link-cut tree

```
namespace lct {
  struct Node {
    int size, fa, son[2];
    bool rev;
  } tree[N];

  bool isRoot(int x) {
    return tree[tree[x].fa].son[0] != x && tree[tree[x].fa].son[1] != x;
  }

  int which(int x) {
    return tree[tree[x].fa].son[1] == x;
  }

  void apply(int x) {
    swap(tree[x].son[0], tree[x].son[1]);
    tree[x].rev ^= 1;
  }

  void pushDown(int x) {
    if (tree[x].rev) {
      apply(tree[x].son[0]), apply(tree[x].son[1]);
      tree[x].rev = 0;
    }
  }

  void pushUp(int x) {
    tree[x].size = 1 + tree[tree[x].son[0]].size + tree[tree[x].son[1]].size;
  }

  void rotate(int x) {
    int y = tree[x].fa, z = tree[y].fa;
    int id = which(x), p = tree[x].son[1 - id];
    if (!isRoot(y)) tree[z].son[which(y)] = x;
    tree[x].fa = z, tree[y].son[id] = p, tree[p].fa = y;
    tree[x].son[1 - id] = y, tree[y].fa = x;
    pushUp(y), pushUp(x);
  }

  void dfs(int root) {
    if (!isRoot(root)) dfs(tree[root].fa);
    pushDown(root);
  }

  void Splay(int x) {
    dfs(x);
    while (!isRoot(x)) {
      int y = tree[x].fa;
      if (!isRoot(y)) {
        rotate((which(x) == which(y)) ? y : x);
      }
      rotate(x);
    }
```

```
  }

  // access u -> root
  int access(int u) {
    int tmp = 0;
    while (u != 0) {
      Splay(u), tree[u].son[1] = tmp, pushUp(u);
      tmp = u, u = tree[u].fa;
    }
    return tmp;
  }

  // find the root of this tree.
  int findRoot(int root) {
    access(root), Splay(root);
    int tmp = root;
    while (tree[tmp].son[0]) {
      pushDown(tmp);
      tmp = tree[tmp].son[0];
    }
    Splay(tmp);
    return tmp;
  }

  // make the node become root.
  void makeRoot(int root) {
    access(root), Splay(root), apply(root);
  }

  // link edge u -> v
  void link(int u, int v) {
    makeRoot(u); tree[u].fa = v;
  }

  // cut edge u -> v
  void cut(int u, int v) {
    makeRoot(u), access(v), Splay(v);
    tree[v].son[0] = tree[u].fa = 0, pushUp(v);
  }
}
```

Steiner Tree

$O(n3^k)$

```
// maximum node num, key node num, infinity.
const int N = 1010, K = 5, inf = 0x3f3f3f3f;

struct edge { int v, w; };

vector<edge> G[N];
int dp[1 << K][N], a[N];
bool inq[N];
queue<int> Q;

void init() {
  memset(dp, 0x3f, sizeof dp);
  for (int i = 0; i < N; i++) {
```

```
    G[i].clear();
  }
}

void addEdge(int u, int v, int w);

void bellmanFord(int *dp) {
  while (!Q.empty()) {
    int u = Q.front(); Q.pop(), inq[u] = 0;
    for (auto& e : G[u]) {
      if (dp[e.v] > dp[u] + e.w) {
        dp[e.v] = dp[u] + e.w;
        if (!inq[e.v]) {
          Q.push(e.v), inq[e.v] = 1;
        }
      }
    }
  }
}

// k -> key node num, n -> node num.
void solve(int k, int n) {
  for (int S = 1; S < (1 << k); S++) {
    for (int i = 0; i < n; i++) {
      for (int s = (S - 1) & S; s; s = (s - 1) & S) {
        dp[S][i] = min(dp[S][i], dp[s][i] + dp[s ^ S][i]);
      }
      if (dp[S][i] < inf) Q.push(i), inq[i] = 1;
    }
    bellmanFord(dp[S]);
  }
}
```

ZhuLiu's Algo

$O(nm)(?)$

```
const int N = 1010, M = 1010101, inf = 0x3f3f3f3f;

// directed edges.
// r -> root, x_i -> u, y_i -> v, z_i -> w
int r, dfn, x[M], y[M], z[M], last[N], weight[N], id[N];
bool vis[N], instk[N];

void dfs(int u) {
  if (u == r) return;
  instk[u] = vis[u] = 1;
  if (!vis[last[u]]) {
    dfs(last[u]);
  } else if (instk[last[u]]) {
    id[u] = ++dfn;
    for (int v = last[u]; v != u; v = last[v]) {
      id[v] = dfn;
    }
  }
  instk[u] = 0;
  if (!id[u]) id[u] = ++dfn;
}
```

```cpp
// n -> nodes num, m -> edges num.
int solve(int n, int m) {
  int ans = 0;
  while (true) {
    for (int i = 1; i <= n; i++) {
      weight[i] = inf;
    }
    for (int i = 1; i <= m; i++) {
      if (z[i] < weight[y[i]]) {
        last[y[i]] = x[i], weight[y[i]] = z[i];
      }
    }
    memset(id, 0, sizeof id);
    memset(vis, 0, sizeof vis);
    id[r] = dfn = 1;
    for (int i = 1; i <= n; i++) {
      if (i == r) continue;
      if (weight[i] == inf) return -1;
      ans += weight[i];
      if (!vis[i]) dfs(i);
    }
    if (dfn == n) return ans;
    int cnt = 0;
    for (int i = 1; i <= m; i++) {
      if (id[x[i]] != id[y[i]]) {
        z[++cnt] = z[i] - weight[y[i]];
        x[cnt] = id[x[i]], y[cnt] = id[y[i]];
      }
    }
    m = cnt, n = dfn, r = id[r];
  }
  return -1;
}
```

CartesianTree

```cpp
// left son, right son, father.
int l[N], r[N], fa[N];

void build(int n, int arr[]) {
  static int top, stack[N];
  top = 0, stack[top++] = 1;
  for (int i = 2; i <= n; i++) {
    while (top && arr[stack[top - 1]] > arr[i]) --top;
    if (top > 0) {
      int x = i, y = stack[top - 1];
      l[x] = r[y], fa[l[x]] = x, fa[x] = y, r[y] = x;
      stack[top++] = x;
    } else {
      fa[stack[0]] = i, l[i] = stack[0];
      stack[top++] = i;
    }
  }
}
```

AC Automaton

```
template <int N, int charset> class acam {
  int tot;
  int fail[N], endpos[N];
  int son[N][charset];

  int encode(int c) { return c - 'A'; }
public:

  int fa[N], length[101010];
  bool vis[N];

  void initNode(int i) {
    fail[i] = 0, endpos[i] = -1;
    memset(son[i], 0, sizeof son[i]);
  }

  void insert(char *s, int index = 0) {
    int cur = 0;
    for (int i = 0; s[i]; i++) {
      int c = encode(s[i]);
      if (!son[cur][c]) {
        son[cur][c] = ++tot;
        initNode(tot);
      }
      fa[son[cur][c]] = cur;
      cur = son[cur][c];
      length[index] = i + 1;
    }
    endpos[index] = cur;
  }
  void build() {
    queue<int> Q;
    for (int i = 0; i < charset; i++) {
      if (son[0][i]) Q.push(son[0][i]);
    }
    while (!Q.empty()) {
      int u = Q.front(); Q.pop();
      for (int i = 0; i < charset; i++) {
        if (son[u][i]) {
          fail[son[u][i]] = son[fail[u]][i];
          Q.push(son[u][i]);
        } else {
          son[u][i] = son[fail[u]][i];
        }
      }
    }
  }
};
```

pb_ds

```
#include<bits/extc++.h>
// 下面是set的例子 map的话将__gnu_pbds::null_type改成想要的结构即可
```

```
typedef __gnu_pbds::tree<int, __gnu_pbds::null_type, less<int>,
__gnu_pbds::rb_tree_tag, __gnu_pbds::tree_order_statistics_node_update> ordered_set;
/*
  iterator find_by_order(size_type order)
    找到第order+1小的迭代器，如果order太大会返回end()
  size_type order_of_key(const_key_reference r_key)
    询问这个tree中有多少个比r_key小的元素
  void join(tree &other)
    把other中所有元素移动到*this中（要求原来other和*this的key不能相交，否则会抛出异常）
*/
typedef __gnu_pbds::priority_queue<int, less<int>> pq;
/*
  void join(priority_queue &other)
    把other合并到*this，然后other会被清空
*/
```

树哈希

$f_{now} = 1 + \sum f_{son_{now,i}} \times prime(siz[son_{now,i}])$

> Notes 其中f_x为以节点x为根的子树对应的哈希值。 $siz[x]$表示以节点x为根的子树大小。 $son_{x,i}$表示x的所有子节点之一。 prime(i)表示第i个质数。 选树的重心作根可保证一棵树仅有两个哈希值

线性递推

```
namespace LinearRecurrence {
  VL ReedsSloane(const VL &s, ll Mod) {
    function<void(VL &, size_t)> extend = [](VL &v, size_t d) {
      if(d <= v.size()) return;
      v.resize(d, 0);
    };
    function<ll(ll, ll)> inverse = [](ll a, ll m) {
      pll ret = crt::exgcd(a, m);
      ll g = a * ret.first + m * ret.second;
      if(g != 1) return -1ll;
      return ret.first >= 0 ? ret.first : ret.first + m;
    };
    function<int(const VL&, const VL&)> L = [](const VL &a, const VL &b) {
      int da = (a.size() > 1 || (a.size() == 1 && a[0])) ? a.size() - 1 : -1000;
      int db = (b.size() > 1 || (b.size() == 1 && b[0])) ? b.size() - 1 : -1000;
      return max(da, db + 1);
    };
    function<pair<VL, VL>(const VL&, ll, ll, ll)> prime_power = [&](const VL &s, ll
Mod, ll p, ll e) {
      vector<VL> a(e), b(e), an(e), bn(e), ao(e), bo(e);
      VL t(e), u(e), r(e), to(e, 1), uo(e), pw(e + 1);
      pw[0] = 1;
      for(int i = pw[0] = 1; i <= e; i++) pw[i] = pw[i - 1] * p;
      for(ll i = 0; i < e; i++) {
        a[i] = {pw[i]}, an[i] = {pw[i]};
        b[i] = {0}, bn[i] = {s[0] * pw[i] % Mod};
        t[i] = s[0] * pw[i] % Mod;
        if(t[i] == 0) t[i] = 1, u[i] = e;
        else {
          for(u[i] = 0; t[i] % p == 0; t[i] /= p, u[i]++);
```

```
      }
    }
    for(size_t k = 1; k < s.size(); k++) {
      for(int g = 0; g < e; g++) {
        if(L(an[g], bn[g]) > L(a[g], b[g])) {
          ao[g] = a[e - 1 - u[g]];
          bo[g] = b[e - 1 - u[g]];
          to[g] = t[e - 1 - u[g]];
          uo[g] = u[e - 1 - u[g]];
          r[g] = k - 1;
        }
      }
      a = an, b = bn;
      for(int o = 0; o < e; o++) {
        ll d = 0;
        for(size_t i = 0; i < a[o].size() && i <= k; i++) {
          d = (d + a[o][i] * s[k - i]) % Mod;
        }
        if(d == 0) t[o] = 1, u[o] = e;
        else {
          for(u[o] = 0, t[o] = d; t[o] % p == 0; t[o] /= p, u[o]++);
          int g = e - 1 - u[o];
          if(L(a[g], b[g]) == 0) {
            extand(bn[o], k + 1);
            bn[o][k] = (bn[o][k] + d) % Mod;
          } else {
            ll coef = t[o] * inverse(to[g], Mod) % Mod * pw[u[o] - uo[g]] % Mod;
            int m = k - r[g];
            extand(an[o], ao[g].size() + m);
            extand(bn[o], bo[g].size() + m);
            for(size_t i = 0; i < ao[g].size(); i++) {
              an[o][i + m] -= coef * ao[g][i] % Mod;
              if(an[o][i + m] < 0) an[o][i + m] += Mod;
            }
            while(an[o].size() && an[o].back() == 0) an[o].pop_back();
            for(size_t i = 0; i < bo[g].size(); i++) {
              bn[o][i + m] -= coef * bo[g][i] % Mod;
              if(bo[o][i + m] < 0) bn[o][i + m] += Mod;
            }
            while(bn[o].size() && bn[o].back() == 0) bn[o].pop_back();
          }
        }
      }
    }
    return make_pair(an[0], bn[0]);
  };

  vector<tuple<ll, ll, int>> fac;
  for(ll i = 2; i * i <= Mod; i++) {
    if(Mod % i == 0) {
      ll cnt = 0, pw = 1;
      while(Mod % i == 0) Mod /= i, ++cnt, pw *= i;
      fac.emplace_back(pw, i, cnt);
    }
  }
  if(Mod > 1) fac.emplace_back(Mod, Mod, 1);
  vector<VL> as;
  size_t n = 0;
  for(auto &&x: fac) {
    ll mod, p, e;
    VL a, b;
```

```
    tie(mod, p, e) = x;
    auto ss = s;
    for(auto &&x: ss) x %= mod;
    tie(a, b) = prime_power(ss, mod, p, e);
    as.emplace_back(a);
    n = max(n, a.size());
  }
  VL a(n);
  vector<pll> c(as.size());
  for(size_t i = 0; i < n; i++) {
    for(size_t j = 0; j < as.size(); j++) {
      c[j].first = get<0>(fac[j]);
      c[j].second = i < as[j].size() ? as[j][i] : 0;
    }
    a[i] = crt::crt(c).second;
  }
  return a;
}

VL BM(const VL &s, ll Mod) {
  VL C(1, 1), B(1, 1);
  int L = 0, m = 1, b = 1;
  for(size_t n = 0; n < s.size(); n++) {
    ll d = 0;
    for(int i = 0; i <= L; i++) {
      d = (d + C[i] * s[n - i]) % Mod;
    }
    if(d == 0) ++m;
    else if(2 * L <= int(n)) {
      VL T = C;
      ll inv = crt::exgcd(b, Mod).first;
      if(inv < 0) inv += Mod;
      ll c = Mod - d * inv % Mod;
      while(C.size() < B.size() + m) C.push_back(0);
      for(size_t i = 0; i < B.size(); i++) C[i + m] = (C[i + m] + c * B[i]) % Mod;
      L = n + 1 - L, B = T, b = d, m = 1;
    } else {
      ll inv = crt::exgcd(b, Mod).first;
      if(inv < 0) inv += Mod;
      ll c = Mod - d * inv % Mod;
      while(C.size() < B.size() + m) C.push_back(0);
      for(size_t i = 0; i < B.size(); i++) C[i + m] = (C[i + m] + c * B[i]) % Mod;
      ++m;
    }
  }
  return C;
}

int m;
VL ini, trans;
ll Mod;
void init(const VL &s, ll md, VL A={}) {
  Mod = md;
  if(A.empty()) {
    A = ReedsSloane(s, Mod);
    // A = BM(s, Mod);
  }
  if(A.empty()) A = {0};
  m = A.size() - 1;
  trans.resize(m);
  for(int i = 0; i < m; i++) {
```

```
      trans[i] = (Mod - A[i + 1]) % Mod;
    }
    reverse(trans.begin(), trans.end());
    ini = {s.begin(), s.begin() + m};
  }

  ll calc(ll n) {
    if(Mod == 1) return n;
    if(n < m) return ini[n];
    VL v(m), u(m << 1);
    int msk = !!n;
    for(ll m = n; m > 1; m >>= 1) msk <<= 1;
    v[0] = 1 % Mod;
    for(int x = 0; msk; msk >>= 1, x <<= 1) {
      fill_n(u.begin(), m * 2, 0);
      x |= !!(n & msk);
      if(x < m) u[x] = 1 % Mod;
      else {
        for(int i = 0; i < m; i++) {
          for(int j = 0, t = i + (x & 1); j < m; j++, t++) {
            u[t] = (u[t] + v[i] * v[j]) % Mod; // to better
          }
        }
        for(int i = m * 2 - 1; i >= m; i--) {
          for(int j = 0, t = i - m; j < m; j++, t++) {
            u[t] = (u[t] + trans[j] * u[i]) % Mod; // to better
          }
        }
      }
      v = {u.begin(), u.begin() + m};
    }
    ll ret = 0;
    for(int i = 0; i < m; i++) {
      ret = (ret + v[i] * ini[i]) % Mod; // to better
    }
    return ret;
  }
}
```

线性基

```
typedef unsigned long long ull;
struct LB {
  const static int L = 64; // insert(x) 0 <= x < (1 << L)
  ull a[L];
  LB() { this->init(); }
  void init() { memset(a, 0, sizeof a); }
  ull &operator[](const size_t &id) { return a[id]; }
  const ull &operator[](const size_t &id) const { return a[id]; }
  // 询问x是否在线性基中可以仿造下面的函数来写
  // 即将`return true;`上面三行删去 然后把返回值取反
  // 插入一个数x ==> obj(x) 一边插入一边高斯消元 O(L)
  bool operator()(ull x) {
    for(int i = L - 1; ~i; i--) {
      if((x >> i) & 1) {
        if(!a[i]) {
          for(int j = 0; j < i; j++) if((x >> j) & 1) x ^= a[j];
```

```cpp
          for(int j = i + 1; j < L; j++) if((a[j] >> i) & 1) a[j] ^= x;
          a[i] = x;
          return true;
        } else {
          x ^= a[i];
        }
      }
      if(!x) {
        return false;
      }
    }
    return true;
  }
  // 线性基求交 O(L^2)
  friend LB operator&(const LB &A, const LB &B) {
    LB C, D, E;
    for(int i = L - 1; ~i; i--) {
      if(A[i]) {
        C(A[i]);
        D[i] = 1ull << i;
      }
    }
    for(int i = 0; i < L; i++) {
      if(!B[i]) {
        continue;
      }
      bool can = true;
      ull v = 0, x = B[i];
      for(int j = L - 1; ~j; j--) {
        if((x >> j) & 1) {
          if(C[j]) {
            x ^= C[j], v ^= D[j];
          } else {
            can = false, C[j] = x, D[j] = v;
            break;
          }
        }
      }
      if(can) {
        ll m = 0;
        for(int j = L - 1; ~j; j--) {
          if((v >> j) & 1) {
            m ^= A[j];
          }
        }
        E(m);
      }
    }
    return E;
  }
  // 线性基求并 O(L^2)
  friend LB operator|(const LB &x, const LB &y) {
    LB z;
    for(int i = 0; i < L; i++) if(x[i]) z(x[i]);
    for(int i = 0; i < L; i++) if(y[i]) z(y[i]);
    return z;
  }
};
```

线性基区间最大值

```
namespace LBRMQ {
  const int N = 1e6 + 10, L = 32;
  int b[N][L], pre[N][L];
  void init() {
    memset(b[0], 0, sizeof b[0]);
    memset(pre[0], 0, sizeof pre[0]);
  }
  // index start from 1
  void add(int x, int r) {
    int maxl = r;
    memcpy(b[r], b[r - 1], sizeof(int) * L);
    memcpy(pre[r], pre[r - 1], sizeof(int) * L);
    for(int i = L - 1; ~i; i--) {
      if((x >> i) & 1) {
        if(!b[r][i]) {
          b[r][i] = x;
          pre[r][i] = maxl;
          return;
        }
        if(pre[r][i] < maxl) {
          swap(pre[r][i], maxl);
          swap(b[r][i], x);
        }
        x ^= b[r][i];
      }
    }
  }
  int query(int l, int r) {
    int ans = 0;
    for(int i = L - 1; ~i; i--) {
      if(pre[r][i] >= l) {
        ans = max(ans, ans ^ b[r][i]);
      }
    }
    return ans;
  }
}
```

多项式全家桶

```
// 引用 exgcd 和 fft::multiply_mod
// 确保所有输入的数在[0, MOD)的区间中
// MOD < 1073741823 以及 最好是质数
// 传入的vector为{a_0, a_1, a_2, ..., a_n} 即认定为 y=\sum_{i=0}^{n}a_i\cdot x^i
namespace poly {
  int MOD = 998244353ll;
  vector<int> inv(const vector<int> &a) {
    if(a.size() == 1) {
      const int inv = exgcd(a[0], MOD).first;
      return vector<int>(1, inv < 0 ? inv + MOD : inv);
    }
    const int na = a.size(), nb = (na + 1) >> 1;
    vector<int> b(a.begin(), a.begin() + nb);
    b = inv(b);
```

```
    vector<int> c = fft::multiply_mod(b, b, MOD);
    c.resize(na);
    c = fft::multiply_mod(a, c, MOD);
    b.resize(na), c.resize(na);
    for(int i = 0; i < na; i++) {
      c[i] = (((2ll * b[i] - c[i]) % MOD) + MOD) % MOD;
    }
    return c;
  }

  // A = B * C + D (mod x^n) (n = A.size())
  // always use with the next function mod
  // make sure A.size() >= B.size() or else it will return an empty vector
  vector<int> divide(const vector<int> &a, const vector<int> &b) {
    const int n = a.size(), m = b.size();
    if(n < m) return {};
    vector<int> A(a), B(b);
    reverse(A.begin(), A.end()), reverse(B.begin(), B.end());
    A.resize(n - m + 1), B.resize(n - m + 1);
    B = inv(B);
    vector<int> C = fft::multiply_mod(A, B, MOD);
    C.resize(n - m + 1), reverse(C.begin(), C.end());
    return C;
  }

  vector<int> mod(const vector<int> &a, const vector<int> &b, const vector<int> &c) {
    const int n = a.size(), m = b.size();
    if(n < m) return a;
    vector<int> e = fft::multiply_mod(b, c, MOD);
    e.resize(m - 1);
    for(int i = 0; i < m - 1; i++) {
      e[i] = a[i] - e[i];
      if(e[i] < 0) {
        e[i] += MOD;
      }
    }
    return e;
  }

  // 构造一个多项式 \prod_{i=left}^{right} (x-vec_i)
  vector<int> buildPoly(const vector<int> &vec, const int left, const int right) {
    if(left == right) {
      vector<int> ret;
      ret.push_back(MOD - vec[left]);
      ret.push_back(1);
      return ret;
    }
    const int mid = (left + right) >> 1;
    return fft::multiply_mod(buildPoly(vec, left, mid), buildPoly(vec, mid + 1,
right), MOD);
  }

  void multipointCalc(const vector<int> &poly, const vector<int> &vec, const int left,
const int right, vector<int> &ret) {
    const int n = poly.size(), mid = (left + right) >> 1;
    if(n == 1) {
      for(int i = left; i <= right; i++) {
        ret[i] = poly[0];
      }
      return;
    }
```

```
    const vector<int> b0 = buildPoly(vec, left, mid);
    multipointCalc(mod(poly, b0, divide(poly, b0)), vec, left, mid, ret);
    if(left != right) {
      const vector<int> b1 = buildPoly(vec, mid + 1, right);
      multipointCalc(mod(poly, b1, divide(poly, b1)), vec, mid + 1, right, ret);
    }
  }

  // 多点求值
  vector<int> multipointCalc(const vector<int> &poly, const vector<int> &vec) {
    const int n = vec.size();
    vector<int> ret(n);
    multipointCalc(poly, vec, 0, n - 1, ret);
    return ret;
  }

  vector<int> multiInv(const vector<int> &vec) {
    const int n = vec.size();
    vector<int> a(n + 1), ret(n); a[0] = 1;
    for(int i = 1; i <= n; i++) {
      a[i] = 1ll * a[i - 1] * vec[i - 1] % MOD;
    }
    int cur = (exgcd(a[n], MOD).first + MOD) % MOD;
    for(int i = n - 1; i >= 0; i--) {
      ret[i] = 1ll * cur * a[i] % MOD;
      cur = 1ll * cur * vec[i] % MOD;
    }
    return ret;
  }

  // 快速插值 {{x0, y0}, {x1, y1}, {x2, y2}, ...}
  vector<int> interpolate(const vector<pair<int, int>> &p) {
    const int n = p.size(), n0 = (n + 1) >> 1, n1 = n - n0;
    if(n == 1) {
      return {p[0].second};
    }
    vector<pair<int, int>> p0(p.begin() + n1, p.end());
    vector<int> f0 = interpolate(p0);
    vector<int> x(n);
    for(int i = 0; i < n; i++) {
      x[i] = p[i].first;
    }
    vector<int> g0 = buildPoly(x, n1, n - 1);
    x.resize(n1);
    vector<int> fx = multipointCalc(f0, x), gx = multipointCalc(g0, x);
    gx = multiInv(gx);
    p0.resize(n1);
    for(int i = 0; i < n1; i++) {
      p0[i].first = p[0].first;
      p0[i].second = (p[i].second - fx[i] + MOD) % MOD;
      p0[i].second = 1ll * p0[i].second * gx[i] % MOD;
    }
    fx = interpolate(p0);
    fx = fft::multiply_mod(fx, g0, MOD);
    fx.resize(n), f0.resize(n);
    for(int i = 0; i < n; i++) {
      fx[i] = (fx[i] + f0[i]) % MOD;
    }
    return fx;
  }
}
```

sais

```
namespace SA {
  const size_t sz = 3e5 + 5;
  int bucket[sz], bucket1[sz], sa[sz], rk[sz], ht[sz];
  bool type[sz << 1];
  bool isLMS(const int i, const bool *type) {
    return i > 0 && type[i] && !type[i - 1];
  }

  template<class T>
  void inducedSort(const T &s, int *sa, const int len, const int sigma, const int
bucketSize, bool *type, int *bucket, int *cntbuf, int *p) {
    memset(bucket, 0, sizeof(int) * sigma);
    memset(sa, -1, sizeof(int) * len);
    for (int i = 0; i < len; i++) {
      bucket[s[i]]++;
    }
    cntbuf[0] = bucket[0];
    for (int i = 1; i < sigma; i++) {
      cntbuf[i] = cntbuf[i - 1] + bucket[i];
    }
    for (int i = bucketSize - 1; i >= 0; i--) {
      sa[--cntbuf[s[p[i]]]] = p[i];
    }
    for (int i = 1; i < sigma; i++) {
      cntbuf[i] = cntbuf[i - 1] + bucket[i - 1];
    }
    for (int i = 0; i < len; i++) {
      if (sa[i] > 0 && !type[sa[i] - 1]) {
        sa[cntbuf[s[sa[i] - 1]]++] = sa[i] - 1;
      }
    }
    cntbuf[0] = bucket[0];
    for (int i = 1; i < sigma; i++) {
      cntbuf[i] = cntbuf[i - 1] + bucket[i];
    }
    for (int i = len - 1; i >= 0; i--) {
      if (sa[i] > 0 && type[sa[i] - 1]) {
        sa[--cntbuf[s[sa[i] - 1]]] = sa[i] - 1;
      }
    }
  }
  template<class T>
  void sais(const T &s, int *sa, int len, bool *type, int *bucket, int *bucket1, int
sigma) {
    int i, j, bucketSize = 0, cnt = 0, p = -1, x, *cntbuf = bucket + sigma;
    type[len - 1] = 1;
    for (i = len - 2; i >= 0; i--) {
      type[i] = s[i] < s[i + 1] || (s[i] == s[i + 1] && type[i + 1]);
    }
    for (i = 1; i < len; i++) {
      if (type[i] && !type[i - 1]) {
        bucket1[bucketSize++] = i;
      }
    }
    inducedSort(s, sa, len, sigma, bucketSize, type, bucket, cntbuf, bucket1);
    for (i = bucketSize = 0; i < len; i++) {
      if (isLMS(sa[i], type)) {
        sa[bucketSize++] = sa[i];
```

```
    }
  }
  for (i = bucketSize; i < len; i++) {
    sa[i] = -1;
  }
  for (i = 0; i < bucketSize; i++) {
    x = sa[i];
    for (j = 0; j < len; j++) {
      if (p == -1 || s[x + j] != s[p + j] || type[x + j] != type[p + j]) {
        cnt++, p = x;
        break;
      } else if (j > 0 && (isLMS(x + j, type) || isLMS(p + j, type))) {
        break;
      }
    }
    x = (~x & 1 ? x >> 1 : (x - 1) >> 1), sa[bucketSize + x] = cnt - 1;
  }
  for (i = j = len - 1; i >= bucketSize; i--) {
    if (sa[i] >= 0) {
      sa[j--] = sa[i];
    }
  }
  int *s1 = sa + len - bucketSize, *bucket2 = bucket1 + bucketSize;
  if (cnt < bucketSize) {
    sais(s1, sa, bucketSize, type + len, bucket, bucket1 + bucketSize, cnt);
  } else {
    for (i = 0; i < bucketSize; i++) {
      sa[s1[i]] = i;
    }
  }
  for (i = 0; i < bucketSize; i++) {
    bucket2[i] = bucket1[sa[i]];
  }
  inducedSort(s, sa, len, sigma, bucketSize, type, bucket, cntbuf, bucket2);
}

void getHeight(const vector<int> &s, int n) {
  int i, j, k = 0;
  for(i = 1; i <= n; i++) {
    rk[sa[i]] = i;
  }
  for(i = 0; i < n; i++) {
    if(k) k--;
    for(j = sa[rk[i] - 1]; s[i + k] == s[j + k]; k++);
    ht[rk[i]] = k;
  }
}

template<class T>
void solve(const T &s) {
  const int n = s.size();
  vector<int> v(n + 1);
  int sigma = 0;
  for(int i = 0; i < n; i++) {
    v[i] = int(s[i]) + 1;
    sigma = max(sigma, v[i] + 1);
  }
  v[n] = 0;
  sais(v, sa, n + 1, type, bucket, bucket1, sigma);
  getHeight(v, n);
}
```

```
}
```

旋转卡壳

```
const int N=1e5+7;
struct point{
  double x,y;
}S[N];
inline double Cross(point a,point b,point c){
  return (a.x-c.x)*(b.y-c.y)-(a.y-c.y)*(b.x-c.x);
}
inline double Dis(point a,point b){
  return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
}
//先求凸包再旋转卡壳
double FarthestPointPair(int top){
  if(top==1)return Dis(S[0],S[1]);
  S[++top]=S[0];int j=2;double ans=0;
  for(int i=0;i<top;++i){
    while(Cross(S[i],S[i+1],S[j])<Cross(S[i],S[i+1],S[j+1])) {
      j=(j+1)%top;
    }
    ans=max(ans,max(Dis(S[i],S[j]),Dis(S[i+1],S[j])));
  }
  return ans;
}
```

化行最简形

```
void Gauss(vector<vector<int>> &v) {
  const int m = v.size(), n = v[0].size();
  vector<int> id(n, -1);
  int i = 0, j = 0;
  while(i < m && j < n) {
    int mi = i;
    for(int k = i + 1; k < m; k++) {
      if(v[k][j] > v[mi][j]) {
        mi = k;
      }
    }
    if(v[mi][j] != 0) {
      if(i != mi) {
        for(int k = 0; k < n; k++) {
          swap(v[i][k], v[mi][k]);
        }
      }
      const int inv = fpow(v[i][j], MOD - 2);
      for(int k = j + 1; k < n; k++) {
        v[i][k] = 1ll * v[i][k] * inv % MOD;
      }
      id[j] = i;
      v[i][j] = 1;
      for(int r = i + 1; r < m; r++) {
        for(int c = j + 1; c < n; c++) {
```

```
            v[r][c] -= 1ll * v[r][j] * v[i][c] % MOD;
            if(v[r][c] < 0) v[r][c] += MOD;
          }
          v[r][j] = 0;
        }
        i++;
      }
      j++;
    }
    for(int i = 0; i < m; i++) {
      for(int j = 0; j < n; j++) {
        if(v[i][j] && id[j] > i) {
          for(int k = j + 1; k < n; k++) {
            v[i][k] -= 1ll * v[i][j] * v[id[j]][k] % MOD;
            if(v[i][k] < 0) v[i][k] += MOD;
          }
          v[i][j] = 0;
        }
      }
    }
  }
```

min_25

min_25用途

1. 筛出质数
2. 求出所有的$\sum_{i=2}^{\lfloor\frac{n}{x}\rfloor}[i是质数]f(i)$
3. 求积性函数前缀和

前提

1. 当i为质数时$f(i)$需要是一个多项式。
2. 对于求积性函数前缀和而言$f(p^k)$需要快速求出，求多个值的时候一般不适用min_25筛。

计算

处理质数

$g(a,b)=\sum_{i=2}^{a}[i是质数 或 pmin_i > prime_b]*i^k$

需要求每一个

$$g(\lfloor\frac{n}{x}\rfloor,\infty)=\sum_{i=2}^{\lfloor\frac{n}{x}\rfloor}[i是质数]f(i)$$

那么有

$$g(a,b)=\begin{cases} g(a,b-1), & a < prime_b^2 \\ g(a,b-1)-prime_b^k\left(g(\lfloor\frac{a}{prime_b}\rfloor,b-1)-g(prime_{b-1},b-1)\right), & a \ge prime_b^2 \end{cases}$$

滚动数组叠上去即可求出。

计算前缀和

$S(a,b)=\sum_{i=2}^{a}[pmin_i \ge prime_b]f(i)$

前缀和即

$$\sum_{i=1}^{n} f(i)=S(n,1)+f(1)$$

那么有

$$S(a,b)=\begin{cases} 0, & a < prime_b \\ g(a,\infty)-g(prime_{b-1},\infty)+ \\ \sum_{i=b}^{\infty}\sum_{t\ge 1, prime_i^{t+1}\le a}\left(S(\lfloor\frac{a}{prime_i^t}\rfloor,i+1)*f(prime_i^t)+f(prime_i^{t+1})\right), & a \ge prime_b \end{cases}$$

无需记忆化，递归求解。

防忘代码

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
void Main();
#ifdef ConanYu
#include "local.hpp"
#else
#define debug(...) do { } while(0)
int main() {
  ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
  Main();
}
#endif

const int MOD = 1e9 + 7;
int fpow(int a, int b) {
  int ans = 1;
  for(; b; b >>= 1, a = 1ll * a * a % MOD) {
    if(b & 1) ans = 1ll * ans * a % MOD;
  }
  return ans;
}

const int N = 1e5 + 10, INV2 = fpow(2, MOD - 2);
ll n, a[N << 1];
int sn, cnt, id, prime[N], g[N << 1], h[N << 1];
int idx(ll x) {
  return x <= sn ? x : id - n / x + 1;
}

void sub(int &a, int b) {
  a -= b;
```

```
  if(a < 0) a += MOD;
}

int solve(long long a, int b) {
  if(a < prime[b]) return 0;
  int ans = g[idx(a)];
  sub(ans, g[idx(prime[b - 1])]);
  for(int i = b; i <= cnt && a / prime[i] >= prime[i]; i++) {
    ll k = prime[i], m = k - 1;
    while(a / k >= prime[i]) {
      ans += 1ll * solve(a / k, i + 1) * m % MOD;
      if(ans >= MOD) ans -= MOD;
      m = m * prime[i] % MOD;
      ans += m;
      if(ans >= MOD) ans -= MOD;
      k *= prime[i];
    }
  }
  return ans;
}
void Main() {
  cin >> n;
  sn = sqrt(n);
  cnt = id = 0;
  for(ll i = 1; i <= n; i = a[id] + 1) {
    a[++id] = n / (n / i);
    g[id] = a[id] % MOD * ((a[id] + 1) % MOD) % MOD * INV2 % MOD;
    sub(g[id], 1);
    h[id] = (a[id] - 1) % MOD;
  }
  for(int i = 2; i <= sn; i++) {
    if(h[i] != h[i - 1]) {
      prime[++cnt] = i;
      for(int j = id; a[j] / i >= i; j--) {
        const int ta = idx(a[j] / i), tb = idx(prime[cnt - 1]);
        int A = g[ta], B = h[ta];
        sub(A, g[tb]), sub(B, h[tb]);
        sub(g[j], 1ll * i * A % MOD);
        sub(h[j], B);
      }
    }
  }
  for(int i = 1; i <= id; i++) {
    g[i] -= h[i];
    if(g[i] < 0) g[i] += MOD;
  }
  cout << ((solve(n, 1) + 1) % MOD) << "\n";
}
```