# Prim算法求最小生成树，原图为密集图，存储在邻接矩阵dis中

# 输出为vector数组形式，存储在G中

# cost\_MST为最小生成树代价

# 需要全局数组visit[ ] ,pre[ ]

void prim(void)

{

memset(visit,0,sizeof(visit));

visit[1] = true;

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

{

pre[i] = 1;

b[i] = dis[1][i];

}

pre[1] = -1;

int min\_id;

double minn = 1000000;

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

{

minn = 100000000;

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

if(!visit[i] && b[i] < minn)

{

minn = b[i];

min\_id = i;

}

visit[min\_id] = true;

cost\_MST += minn;

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

if(!visit[j] && dis[min\_id][j] < b[j])

{

b[j] = dis[min\_id][j];

pre[j] = min\_id;

}

}

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

{

int to = pre[i];

G[to].push\_back(i);

G[i].push\_back(to);

}

}

# C++ 求两个集合的交集和并集

#include<bits/stdc++.h>

using namespace std;

set<int> s1{1,2,3,5};

set<int> s2{5,7,9};

int main(void)

{

    vector<int> ANS;

    set\_union(s1.begin(),s1.end(),s2.begin(),s2.end(),inserter(ANS,ANS.begin()));

    for(auto it:ANS)

        cout<<it<<endl;

    return 0;

}

**交集为set\_intersection**

# 矩阵快速幂

struct Mat

{

    int line;

    int column;

    int a[4][4];//注意!!!根据题目需要更改大小

    Mat()

    {

        line=column=2;//同上！！！这里是初始化为单位矩阵，但是在正式写代码时不建议用这个，用下面那个比较好

        a[1][1]=1;

        a[1][2]=0;

        a[2][1]=0;

        a[2][2]=1;

    }

    Mat(int x,int y)

    {

        line=x;

        column=y;

        memset(a,0,sizeof(a));

    }

};

Mat MulMat(Mat A,Mat B,int MOD)//A's column and B's line must be equal

{

    int aa,bb,cc;

    aa=A.line;

    bb=A.column;

    cc=B.column;

    Mat C(aa,cc);

    for(int i=1;i<=aa;i++)

    {

        for(int j=1;j<=cc;j++)

        {

            for(int k=1;k<=bb;k++)

            {

                C.a[i][j]+=(A.a[i][k]\*B.a[k][j])%MOD;

                C.a[i][j]%=MOD;

            }

        }

    }

    return C;

}

Mat FastMod(Mat A,int n,int Mod)

{

    Mat E;

    if(n==0)

        return E;

    if(n==1)

        return A;

    if(n%2==1)

    {

        //printf("step1\n");

        return(MulMat(A,FastMod(A,n-1,Mod),Mod));

    }

    else

    {

        Mat temp=FastMod(A,n/2,Mod);

        //printf("step 2\n");

        return(MulMat(temp,temp,Mod));

    }

}

# 欧拉筛法 分解质因数

const int maxn=32767;

int prime[50],isprime[maxn];

struct P\_factor

{

    int p,k;

    P\_factor()

    {

        p=k=0;

    }

    P\_factor(int x,int y)

    {

        p=x;

        k=y;

    }

};

vector <P\_factor> divide\_factor(int x)

{

    vector <P\_factor> R;

    R.clear();

    for(int i=1;i<=prime[0];i++)

    {

        int p=prime[i];

        if(p\*p>x)

            break;

        if(x%p==0)

        {

            int k=0;

            while(x%p==0)

            {

                x/=p;

                k++;

            }

            R.push\_back(P\_factor(p,k));

        }

    }

    if(x>1)

        R.push\_back(P\_factor(x,1));

    return R;

}

void Prepare\_Prime(void)

{

    memset(isprime,1,sizeof(isprime));

    isprime[1]=0;

    prime[0]=0;

    for(int i=2;i<maxn;i++)

    {

        if(isprime[i])

        {

            prime[++prime[0]]=i;

        }

        for(int j=1;j<=prime[0];j++)

        {

            if(i\*prime[j]>=maxn)

                break;

            isprime[i\*prime[j]]=0;

            if(i%prime[j]==0)

                break;

        }

    }

}

**vector的取出**

vector <P\_factor> ANS=divide\_factor(x);

        if(t!=0)

            printf("\n");

        t++;

        while(!ANS.empty())

        {

            P\_factor temp=ANS.back();

            ANS.pop\_back();

            printf("%d %d ",temp.p,temp.k);

        }

# 实数的gcd

double gcd(double x,double y)

{

    while(fabs(x)>eps && fabs(y)>eps)//此处的eps根据需要取

    {

        if(x>y)

            x-=floor(x/y)\*y;

        else

            y-=floor(y/x)\*x;

    }

    return x+y;

}

# 向量叉积

double cross(const Point& p1,const Point& p2,const Point& q1,const Point& q2)

{

    return (p2.x-p1.x)\*(q2.y-q1.y)-(p2.y-p1.y)\*(q2.x-q1.x);

}//vector cross product

# 极角排序

bool cmp(const Point& a,const Point& b)

{

    if(a.y==0 && b.y==0 && a.x\*b.x<=0)

        return a.x>b.x;

    if(a.y==0 && a.x>=0 && b.y!=0)

        return true;

    if(b.y==0 && b.x>=0 && a.y!=0)

        return false;

    if(b.y\*a.y<=0)

        return a.y>b.y;

    return cross(O,a,O,b)>0 || (cross(O,a,O,b)==0 && a.x<b.x);

}

# 得到垂心

Point get\_circumcenter(Point a,Point b,Point c)

{

    double a1=b.x-a.x,b1=b.y-a.y,c1=(a1\*a1+b1\*b1)/2;

    double a2=c.x-a.x,b2=c.y-a.y,c2=(a2\*a2+b2\*b2)/2;

    double d=a1\*b2-a2\*b1;

    return Point(a.x+(c1\*b2-c2\*b1)/d,a.y+(a1\*c2-a2\*c1)/d);

}

# Java读写挂

import java.io.IOException;

import java.io.InputStream;

import java.io.PrintWriter;

import java.util.Arrays;

import java.util.InputMismatchException;

import java.util.Random;

public class A {

 public static void main(String[] args) {

// Scanner scan = new Scanner(System.in);

  InputReader in = new InputReader(System.in);

  PrintWriter out = new PrintWriter(System.out);

  int n = in.nextInt();

  int[] a = new int[n];

  for(int i=0; i<n; ++i) {

   a[i] = in.nextInt();

  }

  Random random = new Random();

  for(int i = n-1; i >= 1; --i) {

   int x = random.nextInt(n);

   int temp = a[x];

   a[x] = a[i];

   a[i] = temp;

  }

  Arrays.sort(a);

  int min = Integer.MAX\_VALUE;

  int cnt = 0;

  for(int i=0; i<n-1; ++i) {

   int diff = Math.abs(a[i] - a[i+1]);

   if(diff < min) {

    min = diff;

    cnt = 1;

   } else if(diff == min) {

    cnt++;

   }

  }

  System.out.println(min + " " + cnt);

  out.close();

// scan.close();

 }

}

class InputReader {

 private final InputStream stream;

 private final byte[] buf = new byte[8192];

 private int curChar, snumChars;

 public InputReader(InputStream st) {

  this.stream = st;

 }

 public int read() {

  if (snumChars == -1)

   throw new InputMismatchException();

  if (curChar >= snumChars) {

   curChar = 0;

   try {

    snumChars = stream.read(buf);

   } catch (IOException e) {

    throw new InputMismatchException();

   }

   if (snumChars <= 0)

    return -1;

  }

  return buf[curChar++];

 }

 public int nextInt() {

  int c = read();

  while (isSpaceChar(c)) {

   c = read();

  }

  int sgn = 1;

  if (c == '-') {

   sgn = -1;

   c = read();

  }

  int res = 0;

  do {

   res \*= 10;

   res += c - '0';

   c = read();

  } while (!isSpaceChar(c));

  return res \* sgn;

 }

 public long nextLong() {

  int c = read();

  while (isSpaceChar(c)) {

   c = read();

  }

  int sgn = 1;

  if (c == '-') {

   sgn = -1;

   c = read();

  }

  long res = 0;

  do {

   res \*= 10;

   res += c - '0';

   c = read();

  } while (!isSpaceChar(c));

  return res \* sgn;

 }

 public int[] nextIntArray(int n) {

  int a[] = new int[n];

  for (int i = 0; i < n; i++) {

   a[i] = nextInt();

  }

  return a;

 }

 public String readString() {

  int c = read();

  while (isSpaceChar(c)) {

   c = read();

  }

  StringBuilder res = new StringBuilder();

  do {

   res.appendCodePoint(c);

   c = read();

  } while (!isSpaceChar(c));

  return res.toString();

 }

 public String nextLine() {

  int c = read();

  while (isSpaceChar(c))

   c = read();

  StringBuilder res = new StringBuilder();

  do {

   res.appendCodePoint(c);

   c = read();

  } while (!isEndOfLine(c));

  return res.toString();

 }

 public boolean isSpaceChar(int c) {

  return c == ' ' || c == '\n' || c == '\r' || c == '\t' || c == -1;

 }

 private boolean isEndOfLine(int c) {

  return c == '\n' || c == '\r' || c == -1;

 }

}

# KMP(gzp)

int\* buildNext(char \*P)

{

    int m=strlen(P),j=0;

    int \*N=new int[m];

    int t=N[0]=-1;

    while(j<m-1)

    {

        if(0>t||P[j]==P[t])

        {

            j++;

            t++;

            N[j]=t;

        }

        else

            t=N[t];

    }

    return N;

}

int match(char \*P,char \*T)

{

    int \*next=buildNext(P);

    int n=strlen(T);

    int m=strlen(P);

    int i=0,j=0;

    while(j<m&&i<n)

        if(0>j||T[i]==P[j])

    {

        i++;

        j++;

    }

    else

        j=next[j];

    delete []next;

    if(i==n)//如果i指针到了文本串末尾，说明此时j为i的后缀匹配个数

        return j;

    else

        return 0;

}

# 快速幂函数

long long power(long long a,long long n,long long Mod)

{

    long long pow=1;

    long long p=a%Mod;

    while(n>0)

    {

        if(n&1)

            pow=(pow\*p)%Mod;

        n>>=1;

        p=(p\*p)%Mod;

    }

    return (pow%Mod);

}

# 区间修改的离线查询

## 给出长度为n的序列，m次修改，每次在区间[xi,yi]加上一个数ci，求最后每个数是多少。

## 活用前缀和的思想，如果现在xi出加上ci，yi +1处减去ci，最后用前缀和扫一遍过去后，就相当于在[xi,yi]上加上ci。

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

    while(m--)

    {

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

        a[x]+=c; A[y+1]-=c;

    }

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

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

    printf("%d\n",a[n]);

## 除了整段同时加一个数外，对于整段加上一个等差数列，我们也可以用类似的方法处理。

## 给出长度为n的序列，m次修改，每次在区间[xi,yi]加上一个首项是ri,公差是di的等差数列，求最后每个数是多少。

## 用两个数组，一个记公差di，一个记最终答案

for (int i=0; i<=n; i++) d[i]=a[i]=0;

    while(m--)

    {

        int x,y,r,td; scanf("%d%d%d%d",&x,&y,&r,&td);

        d[x]+=td; d[y+1]-=td;

                        //相当于整段加上同一个数

        a[x]+=r-td; a[y+1]-=(y-x)\*td+ri;

                         //a[x]加去(首项-公差)，a[y+1]减去末项

    }

    for (int i=1; i<=n; i++)

    {

        d[i]+=d[i-1];

        a[i]+=a[i-1]+d[i];

    }

# 求1 - n的异或

# f(1, n) = f(0, n) =

# n n % 4 == 0

# 1 n % 4 == 1

# n +1 n % 4 == 2

# 0 n % 4 == 3

# C++读写挂

#include <bits/stdc++.h>

// #define test TEST

using namespace std;

const int mod=1e9+7;

const int maxn=1e6+6;

typedef long long ll;

namespace IO {

#define BUF\_SIZE 100000

//fread -> read

bool IOerror = 0;

inline char nc() {

    static char buf[BUF\_SIZE], \*p1 = buf + BUF\_SIZE, \*pend = buf + BUF\_SIZE;

    if (p1 == pend) {

        p1 = buf;

        pend = buf + fread(buf, 1, BUF\_SIZE, stdin);

        if (pend == p1) {

            IOerror = 1;

            return -1;

        }

    }

    return \*p1++;

}

inline bool blank(char ch) {

    return ch == ' ' || ch == '\n' || ch == '\r' || ch == '\t';

}

inline bool read(int &x) {

    char ch;

    while (blank(ch = nc()));

    if (IOerror)

        return false;

    for (x = ch - '0'; (ch = nc()) >= '0' && ch <= '9'; x = x \* 10 + ch - '0');

    return true;

}

#undef BUF\_SIZE

};

int main(int argc, char const \*argv[])

{

    #ifdef test

    freopen("test.txt","r",stdin);

    #endif

    int n;

    while(IO::read(n)){

        printf("%d\n",n);

    }

    return 0;

}

# 逆元求组合数

void init(){

    fac[0]=1;

    for(int i=1;i<maxn;i++){

        fac[i]=fac[i-1]\*i%mod;

    }

    inv[1]=1;

    for(int i=2;i<maxn;i++){

        inv[i]=(mod-mod/i)\*inv[mod%i]%mod;

    }

    inv[0]=1;

    for(int i=1;i<maxn;i++){

        inv[i]=inv[i-1]\*inv[i]%mod;

    }

}

ll C(ll a,ll b){

    return fac[a]\*inv[b]%mod\*inv[a-b]%mod;

}

# C++ 高精度

class BigNum

{

private:

    int a[500]; //可以控制大数的位数

    int len;

public:

    BigNum(){len=1;memset(a,0,sizeof(a));} //构造函数

    BigNum(const int); //将一个int类型的变量转化成大数

    BigNum(const \_\_int64); //将一个\_\_int64类型的变量转化成大数

    BigNum(const char\*); //将一个字符串类型的变量转化为大数

    BigNum(const BigNum &); //拷贝构造函数

    BigNum &operator=(const BigNum &); //重载赋值运算符，大数之间进行赋值运算

    friend istream& operator>>(istream&,BigNum&); //重载输入运算符

    friend ostream& operator<<(ostream&,BigNum&); //重载输出运算符

    BigNum operator+(const BigNum &)const; //重载加法运算符，两个大数之间的相加运算

    BigNum operator-(const BigNum &)const; //重载减法运算符，两个大数之间的相减运算

    BigNum operator\*(const BigNum &)const; //重载乘法运算符，两个大数之间的相乘运算

    BigNum operator/(const int &)const; //重载除法运算符，大数对一个整数进行相除运算

    BigNum operator/(const \_\_int64 &)const; //重载除法运算符，大数对一个64位整数进行相除运算

    BigNum operator^(const int &)const; //大数的n次方运算

    int operator%(const int &)const; //大数对一个int类型的变量进行取模运算

    \_\_int64 operator%(const \_\_int64 &)const; //大数对一个\_\_int64类型的变量进行取模运算

    bool operator>(const BigNum &T)const; //大数和另一个大数的大小比较

    bool operator>(const int &t)const; //大数和一个int类型的变量的大小比较

    bool operator>(const \_\_int64 &t)const; //大数和一个\_\_int64类型的变量的大小比较

    void print(); //输出大数

};

BigNum::BigNum(const int b) //将一个int类型的变量转化为大数

{

    int c,d=b;

    len=0;

    memset(a,0,sizeof(a));

    while(d>MAXN) //每位存四位数字，MAXN=9999；

    {

        c=d-(d/(MAXN+1))\*(MAXN+1);

        d=d/(MAXN+1);

        a[len++]=c;

    }

    a[len++]=d;

}

BigNum::BigNum(const \_\_int64 b) //将一个int类型的变量转化为大数

{

    \_\_int64 c,d=b;

    len=0;

    memset(a,0,sizeof(a));

    while(d>MAXN) //每位存四位数字，MAXN=9999；

    {

        c=d-(d/(MAXN+1))\*(MAXN+1);

        d=d/(MAXN+1);

        a[len++]=c;

    }

    a[len++]=d;

}

BigNum::BigNum(const char \*s) //将一个字符串类型的变量转化为大数

{

    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) //拷贝构造函数

{

    int i;

    memset(a,0,sizeof(a));

    for(i=0;i<len;i++)

        a[i]=T.a[i];

}

BigNum & BigNum::operator=(const BigNum &n) //重载赋值运算符，大数之间赋值运算

{

    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)

{

    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) //重载输出运算符

{

    int i;

    cout<<b.a[b.len-1];

    for(i=b.len-2;i>=0;i--)

    {

        printf("%04d",b.a[i]);

    }

    return out;

}

BigNum BigNum::operator+(const BigNum &T)const //两个大数之间的相加运算

{

    BigNum t(\*this);

    int i,big;

    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 //两个大数之间的相减运算

{

    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[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 //两个大数之间的相乘

{

    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 //大数对一个整数进行相除运算

{

    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;

}

BigNum BigNum::operator/(const \_\_int64 &b)const //大数对一个整数进行相除运算

{

    BigNum ret;

    \_\_int64 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 //大数对一个 int类型的变量进行取模

{

    int i,d=0;

    for(i=len-1;i>=0;i--)

        d=((d\*(MAXN+1))%b+a[i])%b;

    return d;

}

\_\_int64 BigNum::operator%(const \_\_int64 &b)const //大数对一个 64位整数类型的变量进行取模

{

    \_\_int64 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 //大数的n次方运算

{

    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;

}

bool BigNum::operator>(const BigNum &T)const //大数和另一个大数的大小比较

{

    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 int &t)const //大数和一个int类型的变量的大小比较

{

    BigNum b(t);

    return \*this>b;

}

void BigNum::print() //输出大数

{

    int i;

    printf("%d",a[len-1]);

    for(i=len-2;i>=0;i--)

      printf("%04d",a[i]);

    printf("\n");

}

# 树状数组

void add(int k,int num)

{

    while(k<=n)

    {

        tree[k]+=num;

        k+=k&-k;

    }

}

int read(int k)//1~k的区间和

{

 int sum=0;

 while(k)

 {

  sum+=tree[k];

  k-=k&-k;

 }

 return sum;

}

# 拓扑排序

#include<cstdio>

#include<iostream>

#include<cstring>

#include<string>

#include<vector>

#include<algorithm>

using namespace std;

vector<int> G[110];

int n,m;

int c[110],topo[110];

int cnt;

bool flag;

bool dfs(int x)

{

 c[x]=-1;

 for(auto it:G[x])

 {

  if(c[it]==-1)

   return false;

  if((!c[it]) && !dfs(it))

   return false;

 }

 c[x]=1;

 topo[--cnt]=x;

 return true;

}

int main(void)

{

 int aa,bb;

 while(scanf("%d%d",&n,&m)!=EOF)

 {

  if(n==0)

   break;

  for(int i=1;i<=n;i++)

   G[i].clear();

  for(int i=0;i<m;i++)

  {

   scanf("%d%d",&aa,&bb);

   G[aa].push\_back(bb);

  }

  cnt=n;

  memset(c,0,sizeof(c));

  for(int i=1;i<=n;i++)

  {

   if(!c[i])

    if(!dfs(i))

     flag=false;

  }

  for(int i=0;i<n;i++)

   printf("%d ",topo[i]);

  printf("\n");

 }

 return 0;

}

# Trie

**struct Trie**

**{**

**int tot,root,go[2][maxn],cnt[maxn];**

**void init(void)**

**{**

**memset(go,-1,sizeof(go));**

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

**tot=0;**

**root=create();**

**}**

**int create(void)**

**{**

**go[0][tot]=go[1][tot]=-1;**

**return tot++;**

**}**

**void insert(long long x)**

**{**

**int now=root;**

**for(int i=31;i>=0;i--)**

**{**

**int flag=x&bits[i];**

**flag=!!flag;**

**//cout<<"x="<<x<<" i="<<i<<" flag="<<flag<<endl;**

**if(go[flag][now]==-1)**

**go[flag][now]=create();**

**now=go[flag][now];**

**cnt[now]++;**

**}**

**}**

**void dele(long long x)**

**{**

**int now=root;**

**for(int i=31;i>=0;i--)**

**{**

**int flag=x&bits[i];**

**flag=!!flag;**

**now=go[flag][now];**

**cnt[now]--;**

**}**

**}**

**long long solve(long long x)**

**{**

**long long ans=0;**

**int now=root;**

**for(int i=31;i>=0;i--)**

**{**

**int flag=x&bits[i];**

**flag=!!flag;**

**if(go[flag][now]==-1)**

**now=go[!flag][now];**

**else**

**{**

**int tempnow=go[flag][now];**

**if(cnt[tempnow]>0)**

**{**

**ans+=pow2[i];**

**now=tempnow;**

**}**

**else**

**now=go[!flag][now];**

**}**

**}**

**return ans;**

**}**

**};**

**Trie mytrie;**

# 次短路

#include<cstdio>

#include<iostream>

#include<cstring>

#include<queue>

#include<string>

#include<vector>

#include<cmath>

#include<algorithm>

using namespace std;

const long long INF=1e16;

struct Node

{

 int id,flag;

 long long val;

 Node(int id=0,int flag=0,long long val=INF):id(id),flag(flag),val(val)

 {

 }

 bool operator <(const Node& X)const

 {

  return this->val>X.val;

 }

};

struct Edge

{

 int to;

 long long val;

 Edge(int to=-1,long long val=INF):to(to),val(val)

 {

 }

};

int n,m;

const int maxn=100000+10;

vector<Edge> G[100000+10];

bool visit[2][maxn];

long long dis[2][maxn];

priority\_queue<Node> pq;

void init(void)

{

 for(int i=0;i<maxn;i++)

  G[i].clear();

 while(!pq.empty())

  pq.pop();

 memset(visit,0,sizeof(visit));

 for(int i=0;i<maxn;i++)

 {

  dis[0][i]=dis[1][i]=INF;

 }

}

int main(void)

{

 int T;

 int aa,bb,ww;

 scanf("%d",&T);

 while(T--)

 {

  init();

  scanf("%d%d",&n,&m);

  for(int i=0;i<m;i++)

  {

   scanf("%d%d%d",&aa,&bb,&ww);

   G[aa].push\_back(Edge(bb,(long long)ww));

   G[bb].push\_back(Edge(aa,(long long)ww));

  }

  dis[0][1]=0;

  pq.push(Node(1,0,0));

  while(!pq.empty())

  {

   Node TEMP=pq.top();

   pq.pop();

   int id\_now=TEMP.id;

   long long dis\_now=TEMP.val;

   int flag\_now=TEMP.flag;

   visit[flag\_now][id\_now]=true;

   for(auto it:G[id\_now])

   {

    int id\_to=it.to;

    long long distance=it.val;

    if( (!visit[0][id\_to]) && (dis\_now+distance<dis[0][id\_to]) )

    {

     dis[1][id\_to]=dis[0][id\_to];

     dis[0][id\_to]=dis\_now+distance;

     pq.push(Node(id\_to,0,dis[0][id\_to]));

    }

    else if((!visit[1][id\_to]) && (dis\_now+distance<dis[1][id\_to]))

    {

     dis[1][id\_to]=dis\_now+distance;

     pq.push(Node(id\_to,1,dis[1][id\_to]));

    }

   }

  }

  printf("%lld\n",dis[1][n]);

 }

 return 0;

}

# myk的单调栈

const int maxn=1000000+10;

struct Interval

{

  int lend,rend;

  Interval(int lend=-1,int rend=-1):lend(lend),rend(rend)

  {

  }

};

Interval minn[maxn],maxx[maxn];

minn[1].lend = 1;

maxx[1].lend = 1;

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

{

  int j = i-1;

  while(j > 0 && a[j] >= a[i])

  {

   j = minn[j].lend;

   j--;

  }

  minn[i].lend = j + 1;

  j = i - 1;

  while(j > 0 && a[j] <= a[i])

  {

   j = maxx[j].lend;

   j--;

  }

  maxx[i].lend = j + 1;

 }

  minn[n].rend = maxx[n].rend = n;

  for(int i = n - 1; i >= 1; i--)

 {

  int j = i + 1;

  while(j <= n && a[j] > a[i])

  {

   j = minn[j].rend;

   j++;

  }

  minn[i].rend = j - 1;

  j = i + 1;

  while(j <= n && a[j] < a[i])

  {

   j = maxx[j].rend;

   j++;

  }

  maxx[i].rend = j - 1;

 }

# 链式前向星

struct Edge

{

 int to,next;

 Edge(int to=-1,int next=-1):to(to),next(next)

 {

 }

};

Edge e[maxn];

void add\_edge(int from,int to)

{

 e[cnt].to=to;

 e[cnt].next=head[from];

 head[from]=cnt++;

}

for(int j=head[x];j!=-1;j=e[j].next)// For every edge from x

# STL中 next\_permutation prev\_permutation的使用

#include<bits/stdc++.h>

using namespace std;

int main(void)

{

 int a[5] = {0,1,2,3,4};

 while(next\_permutation(a,a + 5))

 {

  for(int i = 0; i < 5; i++)

   printf("%d ",a[i]);

  printf("\n");

 }

 return 0;

}

# 博弈论总结：

## 巴什博弈

# 简述

什么是巴什博弈：只有一堆n个物品，两个人轮流从这堆物品中取物， 规定每次至少取一个，最多取m个。最后取光者得胜。

# 分析

我们称先进行游戏的人为先手，另一个人为后手。

1、如果 n=m+1，那么由于一次最多只能取 m 个，所以，无论先手拿走多少个，后手都能够一次拿走剩余的物品，后者取胜。

2、如果 n=(m+1)∗r+s，(r为任意自然数， s≤m),先手要拿走s个物品，如果后手拿走 k(k≤m) 个，那么先手再拿走 m+1−k 个，结果剩下 (m+1)∗(r−1)个，以后保持这样的取法，那么先取者肯定获胜。我们得到如下结论:要保持给对手留下 (m+1)的倍数，就能最后获胜。

# 必胜态必败态

只要 nn 不能整除 m+1 ,那么必然是先手取胜，否则后手取胜。

# 变形

如果我们规定最后取光者输，那么又会如何呢？

(n−1) % (m+1)==0则后手胜利 先手会重新决定策略，所以不是简单的相反的。

例如 n=15，m=3

后手 先手 剩余

   0 2 13

   1 3 9

   2 2 5

   3 1 1

   1 0 0

先手胜利 输的人最后必定只抓走一个，如果>1个，则必定会留一个给对手

# 威佐夫博奕

# 简述

威佐夫博弈(Wythoff Game)：有两堆各若干个物品，两个人轮流从某一堆或同时从两堆中取同样多的物品，规定每次至少取一个，多者不限，最后取光者得胜。

# 分析

我们用 （ak，bk）（ak≤bk,k=0，1，2，...,n) 表示两堆物品的数量并称其为局势，如果甲面对 （0，0） ，那么甲已经输了，这种局势我们称为奇异局势。前几个奇异局势是： （0，0） 、（1，2） 、 （3，5） 、 （4，7） 、 （6，10） 、 （8，13） 、 （9，15） 、 （11，18） 、 （12，20） 。 可以看出： a0=b0=0 , ak是未在前面出现过的最小自然数,而 bk=ak+k 。

# 必胜态必败态

满足 ak=k∗（1+根号5）/2，bk=ak+k ，后手必胜，否则先手必胜。

# 无向图 全图的最小割

#include <cstdio>

#include <cstring>

#include <algorithm>

using namespace std;

#define mem(a) memset(a, 0, sizeof(a))

const int maxv = 500;

const int inf = 0x3f3f3f3f;

int v[maxv], d[maxv];

int G[maxv][maxv];

bool vis[maxv];

int Stoer\_Wanger(int n) {

    int res = inf;

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

    while (n > 1) {

        int k = 1, pre = 1;

        mem(vis); mem(d);

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

            k = -1;

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

                if ( !vis[ v[j] ] ) {

                    d[ v[j] ] += G[ v[pre] ][ v[j] ];

                    if (k == -1 || d[ v[k] ] < d[ v[j] ] ) {

                        k = j;

                    }

                }

            }

            vis[ v[k] ] = true;

            if (i == n) {

                res = min(res, d[ v[k] ]);

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

                    G[ v[pre] ][ v[j] ] += G[ v[j] ][ v[k] ];

                    G[ v[j] ][ v[pre] ] += G[ v[j] ][ v[k] ];

                }

                v[ k ] = v[ n-- ];

            }

            pre = k;

        }

    }

    return res;

}

int main() {

    int n, m, s, u, v, w;

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

        mem(G);

        for (int i = 0; i < m; i++) {

            scanf("%d%d%d", &u, &v, &w);

            G[u][v] += w;

            G[v][u] += w;

        }

        printf("%d\n", Stoer\_Wanger(n));

    }

    return 0;

}

**并查集**

int fa[N];

void init(int n) {

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

}

void unin(int u, int v) {

fa[find(v)] = find(u);

}

int find(int u) {

return fa[u] == u ? fa[u] : fa[u] = find(fa[u]);

}

**S-T表**

#include <iostream>

#include <cmath>

using namespace std;

int a[50001];

int f[50001][16];

int n;

void rmq\_init() //建立： dp(i,j) = min{dp(i,j-1),dp(i+2^(j-1),j-1) O(nlogn)

{

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

f[i][0] = a[i];

int k=floor(log((double)n)/log(2.0)); //C/C++取整函数ceil()大,floor()小

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

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

{

if(i+(1<<(j-1))<=n) //f(i,j) = min{f(i,j-1),f(i+2^(j-1),j-1)

f[i][j]=min(f[i][j-1],f[i+(1<<(j-1))][j-1]);

}

}

int rmq(int i,int j) //查询：返回区间[i,j]的最小值 O(1)

{

int k = floor(log((double)(j-i+1))/log(2.0));

return min(f[i][k],f[j-(1<<k)+1][k]);

}

int main()

{

scanf("%d",&n);

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

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

rmq\_init();

printf("%d\n", rmq(2,5));

}

# Tarjan 强联通 缩点

需要全局变量dfn low cnt\_dfs cnt\_scc sccno

栈s

void dfs(int x)

{

dfn[x]=low[x]=++cnt\_dfs;

s.push(x);

for(int j=head[x];j!=-1;j=e[j].next)

{

int to=e[j].to;

if(!dfn[to])

{

dfs(to);

low[x]=min(low[x],low[to]);

}

else if(!sccno[to])

low[x]=min(low[x],dfn[to]);

}

if(low[x]==dfn[x])

{

int tempcnt=0;

cnt\_scc++;

while(s.top()!=x)

{

sccno[s.top()]=cnt\_scc;

tempcnt++;

s.pop();

}

s.pop();

sccno[x]=cnt\_scc;

num[cnt\_scc]=tempcnt+1;

}

}

## Java

java在比赛中最常用的就是大数了。  
基本函数：  
1.valueOf(parament); 将参数转换为制定的类型  
比如 int a=3;  
BigInteger b=BigInteger.valueOf(a);  
则b=3;  
String s=”12345”;  
BigInteger c=BigInteger.valueOf(s);  
则c=12345；

2.add(); 大整数相加

BigInteger a=new BigInteger(“23”);

BigInteger b=new BigInteger(“34”);

a.add(b);

3.subtract(); 相减  
4.multiply(); 相乘  
5.divide(); 相除取整  
6.remainder(); 取余  
7.pow(); a.pow(b)=a^b  
8.gcd(); 最大公约数  
9.abs(); 绝对值  
10.negate(); 取反数  
11.mod(); a.mod(b)=a%b=a.remainder(b);  
12.max(); min();  
13.punlic int comareTo();  
14.boolean equals(); 是否相等  
15.BigInteger构造函数：  
一般用到以下两种： BigInteger(String val);  
将指定字符串转换为十进制表示形式；  
BigInteger(String val,int radix); 将指定基数的 BigInteger 的字符串表示形式转换为 BigInteger  
Ⅱ.基本常量：

A=BigInteger.ONE 1

B=BigInteger.TEN 10

C=BigInteger.ZERO 0

Ⅲ.基本操作

1. 读入： 用Scanner类定义对象进行控制台读入,Scanner类在java.util.\*包中
2. Scanner cin=new Scanner(System.in);// 读入
3. while(cin.hasNext()) //等同于!=EOF
4. {
5. int n;
6. BigInteger m;
7. n=cin.nextInt(); //读入一个int;
8. m=cin.BigInteger();//读入一个BigInteger;
9. System.out.print(m.toString());
10. }

示例：

import java.util.Scanner;

import java.math.\*;

import java.text.\*;

public class Main

{

public static void main(String args[])

{

Scanner cin = new Scanner ( System.in );

BigInteger a,b;

int c;

char op;

String s;

while( cin.hasNext() )

{

a = cin.nextBigInteger();

s = cin.next();

op = s.charAt(0);

if( op == '+')

{

b = cin.nextBigInteger();

System.out.println(a.add(b));

}

else if( op == '-')

{

b = cin.nextBigInteger();

System.out.println(a.subtract(b));

}

else if( op == '\*')

{

b = cin.nextBigInteger();

System.out.println(a.multiply(b));

}

else

{

BigDecimal a1,b1,eps;

String s1,s2,temp;

s1 = a.toString();

a1 = new BigDecimal(s1);

b = cin.nextBigInteger();

s2 = b.toString();

b1 = new BigDecimal(s2);

c = cin.nextInt();

eps = a1.divide(b1,c,4);

//System.out.println(a + " " + b + " " + c);

//System.out.println(a1.doubleValue() + " " + b1.doubleValue() + " " + c);

System.out.print( a.divide(b) + " " + a.mod(b) + " ");

if( c != 0)

{

temp = "0.";

for(int i = 0; i < c; i ++) temp += "0";

DecimalFormat gd = new DecimalFormat(temp);

System.out.println(gd.format(eps));

}

else System.out.println(eps);

}

}

}

}

# 最大流

struct Dinic

{

 int n,m,s,t;

 vector<Edge> edges;

 vector<int> G[maxn];

 bool visit[maxn];

 long long d[maxn];

 int cur[maxn];

 void init(int n)

 {

  this -> n = n;

  edges.clear();

  for(int i = 0; i < maxn; i++)

   G[i].clear();

 }

 void add\_edge(int from,int to,long long cap)

 {

  edges.push\_back(Edge(from,to,cap,0));

  edges.push\_back(Edge(to,from,0,0));

  m = edges.size();

  G[from].push\_back(m - 2);

  G[to].push\_back(m - 1);

 }

 bool BFS(void)

 {

  memset(visit,0,sizeof(visit));

  queue<int> Q;

  Q.push(s);

  d[s] = 0;

  visit[s] = true;

  while(!Q.empty())

  {

   int x = Q.front();

   Q.pop();

   for(auto it:G[x])

   {

    Edge& e = edges[it];

    if(!visit[e.to] && e.cap > e.flow)

    {

     visit[e.to] = true;

     d[e.to] = d[x] + 1;

     Q.push(e.to);

    }

   }

  }

  return visit[t];

 }

 long long DFS(int x,long long a)

 {

  if(x == t || a == 0)

   return a;

  long long flow = 0, f;

  for(int& i = cur[x]; i < G[x].size(); i++)

  {

   Edge& e = edges[G[x][i]];

   if(d[x] + 1 == d[e.to] && (f = DFS(e.to,min(a,e.cap - e.flow))) > 0)

   {

    e.flow += f;

    edges[G[x][i] ^ 1].flow -= f;

    flow += f;

    a -= f;

    if(a == 0)

     break;

   }

  }

  return flow;

 }

 long long max\_flow(int s,int t)

 {

  this -> s = s;

  this -> t = t;

  long long flow = 0;

  while(BFS())

  {

   memset(cur,0,sizeof(cur));

   flow += DFS(s, INF);

  }

  return flow;

 }

};

Dinic dinic;

# nth\_element 作用

# 把一个数组以某个元素作为pivot，使得其左边的元素都小于它，右边的元素都大于它。

# nth\_element(start, start+8, end) ;