高精度

(1)高精函数

```
                    //高精例程
////////////////////////////////////////////////////////////////
//常用函数                                                     //
//(1)add(bint a, bint b, bint& c)大数加法 c=a+b                //
//(2)add(bint a, type b, bint& c)高精加单精 c=a+b              //
//(3)by(bint a, type b, bint& c)高精乘单精 c=a*b,              //
// **注意:b 应小于 base**                                      //
//(4)by(bint a, bint b, bint& c)大数乘法 c=a*b                 //
//(5)div(bint a, type b, bint& c, type& d)高精除单精           //
//     c = a/b, d = a%b;                                       //
//(6)input(bint& a)输入高精,无效输入返回 0,否则返回 1          //
//(7)output(bint& a)输出高精                                   //
////////////////////////////////////////////////////////////////
//少用函数                                                     //
//(8)move(bint& a)二进制右移,即除 2 操作                       //
//(9)sub(bint a, bint b, bint& c)大数减法 c=a-b,a>=b           //
//(10)sub(bint a, type b, bint& c)高精减单精 c=a-b,a>=b        //
//(11)cmp(bint a, bint b)比较 a 和 b,>,==,<分别返回            //
//      正数,0,负数.                                           //
//(12)give(bint a, bint& b)赋值 b = a;                         //
//(13)give(type a, bint& b)赋值 b = a;                         //
//(14)shift(bint& a, type k)段移位函数,把 a 移动 k 段,变大 mod^k//
//(15)div(bint a, bint b, bint& c, bint& d)大数除法            //
//     c=a/b,d=a%b,**注意:需要函数(1),(2),(4),(9),(11),(13),   //
//   (14)**                                                    //
////////////////////////////////////////////////////////////////
#include <stdio.h>
#include <string.h>
/////////////////////////////////////
#define MAX 100
#define mod 10000
#define baselen 4
#define in(a) scanf("%d",&a)
#define out1(a) printf("%d",a)
#define out2(a) printf("%04d",a)
typedef int type;
/////////////////////////////////////
struct bint{
	type dig[MAX], len;
	bint(){len = 0, dig[0] = 0;}
};
////////////////////////////////////////////
//常用函数
//(1)
void add(bint a, bint b, bint& c){
	type i, carry ;
	for( i = carry = 0; i <= a.len || i <= b.len || carry; i++){
		if(i<=a.len)carry += a.dig[i];
		if(i<=b.len)carry += b.dig[i];
		c.dig[i] = carry%mod;
		carry /= mod;
	}
	c.len = i - 1;
```

用 FinePrint 打印 - 可在 www.fineprint.com.cn 订购
PDF 文件使用 "pdfFactory Pro" 试用版本创建 www.fineprint.com.cn

```
}
//(2)
void add(bint a, type b, bint& c){
      type i;
      for( i    = 0; i <= a.len || b; i++){
            if(i<=a.len)b += a.dig[i];
            c.dig[i] = b%mod;
            b /= mod;
      }
      c.len = i-1;
}
//(3)
void by(bint a, type b, bint& c){
      type i, carry;
      for( i = carry = 0; i <= a.len || carry; i++){
            if( i <= a.len ) carry += b*a.dig[i];
            c.dig[i] = carry%mod;
            carry /= mod;
      }
      i--;
      while( i && !c.dig[i] )i--;
      c.len = i;
}
//(4)
void by(bint a, bint b, bint& c){
      type i, j, carry;
      for( i=a.len+b.len+1; i>=0; i--)c.dig[i] = 0;
      for( i=0; i<=a.len; i++){
            carry = 0;
            for( j=0; j<=b.len||carry; j++){
                  carry    += c.dig[i+j];
                  if(j<=b.len)carry += a.dig[i]*b.dig[j];
                  c.dig[i+j] = carry%mod;
                  carry /= mod;
            }
      }
      i = a.len+b.len+1;
      while(i&&c.dig[i]==0)i--;
      c.len = i;
}
//(5)
void div(bint a, type b, bint& c, type& d){
      type i;
      for(i = a.len,d = 0; i>=0 ; i--){
            d = d*mod + a.dig[i];
            c.dig[i] = d/b;
            d = d%b;
      }
      i = a.len;
      while(i&&c.dig[i]==0)i--;
      c.len = i;
}
//(6)
bool input(bint& a){
      type i, j, w, k, p;
      char data[MAX*baselen+1];
      if(scanf("%s",data)==EOF)return false;
      w = strlen(data) - 1, a.len = 0;
```

```
      for(p=0;p<=w&&data[p]=='0';p++);
      while(1){
            i = j = 0, k = 1;
            while(i<baselen&&w>=p){
                  j = j+ (data[w--] - '0')*k;
                  k *= 10, i++;
            }
            a.dig[a.len++] = j;
            if(w<p)break;
      }
      a.len--;
      return true;
}
//(7)
void output(bint& a){
      type i;
      i = a.len - 1;
      out1(a.dig[a.len]);
      while(i>=0)out2(a.dig[i--]);
}
/////////////////////////////////////////////////////////////////////////////////////
//少用函数
//(8)
void move(bint& a){
      type carry, k, t;
      k = a.len+1, carry = 0;
      while(k--){
            t = a.dig[k]&1;
            a.dig[k] = (a.dig[k]>>1);
            if(carry)a.dig[k] += (mod>>1);
            carry = t;
      }
      if(a.len&&a.dig[a.len]==0)a.len--;
}
//(9)
void sub(bint a, bint b, bint& c){
      type i, carry;
      for( i=carry=0; i<=a.len; i++){
            c.dig[i] = a.dig[i]-carry;
            if(i<=b.len)c.dig[i] -= b.dig[i];
            if(c.dig[i]<0)carry = 1, c.dig[i] += mod;
            else carry = 0;
      }
      i--;
      while(i&&c.dig[i]==0)i--;
      c.len = i;
}
//(10)
void sub(bint a, type b, bint& c){
      type i;
      for( i=0; i<=a.len; i++){
            c.dig[i] = a.dig[i]-b;
            if(c.dig[i]<0)b = 1, c.dig[i] += mod;
            else b = 0;
      }
      i--;
      while(i&&c.dig[i]==0)i--;
      c.len = i;
```

用 FinePrint 打印 - 可在 www.fineprint.com.cn 订购
PDF 文件使用 "pdfFactory Pro" 试用版本创建 www.fineprint.com.cn

```
}
//(11)
int cmp(bint a, bint b){
	if(a.len<b.len)return -1;
	if(a.len>b.len)return 1;
	int i = a.len;
	while(i&&a.dig[i]==b.dig[i])i--;
	return a.dig[i] - b.dig[i];
}
//(12)
void give(bint a, bint& b){
	int i = 0;
	while(i<=a.len){
		b.dig[i] = a.dig[i];
		i++;
	}
	b.len = a.len;
}
//(13)
void give(type a, bint& b){
	b.dig[0] = a%mod;
	a /= mod;
	if(a>0)b.dig[1] = a, b.len = 1;
	else b.len = 0;
}
//(14)
void shift(bint& a, type k){
	int i;
	i = a.len+k;
	while(i>=k){
		a.dig[i] = a.dig[i-k];
		i --;
	}
	while(i>=0)a.dig[i--] = 0;
	a.len += k;
}
//(15)
void div(bint a, bint b, bint& c, bint& d){
	type x, k;
	bint temp;
	give(a, d);
	c.len = c.dig[0] = 0;
	while( cmp(d,b)>0 ){
		k = d.len - b.len;
		if( d.dig[d.len] > b.dig[b.len] )
			x = d.dig[d.len] / (b.dig[b.len] + 1);
		else if( k )
			k--, x =   ( d.dig[d.len]*mod + d.dig[d.len-1])/(b.dig[b.len] + 1);
		else break;
		by( b, x, temp);
		shift( temp, k );
		sub(d, temp, d);
		give( x, temp );
		shift(temp, k);
		add(c, temp, c);
	}
	if(cmp(d,b)>=0) sub(d,b,d), add(c,(type)1, c);
}
```

```
///////////////////////////////////////////////////////////////////////
int main(){
	bint a, b, c, d, start, end, mid;
	while(input(a)){
		give(a,end);
		end.len /= 2;
		end.dig[++end.len] = mod - 1;
		give(a,start);
		start.len /= 2;
		if(start.len==0)start.dig[0] = 0;
		else start.dig[--start.len] = 1;
		while(cmp(end,start)>=0){
			add(end,start,mid);
			move(mid);
			by(mid,mid,d);
			if(cmp(d,a)<=0)add(mid,1,start);
			else sub(mid,1,end);
		}
		output(end);
		printf("\n");
	}
	return 0;
}
```

(2)高精开方

```
//by zhonglei
#include<stdio.h>
#include<string.h>
#include<math.h>
int big(char s1[],char s2[]){
		int len1,len2,i,q;
		q=0;
		while(s1[q]=='0') q++;
		strcpy(s1,s1+q);
		if(strlen(s1)==0){
				s1[0]='0';
				s1[1]=0;
		}
		q=0;
		while(s2[q]=='0') q++;
		strcpy(s2,s2+q);
		if(strlen(s2)==0){
				s2[0]='0';
				s2[1]=0;
		}
		len1=strlen(s1);
		len2=strlen(s2);
		if(len1>len2)
				return 1;
		else if(len1<len2)
				return 0;
		else{
				for(i=0;i<len1;i++){
						if(s1[i]>s2[i])
								return 1;
						else if(s1[i]<s2[i])
```

用 FinePrint 打印 - 可在 www.fineprint.com.cn 订购
PDF 文件使用 "pdfFactory Pro" 试用版本创建 www.fineprint.com.cn

```
                                        return 0;
                        }
                }
                return 0;
        }
        void mul(char s[],int t,char re[]){
                intleft,i,j,k,len;
                char c;
                left=0;
                j=0;
                for(i=strlen(s)-1;i>=0;i--){
                        k=t*(s[i]-'0')+left;
                        re[j++]=(k%10)+'0';
                        left=k/10;
                }
                while(left>0){
                        re[j++]=(left%10)+'0';
                        left/=10;
                }
                re[j]=0;
                len=strlen(re);
                for(i=0;i<len/2;i++){
                        c=re[i];
                        re[i]=re[len-1-i];
                        re[len-1-i]=c;
                }
                return;
        }
        void sub(char a[],char b[]){
                intleft,len1,len2,temp,j;
                len1=strlen(a)-1;
                len2=strlen(b)-1;
                left=0;
                while(len2>=0){
                        temp=a[len1]-b[len2]+left;
                        if(temp<0){
                                temp+=10;
                                left=-1;
                        }
                        else
                                left=0;
                        a[len1]=temp+'0';
                        len1--;
                        len2--;
                }
                while(len1>=0){
                        temp=a[len1]-'0'+left;
                        if(temp<0){
                                temp+=10;
                                left=-1;
                        }
                        else
                                left=0;
                        a[len1]=temp+'0';
                        len1--;
                }
                j=0;
                while(a[j]=='0') j++;
```

```
                strcpy(a,a+j);
                if(strlen(a)==0){
                        a[0]='0';
                        a[1]=0;
                }
                return;
        }
        void sqr(char s[],char re[]){
                char temp[1010];
                char left[1010];
                char p[1010];
                inti,j,k,len1,len2,q;
                len1=strlen(s);
                if(len1%2==0){
                        left[0]=s[0];
                        left[1]=s[1];
                        left[2]=0;
                        j=2;
                }
                else{
                        left[0]=s[0];
                        left[1]=0;
                        j=1;
                }
                re[0]='0';
                re[1]=0;
                q=0;
                while(j<=len1){
                        mul(re,20,temp);
                        len2=strlen(temp);
                        for(i=9;i>=0;i--){
                                temp[len2-1]=i+'0';
                                mul(temp,i,p);
                                if(!big(p,left))
                                        break;
                        }
                        re[q++]=i+'0';
                        re[q]=0;
                        sub(left,p);
                        len2=strlen(left);
                        left[len2]=s[j];
                        left[len2+1]=s[j+1];
                        left[len2+2]=0;
                        j+=2;
                }
        }
        int main(){
                char s[1010],re[1010];
                int i;
                freopen("test.txt","r",stdin);
                while(scanf("%s",s)!=EOF){
                        re[0]=0;
                        sqr(s,re);
                        i=0;
                        while(re[i]=='0') i++;
                        strcpy(re,re+i);
                        printf("%s\n",re);
                }
```

用 FinePrint 打印 - 可在 www.fineprint.com.cn 订购
PDF 文件使用 "pdfFactory Pro" 试用版本创建 www.fineprint.com.cn

```
}
```

(3)高精类

```
                                        //高精类，包括加减乘除取模运算
//written by magic pig on 8th June
#include <iostream.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX 130
#define base 10000
#define baselen 4
long countnub = 0;
//此大整数类用数组 digital[MAX]表示一个大整数;
//一个 digital 表示最大为 9999;
//len 表示目前整数的用到最大 digital 位,sign 表示符号;
class Int {
public :
    //构造函数;
    Int();
    //比较函数,第二个参数为 0 则表示绝对值比较;
    long cmp(Int ,long);
    //判断是否为 0;
    bool zero();
    //判定奇偶性;
    bool odd();
    //右移一个二进制位
    Int move();
    //赋值;
    Int operator = (long);
    Int operator = (Int );
    Int operator = (char*);
    //双目运算;
    Int operator +(Int );
    Int operator -(Int );
    Int operator *(Int );
    Int operator /(Int );
    Int operator %(Int );
    //输入输出;
    friend ostream& operator <<(ostream&,Int );
    friend istream& operator >>(istream& ,Int& );
private :
    long digital[MAX];
    long sign;
    long len;
    //十进制移位
    Int shift(long k);
};

Int ::Int(){digital[len=0] = 0, sign = 1;}

long Int::cmp(Int obj, long sel = 1){
    if(sel&&obj.sign+sign == 0)return sign - obj.sign;    //比较正负号;
    long k = len - obj.len;//比较长度;
    if(k)return sel? sign*k : k;
```

```
    for(k = len; k>0 && obj.digital[k] == digital[k]; k--);    //比较数位;
    return sel? sign * ( digital[k] - obj.digital[k] ): digital[k]-obj.digital[k];
}

bool Int::zero(){ return digital[0]+len ==0; }

bool Int:: odd(){ return digital[0]&1; }

Int Int::move(){
    if(digital[0]<=1&&len==0)digital[0] = 0;
    else {
        int k = len , t, carry=0;
        if (digital[len]==1)len--;
        while(k>=0){
            t = digital[k]&1;
            digital[k]= digital[k]>>1;
            if(carry)digital[k] += base/2;
            k--;
            carry = t;
        }
    }
    if(this->zero())sign = 1;
    return *this;
}
////////////////////////////////////////////////////////////////////////////////
Int Int::operator =(Int obj){
    for(len = 0, sign = obj.sign; len <= obj.len; len++)digital[len]=obj.digital[len];
    len--;
    return *this;
}
Int Int::operator = (long obj){
    if(obj<0)sign = -1, obj = -obj;
    else sign = 1;
    digital[0] = obj%base;
    if(obj/=base){
        digital[1] = obj%base, len = 1;
        if(obj/=base)digital[2] = obj%base, len = 2;
    }
    else len = 0;
    return *this;
}
Int Int::operator = (char *s){
    int i, j, l, k;
    if(s[0] == '-')l = 1,sign = -1;
    else l = 0, sign = 1;
    i=l;
    while(s[i])i++;
    i--;
    k=0;
    while(i-baselen+1>=l){
        for(j=1,digital[k]=0;j<=baselen;j++)
            digital[k]=digital[k]*10+s[i-baselen+j]-'0';
        i = i-baselen,k++;
    }
    digital[k] = 0;
    while(i>=l)digital[k] = digital[k]*10 + s[l++] - '0';
    if(k)len = k-(digital[k]==0);
    else len = 0;
```

用 FinePrint 打印 - 可在 www.fineprint.com.cn 订购
PDF 文件使用 "pdfFactory Pro" 试用版本创建 www.fineprint.com.cn

```
        return *this;
    }
    ////////////////////////////////////////////////////////////////
    Int Int::operator +(Int obj){
        Int sum;
        if(obj.sign==sign){    //同号加;
            long carry;
            long i;
            for(i = carry = 0;i <= len && i <= obj.len;i++)
                carry = carry + digital[i] + obj.digital[i], sum.digital[i]   = carry%base,
carry = carry/base;
            for(;i <= len;i++)
                carry  =  carry  +  digital[i],  sum.digital[i]    =  carry%base,  carry  =
carry/base;
            for(;i <= obj.len;i++)
                carry  =  carry  +obj.digital[i],  sum.digital[i]    =  carry%base,  carry  =
carry/base;
            sum.len = i-!(sum.digital[i] = carry);
            sum.sign = sign;
            return sum;
        }
        else {    //异号变同号减法;
            sum = obj;
            sum.sign = -sum.sign;
            return *this-sum;
        }
    }
    Int  Int::operator -(Int obj){
        Int *sub1, *sub2, quotient;
        if(sign==obj.sign){    //同号减;
            long i, carry;
            i = this->cmp(obj,0);//绝对值比较;
            if(i==0)return quotient;
            else if(i<0)sub1 = &obj, sub2 = this, quotient.sign = -sign;
            else sub1 = this, sub2 = &obj, quotient.sign = sign;
            for(i = carry = 0;i <= sub2->len;i++)
                if(   (quotient.digital[i] = sub1 ->digital[i] - carry - sub2->digital[i]) < 0)
                    carry = 1, quotient.digital[i] += base;//借位;
                else carry = 0;
            for(;i <= sub1->len;i++)
                if(  (quotient.digital[i]  =  sub1  ->digital[i]  -  carry  )<  0  )carry  =  1,
quotient.digital[i] += base;//借位;
                else carry = 0;
            i--;
            while(i&&quotient.digital[i]==0)i--;
            quotient.len = i;
            return quotient;
        }
        else {          //异号变同号加:
            quotient = obj, quotient.sign = -obj.sign;
            return *this + quotient;
        }
    }
    Int Int::operator *(Int obj){
        long carry, i, j, maxlen;
        Int product;
        maxlen = obj.len + len + 2;
        memset( product.digital, 0, sizeof(long)*maxlen );
```

```
        for(i = 0; i <= obj.len; i++){
            for(j =0, carry = 0; j <= len; j++){
                carry += obj.digital[i] * digital[j] +product.digital[j+i];
                product.digital[j+i] = carry%base;
                carry/=base;
            }
            while(carry) product.digital[i+j++] = carry%base, carry /= base;
        }
        i = maxlen-1;
        while(i&&product.digital[i]==0)i--;
        product.len = i;
        if(product.zero())product.sign = 1;//确定符号
        else product.sign = sign*obj.sign;
        return product;
    }
    Int Int::operator /( Int obj){
        long div,    k,    flag;
        Int x, y, z;
        x = *this;
        flag = obj.sign*sign;
        obj.sign = x.sign = 1;
        while( x.cmp(obj) >0 ){
            k = x.len-obj.len;
            if(        x.digital[x.len]        >        obj.digital[obj.len]        )        div        =
x.digital[x.len]/(obj.digital[obj.len]+1);
            else                    if(x.len>obj.len)k--,                    div                    =
(x.digital[x.len]*base+x.digital[x.len-1])/(obj.digital[obj.len]+1);
            else break;
            x = x - ( obj*(z=div) ).shift(k);
            y = y+ z.shift(k);
        }
        if(x.cmp(obj)>=0)y =  y+(z=1);
        if(y.zero())y.sign=1;
        else y.sign=flag;
        return y;
    }
    Int Int::operator %(Int obj){
        long div,    k;
        Int x, y, z;
        x = *this;
        obj.sign = x.sign = 1;
        while( x.cmp(obj) >0 ){
            k = x.len-obj.len;
            if(        x.digital[x.len]        >        obj.digital[obj.len]        )        div        =
x.digital[x.len]/(obj.digital[obj.len]+1);
            else                    if(x.len>obj.len)k--,                    div                    =
(x.digital[x.len]*base+x.digital[x.len-1])/(obj.digital[obj.len]+1);
            else break;
            x = x - ( obj*(z=div) ).shift(k);
        }
        if(x.cmp(obj)>=0)x = x-obj;
        if(x.zero())x.sign = 1;
        else x.sign = sign;
        return x;
    }
    Int Int::shift(long k){
        Int temp;
        int i;
```

用 FinePrint 打印 - 可在 www.fineprint.com.cn 订购
PDF 文件使用 "pdfFactory Pro" 试用版本创建 www.fineprint.com.cn

```
        temp = *this;
        for(i=0;i<=len;i++)temp.digital[i+k]=digital[i];
        for(i=0;i<k;i++)temp.digital[i] = 0;
        temp.sign = sign;
        temp.len = len+k;
        return temp;
}
/////////////////////////////////////////////////////////////////////////
ostream& operator <<(ostream& out,Int obj ){
        int i = obj.len;
        if(obj.sign==-1)out<<'-';
        out<<obj.digital[i--];
        out.fill('0');
        out.setf(ios::right);
        while(i>=0){
                out.width(baselen);
                out<<obj.digital[i--];
        }
        return out;
}
istream& operator >>(istream& in,Int& obj){
        char s[baselen*MAX];
        in>>s;
        obj = s;
        return in;
}
```

计算几何

(1)凸包

```
//水平序凸包
//输入放到 p 中(n)，凸包放在 bag 中(len). O( nlog(n) )的复杂度
#include <stdlib.h>
struct point{
        int x, y;
}p[MAX], bag[MAX];
int len, n;
int cross(point o, point t1, point t2){
        return (t1.x - o.x)*(t2.y-o.y)    - (t1.y - o.y)*(t2.x-o.x);
}
int compare(const void* t1, const void* t2){
        point *p1 = (point*)t1;
        point *p2 = (point*)t2;
        if(p1->y == p2->y)return p1->x - p2->x;
        return p1->y - p2->y;
}
void make_bag(){
        int i, j;
        qsort(p,n,sizeof(point),compare);
        bag[0] = p[0];
        len = 1;
        for(i=1;i<n;i++){
                while(len>=2&&cross(bag[len-2], bag[len-1], p[i])<=0)len--;
                bag[len++] = p[i];
        }
        j = len+1;
```

```
        for(i=n-2;i>=0;i--){
                while(len>=j&&cross(bag[len-2], bag[len-1], p[i])<=0)len--;
                bag[len++] = p[i];
        }
        len--;
}
//jarivs march 凸包,不需要排序,复杂度是 O(n*len),len 是凸包上的点.
#include <math.h>
#define MAX 1001
const double err=1e-10;
struct point{
        double x, y;
}p[MAX], bag[MAX];
int n,len;
double cross(point o, point t1, point t2){
        return (t1.x - o.x)*(t2.y-o.y)    - (t1.y - o.y)*(t2.x-o.x);
}
double dis(point& t1, point& t2){
        return sqrt(pow(t1.x-t2.x,2)+pow(t1.y-t2.y,2));
}
void make_bag(){
        int i, j, k, s;
        double temp;
        for (i=1,j=0;i<n;i++) if (p[i].x<p[j].x || (fabs(p[i].x-p[j].x)<err && p[i].y<p[j].y))
j=i;
        bag[len=0] = p[s=j];
        do {
                len++;
                k= j>0?0:1;
                for (i=0;i<n;i++) if (i!=j) {
                        temp = cross(p[j],p[k],p[i]);
                        if(temp<-err)k = i;
                        else if(fabs(temp)<err && dis(p[j],p[k]) < dis(p[j],p[k]) ) k=i;
                }
                bag[len]=p[j=k];
        } while (s!=j);
}
```

(2)最远点对

```
                                   //最远点对，凸包＋卡壳//
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define MAX 30001
/////////////////////////////////////////凸包
//输入放到 p 中，凸包放在 bag 中(len)，水平直角序凸包
struct point{
        int x, y;
}p[MAX], bag[MAX];
int len, n;
int cross(point o, point t1, point t2){
        return (t1.x - o.x)*(t2.y-o.y)    - (t1.y - o.y)*(t2.x-o.x);
}
int compare(const void* t1, const void* t2){
        point *p1 = (point*)t1;
```

用 FinePrint 打印 - 可在 www.fineprint.com.cn 订购
PDF 文件使用 "pdfFactory Pro" 试用版本创建 www.fineprint.com.cn

```
        point *p2 = (point*)t2;
        if(p1->y == p2->y)return p1->x - p2->x;
        return p1->y - p2->y;
}
void make_bag(){
        int i, j;
        qsort(p,n,sizeof(point),compare);
        bag[0] = p[0];
        len = 1;
        for(i=1;i<n;i++){
                while(len>=2&&cross(bag[len-2], bag[len-1], p[i])<=0)len--;
                bag[len++] = p[i];
        }
        j = len+1;
        for(i=n-2;i>=0;i--){
                while(len>=j&&cross(bag[len-2], bag[len-1], p[i])<=0)len--;
                bag[len++] = p[i];
        }
        len--;
}
////////////////////////////////////////////////
//卡壳
int cross2(point o1,    point t1, point o2, point t2){
        return (t1.x - o1.x )*(t2.y-o2.y)    - (t1.y - o1.y)*(t2.x-o2.x);
}
double dis(point p1, point p2){return sqrt( pow(p1.x-p2.x,2) + pow(p1.y-p2.y,2) );}
double bigger(double x, double y){return x>y?x:y;}
double VertexToVertex(){
         if(len==1)return 0;
         int s, p;
         double t, max;
         bag[len] = bag[s=p=0];
         while(1){
                 if(p==s)p++;
                 if(p>=len)return max;
                 while(p<len&&cross2(bag[s], bag[s+1], bag[p], bag[p+1])>0)
                                 max = bigger(dis(bag[s],bag[p++]), max);
                   max = bigger(dis(bag[s], bag[p]), max);
                 if(cross2(bag[s],bag[s+1],bag[p],bag[p+1])==0)
                             max = bigger( dis(bag[s],bag[p+1]),max);
                 s++;
          }
}
/////////////////////////////////////////////////////
int main(){
//      freopen("test.in","r",stdin);
        int    i;
        while(scanf("%d",&n)!=EOF){
                for(i=0;i<n;i++)scanf("%d%d",&p[i].x,&p[i].y);
                if(n==1){
                        printf("0.00\n");
                        continue;
                }
                make_bag();
                printf("%.2lf\n",VertexToVertex());
        }
        return 0;
}
```

(3)最近点对

```
                                    //最近点对//
//n 为输入点的规模，输入点存在数组 a 中，直接调用 caculate(),答案放在 min 中
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define EPS 1e-10
#define MAX 100001
struct pp{
        double    x, y;
} a[MAX], b[MAX], c[MAX];
double min;
int n;
double abs(double x){return x>0?x:-x;}
double dis(pp a, pp b){return sqrt( pow(a.x-b.x,2)+ pow(a.y-b.y,2) );}
void near_set(pp s_x[], pp s_y[], pp s_temp[], int s, int e){
        if(s==e)return    ;
        int i, j, g, p, mid;
        double dp, x, y;
        mid = (s+e)/2;
        x = s_x[mid].x, y = s_x[mid].y;
        g = s, p = mid+1;
        //分割，注意重点和 x 相同的点。
        for(i=s;i<=e;i++)
        if(s_y[i].x<x||s_y[i].x==x&&s_y[i].y<=y) s_temp[g++] = s_y[i] ;
        else      s_temp[p++] = s_y[i];
        //分治
        near_set(s_x, s_temp, s_y, s, mid);
        near_set(s_x, s_temp, s_y, mid+1, e);
        //合并
        g = s, p = mid+1;
        for(i=s;i<=e&&g<=mid&&p<=e;i++)
        if(s_temp[g].y<s_temp[p].y)s_y[i] = s_temp[g++];
        else s_y[i] = s_temp[p++];
        while(g<=mid)s_y[i++] = s_temp[g++];
        while(p<=e)s_y[i++] = s_temp[p++];
        p = s;
        for(i=s;i<=e;i++)if(abs(x-s_y[i].x)<min)s_temp[p++] = s_y[i];
        for(i=s;i<p;i++)
                for(j=i+1;j<p&&s_temp[j].y - s_temp[i].y<min;j++){
                        dp = dis(s_temp[i], s_temp[j]);
                        if(dp<min)min = dp;
                }
}
int compare1(const void* p, const void* q){
        pp* k1 = (pp*)p;
        pp* k2 = (pp*)q;
        if(k1->x != k2->x)return (k1->x - k2->x)>0?1 : -1;
        return k1->y-k2->y>0?1 : -1;
}
int compare2(const void* p, const void* q){
        pp* k1 = (pp*)p;
        pp* k2 = (pp*)q;
        return k1->y-k2->y>0?1:-1;
```

用 FinePrint 打印 - 可在 www.fineprint.com.cn 订购
PDF 文件使用 "pdfFactory Pro" 试用版本创建 www.fineprint.com.cn

```
}
void caculate(){
      qsort(a,n,sizeof(pp),compare1);
      int i;
      for(i=0;i<n-1;i++)if( dis(a[i], a[i+1])<EPS ){
            min = 0;
            return ;
      }
      for(i=0;i<n;i++)b[i] = a[i];
      qsort(b,n,sizeof(pp),compare2);
      min = 1e20;
      near_set(a,b,c,0,n-1);
}
```

(4)简单多边形的重心

```
//by ant
// 求多边形重心，无论逆正序、凹凸形
#include<stdio.h>
#include<math.h>
#define maxn 1000000

struct point {
              long double x, y;
};

long n;
point p[maxn];
point v, ans;      // v[maxn]

void process() {
              long i,j,k;
              long double s, ss=0;
              ans.x = ans.y = 0;
              for (i=0; i<n; i++) {
                            j = (i + 1) % n;
                            v.x = (p[i].x + p[j].x) / 3.0;
                            v.y = (p[i].y + p[j].y) / 3.0;
                            s = (p[i].x*p[j].y - p[i].y*p[j].x) / 2.0;
                            v.x *= s;
                            v.y *= s;
                            ss += s;
                            ans.x += v.x;
                            ans.y += v.y;
              }
              ans.x /= ss;
              ans.y /= ss;
       if(fabs(ans.x) < 0.005) ans.x = 0;    // printf("%.2lf", -0.001);
       if(fabs(ans.y) < 0.005) ans.y = 0;    // output: -0.00, not: 0.00
              printf("%.2llf %.2llf\n", ans.x, ans.y);
}
```

(5)直线问题

```
//线段相交的判定
struct line{float x1,x2,y1,y2;}obj[2001];
float crossproduct(float x1,float y1,float x2,float y2){
```

```
      return x1*y2-x2*y1;
}
float max(float x,float y){return x>y?x:y;}
float min(float x,float y){return x<y?x:y;}
int intersect(line l1,line l2){
      if(max(l1.x1,l1.x2)<min(l2.x1,l2.x2))return 0;
      if(max(l1.y1,l1.y2)<min(l2.y1,l2.y2))return 0;
      if(min(l1.x1,l1.x2)>max(l2.x1,l2.x2))return 0;
      if(min(l1.y1,l1.y2)>max(l2.y1,l2.y2))return 0;
      if(crossproduct(l2.x1-l1.x1, l2.y1-l1.y1, l1.x2-l1.x1, l1.y2-l1.y1)*
           crossproduct(l1.x2-l1.x1, l1.y2-l1.y1, l2.x2-l1.x1, l2.y2-l1.y1)>=0&&
           crossproduct(l1.x1-l2.x1, l1.y1-l2.y1, l2.x2-l2.x1, l2.y2-l2.y1)*
           crossproduct(l2.x2-l2.x1, l2.y2-l2.y1, l1.x2-l2.x1, l1.y2-l2.y1)>=0)

           return 1;
      return 0;
}
////线段相交的判定  by kk
# include <stdio.h>
# define EPS 10e-8
struct point{
              double x,y;
};
int cross(point p0,point p1,point p2)
{
              double t=(p1.x-p0.x)*(p2.y-p0.y)-(p1.y-p0.y)*(p2.x-p0.x);
              if(t>EPS) return 1;
              if(t<-EPS) return -1;
              return 0;
}
int Intersect(point p1,point p2,point Q1,point Q2)
{
              if(cross(p1,Q1,p2)*cross(p1,p2,Q2)<=0) return 0;
              if(cross(Q2,p1,Q1)*cross(Q2,Q1,p2)<=0) return 0;
              return 1;
}
//线段交点  by kinfkong
//定义:所谓的齐次坐标:
struct point{
              double x,y,flag;
}; //flag  初始值为 1, x,y 就是该点的笛卡尔坐标
point product(point p1,point p2)
{
              point temp;
              temp.x = p1.y*p2.flag - p2.y*p1.flag;
              temp.y = p1.flag*p2.x - p1.x*p2.flag;
              temp.flag = p1.x*p2.y - p2.x*p1.y;
              return temp;
}
//product  这个函数是实上是求:
|    i          j         k        |
|p1.x      p1.y    p1.flag|    展开后，i,j,k  就得到 temp 的 x,y,flag;
|p2.x      p2.y    p2.flag|
很容易记的.
point Intersect(point p1,point p2,point Q1,point Q2)
{
              return  product(product(p1,p2),product(Q1,Q2));
}
```

用 FinePrint 打印 - 可在 www.fineprint.com.cn 订购
PDF 文件使用 "pdfFactory Pro" 试用版本创建 www.fineprint.com.cn

```
//这个函数就是返回交点:
        记返回的是  p;
        if (p.flag=p.x=p.y=0) : 表明直线 p1,p2 和直线 Q1,Q2 重合
        else if (p.flag==0) : 表明直线 p1,p2 和直线 Q1,Q2 平行
        else : 交点的真实坐标为: 横坐标为:p.x/p.flag,  纵坐标为: p.y/p.flag;
```

(6)计算多边形面积(凹凸都适用)

简单的公式.:

(7)判断点线在多边行内

```
//by kinfkong
// zoj 1081  点在多边形内
# include <iostream.h>
# define MAX 1000
# define N 1000
struct point{
       int x,y;
}p[N];
int cross(point p0,point p1,point p2)
{
       return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
}
int Inline(point p0,point p1,point p2)
{
       if(cross(p0,p1,p2)!=0) return 0;
       if((p1.x-p0.x)*(p0.x-p2.x)<0) return 0;
       if((p1.y-p0.y)*(p0.y-p2.y)<0) return 0;
       return 1;
}
int Intersect(point p1,point p2,point Q1,point Q2)
{
       if(cross(p1,Q2,p2)*cross(p1,p2,Q1)<=0) return 0;
       if(cross(Q1,p1,Q2)*cross(Q1,Q2,p2)<=0) return 0;
       return 1;
}
point bigger(point p1,point p2)
{
       if(p1.y>p2.y) return p1;
       else return p2;
}
int check(point p1,int n)
{
       int i,count=0;
       point temp,p2;
       p2.x=MAX;
       p2.y=p1.y;
       for(i=0;i<n;i++){
              if(Inline(p1,p[i],p[i+1])) return 1;
              if(p[i].y!=p[i+1].y){
                     temp=bigger(p[i],p[i+1]);
                     if(Inline(temp,p1,p2)) count++;
                     else if(Intersect(p1,p2,p[i],p[i+1])) count++;
              }
       }
```

```
       return count&1;
}
int main()
{
       int n,m,i,test=1;
       point s;
       cin>>n;
       while(n){
              cin>>m;
              for(i=0;i<n;i++) cin>>p[i].x>>p[i].y;
              p[n]=p[0];
              cout<<"Problem  "<<test++<<':'<<endl;
              for(i=0;i<m;i++){
                     cin>>s.x>>s.y;
                     if(check(s,n)) cout<<"Within"<<endl;
                     else cout<<"Outside"<<endl;
              }
              cin>>n;
              if(n) cout<<endl;
       }
       return 0;
}
```

图论算法

(1)生成树问题

(2)最短路问题

(3)网络流问题

//网络流（最大流程序,标号法)

```
//图的定义:有路则容量不为 0，没路则容量为 0,所有容量假定都为整数
//g 是容量, f 是流量,记得要初始化 f 和 g,点从 0 开始计算;
#include <string.h>
#define MAX 51
#define infinity 0x7fffffff
int g[MAX][MAX], pre[MAX], f[MAX][MAX], v[MAX], s, t, n;
int min(int x,int y){return y<x?y:x;}
void modify(){
       int p, min, q;
       min = v[p=t];
       while(p!=s){
              q = pre[p] ;
              if(q>=n)f[p][q-=n]-=min;
              else f[q][p] += min;
              p = q;
       }
}
int max_flow(){
       int i,   p, q, k, queue[MAX];
       while(1){
              memset( pre, 0xff, sizeof(pre) );
              pre[s] = s;
              queue[p=q=0] = s;
              v[s] = infinity;
```

用 FinePrint 打印 - 可在 www.fineprint.com.cn 订购
PDF 文件使用 "pdfFactory Pro" 试用版本创建 www.fineprint.com.cn

```
            while(p<=q){
                k = queue[p++];
                for( i=0; i<n; i++) if( pre[i]<0 ){
                    if( g[k][i]>f[k][i] ) v[i] = min(g[k][i] - f[k][i], v[k]), pre[i] = k,
queue[++q] = i;
                    else if( f[i][k]>0 )v[i] = min(f[i][k], v[k]), pre[i] = k + n, queue[++q]
= i;
                }
                if(pre[t]>=0)break;
            }
            if(pre[t]<0)break;
            modify();
        }
        for(i=k=0;i<n;i++)k+=f[s][i];
        return k;
}
```

//网络流（最大流程序,前流推进)

```
 #define next(x) ((x)=((x)+1) % 8000)

    int head, tail;
    int queue[8000];
    int flow[200][200];
    int network[200][200];
    int excess[200], height[200], enqueued[200];

    void enqueue(int x)
    {
        if(x == SOURCE || x == SINK) return;
        queue[next(tail)] = x;
    }

    int dequeue(void)
    {
        int ret = queue[next(head)];
        return ret;
    }

    void push(int u, int v)
    {
        int d = excess[u] < network[u][v] - flow[u][v] ? excess[u] : network[u][v] -
flow[u][v];
        int p = excess[v];
        excess[u] -= d;
        excess[v] += d;
        flow[u][v] += d;
        flow[v][u] = -flow[u][v];
        if(!p) enqueue(v);
    }

    int pushFlow(void)
    {
        int i, u, v;
        int minh;
        for(i = 0; i < V; i++)
            excess[i] = height[i] = 0;
```

```
        for(i = 0; i < V; i++)
            if(network[SOURCE][i])
            {
                excess[i] = flow[SOURCE][i] = network[SOURCE][i];
                flow[i][SOURCE] = -flow[SOURCE][i];
                enqueue(i);
            }
        height[SOURCE] = V;
        while(head != tail)
        {
            u = dequeue();
            while(excess[u])
            {
                v = -1;
                minh = 9999999; /* arbitarily large integer */
                for(i = 0; i < V; i++)
                    if(network[u][i] - flow[u][i] > 0 && minh > height[i])
                    {
                        minh = height[i];
                        v = i;
                    }
                if(v == -1) break;

                /* if we can't push, lift u */
                if(minh != height[u] - 1) height[u] = minh + 1;
                push(u, v);
            }
            if(excess[u])
                enqueue(u);
        }
        return excess[SINK];
    }
```

//网络流（最小费用最大流程序) by peipei

```
////////general min cost max flow
const int maxn=100;
//input:
int n,                                  //n>=1
   s,                                    //0<=s<=n
   t,                                    //0<=t<=n && s!=t
   c[maxn+1][maxn+1],
//c[0..n][0..n] !!!, c[i][j]>=0 , c[i][j]==0 then no edge<i,j>
   w[maxn+1][maxn+1];        //w[0..n][0..n],no minus circle!!!!!
//output:
int f[maxn+1][maxn+1];
//local:
const int maxint=2000000000;
int dis[maxn+1],pre[maxn+1],tag[maxn+1];
//if tag[loop]==1 then pre[loop] -> loop
//if tag[loop]==-1 then loop -> pre[loop]
//function:
int find(){
   int x,loop,doit=1;
   for(loop=0;loop<=n;loop++)dis[loop]=maxint;
   dis[s]=0;
```

用 FinePrint 打印 - 可在 www.fineprint.com.cn 订购
PDF 文件使用 "pdfFactory Pro" 试用版本创建 www.fineprint.com.cn

```
    while(doit){
        doit=0;
        for(x=0;x<=n;x++)if(dis[x]!=maxint){
            for(loop=0;loop<=n;loop++){
                if(c[x][loop]>0  &&  f[x][loop]<c[x][loop]
                      && dis[x]+w[x][loop]<dis[loop]){
                    dis[loop]=dis[x]+w[x][loop];
                    pre[loop]=x;tag[loop]=1;doit=1;
                }
                if(c[loop][x]>0  && f[loop][x]>0
                      && dis[x]-w[loop][x]<dis[loop]){
                    dis[loop]=dis[x]-w[loop][x];
                    pre[loop]=x;tag[loop]=-1;doit=1;
                }
            }
        }
    }
    return dis[t]!=maxint;
}
void adjust(){
    int loop,min=maxint;
    for(loop=t;loop!=s;loop=pre[loop]){
        if(tag[loop]>0){
            if(c[pre[loop]][loop]-f[pre[loop]][loop]<min)
                min=c[pre[loop]][loop]-f[pre[loop]][loop];
        }else{
            if(f[loop][pre[loop]]<min)
                min=f[loop][pre[loop]];
        }
    }
    for(loop=t;loop!=s;loop=pre[loop]){
        if(tag[loop]>0){
            f[pre[loop]][loop]+=min;
        }else{
            f[loop][pre[loop]]-=min;
        }
    }
}
void solve(){
    int i,j;
    for(i=0;i<=n;i++)for(j=0;j<=n;j++)f[i][j]=0;
    while(find())adjust();
}
```

(4)二分图问题

//最大基数匹配

```
#include <string.h>
#define MAX 100
#define _clr(x) memset(x,0xff,sizeof(x))
int n;
int match(bool g[][MAX],int n,int m)
{
      int  s[MAX],t[MAX],match1[MAX],match2[MAX],p,q,i,j,k,ret=0;
      for(_clr(match1),_clr(match2),i=0;i<n;ret+=(match1[i]>=0),i++)
```

```
            for(_clr(t),s[p=q=0]=i;p<=q&&match1[i]<0;p++)
                  for(k=s[p],j=0;j<m&&match1[i]<0;j++)
                        if (g[k][j]&&t[j]<0){
                              s[++q]=match2[j],t[j]=k;
                              if (s[q]<0)
                                    for(p=j;p>=0;j=p)
                                          match2[j]=k=t[j],p=match1[k],match1[k]=j;
                        }
      return ret;
}
```

//最大权匹配

```
#include <string.h>
#define MAXN 401
int g[MAXN][MAXN], match1[MAXN], match2[MAXN];
int best_match(int n,int m)
{
    int i,j,k,p,q;
    int l1[MAXN],l2[MAXN];
    int s[MAXN],t[MAXN];
    bool gl[MAXN][MAXN];
    int al,w,ret=0;

    for(i=0;i<n;i++){
        l1[i]=g[i][0];
        for(j=1;j<m;j++) if (g[i][j]>l1[i]) l1[i]=g[i][j];
    }
    memset(l2,0,sizeof(l2));
    for(i=0;i<n;i++)
        for(j=0;j<m;j++) gl[i][j]=(g[i][j]==l1[i]);
    memset(match1,0xff,sizeof(match1));
    memset(match2,0xff,sizeof(match2));
    for(i=0;i<n;i++) {
        memset(t,0xff,sizeof(t));
        for(s[p=q=0]=i;p<=q  &&  match1[i]<0;p++)
            for(k=s[p],j=0;j<m && match1[i]<0;j++)
                if(gl[k][j] && t[j]<0) {
                    s[++q]=match2[j],t[j]=k;
                    if(s[q]<0)
                        for(p=j;p>=0;j=p)
                            match2[j]=k=t[j],p=match1[k],match1[k]=j;
                }
        if(match1[i]<0) {
            al=0x7fffffff;
            for(j=0;j<p;j++)
                for(k=0;k<m;k++)
                    if(t[k]<0 && (w= l1[ s[j] ]+l2[ k ]-g[ s[j] ][ k ])<al) al=w;
            for(j=0;j<p;j++) l1[s[j]]-=al;
            for(j=0;j<m;j++) if(t[j]>=0) l2[j]+=al;
            for(j=0;j<n;j++)
                for(k=0;k<m;k++) gl[j][k]=(g[j][k]==l1[j]+l2[k]);
            i--;
        }
    }
    ret=0;
    for(i=0;i<n;i++)  ret+=g[i][match1[i]];
```

用 FinePrint 打印 - 可在 www.fineprint.com.cn 订购
PDF 文件使用 "pdfFactory Pro" 试用版本创建 www.fineprint.com.cn

```
    return ret;
}
```

(5)Euler 回路

```
//PKU 1041 Euler(原题加了 spj，所以只输出 euler 回路就可以)
//输入格式：两点 + 边标号
//输出 Euler 回路 边标号序列
// 50 个点，2000 条边 无向图
#include <stdio.h>
#include <string.h>
#include <vector>
using namespace std;
#define MAXR 2000
struct node{
    int next, ridx;
    node set(int n, int r){
        next=n,ridx=r;
        return *this;
    }
};
int path[MAXR], rmax, now; // path 答案，倒回来输出。rmax 是总路数，now 是
当输出的路数
bool flag[MAXR];                    //标记
vector<node> adj[50];               //邻接表
vector<node>:: iterator pt[50];  //游标
void euler(int  idx){
    int temp, nx;
    while( pt[idx] != adj[idx].end() ){
        temp = pt[idx]->ridx; //取边
        nx = pt[idx]->next;    //取点
        pt[idx]++;                  //边去了，游标前进
        if(flag[temp])continue;     //之前用了
        flag[temp] = 1;              //标记为用了
        euler(nx);                   //继续搜
        path[now++] = temp;
    }
    return ;
}
int main(){
//  freopen("test.in","r",stdin);
    int i, x, y, r, s;
    node temp;
    while( scanf("%d%d%d",&x,&y,&r)==3 ){
        rmax = 1, now = 0;
        for(i=1;i<50;i++)adj[i].clear();
        if(x<y)s=x;
        else s = y;
        adj[x].push_back(temp.set(y,r));
        adj[y].push_back(temp.set(x,r));
        while( scanf("%d%d",&x,&y),x ){
            scanf("%d",&r);
            adj[x].push_back(temp.set(y,r));
            adj[y].push_back(temp.set(x,r));
            rmax++;
        }
        memset(flag,0,sizeof(flag));
```

```
        for(i=1;i<50;i++)pt[i] = adj[i].begin();
        for(i=1;i<50;i++)if(adj[i].size()&1)break;
        if(i==50)euler(s);
        if(now<rmax)printf("Round trip does not exist.");
        else {
            printf("%d",path[--now]);
            while(now)printf(" %d",path[--now]);
        }
        printf("\n");
    }
    return 0;
}
```

(6)连通性问题

```
//无向图的割顶和桥(pku 1523 测试)

#include <stdio.h>
#define MAX 1010
bool map[MAX][MAX],visited[MAX]; //邻接表和访问标志
//bool brige[MAX][MAX]; //桥标志
int cut[MAX]; //割顶度,即去掉这点，有多少连通块
int  deep[MAX], ans[MAX], n; //深度和访问的最小主先
inline int min(int x, int y){return x<y?x:y;}
void dfs(int idx, int fa, int d){
    int i;
    deep[idx] = ans[idx] = d;
    visited[idx] = 1;
    for(i=0;i<n;i++)if(map[idx][i]){
        if(i==fa)continue;
        if(visited[i]){
            ans[idx] = min(ans[idx], deep[i]);
            continue;
        }
        dfs(i, idx, d+1);
        ans[idx] = min(ans[idx], ans[i]);
//      if(ans[i]>d)brige[i][idx] = brige[idx][i] = 1; //割边
        if(ans[i]>=d) cut[idx]++;
    }
    if(fa!=-1)cut[idx]++;
}

//极大强连通分支(tested by uva 247)

//next 是邻接矩阵，直接调用 solve
#include <string.h>
#define MAXN 30
bool next[MAXN][MAXN], v1[MAXN], v2[MAXN];
int tree[MAXN],f1, f2, n;
int ans[MAXN];//用 ans 存储答案,同一个分支的具有相同的标号。
void dfs1(int idx){
    int i;
    v1[idx] = 1;
    for(i=0;i<n;i++)if(next[idx][i]&&!v1[i])dfs1(i);
    tree[f1++] = idx;
}
void dfs2(int idx, int fa){
```

用 FinePrint 打印 - 可在 www.fineprint.com.cn 订购
PDF 文件使用 "pdfFactory Pro" 试用版本创建 www.fineprint.com.cn

```
        int i;
        v2[idx] = 1, ans[idx] = fa;
        for(i=0;i<f1;i++)
            if(next[tree[i]][idx]&&!v2[tree[i]])
                dfs2(tree[i],fa);
}
void solve(){
        int i, j, k;
        memset(v1,0,sizeof(v1));
        memset(v2,0,sizeof(v2));
        for(i=0;i<n;i++)if(!v1[i]){
            f1 = 0;
            dfs1(i);
            for(j=f1-1;j+1;j--)if(!v2[tree[j]]){
                f2 = 0;
                dfs2(tree[j],tree[j]);
            }
        }
}
```

数据结构

(1)堆

```
//最大堆
#include <stdio.h>
#include <algorithm>
using namespace std;
#define MAX 10001
int n, heap[MAX]; //n 是堆的规模,heap 是堆
void down(int idx){                    //下调
        int sub = (idx<<1)+1;
        while(sub<n){
            if(sub+1<n&&heap[sub]<=heap[sub+1])sub++;
            if(heap[sub]<=heap[idx])break;
            swap(heap[sub],heap[idx]);
            sub = ( (idx = sub)<<1 )+1;
        }
}
void up(int idx){                      //上调
        int fa = (idx-1)>>1;
        while(idx){
            if(heap[fa]>=heap[idx])break;
            swap(heap[fa],heap[idx]);
            fa=((idx=fa)-1)>>1;
        }
}
int pop(){                             //必须有元素
        swap(heap[--n],heap[0]);
        down(0);
        return heap[n];
}
void build(){for(int now=(n>>1)-1;now+1;now--)down(now);}//O(n)建堆
```

(2)线段树

```
//线段树，MAX 表示最大的范围。right,left 表示儿子被覆盖的长度。
//count 表示此段被覆盖的次数。area 是此段被覆盖的长度。
//start 和 end 是此节点代表的起末位置。mid 是中间分割点。
//可以相应的修改一下参数，实现其他功能。
#define MAX 1024*2
struct node {
        int right, left, count;
        int start, end, mid, area;
}tree[MAX];
void construct(int s, int e, int d){
        tree[d].start = s, tree[d].end = e, tree[d].mid = (s+e)/2;
        if(s+1==e)return ;
        construct(s,(s+e)/2,d*2+1), construct((s+e)/2,e,d*2+2);
}
void init(){
        int i;
        for(i=0;i<MAX;i++)tree[i].right = tree[i].left = tree[i].count = tree[i].area = 0;
}
//加线段
int give(int s, int e, int d){
        if( tree[d].start >= s && tree[d].end <= e)tree[d].count++;
        else {
            if( tree[d].mid > s ) tree[d].left = give(s, e, d*2+1);
            if( tree[d].mid < e ) tree[d].right = give(s, e, d*2+2);
        }
        if( tree[d].count ) tree[d].area = tree[d].end-tree[d].start;
        else tree[d].area = tree[d].left + tree[d].right;
        return tree[d].area;
}
//删线段
int del(int s,int e,int d){
        if( tree[d].start >= s && tree[d].end <= e)tree[d].count--;
        else {
            if( tree[d].mid > s ) tree[d].left = del(s, e, d*2+1);
            if( tree[d].mid < e ) tree[d].right = del(s, e, d*2+2);
        }
        if( tree[d].count ) tree[d].area   = tree[d].end-tree[d].start;
        else tree[d].area = tree[d].left + tree[d].right;
        return tree[d].area;
}
```

(3)树状数组

```
//树状数组用于统计
//pku 2352
//输入按 y 有序，没有两点相同
//tree[i]表示 i 所覆盖的数目
#include <stdio.h>
#include <string.h>
#define MAX 15001
#define MAXIDX 32002
struct pp{int x, y; }star[MAX];
bool operator ==( const pp& t1, const pp& t2){return t1.x==t2.x&&t1.y==t2.y;}
int tree[MAXIDX], c[MAX], lowbit[MAXIDX];
inline void add(int x){while(x<MAXIDX){tree[x]++;x += lowbit[x];}}   //增加
```

~~用 FinePrint 打印 - 可在 www.fineprint.com.cn 订购~~
PDF 文件使用 "pdfFactory Pro" 试用版本创建 www.fineprint.com.cn

```
inline int cal(int x){int a=0; while(x){ a+=tree[x]; x-=lowbit[x];}return a;}//计算
int main(){
	int n, i;
	for(i=1;i<MAXIDX;i++)lowbit[i] = (i^(i-1))&i; //最小位
	while(scanf("%d",&n)!=EOF){
		for(i=0;i<n;i++){
			scanf("%d%d",&star[i].x, &star[i].y);
			star[i].x ++; //防止 0， 所以加 1
		}
		memset(c,0,sizeof(c));
		memset(tree,0,sizeof(tree));
		for(i=0;i<n;i++){
			c[ cal(star[i].x)]++;    //统计
			add( star[i].x);//增加
		}
		for(i=0;i<n;i++)printf("%d\n",c[i]);
	}
	return 0;
}
///////////////////////////
//数状数组处理二维图色问题//
//pku 2155                  //
//一个区间取反,询问一个点//
///////////////////////////
#include <stdio.h>
#include <string.h>
#define MAXN 11
int lowbit[MAXN];
bool color[MAXN][MAXN];
int n; //最大范围
bool getcolor(int x, int y){
	bool c = 0;
	int t;
	while(x<=n){
		t = y;
		while(t<=n){
			c^=color[x][t]; //  覆盖点(x,y)的数组
			t += lowbit[t];
		}
		x += lowbit[x];
	}
	return c;
}
void set(int x, int y){
	int t;
	while(x){
		t = y;
		while(t){
			color[x][t]^=1; //  取反
			t^=lowbit[t];
		}
		x^=lowbit[x];
	}
}
int main(){
//	freopen("test.in","r",stdin);
	int i, cas, lx, ly, rx, ry, k, now;
	char op[2];
```

```
	for(i=1;i<MAXN;i++) lowbit[i] = (i^(i-1))&i; // i 的非 0 最小位
	scanf("%d",&cas);
	now = 0;
	while(now<cas){
		if(now)printf("\n");
		now++;
		memset(color,0,sizeof(color));
		scanf("%d%d",&n,&k);
		while(k--){
			scanf("%s%d%d",op,&lx,&ly);
			if(op[0]=='C'){
				scanf("%d%d",&rx,&ry);
				set(lx-1,ly-1),set(rx,ry),set(lx-1,ry), set(rx,ly-1); //取反
			}
			else {
				if(getcolor(lx,ly))printf("1\n"); //取色
				else printf("0\n");
			}
		}
	}
	return 0;
}
```

(4)哈希表

(5)左偏树

```
//zoj 2334 monkey king
#include <stdio.h>
#define MAX 100001
struct node {
	int power, l, r, d;
	void clear(){l=-1,r=-1,d=0;}
}monkey[MAX];

int set[MAX];

int join(int p1, int p2){
	if(monkey[p1].power < monkey[p2].power )p1=p1+p2-(p2=p1);
	if(monkey[p1].r==-1)monkey[p1].r = p2;
	else monkey[p1].r = join(monkey[p1].r, p2);
	int l = monkey[p1].l, r = monkey[p1].r;
	if(l==-1)monkey[p1].l = r,	monkey[p1].r = -1, monkey[p1].d = 1;
	else {
		if(monkey[l].d<monkey[r].d)monkey[p1].l = r, monkey[p1].r = l;
		monkey[p1].d = monkey[r].d+1;
	}
	return p1;
}

int fight(int index){
	int r, l, t, p;
	r = monkey[index].r, l = monkey[index].l;
	if(r!=-1&&l!=-1)t = join(r,l);
	else t = l;
	monkey[index].power/=2;
	monkey[index].clear();
```

用 FinePrint 打印 - 可在 www.fineprint.com.cn 订购
PDF 文件使用 "pdfFactory Pro" 试用版本创建 www.fineprint.com.cn

```
	p = index;
	if(t!=-1){
		p = join(index,t);
		if(p==t){
			set[t] = t;
			set[index] = t;
		}
	}
	return p;
}

int find(int p){
	if(set[p] == p)return p;
	else return set[p] = find(set[p]);
}

int main(){
	freopen("test.in","r",stdin);
	int n, m, i, a, b;
	while( scanf("%d",&n)!=EOF ){
		for(i=0;i<n;i++){
			scanf("%d",&monkey[i].power);
			monkey[i].clear();
			set[i] = i;
		}
		scanf("%d",&m);
		while(m--){
			scanf("%d%d",&a,&b);
			a = find(a-1), b = find(b-1);
			if(a==b)printf("-1\n");
			else {
				a = fight(a), b = fight(b);
				set[a] = set[b] = join(a,b);
				printf("%d\n",monkey[ set[a] ].power);
			}
		}
	}
	return 0;
}
```

数论算法

简单的数论算法(gcd,ext_euclid,中国剩余定义，Euler 函数)

```
//最大公约数

int gcd(int a,int b){
        if(b==0) return a;
        else return gcd(b,a%b);
}
//扩展偶几里得，a*x+b*y=1
int ext_euclid(int a,int b,int &x,int &y)
{
   int t,d;
   if (b==0) {x=1;y=0;return a;}
   d=ext_euclid(b,a %b,x,y);
   t=x;
```

```
	x=y;
	y=t-a/b*y;
	return d;
}

//中国剩余定义，高精度

# include <stdio.h>
# include <string.h>
# define MAX 200

# define INPUT "%I64d"
# define OUTPUT "%I64d "
typedef __int64 I64;

struct number{
	int digit;
	I64 s[500];
};
struct set{
	I64 x,y,d;
};
set Euclid(I64 a,I64 b)
{
	set t,t1;
	if(b==0) {
		t.x=1;
		t.y=0;
		t.d=a;
	}
	else {
		t1=Euclid(b,a%b);
		t.x=t1.y;
		t.y=t1.x-(a/b)*t1.y;
		t.d=t1.d;
	}
	return t;
}
I64 Inverse(I64 a, I64 m)
{
	set t;
	t=Euclid(a,m);
	return (t.x%m+m)%m;
}
void SZDL(I64 r[],I64 m[],int n,I64 d[])
{
	int i,j;
	I64 t[MAX],s;
	for(i=0;i<n;i++) t[i]=r[i]%m[i];
	for(i=1;i<n;i++) {
		d[i-1]=t[i-1];
		for(j=i;j<n;j++) {
			s = ((t[j]-d[i-1])%m[j]+m[j])%m[j];
			t[j]=(s*Inverse(m[i-1],m[j]))%m[j];
		}
	}
	d[i-1]=t[i-1];
```

用 FinePrint 打印 - 可在 www.fineprint.com.cn 订购
PDF 文件使用 "pdfFactory Pro" 试用版本创建 www.fineprint.com.cn

```
}

number add(number a,I64 n)
{
      int carry,i,temp;
      a.s[0]+=n;
      carry=0;
      for(i=0;i<a.digit;i++) {
            temp=a.s[i]+carry;
            a.s[i]=temp%10;
            carry=temp/10;
      }
      while(carry){
            a.s[a.digit++]=carry%10;
            carry/=10;
      }
      return a;
}
number product(number a,I64 n)
{
      int i,temp,carry;
      for(i=0;i<a.digit;i++) a.s[i]*=n;
      carry=0;
      for(i=0;i<a.digit;i++) {
            temp=a.s[i]+carry;
            a.s[i]=temp%10;
            carry=temp/10;
      }
      while(carry){
            a.s[a.digit++]=carry%10;
            carry/=10;
      }
      while(a.s[a.digit-1]==0&&a.digit>1) a.digit--;
      return a;
}
void output(number a)
{
      int i;
      for(i=a.digit-1;i>=0;i--) printf(INPUT,a.s[i]);
}
int main()
{
      int i,n;
      I64 r[MAX],m[MAX],d[MAX];
      number ans;
      while(scanf("%d",&n)!=EOF) {
            for(i=0;i<n;i++) scanf(INPUT,&r[i]);
            for(i=0;i<n;i++) scanf(INPUT,&m[i]);
            SZDL(r,m,n,d);
            ans.s[0]=0;ans.digit=1;
            for(i=n-1;i>=0;i--){
                  ans = add(product(ans,m[i]),d[i]);
            }
            output(ans);
            printf("\n");
      }
      return 0;
}
```

//中国剩余定理,int 版本

```
// zoj 1160
# include <stdio.h>
struct set{
      int d,x,y;
};
set Extended_Euclid(int a,int b)
{
      set temp;
      set temp1;
      if(b==0){
            temp.d=a;temp.x=1;temp.y=0;
            return temp;
      }
      temp1=Extended_Euclid(b,a%b);
      temp.d=temp1.d;
      temp.x=temp1.y;
      temp.y=temp1.x-(a/b)*temp1.y;
      return temp;
}
int Equation_Sol(int a,int b,int n)
{
      set s;
      int i,x0;
      s=Extended_Euclid(a,n);
      return ((s.x*(b/s.d))%n+n)%n;
}
int Chinese_Remainder(int a[],int k,int r[])
{
      int n;
      int i,sum;
      n=1;
      for(i=0;i<k;i++) n*=a[i];
      sum=0;
      for(i=0;i<k;i++) sum=(sum+r[i]*(n/a[i])*Equation_Sol(n/a[i],1,a[i]))%n;
      return sum;
}
int main()
{
      int a[3],r[3],d,t;
      a[0]=23;
      a[1]=28;
      a[2]=33;
      int block,test;
      scanf("%d",&block);
      while(block--){
            test=1;
            while(1){
                  scanf("%d%d%d%d",&r[0],&r[1],&r[2],&d);
                  if(r[0]==-1&&r[1]==-1&&r[2]==-1&&d==-1) break;
                  r[0]%=a[0];
                  r[1]%=a[1];
                  r[2]%=a[2];
                  t=Chinese_Remainder(a,3,r);
                  while(t<=d) t+=a[0]*a[1]*a[2];
                  printf("Case %d: the next triple peak occurs in %d days.\n",test++,t-d);
            }
```

用 FinePrint 打印 - 可在 www.fineprint.com.cn 订购
PDF 文件使用 "pdfFactory Pro" 试用版本创建 www.fineprint.com.cn

```
        if(block) printf("\n");
    }
        return 0;
}
int Euler(int n)
{
        int t,sum,flag,i;
        sum=1;
        for(i=0;Prime[i]<=int(sqrt(n)+eps);i++){
                flag=0;
                while(n%Prime[i]==0){
                        n/=Prime[i];
                        sum*=Prime[i];
                        flag=1;
                }
                if(flag) sum=(sum/Prime[i])*(Prime[i]-1);
        }
        if(n!=1) sum*=n-1;
        return sum;
}
```

(2)随机素数测试与大数分解

```
//miller_rabin 大数检测+Pollard P 素因子分解
//输入 a<2^63
//加大 MAX 可以保证分解的成功率
#include <stdlib.h>
#include <stdio.h>
typedef unsigned __int64 u64;
#define MAX 100
#define MAXN 30
u64 len, dig, limit;
u64 mod(u64 a, u64 b, u64 n){
    if(!a)return 0;
    else return ( ((a&dig)*b)%n + (mod(a>>len,b,n)<<len)%n )%n;
}
u64 by(u64 a, u64 b, u64 n){
    u64 p;
    p = 8, len = 61;
    while(p<n){
        p<<=4;
        len -=4;
    }
    dig = ((limit/p)<<1) - 1; //动态划分段
    return mod(a,b,n);
}
u64 random(){
        u64 a;
        a = rand();
        a *= rand();
        a *= rand();
      a *= rand();
        return a;
}
//////////////////////////////////////////////////////
//Miller_Rabin
u64 square_multiply(u64 x, u64 c, u64 n){
```

```
        u64 z=1;
        while(c){
                if(c%2==1)z = by(z,x,n);;
                x = by(x,x,n);
                c=(c>>1);
        }
        return z;
}
bool Miller_Rabin(u64 n){
    if(n<2)return false;
    if(n==2)return true;
    if(!(n&1))return false;
    u64 k = 0, i, j, m, a;
        m = n - 1;
        while(m%2==0)m=(m>>1),k++;
        for(i=0;i<MAX;i++){
                a = square_multiply(random()%(n-1)+1, m, n);//平方乘
                if(a==1)continue;
                for(j=0;j<k;j++){
                        if(a==n-1)break;
                        a = by(a,a,n);
                }
                if(j<k)continue;
                return false ;
        }
        return true;
}
/////////////////////////////////////////////////////////
//Pollard p,只找出一个因子。
u64 gcd(u64 a,u64 b){
        if(b==0) return a;
        else return gcd(b,a%b);
}
//用公式 f(x) = x^2 + 1 检验碰撞。
u64 f(u64 x, u64 n ){
        return (by(x,x,n)+1)%n;
}
//分解不到，return 0
u64 Pollard(u64 n){
       if(n<=2)return 0;
       if(!(n&1))return 2; //必不可少
        u64 i, p, x,xx;
        for(i=1;i<MAX;i++){
                x = random()%n; //或者直接用  x = i
                xx = f(x,n);
                p = gcd( (xx+n-x)%n , n);
                while(p==1){
                        x = f(x,n);
                        xx = f( f(xx,n),n);
                        p = gcd( (xx+n-x)%n,n)%n;
                }
                if(p)return p;
        }
        return 0;
}
/////////////////////////////////////////////////////////
u64 factor[MAXN], m;
//////////////////////////////////////////////////////
```

用 FinePrint 打印 - 可在 www.fineprint.com.cn 订购
PDF 文件使用 "pdfFactory Pro" 试用版本创建 www.fineprint.com.cn

```
//分解质数因子
u64 prime(u64 a){
      if(Miller_Rabin(a))return 0;
      u64 t = Pollard(a), p;
      if( p = prime(t) )return p;
      else return t;

}
int main(){
      u64 l, a, t;
      limit = 1;
      limit = limit<<63; //动态化分段使用
      while(scanf("%I64u",&a)!=EOF){
            m = 0;
            while(a>1){
                  if(Miller_Rabin(a))break;
                  t = prime(a);
                  factor[m++] = t;
                  a/=t;
            }
            if(a>0)factor[m++]=a;
            for(l=0;l<m;l++)printf("%I64u\n",factor[l]);
      }
      return 0;
}
```

字符串

(1)KMP

```
#include <stdio.h>
//KMP 模式匹配
//sub 为子串
//fa 为匹配的母串，直接调用 KMP 函数就能知道匹配成功与否
#include <string.h>
#define MAX 10000
char sub[MAX], fa[MAX]; //sub 子串， fa 母串
int f[MAX], slen, tlen;
void fail(){
      int i, j;
      f[0]=-1;
      for(i=1;i<slen;i++){
            j = f[i-1];
            while(j>=0&&sub[i]!=sub[j+1])j = f[j];
            if(sub[j+1]==sub[i])f[i]=j+1;
            else f[i] = -1;
      }
}
bool KMP(){
      int ps, pf;
      ps = pf =0, slen =   strlen(sub), tlen =   strlen(fa);
      fail();
      while(ps<slen && pf<tlen)
            if(sub[ps]==fa[pf])ps++,pf++;
            else if(ps==0)pf++;
```

```
            else ps = f[ps-1] +1;
      if(ps<slen)return false;
      return true; //return pf - ps;  返回 fa 中第一个匹配的位置
}
```

(2)后缀数组

(3)LIS(nlogn)

```
//最长严格递增子列
//输入放到 data,元素个数为 n
#define MAX 40010
int data[MAX], n;
int LCS(){
      int i,   pre, ft, la, m;
      for(i=pre=0;i<n;i++){
            ft = 0, la = pre-1;
            while(ft<=la){
                  m = (ft+la)>>1;
                  if(data[m]>=data[i])la = m-1; //去掉'='，最长非严格递增子列
                  else ft = m+1;
            }
            data[ft] = data[i];
            if(ft==pre)pre++;
      }
      return pre;
}
```

(4)最小串表示法(O(N)算法)

```
//MAX 事输入字符串的两倍长度，也可以用取模，省内存
//输出最小子串的最前位置，多个最小子串，输出第一个
#include <string.h>
#include <algorithm>
using namespace std;
#define MAX 200010
inline int max(int x,int y){return x>y?x:y;}
char str[MAX];
int min_string(){
      int i, j, k, len;
      len = strlen(str),i=0,j=1;;
      memcpy(str+len,str,len);
      while(j<len){
            k = 0;
            while(k<len&&str[j+k]==str[i+k])k++;
            if(k==len)break;
            if(str[j+k]<str[i+k]) i=max(j+1,i+k+1);
            else j = j+k+1;
            if(i>j)swap(i,j);
      }
      return i;
}
```

(5)最大公共上升子列(平方算法)

```
//最长公共递增子列(Greatest Common Increasing Subsequence)
//a,b 为两个序列
```

用 FinePrint 打印 - 可在 www.fineprint.com.cn 订购
PDF 文件使用 "pdfFactory Pro" 试用版本创建 www.fineprint.com.cn

```
//平方算法,pku 2127
#include <stdio.h>
#include <string.h>
#define MAX 510
int next[MAX][MAX], maxlen[MAX], a[MAX], b[MAX];
int main(){
    int len1, len2, i, j, k;
    while(scanf("%d",&len1)!=EOF){
        for(i=0;i<len1;i++)scanf("%d",a+i);
        scanf("%d",&len2);
        for(i=0;i<len2;i++)scanf("%d",b+i);
        memset(maxlen,0,sizeof(maxlen));
        //dp,倒回来
        for(i=len1-1;i+1;i--)
            for(j=len2-1,k=len2;j+1;j--)
            if(a[i]==b[j]&&maxlen[j]<maxlen[k]+1)next[j][maxlen[j] = maxlen[k] +
1] = k;
            else if(a[i]<b[j]&&maxlen[j]>maxlen[k])k = j;
        for(k=i=j=0;i<len2;i++)if(maxlen[i]>k)k=maxlen[i],j=i;
        //output 最大值
        printf("%d\n",k);
        //output  路径
        for(i=0;i<k;i++){
            if(i)printf(" ");
            printf("%d",b[j]);
            j = next[j][k-i];
        }
        printf("\n");
    }
    return 0;
}
```

模拟算法

表达式求值

```
//by kinfkong
//zoj 1958
# include <iostream.h>
# include <string.h>
# define N 1000
char table[6][6]={
    '>','>','<','<','>','>',
        '>','>','<','<','>','>',
        '>','>','>','<','>','>',
        '<','<','<','<','=','?',
        '>','>','>','?','>','>',
        '<','<','<','<','?','='
};
int unionset(int a,int b)
{
    return a|b;
}
int diffset(int a,int b)
{
    int i,ans=a;
```

```
    for(i=0;i<26;i++) if((a&(1<<i))&&(b&(1<<i))) ans-=(1<<i);
    return ans;
}
int interset(int a,int b)
{
    return a&b;
}
int oper(int a,char c,int b)
{
    if(c=='+') return unionset(a,b);
    else if(c=='-') return diffset(a,b);
    else return interset(a,b);
}
int getnum(char c)
{
    switch(c){
    case '+':return 0;
    case '-':return 1;
    case '*':return 2;
    case '(':return 3;
    case ')':return 4;
    case '#':return 5;
    }
    return 0;
}
char Precede(char c1,char c2)
{
    return  table[getnum(c1)][getnum(c2)];
}
int Evalute(char s[])
{
    int top1,top2;
    int str2[N];
    char str1[N];
    int i,t,e;
    top1=top2=0;
    t=strlen(s);
    str1[top1++]='#';
    s[t]='#';s[++t]=0;
    i=0;
    while(s[i]!='#'||str1[top1-1]!='#'){
        if(s[i]=='{'){
            i++;e=0;
            while(s[i]!='}') e|=(1<<(s[i++]-'A'));
            i++;
            str2[top2++]=e;
        }
        else{
            switch(Precede(str1[top1-1],s[i])){
            case '<': str1[top1++]=s[i];i++;break;
            case '=': top1--;i++;break;
            case '>': str2[top2-2]=oper(str2[top2-2],str1[to
                              p1-1],
                              str2[top2-1]);
                top2--;top1--;break;
            }
        }
    }
```

用 FinePrint 打印 - 可在 www.fineprint.com.cn 订购
PDF 文件使用 "pdfFactory Pro" 试用版本创建 www.fineprint.com.cn

```
        return str2[0];
}
int main()
{
        char s[1000];
        int i,e;
        while(cin>>s){
                e=Evalute(s);
                cout<<'{';
                for(i=0;i<26;i++)if(e&(1<<i))  cout<<(char)('A'+i);
                cout<<'}'<<endl;
        }
        return 0;
}
```

特殊问题

(1)LCA+RMQ

```
//输入的边存在 edge 链表里面，按照节点标号构图。
//查询为原始标号
#include <stdio.h>
#define MAX1 10010
#define MAX2 20010
#define MAX3 16
#define less(a,b) a>b?b:a
struct node{
        int nub, w;
        node* link;
        node(int a=0, int b = 0,    node*next=NULL){ nub = a, w = b, link = next;}
}edge[MAX1];
int tab[MAX2], dis[MAX1], vetex[MAX1], min[MAX2][MAX3];
int n, m, s, flag;
void search(int a, int w){
        node*temp;
        int k = flag;
        tab[s++] = k, dis[flag] = w;
        temp = edge[a].link;
        while(temp!=NULL){
                flag ++;
                search(temp->nub, w + temp->w);
                tab[s++] = k;
                edge[a].link = temp->link;
                delete temp;
                temp = edge[a].link;
        }
        vetex[a] = s-1;
}

void RMQ(){
        int i, len, j;
        len =    j = 1;
        for(i=0;i<s;i++)min[i][0] = tab[i];
        while(len*2<=s){
                for(i=0;i+len*2<=s;i++)
                        min[i][j] = less( min[i][j-1], min[i+len][j-1] );
```

```
                j++, len*=2;
        }
}
int find(int a, int b){
        if(a==b)return 0;
        int t1, t2, d, i;
        t1 = vetex[a], t2 = vetex[b];
        if(t1>t2) t1 = t2 + t1 - ( t2 = t1);
        i = 0, d = 1;
        while(t1+2*d-1<t2)i++, d*=2;
        return dis[ tab[t2] ] + dis[ tab[t1] ] - 2*dis[ less(min[t1][i], min[t2-d+1][i]) ];
}
int main(){
        int a, b, i, k, cas;
        scanf("%d",&cas);
        while(cas--){
                scanf("%d%d",&n,&m);
                for(i=1;i<=n-1;i++){
                        scanf("%d%d%d",&a,&b,&k);
                        if(a>b)a = a + b - ( b = a );
                        edge[a].link = new node(b, k, edge[a].link);
                }
                s = 0, flag = 1;
                search(1,0);
                RMQ();
                for(i=0;i<m;i++){
                        scanf("%d%d",&a,&b);
                        printf("%d\n",find(a,b));
                }
                printf("\n");
        }
        return 0;
}
```

(2)FFT

```
//卷积
/**
  * Given two sequences {a1, a2, a3.. an} and {b1, b2, b3... bn},
  * their repeat convolution means:
  * r1 = a1*b1 + a2*b2 + a3*b3 + ... + an*bn
  * r2 = a1*bn + a2*b1 + a3*b2 + ... + an*bn-1
  * r3 = a1*bn-1 + a2*bn + a3*b1 + ... + an*bn-2
  * ...
  * rn = a1*b2 + a2*b3 + a3*b4 + ... + an-1*bn + an*b1
  * Notice n >= 2 and n must be power of 2.
  */
#include <vector>
#include <complex>
#include <cmath>
#define for if (0); else for
using namespace std;

const int MaxFastBits = 16;
int **gFFTBitTable = 0;

int NumberOfBitsNeeded(int PowerOfTwo) {
```

用 FinePrint 打印 - 可在 www.fineprint.com.cn 订购
PDF 文件使用 "pdfFactory Pro" 试用版本创建 www.fineprint.com.cn

```
        for (int i = 0;; ++i) {
            if (PowerOfTwo & (1 << i)) {
                return i;
            }
        }
    }

    int ReverseBits(int index, int NumBits) {
        int ret = 0;
        for (int i = 0; i < NumBits; ++i, index >>= 1) {
            ret = (ret << 1) | (index & 1);
        }
        return ret;
    }

    void InitFFT() {
        gFFTBitTable = new int *[MaxFastBits];
        for (int i = 1, length = 2; i <= MaxFastBits; ++i, length <<= 1) {
            gFFTBitTable[i - 1] = new int[length];
            for (int j = 0; j < length; ++j) {
                gFFTBitTable[i - 1][j] = ReverseBits(j, i);
            }
        }
    }
    inline int FastReverseBits(int i, int NumBits) {
        return NumBits <= MaxFastBits ? gFFTBitTable[NumBits - 1][i] : ReverseBits(i,
    NumBits);
    }

    void FFT(bool InverseTransform, vector<complex<double> >& In,
    vector<complex<double> >& Out) {
        if (!gFFTBitTable) { InitFFT(); }
        // simultaneous data copy and bit-reversal ordering into outputs
        int NumSamples = In.size();
        int NumBits = NumberOfBitsNeeded(NumSamples);
        for (int i = 0; i < NumSamples; ++i) {
            Out[FastReverseBits(i, NumBits)] = In[i];
        }
        // the FFT process
        double angle_numerator = acos(-1.) * (InverseTransform ? -2 : 2);
        for (int BlockEnd = 1, BlockSize = 2; BlockSize <= NumSamples; BlockSize
    <<= 1) {
            double delta_angle = angle_numerator / BlockSize;
            double sin1 = sin(-delta_angle);
            double cos1 = cos(-delta_angle);
            double sin2 = sin(-delta_angle * 2);
            double cos2 = cos(-delta_angle * 2);
            for (int i = 0; i < NumSamples; i += BlockSize) {
                complex<double> a1(cos1, sin1), a2(cos2, sin2);
                for (int j = i, n = 0; n < BlockEnd; ++j, ++n) {
                    complex<double> a0(2 * cos1 * a1.real() - a2.real(), 2 * cos1 *
    a1.imag() - a2.imag());
                    a2 = a1;
                    a1 = a0;
                    complex<double> a = a0 * Out[j + BlockEnd];
                    Out[j + BlockEnd] = Out[j] - a;
                    Out[j] += a;
                }
```

```
                }
                BlockEnd = BlockSize;
            }
            // normalize if inverse transform
            if (InverseTransform) {
                for (int i = 0; i < NumSamples; ++i) {
                    Out[i] /= NumSamples;
                }
            }
    }

    vector<double> convolution(vector<double> a, vector<double> b) {
        int n = a.size();
        vector<complex<double> > s(n), d1(n), d2(n), y(n);
        for (int i = 0; i < n; ++i) {
            s[i] = complex<double>(a[i], 0);
        }
        FFT(false, s, d1);
        s[0] = complex<double>(b[0], 0);
        for (int i = 1; i < n; ++i) {
            s[i] = complex<double>(b[n - i], 0);
        }
        FFT(false, s, d2);
        for (int i = 0; i < n; ++i) {
            y[i] = d1[i] * d2[i];
        }
        FFT(true, y, s);
        vector<double> ret(n);
        for (int i = 0; i < n; ++i) {
            ret[i] = s[i].real();
        }
        return ret;
    }

    int main() {
        double a[4] = {1, 2, 3, 4}, b[4] = {1, 2, 3, 4};
        vector<double> r = convolution(vector<double>(a, a + 4), vector<double>(b, b +
    4));
        // r[0] = 30 (1*1 + 2*2 + 3*3 + 4*4)
        // r[1] = 24 (1*4 + 2*1 + 3*2 + 4*3)
        // r[2] = 22 (1*3 + 2*4 + 3*1 + 4*2)
        // r[3] = 24 (1*2 + 2*3 + 3*4 + 4*1)
        return 0;
    }
    ////////////////////////////////////////////////////////////////////////////////////////////////////

    //多项式乘法

    #include <stdio.h>
    #include <math.h>
    #define MAX 65536
    #define EPS 1e-8
    double pi = acos(-1);
    double   cof1[MAX], cof2[MAX];
    int n, k, permutation[MAX];
    struct complex{
        double r, v;
```

用 FinePrint 打印 - 可在 www.fineprint.com.cn 订购
PDF 文件使用 "pdfFactory Pro" 试用版本创建 www.fineprint.com.cn

```
	complex operator + (complex& obj){
		complex temp;
		temp.r = r + obj.r;
		temp.v = v + obj.v;
		return temp;
	}
	complex operator - (complex& obj){
		complex temp;
		temp.r = r - obj.r;
		temp.v = v - obj.v;
		return temp;
	}
	complex operator * ( complex& obj){
		complex temp;
		temp.r = r*obj.r - v*obj.v;
		temp.v = r*obj.v + v*obj.r;
		return temp;
	}
}p1[MAX], p2[MAX], omiga[MAX], result1[MAX], result2[MAX];
void caculate_permutation(int s, int interval, int w, int next){
	if(interval==n){
		permutation[w] = s;
		return ;
	}
	caculate_permutation(s,interval*2, w, next/2);
	caculate_permutation(s+interval, interval*2, w+next, next/2);
}
void fft(complex transform[], complex p[]){
	int i, j, l, num, m;
	complex temp1, temp2;
	for(i=0;i<n;i++)transform[i] = p[ permutation[i] ] ;
	num = 1, m = n;
	for(i=1;i<=k;i++){
		for(j=0;j<n;j+=num*2)
			for(l=0;l<num;l++)
				temp2 = omiga[m*l]*transform[j+l+num],
				temp1 = transform[j+l],
				transform[j+l] = temp1 + temp2,
				transform[j+l+num] = temp1 - temp2;
		num*=2,m/=2;
	}
}


void polynomial_by(int n1,int n2){
	int i;
	double angle;
	k = 0, n = 1;
	while(n<n1+n2-1)k++,n*=2;
	for(i=0;i<n1;i++)p1[i].r = cof1[i], p1[i].v = 0;
	while(i<n)p1[i].r = p1[i].v = 0, i++;
	for(i=0;i<n2;i++)p2[i].r = cof2[i], p2[i].v = 0;
	while(i<n)p2[i].r = p2[i].v = 0, i++;
	caculate_permutation(0,1,0,n/2);
	angle = pi/n;
	for(i=0;i<n;i++)omiga[i].r = cos(angle*i), omiga[i].v = sin(angle*i);
	fft(result1,p1);
	fft(result2,p2);
```

```
	for(i=0;i<n;i++)result1[i] = result1[i]*result2[i];
	for(i=0;i<n;i++)omiga[i].v = -omiga[i].v;
	fft(result2, result1);
	for(i=0;i<n;i++)result2[i].r/=n;
	i = n -1;
	while(i&&fabs(result2[i].r)<EPS)i--;
	n = i+1;
	while(i>=0)  cof1[i] = result2[i].r, i--;
}
```

(3)最大团

```
#include <stdio.h>
bool  con[50][50],found;
int max,n,c[50];
void maxclique(int *que,int l,int    size){
	if(!l){
		if(size>max)max=size,found=true;
		return ;
	}
	int tque[50],i,j,k;
	for(i=0;i<l;i++){
		if(c[que[i]]+size<=max)return ;
		for(j=i+1,k=0;j<l;j++)if(con[que[i]][que[j]])tque[k++]=que[j];
		maxclique(tque,k,size+1);
		 if(found)return ;
	}
}
int main(){
	int i,j,que[50],l;
	while(1){
		scanf("%d",&n);
		if(!n)return 1;
		for(i=0;i<n;i++)for(j=0;j<n;j++)scanf("%d",&con[i][j]);
		max=0;
		for(i=n-1;i>=0;i--){
			found=false ;
			for(j=i+1,l=0;j<n;j++)if(con[i][j])que[l++]=j;
			maxclique(que,l,1);
			c[i]=max;
		}
		printf("%d\n",max);
	}
}
```

排序

(1)快速排序(找第 n 大数)

```
//快速排序
#include <stdlib.h>
void qsort(int array[],int s,int e){
	if(s>=e)return ;
	int p=s,q=e,t;

	//随机取一个下标,取其为基准
```

用 FinePrint 打印 - 可在 www.fineprint.com.cn 订购
PDF 文件使用 "pdfFactory Pro" 试用版本创建 www.fineprint.com.cn

```
        int k = (rand())%(e-s);
        t = array[s+k];
        array[s+k] = array[s];

        while(p<q){
            while( p<q&&array[q]>=t )q--;
            array[p] = array[q];
            while( p<q&&array[p]<=t )p++;
            array[q] = array[p];
        }
        array[p] = t;
        qsort(array,s,p-1);
        qsort(array,p+1,e);
    }
```

(2)归并排序(逆序数)

```
//归并排序,实现起来要附加数组
(1)递归实现
#define MAX 1000
int temp[MAX] ;
//int tot; //统计逆序数
void merge(int a[],int first, int mid, int last){
    int n=0,start=mid+1;
    while(first<=mid||start<=last)
        if(first<=mid&&start<=last){
            if(a[first]<a[start])temp[n++]=a[first++];
            else  temp[n++]=a[start++];//tot  +=  mid-first+1;//统计逆序数.保证元素
不同
        }
        else if(first<=mid)temp[n++]=a[first++];
        else temp[n++]=a[start++];
    while(n)a[last--]=temp[--n];
}
void mergesort(int a[], int first ,int last){
    int mid;
    if(first<last)mid = (first+last)/2, mergesort(a,first,mid), mergesort(a,mid+1,last),
merge(a,first,mid,last);
}
//Byinkfish
#include <string.h>
double inv(int n, int*a){
    int L=n>>1,R=n-L,i,j;
    int *v=new int(n);
    double ret=(R>1?(inv(L,a)+inv(R,a+L): 0 );
    for(i=j=0;i<=L;v[i+j]=a[i],i++)
        for(ret+=j;j<R&&(i==L||a[i]>a[L+j]);v[i+j]=a[L+j],j++);
    memcpy(a,v,sizeof(int)*n);
    return ret;
}
(2)非递归实现
#define MAX 5000
int temp[MAX];
void mergesort(int a[],int n){
    int gap=1,end1,end2,i,j,m;
    while(gap<n){
        i=0;
```

```
        while(i<n){
            j=end1=i+gap;
            if(end1>=n)break;
            end2=j+gap;
            if(end2>n)end2=n;
            m=0;
            while(i<end1||j<end2){
                if(i<end1&&j<end2){
                    if(a[i]<=a[j])temp[m++]=a[i++];
                    else  temp[m++]=a[j++];//amount+=end1-i;统计逆序数.保证元
素不同

                }
                else if(i<end1)temp[m++]=a[i++];
                else temp[m++]=a[j++];
            }
            i=end2;
            while(m)a[--end2] = temp[--m] ;
        }
        gap*=2;
    }
}
```

(3)希尔排序

```
//希尔排序,又叫缩小增量排序
void shellsort(int s[],int n){
    int gap = (n+1)/2, i, j, temp;
    do {
        for(i = gap; i < n; i++){
            j = i, temp = s[i];
            while(j >= gap && temp < s[j-gap])
                s[j] = s[j-gap], j -=gap ;
            s[j] = temp;
        }
        gap = gap==1?0:(gap+1)/2;
    }while(gap);
}
```

(4)基数排序

```
//对整数的基数排序  10*(n+10)
void radixsort(int array[], int n){
    int i, k, *index, *temp, first, head[10], tail[10], power1=10, power2 = 1,sign = 1;
     //数组下标，静态链表的后一项
    index = new int[n];
    first = 0;
    for(i=0;i<n-1;i++)index[i]=i+1;
    index[n-1] = -1;
    //临时数组
    temp = new int[n];
    for(i=0;i<n;i++)temp[i] = array[i];
    while(sign){
        for(i=0;i<10;i++)head[i] = -1;
        i = first;
        //盒子排序做法
```

用 FinePrint 打印 - 可在 www.fineprint.com.cn 订购
PDF 文件使用 "pdfFactory Pro" 试用版本创建 www.fineprint.com.cn

```
                while(i!=-1){
                        //取数位
                        k = temp[i];
                        k = (k%power1)/power2;
                        if(head[k]==-1)head[k] = i;
                        else index[ tail[k] ] = i;
                        tail[k] = i;
                        //下一个数
                        i = index[i] ;
                }
                //收拾
                i = 9;
                while(head[i] == -1)i--;
                first = head[i] ;
                k = tail[i];
                //已经排好了，其实做多了一趟排序
                if(i == 0)sign = 0;
                while(--i>=0)if(head[i]!=-1)index[k] = head[i],k = tail[i];
                index[k] = -1, power1*=10, power2*=10;
        }
          i = n-1;
        while(first!=-1)array[i--] = temp[first], first = index[first];
        delete temp;
        delete index;
        return ;
}
```

(5)STL 的 sort 函数

```
#include <algorithm>
using namespace std;
type data[n];
////////////////////////////////////////////////////////
bool operator <(const type& t1, const type& t2){
        return t1.value<t2.value;
}
sort(data,data+n);
或
bool cmp(const type& t1, const type& t2){
        return t1.value<t2.value;
}
sort(data,data+n,cmp);
/////////////////////////////////////////////////////////////////
```

用 FinePrint 打印 - 可在 www.fineprint.com.cn 订购
PDF 文件使用 "pdfFactory Pro" 试用版本创建 www.fineprint.com.cn