

ICPC Notebook

template	
hash.sh	2
settings.sh	2
template.hpp	2
data-structure	
BIT.hpp	3
FastSet.hpp	3
LazySegmentTree.hpp	4
SegmentTree.hpp	5
SparseTable.hpp	6
UnionFind.hpp	7
waveletmatrix.hpp	7
math	
BinaryGCD.hpp	10
CHT.hpp	10
ChineseRem.hpp	11
Combination.hpp	11
Eratosthenes.hpp	11
ExtGCD.hpp	11
Matrix.hpp	12
floorsum.hpp	13
miller_rabin.hpp	13
subset_zeta.hpp	14
xor_set.hpp	14
modint	
modint.hpp	16
FPS	
FFT.hpp	17
graph	
2SAT.hpp	18
BellmanFord.hpp	18
SCC.hpp	18
dijkstra.cpp	19
graph/tree	
AuxiliaryTree.hpp	20
Cartesiantree.hpp	21
Rerooting.hpp	21
lca.hpp	22
flow	
dinic.hpp	24
mincostflow.hpp	24
string	
AhoCorasick.hpp	26
KMP.hpp	26
Manacher.hpp	26
RollingHash.hpp	27
SuffixArray.hpp	27
Trie.hpp	27
Zalgorithm.hpp	28
algorithm	
mo.hpp	29
geometry	
geometry.hpp	30
memo	
Primes.md	33
母関数.md	34
燃やす埋める.md	35

template

hash.sh

```
1 # 使い方: sh hash.sh -> コピペ -> Ctrl + D
2 # コメント・空白・改行を削除して md5 でハッシュする
3 g++ -dD -E -P -fpreprocessed - | tr -d '[:space:]' | md5sum | cut -c-6
4
```

settings.sh

```
1 # CLion の設定
2 Settings → Build → CMake → Reload CMake Project
3 add_compile_options(-D_GLIBCXX_DEBUG)
4 # Caps Lock を Ctrl に変更
5 setxkbmap -option ctrl:nocaps
6
```

template.hpp

md5: fc725b

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 using ll = long long;
4 const ll INF = LLONG_MAX / 4;
5 #define rep(i, a, n) for(ll i = a; i < n; i++)
6 #define rrep(i, a, n) for(ll i = a; i >= n; i--)
7 #define inr(l, x, r) (l <= x && x < r)
8 #define sz(a) ssize(a)
9 bool chmin(auto& a, auto b) { return a > b ? a = b, 1 : 0; }
10 bool chmax(auto& a, auto b) { return a < b ? a = b, 1 : 0; }
11
12 int main() {
13     cin.tie(0)->sync_with_stdio(0);
14     // your code here...
15 }
16
```

data-structure

BIT.hpp

md5: b7588b

```

1  struct BIT {
2      vector<ll> a;
3      BIT(ll n) : a(n + 1) {}
4      // A[i] += x
5      void add(ll i, ll x){
6          i++;
7          while(i < (int)a.size()){
8              a[i] += x;
9              i += i & -i;
10         }
11     }
12     // sum of A[0, r)
13     ll sum(ll r) {
14         ll s = 0;
15         while(r){
16             s += a[r];
17             r -= r & -r;
18         }
19         return s;
20     }
21     // sum of A[l, r)
22     ll sum(ll l, ll r){
23         return sum(r) - sum(l);
24     }
25 };
26

```

FastSet.hpp

md5: 2cb8c9

```

1  // using u64 = uint64_t;
2  const u64 B = 64;
3  struct FastSet {
4      u64 n;
5      vector<vector<u64>> a;
6      FastSet(u64 n_) : n(n_) {
7          do a.emplace_back(n_ = (n_ + B - 1) / B);
8          while(n_ > 1);
9      }
10     // bool operator[](ll i) const { return a[0][i / B] >> (i % B) & 1; }
11     void set(ll i) {
12         for(auto& v : a) {
13             v[i / B] |= 1ULL << (i % B);
14             i /= B;
15         }
16     }
17     void reset(ll i) {
18         for(auto& v : a) {
19             v[i / B] &= ~(1ULL << (i % B));
20             if(v[i / B]) break;
21             i /= B;
22         }
23     }
24     ll next(ll i) { // i を超える最小の要素
25         rep(h, 0, sz(a)) {
26             i++;
27             if(i / B >= sz(a[h])) break;
28             u64 d = a[h][i / B] >> (i % B);
29             if(d) {
30                 i += countr_zero(d);
31                 while(h--) i = i * B + countr_zero(a[h][i]);
32                 return i;
33             }
34             i /= B;
35         }
36         return n;
37     }
38     ll prev(ll i) { // i より小さい最大の要素
39         rep(h, 0, sz(a)) {
40             i--;
41             if(i < 0) break;
42             u64 d = a[h][i / B] << (~i % B);
43             if(d) {
44                 i -= countl_zero(d);
45                 while(h--) i = i * B + __lg(a[h][i]);
46                 return i;
47             }
48             i /= B;

```

```

49     }
50     return -1;
51 }
52 };
53

```

LazySegmentTree.hpp

md5: 247a93

```

1  template<class T,
2      T (*op)(T, T),
3      T (*e)(),
4      class F,
5      T (*mapping)(F, T),
6      F (*composition)(F, F),
7      F (*id)()>
8  struct LazySegmentTree {
9      LazySegmentTree(const int _n) : n(_n) {
10         while((1 << log) < n) log++;
11         len = 1 << log;
12         d.assign(len * 2, e());
13         lazy.assign(len, id());
14     }
15     void set(const int i, const T x) {
16         assert(0 <= i && i < n);
17         d[i + len] = x;
18     }
19     T get(int p) {
20         assert(0 <= p && p < n);
21         p += len;
22         for(int i = log; i >= 1; i--) push(p >> i);
23         return d[p];
24     }
25     void build(){
26         for(int i = len - 1; i >= 1; i--) update(i);
27     }
28     void update(int l, int r, const F x) {
29         assert(0 <= l && l <= r && r <= n);
30         l += len;
31         r += len;
32         const int l_ctz = __builtin_ctz(l);
33         const int r_ctz = __builtin_ctz(r);
34         for(int i = log; i > l_ctz; i--) push(l >> i);
35         for(int i = log; i > r_ctz; i--) push((r - 1) >> i);
36         const int lt = l, rt = r;
37         while(l < r) {
38             if(l & 1) apply(l++, x);
39             if(r & 1) apply(--r, x);
40             l >>= 1;
41             r >>= 1;
42         }
43         l = lt;
44         r = rt;
45         for(int i = l_ctz + 1; i <= log; i++) update(l >> i);
46         for(int i = r_ctz + 1; i <= log; i++) update((r - 1) >> i);
47     }
48     T query(int l, int r) {
49         assert(0 <= l && l <= r && r <= n);
50         l += len;
51         r += len;
52         const int l_ctz = __builtin_ctz(l);
53         const int r_ctz = __builtin_ctz(r);
54         for(int i = log; i > l_ctz; i--) push(l >> i);
55         for(int i = log; i > r_ctz; i--) push((r - 1) >> i);
56         T left = e(), right = e();
57         while(l < r) {
58             if(l & 1) left = op(left, d[l++]);
59             if(r & 1) right = op(d[--r], right);
60             l >>= 1;
61             r >>= 1;
62         }
63         return op(left, right);
64     }
65     template<class G> int max_right(int l, G g) {
66         assert(0 <= l && l <= n);
67         assert(g(e()));
68         if(l == n) return n;
69         l += len;
70         for(int i = log; i >= 1; i--) push(l >> i);
71         T sm = e();
72         do {
73             l /= l & -l;
74             if(!g(op(sm, d[l]))) {
75                 while(l < len) {

```

```

76         push(l);
77         l <= 1;
78         if(g(op(sm, d[l]))) {
79             sm = op(sm, d[l]);
80             l++;
81         }
82     }
83     return l - len;
84 }
85 sm = op(sm, d[l]);
86 l++;
87 } while(l & (l - 1));
88 return n;
89 }
90 template<class G> int min_left(int r, G g) {
91     assert(0 <= r && r <= n);
92     assert(g(e()));
93     if(r == 0) return 0;
94     r += len;
95     for(int i = log; i >= 1; i--) push((r - 1) >> i);
96     T sm = e();
97     do {
98         r /= r & -r;
99         if(r > 1) r--;
100        if(!g(op(d[r], sm))) {
101            while(r < len) {
102                push(r);
103                r = r * 2 + 1;
104                if(g(op(d[r], sm))) {
105                    sm = op(d[r], sm);
106                    r--;
107                }
108            }
109            return r + 1 - len;
110        }
111        sm = op(d[r], sm);
112    } while(r & (r - 1));
113    return 0;
114 }
115
116 private:
117 vector<T> d;
118 vector<F> lazy;
119 int n = 1, log = 0, len = 0;
120 inline void update(const int k) { d[k] = op(d[2 * k], d[2 * k + 1]); }
121 inline void apply(const int k, const F& x) {
122     d[k] = mapping(x, d[k]);
123     if(k < len) lazy[k] = composition(lazy[k], x);
124 }
125 inline void push(const int k) {
126     apply(2 * k, lazy[k]);
127     apply(2 * k + 1, lazy[k]);
128     lazy[k] = id();
129 }
130 };
131
132 //区間加算・区間和取得
133 struct S{
134     ll value;
135     ll size;
136 };
137 using F = ll;
138
139 S op(S a, S b){ return {a.value+b.value, a.size+b.size}; }
140 S e(){ return {0, 0}; }
141 S mapping(F f, S x){ return {x.value + f*x.size, x.size}; }
142 F composition(F f, F g){ return f+g; }
143 F id(){ return 0; }
144

```

SegmentTree.hpp

md5: 10f106

```

1  template<class T>
2  struct SegmentTree {
3      static constexpr T unit = INT_MAX;
4      T op(T a, T b){ return min(a, b); }
5      vector<T> s;
6      int _n, n;
7      SegmentTree(int n_ = 0, T def = unit): _n(n_) {
8          int log = 1;
9          while((1 << log) < n_) log++;
10         n = 1<<log;

```

```

11     s = vector<T>(n*2, def);
12 }
13 // s[i] = x;
14 void update(int i, T x) {
15     i += n;
16     s[i] = x;
17     while(i >= 1){
18         s[i] = op(s[2 * i], s[2 * i + 1]);
19     }
20 }
21 // s[i] = f(s[i], x);
22 void apply(int i, T x){
23     i += n;
24     s[i] = op(s[i], x);
25     while(i >= 1){
26         s[i] = op(s[2 * i], s[2 * i + 1]);
27     }
28 }
29 // 区間取得: [b, e)
30 T query(int b, int e){
31     T ra = unit, rb = unit;
32     for(b += n, e += n; b < e; b /= 2, e /= 2){
33         if (b % 2) ra = op(ra, s[b++]);
34         if (e % 2) rb = op(s[--e], rb);
35     }
36     return op(ra, rb);
37 }
38 // セグ木上の二分探索 O(log{n}) (optional)
39 // ex int L = lst.max_right(0, [&](int tmp){return tmp < l[i];});
40 template<class F> int max_right(int l, F f){
41     if(l == _n) return _n;
42     l += n;
43     T sm = unit;
44     do{
45         while(l % 2 == 0) l >>= 1;
46         if(!f(op(sm, s[l]))) {
47             while(l < n){
48                 l = (2 * l);
49                 if(f(op(sm, s[l]))) {
50                     sm = op(sm, s[l]);
51                     l++;
52                 }
53             }
54             return l - n;
55         }
56         sm = op(sm, s[l]);
57         l++;
58     }while((l & -l) != l);
59     return _n;
60 }
61 template<class F> int min_left(int r, F f){
62     if(r == 0) return 0;
63     r += n;
64     T sm = unit;
65     do {
66         r--;
67         while(r > 1 && (r % 2)) r >>= 1;
68         if(!f(op(s[r], sm))) {
69             while(r < n){
70                 r = (2 * r + 1);
71                 if(f(op(s[r], sm))) {
72                     sm = op(s[r], sm);
73                     r--;
74                 }
75             }
76             return r + 1 - n;
77         }
78         sm = op(s[r], sm);
79     }while((r & -r) != r);
80     return 0;
81 }
82 };
83

```

SparseTable.hpp

md5: acd1f4

```

1  template<typename T> struct SparseTable {
2      vector<vector<T> > st;
3      vector<int> lookup;
4
5      SparseTable(const vector<T>& v) {
6          int b = 0;

```

```
7         while((1 << b) <= v.size()) ++b;
8         st.assign(b, vector<T>(1 << b));
9         for(int i = 0; i < v.size(); i++) { st[0][i] = v[i]; }
10        for(int i = 1; i < b; i++) {
11            for(int j = 0; j + (1 << i) <= (1 << b); j++) { st[i][j] = min(st[i - 1][j], st[i - 1][j + (1 << (i - 1))]); }
12        }
13        lookup.resize(v.size() + 1);
14        for(int i = 2; i < lookup.size(); i++) { lookup[i] = lookup[i >> 1] + 1; }
15    }
16
17    inline T rmq(int l, int r) {
18        int b = lookup[r - l];
19        return min(st[b][l], st[b][r - (1 << b)]);
20    }
21 };
22
```

UnionFind.hpp

md5: 631ec9

```
1 struct UnionFind {
2     vector<int> e;
3     UnionFind(int n) : e(n, -1) {}
4     bool same(int a, int b) { return find(a) == find(b); }
5     int size(int x) { return -e[find(x)]; }
6     int find(int x) { return e[x] < 0 ? x : e[x] = find(e[x]); }
7     bool join(int a, int b) {
8         a = find(a), b = find(b);
9         if(a == b) return false;
10        if(e[a] > e[b]) swap(a, b);
11        e[a] += e[b];
12        e[b] = a;
13        return true;
14    }
15};
```

waveletmatrix.hpp

md5: 3b1bf0

```
1 // i桁目のビットが1かどうか
2 bool has_bit(ll x, int i) { return (x >> i) & 1; }
3
4 // 長さnの静的なビット列に対して累積和ができるデータ構造
5 class BitCumulativeSum {
6     // 64桁ごとに累積和を作る
7     inline static constexpr int w = 64;
8
9     vector<uint64_t> block; // ビット列をwごとに保持
10    vector<int> sum; // 累積和
11
12    public:
13    BitCumulativeSum() = default;
14    BitCumulativeSum(int n) : block(n / w + 1, 0), sum(1, 0) {
15        sum.reserve(block.size() + 1); // 事前に要素数分のメモリを確保しておく（このサイズになっているわけではない）
16    }
17
18    // i桁目のビットを立てる
19    void set(int i) { block[i / w] |= 1LL << (i % w); }
20
21    // 累積和を作成
22    void build() {
23        for(const auto& b : block) {
24            // popcount : 2進数表記で1の数を数える
25            sum.push_back(sum.back() + popcount(b));
26        }
27    }
28
29    // [0, r) 桁までの1の個数
30    int rank1(int r) const { return sum[r / w] + popcount(block[r / w] & ((1LL << (r % w)) - 1)); }
31
32    // [0, r) 桁まででの0の個数
33    int rank0(int r) const { return r - rank1(r); }
34};
35
36// 本題
37class WaveletMatrix {
38    int n, sigma;
39    vector<BitCumulativeSum> bv;
40
41    public:
42    WaveletMatrix(vector<int> v) : n((int)v.size()) {
43        // sigmaを決定する
44        int mx = 0;
```

```

45     for(auto& x : v) {
46         assert(x >= 0);
47         mx = max(mx, x);
48     }
49     sigma = 0;
50     while((1LL << sigma) - 1 < mx) sigma++;
51
52     // 行列の構築
53     bv.assign(sigma, n);
54     vector<int> nxt_v(n);
55     // 上位の桁から構築していく
56     for(int h = sigma - 1; h >= 0; h--) {
57         auto& B = bv[h]; // h桁目に対応するビットの累積和（ただし0の個数を数える）
58
59         // vでh桁目が0の要素を左に、1の要素を右に寄せる
60         int l = 0, r = n - 1;
61         // 1の方を寄せる
62         for(int i = n - 1; i >= 0; i--) {
63             if(has_bit(v[i], h)) {
64                 B.set(i);
65                 nxt_v[r--] = v[i];
66             }
67         }
68
69         B.build(); // 累積和を構築
70
71         // 0の方も寄せる
72         for(int i = 0; i < n; i++) {
73             if(!has_bit(v[i], h)) { nxt_v[l++] = v[i]; }
74         }
75         swap(v, nxt_v);
76     }
77 }
78
79 // [l, r) でk番目に小さい数 (0-indexed)
80 int kth_smallest(int l, int r, int k) const {
81     assert(0 <= k && k < n);
82     uint32_t res = 0;
83     // 上位の桁から0か1を決定していく
84     for(int h = sigma - 1; h >= 0; h--) {
85         const auto& B = bv[h];
86         int zero_cnt = B.rank0(r) - B.rank0(l); // 区間のビット0の個数
87         if(k >= zero_cnt) {
88             // h桁目が1の場合
89             res |= 1 << h;
90             k -= zero_cnt;
91             // 区間の更新
92             // h桁目が0の要素が左、1の要素が右によっていることを利用する
93             l = B.rank0(n) + B.rank1(l);
94             r = B.rank0(n) + B.rank1(r);
95         } else {
96             // h桁目が0の場合
97             l = B.rank0(l);
98             r = B.rank0(r);
99         }
100     }
101     return res;
102 }
103
104 // [l, r) でk番目に小さい数 (0-indexed)
105 int kth_largest(int l, int r, int k) const {
106     assert(0 <= r - l - k + 1 && r - l - k + 1 < n);
107     return kth_smallest(l, r, r - l - k + 1);
108 }
109
110 // [0, r) でu未満の値の個数
111 int range_freq(int r, int u) {
112     assert(u >= 0);
113     if(u >= (1LL << sigma)) return r;
114
115     int l = 0, ret = 0;
116     for(int h = sigma - 1; h >= 0; --h) {
117         auto& B = bv[h];
118         if(has_bit(u, h)) {
119             // h桁目が1の場合
120             ret += B.rank0(r) - B.rank0(l); // 区間に属しているh桁目が0の要素はu未満
121             l = B.rank0(n) + B.rank1(l);
122             r = B.rank0(n) + B.rank1(r);
123         } else {
124             // h桁目が0の場合

```



```
125         l = B.rank0(l);
126         r = B.rank0(r);
127     }
128 }
129
130     return ret;
131 }
132
133 // [l, r) でu未満の値の個数
134 int range_freq(int l, int r, int u) {
135     assert(u >= 0);
136     return range_freq(r, u) - range_freq(l, u);
137 }
138
139 // [l, r) でd以上u未満の値の個数
140 int range_freq(int l, int r, int d, int u) {
141     assert(d >= 0 && u >= 0);
142     return range_freq(l, r, u) - range_freq(l, r, d);
143 }
144 };
```

math

BinaryGCD.hpp

md5: f3ab31

```

1  u64 ctz(u64 x) { return countr_zero(x); }
2  u64 binary_gcd(u64 x, u64 y) {
3      if(!x || !y) return x | y;
4      u64 n = ctz(x), m = ctz(y);
5      x >>= n, y >>= m;
6      while(x != y) {
7          if(x > y) x = (x - y) >> ctz(x - y);
8          else y = (y - x) >> ctz(y - x);
9      }
10     return x << min(n, m);
11 }
12

```

CHT.hpp

md5: b42f8f

```

1  template<typename T> class CHT {
2  private:
3      struct node {
4          node *left, *right;
5          static const T inf = numeric_limits<T>::max();
6          T a, b;
7          node() : node(0, inf) {}
8          node(const T _a, const T _b) : left(nullptr), right(nullptr), a(_a), b(_b) {}
9          T f(const T x) const { return a * x + b; }
10 };
11 static void swap(node* x, node* y) { std::swap(x->a, y->a), std::swap(x->b, y->b); }
12 void _add_line(node* cur, node* nw, T l, T r) {
13     while(true) {
14         if(nw->f(l) < cur->f(l)) swap(cur, nw);
15         if(cur->f(r - 1) <= nw->f(r - 1)) break;
16         const T mid = (l + r) / 2;
17         if(cur->f(mid) <= nw->f(mid)) {
18             if(!cur->right) {
19                 cur->right = new node(*nw);
20                 break;
21             } else {
22                 cur = cur->right, l = mid;
23             }
24         } else {
25             swap(cur, nw);
26             if(!cur->left) {
27                 cur->left = new node(*nw);
28                 break;
29             } else {
30                 cur = cur->left, r = mid;
31             }
32         }
33     }
34 }
35 T query(node* cur, const T k, T l, T r) const {
36     T ans = numeric_limits<T>::max();
37     while(cur) {
38         ans = min(ans, cur->f(k));
39         const T mid = (l + r) / 2;
40         if(k < mid) {
41             cur = cur->left, r = mid;
42         } else {
43             cur = cur->right, l = mid;
44         }
45     }
46     return ans;
47 }
48 void clear(node* cur) {
49     if(cur->left) clear(cur->left);
50     if(cur->right) clear(cur->right);
51     delete cur;
52 }
53 const T lpos, rpos;
54 node* root;
55
56 public:
57 CHT(const T _lpos, const T _rpos) : lpos(_lpos), rpos(_rpos), root(new node()) { assert(lpos < rpos); }
58 // ~CHT(){ clear(root); }
59 // f(x) = a * x + b を挿入
60 void add_line(const T a, const T b) {
61     node nw(a, b);
62     return _add_line(root, &nw, lpos, rpos);
63 }

```

```
63     }
64     // x = k での最小値
65     T query(const T k) const { return query(root, k, lpos, rpos); }
66 };
```

ChineseRem.hpp

md5: f60d0f

```
1  inline ll mod(ll a, ll m) { return (a % m + m) % m; }
2
3  inline ll mul(ll a, ll b, ll m) {
4      a = mod(a, m);
5      b = mod(b, m);
6      if(b == 0) return 0;
7      ll res = mul(mod(a + a, m), b >> 1, m);
8      if(b & 1) res = mod(res + a, m);
9      return res;
10 }
11
12 // returns gcd(a, b) and assign x, y to integers
13 // s.t. ax + by = gcd(a, b) and |x| + |y| is minimized
14 ll extgcd(ll a, ll b, ll& x, ll& y) {
15     // assert(a >= 0 && b >= 0);
16     if(!b) return x = 1, y = 0, a;
17     ll d = extgcd(b, a % b, y, x);
18     y -= a / b * x;
19     return d;
20 }
21
22 // 中国剰余定理
23 // リターン値を (r, m) とすると解は x = r (mod. m)
24 // 解なしの場合は (0, -1) をリターン
25 pair<ll, ll> chineseRem(const vector<ll>& b, const vector<ll>& m) {
26     ll r = 0, M = 1;
27     rep(i, 0, (int)b.size()) {
28         ll p, q;
29         ll d = extGCD(M, m[i], p, q); // p is inv of m1/d (mod. m[i]/d)
30         if((b[i] - r) % d != 0) return {0, -1};
31         ll tmp = mul(((b[i] - r) / d), p, (m[i] / d));
32         r += M * tmp;
33         M *= m[i] / d;
34     }
35     return {mod(r, M), M};
36 }
```

Combination.hpp

md5: a88ecc

```
1  int maxnum = 200005;
2  vector<ll> fac(maxnum), inv(maxnum), finv(maxnum);
3  void init_fac() {
4      fac[0] = fac[1] = 1;
5      inv[1] = 1;
6      finv[0] = finv[1] = 1;
7      rep(i, 2, maxnum) {
8          fac[i] = fac[i - 1] * i % MOD;
9          inv[i] = MOD - MOD / i * inv[MOD % i] % MOD;
10         finv[i] = finv[i - 1] * inv[i] % MOD;
11     }
12 }
13 ll nCr(ll n, ll r) {
14     if(n < 0 or n - r < 0 or r < 0) return 0;
15     return fac[n] * (finv[n - r] * finv[r] % MOD) % MOD;
16 }
```

Eratosthenes.hpp

md5: 91b6ba

```
1  int max_num = 1000005;
2  vector<int> erat(max_num);
3  void init_e() {
4      for(ll i = 2; i*i <= max_num; i++) {
5          if(erat[i] == 0) {
6              for(ll j = i * i; j <= max_num - 1; j += i) {
7                  if(erat[j] == 0) erat[j] = i;
8              }
9          }
10     }
11 }
```

ExtGCD.hpp

md5: c3fa9b

```
1  // returns gcd(a, b) and assign x, y to integers
2  // s.t. ax + by = gcd(a, b) and |x| + |y| is minimized
```

```

3  ll extgcd(ll a, ll b, ll& x, ll& y) {
4      // assert(a >= 0 && b >= 0);
5      if(!b) return x = 1, y = 0, a;
6      ll d = extgcd(b, a % b, y, x);
7      y -= a / b * x;
8      return d;
9  }
10

```

Matrix.hpp

md5: 25fbc4

```

1  template<class T> struct Matrix {
2      vector<vector<T>>> A;
3
4      Matrix() {}
5
6      Matrix(size_t n, size_t m) : A(n, vector<T>(m, 0)) {}
7
8      Matrix(size_t n) : A(n, vector<T>(n, 0)){};
9
10     size_t height() const { return A.size(); }
11
12     size_t width() const { return A[0].size(); }
13
14     inline const vector<T>& operator[](ll k) const { return A.at(k); }
15
16     inline vector<T>& operator[](ll k) { return A.at(k); }
17
18     static Matrix I(size_t n) {
19         Matrix mat(n);
20         for(ll i = 0; i < n; i++) mat[i][i] = 1;
21         return (mat);
22     }
23
24     Matrix& operator+=(const Matrix& B) {
25         size_t n = height(), m = width();
26         assert(n == B.height() && m == B.width());
27         for(ll i = 0; i < n; i++)
28             for(ll j = 0; j < m; j++) (*this)[i][j] += B[i][j];
29         return (*this);
30     }
31
32     Matrix& operator-=(const Matrix& B) {
33         size_t n = height(), m = width();
34         assert(n == B.height() && m == B.width());
35         for(ll i = 0; i < n; i++)
36             for(ll j = 0; j < m; j++) (*this)[i][j] -= B[i][j];
37         return (*this);
38     }
39
40     Matrix& operator*=(const Matrix& B) {
41         size_t n = height(), m = B.width(), p = width();
42         assert(p == B.height());
43         vector<vector<T>>> C(n, vector<T>(m, 0));
44         for(ll i = 0; i < n; i++)
45             for(ll j = 0; j < m; j++)
46                 for(ll k = 0; k < p; k++) C[i][j] = C[i][j] + (*this)[i][k] * B[k][j];
47         A.swap(C);
48         return (*this);
49     }
50
51     Matrix& operator^=(long long k) {
52         Matrix B = Matrix::I(height());
53         while(k > 0) {
54             if(k & 1) B *= *this;
55             *this *= *this;
56             k >>= 1LL;
57         }
58         A.swap(B.A);
59         return (*this);
60     }
61
62     Matrix operator+(const Matrix& B) const { return (Matrix(*this) += B); }
63
64     Matrix operator-(const Matrix& B) const { return (Matrix(*this) -= B); }
65
66     Matrix operator*(const Matrix& B) const { return (Matrix(*this) *= B); }
67
68     Matrix operator^(const long long k) const { return (Matrix(*this) ^= k); }
69
70     // 行列式
71     T determinant() {

```

```
72     Matrix B(*this);
73     assert(width() == height());
74     T ret = 1;
75     for(ll i = 0; i < width(); i++) {
76         ll idx = -1;
77         for(ll j = i; j < width(); j++) {
78             if(B[j][i] != 0) idx = j;
79         }
80         if(idx == -1) return (0);
81         if(i != idx) {
82             ret *= -1;
83             swap(B[i], B[idx]);
84         }
85         ret *= B[i][i];
86         T vv = B[i][i];
87         for(ll j = 0; j < width(); j++) { B[i][j] /= vv; }
88         for(ll j = i + 1; j < width(); j++) {
89             T a = B[j][i];
90             for(ll k = 0; k < width(); k++) { B[j][k] -= B[i][k] * a; }
91         }
92     }
93     return (ret);
94 }
95 ll rank() {
96     vector<vector<T>> B = A;
97     ll n = B.size();
98     ll m = B[0].size();
99     rep(i, 0, n) {
100         ll id = i;
101         rep(j, i + 1, n) if(B[id] < B[j]) id = j;
102         swap(B[i], B[id]);
103         ll r = -1;
104         rep(j, i, m) if(B[i][j]) {
105             r = j;
106             break;
107         }
108         if(r == -1) return i;
109         rep(j, 0, n) {
110             if(i == j) continue;
111             if(B[j][r]) { rep(k, i, m) B[j][k] ^= B[i][k]; }
112         }
113     }
114     return n;
115 }
116 };
```

floorsum.hpp

md5: 76b016

```
1 long long floor_sum(long long n, long long m, long long a, long long b) {
2     long long ans = 0;
3     if(a >= m) {
4         ans += (n - 1) * n * (a / m) / 2;
5         a %= m;
6     }
7     if(b >= m) {
8         ans += n * (b / m);
9         b %= m;
10    }
11
12    long long y_max = (a * n + b) / m, x_max = (y_max * m - b);
13    if(y_max == 0) return ans;
14    ans += (n - (x_max + a - 1) / a) * y_max;
15    ans += floor_sum(y_max, a, m, (a - x_max % a) % a);
16    return ans;
17 }
18
```

miller_rabin.hpp

md5: b6e3d4

```
1 ll mod_pow(ll a, ll b, ll m) {
2     ll res = 1;
3     a %= m;
4     while(b > 0) {
5         if(b & 1) res = __uint128_t(res) * a % m;
6         a = __uint128_t(a) * a % m;
7         b >>= 1;
8     }
9     return res;
10 }
11
12 // num が素数なら true, そうでなければ false (計算量 O(log{num}^3))
13 bool miller_rabin(ll num) {
```

```
14     const vector<ll> A = {2, 325, 9375, 28178, 450775, 9780504, 1795265022};
15
16     // これらは例外的に判定が必要（なぜかは分らん）
17     if(num == 2 || num == 3 || num == 5 || num == 13 || num == 19 || num == 73 || num == 193 || num == 407521
18         || num == 299210837)
19         return true;
20     // 1 か 2 以外の偶数は素数でない
21     if(num == 1 || !(num & 1)) return false;
22
23     // num-1 = 2^s d (d は奇数) を満たす s, d を求める
24     ll s = 0, d = num - 1;
25     while(!(d & 1)) { s = s + 1, d >>= 1; }
26
27     // 各 a について, 条件をチェックする
28     for(auto a : A) {
29         ll powa = mod_pow(a, d, num);
30         if(powa == 1 || powa == num - 1) continue;
31
32         bool may_prime = false;
33         for(int i = 0; i < s - 1; i++) {
34             powa = __uint128_t(powa) * powa % num;
35             if(powa == 1) return false;
36             if(powa == num - 1) {
37                 may_prime = true;
38                 break;
39             }
40         }
41         if(!may_prime) return false;
42     }
43
44     return true;
45 }
```

subset_zeta.hpp

md5: 748007

```
1  template<class T> vector<T> subset_zeta(vector<T> f, int n, bool inv = false) {
2      for(int i = 0; i < n; i++) {
3          for(int S = 0; S < (1 << n); S++) {
4              if((S & (1 << i)) != 0) { // if i in S
5                  if(!inv) {
6                      f[S] += f[S ^ (1 << i)];
7                  } else {
8                      f[S] -= f[S ^ (1 << i)];
9                  }
10             }
11         }
12     }
13     return f;
14 }
15
16 template<class T> vector<T> supset_zeta(vector<T> f, int n, bool inv = false) {
17     for(int i = 0; i < n; i++) {
18         for(int S = 0; S < (1 << n); S++) {
19             if((S & (1 << i)) == 0) { // if i not in S
20                 if(!inv) {
21                     f[S] += f[S ^ (1 << i)];
22                 } else {
23                     f[S] -= f[S ^ (1 << i)];
24                 }
25             }
26         }
27     }
28     return f;
29 }
```

xor_set.hpp

md5: 70baccd

```
1  class xor_set {
2      private:
3          vector<ll> w;
4
5      public:
6          xor_set() {}
7          void insert(ll x) {
8              for(ll v : w)
9                  if(v & -v & x) x ^= v;
10             if(x == 0) return;
11             for(ll& v : w)
12                 if(x & -x & v) v ^= x;
13             w.push_back(x);
14         }
15         // 独立か判定
```

```
16     ll count(ll x) {
17         for(ll v : w)
18             if(v & -v & x) x ^= v;
19         if(x == 0) return 1;
20         else return 0;
21     }
22     vector<ll> get() { return w; }
23 };
```

modint

modint.hpp

md5: c3f394

```
1  const uint32_t mod = 1000000007;
2  struct mm {
3      uint32_t x;
4      mm() : x(0) {}
5      template<class T> mm(T x_) : x(x_ % mod) {
6          if(x >= mod) x += mod;
7      }
8      friend mm operator+(mm a, mm b) {
9          a.x += b.x;
10         if(a.x >= mod) a.x -= mod;
11         return a;
12     }
13     friend mm operator-(mm a, mm b) {
14         a.x -= b.x;
15         if(a.x >= mod) a.x += mod;
16         return a;
17     }
18     friend mm operator*(mm a, mm b) { return (uint64_t)a.x * b.x; }
19     friend mm operator/(mm a, mm b) { return a * b.inv(); }
20     friend mm& operator+=(mm& a, mm b) { return a = a + b; }
21     friend mm& operator-=(mm& a, mm b) { return a = a - b; }
22     friend mm& operator*=(mm& a, mm b) { return a = a * b; }
23     friend mm& operator/=(mm& a, mm b) { return a = a * b.inv(); }
24     mm inv() const { return pow(mod - 2); }
25     mm pow(ll b) const {
26         mm a = *this, c = 1;
27         while(b) {
28             if(b & 1) c *= a;
29             a *= a;
30             b >>= 1;
31         }
32         return c;
33     }
34 };
35
```


FPS

FFT.hpp

md5: 15dbfb

```
1 // need modint
2 // {998244353, 3}, {1000000007, 5}, {1811939329, 13}, {2013265921, 31}
3 mm g = 3; // 原始根
4 void fft(vector<mm>& a) {
5     ll n = a.size(), lg = __lg(n);
6     assert((1 << lg) == n);
7     vector<mm> b(n);
8     rep(l, 1, lg+1){
9         ll w = n >> l;
10        mm s = 1, r = g.pow(mod >> l);
11        for(ll u = 0; u < n / 2; u += w) {
12            rep(d, 0, w) {
13                mm x = a[u << 1 | d], y = a[u << 1 | w | d] * s;
14                b[u | d] = x + y;
15                b[n >> 1 | u | d] = x - y;
16            }
17            s *= r;
18        }
19        swap(a, b);
20    }
21 }
22 vector<mm> conv(vector<mm> a, vector<mm> b) {
23     if(a.empty() || b.empty()) return {};
24     size_t s = a.size()+b.size()-1, n = bit_ceil(s);
25     a.resize(n), b.resize(n);
26     fft(a), fft(b);
27     mm inv = mm(n).inv();
28     rep(i, 0, n) a[i] *= b[i] * inv;
29     reverse(a.begin()+1, a.end());
30     fft(a);
31     a.resize(s);
32     return a;
33 }
```

graph

2SAT.hpp

md5: 7fb307

```
1 // need SCC
2 struct TWO_SAT {
3     int _n;
4     vector<bool> answer;
5     SCC scc;
6     TWO_SAT(int n=0) : _n(n), answer(n), scc(2 * n) {}
7     // if(f is true) x_i; else \bar{x}_i
8     // if(g is true) x_j; else \bar{x}_j
9     void add_clause(int i, bool f, int j, bool g) {
10         scc.add_edge(2 * i + (f ? 0 : 1), 2 * j + (g ? 1 : 0));
11         scc.add_edge(2 * j + (g ? 0 : 1), 2 * i + (f ? 1 : 0));
12     }
13     bool satisfiable() {
14         auto id = scc.scc_ids().second;
15         for (int i = 0; i < _n; i++) {
16             if (id[2 * i] == id[2 * i + 1]) return false;
17             answer[i] = id[2 * i] < id[2 * i + 1];
18         }
19         return true;
20     }
21 };
```

BellmanFord.hpp

md5: 1090e0

```
1 struct Edge{
2     int from, to;
3     ll cost;
4 };
5 vector<ll> bellman_ford(vector<Edge> &edges, int n, int start){
6     vector<ll> dist(n, INF);
7     dist[start] = 0;
8     rep(i, 0, n-1){
9         for (auto edge : edges){
10             ll d = (dist[edge.from] + edge.cost);
11             if (dist[edge.from] != INF && d < dist[edge.to]){
12                 dist[edge.to] = d;
13             }
14         }
15     }
16     rep(i, 0, n){
17         for (auto edge : edges){
18             ll d = (dist[edge.from] + edge.cost);
19             if(d < dist[edge.to] && dist[edge.from] != INF){
20                 dist[edge.to] = -INF; // 更新されたら無限に小さくなる
21             }
22         }
23     }
24     return dist;
25 }
26 }
27 }
```

SCC.hpp

md5: c364f9

```
1 struct SCC{
2     int _n;
3     vector<vector<int>> g;
4     SCC(int n) : _n(n), g(n) {}
5     // add edge
6     void add_edge(int from, int to){ g[from].push_back(to); }
7     // @return pair of (number of scc components, scc id)
8     pair<int, vector<int>> scc_ids() {
9         int now_ord = 0, group_num = 0;
10        vector<int> visited, low(_n), ord(_n, -1), ids(_n);
11        visited.reserve(_n);
12        auto dfs = [&](auto self, int v) -> void {
13            low[v] = ord[v] = now_ord++;
14            visited.push_back(v);
15            for(auto to: g[v]){
16                if(ord[to] == -1){
17                    self(self, to);
18                    low[v] = min(low[v], low[to]);
19                }else{
20                    low[v] = min(low[v], ord[to]);
21                }
22            }
23            if(low[v] == ord[v]){
```

```

24         while(true){
25             int u = visited.back();
26             visited.pop_back();
27             ord[u] = _n;
28             ids[u] = group_num;
29             if(u == v) break;
30         }
31         group_num++;
32     }
33 };
34 rep(i, 0, _n) if(ord[i] == -1) dfs(dfs, i);
35 for(auto& x: ids) x = group_num-1-x;
36 return {group_num, ids};
37 }
38 // get scc (topological sorted)
39 vector<vector<int>> scc(){
40     auto ids = scc_ids();
41     int group_num = ids.first;
42     vector<int> counts(group_num);
43     for(auto x : ids.second) counts[x]++;
44     vector<vector<int>> groups(ids.first);
45     rep(i, 0, group_num) groups[i].reserve(counts[i]);
46     rep(i, 0, _n) groups[ids.second[i]].push_back(i);
47     return groups;
48 }
49 };

```

dijkstra.cpp

md5: 2345e4

```

1  vector<ll> dijkstra(int s, vector<vector<pair<int, ll>>> &g){
2      int n = (int)g.size();
3      priority_queue<pair<ll, int>, vector<pair<ll, int>>, greater<pair<ll, int>>> que;
4      vector<ll> dist(n, INF);
5      que.push(make_pair(0, s));
6      dist[s] = 0;
7      while(!que.empty()){
8          auto [d, u] = que.top(); que.pop();
9          if(dist[u] < d) continue;
10         for(auto [v, c]: g[u]){
11             if(dist[v] > d+c){
12                 dist[v] = d+c;
13                 que.push({dist[v], v});
14             }
15         }
16     }
17     return dist;
18 }
19

```

graph/tree

AuxiliaryTree.hpp

md5: 1e46db

```

1  vector<int> fs, ls, depth, lg, stk;
2
3  vector<vector<int>> st;
4  int cur;
5  vector<vector<int>> graph;
6
7  void ett_dfs(int v, int p, int d) {
8      st[0][fs[v] = cur++] = v;
9      depth[v] = d;
10     for(int w : graph[v]) {
11         if(w == p) continue;
12         ett_dfs(w, v, d + 1);
13         st[0][cur++] = v;
14     }
15     ls[v] = cur - 1;
16 }
17
18 void AuxiliaryTree(vector<vector<int>> gh) {
19     graph = gh;
20     int n = graph.size();
21     fs.resize(n);
22     ls.resize(n);
23     depth.resize(n);
24     lg.resize(3 * n);
25     stk.resize(2 * n);
26     st.resize(20);
27     for(int i = 0; i < 20; i++) st[i].resize(3 * n);
28     cur = 0;
29     ett_dfs(0, -1, 0);
30     lg[0] = lg[1] = 0;
31     for(int i = 2; i <= cur; ++i) lg[i] = lg[i >> 1] + 1;
32
33     for(int i = 0, b = 1; i < lg[cur]; ++i, b <= 1) {
34         for(int j = 0; j < (cur - (b << 1) + 1); ++j) {
35             st[i + 1][j] = (depth[st[i][j]] <= depth[st[i][j + b]]) ? st[i][j] : st[i][j + b];
36         }
37     }
38 }
39
40 bool cmp_at(int x, int y) { return fs[x] < fs[y]; }
41
42 inline int lca(int u, int v) {
43     int x = fs[u], y = fs[v];
44     if(x > y) swap(x, y);
45     int l = lg[y - x + 1];
46     return (depth[st[l][x]] <= depth[st[l][y - (1 << l) + 1]]) ? st[l][x] : st[l][y - (1 << l) + 1];
47 }
48
49 // 頂点vsを含むAuxiliary Treeを構築する
50 // 結果をg0に入れる
51 // 根頂点を返す
52 inline int auxiliary_tree(vector<int>& vs, vector<vector<int>>& g0) {
53     sort(vs.begin(), vs.end(), cmp_at);
54     int k = vs.size();
55     for(int i = 0; i < k - 1; ++i) { vs.push_back(lca(vs[i], vs[i + 1])); }
56     sort(vs.begin(), vs.end(), cmp_at);
57     int prv = -1;
58     int cur = 0;
59     for(int i = 0; i < vs.size(); ++i) {
60         int v = vs[i];
61         if(prv == v) continue;
62         while(cur > 0 && ls[stk[cur - 1]] < fs[v]) --cur;
63         if(cur > 0) {
64             g0[stk[cur - 1]].push_back(v);
65             g0[v].push_back(stk[cur - 1]);
66         }
67         // 有向
68         // if(cur > 0) {
69         //     g0[stk[cur-1]].push_back(v);
70         // }
71         // g0[v].clear();
72         stk[cur++] = v;
73         prv = v;
74     }
75     return stk[0];
76 }

```

Cartesiantree.hpp

md5: 41ed07

```

1  template<typename T> pair<vector<vector<int>>, int> CartesianTree(vector<T>& a) {
2      int N = (int)a.size();
3      vector<vector<int>> g(N);
4      vector<int> p(N, -1), st;
5      st.reserve(N);
6      for(int i = 0; i < N; i++) {
7          int prv = -1;
8          while(!st.empty() && a[i] < a[st.back()]) {
9              prv = st.back();
10             st.pop_back();
11         }
12         if(prv != -1) p[prv] = i;
13         if(!st.empty()) p[i] = st.back();
14         st.push_back(i);
15     }
16     int root = -1;
17     for(int i = 0; i < N; i++) {
18         if(p[i] != -1) g[p[i]].push_back(i);
19         else root = i;
20     }
21     return make_pair(g, root);
22 }
```

Rerooting.hpp

md5: 866b81

```

1  template<class E, class V, E (*merge)(E, E), E (*e)(), E (*put_edge)(V, int), V (*put_vertex)(E, int)>
2  struct RerootingDP {
3      struct edge {
4          int to, idx, xdi;
5      };
6      RerootingDP(int n_ = 0) : n(n_), inner_edge_id(0) {
7          es.resize(2 * n - 2);
8          start.resize(2 * n - 2);
9          if(n == 1) es_build();
10     }
11     void add_edge(int u, int v, int idx, int xdi) {
12         start[inner_edge_id] = u;
13         es[inner_edge_id] = {v, idx, xdi};
14         inner_edge_id++;
15         start[inner_edge_id] = v;
16         es[inner_edge_id] = {u, xdi, idx};
17         inner_edge_id++;
18         if(inner_edge_id == 2 * n - 2) { es_build(); }
19     }
20     vector<V> build(int root_ = 0) {
21         root = root_;
22         vector<V> subdp(n);
23         subdp[0] = put_vertex(e(), 0);
24         outs.resize(n);
25         vector<int> geta(n + 1, 0);
26         for(int i = 0; i < n; i++) geta[i + 1] = start[i + 1] - start[i] - 1;
27         geta[root + 1]++;
28         for(int i = 0; i < n; i++) geta[i + 1] += geta[i];
29         auto dfs = [&](auto sfs, int v, int f) -> void {
30             E val = e();
31             for(int i = start[v]; i < start[v + 1]; i++) {
32                 if(es[i].to == f) { swap(es[start[v + 1] - 1], es[i]); }
33                 if(es[i].to == f) continue;
34                 sfs(sfs, es[i].to, v);
35                 E nval = put_edge(subdp[es[i].to], es[i].idx);
36                 outs[geta[v]++] = nval;
37                 val = merge(val, nval);
38             }
39             subdp[v] = put_vertex(val, v);
40         };
41         dfs(dfs, root, -1);
42         return subdp;
43     }
44     vector<V> reroot() {
45         vector<E> reverse_edge(n);
46         reverse_edge[root] = e();
47         vector<V> answers(n);
48         auto dfs = [&](auto sfs, int v) -> void {
49             int le = outs_start(v);
50             int ri = outs_start(v + 1);
51             int siz = ri - le;
52             vector<E> rui(siz + 1);
53             rui[siz] = e();
54             for(int i = siz - 1; i >= 0; i--) { rui[i] = merge(outs[le + i], rui[i + 1]); }
55             answers[v] = put_vertex(merge(rui[0], reverse_edge[v]), v);

```

```

56     E lui = e();
57     for(int i = 0; i < siz; i++) {
58         V rdp = put_vertex(merge(merge(lui, rui[i + 1]), reverse_edge[v]), v);
59         reverse_edge[es[start[v] + i].to] = put_edge(rdp, es[start[v] + i].xdi);
60         lui = merge(lui, outs[le + i]);
61         sfs(sfs, es[start[v] + i].to);
62     }
63 };
64 dfs(dfs, root);
65 return answers;
66 }
67
68 private:
69 int n, root, inner_edge_id;
70 vector<E> outs;
71 vector<edge> es;
72 vector<int> start;
73 int outs_start(int v) {
74     int res = start[v] - v;
75     if(root < v) res++;
76     return res;
77 }
78 void es_build() {
79     vector<edge> nes(2 * n - 2);
80     vector<int> nstart(n + 2, 0);
81     for(int i = 0; i < 2 * n - 2; i++) nstart[start[i] + 2]++;
82     for(int i = 0; i < n; i++) nstart[i + 1] += nstart[i];
83     for(int i = 0; i < 2 * n - 2; i++) nes[nstart[start[i] + 1]++] = es[i];
84     swap(es, nes);
85     swap(start, nstart);
86 }
87 };
88
89 using S = ll;
90
91 using T = ll;
92
93 S merge(S a, S b) { return a * b; }
94 S e() { return 1; }
95 S put_edge(T v, int i) { return v + 1; }
96 T put_vertex(S e, int v) { return e; }

```

lca.hpp

md5: 281c40

```

1  template<class graph> struct lca {
2      const graph& G;
3      vector<vector<int>> parent;
4      vector<int> dep;
5      int log = 1;
6      lca(const graph& G, int root = 0) : G(G) { init(root); }
7      void init(int root = 0) {
8          const int n = G.size();
9          dep.assign(n, -1);
10         parent.assign(1, vector<int>(n, -1));
11         queue<int> que({root});
12         dep[root] = 0;
13         int max_dep = 0;
14         while(!que.empty()) {
15             const int pos = que.front();
16             que.pop();
17             max_dep = max(max_dep, dep[pos]);
18             for(const auto& x : G[pos]) {
19                 if(dep[x] == -1) {
20                     dep[x] = dep[pos] + 1;
21                     parent[0][x] = pos;
22                     que.push(x);
23                 }
24             }
25         }
26         while((1 << log) <= max_dep) log++;
27         parent.resize(log, vector<int>(n, -1));
28         for(int k = 0; k < log - 1; k++) {
29             for(int v = 0; v < n; v++) {
30                 if(parent[k][v] < 0) parent[k + 1][v] = -1;
31                 else parent[k + 1][v] = parent[k][parent[k][v]];
32             }
33         }
34     }
35     int query(int u, int v) const {
36         if(dep[u] < dep[v]) swap(u, v);
37         const int sub = dep[u] - dep[v];
38         for(int k = 0; k < log; k++) {
39             if(!(sub >> k)) break;

```

```
40         if(sub >> k & 1) u = parent[k][u];
41     }
42     if(u == v) return u;
43     for(int k = __lg(dep[u]); k >= 0; k--) {
44         if(parent[k][u] != parent[k][v]) {
45             u = parent[k][u];
46             v = parent[k][v];
47         }
48     }
49     return parent[0][u];
50 }
51 int dist(const int u, const int v) const { return dep[u] + dep[v] - dep[query(u, v)] * 2; }
52 int jump(int u, int v, int d) const {
53     const int lc = query(u, v);
54     const int l = dep[u] - dep[lc];
55     const int r = dep[v] - dep[lc];
56     if(d < 0 || d > l + r) return -1;
57     if(l < d) {
58         d = l + r - d;
59         swap(u, v);
60     }
61     for(int k = 0; k < log; k++) {
62         if(!(d >> k)) break;
63         if(d >> k & 1) u = parent[k][u];
64     }
65     return u;
66 }
67 };
```

flow

dinic.hpp

md5: fd2103

```

1  struct Dinic {
2      int V; // 頂点数
3      vector<vector<vector<ll>>> graph; // グラフ
4      vector<int> dis; // 始点からの距離
5      vector<int> next; // 次に処理する頂点のメモ
6      Dinic(int v) : V(v) { graph.resize(V); }
7      void add_edge(int from, int to, ll capacity) {
8          graph[from].push_back({to, capacity, (int)graph[to].size()});
9          graph[to].push_back({from, 0, (int)graph[from].size() - 1});
10     }
11     void bfs(int s) {
12         dis.assign(V, -1);
13         dis[s] = 0;
14         deque<int> pos = {s};
15         while(pos.size()) {
16             int now = pos[0];
17             pos.pop_front();
18             for(auto& to : graph[now]) {
19                 if(dis[to[0]] < 0 and to[1] > 0) {
20                     dis[to[0]] = dis[now] + 1;
21                     pos.emplace_back(to[0]);
22                 }
23             }
24         }
25     }
26     ll dfs(int v, int t, ll f) {
27         if(v == t) return f;
28         for(int& i = next[v]; i < graph[v].size(); i++) {
29             int to = graph[v][i][0];
30             ll& cap = graph[v][i][1];
31             int rev = graph[v][i][2];
32             if(cap > 0 and dis[v] < dis[to]) {
33                 ll d = dfs(to, t, min(f, cap));
34                 if(d > 0) {
35                     cap -= d;
36                     graph[to][rev][1] += d;
37                     return d;
38                 }
39             }
40         }
41         return 0;
42     }
43     ll max_flow(int s, int t) {
44         ll flow = 0;
45         while(1) {
46             bfs(s);
47             if(dis[t] < 0) return flow;
48             next.assign(V, 0);
49             ll f;
50             while((f = dfs(s, t, INF)) > 0) flow += f;
51         }
52     }
53 };

```

mincostflow.hpp

md5: e93128

```

1  struct MinCostFlow {
2      int V;
3      vector<vector<vector<ll>>> g; // g[from] = {{to, 容量, コスト, 逆辺のindex} ... }
4      vector<ll> h, dis; // ポテンシャル, 最短距離
5      vector<int> prevv, preve; // 直前の頂点, 辺
6
7      MinCostFlow(int v) : V(v), g(v), dis(v), prevv(v), preve(v) {}
8
9      void add_edge(int u, int v, ll c, ll d) {
10         g[u].push_back({v, c, d, (int)g[v].size()});
11         g[v].push_back({u, 0, -d, (int)g[u].size() - 1});
12     }
13
14     ll min_cost_flow(int s, int t, ll f) {
15         ll res = 0;
16         h.assign(V, 0);
17         using Q = pair<ll, int>;
18         while(f > 0) {
19             priority_queue<Q, vector<Q>, greater<Q>> que;
20             dis.assign(V, LLONG_MAX);

```



```
21     dis[s] = 0;
22     que.push({0, s});
23     while(que.size()) {
24         Q q = que.top();
25         int v = q.second;
26         que.pop();
27         if(dis[v] < q.first) continue;
28         for(int i = 0; i < g[v].size(); i++) {
29             auto edge = g[v][i];
30             int to = edge[0];
31             ll cap = edge[1], cost = edge[2];
32             if(cap > 0 and dis[to] > dis[v] + cost + h[v] - h[to]) {
33                 dis[to] = dis[v] + cost + h[v] - h[to];
34                 prevv[to] = v;
35                 preve[to] = i;
36                 que.push({dis[to], to});
37             }
38         }
39     }
40     if(dis[t] == INF) return -1;
41     for(int i = 0; i < V; i++) h[i] += dis[i];
42     ll d = f;
43     for(int i = t; i != s; i = prevv[i]) d = min(d, g[prevv[i]][preve[i]][1]);
44     f -= d;
45     res += d * h[t];
46     for(int i = t; i != s; i = prevv[i]) {
47         auto& edge = g[prevv[i]][preve[i]];
48         edge[1] -= d;
49         g[i][edge[3]][1] += d;
50     }
51 }
52 return res;
53 }
54 };
```

string

AhoCorasick.hpp

md5: dc1171

```
1 struct Aho {
2     using MP = unordered_map<char, int>;
3     vector<MP> to;
4     vector<int> cnt, fail;
5     Aho() : to(1), cnt(1) {}
6     int add(const string& s) {
7         int v = 0;
8         for(char c : s) {
9             if(!to[v].count(c)) {
10                 to[v][c] = to.size();
11                 to.push_back(MP());
12                 cnt.push_back(0);
13             }
14             v = to[v][c];
15         }
16         cnt[v]++;
17         return v;
18     }
19     void init() {
20         fail = vector<int>(to.size(), -1);
21         queue<int> q;
22         q.push(0);
23         while(!q.empty()) {
24             int v = q.front();
25             q.pop();
26             for(auto [c, u] : to[v]) {
27                 fail[u] = (*this)(fail[v], c);
28                 cnt[u] += cnt[fail[u]];
29                 q.push(u);
30             }
31         }
32     }
33     int operator()(int v, char c) const {
34         while(v != -1) {
35             auto it = to[v].find(c);
36             if(it != to[v].end()) return it->second;
37             v = fail[v];
38         }
39         return 0;
40     }
41     int operator[](int v) const { return cnt[v]; }
42 };
43
```

KMP.hpp

md5: 886c63

```
1 // kmp[i] := max{ l ≤ i | s[:l] == s[(i+1)-l:i+1] }
2 // abacaba -> 0010123
3 auto KMP(string s) {
4     vector<ll> p(sz(s));
5     rep(i, 1, sz(s)) {
6         ll g = p[i - 1];
7         while(g && s[i] != s[g]) g = p[g - 1];
8         p[i] = g + (s[i] == s[g]);
9     }
10     return p;
11 }
12
```

Manacher.hpp

md5: 5882fb

```
1 // 各位置での回文半径を求める
2 // aaabaaa -> 1214121
3 // 偶数長の回文を含めて直径を知るには、N+1 個の $ を挿入して 1 を引く
4 // $a$a$a$b$a$a$a$a$ -> 123432181234321
5 auto manacher(string s) {
6     ll n = sz(s), i = 0, j = 0;
7     vector<ll> r(n);
8     while(i < n) {
9         while(i >= j && i + j < n && s[i - j] == s[i + j]) j++;
10         r[i] = j;
11         ll k = 1;
12         while(i >= k && i + k < n && k + r[i - k] < j) {
13             r[i + k] = r[i - k];
14             k++;
15         }
16     }
17 }
```

```
16         i += k, j -= k;
17     }
18     return r;
19 }
20
```

RollingHash.hpp

md5: adb8d3

```
1 // using u64 = uint64_t;
2 const u64 mod = INF;
3 u64 add(u64 a, u64 b) {
4     a += b;
5     if(a >= mod) a -= mod;
6     return a;
7 }
8 u64 mul(u64 a, u64 b) {
9     auto c = (__uint128_t)a * b;
10    return add(c >> 61, c & mod);
11 }
12 random_device rnd;
13 const u64 r = ((u64)rnd() << 32 | rnd()) % mod;
14 struct RH {
15     ll n;
16     vector<u64> hs, pw;
17     RH(string s) : n(sz(s)), hs(n + 1), pw(n + 1, 1) {
18         rep(i, 0, n) {
19             pw[i + 1] = mul(pw[i], r);
20             hs[i + 1] = add(mul(hs[i], r), s[i]);
21         }
22     }
23     u64 get(ll l, ll r) const { return add(hs[r], mod - mul(hs[l], pw[r - l])); }
24 };
25
```

SuffixArray.hpp

md5: 1d70ce

```
1 // returns pair{sa, lcp}
2 // sa 長さ n : s[sa[0]:] < s[sa[1]:] < ... < s[sa[n-1]:]
3 // lcp 長さ n-1 : lcp[i] = LCP(s[sa[i]:], s[sa[i+1]:])
4 auto SA(string s) {
5     ll n = sz(s) + 1, lim = 256;
6     // assert(lim > ranges::max(s));
7     vector<ll> sa(n), lcp(n), x(all(s) + 1), y(n), ws(max(n, lim)), rk(n);
8     iota(all(sa), 0);
9     for(ll j = 0, p = 0; p < n; j = max(1LL, j * 2), lim = p) {
10         p = j;
11         iota(all(y), n - j);
12         rep(i, 0, n) if(sa[i] >= j) y[p++] = sa[i] - j;
13         fill(all(ws), 0);
14         rep(i, 0, n) ws[x[i]]++;
15         rep(i, 1, lim) ws[i] += ws[i - 1];
16         for(ll i = n; i--;) sa[--ws[x[y[i]]]] = y[i];
17         swap(x, y);
18         p = 1;
19         x[sa[0]] = 0;
20         rep(i, 1, n) {
21             ll a = sa[i - 1], b = sa[i];
22             x[b] = (y[a] == y[b] && y[a + j] == y[b + j]) ? p - 1 : p++;
23         }
24     }
25     rep(i, 1, n) rk[sa[i]] = i;
26     for(ll i = 0, k = 0; i < n - 1; lcp[rk[i++]] = k) {
27         if(k) k--;
28         while(s[i + k] == s[sa[rk[i] - 1] + k]) k++;
29     }
30     sa.erase(begin(sa));
31     lcp.erase(begin(lcp));
32     return pair{sa, lcp};
33 }
34
```

Trie.hpp

md5: 13af70

```
1 struct Trie {
2     static constexpr int C_SIZE = 26; // C_SIZE : 文字の種類数
3     static constexpr int C_BEGIN = 'a'; // C_BEGIN : 開始文字
4     static constexpr int ROOT = 0;
5     struct Node {
6         array<int, C_SIZE> to = {}; // 子ノードの番号, 存在しなければ-1
7         vector<int> ids; // そのノードが終端である文字列のIDリスト
8         Node() { to.fill(-1); }
```

```

9     };
10    vector<Node> nodes;
11    int cnt = 0; // 追加した文字列の個数
12    Trie() : nodes(1) {}
13    // nodes[idx]から文字cで遷移したときの頂点のindex
14    int next(int idx, char c) { return nodes[idx].to[c - C_BEGIN]; }
15    // 文字列の追加
16    int insert(const string& s) {
17        int now = ROOT;
18        for(char c : s) {
19            int k = c - C_BEGIN;
20            if(nodes[now].to[k] == -1) {
21                nodes[now].to[k] = nodes.size();
22                nodes.push_back(Node());
23            }
24            now = nodes[now].to[k];
25        }
26        nodes[now].ids.push_back(cnt++);
27        return now;
28    }
29    // 文字列に対応するnodeのindexを検索, 存在しなければ-1
30    int find(const string& s) {
31        int now = ROOT;
32        for(char c : s) {
33            now = next(now, c);
34            if(now == -1) return -1;
35        }
36        return now;
37    }
38 };

```

Zalgorithm.hpp

md5: b20b04

```

1 // Z[i] := LCP(s, s[i:])
2 // abacaba -> 7010301
3 auto Z(string s) {
4     ll n = sz(s), l = -1, r = -1;
5     vector<ll> z(n, n);
6     rep(i, 1, n) {
7         ll& x = z[i] = i < r ? min(r - i, z[i - l]) : 0;
8         while(i + x < n && s[i + x] == s[x]) x++;
9         if(i + x > r) l = i, r = i + x;
10    }
11    return z;
12 }
13

```

algorithm

mo.hpp

md5: 934d7d

```

1  struct Mo {
2      int n;
3      vector<pair<int, int> > lr;
4
5      explicit Mo(int n) : n(n) {}
6
7      void add(int l, int r) { /* [l, r) */
8          lr.emplace_back(l, r);
9      }
10
11     template<typename AL, typename AR, typename EL, typename ER, typename O>
12     void build(const AL& add_left, const AR& add_right, const EL& erase_left, const ER& erase_right, const O& out) {
13         int q = (int)lr.size();
14         int bs = n / min<int>(n, sqrt((double)q));
15         vector<int> ord(q);
16         iota(begin(ord), end(ord), 0);
17         sort(begin(ord), end(ord), [&](int a, int b) {
18             int ablock = lr[a].first / bs, bblock = lr[b].first / bs;
19             if(ablock != bblock) return ablock < bblock;
20             return (ablock & 1) ? lr[a].second > lr[b].second : lr[a].second < lr[b].second;
21         });
22         int l = 0, r = 0;
23         for(auto idx : ord) {
24             while(l > lr[idx].first) add_left(--l);
25             while(r < lr[idx].second) add_right(r++);
26             while(l < lr[idx].first) erase_left(l++);
27             while(r > lr[idx].second) erase_right(--r);
28             out(idx);
29         }
30     }
31
32     template<typename A, typename E, typename O> void build(const A& add, const E& erase, const O& out) {
33         build(add, add, erase, erase, out);
34     }
35 };
36
37 int main() {
38     int N;
39     cin >> N;
40     vector<int> A(N);
41     for(auto& a : A) cin >> a;
42     int Q;
43     cin >> Q;
44     Mo mo(N);
45     for(int i = 0; i < Q; i++) {
46         int a, b;
47         cin >> a >> b;
48         mo.add(a - 1, b);
49     }
50     vector<int> cnt(1000001), ans(Q);
51     int sum = 0;
52     auto add = [&](int i) {
53         if(cnt[A[i]]++ == 0) ++sum;
54     };
55     auto erase = [&](int i) {
56         if(--cnt[A[i]] == 0) --sum;
57     };
58     auto out = [&](int q) { ans[q] = sum; };
59     mo.build(add, erase, out);
60     for(auto& p : ans) cout << p << "\n";
61 }
62

```

geometry

geometry.hpp

md5: f4e0fc

```

1  /*
2  前提
3  - 点(位置ベクトル)を複素数型で扱う
4  - 実軸(real)をx軸、虚軸(imag)をy軸として見る
5  - 比較するときは、計算誤差を意識して EPS で判定 (equal関数)
6  */
7
8  namespace geometry {
9  using D = long double;
10 using Point = std::complex<D>;
11 using Polygon = vector<Point>;
12 const D EPS = 1e-8;
13 const D PI = M_PI;
14
15 // 入出力ストリーム
16 istream& operator>>(istream& is, Point& p) {
17     D a, b;
18     is >> a >> b;
19     p = Point(a, b);
20     return is;
21 }
22
23 ostream& operator<<(ostream& os, Point& p) { return os << fixed << setprecision(20) << p.real() << ' ' << p.imag(); }
24
25 // d 倍する
26 Point operator*(Point p, D d) { return Point(p.real() * d, p.imag() * d); }
27
28 // 偏角 (0 ≤ θ < 2π)
29 D argument(Point p) {
30     D res = arg(p);
31     if(res < 0.0) res += 2.0 * PI; // [-π, π] -> [0, 2π)
32     return res;
33 }
34
35 // 等しいかどうか (誤差で判定)
36 inline bool equal(D a, D b) { return fabs(a - b) < EPS; }
37
38 // 単位ベクトル
39 Point unit_vector(Point a) { return a / abs(a); };
40
41 // 法線ベクトル (逆向きがよければ (0, -1) をかける)
42 Point normal_vector(Point a, D dir = 1) { return a * Point(0, dir); }
43
44 // 内積: a・b = |a||b|cosθ
45 D dot(Point a, Point b) { return (a.real() * b.real() + a.imag() * b.imag()); }
46
47 // 外積: a×b = |a||b|sinθ (外積の大きさではないか?)
48 D cross(Point a, Point b) { return (a.real() * b.imag() - a.imag() * b.real()); }
49
50 // 反時計回りに theta 回転
51 Point rotate(Point a, D theta) {
52     D c = cos(theta), s = sin(theta);
53     return Point(c * a.real() - s * a.imag(), s * a.real() + c * a.imag());
54 }
55
56 // 直線
57 struct Line {
58     Point a, b;
59     Line() = default;
60     Line(Point a_, Point b_) : a(a_), b(b_) { assert(a_ != b_); };
61     // Ax+By=C
62     Line(D A, D B, D C) {
63         if(equal(A, 0)) {
64             a = Point(0, C / B), b = Point(1, C / B);
65         } else if(equal(B, 0)) {
66             b = Point(C / A, 0), a = Point(C / A, 1);
67         } else {
68             a = Point(0, C / B), b = Point(C / A, 0);
69         }
70     }
71 };
72
73 // 線分 (Line と同じ)
74 struct Segment : Line {
75     Segment() = default;

```

```

76     Segment(Point a_, Point b_) : Line(a_, b_){};
77 };
78
79 // 円 (中心と半径)
80 struct Circle {
81     Point p;
82     D r;
83     Circle(Point p_, D r_) : p(p_), r(r_) {}
84 };
85
86 // 射影: 直線 (線分) に 点p から引いた垂線の足を求める
87 Point projection(Line l, Point p) {
88     D t = dot(p - l.a, l.a - l.b) / norm(l.a - l.b);
89     return l.a + (l.a - l.b) * t;
90 }
91 Point projection(Segment l, Point p) {
92     D t = dot(p - l.a, l.a - l.b) / norm(l.a - l.b);
93     return l.a + (l.a - l.b) * t;
94 }
95
96 // 反射: 直線を対象軸として 点p と線対称の位置にある点を求める
97 Point reflection(Line l, Point p) { return p + (projection(l, p) - p) * 2.0; }
98
99 // 3点 a, b, c の位置関係
100 int ccw(Point a, Point b, Point c) {
101     b -= a, c -= a;
102     // 点 a, b, c が
103     if(cross(b, c) > EPS) return 1;    // 反時計回りのとき
104     if(cross(b, c) < -EPS) return -1; // 時計回りのとき
105
106     // 同一直線上にある場合
107     if(dot(b, c) < 0) return 2;        // c, a, b の順
108     if(norm(b) < norm(c)) return -2;   // a, b, c の順
109     return 0;                          // a, c, b の順
110 }
111
112 // 垂直 (内積 == 0)
113 bool is_vertical(Line a, Line b) { return equal(dot(a.b - a.a, b.b - b.a), 0); }
114
115 // 平行 (外積 == 0)
116 bool is_parallel(Line a, Line b) { return equal(cross(a.b - a.a, b.b - b.a), 0); }
117
118 // 線分の交差判定 (線分 s に対して, 線分 t の端点が反対側にあればよい)
119 bool is_intersect(Segment s, Segment t) {
120     return (ccw(s.a, s.b, t.a) * ccw(s.a, s.b, t.b) <= 0) && (ccw(t.a, t.b, s.a) * ccw(t.a, t.b, s.b) <= 0);
121 }
122
123 // 交点 (交差する前提)
124 Point cross_point(Line s, Line t) {
125     D d1 = cross(s.b - s.a, t.b - t.a);
126     D d2 = cross(s.b - s.a, s.b - t.a);
127     // s, t が一致する場合 (適当な1点を返す)
128     if(equal(abs(d1), 0) && equal(abs(d2), 0)) return t.a;
129
130     return t.a + (t.b - t.a) * (d2 / d1);
131 }
132 Point cross_point(Segment s, Segment t) {
133     assert(is_intersect(s, t)); // 交差する前提
134     return cross_point(Line(s), Line(t));
135 }
136
137 // 点間の距離
138 D dist(Point a, Point b) { return abs(a - b); }
139
140 // 点と直線の距離 (垂線の足との距離)
141 D dist_line_point(Line l, Point p) { return abs(p - projection(l, p)); }
142
143 // 線分と点の距離 (点p から線分のどこかへの最短距離)
144 D dist_segment_point(Segment l, Point p) {
145     if(dot(l.b - l.a, p - l.a) < EPS) return abs(p - l.a);
146     if(dot(l.a - l.b, p - l.b) < EPS) return abs(p - l.b);
147     return abs(cross(l.b - l.a, p - l.a)) / abs(l.b - l.a);
148 }
149
150 // 線分と線分の距離
151 D dist_segment_segment(Segment s, Segment t) {
152     if(is_intersect(s, t)) return 0.0;
153     D res = min({
154         dist_segment_point(s, t.a),
155         dist_segment_point(s, t.b),

```

```

156         dist_segment_point(t, s.a),
157         dist_segment_point(t, s.b),
158     });
159     return res;
160 }
161
162 // 2つの円の交点
163 pair<Point, Point> crosspoint(const Circle& c1, const Circle& c2) {
164     D d = abs(c1.p - c2.p);
165     D a = acos((c1.r * c1.r + d * d - c2.r * c2.r) / (2 * c1.r * d));
166     D t = atan2(c2.p.imag() - c1.p.imag(), c2.p.real() - c1.p.real());
167     Point p1 = c1.p + Point(cos(t + a) * c1.r, sin(t + a) * c1.r);
168     Point p2 = c1.p + Point(cos(t - a) * c1.r, sin(t - a) * c1.r);
169     return {p1, p2};
170 }
171
172 ll cross_cht(Point o, Point a, Point b) {
173     return (a.real() - o.real()) * (b.imag() - o.imag()) - (a.imag() - o.imag()) * (b.real() - o.real());
174 }
175
176 // 凸包
177 Polygon convex_hull(Polygon ps) {
178     int n = ps.size(), k = 0;
179     if(n <= 2) return ps;
180     sort(ps.begin(), ps.end(),
181          [](const Point& a, const Point& b) { return real(a) != real(b) ? real(a) < real(b) : imag(a) < imag(b); });
182     Polygon res;
183     for(auto p : ps) {
184         while((int)res.size() >= 2 && cross_cht(res[k - 1], res[k - 2], p) >= 0) {
185             res.pop_back();
186             k--;
187         }
188         res.push_back(p);
189         k++;
190     }
191     int t = res.size();
192     rrep(i, n - 2, 0) {
193         while((int)res.size() > t && cross_cht(res[k - 1], res[k - 2], ps[i]) >= 0) {
194             res.pop_back();
195             k--;
196         }
197         res.push_back(ps[i]);
198         k++;
199     }
200     return res;
201 }
202 }; // namespace geometry
203 using namespace geometry;
204

```

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

競プロ用 母関数・形式的冪級数 (FPS) チートシート

1. 母関数の種類と定義

名称	英語略称	定義式	主な用途
通常型母関数	OGF	$A(x) = \sum_{n=0}^{\infty} a_n x^n$	区別しないものの数え上げ (組合せ, 硬貨の支払い, 分割数)
指数型母関数	EGF	$A(x) = \sum_{n=0}^{\infty} a_n \frac{x^n}{n!}$	区別するものの数え上げ (順列, ラベル付きグラフ, 部屋割り)

2. 頻出展開公式 (OGF / EGF)

係数 a_n (または $a_n/n!$) が「1通りの操作」に対応する基本的な部品です。

関数 $f(x)$	級数展開 $\sum c_n x^n$	組合せ論的意味
$\frac{1}{1-x}$	$\sum_{n=0}^{\infty} x^n$	何個でも選べる (0個, 1個, 2個...)
$\frac{1}{1-x^2}$	$\sum_{n=0}^{\infty} x^{2n}$	偶数個選べる (0個, 2個, 4個...)
$\frac{x}{1-x^2}$	$\sum_{n=0}^{\infty} x^{2n+1}$	奇数個選べる (1個, 3個, 5個...)
$\frac{1-x^{k+1}}{1-x}$	$\sum_{i=0}^k x^i$	k 個まで選べる (個数制限付き)
$(1+x)^n$	$\sum_{k=0}^n \binom{n}{k} x^k$	n 個から k 個選ぶ (各要素を選ぶ/選ばない)
$\frac{1}{(1-x)^n}$	$\sum_{k=0}^{\infty} \binom{n+k-1}{k} x^k$	重複組合せ ${}_nH_k$ (n 種類から重複を許して k 個)
e^x	$\sum_{n=0}^{\infty} \frac{x^n}{n!}$	区別できる要素の基本構成 (EGF で頻出)
e^{ax}	$\sum_{n=0}^{\infty} \frac{a^n x^n}{n!}$	区別できる要素の基本構成 (EGF で頻出)
$-\log(1-x)$	$\sum_{n=1}^{\infty} \frac{x^n}{n}$	サイクル、連結成分の数え上げ

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, \dots 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, \dots がすべて 1 のとき z 得る	無条件で z 得る; $(w, T, z), (x, w, \infty), (y, w, \infty)$

--