Contents

1 Math

1.1 Number Theory

1.1.1 Modular

```
template<typename T> struct M {
  static T MOD;
  T v;
  M() : v(0) {}
  M(T x) {
    v = (-MOD <= x && x < MOD) ? x : x % MOD;
    if (v < 0) v += MOD;
  }
  explicit operator T() { return v; }
  bool operator==(M b) { return v == b.v; }
  bool operator!=(M b) { return v != b.v; }
  M operator+(M b) { return M(v + b.v); }
  M operator-(M b) { return M(v - b.v); }
  M operator*(M b) { return M((__int128)v * b.v % MOD); }
  M operator/(M b) { return *this * (b ^ (MOD - 2)); }
  friend M operator^(M a, ll b) {
    M ans(1);
    for(; b; b >>= 1, a *= a) if(b & 1) ans *= a;
    return ans;
  }
  friend M& operator+=(M& a, M b) { return a = a + b; }
  friend M& operator-=(M& a, M b) { return a = a - b; }
  friend M& operator*=(M& a, M b) { return a = a * b; }
  friend M& operator/=(M& a, M b) { return a = a / b; }
};
using Mod = M<ll>;
template<>ll Mod::MOD = 1000000007;

/* Safe primes
 * 21673, 26497, 22621, 21817, 28393, 26821, 30181, 22093
 * 977680993, 971939533, 970479637, 910870273, 1041012121
 * 741266610070171837, 1110995545625882557
 * NTT prime           | p - 1       | primitive root
 * 65537               | (2^16)      | 3
 * 998244353           | (2^23)*119  | 3
 * 2748779069441       | (2^39)*5    | 3
 * 1945555039024054273 | (2^56)*27   | 5           */
```

1.1.2 Extended GCD

```
tuple<ll, ll, ll> extgcd(ll a, ll b) {
  if (b == 0) return { 1, 0, a };
  else {
    auto [p, q, g] = extgcd(b, a % b);
    return { q, p - q * (a / b), g };
  }
}
```

1.1.3 Chinese Remainder

```
ll crt(ll a, ll m, ll b, ll n) {
  if (n > m) swap(a, b), swap(m, n);
  auto [x, y, g] = extgcd(m, n);
  assert((a - b) % g == 0);  // no solution
  x = (x * (b - a) / g) % (n / g) * m + a;
  return x < 0 ? x + m * n / g : x;
}
```

1.1.4 Tonelli-Shanks

```
int legendre(Mod a) {
  if (a == 0) return 0;
  return (a ^ ((a.MOD - 1) / 2)) == 1 ? 1 : -1;
}
// O(log^2(p)) worst, O(log(p)) most of the time
Mod sqrt(Mod a) {
  assert(legendre(a) != -1);  // no solution
  ll p = a.MOD, s = p - 1;
  if (a == 0) return 0;
  if (p == 2) return 1;
  if (p % 4 == 3) return a ^ ((p + 1) / 4);
  int r, m;
  for (r = 0; !(s & 1); r++) s >>= 1;
  Mod n = 2;
  while (legendre(n) != -1) n += 1;
  Mod x = a ^ ((s + 1) / 2), b = a ^ s, g = n ^ s;
  while (b != 1) {
    Mod t = b;
    for (m = 0; t != 1; m++) t *= t;
    Mod gs = g ^ (1LL << (r - m - 1));
    g = gs * gs, x *= gs, b *= g, r = m;
  }
  return x;
}
```