ICPC Notebook

template

hash.sh

```sh
# □□□: sh hash.sh -> □□□ -> Ctrl + D
# □□□□□□□□□□□□□□□ md5 □□□□□□□
g++ -dD -E -fpreprocessed - | tr -d '[:space:]' | md5sum | cut
-c-6
```

settings.sh

```sh
# CLion □□□
Settings → Build → CMake → Reload CMake Project
add_compile_options(-D_GLIBCXX_DEBUG)
# Caps Lock □ Ctrl □□□
setxkbmap -option ctrl:nocaps
```

template.hpp

md5: 365d7f

```cpp
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
const ll INF = LLONG_MAX / 4;
#define rep(i, a, b) for(ll i = a; i < (b); i++)
#define all(a) begin(a), end(a)
ll sz(const auto& a) { return size(a); }
bool chmin(auto& a, auto b) {
   if(a <= b) return 0;
   a = b;
   return 1;
}
bool chmax(auto& a, auto b) {
   if(a >= b) return 0;
   a = b;
   return 1;
}

int main() {
   cin.tie(0)->sync_with_stdio(0);
   // your code here...
}
```

data-structure

BIT.hpp

md5: 1fe3e2

```cpp
struct BIT {
   vector<ll> a;
   BIT(ll n) : a(n + 1) {}
   void add(ll i, ll x) {  // A[i] += x
      i++;
      while(i < sz(a)) {
         a[i] += x;
         i += i & -i;
      }
   }
   ll sum(ll r) {
      ll s = 0;
      while(r) {
         s += a[r];
         r -= r & -r;
      }
      return s;
   }
   ll sum(ll l, ll r) {  // sum of A[l, r)
      return sum(r) - sum(l);
   }
};
```

FastSet.hpp

md5: f86f47

```cpp
// using u64 = uint64_t;
const u64 B = 64;
struct FastSet {
   u64 n;
   vector<vector<u64>> a;
   FastSet(u64 n_) : n(n_) {
      do a.emplace_back(n_ = (n_ + B - 1) / B);
      while(n_ > 1);
   }
   // bool operator[](ll i) const { return a[0][i / B] >> (i %
B) & 1; }
   void set(ll i) {
      for(auto& v : a) {
         v[i / B] |= 1ULL << (i % B);
         i /= B;
      }
   }
   void reset(ll i) {
      for(auto& v : a) {
         v[i / B] &= ~(1ULL << (i % B));
         if(v[i / B]) break;
         i /= B;
      }
   }
   ll next(ll i) {  // i □□□□□□□□□
      rep(h, 0, sz(a)) {
         i++;
         if(i / B >= sz(a[h])) break;
         u64 d = a[h][i / B] >> (i % B);
         if(d) {
            i += countr_zero(d);
            while(h--) i = i * B + countr_zero(a[h][i]);
            return i;
         }
         i /= B;
      }
      return n;
   }
   ll prev(ll i) {  // i □□□□□□□□□□
      rep(h, 0, sz(a)) {
         i--;
         if(i < 0) break;
         u64 d = a[h][i / B] << (~i % B);
         if(d) {
            i -= countl_zero(d);
            while(h--) i = i * B + __lg(a[h][i]);
            return i;
         }
      }
      return -1;
   }
};
```

math

modint

BarrettReduction.hpp

md5: b61c28

```cpp
// using u64 = uint64_t;
struct Barrett {  // mod < 2^32
   u64 m, im;
   Barrett(u64 mod) : m(mod), im(-1ULL / m + 1) {}
   // input: a * b < 2^64, output: a * b % mod
   u64 mul(u64 a, u64 b) const {
      a *= b;
      u64 x = ((__uint128_t)a * im) >> 64;
      a -= x * m;
      if((ll)a < 0) a += m;
      return a;
   }
};
```

modint.hpp

md5: ade70b

```cpp
const ll mod = 998244353;
struct mm {
   ll x;
   mm(ll x_ = 0) : x(x_ % mod) {
      if(x < 0) x += mod;
   }
   friend mm operator+(mm a, mm b) { return a.x + b.x; }
   friend mm operator-(mm a, mm b) { return a.x - b.x; }
   friend mm operator*(mm a, mm b) { return a.x * b.x; }
   friend mm operator/(mm a, mm b) { return a * b.inv(); }
```

```
   // 4 □□□□  Alt + Shift + □□□□□□□□□□□□
   friend mm& operator+=(mm& a, mm b) { return a = a.x + b.x; }
   friend mm& operator-=(mm& a, mm b) { return a = a.x - b.x; }
   friend mm& operator*=(mm& a, mm b) { return a = a.x * b.x; }
   friend mm& operator/=(mm& a, mm b) { return a = a * b.inv(); }
}
   mm inv() const { return pow(mod - 2); }
   mm pow(ll b) const {
      mm a = *this, c = 1;
      while(b) {
         if(b & 1) c *= a;
         a *= a;
         b >>= 1;
      }
      return c;
   }
};
```

FPS

FFT.hpp

md5: 5e6cea

```
// {998244353, 3}, {754974721, 11}, {167772161, 3}, {469762049,
3}, {2130706433, 3}
mm g = 3;  // □□□
void fft(vector<mm>& a) {
   ll n = sz(a), lg = __lg(n);
   static auto z = [] {
      vector<mm> z(30);
      mm s = 1;
      rep(i, 2, 32) {
         z[i - 2] = s * g.pow(mod >> i);
         s *= g.inv().pow(mod >> i);
      }
      return z;
   }();
   rep(l, 0, lg) {
      ll w = 1 << (lg - l - 1);
      mm s = 1;
      rep(k, 0, 1 << l) {
         ll o = k << (lg - l);
         rep(i, o, o + w) {
            mm x = a[i], y = a[i + w] * s;
            a[i] = x + y;
            a[i + w] = x - y;
         }
         s *= z[__builtin_ctzll(~k)];
      }
   }
}
// □□□
void ifft(vector<mm>& a) {
   ll n = sz(a), lg = __lg(n);
   static auto z = [] {
      vector<mm> z(30);
      mm s = 1;
      rep(i, 2, 32) {  // g □□□□
         z[i - 2] = s * g.inv().pow(mod >> i);
         s *= g.pow(mod >> i);
      }
      return z;
   }();
   for(ll l = lg; l--;) {  // □□□
      ll w = 1 << (lg - l - 1);
      mm s = 1;
      rep(k, 0, 1 << l) {
         ll o = k << (lg - l);
         rep(i, o, o + w) {
            mm x = a[i], y = a[i + w];  // *s □□□
            a[i] = x + y;
            a[i + w] = (x - y) * s;
         }
         s *= z[__builtin_ctzll(~k)];
      }
   }
}
vector<mm> conv(vector<mm> a, vector<mm> b) {
   if(a.empty() || b.empty()) return {};
   size_t n_ = sz(a) + sz(b) - 1, n = bit_ceil(n_);
   // if(min(sz(a), sz(b)) <= 60) □□□□□□
   a.resize(n);
   b.resize(n);
   fft(a);
   fft(b);
   mm x = mm(n).inv();
   rep(i, 0, n) a[i] *= b[i] * x;
   ifft(a);
   a.resize(n_);
   return a;
}
```

graph

graph/tree

flow

□□□□□□.md

□□□□□□	□□□□□□
x □ 0 □□□ z □□	(x, T, z)
x □ 0 □□□ z □□	□□□□ z □□; (S, x, z)
x □ 1 □□□ z □□	(S, x, z)
x □ 1 □□□ z □□	□□□□ z □□; (x, T, z)
$x, y, \ldots$ □□□□ 0 □□□ z □□	□□□□ z □□; $(S, w, z), (w, x, \infty), (w, y, \infty)$
$x, y, \ldots$ □□□□ 1 □□□ z □□	□□□□ z □□; $(w, T, z), (x, w, \infty), (y, w, \infty)$

string

algorithm

geometry