目录

[目录 1](#_Toc495217832)

[矩阵乘法 快速幂 2](#_Toc495217833)

[矩阵运算 高斯消元 4](#_Toc495217834)

[中国剩余定理 8](#_Toc495217835)

[欧拉函数 9](#_Toc495217836)

[博弈论 威佐夫博弈 10](#_Toc495217837)

[直角坐标 11](#_Toc495217838)

[空间坐标 18](#_Toc495217839)

[两圆香蕉 24](#_Toc495217840)

[全排列 26](#_Toc495217841)

[快速幂 27](#_Toc495217842)

[树状数组 28](#_Toc495217843)

[离散化 30](#_Toc495217844)

[线段树 30](#_Toc495217845)

[Bfs queue 33](#_Toc495217846)

[大数 34](#_Toc495217847)

[莫比乌斯反演 43](#_Toc495217848)

[Kmp 45](#_Toc495217849)

[Kmp（不可重叠） 48](#_Toc495217850)

矩阵乘法 快速幂

**#include <iostream>**

**#include <cstdio>**

**#include <cstring>**

**#include <cstdlib>**

**#include <algorithm>**

**#include <queue>**

**#include <vector>**

**#include <stack>**

**#include <map>**

**#include <cmath>**

**using namespace std;**

**typedef long long ll;**

**typedef unsigned long long ull;**

**typedef unsigned int uint;**

**const int mod = 1e9 + 7;**

**const int INF = 0x7fffffff;**

**int k, m;**

**struct Matrix {**

**int det[20][20];**

**};**

**Matrix yy;**

**Matrix matrix;**

**Matrix Mul(Matrix x, Matrix y) //????**

**{**

**Matrix ret;**

**for (int i = 0; i < 10; ++i) {**

**for (int j = 0; j < 10; ++j) {**

**ret.det[i][j] = 0;**

**for (int k = 0; k < 10; ++k) {**

**ret.det[i][j] += x.det[i][k] \* y.det[k][j];**

**ret.det[i][j] %= m;**

**}**

**}**

**}**

**return ret;**

**}**

**Matrix Pow(Matrix matrix,int n) //???**

**{**

**if (n == 1) {**

**return matrix;**

**}**

**if (n & 1) {**

**return Mul(matrix, Pow(matrix,n - 1));**

**}**

**Matrix t = Pow(matrix,n >> 1);**

**return Mul(t, t);**

**}**

**int main()**

**{**

**m = 9973;**

**int T;**

**scanf("%d",&T);**

**while(T--)**

**{**

**int n,p;**

**scanf("%d%d",&n,&p);**

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

**{**

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

**{**

**scanf("%d",&yy.det[i][j]);**

**}**

**}**

**Matrix xx = Pow(yy,p);**

**int ans = 0;**

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

**{**

**ans += xx.det[i][i];**

**ans %= m;**

**}**

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

**}**

**return 0;**

**}**

矩阵运算 高斯消元

**#include <set>**

**#include <map>**

**#include <queue>**

**#include <math.h>**

**#include <vector>**

**#include <string>**

**#include <stdio.h>**

**#include <string.h>**

**#include <stdlib.h>**

**#include <iostream>**

**#include <algorithm>**

**#define eps 1e-6**

**#define pi acos(-1.0)**

**#define inf 107374182**

**#define inf64 1152921504606846976**

**#define iabs(x) ((x) > 0 ? (x) : -(x))**

**#define clear1(A, X, SIZE) memset(A, X, sizeof(A[0]) \* (SIZE))**

**#define clearall(A, X) memset(A, X, sizeof(A))**

**#define memcopy1(A , X, SIZE) memcpy(A , X ,sizeof(X[0])\*(SIZE))**

**#define memcopyall(A, X) memcpy(A , X ,sizeof(X))**

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

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

**using namespace std;**

**struct node**

**{**

**long long num[75];**

**node()**

**{**

**clearall(num,0);**

**}**

**void clen()**

**{**

**clearall(num,0);**

**}**

**};**

**struct node matrix[75];**

**int n,m,len;**

**bool free\_x[75];**

**long long X[75],p;**

**void Debug(void)**

**{**

**puts("");**

**int i, j;**

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

**{**

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

**{**

**printf("%4d ", matrix[i].num[j] );**

**}**

**cout << endl;**

**}**

**cout << endl;**

**}**

**int Guass()**

**{**

**int i,j,k,col;**

**clearall(X,0);**

**clearall(free\_x,1);//把解集清空，所有变量都标为自由变量**

**//Debug();**

**for (k = 0,col = 0; k < m && col < n; ++k, ++col) //枚举行列**

**{**

**int max\_r = k;//找到该col列元素绝对值最大的那行与第k行交换.(为了在除法时减小误差)**

**for (i = k + 1; i < m; ++i)**

**{**

**if (iabs(matrix[i].num[col]) > iabs(matrix[max\_r].num[col])) max\_r = i;**

**}**

**if (max\_r != k) //交换**

**{**

**for (i = k; i < n + 1; ++i) swap(matrix[k].num[i],matrix[max\_r].num[i]);**

**}**

**/\*if (matrix[k].num[col]!=0 ) //如果对应该列都为0，枚举该行的下一列**

**{**

**k--;**

**continue;**

**}\*/**

**for (i = k + 1; i < m; ++i) //将k后边的col进行初等变换成行阶梯矩阵**

**{**

**if (matrix[i].num[col]!=0)**

**{**

**long long x1=matrix[i].num[col],x2=matrix[k].num[col];**

**for (j = col; j < n + 1; ++j)**

**{**

**matrix[i].num[j] = matrix[i].num[j] \*x2- x1\*matrix[k].num[j];**

**matrix[i].num[j] = (matrix[i].num[j]%p+p)+p;**

**}**

**}**

**}**

**}**

**// 1. 无解的情况: 化简的增广阵中存在(0, 0, ..., a)这样的行(a != 0). 即R(A) != R(A')无解**

**/\*for (i = k; i < m; ++i)**

**{**

**if (iabs(matrix[i].num[col]) >eps) return -1;**

**}\*/**

**// 2. 无穷解的情况: 在n \* (n + 1)的增广阵中出现(0, 0, ..., 0)这样的行，即说明没有形成严格的上三角阵.**

**// 且出现的行数即为自由变元的个数. 即R(A) = R(A') < n**

**//printf("%d %d\n",k,n);**

**/\*if (k < n)**

**{**

**//注释处为求多解的自由变量**

**// 首先，自由变元有n - k个，即不确定的变元至少有n - k个.**

**int num = 0,freeidx;**

**for (i = k - 1; i >= 0; --i)**

**{**

**num = 0;// 用于判断该行中的不确定的变元的个数，如果超过1个，则无法求解，它们仍然为不确定的变元.**

**double tmp = matrix[i].num[n];**

**// 第i行一定不会是(0, 0, ..., 0)的情况，因为这样的行是在第k行到第m行.**

**// 同样，第i行一定不会是(0, 0, ..., a), a != 0的情况，这样的无解的.**

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

**{**

**if (iabs(matrix[i].num[j]) > eps && free\_x[j])**

**{**

**num++;**

**freeidx = j;**

**}**

**}**

**if (num > 1) continue; // 无法求解出确定的变元.**

**// 说明就只有一个不确定的变元free\_index，那么可以求解出该变元，且该变元是确定的.**

**tmp = matrix[i].num[n];**

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

**{**

**if (iabs(matrix[i].num[j])>eps && j != freeidx) tmp -= matrix[i].num[j]\*X[j];**

**}**

**X[freeidx] = tmp/matrix[i].num[freeidx];**

**free\_x[freeidx] = 0;**

**}**

**return n - k;**

**}\*/**

**// 3. 唯一解的情况: 在n \* (n + 1)的增广阵中形成严格的上三角阵.**

**// 计算出Xn-1, Xn-2 ... X0.**

**for (i = k - 1; i >= 0; --i)**

**{**

**long long tmp = matrix[i].num[n];**

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

**{**

**tmp =((tmp- matrix[i].num[j]\*X[j])%p+p)%p;**

**}**

**while(tmp%matrix[i].num[i])tmp+=p;**

**X[i] = ((tmp/matrix[i].num[i])%p+p)%p;**

**} Debug();**

**return 0;**

**}**

**char s[75];**

**int main()**

**{**

**freopen("in.txt", "r", stdin);**

**int T;**

**scanf("%d",&T);**

**while(T--)**

**{**

**clearall(matrix,0);**

**scanf("%lld%s",&p,s);**

**n=strlen(s);**

**m=n;**

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

**{**

**if(s[i]!='\*')**

**matrix[i].num[n]=(s[i]-'a'+1)%p;**

**matrix[i].num[0]=1;**

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

**{**

**matrix[i].num[j]=(matrix[i].num[j-1]\*(i+1))%p;**

**}**

**}**

**Debug();**

**Guass();**

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

**{**

**if(i!=0)printf(" ");**

**printf("%lld",X[i]);**

**}**

**//Debug();**

**puts("");**

**}**

**return 0;**

**}**

中国剩余定理

**#include<stdio.h>**

**//????????**

**//24 29 34 0 1**

**//24 29 34 1 21252**

**//24 29 34 2 21251**

**//0 0 0 0 21252**

**long long W[] = {(long long)23,(long long)28,(long long)33};**

**long long ext\_euclid(long long a, long long b, long long &x, long long &y)**

**{**

**long long ret, tmp;**

**if (!b)**

**{**

**x = 1;**

**y = 0;**

**return a;**

**}**

**ret = ext\_euclid(b, a % b, x, y);**

**tmp = x;**

**x = y;**

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

**return ret;**

**}**

**long long China(long long B[],long long W[],long long k) //B 余数 W 质数**

**{**

**long long i;**

**long long d,x,y,a=0,m,n=1;**

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

**n\*=W[i];**

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

**{**

**m=n/W[i];**

**d=ext\_euclid(W[i],m,x,y);**

**a=(a+y\*m\*B[i])%n;**

**}**

**if(a>0)**

**return a;**

**else**

**return(a+n);**

**}**

**int main()**

**{**

**long long b[3],d;**

**int Case = 0 ;**

**while(scanf("%I64d%I64d%I64d%I64d",&b[0],&b[1],&b[2],&d),(b[0]!=-1))**

**{**

**long long ans = China(b,W,(long long)3)-d;**

**if(ans <= 0)**

**{**

**ans += 21252;**

**}**

**printf("Case %d: the next triple peak occurs in %I64d days.\n",++Case,ans);**

**}**

**}**

欧拉函数

**#include<stdio.h>**

**#include<string.h>**

**#define ll long long**

**ll eul(ll n)**

**{**

**ll ret=1,i;**

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

**{**

**if(n%i==0)**

**{**

**n/=i,ret\*=i-1;**

**while(n%i==0)**

**n/=i,ret\*=i;**

**}**

**}**

**if(n>1)**

**ret\*=n-1;**

**return ret;**

**}**

**int main()**

**{**

**ll n;**

**while(~scanf("%I64d",&n))**

**{**

**ll ans = 0;**

**for(ll i = 1; i\*i<=n;i++)**

**{**

**if((n%i) == 0)**

**{**

**ans += eul(i)\*n/i;**

**if(i\*i!=n)**

**ans += eul(n/i)\*i;**

**// printf("%I64d\n",i);**

**}**

**}**

**printf("%I64d\n",ans);**

**}**

**}**

博弈论 威佐夫博弈

**#include<stdio.h>**

**#include<math.h>**

**#include<algorithm>**

**using namespace std;**

**int main()**

**{**

**int T,a,b;**

**scanf("%d",&T);**

**while(T--)**

**{**

**scanf("%d%d",&a,&b);**

**if(b<a)**

**swap(a,b);**

**int k ;**

**k = b - a;**

**int a\_ = (int)(k\*(1.0+sqrt(5.0))/2);**

**printf("%s\n",(a == a\_) ? "B" : "A");**

**}**

**}**

直角坐标

**//http://blog.sina.com.cn/s/blog\_71dbfe2e0101f7zb.html**

**//http://dev.gameres.com/Program/Abstract/Geometry.htm#矢量的概念**

**#include<stdio.h>**

**#include<string.h>**

**#include<algorithm>**

**#include<iostream>**

**#include<stdlib.h>**

**#include<math.h>**

**using namespace std;**

**const double eps=1e-8;**

**#define zero(x) (((x) > 0 ? (x) : (-x)) < eps)**

**double max(double a,double b)**

**{**

**if(a > b)**

**return a;**

**return b ;**

**}**

**double min(double a,double b)**

**{**

**if(a < b)**

**return a;**

**return b ;**

**}**

**struct point2d//点也是向量**

**{**

**double x,y;**

**point2d(double x=0,double y=0):x(x),y(y){}**

**};**

**struct polygon//多边形 点的集合;**

**{**

**point2d pp[1010];**

**int size;**

**};**

**int isbanana(point2d p1,point2d p2,point2d p3,point2d p4);**

**int fun(point2d a,point2d b,point2d c,point2d d);**

**int isclw(point2d a,point2d b,point2d c);**

**point2d line\_cross(point2d a,point2d b,point2d c,point2d d);**

**polygon p\_h(polygon po,point2d a,point2d b);**

**polygon p\_hop(polygon po,point2d a,point2d b);**

**void see(polygon pol);**

**int ispolcer(polygon pol);**

**double dis(point2d a,point2d b);**

**void see\_p(point2d a);**

**//重载**

**point2d operator + (point2d a,point2d b)**

**{**

**return point2d(a.x+b.x,a.y+b.y);**

**}**

**point2d operator - (point2d a,point2d b)**

**{**

**return point2d(a.x-b.x,a.y-b.y);**

**}**

**point2d operator \* (point2d a,double b)**

**{**

**return point2d(a.x\*b,a.y\*b);**

**}**

**point2d operator / (point2d a,double b)**

**{**

**return point2d(a.x/b,a.y/b);**

**}**

**//×乘**

**//矢量叉积：**

**//**

**//　　计算矢量叉积是与直线和线段相关算法的核心部分。**

**//设矢量P = ( x1, y1 )，Q = ( x2, y2 )，则矢量叉积定义为由(0,0)、**

**//p1、p2和p1+p2所组成的平行四边形的带符号的面积，即：P × Q = x1\*y2 - x2\*y1，**

**//其结果是一个标量。显然有性质 P × Q = - ( Q × P ) 和 P × ( - Q ) = - ( P × Q )。**

**//一般在不加说明的情况下，本文下述算法中所有的点都看作矢量，**

**//两点的加减法就是矢量相加减，而点的乘法则看作矢量叉积。**

**//　　叉积的一个非常重要性质是可以通过它的符号判断两矢量相互之间的顺逆时针关系：**

**//　　若 P × Q > 0 , 则P在Q的顺时针方向。**

**//　　若 P × Q < 0 , 则P在Q的逆时针方向。**

**//　　若 P × Q = 0 , 则P与Q共线，但可能同向也可能反向。**

**double operator \* (point2d a,point2d b)**

**{**

**return a.x\*b.y-b.x\*a.y;**

**}**

**//·乘**

**double pdp(point2d a,point2d b)**

**{**

**return a.x\*b.x+a.y\*b.y;**

**}**

**double dis(point2d a,point2d b)**

**{**

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

**}**

**//两线段是否平行**

**int isl\_l(point2d aa,point2d bb,point2d cc,point2d dd)**

**{**

**point2d ab = aa-bb;**

**point2d cd = cc-dd;**

**if(cd\*ab)**

**{**

**return 0;**

**}**

**return 1;**

**}**

**//int main()**

**//{**

**//**

**// point2d a,b,c,d;**

**// while(~scanf("%lf%lf%lf%lf",&a.x,&a.y,&b.x,&b.y))**

**// {**

**// scanf("%lf%lf%lf%lf",&c.x,&c.y,&d.x,&d.y);**

**// printf("is %d\n",isl\_l(a,b,c,d));**

**//**

**// }**

**//}**

**//两线段是否香蕉(包括边界香蕉)，快速排斥&&跨立实验**

**int isbanana(point2d aa,point2d bb,point2d cc,point2d dd)**

**{**

**int f1,f2;**

**f1=fun(aa,bb,cc,dd);**

**f2=fun(cc,dd,aa,bb);**

**if((f1==1&&f2==1)||f1==0||f2==0||(f1==0&&f2==0))**

**return 1;**

**else**

**return 0;**

**}**

**int fun(point2d a,point2d b,point2d c,point2d d)**

**{**

**point2d p[3];**

**int flag,m,n;**

**p[0].x=b.x-a.x;**

**p[0].y=b.y-a.y;**

**p[1].x=c.x-a.x;**

**p[1].y=c.y-a.y;**

**p[2].x=d.x-a.x;**

**p[2].y=d.y-a.y;**

**m=(p[0].x\*p[1].y-p[0].y\*p[1].x);**

**n=(p[0].x\*p[2].y-p[0].y\*p[2].x);**

**if((m%1000)!=0)**

**m=m%1000;**

**else**

**{**

**while((fabs((double)m/10))>10)**

**m=m/10;**

**}**

**if((n%1000)!=0)**

**n=n%1000;**

**else**

**{**

**while((fabs((double)n/10))>10)**

**n=n/10;**

**}**

**if(n\*m<0)**

**flag=1;**

**else if(m\*n==0)**

**{**

**if((c.x>=min(a.x,b.x)&&c.x<=max(a.x,b.x))&&(c.y>=min(a.y,b.y)&&c.y<=max(a.y,b.y))||(d.x>=min(a.x,b.x)&&d.x<=max(a.x,b.x))&&(d.y>=min(a.y,b.y)&&d.y<=max(a.y,b.y)))**

**flag=0;**

**else**

**flag=-1;**

**}**

**else**

**flag=-1;**

**return flag;**

**}**

**//clockwise;也可以判断a b c是否在同一直线或c在ab那一侧**

**// ab有顺序**

**int isclw(point2d a,point2d b,point2d c)**

**{**

**double ans = (a-b)\*(b-c);**

**if(ans<0)//叉乘< 0 为顺时针返回1**

**return 1;**

**if(ans==0)//同一直线**

**return 0;**

**if(ans>0)**

**return -1;**

**}**

**//int main()**

**//{**

**// point2d a,b,c;**

**// a = point2d(0,0);**

**// b = point2d(1,0);**

**// c = point2d(1,-1);**

**// printf("%d\n",isclw(a,b,c));**

**//}**

**//两直线焦点**

**point2d line\_cross(point2d a,point2d b,point2d c,point2d d)**

**{**

**double a1,b1,c1,a2,b2,c2;**

**// a1 = a.y - b.y;**

**// b1 = b.x - a.x;**

**// c1 = a.x\*b.y - b.x\*a.y; //a\*b;**

**// a2 = c.y - d.y;**

**// b2 = d.x - c.x;**

**// c1 = c.x\*d.y - d.x\*c.y;//c\*d;**

**// double dd = a1 \* b2 - a2 \* b1;**

**// printf("dd %f\n",dd);**

**// return point2d((b1\*c2-b2\*c1)/dd,(c1\*a2-c2\*a1)/dd);**

**// a1 = (a.y - b.y) / (a.x - b.x);**

**// b1 = a.y - a1\*(a.x);**

**// a2 = (c.y - d.y) / (c.x - d.x);**

**// b2 = c.y - a1\*(c.x);**

**// double xx = (b1 - b2) / (a2 - a1);**

**// double yy = a1\*xx+b1;**

**// return point2d(xx,yy);**

**// double lamuda = (fabs((d-a)\*(b-a)))/(fabs((c-a)\*(b-a)));**

**// double xx = c.x+lamuda\*(d.x-c.x);**

**// double yy = c.y+lamuda\*(d.y-c.y);**

**// return point2d(xx,yy);**

**double s2 = fabs((d-a)\*(b-a))/2;**

**double s1 = fabs((c-a)\*(b-a))/2;**

**return point2d((d.x\*s1+c.x\*s2)/(s1+s2)**

**,(d.y\*s1+c.y\*s2)/(s1+s2));**

**}**

**//一个多边形与一个半平面的交集 polygon & half**

**polygon p\_h(polygon po,point2d a,point2d b)**

**{**

**// printf("po.size %d\n",po.size);**

**polygon ans;**

**ans.size = 0;**

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

**{**

**if(isclw(a,b,po.pp[i])>=0)**

**{**

**ans.pp[ans.size++] = po.pp[i];**

**}**

**else if(isclw(a,b,po.pp[i])<0**

**&&isclw(a,b,po.pp[(i+1)%po.size])>0)**

**{**

**ans.pp[ans.size++] = line\_cross(a,b,**

**po.pp[i],po.pp[(i+1)%po.size]);**

**}**

**}**

**// printf("ans %d\n",ans.size);**

**return ans;**

**}**

**void see(polygon pol)**

**{**

**printf("size %d\n",pol.size);**

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

**{**

**printf("%.2f %.2f \n",pol.pp[i].x,pol.pp[i].y);**

**}**

**}**

**//多边形有无核**

**int ispolcer(polygon pol)**

**{**

**polygon now = pol;**

**point2d a,b;**

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

**{**

**now = p\_h(now,pol.pp[i],pol.pp[(i+1)%pol.size]);**

**// printf("!ans %d",now.size);**

**// see(now);**

**if(now.size == 0)**

**{**

**return 0;**

**}**

**// printf("i %d\n",i);**

**}**

**return 1;**

**}**

**//int main()**

**//{**

**// point2d a[4];**

**// point2d b[4];**

**// point2d c[4];**

**// point2d d[4];**

**// a[0] = point2d(0,0);**

**// a[1] = point2d(-4,-4);**

**// a[2] = point2d(-1,0);**

**// a[3] = point2d(0,-1);**

**// b[0] = point2d(-1,0);**

**// b[1] = point2d(1,0);**

**// b[2] = point2d(0,1);**

**// b[3] = point2d(0,4);**

**// c[0] = point2d(1,4);**

**// c[1] = point2d(4,1);**

**// c[2] = point2d(1,1);**

**// c[3] = point2d(4,4);**

**// d[0] = point2d(4,4);**

**// d[1] = point2d(1,1);**

**// d[2] = point2d(5,4);**

**// d[3] = point2d(7,1);**

**//**

**// point2d cross = line\_cross(a[0],a[1],a[2],a[3]);**

**// printf("%.2f %.2f\n",cross.x,cross.y);**

**//**

**//**

**//// point2d e = point2d(3,1);**

**//// isbanana(a[0],a[1],a[2],a[3]);**

**//// isbanana(b[0],b[1],b[2],b[3]);**

**//// isbanana(e,c[1],c[2],c[3]);**

**//// isbanana(d[0],d[1],d[2],d[3]);**

**//**

**//**

**//**

**//**

**//}**

**//**

**int main()**

**{**

**point2d p[60];**

**int n;**

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

**{**

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

**{**

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

**}**

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

**{**

**printf("%lf %lf\n",p[i].x,p[i].y);**

**}**

**}**

**}**

空间坐标

**#include<stdio.h>**

**#include<string.h>**

**#include<algorithm>**

**#include<iostream>**

**#include<stdlib.h>**

**#include<math.h>**

**const double eps=1e-8;**

**#define zero(x) (((x) > 0 ? (x) : (-x)) < eps)**

**struct point3d//点也是向量**

**{**

**double x,y,z;**

**point3d(double x=0,double y=0,double z=0):x(x),y(y),z(z){}**

**};**

**double pdp(point3d a,point3d b);**

**double plen(point3d a);**

**double pplen(point3d a,point3d b);**

**point3d pnv(point3d a,point3d b,point3d c);**

**double psurlen(point3d p,point3d s1,point3d s2,point3d s3);**

**int issur(point3d p,point3d s1,point3d s2,point3d s3);**

**int isline(point3d p,point3d s1,point3d s2) ;**

**double pllen(point3d p,point3d s1,point3d s2);**

**double S(point3d p1,point3d p2,point3d p3);**

**double v4(point3d p1,point3d p2,point3d p3,point3d p4);**

**//重载**

**point3d operator + (point3d a,point3d b)**

**{**

**return point3d(a.x+b.x,a.y+b.y,a.z+b.z);**

**}**

**point3d operator - (point3d a,point3d b)**

**{**

**return point3d(a.x-b.x,a.y-b.y,a.z-b.z);**

**}**

**point3d operator \* (point3d a,double b)**

**{**

**return point3d(a.x\*b,a.y\*b,a.z\*b);**

**}**

**point3d operator / (point3d a,double b)**

**{**

**return point3d(a.x/b,a.y/b,a.z/b);**

**}**

**//×乘**

**point3d operator \* (point3d a,point3d b)**

**{**

**return point3d(a.y\*b.z-a.z\*b.y,a.z\*b.x-a.x\*b.z,a.x\*b.y-a.y\*b.x);**

**}**

**//·乘**

**double pdp(point3d a,point3d b)**

**{**

**return a.x\*b.x+a.y\*b.y+a.z\*b.z;**

**}**

**//向量长度 AKA 模长**

**double plen(point3d a)**

**{**

**return sqrt(pdp(a,a));**

**}**

**//两点距离**

**double pplen(point3d a,point3d b)**

**{**

**return sqrt(fabs(pdp(a-b,a-b)));**

**}**

**//**

**//int main()**

**//{**

**// point3d a = point3d(0,0,0);**

**// point3d b = point3d(0,0,1);**

**// point3d c = point3d(0,1,0);**

**// point3d d = point3d(1,0,0);**

**// point3d e = point3d(1,1,1);**

**//**

**// printf("%.2f\n",pplen(a,e));**

**//}**

**//平面法向量 Plane normal vector 两边叉乘**

**point3d pnv(point3d a,point3d b,point3d c)**

**{**

**return (a-b)\*(a-c);**

**}**

**//int main()**

**//{**

**// point3d a = point3d(0,0,0);**

**// point3d b = point3d(1,0,0);**

**// point3d c = point3d(0,1,0);**

**// point3d d = point3d(0,0,1);**

**// point3d e = point3d(1,1,1);**

**// point3d f = pnv(a,b,c);**

**// printf("%.2f %.2f %.2f\n",f.x,f.y,f.z);**

**//}**

**//点到平面的距离 p surface len（法向量）乘点（到平面上任意一点的距离）除以（法向量模长）**

**double psurlen(point3d p,point3d s1,point3d s2,point3d s3)**

**{**

**point3d f = pnv(s1,s2,s3);**

**return fabs((pdp(f,p-s1))/plen(f));**

**}**

**//int main()**

**//{**

**// point3d a = point3d(0,0,0);**

**// point3d b = point3d(1,0,0);**

**// point3d c = point3d(0,1,0);**

**// point3d d = point3d(0,0,1);**

**// point3d e = point3d(1,1,1);**

**//**

**// printf("%.2f\n",psurlen(a,b,c,d)\*3);// = 根号3 (b,c,d)这个平面截正方体对角线三分之一**

**//**

**//}**

**// 四点共面 到平面的距离==零**

**int issur(point3d p,point3d s1,point3d s2,point3d s3)**

**{**

**if(isline(p,s1,s2)||isline(s3,s1,s2)**

**||isline(p,s1,s3)||isline(p,s3,s2))**

**return 1;//任意三点共线**

**return zero(psurlen(p,s1,s2,s3));**

**}**

**//int main()**

**//{**

**// point3d a = point3d(0,0,0);**

**// point3d b = point3d(1,0,0);**

**// point3d c = point3d(0,1,0);**

**// point3d d = point3d(0,0,1);**

**// point3d e = point3d(1,1,1);**

**// point3d f = point3d(1,1,0);**

**//**

**// printf("%d\n",issur(a,b,c,f));**

**//**

**//}**

**//三点共线 任意两点连线 叉乘为0**

**int isline(point3d p,point3d s1,point3d s2)**

**{**

**return zero(plen((s1-s2)\*(s2-p)));**

**}**

**// int main()**

**//{**

**// point3d a = point3d(0,0,0);**

**// point3d b = point3d(1,0,0);**

**// point3d c = point3d(0,1,0);**

**// point3d d = point3d(2,2,2);**

**// point3d e = point3d(1,1,1);**

**// printf("%.2f\n",pllen(e,d,a));**

**// printf("%d\n",isline(e,d,a));**

**//**

**//}**

**//点到直线距离 p line len（法向量）乘点（到线上任意一点的距离）除以（法向量模长）**

**double pllen(point3d p,point3d s1,point3d s2)**

**{**

**if(isline(p,s1,s2)) //共线为零**

**return 0;**

**point3d f = s1-s2;**

**if(zero((pdp(f,p-s1))))//s1为投影点的话**

**return plen(p-s1);**

**// printf("%.2f\n",pdp(f,p-s1));**

**return fabs((plen(f\*(p-s1)))/plen(f)); //直线p-s1 叉乘直线s1-s2 除以 直线s1-s2模长**

**}**

**// int main()**

**//{**

**// point3d a = point3d(0,0,0);**

**// point3d b = point3d(1,0,0);**

**// point3d c = point3d(0,1,0);**

**// point3d d = point3d(0,0,1);**

**// point3d e = point3d(1,1,1);**

**// point3d f = point3d(2,2,2);**

**//**

**// printf("%.2f\n",pllen(d,b,c));**

**//**

**//}**

**//三点面积 square of surface**

**double S(point3d p1,point3d p2,point3d p3)**

**{**

**if(isline(p1,p2,p3))**

**return 0;**

**// printf("%.2f %.2f\n",pplen(p2,p3),pllen(p1,p2,p3)\*2);**

**return fabs(pplen(p2,p3)\*pllen(p1,p2,p3)/2);**

**}**

**// int main()**

**//{**

**// point3d a = point3d(0,0,0);**

**// point3d b = point3d(1,0,0);**

**// point3d c = point3d(0,1,0);**

**// point3d d = point3d(0,0,1);**

**// point3d e = point3d(1,1,1);**

**// point3d f = point3d(2,2,2);**

**//**

**// printf("%.2f\n",S(d,b,c)\*2);**

**//**

**//}**

**//四面体体积 V4**

**double v4(point3d p1,point3d p2,point3d p3,point3d p4)**

**{**

**if(issur(p1,p2,p3,p4))**

**return 0;//共面为零**

**return S(p1,p2,p3)\*psurlen(p4,p1,p2,p3)/3;//底乘高/3**

**return fabs(pdp((p2-p1)\*(p3-p1),(p4-p1))/6);//混合积几何意义**

**}**

**// int main()**

**//{**

**// point3d a = point3d(0,0,0);**

**// point3d b = point3d(1,0,0);**

**// point3d c = point3d(0,1,0);**

**// point3d d = point3d(0,0,1);**

**// point3d e = point3d(1,1,1);**

**// point3d f = point3d(2,2,2);**

**// printf("%f\n",pdp((b-a)\*(c-a),(d-a))/6);**

**// printf("%f\n",v4(d,b,c,a));**

**//**

**//}**

**int main()**

**{**

**point3d a,b,c,d;**

**while(~scanf("%lf%lf%lf",&a.x,&a.y,&a.z))**

**{**

**scanf("%lf%lf%lf",&b.x,&b.y,&b.z);**

**scanf("%lf%lf%lf",&c.x,&c.y,&c.z);**

**scanf("%lf%lf%lf",&d.x,&d.y,&d.z);**

**// printf("%d\n",issur(a,b,c,d));**

**if(issur(a,b,c,d))**

**{**

**printf("O O O O\n");**

**continue;**

**}**

**double s1 = S(d,b,c);**

**double s2 = S(a,d,c);**

**double s3 = S(a,b,d);**

**double s4 = S(a,b,c);**

**double sum = s1+s2+s3+s4;**

**double r = 3\*v4(a,b,c,d)/(sum);**

**double xx = (s1\*a.x+s2\*b.x+s3\*c.x+s4\*d.x)/sum;**

**double yy = (s1\*a.y+s2\*b.y+s3\*c.y+s4\*d.y)/sum;**

**double zz = (s1\*a.z+s2\*b.z+s3\*c.z+s4\*d.z)/sum;**

**printf("%.4f %.4f %.4f %.4f\n",xx,yy,zz,r);**

**}**

**}**

两圆香蕉

**#include<iostream>**

**#include<algorithm>**

**#include<string.h>**

**#include<stdio.h>**

**#include<map>**

**#include<string>**

**#include<set>**

**#include<math.h>**

**using namespace std;**

**const double pi = acos(-1);**

**struct cyc**

**{**

**int x,y;**

**int r;**

**double ran;**

**double s;**

**double ts;**

**};**

**cyc t,c1,c2;**

**double d;**

**double dis(int x1,int y1,int x2,int y2)**

**{**

**double ans1;**

**double ans2;**

**double ans;**

**ans1 = (double)(x2-x1)\*(x2-x1);**

**ans2 = (double)(y2-y1)\*(y2-y1);**

**ans = sqrt(ans1 + ans2);**

**return ans;**

**}**

**int abs(int a)**

**{**

**if(a<0)**

**return 0 - a;**

**return a;**

**}**

**double findx()**

**{**

**double x;**

**c2.ran = acos(((c2.r\*c2.r)+(d\*d)-(c1.r\*c1.r))/(double)(2\*d\*c2.r));**

**c1.ran = acos(((c1.r\*c1.r)+(d\*d)-(c2.r\*c2.r))/(double)(2\*d\*c1.r));**

**x = c2.r\*(double)sin(c2.ran);**

**return x;**

**}**

**double lwtg()//long way to go**

**{**

**double xx = findx();**

**c1.s = c1.r\*c1.r\*c1.ran;**

**c2.s = c2.r\*c2.r\*c2.ran;**

**c1.ts = c1.r\*d\*sin(c1.ran);**

**return 0;**

**}**

**double banana()**

**{**

**double ans;**

**if(c1.r<c2.r)**

**{**

**t = c1;c1 = c2;c2 = t;**

**}**

**d = dis(c1.x,c1.y,c2.x,c2.y);**

**// printf("dis = %.2f\n",d);**

**double dd = (double)c1.r + c2.r;**

**double dd\_ = (double)abs(c1.r - c2.r);**

**if(d>=dd)**

**{**

**ans = 0.000000;**

**}**

**else if(d<=dd\_)**

**{**

**ans=pi\*(double)(c2.r\*c2.r);**

**}**

**else**

**{**

**lwtg();**

**ans = c1.s+c2.s-c1.ts;**

**}**

**return ans;**

**}**

**int main()**

**{**

**int T;**

**scanf("%d",&T);**

**for(int TT = 0;TT<T;TT++)**

**{**

**// c1.r = 0;c1.ran = 0;c1.s = 0;c1.ts = 0;c1.x = 0;c1.y = 0;**

**// c2.r = 0;c2.ran = 0;c2.s = 0;c2.ts = 0;c2.x = 0;c2.y = 0;**

**int c1x,c1y,c1r,c2x,c2y,c2r;**

**scanf("%d%d",&c2r,&c1r);**

**scanf("%d%d",&c2.x,&c2.y);**

**scanf("%d%d",&c1.x,&c1.y);**

**double ans1,ans2,ans3,ans4;**

**c1.r = c1r;**

**c2.r = c1r;**

**ans1 = banana();**

**c1.r = c2r;**

**c2.r = c2r;**

**ans2 = banana();**

**c1.r = c1r;**

**c2.r = c2r;**

**ans3 = banana();**

**c2.r = c1r;**

**c1.r = c2r;**

**ans4 = banana();**

**printf("Case #%d: %.6f\n",TT+1,ans1+ans2-ans3-ans4);**

**}**

**}**

全排列

**#include<stdio.h>**

**#include<algorithm>**

**using namespace std;**

**int main()**

**{**

**int a[10];**

**long long ans=0;**

**for(int i=0;i<10;i++)**

**a[i]=i;**

**do**

**{**

**ans++;**

**// for(int i=0;i<10;i++)**

**// printf("%d",a[i]);//必do while**

**// printf("\n");**

**}**

**while(next\_permutation(a, a + 10)) ;//顺序过 看7行 一定注意范围**

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

**a[0]=3;a[1]=2;a[2]=1;**

**do**

**{**

**for(int i=0;i<3;i++)**

**printf("%d",a[i]);//必do while**

**printf("\n");**

**}**

**while(prev\_permutation(a, a + 3)) ;//逆序过 看19行**

**}**

快速幂

**#include<stdio.h>**

**#include<algorithm>**

**#include<stdlib.h>**

**using namespace std;**

**int pow(int x,int n)**

**{**

**if(x == 0) return 0;**

**if(n==0) return 1;**

**else**

**{**

**while((n&1)==0)**

**{**

**n>>=1;**

**x\*=x;**

**}**

**}**

**int result=x;**

**n>>=1;**

**while(n!=0)**

**{**

**x\*=x;**

**if((n&1)!=0)**

**result\*=x;**

**n>>=1;**

**}**

**return result;**

**}**

**int main()**

**{**

**int n,i;**

**while(~scanf("%d%d",&n,&i))**

**{**

**printf("%d\n",pow(n,i));**

**}**

**}**

树状数组

**#include<stdio.h>**

**#include<string.h>**

**#include<algorithm>**

**#include<iostream>**

**#include<stdlib.h>**

**#include<math.h>**

**using namespace std;**

**///////////////////记得按 for(int i = 1;i <= 10;i++)**

**///////////////////不要按 for(int i = 0;i < 10;i++)**

**/////////////////// ALL**

**int Lowbit(int x) //2^k**

**{**

**return x & (-x);**

**}**

**struct fenwick**

**{**

**int fa[100100];**

**int fn;//数组大小 //一定要赋值**

**void update(int t, int val)//位置，值**

**{**

**for(int i = t;i <= fn ;i += Lowbit(i))**

**{**

**fa[i] += val;**

**}**

**}**

**int getsum(int x)**

**{**

**int ans=0;**

**for(int i=x; i>0; i-=Lowbit(i))**

**ans += fa[i];**

**return ans;**

**}**

**}fen;**

**//int n;//一共多少个数**

**//int c[110101];**

**//void Update(int t,int val)**

**//{**

**// for(int i=t; i<=n; i+=Lowbit(i))**

**// c[i] += val;**

**//}**

**//int getSum(int x)**

**//{**

**// int ans=0;**

**// for(int i=x; i>0; i-=Lowbit(i))**

**// ans += c[i];**

**// return ans;**

**//}**

**int main()**

**{**

**fen.fn =10;**

**for(int i = 1;i <= 10;i++)**

**{**

**fen.update(i,1);**

**}**

**for(int i = 1;i <= 10;i++)**

**{**

**printf("!%d %d\n",i,fen.getsum(i));**

**}**

**}**

**2D**

**#include<stdio.h>**

**#include<string.h>**

**#include<algorithm>**

**#include<iostream>**

**#include<stdlib.h>**

**#include<math.h>**

**#define N 2100**

**#define ll long long**

**using namespace std;**

**int cc[N][N];**

**int ccn;**

**int num[N][N];**

**int lowbit(int x)**

**{**

**return x&(-x);**

**}**

**void update2(int x,int y,int val)**

**{**

**int tem = val;**

**val -= num[x][y];**

**num[x][y] = tem;**

**for(int i = x;i < N; i += lowbit(i))**

**{**

**for(int j = y; j < N ;j += lowbit(j))**

**{**

**cc[i][j] += val;**

**}**

**}**

**}**

**int sum2(int x,int y)**

**{**

**int ans = 0;**

**for(int i = x;i > 0;i -= lowbit(i))**

**{**

**for(int j = y; j > 0; j -= lowbit(j))**

**{**

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

**}**

**}**

**return ans;**

**}**

**int main()**

**{**

**int T;**

**scanf("%d",&T);**

**while(T--)**

**{**

**scanf("%d",&ccn);**

**int xx,yy,v,xxx,yyy;**

**int s1,s2,s3,s4,s5;**

**char C[10];**

**for(int i = 0;i <= ccn;i++)**

**for(int j = 0;j <= ccn;j++)**

**{**

**cc[i][j] = num[i][j] = 0;**

**}**

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

**{**

**if(C[0]=='E')**

**break;**

**switch(C[1])**

**{**

**case 'E' :**

**int xx,yy,v;**

**scanf("%d%d%d",&xx,&yy,&v);**

**xx++;yy++;**

**update2(xx,yy,v);**

**break;**

**case 'U' :**

**scanf("%d%d%d%d",&xx,&yy,&xxx,&yyy);**

**xx++;yy++;xxx++;yyy++;**

**s1 = sum2(xx-1,yy-1);**

**s2 = sum2(xx-1,yyy);**

**s3 = sum2(xxx,yy-1);**

**s4 = sum2(xxx,yyy);**

**s5 = s4 - s3 - s2 + s1;**

**// printf("%lld %lld %lld %lld\n",s1,s2,s3,s4);**

**printf("%d\n",s5);**

**break;**

**}**

**}**

**}**

**}**

离散化

**/\***

**有些数据本身很大， 自身无法作为数组的下标保存对应的属性。**

**如果这时只是需要这堆数据的相对属性， 那么可以对其进行离散化处理！**

**离散化：当数据只与它们之间的相对大小有关，而与具体是多少无关时，可以进行离散化。**

**http://blog.csdn.net/gokou\_ruri/article/details/7723378**

**\*/**

**#include<stdio.h>**

**#include<string.h>**

**#include<algorithm>**

**#include<iostream>**

**#include<stdlib.h>**

**#include<math.h>**

**using namespace std;**

**int main()**

**{**

**int a[] = {1,2,3,4,5,6,700,700,9};**

**int b[] = {1,2,3,4,5,6,700,700,9};**

**sort(a,a+9);**

**int size=unique(a,a+9)-a;**

**//size为离散化后元素个数**

**for(int i=0;i<9;i++)**

**b[i]=lower\_bound(a,a+size,b[i])-a + 1;**

**//k为b[i]经离散化后对应的值**

**for(int i = 0;i < 9;i++)**

**{**

**printf("%d ",b[i]);**

**}**

**}**

线段树

**//#pragma comment(linker, "/STACK:102400000, 102400000")**

**#include <algorithm>**

**#include <iostream>**

**#include <cstdlib>**

**#include <cstring>**

**#include <cstdio>**

**#include <vector>**

**#include <cmath>**

**#include <ctime>**

**#include <list>**

**#include <set>**

**#include <map>**

**using namespace std;**

**typedef long long LL;**

**const int N = 1e5 + 5;**

**struct SegTree {**

**int l, r, mid;**

**LL sum, Max, Min;**

**}T[N << 2];**

**void build(int p, int l, int r) {**

**int ls = p << 1, rs = (p << 1)|1;**

**T[p].l = l, T[p].r = r, T[p].mid = (l + r) >> 1;**

**if(l == r) {**

**scanf("%lld", &T[p].sum);**

**T[p].Max = T[p].Min = T[p].sum;**

**return ;**

**}**

**build(ls, l, T[p].mid);**

**build(rs, T[p].mid + 1, r);**

**T[p].sum = (T[ls].sum + T[rs].sum);**

**T[p].Min = min(T[ls].Min, T[rs].Min);**

**T[p].Max = max(T[ls].Max, T[rs].Max);**

**}**

**void update\_add(int p, int l, int r, LL val) {**

**int ls = p << 1, rs = (p << 1)|1;**

**if(T[p].l == l && T[p].r == r) {**

**T[p].Min += val;**

**T[p].Max += val;**

**T[p].sum += (r - l + 1) \* val;**

**return ;**

**}**

**if(r <= T[p].mid) {**

**update\_add(ls, l, r, val);**

**}**

**else if(l > T[p].mid) {**

**update\_add(rs, l, r, val);**

**}**

**else {**

**update\_add(ls, l, T[p].mid, val);**

**update\_add(rs, T[p].mid + 1, r, val);**

**}**

**T[p].sum = (T[ls].sum + T[rs].sum);**

**T[p].Min = min(T[ls].Min, T[rs].Min);**

**T[p].Max = max(T[ls].Max, T[rs].Max);**

**}**

**LL query(int p, int l, int r) {**

**int ls = p << 1, rs = (p << 1)|1;**

**if(T[p].l == l && T[p].r == r) {**

**// sum**

**// return T[p].sum;**

**// max**

**// return T[p].Max;**

**// min**

**// return T[p].Min;**

**}**

**if(r <= T[p].mid) {**

**return query(ls, l, r);**

**}**

**else if(l > T[p].mid) {**

**return query(rs, l, r);**

**}**

**else {**

**//sum**

**// return query(ls, l, T[p].mid) + query(rs, T[p].mid + 1, r);**

**//max**

**// return max(query(ls, l, T[p].mid) , query(rs, T[p].mid + 1, r));**

**//min**

**// return min(query(ls, l, T[p].mid) , query(rs, T[p].mid + 1, r));**

**}**

**}**

**int main()**

**{**

**int n, m, t, c, l, r;**

**LL val;**

**scanf("%d", &t);**

**while(t--) {**

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

**build(1, 1, n);**

**while(m--) {**

**scanf("%d %d %d", &c, &l, &r);**

**if(c == 1) {**

**scanf("%lld", &val);**

**update\_add(1, l, r, val);//区间+= 而不是单点 而不是=**

**}**

**else {**

**printf("%lld\n", query(1, l, r));**

**}**

**}**

**}**

**return 0;**

**}**

Bfs queue

**#include<stdio.h>//上下电梯 输入5 1 53 3 1 2 5 0**

**#include<string.h>**

**#include<queue>**

**using namespace std;**

**int st,endd,all;**

**struct lift**

**{**

**int num;**

**int step;**

**} ;**

**int np[205];**

**bool map[205];**

**int bfs()**

**{**

**lift a,next;**

**queue<lift> q;**

**a.num=st;**

**a.step=0;**

**q.push(a);**

**map[st]=1;**

**while(!q.empty())**

**{**

**a=q.front();**

**if(a.num==endd)**

**return a.step;**

**q.pop() ;**

**next=a;**

**next.num=a.num+np[a.num];**

**if(!map[next.num]&&next.num>0&&next.num<=all)**

**{**

**next.step=a.step+1;**

**map[next.num]=1;**

**q.push(next);**

**}**

**next.num=a.num-np[a.num];**

**if(!map[next.num]&&next.num>0&&next.num<=all)**

**{**

**next.step=a.step+1;**

**map[next.num]=1;**

**q.push(next);**

**}**

**}**

**return -1;**

**}**

**int main()**

**{**

**while(scanf("%d",&all),all)**

**{**

**int i;**

**memset(map,0,sizeof(map));**

**scanf("%d%d",&st,&endd);**

**for(i=1;i<=all;i++)**

**{**

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

**}**

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

**}**

**}**

大数

**#include<iostream>**

**#include<string>**

**#include<iomanip>**

**#include<algorithm>**

**#include<bits/stdc++.h>**

**using namespace std;**

**#define MAXN 9999**

**#define MAXSIZE 10**

**#define DLEN 4**

**class BigNum**

**{**

**private:**

**int a[500]; //¿ÉÒÔ¿ØÖÆ´óÊýµÄÎ»Êý**

**int len; //´óÊý³¤¶È**

**public:**

**BigNum(){ len = 1;memset(a,0,sizeof(a)); } //¹¹Ôìº¯Êý**

**BigNum(const int); //½«Ò»¸öintÀàÐÍµÄ±äÁ¿×ª»¯Îª´óÊý**

**BigNum(const char\*); //½«Ò»¸ö×Ö·û´®ÀàÐÍµÄ±äÁ¿×ª»¯Îª´óÊý**

**BigNum(const BigNum &); //¿½±´¹¹Ôìº¯Êý**

**BigNum &operator=(const BigNum &); //ÖØÔØ¸³ÖµÔËËã·û£¬´óÊýÖ®¼ä½øÐÐ¸³ÖµÔËËã**

**friend istream& operator>>(istream&, BigNum&); //ÖØÔØÊäÈëÔËËã·û**

**friend ostream& operator<<(ostream&, BigNum&); //ÖØÔØÊä³öÔËËã·û**

**BigNum operator+(const BigNum &) const; //ÖØÔØ¼Ó·¨ÔËËã·û£¬Á½¸ö´óÊýÖ®¼äµÄÏà¼ÓÔËËã**

**BigNum operator-(const BigNum &) const; //ÖØÔØ¼õ·¨ÔËËã·û£¬Á½¸ö´óÊýÖ®¼äµÄÏà¼õÔËËã**

**BigNum operator\*(const BigNum &) const; //ÖØÔØ³Ë·¨ÔËËã·û£¬Á½¸ö´óÊýÖ®¼äµÄÏà³ËÔËËã**

**BigNum operator/(const int &) const; //ÖØÔØ³ý·¨ÔËËã·û£¬´óÊý¶ÔÒ»¸öÕûÊý½øÐÐÏà³ýÔËËã**

**BigNum operator!() const ; //´óÊýµÄ½×³Ë**

**BigNum operator^(const int &) const; //´óÊýµÄn´Î·½ÔËËã**

**int operator%(const int &) const; //´óÊý¶ÔÒ»¸öintÀàÐÍµÄ±äÁ¿½øÐÐÈ¡Ä£ÔËËã**

**bool operator>(const BigNum & T)const; //´óÊýºÍÁíÒ»¸ö´óÊýµÄ´óÐ¡±È½Ï**

**bool operator<(const BigNum & T)const;**

**bool operator==(const BigNum & T)const; //´óÊýºÍÁíÒ»¸ö´óÊýµÄÊÇ·ñµÈÓÚ**

**bool operator>(const int & t)const; //´óÊýºÍÒ»¸öintÀàÐÍµÄ±äÁ¿µÄ´óÐ¡±È½Ï**

**bool operator<(const int & t)const; //´óÊýºÍÒ»¸öintÀàÐÍµÄ±äÁ¿µÄ´óÐ¡±È½Ï**

**void print(); //Êä³ö´óÊý**

**};**

**BigNum::BigNum(const int b) //½«Ò»¸öintÀàÐÍµÄ±äÁ¿×ª»¯Îª´óÊý**

**{**

**int c,d = b;**

**len = 0;**

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

**while(d > MAXN)**

**{**

**c = d - (d / (MAXN + 1)) \* (MAXN + 1);**

**d = d / (MAXN + 1);**

**a[len++] = c;**

**}**

**a[len++] = d;**

**}**

**BigNum::BigNum(const char\*s) //½«Ò»¸ö×Ö·û´®ÀàÐÍµÄ±äÁ¿×ª»¯Îª´óÊý**

**{**

**int t,k,index,l,i;**

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

**l=strlen(s);**

**len=l/DLEN;**

**if(l%DLEN)**

**len++;**

**index=0;**

**for(i=l-1;i>=0;i-=DLEN)**

**{**

**t=0;**

**k=i-DLEN+1;**

**if(k<0)**

**k=0;**

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

**t=t\*10+s[j]-'0';**

**a[index++]=t;**

**}**

**}**

**BigNum::BigNum(const BigNum & T) : len(T.len) //¿½±´¹¹Ôìº¯Êý**

**{**

**int i;**

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

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

**a[i] = T.a[i];**

**}**

**BigNum & BigNum::operator=(const BigNum & n) //ÖØÔØ¸³ÖµÔËËã·û£¬´óÊýÖ®¼ä½øÐÐ¸³ÖµÔËËã**

**{**

**int i;**

**len = n.len;**

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

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

**a[i] = n.a[i];**

**return \*this;**

**}**

**istream& operator>>(istream & in, BigNum & b) //ÖØÔØÊäÈëÔËËã·û**

**{**

**char ch[MAXSIZE\*4];**

**int i = -1;**

**in>>ch;**

**int l=strlen(ch);**

**int count=0,sum=0;**

**for(i=l-1;i>=0;)**

**{**

**sum = 0;**

**int t=1;**

**for(int j=0;j<4&&i>=0;j++,i--,t\*=10)**

**{**

**sum+=(ch[i]-'0')\*t;**

**}**

**b.a[count]=sum;**

**count++;**

**}**

**b.len =count++;**

**return in;**

**}**

**ostream& operator<<(ostream& out, BigNum& b) //ÖØÔØÊä³öÔËËã·û**

**{**

**int i;**

**cout << b.a[b.len - 1];**

**for(i = b.len - 2 ; i >= 0 ; i--)**

**{**

**cout.width(DLEN);**

**cout.fill('0');**

**cout << b.a[i];**

**}**

**return out;**

**}**

**BigNum BigNum::operator+(const BigNum & T) const //Á½¸ö´óÊýÖ®¼äµÄÏà¼ÓÔËËã**

**{**

**BigNum t(\*this);**

**int i,big; //Î»Êý**

**big = T.len > len ? T.len : len;**

**for(i = 0 ; i < big ; i++)**

**{**

**t.a[i] +=T.a[i];**

**if(t.a[i] > MAXN)**

**{**

**t.a[i + 1]++;**

**t.a[i] -=MAXN+1;**

**}**

**}**

**if(t.a[big] != 0)**

**t.len = big + 1;**

**else**

**t.len = big;**

**return t;**

**}**

**BigNum BigNum::operator-(const BigNum & T) const //Á½¸ö´óÊýÖ®¼äµÄÏà¼õÔËËã**

**{**

**int i,j,big;**

**bool flag;**

**BigNum t1,t2;**

**if(\*this>T)**

**{**

**t1=\*this;**

**t2=T;**

**flag=0;**

**}**

**else**

**{**

**t1=T;**

**t2=\*this;**

**flag=1;**

**}**

**big=t1.len;**

**for(i = 0 ; i < big ; i++)**

**{**

**if(t1.a[i] < t2.a[i])**

**{**

**j = i + 1;**

**while(t1.a[j] == 0)**

**j++;**

**t1.a[j--]--;**

**while(j > i)**

**t1.a[j--] += MAXN;**

**t1.a[i] += MAXN + 1 - t2.a[i];**

**}**

**else**

**t1.a[i] -= t2.a[i];**

**}**

**t1.len = big;**

**while(t1.a[t1.len - 1] == 0 && t1.len > 1)**

**{**

**t1.len--;**

**big--;**

**}**

**if(flag)**

**t1.a[big-1]=0-t1.a[big-1];**

**return t1;**

**}**

**BigNum BigNum::operator\*(const BigNum & T) const //Á½¸ö´óÊýÖ®¼äµÄÏà³ËÔËËã**

**{**

**BigNum ret;**

**int i,j,up;**

**int temp,temp1;**

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

**{**

**up = 0;**

**for(j = 0 ; j < T.len ; j++)**

**{**

**temp = a[i] \* T.a[j] + ret.a[i + j] + up;**

**if(temp > MAXN)**

**{**

**temp1 = temp - temp / (MAXN + 1) \* (MAXN + 1);**

**up = temp / (MAXN + 1);**

**ret.a[i + j] = temp1;**

**}**

**else**

**{**

**up = 0;**

**ret.a[i + j] = temp;**

**}**

**}**

**if(up != 0)**

**ret.a[i + j] = up;**

**}**

**ret.len = i + j;**

**while(ret.a[ret.len - 1] == 0 && ret.len > 1)**

**ret.len--;**

**return ret;**

**}**

**BigNum BigNum::operator/(const int & b) const //´óÊý¶ÔÒ»¸öÕûÊý½øÐÐÏà³ýÔËËã**

**{**

**BigNum ret;**

**int i,down = 0;**

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

**{**

**ret.a[i] = (a[i] + down \* (MAXN + 1)) / b;**

**down = a[i] + down \* (MAXN + 1) - ret.a[i] \* b;**

**}**

**ret.len = len;**

**while(ret.a[ret.len - 1] == 0 && ret.len > 1)**

**ret.len--;**

**return ret;**

**}**

**int BigNum::operator %(const int & b) const //´óÊý¶ÔÒ»¸öintÀàÐÍµÄ±äÁ¿½øÐÐÈ¡Ä£ÔËËã**

**{**

**int i,d=0;**

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

**{**

**d = ((d \* (MAXN+1))% b + a[i])% b;**

**}**

**return d;**

**}**

**BigNum BigNum::operator^(const int & n) const //´óÊýµÄn´Î·½ÔËËã**

**{**

**BigNum t,ret(1);**

**int i;**

**if(n<0)**

**exit(-1);**

**if(n==0)**

**return 1;**

**if(n==1)**

**return \*this;**

**int m=n;**

**while(m>1)**

**{**

**t=\*this;**

**for( i=1;i<<1<=m;i<<=1)**

**{**

**t=t\*t;**

**}**

**m-=i;**

**ret=ret\*t;**

**if(m==1)**

**ret=ret\*(\*this);**

**}**

**return ret;**

**}**

**BigNum BigNum::operator! ()const //´óÊýµÄ½×³Ë**

**{**

**BigNum t = \*this;**

**if(\*this == BigNum(0)||\*this == BigNum(1))**

**{**

**return BigNum(1);**

**}**

**BigNum ans = BigNum(1);**

**while(t>0)**

**{**

**ans = ans \* t;**

**t = (t - BigNum(1));**

**}**

**return ans;**

**}**

**bool BigNum::operator>(const BigNum & T) const //´óÊýºÍÁíÒ»¸ö´óÊýµÄ´óÐ¡±È½Ï**

**{**

**int ln;**

**if(len > T.len)**

**return true;**

**else if(len == T.len)**

**{**

**ln = len - 1;**

**while(a[ln] == T.a[ln] && ln >= 0)**

**ln--;**

**if(ln >= 0 && a[ln] > T.a[ln])**

**return true;**

**else**

**return false;**

**}**

**else**

**return false;**

**}**

**bool BigNum::operator < (const BigNum& T) const**

**{**

**int ln;**

**if(len<T.len)**

**return true;**

**else if(len==T.len)**

**{**

**ln = len - 1;**

**while(a[ln] == T.a[ln] && ln >= 0)**

**ln--;**

**if(ln >= 0 && a[ln] < T.a[ln])**

**return true;**

**else**

**return false;**

**}**

**else**

**return false;**

**}**

**bool BigNum::operator==(const BigNum & T) const //´óÊýºÍÁíÒ»¸ö´óÊýÊÇ·ñµÈÓÚ**

**{**

**int ln;**

**if(len > T.len)**

**return false;**

**else if(len == T.len)**

**{**

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

**{**

**if(a[i]!=T.a[i])**

**{**

**return false;**

**}**

**}**

**return true;**

**}**

**else**

**return false;**

**}**

**bool BigNum::operator >(const int & t) const //´óÊýºÍÒ»¸öintÀàÐÍµÄ±äÁ¿µÄ´óÐ¡±È½Ï**

**{**

**BigNum b(t);**

**return \*this>b;**

**}**

**void BigNum::print() //Êä³ö´óÊý**

**{**

**int i;**

**cout << a[len - 1];**

**for(i = len - 2 ; i >= 0 ; i--)**

**{**

**cout.width(DLEN);**

**cout.fill('0');**

**cout << a[i];**

**}**

**cout << endl;**

**}**

**int main(void)**

**{**

**int i,n;**

**BigNum x,xx,xxx; //¶¨Òå´óÊýµÄ¶ÔÏóÊý×é**

**char s[100];**

**while(~scanf("%d%s",&n,s))**

**{**

**x = BigNum(s);**

**xx = BigNum(n);**

**x.print();**

**xx.print();**

**printf("%d\n",x==xx);**

**xxx = !x;**

**xxx.print();**

**}**

**}**

莫比乌斯反演

**#include<stdio.h>**

**#include<iostream>**

**#include<string.h>**

**using namespace std;**

**const int mod = (int)1e9+7;**

**const int MAXN=50000+50;**

**const int INF=0x7fffffff;**

**int miu[MAXN] ;**

**int pre[MAXN] ;**

**#define FI(n) FastIO::read(n)**

**namespace FastIO {**

**const int SIZE = 1 << 16;**

**char buf[SIZE], obuf[SIZE], str[60];**

**int bi = SIZE, bn = SIZE, opt;**

**int read(char \*s) {**

**while (bn) {**

**for (; bi < bn && buf[bi] <= ' '; bi++);**

**if (bi < bn) break;**

**bn = fread(buf, 1, SIZE, stdin);**

**bi = 0;**

**}**

**int sn = 0;**

**while (bn) {**

**for (; bi < bn && buf[bi] > ' '; bi++) s[sn++] = buf[bi];**

**if (bi < bn) break;**

**bn = fread(buf, 1, SIZE, stdin);**

**bi = 0;**

**}**

**s[sn] = 0;**

**return sn;**

**}**

**bool read(int& x) {**

**int n = read(str), bf;**

**if (!n) return 0;**

**int i = 0; if (str[i] == '-') bf = -1, i++; else bf = 1;**

**for (x = 0; i < n; i++) x = x \* 10 + str[i] - '0';**

**if (bf < 0) x = -x;**

**return 1;**

**}**

**};**

**void get\_miu(int maxn)**

**{**

**int prime[MAXN],pnum=0;**

**miu[1]= pre[1] = 1;**

**for (int i=2;i<maxn;i++) miu[i]=-INF;**

**for (int i=2;i<maxn;i++)**

**{**

**if (miu[i]==-INF)**

**{**

**miu[i]=-1;**

**prime[++pnum]=i;**

**}**

**for (int j=1;j<=pnum;j++)**

**{**

**if (i\*prime[j]>=maxn) break;**

**if (i%prime[j]==0) miu[i\*prime[j]]=0;**

**else miu[i\*prime[j]]=-miu[i];**

**}**

**pre[i] = pre[i-1]+miu[i];**

**}**

**}**

**int solve(int a,int b,int c,int d,int k)//Ä¬ÈÏa==1 c==1**

**{**

**if(k == 0)**

**{**

**return 0;**

**}**

**b = b/k;**

**d = d/k;**

**if(b>d)**

**{**

**int t = b;**

**b = d;**

**d = t;**

**}**

**int ans = 0;**

**for(int i = 1,pos;i <= b;i = pos+1)**

**{**

**pos = min(b/(b/i),d/(d/i));**

**ans =ans +((pre[pos]-pre[i-1])\*(b/i)\*(d/i));**

**}**

**return ans;**

**}**

**int main()**

**{**

**memset(miu,0,sizeof(miu));**

**get\_miu(MAXN);**

**// for(int i = 1; i <= 10;i++)**

**// {**

**// printf("%d %d\n",i,miu[i]);**

**// }**

**int T,Tc = 1;**

**FI(T);**

**while(T--)**

**{**

**int a,b,c,d,k;**

**FI(a);FI(b);FI(c);FI(d);FI(k);**

**int ans = solve(1,b,1,d,k)+solve(1,a-1,1,c-1,k)-solve(1,b,1,c-1,k)-solve(1,a-1,1,d,k);**

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

**}**

**return 0;**

**}**

Kmp

**/\***

**pku3461(Oulipo), hdu1711(Number Sequence)**

**这个模板 字符串是从0开始的**

**Next数组是从1开始的**

**\*/**

**#include <iostream>**

**#include <cstring>**

**using namespace std;**

**const int N = 1000002;**

**int nnext[N];**

**char S[N], T[N];**

**int slen, tlen;**

**void getNext(char T[])**

**{**

**int j, k;**

**tlen = strlen(T);**

**j = 0; k = -1; nnext[0] = -1;**

**while(j <= tlen)**

**if(k == -1 || T[j] == T[k])**

**nnext[++j] = ++k;**

**else**

**k = nnext[k];**

**}**

**void getNext()**

**{**

**int j, k;**

**j = 0; k = -1; nnext[0] = -1;**

**while(j < tlen)**

**if(k == -1 || T[j] == T[k])**

**nnext[++j] = ++k;**

**else**

**k = nnext[k];**

**}**

**/\***

**返回模式串T在主串S中首次出现的位置**

**返回的位置是从0开始的。**

**\*/**

**int KMP\_Index()**

**{**

**int i = 0, j = 0;**

**getNext();**

**while(i < slen && j < tlen)**

**{**

**if(j == -1 || S[i] == T[j])**

**{**

**i++; j++;**

**}**

**else**

**j = nnext[j];**

**}**

**if(j == tlen)**

**return i - tlen+1;**

**else**

**return -1;**

**}**

**/\***

**返回模式串在主串S中出现的次数**

**\*/**

**int KMP\_Count()**

**{**

**int ans = 0;**

**int i, j = 0;**

**if(slen == 1 && tlen == 1)**

**{**

**if(S[0] == T[0])**

**return 1;**

**else**

**return 0;**

**}**

**getNext();**

**for(i = 0; i < slen; i++)**

**{**

**while(j > 0 && S[i] != T[j])**

**j = nnext[j];**

**if(S[i] == T[j])**

**j++;**

**if(j == tlen)**

**{**

**ans++;**

**j = nnext[j];**

**}**

**}**

**return ans;**

**}**

**int main()**

**{**

**int TT;**

**int i, cc;**

**cin>>TT;**

**while(TT--)**

**{**

**cin>>S>>T;**

**slen = strlen(S);**

**tlen = strlen(T);**

**cout<<"模式串T在主串S中首次出现的位置是: "<<KMP\_Index()<<endl;**

**cout<<"模式串T在主串S中出现的次数为: "<<KMP\_Count()<<endl;**

**}**

**return 0;**

**}**

**/\***

**test case**

**4**

**aaaaaa a**

**abcd d**

**aabaa b**

**\*/**

Kmp（不可重叠）

**#include<stdio.h>**

**#include<string.h>**

**void makeNext(const char P[],int next[])**

**{**

**int q,k;**

**int m = strlen(P);**

**next[0] = 0;**

**for (q = 1,k = 0; q < m; ++q)**

**{**

**while(k > 0 && P[q] != P[k])**

**k = next[k-1];**

**if (P[q] == P[k])**

**{**

**k++;**

**}**

**next[q] = k;**

**}**

**}**

**int kmp(const char T[],const char P[],int next[])**

**{**

**int n,m;**

**int i,q;**

**n = strlen(T);**

**m = strlen(P);**

**makeNext(P,next);**

**for (i = 0,q = 0; i < n; ++i)**

**{**

**while(q > 0 && P[q] != T[i])**

**q = next[q-1];**

**if (P[q] == T[i])**

**{**

**q++;**

**}**

**if (q == m)**

**{**

**printf("Pattern occurs with shift:%d\n",(i-m+1));**

**q = 0;**

**}**

**}**

**}**

**int main()**

**{**

**int i;**

**int next[20]={0};**

**char T[] = "aaaaaaa";**

**char P[] = "aa";**

**printf("%s\n",T);**

**printf("%s\n",P );**

**// makeNext(P,next);**

**kmp(T,P,next);**

**for (i = 0; i < strlen(P); ++i)**

**{**

**printf("%d ",next[i]);**

**}**

**printf("\n");**

**return 0;**

**}**