Standard Code Library

Hao Yuan
Department of Computer Science and Engineering
Shanghai Jiaotong University

October 23, 2003

Contents

Chapter 1

Algorithms and Datastructures

1.1 High Precision in C

```
#define maxlen 1000

struct HP { int len,s[maxlen];};

void PrintHP(HP x){ for(int i=x.len;i>=1;i--) cout<<x.s[i]; }

void Str2HP(const char *s,HP &x)
{
    x.len=strlen(s);
    for(int i=1;i<=x.len;i++) x.s[i]=s[x.len-i]-'0';
}

void Int2HP(int inte,HP &x)
{
    if(inte==0) { x.len=1; x.s[1]=0; return;};
    for(x.len=0;inte>0;){ x.s[++x.len]=inte%10; inte/=10;};
}

void Multi(const HP a,const HP b,HP &c)
{
    int i,j; c.len=a.len+b.len;
    for(i=1;i<=c.len;i++) c.s[i]=0;
    for(i=1;i<=a.len;i++) for(j=1;j<=b.len;j++) c.s[i+j-1]+=a.s[i]*b.s[j];
    for(i=1;i<c.len;i++) { c.s[i+1]+=c.s[i]/10; c.s[i]%=10; }
    while(c.s[i]) { c.s[i+1]=c.s[i]/10; c.s[i]%=10; i++; }
    while(i>1 && !c.s[i]) i--; c.len=i;
}

void Plus(const HP a,const HP b,HP &c)
{
    int i; c.s[1]=0;
    for(i=1; i<=a.len || i<=b.len || c.s[i] ;i++){
        if(i<=a.len) c.s[i]+=a.s[i];
        if(i<=b.len) c.s[i]+=b.s[i];
        c.s[i+1]=c.s[i]/10;  c.s[i]%=10;
    }
    c.len=i-1;  if( c.len==0 ) c.len=1;
}

void Subtract(const HP a,const HP b,HP &c)
{
    for(int i=1,j=0; i<=a.len ;i++){
        c.s[i]=a.s[i]-j; if(i<=b.len) c.s[i]-=b.s[i];
        if(c.s[i]<0){ j=1 ; c.s[i]+=10; } else j=0;
    }
```

```
    c.len=a.len; while(c.len>1 && !c.s[c.len]) c.len--;
}

int HPCompare(const HP x,const HP y)
{
    if(x.len>y.len) return 1;
    if(x.len<y.len) return -1;
    int i=x.len;
    while((i>1)&&(x.s[i]==y.s[i])) i--;
    return x.s[i]-y.s[i];
}

void Divide(const HP a, const HP b, HP &c,HP &d)
{
    int i,j;  d.len=1; d.s[1]=0;
    for(i=a.len;i>0;i--) {
        if(!(d.len==1 && d.s[1]==0))
            { for(j=d.len;j>0;j--) d.s[j+1]=d.s[j]; ++d.len; }
        d.s[1]=a.s[i];  c.s[i]=0;
        while( (j=HPCompare(d,b))>=0 )
            { Subtract(d,b,d);  c.s[i]++;  if(j==0) break; }
    }
    c.len=a.len;  while((c.len>1)&&(c.s[c.len]==0)) c.len--;
}
```

1.2 High Precision in C Plus Plus

```
const int maxlen = 10000;

class HP { public:
    int len, s[maxlen];  HP() { (*this)=0; };
    HP(int inte) { (*this)=inte; };  HP(const char*str) { (*this)=str; };
    friend ostream& operator<<(ostream &cout,const HP &x);
    HP operator=(int inte);       HP operator=(const char*str);
    HP operator*(const HP &b);   HP operator+(const HP &b);
    HP operator-(const HP &b);   HP operator/(const HP &b);
    HP operator%(const HP &b);   int Compare(const HP &b);
};

ostream& operator<<(ostream &cout,const HP &x)
{ for(int i=x.len;i>=1;i--) cout<<x.s[i]; return cout; }

HP HP::operator=(const char *str)
{
    len=strlen(str);
    for(int i=1;i<=len;i++) s[i]=str[len-i]-'0';
    return *this;
}

HP HP::operator=(int inte)
{
    if(inte==0) { len=1; s[1]=0; return (*this);};
    for(len=0;inte>0;){ s[++len]=inte%10; inte/=10;};
    return (*this);
}

HP HP::operator*(const HP &b)
{
    int i,j; HP c; c.len=len+b.len;
    for(i=1;i<=c.len;i++) c.s[i]=0;
    for(i=1;i<=len;i++) for(j=1;j<=b.len;j++) c.s[i+j-1]+=s[i]*b.s[j];
    for(i=1;i<c.len;i++) { c.s[i+1]+=c.s[i]/10; c.s[i]%=10; }
```

```
    while(c.s[i]) { c.s[i+1]=c.s[i]/10; c.s[i]%=10; i++; }
    while(i>1 && !c.s[i]) i--; c.len=i;
    return c;
}

HP HP::operator+(const HP &b)
{
    int i;  HP c;  c.s[1]=0;
    for(i=1; i<=len || i<=b.len || c.s[i] ;i++){
        if(i<=len) c.s[i]+=s[i];
        if(i<=b.len) c.s[i]+=b.s[i];
        c.s[i+1]=c.s[i]/10;  c.s[i]%=10;
    }
    c.len=i-1;  if( c.len==0 ) c.len=1;
    return c;
}

HP HP::operator-(const HP &b)
{
    int i,j; HP c;
    for(i=1,j=0; i<=len ;i++){
        c.s[i]=s[i]-j; if(i<=b.len) c.s[i]-=b.s[i];
        if(c.s[i]<0){ j=1 ; c.s[i]+=10; } else j=0;
    }
    c.len=len; while(c.len>1 && !c.s[c.len]) c.len--;
    return c;
}

int HP::Compare(const HP &y)
{
    if(len>y.len) return 1;
    if(len<y.len) return -1;
    int i=len;
    while((i>1)&&(s[i]==y.s[i])) i--;
    return s[i]-y.s[i];
}

HP HP::operator/(const HP &b)
{
    int i,j; HP d(0),c;
    for(i=len;i>0;i--){
        if(!(d.len==1 && d.s[1]==0))
            { for(j=d.len;j>0;j--) d.s[j+1]=d.s[j]; ++d.len; }
        d.s[1]=s[i];  c.s[i]=0;
        while( (j=d.Compare(b))>=0 )
            { d=d-b;  c.s[i]++;  if(j==0) break; }
    }
    c.len=len;  while((c.len>1)&&(c.s[c.len]==0)) c.len--;
    return c;
}

HP HP::operator%(const HP &b)
{
    int i,j; HP d(0);
    for(i=len;i>0;i--){
        if(!(d.len==1 && d.s[1]==0))
            { for(j=d.len;j>0;j--) d.s[j+1]=d.s[j]; ++d.len; }
        d.s[1]=s[i];
        while( (j=d.Compare(b))>=0 ){ d=d-b; if(j==0) break; }
    }
    return d;
}
```

1.3 High Precision Floating-point Number

```
const int fprec = 100; // floating-point precision

HP zero=0;

class FS{public:
    FS();  void SetZero();
    FS(int inte) { (*this)=inte; }
    FS(char *s)  { (*this)=s;    }
    FS operator=(char *s); FS operator=(int inte);
    FS operator+(FS b); FS operator-(FS b);
    FS operator*(FS b); FS operator/(FS b);
    friend ostream& operator<<(ostream &cout, FS x);
    int sign,prec;
    HP num;
};

void FS::SetZero(){ sign=1; num=0; prec=0; }

FS::FS() { SetZero(); }

ostream& operator<<(ostream &cout,FS x)
{
    if(x.sign<0) cout<<"-";
    int i,k,low=1;
    for(i=x.num.len; i>x.prec; i--) cout<<x.num.s[i];
    if( x.num.len<=x.prec) cout<<"0";
    if( x.num.Compare(zero)==0 ) { cout<<".0"; return cout; }
    k=i;
    while( k>0 && x.num.s[k]==0 ) k--;
    if(k==0) { cout<<".0"; return cout; }
    cout<<".";
    if( x.num.len<x.prec ) for(int j=0;j<x.prec-x.num.len;j++) cout<<"0";
    while(x.num.s[low]==0) low++;
    while(i>=low ) cout<<x.num.s[i--];
    return cout;
}

FS FS::operator=(int inte)
{
    prec = 0;
    if( inte>=0 ) { sign =  1; num =  inte; }
        else      { sign = -1; num = -inte; }
    return (*this);
}

FS FS::operator=(char *s)
{
    int p,i,j,l;
    SetZero();
    if( s[0]=='-' ) { sign = -1; s++; };
    if( s[0]=='+' ) { sign =  1; s++; };
    l = strlen(s);
    for(p=0; p<l; p++) if( s[p]=='.' ) break;
    if( p==l ) prec = 0; else prec = l-1-p;
    for(i=l-1,j=0; i>=0; i--) if( s[i]!='.' ) num.s[++j] = s[i]-'0';
    while(j>1 && num.s[j]==0) --j;  num.len = j;
    return (*this);
}
```

```
void LShift(FS &a,int sl)
{
    a.prec+=sl; a.num.len+=sl; int i;
    for(i=a.num.len; i>sl; i--) a.num.s[i]=a.num.s[i-sl];
    while(i>0) a.num.s[i--]=0;
}

void RShift(FS &a,int sl)
{
    a.prec-=sl; a.num.len-=sl; int i;
    for(i=1; i<=a.num.len; i++) a.num.s[i]=a.num.s[i+sl];
}

FS FS::operator+(FS b)
{
    FS c;
    if( prec>b.prec ) LShift(b,prec-b.prec); else
    if( prec<b.prec ) LShift((*this),b.prec-prec);
    if( sign==b.sign ) {
        c.sign=sign; c.prec=prec; c.num=num+b.num;
        if( c.num.Compare(zero)==0 ) c.SetZero();
    } else  {
        c.prec=prec;
        if( num.Compare(b.num)==0) c.SetZero(); else
        if( num.Compare(b.num)>0 ) { c.sign=sign; c.num=num-b.num; } else
        if( num.Compare(b.num)<0 ) { c.sign=b.sign; c.num=b.num-num; }
        if( c.num.Compare(zero)==0 ) c.SetZero();
    }
    if( c.prec > fprec ) RShift( c, c.prec - fprec );
    return c;
}

FS FS::operator-(FS b)
{
    b.sign = - b.sign;
    FS c = (*this) + b;
    b.sign = - b.sign;
    return c;
}

FS FS::operator*(FS b)
{
    FS c;
    c.sign = sign * b.sign ;
    c.prec = prec + b.prec ;
    c.num  = num  * b.num  ;
    if( c.num.Compare(zero)==0 ) c.SetZero();
    if( c.prec > fprec ) RShift( c, c.prec - fprec );
    return c;
}

FS FS::operator/(FS b)  // 355/133 = 3.14159292035398230088495575221 24
{
    FS c,d;  // c = d / b
    d = (*this); LShift(d, fprec);
    c.sign = d.sign * b.sign ;
    c.prec = d.prec;
    LShift(d , b.prec);
    c.num =  d.num / b.num;
    if( c.prec > fprec ) RShift( c, c.prec - fprec );
    return c;
}
```

1.4 Fraction Class

```
int gcd(int a,int b){ if (b==0) return a; return gcd(b,a%b); }
int lcm(int a,int b){ return a/gcd(a,b) * b; }

class Fraction { public:
    int a,b; // ( a/b = numerator/denominator )
    int sign(int x) { return (x>0?1:-1); }
    Fraction():a(0),b(1){}
    Fraction(int x):a(x),b(1){}
    Fraction(int x,int y){
        int m = gcd(abs(x),abs(y));
        a = x/m * sign(y);
        if (a==0) b = 1; else b = abs( y/m );
    }
    int get_denominator() {return b;}
    int get_numerator() {return a;}
    Fraction operator+(const Fraction &f) {
        int m = gcd(b,f.b);
        return Fraction(f.b/m * a + b/m * f.a, b/m * f.b);
    }
    Fraction operator-(const Fraction &f) {
        int m = gcd(b,f.b);
        return Fraction(f.b/m * a - b/m * f.a, b/m * f.b);
    }
    Fraction operator*(const Fraction &f) {
        int m1 = gcd(abs(a),f.b);
        int m2 = gcd(b,abs(f.a));
        return Fraction( (a/m1)*(f.a/m2), (b/m2)*(f.b/m1) );
    }
    Fraction operator/(const Fraction &f)
        { return (*this)*Fraction(f.b,f.a); }
    friend ostream &operator<<(ostream &out,const Fraction &f) {
        if (f.a==0) cout << 0; else
            if (f.b==1) cout << f.a; else cout << f.a << '/' << f.b;
        return out;
    }
};
```

1.5 Binary Heap

```
#define MAXN 1048576
int n,HeapSize,Heap[MAXN+1];

void HeapUp(int p)
{
    int q=p>>1,a=Heap[p];
    while(q){
        if(a<Heap[q]) Heap[p]=Heap[q]; else break;
        p=q;   q=p >> 1;
    }
    Heap[p]=a;
}

void AddToHeap(int a)
{
    Heap[++HeapSize]=a;
    HeapUp(HeapSize);
}
```

```
void HeapDown(int p)
{
    int q=p<<1,a=Heap[p];
    while( q <= HeapSize ) {
        if( q<HeapSize && Heap[q+1]<Heap[q] ) q++;
        if( Heap[q] < a ) Heap[p] = Heap[q]; else break;
        p=q; q=p << 1;
    }
    Heap[p] = a;
}

int GetTopFromHeap()
{
    int TopElement = Heap[1];
    Heap[1] = Heap[HeapSize--];
    HeapDown(1);
    return TopElement;
}

void BuildHeap() // Remember to Let HeapSize = N
{ for(int i=HeapSize;i>0;i--) HeapDown(i); }
```

1.6 Winner Tree

```
const int inf=10000000;
const int maxsize=1048576; // 2^floor(log(n))

int heap[maxsize*2],pos[maxsize*2],n,base;

void Update(int i)
{
    int j=i<<1;
    if(heap[j+1]<heap[j]) j++;
    heap[i]=heap[j]; pos[i]=pos[j];
}

int GetTopFromHeap(int &ps)
{
    int ret=heap[1],p=pos[1];
    ps=p;    heap[p]=inf;
    while(p>1) { p>>=1; Update(p); }
    return ret;
}

int main()
{
    int i,j;
    cin >> n;
    for(base=1;base<n;base<<=1);    base--;
    for(i=base+1;i<=(base<<1)+1;i++) {
        pos[i]=i;
        if( i<=base+n) cin>> heap[i]; else heap[i]=inf;
    }
    for(i=base;i>0;i--) Update(i);
    for(i=1;i<=n;i++) cout<<GetTopFromHeap(j)<<endl;
    return 0;
}
```

1.7 Digital Tree

```
#define maxlen 100
#define maxsize 1000000
#define DataType int

char tree[maxsize],s[maxlen];
int son[maxsize],bro[maxsize],num,k,n;
DataType data[maxsize];

DataType find(const char*s)
{
    int i,j=0;
    for(i=0;s[i];i++) {
        j=son[j];
        while(j && tree[j]!=s[i]) j=bro[j];
        if(!j) return -1;
    }
    return data[j];
}

void add(const char*s,DataType x)
{
    int i,j=0,p;
    for(i=0;s[i];i++) {
        p=j;   j=son[j];
        while(j && tree[j]!=s[i]) j=bro[j];
        if(!j) {
            tree[++num]=s[i];  son[num]=0;
            bro[num]=son[p];   son[p]=num;
            data[num]=-1;      j=num;
        }
    }
    data[j]=x;
}

void init()
{ num=0; bro[num]=0; son[num]=0; data[0]=-1; }
```

1.8 Segment Tree

```
int cc[1 << 22],m,n; // memset cc first

void update(int ii,int s,int t,int ss,int tt,bool insert) {
    if(ss>tt) return; int mid((s+t)/2);
    if(s==ss && t==tt){ if(insert) cc[ii]=t-s+1; else cc[ii]=0; return;}
    if(cc[ii]==0) if (!insert) return; else cc[ii*2]=cc[ii*2+1]=0;
    else if(cc[ii]==t-s+1) if(insert) return;
        else { cc[ii*2]=mid-s+1; cc[ii*2+1]=t-mid; }
    update(ii*2,s,mid,ss,__min(mid,tt),insert);
    update(ii*2+1,mid+1,t,__max(mid+1,ss),tt,insert);
    cc[ii]=cc[ii*2]+cc[ii*2+1];
}

int query(int ii,int s,int t,int ss,int tt) {
    if(ss>tt) return 0; int mid((s+t)/2);
    if(s==ss && t==tt) return cc[ii];
    if(cc[ii]==0) cc[ii*2] = cc[ii*2+1] = 0;
    if(cc[ii]==t-s+1) {cc[ii*2]=mid-s+1; cc[ii*2+1]=t-mid;}
    return query(ii*2,s,mid,ss,__min(mid,tt))
         +query(ii*2+1,mid+1,t,__max(mid+1,ss),tt);
}
```

1.9 Segment Tree in IOI'2001

```
// upper : maximum possible right point of intervals
int upper, tree[maxinterval+1];

void init()
{ upper=0; memset(tree,0,sizeof(tree)); }

void update( int r,int x ) // sum[1..r] +=x
{ while(r<=upper){ tree[r]+=x; r+=(r&(r^(r-1))); } }

int sum(int r)              // return sum[1..r]
{
    int res = 0;
    while ( r>0 ) { res+=tree[r]; r-=(r&(r^(r-1))); }
    return res;
}
```

1.10 Union-Find Set

```
int rank[maxn],pnt[maxn];

void makeset(int x)
{ rank[pnt[x]=x]=0; }

int find(int x)
{
    int px=x,i;
    while(px!=pnt[px]) px=pnt[px];
    while(x!=px){ i=pnt[x]; pnt[x]=px; x=i; };
    return px;
}

void merge(int x,int y) // or just pnt[find(x)]=find(y)
{
    if( rank[x=find(x)]>rank[y=find(y)]) pnt[y]=x;
        else { pnt[x]=y; rank[y]+=(rank[x]==rank[y]); };
}
```

1.11 Quick Sort

```
void quicksort(int b,int e,int a[])
{
    int i=b,j=e,x=a[(b+e)/2];
    do{
      while(a[i]<x) i++;
      while(a[j]>x) j--;
      if(i<=j) std::swap(a[i++],a[j--]);
    }while(i<j);
    if(i<e) quicksort(i,e,a);
    if(j>b) quicksort(b,j,a);
}
```

1.12 Merge Sort

```
void sort(int b,int e)
{
    if(e-b<=0) return;
    int mid=(b+e)/2,p1=b,p2=mid+1,i=b;
    sort(b,mid); sort(mid+1,e);
    while( p1<=mid || p2<=e )
        if( p2>e|| (p1<=mid && a[p1]<=a[p2]) )
            t[i++]=a[p1++]; else t[i++]=a[p2++];
    for(i=b;i<=e;i++)a[i]=t[i];
}
```

1.13 Radix Sort

```
#define base (1<<16)
int n,a[maxn],t[maxn],bucket[base+2];

void RadixSort(int n,int a[],int t[],int bucket[])
{
    int k,i,j;
    for(j=0;j<base;j++) bucket[j]=0;
    for(k=base-1,i=0; i<2; i++,k<<=16){
        for(j=0;j<n;j++)        bucket[a[j]&k]++;
        for(j=1;j<base;j++)     bucket[j]+=bucket[j-1];
        for(j=n-1;j>=0;j--)     t[--bucket[a[j]&k]]=a[j];
        for(j=0;j<n;j++)        a[j]=t[j];
    }
}
```

1.14 Select K^{th} Smallest Element

```
int select(int* a, int b, int e, int k)
{
    if( b==e) return a[b];
    int x = a[b+rand()%(e-b+1)], i = b, j = e;
    i--; j++;
    while(i<j) {
        while( a[++i] < x ); while( a[--j] > x );
        if( i<j ) std::swap(a[i],a[j]);
    }
    if(j==e) j--; i = j-b+1;
    if( k <= i ) return select(a, b, j, k);
        else return select(a, j+1, e, k-i );
}
```

1.15 KMP

```
int fail[maxlen];

void makefail( char *t, int lt )
{
    --t;
    for(int i=1,j=0;i<=lt;i++,j++){
        fail[i]=j;
        while(j>0 && t[i]!=t[j]) j=fail[j];
    }
}

// start matching pattern T in S[i..)
// return match pos or longest match length with corresponding pos
```

```
int kmp(char *s, int ls, char *t, int lt, int i,int &longest,int &lp)
{
    longest = lp = 0; --s; --t;
    for(int j=1; i<=ls; i++,j++) {
        while( j>0 && s[i]!=t[j] ) j=fail[j];
        if( j>longest ) { longest = j; lp = i-j; }
        if( j==lt ) return i-lt;
    }
    return -1;
}
```

1.16 Suffix Sort

SuffixSort : input $s[0..n)$ **, output** $id[0..n)$

```
#define nb next       // "new bucket" overlaid on "next"
#define head height  // head is never used when computing height
#define rank b       // after SuffixSort, "rank" overlaid on "bucket"

char s[maxn]; int n, id[maxn], height[maxn], b[maxn], next[maxn];

bool cmp(const int &i,const int &j){ return s[i]<s[j]; }

void SuffixSort()
{
    int i,j,k,h;
    for(i=0; i<n; i++) id[i]=i;
    std::sort(id,id+n,cmp);
    for(i=0; i<n; i++)
        if(i==0 || s[id[i]]!=s[id[i-1]] ) b[id[i]] = i;
            else b[id[i]]=b[id[i-1]];
    for(h=1; h<n; h<<=1)
    {
        for(i=0;i<n;i++) head[i]=next[i]=-1;
        for(i=n-1; i>=0; i--) if(id[i])
        {
            j = id[i]-h; if( j<0 ) j+=n;
            next[j] = head[b[j]]; head[b[j]] = j;
        }
        j=n-h; next[j] = head[b[j]]; head[b[j]] = j;
        for(i=k=0; i<n; i++) if( head[i]>=0 )
            for(j=head[i]; j>=0; j=next[j]) id[k++]=j;
        for(i=0; i<n; i++) if( i>0 && id[i]+h<n && id[i-1]+h<n
            && b[id[i]] == b[id[i-1]] && b[id[i]+h] == b[id[i-1]+h] )
                nb[id[i]] = nb[id[i-1]]; else nb[id[i]] = i;
        for(i=0; i<n; i++) b[i] = nb[i];
    }
}
```

GetHeight : $height[i] = LCP(s[id[i]], s[id[i]-1]$

```
void GetHeight()
{
    int i, j, h; height[0] = 0;
    for(i=0; i<n; i++) rank[ id[i] ] = i;
    for( h=0, i=0; i<n; i++ ) if( rank[i] > 0 )
    {
        j = id[ rank[i] -1 ];
        while( s[i+h] == s[j+h] ) ++h;
        height[ rank[i] ] = h;
        if( h>0 ) --h;
    }
}
```

Chapter 2

Graph Theory and Network Algorithms

2.1 SSSP — Dijkstra + Binary Heap

```
const int inf = 1000000000;

int n,m,num,len,next[maxm],ev[maxm],ew[maxm];
int value[maxn],mk[maxn],nbs[maxn],ps[maxn],heap[maxn];

void update(int r)
{
    int q=ps[r],p=q>>1;
    while(p && value[heap[p]]>value[r]) {
        ps[heap[p]]=q; heap[q]=heap[p];
        q=p; p=q>>1;
    }
    heap[q]=r; ps[r]=q;
}

int getmin()
{
    int ret=heap[1],p=1,q=2,r=heap[len--];
    while(q<=len) {
        if( q<len && value[heap[q+1]]<value[heap[q]]) q++;
        if( value[heap[q]]<value[r]) {
            ps[heap[q]]=p; heap[p]=heap[q];
            p=q; q=p<<1;
        } else break;
    }
    heap[p]=r; ps[r]=p;
    return ret;
}

void dijkstra(int src,int dst)
{
    int i,j,u,v;
    for(i=1;i<=n;i++) {value[i]=inf; mk[i]=ps[i]=0; };
    value[src]=0; heap[len=1]=src; ps[src]=1;
    while(!mk[dst]) {
        if(len==0) return;
        u=getmin(); mk[u]=1;
        for(j=nbs[u];j;j=next[j]) {
            v=ev[j]; if( !mk[v] && value[u]+ew[j]<value[v] ) {
                if(ps[v]==0){ heap[++len]=v; ps[v]=len; }
                value[v]=value[u]+ew[j]; update(v);
            }
        }
    }
}
```

```
void readdata()
{
    int i,u,v,w;
    cin>>n>>m; num=0;
    for(i=1;i<=n;i++) nbs[i]=0;
    while(m--){
        cin>>u>>v>>w;
        next[++num]=nbs[u]; nbs[u]=num;
        ev[num]=v;  ew[num]=w;
    }
    dijkstra(1,n); // Minimum Distance saved at value[1..n]
}
```

2.2 SSSP — Bellman Ford + Queue

```
const int maxn = maxm = 1000005
const int inf = 1000000000

int nbs[maxn],next[maxm],value[maxn],open[maxn],open1[maxn];
int ev[maxm],ew[maxm],mk[maxn], n,m,num,cur,tail;

void BellmanFord(int src)
{
    int i,j,k,l,t,u,v,p=0;
    for(i=1;i<=n;i++) { value[i]=inf; mk[i]=0; }
    value[src]=tail=0;  open[0]=src;
    while(++p, tail>=0){
        for(i=0;i<=tail;i++) open1[i]=open[i];
        for(cur=0,t=tail,tail=-1; cur<=t; cur++)
            for(u=open1[cur],i=nbs[u]; i; i=next[i]) {
                v=ev[i]; if( value[u]+ew[i]<value[v]){
                    value[v]=value[u]+ew[i];
                    if(mk[v]!=p) { open[++tail]=v; mk[v]=p; }
                }
            }
    }
}
```

2.3 MST — Kruskal

```
#define maxn 1000005
#define maxm 1000005

int id[maxm],eu[maxm],ev[maxm],ew[maxm],n,m,pnt[maxn];
int cmp(const int &i,const int &j){ return ew[i]<ew[j]; }
int find(int x){ if(x!=pnt[x]) pnt[x]=find(pnt[x]); return pnt[x]; }

int Kruskal()
{
    int ret=0,i,j,p;
    for(i=1;i<=n;i++) pnt[i]=i; // node[1..n]
    for(i=0;i<m;i++) id[i]=i;    // ew[0..m-1]
    std::sort(id,id+m,cmp);
    for(j=-1,i=1; i<n; i++){
        while( p=id[++j], find(eu[p])==find(ev[p]) );
        ret+=ew[p]; pnt[find(ev[p])]=find(eu[p]);
    }
    return ret;
}
```

2.4 Minimum Directed Spanning Tree

```
int n,g[maxn][maxn],used[maxn],pass[maxn],eg[maxn],more,queue[maxn];

void combine(int id, int& sum) {
    int tot = 0, from, i, j, k;
    for (;id!=0&&!pass[id]; id=eg[id]) {queue[tot++]=id; pass[id]=1;}
    for (from=0; from<tot&&queue[from]!=id; from++);
    if (from==tot) return;   more = 1;
    for(i=from; i<tot; i++) {
        sum+=g[eg[queue[i]]][queue[i]];
        if(i!=from){   used[queue[i]]=1;
            for (j = 1; j <= n; j++) if (!used[j])
                if(g[queue[i]][j]<g[id][j]) g[id][j]=g[queue[i]][j];
        }
    }
    for (i=1; i<=n; i++) if (!used[i]&&i!=id) {
        for (j=from; j<tot; j++){  k=queue[j];
        if(g[i][id]>g[i][k]-g[eg[k]][k]) g[i][id]=g[i][k]-g[eg[k]][k];
        }
    }
}

int msdt(int root) { // return the total length of MDST
    int i, j, k, sum = 0;
    memset(used, 0, sizeof(used));
    for(more=1; more;){ more = 0;
        memset(eg, 0, sizeof(eg));
        for (i = 1; i <= n; i++) if (!used[i] && i != root) {
            for (j = 1, k = 0; j <= n; j++) if (!used[j] && i != j)
                if (k == 0 || g[j][i] < g[k][i]) k = j;
            eg[i] = k;
        } memset(pass, 0, sizeof(pass));
        for(i=1;i<=n;i++) if(!used[i]&&!pass[i]&&i!=root) combine(i,sum);
    }
    for(i=1; i<=n; i++) if (!used[i] && i!=root) sum+=g[eg[i]][i];
    return sum;
}
```

2.5 Maximum Matching on Bipartite Graph

```
int nx,ny,m, g[MAXN][MAXN],sy[MAXN],cx[MAXN],cy[MAXN];

int path(int u)
{
    for(int v=1;v<=ny;v++) if(g[u][v] && !sy[v]){ sy[v]=1;
        if(!cy[v] || path(cy[v])) { cx[u]=v; cy[v]=u; return 1;}
    } return 0;
}

int MaximumMatch()
{
    int i,ret=0;
    memset(cx,0,sizeof(cx)); memset(cy,0,sizeof(cy));
    for(i=1;i<=nx;i++)if(!cx[i]){memset(sy,0,sizeof(sy));ret+=path(i);}
    return ret;
}
```

2.6 Maximum Cost Perfect Matching on Bipartite Graph

```
int cx[maxn],cy[maxn],sx[maxn],sy[maxn],lx[maxn],ly[maxn];
int nx,ny,match,g[maxn][maxn];

int path(int u)
{
    sx[u]=1; for(int v=1;v<=ny;v++) if(g[u][v]==lx[u]+ly[v] && !sy[v]) {
    sy[v]=1; if(!cy[v] || path(cy[v])) { cx[u]=v; cy[v]=u; return 1;}
    } return 0;
}

void KuhnMunkres()
{
    int i,j,u,min;
    memset(lx,0,sizeof(lx));    memset(ly,0,sizeof(ly));
    memset(cx,0,sizeof(cx));    memset(cy,0,sizeof(cy));
    for(i=1;i<=nx;i++) for(j=1;j<=ny;j++) if(lx[i]<g[i][j]) lx[i]=g[i][j];
    for(match=0, u=1; u<=nx; u++) if(!cx[u]) {
        memset(sx,0,sizeof(sx));   memset(sy,0,sizeof(sy));
        while(!path(u)){
            min=0x3fffffff;
            for(i=1;i<=nx;i++) if(sx[i]) for(j=1;j<=ny;j++) if(!sy[j])
                if( lx[i]+ly[j]-g[i][j]<min ) min=lx[i]+ly[j]-g[i][j];
            for(i=1;i<=nx;i++) if(sx[i]) { lx[i]-=min; sx[i]=0; }
            for(j=1;j<=ny;j++) if(sy[j]) { ly[j]+=min; sy[j]=0; }
        };
    }
}
```

2.7 Maximum Matching on General Graph

```
// total is the maximum cardinality, p[1..n] means a match: i <-> p[i]
int g[maxn][maxn],p[maxn],l[maxn][3],n,total,status[maxn],visited[maxn];

void solve()
{
    int i,j,k,pass;
    memset(p,0,sizeof(p));
    do{ i=0;
        do{ if(p[++i]) pass=0; else {
                memset(l,0,sizeof(l));
                l[i][2]=0xff; pass=path(i);
                for(j=1;j<=n;j++) for(k=1;k<=n;k++)
                    if(g[j][k]<0) g[j][k]=-g[j][k];
            };
        }while( i!=n && !pass);
        if(pass) total+=2;
    }while(i!=n && total!=n);
}

void upgrade(int r)
{
    int j=r, i=l[r][1];
    for(p[i]=j; l[i][2]<0xff;){
        p[j]=i; j=l[i][2]; i=l[j][1]; p[i]=j;
    }   p[j]=i;
}
```

```
int path(int r)
{
    int i,j,k,v,t,quit;
    memset(status,0,sizeof(status)); status[r]=2;
    do{ quit=1;
        for(i=1;i<=n;i++) if(status[i]>1)
            for(j=1;j<=n;j++) if(g[i][j]>0 && p[j]!=i)
                if(status[j]==0) {
                    if(p[j]==0){ l[j][1]=i; upgrade(j); return 1;} else
                    if(p[j]>0) {
                        g[i][j]=g[j][i]=-1; status[j]=1;
                        l[j][1]=i;  g[j][p[j]]=g[p[j]][j]=-1;
                        l[p[j]][2]=j;   status[p[j]]=2;
                        quit=0;
                    }
                } else
                if(status[j]>1 && (status[i]+status[j]<6)){
                    quit=0; g[i][j]=g[j][i]=-1;
                    memset(visited,0,sizeof(visited));
                    visited[i]=1;   k=i; v=2;
                    while(l[k][v]!=0xff){k=l[k][v]; v=3-v; visited[k]=1;}
                    k=j; v=2;
                    while(!visited[k]) { k=l[k][v]; v=3-v; }
                    if(status[i]!=3) l[i][1]=j;
                    if(status[j]!=3) l[j][1]=i;
                    status[i]=status[j]=3;  t=i; v=2;
                    while(t!=k) {
                        if(status[l[t][v]]!=3) l[l[t][v]][v]=t;
                        t=l[t][v]; status[t]=3; v=3-v;
                    }
                    t=j; v=2;
                    while(t!=k) {
                        if(status[l[t][v]]!=3) l[l[t][v]][v]=t;
                        t=l[t][v]; status[t]=3; v=3-v;
                    }
                }
    }while(!quit);
    return 0;
}
```

2.8 Maximum Flow — Ford Fulkson in Matrix

```
// Remember to memset C[maxn][maxn] for a new case
int c[maxn][maxn],f[maxn][maxn],pnt[maxn],open[maxn],d[maxn],mk[maxn];

int maxflow(int n,int s,int t)
{
    int cur,tail,i,j,u,v,flow=0; memset(f,0,sizeof(f));
    do{ memset(mk,0,sizeof(mk));   memset(d,0,sizeof(d));
        open[0]=s; mk[s]=1; d[s]=0x3fffffff;
        for(pnt[s]=cur=tail=0; cur<=tail && !mk[t]; cur++)
            for(u=open[cur],v=1;v<=n;v++) if(!mk[v]&&f[u][v]<c[u][v]){
                mk[v]=1; open[++tail]=v; pnt[v]=u;
                if(d[u]<c[u][v]-f[u][v] ) d[v]=d[u];
                    else d[v]=c[u][v]-f[u][v];
            }
        if(!mk[t]) break; flow+=d[t];
        for(u=t;u!=s;){ v=u; u=pnt[v]; f[u][v]+=d[t]; f[v][u]=-f[u][v];}
    } while(d[t]>0); return flow;
}
```

2.9 Maximum Flow — Ford Fulkson in Link

```
#define maxn 1000
#define maxm 2*maxn*maxn

int c[maxm],f[maxm],ev[maxm],be[maxm],next[maxm],num=0;
int nbs[maxn],pnt[maxn],open[maxn],d[maxn],mk[maxn];

void AddEdge(int u,int v,int cc) // Remember to set nbs[1..n]=num=0
{
    next[++num]=nbs[u];     nbs[u]=num;  be[num]=num+1;
    ev[num]=v;  c[num]=cc;  f[num]=0;
    next[++num]=nbs[v];     nbs[v]=num;  be[num]=num-1;
    ev[num]=u;  c[num]=0 ;  f[num]=0;
}

int maxflow(int n,int s,int t)
{
    int cur,tail,i,j,u,v,flow=0; // f has been set zero when AddEdge
    do{ memset(mk,0,sizeof(mk));   memset(d,0,sizeof(d));
        open[0]=s; mk[s]=1; d[s]=0x3fffffff;
        for(pnt[s]=cur=tail=0; cur<=tail && !mk[t]; cur++)
            for(u=open[cur],j=nbs[u];j;j=next[j]) { v=ev[j];
                if(!mk[v]&&f[j]<c[j]){
                    mk[v]=1; open[++tail]=v; pnt[v]=j;
                    if(d[u]<c[j]-f[j]) d[v]=d[u]; else d[v]=c[j]-f[j];
                }
            }
        if(!mk[t]) break; flow+=d[t];
        for(u=t;u!=s;u=ev[be[j]]){j=pnt[u];f[j]+=d[t];f[be[j]]=-f[j];}
    } while(d[t]>0); return flow;
}
```

2.10 Minimum Cost Maximum Flow in Matrix

```
const int inf=0x3fffffff;

int c[maxn][maxn], f[maxn][maxn], w[maxn][maxn], pnt[maxn];
int value[maxn], d[maxn], mk[maxn], open[maxn], oldque[maxn];

void mincost(int n, int s, int t, int &flow, int &cost)
{
    int cur, tail, tl, i, j, u, v;
    memset(f, 0, sizeof(f)); flow=0; cost=0;
    do{ memset(d, 0, sizeof(d));
        for(i=1;i<=n;i++) value[i]=inf;
        open[0]=s; d[s]=0x3fffffff; tail=value[s]=0;
        while(tail>=0){
            memset(mk, 0, sizeof(mk));
            memcpy(oldque, open, sizeof(open));
            for(tl=tail, pnt[s]=cur=0, tail=-1; cur<=tl; cur++)
            for(u=oldque[cur], v=1;v<=n;v++)
                if( f[u][v]<c[u][v] && value[u]<inf
                    && value[u]+w[u][v]<value[v] ){
                    if(!mk[v]){ mk[v]=1; open[++tail]=v;};
                    pnt[v]=u; value[v]=value[u]+w[u][v];
                    if(d[u]<c[u][v]-f[u][v]) d[v]=d[u];
                        else d[v]=c[u][v]-f[u][v];
                }
        }
        if(value[t]==inf) return;
        flow+=d[t]; cost+=d[t]*value[t];
        for(u=t;u!=s;){
            v=u; u=pnt[v]; f[u][v]+=d[t]; f[v][u]=-f[u][v];
            if( f[u][v]<0 ) w[u][v]=-w[v][u]; else
            if( f[v][u]<0 ) w[v][u]=-w[u][v];
        }
    } while(d[t]>0);
}
```

2.11 Minimum Cost Maximum Flow in Link

```
#define maxn 350
#define maxm 100000 // maxm*2
const int inf=0x3fffffff;

int c[maxm],f[maxm],w[maxm],ev[maxm],be[maxm],next[maxm],value[maxn];
int nbs[maxn],pnt[maxn],open[maxn],oldque[maxn],d[maxn],mk[maxn],num=0;

void AddEdge(int u,int v,int cc,int ww) // Remember to set nbs[1..n]=num=0
{
    next[++num]=nbs[u];     nbs[u]=num;  be[num]=num+1;
    ev[num]=v;  c[num]=cc;  f[num]=0;  w[num]=ww;
    next[++num]=nbs[v];     nbs[v]=num;  be[num]=num-1;
    ev[num]=u;  c[num]=0 ;  f[num]=0;  w[num]=-ww;
}

void mincost(int n,int s,int t,int &flow,int &cost)
{
    int cur,tail,tl,i,j,u,v;
    memset(f,0,sizeof(f));  flow=0;  cost=0;
    do{ memset(d,0,sizeof(d));
        for(i=1;i<=n;i++) value[i]=inf;
        open[0]=s;  d[s]=0x3fffffff;  tail=value[s]=0;
        while(tail>=0){
            memset(mk,0,sizeof(mk));
            memcpy(oldque,open,sizeof(open));
            for(tl=tail,pnt[s]=cur=0,tail=-1; cur<=tl;  cur++)
            for(u=oldque[cur],j=nbs[u];j;j=next[j]){  v=ev[j];
                if(f[j]<c[j] && value[u]<inf && value[u]+w[j]<value[v]){
                    if(!mk[v]){  mk[v]=1;  open[++tail]=v;};
                    pnt[v]=j;  value[v]=value[u]+w[j];
                    if(d[u]<c[j]-f[j])  d[v]=d[u];  else  d[v]=c[j]-f[j];
                }
            }
        }
        if(value[t]==inf)  return;
        flow+=d[t];  cost+=d[t]*value[t];
        for(u=t;u!=s;u=ev[be[j]]){j=pnt[u];f[j]+=d[t];f[be[j]]=-f[j];}
    } while(d[t]>0);
}
```

2.12 Recognizing Chordal Graph

```
int n,m,mk[maxn],degree[maxn],PEO[maxn],g[maxn][maxn];

int Chordal()
{
    memset(mk,0,sizeof(mk));  memset(degree,0,sizeof(degree));
    for(int j,k,u,v,i=0;i<n;i++) {
        j=-1;    u=-1;
        for(k=0;k<n;k++) if(!mk[k] && (j<0 || degree[k]>degree[j]))  j=k;
        mk[j]=1;  PEO[i]=j;
        for(k=i-1;k>=0;k--) if( g[j][PEO[k]] )
            if( u<0 ) u=PEO[k];  else if( !g[u][PEO[k]])  return 0;
        for(k=0;k<n;k++) if(!mk[k] && g[j][k])  degree[k]++;
    }
    return 1;
}
```

2.13 DFS — Bridge

```
int n,g[maxn][maxn],mk[maxn],d[maxn],low[maxn];
int color,ti,bridgenum,bridgeu[maxn],bridgev[maxn];

void dfsvisit(int u,int p)
{
    int v,s=0,bBridge=0; low[u]=d[u]=++ti; mk[u]=-color;
    for(v=1; v<=n; v++) if(g[u][v] && v!=p)
        if(mk[v]==0){ dfsvisit(v,u); s++;
            if(low[v]<low[u]) low[u]=low[v];
            if(low[v]==d[v]) {
                bridgeu[bridgenum   ]=u;
                bridgev[bridgenum++]=v;
            }
        } else if(d[v]<low[u]) low[u]=d[v];
    mk[u]=color;
}

void dfs()
{
    int i,j,k; memset(mk,0,sizeof(mk));
    color=ti=bridgenum=0;
    for(i=1; i<=n; i++) if(!mk[i]){ ++color; dfsvisit(i,0); }
    cout<<bridgenum<<endl;
}
```

2.14 DFS — Cutvertex

```
int n,g[maxn][maxn],mk[maxn],d[maxn],low[maxn];
int color,ti,cutvertexnum,cutvertexlist[maxn];

void dfsvisit(int u,int p)
{
    int v,s=0,bVertex=0; low[u]=d[u]=++ti; mk[u]=-color;
    for(v=1; v<=n; v++) if(g[u][v] && v!=p)
        if(mk[v]==0){ dfsvisit(v,u); s++;
            if(low[v]<low[u]) low[u]=low[v];
            if(low[v]>=d[u]) bVertex=1;
        } else if(d[v]<low[u]) low[u]=d[v];
    if((p && bVertex) || (!p && s>1)) cutvertexlist[cutvertexnum++]=u;
    mk[u]=color;
}

void dfs()
{
    int i,j,k; memset(mk,0,sizeof(mk));
    color=ti=cutvertexnum=0;
    for(i=1; i<=n; i++) if(!mk[i]){ ++color; dfsvisit(i,0); }
    cout<<cutvertexnum<<endl;
    for(i=0;i<cutvertexnum;i++) cout<<cutvertexlist[i]<<" "; cout<<endl;
}
```

2.15 DFS — Block

```
int n,g[maxn][maxn],mk[maxn],d[maxn],low[maxn],len,que[maxn];
int color,ti,cutvertexnum,cutvertexlist[maxn],blocknum;

void dvsvisit(int u,int p)
{
    int v,s=0,bCutvertex=0; low[u]=d[u]=++ti; mk[u]=-color; que[++len]=u;
    for(v=1; v<=n; v++) if(g[u][v] && v!=p)
        if(mk[v]==0){ dvsvisit(v,u); s++;
            if(low[v]<low[u]) low[u]=low[v];
            if(low[v]>=d[u]){
                while(que[len]!=v) cout<<que[len--]<<" ";
                cout<<que[len--]<<" "<<u<<endl;
                bCutvertex=1; blocknum++;
            }
        } else if(d[v]<low[u]) low[u]=d[v];
    if((p && bCutvertex) || (!p && s>1)) cutvertexlist[cutvertexnum++]=u;
    mk[u]=color;
}

void dfs()
{
    int i,j,k; memset(mk,0,sizeof(mk));
    color=ti=cutvertexnum=blocknum=0;
    for(i=1; i<=n; i++) if(!mk[i]){
        ++color; len=0; dvsvisit(i,0);
        if(len>1 || d[i]==ti){
            while(len>1) cout<<que[len--]<<" ";
            cout<<i<<endl; blocknum++;
        }
    }
    cout<<"Block Number : "<<blocknum<<endl;
    cout<<"Cutvertex Number: "<<cutvertexnum<<endl;
    for(i=0;i<cutvertexnum;i++) cout<<cutvertexlist[i]<<" ";
    cout<<endl<<endl;
}
```

2.16 DFS — Topological Sort

```
int n,mk[maxn],topo[maxn],g[maxn][maxn],ps,topook;

void dfs(int u)
{
    if(mk[u]<0){topook=0; return;}; if(mk[u]>0) return; else mk[u]=-1;
    for(int v=1; topook && v<=n; v++) if(g[u][v]) dfs(v);
    topo[ps--]=u; mk[u]=1;
}

void toposort()
{
    int i,j,k; topook=1; ps=n; memset(mk,0,sizeof(mk));
    for(i=1;topook && i<=n;i++) if(!mk[i]) dfs(i);
}

int main()
{
    int i,m,u,v;
    while(cin>>n>>m,n && !cin.fail()){
        memset(g,0,sizeof(g));
        while(m--){ cin>>u>>v; g[u][v]=1; }; toposort();
        for(i=1;i<n;i++) cout<<topo[i]<<" "; cout<<topo[n]<<endl;
    }
    return 0;
}
```

2.17 Strongly Connected Component

```
int g[maxn][maxn],n, mk[maxn], list[maxn],num;

void back(int v)
{
    mk[v]=1; cout<<v<<" ";
    for(int u=1; u<=n; u++) if(!mk[u] && g[u][v]) back(u);
}

void dfs(int u)
{
    mk[u]=1;
    for(int v=1; v<=n; v++) if(!mk[v] && g[u][v]) dfs(v);
    list[num--]=u;
}

int main()
{
    int i,j,k,l;
    cin>>n; for(i=1;i<=n;i++) for(j=1;j<=n;j++) cin>>g[i][j];
    memset(mk,0,sizeof(mk)); num=n;
    for(i=1;i<=n;i++) if(!mk[i]) dfs(i);
    memset(mk,0,sizeof(mk));
    for(i=1;i<=n;i++) if(!mk[list[i]]) { back(list[i]); cout<<endl; }
    return 0;
}
```

Chapter 3

Number Theory

3.1 Greatest Common Divisor

```
void gcd(int a,int b,int &d,int &x,int &y)
{
    if( b==0 ){ d=a; x=1; y=0; return; }
    gcd( b, a%b, d, y, x );
    y -= x * (a/b);
}
```

3.2 Chinese Remainder Theorem

$extended\_euclid(a, b) = ax + by$

```
int extended_euclid(int a,int b,int &x,int &y)
{
    if (b==0){ x=1,y=0; return a; } else {
        int res=extended_euclid(b,a%b,x,y);
        int t=x; x=y; y=t-(a/b)*y;
        return res;
    }
}
```

$ax \equiv b \pmod{n}$, $n > 0$

```
void modular_linear_equation_solver(int a,int b,int n)
{
    int d,x,y,e,i;
    d=extended_euclid(a,n,x,y);
    if (b%d!=0) cout<<"No answer!"; else {
        e=x*(b/d)%n;      // x=e is a basic solution
        for(i=0;i<d;i++) cout<<(e+i*(n/d))%n<<endl;
    }
}
```

Given b_i , w_i , $i = 0 \cdots len - 1$ which $w_i > 0$, $i = 0 \cdots len - 1$ and $(w_i, w_j) = 1$, $i \neq j$
Find an x which satisfies: $x \equiv b_i \pmod{w_i}$, $i = 0 \cdots len - 1$

```
int china(int b[],int w[],int len)
{
    int i,d,x,y,x,m,n;
    x=0; n=1; for(i=0;i<len;i++) n*=w[i];
    for(i=0;i<len;i++){
        m=n/w[i];
        d=extended_euclid(w[i],m,x,y);
        x=(x+y*m*b[i])%n;
    }
    return (n+x%n)%n;
}
```

3.3 Prime Generator

```
#define maxn 10000000
#define maxp 1000000

char mk[maxn];
int prime[maxp], pnum;

void GenPrime(int n)
{
    int i,j,k; pnum = 0; memset(mk, 0, n+1);
    for(i=2,k=4; i<=n; i++,k+=i+i-1) if( !mk[i] )
    {
        prime[pnum++] = i;
        if(k<=n) for(j=i+i; j<=n; j+=i) mk[j] = 1;
    }
}
```

3.4 ϕ Generator

$\phi(n) = n \prod_{p|n}(1 - \frac{1}{p})$, where p is a prime.
$\phi(846720) = 193536$

```
int Phi(int n) // O( Sqrt(N) )
{
    int i,j,ret=n;
    for(i=2,j=4; j<=n; i++,j+=i+i-1) if (!(n%i))
    {
        ret = ret / i * (i-1);
        while( !(n%i) ) n/=i;
    }
    if( n>1 ) ret = ret / n * (n-1);
    return ret;
}

#define maxn 10000000
#define maxp 1000000

int phi[maxn], prime[maxp], pnum;

void GenPhi(int n)  // O( N loglog N )
{
    int i,j,k; pnum = 0;
    memset(phi, 0, (n+1)*sizeof(phi[0]));
    phi[1] = 1;
    for(i=2; i<=n; i++) if( !phi[i] )
    {
        prime[pnum++] = i;
        for(j=i; j<=n; j+=i)
        {
            if(!phi[j]) phi[j]=j;
            phi[j] = phi[j]/i*(i-1);
        }
    }
}
```

3.5 Discrete Logarithm

```
#define llong __int64
inline int mod(int x,int n) {return (x%n+n)%n;}
```

// $ax \equiv 1 \pmod n$

```
int Inv(int a,int n)
{
    int d,x,y; Gcd(a,n,d,x,y);
    if (d==1) return mod(x,n); else return -1;
}
```

// $x \equiv a^b \pmod n$, $a, b >= 0$

```
int ModPow(int a,int b,int n)
{
    llong d(1),i(0); while (b>=((llong)1<<i)) i++;
    for(--i;i>=0;--i){ d=d*d%n; if(b&(1<<i)) d=d*a%n; }
    return d;
}
```

// $a^x \equiv b \pmod n$, n is prime!

```
int mexp[50000],id[50000];

bool logcmp(const int &a,const int &b) {return mexp[a]<mexp[b];}

int ModLog(int a,int b,int n)
{
    int i,j,m = (int)ceil(sqrt(n)), inv = Inv(ModPow(a,m,n),n);
    for (id[0]=0,mexp[0]=i=1;i<m;i++)
        { id[i]=i;mexp[i] = (mexp[i-1]*(llong)a)%n; }
    std::stable_sort(id,id+m,logcmp);
    std::sort(mexp,mexp+m);
    for (i=0; i<m; i++) {     // i*m < n
        j = std::lower_bound(mexp,mexp+m,b)-mexp;
        if (j<m && mexp[j]==b) return i*m+id[j];
        b = (b*(llong)inv)%n;
    }
    return -1;
}
```

3.6 Square Roots in Z_p

```
#define llong __int64

int ModPow(int a,int b,int n) // a^b mod n a,b>=0
{
    llong d(1),i(0);
    while (b>=((llong)1<<i)) i++;
    for (--i;i>=0;--i){ d=d*d%n; if(b&(1<<i)) d=d*a%n;}
    return d;
}

// x*x = a (mod n) n should be a prime and gcd(a,n)==1
int ModSqrt(int a,int n)
{
    int b,k,i,x;
    if (n==2) return a%n;
    if (ModPow(a,(n-1)/2,n)==1) {
        if (n%4==3) x = ModPow(a,(n+1)/4,n); else {
            for(b=1; ModPow(b,(n-1)/2,n)==1; b++);
            i=(n-1)/2; k=0; do{ i/=2; k/=2;
                if((ModPow(a,i,n)*(llong)ModPow(b,k,n)+1)%n==0) k+=(n-1)/2;
            } while (i%2==0);
            x=( ModPow(a,(i+1)/2,n)*(llong)ModPow(b,k/2,n) ) % n;
        }   if(x*2>n) x=n-x; return x;
    } return -1;
}

int main()
{
    int a,n,casec,x; cin >> casec;
    while (casec--) {
        cin >> a >> n;  x = ModSqrt(a,n);
        if (x<0) cout << "No root" << endl;
        else if (x*2==n) cout << x << endl;
        else cout << x << ' ' << n-x << endl;
    }
    return 0;
}
```

Chapter 4

Algebraic Algorithms

4.1 Linear Equations in Z_2

// Gauss Elimination : $\bigoplus_{0\leq j<nn} a_{i,j}x_{i,j} = a_{i,nn}$

```
int m,nn,num,list[maxn]; char a[maxn][maxn];

int reduce()
{
    int i,j,k,r;
    for(i=r=0; i<nn; i++){
        for(j=r; j<m && !a[j][i]; j++); if(j>=m) continue;
        if(j>r) for(k=0;k<=nn;k++) std::swap(a[r][k],a[j][k]);
        for(num=0,k=i;k<=nn;k++) if( a[r][k] ) list[num++]=k;
        for(j=0;j<m;j++) if(j!=r && a[j][i])
            for(k=0;k<num;k++) a[j][list[k]]^=1;
        ++r;
    }
    for(i=0;i<m;i++)
        if(a[i][nn]){
            for(j=0;j<nn && !a[i][j];j++);
            if(j==nn) return 0; // else x[j]=a[i][nn]/a[i][j];
        }
    return 1;
}
```

4.2 Linear Equations in Z

// Gauss Elimination : $\sum_{0\le j<nn} a_{i,j}x_{i,j} = a_{i,nn}$

```
int m,nn, a[maxn][maxn];

int gcd(int x,int y)
{ if(y==0) return x; else return gcd(y,x%y); }

void yuefen(int b[],int ct)
{
    int i,j=0,k;
    for(i=0;i<ct;i++) if(b[i]) if(j) k=gcd(b[i],k); else {k=b[i]; j=1;}
    if(k!=0) for(i=0;i<ct;i++) b[i]/=k;
}

int reduce() // return 0 means no solution!
{
    int i,j,k,r,tmp;
    for(i=r=0; i<nn; i++){
        for(j=r; j<m && !a[j][i]; j++);  if(j>=m) continue;
        if(j>r) for(k=0;k<=nn;k++) std::swap(a[r][k],a[j][k]);
        for(j=0;j<m;j++) if(j!=r && a[j][i]){
            tmp=a[j][i];
            for(k=0;k<=nn;k++) a[j][k]=a[j][k]*a[r][i]-tmp*a[r][k];
            yuefen(a[j],nn+1);
        } ++r;
    }
    for(i=0;i<m;i++) if(a[i][nn]) {
        for(j=0;j<nn && !a[i][j];j++);
        if(j==nn) return 0; // else x[j]=a[i][nn]/a[i][j];
    }
    return 1;
}
```

4.3 Linear Equations in Q

Note: *fraction.h* contains a *Fraction Class* (Section 1.4 on Page 8)

```
#include<fraction.h>

int m,nn; Fraction a[maxn][maxn];
int dcmp(Fraction x){return x.a;}

int reduce()
{
    int i,j,k,r;  double tmp;
    for(i=r=0; i<nn; i++){
        for(j=r; j<m && !dcmp(a[j][i]); j++);  if(j>=m) continue;
        if(j>r) for(k=0;k<=nn;k++) std::swap(a[r][k],a[j][k]);
        for(j=0;j<m;j++) if(j!=r && dcmp(a[j][i])){
            tmp=a[j][i]/a[r][i];
            for(k=0;k<=nn;k++) a[j][k]=a[j][k]-tmp*a[r][k];
        } ++r;
    }
    for(i=0;i<m;i++) if(dcmp(a[i][nn])){
        for(j=0;j<nn && !dcmp(a[i][j]);j++);
        if(j==nn) return 0; // else x[j]=a[i][nn]/a[i][j];
    }   return 1;
}
```

4.4 Linear Equations in R

```
const double eps=1e-8;
int m,nn; double a[maxn][maxn];

int dcmp(double x){ if(x>eps) return 1; if(x<-eps) return -1; return 0;}

int reduce() // r is rank
{
    int i,j,k,r;  double tmp;
    for(i=r=0; i<nn; i++){
        for(j=r; j<m && !dcmp(a[j][i]); j++);  if(j>=m) continue;
        if(j>r) for(k=0;k<=nn;k++) std::swap(a[r][k],a[j][k]);
        for(j=0;j<m;j++) if(j!=r && dcmp(a[j][i])){
            tmp=a[j][i]/a[r][i];
            for(k=0;k<=nn;k++) a[j][k]-=tmp*a[r][k];
        } ++r;
    }
    for(i=0;i<m;i++) if(dcmp(a[i][nn])){
        for(j=0;j<nn && !dcmp(a[i][j]);j++);
        if(j==nn) return 0; // else x[j]=a[i][nn]/a[i][j];
    }   return 1;
}
```

4.5 Roots of Polynomial

Find the roots of $f_a(x) = \sum_{i=0}^{n} a_i x^i$ using *Newton Iterations*, $f_b(x) = f_a(x)\frac{d}{dx}$

```
const double eps=1e-5;
#define genx (rand()%1000)/100.0

int dcmp(double x)
{ if(x>eps) return 1; else if(x<-eps) return -1; else return 0;}

double f(double a[],int n,double x)
{
    double ret=0,xx=1;
    for(int i=0;i<=n;i++){ ret+=a[i]*xx; xx*=x; }
    return ret;
}

double newton(double a[],double b[],int n)
{
    double dy,y,x=genx,lastx=x-1;
    while(y=f(a,n,x), dcmp(lastx-x)){
        lastx=x; dy=f(b,n-1,x);
        if(!dcmp(dy)) x=genx; else x=x-y/dy;
    }
    return x;
}

void solve(double a[],double x[],int n)
{
    int i,j; double b[maxn];
    for(j=n;j>0;j--){
        for(i=0;i<j;i++) b[i]=a[i+1]*(i+1);
        x[j-1]=newton(a,b,j);
        for(b[j]=0, i=j-1; i>=0; i--) b[i]=a[i+1]+b[i+1]*x[j-1];
        for(i=0;i<j;i++) a[i]=b[i];
    }
}
```

4.6 Roots of Cubic and Quartic

$$c_0 + c_1 * x + c_2 * x^2 + c_3 * x^3 + c_4 * x^4 = 0$$

The functions return the number of distinct non-complex roots and put the values into the s array.

```
const double pi = acos(-1.0); // 3.14159265358979323846

double cbrt(double x)
{
    if( x> eps ) return  pow(  x, 1/3.0);
    if( x<-eps ) return -pow( -x, 1/3.0);
    return 0;
}

int SolveQuadric(double c[3], double s[2])
{
    double p, q, d;  // normal form: x^2 + px + q = 0
    p = c[1]/(2*c[2]); q = c[0]/c[2]; d = p*p-q;
    if( dcmp(d)==0 ) { s[0] = - p; return 1; }
    if( dcmp(d) < 0 ) return 0;
    d = sqrt( d );
    s[0] = - p + d;
    s[1] = - p - d;
    return 2;
}

int SolveCubic(double c[4], double s[3])
{
    int     i, num;   // normal form: x^3 + Ax^2 + Bx + C = 0
    double  sub, A, B, C, sqa, p, q, cbp, d;
    A = c[2]/c[3];  B = c[1]/c[3];  C = c[0]/c[3];
    sqa = A * A;       //  x = y - A/3  =>  x^3 +px + q = 0
    p = 1.0/3 * (- 1.0/3 * sqa + B);
    q = 1.0/2 * (2.0/27 * A * sqa - 1.0/3 * A * B + C);
    cbp = p * p * p;  // use Cardano's formula
    d = q * q + cbp;
    if( dcmp(d)==0 ) {
        if( dcmp(q)==0 ) { s[0] = 0; num = 1; } // one triple solution
        else { // one single and one double solution
            double u = cbrt( -q );
            s[0] = 2 * u;  s[1] = - u;  num = 2;
        }
    } else if( dcmp(d)<0 ) { // Casus irreducibilis: three real solutions
        double phi = 1.0/3 * acos(-q / sqrt(-cbp));
        double t = 2 * sqrt(-p);
        s[ 0 ] =   t * cos(phi);
        s[ 1 ] = - t * cos(phi + pi / 3);
        s[ 2 ] = - t * cos(phi - pi / 3);
        num = 3;
    } else { /* one real solution */
        d = sqrt(d); double u = cbrt(d-q), v = - cbrt(d+q);
        s[ 0 ] = u + v; num = 1;
    }
    /* resubstitute */
    sub = 1.0/3 * A;  for( i=0; i<num; ++i ) s[i] -= sub;
    return num;
}
```

```
int SolveQuartic(double c[5], double s[4])
{
    double  e[4], z, u, v, sub, A, B, C, d, sqa, p, q, r;
    int     i, num;            // x^4 + Ax^3 + Bx^2 + Cx + D = 0
    A = c[3]/c[4];  B = c[2]/c[4];  C = c[1]/c[4];  d = c[0]/c[4];
    sqa = A * A;               // x = y - A/4  =>  x^4 + px^2 + qx + r = 0
    p = - 3.0/8 * sqa + B;
    q = 1.0/8 * sqa * A - 1.0/2 * A * B + C;
    r = - 3.0/256*sqa*sqa + 1.0/16*sqa*B - 1.0/4*A*C + d;
    if( dcmp(r)==0 ) {         // no absolute term: y(y^3 + py + q) = 0
        e[0] = q;  e[1] = p;  e[2] = 0;  e[3] = 1;
        num = SolveCubic(e, s);  s[ num++ ] = 0;
    } else {                   // solve the resolvent cubic ...
        e[0] = 1.0/2 * r * p - 1.0/8 * q * q;   e[1] = - r;
        e[2] = - 1.0/2 * p;                     e[3] = 1;
        SolveCubic(e, s);
        z = s[ 0 ];            // ... and take the one real solution
        u = z*z-r;  v = 2*z-p; // ... to build two quadric equations
        if(dcmp(u)==0) u=0; else if(dcmp(u)>0) u=sqrt(u); else return 0;
        if(dcmp(v)==0) v=0; else if(dcmp(v)>0) v=sqrt(v); else return 0;
        e[0] = z-u;  e[1] = dcmp(q)<0 ? -v : v;  e[2] = 1;
        num = SolveQuadric(e, s);
        e[0] = z+u;  e[1] = dcmp(q)<0 ? v : -v;  e[2] = 1;
        num += SolveQuadric(e, s + num);
    }
    sub = 1.0/4*A;  for( i=0; i<num; ++i ) s[i] -= sub; // resubstitute
    return num;
}
```

4.7 Fast Fourier Transform

```
const double eps=1e-8;
const double pi=acos(-1.0);

#define cp complex<double>

inline int max(int a,int b){ if(a>b) return a; else return b; }
inline int dcmp(double a){ if(a<-eps) return -1; return (a>eps); }

void fft(cp *x,int n,cp *y,int bInv) // y=Wx,  w[j,k]=e^ijk
{
    if(n==1) { y[0] = x[0]; return; }
    cp *xeven = new cp[n/2], *xodd = new cp[n/2], w(1,0),
       *yeven = new cp[n/2], *yodd = new cp[n/2], wn; int i;
    if(bInv) wn=cp( cos(-2*pi/n), sin(-2*pi/n) );
        else wn=cp( cos( 2*pi/n), sin( 2*pi/n) );
    for(i=0; i<n/2; i++)
    {
        xeven[i] = x[i*2  ];
        xodd [i] = x[i*2+1];
    }
    fft(xeven, n/2, yeven, bInv);
    fft(xodd , n/2, yodd , bInv);
    for(i=0; i<n/2; i++)
    {
        y[i    ] = yeven[i] + w*yodd[i];
        y[i+n/2] = yeven[i] - w*yodd[i];
        w *= wn;
    }
    delete xeven; delete yeven; delete xodd; delete yodd;
}
```

4.8 FFT - Polynomial Multiplication

```
void PolyMulti(double *a,int na,double *b,int nb,double *c,int &nc)
{
    int i,j,n=(na>nb)?na:nb;
    n=1<<((int)ceil(log(2*n)/log(2)-eps));
    cp *x=new cp[n], *ya=new cp[n], *yb=new cp[n], *yc=new cp[n];
    for(i=0;i<n;i++) x[i]=(i<na)?a[i]:0;      fft(x,n,ya,0);
    for(i=0;i<n;i++) x[i]=(i<nb)?b[i]:0;      fft(x,n,yb,0);
    for(i=0;i<n;i++) yc[i]=ya[i]*yb[i];       fft(yc,n,x,1);
    for(i=0;i<n;i++) c[i]=x[i].real()/n;
    for(nc=n; nc>0 && dcmp(c[nc-1])==0; nc--);
    delete x; delete ya; delete yb; delete yc;
}
```

4.9 FFT - Convolution

$r_k = \sum_{i=0}^{n-1} a[i] * b[i-k]$

```
void Convolution1(int *a,int *b,int *c,int n)
{
    int m,i,j,*rb=new int[n]; rb[0]=b[0];
    for(i=1;i<n;i++) rb[i]=b[n-i];
    PolyMulti1(a,n,rb,n,c,m);
    for(i=0;i<n;i++) c[i]+=c[i+n];
    delete[] rb;
}

\\ N must be power of 2
void Convolution2(int *a,int *b,int *c,int n)
{
    int i,j;
    cp *x=new cp[n], *ya=new cp[n], *yb=new cp[n], *yc=new cp[n];
    x[0]=b[0];
    for(i=1;i<n;i++) x[i]=(i<n)?b[n-i]:0;     fft(x,n,yb,0);
    for(i=0;i<n;i++) x[i]=(i<n)?a[i  ]:0;     fft(x,n,ya,0);
    for(i=0;i<n;i++) yc[i]=ya[i]*yb[i];       fft(yc,n,x,1);
    for(i=0;i<n;i++) c[i]=int(x[i].real()/n+0.5);
    delete x; delete ya; delete yb; delete yc;
}
```

4.10 FFT - Reverse Bits

```
#define for if(0); else for
const double pi = acos(-1.0);
const int MFB = 16;
int **bt = 0;

struct cp { double re,im;} ;

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

```
void InitFFT() {
    bt = new int *[MFB]; int i,j,length;
    for(i=1, length=2; i<=MFB; ++i, length<<=1) {
        bt[i-1] = new int[length];
        for(j=0; j<length; ++j) bt[i-1][j] = ReverseBits(j, i);
    }
}

inline int FRB(int i, int bitnum) {
    return bitnum <= MFB ? bt[bitnum - 1][i] : ReverseBits(i, bitnum);
}

void FFT(cp *in, cp *out,int n,bool bInv)
{
    int i, j, k, ed, len, bitnum=0;  if(!bt) InitFFT();
    while( !((1<<bitnum)&n) ) bitnum++;
    for(i=0; i<n; ++i) out[FRB(i, bitnum)] = in[i];
    double basicangle = pi * (bInv ? -2 : 2);
    cp a0,a1,a2,a,b;
    for(ed = 1, len = 2; len <= n; len <<= 1) {
        double delta_angle = basicangle / len;
        double sin1 = sin(-delta_angle), sin2 = sin(-delta_angle * 2);
        double cos1 = cos(-delta_angle), cos2 = cos(-delta_angle * 2);
        for(i=0; i<n; i+=len) {
            a1.re=cos1; a1.im=sin1; a2.re=cos2; a2.im=sin2;
            for(j=i, k=0; k<ed; ++j, ++k) {
                a0.re=2*cos1*a1.re-a2.re;  a0.im=2*cos1*a1.im-a2.im;
                a2 = a1; a1 = a0; b=out[j+ed];
                a.re = a0.re*b.re - a0.im*b.im;
                a.im = a0.im*b.re + a0.re*b.im;
                out[j+ed].re=out[j].re-a.re;
                out[j+ed].im=out[j].im-a.im;
                out[j].re+=a.re;
                out[j].im+=a.im;
            }
        }
        ed = len;
    }
    if (bInv) for (int i = 0; i < n; ++i) { out[i].re/= n; out[i].im/=n; }
}

// n must be power of 2
void convolution(double *a, double *b,double *r,int n) {
    int i;
    cp *s=new cp[n], *d1=new cp[n], *d2=new cp[n], *y=new cp[n];
    s[0].im=b[0]; s[0].re=0;
    for(i=1; i<n; ++i) s[i].re=b[n-i],s[i].im=0;    FFT(s, d2, n, false);
    for(i=0; i<n; ++i) s[i].re=a[i  ],s[i].im=0;    FFT(s, d1, n, false);
    for(i=0; i<n; ++i) {
        y[i].re = d1[i].re*d2[i].re - d1[i].im*d2[i].im;
        y[i].im = d1[i].re*d2[i].im + d1[i].im*d2[i].re;
    }
    FFT(y, s , n, true);
    for(i=0; i<n; ++i) r[i] = s[i].re;
    delete s; delete d1; delete d2; delete y;
}
```

4.11 Linear Programming - Primal Simplex

Primal Simplex Method for solving Linear Programming problem in Standard Form

maximize

$$c_1x_1 + c_2x_2 + \cdots + c_nx_n + \text{ans}$$

subject to

$$\begin{aligned} a_{1,1}\,x_1 + a_{1,2}\,x_2 + \cdots + a_{1,n}\,x_n &\leq rhs_1 \\ a_{2,1}\,x_1 + a_{2,2}\,x_2 + \cdots + a_{2,n}\,x_n &\leq rhs_2 \\ &\vdots \\ a_{m,1}\,x_1 + a_{m,2}\,x_2 + \cdots + a_{m,n}\,x_n &\leq rhs_m \end{aligned}$$

```
const double eps = 1e-8;
const double inf = 1e15;

#define OPTIMAL -1
#define UNBOUNDED -2
#define FEASIBLE -3
#define INFEASIBLE -4
#define PIVOT_OK 1

int basic[maxn],row[maxm],col[maxn];
double c0[maxn];

double dcmp(double x)
{
    if( x >  eps ) return 1;
    if( x < -eps ) return -1;
    return 0;
}

int Pivot(int n, int m, double *c,double a[maxn][maxn],
    double *rhs,int &i,int &j)
{
    double min = inf; int k = -1;
    for(j=0; j<=n; j++) if( !basic[j] && dcmp(c[j])>0 )
        if( k<0 || dcmp(c[j]-c[k])>0 ) k=j;
    j=k; if( k < 0 ) return OPTIMAL;
    for(k=-1,i=1; i<=m; i++) if( dcmp(a[i][j])>0 )
        if( dcmp(rhs[i]/a[i][j]-min) < 0 ) { min = rhs[i]/a[i][j]; k=i; }
    i=k; if( k < 0 ) return UNBOUNDED; else return PIVOT_OK;
}

int PhaseII(int n, int m, double *c, double a[maxn][maxn],
    double *rhs, double &ans, int PivotIndex)
{
    int i,j,k,l; double tmp;
    while(k=Pivot(n,m,c,a,rhs,i,j),k==PIVOT_OK || PivotIndex)
    {
        if( PivotIndex ) { j=0; i=PivotIndex; PivotIndex=0; }
        basic[row[i]]=0; col[row[i]]=0; basic[j]=1; col[j]=i; row[i]=j;
        tmp=a[i][j]; for(k=0;k<=n;k++) a[i][k]/=tmp; rhs[i]/=tmp;
        for(k=1;k<=m;k++) if(k!=i && dcmp(a[k][j]))
        {
            tmp = -a[k][j]; for(l=0;l<=n;l++) a[k][l]+=tmp*a[i][l];
            rhs[k] += tmp*rhs[i];
        }
        tmp=-c[j]; for(l=0;l<=n;l++) c[l]+=a[i][l]*tmp; ans-=tmp*rhs[i];
    }
    return k;
}
```

```
int PhaseI(int n,int m,double *c,double a[maxn][maxn],double *rhs,double &ans)
{
    int i,j,k = -1; double tmp, min = 0, ans0 = 0;
    for(i=1; i<=m; i++) if( dcmp(rhs[i]-min)<0 ) { min=rhs[i]; k=i; }
    if( k<0 ) return FEASIBLE;
    for(i=1; i<=m; i++) a[i][0] = -1;
    for(j=1; j<=n; j++) c0[j]=0;  c0[0] = -1;
    PhaseII(n, m, c0, a, rhs, ans0, k);
    if( dcmp(ans0)<0 ) return INFEASIBLE;
    for(i=1; i<=m; i++) a[i][0] = 0;
    for(j=1; j<=n; j++) if( dcmp(c[j]) &&  basic[j] )
    {
        tmp = c[j]; ans += rhs[col[j]]*tmp;
        for(i=0; i<=n; i++) c[i] -= tmp*a[col[j]][i];
    }
    return FEASIBLE;
}

int simplex(int n, int m, double *c, double a[maxn][maxn],
    double *rhs, double &ans, double *x) // standard form
{
    int i,j,k;
    for(i=1; i<=m; i++)
    {
        for(j=n+1; j<=n+m; j++) a[i][j]=0;
        a[i][n+i] = 1; a[i][0] = 0;
        row[i] = n+i; col[n+i] = i;
    }
    k = PhaseI (n+m, m, c, a, rhs, ans);
    if( k == INFEASIBLE ) return k;
    k = PhaseII(n+m, m, c, a, rhs, ans, 0);
    for(j=0;j<=n+m;j++) x[j]=0;
    for(i=1;i<=m;i++) x[row[i]]=rhs[i];
    return k;
}

int n, m; double c[maxn],ans, a[maxm][maxn], rhs[maxm], x[maxn];

int main()
{
    ifstream cin("lp.in");
    int i,j;
    while( cin>>n>>m && !cin.fail() )
    {
        for(j=1; j<=n; j++) cin>>c[j]; cin>>ans; c[0]=0;
        for(i=1; i<=m; i++){ for(j=1; j<=n; j++) cin>>a[i][j]; cin>>rhs[i]; }
        switch( simplex(n, m, c, a, rhs, ans, x) )
        {
            case OPTIMAL :
                printf("OPTIMAL\n%10lf\n",ans);
                for(j=1;j<=n;j++) printf("x[ %2d ] = %10lf\n",j,x[j]);
                break;
            case UNBOUNDED :
                printf("UNBOUNDED\n");  break;
            case INFEASIBLE :
                printf("INFEASIBLE\n"); break;
        }   printf("\n");
    }
    return 0;
}
```

Chapter 5

Computational Geometry

5.1 Basic Operations

```
const double eps = 1e-8;
const double pi  = acos(-1.0);

struct CPoint{ double x,y; };

double min(double x,double y){ if( x<y ) return x; else return y; }

double max(double x,double y){ if( x>y ) return x; else return y; }

double sqr(double x){ return x*x; }

int dcmp(double x)
{
    if(x<-eps) return -1; else return (x>eps);
}

double cross(CPoint p0,CPoint p1,CPoint p2)
{
    return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
}

double dot(CPoint p0,CPoint p1,CPoint p2)
{
    return (p1.x-p0.x)*(p2.x-p0.x)+(p1.y-p0.y)*(p2.y-p0.y);
}

double dissqr(CPoint p1,CPoint p2)
{
    return sqr(p1.x-p2.x)+sqr(p1.y-p2.y);
}

double dis(CPoint p1,CPoint p2)
{
    return sqrt(sqr(p1.x-p2.x)+sqr(p1.y-p2.y));
}

int PointEqual(const CPoint &p1,const CPoint &p2)
{
    return dcmp(p1.x-p2.x)==0 && dcmp(p1.y-p2.y)==0;
}
```

5.2 Extended Operations

```
// Crossing Angle of P0P1 -> P0P2, range in (-pi,pi]
double angle(CPoint p0,CPoint p1,CPoint p2)
{
    double cr = cross(p0,p1,p2);
    double dt =  dot (p0,p1,p2);
    if(dcmp(cr)==0) cr=0.0;
    if(dcmp(dt)==0) dt=0.0;
    return atan2(cr,dt);
}

int PointOnLine(CPoint p0,CPoint p1,CPoint p2)
{
    return dcmp(cross(p0,p1,p2))==0;
}

int PointOnSegment(CPoint p0,CPoint p1,CPoint p2)
{
    return dcmp(cross(p0,p1,p2))==0 && dcmp(dot(p0,p1,p2))<=0;
}

// 1 = cross;   0 = parallel;   -1 = overlap
int LineIntersection(CPoint p1,CPoint p2,CPoint p3,CPoint p4,CPoint &cp)
{
    double u=cross(p1,p2,p3), v=cross(p2,p1,p4);
    if( dcmp(u+v) )
    {
        cp.x=(p3.x*v + p4.x*u) / (v+u);
        cp.y=(p3.y*v + p4.y*u) / (v+u);
        return 1;
    }
    if( dcmp(u) ) return 0;      // else u=v=0;
    if( dcmp(cross(p3,p4,p1)) ) return 0;
    return -1;
}

int SegmentIntersection(CPoint p1,CPoint p2,CPoint p3,CPoint p4,CPoint &cp)
{
    int ret=LineIntersection(p1,p2,p3,p4,cp);
    if(ret==1) return PointOnSegment(cp,p1,p2) && PointOnSegment(cp,p3,p4);
    if(ret==-1 && ( PointOnSegment(p1,p3,p4) || PointOnSegment(p2,p3,p4)
                 || PointOnSegment(p3,p1,p2) || PointOnSegment(p4,p1,p2) ))
        return -1;
    return 0;
}

int SegmentIntersecTest(CPoint p1,CPoint p2,CPoint p3,CPoint p4)
{
    if(  max(p1.x, p2.x) + eps < min(p3.x, p4.x) ||
         max(p3.x, p4.x) + eps < min(p1.x, p2.x) ||
         max(p1.y, p2.y) + eps < min(p3.y, p4.y) ||
         max(p3.y, p4.y) + eps < min(p1.y, p2.y) ) return 0;
    int d1=dcmp(cross(p3,p4,p2));
    int d2=dcmp(cross(p3,p4,p1));
    int d3=dcmp(cross(p1,p2,p4));
    int d4=dcmp(cross(p1,p2,p3));
    if( d1*d2==1 || d3*d4 ==1 ) return 0;
    if( d1==0 && d2==0 && d3==0 && d4==0 ) return -1;
    return 1;
}
```

```
// 0 = outside;     1 = inside;     2 = boundary
int PointInPolygon(CPoint cp,CPoint p[],int n)
{
    int i,k,d1,d2,wn=0;
    double sum=0;
    p[n]=p[0];
    for(i=0;i<n;i++)
    {
        if( PointOnSegment(cp,p[i],p[i+1]) ) return 2;
        k  = dcmp( cross(p[i],p[i+1],cp) );
        d1 = dcmp( p[i+0].y - cp.y );
        d2 = dcmp( p[i+1].y - cp.y );
        if(k>0 && d1<=0 && d2>0) wn++;
        if(k<0 && d2<=0 && d1>0) wn--;
    }
    return wn!=0;
}

double PointToLine(CPoint p0,CPoint p1,CPoint p2,CPoint &cp)
{
    double d=dis(p1,p2);
    double s = cross(p1,p2,p0)/d;
    cp.x = p0.x + s*(p2.y-p1.y)/d;
    cp.y = p0.y - s*(p2.x-p1.x)/d;
    return s; // **************** Signed Magnitude *****************
}

void PointProjLine(CPoint p0,CPoint p1,CPoint p2,CPoint &cp)
{
    double t = dot(p1,p2,p0)/dot(p1,p2,p2);
    cp.x = p1.x + t*(p2.x-p1.x);
    cp.y = p1.y + t*(p2.y-p1.y);
}
```

5.3 Convex Hull

Graham Scan, $O(N \log N)$

```
CPoint bp; // for polar sorting

int PolarCmp(const CPoint &p1,const CPoint &p2)
{
    int u=dcmp(cross(bp,p1,p2));
    return u>0 || ( u==0 && dcmp(dissqr(bp,p1)-dissqr(bp,p2))<0 );
}

void GrahamScan(CPoint pin[], int n, CPoint ch[], int &m)
{
    int i,j,k,u,v;
    memcpy(ch, pin, n*sizeof(CPoint));
    for(i=k=0; i<n; i++)
    {
        u = dcmp( ch[i].x - ch[k].x );
        v = dcmp( ch[i].y - ch[k].y );
        if( v<0 || (v==0 && u<0) ) k=i;
    }
    bp = ch[k];
    std::sort(ch, ch+n, PolarCmp);
    n = std::unique(ch, ch+n, PointEqual)-ch;
    if( n<=1 ) { m = n; return; }
    if( dcmp(cross(ch[0],ch[1],ch[n-1]))==0 )
        { m=2; ch[1]=ch[n-1]; return; }
    ch[n++]=ch[0];
    for(i=1,j=2; j<n; j++)
    {
        while( i>0 && dcmp(cross(ch[i-1],ch[i],ch[j]))<=0 ) i--;
        ch[++i] = ch[j];
    }
    m=i;
}

void GrahamScanReserved(CPoint pin[], int n, CPoint ch[], int &m)
{
    int i,j,k,u,v;
    memcpy(ch, pin, n*sizeof(CPoint));
    for(i=k=0; i<n; i++)
    {
        u = dcmp( ch[i].x - ch[k].x );
        v = dcmp( ch[i].y - ch[k].y );
        if( v<0 || (v==0 && u<0) ) k=i;
    }
    bp = ch[k];
    std::sort(ch, ch+n, PolarCmp);
    n = std::unique(ch, ch+n, PointEqual)-ch;
    if( n>0 && dcmp(cross(ch[0],ch[1],ch[n-1])) )
    {
        for(i=n-1; dcmp(cross(ch[0],ch[n-1],ch[i]))==0; i--);
        std::reverse(ch+i+1,ch+n);
    }
    for(m=0,i=0; i<n; i++)
    {
        while( m>=2 && dcmp(cross(ch[m-2],ch[m-1],ch[i]))<0 ) m--;
        ch[m++] = ch[i];
    }
}
```

Montone Chain, $O(N \log N)$

```
int VerticalCmp(const CPoint &p1,const CPoint &p2)
{
    return p1.y+eps<p2.y || (p1.y<p2.y+eps && p1.x+eps<p2.x);
}

void MontoneChain(CPoint pin[], int n, CPoint ch[], int &m)
{
    int i,k; CPoint *p = new CPoint[n];
    memcpy(p, pin, n*sizeof(CPoint));
    std::sort(p,p+n,VerticalCmp);
    n = std::unique(p,p+n,PointEqual)-p;
    for( m=i=0; i<n; i++ )
    {
        while( m>1 && dcmp(cross(ch[m-2],ch[m-1],p[i]))<=0 ) m--;
        ch[m++]=p[i];
    }
    k=m;
    for( i=n-2; i>=0; i-- )
    {
        while( m>k && dcmp(cross(ch[m-2],ch[m-1],p[i]))<=0 ) m--;
        ch[m++]=p[i];
    }
    if(n>1) m--;
    delete p;
}

void MontoneChainReserved(CPoint pin[], int n, CPoint ch[], int &m)
{
    int i,k;
    CPoint *p = new CPoint[n]; memcpy(p, pin, n*sizeof(CPoint));
    std::sort(p,p+n,VerticalCmp);
    n = std::unique(p,p+n,PointEqual)-p;
    for( m=i=0; i<n; i++ )
    {
        while( m>1 && dcmp(cross(ch[m-2],ch[m-1],p[i]))<0 ) m--;
        ch[m++]=p[i];
    }
    if( n==m ) return;
    k=m;
    for( i=n-2; i>=0; i-- )
    {
        while( m>k && dcmp(cross(ch[m-2],ch[m-1],p[i]))<0 ) m--;
        ch[m++]=p[i];
    }
    if(n>1) m--;
    delete p;
}
```

Javis March, $O(N\,H)$

```
int ConvexJavisMarchCmp(CPoint p0,CPoint p1,CPoint pnew)
{
    int u=dcmp(cross(p0,p1,pnew));
    return (u<0 || (u==0 && dcmp(dissqr(pnew,p0)-dissqr(p1,p0))>0));
}

void ConvexJavisMarch(CPoint pin[], int n, CPoint ch[], int &m)
{
    int i,j,k,u,v;
    char *mk = new char[n];
    CPoint *p = new CPoint[n];
    memcpy(p,pin,n*sizeof(CPoint));
    memset(mk,0,n);
    for(i=k=0;i<n;i++)
    {
        u=dcmp(p[i].x-p[k].x);
        v=dcmp(p[i].y-p[k].y);
        if( v<0 || (v==0 && u<0) ) k=i;
    }
    for(m=0; !mk[k]; m++)
    {
        mk[k]=1; ch[m]=p[k];
        for(j=k=0;j<n;j++) if(ConvexJavisMarchCmp(ch[m],p[k],p[j])) k=j;
    }
    delete p;
    delete mk;
}
```

5.4 Point Set Diameter

P must be convex in ccw order and no trhee points on an edge and will be changed after computing it's convex hull

```
double Diameter(CPoint *p, int n)
{
    Convex( p, n, p, n );
    if( n==1 ) return 0;
    if( n==2 ) return dis( p[0], p[1] );
    int u, nu, v, nv, k; double ret = 0;
    p[n] = p[0];
    for( u=0,v=1; u<n; u=nu )
    {
        nu = u+1;
        while(1) {
            nv = (v+1)%n;
            k = dcmp(  (p[nu].x-p[u].x) * (p[nv].y-p[v].y)
                     - (p[nv].x-p[v].x) * (p[nu].y-p[u].y) );
            if( k<=0 ) break;
            v=nv;
        }
        ret = max( ret, dis(p[u],p[v]) );
        if( k==0 ) ret = max( ret, dis(p[u],p[nv]) );
    }
    return ret;
}
```

5.5 Closest Pair

```
#define sqr(z) ((z)*(z))
struct point { double x,y; } pt[maxn]; // [1..n]
int n,o[maxn],on;

int dcmp(double a,double b) {
    if (a - b < 1e-10 && b - a < 1e-10) return 0;
    if (a > b) return 1; return -1;
}

bool cmp(const point& a,const point& b)
{ return dcmp(a.x,b.x) < 0; }

bool cmp2(const int& a,const int& b)
{ return dcmp(pt[a].y,pt[b].y) < 0; }

double dis(point a,point b)
{ return sqrt( sqr(a.x - b.x) + sqr(a.y - b.y) ); }

double min(double a,double b) { return a < b ? a : b; }

double search(int s,int t) {
    int mid = (s + t) / 2,i,j; double ret(1e300);
    if (s >= t) return ret;
    for(i=mid; i>=s && !dcmp(pt[i].x,pt[mid].x); i--); ret=search(s,i);
    for(i=mid; i<=t && !dcmp(pt[i].x,pt[mid].x); i++);
    ret=min(ret,search(i,t)); on=0;
    for(i=mid; i>=s && dcmp(pt[mid].x-pt[i].x,ret)<=0; i--) o[++on]=i;
    for(i=mid+1; i<=t && dcmp(pt[i].x-pt[mid].x,ret)<= 0; i++) o[++on]=i;
    std::sort(o+1,o+on+1,cmp2);
    for(i=1; i<=on; i++) for(j=1; j<=10; j++) if(i+j<=on)
        ret = min(ret,dis(pt[o[i]],pt[o[i+j]]));
    return ret;
}

double solve() { std::sort(pt+1,pt+1+n,cmp); return search(1,n); }
```

5.6 Circles

Crossing of $|P - P_0| = r$ and $ax + by + c = 0$

```
int CircleCrossLine_1( CPoint p0, double r,
    double a, double b, double c, CPoint &cp1, CPoint &cp2)
{
    double aa = a * a, bb = b * b, s = aa + bb;
    double d = r*r*s - sqr(a*p0.x+b*p0.y+c);
    if( d+eps<0 ) return 0;
    if( d<eps ) d = 0; else d = sqrt( d );
    double ab = a * b, bd = b * d, ad = a * d;
    double xx = bb * p0.x - ab * p0.y - a * c;
    double yy = aa * p0.y - ab * p0.x - b * c;
    cp2.x = ( xx + bd ) / s; cp2.y = ( yy - ad ) / s;
    cp1.x = ( xx - bd ) / s; cp1.y = ( yy + ad ) / s;
    if( d>eps ) return 2; else return 1;
}
```

Crossing of $|P - P_0| = r$ and $\overrightarrow{P_1P_2}$

```
int CircleCrossLine_2( CPoint p0, double r,
                CPoint p1, CPoint p2, CPoint &cp1, CPoint &cp2)
{
    double d, d12, dx, dy;
    d = fabs(PointToLine( p0, p1, p2, cp1 ));
    if( dcmp(d-r) >0 ) return 0;
    if( dcmp(d-r)==0 ) { cp2 = cp1; return 1; }
    d = sqrt( r*r - d*d ) / dis( p1, p2 );
    dx = ( p2.x - p1.x ) * d;
    dy = ( p2.y - p1.y ) * d;
    cp2.x = cp1.x + dx;  cp2.y = cp1.y + dy;
    cp1.x = cp1.x - dx;  cp1.y = cp1.y - dy;
    return 2;
}
```

Crossing of $|P - P_1| = r_1$ and $|P - P_2| = r_2$

```
int CircleCrossCircle_1( CPoint p1, double r1, CPoint p2, double r2,
    CPoint &cp1, CPoint &cp2 )
{
    double mx = p2.x-p1.x, sx = p2.x+p1.x, mx2 = mx*mx;
    double my = p2.y-p1.y, sy = p2.y+p1.y, my2 = my*my;
    double sq = mx2 + my2, d = -(sq-sqr(r1-r2))*(sq-sqr(r1+r2));
    if( d+eps<0 ) return 0; if( d<eps ) d = 0; else d = sqrt(d);
    double x = mx*( (r1+r2)*(r1-r2) + mx*sx ) + sx*my2;
    double y = my*( (r1+r2)*(r1-r2) + my*sy ) + sy*mx2;
    double dx = mx*d, dy = my*d;  sq *= 2;
    cp1.x = ( x - dy ) / sq;  cp1.y = ( y + dx ) / sq;
    cp2.x = ( x + dy ) / sq;  cp2.y = ( y - dx ) / sq;
    if( d>eps ) return 2; else return 1;
}
```

Crossing of $|P - P_1| = r_1$ and $|P - P_2| = r_2$

```
int CircleCrossCircle_2( CPoint p1, double r1, CPoint p2, double r2,
    CPoint &cp1, CPoint &cp2 )
{
    double a, b, c; CommonAxis( p1, r1, p2, r2, a, b, c);
    return CircleCrossLine_1( p1, r1, a, b, c, cp1, cp2);
}
```

Common Axis of $|P - P_1| = r_1$ and $|P - P_2| = r_2$ of the $ax + by + c = 0$ form

```
void CommonAxis(CPoint p1, double r1, CPoint p2, double r2,
                          double &a, double &b, double &c )
{
    double sx = p2.x + p1.x, mx = p2.x - p1.x;
    double sy = p2.y + p1.y, my = p2.y - p1.y;
    a = 2*mx; b = 2*my; c = - sx*mx - sy*my - (r1+r2)*(r1-r2);
}
```

5.7 Largest Empty Convex Polygon

```
#define ABS(x)                  ( (x)>=0 ? (x) : -(x) )
#define CROSS(x1, y1, x2, y2)   ( (x1)*(y2)-(x2)*(y1) )

const double eps = 1e-8;

struct CPoint { int x, y; };

int n; CPoint p[maxn]; double ans;

bool cmp(const CPoint &a, const CPoint &b) {
    int v = CROSS( a.x, a.y, b.x, b.y );
    if ( v>0 ) return true; if ( v<0 ) return false;
    return ( a.x*a.x + a.y*a.y < b.x*b.x + b.y*b.y );
}

CPoint c[maxn]; int nc; double fm[maxn][maxn];

void sweep(int x, int y) {
    int i, j, k, m; double v, best = 0;
    for(nc=i=0; i<n; ++i ) if( p[i].y<y || p[i].y==y && p[i].x<x )
        { c[nc].x=p[i].x-x;  c[nc++].y=p[i].y-y; }
    if( nc<2 ) return;
    std::sort(c, c + nc, cmp);
    memset(fm, 0, sizeof(fm));
    for( i=1; i<nc; ++i ) {
        j=i-1; while(j>=0 && CROSS(c[i].x, c[i].y, c[j].x, c[j].y)==0) --j;
        int nev = 0, ev[maxn];
        while( j>=0 ) {
            v = CROSS(c[j].x, c[j].y, c[i].x, c[i].y)/2.0;  k=j-1;
            while( k>=0 && CROSS( c[j].x-c[i].x, c[j].y-c[i].y,
                                  c[k].x-c[i].x, c[k].y-c[i].y ) >0 ) --k;
            if( k>=0 ) v += fm[j][k];
            if( v-best>eps ) best = v;
            if( CROSS(c[i].x, c[i].y, c[i-1].x, c[i-1].y) )
                if( v-fm[i][j]>eps ) fm[i][j]=v;
            ev[ nev++ ]=j; j=k;
        }
        if ( CROSS(c[i].x, c[i].y, c[i-1].x, c[i-1].y) )
            for(j=nev-2; j>=0; --j) if ( fm[i][ev[j+1]]-fm[i][ev[j]]>eps )
                fm[i][ev[j]]=fm[i][ev[j+1]];
    }
    if( best-ans>eps ) ans = best;
}

void main() {
    int t, i; for( scanf("%d", &t); t; --t ) { scanf("%d", &n);
        for(i=0; i<n; ++i ) scanf("%d %d", &p[i].x, &p[i].y);
        for(ans=i=0; i<n; ++i ) sweep(p[i].x, p[i].y); // main procedure
        printf("%.1lf\n", ans);
    }
}
```

5.8 Triangle Centers

```
// INPUT: (242, 89), (212, 185), (71, 128), OUTPUT: (158.0885, 115.4652)
void Circumcenter(CPoint p0,CPoint p1,CPoint p2,CPoint &cp)
{
    double a1=p1.x-p0.x, b1=p1.y-p0.y, c1=(sqr(a1)+sqr(b1))/2 ;
    double a2=p2.x-p0.x, b2=p2.y-p0.y, c2=(sqr(a2)+sqr(b2))/2 ;
    double d = a1 * b2 - a2 * b1;
    cp.x = p0.x + (c1*b2 - c2*b1) / d;
    cp.y = p0.y + (a1*c2 - a2*c1) / d;
}

// INPUT: (242, 89), (212, 185), (71, 128), OUTPUT: (189.5286, 137.4987)
double Incenter(CPoint A, CPoint B, CPoint C, CPoint &cp)
{
    double s, p , r, a, b, c;
    a = dis(B, C), b = dis(C, A), c = dis(A, B); p = ( a + b + c ) / 2;
    s = sqrt( p * (p-a) * (p-b) * (p-c) );        r = s / p;
    cp.x = ( a*A.x + b*B.x + c*C.x ) / ( a + b + c );
    cp.y = ( a*A.y + b*B.y + c*C.y ) / ( a + b + c );
    return r;
}

// INPUT: (242, 89), (212, 185), (71, 128), OUTPUT: (208.8229, 171.0697)
void Orthocenter(CPoint A, CPoint B, CPoint C, CPoint &cp)
{
    Circumcenter(A, B, C, cp);
    cp.x = A.x + B.x + C.x - 2 * cp.x;
    cp.y = A.y + B.y + C.y - 2 * cp.y;
}
```

Find three numbers r, s, t which make $P = rA + sB + tC$ and $r + s + t = 1$

```
void Parametric( CPoint P, CPoint A , CPoint B , CPoint C ,
                           double &r , double &s , double &t )
{
    double d;
    d = cross( A, B, C);
    r = cross( P, B, C) / d;
    s = cross( A, P, C) / d;
    t = cross( A, B, P) / d;
}

void PolygonCentroids(CPoint p[], int n, CPoint &cp)
{
    double sum=0, s=0; cp.x=0; cp.y=0;
    for( int i=1; i<n-1; i++,sum+=s )
    {
        s= cross( p[0], p[i], p[i+1] );
        cp.x += s*( p[0].x + p[i].x + p[i+1].x );
        cp.y += s*( p[0].y + p[i].y + p[i+1].y );
    }
    cp.x/=sum*3; cp.y/=sum*3;
}
```

5.9 Polyhedron Volume

Remark : All faces are assumed oriented **counterclockwise** from the outside; Volume6 returns six times the volume of the tetrahedron determined by abc and the origin d. Volume6 is positive iff d is on the negative side of abc, where the positive side is determined by the rh-rule. So the volume is positive if the ccw normal to abc points outside the tetrahedron.

```
struct TPoint { double x, y, z; };
typedef int TFace[3];

double Volume6( TPoint a, TPoint b, TPoint c, TPoint d ) // d = origin
{
    double  vol, bdx, bdy, bdz, cdx, cdy, cdz;
    bdx = b.x-d.x;  bdy = b.y-d.y;  bdz = b.z-d.z;
    cdx = c.x-d.x;  cdy = c.y-d.y;  cdz = c.z-d.z;
    vol =  ( a.z - d.z ) * ( bdx * cdy - bdy * cdx)
         + ( a.y - d.y ) * ( bdz * cdx - bdx * cdz)
         + ( a.x - d.x ) * ( bdy * cdz - bdz * cdy);
    return vol;
}

void main()
{
  int n, F, i, j; double vol;
  TPoint p[maxn]; TFace face[maxn*2-4];
  cin>>n; for(i=0; i<n; i++) cin >> p[i].x >> p[i].y >> p[i].z;
  cin>>F; for(i=0; i<F; i++) for(j=0;j<3;j++) cin >> face[i][j];
  if( F != 2 * n - 4 ) { printf( "Not a simple polyhedron!\n" ); return;}
  for( vol = i = 0; i < F; i++ )
    vol += Volume6( p[face[i][0]], p[face[i][1]], p[face[i][2]], p[0] );
  vol /= 6.0;  cout << vol <<endl;
}
```

5.10 Planar Graph Contour

```
int x[maxn],y[maxn],g[maxn][maxn],num[maxn],base,n,size,mk[maxn][maxn];
int s[maxn], used[maxn], ans; double angle[maxn];

bool cmp(const int &i,const int&j){ return angle[i] < angle[j]; }

void dfs(int d,int u,int v)
{
    int i,j,w; s[d] = u; used[u]++;
    if( mk[u][v] ) {
        if( d==size ) {
            used[u]--;
            for(j=1;j<=n;j++) if(used[j]>1) break; if(j<=n) return;
            if(j>n) ++ans;
        }
        return;
    }
    mk[u][v]=1;
    for(j=0; j<num[v]; j++) if( g[v][j]==u ) break;
    j = (j+1)%num[v]; w = g[v][j]; dfs(d+1, v, w);
}

void solve()
{
    int i,j,k,l,u,v;
    for(i=1; i<=n; i++){
        base=i;
        for(j=1; j<=n; j++) angle[j] = atan2(y[j]-y[i],x[j]-x[i]);
        std::sort(g[i], g[i]+num[i], cmp);
    }
    u = 1; memset(mk, 0, sizeof(mk));
    for(i=2; i<=n; i++) if(y[i]<y[u] || (y[i]==y[u] && x[i]<x[u])) u=i;
    for(v=-1,i=0; i<num[u]; i++) {
        j = g[u][i]; if( j==u || j==v ) continue;
        if( v<0 ) { v=j; continue; }
        k = (x[j]-x[u])*(y[v]-y[u])-(y[j]-y[u])*(x[v]-x[u]);
        if( k<0 ) v=j; else
        if( k==0 ) if( y[j]<y[v] || (y[j]==y[v] && x[j]<x[v]) ) v=j;
    }
    dfs(0, v, u); ans = 0;  // outer contour
    for(i=1; i<=n; i++) for(j=0; j<num[i]; j++)
        if( !mk[i][g[i][j]] )
        {
            memset(used,0,sizeof(used));
            dfs(0,i,g[i][j]);
        }
}

int main()
{
    int t,i,j,k,l;
    cin>>t; while(t-->0) {
        cin>>n;
        for(k=0; k<n; k++) {
            cin>>i; cin>>x[i]>>y[i]; cin>>num[i];
            for(j=0;j<num[i];j++) cin>>g[i][j];
        }
        cin>>size; ans=0; if(size<3) size=3;
        solve(); cout<<ans<<endl;
    } return 0;
}
```

5.11 Rectangles Area

```
struct TSegNode {
    TSegNode(int x, int y):L(x),R(y),Lch(-1),Rch(-1),count(0),len(0){}
    TSegNode(){TSegNode(-1,-1);}
    int L, R, Lch, Rch, count, len;
};

struct Tevent {
    int L, R, x;
    bool style;
    friend const bool operator< (Tevent a, Tevent b) { return a.x<b.x; }
};

int nlist, list[MAXN*4], total, n, nevent;
TSegNode node[MAXN*4]; Tevent event[MAXN*4];

void CreateTree(int r) {
    if ( node[r].R-node[r].L>1 ) {
        int mid = (node[r].L+node[r].R)>>1;
        node[total] = TSegNode(node[r].L, mid);
        node[r].Lch = total;  CreateTree(total++);
        node[total] = TSegNode(mid, node[r].R);
        node[r].Rch = total;  CreateTree(total++);
    }
}

void Update(int r, int L, int R, int v) {
    if ( L>=node[r].R || R<=node[r].L ) return;
    if ( L<=node[r].L && R>=node[r].R ) {
        node[r].count+=v;
        if ( v>0 && v==node[r].count ) node[r].len = node[r].R-node[r].L;
        if ( v<0 && node[r].count==0 ) if( node[r].Lch<0 ) node[r].len = 0;
        else node[r].len = node[node[r].Lch].len + node[node[r].Rch].len;
    } else {
        Update(node[r].Lch, L, R, v); Update(node[r].Rch, L, R, v);
        if ( node[r].count==0 ) node[r].len =
            node[node[r].Lch].len + node[node[r].Rch].len;
    }
}
```

```
int main() {
    int i, j, res, last;
    scanf("%d", &n);
    nevent=0; nlist=0;
    for ( i=0; i<n; ++i ) {
        int lx, ly, ux, uy;
        scanf("%d %d %d %d", &lx, &ly, &ux, &uy);
        if ( lx<ux && ly<uy ) {
            event[nevent].x = lx;   event[nevent].L = ly;
            event[nevent].R = uy;   event[nevent++].style = true;
            event[nevent].x = ux;   event[nevent].L = ly;
            event[nevent].R = uy;   event[nevent++].style = false;
        }
        list[nlist++] = ly;  list[nlist++] = uy;
    }
    std::sort(event, event+nevent);
    std::sort(list, list+nlist);
    nlist = std::unique(list, list+nlist)-list;
    node[total=0, total++] = TSegNode(0, nlist-1);
    CreateTree( 0 );
    for( i=0; i<total; ++i )
        { node[i].L = list[node[i].L]; node[i].R = list[node[i].R]; }
    res = i = 0;
    while ( i<nevent ) {
        for(last=event[i].x; event[i].x==last; ++i)
            Update(0, event[i].L, event[i].R, event[i].style ? 1 : -1);
        if ( i < nevent ) res += (event[i].x - last) * node[0].len;
    }
    printf("%d\n", res);
    return 0;
}
```

5.12 Rectangles Perimeter

```
#define ABS(x) ( (x)>=0 ? (x) : -(x) )

struct TSegNode {
    TSegNode(int x, int y):L(x),R(y),Lch(-1),Rch(-1),count(0),len(0){}
    TSegNode(){TSegNode(-1,-1);}
    int L, R, Lch, Rch, count, len;
};

struct Tevent {
    int L, R, x; bool style;
    friend const bool operator< (Tevent a, Tevent b)
    { if ( a.x!=b.x ) return a.x<b.x; return ( a.style && !b.style ); }
};

int n, lx[MAXN], ly[MAXN], ux[MAXN], uy[MAXN], total, nevent, res;
TSegNode node[MAXN*4]; Tevent event[MAXN*4];

void CreateTree(int r) {
    if ( node[r].R-node[r].L>1 ) {
        int mid = (node[r].L+node[r].R)>>1;
        node[total] = TSegNode(node[r].L, mid);
        node[r].Lch = total;  CreateTree(total++);
        node[total] = TSegNode(mid, node[r].R);
        node[r].Rch = total;  CreateTree(total++);
    }
}

void Update(int r, int L, int R, int v) {
    if ( L>=node[r].R || R<=node[r].L ) return;
    if ( L<=node[r].L && R>=node[r].R ) {
        node[r].count+=v;
        if ( v>0 && v==node[r].count ) node[r].len = node[r].R-node[r].L;
        if ( v<0 && node[r].count==0 ) if( node[r].Lch<0 ) node[r].len = 0;
        else node[r].len = node[node[r].Lch].len + node[node[r].Rch].len;
    } else {
        Update(node[r].Lch, L, R, v); Update(node[r].Rch, L, R, v);
        if ( node[r].count==0 ) node[r].len =
            node[node[r].Lch].len + node[node[r].Rch].len;
    }
}
```

```
void process() {
    int nlist, list[MAXN*2], last, i, now;
    nevent = 0;  nlist = 0;
    for( i=0; i<n; ++i ) {
        event[nevent].x = lx[i];  event[nevent].L      = ly[i];
        event[nevent].R = uy[i];  event[nevent++].style = true;
        event[nevent].x = ux[i];  event[nevent].L      = ly[i];
        event[nevent].R = uy[i];  event[nevent++].style = false;
        list[nlist++]   = ly[i];  list[nlist++]        = uy[i];
    }
    std::sort(event, event+nevent);
    std::sort(list, list+nlist);
    nlist = int(std::unique(list, list+nlist)-list);
    node[total=0, total++] = TSegNode(0, nlist-1);
    CreateTree( 0 );
    for(i=0; i<total; ++i )
        { node[i].L = list[node[i].L];  node[i].R = list[node[i].R]; }
    last = i = 0;
    while( i<nevent ) {
        now = event[i].x;
        while( i<nevent && event[i].x==now && event[i].style )
            { Update(0, event[i].L, event[i].R, 1); ++i; }
        res += ABS(node[0].len-last); last = node[0].len;
        while( i<nevent && event[i].x==now )
            { Update(0, event[i].L, event[i].R, -1); ++i; }
        res += ABS(node[0].len-last); last = node[0].len;
    }
}

int main() {
    int i;
    scanf("%d", &n);
    for(i=0; i<n; ++i) scanf("%d %d %d %d",&lx[i],&ly[i],&ux[i],&uy[i]);
    res=0;  process();
    for(i=0; i<n; ++i){std::swap(lx[i], ly[i]); std::swap(ux[i], uy[i]);}
    process(); printf("%d\n", res);
    return 0;
}
```

5.13 Smallest Enclosing Circle

$O(N^3)$, compute Convex Hull first! or it will be quite slow!

```
double GetCos(CPoint p0,CPoint p1,CPoint p2)
{ return dot(p0,p1,p2)/dis(p0,p1)/dis(p0,p2); }

int allin(CPoint p[],int n,int i,int j,int k)
{
    for(int l=0; l<n; l++) if(l!=i && l!=j && l!=k) {
        if( (cross(p[i],p[j],p[k])>0)^(cross(p[i],p[j],p[l])>0) &&
          dcmp(GetCos(p[k],p[i],p[j])+GetCos(p[l],p[i],p[j]))>0) return 0;
        if( (cross(p[j],p[k],p[i])>0)^(cross(p[j],p[k],p[l])>0) &&
          dcmp(GetCos(p[i],p[k],p[j])+GetCos(p[l],p[k],p[j]))>0) return 0;
        if( (cross(p[i],p[j],p[k])>0)^(cross(p[i],p[l],p[k])>0) &&
          dcmp(GetCos(p[j],p[k],p[i])+GetCos(p[l],p[k],p[i]))>0) return 0;
    }
    return 1;
}

double SmallestEnclosingCircle(CPoint p[],int n,CPoint &cp)
{
    int i,j,k; double di,cos1,cos2,co,si,r=0;
    if( n == 1 ) { cp = p[0]; return 0; }
    if( n == 2 )
    {
        cp.x = ( p[0].x + p[1].x )/2;
        cp.y = ( p[0].y + p[1].y )/2;
        return dis(p[0],p[1])/2;
    }
    for(i=0; i<n; i++) for(j=i+1; j<n; j++)
    {
        di = dis(p[i],p[j]);   cos1 = cos2 = -2;
        if( dcmp(di-r*2)>0 ) r = di/2;
        for(k=0; k<n; k++) if( k!=i && k!=j )
        {
            co = GetCos(p[k],p[i],p[j]);
            if( dcmp(cross(p[i],p[j],p[k]))>0 )
                { if( co>cos1 ) cos1=co; }
            else  if( co>cos2 ) cos2=co;
        }
        if( dcmp(cos1)<=0 && dcmp(cos2)<=0 )
        {
            cp.x = ( p[i].x + p[j].x )/2;
            cp.y = ( p[i].y + p[j].y )/2;
            return di/2;
        }
    }
    r = 1e30;
    for(i=0; i<n; i++) for(j=i+1; j<n; j++) {
        di = dis( p[i], p[j] );
        for(k=j+1; k<n; k++) {
            co = GetCos( p[k], p[j], p[i] );
            si = sqrt(1-sqr(co));
            if( dcmp(di/si/2-r)<0 && allin(p,n,i,j,k) ) {
                r=di/si/2;
                GetCircleCenter(p[i],p[j],p[k],cp);
            }
        }
    }
    return r;
}
```

5.14 Smallest Enclosing Ball

```
const double eps = 1e-10;

struct point_type { double x, y, z; };

int npoint, nouter;
point_type point[10000], outer[4],res;
double radius,tmp;

inline double dist(point_type p1, point_type p2)
{
    double dx=p1.x-p2.x, dy=p1.y-p2.y, dz=p1.z-p2.z;
    return ( dx*dx + dy*dy + dz*dz );
}

inline double dot(point_type p1, point_type p2)
{
    return p1.x*p2.x + p1.y*p2.y + p1.z*p2.z;
}

void minball(int n)
{
    ball();
    if( nouter<4 )
        for(int i=0; i<n; ++i)
            if( dist(res, point[i])-radius>eps )
            {
                outer[nouter]=point[i];
                ++nouter;
                minball(i);
                --nouter;
                if( i>0 )
                {
                    point_type Tt = point[i];
                    memmove(&point[1], &point[0], sizeof(point_type)*i);
                    point[0]=Tt;
                }
            }
}
```

```
void ball() {
    point_type q[3]; double m[3][3], sol[3], L[3], det; int i,j;
    res.x = res.y = res.z = radius = 0;
    switch ( nouter ) {
        case 1: res=outer[0]; break;
        case 2:
            res.x=(outer[0].x+outer[1].x)/2;
            res.y=(outer[0].y+outer[1].y)/2;
            res.z=(outer[0].z+outer[1].z)/2;
            radius=dist(res, outer[0]);
            break;
        case 3:
            for(i=0; i<2; ++i ) {
                q[i].x=outer[i+1].x-outer[0].x;
                q[i].y=outer[i+1].y-outer[0].y;
                q[i].z=outer[i+1].z-outer[0].z;
            }
            for(i=0; i<2; ++i ) for(j=0; j<2; ++j )
                m[i][j]=dot(q[i], q[j])*2;
            for (i=0; i<2; ++i ) sol[i]=dot(q[i], q[i]);
            if( fabs(det=m[0][0]*m[1][1]-m[0][1]*m[1][0])<eps ) return;

            L[0]=(sol[0]*m[1][1] - sol[1]*m[0][1])/det;
            L[1]=(sol[1]*m[0][0] - sol[0]*m[1][0])/det;

            res.x=outer[0].x+q[0].x*L[0]+q[1].x*L[1];
            res.y=outer[0].y+q[0].y*L[0]+q[1].y*L[1];
            res.z=outer[0].z+q[0].z*L[0]+q[1].z*L[1];
            radius=dist(res, outer[0]);
            break;
        case 4:
            for(i=0; i<3; ++i){
                q[i].x=outer[i+1].x-outer[0].x;
                q[i].y=outer[i+1].y-outer[0].y;
                q[i].z=outer[i+1].z-outer[0].z;
                sol[i]=dot(q[i], q[i]);
            }
            for(i=0;i<3;++i) for(j=0;j<3;++j) m[i][j]=dot(q[i],q[j])*2;
            det= m[0][0]*m[1][1]*m[2][2] + m[0][1]*m[1][2]*m[2][0]
                + m[0][2]*m[2][1]*m[1][0] - m[0][2]*m[1][1]*m[2][0]
                - m[0][1]*m[1][0]*m[2][2] - m[0][0]*m[1][2]*m[2][1];

            if ( fabs(det)<eps ) return;

            for(j=0; j<3; ++j){
                for(i=0; i<3; ++i) m[i][j]=sol[i];
                L[j]=( m[0][0]*m[1][1]*m[2][2] + m[0][1]*m[1][2]*m[2][0]
                      + m[0][2]*m[2][1]*m[1][0] - m[0][2]*m[1][1]*m[2][0]
                      - m[0][1]*m[1][0]*m[2][2] - m[0][0]*m[1][2]*m[2][1]
                     )/ det;
                for(i=0; i<3; ++i) m[i][j]=dot(q[i], q[j])*2;
            }
            res=outer[0];
            for (i=0; i<3; ++i ) {
                res.x += q[i].x * L[i];
                res.y += q[i].y * L[i];
                res.z += q[i].z * L[i];
            }
            radius=dist(res, outer[0]);
    }
}
```

Chapter 6

Classic Problems

6.1 Bernoulli Number Generator

Note: *fraction.h* contains a *Fraction Class* (Section 1.4 on Page 8)

```
#include<fraction.h>

Fraction a[22];
int c[22][22];

int main()
{
    int i,j,k,m;
    c[0][0] = 1;
    for (i=1;i<=21;i++) {
        c[i][0] = 1; c[i][i] = 1;
        for (j=1;j<i;j++) c[i][j] = c[i-1][j] + c[i-1][j-1];
    }
    a[0] = 0;
    while( cin>>k ) {
        a[k+1] = Fraction(1,k+1);  m = k+1;
        for (i=k;i>=1;i--) {
            a[i] = 0;
            for (j=i+1;j<=k+1;j++)
                if ((j-i+1)%2==0) a[i] = a[i]+a[j]*c[j][j-i+1];
                    else a[i] = a[i]-a[j]*c[j][j-i+1];
            a[i] = a[i] * Fraction(1,i);
            m = lcm(m,a[i].get_denominator());
        }
        cout << m << ' ';
        for (i=k+1;i>0;i--) cout << a[i] * m<< ' ';
        cout << 0 << endl;
    }
    return 0;
}
```

6.2 Baltic OI'99 Expressions

$f(n, d)$ is the number of trees whose depth is less or equal than d.

```
int f[maxn][maxd], h[maxn][maxn], n, d;

int main()
{
    ifstream cin("input.txt");
    int i, j, k;
    for(d=1; d<maxd; d++) {
        memset(h, 0, sizeof(h));
        for(i=0; i<=d; i++) h[i][0]=1;
        for(i=1; i<maxn; i++) for(j=i-d; j<=i; j++)
            if(j>=0) h[i][j]=h[i-1][j]+h[i][j-1];
        for(i=1;i<maxn;i++) f[i][d]=h[i][i];
    }
    while(cin>>n>>d && n) cout<< f[n/2][d]- f[n/2][d-1]<<endl;
    return 0;
}
```

6.3 Bead Coloring — Pólya Theory

Use C colors to color L-bead necklace , the non-isomorphic number of the necklaces is :

If L is odd ,

$$f(C, L) = \frac{1}{2L}(LC^{\frac{L+1}{2}} + \sum_{K=1}^{L} C^{(K,L)})$$

If L is even,

$$f(C, L) = \frac{1}{2L}(\frac{L}{2}(C^{\frac{L}{2}} + C^{\frac{L}{2}+1}) + \sum_{K=1}^{L} C^{(K,L)})$$

```
int ans, n, m, mk[maxn], id[maxn], num;

int main()
{
    int i, j, k, l, d, u, p[maxn];
    while(cin>>n>>m && n && m){
        for(p[0]=i=1; i<=m; i++) p[i]=p[i-1]*n;
        for(ans=num=i=0;i<m;i++) id[i]=i;
        for(l=0;l<2;l++){
            for(i=0;i<m;i++) {
                memset(mk, 0, sizeof(mk));
                for(k=j=0;j<m;j++) if(!mk[id[j]])
                    for(k++, u=id[j];!mk[u];u=id[(u+i)%m]) mk[u]=1;
                num++; ans+=p[k];
            }
            std::reverse(id, id+m);
        }   cout<<ans/num<<endl;
    }
    return 0;
}
```

6.4 Binary Stirling Number

Parity of the Stirling number of the second kind

```
#define int long long

int calc(int n,int k)
{
        if( k==0 ) if( n==0 ) return 1; else return 0;
                else if( k==1 ) return 1; else
        {
                int p = 0, p2 = 1;
                while( k>p2*2 || n-k/2>p2 ) { p++; p2<<=1; }
                if( k>p2) return calc(n-p2,k-p2);
                if( n-k>=p2/2 ) return calc(n-p2/2,k);
                return 0;
        }
}
```

6.5 Box Surface Distance

```
int r,L,H,W,x1,y1,z1,x2,y2,z2;

void turn(int i,int j,int x,int y,int z,int x0,int y0,int L,int W,int H){
    if(z==0){ int R=x*x+y*y; if(R<r) r=R; }else{
        if(i>=0 && i< 2) turn(i+1,j, x0+L+z,y,x0+L-x, x0+L,y0, H,W,L);
        if(j>=0 && j< 2) turn(i,j+1, x,y0+W+z,y0+W-y, x0,y0+W, L,H,W);
        if(i<=0 && i>-2) turn(i-1,j, x0-z,y,x-x0,     x0-H,y0, H,W,L);
        if(j<=0 && j>-2) turn(i,j-1, x,y0-z,y-y0,     x0,y0-H, L,H,W);
    }
}

int main(){
    while(cin>> L >>W>> H >> x1 >> y1 >> z1 >> x2 >>y2 >> z2){
        if(z1!=0 && z1!=H) if(y1==0 || y1==W)
            { std::swap(y1,z1); std::swap(y2,z2); std::swap(W,H); } else
            { std::swap(x1,z1); std::swap(x2,z2); std::swap(L,H); }
        if(z1==H) z1=0,z2=H-z2;
        r=0x3fffffff; turn(0,0,x2-x1,y2-y1,z2,-x1,-y1,L,W,H);
        cout<<r<<endl;
    }
    return 0;
}
```

6.6 Calculate Expression

```
char expr[MAX+1]; int next[MAX],stack[MAX],top;
double calc(int L, int R);

void prefix() {
    int i; top=-1;
    for ( i=0; expr[i]; ++i ) {
        next[i]=-1;
        if ( expr[i]=='(' ) stack[++top]=i;
        else if ( expr[i]==')' ) next[stack[top--]]=i;
    }
}
```

```
double getnum(int &L) {
    double res = 0;
    if ( expr[L]=='(' ) { res=calc(L+1, next[L]-1); L=next[L]+1; }
        else while ( isdigit(expr[L]) ) res=res*10+expr[L++]-'0';
    return res;
}

void process(double &a, double b, char op) {
    switch ( op ) {
        case '+': a += b; break;
        case '-': a -= b; break;
        case '*': a *= b; break;
        default : a /= b;
    }
}

double calc(int L, int R) {
    double a, b, c;
    char op1, op2;
    if ( next[L] == R ) return calc(L+1, R-1);
    a = 0; op1 =( expr[L]=='-' ? '-' : '+' );
    L = ( expr[L]=='+' || expr[L]=='-' ? L+1 : L );
    for( b = getnum(L); L<R ; ) {
        op2=expr[L++]; c=getnum(L);
        if ( op2=='+' || op2=='-' || op1=='*' || op1=='/' ) {
            process(a, b, op1); b=c; op1=op2;
        } else process(b, c, op2);
    }
    process(a, b, op1);
    return a;
}

void main() {
    scanf("%s", expr); prefix();
    printf("%.10lf\n", calc(0, strlen(expr)));
}
```

6.7 Cartesian Tree

```
int lson[maxn],rson[maxn],pnt[maxn],root,n;

void BuildCartesianTree(int a[],int n)
{
    int i,j;
    for(i=0; i<n; j=i++) {
        pnt[i]=i-1; lson[i]=rson[i]=-1; j=i;
        while( pnt[j]>=0 && a[i]>a[pnt[j]] ) j = pnt[j];
        if( j!=i ) { pnt[i]=pnt[j]; lson[i]=j; pnt[j]=i; };
        if(pnt[i]>=0) rson[pnt[i]]=i;
    }
    for(i=0;i<n;i++) if(pnt[i]<0) root=i;
}
```

6.8 Catalan Number Generator

```
\\ Catalan[ 19] = 1767263190            < 2^31
\\ Catalan[ 35] = 3116285494907301262   < 2^63
\\ Catalan[100] = 896519947090131496687170070074100632420837521538745909320

#define maxn 1000
#define maxlen 700
#define maxpnum 400

int prime[maxpnum],primepos[maxn*2],num[maxpnum],pnum;

struct HP{ int len; int s[maxlen]; };

void PrintHP(HP x) { for(int i=x.len; i>0; i--) cout<<x.s[i]; }

void Multi(HP &x,int k)
{
    int i; for(i=1;i<=x.len;i++) x.s[i]*=k;
    x.len+=8; // log(10,maxn*2);
    for(i=1;i<=x.len;i++) { x.s[i+1]+=x.s[i]/10; x.s[i]%=10; }
    while( x.len>1 && !x.s[x.len]) x.len--;
}

void Factorize(int x,int flag)
{
    for(int i=0; prime[i]*prime[i]<=x; i++)
        while( x%prime[i]==0 ) { x/=prime[i]; num[i]+=flag; }
    if(x>1) num[primepos[x]]+=flag;
}

HP Catalan(int n)
{
    HP x; memset(&x,0,sizeof(x)); x.len=1; x.s[1]=1;
    memset(num,0,sizeof(num)); int i,j;
    for(i=1;i<=n;i++) { Factorize(2*n+1-i , 1); Factorize(i,-1); }
    Factorize(n+1,-1);
    for(i=0;i<pnum;i++) while(num[i]-->0) Multi(x,prime[i]);
    return x;
}

void InitPrimes()
{
    int i,j; pnum=0; memset(primepos,0,sizeof(primepos));
    for(i=2;i<=maxn*2;i++) if(!primepos[i]) {
        primepos[i]=pnum; prime[pnum++]=i;
        for(j=i+i;j<=maxn*2;j+=i) primepos[j]=-1;
    }
}

void main()
{
    InitPrimes(); int n;
    while(cin>>n) { PrintHP(Catalan(n));cout<<endl; }
}
```

6.9 Coloring Regular Polygon

Coloring regular n-vertex polygon with m white and $n-m$ black. When $n=17$ and $m=8$ OUTPUT : 750

```
int c[maxn][maxn], ans, n, m;

int gcd(int i,int j){ if(j==0) return i; else return gcd(j,i%j); }

int main()
{
    cin>>n>>m;
    int i,j,k,l,d;
    c[0][0]=1;
    for(i=1; i<maxn; i++) {
        c[i][0]=1;
        for(j=1; j<=i; j++) c[i][j]=c[i-1][j]+c[i-1][j-1];
    }
    for(k=0;k<m;k++) {
        d=gcd(m,k);
        if( n*d % m == 0 ) { l=n*d/m; ans+=c[l-1][d-1]; }
    }
    if(m%2==0) {
        if(n%2==0) ans+=(m/2)*c[n/2-1][m/2-1];
        if(m==2) ans+=(m/2)*(n-1); else
            for(i=2-n%2; i<=n-(m-2); i+=2)
                ans+=(m/2)*(i-1)*c[(n-i)/2-1][(m-2)/2-1];
    } else for(i=2-n%2; i<=n-(m-1); i+=2) ans+=m*c[(n-i)/2-1][(m-1)/2-1];
    cout<<ans/(2*m)<<endl;
    return 0;
}
```

6.10 Counting Inverse Pairs

```
#include<iostream.h>
#include<fstream.h>
#include<algorithm>

#define maxn 10000
int a[maxn],t[maxn],n,ans;

void sort(int b,int e)
{
    if(e-b<=0) return;
    int mid=(b+e)/2,p1=b,p2=mid+1,i=b;
    sort(b,mid); sort(mid+1,e);
    while( p1<=mid || p2<=e )
        if( p2>e || (p1<=mid && a[p1]<=a[p2]) ) t[i++]=a[p1++];
            else{ t[i++]=a[p2++]; ans+=mid-p1+1; }
    for(i=b;i<=e;i++)a[i]=t[i];
}

int main()
{
    ifstream cin("input.txt");
    int i,j;
    while(cin>>n) {
        for(i=0;i<n;i++) cin>>a[i];
        ans=0; sort(0,n-1); // Counting Inverse Number
        cout<<"Minimum exchange operations : "<<ans<<endl;
    }
    return 0;
}
```

6.11 Counting Trees

```
// Rooted     {1, 5, 11, 20, 30} => {1, 9, 1842, 12826228, 354426847597 }
// Non-Rooted {1, 3, 10, 25, 30} => {1, 1, 106, 104636890,  14830871802 }
void main()
{
    ifstream cin("input.txt");
    int i,j,n;
    memset(s, 0, sizeof(s)); a[0] = 0; a[1] = 1;
    for(i=1; i<maxn-1; i++)
    {
        a[i+1] = 0;
        for(j=1; j<=i; j++)
        {
            s[i][j] = s[i-j][j] + a[i+1-j];
            a[i+1] += j*a[j]*s[i][j];
        }
        a[i+1] /= i;
    }
    while(cin>>n) // a[n] = Rooted;   ans = Non-Rooted
    {
        int ans = a[n];
        for(i=1; 2*i<=n; i++) ans -= a[i] * a[n-i];
        if( n%2==0 ) ans += (a[n/2]+1) * a[n/2] / 2;
        cout << a[n] << " " << ans << endl;
    }
}
```

6.12 Eight Puzzle Problem

Input: 012345678   123456780 Output: $STEP = 22$

Common Part

```
#define maxlen 10
#define size 362880+1

const int link[9][5]={ {2,1,3}, {3,0,2,4}, {2,1,5}, {3,0,4,6},
    {4,1,3,5,7}, {3,2,4,8}, {2,3,7}, {3,4,6,8}, {2,5,6} };

int s[maxlen],p[maxlen],mk[size],open[size],cur,tail;

void encode(int *s,int len,int &x)
{
    int i,j,k,l; for(x=0,i=len-1; i>=0; x+=k*p[i--])
    for(k=s[i],j=i+1; j<len; j++) if(s[j]<s[i]) k--;
}

void decode(int *s,int len,int x)
{
    int i,j,k,l; for(i=len-1; i>=0; i--){ s[i]=x/p[i]; x%=p[i]; }
    for(i=0; i<len; i++) for(j=0; j<i; j++) if(s[j]>=s[i]) s[j]++;
}

void print(int *s,int len)
{
        for(int i=0;i<len;i++)
                cout<<s[i];
        cout<<endl;
}
```

```
int main()
{
    ifstream cin("input.txt");
    char ch; int i,src,dst;
    for(p[0]=i=1; i<maxlen; i++) p[i]=p[i-1]*i;
    for(i=0;i<9;i++) { cin>>ch; s[i]=ch-'0'; } encode(s,9,src);
    for(i=0;i<9;i++) { cin>>ch; s[i]=ch-'0'; } encode(s,9,dst);
    solve(src,dst); cout<<cur<<" "<<tail<<endl;
    return 0;
}
```

Simple Breadth First Search

```
void output(int pos,int num)
{
    if(pos==1) cout<<"Total number of steps = "<<num<<endl;
        else output(mk[open[pos]],num+1);
    decode(s,9,open[pos]); print(s,9);
}

void solve(int src,int dst)
{
    int i,j,k,x,l,ps;
    if(src==dst){ cout<<"SRC DST is the same!"<<endl; return; }
    cur=0; tail=1; open[1]=src; mk[src]=1;
    while(++cur<=tail){
        decode(s,9,open[cur]); for(ps=0; s[ps]; ps++);
        for(k=1; k<=link[ps][0]; k++){
            std::swap(s[ps],s[link[ps][k]]); encode(s,9,x);
            if(!mk[x]){
                mk[x]=cur; open[++tail]=x;
                if(x==dst) { output(tail,0); return; }
            }
            std::swap(s[ps],s[link[ps][k]]);
        }
    }
    cout<<"No solution!"<<endl;
}
```

Heuristic Breadth First Search

```
int d[size],heap[size],hlen,h[size],dsts[maxlen];
int cmp(const int &i,const int &j){ return h[i]>h[j]; }

void calch(int pos)
{
    int i,j,k;  h[pos]=d[pos];
    for(i=0;i<9;i++) if(s[i]!=dsts[i]) h[pos]++;
}

void output(int pos,int num)
{
    if(pos==1) cout<<"Total number of steps = "<<num<<endl;
        else output(mk[open[pos]],num+1);
    decode(s,9,open[pos]); print(s,9);
}
```

```
void solve(int src,int dst)
{
    int i,j,k,x,l,ps;
    if(src==dst){ cout<<"SRC DST is the same!"<<endl; return; }
    tail=1; open[1]=src; mk[src]=1; hlen=1; heap[0]=1; d[1]=0;
    decode(s,9,src); decode(dsts,9,dst); calch(1);
    while(hlen>0){
        cur=heap[0]; std::pop_heap(heap,heap+(hlen--),cmp);
        decode(s,9,open[cur]); for(ps=0; s[ps]; ps++);
        for(k=1; k<=link[ps][0]; k++) {
            std::swap(s[ps],s[link[ps][k]]); encode(s,9,x);
            if(!mk[x]){
                mk[x]=cur; open[++tail]=x; d[tail]=d[cur]+1; calch(tail);
                heap[hlen++]=tail; std::push_heap(heap,heap+hlen,cmp);
                if(x==dst) { output(tail,0); return; }
            }
            std::swap(s[ps],s[link[ps][k]]);
        }
    }
    cout<<"No solution!"<<endl;
}
```

Double Breadth First Search

```
int step,di[size];

void out1(int pos)
{
    if(pos>2) out1(mk[open[pos]]); step++;
    decode(s,9,open[pos]); print(s,9);
}

void out2(int pos)
{
    decode(s,9,open[pos]); print(s,9);
    if(pos>2) out2(mk[open[pos]]); step++;
}

void solve(int src,int dst)
{
    int i,j,k,x,l,ps;
    if(src==dst){ cout<<"SRC DST is the same!"<<endl; return; }
    open[1]=src; mk[src]=1; di[src]=1; cur=0;
    open[2]=dst; mk[dst]=2; di[dst]=2; tail=2;
    while(++cur<=tail){
        decode(s,9,open[cur]); for(ps=0; s[ps]; ps++);
        for(k=1; k<=link[ps][0]; k++) {
            std::swap(s[ps],s[link[ps][k]]); encode(s,9,x);
            if(!mk[x]){ mk[x]=cur; open[++tail]=x; di[x]=di[open[cur]]; }
                else if(di[x]!=di[open[cur]]){
                    step=0;
                    if(di[x]==1) { out1(mk[x]); out2(cur); }
                        else     { out1(cur);   out2(mk[x]); }
                    cout<<"Total number of steps = "<<step<<endl;
                    return;
                }
            std::swap(s[ps],s[link[ps][k]]);
        }
    }
    cout<<"No solution!"<<endl;
}
```

6.13 Extended Honai Tower

```
int P[ML][ML],D[ML][ML],T[ML][ML];

void init()
{
    int i,j,x,k,l;
    for(P[0][0]=l=1;l<ML;l++) {
        P[0][l] = P[l][0] = 1;
        for(i=1;i<l;i++) P[i][l-i] = P[i-1][l-i]+P[i][l-i-1];
    }
    for(i=0;i<ML;i++) for (k=j=0; j<ML-i && k!=ML;j++)
        for(x=0;x<P[i][j];x++) { if (k==ML) break; D[i][k++] = 1<<j; }
    for(i=0;i<ML;i++) T[i][0] = 0;
    for(j=1;j<ML;j++) for(i=0;i<20;i++) T[i][j] = T[i][j-1]+D[i][j-1];
}

int main()
{
    init();
    for (int a,b,casec=1 ; cin>>a>>b && (a||b) ;casec++)
        cout << "Case " << casec << ": " << T[b-3][a] << endl;
    return 0;
}
```

6.14 High Precision Square Root

```
int x[maxlen],y[maxlen],z[maxlen],bck[maxlen],lx,ly,lz;

int IsSmaller()  // is z<=y ?
{int i=ly; while(i>1 && z[i]==y[i]) i--; return (z[i]<=y[i]); }

void Solve() // y^2=x
{
    int i,j,k;
    lx=(ly+1)/2;  ly=lx*2;
    memset(x,0,sizeof(x)); memset(z,0,sizeof(z));
    for(i=lx;i>0;i--){
        for(j=1; j<10; x[i]=j++){
            memcpy(bck,z,sizeof(z));
            z[2*i-1]++; for(k=i;k<=lx;k++)
            { z[i-1+k]+=2*x[k]; z[i+k]+=z[i-1+k]/10; z[i-1+k]%=10; }
            for(k=lx+i;k<=ly;k++){ z[k+1]+=z[k]/10;  z[k]%=10; }
            if(!IsSmaller()) break;
        };
        if(j<10) memcpy(z,bck,sizeof(bck));
    };
    for(i=lx;i>0;i--) cout<<x[i]; cout<<endl;
}

int main()
{
    char ch,s[maxlen]; int i,j;
    memset(y,0,sizeof(y));
    cin>>s; ly=strlen(s);
    for(i=0;i<ly;i++) y[i+1]=s[ly-1-i]-'0';
    Solve();
    return 0;
}
```

6.15 Largest Empty Rectangle

$O(N^2)$

```
int n,wx,wy,x[maxn],y[maxn],id[maxn];
int xx[maxn],yy[maxn],ans;

bool cmp(const int&i,const int&j)
{
    return x[i]<x[j];
}

void calc(int i,int px,int py)
{
    int ret,j,low=0,high=wy;
    for(;i<n;i++) if( x[i]>px )
    {
        j=(high-low)*(x[i]-px); if(j>ans) ans=j;
        if( y[i]<py && y[i]>=low ) low = y[i];
        if( y[i]>=py && y[i]<=high ) high = y[i];
    }
}

int main()
{
    int i,j,k;
    cin>>wx>>wy>>n; for(i=0;i<n;i++) cin>>x[i]>>y[i];
    x[n]=y[n]=0; n++; x[n]=wx; y[n]=wy; n++;
    for(i=0;i<n;i++) id[i]=i; std::sort(id,id+n,cmp);
    for(i=0;i<n;i++) { xx[i]=x[id[i]]; yy[i]=y[id[i]]; }
    for(i=0;i<n;i++) { x[i]=xx[i]; y[i]=yy[i];}
    std::sort(yy,yy+n); k=std::unique(yy,yy+n)-yy;
    ans=0;
    for(i=0;i<n;i++) calc(i,x[i],y[i]);
    for(j=0;j<k;j++) calc(0,0,yy[j]);
    cout<<ans<<endl;
    return 0;
}
```

$O(D^2)$

```
int x[maxn],y[maxn],xlist[maxn],ylist[maxn],nx,ny,ans,n,wx,wy;
char g[maxd][maxd]; int u[maxd],d[maxd],l[maxd];

int main()
{
    int i,j,px,py,up,down,tmp; ans=0;
    cin>>wx>>wy>>n; for(i=0;i<n;i++) cin>>x[i]>>y[i];
    nx=ny=n;  for(i=0;i<n;i++) { xlist[i]=x[i]; ylist[i]=y[i]; }
    xlist[nx++]=ylist[ny++]=0; xlist[nx++]=wx; ylist[ny++]=wy;
    std::sort(xlist,xlist+nx); nx=std::unique(xlist,xlist+nx)-xlist;
    std::sort(ylist,ylist+ny); ny=std::unique(ylist,ylist+ny)-ylist;

    for(i=0; i<nx; i++) memset(g, 0, n);

    for(i=0; i<n; i++)
    {
        px = std::lower_bound(xlist,xlist+nx,x[i]) - xlist;
        py = std::lower_bound(ylist,ylist+ny,y[i]) - ylist;
        g[px][py]=1;
    }
```

```
    for(j=0; j<ny-1; j++)
    {
        tmp = wx * ( ylist[j+1] - ylist[j] );
        if( tmp > ans ) ans = tmp;
    }
    for(i=1; i<nx; i++)
    {
        down=0; up=ny-1;
        for(j=0; j<ny; j++) if( i==1 || *(*(g+i-1)+j) )
            { l[j]=i-1; d[j]=0; down=j; } else
                if( down > d[j] ) d[j] = down;
        for(j=ny-1; j>=0; j--)
        {
            if( i==1 || *(*(g+i-1)+j) ) { u[j]=ny-1; up=j; } else
                    if( up < u[j] ) u[j] = up;
            tmp = ( xlist[i] - xlist[l[j]] ) * ( ylist[u[j]]-ylist[d[j]] );
            if(tmp>ans) ans=tmp;
        }
    }
    cout<<ans<<endl;
    return 0;
}
```

$O(N^2)$

```
int n,wx,wy,id[maxn],x[maxn],y[maxn],ans,xx[maxn],yy[maxn];

bool xcmp(const int&i,const int &j) { return x[i]<x[j]; }
bool ycmp(const int&i,const int &j) { return y[i]<y[j]; }

int main()
{
    int i,j,k,l,tmp,low,high,last;
    cin>>wx>>wy>>n; for(i=0;i<n;i++) cin>>x[i]>>y[i];
    x[n]=y[n]=0; n++; x[n]=wx; y[n]=wy; n++;
    for(i=0;i<n;i++) id[i]=i;
    std::sort(id,id+n,xcmp);
    for(i=0;i<n;i++) { xx[i]=x[id[i]]; yy[i]=y[id[i]]; }
    for(i=0;i<n;i++) { x[i]=xx[i]; y[i]=yy[i]; }
    std::sort(id,id+n,ycmp);
    for(i=0;i<n;i++)
    {
        l=0; last=0;
        for(j=0;j<n;j++) if( x[id[j]]<x[i] && y[id[j]]>last)
        {
            if( y[id[j]]-last > l ) l=y[id[j]]-last;
            last=y[id[j]];
        }
        if( wy-last>l ) l=wy-last;
        if( l*x[i] > ans ) ans =l*x[i];
        low=0; high=wy; for(j=i+1;j<n;j++)
        {
            tmp = (high-low)*(x[j]-x[i]) ;
            if( tmp> ans ) ans=tmp;
            if( y[j]>=y[i] && y[j]<high ) high = y[j];
            if( y[j]<=y[i] && y[j]>low  ) low  = y[j];
        }
    }
    cout<<ans<<endl;
    return 0;
}
```

6.16 Last Non-Zero Digit of N!

Smart Edition

```
const int ff[10] = {1, 1, 2, 6, 4, 4, 4, 8, 4, 6 };

int fact(int n)
{
    int i,x;
    if(n<5) return ff[n];
    x = ( ff[n%10] * 6 ) %10;
    for( i=1; i<=(n/5)%4; i++)
        if( x==6 || x==2 ) x=(x+10)/2; else x/=2;
    return ( fact(n/5) * x ) % 10;
}
```

High Precision Edition

```
int a[10] = {6,1,2,6,4,4,4,8,4,6};
int b[ 4] = {1,8,4,2};

void divide(char s[],int &len)
{
    int i;
    char temp[200];
    for(i=0; i<len; i++) temp[i] = s[i]*2; temp[len] = 0;
    for(i=0; i<len; i++) if ( temp[i]>9 ){ temp[i]-=10; temp[i+1]++; }
    for(i=0; i<len; i++) s[i] = temp[i+1];
    if( temp[len]==0 ) len--;
}

int fact(char s[])
{
    int resulent=1,power=0,len=strlen(s),i;
    char temp;
    if (len==1&&s[0]=='0') return 1;
    for(i=0;i<len;i++) s[i]-='0';
    for(i=0;i<len/2;i++){ temp=s[i]; s[i]=s[len-1-i]; s[len-1-i]=temp; }
    while(len){
        resulent=resulent*a[s[0]%10]%10;
        divide(s,len);
        power+=(s[1]*10+s[0])%4;
    }
    resulent=resulent*b[power%4]%10;
    return resulent;
}
```

6.17 Least Common Ancestor

```
int n,h,root;   // maxh-1 = h = floor( log ( 2, n-1 ) )
int pnt[maxn][maxh],son[maxn],next[maxn],depth[maxn];
int stack[maxn], mylog[maxn];

int GetParent(int x,int len)
{
    while( len>0 ) {
        x = pnt[x][ mylog[len] ];
        len -= ( 1<<mylog[len] );
    }
    return x;
}
```

```
int LCA(int x,int y)  // O( log N )
{
    int nx,ny,px,py,low,mid,high;
    low=0; high = depth[x]<depth[y] ? depth[x] : depth[y];
    px = GetParent(x, depth[x]-high) ;
    py = GetParent(y, depth[y]-high) ;
    if( px == py ) return px;
    while(high-low>1)
    {
        mid = mylog[high-low-1];
        nx = pnt[px][mid];
        ny = pnt[py][mid];
        mid = high - (1<<mid);
        if( nx == ny ) low = mid; else { high = mid; px = nx; py = ny; }
    }
    return pnt[px][mylog[high-low]];
}

int LCA_2(int x,int y) // O( log^2 N )
{
    int low,mid,high;
    low = 0;  mid = high = depth[x]<depth[y] ? depth[x] : depth[y];
    if( GetParent(x, depth[x]-mid) != GetParent(y, depth[y]-mid) )
    while(low+1<high)
    {
        mid = (low + high) / 2;
        if( GetParent(x, depth[x]-mid) != GetParent(y, depth[y]-mid) )
            high = mid; else low = mid;
    } else low = high;
    return GetParent(x, depth[x]-low) ;
}

void dfs(int d,int cur)
{
    int i,j;  stack[d] = cur;  depth[cur] = d;
    for(j=1,i=2; i<=d; j++,i*=2 ) pnt[cur][j]=stack[d-i];
    for(j=son[cur]; j; j=next[j]) dfs( d+1, j );
}

void main()
{
    int i,j,k,l;
    for(i=0,j=1; j<maxn; i++)
    {
        k = j*2; if( k>maxn ) k = maxn;
        while( j<k ) mylog[j++] = i;
    }
    cin>>n;
    for(i=1; i<=n; i++) {
        son[i] = next[i] = 0;
        for(j=0; j<=h; j++) pnt[i][j] = 0;
    }
    for(i=1; i<n; i++) {
        cin >> j >> k;  pnt[j][0] = k;
        next[j]=son[k]; son[k]=j;
    }
    for(i=1; i<=n; i++) if( pnt[i][0]==0 ) { root=i; break; };

    dfs( 0, root );   // Preprocess Parent Array

    for(cin>>k; k; k--) { cin >> i >> j; cout << LCA(i,j) <<endl; }
}
```

6.18 Longest Common Substring

$O(N \log N)$, using Suffix Sort with LCP information

```
int LCS(char *s1, int l1, char *s2, int l2, int &i1, int &i2)
{
    strcpy(s, s1);          s[l1]='$';
    strcpy(s+l1+1, s2);     n=l1+l2+1;
    SuffixSort();           GetHeight();     // s[l1]=0;
    int i, j, l=0; i1 = i2 = 0;
    for(i=1; i<n; i++)
    {
        if( height[i]>=l && id[i-1]<l1 && id[i]>l1 )
            { l = height[i]; i1 = id[i-1];  i2 = id[i]-l1-1; }
        if( height[i]>=l && id[i]<l1 && id[i-1]>l1 )
            { l = height[i]; i1 = id[i];  i2 = id[i-1]-l1-1; }
    }
    return l;
}
```

$O(N^2)$, using KMP

```
int LCS(char *s1, int l1, char *s2, int l2, int &ansi, int &ansj)
{
    int i,j,k,l,ans=0; ansi=0; ansj=0;
    for(i=0; i<l1-ans; i++)
    {
        makefail( s1+i, l1-i );
        kmp( s2, l2, s1+i, l1-i, 0, l, j);
        if(l>ans) { ans=l; ansi=i; ansj=j; }
    }
    return ans;
}
```

Example Part

```
char s1[maxlen],s2[maxlen]; int l1,l2;

int main()
{
    ifstream cin("input.txt");
    cin>>s1>>s2; l1=strlen(s1); l2=strlen(s2);
    int i1,i2,i,l = LCS(s1,l1,s2,l2,i1,i2);
    cout<<l<<" "<<i1<<" "<<i2<<endl;
    for(i=0;i<l;i++) cout<<s1[i1+i]; cout<<endl;
    for(i=0;i<l;i++) cout<<s2[i2+i]; cout<<endl;
    return 0;
}
```

M^{th} Longest Common Substring

```
#define h next // h[i] = Longest Common Substring of s1+0 and s2+i
int mk[maxn]; // already found a common substring = s2[i..mk[i])
struct CAnswer{ int pos,len; } ans[maxn];

bool newcmp(const CAnswer &a, const CAnswer &b)
{
    if(a.len != b.len ) return a.len>b.len;
    return a.pos<b.pos;
}
```

```
void LCS(char *s1, int l1, char *s2, int l2,int m)
{
    strcpy(s, s1);          s[l1]='$';
    strcpy(s+l1+1, s2);     n=l1+l2+1;
    SuffixSort();           GetHeight();     // s[l1]=0;
    int i, j, k, p, u, v;
    // computing longest common prefix between s1+0 and s2+i
    memset(h,0,sizeof(h));
    for(i=0; i<n; i++) if( i<n-1 && id[i]<l1 && id[i+1]>l1 ) {
        k=maxlen;
        for(j=i+1; j<n; j++)
            { if(id[j]<l1) break; if(height[j]<k) k=height[j]; h[j]=k; }
        i=j-1;
    }
    for(i=n-1; i>0; i--) if( id[i]<l1 && id[i-1]>l1) {
        k=maxlen;
        for(j=i-1; j>=0; j--) {
            if(id[j]<l1) break; if(height[j+1]<k) k=height[j+1];
            if(k>h[j]) h[j]=k;
        }
        i=j+1;
    }
    num=0; // Collect Non-Position-Covering Answer
    for(i=0;i<n;i++)
        if( h[rank[i]]!=0 && (i==0 || h[rank[i-1]]<=h[rank[i]]) )
            { k=rank[i]; ans[num].pos=id[k]; ans[num].len=h[k]; num++;}
    std::sort(ans,ans+num,newcmp);
    memset(mk,0,sizeof(mk));
    for(i=j=0; i<num && j<m; i++) {
        k=rank[ans[i].pos]; // Check Non-Substring-Covering
        if( mk[k]>=h[k] ) continue;
        int ok=1;
        for(u=maxlen, p=k+1; p<n; p++) {
            if( height[p]<u ) u=height[p];
            if(u<h[k]) break;
            if(mk[p]>=h[k]) { ok=0; break; }
        }
        if(!ok) continue;
        for(u=maxlen,p=k-1; p>=0; p--) {
            if( height[p+1]<u ) u=height[p+1];
            if(u<h[k]) break;
            if(mk[p]>=h[k]) { ok=0; break; }
        }
        if(!ok) continue;
        j++; // Check Passed, Set Already Found Substring
        for(v=0; v<h[k]; v++)
            if( mk[rank[id[k]+v]] < h[k]-v ) mk[rank[id[k]+v]] = h[k]-v;
        // LENGTH h[rank[ans[i].pos]] POSITION ans[i].pos-l1-1
        char ch = s[ans[i].pos + h[rank[ans[i].pos]]];
        s[ans[i].pos + h[rank[ans[i].pos]]] = 0;
        cout << s+ans[i].pos << endl;
        s[ans[i].pos + h[rank[ans[i].pos]]] = ch;
    }
}
```

6.19 Longest Non Descending Sub Sequence

```
int LNDSS(int a[],int n) // Longest Non-descending Sub Sequence
{
    int i,j,k,*b=new int[n+1],ans=0;
    b[ans]=-0x3f3f3f3f;
    for(i=0;i<n;i++){   // lower_bound for Asending Sub Squence
        j=std::upper_bound(b,b+ans+1,a[i])-b;
        if(j>ans) b[++ans]=a[i];else if(a[i]<b[j]) b[j]=a[i];
    }
    delete b; return ans;
}
```

6.20 Join and Disjoin

Note: UnionFind.h contains a Union-Find Set (Section 1.10 on Page 11)

```
#include<unionfind.h>

int Gather(int x,int y)
{
    if(!x && !y) return 0;
    if(!x) return find(y);
    if(!y) return find(x);
    Merge(x,y);
    return find(x);
}

void Join(int x,int y)
{
    int a=Gather(x,y); // x,y nerver be zero
    int b=Gather(vs[x],vs[y]);
    vs[a]=b; vs[b]=a;
}

void Disjoin(int x,int y)
{
    int a=Gather(x,vs[y]);
    int b=Gather(y,vs[x]);
    vs[a]=b; vs[b]=a;
}
```

6.21 Magic Square

```
#define maxn 1000
int a[maxn][maxn],n;

void build(int n,int a[][maxn]) // No solutions when n=2!
{
    int i,j,k,n2=n*n,m=n/2,m2=m*m;
    for(i=0;i<n;i++) for(j=0;j<n;j++) a[i][j]=0;
    if (n==2) return;              // No solutions
    if (n%2==1)
        for(i=0,j=n/2,k=1;k<=n2;k++) {
            a[i][j] = k;
            if(!a[(i+n-1)%n][(j+1)%n])
                { i=(i+n-1)%n;  j=(j+1)%n; } else i=(i+1)%n;
        }
    else if(n%4==0)
        for(k=0,i=0;i<n;i++) for(j=0;j<n;j++) {
            a[i][j] = ++k;
            if (i%4==j%4||i%4+j%4==3) a[i][j] = n2+1-a[i][j];
        }
    else if(n%4==2)
        for(i=0,j=m/2,k=0;k<m2;k++) {
            if((i<=m/2 && !(i==m/2&&j==m/2))||(i==m/2+1&&j==m/2)){ // L
                a[i*2  ][j*2+1]=k*4+1;  a[i*2+1][j*2]=k*4+2;
                a[i*2+1][j*2+1]=k*4+3;  a[i*2  ][j*2]=k*4+4;
            } else if (i>m/2+1) { // X
                a[i*2  ][j*2]=k*4+1;    a[i*2+1][j*2+1]=k*4+2;
                a[i*2+1][j*2]=k*4+3;    a[i*2  ][j*2+1]=k*4+4;
            } else { // U
                a[i*2  ][j*2  ]=k*4+1; a[i*2+1][j*2]=k*4+2;
                a[i*2+1][j*2+1]=k*4+3; a[i*2][j*2+1]=k*4+4;
            }
            if (!a[(i+m-1)%m*2][(j+1)%m*2]) i=(i+m-1)%m, j=(j+1)%m;
                else i=(i+1)%m;
        }
}

int main()
{
    while (cin>>n) {
        build(n,a); cout<<"Order "<<n<<":"<<endl;
        for (int j,i=0;i<n;i++)
            { for (j=0;j<n;j++) cout<<a[i][j]<<' '; cout<<endl;}
    }
    return 0;
}
```

6.22 Optimal Binary Search Tree

```
int n,a[maxn],s[maxn][maxn],h[maxn][maxn],kk[maxn][maxn];

int solve()
{
    int i,j,k,l;  memset(h,0,sizeof(h));
    for(i=1;i<=n;i++) { s[i][i]=a[i]; h[i][i]=0; kk[i][i]=i;
        for(j=i+1;j<=n;j++) s[i][j]=s[i][j-1]+a[j];
    }
    for(l=1; l<n; l++) {
        for(i=1;i<n;i++) { j=i+l; h[i][j]=0x0fffffff;
            for(k=kk[i][j-1]; k<=kk[i+1][j]; k++)
                if( h[i][k-1]+h[k+1][j]-a[k]+s[i][j] < h[i][j]) {
                    h[i][j] = h[i][k-1]+h[k+1][j]+s[i][j]-a[k];
                    kk[i][j] = k;
                }
        }
    }
    return h[1][n];
}
```

6.23 Pack Rectangles — Cut Rectangles

```
struct rect{int x1,y1,x2,y2;} r[maxm];
int mk[maxm];

int intersect(rect a,const rect &b,rect out[4]) // b cut a
{
    if( b.x2<=a.x1 || b.x1>=a.x2 || b.y2<=a.y1 || b.y1>=a.y2) return 0;
    if( b.x1<=a.x1 && b.x2>=a.x2 && b.y1<=a.y1 && b.y2>=a.y2) return -1;
    rect t; int nout=0;
    if( b.x1>a.x1 ){ t=a; t.x2=b.x1; a.x1=b.x1; out[nout++]=t; }
    if( b.x2<a.x2 ){ t=a; t.x1=b.x2; a.x2=b.x2; out[nout++]=t; }
    if( b.y1>a.y1 ){ t=a; t.y2=b.y1; a.y1=b.y1; out[nout++]=t; }
    if( b.y2<a.y2 ){ t=a; t.y1=b.y2; a.y2=b.y2; out[nout++]=t; }
    return nout;
}

int main()
{
    rect curr,t[4]; int i,j,k,nn,nr,ans,rr,n;
    cin>>n; rr=0;
    for(i=0;i<n;i++){
        cin >> curr.x1 >> curr.y1 >> curr.x2 >> curr.y2;
        nr=rr; mk[rr]=1; r[rr++]=curr;
        for(j=0; j<nr; j++) {
            mk[j]=1; nn=intersect(r[j],curr,t); if(!nn) continue;
             if( nn<0 ) mk[j] = 0; else { r[j] = t[--nn];
                while(nn) { mk[rr] = 1; r[rr++] = t[--nn]; }
             }
        }
        for(k=j=0; j<rr; j++) if(mk[j]) r[k++]=r[j]; rr=k;
    }
    for(ans=i=0; i<rr; i++) ans+=(r[i].x2-r[i].x1)*(r[i].y2-r[i].y1);
    cout<<ans<<endl;
    return 0;
}
```

6.24 Pack Rectangles — O(N^2)

```
int x1[maxn],y1[maxn],x2[maxn],y2[maxn];
int ylist[maxn*2],id[maxn],n,ny;

bool cmp(const int&i,const int&j){ return x1[i]<x1[j]; }

int GetAreaUnion()
{
    int i,j,k,rx,l,ans=0;
    for(ny=0,i=0; i<n; i++) { ylist[ny++]=y1[i]; ylist[ny++]=y2[i]; }
    std::sort(ylist, ylist+ny); ny=std::unique(ylist, ylist+ny)-ylist;
    for(i=0; i<n; i++) id[i]=i; std::sort(id,id+n,cmp);
    for(j=0; j<ny-1; j++){
        rx = -0x3f3f3f; l=0;
        for(k=0;k<n;k++){ i = id[k];
            if( y1[i]<=ylist[j] && y2[i]>=ylist[j+1] && x2[i]>rx ) {
                if( x1[i]>rx ) l+=x2[i]-x1[i]; else l+=x2[i]-rx;
                rx = x2[i];
            }
        }
        ans += l * (ylist[j+1]-ylist[j]);
    }
    return ans;
}
```

6.25 Parliament

Given $n > 0$, find distinct positive numbers $a_1 + a_2 + ... + a_k = n$ that maximize $a_1 \cdot a_2 \cdot ... \cdot a_k$.

```
int main()
{
    int n,k,p,i,caseno;
    for(cin>>caseno; caseno--; ){ cin>>n;
        for(p=n, k=2; p>=k; k++) p-=k; k--;
        if(p<=1){ for(i=2;i<k;i++) cout<<i<<" "; cout<<k+p<<endl;  } else
        if(p==k){ for(i=3;i<=k;i++) cout<<i<<" "; cout<<k+2<<endl; } else
            {   for(i=2+(p==k-1);i<=k;i++) if(i!=k-p+1) cout<<i<<" ";
                cout<<k+1<<endl; }
        if(caseno) cout<<endl;
    }
    return 0;
}
```

6.26 π Generator

```
int a=10000,b,c=2800,d,e,f[2801],g;

void GenPI() {
    for(;b-c;) f[b++]=a/5;
    for(;d=0,g=c*2; c-=14,printf("%.4d",e+d/a),e=d%a)
        for(b=c; d+=f[b]*a,f[b]=d%--g,d/=g--,--b; d*=b);
}
```

6.27 Plant Trees — Iteration

```
const int maxlen = 50005;
const int maxn   = 50000;

int n,st[maxlen],a[maxn],b[maxn],c[maxn],up;

int main(){
    int i,more;
    while(cin>>n){
        for(i=0; i<n; i++){
            cin>>a[i]>>b[i]>>c[i];
            if(++b[i]>up) up=b[i];
        }
        memset(st, 0, sizeof(st));
        for(more=1; more; ){
            more = 0;
            for(i=0; i<n; i++) if (st[a[i]]+c[i]>st[b[i]])
                { st[b[i]]=st[a[i]]+c[i];   more=1; }
            for(i=1; i<=up; i++) {
                if(st[i-1]+1<st[i]){ st[i-1]=st[i]-1; more=1; }
                if(st[i-1]  >st[i]){ st[i  ]=st[i-1]; more=1; }
            }
            for(i=up; i>0; i--){
                if(st[i]-1>st[i-1]){ st[i-1]=st[i]-1; more=1; }
                if(st[i-1]>st[i  ]){ st[i  ]=st[i-1]; more=1; }
            }
        }
        cout<<st[up] << endl;
    }
    return 0;
}
```

6.28 Plant Trees — Segment Tree

```
#define maxn 50000
#define maxup 50006

int nspan,span[maxn][3],up,tree[maxup];
int iteam[maxup],next[maxup],num;

int funt_comp(const void *a, const void *b)
{ return ((const int *)b)[0]-((const int *)a)[0]; }

void add(int r)
{ for(;r<=up; r+=r&(r^(r-1))) ++tree[r]; }

int sum(int r)
{ int ans = 0; for(;r>0; r-=r&(r^(r-1))) ans+=tree[r]; return ans; }

void go()
{
    int j,k,i,ans=0; up=0;
    for(i=0; i<nspan; ++i){
        scanf("%d %d %d", &span[i][0], &span[i][1], &span[i][2]);
        ++span[i][0]; ++span[i][1];
        if (span[i][1]>up) up=span[i][1];
    }
    qsort(span, nspan, sizeof(int)*3, funt_comp);
    for(j=0;j<=up;j++) next[j]=j+1;
    next[up]=0; memset(tree,0,(up+1)*sizeof(int));
    for(i=0; i<nspan; i++){
```

```
        k=sum(span[i][1])-sum(span[i][0]-1);
        if(k>=span[i][2]) continue; else k=span[i][2]-k;
        j=span[i][0]; if(next[j-1]!=j) j=next[j];
        while(k--){
            next[span[i][0]]=next[span[i][0]-1]=next[j];
            ans++; add(j); j=next[j];
        }
    }
    printf("%d\n", ans);
}

int main() {
    while( 1==scanf("%d", &nspan) ) go();
    return 0;
}
```

6.29 Range Maximum Query

O($N \log N$) Preprocess, O(1) Query

```
int n,L,q, a[maxn],h[maxn][maxL]; // maxL = sqrt{N} + 3

void PreProcess()
{
    int i,j,l;
    for(i=0;i<n;i++) h[i][0]=a[i];
    for(j=1,l=1; l*2<=n; j++,l*=2) for(i=0; i<=n-l*2; i++)
        h[i][j] = (h[i][j-1]>h[i+l][j-1]) ? h[i][j-1] : h[i+l][j-1];
}

int Query(int be,int ed) // return max{a[op..ed]}
{
    int j=0,l=1; while( 2*l<=ed-be+1 ) { j++; l*=2; }
    return (h[be][j]>h[ed+1-l][j]) ? h[be][j] : h[ed+1-l][j];
}
```

O(N) Preprocess, O($\sqrt{N}$) Query

```
int a[maxn],b[maxL],n,L,q;

void PreProcess()
{
    int i,j,up,k; L = (int)sqrt(n);
    for(i=k=0; i<n; k++ ) {
        up=i+L; if( up>n ) up = n;
        for(j=i+1; j<up; j++) if( a[j]>a[i] ) i=j;
        b[k]=i; i=up;
    }
}

int Query(int be,int ed) // return max{a[op..ed]}
{
    int i,up,u,v,k;
    u = be / L; v = ed / L; k = be;
    if( u<v ) {
        k=be; up=(u+1)*L;
        for(i=u+1; i<v ; i++) if( a[b[i]]>a[k] ) k = b[i];
        for(i=be; i<up; i++)      if( a[i]>a[k] ) k = i;
        for(i=v*L; i<=ed; i++)    if( a[i]>a[k] ) k = i;
    } else for(i=be; i<=ed; i++) if( a[i]>a[k] ) k = i;
    return k;
}
```

6.30 Travelling Salesman Problem

```
int n,x[maxn],y[maxn],id[maxn];
double g[maxn][maxn];

double dis(int x1,int y1,int x2,int y2)
{ return sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2)); }

double solve()
{
    int i,j,k,l,loop;
    double cur,ans=1e30;

    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            g[i][j]=dis(x[i],y[i],x[j],y[j]);

    for(k=0;k<n;k++)
    {
        for(l=0;l<50;l++)
        {
            for(i=0;i<n;i++) id[i]=i;
            std::swap(id[0],id[k]);
            std::random_shuffle(id+1,id+n);

            loop=1;
            while(loop){
                loop=0;
                for(i=1;i<n;i++) for(j=i+1;j<n-1;j++)
                    if( g[id[i-1]][id[i]] + g[id[j]][id[j+1]]
                      > g[id[i-1]][id[j]] + g[id[i]][id[j+1]] + 1e-8 )
                    {
                        loop=1;
                        std::reverse(id+i,id+j+1);
                    }
            };

            for(cur=0,i=0; i<n-1; i++)
                cur+=g[id[i]][id[i+1]];

            if(cur<ans) ans=cur;
        }
    }
    return ans;
}
```

6.31 Tree Heights

```
#define maxn 5003*4

int n,num,nbs[maxn],next[maxn*2],h[maxn];
int out1[maxn],out2[maxn],son1[maxn],in[maxn],value[maxn];
int son[maxn],pnt[maxn],bro[maxn],weight[maxn],id[maxn];

void solve()
{
    int i,j,k(1),l;
    id[1]=1; weight[1]=in[1]=0;
    for(i=1;i<=k;i++) for(j=son[id[i]];j;j=bro[j]) id[++k]=j;
    for(i=2;i<=n;i++) weight[i]=1;
    for(k=n;k>0;k--){ i=id[k];
        for(j=son[i];j;j=bro[j])
            if(out1[j]+weight[j]>=out1[i])
            { out2[i]=out1[i]; out1[i]=out1[j]+weight[j]; son1[i]=j; }
            else if(out1[j]+weight[j]>out2[i]) out2[i]=out1[j]+weight[j];
    }
    for(k=2;k<=n;k++){ i=id[k]; in[i]=0;
        if(in[pnt[i]]>in[i]) in[i]=in[pnt[i]];
        if(i==son1[pnt[i]])  l=out2[pnt[i]]; else l=out1[pnt[i]];
        if(l>in[i]) in[i]=l; in[i]+=weight[i];
    }
}

void dfs(int node)
{
    for(int j,i=nbs[node];i;i=next[i]){
        j=value[i]; if(j==pnt[node]) continue;
        pnt[j]=node; bro[j]=son[node]; son[node]=j; dfs(j);
    }
}

void out()
{
    int maxh=-1, minh=n+1,i;
    for(i=1;i<=n;i++) {
        if(in[i]<out1[i]) h[i]=out1[i]; else h[i]=in[i];
        if(h[i]>maxh) maxh=h[i];  if(h[i]<minh) minh=h[i];
    }
    cout<<"Best Roots  :";
    for(i=1;i<=n;i++) if(h[i]==minh) cout<<" "<<i;
    cout<<endl <<"Worst Roots :";
    for(i=1;i<=n;i++) if(h[i]==maxh) cout<<" "<<i;
    cout<<endl;
}

int main()
{
    int i,j,k,l;
    while(cin>>n){
        for(i=1;i<=n;i++)
            out1[i]=out2[i]=son1[i]=son[i]=bro[i]=pnt[i]=next[i]=nbs[i]=0;
        for(num=1,i=1;i<=n;i++){ cin>>l; for(k=0;k<l;k++)
            { cin>>j; value[num]=j; next[num]=nbs[i]; nbs[i]=num++;} }
        dfs(1); solve(); out();
    }
    return 0;
}
```

6.32 Minimum Cyclic Presentation

```
int MinimumCyclicPresentation(char *s, int n)
{
    int i,j,x,y,u,v;
    for(x=0,y=1; y<n; y++) if( s[y]<=s[x] )
    {
        i=u=x;   j=v=y;
        while( s[i]==s[j] )
        {
            ++u; if( ++i == n ) i=0;
            ++v; if( ++j == n ) j=0;
             if( i==x ) break;
        }
        if( s[i]<=s[j] ) y = v; else
            { x = y; if( u>y ) y = u; }
    }
    return x;
}
```

6.33 Maximum Clique

```
int list[maxn][maxn],g[maxn][maxn],s[maxn],degree[maxn],behide[maxn];
int found,n,curmax,curobj;

void sortdegree()
{
    for(int j,k,l,i=1;i<=n;i++) {
        for(k=i, j=i+1;j<=n;j++) if( degree[j]<degree[k]) k=j;
        if(k!=i){
            std::swap(degree[i],degree[k]);
            for(l=1;l<=n;l++) std::swap(g[i][l],g[k][l]);
            for(l=1;l<=n;l++) std::swap(g[l][i],g[l][k]);
        }
    }
}

void dfs(int d)
{
    if( d-1>curmax ) { found=1;return;};
    int i,j;
    for(i=1;i<list[d-1][0]-curmax+d;i++)
        if(!found && d+behide[list[d-1][i]+1]>curmax &&
           (list[d-1][0]==i || d+behide[list[d-1][i+1]]>curmax )) {
            for(j=i+1,list[d][0]=0; j<=list[d-1][0]; j++)
                if( g[list[d-1][j]][list[d-1][i]] )
                    list[d][++list[d][0]]=list[d-1][j];
            if( list[d][0]==0 || d + behide[list[d][1]] >curmax) dfs(d+1);
        }
}

void solve()
{
    sortdegree(); behide[n+1]=0; behide[n]=1;
    for(int j,i=n-1;i>0;i--) {
        curmax=behide[i+1]; found=list[1][0]=0;
        for(j=i+1;j<=n;j++) if(g[j][i]) list[1][++list[1][0]]=j;
        dfs(2); behide[i]=curmax+found;
    } cout<<behide[1]<<endl;
}

int main()
{
    int i,j;
    while(cin>>n,n) {
        for(i=1;i<=n;i++) for(j=1,degree[i]=0; j<=n; j++) {
            cin >> g[i][j];
            degree[i]+=(g[i][j]!=0);
        } solve();
    }
    return 0;
}
```

6.34 Maximal Non-Forbidden Submatrix

```
#define forbidden 1

int wx,wy,g[maxn][maxn],h[maxn],r[maxn],l[maxn];

int solve()
{
    int i,j,k,ans,left,right;
    ans=0; memset(h,0,sizeof(h));
    for(i=0;i<wx;i++) {
        for(j=0;j<wy;j++) if(g[i][j]!=forbidden) h[j]++; else h[j]=0;
        for(j=0;j<wy;j++) if(h[j]) {
            if(j==0 || h[j-1]==0) left=j;
            if(i==0 || g[i-1][j]==forbidden) l[j]=left;
            if(left>l[j]) l[j]=left;
        }
        for(j=wy-1;j>=0;j--) if(h[j]) {
            if(j==wy-1 || h[j+1]==0 ) right=j;
            if(i==0 || g[i-1][j]==forbidden) r[j]=right;
            if(right<r[j]) r[j]=right;
        }
        for(j=0;j<wy;j++)
            if((r[j]-l[j]+1)*h[j] > ans) ans = (r[j]-l[j]+1)*h[j];
    }
    return ans;
}
```

6.35 Maximum Two Chain Problem

```
typedef struct { int x, y; } point;

int cmp(const void* e1, const void* e2) {
   const point* p1 = (const point*)e1;
   const point* p2 = (const point*)e2;
   if (p1->x != p2->x) return p1->x - p2->x;
   return p1->y - p2->y;
}

int n;
point p[MAX];

void initialize() {
   int i;
   for (scanf("%d", &n), i = 1; i <= n; i++)
      scanf("%d%d", &p[i].x, &p[i].y);

   qsort(&p[1], n, sizeof(point), cmp);
   p[0].x = p[0].y = 0;
}

int deg[MAX] = {0}, queue[MAX];
int maxlevel, level[MAX] = {0};
int left[MAX] = {0}, right[MAX] = {0}, mark[MAX] = {0};
```

```
void local_chain() {
   int i, j;
   for (i = 1; i <= n; i++)
      for (j = i + 1; j <= n; j++)
         if (p[i].y <= p[j].y)
            deg[i]++;

   for (queue[0] = 0, i = 1; i <= n; i++)
      if (deg[i] == 0)
         queue[++queue[0]] = i;
   for (i = 1, maxlevel = -1; i <= queue[0]; i++)
      for (j = 1; j < queue[i]; j++)
         if (p[j].y <= p[queue[i]].y)
            if (--deg[j] == 0) {
               queue[++queue[0]] = j, level[j] = level[queue[i]] + 1;
               if (level[j] > maxlevel) maxlevel = level[j];
            }
   for (maxlevel++, i = 1; i <= n; i++)
      level[i] = maxlevel - level[i];

   for (mark[0] = n + 1, i = 1; i <= n; i++) {
      for (j = 0; j < i; j++)
         if (mark[j] && level[j] == level[i] - 1 && p[j].y <= p[i].y)
            break;
      if (j < i) {
         if (left[level[i]] == 0) left[level[i]] = i, mark[i] = n + 1;
         mark[right[level[i]]]--;
         mark[right[level[i]] = i]++;
      }
   }
}

int index[MAX], value[MAX] = {0}, levvalue[MAX];

int index_cmp(const void* e1, const void* e2) {
   return level[*(const int*)e1] - level[*(const int*)e2];
}

void calc_value() {
   int q, i, j, lev;
   for (i = 1; i <= n; i++)
      index[i] = i;
   qsort(index, n, sizeof(int), index_cmp);

   for (q = 1; q <= n; q++) {
      lev = level[i = index[q]];

      if (left[lev] == i && right[lev] == i)
         value[i] = levvalue[lev - 1] + 1;
      else if (left[lev] == i || right[lev] == i)
         value[i] = levvalue[lev - 1] + 2;
      else
         for (j = 0; j < i; j++) {
            if (mark[j]) value[j] = levvalue[level[j]];
            if (p[j].y <= p[i].y && value[j] + level[i] - level[j] + 1 > value[i])
               value[i] = value[j] + level[i] - level[j] + 1;
         }

      if (value[i] > levvalue[lev])
         levvalue[lev] = value[i];
   }
}
```

```
void put_answer() {
   int i, max = 0;
   for (i = 1; i <= n; i++)
      if (value[i] > max)
         max = value[i];
   printf("%d\n", max);
}

void main() {
   initialize();
   local_chain();
   calc_value();
   put_answer();
}
```

6.36 N Queens Problem

```
int main() {
    int n, i, odd;
    while(cin>>n){
         if(n<4) cout<<"Impossible"; else
         if((n/2)%3!=1){
              cout<<2;
              for(i=4; i<=n; i+=2) cout<<" "<<i;
              for(i=1; i<=n; i+=2) cout<<" "<<i;
         } else {
              if(n&1) { n--; odd = 1; } else odd=0;
              cout<< n/2 ;
              for(i=n/2+1; i!=n/2-1; i=(i+2)%n) cout<<" "<<i+1;
              for(i=(i+n-2)%n; i!=n/2-1; i=(i+n-2)%n) cout<<" "<<n-i;
              cout<<" "<<n-i; if(odd) cout<<" "<<n+1;
         }
         cout<<endl;
    }
    return 0;
}
```

6.37 de Bruijn Sequence Generator

```
int go[1<<maxn],start,now,n,k,a[(1<<maxn)+maxn],i,ans,caseno;

int main()
{
    ifstream cin("input.txt");
    for(cin>>caseno; caseno--;){ cin>>n>>k;
         memset(go,0,sizeof(go));   memset(a,0,sizeof(a));
         now=start=(1<<(n-1))-1;      i=0;
         do { if(go[now]){ a[i++]=1; now=(now*2+1)&start; }
           else { go[now]=1; a[i++]=0; now=(now*2)&start; }
         } while( now!=start );
         a[i++]=1;
         for(i=0; i<n; i++) a[i+(1<<n)]=a[i];
         for(ans=i=0; i<n; i++) ans=ans*2+a[k+i];
         cout<<ans<<endl;
    }
    return 0;
}
```

6.38 ZOJ 1482 Partition

```
#define maxn 3010

int n,pnt[maxn],rank[maxn];

int find(int x)
{
    if(x!=pnt[x]) pnt[x]=find(pnt[x]);
    return pnt[x];
}

int main() {
    int i,j,ans(0),x;
    cin >> n;
    memset(pnt,0,sizeof(pnt));
    for(i=1; i<=n; i++) for(j=1; j<=n; j++){
        cin>>x;
        if(!x){
            if(pnt[j])pnt[find(j)]=j;
            if(pnt[j-1])pnt[j-1]=j;
            pnt[j]=j;
        } else { if(pnt[j]==j) ans++; pnt[j] = 0; }
    }
    for(i=1; i<=n; i++) if(pnt[i]==i) ans++;
    cout<<ans<<endl;
    return 0;
}
```