ACM_LIBRARY

AlphaBet

2015-12-10

XDU

目录

头文件

```
//#pragma comment(linker,"/STACK:102400000,102400000")
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <cassert>
#include <climits>
#include <ctime>
#include <numeric>
#include <vector>
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstring>
#include <iomanip>
#include <complex>
#include <deque>
#include <functional>
#include <list>
#include <map>
#include <string>
#include <sstream>
#include <set>
#include <stack>
#include <queue>
#include <stdio.h>
#include <string.h>
typedef long long LL;
typedef unsigned long long ULL;
typedef long double LD;
#define FOR(i,n) for(int i=0;i<(n);++i)
const double EPS = 1e-6;
const int INF = 0x3fffffff;
const LL LINF = INF * 1ll * INF;
const double PI = acos(-1.0);

using namespace std;
int main()
{
    return 0;
}
```

数据结构

并查集

```
/**********************************************************************
*****
    > File Name: 并查集.cpp
    > Author:
    > Mail:
    > Created Time: 2015年12月09日 星期三 12时49分08秒

**********************************************************************
***/



// 并查集:
 int par[maxn];
 void init(int n)
 {
       for(int i=0; i<n; i++)
          par[i] = i;

 }
 int Find(int x)
 {
       if(par[x] == x) return x;
       else return par[x] = Find(par[x]);

 }


  // 切记执行合并集合的操作时找到两个元素的根然后 par[x] = y
int x = Find(fir), y = Find(sec);
if(x != y)
{
   par[x] = y;
}
```

动态树 LCT

```
//动态树
typedef struct link_cut_tree{

    int son[N][2],sum[N],f[N],key[N],tag[N];

    int up(int x){
        sum[x] = sum[son[x][0]] + sum[son[x][1]] + key[x];
        //
        //
    }

    bool isroot(int x){
        return !f[x] ||  son[f[x]][0] != x && son[f[x]][1] !=x;
    }

    int reverse(int x){
        swap(son[x][0],son[x][1]);
        tag[x] ^= 1;
    }

    int pd(int x){
        if (tag[x]){
            reverse(son[x][0]);
            reverse(son[x][1]);
            tag[x] = 0;
        }
        //
        //
    }

    int rotate(int x){
        int y = f[x];
        int w = (son[y][0] == x);
        son[y][!w] = son[x][w];
        if (son[x][w]) f[son[x][w]] = y;
        if (f[y]){
            int z = f[y];
            if (son[z][0] == y) son[z][0] = x;
            if (son[z][1] == y) son[z][1] = x;
        }
        f[x] = f[y]; f[y] = x; son[x][w] = y;
        up(y); up(x);
```

```
}

stack<int>S;

int splay(int x){
    for (int y = x;!isroot(y);y = f[y]) S.push(y);
    for (;!S.empty();S.pop()) pb(S.top());
    while (!isroot(x)){
        int y = f[x];
        if (!isroot(y)){
            if ((son[f[y]][0] == y)^(son[y][0] == x))
                rotate(x); else rotate(y);
            rotate(x);
        }
    }
    up(x);
}

int access(int x){
    for (int y = 0;x;y = x,x = f[x]){
        splay(x); son[x][1] = y; up(x);
    }
}

int root(int x){
    access(x);
    splay(x);
    while (son[x][0]) x = son[x][0];
    return x;
}

int makeroot(int x){
    access(x);
    splay(x);
    reverse(x);
}

int link(int x,int y){
    makeroot(x);
    f[x] = y;
    access(x);
}

int ask_sum(int x,int y){
        makeroot(x);
        access(y);
        splay(x);
        return sum[x];
    }

}link_cut_tree;
```

函数式线段树

```
//函数式线段树 FSeg.cpp
int siz[N],root[N],sl[trsize],sr[trsize];

void insert(int a,int b,int fx,int &nx,int p,int c){
    nx = ++tot;
    siz[nx] = siz[fx] + c;
    if (a == b) return;
    sl[nx] = sl[fx];
    sr[nx] = sr[fx];
    int mid = (a + b) >> 1;
    if (p <= mid) insert(a,mid,sl[fx],sl[nx],p,c);
    else insert(mid + 1,b,sr[fx],sr[nx],p,c);
}

int ask(int a,int b,int x,int y,int k){
    if (a == b) return a;
    int mid = (l + r) >> 1;
    if (siz[sl[y]] - siz[sl[x]] >= k)
        return ask(l,mid,sl[x],sl[y],k);
    else return ask(mid + 1,r,sr[x],sr[y],k - siz[sl[y]] +
siz[sl[x]]);
}
```

红黑树

```
//红黑树
#include <map>

using namespace std;
```

```
const int N = 1100000;

map<int,int>M;

if (!M.count(x)) M[x]++;
```

集合

```
//集合
#include <set>

set<int>S;

S.insert();

set<int>::iterator i;

i = S.begin();

while (i != S.end())
    printf("%d\n",*i++);

S.clear();

S.count(4);

S.empty();

S.erase();

S.find();

S.lower_bound();

//大于或等于某值的第一个位置

S.rbegin();

S.upper_bound();

//大于某个数的第一个位置
```

```
离散化/*离散化
  fist edited by williamchen 2015/11/24

 */

typedef struct discrete{

    int tmp[N],n,tot;

    int search(int x){
        int l = 1,r = tot;
        do{
            int mid = (l + r) >> 1;
            if (tmp[mid] == x) return mid;
            if (tmp[mid] > x) r = mid - 1;
            else l = mid + 1;
            }while(l <= r);
        return 0;
    }

    int init(int x,int a[]){
        n = x;
        tot = 0;
        for (int i = 1;i <= n;i++) tmp[i] = a[i];
        sort(tmp+1,tmp+n+1);
        for (int i = 1;i < n;i++)
            if (tmp[i] != tmp[i+1]) tmp[++tot] = tmp[i];
        tmp[++tot] = tmp[n];
        for (int i = 1;i <= n;i++)
            a[i] = search(a[i]);
        return tot;
    }

}discrete;

discrete DisC;
```

莫队算法

```
//莫队算法
typedef struct que{
    int l,r,id;
}que;
```

```
que qu[N];

bool cmp(que a,que b){
    if (pos[a.l] == pos[b.l]) return a.r < b.r;
    return a.l < b.l;
}

int bkn;

int init(){
    bkn = int(sqrt(n));
    for (int i = 1;i <= n;i++)
        pos[i] = (i - 1)/bkn + 1;
}
```

伸展树 Splay

```
//伸展树 Splay.cpp
int tr[80001][2],num[80001],fa[80001];
void rotate(int x,int &k)
{
    int y=fa[x],z=fa[y],l,r;
    if(tr[y][0]==x)l=0;else l=1;r=l^1;
    if(y==k)k=x;
    else{if(tr[z][0]==y)tr[z][0]=x;else tr[z][1]=x;}
    fa[x]=z;fa[y]=x;fa[tr[x][r]]=y;
    tr[y][l]=tr[x][r];tr[x][r]=y;
}
void splay(int x,int &k)
{
    int y,z;
    while(x!=k)
    {
        y=fa[x],z=fa[y];
        if(y!=k)
        {
            if((tr[y][0]==x)^(tr[z][0]==y))rotate(x,k);
            else rotate(y,k);
        }
        rotate(x,k);
    }
}
void ins(int &k,int x,int last)
```

```
{
    if(k==0){size++;k=size;num[k]=x;fa[k]=last;splay(k,rt);return;}
    if(x<num[k])ins(tr[k][0],x,k);else ins(tr[k][1],x,k);
}
void del(int x)
{
    splay(x,rt);
    if(tr[x][0]*tr[x][1]==0)
    {rt=tr[x][0]+tr[x][1];}
    else
    {
        int k=tr[x][1];
        while(tr[k][0])k=tr[k][0];
        tr[k][0]=tr[x][0];fa[tr[x][0]]=k;
        rt=tr[x][1];
    }
    fa[rt]=0;
}
void ask_before(int k,int x)
{
    if(k==0)return;
    if(num[k]<=x){t1=k;ask_before(tr[k][1],x);}
    else ask_before(tr[k][0],x);
}
void ask_after(int k,int x)
{
    if(k==0)return;
    if(num[k]>=x){t2=k;ask_after(tr[k][0],x);}
    else ask_after(tr[k][1],x);
}
```

绳 Rope

```
//绳 Rope
#include<iostream>
#include<cstdio>
#include<ext/rope>
using namespace std;
using namespace __gnu_cxx;
crope list;
int t,now;
char ch[3000005];
```

```
inline int read()
{
    int x=0,f=1;char ch=getchar();
    while(ch>'9'||ch<'0'){if(ch=='-')f=-1;ch=getchar();}
    while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}
    return x*f;
}
int main()
{
    t=read();
    char s[10];int x;
    while(t--)
    {
        scanf("%s",s);
        switch(s[0])
        {
        case 'M':now=read();break;
        case 'P':now--;break;
        case 'N':now++;break;
        case 'I':
            x=read();
            for(int i=0;i<x;i++)
            {
                ch[i]=getchar();
                while(ch[i]=='\n')ch[i]=getchar();
            }
            ch[x]=0;
            list.insert(now,ch);
            break;
        case 'D':x=read();list.erase(now,x);break;
        case 'G':x=read();list.copy(now,x,ch);ch[x]=0;puts(ch);
        }
    }
    return 0;
}
```

树套树

```
//树套树
int build(int a,int b){
    int x = ++tot;
    if (a == b) {root[a] = 1; return;}
    int mid = (a + b) >> 1;
    sl[x] = build(a,mid);
    sr[x] = build(mid + 1,b);
}

int change(int a,int b,int x,int p,int delta){
    int v = ++tot;
    val[v] = val[x] + delta;
    if (a == b) return v;
    if (p < mid){
        sr[v] = sr[x];
        sl[v] = change(a,mid,sl[x],p,delta);
    }else{
        sl[v] = sl[x];
        sr[v] = change(mid+1,b,sr[x],p,delta);
    }
    return v;
}

int ask(int a,int b,int x,int c,int d){
    if (c <= a && b <= d) return val[x];
    int mid = (a + b) >> 1;
    int ans = 0;
    if (c <= mid) ans += ask(a,mid,sl[x],c,d);
    if (mid < d) ans += ask(mid+1,d,sr[x],c,d);
    return ans;
}

int lowbit(int x){
    return x & (-x);
}

int segchange(int x,int p,int c){
    for (int t;x <= n;x += lowbit(x)){
        t = root[x]; root[x] = tot++;
        change(1,n,t,p,c);
    }
}

int segsum(int x,int c,int d){
    if (c > d) return 0;
    int ans = 0;
    for (;x;x -= lowbit(x))
      ans += ask(1,n,root[x],c,d);
    return ans;
}
```

树状数组 BIT

```
//树状数组
typedef struct Fenwick_Tree{

    ll C[N];
    int siz;

    int init(int x){
        for (int i = 0;i <= x+1;i++) C[i] = 0;
        return siz = x;
    }

    int lowbit(int x){
        return x & (-x);
    }

    int add(int x,ll t){
        for (;x <= siz;x += lowbit(x))
          C[x] += t;
    }

    ll sum(int x){
        ll ans = 0;
        for (;x;x -= lowbit(x)){
            ans += C[x];
        }
        return ans;
    }

    ll intersum(int l,int r){
        if (l > r) return 0;
        return sum(r) - sum(l-1);
    }

}Fenwick_Tree;

Fenwick_Tree Fen;
```

位集 Bitset

```
//位集Bitset.cpp
#include <bitset>

using namespace std;

const int N = 200010;

bitset<N>a;

int example(){
    a = a ^ a;
    a ^= (a << x);
    a[0] = 1;
    a[1] = 0;
}
```

线段树

```
//线段树
typedef struct segment_tree{
    int siz,tot;
    int sl[N],sr[N];
    int tl[N],tr[N];
    ll sum[N],flag[N];

    int init(int x){
        siz = x; tot = 0;
        memset(sl,0,sizeof(l));
        memset(sr,0,sizeof(r));
        memset(sum,0,sizeof(sum));
        memset(flag,0,sizeof(flag));
    }

    int up(int x){
        if (!flag[x]) return 0;
        int mid = (l + r) >> 1;
        sum[x] += flag[x] * (tr[x] - tl[x] + 1);
        flag[sl[x]] += flag[x];
        flag[sr[x]] += flag[x];
        return flag[x] = 0;
    }

    int update(in   ){
        if (tl[x] == tr[x]) return 0;
        int mid = (l + r) >> 1;
        up(x); up(sl[x]); up(sr[x]);
        sum[x] = sum[sl[x]] + sum[sr[x]];
    }

    int build(int l,int r){
        int x = ++tot;
        tl[x] = l; tr[x] = r;
        if (l == r) return x;
        int mid = (l + r) >> 1;
        sl[x] = build(l,mid);
        sr[x] = build(mid + 1,r);
        update(x);
    }

    int add(int x,int lx,int rx,ll c){
        if (tl[x] >= lx && tr[x] <= rx) return flag[x] += c;
        int mid = (l + r) >> 1; up(x);
        if (mid >= lx) add(sl[x],lx,rx,c);
        if (mid < rx) add(sr[x],lx,rx,c);
        update(x);
    }

    int query(int x,int lx,int rx){
        update(x);
        if (tl[x] >= lx && tr[x] <= rx) return sum[x];
        int mid = (l + r) >> 1; ll ans = 0;
        if (mid >= lx) ans += query(sl[x],lx,rx);
        if (mid < rx) ans += query(sr[x],lx,rx);
        return ans;
    }

}segment_tree;

segment_tree Seg;
```

主席树

```
//主席树求区间众数+读入优化
#include<cstdio>
#include<iostream>
using namespace std;
int n,m,sz;
int root[500010],ls[10000010],rs[10000010],sum[10000010];

inline int read()
{
    char ch=getchar();
    while(!(ch>='0'&&ch<='9'))ch=getchar();
    int x=0;
    while(ch>='0'&&ch<='9'){x=x*10+(ch-'0');ch=getchar();}
    return x;
}

void update(int l,int r,int x,int &y,int v)
{
    y=++sz;
    sum[y]=sum[x]+1;
    if(l==r)return;
    ls[y]=ls[x];rs[y]=rs[x];
    int mid=(l+r)>>1;
    if(v<=mid)update(l,mid,ls[x],ls[y],v);
    else update(mid+1,r,rs[x],rs[y],v);
}

int que(int L,int R)
{
    int l=1,r=n,mid,x,y,tmp=(R-L+1)>>1;
    x=root[L-1];y=root[R];
    while(l!=r)
    {
        if(sum[y]-sum[x]<=tmp)return 0;
        mid=(l+r)>>1;
        if(sum[ls[y]]-sum[ls[x]]>tmp)
        {r=mid;x=ls[x];y=ls[y];}
        else if(sum[rs[y]]-sum[rs[x]]>tmp)
        {l=mid+1;x=rs[x];y=rs[y];}
        else return 0;
    }
    return l;
}

int main()
{
    n=read();m=read();
    for(int i=1;i<=n;i++)
    {
        int x;x=read();
        update(1,n,root[i-1],root[i],x);
    }
    for(int i=1;i<=m;i++)
    {
        int l,r;l=read();r=read();
        printf("%d\n",que(l,r));
    }
    return 0;
}
```

数学

组合数 Lucas

```
/**********************************************************************
*****
    > File Name: lucas.cpp
    > Author:
    > Mail:
    > Created Time: 2015年12月08日 星期二 13时35分34秒

**********************************************************************
***/
#include<string.h>
#include<algorithm>
#include<stdio.h>
using namespace std;
long long m,n,p;
long long  pow(long long  a,long long  b,long long  mod)
{
       long long  ans=1;
       while(b)
    {
```

```
                if(b&1)
        {
                        b--;
                        ans=(ans*a)%mod;

        }
                else
        {
                        b/=2;
                        a=(a*a)%mod;

        }

    }
        return ans;

}
long long  fun1(long long  n,long long  m)
{
        if(n<m)
            return 0;
       long long  ans=1;
        for(int i=1;i<=m;i++)
    {
                ans=ans*(((n-m+i)%p)*pow(i,p-2,p)%p)%p;

    }
        return ans;

}
long long  lucas(long long  n,long long  m)
{
        if(m==0)
            return 1;
        return (lucas(n/p,m/p)*fun1(n%p,m%p))%p;

}
int main()
{
     p=10007;
    while(scanf("%lld%lld",&n,&m)!=EOF)
    {

      printf("%lld\n",lucas(n,m));    //C(N,M)
    }
        return 0;

}
```

Miller_Rabin 判断素数

```
/*********************************************************************
*****
    > File Name: Miller_Rabin——判断素数.cpp
    > Author:
    > Mail:
    > Created Time: 2015年12月08日 星期二 14时05分12秒

*********************************************************************
***/
bool Miller_Rabin(LL n)
{
        int T=S;
        if(n==1||n==0)
            return 0;
        if(n==2)
            return 1;
        if(!(n&1))
            return 0;
        LL x=n-1;
        LL y,s=0,r,a;
        while((x&1)==0)
    {
                s++;
                x>>=1;

    }
        r=x;
        while(T--)
    {
                a=rand()%(n-1)+1;
                y=pow_mod(a,r,n);
                if(y!=1&&y!=n-1)
        {
                        for(LL j=1;j<=s-1&&y!=n-1;j++)
            {
```

```
                        y=mult_mod(y,y,n);
                        if(y==1)
                            return 0;

            }
                    if(y!=n-1)
                        return 0;

        }
                else
                    return 1;

    }
        return 1;


}
```

分解质因数

```
/**********************************************************************
*****
    > File Name: pollard_rho——质因数分解.cpp
    > Author:
    > Mail:
    > Created Time: 2015年12月08日 星期二 14时08分51秒

**********************************************************************
***/

long long Pollard_rho(long long x,long long c)
{
        long long i=1,k=2;
        long long x0=rand()%x;
        long long y=x0;
        while(1)
    {
                i++;
                x0=(mult_mod(x0,x0,x)+c)%x;
                long long d=gcd(y-x0,x);
                if(d!=1&&d!=x) return d;
                if(y==x0) return x;
```

```
            if(i==k){y=x0;k+=k;}

   }

}
void findfac(long long n)
{
       if(Miller_Rabin(n))
   {
             factor[tol++]=n;
             return;

   }
      long long p=n;
      while(p>=n)p=Pollard_rho(p,rand()%(n-1)+1);
      findfac(p);
      findfac(n/p);

}
```

博弈论——SG 函数

```
/***********************************************************************
*****
   > File Name: 博弈论——SG 函数.cpp
   > Author:
   > Mail:
   > Created Time: 2015 年 12 月 08 日 星期二 17 时 16 分 29 秒

***********************************************************************
***/

/*
* SG 函数
*  N 求解范围 S[]数组是可以每次取的值，t 是 s 的长度。
*  */
int sg[N];
bool hash[N];
void sg_solve(int *s,int t,int N)
{
      int i,j;
      memset(sg,0,sizeof(sg));
      for(i=1;i<=N;i++)
```

```
    {
              memset(hash,0,sizeof(hash));
              for(j=0;j<t;j++)
                 if(i - s[j] >= 0)
                    hash[sg[i-s[j]]] = 1;
              for(j=0;j<=N;j++)
                 if(!hash[j])
                    break;
              sg[i] = j;

    }

}


/*
 * 注意 S 数组要按从小到大排序 SG 函数要初始化为-1 对于每个集合只需初始化 1 遍
 * n 是集合 s 的大小 S[i]是定义的特殊取法规则的数组
 * */
 int s[110],sg[10010],n;
 int SG_dfs(int x)
 {
        int i;
        if(sg[x]!=-1)
           return sg[x];
        bool vis[110];
        memset(vis,0,sizeof(vis));
        for(i=0;i<n;i++)
     {
               if(x>=s[i])
        {
                      SG_dfs(x-s[i]);
                      vis[sg[x-s[i]]]=1;

        }

     }
        int e;
        for(i=0;;i++)
           if(!vis[i])
     {
                   e=i;
                   break;
    }
        return sg[x]=e;
 }
```

大步小步法 BSGS

```
//大步小步法 用于求(Y^X = Z) mod P
#define ll long long
using namespace std;
int T,K;
int read()
{
    int x=0;char ch=getchar();
    while(ch<'0'||ch>'9')ch=getchar();
    while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}
    return x;
}
int gcd(int a,int b)
{
    return b==0?a:gcd(b,a%b);
}
void exgcd(int a,int b,int &x,int &y)
{
    if(b==0){x=1,y=0;return;}
    exgcd(b,a%b,x,y);
    int t=x;x=y;y=t-a/b*y;
}
int solve1(ll y,int z,int p)
{
    y%=p;
    ll ans=1;
    for(int i=z;i;i>>=1,y=y*y%p)
        if(i&1)ans=ans*y%p;
    return ans;
}
void solve2(int y,int z,int p)
{
    p=-p;
    int t=gcd(y,p);
    if(z%t){puts("Orz, I cannot find x!");return;}
    y/=t;z/=t;p/=t;
    int a,b;exgcd(y,p,a,b);
    a=(ll)a*z%p;
    while(a<0)a+=p;
    printf("%d\n",a);
}
map<int,int> mp;
void solve3(int y,int z,int p)
{
    y%=p;
    if(!y&&!z){puts("1");return;}
    if(!y){puts("Orz, I cannot find x!");return;}
    mp.clear();
    ll m=ceil(sqrt(p)),t=1;
    mp[1]=m+1;
    for(ll i=1;i<m;i++)
    {
        t=t*y%p;
        if(!mp[t])mp[t]=i;
    }
    ll tmp=solve1(y,p-m-1,p),ine=1;
    for(ll k=0;k<m;k++)
    {
        int i=mp[z*ine%p];
        if(i)
        {
            if(i==m+1)i=0;
            printf("%lld\n",k*m+i);
            return;
        }
        ine=ine*tmp%p;
    }
    puts("Orz, I cannot find x!");
}
```

二进制一的个数

```
/**********************************************************************
*****
    > File Name: 二进制中 1 的个数.cpp
    > Author:
```

```
    > Mail:
    > Created Time: 2015年12月08日 星期二 13时42分34秒
 ************************************************************************/

int BitCount2(unsigned int n)
{
        unsigned int c =0 ;
        for (c =0; n; ++c)
    {
                n &= (n -1) ; // 清除最低位的1

    }
        return c ;

}
//或者
int cal(int n)
{
         n = (n &0x55555555) + ((n >>1) &0x55555555) ;
        n = (n &0x33333333) + ((n >>2) &0x33333333) ;
        n = (n &0x0f0f0f0f) + ((n >>4) &0x0f0f0f0f) ;
        n = (n &0x00ff00ff) + ((n >>8) &0x00ff00ff) ;
        n = (n &0x0000ffff) + ((n >>16) &0x0000ffff) ;
        return n;

}
```

反素数

```
/*************************************************************************
    > File Name: 反素数.cpp
    > Author:
    > Mail:
    > Created Time: 2015年12月08日 星期二 17时13分49秒
 ************************************************************************/

/*
* 网上模板:
```

```
* 对于任何一个正整数 x，其约数为 g(x)，对于任意 i<x,都有 g(i)<g(x),则 x 为反素数
* */
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cmath>
#include<cstring>

using namespace std;

typedef long long lld;

lld prime[20]={2,3,5,7,11,13,17,19,23,29,31,37,39,41,43,47,53};
lld n;
lld bestcurr,largecnt;/*bestcurr 相同最大因数个数中值最小的数, largecnt: n
范围内最大的因数个数*/
void getarcprime(lld curr,int cnt,int limit,int k)
{
       if(curr>n)
          return ;
       if(largecnt<cnt)/*此时枚举到的因数个数比之前记录的最大的因数个数要大，
就替换最大因数个数*/
    {
             largecnt=cnt;
             bestcurr=curr;

    }
       if(largecnt==cnt && bestcurr>curr)/*替换最优值*/
          bestcurr=curr;
       lld temp=curr;
       for(int i=1;i<=limit;i++)
    {
             temp=temp*prime[k];
             if(temp>n)
                return;
             getarcprime(temp,cnt*(i+1),i,k+1);


    }

}
int main()
{
       int i,cas;
```

```
        scanf("%d",&cas);
        for(i=1;i<=cas;i++)
    {
                scanf("%lld",&n);
                bestcurr=0;
                largecnt=0;
                getarcprime(1,1,50,0);
                printf("Case #%d: %lld\n",i,bestcurr);

    }
        return 0;

}
```

高斯消元

```
//高斯消元 Gauss.cpp
int swap(int x,int y){
      for (int i=0;i<=n;i++){
        double tmp=c[x][i];
        c[x][i]=c[y][i];
        c[y][i]=tmp;
       }
 }

int guass(){
     for (int i=1;i<=n;i++)
         for (int k=i,j=i;j<=n;j++)
         {if (fabs(c[j][i-1])>fabs(c[k][i-1])) k=j;
         swap(i,k);
         for (int j=i+1;j<=n;j++)
         for (int k=n;k>=i-1;k--)
         c[j][k]-=c[i][k]*c[j][i-1]/c[i][i-1];}

}

int solve(){
    double ans[N];
    ans[n]=1;
    for (int i=n;i>=1;i--)
      {  double sum=0;
         for (int j=i;j<=n;j++)
         sum-=ans[j]*c[i][j];
         ans[i-1]=sum/c[i][i-1];
         }
     for (int i=0;i<n;i++)
      if (i) printf(" %.3lf",ans[i]);
      else printf("%.3lf",ans[i]);
      printf("\n");
}
```

矩阵乘法

```
//矩阵乘法
typedef long long ll;

const int N = 105;

typedef struct matrix{

   int n,m;

   int va[N][N];

   int init(int sn,int sm){
      n = sn;
      m = sm;
      memset(va,0,sizeof(va));
   }

}matrix;

matrix mul(matrix a,matrix b,ll M){
   matrix c;
   c.init(a.n,b.m);
   for (int i = 1;i <= a.n;i++)
      for (int j = 1;j <= b.m;j++)
         for (int k = 1;k <= a.m;k++)
           (c.va[i][j]+=a.va[i][k]* b.va[k][j])%=M;
   reuturn c;
}

matrix power(matrix a,ll b,ll M){
   if (b == 1) return a;
   matrix tmp = power(a,b >> 1,M);
```

```
    tmp = mul(tmp,tmp,M);
    if (b & 1) tmp = mul(tmp,a,M);
    return tmp;
}
```

大数

```
/**********************************************************************
*****
    > File Name: 大数 BigNum.cpp
    > Author:
    > Mail:
    > Created Time: 2015 年 12 月 08 日  星期二  13 时 41 分 40 秒

**********************************************************************
***/

#include <cstdlib>
#include <cstring>
#include <string>
#include <algorithm>
using namespace std;

const int MAXN = 410;

struct bign
{
       int len, s[MAXN];
       bign ()
       {
           memset(s, 0, sizeof(s));
           len = 1;

       }
       bign (int num) { *this = num;  }
       bign (const char *num) { *this = num;  }
       bign operator = (const int num)
       {
           char s[MAXN];
           sprintf(s, "%d", num);
           *this = s;
           return *this;
```

```
}
bign operator = (const char *num)
{
    for(int i = 0; num[i] == '0'; num++) ;  //去前导0
    len = strlen(num);
    for(int i = 0; i < len; i++) s[i] = num[len-i-1] - '0';
    return *this;

}
bign operator + (const bign &b) const //+
{
    bign c;
    c.len = 0;
    for(int i = 0, g = 0; g || i < max(len, b.len); i++)
    {
        int x = g;
        if(i < len) x += s[i];
        if(i < b.len) x += b.s[i];
        c.s[c.len++] = x % 10;
        g = x / 10;

    }
        return c;

}
bign operator += (const bign &b)
{
    *this = *this + b;
    return *this;

}
void clean()
{
    while(len > 1 && !s[len-1]) len--;

}
bign operator * (const bign &b) //*
{
    bign c;
    c.len = len + b.len;
    for(int i = 0; i < len; i++)
    {
        for(int j = 0; j < b.len; j++)
        {
            c.s[i+j] += s[i] * b.s[j];
        }

    }
        for(int i = 0; i < c.len; i++)
        {
                c.s[i+1] += c.s[i]/10;
                c.s[i] %= 10;

        }
        c.clean();
        return c;

}
bign operator *= (const bign &b)
{
    *this = *this * b;
    return *this;

}
bign operator - (const bign &b)
{
    bign c;
    c.len = 0;
    for(int i = 0, g = 0; i < len; i++)
    {
        int x = s[i] - g;
        if(i < b.len) x -= b.s[i];
            if(x >= 0) g = 0;
            else
            {
                g = 1;
                x += 10;

            }
                c.s[c.len++] = x;

    }
    c.clean();
    return c;

}
bign operator -= (const bign &b)
```

```
{
    *this = *this - b;
    return *this;

}
bign operator / (const bign &b)
{
    bign c, f = 0;
    for(int i = len-1; i >= 0; i--)
    {
        f = f*10;
        f.s[0] = s[i];
        while(f >= b)
        {
            f -= b;
            c.s[i]++;

        }

    }
    c.len = len;
    c.clean();
    return c;

}
bign operator /= (const bign &b)
{
    *this  = *this / b;
    return *this;

}
bign operator % (const bign &b)
{
    bign r = *this / b;
    r = *this - r*b;
    return r;

}
bign operator %= (const bign &b)
{
    *this = *this % b;
    return *this;

}
```

```
bool operator < (const bign &b)
{
    if(len != b.len) return len < b.len;
    for(int i = len-1; i >= 0; i--)
    {
        if(s[i] != b.s[i]) return s[i] < b.s[i];

    }
    return false;

}
bool operator > (const bign &b)
{
    if(len != b.len) return len > b.len;
    for(int i = len-1; i >= 0; i--)
    {
        if(s[i] != b.s[i]) return s[i] > b.s[i];

    }
   return false;

}
bool operator == (const bign &b)
{
    return !(*this > b) && !(*this < b);

}

bool operator != (const bign &b)
{
        return !(*this == b);

}
bool operator <= (const bign &b)
{
    return *this < b || *this == b;

}
bool operator >= (const bign &b)
{
    return *this > b || *this == b;

}
string str() const
```

```
        {
            string res = "";
            for(int i = 0; i < len; i++) res = char(s[i]+'0') + res;
            return res;

        }

};

istream& operator >> (istream &in, bign &x)
{
        string s;
        in >> s;
        x = s.c_str();
        return in;

}

ostream& operator << (ostream &out, const bign &x)
{
        out << x.str();
        return out;

}

int main()
{
        bign a, b, c, d, e, f, g;
        while(cin>>a>>b)
    {
                a.clean(), b.clean();
                c = a+b;
                d = a-b;
                e = a*b;
                f = a/b;
                g = a%b;
                cout<<"a+b"<<"="<<c<<endl; // a += b
                cout<<"a-b"<<"="<<d<<endl; // a -= b;
                cout<<"a*b"<<"="<<e<<endl; // a *= b;
                cout<<"a/b"<<"="<<f<<endl; // a /= b;
                cout<<"a%b"<<"="<<g<<endl; // a %= b;
                if(a != b) printf("YES\n");
                else printf("NO\n");
    }
        return 0;

}
```

欧拉函数

```
//欧拉函数
//筛法
for(int i=1;i<=n;i++) phi[i]=i;
   for (int i=2;i<=n;i++)
   if (!b[i]){
         phi[i]--;
         for (int j=2;j*i<=n;j++)
         phi[i*j]=(phi[i*j]/i)*(i-1),b[i*j]=1;
   }


//求单个数的欧拉函数
int eular(int n)
{
        int ret = 1,i;
        for (i = 2;i * i <= n;i++)
            if (n % i == 0)
    {
                    n /= i;
                    ret *= (i - 1);
                    while (n % i == 0)
        {
                            n /= i;
                            ret *= i;

        }

    }
        if (n > 1)
            ret *= (n - 1);
        return ret;
}
```

素数筛

```
/*********************************************************************
*****
    > File Name: 素数筛.cpp
    > Author:
    > Mail:
    > Created Time: 2015年12月08日 星期二 13时43分25秒
*********************************************************************
***/


#include<iostream>
using namespace std;
const long N = 200000;
long prime[N] = {0},num_prime = 0;
int isNotPrime[N] = {1, 1};
int main()
{
       for(long i = 2 ; i < N ; i ++)
       {
             if(! isNotPrime[i])
                 prime[num_prime ++]=i;
             for(long j = 0 ; j < num_prime && i * prime[j] <  N ; j
++)
                {
                             isNotPrime[i * prime[j]] = 1;
                         if( !(i % prime[j] )  )
                             break;
                      }
          }
    return 0;
}
```

随机数生成

```
//随机数生成
typedef struct Random{
    int seed,modnum,now;

    int init(int x,int y){
        seed = x;
        modnum = y;
        now = seed;
    }

    int getRand(){
        now = (now * 31 + 997) % modnum;
        return now;
    }
}Ran;
```

扩展欧几里得

```
//扩展欧几里得
ll ex_gcd(ll a,ll b,ll &x,ll &y){
    if (!b) return x = 1,y = 0,a;
    int gcd = ex_gcd(b,a % b,x,y);
    int t = x; x = y; y = t - a / b *y;
    return gcd;
}
```

唯一分解定理

```
/*********************************************************************
*****
    > File Name: 唯一分解定理.cpp
    > Author:
    > Mail:
    > Created Time: 2015年12月09日 星期三 12时57分44秒
*********************************************************************
***/


 //唯一分解定理:
```

```
int res[maxn];
void getwy(int n)
{
      for(int i=0; i<num; i++)   //num为素数的个数
   {
            while(n%prime[i] == 0)
      {
                   res[i]++;
                   n/=prime[i];

      }
            if(n == 1) break;

   }

}
```

微型素数表

```
//微型素数表
typedef long long ll;

ll lucky[25] = {
   2,3,5,7,11,
   31,131,283,251,257,
   283,353,373,389,409,
   32713,32869,33347,33377,34583,
   9999999997,10000000000007,61,99999999997,10000007
};
```

约瑟夫

```
/*********************************************************************
*****
   > File Name: 约瑟夫.cpp
   > Author:
   > Mail:
   > Created Time: 2015年12月08日 星期二 14时10分01秒

*********************************************************************
***/
```

```
LL fun(LL n)
{
        if(n==0)
            return 1;
        LL f=0;
        for(LL i=1;i<=n;i++)
            f=(f+m)%i;
        return f+1;

}

//M=2 时

LL fun(LL n)
{
        if(n==1)
            return 1;
        if(n&1)
            return 2*fun((n-1)/2)+1;
        else
            return  2*fun(n/2)-1;

}

//某些约瑟夫可采用线段树来做
```

卡特兰数

catalan Catlan(n)= C(n,2n) - c(n+1,2n)
最基本应用 n 个位置 合法的括号序列 Catalan(n)
衍生应用 n 个节点组成二叉树个数 Catlan(n)
这种题找规律是王道 1,1,2,5,14,42,132,429

图论

最小生成树

```
/***********************************************************************
*****
    > File Name: 最小生成树 Kruskal.cpp
    > Author:
    > Mail:
    > Created Time: 2015年12月09日 星期三 13时10分23秒

***********************************************************************
***/

#include<iostream>
using namespace std;
int par[maxn];
void init(int n)
{
      for(int i=0; i<=n; i++)
         par[i] = i;

}
int Find(int x)
{
      if(par[x] == x) return x;
      else return par[x] = Find(par[x]);

}
ll kruskal()
{
      init(n);                                      //并查集初始化
      sort(e, e+num);
      int m = 0; ll ans = 0;
      for(int i=0; i<num; i++)
   {
             int x = Find(e[i].u), y = Find(e[i].v);
             if(x != y)
       {
                    ans += e[i].cost;
                    m++;
                    par[x] = y;                //注意并查集合并操作 之前
写错了

       }
```

```
            if(m == n-1) break;                         //n
是顶点数

    }
      if(m != n-1)   //最小生成树不存在
      return ans;

}
```

最近公共祖先 LCA

```
//最近公共祖先 LCA.cpp
int dfs(int u){
    vis[u]=1;
    for (int i=base[u];i;i=next[i])
      if (!vis[now[i]]){
         int x=now[i];
         dp[x][0]=u; d[x]=d[u]+1;
         dfs(x);
         }
    }


int lca(int x,int y){
   if (x==y) return x;

   if (d[x]<d[y]) swap(x,y);

   for (int i=k;i>=0;i--)
   if (d[dp[x][i]]>=d[y]) x=dp[x][i];

   if (x==y) return x;

   for (int i=k;i>=0;i--)
   if (dp[x][i]!=dp[y][i])
    x=dp[x][i],y=dp[y][i];
   return dp[x][0];
}
```

最短路 SPFA

```
/* ID 03
   get minipath
   and find strange circle
 */
stack<int>S;

bool spfa(){

  for (int i=1;i<=n;i++) {S.push(i); vis[i]=1;}

  while (!S.empty()){
      int u=S.top(); S.pop(); vis[u]=0;
      for (int i=base[u];i;i=Nx[i])
          if (d[u]+c[i]<d[now[i]]){
             int x = now[i]; cnt[i]++;
             if (cnt[i]==n) return 0;
             if (!vis[x]) S.push(x);
             d[x] = d[u]+c[i];
             vis[x] = 1;
             }
         }return 1;
   }
```

最大流 SAP

```
//最大流 SAP
int sap(int u,int flow){
    if (u == T) return flow;
    int tmp,rec = 0;
    for (int i = base[u]; i;i = nex[i]){
        int x = to[i];
        if (fl[i] <= 0 || d[u] != d[x] + 1) continue;
        tmp = sap(x,min(flow - rec,fl[i]));
        rec += tmp; fl[i] -= tmp; fl[op[i]] += tmp;
        if (rec == flow) return flow;
    }
    if (d[S] >= ntot) return rec;
    num[d[u]]--; if (!num[d[u]]) d[S] = ntot;
    num[++d[u]]++; return rec;
}
```

```
int nadd(int x,int y,int f){
   to[++tot] = y; fl[tot] = f;
   nxt[tot] = base[x]; base[x] = tot;
   op[tot] = tot + 1;
   to[++tot] = x; fl[tot] = f;
   nxt[tot] = base[y]; base[y] = tot;
   op[tot] = tot - 1;
}
```

ZKW 费用流

```
//ZKW.cpp
#include <cstdio>
#include <cstring>
#include <iostream>

#define debug printf("xxxxxx");

using namespace std;

const int maxn=2010;

const int inf=~0U>>1;

int f,fa,fb,n,a,b;

int tmp,ans=0,tot;

typedef struct edge{int x,op,f,cost,next;}edge;

edge v[maxn*2100]; int dis[maxn],base[maxn]; bool vis[maxn];

int S,T; int cur[maxn];

int min(int a,int b) {if (a<b) return a;else return b;}

int aug(int u,int flow)
  {   if (u==T) {ans+=flow*dis[S]; return flow;}
     vis[u]=1; int now=0;
```

```
      for (int i=base[u];i;i=v[i].next)
      { int x=v[i].x;
        if (vis[x]||!v[i].f||dis[u]!=dis[x]+v[i].cost)
          continue;
       int tmp=aug(x,min(flow-now,v[i].f));
       if (tmp) v[i].f-=tmp; v[v[i].op].f+=tmp;
       now+=tmp;  if (now==flow) return flow;
     } return now;
  }

int modlable()
  {  int del=inf;
     for (int i=S;i<=T;i++)
     if (vis[i])
     for (int j=base[i];j;j=v[j].next)
     if (v[j].f) {int x=v[j].x;
      if (!vis[x]) del=min(del,dis[x]+v[j].cost-dis[i]);}
     if (del==inf) return 0;
     for (int i=S;i<=T;i++)
       if (vis[i]) vis[i]=0,dis[i]+=del,cur[i]=base[i]; return 1;
   }
int zkw()
  {  for (int i=S;i<=T;i++) cur[i]=base[i];
     do {while (aug(S,inf)) memset(vis,0,sizeof(vis));}
     while (modlable()); printf("%d\n",ans);
   }

int add(int x,int y,int f,int c)
   {  v[++tot].x=y; v[tot].op=tot+1;
      v[tot].f=f; v[tot].cost=c;
      v[tot].next=base[x]; base[x]=tot;
      v[++tot].x=x; v[tot].op=tot-1;
      v[tot].f=0; v[tot].cost=-c;
      v[tot].next=base[y]; base[y]=tot;
   }

int main(){
     scanf("%d%d%d%d%d%d",&n,&a,&b,&f,&fa,&fb);
     S=0; T=2*n+1;
      for (int i=1;i<=n;i++)
         { scanf("%d",&tmp);
           add(S,i,tmp,f);
           add(i,T,tmp,0);
           add(S,n+i,tmp,0);
```

```
            for (int j=i+a+1;j<=n;j++)
            add(n+i,j,inf,fa);
            for (int j=i+b+1;j<=n;j++)
            add(n+i,j,inf,fb);
          }
     zkw();
}
```

匈牙利

```
//匈牙利
#include <bitset>

using namespace std;

const int N = 1000001;

bitset<N>pos;

int find(int x){
    for (int i = 1;i <= n;i++)
        if (g[x][i] && !vis[i]){
            vis[i] = 1;
            if (lk[i] == 0||find(lk[i])){
                lk[i] = x; return 1;
            }
        }
        return 0;
}

for (int i = 1;i <= n;i++){
    vis.reset();
    if (find(i)) ans++;
}
```

堆优化 DIJ

```
//堆优化 Dij
typedef struct seg{

    int id,va;
    seg(int a,int b){id = a; va = b;}

    bool operator <(const seg &x)const{
        if (va == x.va) return id < x.id;
        return va > x.va;
    }

}seg;

priority_queue<seg>Q;

bool vis[N];

int dij(int s){

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

    d[s] = 0;

    memset(vis,0,sizeof(vis));

    Q.push(seg(s,d[s]));

    while (!Q.empty()){
        seg tmp = Q.top();
        while (vis[tmp.id]) {Q.pop(); tmp = Q.top();}
        int u = tmp.id;
        for (int i = base[u];i;i = nxt[i]){
            int v = now[i];
            if (d[v] > d[u] + va[i]){
                d[v] = d[u] + va[i];
                Q.push(seg(v,d[v]));
            }
        }
        vis[u] = 1;
        Q.pop();
    }
}


int main(){
}
```

拓扑排序

```
/***********************************************************************
*****
    > File Name: 拓扑排序.cpp
    > Author:
    > Mail:
    > Created Time: 2015年12月09日 星期三 13时03分50秒

***********************************************************************
***/

voi topo()
{
      priority_queue<int, vector<int>, greater<int> > que;
      for(int i=1; i<=N; i++)
         if(indegree[i] == 0) que.push(i);
      int c = 1;
      while(!que.empty())
       {
         int v = que.top(); que.pop();
         printf("%d%c", v, c==N?'\n':' ');
          c++;
         for(int i=1; i<=N; i++)
       {
           if(Map[v][i])
          {
             indegree[i]--;
             if(indegree[i]==0) que.push(i);

          }

       }

   }

}
```

Floyd

```
/**********************************************************************
*****
    > File Name: Floyd.cpp
    > Author:
    > Mail:
    > Created Time: 2015年12月09日 星期三 13时03分06秒

**********************************************************************
***/

int n, d[maxn][maxn];

 void floyd()
 {
        for(int k=1; k<=n; k++)
            for(int i=1; i<=n; i++)
            for(int j=1; j<=n; j++)
            d[i][j] = min(d[i][j], d[i][k]+d[k][j]);

 }
```

判断是否有环

```
/**********************************************************************
*****
    > File Name: 判断是否有环.cpp
    > Author:
    > Mail:
    > Created Time: 2015年12月09日 星期三 13时07分08秒

**********************************************************************
***/

int G[maxn][maxn]
int c[maxn];
```

```
bool dfs(int u)
{
       c[u] = -1;
       for(int v=1; v<=n; v++) if(G[u][v])
    {
              if(c[v] < 0) return true;
              if(!c[v] && dfs(v)) return true;

    }
       c[u] = 1;
       return false;

}

 bool pan()
 {
        memset(c, 0, sizeof(c));
        for(int u=1; u<=n; u++) if(!c[u])
            if(dfs(u)) return true;
        return false;

 }
```

一些注意点

```
/*****************************************************************************
   > File Name: 一些注意点.cpp
   > Author:
   > Mail:
   > Created Time: 2015年12月09日 星期三 13时13分47秒

****************************************************************************/
```

做图论时候应该注意的事情

首先应该看看有没有重边和自环

分析图的定点数为 0 等小情况是的答案

1、用 int64 存储信息

2、似乎 edge 要开 200000

3、n=0 或 1 时输出 0

4、确认你的 spfa 没写错，访问下一个节点前先对当前节点置 v[x]=true（环！），最后置为 false
5、最后检验时要记得从 i=2 时开始检验
6、inf 要是 1<<61 左右，太大会爆，太小不行
7、双向边，重边
做 MST 的题的时候应该考虑一下 MST 的存在性

字符串

匹配 KMP

```
//KMP
typedef struct PString{
    char s[N];

    int p[N],len;

    int OnCreate(){
        len = strlen(s);
        int j = -1; p[0] = -1;
        for (int i = 1;i < len;i++){
            if (s[i] != s[j + 1] && j != -1) j = p[j];
            if (s[i] == s[j + 1]) j++;
            p[i] = j;
        }
    }

    int kmp(char pa[]){
        int lenx = strlen(pa);
        int j = -1;
        for (int i = 0;i < lenx;i++){
            if (pa[i] != s[j + 1] && j != -1) j = p[j];
            if (pa[i] == s[j + 1]) j++;
            if (j == len - 1) return 1;
        }
        return 0;
    }

}PString;
```

回文字符串 Manacher

```
//回文字串 Manacher
void manacher(){
    int m = 2 * n + 1;
    for (int i = 1;i <= n;i++){
        a[i << 1] = ch[i];
        a[i << 1 | 1] = '#';
    }
    a[0] = '+'; a[1] = '#'; a[m+1] = '-';
    int mx = 0,id;
    for (int i = 1;i <= m;i++){
        if (mx >= i) p[i] = min(mx - i,p[2*id-i]);
        else p[i] = 0;
        for (;a[i+p[i]+1] == a[i - p[i]];p[i]++);
        if (p[i]+i > mx) id = i,mx = p[i]+1;
    }
}
```

后缀自动机 SAM

```
//后缀自动机


int last = 1,tot = 1;
void add(int w){
    int p = ++tot,x = last,r,q;
    ml[last = p] = ml[x] + 1;
    for (;x && !son[x][w];x = pre[x]) son[x][w] = p;
    if (!x) {pre[p] = 1; return;}
    if (ml[x] + 1 == ml[q = son[x][w]]) {pre[p] = q; return;}
    pre[r = ++tot] = pre[q];
    memcpy(son[r],son[q],sizeof son[r]);
    ml[r] = ml[x] + 1;.
    pre[p] = pre[q] = r;
    for (;x && son[x][w] == q;x = pre[x]) son[x][w] = r;
}
```

哈希

```
/*
    BKDR_HASH 是一种可动态变换的 Hash 函数
    可以直接溢出
*/

typedef unsigned long long ull;

ull seed = 137;

int bkdr_hash(){
    int len = s.length();
    ull ans = 0;
    for (int i = 1;i < s.length();i++)
        ans = ans * seed + s[i];
}
```

AC 自动机

```
//AC 自动机
int trie[N][27],siz[N],tot = 0;

int ins(string s){
    int now = 1,c,len = strlen(s);
    for (int i = 0;i < len;i++){
        c = s[i] - 'A';
        if (trie[now][c]) now = trie[now][c];
        else now = a[now][c] = ++tot;
    }
    siz[now]++;
}

int Q[N],fail[N];

int acmatch(){
    int h = 1,t = 1,now;
    Q[1] = 1; fail[1] = 0;
    for (int i = 0;i < 26;i++) trie[0][i] = 1;
    while (t <= w){
        now = Q[++tot];
        for (int i = 0;i < 26;i++){
            if (!trie[now][i]) continue;
            int k = fail[now];
            while (!trie[now][i]) k = fail[k];
            fail[trie[now][i]] = trie[k][i];
            Q[++t] = trie[now][i];
        }
    }
}
```

几何

Pick 定理

```
//pick 定理
#include <iostream>
#include <cstring>
#include <cmath>

using namespace std;

typedef struct line{point a,b;}line;

typedef struct point{int x,y}point;

point p[101],o;

point sub(point a,point b){
    point t;
    t.x = a.x - b.x;
    t.y = a.x - b.y;
    return t;
}


int cross(point a,point b){
    return a.x * b.y - a.y * b.x;
}

int gcd(int x,int y){
    return y == 0?x:gcd(y,x%y);
}

int area(point a,point b,point c){
    return cross(sub(b,a),sub(c,a));
}

int calc(point a,point b){
    int dx = abs(a.x - b.x),dy = (a.y - b.y);
    return gcd(dx,dy);
}

/*
    S = a + b/2 - 1
    S 表示面积
    a 为内部点
    b 为边上的点
*/
```

凸包

```
//凸包


//注意凸包的判重和退化

struct point
{
    double x,y;
};

double det(point a,point b,point c,point d)
{
    double x1=b.x-a.x,y1=b.y-a.y;
    double x2=d.x-c.x,y2=d.y-c.y;
    return x1*y2-x2*y1;
}
```

```
double dot(point a,point b,point c,point d)
{
    double x1=b.x-a.x,y1=b.y-a.y;
    double x2=d.x-c.x,y2=d.y-c.y;
    return x1*x2+y1*y2;
}
double dist(point a,point b)
{
    double xx=(a.x-b.x)*(a.x-b.x);
    double yy=(a.y-b.y)*(a.y-b.y);
    return sqrt(xx+yy);
}
point ps[100000+5];
int N;
bool cmp_x(const point &a,const point &b)
{
    if(a.x!=b.x)
    {
        return a.x<b.x;
    }
    else{
        return a.y<b.y;
    }
}
vector<point> convex_hull(point *ps,int n)
{

    sort(ps,ps+N,cmp_x);
    int k=0;
    vector<point>qs(2*n);
    for(int i=0;i<n;i++)
    {
        while(k>1&&(det(qs[k-2],qs[k-1],qs[k-1],ps[i])<=0))
            k--;

        qs[k++]=ps[i];
    }
    for(int i=n-2,t=k;i>=0;i--)
    {
        while(k>t&&(det(qs[k-2],qs[k-1],qs[k-1],ps[i])<=0))
            k--;

        qs[k++]=ps[i];
    }
    qs.resize(k-1);
    return qs;
}
double area(point a,point b,point c)
{
    double ans=det(b,a,b,c);
    return fabs(ans)/2;
}
void solve()
{
    vector<point >qs=convex_hull(ps,N);



}
//旋转卡壳法

int i=0,j=0;

        int n=qs.size();
        for(int k=0;k<n;k++)
        {
            if(!cmp_x(qs[i],qs[k])) i=k;
            if(cmp_x(qs[j],qs[k])) j=k;
        }
        int si=i,sj=j;
        while(i!=sj||j!=si)
        {
            //double xa=qs[i].x,ya=qs[i].y,xb=qs[j].x,yb=qs[j].y;
            //double f1=check(x1,y1,x2,y2,xa,ya);
            //double f2=check(x1,y1,x2,y2,xb,yb);
            //if(f1*f2<=0)
            //{flag=0;break;}
            if(det(qs[i],qs[(i+1)%n],qs[j],qs[(j+1)%n])<0)
            i=(i+1)%n;
            else
            j=(j+1)%n;
        }
```

其他。。。

```
//foundation of Geometry
//some funtion refere to kuangbin
//take out! eps should be 1e-8
const double EPS = 1e-8
const double PI = acos(-1.0);
int sgn(double x)
{
    if(fabs(x)<EPS) return 0;
    if(x<0) return -1;
    return 1;
}

struct point
{
    double x,y;
    point(){};
    point (double x0,double y0)
    {
        x=x0,y=y0;
    }
    point operator - (const point &b)const
    {
        return point(x-b.x,y-b.y);
    }
    point operator + (const point &b)const
    {
        return point(x+b.x,y+b.y);
    }
    double operator * (const point &b) const
    {
        return x*b.x+y*b.y;
    }
    double operator ^ (const point &b) const
    {
        return x*b.y-y*b.x;
    }

    void transXY(double xita)//旋转西塔角度
    {
        double tx=x,ty=y;
        x = tx*cos(B)-ty*sin(B);
        y = tx*sin(B)+ty*cos(B);
    }


};

struct  line
{
    point s,e;
    line(){};
    line(point a,point b)
    {
        s=a,e=b;
    }
    pair<int ,point> operator &(const line &b)const//直线相交,0 重合, 1
平行, 2 返回交点
    {
        point res=s;
        if(sgn((s-e)^(b.s-b.e))==0)
        {
            if(sgn((s-b.e)^(b.s-b.e))==0)
                return make_pair(0,res);
            else
                return make_pair(1,res);
        }
        double t = ((s-b.s)^(b.s-b.e))/((s-e)^(b.s-b.e));
        res.x += (e.x-s.x)*t;
        res.y += (e.y-s.y)*t;
        return make_pair(2,res);
    }



};

struct  circle
{
    point heart;
    double r;

};

double dist(point a,point b)
{
    double xx=(a.x-b.x),yy=(a.y-b.y);
    return sqrt(xx*xx+yy*yy);
}

//*判断线段相交
bool inter(line l1,line l2)
{
    return
    max(l1.s.x,l1.e.x) >= min(l2.s.x,l2.e.x) &&
    max(l2.s.x,l2.e.x) >= min(l1.s.x,l1.e.x) &&
    max(l1.s.y,l1.e.y) >= min(l2.s.y,l2.e.y) &&
    max(l2.s.y,l2.e.y) >= min(l1.s.y,l1.e.y) &&
    sgn((l2.s-l1.e)^(l1.s-l1.e))*sgn((l2.e-l1.e)^(l1.s-l1.e)) <= 0 &&
    sgn((l1.s-l2.e)^(l2.s-l2.e))*sgn((l1.e-l2.e)^(l2.s-l2.e)) <= 0;
}

bool Seg_inter_line(line l1,line l2)//直线 l1 与线段 l2 是否相交
{
    return sgn((l2.s-l1.e)^(l1.s-l1.e))*sgn((l2.e-l1.e)^(l1.s-
l1.e))<=0;
}

point point_to_line(point p,line l)//点到直线距离
{
    point ret;
    double t=(l.e-l.s)*(p-l.s)/dist(l.s,l.e);
    ret.x=l.s.x+(l.e.x-l.s.x)*t;
    ret.y=l.s.y+(l.e.y-l.s.y)*t;
    return ret;
}

point point_to_segline(point p,line l) //点到线段距离
{
    point ret;
    double t=(l.e-l.s)*(p-l.s);
    t=fabs(t)/dist(l.s,l.e)/dist(l.s,l.e);
    if(t>=0&&t<=1)
    {
        ret.x=l.s.x+(l.e.x-l.s.x)*t;
        ret.y=l.s.y+(l.e.y-l.s.y)*t;
    }
    else
    {
        if(dist(p,l.s)<dist(p,l.e))
            ret=l.s;
        else
            ret=l.e;
    }
    return ret;
}


double areaMulti(point p[],int n)//计算多边形
{
    double ans=0;
    for (int i = 1; i < n-2; ++i)
    {
        /* code */
        double temp=(p[i+1]-p[i])^(p[0]-p[i]);
        temp/=2;
        ans+=fabs(temp);
    }
    return ans;
}

bool onSeg(point p,line l)
{
    return
    sgn(((l.s-p)^(l.e-p))==0)&&
    sgn(((p.x-l.s.x)*(p.x-l.e.x))<=0)&&
    sgn(((p.y-l.s.y)*(p.y-l.e.y))<=0);
}
/*
点是否在凸边形内，-1 外，0 上个，1 内
*/
int inConvexPoly(point a,point p[],int n)
{
    for(int i = 0;i < n;i++)
    {
        if(sgn((p[i]-a)^(p[(i+1)%n]-a))<0) return -1;
        else if(onSeg(a,line(p[i],p[(i+1)%n]))) return 0;

    }
    return 1;
}

/*
点是否在多边形内，是 1，否-1，边界上 0
*/
```

```
int inPoly(point p,point poly[],int n)
{
    int cnt=0;
    line ray,side;
    ray.s=p;
    ray.e.y=p.y;
    ray.e.x=-(double)INF;
    for (int i = 0; i < n; ++i)
    {
        /* code */
        side.s=poly[i];
        side.e=poly[(i+1)%n];
        if (onSeg(p,side))
        {
            return 0;
        }

        if(sgn(side.s.y-side.e.y)==0)
            continue;

        if(onSeg(side.s,ray))
        {
            if(sgn(side.s.y-side.e.y)>0)
                cnt++;
        }
        else if(onSeg(side.e,ray))
        {
            if(sgn(side.e.y-side.s.y)>0)
                cnt++;
        }
        else if(inter(ray,side))
            cnt++;


    }
    if(cnt%2==1)
        return 1;
    else
        return -1;
}

/*
判断是否是凸边形
*/
```

```
bool inConvex(point poly[],int n)
{
    bool s[3];
    memset(s,0,sizeof(s));
    for(int i=0;i<n;i++)
    {
        s[sgn((poly[(i+1)%n]-poly[i])^(poly[(i+2)%n]-poly[i]))+1] = 1;
        if(s[0]&&s[2]) return 0;
    }
    return 1;
}

//两个圆的公共部分面积
double Area_of_overlap(Point c1,double r1,Point c2,double r2)
{
    double d = dist(c1,c2);
    if(r1 + r2 < d + eps)return 0;
    if(d < fabs(r1 - r2) + eps)
    {
        double r = min(r1,r2);
        return PI*r*r;
    }
    double x = (d*d + r1*r1 - r2*r2)/(2*d);
    double t1 = acos(x / r1);
    double t2 = acos((d - x)/r2);
    return r1*r1*t1 + r2*r2*t2 - d*r1*sin(t1);
}

/*
三点求圆心坐标（三角形外心）
*/

point waixin(point a,point b,point c)
{
    double a1 = b.x - a.x, b1 = b.y - a.y, c1 = (a1*a1 + b1*b1)/2;
    double a2 = c.x - a.x, b2 = c.y - a.y, c2 = (a2*a2 + b2*b2)/2;
    double d = a1*b2 - a2*b1;
    return point(a.x+(c1*b2-c2*b1)/d,a.y+(a1*c2-a2*c1)/d);
}


/*
半平面交 （From UESTC）
直线左边代表有效区域
```

```
*/

bool HPIcmp(line a,line b)
{
    if(fabs(a.k-b.k)>eps) return a.k<b.k;
    return ((a.s-b.s)^(b.e-b.s))<0;
}
line Q[110];
void HPI(line l[],int n,point res[],int &resn)
{
    int tol=n;
    sort(l,l+n,HPIcmp);
    tol=1;
    for(int i=1;i<n;i++)
        if(fabs(l[i].k-l[i-1].k)>EPS)
            l[tol++]=l[i];
    int head=0,tail=1;
    Q[0] = l[0];
    Q[1] = l[1];
    resn=0;
    for(int i=2;i<tol;i++)
    {
        if(fabs((Q[tail].e-Q[tail].s)^(Q[tail-1].e-Q[tail-1].s)) < eps
||
         fabs((Q[head].e-Q[head].s)^(Q[head+1].e-Q[head+1].s)) < eps)
            return;
        while(head < tail && (((Q[tail]&Q[tail-1]) -
line[i].s)^(line[i].e-line[i].s)) > eps)
            tail--;
        while(head < tail && (((Q[head]&Q[head+1]) -
line[i].s)^(line[i].e-line[i].s)) > eps)
            head++;
        Q[++tail] = line[i];
    }
    while(head < tail && (((Q[tail]&Q[tail-1]) -
Q[head].s)^(Q[head].e-Q[head].s)) > eps)
        tail--;
    while(head < tail && (((Q[head]&Q[head-1]) -
Q[tail].s)^(Q[tail].e-Q[tail].e)) > eps)
        head++;
    if(tail <= head + 1)
        return;
    for(int i = head; i < tail; i++)
        res[resn++] = Q[i]&Q[i+1];
    if(head < tail - 1)
        res[resn++] = Q[head]&Q[tail];

}
```

This document was created with Win2PDF available at http://www.win2pdf.com.
The unregistered version of Win2PDF is for evaluation or non-commercial use only.
This page will not be added after purchasing Win2PDF.