ICPC Notebook

template

hash.sh

```
# 使い方: sh hash.sh -> コピペ -> Ctrl + D
# コメント・空白・改行を削除して md5 でハッシュする
g++ -dD -E -P -fpreprocessed - | tr -d '[:space:]' | md5sum | cut -c-6
```

settings.sh

```
# CLion の設定
Settings → Build → CMake → Reload CMake Project
add_compile_options(-D_GLIBCXX_DEBUG)
# Caps Lock を Ctrl に変更
setxkbmap -option ctrl:nocaps
```

template.hpp

md5: 136d85

```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)
#define sz(a) ssize(a)
bool chmin(auto& a, auto b) { return a > b ? a = b, 1 : 0; }
bool chmax(auto& a, auto b) { return a < b ? a = b, 1 : 0; }

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

data-structure

BIT.hpp

md5: 8133c8

```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: 2cb8c9

```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 を超える最小の要素
      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 より小さい最大の要素
      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;
         }
         i /= B;
      }
      return -1;
   }
};
```

UnionFind.md

```cpp
struct UnionFind {
    vector<int> parents; // parents[i] := 頂点iのroot
    vector<int> sizes; // sizes[i] := 頂点iが属するグループの要素数
    UnionFind(int n){
        parents = vector<int>(n);
        for(int i=0; i<n; i++) parents.at(i) = i;
        sizes = vector<int>(n, 1);
    }
    int find(int i){
        if(parents.at(i)==i) return i; // 自身のroot == iのとき自分がroot
        return (parents.at(i) = find(parents.at(i))); // 経路圧縮
    }
    void merge(int a, int b){
        a = find(a);
        b = find(b);
        if(a!=b){
            sizes.at(a) += sizes.at(b);
            parents.at(b) = a;
        }
    }
```

```
    bool connected(int a, int b){
        return (find(a)==find(b));
    }
    long long size(int i){
        return (long long)sizes.at(find(i));
    }
};
```

内容	コード	計算量
宣言	UnionFind uf(N)	$O(N)$
aの根の取得	uf.find(a)	$O(\alpha(N))$
aとbを統合	uf.merge(a, b)	$O(\alpha(N))$
aとbが同じグループか判定	uf.connected(a, b)	$O(\alpha(N))$
aと同じグループの要素数	uf.size(a)	$O(\alpha(N))$

$\alpha(N)$はアッカーマンの逆関数

math

BinaryGCD.hpp

md5: f3ab31

```
u64 ctz(u64 x) { return countr_zero(x); }
u64 binary_gcd(u64 x, u64 y) {
   if(!x || !y) return x | y;
   u64 n = ctz(x), m = ctz(y);
   x >>= n, y >>= m;
   while(x != y) {
      if(x > y) x = (x - y) >> ctz(x - y);
      else y = (y - x) >> ctz(y - x);
   }
   return x << min(n, m);
}
```

ExtGCD.hpp

md5: c3fa9b

```
// returns gcd(a, b) and assign x, y to integers
// s.t. ax + by = gcd(a, b) and |x| + |y| is minimized
ll extgcd(ll a, ll b, ll& x, ll& y) {
   // assert(a >= 0 && b >= 0);
   if(!b) return x = 1, y = 0, a;
   ll d = extgcd(b, a % b, y, x);
   y -= a / b * x;
   return d;
}
```

modint

BarrettReduction.hpp

md5: 2ca7f3

```
// 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;
   }
};
```

combination.md

```
const long long MAX = 5e5; // 関数comの引数の最大値
const long long mod = 1e9+7; // 何で割った余りか
vector<long long> fac(MAX), finv(MAX), inv(MAX);

void COMinit(){
    fac[0] = 1; fac[1] = 1;
    finv[0] = 1; finv[1] = 1;
    inv[1] = 1;
    for(int i=2; i<MAX; i++){
        fac[i] = fac[i-1] * i % mod;
        inv[i] = mod - inv[mod%i] * (mod / i) % mod;
        finv[i] = finv[i-1] * inv[i] % mod;
    }
}

long long com(int n, int k){
    if(n<k) return 0;
    if(n<0 || k<0) return 0;
    return fac[n] * (finv[k] * finv[n-k] % mod) % mod;
}
```

内容	コード	計算量
二項係数を求める前処理	COMinit()	$O(MAX)$
二項係数${}_nC_k \bmod p$の計算	com(n, k)	$O(1)$

com(n, k)を実行する前に必ずCOMinit()を実行する（一度だけで良い）。$1 \leq k \leq n \leq 10^7$
また、$\bmod p$のpは素数かつ$n < p$となる必要がある。

modint.hpp

md5: 81b530

```
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 行コピペ  Alt + Shift + クリックで複数カーソル
   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: 3138c7

```
// {998244353, 3}, {1811939329, 13}, {2013265921, 31}
mm g = 3;  // 原始根
void fft(vector<mm>& a) {
   ll n = sz(a), lg = __lg(n);
   assert((1 << lg) == n);
   vector<mm> b(n);
   rep(l, 1, lg + 1) {
      ll w = n >> l;
      mm s = 1, r = g.pow(mod >> l);
      for(ll u = 0; u < n / 2; u += w) {
         rep(d, 0, w) {
            mm x = a[u << 1 | d], y = a[u << 1 | w | d] * s;
            b[u | d] = x + y;
            b[n >> 1 | u | d] = x - y;
         }
         s *= r;
      }
      swap(a, b);
   }
}
vector<mm> conv(vector<mm> a, vector<mm> b) {
   if(a.empty() || b.empty()) return {};
   size_t s = sz(a) + sz(b) - 1, n = bit_ceil(s);
   // if(min(sz(a), sz(b)) <= 60) 愚直に掛け算
   a.resize(n);
   b.resize(n);
   fft(a);
   fft(b);
   mm inv = mm(n).inv();
   rep(i, 0, n) a[i] *= b[i] * inv;
   reverse(1 + all(a));
   fft(a);
   a.resize(s);
   return a;
}
```

FFT_fast.hpp

md5: c8c567

```
// modint を u32 にして加減算を真面目にやると速い
mm g = 3;  // 原始根
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[countr_zero<uint64_t>(~k)];
      }
   }
}
// コピペ
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 を逆数に
```

```
        z[i - 2] = s * g.inv().pow(mod >> i);
        s *= g.pow(mod >> i);
      }
      return z;
   }();
   for(ll l = lg; l--;) {  // 逆順に
      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 を下に移動
            a[i] = x + y;
            a[i + w] = (x - y) * s;
         }
         s *= z[countr_zero<uint64_t>(~k)];
      }
   }
}
vector<mm> conv(vector<mm> a, vector<mm> b) {
   if(a.empty() || b.empty()) return {};
   size_t s = sz(a) + sz(b) - 1, n = bit_ceil(s);
   // if(min(sz(a), sz(b)) <= 60) 愚直に掛け算
   a.resize(n);
   b.resize(n);
   fft(a);
   fft(b);
   mm inv = mm(n).inv();
   rep(i, 0, n) a[i] *= b[i] * inv;
   ifft(a);
   a.resize(s);
   return a;
}
```

graph

dijkstra.md

```
using ll = long long;
using pll = pair<ll, ll>;
const ll INF = 2e18;
vector<ll> dijkstra(long long start, vector<vector<pll>> &g){
    priority_queue<pll, vector<pll>, greater<pll>> pq;
    int n = g.size();
    vector<ll> dist(n, INF);
    dist[start] = 0; pq.push({0, start});
    while(!pq.empty()){
        auto [d, pos] = pq.top(); pq.pop();
        if(dist[pos] < d) continue;
        for(int i=0; i<g[pos].size(); i++){
            auto [cost, to] = g[pos][i];
            if(dist[to]>dist[pos]+cost){
                dist[to] = dist[pos] + cost;
                pq.push({dist[to], to});
            }
        }
    }
    return dist;
}
```

内容	コード	計算量
頂点数nのグラフの宣言	vector<vector<pll>> g; g.resize(n);	
頂点sからグラフgに対してダイクストラの実行	dijkstra(s, g)	$O(\|E\|log\|V\|)$
u→v(コスト:c)の辺を追加	g[u].push_back({c, v})	

(ただし$|E|$は辺の数, $|V|$は頂点の数.)

graph/tree

flow

燃やす埋める.md

変形前の制約	変形後の制約
x が 0 のとき z 失う	(x, T, z)
x が 0 のとき z 得る	無条件で z 得る; (S, x, z)
x が 1 のとき z 失う	(S, x, z)
x が 1 のとき z 得る	無条件で z 得る; (x, T, z)
$x, y, \ldots$ がすべて 0 のとき z 得る	無条件で z 得る; $(S, w, z), (w, x, \infty), (w, y, \infty)$
$x, y, \ldots$ がすべて 1 のとき z 得る	無条件で z 得る; $(w, T, z), (x, w, \infty), (y, w, \infty)$

string

KMP.hpp

md5: 886c63

```
// kmp[i] := max{ l ≤ i | s[:l] == s[(i+1)-l:i+1] }
// abacaba -> 0010123
auto KMP(string s) {
   vector<ll> p(sz(s));
   rep(i, 1, sz(s)) {
      ll g = p[i - 1];
      while(g && s[i] != s[g]) g = p[g - 1];
      p[i] = g + (s[i] == s[g]);
   }
   return p;
}
```

Manacher.md

```
vector<long long> manacher(const string &s){
    int i = 0, j = 0;
    vector<long long> r(s.size(), 0);
    while(i<s.size()){
        while(i-j>=0 && i+j<s.size() && s[i-j]==s[i+j]) j++;
        r[i] = j;
        int k = 1;
        while(i-k>=0 && k+r[i-k]<j) r[i+k] = r[i-k], k++;
        i += k; j -= k;
    }
    return r;
}
```

内容	コード	計算量
i文字目を中心とする最長の回文半径を記録した配列の取得	r = manacher(S)	$O(\|S\|)$

偶数長の回文を判定する場合はダミーを交互に挿入する。(例: abba → #a#b#b#a#)

RollingHash.md

```
struct RollingHash{

    using ll = long long;

    const ll mod = (1ll<<61)-1;
    ll n, rnd;
    vector<ll> hs, pw;
    RollingHash(string s): n(s.size()), hs(n+1), pw(n+1, 1) {
        // rnd = rand()%mod;
        rnd = 93842734347298748;
        for(int i=0; i<n; i++){
            pw[i+1] = mul(pw[i], rnd);
            hs[i+1] = add(mul(hs[i], rnd), s[i]);
        }
    };
    ll add(ll a, ll b){
        return (a+b)%mod;
    }
    ll mul(ll a, ll b){
        auto c = (__uint128_t)a*b;
        return add(c>>61, c&mod);
    }
    ll get(ll l, ll r){
        return add(hs[r], mod - mul(hs[l], pw[r-l]));
    }
};
```

内容	コード	計算量
宣言	RollingHash rh(s)	$O(s.size())$
$[l, r)$のハッシュ値の取得	rh.get(l, r)	$O(1)$

SuffixArray.hpp

md5: 1d70ce

```
// returns pair{sa, lcp}
// sa 長さ n : s[sa[0]:] < s[sa[1]:] < … < s[sa[n-1]:]
// lcp 長さ n-1 : lcp[i] = LCP(s[sa[i]:], s[sa[i+1]:])
auto SA(string s) {
   ll n = sz(s) + 1, lim = 256;
   // assert(lim > ranges::max(s));
   vector<ll> sa(n), lcp(n), x(all(s) + 1), y(n), ws(max(n, lim)), rk(n);
   iota(all(sa), 0);
   for(ll j = 0, p = 0; p < n; j = max(1LL, j * 2), lim = p) {
      p = j;
      iota(all(y), n - j);
      rep(i, 0, n) if(sa[i] >= j) y[p++] = sa[i] - j;
      fill(all(ws), 0);
      rep(i, 0, n) ws[x[i]]++;
      rep(i, 1, lim) ws[i] += ws[i - 1];
      for(ll i = n; i--;) sa[--ws[x[y[i]]]] = y[i];
      swap(x, y);
      p = 1;
      x[sa[0]] = 0;
      rep(i, 1, n) {
         ll a = sa[i - 1], b = sa[i];
         x[b] = (y[a] == y[b] && y[a + j] == y[b + j]) ? p - 1 : p++;
      }
   }
   rep(i, 1, n) rk[sa[i]] = i;
   for(ll i = 0, k = 0; i < n - 1; lcp[rk[i++]] = k) {
      if(k) k--;
      while(s[i + k] == s[sa[rk[i] - 1] + k]) k++;
   }
   sa.erase(begin(sa));
   lcp.erase(begin(lcp));
```

```
   return pair{sa, lcp};
}
```

Zalgorithm.md

```
vector<long long> z_algorithm(const string &s){
    vector<long long> z(s.size(), 0);
    z[0] = s.size();
    int i = 1, j = 0;
    while(i<s.size()){
        while(i+j<s.size() && s[j]==s[i+j]) j++;
        z[i] = j;
        if(j==0){ i++; continue; }
        int k = 1;
        while(i+k<s.size() && k+z[k]<j) z[i+k] = z[k], k++;
        i += k; j -= k;
    }
    return z;
}
```

内容	コード	計算量
z[i]=「S」と「Sのi文字目以降」の最長共通接頭辞の長さを計算	z = z_algorithm(S)	$O(\|S\|)$

algorithm

geometry

memo

Primes.md

素数の個数

n	10^2	10^3	10^4	10^5	10^6	10^7	10^8	10^9	10^{10}
$\pi(n)$	25	168	1229	9592	78498	664579	5.76e+6	5.08e+7	4.55e+8

高度合成数

$\leq n$	10^3	10^4	10^5	10^6	10^7	10^8	10^9
x	840	7560	83160	720720	8648640	73513440	735134400
$d^0(x)$	32	64	128	240	448	768	1344

$\leq n$	10^{10}	10^{11}	10^{12}	10^{13}	10^{14}	10^{15}	10^{16}	10^{17}	10^{18}
$d^0(x)$	2304	4032	6720	10752	17280	26880	41472	64512	103680

素数階乗

n	2	3	5	7	11	13	17	19	23	29
$n\#$	2	6	30	210	2310	30030	510510	9.70e+6	2.23e+8	6.47e+9

階乗

4!	5!	6!	7!	8!	9!	10!	11!	12!	13!
24	120	720	5040	40320	362880	3.63e+6	3.99e+7	4.79e+8	6.23e+9