}

BigNum operator / (int & y){//高精/单精

BigNum x = \* this ;

BigNum c=x;

int i,tt=0;

if(y<0)tt=1,y=-y;

for (int i=c.v[0];i>=1;i--){

if (i-1>0){

int ret=c.v[i-1];

c.v[i-1]=(c.v[i]%y)\*10+ret;

}

c.v[i]=c.v[i]/2;

}

while((c.v[c.v[0]]==0)&&(c.v[0]>1)) c.v[0]--;

c.flag=x.flag^tt;

return c;

}

BigNum operator \* (const BigNum & y){//高精\*高精

BigNum x = \* this ;BigNum c;

int i,j;

c.v[0]=x.v[0]+y.v[0]-1;

for(i=1;i<=x.v[0];i++)

for(j=1;j<=y.v[0];j++)

c.v[i+j-1]+=x.v[i]\*y.v[j];

i=1;

for(; c.v[i]||i<=c.v[0] ; i++ )c.v[i+1]+=c.v[i]/10,c.v[i]%=10;

c.v[0]=i-1;

while((c.v[c.v[0]]==0)&&(c.v[0]>1))c.v[0]--;

c.flag=x.flag^y.flag;

return c;

}

BigNum operator / (const BigNum & y){//高精/高精

BigNum x = \* this ;

BigNum c,tmp,one(1);

if (x.v[0]==1&&x.v[1]==0)c=x;

else if (x==y)c=BigNum(1);

else if (x<=y)c=BigNum(0);

else {

BigNum L(1),R,Mid;

R=x;

while (L<=R){

Mid=(L+R)/2;

tmp=Mid\*y;

if (tmp<=x){

c=Mid;

L=Mid+one;

}

else R=Mid-one;

}

c.flag=x.flag^y.flag;

}

return c;

}

void Print(){

if (flag)cout<<"-";

for(int i=v[0];i>0;i--)printf("%d",v[i]);

printf("\n");

}

};

**差分约束：**

/\*

有n 个人，按从1?n的顺序排队。对于不同编号的两人A和B有两种关系，分别为喜欢关系和不喜欢关系。输入数据中有X对喜欢关系的人和Y对不喜欢关系的人。对于喜欢关系的两人有一个距离C表示这两个人最多相距C。对于不喜欢关系的两人有一个距离C表示这两人至少相距C。

你的任务是求出第1个人到第n个人的最长距离。

Input

第一行一个数T表示有T组测试数据。

每组数据第一行是N,X,Y （2<=N<=1000)(1<=X,Y<=10000)

接下来X行每行三个数A,B,C。(1<=A<B<=N ) (1<=C<=1000000)

接下来Y行每行三个数A,B,C。(1<=A<B<=C) (1<=C<=1000000)

Output

一个数。

若不可能存在一个可行的排队方式输出-1

如果1到N的距离太大(即1到N不连通）输出-2。

否则输出1到N的最短距离。

Sample Input

1

4 2 1

1 3 8

2 4 15

2 3 4

Sample Output

19

\*/

#include<iostream>

using namespace std;

int n,ml,md,st,len,list[1010],d[1010],first[1010],ru[1010];

bool v[1010];

struct node

{ int x,y,next,d;

}edge[21000];

void add(int x,int y,int d)

{ edge[++len].x=x;edge[len].y=y;edge[len].d=d;

edge[len].next=first[x];first[x]=len;

}

bool spfa(int st)

{ int head=0,i,x,y;

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

v[i]=false;d[i]=2000000000;ru[i]=0;

if (first[i]!=-1){

list[++head]=i;

v[i]=true;ru[i]++;

}

}d[list[head]]=0;

bool bo=true;

while (head!=0)

{ x=list[head];head--; v[x]=false;;

int k=first[x];

while (k!=-1)

{

y=edge[k].y;

if (d[y]>d[x]+edge[k].d)

{ d[y]=d[x]+edge[k].d;

if (!v[y])

{ v[y]=true;

list[++head]=y;

ru[y]++;

if (ru[y]>n+1)return false;

}

}

k=edge[k].next;

}

}

return true;

}

int main()

{ int t,i,x,y,c;char s[5];

scanf("%d",&t);

while (t-->0){

scanf("%d%d%d",&n,&ml,&md);

len=0;memset(first,-1,sizeof(first));

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

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

add(x,y,c);

}

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

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

add(y,x,-c);

}

if (spfa(1)){

if (d[n]>1000000000)printf("-2\n");

else printf("%d\n",d[n]);

}

else printf("-1\n");

}

}

**欧拉筛素数：**

#include<iostream>

#include<cstring>

using namespace std;

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

int p[80000],v[1000001],a[110];

int tot;

void init(){

rep(i,2,1000000){

if (!v[i])p[++tot]=i;

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

if ((long long)p[j]\*i>1000000)break;

v[p[j]\*i]=1;

if (i%p[j]==0)break;

}

}

}

**拓展欧几里得：**

/\*

有两只青蛙，在地球的同一个纬度上，但是它们所在经度位置不同。青蛙A一开始在位置x，每次向西跳m米；青蛙B一开始在位置y，每次向西跳n米；这个纬度绕地球一圈是L米。两只青蛙一起跳，问它们最早会在跳几次后相遇，如果不能相遇则输出“Impossible”。

x、y、m、n、L<=2100000000，x!=y。

设它们一起跳了p次，另有一整数q，可以列出方程：(x+p∙m)-(y+p∙n)=q∙L。写成关于p、q的二元一次方程是：(m–n)∙p+L∙q=y–x。

设A=(m–n),B=L ,C=y–x。这道题目就可以转化为求方程A∙p+B∙q=C的p不小于0且最小的解。

这是可以运用拓展欧几里得算法做到的。我们先求出A∙p0+B∙q0=(A,B)的解，如果C mod(A,B)≠0，则无解；否则p=p0∙C/((A,B))就是一个解。然后我们发现，如果给p减去 B/d 同时给q加上 A/d ，等式仍然成立，所以我们可以把p对 B/d 取模，得到想要的答案。

\*/

#include<algorithm>

#include<iostream>

#include<cstdio>

using namespace std;

long long extended\_gcd( long long A , long long B , long long &x , long long &y ) {

//solve ax+by=gcd(a,b)

if ( B == 0 ) {

x = 1;

y = 0;

return A;

}

long long d = extended\_gcd( B , A % B , x , y );

long long tmp = x;

x = y;

y = tmp - A / B \* y;

return d;

}

void caln( long long A , long long B , long long C ) {

long long x , y;

long long d = extended\_gcd( A , B , x , y );

if ( d == 0 || C % d != 0 ) {

cout << "Impossible" << endl;

return;

}

x = x \* ( C / d );

long long p = abs( B / d );

cout << ( x % p + p ) % p << endl;

}

void solve() {

long long x , y , m , n , L;

cin >> x >> y >> m >> n >> L;

caln( m - n , L , y - x );

}

int main() {

solve();

return 0;

}

已知a,b,n,求x，使得http://images2015.cnblogs.com/blog/803405/201602/803405-20160201214244304-1395370709.png，可以转化为：http://images2015.cnblogs.com/blog/803405/201602/803405-20160201214254632-2031386830.png，则要求gcd(a,n)|b，否则无解。

**解同余方程组：**

/\*

解同余方程组：

x≡r1(moda1)x≡r2(moda2)......x≡rn(modan)

其中模数不一定互质。

题解

若模数两两互质，我们可以用中国剩余定理来解。

这里我们先考虑两个方程：

x≡r1(moda1)x≡r2(moda2)

我们可以写成:

x+y1a1=r1x−y2a2=r2

相减得：y1a1+y2a2=r1−r2也就是ax+by=m的形式。

这是可以用扩展欧几里德解的。

若gcd(a,b)/|m那么方程就无解，直接输出-1。

否则我们可以解出上式的y1。回带得到一个特解x0=r1−y1a1。

通解可以写成x=x0+k∗lcm(a1,a2)也就是x≡x0(modlcm(a1,a2))。

这样我们就将两个方程合并为了一个。重复进行以上操作，我们最终能将n个方程全部合并，再用扩展欧几德得解出来就好了。

\*/

#include<cstdio>

#include<cstring>

#include<iostream>

#include<algorithm>

using namespace std;

typedef long long ll;

ll a[100005],r[100005];

int n;

ll exgcd(ll a,ll b,ll &x,ll &y){

if(b==0) return x=1,y=0,a;

ll tmp=exgcd(b,a%b,y,x);

y-=a/b\*x;

return tmp;

}

ll solve(){

ll M=a[1],R=r[1],x,y,d;

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

d=exgcd(M,a[i],x,y);

if((R-r[i])%d!=0) return -1;

x=(R-r[i])/d\*x%a[i];

R-=x\*M;

M=M/d\*a[i];

R%=M;

}

return (R%M+M)%M;

}

int main(){

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

for(int i=1;i<=n;i++)scanf("%lld%lld",&a[i],&r[i]);

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

}

return 0;

}

**中国剩余定理：**

中国剩余定理(西方数学史中的叫法)，就是上一题目的一般情况。

设m1,m2...mk是两两互素的正整数，即: gcd(mi, mj) = 1 (其中 i != j, i, j >= 1且 <=k).

则同余方程组:

x ≡ a1(mod m1)

x ≡ a2(mod m2)

... ...

x ≡ ak(mod mk)

存在唯一[m1,m2...mk]使方程成立.

解法同物不知数是一致的.我们可以稍微模仿一下.

唯一的难题就是如何把上面70, 15, 21的求法,对应到一般情况来.

假设: N1, N2, ... ,Nk.就是对应的权值, 满足如下条件:

N1 能够被 m2, m3..., mk整除,但是除以m1正好余1.

N2 能够被 m1, m3..., mk整除,但是除以m2正好余1.

... ...

Nk能够被m1, m2,...,mk-1整除,但是除以mk正好余1.

N1->Nk如果求出来了,那么假设:

x1 = N1\*a1 + N2\*a2 + ... + Nk\*ak就是我们要求的x一个解, 同物不知数一样,我们把x1 mod (m1\*m2\*...\*mk)的结果

就是x的最小整数解,若为负数,则再加上一个m1\*m2\*...\*mk.因为加减整数倍个m1\*m2\*...\*mk所得结果都是x的解.

所以问题只剩下一个,就是求N1, N2,...,Nk.

怎么求呢?我需要先化简一番:

设m = m1\*m2\*...\*mk, L, J为任意整数.

因为Ni能被m1, m2,...,mi-1, mi+1,...,mk整除(其中i+1<k)

因此: Ni = m/mi \*L

又因为Ni除以mi余1

因此: Ni = mi\*J + 1

即: mi\*J + 1 = m/mi \*L ==> (-mi)\*J + m/mi\*L = 1

而m1-->mk这些数都是互质数,所以(-mi) 同 m/mi也是互质数.即:

gcd(mi, m/mi) = 1也就是说:

 (m/mi)\*L + (-mi)\*J  = gcd(m/mi, -mi)==>其中-mi和m/mi都是已知的,J和L未知

这就是经典扩展欧几里德定理的原型(由定理知J和L是唯一的, 因此,N1-->Nk有唯一解).

按照扩展欧几里德定理求解即可.

|  |
| --- |
| 扩展欧几里德定理： a和b都是不全为0的正整数，则： a\*x + b\*y = gcd(a, b) 存在唯一的x, y使得上面等式成立。 (当然，容易得知，如果，a和b中有负数，那么也是成立的。) 本题中，m/mi相当于a, -mi相当于b, L相当于x, J相当于y。求出L, J就能求出Ni。 |

此时Ni求解完毕.

我们要求的x的最小整数解也就呼之欲出了.

#include <iostream>

using namespace std;

//参数可为负数的扩展欧几里德定理

void exOJLD(int a, int b, int &x, int &y){

    //根据欧几里德定理

    if(b == 0){//任意数与0的最大公约数为其本身。

        x = 1;

        y = 0;

    }else{

        int x1, y1;

        exOJLD(b, a%b, x1, y1);

        if(a\*b < 0){//异号取反

            x = - y1;

            y = a/b\*y1 - x1;

        }else{//同号

            x = y1;

            y = x1 - a/b\* y1;

        }

    }

}

//剩余定理

int calSYDL(int a[], int m[], int k){

    int N[k];//这个可以删除

    int mm = 1;//最小公倍数

    int result = 0;

    for(int i = 0; i < k; i++){

        mm \*= m[i];

    }

    for(int j = 0; j < k; j++){

        int L, J;

        exOJLD(mm/m[j], -m[j], L, J);

        N[j] = m[j] \* J + 1;//1

        N[j] = mm/m[j] \* L;//2 【注】1和2这两个值应该是相等的。

        result += N[j]\*a[j];

    }

    return (result % mm + mm) % mm;//落在(0, mm)之间，这么写是为了防止result初始为负数，本例中不可能为负可以直接 写成：return result%mm;即可。

}

int main(){

    int a[3] = {2, 3, 2};

    int m[3] = {3, 5, 7};

    cout<<"结果:"<<calSYDL(a, m, 3)<<endl;

}

**模拟退火：**

#include<iostream>

#include<cstdlib>

#include<cstdio>

#include<cmath>

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

using namespace std;

const int maxn = 60 , Times = 20;

const double Pi = 3.1415926536 , Eps = 1e-8 , Ini = 1000 , Dec = 0.80;

struct point {

double x , y;

};

int n;

double M[maxn];

point A[maxn] , B[maxn];

void input() {

scanf( "%d" , &n );

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

scanf( "%lf%lf" , &A[i].x , &A[i].y );

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

scanf( "%lf" , &M[i] );

M[i] = M[i] \* Pi / 180;

}

}

double dist2( point P , point Q ) {

return sqr( P.x - Q.x ) + sqr( P.y - Q.y );

}

point rotate( point O , point P , double angle ) {

point ret = O;

P.x -= O.x;

P.y -= O.y;

ret.x += P.x \* cos( angle ) - P.y \* sin( angle );

ret.y += P.x \* sin( angle ) + P.y \* cos( angle );

return ret;

}

double caln( point S ) {

B[1] = S;

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

B[i + 1] = rotate( A[i] , B[i] , M[i] );

return dist2( B[1] , B[n + 1] );

}

void solve() {

srand( 1995 - 05 - 12 );

point P; P.x = P.y = 0;

double D = caln( P );

for ( double E = Ini ; E > Eps ; E \*= Dec )

for ( int T = 1 ; T <= Times ; T++ ) {

point \_P = P;

\_P.x += E \* ( rand() % 10001 - 5000 );

\_P.y += E \* ( rand() % 10001 - 5000 );

double \_D = caln( \_P );

if ( \_D < D ) {

P = \_P;

D = \_D;

}

}

caln( P );

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

printf( "%d %d\n" , ( int ) floor( B[i].x + 0.5 ) , ( int ) floor( B[i].y + 0.5 ) );

}

int main() {

input();

solve();

return 0;

}

**最近点对：**

#include<cstdio>

#include<cstring>

#include<cstdlib>

#include<iostream>

#include<algorithm>

#include<cmath>

using namespace std;

#define N 100010

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

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

const double INF=1e17;

struct node{

double x,y;

}a[N],b[N],tmp[N];

int n;

double ans;

bool cmp1(const node &n1,const node &n2){

if (n1.x<n2.x)return true;

if (n1.x==n2.x&&n1.y<n2.y)return true;

return false;

}

bool cmp2(node n1,node n2){return n1.y<n2.y;}

void meger(int l,int m,int r){

int k=l,k1=l,k2=m+1;

while (k<=r){

if ((k2>r)||(k1<=m&&a[k1].y<a[k2].y))tmp[k++]=a[k1++];

else tmp[k++]=a[k2++];

}

rep(i,l,r)a[i]=tmp[i];

}

void find(int l,int r){

if (l==r)return ;

if (l+1==r){

double tmp=sqr(a[l].x-a[r].x)+sqr(a[l].y-a[r].y);

tmp=sqrt(tmp);

if (tmp<ans)ans=tmp;

return;

}

int mid=(l+r)/2,tot=0;

double midl=(a[mid].x+a[mid+1].x)/2.0;

find(l,mid);find(mid+1,r);

meger(l,mid,r);

rep(i,l,r)

if (fabs(a[i].x-midl)<ans)b[++tot]=a[i];

if(tot!=0){

rep(i,1,tot-1)

rep(j,i+1,i+6){

if (j>tot)break;

double tmp=sqr(b[i].x-b[j].x)+sqr(b[i].y-b[j].y);

tmp=sqrt(tmp);

if (tmp<ans)ans=tmp;

}

}

}

int main(){

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

rep(i,1,n)scanf("%lf%lf",&a[i].x,&a[i].y);

sort(a+1,a+1+n,cmp1);

ans=INF;find(1,n);

printf("%.2lf\n",ans/2.0);

}

}

**卡特兰数：**

递推：h(n)=C(2n,n)/(n+1) (n=0,1,2,...)

h(n)=h(n-1)\*(4\*n-2)/(n+1);

h(n)=c(2n,n)-c(2n,n-1)(n=0,1,2,...)

令h(0)=1,h(1)=1，catalan数满足递推式[2]  ：

h(n)= h(0)\*h(n-1)+h(1)\*h(n-2) + ... + h(n-1)\*h(0) (n>=2)

**应用：**

**矩阵连乘：** P=a1×a2×a3×……×an，依据乘法结合律，不改变其顺序，只用括号表示成对的乘积，试问有几种括号化的方案？(h(n)种)[4

**出栈次序**

一个栈(无穷大)的进栈序列为1，2，3，…，n，有多少个不同的出栈序列? h（n）

**凸多边形三角划分**

在一个凸多边形中，通过若干条互不相交的对角线，把这个多边形划分成了若干个三角形。任务是键盘上输入凸多边形的边数n，求不同划分的方案数f（n）。比如当n=6时，f（6）=14。

答案为f（n）=h（n-2） （n=2，3，4，……）

类似问题

一位大城市的律师在她住所以北n个街区和以东n个街区处工作。每天她走2n个街区去上班。如果她从不穿越（但可以碰到）从家到办公室的对角线，那么有多少条可能的道路？

在圆上选择2n个点,将这些点成对连接起来使得所得到的n条线段不相交的方法数？

**给定节点组成二叉搜索树**

给定N个节点，能构成多少种不同的二叉搜索树？

（能构成h（N）个）

（这个公式的下标是从h(0)=1开始的）

**n对括号正确匹配数目**

给定n对括号，求括号正确配对的字符串数，例如：

1对括号：() 1种可能

2对括号：()() (()) 2种可能

3对括号：((())) ()(()) ()()() (())() (()()) 5种可能

那么问题来了，n对括号有多少种正确配对的可能呢？

实际上假设S(n)为n对括号的正确配对数目，那么有递推关系S(n)=S(1)S(n)+S(2)S(n-1) +...+S(n)S(1)，显然S(n)是一个卡特兰数。

对于在n位的2进制中，有m个0，其余为1的catalan数为：C（n,m）-C(n,m-1)。证明可以参考标准catalan数的证明。[7]

**问题1的描述：**有n个1和m个-1（n>m），共n+m个数排成一列，满足对所有0<=k<=n+m的前k个数的部分和Sk > 0的排列数。 问题等价为在一个格点阵列中，从（0，0）点走到（n，m）点且不经过对角线x==y的方法数（x > y）。

考虑情况I：第一步走到（0，1），这样从（0，1）走到（n，m）无论如何也要经过x==y的点，这样的方法数为(( n+m-1,m-1 ));

考虑情况II：第一步走到（1，0），又有两种可能：

a . 不经过x==y的点；（所要求的情况）

b . 经过x==y的点，我们构造情况II.b和情况I的一一映射，说明II.b和I的方法数是一样的。设第一次经过x==y的点是（x1，y1），将（0，0）到（x1，y1）的路径沿对角线翻折，于是唯一对应情况I的一种路径；对于情况I的一条路径，假设其与对角线的第一个焦点是（x2，y2），将（0，0）和（x2，y2）之间的路径沿对角线翻折，唯一对应情况II.b的一条路径。

问题的解就是总的路径数 ((n+m, m)) - 情况I的路径数 - 情况II.b的路径数。

((n+m , m)) - 2\*((n+m-1, m-1))

或：((n+m-1 , m)) - ((n+m-1 , m-1))

**问题2的描述：**有n个1和m个-1（n>=m），共n+m个数排成一列，满足对所有0<=k<=n+m的前k个数的部分和Sk >= 0的排列数。（和问题1不同之处在于此处部分和可以为0，这也是更常见的情况） 问题等价为在一个格点阵列中，从（0，0）点走到（n，m）点且不穿过对角线x==y的方法数（可以走到x==y的点）。

把（n，m）点变换到（n+1，m）点，问题变成了问题1。

方法数为：

((n+m+1, m)) - 2\*((n+m+1-1, m-1))

或：((n+m+1-1, m)) - ((n+m+1-1, m-1))

**米勒拉宾素数测试+大整数分解：**

/\*

判断所有质数 i<=k， 2^i-1 是否是质数，不是的话就要将它分解质

因数输出来。

k<=63

\*/

#include<iostream>

#include<cmath>

#include<cstdlib>

#include<algorithm>

using namespace std;

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

typedef long long ll;

ll fac[110];

int tot;

int isPrime(int x){

if (x==2)return 1;

rep(i,2,(int)sqrt(x))

if (x%i==0)return 0;

return 1;

}

//米勒-拉宾素数测试

ll qmul(ll a,ll b,ll c){//快速乘法a\*b%c

a%=c;b%=c;

ll ans=0;

while (b){

if (b&1)ans=(ans+a)%c;

a=(a<<1)%c;

b>>=1;

}

return ans;

}

ll qpow(ll a,ll b,ll c){//快速幂a^b % c

if (b==1)return a%c;

if (!b)return 1%c;

ll tmp=qpow(a,b/2,c);

tmp=qmul(tmp,tmp,c);

if (b&1)tmp=qmul(tmp,a,c);

return tmp;

}

int check(ll a,ll d,int r,ll n){

//验证a^(d\*2^r) %n=1 或 n-1

ll tmp=qpow(a,d,n);

ll last=tmp;

rep(i,1,r){

tmp=qmul(tmp,tmp,n);

if (tmp==1&&last!=1&&last!=n-1)return false;

last=tmp;

}

if (tmp!=1)return false;

return true;

}

int Miller\_Rabin(ll n){

if ( n<2 || n&1==0 )return 0;

if (n==2)return 1;

ll d=n-1; int r=0;

while ( (d&1)==0 )d>>=1,r++; //将n分解成d\*2^r

rep(i,1,30){//随机测试30次，减小出错概率

ll a=rand()%(n-1)+1;

if (!check(a,d,r,n))return false;//不通过测试则是合数

}

return true;

}

///////////////////

///Pollard\_rho大整数质因数分解

ll gcd(ll a,ll b){

if (a<0)a=-a;

if (b<0)b=-b;

if (!b)return a;

else return gcd(b,a%b);

}

ll Pollard\_rho(ll x,ll c){

ll i=1,k=2;

ll x0=rand()%x;

ll y=x0;

while(1){

i++;

x0=(qmul(x0,x0,x)+c)%x; //x^2+c

ll d=gcd(y-x0,x);

if(d!=1&&d!=x) return d;

if(y==x0) return x; //找到循环，算法失败，重来

if(i==k){y=x0;k+=k;} //将对y的寻找范围扩大2倍，是一个优化

}

}

void findfac(ll n){

if (Miller\_Rabin(n)){

fac[++tot]=n;

return;

}

ll p=n;

while (p>=n)p=Pollard\_rho(p,rand()%(n-1)+1);

findfac(p); //找到了一个因子，那么递归下去继续找

findfac(n/p);

}

/////////////////////////////

void work(int k){

ll n=(1LL<<(k))-1;

tot=0;

findfac(n);

if (tot>1){

sort(fac+1,fac+tot+1);

printf("%lld",fac[1]);

rep(i,2,tot) printf(" \* %lld",fac[i]);

printf(" = %lld = ( 2 ^ %d ) - 1\n", n, k);

}

}

int main(){

srand(1996-04-22);

int k;

scanf("%d",&k);

rep(i,2,k)

if (isPrime(i))

work(i);

return 0;

}

**点双连通分量：**

#include<cstdio>

#include<cstring>

#include<iostream>

#include<cstdlib>

using namespace std;

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

#define N 1010

int bo[N][N],who[N][N],cnt[N],color[N],sta[N],dfn[N],low[N],bk[N];

int topp,n,tot,times;

void tarjan(int u){

int i,v;

dfn[u]=low[u]=++times;

sta[topp++]=u;

rep(v,1,n)

if(!bo[u][v]){

if(!dfn[v]){

tarjan(v);

if(low[v]<low[u])low[u]=low[v];

if(low[v]==dfn[u]){

who[tot][0]=u;

for(i=1,sta[topp]=-1;sta[topp]!=v;i++)

who[tot][i]=sta[--topp];

if(i>2){

cnt[tot]=i;

tot++;

}

}

}

else if(low[u]>dfn[v])low[u]=dfn[v];

}

}

int paint(int fa,int x,int now,int k){

rep(y,0,cnt[k]-1)

if((!bo[who[k][x]][who[k][y]])&&y!=fa){

if(!color[y]){

color[y]=3-now;

if(paint(x,y,3-now,k))return 1;

}

else if(color[x]==color[y])return 1;

}

return 0;

}

int main(){

freopen("ball.in","r",stdin);

freopen("ball.out","w",stdout);

int m,x,y,ans;

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

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

tot=topp=times=0;

rep(i,1,m){

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

bo[x][y]=bo[y][x]=1;

}

rep(i,1,n)bo[i][i]=1;

rep(i,1,n)

if(!dfn[i])tarjan(i);

rep(k,0,tot-1){

rep(i,0,cnt[k]-1)color[i]=0;

color[0]=1;

if(paint(-1,0,1,k))

rep(i,0,cnt[k]-1)

bk[who[k][i]]=1;

}

ans=0;

rep(i,1,n)

if(!bk[i])ans++;

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

return 0;

}

**Trie：**

/\*

题目大意：将给出的火星字符串翻译成英文字符串，每个单词有对应的英文单词，如果不存在对应的就输出原有的即可。

解题思路：对火星字符串进行建树，在最后一个字母上存下对应的英文字符串。这里要注意一个标记，不然会超内存。

\*/

#include<iostream>

using namespace std;

int tot=0;

struct node

{

int c[27],times;

char st[11];

void init()

{ memset(c,0,sizeof(c));

times=0;

}

}t[510001];

void insert(char \*s1,char \*s2)

{ int p=0;

while(\*s2)

{

int index=\*s2-'a'+1;

if (t[p].c[index]==0)

{ t[p].c[index]=++tot;

t[tot].init();

}

p=t[p].c[index];

s2++;

}

t[p].times++;

strcpy(t[p].st,s1);

}

char\* search(char \*s)

{ int p=0,i;

for (i=0;s[i];i++)

{

int index=s[i]-'a'+1;

if (t[p].c[index]==0)return s;

p=t[p].c[index];

}

if (t[p].times==1)return t[p].st;

else return s;

}

int main()

{ int i,j,k,num,len;

char s1[25],s2[25],s[3005];

scanf("%s",s1);

t[0].init();

while(scanf("%s",s1)!=EOF)

{

if (strcmp(s1,"END")==0)break;

scanf("%s",s2);

insert(s1,s2);

}

scanf("%s",s1);getchar();

while(1)

{

gets(s);

if (strcmp(s,"END")==0)break;

len=strlen(s);num=0;

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

{

if (s[i]>='a'&&s[i]<='z')

s1[num++]=s[i];

else

{

s1[num]=0;num=0;

printf("%s",search(s1));

printf("%c",s[i]);

}

}

printf("\n");

}

}

**凸包：**

/\*

草地上有些树，用树做篱笆围一块最大的面积来养牛，每头牛要50平方米才能养活，问最多能养多少只羊

凸包求面积，分解成三角形用叉积求面积。

\*/

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

struct node

{

int x,y;

}list[10001],sta[10001];

int n,top;

int multi(node p1,node p2,node p0)

{

int x1,y1,x2,y2;

x1= p1.x-p0.x;

y1= p1.y-p0.y;

x2= p2.x-p0.x;

y2= p2.y-p0.y;

return x1\*y2 - x2\*y1;

}

double dis(node p1,node p2)

{

return sqrt( double((p1.x-p2.x)\*(p1.x-p2.x)) + double((p1.y-p2.y)\*(p1.y-p2.y)) );

}

int cmp(node p1,node p2)

{

int tt= multi(p1,p2,list[1]);

if( tt>0) return 1;

if ( tt==0 && dis(p1,list[1])<dis(p2,list[1]) ) return 1;

return 0;

}

void qs(int l ,int r)

{

if (l>=r) return ;

int i=l,j=r;

node no= list[(l+r)/2];

do{

while( cmp(list[i],no) )i++;

while( cmp(no,list[j]) )j--;

if(i<=j)

{

node tt=list[i]; list[i]=list[j]; list[j]=tt;

i++;j--;

}

}while(i<=j);

qs(i,r);

qs(l,j);

}

void tubao()

{

qs(2,n);

int i;

for(i=1;i<=2;i++) sta[i]= list[i];

top=2;

for(i=3;i<=n;i++)

{

while(top>1 && multi(sta[top],list[i],sta[top-1])<=0) top--;

sta[++top]=list[i];

}

}

int main()

{ long long t;

int i,j;

scanf("%d",&n);

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

{

scanf("%d%d",&list[i].x,&list[i].y);

if( list[i].y<list[1].y || (list[i].y==list[1].y && list[i].x<list[1].x) )

{

node ttt=list[1]; list[1]=list[i]; list[i]= ttt;

}

}

tubao();

t=0;

for (i=2;i<top;i++)t+=multi(sta[i],sta[i+1],sta[1]);

if (t<0)t=-t;t/=2;

printf("%I64d\n",t/50);

return 0;

}

**Dancinglinks：**

/\*

给出一个n\*m的01矩阵，要求选出一些行，使得每列有且仅有一个1。问是否可能。

n<=16，m<=300。多组数据，T<=500。

\*/

#include<cstdio>

#include<iostream>

using namespace std;

#define N 20

#define M 310

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

struct node

{ int l,r,u,d;

}p[N\*M];//双向链表

//链头编号为1～m 则 原矩阵每个点的编号为 i\*m+j

int data[N][M],lev[M];//data为原矩阵，lev[i]第i列 有多少个1

int n,m,s;//s 为左上角虚拟点编号 n\*m+m+1

int tot,sta[N\*M];//sta 是记录被删除节点的栈，tot是栈顶节点

void input()

{ rep(i,1,n)

rep(j,1,m)

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

s=n\*m+m+1;tot=0;

}

void remove(int t)

{ sta[++tot]=t;

if (t>m)lev[(t-1)%m+1]--;

p[p[t].l].r=p[t].r;p[p[t].r].l=p[t].l;

p[p[t].u].d=p[t].d;p[p[t].d].u=p[t].u;

}

void rebuild()

{ int t=sta[tot--];

if (t>m)lev[(t-1)%m+1]++;

p[p[t].l].r=t;p[p[t].r].l=t;

p[p[t].u].d=t;p[p[t].d].u=t;

}

void delcol(int t)

{ for (int i=p[t].u;i;i=p[t].u)

{ if (i<=m){remove(i);break;}

int l=i;while (p[l].l)l=p[l].l;

for (int j=l;j;j=p[j].r)remove(j);

}

for (int i=p[t].d;i;i=p[t].d)

{ int l=i;while (p[l].l)l=p[l].l;

for (int j=l;j;j=p[j].r)remove(j);

}

}

bool search()

{ if (!p[s].r)return true;

int c=0;

for (int i=p[s].r;i;i=p[i].r)

if (!c||lev[i]<lev[c])c=i;

if (!lev[c])return false;

int cnt=tot;

for (int i=p[c].d;i;i=p[i].d)

{ int l=i;while (p[l].l)l=p[l].l;

for (int j=l;j;j=p[j].r)delcol(j);

for (int j=l;j;j=p[j].r)remove(j);

if (search())return true;

while (tot>cnt)rebuild();

}

return false;

}

void solve()

{ memset(p,0,sizeof(p));

p[s].r=1;p[1].l=s;

rep(i,0,n)

rep(j,1,m-1)

{ p[i\*m+j].r=i\*m+j+1;

p[i\*m+j+1].l=i\*m+j;

}

rep(i,0,n-1)

rep(j,1,m)

{ p[i\*m+j].d=i\*m+j+m;

p[i\*m+m+j].u=i\*m+j;

}

rep(i,1,m)lev[i]=n;

rep(i,1,n)

rep(j,1,m)

if (!data[i][j])remove(i\*m+j);

if (search())printf("Yes, I found it\n");

else printf("It is impossible\n");

}

int main()

{ while (scanf("%d%d",&n,&m)==2)

{ input();

solve();

}

}

**莫队算法：**

#include<cstdio>

#include<cstring>

#include<cstdlib>

#include<iostream>

#include<algorithm>

#include<cmath>

using namespace std;

#define N 50010

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

struct node{

    int l,r,idx;

    long long a,b;

}ask[N],ans[N];

int n,m,mm,tot;

int c[N],s[N],block[N];

bool cmp(node n1,node n2){

    return block[n1.l]<block[n2.l]||(block[n1.l]==block[n2.l]&&n1.r<n2.r);

}

void init(){

    scanf("%d%d",&n,&m);

    rep(i,1,n)scanf("%d",&c[i]);

    rep(i,1,m){

        scanf("%d%d",&ask[i].l,&ask[i].r);

        ask[i].idx=i;

    }

    mm=(int)sqrt(n\*1.0);

    rep(i,1,n)block[i]=(i-1)/mm+1;

    sort(ask+1,ask+1+m,cmp);

}

/\*sigma(S[i]\*(S[i]-1)/2)/[(r-l+1)\*(r-l)/2]

   =sigma(S[i]\*(S[i]-1))/[(r-l+1)\*(r-l)]

   =[sigma(S[i]\*S[i])-(r-l+1)]/[(r-l+1)\*(r-l)]

\*/

long long gcd(long long x,long long y){

    if (!y)return x;

    else return gcd(y,x%y);

}

void caln(int p,int delta){

    tot-=s[c[p]]\*s[c[p]];

    s[c[p]]+=delta;

    tot+=s[c[p]]\*s[c[p]];

}

void solve(){

    int l=1,r=0;tot=0;long long k;

    rep(i,1,m){

        if (r<ask[i].r){

            for (r=r+1;r<=ask[i].r;r++)caln(r,1);

            r--;

        }

        else if (r>ask[i].r){

            for (;r>ask[i].r;r--)caln(r,-1);

        }

        if (l<ask[i].l){

            for (;l<ask[i].l;l++)caln(l,-1);

        }

        else if (l>ask[i].l){

            for (l=l-1;l>=ask[i].l;l--)caln(l,1);

            l++;

        }

        if (l==r){

            ans[ask[i].idx].a=0;ans[ask[i].idx].b=1;

            continue;

        }

        ans[ask[i].idx].a=tot-(r-l+1);

        ans[ask[i].idx].b=(long long)(r-l+1)\*((long long)(r-l));

        k=gcd(ans[ask[i].idx].a,ans[ask[i].idx].b);

        ans[ask[i].idx].a/=k;

        ans[ask[i].idx].b/=k;

    }

    rep(i,1,m)printf("%lld/%lld\n",ans[i].a,ans[i].b);

}

int main(){

    init();

    solve();

    return 0;

}

HW：

**最长上升子序列nlogn算法 ：**

#include<cstdio>

#include<cstring>

#define MAXN 40005

int arr[MAXN],ans[MAXN],len;

/\*

二分查找。 注意，这个二分查找是求下界的; (什么是下界？详情见《算法入门经典》 P145)

即返回 >= 所查找对象的第一个位置（想想为什么）

也可以用STL的lowe\_bound二分查找求的下界

\*/

int binary\_search(int i){

int left,right,mid;

left=0,right=len;

while(left<right){

mid = left+(right-left)/2;

if(ans[mid]>=arr[i]) right=mid;

else left=mid+1;

}

return left;

}

int main() {

int T,p,i,j,k;

scanf("%d",&T);

while(T--){

scanf("%d",&p);

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

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

ans[1] = arr[1];

len=1;

for(i=2; i<=p; ++i){

if(arr[i]>ans[len])

ans[++len]=arr[i];

else{

int pos=binary\_search(i);

// 如果用STL： pos=lower\_bound(ans,ans+len,arr[i])-ans;

ans[pos] = arr[i];

}

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

}

return 0;

}

**最长公共子序列nlogn**

#include<cstdio>

#include<algorithm>

#include<vector>

using namespace std;

const int maxn = 100005;

/\*a,b为两个序列，lis记录a中元素在b中的位置，d[i]为

长度i的上升子序列最后一个元素的值，pos[i][j]记录a中

值为i的元素在b中出现的第j个位置的序号\*/

int a[maxn], b, lis[maxn\*20],d[maxn\*20];

vector<int> pos[maxn];

int main(){

int len\_a, len\_b;

scanf("%d%d", &len\_a, &len\_b);

for (int i = 1; i <= len\_a; i++)

scanf("%d", a + i);

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

scanf("%d", &b);

pos[b+5000].push\_back(i);

}

int len\_lis = 1;

for (int i = 0; i <= len\_a; i++)

for (int k = pos[a[i]+5000].size() - 1; k >= 0; k--)

lis[len\_lis++] = pos[a[i]+5000][k];

d[1] = lis[1];

int max\_len\_lcs = 1;

for (int i = 2; i <= len\_lis; i++){

/\*如果lis[i]大于目前最长子序列的最后一个元素，

把lia[i]附在改序列后边，构成长度+1的最长序列，

同时更新d[]的值.

否则找到d[]中第一个大于lis[i]的位置k，将lis[i]

附在长度为k-1的序列后边，即变为长度为k的序列，因此

更新d[k]的值为lis[i]\*/

if (lis[i] > d[max\_len\_lcs])

d[++max\_len\_lcs] = lis[i];

else {

int pos\_greater\_than\_lis\_i = lower\_bound(d, d + max\_len\_lcs, lis[i]) - d;

d[pos\_greater\_than\_lis\_i] = lis[i];

}

}

printf("%d\n", max\_len\_lcs);

return 0;

}

**完全背包：**

/\*

一个背包总容量为V，现在有N个物品，第i个 物品体积为weight[i]，

价值为value[i]，每个物品都有无限多件，现在往背包里面装东西，

怎么装能使背包的内物品价值最大？

\*/

#include<iostream>

using namespace std;

#define V 1500

unsigned int f[V];//全局变量，自动初始化为0

unsigned int weight[10];

unsigned int value[10];

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

int main()

{

int N,M;

cin>>N;//物品个数

cin>>M;//背包容量

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

{

cin>>weight[i]>>value[i];

}

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

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

{

if (weight[i]<=j)

{

f[j]=max(f[j],f[j-weight[i]]+value[i]);

}

}

cout<<f[M]<<endl;//输出最优解

}

**多重背包：**

**/\***

**一个背包总容量为M，现在有6个物品，第i个 物品体积为w[i]，**

**价值为v[i]，每个物品都有A[i]，现在往背包里面装东西，**

**怎么装能使背包的内物品价值最大？**

**\*/**

#include<iostream>

#include<cstring>

using namespace std;

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

#define dto(i,u,v) for (int i=(u);i>=(v);i--)

const int w[]={3,5,2,6,11,8};

const double v[]={0.01,0.05,0.10,0.25,0.50,1.00};

int a[6];

double f[10010];

int main(){

int M;

while (cin>>M){

rep(i,0,5)cin>>a[i];

memset(f,0,sizeof(f));

rep(i,0,5){

if (a[i]\*w[i]>M){

rep(j,w[i],M)

f[j]=max(f[j],f[j-w[i]]+v[i]);

}

else {

int k=1;

while (k<a[i]){

dto(j,M,k\*w[i])

f[j]=max(f[j],f[j-k\*w[i]]+k\*v[i]);

a[i]-=k;

k\*=2;

}

dto(j,M,a[i]\*w[i])

f[j]=max(f[j],f[j-a[i]\*w[i]]+a[i]\*v[i]);

}

}

printf("$%.2lf\n",f[M]);

}

return 0;

}

**TSP：**

/\*

有编号1到N的N个城市，问从1号城市出发，

遍历完所有的城市并最后停留在N号城市的最短路径长度。

\*/

#include <bits/stdc++.h>

const double INF=10e7;

using namespace std;

int T,n,cnt;

double a[25][25],dp[25][1100000];

struct point{ //结点结构体

int x,y;

}pt[25];

double d(point a,point b){ //结点间距离

return sqrt((a.x-b.x)\*(a.x-b.x)+(a.y-b.y)\*(a.y-b.y));

}

int main() {

scanf("%d",&T);

while(T--){

cnt=1;

scanf("%d",&n);

for(int i=2;i<n;i++) cnt<<=1; //组合数（除起点终点外）

for(int i=0;i<n;i++) //输入

scanf("%d %d",&pt[i].x,&pt[i].y);

for(int i=0;i<n;i++) //建边

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

a[i][j]=d(pt[i],pt[j]);

for(int i=0;i<n;i++) //初始化

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

dp[i][j]=INF;

for(int i=0;i<n;i++) //起点确定，定下初始条件

dp[i][0]=a[i][0];

for(int i=1;i<cnt;i++) //从有元素考虑起

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

for(int k=1;k<n-1;k++) {

if((1<<k-1)&i) //k is in the set

dp[j][i]=min(dp[j][i],a[j][k]+dp[k][i-(1<<k-1)]); //状态转移方程

}

}

double ans=INF;

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

ans=min(ans,dp[i][cnt-1]+a[i][n-1]);

printf("%.2lf\n",ans);

}

return 0;

}

**Bellman\_ford：**

#include <iostream>

using namespace std;

const int maxnum = 110;

const int maxint = 100000000;

// 边，

typedef struct Edge{

int u, v; // 起点，重点

int weight; // 边的权值

}Edge;

Edge edge[4002]; // 保存边的值

int dist[maxnum]; // 结点到源点最小距离

int nodenum, edgenum, source; // 结点数，边数，源点

// 初始化图

void init()

{

// 输入结点数，边数，源点

cin >> nodenum >> edgenum;

source=1;

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

dist[i] = maxint;

dist[source] = 0;

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

{

cin >> edge[i].u >> edge[i].v >> edge[i].weight;

if(edge[i].u == source) //注意这里设置初始情况

dist[edge[i].v] = edge[i].weight;

}

}

// 松弛计算

void relax(int u, int v, int weight){

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

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

}

bool Bellman\_Ford()

{

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

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

relax(edge[j].u, edge[j].v, edge[j].weight);

bool flag = 1;

// 判断是否有负环路

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

if(dist[edge[i].v] > dist[edge[i].u] + edge[i].weight)

{

flag = 0;

break;

}

return flag;

}

int main(){

init();

if(Bellman\_Ford()){

if (dist[nodenum] ==maxint )cout<<-1<<endl;

else cout << dist[nodenum] << endl;

}

else {

if (dist[nodenum] ==maxint )cout<<-1<<endl;

else cout<<0<<endl;

}

return 0;

}

**网络流pannic room：**

#include<iostream>

#include<cstring>

#include<cstdio>

#include<cstdlib>

using namespace std;

#define INF 0x5fffffff

struct node

{ int y,next,c,oth;

}edge[1500];

int len,n,ed,st,first[22],h[22],list[32];

void ins(int x,int y,int c)

{ int k1=++len,k2=++len;

edge[k1].y=y;edge[k1].c=c;edge[k1].next=first[x];

edge[k1].oth=k2;first[x]=k1;

edge[k2].y=x;edge[k2].c=0;edge[k2].next=first[y];

edge[k2].oth=k1;first[y]=k2;

}

bool bfs()

{ int x,y,k,head,tail;

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

list[1]=st;h[st]=0;

for (head=tail=1;head<=tail;head++){

x=list[head];

for (k=first[x];k!=-1;k=edge[k].next){

y=edge[k].y;

if (edge[k].c&&h[y]==-1){

h[y]=h[x]+1;

list[++tail]=y;

}

}

}

return h[ed]!=-1;

}

int dfs(int x,int flow){

int ff,minf,k,y;

if (x==ed)return flow;ff=0;

for (k=first[x];k!=-1;k=edge[k].next)

{ y=edge[k].y;

if (edge[k].c&&h[y]==h[x]+1&&ff<flow&&(minf=dfs(y,min(edge[k].c,flow-ff))))

{ edge[k].c-=minf;ff+=minf;

edge[edge[k].oth].c+=minf;

}

}

if (!ff)h[x]=-1;

return ff;

}

int main()

{ int cases,i,ans,tt,x,j,k;char s[5];

scanf("%d",&cases);

while (cases-->0){ans=0;len=0;

scanf("%d%d",&n,&ed);st=0;ed++;

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

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

scanf("%s",s);

if (s[0]=='I')ins(st,i,INF);

scanf("%d",&k);

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

scanf("%d",&x);

ins(i,x+1,INF);

ins(x+1,i,1);

}

}

while (bfs())

while (tt=dfs(st,INF))ans+=tt;

if (ans>=INF)printf("PANIC ROOM BREACH\n");

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

}

}

**部分背包：**

/\*

问题描述：n件物品，第i件物品价值 vi 元，重wi 磅。

希望用 W磅的背包 拿走最重的物品。第i件物品可以都拿走，

也可以拿走一部分。（物品可以分割所以称为部分背包）

\*/

#include<cstdio>

#incldue<iostream>

#include<algorithm>

#include<cstdlib>

using namespace std;

int main(){

int w[1000],v[1000];//w为质量，v为价值，r为价值与质量的比

float r[1000];//r为价值与质量的比

int n;//物品的数量

int m; //背包的容量

scanf("%d %d",&m,&n); //输入背包总量和数量

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

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

r[i]=v[i]\*1.0/w[i];

}

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

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

if(r[j]<r[j+1]){

swap(r[j],r[j+1]);

swap(w[j],w[j+1]);

swap(v[j],v[j+1]);

}

}

}

int i=0;

while(m>0){

if(w[i]<=m){

m-=w[i];

printf("价值：%d取：%d\n",v[i],w[i]);

i++;

}

else{

printf("价值：%d取：%d\n",v[i],m);

m=0;

}

}

return 0;

}

**矩阵连乘计数：**

#include<stdlib.h>

#include<stdio.h>

#include <memory.h>

#define N 6

void Print\_OPTIMAL\_PARENS(int s[N+1][N+1],int i,int j) //定义函数打印最优全括号的结果

{

if(i==j)

printf("A%d",i);

else

{

printf("(");

Print\_OPTIMAL\_PARENS(s,i,s[i][j]); //在分裂处进行递归调用

Print\_OPTIMAL\_PARENS(s,s[i][j]+1,j);

printf(")");

}

}

int main()

{

int matrix[N+1]; //matrix中记录矩阵的维数

int i,j,k,q;

int m[N+1][N+1]; //m中记录矩阵连乘的次数

int s[N+1][N+1]; //s[i][j]中记录了对Ai...Aj进行分裂的最优的k值

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

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

memset(m,0,(N+1)\*(N+1)\*sizeof(int));

for(j=1;j<=N;j++)

for (i=j;i>=1;i--) //当i=j时,m[i][j]=0，

{ //当i<j时,m[i][j]=min{m[i][k]+m[k+1][j]+p(i-1)p(k)p(j)} i=<k<j

if (j==i)

m[i][j]=0;

else

{

m[i][j]=600000;

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

{

q=m[i][k]+m[k+1][j]+matrix[i-1]\*matrix[k]\*matrix[j];

if (q<m[i][j])

{

m[i][j]=q;

s[i][j]=k;

}

}

}

}

printf("%d/n",m[1][N]);

Print\_OPTIMAL\_PARENS(s,1,N);

return 0;

}

void MatrixChain()

{

int i, j, k, t;

////////此为初始化底层，即一个矩阵的情况///////////

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

m[i][i] = 0;//赋值为0，是因为1个矩阵需做0次相乘

for(int r = 2; r <= n; r++)

{//计算r个矩阵连乘的情况

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

{//计算从i个矩阵开始的连续r个矩阵相乘的最少次数

j = i + r - 1;//A[i : j],连续r个矩阵

////////////以下为其中一种情况，断开点i，即第一个矩阵独立/////////////////

m[i][j] = m[i + 1][j] + p[i - 1] \* p[i] \* p[j];

////////////////开始寻找最优值///////////////////////

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

{

t = m[i][k] + m[k + 1][j] + p[i - 1] \* p[k] \* p[j];

if( t < m[i][j])

m[i][j] = t;

}

}

}

}

**实验课：**

**1.拥有n根火柴，求出可以摆出的最大和最小的数**

#include<iostream>

using namespace std;

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

string finmin(int x){

string s="";

int num[10] = {6, 2, 5, 5, 4, 5, 6, 3, 7, 6};

if (x<=7){

rep(i,1,9)

if (x==num[i]){

s+=char(i+48);

break;

}

}

else {

int eight=x/7;

x%=7;

if (x){

int tem;

if (x==1)tem=9;

else{

rep(i,1,9)

if (x==num[i]){

tem=i;

break;

}

}

bool start=true;

while (eight){

bool flag = true;

int i;

if (start) i = 1;

else i = 0;

start=false;

for (i;i<=tem;i++){

rep(j,0,7)

if (num[i]+num[j]==x+num[8]){

s+=char(i+48);

tem=j;

flag=false;

eight--;

break;

}

if (!flag)break;

}

if (flag){

s+=char(tem+48);

break;

}

x=num[tem];

if (!eight){

s+=char(tem+48);

break;

}

}

}

rep(i,1,eight)

s+="8";

}

return s;

}

string finmax(int x){

string s="";

if (x%2==1){

x-=3;

s="7";

}

rep(i,1,x/2)s+="1";

return s;

}

int main(){

int T,n;

cin>>T;

rep(t,1,T){

cin>>n;

string minn=finmin(n);

string maxx=finmax(n);

cout<<minn<<" "<<maxx<<endl;

}

return 0;

}

**2.拥有n根火柴，求出可以摆出的最大和最小的数**

一开始你有N个农民和M单位的资源，每个农民一秒钟可  
以得到C单位的资源，每个农民可以用P单位的资源立即生产，周  
围资源没有限制，没有人口限制（开了作弊？） ,而且农民采集资  
源的过程是离散的(例如不会在0.5秒的时候收入0.5C),问在S秒后所  
收集的最大资源是多少（不包括已经花出去的）。

#include<iostream>

using namespace std;

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

int main(){

int T,N,M,C,P,S;

cin>>T;

while (T--){

cin>>N>>M>>C>>P>>S;

rep(i,1,S){

if (M>=P&&C\*(S-i+1)>=P){

N+=M/P;

M%=P;

}

M+=N\*C;

}

cout << M << endl;

}

return 0;

}

**3. 要求你构造一个由字符'A', 'B'组成的字符串, 满足以下几个条件:  
• 1) A的个数<=countA  
• 2) B的个数<=countB  
• 3) 连续的A的个数不可以超过maxA.  
• 4) 连续的B的个数不可以超过maxB.  
• 5) 这个字符串的长度最长.  
• 给你countA,countB,maxA,maxB,要求你输出字符串的最大长度.**

#include<iostream>

#include<algorithm>

#include<cmath>

using namespace std;

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

int main(){

int countA,countB,maxA,maxB,ans;

cin>>countA>>countB>>maxA>>maxB;

if (maxA==0 && maxB==0)cout<<0<<endl;

else if (maxA==0)cout<<min(countB,maxB)<<endl;

else if (maxB==0)cout<<min(countA,maxA)<<endl;

else {

int blocka=ceil((double)countA/maxA);

int blockb=ceil((double)countB/maxB);

ans=0;

if(countA<blockb)ans+=countA+(countA+1)\*maxB;

else if (countB<blocka)ans+=countB+(countB+1)\*maxA;

else ans=countA+countB;

cout<<ans<<endl;

}

return 0;

}

**4. 给出N， M，问有多少个长度为N的整数序列，**

**满足所有数都在[1,M]内，并且每一个数至少**

**是前一个数的两倍**

#include<iostream>

#include<cstdio>

#include<cstring>

using namespace std;

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

int dp[31],sum[31];

int main(){

int n;

dp[0]=1;

sum[0]=dp[0]\*2;

dp[2]=3;

sum[2]=dp[2]\*2+sum[0];

rep(i,3,30){

if (i%2==1){//n为奇数无方案

dp[i]=0;

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

}

else {//n为偶数

dp[i]=dp[i-2]\*dp[2]+sum[i-4];

sum[i]=sum[i-1]+dp[i]\*2;

}

}

while (cin>>n){

if (n==-1)break;

cout<<dp[n]<<endl;

}

return 0;

}

**5. 能量项链**

**给出一串项链，每次可以选相邻两个珠子进行聚合，释放**

**出一定的能量，并产生一个新珠子，项链是头尾相接的，求释放的能量的总和的最大值**

#include<iostream>

#include<cstring>

using namespace std;

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

int a[210];

int f[210][210];

int dp(int l,int r){

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

if (l+1==r)return f[l][r]=a[l]\*a[l+1]\*a[r+1];

int maxx=0;

rep(i,l,r-1){

int tmp=dp(l,i)+dp(i+1,r)+a[l]\*a[i+1]\*a[r+1];

maxx=max(maxx,tmp);

}

return f[l][r]=maxx;

}

int main(){

int n;

while (cin>>n){

rep(i,1,n){

cin>>a[i];

a[i+n]=a[i];

}

int ans=0;

memset(f,0,sizeof(f));

rep(i,1,n)

ans=max(ans,dp(i,i+n-1));

cout<<ans<<endl;

}

return 0;

}

**6.给出两个集合S1和S2，**

**在S2中选出一些不重复的数与S1的每个数匹配，使得匹配**

**的数的差的绝对值之和尽量小**

#include<iostream>

#include<algorithm>

#include<cmath>

using namespace std;

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

int s1[502],s2[502],dp[502][502];

int main(){

int T,n,m;

cin>>T;

while (T--){

cin>>n>>m;

rep(i,1,n)cin>>s1[i];

rep(i,1,m)cin>>s2[i];

sort(s1+1,s1+1+n);

sort(s2+1,s2+1+m);

rep(i,1,n)

rep(j,i,m){

if (i==j)dp[i][i]=dp[i-1][i-1]+abs(s1[i]-s2[i]);

else dp[i][j]=min(dp[i-1][j-1]+abs(s1[i]-s2[j]),dp[i][j-1]);

}

cout<<dp[n][m]<<endl;

}

return 0;

}

**7. 过河**

**桥的起点为0，终点为L，其中地有M个石子**

**青蛙每次跳的范围为[S,T]，问要跳过桥最小踩到石子次数**

**限制1<=L<=10^9 1<=S<=T<=10 1<=M<=100**

#include<iostream>

#include<algorithm>

#include<cstring>

using namespace std;

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

int w[110];

int dp[11000],bo[11000];

int main(){

int L,S,T,M;

cin>>L>>S>>T>>M;

rep(i,1,M)cin>>w[i];

sort(w+1,w+1+M);

if (S==T){//单独处理S==T的情况

int ans=0;

rep(i,1,M)

if (w[i]%S==0)ans++;

cout<<ans<<endl;

}

else {

int k=S\*T;

w[0]=0;w[M+1]=L;

rep(i,1,M+1){

if (w[i]-w[i-1]>k){

int delta=w[i]-w[i-1]-k;

rep(j,i,M+1)

w[j]-=delta;

}

bo[w[i]]=1;

}

bo[w[M+1]]=0;

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

dp[0]=0;

rep(i,0,w[M+1]){

if (dp[i]==-1)continue;

rep(j,i+S,i+T){

if (dp[j]==-1||dp[i]+bo[j]<dp[j])

dp[j]=dp[i]+bo[j];

}

}

int ans=110;

rep(i,w[M+1],w[M+1]+T)ans=min(ans,dp[i]);

cout<<ans<<endl;

}

return 0;

}

**7. tour**

**给定平面上 n 个点(n<=100)，求从最左边的点到最右边的点，再回到最左边的点的最短路程，中间不得经过重复的点且 n 个点都要经过。**

#include<iostream>

#include<cstring>

#include<cstdio>

#include<cmath>

#include<algorithm>

using namespace std;

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

#define INF 100000000.0

struct node{

double x,y;

}a[110];

int n;

double dp[110][110];

bool cmp(const node &n1,const node &n2){

return n1.x<n2.x;

}

double dist(node &n1,node &n2){

double x=n1.x-n2.x;

double y=n1.y-n2.y;

return sqrt(x\*x+y\*y);

}

int main(){

while(cin>>n){

rep(i,1,n)

cin>>a[i].x>>a[i].y;

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

rep(i,0,n)

rep(j,0,n)

if (!i || !j)dp[i][j]=0;

else dp[i][j]=INF;

dp[1][1]=0;//起点到自己是0

rep(i,2,n)dp[1][i]=dp[i][1]=dp[i-1][1]+dist(a[i],a[i-1]);

//上面这步是设置边界，即一个人走完全程

rep(i,1,n)//枚举第一个人走的点

rep(j,1,i){//枚举第二个人走的点

if(i>j+1)dp[i][j]=dp[j][i]=dp[i-1][j]+dist(a[i],a[i-1]);

else if (i==j+1){

rep(k,1,j-1)//枚举第一个人是从k点走到i点的

dp[i][j]=dp[j][i]=min(dp[i][j],dp[k][j]+dist(a[i],a[k]));

}

else{//i==j

rep(k,1,j-1)

dp[i][j]=dp[j][i]=min(dp[i][j],

dp[i-1][k]+dist(a[i],a[i-1])+dist(a[k],a[j]));

}

}

printf("%.2lf\n",dp[n][n]);

}

return 0;

}

**8. 国王的遗产**

有一个国王拥有一个n个金块组成的树，在

他死后由他的k个儿子轮流分金块。

• 每个人可以选择一条边将它断开，然后选

择金块数量少的那一块，如果金块数量相

同，则选择剩余编号小的金块所在的那一

块。求在每个人获得尽量多的金块条件下，每个人获得几个金块，按照拿的顺序输出。

#include<iostream>

#include<vector>

#include<cstdio>

using namespace std;

#define N 30010

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

struct node{

vector<int>child;

int minID,sizes;

}t[N];

int n,k,tot,ans,root,st,cnt=0,start\_from\_root,minID;

int bo[N];

vector<int>g[N];

void build\_tree(int x){

bo[x]=-1;

t[x].child.clear();

t[x].sizes=1;t[x].minID=x;

int len=g[x].size();

rep(i,0,len-1){

int y=g[x][i];

if (!bo[y]){

build\_tree(y);

t[x].child.push\_back(y);

t[x].sizes+=t[y].sizes;

t[x].minID=min(t[x].minID,t[y].minID);

}

}

bo[x]=0;

}

void dfs(int x){

//深搜寻找取哪一部分，start\_from\_root=1表示取切掉的边的靠近

//根节点那部分start\_from\_root=0表示取切掉的边远离根结点那边

//st记录的是被切那条边远离根结点的那个结点

int len=t[x].child.size();

rep(i,0,len-1){

int y=t[x].child[i];

int res=tot-t[y].sizes;

if (t[y].sizes>=res){//取靠近根结点那部分

if (res<ans)continue;

else if (res>ans||res==ans&&t[y].minID>t[st].minID){

ans=res;

start\_from\_root=1;

minID=root;

st=y;

}

}

else {//取远离根结点那部分

if (t[y].sizes<ans)continue;

else if (t[y].sizes>ans||(!start\_from\_root&&t[y].minID<minID)){

ans=t[y].sizes;

start\_from\_root=0;

minID=t[y].minID;

st=y;

}

}

dfs(y);

}

}

void del(int x, int v){

int len=t[x].child.size();

rep(i,0,len-1){

int y=t[x].child[i];

del(y,v);

}

bo[x]=v;

}

int main(){

int x,y;

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

rep(i,1,n-1){

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

g[x].push\_back(y);

g[y].push\_back(x);

}

rep(i,1,k-1){

root=1;

//用还没被拿走的编号最小的金块为根节点建树

while (bo[root] && root<n)root++;

build\_tree(root);

///////////////////////////////

//////////////////深搜寻找切哪条边

tot=t[root].sizes;

ans=0;

dfs(root);

///////////////////

//\*\*删去被拿走的金块

if (start\_from\_root)del(root,1),del(st,0);//删除靠近根那部分

else del(st,1);//删除远离根那部分

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

printf("%d ", ans);

cnt+=ans;

}

printf("%d\n",n-cnt);

return 0;

}

**9. 新红黑树**

一棵树由红枝和黑枝组成的树， A和B轮流砍树， A只砍红

枝， B只砍黑枝 砍枝后不与根相连的枝都去掉。每个树枝上有权值，砍掉的枝的权值加到自己的分数上A想使A-B之差越高越好， B想它越低越好。在最佳策略下A-B之差

#include<iostream>

#include<cstdio>

#include<algorithm>

#include<cstring>

using namespace std;

#define INF 0x3f3f3f3f

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

int n;

int f[1<<20][2];

int c[22][22],w[22][22],pre[22],bo[22],sta[22];

void dfs(int x){//预处理砍掉某条边后会消失哪条边

bo[x]=1;

if (x>0)sta[x]|=(1<<(x-1));

rep(i,1,n){

if(!bo[i]&&w[x][i]!=0){

dfs(i);

sta[x]|=sta[i];

pre[i]=x;

}

}

}

int dp(int s,int turn){

if (!s)return 0;

int k=(turn+1)/2;

if (f[s][k]!=INF)return f[s][k];//搜过的状态不再搜

int maxx=-INF,minn=INF;

rep(i,1,n){

if (s&(1<<(i-1))){

if (turn!=c[pre[i]][i])

continue;

int temp=dp(s&~sta[i],-turn);

maxx=max(maxx,w[pre[i]][i]+temp);//A砍

minn=min(minn,w[pre[i]][i]+temp);//B砍

}

}

if(turn==1&&maxx==-INF)maxx=dp(s,-turn);//A没得砍

if(turn==-1&&minn==INF)minn=dp(s,-turn);//B没得砍

if (turn==1)f[s][k]=maxx;

else f[s][k]=minn;

return f[s][k];

}

int main(){

int x,y,color,weight;

cin>>n;

rep(i,1,n){

cin>>x>>y>>color>>weight;

c[x][y]=c[y][x]=color;

w[x][y]=w[y][x]=weight\*color;

}

pre[0]=-1;

dfs(0);

memset(f,INF,sizeof(f));

cout<<dp((1<<n)-1,1)<<endl;

return 0;

}

**10.** **给你n个电话号码，问你是否存在一个电话**

**号码是另一个电话号码的前缀**

#include<iostream>

#include<cstring>

using namespace std;

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

#define N

struct node{

int flag;

int ch[10];

void init(){

memset(ch,0,sizeof(ch));

flag=0;

}

}t[101000];

int flag,m;

void insert(char \*s){

int p=0;

int len=strlen(s);

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

int x=s[i]-'0';

if (!t[p].ch[x]){

t[p].ch[x]=++m;

t[t[p].ch[x]].init();

}

p=t[p].ch[x];

if (t[p].flag==2){//当前点是某个字符串的终点

flag=0;

return;

}

else if(t[p].flag==1&&i==len-1){

//当前点是当前字符串的终点，且同时被另一个字符串经过

flag=0;

return;

}

t[p].flag=1;

}

t[p].flag=2;

}

int main(){

int T,n;

char s[15];

cin>>T;

rep(tt,1,T){

cin>>n;

flag=1;m=0;

t[0].init();

rep(i,1,n){

cin>>s;

insert(s);

}

if (flag)cout<<"YES"<<endl;

else cout<<"NO"<<endl;

}

return 0;

}

**11. countdown**

#include<iostream>

#include<cstring>

#include<algorithm>

#include<map>

#include<vector>

using namespace std;

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

int n,m,d;

int root[1002],sizes[1002];

map<string,int>num;

map<int,string>name;

vector<int>g[1002];

vector<pair<string,int> >ans;

bool cmp(pair<string,int>n1,pair<string,int>n2){

return n1.second>n2.second ||

n1.second==n2.second && n1.first<n2.first;

}

void dfs(int now,int x,int dai){

sizes[now]=0;

for (int i=0;i<g[now].size();i++){

if (dai+1==d)sizes[x]++;

else {

int y=g[now][i];

dfs(y,x,dai+1);

}

}

}

int main(){

int T,k;

cin>>T;

string s1,s2;

rep(tt,1,T){

cin>>n>>d;

num.clear();name.clear();

ans.clear();m=0;

memset(root,0,sizeof(root));

rep(i,1,n){

cin>>s1>>k;

int &x=num[s1];

if (!x){

x=++m;

name[x]=s1;

g[x].clear();

}

rep(j,1,k){

cin>>s2;

int &y=num[s2];

if (!y){

y=++m;root[y]=1;

name[y]=s2;

g[y].clear();

}

g[x].push\_back(y);

}

}

rep(i,1,m){

dfs(i,i,0);

if (sizes[i]>0)

ans.push\_back(make\_pair(name[i],sizes[i]));

}

if (tt>1)cout<<endl;

cout<<"Tree "<<tt<<":"<<endl;

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

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

if (i>2&&ans[i].second!=ans[i-1].second)break;

cout<<ans[i].first<<" "<<ans[i].second<<endl;

}

}

return 0;

}

**12. 1709 PropBot一开始机器人在点(0,0)上，指向+x方**

**向。机器人每秒有两种行动方法：向当前方向走10个单位长度，或者向右转45度。现给出目标点以及最多可以行动的时间，求出在这个过程中，机器人可以达到的离目标点最近的距离，（可以自由选择机器人的行动方法）**

#include <iostream>

#include <algorithm>

#include <stdio.h>

#include <string.h>

#include <math.h>

using namespace std;

double x,y,dl;

int tme,dir;

double Min;

double dis(double dx,double dy)

{

return sqrt( ( x - dx ) \* ( x - dx ) + ( y - dy ) \* ( y - dy ));

}

void DFS(int dir,int tim,double dx,double dy)

{

double d = dis(dx,dy);

if(d < Min)

Min = d;

if(tim == tme || Min < (d - 10 \* (tme - tim)))

return ;

dir %= 8;

if(dir == 0)

DFS(dir,tim+1,dx + 10,dy);

else if(dir == 1)

DFS(dir,tim+1,dx + dl,dy - dl);

else if(dir == 2)

DFS(dir,tim+1,dx,dy - 10);

else if(dir == 3)

DFS(dir,tim+1,dx - dl,dy - dl);

else if(dir == 4)

DFS(dir,tim+1,dx - 10,dy);

else if(dir == 5)

DFS(dir,tim+1,dx - dl,dy + dl);

else if(dir == 6)

DFS(dir,tim+1,dx,dy + 10);

else if(dir == 7)

DFS(dir,tim+1,dx + dl,dy + dl);

DFS(dir+1,tim+1,dx,dy);

}

int main()

{

int t;

scanf("%d",&t);

while(t--)

{

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

dl = 5.0\*sqrt(2.0);

Min = dis(0,0);

DFS(0,0,0,0);

printf("%.6lf\n",Min);

}

return 0;

}

**13Context-Free Clock**

题意：给出一个时刻t和角度theta，求时刻ans，使

得在ans时刻中，时针按顺时针到分针的角度为

theta，且该时刻离t最近。得到ans后将ans按照秒截

断后输出（例如答案是12:34:45’54的话，输出

12:34:35）

#include<stdio.h>

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

double degree[86401];

double abs(double x){return x<0?-x:x;}

void prepare(){

double hourdegree,minutedegree,d;

rep(i,0,23){

rep(j,0,59){

rep(k,0,59) {

hourdegree=(i % 12) \* 30 + (60.0 \* j + k) / 120;

minutedegree=j \* 6 + 1.0 \* k / 10;

d=hourdegree>minutedegree ? 360-hourdegree+minutedegree : minutedegree-hourdegree;

degree[i \* 3600 + j \* 60 + k] = d;

}

}

}

}

const double eps = 0.08;

bool check(double d1, double d2) {

if (d1 == 0) {

return abs(d1-d2)<eps || abs(360.0-d2)<eps;

}

return abs(d1-d2)<eps;

}

int main(){

int h, m, s;

double A;

prepare();

while (scanf("%lf %d:%d:%d", &A, &h, &m, &s)!=EOF){

if (A==-1)break;

int st=h\*3600+m\*60+s;

rep(i,st,86399){

if (check(A,degree[i])){

printf("%02d:%02d:%02d\n",i/3600,i%3600/60,i%3600%60);

break;

}

if (i==86399)i=-1;

}

}

return 0;

}

**14Soj 1203 The Cubic End**

**题目给出了一个有趣的现象：如果一个数字串，以1， 3， 7， 9结尾，则会有一个数，它的三次方以这个数字串结尾，且长度不会超过这个数字串。现在给出一个数字串，找到一个数的三次方以这个数字串结尾**

#include<iostream>

#include<cstring>

using namespace std;

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

#define dto(i,u,v) for (int i=(u);i>=(v);i--)

int a[15],b[15];

int main(){

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

int T,n;

char s[15];

cin>>T;

rep(tt,1,T){

cin>>s;

n=strlen(s);

rep(i,0,n-1)a[i]=s[n-1-i]-'0';

int carry=0;

rep(i,0,n-1){

rep(j,0,9){

int tmp=carry;

b[i]=j;

rep(x,0,i)

rep(y,0,i){

if (x+y>i)break;

int z=i-x-y;

tmp+=b[x]\*b[y]\*b[z];

}

if (tmp%10==a[i]){

carry=tmp/10;

break;

}

}

}

while (b[n-1]==0&&n-1>0)n--;

dto(i,n-1,0)cout<<b[i];

cout<<endl;

}

return 0;

}

**15 Soj 1203 The Cubic End**

**介绍了一种加密算法：将明文（要加密的信息）从左到右一列一列地写入给定列数的矩形中，再添加字符‘x’将它补满，接着从上到下一行一行蛇形(先从左到右读，然后下一行从右到左读，交替进行)地读出来形成密文（加密后的信息）。现在给你密文，要将它翻译成明文。**

#include<iostream>

#include<cstring>

using namespace std;

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

int main(){

int r,c,len;

string str;

while(cin>>c&&c){

char a[201][201];

cin>>str;

len=str.length();

r=len/c;

int x=0,y=0;//x和y表示当前填到x行y列

rep(i,0,len-1){

if (x%2==0){//从左到右填

a[x][y]=str[i];

if (++y==c){

x++;y=c-1;

}

}

else {//从右到左填

a[x][y]=str[i];

if (--y==-1){

x++;y=0;

}

}

}

rep(j,0,c-1)

rep(i,0,r-1)

cout<<a[i][j];

cout<<endl;

}

return 0;

}

**16 Soj 1036 Crypto Columns**

**题意：一个字符串，去掉空格和符号后变成"plaintext"。另一个字符串叫"keyword"。将plaintext排成多行，每行有keyword.size的宽度，不能排满一列的用别的字符填满，生成了字符矩阵。每次选取keyword中字典序最小的字符对应的列，作为需要选取的矩阵的列，被选择过的字符不再被选择，最后形成一行新的字符串，作为输出字符串。先给出keyword和密文，求明文。**

**约束：keyword长度n<=10，密文长度m<=100**

#include<cstdio>

#include<cstring>

#include<cstdlib>

#include<iostream>

using namespace std;

char keys[11];

char flag[11];//±íÊ¾ÕâÒ»ÁÐÊÇ·ñÒÑ¾­±»´¦Àí¹ý

char line[101];//ÃÜÎÄ

char plaintext[101];//Ã÷ÎÄ

void Decrypt(){//½âÃÜ

int i,index,column, count,keysLength, lineLength;

char c;

memset(flag, 0, sizeof(flag));

lineLength=strlen(line);//ÃÜÎÄ×Ü×Ö·ûÊý

keysLength=strlen(keys);//ÁÐÊý

count=lineLength/keysLength;

plaintext[lineLength]=0;

//ÕÒ³ö×îÐ¡×ÖÄ¸

for(column=0;column<keysLength;column++){

c='Z'+1;

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

if(flag[i]==0 && keys[i]<c)

c=keys[i],index=i;

flag[index]=1;

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

//Ã÷ÎÄµÄµÚindexÁÐ

plaintext[keysLength\*i + index]=line[count\*column+i];

}

}

}

int main(){

while(1){

gets(keys);

if(strcmp(keys,"THEEND")==0) break;

gets(line);

Decrypt();

printf("%s\n", plaintext);

}

}

**17 Soj 1028 Hanoi Tower Sequence**

**题意：汉诺塔问题，将从上到下的方向从小到大排列的盘子，从第一个柱子中移动到另一个柱子，其中可以借助第三个柱子，并且每次移动后每根柱子从上到下方向柱子大小总是从小到大的。现在给定了一个移动规则，问第p个移动的盘子编号为多少**

**约束: 1<=p<=10^100**

#include <iostream>

#include <string>

using namespace std;

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

int calc(string ss){

int len=(int)ss.length(),sum=0;

rep(i,0,len-1)ss[i] = ss[i] - '0';

while (ss[len-1]%2==0) { //不断/2

rep(i,0,len-1){

if (i+1<len)ss[i+1]+=(ss[i]%2)\*10;

ss[i]=ss[i]/2;

}

sum++;

}

return sum + 1;

}

int main(){

int T,cases=0;

string s;

cin>>T;

while (cases!=T) {

cin>>s;

cases++;

cout<<"Case "<<cases<< ": "<<calc(s)<<endl;

if (cases!=T)cout << endl;

}

return 0;

}

**18 Stacking Cylinders**

**题意：给出最底层的n个圆柱的圆心位置，圆柱半径都为1，保证相邻两个圆柱的圆心距离在最少为2（so the cylinders do not overlap），最多为3.4(so cylinders at level k cannot touch cylinders at level k - 2)。求最顶层的圆柱的位置。一个圆柱要在上一层，它下一层必须有两个圆柱托住它。输出最顶层圆柱圆心位置包含两个浮点数x，y，x是横坐标，y是高度（最底层那些圆柱圆心的高度是1）。**

#include<iostream>

#include<cstdio>

#include<cmath>

#include<algorithm>

using namespace std;

#define rep(i,u,v) for (int i=(u);i<=(v);i++)

struct node{

double x,y;

node(){}

}p[100];

int n;

bool cmp(const node &n1,const node &n2){return n1.x<n2.x;}

double distances(node &n1, node &n2){

double x=n1.x-n2.x,y=n1.y-n2.y;

return sqrt(x\*x+y\*y);

}

void calc(node &n1,node &n2){

n++;

double len=distances(n1,n2);

double mid=sqrt(4.0-len\*len/4);

double midx=(n1.x+n2.x)/2.0;

double midy=(n1.y+n2.y)/2.0;

double sin\_=(n1.y-n2.y)/len;

double cos\_=(n2.x-n1.x)/len;

p[n].x=sin\_\*mid+midx;

p[n].y=cos\_\*mid+midy;

}

int main(){

int T;

cin>>T;

rep(t,1,T){

cin>>n;

if (!n)break;

rep(i,1,n){

cin>>p[i].x;

p[i].y=1.0;

}

sort(p+1,p+1+n,cmp);

int l=1,r=n;

while (r-l){

rep(i,l,r-1)calc(p[i],p[i+1]);

l=r+1;r=n;

}

printf("%d: %.4lf %.4lf\n",t,p[l].x,p[l].y);

//printf("%.4lf %.4lf\n",p[l].x,p[l].y);

}

return 0;

}