0 Basic

0.1 Compiling

```
g++ std=c++17 -O2 -Wall -Wextra -Wshadow -Wconversion -
    fsanitize=address -fsanitize=undefined "%" -o "%:r"
```

0.2 .vimrc

```
sy on
set ru nu cin cul sc so=3 ts=4 sw=4 bs=2 ls=2
inoremap {<CR> {<CR>}<C-o>O
map <F7> :w<CR>:!g++ "%" -Wall -Wextra -Wshadow -Wconversion
     -fsanitize=address -fsanitize=undefined -
    D_GLIBCXX_DEBUG -o /owo/run<CR>:!/owo/run<CR>
```

0.3 spiltmix64

```
struct custom_hash
{
    static uint64_t splitmix64(uint64_t x)
    {
        x += 0x9e3779b97f4a7c15;
        x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
        x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
        return x ^ (x >> 31);
    }
    size_t operator()(uint64_t x) const
    {
        static const uint64_t FIXED_RANDOM = chrono::
            steady_clock::now().time_since_epoch().count();
        return splitmix64(x + FIXED_RANDOM);
    }
};
```

0.4 mt19937

```
mt19937 rd(chrono::steady_clock::now().time_since_epoch().
    count());
```

0.5 PBDS

```
#include <bits/extc++.h>
using namespace __gnu_pbds;

#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
tree<int, null_type, less<int>, rb_tree_tag,
    tree_order_statistics_node_update> bst;
// order_of_key(n): # of elements <= n
// find_by_order(n): 0-indexed

#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/priority_queue.hpp>
__gnu_pbds::priority_queue<int, greater<int>,
    pairing_heap_tag> pq;
```

1 Math

1.1 Extended GCD

```
// beware of negative numbers!
void extgcd(ll a, ll b, ll c, ll &x, ll &y)
{
    if (b == 0)
        x = c / a, y = 0;
    else
    {
        extgcd(b, a % b, c, y, x);
        y -= x * (a / b);
    }
}
```

2 Polynomial

2.1 NTT (FFT)

```
#define base ll // complex<double>
#define N 524288
// const double PI = acosl(-1);
const ll mod = 998244353, g = 3;
base omega[4 * N], omega_[4 * N];
int rev[4 * N];

ll fpow(ll b, ll p);

ll inverse(ll a) { return fpow(a, mod - 2);}

void calcW(int n)
{
    ll r = fpow(g, (mod - 1) / n), invr = inverse(r);
    omega[0] = omega_[0] = 1;
    for (int i = 1; i < n; i++)
    {
        omega[i] = omega[i - 1] * r % mod;
        omega_[i] = omega_[i - 1] * invr % mod;
    }
    // double arg = 2.0 * PI / n;
    // for (int i = 0; i < n; i++)
    // {
    //     omega[i] = base(cos(i * arg), sin(i * arg));
    //     omega_[i] = base(cos(-i * arg), sin(-i * arg));
    // }
}

void calcrev(int n)
{
    int k = __lg(n);
    for (int i = 0; i < n; i++)
        rev[i] = 0;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < k; j++)
            if (i & (1 << j))
                rev[i] ^= 1 << (k - j - 1);
}

vector<base> NTT(vector<base> poly, bool inv)
{
    base *w = (inv ? omega_ : omega);
    int n = poly.size();
    for (int i = 0; i < n; i++)
        if (rev[i] > i)
            swap(poly[i], poly[rev[i]]);

    for (int len = 1; len < n; len <<= 1)
    {
        int arg = n / len / 2;
        for (int i = 0; i < n; i += 2 * len)
            for (int j = 0; j < len; j++)
            {
                base odd = w[j * arg] * poly[i + j + len] %
                    mod;
                poly[i + j + len] = (poly[i + j] - odd + mod
                    ) % mod;
                poly[i + j] = (poly[i + j] + odd) % mod;
            }
    }
    if (inv)
        for (auto &a : poly)
            a = a * inverse(n) % mod;
    return poly;
}

vector<base> mul(vector<base> f, vector<base> g)
{
    int sz = 1 << (__lg(f.size() + g.size() - 1) + 1);
    f.resize(sz), g.resize(sz);
    calcrev(sz);
    calcW(sz);
    f = NTT(f, 0), g = NTT(g, 0);
    for (int i = 0; i < sz; i++)
        f[i] = f[i] * g[i] % mod;
    return NTT(f, 1);
}
```

$p = 1004535809 = 479 \times 2^{21} + 1, g = 3$
$p = 998244353 = 119 \times 2^{23} + 1, g = 3$
$p = 9223372036737335297 = 54975513881 \cdot 2^{24} + 1, g = 3$
$\omega = g^{\frac{p-1}{n}}$

2.2 Inverse

```
vector<base> inv(vector<base> A)
{
    A.resize(1 << (__lg(A.size() - 1) + 1));
    vector<base> B = {inverse(A[0])};
    for (int n = 1; n < A.size(); n += n)
    {
        vector<base> pA(A.begin(), A.begin() + 2 * n);
        calcrev(4 * n);
        calcW(4 * n);
        pA.resize(4 * n);
        B.resize(4 * n);
        pA = NTT(pA, 0);
        B = NTT(B, 0);
        for (int i = 0; i < 4 * n; i++)
            B[i] = ((B[i] * 2 - pA[i] * B[i] % mod * B[i]) %
                mod + mod) % mod;
        B = NTT(B, 1);
        B.resize(2 * n);
    }
    return B;
}
```

2.3 Division

$A = QB + R.$

```
pair<vector<base>, vector<base>> div(vector<base> A, vector<
    base> B)
{
    if (A.size() < B.size())
        return make_pair(vector<base>(), A);
    int n = A.size(), m = B.size();
    vector<base> revA = A, invrevB = B;
    reverse(revA.begin(), revA.end());
    reverse(invrevB.begin(), invrevB.end());
    revA.resize(n - m + 1);
    invrevB.resize(n - m + 1);
    invrevB = inv(invrevB);

    vector<base> Q = mul(revA, invrevB);
    Q.resize(n - m + 1);
    reverse(Q.begin(), Q.end());
    vector<base> R = mul(Q, B);
    R.resize(m - 1);
    for (int i = 0; i < m - 1; i++)
        R[i] = (A[i] - R[i] + mod) % mod;
    return make_pair(Q, R);
}
```

2.4 Linear Recurrence

```
ll fast_kitamasa(ll k, vector<base> A, vector<base> C)
{
    int n = A.size();
    C.emplace_back(mod - 1);
    vector<base> Q, R = {0, 1}, F = {1};

    R = div(R, C);
    while (k)
    {
        if (k & 1)
            F = div(mul(F, R), C);
        R = div(mul(R, R), C);
        k >>= 1;
    }
    ll ans = 0;
    for (int i = 0; i < F.size(); i++)
        ans = (ans + A[i] * F[i]) % mod;
    return ans;
}
```

2.5 Sparse Power

Find first m term of f^p in $\mathcal{O}(nm)$

```
vector<ll> fpow(vector<ll> f, ll p, ll m)
{
    int b = 0;
    while (b < f.size() && f[b] == 0)
        b++;
    f = vector<ll>(f.begin() + b, f.end());
    int n = f.size();
    f.emplace_back(0);
    vector<ll> q(min(m, b * p), 0);
    q.emplace_back(fpow(f[0], p));
    for (int k = 0; q.size() < m; k++)
    {
        ll res = 0;
        for (int i = 0; i < min(n, k + 1); i++)
            res = (res + p * (i + 1) % mod * f[i + 1] % mod
                * q[k - i + b * p]) % mod;
        for (int i = 1; i < min(n, k + 1); i++)
            res = (res - f[i] * (k - i + 1) % mod * q[k - i
                + 1 + b * p]) % mod;
        res = (res < 0 ? res + mod : res) * inv(f[0] * (k +
            1) % mod) % mod;
        q.emplace_back(res);
    }
    return q;
}
```

3 Geometry

3.1 Pick's Theorem

$$A = I + \frac{B}{2} - 1$$

Area A, lattice points on Boundary B, in interior I.

4 String

4.1 Hash Primes

1000000021, 1066636547, 1111111121, 1234567891, 1800399103
1820303297, 2000000011, 5681288813, 21182087419

4.2 KMP

```
int KMP(string s, string t)
{
    cin >> s >> t;
    int n = t.size(), ans = 0;
    vector<int> f(t.size(), 0);
    f[0] = -1;
    for (int i = 1, j = -1; i < t.size(); i++)
    {
        while (j >= 0)
            if (t[j + 1] == t[i]) break;
            else j = f[j];
        f[i] = ++j;
    }
    for (int i = 0, j = 0; i < s.size(); i++)
    {
        while (j >= 0)
            if (t[j + 1] == s[i]) break;
            else j = f[j];
        if (++j + 1 == t.size())
            ans++, j = f[j];
    }
    return ans;
}
```

4.3 Z

```
void z(string s)
{
    for (int i = 1, mx = 0; i < s.size(); i++)
    {
        if (i < Z[mx] + mx)
            Z[i] = min(Z[mx] - i + mx, Z[i - mx]);
        while (Z[i] + i < s.size() && s[i + Z[i]] == s[Z[i
            ]])
            Z[i]++;
        if (Z[i] + i > Z[mx] + mx)
            mx = i;
    }
}
```

4.4 Manacher

```
int man[2000006];
int manacher(string s)
{
    string t;
    for (int i = 0; i < s.size(); i++)
    {
        if (i)
            t.push_back('$');
        t.push_back(s[i]);
    }
    int mx = 0, ans = 0;
    for (int i = 0; i < t.size(); i++)
    {
        man[i] = 1;
        man[i] = min(man[mx] + mx - i, man[2 * mx - i]);
        while (man[i] + i < t.size() && i - man[i] >= 0 && t
            [i + man[i]] == t[i - man[i]])
            man[i]++;
        if (i + man[i] > mx + man[mx])
            mx = i;
    }
    for (int i = 0; i < t.size(); i++)
        ans = max(ans, man[i] - 1);
    return ans;
}
```

4.5 Minimum Rotation

```
int mincyc(string s)
{
    int n = s.size();
    s = s + s;
    int i = 0, ans = 0;
    while (i < n)
    {
        ans = i;
        int j = i + 1, k = i;
        while (j < s.size() && s[j] >= s[k])
        {
            k = (s[j] > s[k] ? i : k + 1);
            ++j;
        }
        while (i <= k)
            i += j - k;
    }
    return ans;
}
```

4.6 Aho Corasick

```
#define sigma 26
struct AhoCorasick
{
    int ch[500005][sigma] = {{}}, f[500005] = {-1}, cnt
        [500005];
    int idx = 0;
    int insert(string &s)
    {
        int j = 0;
        for (int i = 0; i < s.size(); i++)
        {
            if (!ch[j][s[i] - 'a'])
                ch[j][s[i] - 'a'] = ++idx;
            j = ch[j][s[i] - 'a'];
        }
        return j;
    }
    void build()
    {
        queue<int> q;
        q.push(0);
        while (!q.empty())
        {
            int u = q.front();
            q.pop();
            for (int v = 0; v < sigma; v++)
                if (ch[u][v])
                {
                    int cur = f[u];
                    while (cur >= 0)
                        if (ch[cur][v])
                            break;
                        else
                            cur = f[cur];
                    f[ch[u][v]] = (cur < 0 ? 0 : ch[cur][v])
                        ;
                    q.push(ch[u][v]);
                }
        }
    }
    void match(string &s)
    {
        for (int i = 0; i <= idx; i++)
            cnt[i] = 0;
        for (int i = 0, j = 0; i < s.size(); i++)
        {
            while (j >= 0)
                if (ch[j][s[i] - 'a'])
                    break;
                else
                    j = f[j];
            j = (j < 0 ? 0 : ch[j][s[i] - 'a']);
            cnt[j]++;
        }
        vector<int> v;
        v.emplace_back(0);
        for (int i = 0; i < v.size(); i++)
            for (int j = 0; j < sigma; j++)
                if (ch[v[i]][j])
                    v.emplace_back(ch[v[i]][j]);
        reverse(v.begin(), v.end());
        for (int i : v)
            if (f[i] > 0)
                cnt[f[i]] += cnt[i];
    }
};
```

4.7 Suffix Array

```
void SA(string s)
{
    int n = s.size();
    sa.resize(n), cnt.resize(n), rk.resize(n), tmp.resize(n)
        ;
    iota(sa.begin(), sa.end(), 0);
    sort(sa.begin(), sa.end(), [&](int i, int j)
        { return s[i] < s[j]; });
    for (int i = 1; i < n; i++)
        rk[sa[i]] = rk[sa[i - 1]] + (s[sa[i - 1]] != s[sa[i
            ]]);
    for (int k = 1; k <= n; k <<= 1)
    {
        fill(cnt.begin(), cnt.end(), 0);
        for (int i = 0; i < n; i++)
            cnt[rk[(sa[i] - k + n) % n]]++;
        for (int i = 1; i < n; i++)
            cnt[i] += cnt[i - 1];
        for (int i = n - 1; i >= 0; i--)
            tmp[--cnt[rk[(sa[i] - k + n) % n]]] = (sa[i] - k
                + n) % n;
        sa.swap(tmp);
        tmp[sa[0]] = 0;
        for (int i = 1; i < n; i++)
            tmp[sa[i]] = tmp[sa[i - 1]] + (rk[sa[i - 1]] !=
                rk[sa[i]] || rk[(sa[i - 1] + k) % n] != rk
                [(sa[i] + k) % n]);
        rk.swap(tmp);
    }
}

void LCP(string s)
{
    int n = s.size(), k = 0;
    lcp.resize(n);
    for (int i = 0; i < n; i++)
        if (rk[i] == 0)
            lcp[rk[i]] = 0;
        else
        {
            if (k)
                k--;
            int j = sa[rk[i] - 1];
            while (i + k < n && j + k < n && s[i + k] == s[j
                + k])
                k++;
            lcp[rk[i]] = k;
        }
}
```