
# HUS.HAD

Hanoi University of Science

# Mục lục



# 1 Miscellanous

### 1.0.1 Commands on Shell

```
# compile
g++ $1.cpp --std=c++17 -Wall -Wextra -O2 -o $1
diff output.txt answer.txt
# before running shell file
chmod 700 any_shell_file.sh
```

### 1.0.2 Template

```
#include <bits/stdc++.h>
using namespace std;
/*
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
typedef tree<int, null_type, less<int>, rb_tree_tag,
  ↪ tree_order_statistics_node_update> ordered_set;
*/
typedef long long ll;
typedef long double ld;
template<class T>using PQMax=priority_queue<T>;
template<class T>using PQMin=priority_queue<T, vector<T>,
  ↪ greater<T>>;
template<class T1, class T2>
void maximize(T1 &a, T2 b){ if (b > a) a = b; }
template<class T1, class T2>
void minimize(T1 &a, T2 b){ if (b < a) a = b; }
template<class T> void read(T &number) {
  bool negative = false; register int c;
  number = 0; c = getchar();
  while (c!='-' && !isalnum(c)) c = getchar();
  if (c=='-') negative = true, c = getchar();
  for (; (c>47 && c<58); c=getchar())
    number = number*10 + c-48;
  if (negative) number *= -1;
}
template<class T, class ...Ts>
void read(T &a, Ts&... args){
  read(a); read(args...);
}

#define fi first
#define se second
#define FOR(type,i,a,b) for(type i=(a); i<=(b); i++)
#define REV(type,i,b,a) for(type i=(b); i>=(a); i--)
#define testBit(n, bit) (((n) >> (bit)) & 1)
#define flipBit(n, bit) ((n) ^ (1ll << (bit)))
#define cntBit(n) __builtin_popcount(n)
#define cntBitll(n) __builtin_popcountll(n)
#define log2(n) (31 - __builtin_clz(n))
#define log2ll(n) (63 - __builtin_clzll(n))
#define CURRENT_TIMESTAMP
  ↪ chrono::steady_clock::now().time_since_epoch().count()
#define randomize mt19937_64 mt(CURRENT_TIMESTAMP)

// remember to fill in:
// #define MAX ???
// #define MOD ???

// int main()
// {ios_base::sync_with_stdio(0);cin.tie(0);}
```

# 2 Mathematics

## 2.1 Data structure

### 2.1.1 Modulo

```
#include "../../miscellanous/template.hpp"

typedef unsigned long long ull;
ll MOD = 1000000007;
struct modint{
  int v;
  static inline ll mod(ll num) {
    ll val=num-ull((__uint128_t(-1ULL/MOD)*num)>>64)*MOD;
    return val-(val>=MOD)*MOD;
  }
  modint inv() const{
    ll answer = 1, a = v, n = MOD - 2;
    while (n) {
      if (n & 1) answer = mod(answer * a);
      a = mod(a * a); n >>= 1;
    }
    return answer;
  }

  modint(ll a = 0)
    { v=(a<0)?(MOD-mod(-a)):mod(a);v-=(v>=MOD)*MOD; }
  inline modint& operator += (modint b)
    { v+=b.v; v-=(v>=MOD)*MOD; return *this; }
  inline modint& operator -= (modint b)
    { v+=MOD-b.v; v-=(v>=MOD)*MOD; return*this; }
  inline modint& operator *= (modint b)
    { v = mod(1ll * v * b.v); return *this; }
  inline modint& operator /= (modint b)
    { return (*this)*=b.inv(); }
  inline modint& operator ^= (ll n) {
    modint a = v; v = 1;
    while (n) {if (n & 1) *this *= a; a *= a, n >>= 1;}
    return *this;
  }
};
inline modint operator+(modint a, modint b){return a+=b;}
inline modint operator-(modint a, modint b){return a-=b;}
inline modint operator*(modint a, modint b){return a*=b;}
inline modint operator/(modint a, modint b){return a/=b;}
inline modint operator^(modint a, ll n){return a^=n;}
inline bool operator == (modint a, modint b)
  { return a.v==b.v; }
inline bool operator != (modint a, modint b)
  { return a.v!=b.v; }
inline bool operator < (modint a, modint b)
  { return a.v<b.v; }
inline bool operator > (modint a, modint b)
  { return a.v>b.v; }
inline bool operator <= (modint a, modint b)
  { return a.v<=b.v; }
inline bool operator >= (modint a, modint b)
  { return a.v>=b.v; }
inline istream& operator >> (istream& s, modint &i)
  { ll tmp; s >> tmp; i = tmp; return s; }
inline ostream& operator << (ostream& s, modint i)
  {return s << i.v;}
```

### 2.1.2 BigInteger

```

```

## 2.2 Number Theory

### 2.2.1 Mildly-optimized sieve(Factorable)

```
#include "../../miscellanous/template.hpp"

// remember to call sieve() before any factorize()
namespace Eratos_Factorable {
  constexpr ll MAX = 100000000;
  constexpr ll EST = 53000000; // 1.1*MAX/ln(MAX)
  int smallDiv[MAX + 1] = {};
  int primes[EST], cntPrime = 0;
  inline void sieve(int size = MAX) {
    memset(smallDiv, false, sizeof(smallDiv));
    primes[++cntPrime] = 2; primes[++cntPrime] = 3;
    for (int i=2; i<=size; i += 2) smallDiv[i] = 2;
    for (int i=3; i<=size; i += 6) smallDiv[i] = 3;
    for (int mul = 1; 6 * mul - 1 <= size; mul++) {
      bool pass = false;
      for(int i:{6*mul-1,6*mul+1})if(!smallDiv[i]){
        primes[++cntPrime] = i; smallDiv[i] = i;
        for (ll j = 1ll*i*i; j <= size; j += i*2)
          if (not smallDiv[j]) smallDiv[j] = i;
        if (1ll * i * i > size) pass = true;
```

```
      }
      if (pass) break;
    }
  }
  inline vector<int> factorize(ll number) {
    vector<int> ans;
    while (number > 1) {
      int d = smallDiv[number];
      while (number % d == 0)
        ans.push_back(d), number /= d;
    }
    return ans;
  }
}
```

### 2.2.2 Primality check

```
#include "../../miscellanous/template.hpp"

#define MAX 1000001
#define MOD 1000000007

namespace MillerRabin{
  typedef __int128_t i128;
  constexpr int SMALL_PRIMES[12] =
  ↪  {2,3,5,7,11,13,17,19,23,29,31,37};

  inline ll _power(ll a, ll n, ll mod) {
    ll ans = 1;
    while (n) {
      if (n & 1) ans = (i128)ans * a % mod;
      a = (i128)a * a % mod; n >>= 1;
    }
    return ans;
  }
  inline bool _fermatCheck(ll n, ll a, ll pw, int p2) {
    ll num = _power(a, pw, n);
    if (num == 1 || num == n - 1) return true;
    FOR(int, i, 1, p2-1) {
      num = (i128) num * num % n;
      if (num == n - 1) return true;
    }
    return false;
  }

  inline bool checkPrime(const ll n) {
    if (n == 2 || n == 3 || n == 5 || n == 7) return true;
    if (n < 11 || (n & 1) == 0) return false;
    ll d = n-1; int p2 = 0;
    while ((d & 1) == 0) d >>= 1, p2++;
    for (int a: SMALL_PRIMES)
      if (n == a) return true;
      else if (not _fermatCheck(n, a, d, p2))
        return false;
    return true;
  }
}
```

### 2.2.3 Prime factorization

```
#include "millerrabin.hpp"
randomize;
namespace Pollards{
  typedef __int128_t i128;
  #define sqp1(a, b, mod) (((i128)(a)*(a)+(b)))%(mod))
  #define rand (mt()%1'000'000'000'000 + 1)
  inline vector<ll> rho(ll n) {
    if (n == 1) return {};
    if (MillerRabin::checkPrime(n)) return {n};
    vector<ll> ans;
    if (n % 2 == 0) {
      while (n % 2 == 0) ans.push_back(2), n /= 2;
      vector<ll> others = rho(n);
      ans.insert(ans.end(), others.begin(), others.end());
      return ans;
    }
    ll x = 2, y = 2, g = 1, b = 1;
    while (g == 1) {
      x=sqp1(x,b,n); y=sqp1(y,b,n); y=sqp1(y,b,n);
      g = __gcd(abs(x-y), n);
      if (g == n) x=y=rand, b=rand, g=1;
    }
    vector<ll> tmp1 = rho(g), tmp2 = rho(n / g);
    ans.insert(ans.end(), tmp1.begin(), tmp1.end());
    ans.insert(ans.end(), tmp2.begin(), tmp2.end());
    return ans;
  }
}
```

## 2.3 Numerical

### 2.3.1 FFT

```
typedef complex<ld> cd;
namespace FFT{
  constexpr ld TAU =
    3.14159265358979323846264338327950288419716939937510582
    ↪  * 2;

  constexpr int BIT = 20, MAX_LEN = 1 << BIT;
  vector<int> _rev[BIT + 1];
  cd _root[MAX_LEN + 1];
  void buildRoot() {
    _root[0] = _root[MAX_LEN] = 1;
    for (int i=BIT-1, dist=1<<(BIT-1); i>=0; i--,
    ↪  dist>>=1) {
      cd w = polar(ld(1.0), TAU * dist / MAX_LEN);
      for (int pos = 0; pos < MAX_LEN; pos += 2 * dist)
        _root[pos + dist] = _root[pos] * w;
    }
    _rev[0] = {0};
    for (int bit=1, len=2; len<=MAX_LEN; bit++, len*=2) {
      _rev[bit].resize(len, 0);
      for (int i = 0; i < len; i++)
        _rev[bit][i]=((i&1)<<(bit-1))|_rev[bit-1][i/2];
    }
  }
  void dft(vector<cd> &poly, bool invert = false) {
    assert((cntBit((int)poly.size()) == 1));
    const int n = poly.size(), coef = MAX_LEN / n;
    for (int dist=n/2, span=n; dist>0; dist/=2, span/=2)
    for (int pos = 0; pos < dist; pos++)
    for (int i=pos, k=0; i<n; i+=span, k+=dist) {
      int len = log2(n / span);
      int newK = _rev[len][k / dist] * dist;
      int tmp = (invert ? (n - newK) : newK) * coef;
      cd a = poly[i], b = _root[tmp] * poly[i + dist];
      poly[i] = a + b, poly[i + dist] = a - b;
    }
    if (invert) for (cd &x: poly) x /= n;
  }
  template<class T>vector<T>conv(vector<T>_a,vector<T>_b){
    int len = int(_a.size() + _b.size()) - 1;
    int bit = log2(len)+(cntBit(len)>1); len=1<<bit;
    vector<cd> a(len);
    for (int i = 0; i < len; i++)
      a[i] = cd((i < _a.size())?_a[i]:0,
      ↪  (i<_b.size())?_b[i]:0);
    dft(a, false);
    for (int i = 0; i < len; i++)
      if (i < _rev[bit][i]) swap(a[i], a[_rev[bit][i]]);
    for (int i = 0; i < len; i++) a[i] *= a[i];
    vector<cd> b(a.begin(), a.end());
    for (int i = 0; i < len; i++)
      b[i] = a[i] - conj(a[-i & (len-1)]); //(n-i)%n
    dft(b, true);
    for (int i = 0; i < len; i++)
      if (i < _rev[bit][i]) swap(b[i], b[_rev[bit][i]]);

    while (len > 0 && b[len - 1].imag() < 1e-9) len--;
    vector<T> ans(len);
    FOR(int, i, 0, len-1) ans[i]=round(b[i].imag()/4);
    return ans;
  }
}
```

### 2.3.2 NTT

```
#include "../../math/ds/modint.hpp"

// MOD=998244353=c*2^k+1 => ROOT=g^c, any g: gcd(g,MOD)=1
namespace FFT{
  constexpr int K_MOD = 23, BIT = 20, MAX_LEN = 1 << BIT;
  constexpr int ROOT = 15311432;

  modint _root[MAX_LEN + 1];
  vector<int> _rev[BIT + 1];
  void buildRoot() {
    _root[0] = 1; modint mul=ROOT;
```

```
    FOR(int, i, 1, K_MOD-BIT) mul*=mul;
    FOR(int, i, 1, MAX_LEN) _root[i] = _root[i-1] * mul;
    for (int bit=0, len=1; len<=MAX_LEN; bit++, len*=2) {
      _rev[bit].resize((1 << bit), 0);
      for (int i = 1; i < (1 << bit); i++)
        _rev[bit][i]=((i&1)<<(bit-1))|_rev[bit-1][i/2];
    }
  }

  void transform(vector<modint> &v, bool invert){
    const int len = v.size(), coef = MAX_LEN / len;
    const int bit = log2(len);
    FOR(int, i, 0, len-1) if (i < _rev[bit][i])
      swap(v[i], v[_rev[bit][i]]);

    for (int jmp=1, span=2; span<=len; jmp*=2, span*=2)
    for (int beg = 0; beg < len; beg += span)
    for (int i = 0; i < jmp; i++) {
      int k=coef*len/jmp*i/2; if (invert) k=MAX_LEN-k;
      modint a = v[beg+i], b = _root[k] * v[beg+i+jmp];
      v[beg + i] = a + b; v[beg + i + jmp] = a - b;
    }
    if (invert) FOR(int, i, 0, len-1) v[i] /= len;
  }
  template<class T>
  vector<modint> multiply(vector<T> &_a, vector<T> &_b) {
    int len = int(_a.size() + _b.size()) - 1;
    len = 1 << (log2(len) + (cntBit(len) > 1));
    vector<modint>a(_a.begin(),_a.end()); a.resize(len,0);
    vector<modint>b(_b.begin(),_b.end()); b.resize(len,0);
    transform(a, false); transform(b, false);
    FOR(int, i, 0, len - 1) a[i] *= b[i];
    transform(a, true); return a;
  }
}
```