数学-HYX

三种集合幂卷积的正变换和逆变换

```
void fwtXor(ll* a, int len) {
    if(len == 1) return;
    int h = len >> 1;
    fwtXor(a, h);
    fwtXor(a + h, h);
    for(int i = 0; i < h; ++i) {
        ll x1 = a[i];
        ll x2 = a[i + h];
        a[i] = (x1 + x2) % MOD;
        a[i + h] = (x1 - x2 + MOD) % MOD;
    }
}
void ifwtXor(ll* a, int len) {
    if(len == 1) return;
    int h = len >> 1;
    for(int i = 0; i < h; ++i) {
        // y1=x1+x2
        // y2=x1-x2
        ll y1 = a[i];
        ll y2 = a[i + h];
        a[i] = (y1 + y2) * invTwo % MOD;
        a[i + h] = (y1 - y2 + MOD) * invTwo % MOD;
    }
    ifwtXor(a, h);
    ifwtXor(a + h, h);
}
void fwtAnd(ll* a, int len) {
    if(len == 1) return;
    int h = len >> 1;
    fwtAnd(a, h);
    fwtAnd(a + h, h);
    for(int i = 0; i < h; ++i) {
        ll x1 = a[i];
        ll x2 = a[i + h];
        a[i] = (x1 + x2) % MOD;
        a[i + h] = x2;
    }
}
void ifwtAnd(ll* a, int len) {
    if(len == 1) return;
    int h = len >> 1;
    for(int i = 0; i < h; ++i) {
        // y1=x1+x2
        // y2=x2
        ll y1 = a[i];
        ll y2 = a[i + h];
        a[i] = (y1 - y2 + MOD) % MOD;
        a[i + h] = y2;
    }
    ifwtAnd(a, h);
    ifwtAnd(a + h, h);
}
void fwtOr(ll* a, int len) {
    if(len == 1) return;
    int h = len >> 1;
    fwtOr(a, h);
    fwtOr(a + h, h);
    for(int i = 0; i < h; ++i) {
        ll x1 = a[i];
        ll x2 = a[i + h];
        a[i] = x1;
        a[i + h] = (x2 + x1) % MOD;
    }
}
void ifwtOr(ll* a, int len) {
    if(len == 1) return;
    int h = len >> 1;
    for(int i = 0; i < h; ++i) {
        // y1=x1
        // y2=x2+x1
        ll y1 = a[i];
        ll y2 = a[i + h];
        a[i] = y1;
        a[i + h] = (y2 - y1 + MOD) % MOD;
    }
    ifwtOr(a, h);
    ifwtOr(a + h, h);
}
```

大步小步算法

```
struct Node {
    ll x, y;
};
vector <Node> G[MAX];
ll hash(ll a) {
    return a % MOD;
}
ll find(ll a) {
    ll u = hash(a);
    for (ll i = 0; i < G[u].size(); i++){
        if (G[u][i].x == a) return G[u][i].y;
    }
    return -1LL;
}
void init() {
    for (ll i = 0; i <= MOD; i++) G[i].clear();
}
ll mod_mul(ll a, ll b, ll N) {
    ll ret = 0;
    while(b) {
        if(b & 1) ret = (ret + a) % N;
        a = 2 * a % N;
        b >>= 1;
    }
    return ret;
}
ll mod_pow(ll x, ll n, ll N) {
    ll ret = 1;
    x %= N;
    while(n) {
        if (n & 1) ret = mod_mul(ret, x, N);
        x = mod_mul(x, x, N);
        n >>= 1;
    }
    return ret;
}
void ex_gcd(ll a, ll b, ll& d,ll& x, ll& y){
    if (b == 0){
        d = a;
        x = 1;
        y = 0;
        return;
    }
    ex_gcd(b, a%b, d, y, x);
    y -= x * (a / b);
}
ll inv(ll a, ll N){
    ll d, x, y;
    ex_gcd(a, N, d, x, y);
    if (d == 1) return (x + N) % N;
    return -1;
}
// 大步小步算法求 a ^ x = b mod N 中的 x
ll mod_log(ll a, ll b, ll N){
    ll m = (ll)sqrt(N + 0.5);
    ll s = 1LL;
    for (ll i = 0; i <= m; i++){
        if(find(s) < 0) {
            G[hash(s)].push_back((Node){s,i});
        }
        s = mod_mul(s, a, N);
    }
    ll v = inv(mod_pow(a, m, N), N);                    //求a^m的逆元
    for (ll i = 0; i < m; i++){
        ll res = find(b);
        if (res != -1) return i * m + res;
        b = mod_mul(b, v, N);
    }
    return -1LL; //无解
}
```

轮廓线dp，$n*m$填$1*2$

```
int n, m, cur;
ull dp[2][(1 << 10) + 5];
void update(int past, int now) {
    if(now >> m & 1) dp[cur][now ^ (1 << m)] = (dp[cur][now ^ (1 << m)] + dp[1-cur][past]);
}
int main() {
    while(scanf("%d%d", &n, &m) != EOF) {
        if(n < m) swap(n, m);
```

```
        cur = 0;
        dp[0][(1 << m) - 1] = 1;
        for(int i = 0; i < n; i++) {
            for(int j = 0; j < m; j++) {
                cur ^= 1;
                CLR(dp[cur]);
                for(int state = 0; state < 1 << m; state++) {
                    update(state, state << 1);
                    if(i && !((1 << m - 1) & state)) update(state, (state << 1) ^ (1 << m) ^ 1);
                    if(j && !(state & 1)) update(state, (state << 1) ^ 3);
                }
            }
        }
        printf("%llu\n", dp[cur][(1 << m) - 1]);
    }
    return 0;
}
```

CDQ-NTT

```
const int N = 5e5 + 10, INF = 0x3f3f3f3f, MOD = 1004535809;
const int G = 3, P = 1004535809; // MOD的原根,当且仅当 g^(MOD-1) = 1 % MOD

typedef long long ll;

ll Pow(ll x, ll n) {
    ll ret = 1;
    for(; n; n >>= 1) {
        if(n & 1) ret = ret * x % P;
        x = x * x % P;
    }
    return ret;
}

ll A[N], B[N];
void rader(ll* y, int len) {
    for(int i = 1, j = len / 2; i < len - 1; i++) {
        if(i < j) swap(y[i], y[j]);
        int k = len / 2;
        while(j >= k) {j -= k; k /= 2;}
        if(j < k) j += k;
    }
}
void ntt(ll* y, int len, int op) {
    rader(y, len);
    for(int h = 2; h <= len; h <<= 1) {
        ll wn = Pow(G, (P - 1) / h);
        if(op == -1) wn = Pow(wn, P - 2);
        for(int j = 0; j < len; j += h) {
            ll w = 1;
            for(int k = j; k < j + h / 2; k++) {
                ll u = y[k];
                ll t = w * y[k + h / 2] % P;
                y[k] = (u + t) % P;
                y[k + h / 2] = (u - t + P) % P;
                w = w * wn % P;
            }
        }
    }
    if(op == -1) {
        ll inv = Pow(len, P - 2);
        for(int i = 0; i < len; i++) y[i] = y[i] * inv % P;
    }
}

ll f[N], g[N];


// dp[i] = (i-1)! * sigma(a[j] * b[i-1-j])
void cdq(int l, int r) {
    if(l == r) return;
    int mid = (l + r) >> 1;
    cdq(l, mid);

    int len = 1;
    while(len <= r - l + 1) len <<= 1;
    for(int i = 0; i < len; i++) {
        A[i] = a[i];
        B[i] = (l + i <= mid ? finv[l + i] * f[l + i] % P : 0);
    }
    ntt(A, len, 1);
    ntt(B, len, 1);
    for(int i = 0; i < len; i++) A[i] = A[i] * B[i] % P;
    ntt(A, len, -1);
```

```
    for(int i = mid + 1; i <= r; i++) {
        f[i] += fact[i - 1] * A[i - 1 - 1] % P;
        f[i] %= P;
    }
    cdq(mid + 1, r);
}
```

快速计算1-n素数个数

```
#define MAXN 100
#define MAXM 50010
#define MAXP 666666
#define MAX 1000001
#define clr(ar) memset(ar, 0, sizeof(ar))
#define chkbit(ar, i) (((ar[(i) >> 6]) & (1 << (((i) >> 1) & 31))))
#define setbit(ar, i) (((ar[(i) >> 6]) |= (1 << (((i) >> 1) & 31))))
#define isprime(x) (( (x) && ((x)&1) && (!chkbit(ar, (x)))) || ((x) == 2))

namespace pcf {
long long dp[MAXN][MAXM];
unsigned int ar[(MAX >> 6) + 5] = {0};
int len = 0, primes[MAXP], counter[MAX];

void Sieve() {
    setbit(ar, 0), setbit(ar, 1);
        for(int i = 3; (i * i) < MAX; i++, i++) {
            if(!chkbit(ar, i)) {
                int k = i << 1;
                for(int j = (i * i); j < MAX; j += k) setbit(ar, j);
            }
        }

        for(int i = 1; i < MAX; i++) {
            counter[i] = counter[i - 1];
            if(isprime(i)) primes[len++] = i, counter[i]++;
        }
    }

    void init() {
        Sieve();
        for(int n = 0; n < MAXN; n++) {
            for(int m = 0; m < MAXM; m++) {
                if(!n) dp[n][m] = m;
                else dp[n][m] = dp[n - 1][m] - dp[n - 1][m / primes[n - 1]];
            }
        }
    }

    long long phi(long long m, int n) {
        if(n == 0) return m;
        if(primes[n - 1] >= m) return 1;
        if(m < MAXM && n < MAXN) return dp[n][m];
        return phi(m, n - 1) - phi(m / primes[n - 1], n - 1);
    }

    long long Lehmer(long long m) {
        if(m < MAX) return counter[m];

        long long w, res = 0;
        int i, a, s, c, x, y;
        s = sqrt(0.9 + m), y = c = cbrt(0.9 + m);
        a = counter[y], res = phi(m, a) + a - 1;
        for(i = a; primes[i] <= s;
                i++) res = res - Lehmer(m / primes[i]) + Lehmer(primes[i]) - 1;
        return res;
    }
}

long long solve(long long n) {
    int i, j, k, l;
    long long x, y, res = 0;

    for(i = 0; i < pcf::len; i++) {
        x = pcf::primes[i], y = n / x;
        if((x * x) > n) break;
        res += (pcf::Lehmer(y) - pcf::Lehmer(x));
    }

    for(i = 0; i < pcf::len; i++) {
        x = pcf::primes[i];
        if((x * x * x) > n) break;
        res++;
    }
```

```
    return res;
}
// 初始化
pcf::init();
// 调用输出1-n的素数个数
cout << pcf::Lehmer(n) << endl;
```

中国剩余定理

```
void exgcd(ll a, ll b, ll& d, ll& x, ll& y) {
    if(!b) {d = a; x = 1; y = 0;}
    else {exgcd(b, a%b, d, y, x); y -= x * (a / b);}
}
ll inv(ll a, ll n) { //mod n
    ll d, x, y;
    exgcd(a, n, d, x, y);
    return d == 1 ? (x + n) % n : -1;
}
/*
x = ai(mod mi)
m1 * m2 * ... * mi = M
mi之间互素
x = sigma(ai * Mi * Mi(-1)) % M
Mi = M / mi
Mi(-1) = Mi 模mi的逆
*/
ll mul(ll a, ll b, ll M) {
    ll ret = 0;
    while(b) {
        if(b & 1) ret = (ret + a) % M;
        a = a * 2 % M;
        b >>= 1;
    }
    return ret;
}
ll CRT(ll a[], ll m[], ll n) {
    ll M = 1;
    ll ans = 0;
    for(int i = 1; i <= n; i++)
        M *= m[i];
    for(int i = 1; i <= n; i++) {
        ll Mi = M / m[i];
        ll x, y, d;
        exgcd(Mi, m[i], d, x, y);
        ans = (ans + mul(x, mul(a[i], Mi, M), M)) % M;
    }
    if(ans < 0) ans += M;
    return ans;
}
/*
当mi不互素的时候
两两合并求答案
*/
bool Merge(ll a1, ll m1, ll a2, ll m2, ll &a3, ll &m3)
{
    ll d = __gcd(m1, m2);
    ll c = a2 - a1;
    if(c % d) return false;
    c = (c % m2 + m2) % m2;
    m1 /= d;
    m2 /= d;
    c /= d;
    c *= inv(m1, m2);
    c %= m2;
    c *= m1 * d;
    c += a1;
    m3 = m1 * m2 * d;
    a3 = (c % m3 + m3) % m3;
    return true;
}

ll CRT(ll a[], ll m[], int n) {
    ll a1 = a[1];
    ll m1 = m[1];
    for(int i=2; i<=n; i++) {
        ll a2 = a[i];
        ll m2 = m[i];
        ll m3, a3;
        if(!Merge(a1, m1, a2, m2, a3, m3))
            return -1;
        a1 = a3;
        m1 = m3;
    }
    return (a1 % m1 + m1) % m1;
```

```
}
```

FFT

```
struct Virt {
    double r, i;
    Virt(double _r = 0, double _i = 0) : r(_r), i(_i) {}
    Virt operator+ (Virt& rhs) {
        return Virt(r + rhs.r, i + rhs.i);
    }
    Virt operator- (Virt& rhs) {
        return Virt(r - rhs.r, i - rhs.i);
    }
    Virt operator* (Virt& rhs) {
        return Virt(r * rhs.r - i * rhs.i, r * rhs.i + i * rhs.r);
    }
};
void Rader(Virt F[], int len) {
    int j = len >> 1;
    for(int i = 1; i < len - 1; i++) {
        if(i < j) swap(F[i], F[j]);
        int k = len >> 1;
        while(j >= k) {
            j -= k;
            k >>= 1;
        }
        if(j < k) j += k;
    }
}
void FFT(Virt F[], int len, int on) {
    Rader(F, len);
    for(int h = 2; h <= len; h <<= 1) {
        Virt wn(cos(-on*2*PI/h), sin(-on*2*PI/h));
        for(int j = 0; j < len; j += h) {
            Virt w(1, 0);
            for(int k = j; k < j + h / 2; k++) {
                Virt u = F[k];
                Virt t = w * F[k + h / 2];
                F[k] = u + t;
                F[k + h / 2] = u - t;
                w = w * wn;
            }
        }
    }
    if(on == -1)
        for(int i = 0; i < len; i++)
            F[i].r /= len;
}
// 黑科技，FFT取膜 cov(p, q)
void Cov(int p[], int q[], int m) {
    int t = sqrt(MOD), len = 1;
    while(len < 2 * m) len <<= 1;
    for(int i = 0; i < len; i++) {
        p1[i] = (i < m ? p[i] / t : 0);
        p2[i] = (i < m ? p[i] % t : 0);
        p3[i] = 0;
        q1[i] = (i < m ? q[i] / t : 0);
        q2[i] = (i < m ? q[i] % t : 0);
    }
    FFT(p1, len, 1), FFT(p2, len, 1), FFT(q1, len, 1), FFT(q2, len, 1);
    for(int i = 0; i < len; i++) {
        p3[i] = p1[i] * q2[i] + p2[i] * q1[i];
        p1[i] = p1[i] * q1[i];
        p2[i] = p2[i] * q2[i];
    }
    FFT(p1, len, -1), FFT(p2, len, -1), FFT(p3, len, -1);
    for(int i = 0; i < len; i++) {
        ll t1 = p1[i].r + 0.5;
        ll t2 = p2[i].r + 0.5;
        ll t3 = p3[i].r + 0.5;
        p[i] = (t1 * t * t + t * t3 + t2) % MOD;
    }
    for(int i = m; i < len; i++) {
        p[i%m] = (p[i%m] + p[i]) % MOD;
        p[i] = 0;
    }
}
```

康拓展开

```
int fac[10] = {1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880};
int use[10];
```

```
string Kangtuo(int x) {
    CLR(use);
    string res = "";
    for (int i = 8; ~i; i--) {
        int d = x / fac[i];
        for (int i = 0; i <= d; i++) if (use[i]) d++;
        res += d + '1';
        use[d] = 1;
        x %= fac[i];
    }
    return res;
}

int kangtuo(string &x) {
    CLR(use);int res = 0;
    for (int i = 0; x[i]; i++) {
        int t = x[i] - '0', d = t--;
        for (int j = 1; j < d; j++) if (use[j]) t--;
        res += t * fac[8 - i];
        use[d] = 1;
    }
    return res;
}
```

Lucas定理

```
ll F[MOD + 10];
void init(ll p) {
    F[0] = 1;
    for(int i = 1;i <= p;i++)
        F[i] = F[i-1]*i % MOD;
}

ll inv(ll a, ll m) {
    if(a == 1) return 1;
    return inv(m % a, m) * (m - m / a) % m;
}
返回C(n, m) % p
ll lucas(ll n, ll m, ll p)
{
    ll ans = 1;
    while(n && m)
    {
        ll a = n % p;
        ll b = m % p;
        if(a < b) return 0;
        ans = ans * F[a] % p * inv(F[b] * F[a-b] % p, p) % p;
        n /= p;
        m /= p;
    }
    return ans;
}
```

大素数判断以及大整数分解

```
map <ll, ll> mp;
map <ll, ll> :: iterator iter;
ll Random(ll n) {
    return ((double) rand() / RAND_MAX * n + 0.5);
}
ll q_mul(ll a, ll b, ll mod) {
    ll ans = 0;
    while(b) {
        if(b & 1) ans = (a + ans) % mod;
        b >>= 1;
        a = (a + a) % mod;
    }
    return ans;
}
ll q_pow(ll a, ll b, ll mod) {
    ll ret = 1;
    while(b) {
        if(b & 1) ret = q_mul(ret, a, mod);
        b >>= 1;
        a = q_mul(a, a, mod);
    }
    return ret;
}
bool witness(ll a, ll n) {
    ll d = n - 1;
    while(d % 2 == 0) d >>= 1;
    ll t = q_pow(a, d, n);
    while(d != n - 1 && t != 1 && t != n - 1) {
```

```
        t = q_mul(t, t, n);
        d <<= 1;
    }
    return t == n - 1 || d & 1;
}
bool Isprime(ll p, ll n) {
    if(n == 2) return true;
    if(n < 2 || !(n & 1)) return false;
    int cnt = 20;
    while(cnt--) {
        ll base = Random(n - 2) + 1;
        if(!witness(base, p)) return false;
    }
    return true;
}
ll pollard_rho(ll n, ll c) {
    ll x, y, d, i = 1, k = 2;
    x = Random(n - 2) + 1;
    y = x;
    while(1) {
        i++;
        x = (q_mul(x, x, n) + c) % n;
        d = __gcd(y - x, n);
        if(1 < d && d < n) return d;
        if(y == x) return n;
        if(i == k) {
            y = x;
            k <<= 1;
        }
    }
}
// 分解n，c是任意传入的一个数，分解的数放在mp里面
void find(ll n, ll c) {
    if(n == 1) return;
    if(Isprime(n, n)) {
        mp[n]++;
        return;
    }
    ll p = n;
    while(p >= n)
        p = pollard_rho(p, c--);
    find(p, c);
    find(n / p, c);
}
```

莫比乌斯反演

```
int mu[maxn];
int primes[maxn], tot = 0;
int vis[maxn];
ll sum[maxn];
void Mobius(int n) {
    mu[1] = 1;
    tot = 0;
    for(int i = 2; i <= n; i++) {
        if(!vis[i]) {
            mu[i] = -1;
            primes[tot++] = i;
        }
        for(int j = 0; j < tot && i * primes[j] <= n; j++) {
            vis[i*primes[j]] = 1;
            if(i % primes[j] != 0) mu[i*primes[j]] = -mu[i];
            else {
                mu[i*primes[j]] = 0;
                break;
            }
        }
    }
    for(int i = 1; i <= n; i++) {
        sum[i] = sum[i - 1] + mu[i];
    }
}
// 求（a,b）互质的点对数
ll solve(ll a, ll b) {
    ll ans = 0;
    if(a > b) swap(a, b);
    for(ll i = 1, la = 0; i <= a; i = la + 1) {
        la = min(a / (a / i), b / (b / i));
        ans += (ll)(sum[la] - sum[i - 1]) * (a / i) * (b / i);
    }
    return ans;
}
```

simpson公式（积分）

```
//calc1是可以直接积分的部分，减少精度误差
double calc1(double x) {
    return a/4.0*x*x*x*x+b/3.0*x*x*x;
}
//f是除了直接积分部分的原函数
double f(double x)
{
    return c * x * log(x);//自定义函数
}
double simpson(double a,double b)
{
    double c = a + (b - a) / 2;
    return (f(a) + 4 * f(c) + f(b)) * (b - a) / 6;
}
//递归过程，可以保存la,rb,和simpson的值，若la,rb未变
//则直接使用保存的simpson值不用再调用simpson函数，
//若改变则调用函数并保存修改后的值
//eps为自定义精度
double asr(double a, double b, double eps, double A)
{
    double c = a + (b - a) / 2;
    double L = simpson(a, c), R = simpson(c, b);
    if(abs(L + R - A) <= 15 * eps) return L + R + (L + R - A) / 15.0;
    return asr(a, c, eps/2, L) + asr(c, b, eps/2, R);
}
//调用此函数
double asr_main(double a, double b, double eps)
{
    return asr(a, b, eps, simpson(a,b));
}
```

线性规划

```
// 改进单纯性法的实现
// 参考：http://en.wikipedia.org/wiki/Simplex_algorithm
// 输入矩阵a描述线性规划的标准形式。a为m+1行n+1列，其中行0~m-1为不等式，行m为目标函数（最大化）。列0~n-1为变量0~n-1的系数，列n为常数项
// 第i个约束为a[i][0]*x[0] + a[i][1]*x[1] + ... <= a[i][n]
// 目标为max(a[m][0]*x[0] + a[m][1]*x[1] + ... + a[m][n-1]*x[n-1] - a[m][n])
// 注意：变量均有非负约束x[i] >= 0
const int maxm = 500; // 约束数目上限
const int maxn = 500; // 变量数目上限
const double INF = 1e100;
const double eps = 1e-10;

struct Simplex {
  int n; // 变量个数
  int m; // 约束个数
  double a[maxm][maxn]; // 输入矩阵
  int B[maxm], N[maxn]; // 算法辅助变量

  void pivot(int r, int c) {
    swap(N[c], B[r]);
    a[r][c] = 1 / a[r][c];
    for(int j = 0; j <= n; j++) if(j != c) a[r][j] *= a[r][c];
    for(int i = 0; i <= m; i++) if(i != r) {
      for(int j = 0; j <= n; j++) if(j != c) a[i][j] -= a[i][c] * a[r][j];
      a[i][c] = -a[i][c] * a[r][c];
    }
  }

  bool feasible() {
    for(;;) {
      int r, c;
      double p = INF;
      for(int i = 0; i < m; i++) if(a[i][n] < p) p = a[r = i][n];
      if(p > -eps) return true;
      p = 0;
      for(int i = 0; i < n; i++) if(a[r][i] < p) p = a[r][c = i];
      if(p > -eps) return false;
      p = a[r][n] / a[r][c];
      for(int i = r+1; i < m; i++) if(a[i][c] > eps) {
        double v = a[i][n] / a[i][c];
        if(v < p) { r = i; p = v; }
      }
      pivot(r, c);
    }
  }

  // 解有界返回1，无解返回0，无界返回-1。b[i]为x[i]的值，ret为目标函数的值
  int simplex(int n, int m, double x[maxn], double& ret) {
    this->n = n;
```

```
    this->m = m;
    for(int i = 0; i < n; i++) N[i] = i;
    for(int i = 0; i < m; i++) B[i] = n+i;
    if(!feasible()) return 0;
    for(;;) {
      int r, c;
      double p = 0;
      for(int i = 0; i < n; i++) if(a[m][i] > p) p = a[m][c = i];
      if(p < eps) {
        for(int i = 0; i < n; i++) if(N[i] < n) x[N[i]] = 0;
        for(int i = 0; i < m; i++) if(B[i] < n) x[B[i]] = a[i][n];
        ret = -a[m][n];
        return 1;
      }
      p = INF;
      for(int i = 0; i < m; i++) if(a[i][c] > eps) {
        double v = a[i][n] / a[i][c];
        if(v < p) { r = i; p = v; }
      }
      if(p == INF) return -1;
      pivot(r, c);
    }
  }
};
//////////////// 题目相关
#include<cmath>
Simplex solver;
int main() {
  int n, m;
  while(scanf("%d%d", &n, &m) == 2) {
    for(int i = 0; i < n; i++) scanf("%lf", &solver.a[m][i]); // 目标函数
    solver.a[m][n] = 0; // 目标函数常数项
    for(int i = 0; i < m; i++)
      for(int j = 0; j < n+1; j++)
        scanf("%lf", &solver.a[i][j]);
    double ans, x[maxn];
    assert(solver.simplex(n, m, x, ans) == 1);
    ans *= m;
    printf("Nasa can spend %d taka.\n", (int)floor(ans + 1 - eps));
  }
  return 0;
}
```