ICPC TEMPLATE

AC mountain(ac 山岳险天下)

2019 年 11 月 6 日

目录

1 一切的开始

1.1 宏定义

by 杜教

```
#include <bits/stdc++.h>
using namespace std;
#define rep(i,a,n) for (int i=a;i<n;i++)//注意范围[a,n)
#define per(i,a,n) for (int i=n-1;i>=a;i--)//注意范围[a,n-1]
#define pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define fi first
#define se second
#define SZ(x) ((int)(x).size())
typedef vector<int> VI;
typedef long long ll;
typedef pair<int,int> PII;
mt19937 mrand(random_device{}());
const ll mod=1000000007;
int rnd(int x) { return mrand() % x;}
ll powmod(ll a,ll b) {ll res=1;a%=mod; assert(b>=0);
    for(;b;b>>=1){if(b&1)res=res*a%mod;a=a*a%mod;}return res;}
ll gcd(ll a,ll b) { return b?gcd(b,a%b):a;}
// head ----------------------------------------------------------------------
```

- CMakeLists.txt (for CLion)

```
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -O2 -Dzerol -Wall")
```

- HDU Assert Patch

```
#ifdef ONLINE_JUDGE
#define assert(condition) if (!(condition)) { int x = 1, y = 0; cout << x / y << endl; }
#endif
```

1.2 快速读

```
inline char nc() {
    static char buf[100000], *p1 = buf, *p2 = buf;
    return p1 == p2 && (p2 = (p1 = buf) + fread(buf, 1, 100000, stdin), p1 == p2) ? EOF : *p1++;
}
template <typename T>
bool rn(T& v) {
    static char ch;
    while (ch != EOF && !isdigit(ch)) ch = nc();
    if (ch == EOF) return false;
    for (v = 0; isdigit(ch); ch = nc())
        v = v * 10 + ch - '0';
    return true;
}

template <typename T>
void o(T p) {
    static int stk[70], tp;
    if (p == 0) { putchar('0'); return; }
    if (p < 0) { p = -p; putchar('-'); }
    while (p) stk[++tp] = p % 10, p /= 10;
    while (tp) putchar(stk[tp--] + '0');
}
```

- 需要初始化
- 需要一次读入
- 不支持负数

```
const int MAXS = 100 * 1024 * 1024;
char buf[MAXS];
template<typename T>
inline bool read(T& x) {
    static char* p = buf;
    x = 0;
    while (*p && !isdigit(*p)) ++p;
    if (!*p) return false;
    while (isdigit(*p)) x = x * 10 + *p++ - 48;
```

```
    return true;
}

fread(buf, 1, MAXS, stdin);
```

1.3 对拍

```
#!/usr/bin/env bash
g++ -o r main.cpp -O2 -std=c++11
g++ -o std std.cpp -O2 -std=c++11
while true; do
    python gen.py > in
    ./std < in > stdout
    ./r < in > out
    if test $? -ne 0; then
        exit 0
    fi
    if diff stdout out; then
        printf "AC\n"
    else
        printf "GG\n"
        exit 0
    fi
done
```

- 快速编译运行

```
#!/bin/bash
g++ $1.cpp -o $1 -O2 -std=c++14 -Wall -Dzerol -g
if $? -eq 0; then
    ./$1
fi
```

2 数据结构

2.1 BIT

```
struct Bit
{
    vector<int> a;
    int sz;
    void init(int n)
    {
```

```
        sz=n+5;
        for(int i=1;i<=n+5;i++)
            a.push_back(0);
    }
    int lowbit(int x)
    {
        return x&(-x);
    }
    int query(int x)
    {
        int ans = 0;
        for(;x;x-=lowbit(x))ans+=a[x];
        return ans;
    }
    void update(int x,int v)
    {
        for(;x<sz;x+=lowbit(x))
            a[x]+=v;
    }
}bit;
```

3 图论

3.1 最短路

3.1.1 floyd

```
for (int k = 1; k <= n; k++) {
  for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= n; j++) {
      f[i][j] = min(f[i][j], f[i][k] + f[k][j]);
    }
  }
}
```

- 找最小环

```
int val[maxn + 1][maxn + 1]; // 原图的邻接矩阵
int floyd(const int &n) {
  static int dis[maxn + 1][maxn + 1]; // 最短路矩阵
  for (int i = 1; i <= n; ++i)
    for (int j = 1; j <= n; ++j) dis[i][j] = val[i][j]; // 初始化最短路矩阵
  int ans = inf;
  for (int k = 1; k <= n; ++k) {
```

```
    for (int i = 1; i < k; ++i)
      for (int j = 1; j < i; ++j)
        ans = std::min(ans, dis[i][j] + val[i][k] + val[k][j]); // 更新答案
    for (int i = 1; i <= n; ++i)
      for (int j = 1; j <= n; ++j)
        dis[i][j] = std::min(
            dis[i][j], dis[i][k] + dis[k][j]); // 正常的 floyd 更新最短路矩阵
  }
  return ans;
}
```

- 利用 floyd 的 dp 思路求解

```
int dp[maxn][maxn][maxn];
int w[maxn];
int s[maxn];
bool cmp(int a,int b)
{
    return w[a] < w[b];
}
rep(i,1,n+1)
{
    rep(j,1,n+1)
    {
       scanf("%d",&dp[i][j][0]);
        rep(k,1,n+1)
        {
            dp[i][j][k] = 1e9;
        }
     }
    s[i] = i;
}
sort(s+1,s+n+1,cmp);
rep(k,1,n+1)
{
    rep(i,1,n+1)
    {
        rep(j,1,n+1)
        {
            dp[i][j][k] = min(dp[i][j][k-1],dp[i][s[k]][k-1]+dp[s[k]][j][k-1]);
        }
    }
}
```

- 传递闭包已知一个有向图中任意两点之间是否有连边，要求判断任意两点是否连通。

```
for (int k = 1; k <= n; k++)
  for (int i = 1; i <= n; i++)
    if (f[i][k]) f[i] = f[i] & f[k];
```

3.2 网络流

- dinic

```
const int maxn = 4e3+100;
const int maxm = 1e5+100;
const int inf = 0x7f7f7f7f;

typedef struct Dinic
{
    typedef struct Edge
    {
        int u,v,w,nxt;
    } Edge;
    int head[maxn],hcnt;
    int dep[maxn];
    int cur[maxn];
    Edge e[maxm];
    int S,T,N;
    void init()
    {
        memset(head,-1,sizeof head);
        hcnt = 0;
        S = T = N = 0;
    }
    void adde(int u,int v,int w)
    {
        e[hcnt].u = u,e[hcnt].v = v,e[hcnt].w = w;
        e[hcnt].nxt = head[u];head[u] = hcnt++;
        e[hcnt].u = v,e[hcnt].v = u,e[hcnt].w = 0;
        e[hcnt].nxt = head[v];head[v] = hcnt++;
    }
    int bfs()
    {
        rep(i,0,N)
        {
            dep[i] = inf;
        }
        queue<int> q;
        q.push(S); dep[S] = 0;
        while(!q.empty())
```

```
38          {
39              int u = q.front();q.pop();
40              for(int i = head[u];~i;i = e[i].nxt)
41              {
42                  int v = e[i].v,w = e[i].w;
43                  if(w > 0 && dep[u] + 1 < dep[v])
44                  {
45                      dep[v] = dep[u] + 1;
46                      if(v == T)
47                      {
48                          return 1;
49                      }
50                      q.push(v);
51                  }
52              }
53          }
54          return dep[T] != inf;
55      }
56      int dfs(int s,int mw)
57      {
58          if(s == T) return mw;
59          for(int i = cur[s];~i;i=e[i].nxt)
60          {
61              cur[s] = i;
62              int v = e[i].v,w=e[i].w;
63              if(w <= 0 || dep[v] != dep[s] + 1)
64              {
65                  continue;
66              }
67              int cw = dfs(v,min(w,mw));
68              if(cw <= 0)
69                  continue;
70              e[i].w -= cw;
71              e[i^1].w += cw;
72              return cw;
73          }
74          return 0;
75      }
76      ll dinic()
77      {
78          ll res = 0;
79          while(bfs())
80          {
81              rep(i,0,N)
82              {
83                  cur[i] = head[i];
```

```
84          }
85          while(int d = dfs(S,inf))
86          {
87              res += 1ll * d;
88          }
89      }
90      return res;
91  }
92  } Dinic;
```

- MCMF1

```
1   namespace mincostflow {
2       const int INF=0x3f3f3f3f;
3       struct node {
4           int to; int cap,cost; int rev;
5           node(int t=0,int c=0,int _c=0,int n=0):
6               to(t),cap(c),cost(_c),rev(n) {};
7       }; vector<node> edge[maxn];
8       void addedge(int from,int to,int cap,int cost) {
9           edge[from].push_back(node(to,cap,cost,edge[to].size()));
10          edge[to].push_back(node(from,0,-cost,edge[from].size()-1));
11      }
12      int dis[maxn];
13      bool mark[maxn];
14      void spfa(int s,int t,int n) {
15          memset(dis+1,0x3f,n*sizeof(int));
16          memset(mark+1,0,n*sizeof(bool));
17          static int Q[maxn],ST,ED;
18          dis[s]=0; ST=ED=0; Q[ED++]=s;
19          while (ST!=ED) {
20              int v=Q[ST]; mark[v]=0;
21              if ((++ST)==maxn) ST=0;
22              for (node &e:edge[v]) {
23                  if (e.cap>0&&dis[e.to]>dis[v]+e.cost) {
24                      dis[e.to]=dis[v]+e.cost;
25                      if (!mark[e.to]) {
26                          if (ST==ED||dis[Q[ST]]<=dis[e.to]) {
27                              Q[ED]=e.to,mark[e.to]=1;
28                              if ((++ED)==maxn) ED=0;
29                          } else {
30                              if ((--ST)<0) ST+=maxn;
31                              Q[ST]=e.to,mark[e.to]=1;
32                          }
33                      }
34                  }
```

```
            }
        }
    } int cur[maxn];
    int dfs(int x,int t,int flow) {
        if (x==t||!flow) return flow;
        int ret=0; mark[x]=1;
        for (int &i=cur[x];i<(int)edge[x].size();i++) {
            node &e=edge[x][i];
            if (!mark[e.to]&&e.cap) {
                if (dis[x]+e.cost==dis[e.to]) {
                    int f=dfs(e.to,t,min(flow,e.cap));
                    e.cap-=f; edge[e.to][e.rev].cap+=f;
                    ret+=f; flow-=f;
                    if (flow==0) break;
                }
            }
        } mark[x]=0;
        return ret;
    }
    pair<int,int> min_costflow(int s,int t,int n) {
        int ret=0,ans=0;
        int flow = INF;
        while (flow) {
            spfa(s,t,n); if (dis[t]==INF) break;
            memset(cur+1,0,n*sizeof(int));
            int len=dis[t],f;
            while ((f=dfs(s,t,flow))>0)
                ret+=f,ans+=len*f,flow-=f;
        } return make_pair(ret,ans);//最大流，最小费用
    }
    void init(int n) {
        int i; for (int i = 1; i <= n; i++) edge[i].clear();
    }
}
```

- MCMF2

```
const int maxn = 2e4+10;
namespace MCMF {
    const int inf=0x3f3f3f3f;
    struct Edge {
        int to; int cap,cost; int rev;
        Edge(int t=0,int c=0,int _c=0,int n=0):
            to(t),cap(c),cost(_c),rev(n) {};
    };
    vector<Edge> edge[maxn];
```

```
10  void adde(int from,int to,int cap,int cost)
11  {
12      edge[from].push_back(Edge(to,cap,cost,edge[to].size()));
13      edge[to].push_back(Edge(from,0,-cost,edge[from].size()-1));
14  }
15
16  int dis[maxn];
17  bool mark[maxn];
18
19  void spfa(int s,int t,int n)
20  {
21      memset(dis,0x3f,sizeof dis);
22      memset(mark,0,sizeof mark);
23      static int Q[maxn],ST,ED;
24      dis[s]=0; ST=ED=0; Q[ED++]=s;
25      while (ST!=ED)
26      {
27          int v=Q[ST]; mark[v]=0;
28          if ((++ST)==maxn) ST=0;
29          for (Edge &e:edge[v])
30          {
31              if (e.cap>0&&dis[e.to]>dis[v]+e.cost)
32              {
33                  dis[e.to]=dis[v]+e.cost;
34                  if (!mark[e.to])
35                  {
36                      if (ST==ED||dis[Q[ST]]<=dis[e.to])
37                      {
38                          Q[ED]=e.to,mark[e.to]=1;
39                          if ((++ED)==maxn) ED=0;
40                      }
41                      else
42                      {
43                          if ((--ST)<0) ST+=maxn;
44                          Q[ST]=e.to,mark[e.to]=1;
45                      }
46                  }
47              }
48          }
49      }
50  }
51  int cur[maxn];
52  int dfs(int x,int t,int flow)
53  {
54      if (x==t||!flow) return flow;
55      int ret=0; mark[x]=1;
```

```
        for (int &i=cur[x];i<(int)edge[x].size();i++)
        {
            Edge &e=edge[x][i];
            if (!mark[e.to]&&e.cap)
            {
                if (dis[x]+e.cost==dis[e.to])
                {
                    int f=dfs(e.to,t,min(flow,e.cap));
                    e.cap-=f; edge[e.to][e.rev].cap+=f;
                    ret+=f; flow-=f;
                    if (flow==0) break;
                }
            }
        }
        mark[x]=0;
        return ret;
    }
    pair<int,ll> mc(int s,int t,int n)
    {
        int ret=0;
        ll ans=0;
        int flow = inf;
        while(flow)
        {
            spfa(s,t,n); if (dis[t]==inf) break;
            memset(cur,0,sizeof cur);
            int len=dis[t],f;
            while ((f=dfs(s,t,flow))>0)
                ret+=f,ans+=(ll)len*(ll)f,flow-=f;
        }
        return make_pair(ret,ans);//最大流，最小费用
    }
    void init(int n)
    {
        for(int i = 1; i <= n; i++) edge[i].clear();
    }
}
```

4 数学

4.1 BSGS

- 用于计算方程 $ a^x = b (% p)$

```
int BSGS(int a,int b,int p)
```

```
{
    map<int,int > hash;
    b %= p;
    int t = (int)sqrt(p) + 1;
    for(int j = 0; j < t; j ++){
        int val = 1ll * b * fpow(a,j,p) % p;
        hash[val] = j;
    }
    a = fpow(a,t,p);
    if(!a) return b == 0 ? 1 : -1;
    for(int i = 0; i <= t;++i){
        int val = fpow(a,i,p);
        int j = hash.find(val) == hash.end() ? -1 : hash[val];
        if(j >= 0 && i * t - j > 0) return i * t - j;
    }
    return -1;
}
```

4.2 扩展中国剩余定理

4.2.1 值在 ___int128 以内

```
typedef __int128 ll;
void exgcd(ll a,ll b,ll &x,ll &y)
{
    if(!b){
        x = 1;y = 0;
        return ;
    }
    exgcd(b,a % b,y,x);
    y -= a / b * x;
}
ll inv(ll a,ll p)
{
    ll x,y;
    exgcd(a,p,x,y);
    if(x < 0) x += p;
    return x;
}
ll ex_crt()
{
    bool flag = true;
    ll m1,m2,c1,c2,tmp;
    for(int i = 2;i <= n;++i){//m[i]为模,c[i]为余数
        m1 = m[i - 1];m2 = m[i];
        c1 = c[i - 1];c2 = c[i];
```

```
        tmp = gcd(m1,m2);
        if((c2 - c1) % tmp != 0) {
            flag = false;
            return -1;
        }
        m[i] = m1 / tmp * m2;
        c[i] = inv(m1 / tmp,m2 / tmp) * (c2 - c1) / tmp % (m2 / tmp) * m1 + c1;
        c[i] = (c[i] % m[i] + m[i]) % m[i];
    }
    return c[n];
}
```

4.2.2 值在 ___int128 以外

```
import sys
def exgcd(a, b):
    if b == 0:
        return (a, 1, 0)
    q = a // b
    g, y, x = exgcd(b, a - q * b)
    y -= q * x
    return (g, x, y)

def gcd(a, b):
    if b == 0:
        return a
    else:
        return gcd(b, a % b)

N, lim = map(int, input().split())
M, ans = map(int, input().split())
f = 0
for i in range(0, N - 1):
    m, r = map(int, input().split())
    A = M
    B = m
    C = ((r - ans) % m)
    if (ans - r) % gcd(M, m) != 0:
        f = 1
    g, x, y = exgcd(A, B)
    x = (x * C // g) % (B // g)
    ans += x * M
    M *= B // g
    ans %= M

```

```python
if ans < 0:
    ans += M
if f == 1:
    ans = -1
    print(ans)
else:
    print(ans)
# print ans
```

4.3 扩展卢卡斯

```cpp
ll n, m, p;
ll exgcd(ll a, ll b, ll &x, ll &y){
    if(!b){
        x = 1, y = 0;
        return a;
    }
    ll res = exgcd(b, a % b, x, y);
    ll t = x;
    x = y;
    y = t - a / b * y;
    return res;
}

ll fpow(ll a, ll b, ll mod){
    ll res = 1;
    while(b){
        if(b & 1) res = (res * a) % mod;
        a = (a * a) % mod;
        b >>= 1;
    }
    return res;
}

ll inv(ll a, ll p){
    ll x, y;
    exgcd(a, p, x, y);
    if(x + p > p) return x;
    return x + p;
}

inline ll crt(ll n, ll mod)
{
    return n * (p / mod) % p * inv(p / mod, mod) % p;
}
```

```

ll fac(ll n, ll p, ll k){   //k = p^x
      if(!n) return 1;
      ll ans = 1;
    for(int i = 2; i <= k;++i)
        if(i % p) ans = ans * i % k;
    ans = fpow(ans, n / k, k);
    for(int i = 2; i <= n % k;++i)
        if(i % p) ans = ans * i % k;
    return ans * fac(n / p, p, k) % k;
}

ll C(ll n, ll m, ll p, ll k){ //k = p^x
      if(n < m) return 0;
    ll a = fac(n,p,k), b = fac(m,p,k), c = fac(n - m,p,k);
    ll cnt = 0;
    for(ll i = p; i <= n; i *= p) cnt += n / i;
    for(ll i = p; i <= m; i *= p) cnt -= m / i;
    for(ll i = p; i <= n-m; i *= p) cnt -= (n - m) / i;
    return a*inv(b, k) % k * inv(c, k) % k * fpow(p, cnt, k) % k ;
}

ll ex_Lucas()
{
    ll t = p, ans =i;
          t /= i;
       }
       ans = (ans + crt(C(n, m, i, tmp), tmp))%p;
    }
    if(t > 1) ans = (ans + crt(C(n, m, t, t), t))%p;
    return ans%p;
}
```

4.4 fft

- 多项式快速计算

```
const double PI = acos(-1.0);
struct Complex{
    double x, y; // 实部和虚部 x+yi
    Complex(double _x = 0.0, double _y = 0.0) {
        x = _x;
        y = _y;
    }
    Complex operator-(const Complex &b) const {
        return Complex(x - b.x, y - b.y);
```

```
    }
    Complex operator+(const Complex &b) const {
        return Complex(x + b.x, y + b.y);
    }
    Complex operator*(const Complex &b) const {
        return Complex(x * b.x - y * b.y, x * b.y + y * b.x);
    }
};
/*
* 进行 FFT 和 IFFT 前的反转变换。
* 位置 i 和 (i 二进制反转后位置)互换
* len 必须去 2 的幂
*/
void change(Complex y[], int len){
    for (int i = 1, j = len / 2,k;i < len - 1;++i){
        if (i < j)swap(y[i], y[j]);
        //交换互为小标反转的元素,i<j 保证交换一次
        //i 做正常的+1,j 左反转类型的+1,始终保持 i 和 j 是反转的
        k = len / 2;
        while(j >= k){
            j -= k;
            k >>= 1;
        }
        if (j < k) j += k;
    }
}
/*
* 做 FFT
* len 必须为 2^k 形式,
* on==1 时是 DFT,on==-1 时是 IDFT
*/
void fft(Complex y[], int len, int on) {
    change(y, len);
    for (int h = 2; h <= len; h <<= 1) {
        Complex wn(cos(-on * 2 * PI / h), sin(-on * 2 * PI / h));
        for (int j = 0; j < len; j += h) {
            Complex w(1, 0);
            for (int k = j; k < j + h / 2; k++) {
                Complex u = y[k];
                Complex t = w * y[k + h / 2];
                y[k] = u + t;
                y[k + h / 2] = u - t;
                w = w * wn;
            }
        }
    }
```

```
    if (on == -1)
        for (int i = 0; i < len; i++) y[i].x /= len;
}
```

4.5 fwt

- 计算集合卷积

```
void fwt(ll *a, int n)
{
    for(int d = 1;d < n;d <<= 1){
        for(int m = d << 1,i = 0;i < n;i += m){
            for(int j = 0;j < d;++j){
                ll x = a[i + j],y = a[i + j + d];
                a[i + j] = (x + y),a[i + j + d] = (x - y);
                //xor:a[i + j] = x + y,a[i + j + d] = (x - y + mod) % mod;
                //and:a[i + j] = x + y;
                //or:a[i + j + d] = x + y;
            }
}
void ufwt(ll *a, int n)
{
    for(int d = 1;d < n;d <<= 1){
        for(int m = d << 1,i = 0;i < n;i += m){
            for(int j = 0;j < d;++j){
                ll x = a[i + j],y = a[i + j + d];
                a[i + j] = 1LL * (x + y) / 2,a[i + j + d] = (1LL * (x - y) / 2);
                //xor:a[i + j] = (x + y) / 2,a[i + j + d] = (x - y) / 2;
                //and:a[i + j] = x - y;
                //or:a[i + j + d] = y - x;
            }
}
void work(ll *a,ll *b, int n)
{
    fwt(a,n);
    fwt(b,n);
    for(int i = 0;i < n;++i) a[i] *= b[i];
    ufwt(a,n);
}
```

4.6 polya

- m 种颜色，n 个长度

```

```

```
const int N = 1e5 + 7;
int euler(int n){
    int ans = n;
    for(int i = 1;i <= cnt && prime[i] * prime[i] <= n;++i)
        if(n % prime[i] == 0){
            ans -= ans / prime[i];
            while(n % prime[i] == 0) n /= prime[i];
        }
    if(n > 1) ans -= ans / n;
    return ans;
}
int n,m;
int main()
{
    getprime();
    while(scanf("%d%d",&m,&n) && (n || m)){
        ll ans = 0;
        for(int i = 1; i * i <= n;++i){
            if(n % i == 0)
                ans += i * i == n ? euler(i) * fpow(m,n / i) : euler(i) * fpow(m,n / i) + euler(n / i)
                    * fpow(m,i);
        }
        ans += n & 1 ? n * fpow(m,n / 2 + 1) : (fpow(m,n >> 1) + fpow(m,n / 2 + 1)) * (n >> 1);
        printf("%lld\n",ans / (n << 1LL));
    }
    return 0;
}
```

4.7 大素数

- 包含分解，判断过程为 $O(n^{\frac{1}{4}})$

```
typedef long long ll;
int pri[] = {2,3,5,7,11,13,17,19,23,29,31};
ll minfacotr;
ll mult(ll a, ll b, ll mod) // 大数乘法
{
    ll ans = 0;
    while(b)
    {
        if (b&1)
        {
            ans+= a;
            if (ans >= mod)
                ans -= mod;
        }
```

```
        b >>= 1;
        a <<= 1;
        if (a >= mod)
            a-= mod;
    }
    return ans;
}
ll qpow(ll x, ll n, ll mod)
{
    ll ans = 1;
    while(n)
    {
        if (n&1)
            ans = mult(ans, x, mod);
        x = mult(x, x, mod);
        n >>= 1;
    }
    return ans;
}
bool wintness(ll n, ll a)
{
    ll p = qpow(a, n-1, n);
    if(p != 1)
        return false;
    ll s = n - 1;
    while(!(s&1)&& p == 1)
    {
        s >>= 1;
        p = qpow(a, s, n);
    }
    if (p == 1 || p == n - 1)
        return true;typedef long long ll;
int pri[] = {2,3,5,7,11,13,17,19,23,29,31};
ll minfacotr;
ll mult(ll a, ll b, ll mod) // 大数乘法
{
    ll ans = 0;
    while(b)
    {
        if (b&1)
        {
            ans+= a;
            if (ans >= mod)
                ans -= mod;
        }
        b >>= 1;
```

```
61          a <<= 1;
62          if (a >= mod)
63              a-= mod;
64      }
65      return ans;
66  }
67  ll qpow(ll x, ll n, ll mod)
68  {
69      ll ans = 1;
70      while(n)
71      {
72          if (n&1)
73              ans = mult(ans, x, mod);
74          x = mult(x, x, mod);
75          n >>= 1;
76      }
77      return ans;
78  }
79  bool wintness(ll n, ll a)
80  {
81      ll p = qpow(a, n-1, n);
82      if(p != 1)
83          return false;
84      ll s = n - 1;
85      while(!(s&1)&& p =
86      return false;
87  }
88  bool miller_rabin(ll n) // 判断素数
89  {
90      if (n < 32)
91      {
92          for(int i = 0; i < 11; ++i)
93              if(n == pri[i])
94                  return true;
95          return false;
96      }
97      for(int i = 0 ; i < 10; ++i)
98          if(!wintness(n,pri[i]))
99              return false;
100     return true;
101 }
102 ll gcd(ll a, ll b)
103 {
104     return b ? gcd(b, a % b) : a;
105 }
106 ll pollard_rho(ll n, ll c) // Pollard_rho 大数分解
```

```
{
    ll x= rand() % n, y = x, i = 1, k = 2, d;
    while(1)
    {
        i++;
        x =(mult(x, x, n) + c) % n;
        d = gcd(y-x+n,n);
        if(d > 1 && d < n)
            return d;
        if(y == x)
            return n;
        if (i == k)
        {
            y = x;
            k <<= 1;
        }
    }
}
ll fac[maxn];
int cnt = 0;
void findfactor(ll n)
{
    if(miller_rabin(n))
    {
        minfacotr = min(minfacotr,n);// 求最小素数因子
        fac[++cnt] = n;//储存素数
        return ;
    }
    ll p = n;
    while(p >= n)
        p = pollard_rho(n, rand() % (n - 1) + 1);
    findfactor(p);
    findfactor(n / p);
}
```

4.8 线性基

```
struct L_B{
    ll d[64],p[64];
    int cnt;
    void init(){// 初始化
        memset(d,0,sizeof d);
        memset(p,0,sizeof p); // 单位基
        cnt = 0; // 单位基中1的数量
    }
```

```
    bool insert(ll val){ // 插入 如果x的1数位不存在则插入val, 存在则赋值为0
        for(int i = 63;i >= 0;--i)
            if(val & (1LL << i)){
                if(!d[i]){
                    d[i] = val;
                    break;
                }
                val ^= d[i];
            }
        return val > 0;
    }
    ll query_max(){ // 查询最大异或和
        ll ret = 0;
        for(int i = 63;i >= 0;--i)
            if((ret ^ d[i]) > ret)
                ret ^= d[i];
        return ret;j >= 0;--j)
                if(d[i] & (1LL << j)) d[i] ^= d[j];
        for(int i = 0;i <= 63;++i)
            if(d[i]) p[cnt++] = d[i];
    }
    ll kth_query(ll k){ // 查询第k小值
        int ret = 0;
        if(k >= (1LL << cnt))
            return -1;
        for(int i = 63;i >= 0;--i)
            if(k & (1LL << i)) ret ^= p[i];
        return ret;
    }
}
L_B merge(const L_B &n1,const L_B &n2) // 暴力合并两个线性基
{
    L_B ret = n1;
    for(int i = 63;i >= 0;--i)
        if(n2.d[i]) ret.insert(n1.d[i]);
    return ret;
}
```

4.9 洲阁筛

4.10 计算素数 k 次幂前缀和

```
const int N = 1e6 + 7;
ll n,k,sqrtn;
int m;
```

```
ll a[N << 1],cnt;
int pos1[N],pos2[N];
ull g[N << 1]; // 构造函数
int pri[N],pcnt;
bool ispri[N];
void get_pri()
{
  for(int i = 2;i < N;++i){
    if(!ispri[i]) pri[++pcnt] = i;
    for(int j = 1;j <= pcnt && i * pri[j] < N;++j){
      ispri[i * pri[j]] = 1;
      if(i % pri[j] == 0) break;
    }
  }
}

inline int get_pos(ll x)
{
  return x <= sqrtn ? pos1[x] : pos2[n / x];
}
void Discretization(){ // 离散化处理
  for(ll i = 1,j;i <= n;i = j + 1){
    a[++cnt] = n / i;
    j = n / (n / i);
  }
  reverse(a + 1,a + 1 + cnt);
  for(int i = 1;i <= cnt;++i)
    if(a[i] <= sqrtn) pos1[a[i]] = i;
    else pos2[n / a[i]] = i;
}
void calc_g() // 预处理构造函数
{
  for(int i = 1;i <= cnt;++i) g[i] = a[i] - 1;
  for(int i = 1;i <= m;++i)
    for(int j = cnt;j >= 1 && a[j] >= pri[i] * pri[i];--j)
      g[j] -= g[get_pos(a[j] / pri[i])] - g[get_pos(pri[i] - 1)];
}
ull calc_h(ll i,ll j){ // 计算递归函数
  if(i <= 1) return 0;
  ull sum = 0;
  int res;
  for(res = j;res <= m && pri[res] * pri[res] <= i;++res)
    for(ll o = pri[res],e = 1;o <= i;o *= pri[res],++e)
      sum += (ull)(e * k + 1) * (calc_h(i / o,res + 1) + 1);
  if(pri[res - 1] <= i)
    sum += (ull)(k + 1) * (g[get_pos(i)] - g[get_pos(pri[res - 1])]);
```

```
    return sum;
}
int t;
int main()
{
    get_pri();
    scanf("%lld%lld",&n,&k);
    sqrtn = (ll)sqrt(n);cnt = 0;
    m = upper_bound(pri + 1,pri + 1 + pcnt,sqrtn) - pri - 1;
    Discretization();
    calc_g();
    ull ans = (ull)(k + 1) * (g[get_pos(n)] - m);
    for(int i = 1;i <= m;++i)
        for(ll j = pri[i],e = 1;j <= n;j *= pri[i],e++)
            ans += (ull)(e * k + 1) * (calc_h(n / j,i + 1) + 1);
    ans++;
    printf("%ull\n",ans);
    return 0;
}
```

4.11 自然幂数和

```
void init()
{
//预处理组合数
    for(int i = 0; i < N;++i){
        C[i][0] = C[i][i] = 1;
        if (i == 0)
            continue;
        for (int j = 1; j < i;++j)
            C[i][j] = (C[i - 1][j] % mod + C[i - 1][j - 1] % mod) % mod;
    }
//预处理逆元
    inv[1] = 1;
    for (int i = 2; i < N; ++i)
        inv[i] = (mod - mod / i) * inv[mod % i] % mod;
//预处理伯努利数
    B[0] = 1;
    for (int i = 1; i < N; ++i)
    {
        ll ans = 0;
        if (i == N - 1)
            break;
        for (int j = 0; j < i; ++j)
        {
```

```
            ans += C[i + 1][j] * B[j];
            ans %= mod;
        }
        ans *= -inv[i + 1];
        ans = (ans % mod + mod) % mod;
        B[i] = ans;
    }
}
ll Work(int k)
{
    ll ans = inv[k + 1];
    ll sum = 0;
    for(int i=1; i<=k+1; ++i)
    {
        sum += C[k + 1][i] * tmp[i] % mod * B[k + 1 - i] % mod;
        sum %= mod;
    }
    ans *= sum;
    ans %= mod;
    return ans;
}
int main()
{
    int t;
    init();
    scanf("%d", &t);
    while(t--)
    {
        int k;
        scanf("%lld %d", &n, &k);
        n %= mod;
        tmp[0] = 1;
        for(int i = 1; i < N; ++i)
            tmp[i] = tmp[i - 1] * (n + 1) % mod;
        printf("%lld\n", Work(k));
    }
    return 0;
}
```

4.12 组合数打表

```
ll f[N],inv[N];
inline ll C(ll a,ll b){return a < 0 || b < 0 || a < b ? 0 : f[a] * inv[b] % mod * inv[a - b] % mod;}
void get_table(int index)
{
```

```
    f[0] = inv[0] = 1;
    for(int i = 1;i <= index;++i) f[i] = f[i - 1] * i % mod;
    inv[index] = fpow(f[index],mod - 2);
    for(;index;--index)
        inv[index - 1] = inv[index] * index % mod;
}
```

4.13 杜教筛

```
const int N = 4e6 + 7;
const int mod = 1e9 + 7;
int prime[N],cnt;
ll g[N],phi[N];
bool isprime[N];
ll n,m;
inline ll gao(ll x)
{
    x %= mod;
    return (x + 1) % mod * x % mod * inv2 % mod;
}
void get_phi() // 预处理phi或者Mobius,n^(2/3)的前缀和
{
   phi[1] = 1;
   for(int i = 2;i < N;++i){
       if(!isprime[i]){
            prime[++cnt] = i;
            phi[i] = i - 1;
        }
       for(int j = 1;j <= cnt && i * prime[j] < N;++j)
       {
          isprime[i * prime[j]] = 1;
          if(i % prime[j] == 0){
             phi[i * prime[j]] = phi[i] * prime[j];
             break;
          }
          phi[i * prime[j]] = phi[i] * (prime[j] - 1);
       }
   }
   for(int i = 1;i < N;++i) g[i] = (g[i - 1] + phi[i] * i % mod) % mod;
}

inline ll cal(ll x) //
{
    x %= mod;
    return x * (2 * x + 1) % mod * (x + 1) % mod * inv6 % mod;
```

```
}
map<ll ,ll>mp; // 存已到达情况
ll solve(ll pos) // 递归
{
    if(pos < N) return g[pos];
    if(mp[pos]) return mp[pos];
    ll res = cal(pos),last;
    for(ll i = 2;i <= pos;i = last + 1){
        last = pos / (pos / i);
        res = ((res - (gao(last) - gao(i - 1)) * solve(pos / i) % mod) % mod + mod) % mod;
    }
    mp[pos] = res;
    return res;
}
ll work(ll pos)
{
    ll last,ans = 0;
    for(ll i = 1;i <= pos;i = last + 1){
        last = pos / (pos / i);
        ans = (ans + (last - i + 1) % mod * solve(pos / i) % mod) % mod + mod;
        ans %= mod;
    }
    return (ans + pos) % mod * inv2 % mod;
}
int main()
{
    get_phi();
    scanf("%lld%lld",&n,&m);
    printf("%lld\n",((work(m) - work(n - 1)) % mod + mod) % mod);
    return 0;
}
```

4.14 二次剩余

4.14.1 解 $x^2 = a(mod\, p)$

```
struct T{
    ll p, d;
};
ll w;
//二次域乘法
T multi_er(T a, T b, ll m)
{
    T ans;
    ans.p = (a.p * b.p % m + a.d * b.d % m * w % m) % m;
    ans.d = (a.p * b.d % m + a.d * b.p % m) % m;
```

```
    return ans;
}
//二次域上快速幂
T power(T a, ll b, ll m)
{
    T ans;
    ans.p = 1;
    ans.d = 0;
    while(b){
        if(b & 1)
        ans = multi_er(ans, a, m);
        b >>= 1;
        a = multi_er(a, a, m);
    }
    return ans;
}
//求勒让德符号
ll Legendre(ll a, ll p)
{
    return qpow(a, (p-1)>>1, p);
}
ll mod(ll a, ll m)
{
    a %= m;
    if(a < 0) a += m;
    return a;
}
ll Solve(ll n,ll p)
{
    if(p == 2) return 1;
    if (Legendre(n, p) + 1 == p)
        return -1;
    ll a = -1, t;
    while(true) {
        a = rand() % p;
}
int main()
{
    int t;
    scanf("%d", &t);
    while(t--){
        int n, p;
        scanf("%d %d",&n,&p);
        n %= p;
        int a = Solve(n, p);//x * x = n % p
        if(a == -1) {
```

```
57            puts("No root");
58            continue;
59        }
60        int b = p - a;
61        if(a > b) swap(a, b);
62        if(a == b)
63            printf("%d\n",a);
64        else printf("%d %d\n",a,b);
65    }
66    return 0;
67 }
```

4.15 高精度

```
1  struct BigInteger {
2      typedef unsigned long long ll;
3
4      static const int BASE = 100000000;
5      static const int WIDTH = 8;
6      vector<int> s;
7
8      BigInteger& clean(){while(!s.back()&&s.size()>1)s.pop_back(); return *this;}
9      BigInteger(ll num = 0) {*this = num;}
10     BigInteger(string s) {*this = s;}
11     BigInteger& operator = (long long num) {
12         s.clear();
13         do {
14             s.push_back(num % BASE);
15             num /= BASE;
16         } while (num > 0);
17         return *this;
18     }
19     BigInteger& operator = (const string& str) {
20         s.clear();
21         int x, len = (str.length() - 1) / WIDTH + 1;
22         for (int i = 0; i < len; i++) {
23             int end = str.length() - i*WIDTH;
24             int start = max(0, end - WIDTH);
25             sscanf(str.substr(start,end-start).c_str(), "%d", &x);
26             s.push_back(x);
27         }
28         return (*this).clean();
29     }
30
31     BigInteger operator + (const BigInteger& b) const {
```

```
32          BigInteger c; c.s.clear();
33          for (int i = 0, g = 0; ; i++) {
34              if (g == 0 && i >= s.size() && i >= b.s.size()) break;
35              int x = g;
36              if (i < s.size()) x += s[i];
37              if (i < b.s.size()) x += b.s[i];
38              c.s.push_back(x % BASE);
39              g = x / BASE;
40          }
41          return c;
42      }
43      BigInteger operator - (const BigInteger& b) const {
44          assert(b <= *this); // 减数不能大于被减数
45          BigInteger c; c.s.clear();
46          for (int i = 0, g = 0; ; i++) {
47              if (g == 0 && i >= s.size() && i >= b.s.size()) break;
48              int x = s[i] + g;
49              if (i < b.s.size()) x -= b.s[i];
50              if (x < 0) {g = -1; x += BASE;} else g = 0;
51              c.s.push_back(x);
52          }
53          return c.clean();
54      }
55      BigInteger operator * (const BigInteger& b) const {
56          int i, j; ll g;
57          vector<ll> v(s.size()+b.s.size(), 0);
58          BigInteger c; c.s.clear();
59          for(i=0;i<s.size();i++) for(j=0;j<b.s.size();j++) v[i+j]+=ll(s[i])*b.s[j];
60          for (i = 0, g = 0; ; i++) {
61              if (g ==0 && i >= v.size()) break;
62              ll x = v[i] + g;
63              c.s.push_back(x % BASE);
64              g = x / BASE;
65          }
66          return c.clean();
67      }
68      BigInteger operator / (const BigInteger& b) const {
69          assert(b > 0); // 除数必须大于0
70          BigInteger c = *this;    // 商:主要是让c.s和(*this).s的vector一样大
71          BigInteger m;            // 余数:初始化为0
72          for (int i = s.size()-1; i >= 0; i--) {
73              m = m*BASE + s[i];
74              c.s[i] = bsearch(b, m);
75              m -= b*c.s[i];
76          }
77          return c.clean();
```

```
78      }
79      BigInteger operator % (const BigInteger& b) const { //方法与除法相同
80          BigInteger c = *this;
81          BigInteger m;
82          for (int i = s.size()-1; i >= 0; i--) {
83              m = m*BASE + s[i];
84              c.s[i] = bsearch(b, m);
85              m -= b*c.s[i];
86          }
87          return m;
88      }
89
90      int bsearch(const BigInteger& b, const BigInteger& m) const{
91          int L = 0, R = BASE-1, x;
92          while (1) {
93              x = (L+R)>>1;
94              if (b*x<=m) {if (b*(x+1)>m) return x; else L = x;}
95              else R = x;
96          }
97      }
98      BigInteger& operator += (const BigInteger& b) {*this = *this + b; return *this;}
99      BigInteger& operator -= (const BigInteger& b) {*this = *this - b; return *this;}
100     BigInteger& operator *= (const BigInteger& b) {*this = *this * b; return *this;}
101     BigInteger& operator /= (const BigInteger& b) {*this = *this / b; return *this;}
102     BigInteger& operator %= (const BigInteger& b) {*this = *this % b; return *this;}
103
104     bool operator < (const BigInteger& b) const {
105         if (s.size() != b.s.size()) return s.size() < b.s.size();
106         for (int i = s.size()-1; i >= 0; i--)
107             if (s[i] != b.s[i]) return s[i] < b.s[i];
108         return false;
109     }
110     bool operator >(const BigInteger& b) const{return b < *this;}
111     bool operator<=(const BigInteger& b) const{return !(b < *this);}
112     bool operator>=(const BigInteger& b) const{return !(*this < b);}
113     bool operator!=(const BigInteger& b) const{return b < *this || *this < b;}
114     bool operator==(const BigInteger& b) const{return !(b < *this) && !(b > *this);}
115 };
116
117 ostream& operator << (ostream& out, const BigInteger& x) {
118     out << x.s.back();
119     for (int i = x.s.size()-2; i >= 0; i--) {
120         char buf[20];
121         sprintf(buf, "%08d", x.s[i]);
122         for (int j = 0; j < strlen(buf); j++) out << buf[j];
123     }
```

```
    return out;
}

istream& operator >> (istream& in, BigInteger& x) {
    string s;
    if (!(in >> s)) return in;
    x = s;
    return in;
}
```

4.16 反素数

4.16.1 求小于 n 并且因子个数最多的那个数

```
int pri[]= {2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61};
ll n, ans, tmp;
const ll INF = 0x3f3f3f3f;
void dfs(int step, ll sum, ll num) //num 为因子个数
{
    if(step == 16) return;
    if(num > tmp){
        ans = sum;
        tmp = num;
    }
    if(sum > n) return ;

    if(num == tmp && sum < ans)//因子个数一样,选取较小的那个数
        ans = sum;
    for(int i = 1; i <= 63; ++i){
        if(n / pri[step] < sum) break;
        dfs(step + 1, sum *= pri[step], num * (i + 1));
    }
}

//求一个最小的正整数,使得它的因子个数为 n
int p[16] = {2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53};
void dfs(int dept,ull tmp,int num)
{
    if(num > n) return;
    if(num == n && ans > tmp)
        ans = tmp;
    for(int i=1; i <= 63;++i){
        if(ans / p[dept] < tmp)
            break;
        dfs(dept+1,tmp *= p[dept],num * (i + 1));
    }
```

```
}
```

4.17 高斯消元

```
const int N = 307;
int x[N],a[N][N];// x[N]解集, a[N][N]系数
bool free_x[N];
int gcd(int a,int b){return b ? gcd(b,a % b) : a;}
int lcm(int a,int b){return a / gcd(b,a % b) * b;}
int Gauss(int equ,int var)//equ个方程, var个变元
{
    int free_x_num,i,j,row,max_r,col;//row表示行,col表示列,max_r表示列最大的行,free_x_num变元数量
    int free_index,LCM,ta,tb,temp;// free_index变元下标
    for(i = 0;i <= var;++i){
        x[i] = 0;
        free_x[i] = true;//第i个元素是否是变元
    }
    for(row = 0,col = 0;row < equ && col < var;++row,++col){
        max_r = row;
        //找到col最大的行, 进行交换（除法时减小误差）
        for(i = row + 1;i < equ;++i) if(abs(a[i][col]) > abs(a[max_r][col])) max_r = i;
        //与第row行交换
        if(max_r != row) for(j = row;j < var + 1;++j) swap(a[row][j],a[max_r][j]);
        if(a[row][col]==0){
            //说明该col列第row行以下全是0了, 则处理当前行的下一列.
            row--;
            continue;
        }
        for(i = row + 1;i < equ;++i)//枚举被删行
            if(a[i][col]){
                LCM = lcm(abs(a[i][col]),abs(a[row][col]));
                ta = LCM / abs(a[i][col]);
                tb = LCM / abs(a[row][col]);
                if(a[i][col] * a[row][col] < 0)tb = -tb;//异号的情况是相加
                for(j = col;j < var + 1;++j)
                    a[i][j] = a[i][j] * ta - a[row][j] * tb;
            }
        /*求解小数解,防止溢出
        for(int i = row + 1; i < equ; ++i)
            if(fabs(a[i][col]) > eps){
                double t1 = a[i][col]/a[row][col];
                for(int j = col; j <= var;++j) a[i][j] -= a[row][j] * t1;
            }*/
    }
    for (i = row;i < equ;++i) if(a[i][col]) return -1; // 无解
```

```
    if (row < var){// 多解
        for(i = row - 1;i >= 0;--i){
            free_x_num = 0;
            for (j = 0;j < var;++j)
                if(a[i][j] && free_x[j]) free_x_num++,free_index = j;
            if (free_x_num > 1) continue; // 无法求解出确定的变元.
            temp = a[i][var];
            for (j = 0;j < var;++j) if (a[i][j] && j != free_index) temp -= a[i][j] * x[j];
            x[free_index] = temp / a[i][free_index]; //求出该变元.
            free_x[free_index] = 0; //该变元是确定的.
        }
        return var - row; //自由变元有 var - row 个.
    }
    for (i = var - 1; i >= 0;--i){// 唯一解
        temp = a[i][var];
        for (j = i + 1;j < var;++j)
            if (a[i][j]) temp -= a[i][j] * x[j];
        if (temp % a[i][i]) return -2; // 说明有浮点数解,但无整数解.
        x[i] = temp / a[i][i];
    }
    return 0;
}
```

4.18 高斯消元解异或方程

```
int Guass_xor(int equ,int var)
{
    int row,col;
    for(row = 0, col = 0; row < equ && col < var; ++row, ++col){
        int maxr = row;
        for(int i = row; i < equ; ++i)
            if(a[i][col] > a[maxr][col]) maxr = i;
            if(maxr != row) for(int j = col; j <= var; ++j) swap(a[maxr][j],a[row][j]);
            if(!a[row][col]){
                row--;
                continue;
            }
            for(int i = row + 1; i < equ; ++i)
                if(a[i][col])
                    for(int j = col; j <= var; ++j) a[i][j] ^= a[row][j];
    }
    for(int i = row; i < equ; ++i) if(a[i][col]) return -1;//无可行解
    if(row < var) // 存在多解
            for(int i = row - 1; i >= 0; --i){
                int num = 0, index;
```

```
            for(int j = 0; j < var; ++j)
                if(a[i][j] && vis[j]) num++, index = j;
            if(num > 1) continue;
            for(int j = 0; j < var; ++j)
                if(j != index) x[index] ^= (a[i][j] && x[j]);
            vis[index] = false;
        }
    for(int i = var - 1; i >= 0; --i){ //唯一解
        x[i] = a[i][var];
        for(int j = i + 1; j < var; ++j) x[i] ^=(a[i][j] && x[j]);
    }
    return 0;
}
```

4.19 矩阵类（快速幂）

```
int sz;
struct Matrix{
    ll a[10][10];
    Matrix(){
        for(int i = 0;i < sz;++i)
            for(int j = 0;j < sz;++j)
                if(i == j) a[i][j] = 1;
                else a[i][j] = 0;
    }
    void init(){
        memset(a,0,sizeof a);
    }
    void show(){
        for(int i = 0;i < sz;++i)
            for(int j = 0;j < sz;++j)
                dd(i),dd(j),de(a[i][j]);
    }
    void Relation_matrix(ll *s){
        init();
        for(int i = 0;i < sz;++i){
            if(i + 1 < sz) a[i][i + 1] = 1;
            a[sz - 1][i] = s[i];
        }
    }
    Matrix operator *(const Matrix& tmp) const{
        Matrix res;
        res.init();
        for(int i = 0;i < sz;++i)
            for(int j = 0;j < sz;++j)
```

```
            for(int k = 0;k < sz;++k)
                res.a[i][j] = (res.a[i][j] + a[i][k] * tmp.a[k][j] % mod) % mod;
        return res;
    }
    Matrix operator +(Matrix tmp){
        for(int i = 0;i < sz;++i)
            for(int j = 0;j < sz;++j)
                a[i][j] += tmp.a[i][j];
    }
    friend Matrix fpow(Matrix a,ll b){
        Matrix res;
        while(b){
            if(b & 1) res = res * a;
            a = a * a;
            b >>= 1;
        }
        return res;
    }
    ll get_fib(){
        Matrix tmp;
        tmp.init();
        for(int i = 0;i < sz;++i)
            tmp.a[i][0] = 1;
        Matrix ans = *this;
        ans = ans * tmp;
        return ans.a[0][0];
    }
};
```

4.20 类欧几里得

- 求解 $f(a,b,c,n)=\sum_{i=0}^{n}\lfloor\frac{a*i+b}{c}\rfloor$

```
ll f(ll a,ll b,ll c,ll n)
{
    if(a == 0)
        return (b / c) * (n + 1) % mod;
    if(a >= c)
        return (n * (n + 1) / 2 % mod * (a / c) % mod + f(a % c,b,c,n)) % mod;
    if(b >= c)
        return ((b / c) * (n + 1) % mod + f(a,b % c,c,n)) % mod;
    //ll m = (a * n + b) / c % mod;
    ll m = (n / c * a + (n % c * a + b) / c);
    return ((m % mod) * (n % mod) % mod - f(c,c - b - 1,a,m - 1) + mod) % mod;
}
```

4.21 牛顿迭代

```
#define f(x) () //原函数
#define f2(x) () //导函数
double Newton(double x){
    for(int i = 0;i < 100;++i){
        x = x - f(x) / f2(x); //牛顿迭代法
        if(fabs(f(x)) < eps) return x;
    }
    return -1;
}
```

4.22 母函数

- n 个人分成任意组, 每组人数必须为素数, 有多少种分法,n $<$ 150;

```
void solve()
{
    memset(a, 0, sizeof a);
    for(int i = 0; i <= 160; i += 2)
        a[i] = 1;
    memset(b, 0, sizeof b);
    for(int i = 1; pri[i] <= 150;++i){
        for(int j = 0; j <= 150;++j)
            for(int k = 0;k + j <= 150;k += pri[i])
                b[k + j] += a[j];
        for(int j = 0; j <= 150; ++j){
            a[j] = b[j];
            b[j] = 0;
        }
    }
}
```

4.22.1 莫比乌斯反演

- 求有多少个数对 (x,y) , 满足 a x b , c y d , 且 gcd(x,y) = 1

```
void Mobius()
{
    cnt = 0;
    mu[1] = 1;sum[1] = 1;
    for(int i = 2; i < maxn; ++i){
        if(!vis[i]){
            prime[++cnt] = i;
            mu[i] = -1;
```

```
        }
        for(int j = 1; j <= cnt; ++j){
            if(i * prime[j] >= maxn) break;
            vis[i * prime[j]] = true;
            if(i % prime[j]== 0){
                mu[i * prime[j]] = 0;
                break;
            }
            mu[i * prime[j]] = -mu[i];
        }
        sum[i] = sum[i-1] + mu[i];
    }
}
ll solve(int b, int d)
{
    if(b > d) swap(b, d);
    ll ans = 0,pos;
    for(int i = 1; i <= b; i = pos+1){
        pos = min(b / (b / i),d / (d / i));
        ans += 1LL * (b / i) * (d / i) * (sum[pos] - sum[i - 1]);
    }
    return ans;
}
```

4.23 斐波那契广义循环节

- $f(n) = af(n-1) + bf(n-2) \quad f(1) = c \quad f(2) = d$

```
const int N = 2;
const ll mod = 1000000007;

ll fac[2][505];
int cnt,ct;

ll pri[6] = {2, 3, 7, 109, 167, 500000003};
ll num[6] = {4, 2, 1, 2, 1, 1};

struct Matrix
{
    ll m[N][N];
} ;

Matrix A;
Matrix I = {1, 0, 0, 1};

Matrix multi(Matrix a,Matrix b)
```

```
{
    Matrix c;
    for(int i=0; i<N; i++)
    {
        for(int j=0; j<N; j++)
        {
            c.m[i][j] =0;
            for(int k=0; k<N; k++)
            {
                c.m[i][j] += a.m[i][k] * b.m[k][j];
                c.m[i][j] %= mod;
            }
        }
    }
    return c;
}

Matrix power(Matrix A,ll n)
{
    Matrix ans = I, p = A;
    while(n)
    {
        if(n & 1)
        {
            ans = multi(ans,p);
            n--;
        }
        n >>= 1;
        p = multi(p,p);
    }
    return ans;
}

ll quick_mod(ll a,ll b)
{
    ll ans = 1;
    a %= mod;
    while(b)
    {
        if(b & 1)
        {
            ans = ans * a % mod;
            b--;
        }
        b >>= 1;
        a = a * a % mod;
```

```
    }
    return ans;
}

ll Legendre(ll a,ll p)
{
    ll t = quick_mod(a,(p-1)>>1);
    if(t == 1) return 1;
    return -1;
}

void dfs(int dept,ll product = 1)
{
    if(dept == cnt)
    {
        fac[1][ct++] = product;
        return;
    }
    for(int i=0; i<=num[dept]; i++)
    {
        dfs(dept+1,product);
        product *= pri[dept];
    }
}

bool OK(Matrix A,ll n)
{
    Matrix ans = power(A,n);
    return ans.m[0][0] == 1 && ans.m[0][1] == 0 &&
           ans.m[1][0] == 0 && ans.m[1][1] == 1;
}

int main()
{
    fac[0][0] = 1;
    fac[0][1] = 2;
    fac[0][2] = 500000003;
    fac[0][3] = 1000000006;
    ll a,b,c,d;
    while(cin>>a>>b>>c>>d)
    {
        ll t = a * a + 4 * b;
        A.m[0][0] = a;
        A.m[0][1] = b;
        A.m[1][0] = 1;
        A.m[1][1] = 0;
```

```
        if(Legendre(t,mod) == 1)
        {
            for(int i=0; i<4; i++)
            {
                if(OK(A,fac[0][i]))
                {
                    cout<<fac[0][i]<<endl;
                    break;
                }
            }
        }
        else
        {
            ct = 0;
            cnt = 6;
            dfs(0,1);
            sort(fac[1],fac[1]+ct);
            for(int i=0;i<ct;i++)
            {
                if(OK(A,fac[1][i]))
                {
                    cout<<fac[1][i]<<endl;
                    break;
                }
            }
        }
    }
    return 0;
}
```

4.24 无名小定理

- 1. 设 $x = gcd(\sum_{i=1}^{n1} C(n,i))$
- 当 n 是素数是 x = n (根据 1.)
- 当 n 有多个素因子 x = 1(根据 1.)
- n 只有一个素因子，答案为该素因子 (根据 1.)
- 当 a > b , gcd(a,b) = 1, 那么 $ gcd(a^m − b^m, a^n − b^n) = a^{gcd(m,n)} − b^{gcd(n,m)}$
- $(n+1) * lcm(\sum_{i=0}^{n} C(n,i)) = lcm(\sum_{i=1}^{n+1} i)$
- 判断组合数 C(n,m) 的奇偶性, 当 n & m == m 为奇数, 反之就是偶数.
- 三角形求圆半径: 边长为 a, b, c 的三角形面积为 S, 则外接圆半径为 $a * b * c/(4S)$, 内切圆半径为 $2S/(a+b+c)$

4.25 自然常数

e= 2.71828182845904523536028747135266 24

4.26 欧拉常数

C = 0.57721566490153286060651209

4.27 错排公式

$D_1 = 0$
$D_2 = 1$
$D_n = (n-1)(D_{n-1} + D_{n-2})$

4.28 伯努利数

$ F_0 = 1, F_n = _{i=0}^{n-1}C(n,i)F_i$

4.29 自然幂数和

$\sum_{i=1}^{n} i^k = \frac{1}{k+1} \sum_{i=1}^{k+1} C(k+1,i) * F_{k+1-i} * (n+1)^i$

4.30 Catalan

1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012…
$h(n) = h(n-1)»(4»n2)/(n+1) h(n) = C(2n,n)/(n+1) (n = 0, 1, 2, ...)$
$h(n)=c(2n,n)-c(2n,n-1) (n=0,1,2,...) $

- 将 n + 2 边形沿弦切割成 n 个三角形的不同切割数
- n + 1 个数相乘, 给每两个元素加上括号的不同方法数
- n 个节点的不同形状的二叉树数
- 从 n * n 方格的左上角移动到右下角不升路径数

4.31 pick 定理

计算点阵中顶点在格点上的多边形面积公式, 该公式可以表示为: $2 * S = 2 * a + b - 2$, 其中 a 表示多边形内部的点数,b 表示多边形边界上的点数,S 表示多边形的面积.

4.32 正多边形外接圆半径

$R = \frac{a}{2sin(\frac{pi}{n})}$

4.33 Fibonacci

- $-F(n+m) = F(n+1)F(m) + F(n)F(m-1)$
- $F(n)^2 = (-1)^{n+1} + F(n-1)F(n+1)$
- 前 n 项有 $\lceil \frac{2n}{3} \rceil$ 个奇数项
- 如果 fib(k) 能被 x 整除，则 fib(k*i) 都可以被 x 整除。
- $ f(0)+f(1)+f(2)+…+f(n)=f(n+2)-1$
- $f(1) + f(3) + f(5) + \ + f(2n-1) = f(2n)$
- $f(2) + f(4) + f(6) + \ + f(2n) = f(2n+1) - 1$

- $[f(0)]^2+[f(1)]^2+ +[f(n)]^2=f(n)*f(n+1)$
- $f(0)-f(1)+f(2)- +(-1)^n$
- $f(n)=(-1)^n*[f(n+1)-f(n)]+1$
- $f(n+m)=f(n+1)*f(m)+f(n)*f(m-1)$
- $[f(n)]^2=(-1)^{n-1}+f(n-1)*f(n+1)*f(2n-1)=[f(n)]^2-[f(n-2)]^2$
- $3f(n)=f(n+2)+f(n-2)$
- \$f(2n-2m-2)[f(2n)+f(2n+2)]=f(2m+2)+f(4n-2m) [n>m>=-1, 且 n>=1] \$
- $Fn=\frac{1}{\sqrt{5}}*(\frac{1+\sqrt{5}}{2})-\frac{1}{\sqrt{5}}(\frac{1\sqrt{5}}{2})$

5 计算几何

5.1 处理平面内所有直线围成的所有多边形

```
const int MAXN=1e6+10;
const double eps=1e-8;
const double pi=acos(-1.0);
const ll INF=0x3f3f3f3f3f3f3f3f;

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

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

struct Point{
    double x,y;
    Point(){ x=0,y=0; }
    Point(double _x,double _y):x(_x),y(_y){}
    void input(){ scanf("%lf%lf",&x,&y); }
    void output(){ printf("%.2f %.2f\n",x,y); }
    friend istream &operator >>(istream &os,Point &b){
        os>>b.x>>b.y;
        return os;
    }
    friend ostream &operator <<(ostream &os,Point &b){
        os<<b.x<<' '<<b.y;
        return os;
    }
    bool operator ==(const Point &b)const{
        return (dcmp(x-b.x)==0&&dcmp(y-b.y)==0);
    }
    bool operator !=(const Point &b)const{
        return !((dcmp(x-b.x)==0&&dcmp(y-b.y)==0));
    }
```

```
33      bool operator <(const Point &b)const{
34          return (dcmp(x-b.x)==0? dcmp(y-b.y)<0 : x<b.x);
35      }
36      double operator ^(const Point &b)const{ //叉积
37          return x*b.y-y*b.x;
38      }
39      double operator *(const Point &b)const{ //点积
40          return x*b.x+y*b.y;
41      }
42      Point operator +(const Point &b)const{
43          return Point(x+b.x,y+b.y);
44      }
45      Point operator -(const Point &b)const{
46          return Point(x-b.x,y-b.y);
47      }
48      Point operator *(double a){
49          return Point(x*a,y*a);
50      }
51      Point operator /(double a){
52          return Point(x/a,y/a);
53      }
54      double len2(){ //长度平方
55          return sqr(x)+sqr(y);
56      }
57      double len(){ //长度
58          return sqrt(len2());
59      }
60      double polar(){ //向量的极角
61          return atan2(y,x); //返回与x轴正向夹角(-pi~pi]
62      }
63      Point change_len(double r){ //转化为长度为r的向量
64          double l=len();
65          if(dcmp(l)==0) return *this; //零向量
66          return Point(x*r/l,y*r/l);
67      }
68      Point rotate_left(){ //逆时针旋转90度
69          return Point(-y,x);
70      }
71      Point rotate_right(){ //顺时针旋转90度
72          return Point(y,-x);
73      }
74      Point rotate(Point p,double ang){ //绕点p逆时针旋转ang度
75          Point v=(*this)-p;
76          double c=cos(ang),s=sin(ang);
77          return Point(p.x+v.x*c-v.y*s,p.y+v.x*s+v.y*c);
78      }
```

```
79      Point normal(){  //单位化，逆时针旋转90°
80          return Point(-y/len(),x/len());
81      }
82  };
83
84  inline double cross(Point a,Point b){ //叉积
85      return a.x*b.y-a.y*b.x;
86  }
87
88  inline double dot(Point a,Point b){ //点积
89      return a.x*b.x+a.y*b.y;
90  }
91
92
93  double rad(Point a,Point b){ //两个向量的夹角
94      return fabs(atan2(fabs(cross(a,b)),dot(a,b)));
95  }
96
97  bool is_parallel(Point a,Point b){ //判断向量是否平行
98      double p=rad(a,b);
99      return dcmp(p)==0||dcmp(p-pi)==0;
100 }
101
102 struct Line{
103     Point s,e;
104     Line(){}
105     Line(Point _s,Point _e):s(_s),e(_e){} //两点确定直线
106     Line(Point p,double ang){ //一个点和斜率(弧度制)确定直线
107         s=p;
108         if(dcmp(ang-pi/2)==0){
109             e=s+Point(0,1);
110         }
111         else{
112             e=s+Point(1,tan(ang));
113         }
114     }
115     Line(double a,double b,double c){ //ax+by+c=0
116         if(dcmp(a)==0){
117             s=Point(0,-c/b);
118             e=Point(1,-c/b);
119         }
120         else if(dcmp(b)==0){
121             s=Point(-c/a,0);
122             e=Point(-c/a,1);
123         }
124         else{
```

```
125             s=Point(0,-c/b);
126             e=Point(1,(-c-a)/b);
127         }
128     }
129     void input(){
130         s.input();
131         e.input();
132     }
133     void adjust(){
134         if(e<s) swap(e,s);
135     }
136     double polar(){ //极角
137         return atan2(e.y-s.y,e.x-s.x); //返回与x轴正向夹角(-pi~pi]
138     }
139     double angle(){  //倾斜角
140         double k=atan2(e.y-s.y,e.x-s.x);
141         if(dcmp(k)<0) k+=pi;
142         if(dcmp(k-pi)==0) k-=pi;
143         return k;
144     }
145     Point operator &(const Line &b)const{ //求两直线交点
146         Point res=s;
147         double t=((s-b.s)^(b.s-b.e))/((s-e)^(b.s-b.e));
148         res.x+=(e.x-s.x)*t;
149         res.y+=(e.y-s.y)*t;
150         return res;
151     }
152 };
153
154 double polygon_area(vector<Point> p){ //多边形的有向面积，加上绝对值就是面积 正值表示输入点按照逆时针
      否则为顺时针
155     int n=p.size(); double area=0;
156     for(int i=1;i<n-1;i++) area+=cross(p[i]-p[0],p[i+1]-p[0]);
157     return fabs(area/2);
158 }
159
160 struct PSLG{  //平面直线图 处理平面内所有直线围成的所有多边形 传入直线交点之间的每条线段
161     struct Edge{
162         int from,to;
163         double ang;
164         Edge(){ ang=from=to=0; }
165         Edge(int s,int t,double a){ from=s,to=t,ang=a; }
166     };
167     int n,m,face_cnt; //平面个数 包括外面最大的多边形
168     double area[MAXN]; //每个多边形面积
169     Point point[MAXN]; //平面内所有的点
```

```
170     vector<Edge>edge;
171     vector<int>G[MAXN];
172     vector<vector<Point> >face;
173     int vis[2*MAXN],left[2*MAXN],pre[2*MAXN]; //left表示这条边的左侧属于哪个面
174     void Init(){
175         face.clear();
176         edge.clear();
177         for(int i=0;i<n;i++) G[i].clear();
178         n=m=0;
179     }
180     PSLG(){ Init(); }
181     void AddEdge(int from, int to){       //需要建立反向边帮助寻找下一条边
182       edge.pb(Edge(from,to,(point[to]-point[from]).polar()));
183       edge.pb(Edge(to,from,(point[from]-point[to]).polar()));
184       m=edge.size();
185       G[from].pb(m-2);
186       G[to].pb(m-1);
187     }
188     void Build(){
189         for(int u=0;u<n;u++){
190             int d=G[u].size();
191             for(int i=0;i<d;i++)
192                 for(int j=i+1;j<d;j++)
193                     if(edge[G[u][i]].ang>edge[G[u][j]].ang)
194                         swap(G[u][i],G[u][j]);
195             for(int i=0;i<d;i++) pre[G[u][(i+1)%d]]=G[u][i]; //从u出发的i条边顺时针旋转的第一条边是pre[i]
196         }
197         face_cnt=0; memset(vis,0,sizeof(vis));
198         for(int u=0;u<n;u++){
199             for(int i=0;i<G[u].size();i++){
200                 int e=G[u][i];
201                 if(!vis[e]){
202                     face_cnt++;
203                     vector<Point> polygon;
204                     while(1){
205                         vis[e]=1;
206                         left[e]=face_cnt;
207                         int from=edge[e].from;
208                         polygon.pb(point[from]);
209                         e=pre[e^1];       //逆时针旋转最多的一条边即为顺时针转动的第一条边
210                         if(e==G[u][i]) break;
211                     }
212                     face.pb(polygon);
213                 }
214             }
215         }
```

```
        for(int i=0;i<face_cnt;i++) area[i]=polygon_area(face[i]);
    }
    vector<pair<double,int> >tmp[MAXN];
    void Insert(Line *line,int m){
        for(int i=0;i<m;i++){
            for(int j=i+1;j<m;j++){
                if(!is_parallel(line[i].e-line[i].s,line[j].e-line[j].s)){
                    Point inter=line[i]&line[j];
                    point[n++]=inter;
                    tmp[i].pb({dot(inter-line[i].s,line[i].e-line[i].s),n-1});
                    tmp[j].pb({dot(inter-line[j].s,line[j].e-line[j].s),n-1});
                }
            }
            sort(tmp[i].begin(),tmp[i].end());
            for(int j=1;j<tmp[i].size();j++) AddEdge(tmp[i][j-1].se,tmp[i][j].se);
        }
        Build();
    }
}pslg;

Line line[MAXN];

int main(void){
    int n; scanf("%d",&n);
    for(int i=0;i<n;i++) line[i].input();
    pslg.Insert(line,n);
    sort(pslg.area,pslg.area+pslg.face_cnt);
    printf("%d %.6f %.6f\n",pslg.face_cnt-1,pslg.area[pslg.face_cnt-2],pslg.area[0]);
    int q; scanf("%d",&q);
    while(q--){
        int p; scanf("%d",&p);
        if(p>=pslg.face_cnt) puts("Invalid question");
        else   printf("%.6f\n",pslg.area[pslg.face_cnt-p-1]);
    }
    return 0;
}
```

6 字符串

6.1 kmp

- border

```
void get_fail(int f[],char s[])
{
```

```
    int j = f[0] = 0;
    int n = strlen(s);
    rep(i,1,n)
    {
        while(j && s[i] != s[j]) j = f[j-1];
        f[i] = j += s[i] == s[j];
    }
}
```

- kmp

```
void kmp(int f[],char p[],char s[])
{
    int n = strlen(s);
    int m = strlen(p);
    int j = 0;
    rep(i,0,n)
    {
        while(j && s[i] != p[j]) j = f[j-1];
        if(s[i] == p[j]) j++;
        if(j == m)
        {
            cout << i - j + 2 << endl;
            j = f[j-1];
        }
    }
}
```

6.2 SA

```
const int N=4e5+100;
const int maxn = 2e5+100;
const int inf=1e9+9;

namespace SA {
    char s[N];
    int sa[N],x[N],y[N],hep[N],height[N],n,m;
    void init()
    {
        n = 0;
    }
    void add(char c)
    {
        // c -= 'a';
```

```
15      n++;
16      s[n]=c;
17  }
18  void Sort() {
19      for(int i=0;i<=m;++i) hep[i]=0;
20      for(int i=1;i<=n;++i) ++hep[x[i]];
21      for(int i=1;i<=m;++i) hep[i]+=hep[i-1];
22      for(int i=n;i>=1;--i) sa[hep[x[y[i]]]--]=y[i];
23  }
24  void Pre_sa() {
25      for(int i=1;i<=n;++i) x[i]=s[i],y[i]=i;
26      m=223;Sort();
27      for(int w=1,p=0;m=p,p<n;w<<=1) {
28          p=0;
29          for(int i=1;i<=w;++i) y[++p]=n-w+i;
30          for(int i=1;i<=n;++i) if(sa[i]>w) y[++p]=sa[i]-w;
31          Sort(),swap(x,y),x[sa[1]]=p=1;
32          for(int i=2;i<=n;++i)
33              x[sa[i]]=(y[sa[i]]==y[sa[i-1]]&&y[sa[i]+w]==y[sa[i-1]+w])?p:++p;
34      }return;
35  }
36  ll Pre_height() {
37      for(int i=1;i<=n;++i) x[sa[i]]=i;
38      int k=0,res=0;
39      for(int i=1;i<=n;++i) {
40          k-=k>0;
41          int j=sa[x[i]-1];
42          while(i+k<=n&&j+k<=n&&s[i+k]==s[j+k]) ++k;
43          height[x[i]]=k,res+=k;
44      }return res;//直接返回height数组的和
45  }
46  ll solve()
47  {
48      /**
49      给你一个长为N的字符串，求不同的子串的个数?
50      对于一个后缀sa[i]，它产生了n-sa[i]个前缀，减去height[i]个相同的前缀(与前一个比较)，
51      则产生了n-sa[i]-height[i]个子串。累加后即结果。
52      */
53      ll ans = 0;
54      for (int i = 1; i <= n; i++)
55      {
56          ans += n + 1 - sa[i] - height[i];
57      }
58      return ans;
59  }
60  ll gao()
```

```
61      {
62          Pre_sa();
63          Pre_height();
64          return solve();
65      }
66  }
```

6.3 回文树 1

```
1   struct Pal
2   {
3       int ch[maxn][26],f[maxn],len[maxn],s[maxn];
4       int cnt[maxn];
5       int num[maxn];
6       int last,sz,n;
7
8       int newnode(int x)
9       {
10          memset(ch[sz],0,sizeof(ch[sz]));
11          cnt[sz] = num[sz] = 0, len[sz] = x;
12          return sz++;
13      }
14      void init()
15      {
16          sz = 0;
17          newnode(0),newnode(-1);
18          last = n = 0,s[0] = -1,f[0] = 1;
19      }
20
21      int get_fail(int u)
22      {
23          while(s[n - len[u] - 1] != s[n])
24              u = f[u];
25          return u;
26      }
27
28      void add(int c)
29      {
30          c -= 'a';
31          s[++n] = c;
32          int u = get_fail(last);
33          if(!ch[u][c])
34          {
35              int np = newnode(len[u] + 2);
36              f[np] = ch[get_fail(f[u])][c];
```

```
            num[np] = num[f[np]] + 1;
            ch[u][c] = np;
        }
        last = ch[u][c];
        cnt[last]++;
    }

    void count()
    {
        for(int i = sz - 1;~i;i--)
        cnt[f[i]] += cnt[i];
    }
} pa;
```

6.4 回文树 2

```
struct Palindromic_Tree {
    int son[N][26]; //转移边
    int fail[N]; //fail 指针
    int cnt[N]; //当前节点表示的回文串在原串中出现了多少次
    int num[N]; //当前节点 fail 可以向前跳多少次
    int len[N]; //当前节点表示的回文串的长度
    int S[N]; //插入的字符串
    int last; //最后一次访问到的节点，类似 SAM
    int n; //插入的字符串长度
    long long p; //自动机的总状态数

    int newnode(int l) {
        memset(son[p], 0, sizeof(son[p]));
        cnt[p] = 0;
        num[p] = 0;
        len[p] = l;
        return p++;
    }

    void init() {
        p = 0;
        newnode(0);
        newnode(-1);
        last = 0;
        n = 0;
        S[n] = -1;
        fail[0] = 1;
    }

```

```
    int get_fail(int x) {
        while (S[n - len[x] - 1] != S[n]) x = fail[x];
        return x;
    }

    void add(int c) {
        c -= 'a';
        S[++n] = c;
        int cur = get_fail(last); //通过上一次访问的位置去扩展
        if (!son[cur][c]) { //如果没有对应的节点添加一个新节点
            int now = newnode(len[cur] + 2);
            fail[now] = son[get_fail(fail[cur])][c]; //通过当前节点的 fail 去扩展出新的 fail
            son[cur][c] = now;
            num[now] = num[fail[now]] + 1; //记录 fail 跳多少次
        }
        last = son[cur][c];
        cnt[last]++; //表示当前节点访问了一次
    }
    void count() {
        //如果某个节点出现一次，那么他的 fail 也一定会出现一次，并且在插入的时候没有计数
        for (int i = p - 1; i >= 0; i--) cnt[fail[i]] += cnt[i];
    }
} AUT;
```

7 杂项

7.1 退火

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

using namespace std;

const int maxn = 1e5 + 10;
const double eps = 1e-8;
const double delta = 0.98;
const double inf = 1e18;

struct Point { double x, y; } p[maxn];

double dis(Point A, Point B) { return sqrt((A.x - B.x) * (A.x - B.x) + (A.y - B.y) * (A.y - B.y)); };

double Simulate_Annea(int n)
{
    Point S;
    S.x = S.y = 0;
```

```
    double t = 1000;
    double res = inf;
    while(t > eps)
    {
        int k = 0;
        for(int i = 0; i < n; i ++) if(dis(S, p[i]) > dis(S, p[k])) k = i;
        double d = dis(S, p[k]);
        res = min(res, d);
        S.x += (p[k].x - S.x) / d * t;
        S.y += (p[k].y - S.y) / d * t;
        t *= delta;
    }
    return res;
}

int main()
{
    int n;
    scanf("%d", &n);
    for(int i = 0; i < n; i ++) scanf("%lf%lf", &p[i].x, &p[i].y);
    printf("%.3f\n", Simulate_Annea(n));
    return 0;
}
```

7.2 博弈

7.2.1 Bash

只有一堆 n 个物品, 两个人轮流从这堆物品中取物, 规定每次至少取一个, 最多取 m 个. 最后取光者得胜. 考虑 n = k*(1+m)+r 即可 r = 0, 先手必败，无论先手怎么取（设取的是 a），后手都能凑出 1+m-a

7.2.2 Wythoff

有两堆火柴棍, 每次可以从某一堆取至少 1 根火柴棍 (无上限)，或者从两堆取相同的火柴棍数。最后取完的是胜利者。考虑状态 (a,b) a[i] = i*(1+√5)/2 b[i] = a[i]+i; 因为有浮点数参与运算，范围过大要考虑手写高精度

7.2.3 Fibonacci's Game

有一堆个数为 n 的石子，游戏双方轮流取石子，满足：1) 先手不能在第一次把所有的石子取完；2) 之后每次可以取的石子数介于 1 到对手刚取的石子数的 2 倍之间（包含 1 和对手刚取的石子数的 2 倍）。先手胜当且仅当 n 不是 Fibonacci 数。

7.2.4 staircase nim

每层有若干个石子，每次可以选择任意层的任意个石子将其移动到该层的下一层。最后不能操作的人输把所有奇数阶梯看成 N 堆石子做 nim。把石子从奇数堆移动到偶数堆可以理解为拿走石子，就相当于几个奇数堆的石子在做 Nim。

7.2.5 anti-nim

正常的 nim 游戏是取走最后一颗的人获胜，而反 nim 游戏是取走最后一颗的人输。一个状态为必胜态，当且仅当: 1）所有堆的石子个数为 1，且 NIM_sum（异或和）=0 2）至少有一堆的石子个数大于 1，且 NIM_sum 0

7.2.6 约数博弈

游戏初始状态包含 1-n, 这 n 个正整数。甲乙两个人轮流玩这个游戏. 每轮游戏中，游戏者任意选择一个还存在的数，然后删掉它和它所有的约数。第一个删掉所有数的人获胜。先手必胜反证法 - 考虑一个新的规则 “不准写数字 1”。依然先手必胜证明：如果在新规则下后写者必胜，则原游戏中的先写者写下数字 1，然后他就变成了新规则下的后写者。

7.2.7 约数和倍数博弈

游戏初始状态包含 2-n, 这 n-1 个正整数。甲乙两个人轮流玩这个游戏，每轮游戏中，游戏者任意选择一个还存在的数，然后删掉它和它所有的约数、倍数。第一个删掉所有数的人获胜。先手必败

7.2.8 Chomp 博弈

有一个 n * m 的棋盘，每次可以取走一个方格并拿掉它右边和上面的所有方格。拿掉左下角的格子 (1,1) 者输。- 除了 1*1 的棋盘，先手必败 - 对于其他大小的棋盘，先手必胜。

7.2.9 树上删边游戏

给出一个有 N 个点的树，有一个点作为树的根节点。游戏者轮流从树中删去边，删去一条边后，不与根节点相连的部分将被移走。谁无法移动谁输。叶子节点的 SG 值为 0；中间节点的 SG 值为它的所有子节点的 SG 值加 1 后的异或和。HDU 3590 N<=100 颗树, 每棵数有 m<=100 个点, 双方每次可以选择一棵树的一条边删去, 并且把不与根相连的边一并删去, 不能操作者赢先手必胜当且仅当: 1) 游戏的 SG 函数不为 0 且游戏中某个单一游戏的 SG 函数大于 1
2) 游戏的 SG 函数为 0 且游戏中没有单一游戏的 SG 函数大于 1

```
VI v[100005];
int getsg(int x,int pre)
{
    int ans = 0;
    rep(i,0,v[x].size())
        if(v[x][i] != pre)
        {
            ans ^= (1+getsg(v[x][i],x));
        }
    return ans;
}

int main()
{
    int t,n;
    while(scanf("%d",&t)!=EOF)
    {
```

```
        int sum = 0,ans = 0;
        while(t--)
        {
            scanf("%d",&n);
            rep(i,1,n+1)
            {
                v[i].clear();
            }
            rep(i,1,n)
            {
                int x,y;
                scanf("%d%d",&x,&y);
                v[x].push_back(y);
                v[y].push_back(x);
            }
            int s = getsg(1,-1);
            if(s > 1)
                ans++;
            sum ^= s;
        }
        if((sum == 0&&ans==0)||(sum!=0&&ans>=1))
        {
            cout<<"PP"<<endl;
        }
        else
        {
            cout<<"QQ"<<endl;
        }
    }
    return 0;
}
```

7.2.10 SG

```
define MAX 1005
/* 计算从1-n范围内的SG值。
   Array(存储可以走的步数，Array[0]表示可以有多少种走法)
   Array[]需要从小到大排序 */
/*HDU1847博弈SG函数
1.可选步数为1-m的连续整数，直接取模即可，SG(x) = x % (m+1);
2.可选步数为任意步，SG(x) = x;
3.可选步数为一系列不连续的数，用GetSG(计算) */
int SG[MAX], hash[MAX];
void GetSG(int Array[],int n = MAX-1)
{
```

```
    memset(SG, 0, sizeof(SG));
    for(int i = 0; i <= n; ++i)
    {
        memset(hash, 0, sizeof(hash));
        for(int j = 1; j <= Array[0]; ++j)
        {
            if(i < Array[j])
                break;
            hash[SG[i - Array[j]]] = 1;
        }
        for(int j = 0; j <= n; ++j)
            if(!hash[j])
            {
                SG[i] = j;
                break;
            }
    }
}
```

S-Nim 游戏仅仅是限制了每一次从每一堆中选取的个数，依旧用 sg 函数计算即可。经典的 Nim 游戏中 sg(x) = x，所以结果就是每一堆的状态直接 xor 即可。S-Nim 游戏先计算每一堆的 sg 函数值，然后判断方法依旧是用 xor.

```
const int maxn = 10100;
const int N = 110;
int SG[maxn],f[N],s[maxn];

int k,n,m,t;

void getsg(int n,int nn)
{
    SG[0] = 0;
    int tt = 0;
    for(int i = 1;i<=10000;i++)
    {
        tt++;
        //memset(s,0,sizeof s);
        for(int j = 0;f[j]<=i&&j<nn;j++)
        {
            s[SG[i-f[j]]] = tt;
        }
        for(int j = 0;;j++)
            if(s[j]!=tt)
        {
            SG[i] = j;
            break;
```

```
        }
    }
}

int main()
{
    while(scanf("%d",&k)!=EOF&&k)
    {
        for(int i =0;i<k;i++)
            scanf("%d",&f[i]);
        sort(f,f+k);
        scanf("%d",&n);
        getsg(10000,k);
        for(int i = 0;i<n;i++)
        {
            int ans = 0;
            scanf("%d",&m);
            for(int i = 0;i<m;i++)
                {
                    scanf("%d",&t);
                    ans^=SG[t];
                }
                if(ans)
                    cout<<"W";
                else
                    cout<<"L";
        }

        cout<<endl;
    }
}
```

7.3 indiewar 的私人题

7.3.1 半圆概率

在一个圆内任取 n 个点，则这 n 个点出现在同一个半圆内的概率是多少转化为 2n 条半径的古典概型问题：$P_n = 2n/(2^n) = n/2^{n-1}$

7.3.2 百囚徒挑战

监狱决定给关押的 100 名囚徒一次特赦的机会，条件是囚徒通过一项挑战。

所有囚徒被编号为 1-100，对应他们编号的 100 个号码牌被打乱顺序放在了 100 个抽屉里。

每个囚徒需要从所有抽屉里打开至多半数 (50 个)，并从中找出对应自己编号的号码牌。

如果找到了则该名囚徒的任务成功。所有囚徒会依次单独进入挑战室完成任务，并且从第一个囚徒进入挑战室开始，直到所有囚徒结束挑战为止囚徒之间任何形式的交流都是禁止的。

当一名囚徒完成任务后，挑战室会被恢复为他进入之前的样子（号码牌当然也放回原来的抽屉里）。

在这 100 名囚徒中，任意一名囚徒的失败都会导致整个挑战失败，只有当所有囚徒全部成功完成任务时，他们才会统一得到特赦的机会。最后，在开始挑战之前，监狱给了所有囚徒一个月时间商量对策。那么，囚徒究竟有多大的几率得到释放？$p(n) = 1 - \sum_{m=n+1}^{2n} \frac{1}{m}$

7.3.3 投针问题

假定有一块画着许多等距的平行线的地板，现往板上投放一根细针，求细针与平行线相交的概率。假设平行线间距为 a ，针长为 l 且满足 l < a $p = \frac{2l}{a*pi}$

l >= a $p = 1 - \frac{1}{pi}(2arcsin\frac{a}{l} - \frac{2l}{a} + \frac{2}{a}\sqrt{l^2 - a^2})$

7.3.4 打怪兽

假设你打一下怪, 怪掉总血量的 0% 到 100% 之间的一个随机数,(包括 [0,1] 区间内的所有实数诸如 0.5%) 那么, 求你把怪打死的一共要打几次的数学期望. 不妨设怪物血量为 1, 你每次攻击输出 [公式] 间均匀的随机伤害. 假设 f(x) 为打死血量为 x 的怪物所用的期望次数. 这个比较反常识, 虽然我们是要求 f(1), 但是, 我们却选择求看起来似乎更难的 f(x) $f(x) = e^x$

7.3.5 互素

随机地取两个正整数，他们互素的概率为 $p = \frac{6}{pi*pi}$

7.3.6 半球

在一个球内任取 n 个点，则这 n 个点落在同一个半球内的概率 $p = \frac{n^2-n+2}{2^n}$

圆 $p = \frac{2n}{2^n}$

8 DP

8.1 背包

8.1.1 01 背包

- normal $f_i = max(f_i, f_i v + w)$ 需要按照 i 从大到小的顺序更新，确保每个物品只会选一次

```
memset(dp,0xcf,sizeof dp);
dp[0] = 0;
rep(i,0,n)
{
    cin >> v[i] >> w[i];
    per(j,v[i],m+1)
    {
        dp[j] = max(dp[j],dp[j-v[i]]+w[i]);
    }
}
```

- 计数不超过 m 的方案数 $f_i += f_i v$

- 删除加入物品的顺序不影响结果，假设被删除的物品是最后一次加入的，那么倒过来还原即可。$f_i -= f_i v$ 需要按照 i 从小到大的顺序更新. 给定 ai 和 bi，表示第 i 个商店有 ai 个商品可以买，单价为 bi 元，给出 m 个询问，问用 c 元在 l~r 商店买东西的方案数一种物品的背包可以看成 $\sum_{i=0}^{a} x^{ib} = \frac{1-x^{(a+1)b}}{1-x^b}$，所以可以先用 (a+1)b 去做一个 01 背包 (系数为负)，再除以一个 x^b 的 (系数为负)01 背包。从生成函数来看，$\frac{1}{1-x^b} = \sum_{i=0}^{\infty} x^{bi}$，即做一遍完全背包就可以等效

然后对可逆背包的预处理，由于 $\frac{1-x^b}{1-x^{(a+1)*b}} = (1-x^b) * \sum_{i=0}^{\infty} x^{i*(a+1)b}$，于是反过来对 x^b 做 01 背包，对 (a+1)b 做完全背包就可以

```
int n,m;
int a[maxn*10],b[maxn*10],f[maxn*10][maxn],g[maxn*10][maxn];

void gao(int *dp,int w)
{
    per(i,w,maxn)
    {
        dp[i] = (dp[i] - dp[i-w]+mod)%mod;
    }
}

void gao2(int *dp,int w)
{
    rep(i,w,maxn)
    {
        dp[i] = (dp[i] + dp[i-w]+mod)%mod;
    }
}

int main(int argc, char const *argv[])
{
    ios_base::sync_with_stdio(false), cin.tie(0);
    // cout.tie(0);
    int T,cas;
    cin >> T;
    cas = 1;
    f[0][0] = 1;
    rep(i,0,maxn)
        g[0][i] = 1;
    while(T--)
    {
        prr(cas++);
        cin >> n >> m;
        rep(i,1,n+1)
        {
            cin >> a[i] >> b[i];
            a[i] = (a[i] + 1) * b[i];
        }
```

```
        rep(i,1,n+1)
        {
            memcpy(f[i],f[i-1],sizeof(f[i]));
            memcpy(g[i],g[i-1],sizeof(g[i]));
            gao(f[i],a[i]);gao2(f[i],b[i]);
            gao(g[i],b[i]);gao2(g[i],a[i]);
        }
        int ans = 0;
        rep(i,0,m)
        {
            int l,r,c;
            cin >> l >> r >> c;
            l = (l+ans)%n+1;
            r = (r+ans)%n+1;
            if(l > r)
            {
                swap(l,r);
            }
            ans = 0;
            rep(i,0,c+1)
            {
                ans = (ans + 1ll*f[r][i]*g[l-1][c-i])%mod;
            }
            printf("%d\n",ans);
        }
    }
    return 0;
}
```

8.1.2 完全背包

- normal 需要按照 i 从小到大的顺序更新，意为要么停止选，要么接着多选一个。

```
dp[0] = 0;
rep(i,0,n)
{
    cin >> v >> w;
    rep(j,v,m+1)
    {
        dp[j] = max(dp[j],dp[j-v]+w);
    }
}
cout << dp[m] << endl;
```

- 计数 same

- 删除 same

8.1.3 多组背包

- 二进制拆分二进制拆分，将一个物品拆成 O(logk) 个 01 背包的物品。eg:$10 = 1 + 2 + 4 + 3$，可以表示 $1 - 10$ O(nmlog(k))

```
memset(dp,0,sizeof dp);
    rep(i,0,n)
    {
        cin >> v[i] >> w[i] >> s[i];
    }
    int cnt = 0;
    rep(i,0,n)
    {
        for(int j = 1;j < s[i];j <<= 1)
        {
            ww[cnt] = j * w[i];
            vv[cnt] = j * v[i];
            s[i] -= j;
            cnt++;
        }
        if(s[i])
        {
            ww[cnt] = s[i] * w[i];
            vv[cnt++] = s[i] * v[i];
        }
    }
    rep(i,0,cnt)
    {
        per(j,vv[i],m+1)
        {
            dp[j] = max(dp[j],dp[j-vv[i]] + ww[i]);
        }
    }
    cout << dp[m] << endl;
```

- 单调队列按%v 的余数分组，每组滑窗求区间最大值 O(nm), 但不见得比上面快

8.1.4 分组背包

n 个物品，每个物品只能选一个，体积为 vi，种类为 ki。求总体积恰好 m 的情况下能拿走物品种类数的最大值。将所有物品按 k 分组状态：$f_{i,j,k,s}$ 表示考虑前 i 组，这一组内考虑了前 j 个物品，总体积为 k，第 i 组物品是否被选择的情况为 s 时，种类数的最大值。

8.1.5 树形依赖背包

以 1 为根的树上有 n 个节点，每个节点有一个物品，体积 vi，价值 wi。选了一个点就必须选它的父亲。求总体积不超过 m 的情况下能拿走物品总价值的最大值。

按照 DFS 的顺序进行 DP。往下搜的时候，强行将儿子选入背包中。往上回溯的时候，可以选择要这棵子树的 DP 值，或者不要。