数学

高斯消元

```
const double EPS = 1E-9;
int n;
vector<vector<double> > a(n, vector<double>(n));

double det = 1;
for (int i = 0; i < n; ++i) {
  int k = i;
  for (int j = i + 1; j < n; ++j)
    if (abs(a[j][i]) > abs(a[k][i])) k = j;
  if (abs(a[k][i]) < EPS) {
    det = 0;
    break;
  }
  swap(a[i], a[k]);
  if (i != k) det = -det;
  det *= a[i][i];
  for (int j = i + 1; j < n; ++j) a[i][j] /= a[i][i];
  for (int j = 0; j < n; ++j)
    if (j != i && abs(a[j][i]) > EPS)
      for (int k = i + 1; k < n; ++k) a[j][k] -= a[i][k] * a[j][i];
}

cout << det;
```

高斯消元解异或方程组

```
std::bitset<1010> matrix[2010];  // matrix[1~n]：增广矩阵，0 位置为常数
// n 为未知数个数，m 为方程个数，返回方程组的解，多解 / 无解返回一个空的 vector)
std::vector<bool> GaussElimination(int n, int m) {
  for (int i = 1; i <= n; i++) {
    int cur = i;
    while (cur <= m && !matrix[cur].test(i)) cur++;
    if (cur > m) return std::vector<bool>(0);
    if (cur != i) swap(matrix[cur], matrix[i]);
    for (int j = 1; j <= m; j++)
      if (i != j && matrix[j].test(i)) matrix[j] ^= matrix[i];
  }
  std::vector<bool> ans(n + 1, 0);
  for (int i = 1; i <= n; i++) ans[i] = matrix[i].test(0);
  return ans;
}
```

解线性方程组

```
int n, m, negative = 0, isd[500], rank;
struct Fac {
    long long a, b;
    void simple() {
        long long g = gcd(a, b);
        a /= g;
        b /= g;
    }
    Fac operator+(Fac y) const {
        long long g = gcd(b, y.b);
        long long x1 = g * y.b, x2 = g * b;
        Fac n;
        n.a = a * x1 + y.a * x2;
        n.b = g * y.b * b;
        n.simple();
        return n;
    }
    Fac operator-(Fac y) const {
        Fac p = y;
        p.a *= -1;
        return *this + p;
    }
    Fac operator*(Fac y) const {
        Fac n;
        n.a = a * y.a;
        n.b = b * y.b;
        n.simple();
        return n;
    }
    Fac operator/(Fac y) const {
        Fac n;
        n.a = y.b;
        n.b = y.a;
        return *this * n;
    }
    void print(char c = '\0', char sep = '\0') {
        if (!a)
            putchar('0');
        else if (b == 1) {
            if (a < 0) printf("%c-%c", sep, sep);
            printf("%lld", abs(a));
        } else {
            if (a < 0 && b > 0 || a > 0 && b < 0) printf("%c-%c", sep, sep);
            printf("%lld/%lld", abs(a), abs(b));
        }
        putchar(c);
    }
```

```
} det[100][100];
Fac one = {1, 1};
void lineSwap(int line1, int line2) {
    for (int j = 1; j <= m; j++) {
        Fac tmp = det[line1][j];
        det[line1][j] = det[line2][j];
        det[line2][j] = tmp;
    }
    negative = !negative;
}
int checkFirstLineWithout0(int line) {
    for (int i = line; i <= n; i++)
        if (det[i][line].a) return i;
    return 0;
}
void print(int d, int a = 0, int b = 0) {
    static int id[2] = {0, 0};
    for (int i = 1; i <= 30; i++) putchar(d == 1 ? '=' : '*');
    if (d == 1)
        printf("\nRow Echelon Form #%d:\n", ++id[0]);
    else
        printf("\nRow Simplest Form #%d:\n", ++id[1]);
    putchar('\n');
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            det[i][j].print('\t');
        }
        puts(i == a || i == b ? " *" : "");
    }
    putchar('\n');
}
void sol(int x, int y) {
    for (int i = 1; i <= n; i++) {
        if (i == x) continue;
        Fac ratio = det[i][y] / det[x][y];
        for (int j = y; j <= m; j++) det[i][j] = det[i][j] - ratio * det[x][j];
    }
}
Fac toFac(double f) {
    int a = int(f), b = 1;
    for (int i = 1; i <= 10; i++) {
        if (fabs(f * b - a) < 1e-6) {
            Fac n = {a, b};
            n.simple();
            return n;
        }
        b *= 10;
        a = int(f * b);
    }
    return (Fac){ 0, 1 };
}
```

```
int main() {
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++) {
            double k;
            scanf("%lf", &k);
            det[i][j] = toFac(k);
        }
    for (int j = 1; j <= m; j++) {
        int a = 0, b = 0, t = checkFirstLineWithout0(j);
        if (j != t) {
            if (t) {
                lineSwap(j, t);
                a = j;
                b = t;
            } else continue;
        }
        for (int i = j + 1; i <= n; i++) {
            Fac ratio = det[i][j] / det[j][j];
            for (int k = j; k <= m; k++) {
                det[i][k] = det[i][k] - det[j][k] * ratio;
            }
        }
        print(1, a, b);
    }
    Fac result;
    result.a = result.b = 1;
    for (int i = 1; i <= n; i++) result = result * det[i][i];
    for (int i = 1, l = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            if (det[i][j].a) {
                rank++;
                isd[j] = 1;
                Fac ratio = one / det[i][j];
                for (int k = 1; k <= m; k++) {
                    if (det[i][k].a) {
                        det[i][k] = ratio * det[i][k];
                    }
                }
                sol(i, j);
                print(2);
                break;
            }
        }
    }
    if (n == m) {
        printf("det = ");
        if (negative) result.a *= -1;
        result.print('=');
        printf("%lf\n", (double) result.a / result.b);
    }
```

```
    printf("rank = %d\n", rank);
    for (int j = 1; j <= m; j++) {
        if (isd[j]) continue;
        printf("V%d = ", j);
        int flag = 0;
        for (int k = 1; k <= m; k++) {
            if (!isd[k]) continue;
            for (int i = 1; i <= n; i++) {
                if (det[i][k].a) {
                    if (det[i][j].a) {
                        if (det[i][j].a * det[i][j].b >= 0 && flag) {
                            printf(" + ");
                        }
                        if (det[i][j].a == -det[i][j].b) {
                             printf("%c- V%d", flag ? ' ' : '\0', k);
                        } else if (det[i][j].a != det[i][j].b) {
                            det[i][j].print('\0', ' ');
                            printf(" * V%d", k);
                        } else {
                            printf("%cV%d", flag ? ' ' : '\0', k);
                        }
                        flag = 1;
                    }
                    break;
                }
            }
        }
        putchar('\n');
    }
    return 0;
}
```

多项式三角函数

```
constexpr int maxn = 262144;
constexpr int mod = 998244353;
constexpr int imgunit = 86583718; /* sqrt(-1) = sqrt(998233452) */

using i64 = long long;
using poly_t = int[maxn];
using poly = int *const;

void polytri(const poly &h, const int n, poly &sin_t, poly &cos_t) {
  /* sin(f) = (exp(i * f) - exp(- i * f)) / 2i */
  /* cos(f) = (exp(i * f) + exp(- i * f)) / 2 */
  /* tan(f) = sin(f) / cos(f) */
  assert(h[0] == 0);
  static poly_t tri1_t, tri2_t;

  for (int i = 0; i != n; ++i) tri2_t[i] = (i64)h[i] * imgunit % mod;
  polyexp(tri2_t, n, tri1_t);
  polyinv(tri1_t, n, tri2_t);

  if (sin_t != nullptr) {
    const int invi = fpow(pls(imgunit, imgunit), mod - 2);
    for (int i = 0; i != n; ++i)
      sin_t[i] = (i64)(tri1_t[i] - tri2_t[i] + mod) * invi % mod;
  }
  if (cos_t != nullptr) {
    for (int i = 0; i != n; ++i) cos_t[i] = div2(pls(tri1_t[i], tri2_t[i]));
  }
}
```

多项式反三角函数

```
constexpr int maxn = 262144;
constexpr int mod = 998244353;

using i64 = long long;
using poly_t = int[maxn];
using poly = int *const;

inline void derivative(const poly &h, const int n, poly &f) {
  for (int i = 1; i != n; ++i) f[i - 1] = (i64)h[i] * i % mod;
  f[n - 1] = 0;
}

inline void integrate(const poly &h, const int n, poly &f) {
  for (int i = n - 1; i; --i) f[i] = (i64)h[i - 1] * inv[i] % mod;
  f[0] = 0; /* C */
}

void polyarcsin(const poly &h, const int n, poly &f) {
  /* arcsin(f) = ∫ f' / sqrt(1 - f^2) dx  */
  static poly_t arcsin_t;
  const int t = n << 1;
  std::copy(h, h + n, arcsin_t);
  std::fill(arcsin_t + n, arcsin_t + t, 0);

  DFT(arcsin_t, t);
  for (int i = 0; i != t; ++i) arcsin_t[i] = sqr(arcsin_t[i]);
  IDFT(arcsin_t, t);

  arcsin_t[0] = sub(1, arcsin_t[0]);
  for (int i = 1; i != n; ++i)
    arcsin_t[i] = arcsin_t[i] ? mod - arcsin_t[i] : 0;

  polysqrt(arcsin_t, n, f);
  polyinv(f, n, arcsin_t);
  derivative(h, n, f);

  DFT(f, t);
  DFT(arcsin_t, t);
  for (int i = 0; i != t; ++i) arcsin_t[i] = (i64)f[i] * arcsin_t[i] % mod;
  IDFT(arcsin_t, t);

  integrate(arcsin_t, n, f);
}

void polyarccos(const poly &h, const int n, poly &f) {
  /* arccos(f) = - ∫ f' / sqrt(1 - f^2) dx  */
  polyarcsin(h, n, f);
```

```
  for (int i = 0; i != n; ++i) f[i] = f[i] ? mod - f[i] : 0;
}

void polyarctan(const poly &h, const int n, poly &f) {
  /* arctan(f) = ∫ f' / (1 + f^2) dx  */
  static poly_t arctan_t;
  const int t = n << 1;
  std::copy(h, h + n, arctan_t);
  std::fill(arctan_t + n, arctan_t + t, 0);

  DFT(arctan_t, t);
  for (int i = 0; i != t; ++i) arctan_t[i] = sqr(arctan_t[i]);
  IDFT(arctan_t, t);

  inc(arctan_t[0], 1);
  std::fill(arctan_t + n, arctan_t + t, 0);

  polyinv(arctan_t, n, f);
  derivative(h, n, arctan_t);

  DFT(f, t);
  DFT(arctan_t, t);
  for (int i = 0; i != t; ++i) arctan_t[i] = (i64)f[i] * arctan_t[i] % mod;
  IDFT(arctan_t, t);

  integrate(arctan_t, n, f);
}
```

埃拉托斯特尼筛法

```
int Eratosthenes(int n) {
  int p = 0;
  for (int i = 0; i <= n; ++i) is_prime[i] = 1;
  is_prime[0] = is_prime[1] = 0;
  for (int i = 2; i <= n; ++i) {
    if (is_prime[i]) {
      prime[p++] = i;  // prime[p]是i,后置自增运算代表当前素数数量
      if ((long long)i * i <= n)
        for (int j = i * i; j <= n; j += i)
          // 因为从 2 到 i - 1 的倍数我们之前筛过了，这里直接从 i
          // 的倍数开始，提高了运行速度
          is_prime[j] = 0;  // 是i的倍数的均不是素数
    }
  }
  return p;
}
```

线性筛素数

```cpp
// C++ Version
void init() {
  for (int i = 2; i < MAXN; ++i) {
    if (!vis[i]) {
      pri[cnt++] = i;
    }
    for (int j = 0; j < cnt; ++j) {
      if (1ll * i * pri[j] >= MAXN) break;
      vis[i * pri[j]] = 1;
      if (i % pri[j] == 0) {
        break;
      }
    }
  }
}
```

线性筛欧拉函数

```cpp
void pre() {
  memset(is_prime, 1, sizeof(is_prime));
  int cnt = 0;
  is_prime[1] = 0;
  phi[1] = 1;
  for (int i = 2; i <= 5000000; i++) {
    if (is_prime[i]) {
      prime[++cnt] = i;
      phi[i] = i - 1;
    }
    for (int j = 1; j <= cnt && i * prime[j] <= 5000000; j++) {
      is_prime[i * prime[j]] = 0;
      if (i % prime[j])
        phi[i * prime[j]] = phi[i] * phi[prime[j]];
      else {
        phi[i * prime[j]] = phi[i] * prime[j];
        break;
      }
    }
  }
}
```

线性筛莫比乌斯函数

```
void pre() {
  mu[1] = 1;
  for (int i = 2; i <= 1e7; ++i) {
    if (!v[i]) mu[i] = -1, p[++tot] = i;
    for (int j = 1; j <= tot && i <= 1e7 / p[j]; ++j) {
      v[i * p[j]] = 1;
      if (i % p[j] == 0) {
        mu[i * p[j]] = 0;
        break;
      }
      mu[i * p[j]] = -mu[i];
    }
  }
```

线性筛约数个数

```
void pre() {
  d[1] = 1;
  for (int i = 2; i <= n; ++i) {
    if (!v[i]) v[i] = 1, p[++tot] = i, d[i] = 2, num[i] = 1;
    for (int j = 1; j <= tot && i <= n / p[j]; ++j) {
      v[p[j] * i] = 1;
      if (i % p[j] == 0) {
        num[i * p[j]] = num[i] + 1;
        d[i * p[j]] = d[i] / num[i * p[j]] * (num[i * p[j]] + 1);
        break;
      } else {
        num[i * p[j]] = 1;
        d[i * p[j]] = d[i] * 2;
      }
    }
  }
}
```

线性筛求约数和

```
void pre() {
  g[1] = f[1] = 1;
  for (int i = 2; i <= n; ++i) {
    if (!v[i]) v[i] = 1, p[++tot] = i, g[i] = i + 1, f[i] = i + 1;
    for (int j = 1; j <= tot && i <= n / p[j]; ++j) {
      v[p[j] * i] = 1;
      if (i % p[j] == 0) {
        g[i * p[j]] = g[i] * p[j] + 1;
        f[i * p[j]] = f[i] / g[i] * g[i * p[j]];
        break;
      } else {
        f[i * p[j]] = f[i] * f[p[j]];
        g[i * p[j]] = 1 + p[j];
      }
    }
  }
}
```

杜教筛

```cpp
#include <algorithm>
#include <cstdio>
#include <cstring>
#include <map>
using namespace std;
const int maxn = 2000010;
long long T, n, pri[maxn], cur, mu[maxn], sum_mu[maxn];
bool vis[maxn];
map<long long, long long> mp_mu;

long long S_mu(long long x) {  // 求mu的前缀和
  if (x < maxn) return sum_mu[x];
  if (mp_mu[x]) return mp_mu[x];  // 如果map中已有该大小的mu值，则可直接返回
  long long ret = (long long)1;
  for (long long i = 2, j; i <= x; i = j + 1) {
    j = x / (x / i);
    ret -= S_mu(x / i) * (j - i + 1);
  }
  return mp_mu[x] = ret;  // 路径压缩，方便下次计算
}

long long S_phi(long long x) {  // 求phi的前缀和
  long long ret = (long long)0;
  long long j;
  for (long long i = 1; i <= x; i = j + 1) {
    j = x / (x / i);
    ret += (S_mu(j) - S_mu(i - 1)) * (x / i) * (x / i);
  }
  return (ret - 1) / 2 + 1;
}

int main() {
  scanf("%lld", &T);
  mu[1] = 1;
  for (int i = 2; i < maxn; i++) {  // 线性筛预处理mu数组
    if (!vis[i]) {
      pri[++cur] = i;
      mu[i] = -1;
    }
    for (int j = 1; j <= cur && i * pri[j] < maxn; j++) {
      vis[i * pri[j]] = true;
      if (i % pri[j])
        mu[i * pri[j]] = -mu[i];
      else {
        mu[i * pri[j]] = 0;
        break;
      }
```

```
    }
  }
  for (int i = 1; i < maxn; i++)
    sum_mu[i] = sum_mu[i - 1] + mu[i];  // 求mu数组前缀和
  while (T--) {
    scanf("%lld", &n);
    printf("%lld %lld\n", S_phi(n), S_mu(n));
  }
  return 0;
}
```

Powerful Number 筛

```cpp
#include <bits/stdc++.h>
using namespace std;

const int MOD = 1e9 + 7;

template <typename T>
inline int mint(T x) {
  x %= MOD;
  if (x < 0) x += MOD;
  return x;
}

inline int add(int x, int y) { return x + y >= MOD ? x + y - MOD : x + y; }

inline int mul(int x, int y) { return (long long)1 * x * y % MOD; }

inline int sub(int x, int y) {
  return x < y ? x - y + MOD : x - y;
}  // 防止负数

inline int qp(int x, int y) {
  int r = 1;
  for (; y; y >>= 1) {
    if (y & 1) r = mul(r, x);
    x = mul(x, x);
  }
  return r;
}

inline int inv(int x) { return qp(x, MOD - 2); }

namespace PNS {
const int N = 2e6 + 5;
const int M = 35;

long long global_n;

int g[N], sg[N];

int h[N][M];
bool vis_h[N][M];

int ans;

int pcnt, prime[N], phi[N];
bool isp[N];
```

```
void sieve(int n) {
  pcnt = 0;
  for (int i = 2; i <= n; ++i) isp[i] = true;  // 判断质数数组
  phi[1] = 1;
  for (int i = 2; i <= n; ++i) {
    if (isp[i]) {
      ++pcnt;
      prime[pcnt] = i;
      phi[i] = i - 1;
    }
    for (int j = 1; j <= pcnt; ++j) {  // 筛去非质数
      long long nxt = (long long)1 * i * prime[j];
      if (nxt > n) break;
      isp[nxt] = false;
      if (i % prime[j] == 0) {  // i是非质数的情况
        phi[nxt] = phi[i] * prime[j];
        break;
      }
      phi[nxt] = phi[i] * phi[prime[j]];
    }
  }

  for (int i = 1; i <= n; ++i) g[i] = mul(i, phi[i]);

  sg[0] = 0;
  for (int i = 1; i <= n; ++i) sg[i] = add(sg[i - 1], g[i]);  // g函数的前缀和
}

int inv2, inv6;

void init() {
  sieve(N - 1);
  for (int i = 1; i <= pcnt; ++i) h[i][0] = 1, h[i][1] = 0;
  for (int i = 1; i <= pcnt; ++i) vis_h[i][0] = vis_h[i][1] = true;
  inv2 = inv(2);
  inv6 = inv(6);
}

int S1(long long n) { return mul(mul(mint(n), mint(n + 1)), inv2); }

int S2(long long n) {
  return mul(mul(mint(n), mul(mint(n + 1), mint(n * 2 + 1))), inv6);
}

map<long long, int> mp_g;

int G(long long n) {
  if (n < N) return sg[n];
  if (mp_g.count(n)) return mp_g[n];

  int ret = S2(n);
```

```
  for (long long i = 2, j; i <= n; i = j + 1) {
    j = n / (n / i);
    ret = sub(ret, mul(sub(S1(j), S1(i - 1)), G(n / i)));
  }
  mp_g[n] = ret;
  return ret;
}

void dfs(long long d, int hd, int pid) {
  ans = add(ans, mul(hd, G(global_n / d)));

  for (int i = pid, p; i <= pcnt; ++i) {
    if (i > 1 && d > global_n / prime[i] / prime[i]) break;  // 剪枝

    int c = 2;
    for (long long x = d * prime[i] * prime[i]; x <= global_n;
         x *= prime[i], ++c) {  // 计算f.g函数
      if (!vis_h[i][c]) {
        int f = qp(prime[i], c);
        f = mul(f, sub(f, 1));
        int g = mul(prime[i], prime[i] - 1);
        int t = mul(prime[i], prime[i]);

        for (int j = 1; j <= c; ++j) {
          f = sub(f, mul(g, h[i][c - j]));
          g = mul(g, t);
        }
        h[i][c] = f;
        vis_h[i][c] = true;
      }

      if (h[i][c]) dfs(x, mul(hd, h[i][c]), i + 1);
    }
  }
}

int solve(long long n) {
  global_n = n;
  ans = 0;
  dfs(1, 1, 1);
  return ans;
}
}  // namespace PNS

int main() {
  PNS::init();
  long long n;
  scanf("%lld", &n);
  printf("%d\n", PNS::solve(n));
```

```
  return 0;
}
```

Min_25 筛

```
#define JUDGE
#include<cstdio>
#include<algorithm>
#include<ctime>
#include<cmath>
typedef unsigned U;
bool m1;
#ifdef TEST
FILE *Input=fopen("In.in","r"),*Output=stdout;
#endif
#ifdef JUDGE
FILE *Input=stdin,*Output=stdout;
#endif
#ifdef CHECK
FILE *Input=fopen("In.in","r"),*Output=fopen("a.out","w");
#endif
const int maxn=2e6+5,maxsqr=5e4+5;
U Prime[maxn],Mv[maxn],f1[maxn];
U f2[maxn],Sumf2[maxn],Idk[maxn];
U PSumId0[maxsqr<<1],NPSumf1[maxsqr<<1];
U AllSumf1[maxsqr<<1],AllSumf2[maxsqr<<1];
int All[maxsqr<<1],BreakFrom[maxsqr<<1];
int Cube_Size1,Cube_Size2;
bool notPrime[maxn],vis2[maxsqr<<1];
bool m2;
U qpow(U ar1,int ar2){
        U h1=ar1,ans=1;int h2=ar2;
        while(h2){
                if(h2&1) ans*=h1;
                h1*=h1;h2>>=1;
        }return ans;
}void Init_Multive(int ar1,int ar2){
        Cube_Size1=pow(ar1,0.67);
        Cube_Size2=sqrt(ar1);
        notPrime[0]=notPrime[1]=1;
        f1[1]=0;Mv[1]=1;Idk[1]=1;
        for(int i=2;i<=Cube_Size1;i++){
                if(!notPrime[i]){
                        Prime[++Prime[0]]=i;
                        f1[i]=1;Mv[i]=-1;
                        Idk[i]=qpow(i,ar2);
                }for(int j=1;j<=Prime[0]&&i*Prime[j]<=Cube_Size1;j++){
                        notPrime[i*Prime[j]]=1;
                        Idk[i*Prime[j]]=Idk[i]*Idk[Prime[j]];
                        if(f1[i]==1) f1[i*Prime[j]]=Idk[Prime[j]];
                        else f1[i*Prime[j]]=f1[i];
                        if(i%Prime[j]==0){
```

```
                        Mv[i*Prime[j]]=0;break;
                }else Mv[i*Prime[j]]=-Mv[i];
        }
    }for(int i=1;i<=Cube_Size1;i++){
        for(int j=i,k=1;j<=Cube_Size1;j+=i,k++) f2[j]+=f1[i]*Mv[k];
        Sumf2[i]=Sumf2[i-1]+f2[i];
    }
}void Init_All(int ar1){
    for(int l=1,r;l<=ar1;l=r+1){
        r=ar1/(ar1/l);
        All[++All[0]]=ar1/l;
    }std::reverse(All+1,All+All[0]+1);
    for(int i=1,j=1;i<=All[0];i++){
        while(j<=Prime[0]&&Prime[j]*Prime[j]<=All[i]) j++;
        BreakFrom[i]=j-1;
    }
}int Get_Num(int ar1){
    return ar1<=Cube_Size2?ar1:All[0]-All[All[0]]/ar1+1;
}void Init_PSumf1(int ar1){
    for(int i=1;i<=All[0];i++) PSumId0[i]=All[i]-1;
    for(int i=1;i<=Prime[0];i++){
        for(int j=All[0];j>0;j--){
            int nowfrom=All[j]/Prime[i];
            if(nowfrom<Prime[i]) break;
            PSumId0[j]-=PSumId0[Get_Num(nowfrom)]-(i-1);
        }
    }
}void Init_NPSumf1(int ar1){
    for(int i=1;i<=All[0];i++) NPSumf1[i]=0;
    for(int i=Prime[0];i>0;i--){
        for(int j=All[0];j>0;j--){
            if(BreakFrom[j]<i) break;
            int nowfrom=All[j]/Prime[i];
            for(int k=1;nowfrom;k++,nowfrom/=Prime[i]){
                int nowpos=Get_Num(nowfrom);
                NPSumf1[j]+=NPSumf1[nowpos]+(std::max((int)PSumId0[nowpo
            }
        }
    }
}void Init_AllSumf1(int ar1){
    Init_All(ar1);Init_PSumf1(ar1);Init_NPSumf1(ar1);
    for(int i=1;i<=All[0];i++) AllSumf1[i]=PSumId0[i]+NPSumf1[i];
}U Get_Sumf2(int ar1){
    if(ar1<=Cube_Size1) return Sumf2[ar1];
    int now=Get_Num(ar1);
    if(!vis2[now]){
        AllSumf2[now]=AllSumf1[now];
        for(int l=2,r;l<=ar1;l=r+1){
            r=ar1/(ar1/l);
            AllSumf2[now]-=(r-l+1)*Get_Sumf2(ar1/l);
        }vis2[now]=1;
```

```
        }return AllSumf2[now];
}int main(){
        #ifndef JUDGE
        clock_t t0=clock();
        #endif
        int N,k;fscanf(Input,"%d %d",&N,&k);
        Init_Multive(N,k);Init_AllSumf1(N);
        U ans=0;
        for(int l=1,r;l<=N;l=r+1){
                r=N/(N/l);
                ans+=(Get_Sumf2(r)-Get_Sumf2(l-1))*(N/l)*(N/l);
        }fprintf(Output,"%u\n",ans);
        #ifndef JUDGE
        clock_t t1=clock();
        fprintf(stdout,"a.cpp %lfms %lfMB\n",(t1-t0)*1000.0/CLOCKS_PER_SEC,(&m2-&m1)/10
        #endif
        fclose(Input);fclose(Output);return 0;
}
```

洲阁筛

```
#define JUDGE
#include<cstdio>
#include<ctime>
#include<algorithm>
#include<cmath>
typedef long long LL;
bool m1;
#ifdef TEST
FILE *Input=fopen("In.in","r"),*Output=stdout;
#endif
#ifdef JUDGE
FILE *Input=stdin,*Output=stdout;
#endif
#ifdef CHECK
FILE *Input=fopen("In.in","r"),*Output=fopen("a.out","w");
#endif
const int maxsqr=1e5+5,Mod=1e9+7;
int Prime[maxsqr],f[maxsqr],Min_Pow[maxsqr];
int SumPf[maxsqr],Anti2,Min_cnt[maxsqr];
int SumPId[maxsqr],PSumf[maxsqr<<1];
bool notPrime[maxsqr];
LL All[maxsqr<<1],Cube_Size;
int BreakFrom[maxsqr<<1],PSumId0[maxsqr<<1];
int PSumId[maxsqr<<1],NPSumf[maxsqr<<1];
bool m2;
int Get_Mod(LL ar1,int ar2){
        return ar1<ar2?ar1:ar1%ar2;
}int Plus(int ar1,int ar2,int ar3){
        return ar3-ar1<=ar2?ar1-ar3+ar2:ar1+ar2;
}int Minu(int ar1,int ar2,int ar3){
        return ar1-ar2<0?ar1-ar2+ar3:ar1-ar2;
}int Mul(int ar1,int ar2,int ar3){
        return 1LL*ar1*ar2>=ar3?1LL*ar1*ar2%ar3:ar1*ar2;
}int qpow(int ar1,int ar2,int ar3){
        int h1=ar1,ans=1,h2=ar2;
        while(h2){
                if(h2&1) ans=Mul(ans,h1,ar3);
                h1=Mul(h1,h1,ar3);h2>>=1;
        }return ans;
}int Get_SumId(LL ar1,int ar2){
        int now=Get_Mod(ar1,ar2);
        return Mul(Mul(now,Plus(now,1,Mod),Mod),Anti2,Mod);
}void Init_Multive(int ar1){
        notPrime[0]=notPrime[1]=1;f[1]=1;
        for(int i=2;i<=ar1;i++){
                if(!notPrime[i]){
                        Prime[++Prime[0]]=i;
```

```
                        Min_Pow[i]=i;Min_cnt[i]=1;f[i]=(i^1);
                        SumPId[i]=Plus(SumPId[i-1],i,Mod);
                        SumPf[i]=Plus(SumPf[i-1],f[i],Mod);
                }else{
                        SumPf[i]=SumPf[i-1];SumPId[i]=SumPId[i-1];
                }for(int j=1;j<=Prime[0]&&i*Prime[j]<=ar1;j++){
                        notPrime[i*Prime[j]]=1;
                        if(i%Prime[j]==0){
                                Min_Pow[i*Prime[j]]=Min_Pow[i]*Prime[j];
                                Min_cnt[i*Prime[j]]=Min_cnt[i]+1;
                                f[i*Prime[j]]=Mul(f[i/Min_Pow[i]],Prime[j]^(Min_cnt[i]+1
                                break;
                        }else{
                                Min_Pow[i*Prime[j]]=Prime[j];
                                Min_cnt[i*Prime[j]]=1;
                                f[i*Prime[j]]=Mul(f[i],f[Prime[j]],Mod);
                        }
                }
        }
}void Init_All(LL ar1){
        for(LL l=1,r;l<=ar1;l=r+1){
                r=ar1/(ar1/l);
                All[++All[0]]=ar1/l;
        }std::reverse(All+1,All+All[0]+1);
        for(int i=1,j=1;i<=All[0];i++){
                while(j<=Prime[0]&&1LL*Prime[j]*Prime[j]<=All[i]) j++;
                BreakFrom[i]=j-1;
        }
}void Init_PSumf(LL ar1){
        for(int i=1;i<=All[0];i++){
                PSumId0[i]=Minu(Get_Mod(All[i],Mod),1,Mod);
                PSumId[i]=Minu(Get_SumId(All[i],Mod),1,Mod);
        }for(int i=1;i<=Prime[0];i++){
                for(int j=All[0];j>0;j--){
                        LL nowfrom=All[j]/Prime[i];
                        if(nowfrom<Prime[i]) break;
                        int nowpos=nowfrom<=Cube_Size?nowfrom:All[0]-(ar1/nowfrom)+1;
                        PSumId0[j]=Minu(PSumId0[j],Minu(PSumId0[nowpos],(i-1),Mod),Mod)
                        PSumId[j]=Minu(PSumId[j],Mul(Minu(PSumId[nowpos],i>1?SumPId[Pri
                }
        }for(int i=1;i<=All[0];i++){
                PSumf[i]=Plus(Minu(PSumId[i],PSumId0[i],Mod),All[i]>1?2:0,Mod);
                if(All[i]>=Cube_Size) PSumf[i]=Minu(PSumf[i],SumPf[Cube_Size],Mod);
        }
}int Get_NPSumf(LL ar1){
        for(int i=1;i<=All[0];i++) NPSumf[i]=1;
        for(int i=Prime[0];i>0;i--){
                for(int j=All[0];j>0;j--){
                        if(BreakFrom[j]<i) break;
                        LL nowfrom=All[j]/Prime[i];
                        for(int k=1;nowfrom;k++,nowfrom/=Prime[i]){
```

```
                                int nowpos=nowfrom<=Cube_Size?nowfrom:All[0]-(ar1/nowfr
                                int LastFrom=Prime[std::max(i,BreakFrom[nowpos])];
                                int LastNeed=std::min(nowfrom,Cube_Size);
                                NPSumf[j]=Plus(NPSumf[j],
                                Mul(Plus(NPSumf[nowpos],LastFrom<LastNeed?Minu(SumPf[La
                        }
                }
        }return NPSumf[All[0]];
}int Get_Sumf(LL ar1){
        Cube_Size=sqrt(ar1);Init_Multive(Cube_Size);
        Init_All(ar1);Init_PSumf(ar1);
        int ans=Get_NPSumf(ar1);
        for(int i=1,j=All[0];i<=Cube_Size;i++,j--) ans=Plus(ans,Mul(f[i],PSumf[j],Mod),
        return ans;
}int main(){
        #ifndef JUDGE
        clock_t t0=clock();
        #endif
        LL n;fscanf(Input,"%lld",&n);
        Anti2=qpow(2,Mod-2,Mod);
        fprintf(Output,"%d\n",Get_Sumf(n));
        #ifndef JUDGE
        clock_t t1=clock();
        fprintf(stdout,"a.cpp %lfms %lfMB\n",(t1-t0)*1000.0/CLOCKS_PER_SEC,(&m2-&m1)/10
        #endif
        fclose(Input);fclose(Output);return 0;
}
```

快速数论变换

```
int r[N];
void ntt(int *x, int lim, int opt) {
  int i, j, k, m, gn, g, tmp;
  for (i = 0; i < lim; ++i)
    if (r[i] < i) swap(x[i], x[r[i]]);
  for (m = 2; m <= lim; m <<= 1) {
    k = m >> 1;
    gn = qpow(3, (P - 1) / m);
    for (i = 0; i < lim; i += m) {
      g = 1;
      for (j = 0; j < k; ++j, g = 1ll * g * gn % P) {
        tmp = 1ll * x[i + j + k] * g % P;
        x[i + j + k] = (x[i + j] - tmp + P) % P;
        x[i + j] = (x[i + j] + tmp) % P;
      }
    }
  }
  if (opt == -1) {
    reverse(x + 1, x + lim);
    int inv = qpow(lim, P - 2);
    for (i = 0; i < lim; ++i) x[i] = 1ll * x[i] * inv % P;
  }
}
int A[N], B[N], C[N];
char a[N], b[N];
int main() {
  int i, lim(1), n;
  scanf("%s", &a);
  n = strlen(a);
  for (i = 0; i < n; ++i) A[i] = a[n - i - 1] - '0';
  while (lim < (n << 1)) lim <<= 1;
  scanf("%s", &b);
  n = strlen(b);
  for (i = 0; i < n; ++i) B[i] = b[n - i - 1] - '0';
  while (lim < (n << 1)) lim <<= 1;
  for (i = 0; i < lim; ++i) r[i] = (i & 1) * (lim >> 1) + (r[i >> 1] >> 1);
  ntt(A, lim, 1);
  ntt(B, lim, 1);
  for (i = 0; i < lim; ++i) C[i] = 1ll * A[i] * B[i] % P;
  ntt(C, lim, -1);
  int len(0);
  for (i = 0; i < lim; ++i) {
    if (C[i] >= 10) len = i + 1, C[i + 1] += C[i] / 10, C[i] %= 10;
    if (C[i]) len = max(len, i);
  }
  while (C[len] >= 10) C[len + 1] += C[len] / 10, C[len] %= 10, len++;
  for (i = len; ~i; --i) putchar(C[i] + '0');
```

```
  puts("");
  return 0;
}
```