# ICPC Notebook

template
hash.sh ..... 1
settings.sh ..... 1
template.hpp ..... 1
data-structure
BIT.hpp ..... 1
FastSet.hpp ..... 2
Rangetree.hpp ..... 2
ST.hpp ..... 3
SWAG.hpp ..... 3
Segtree-easy.hpp ..... 3
math
BinaryGCD.hpp ..... 4
ExtGCD.hpp ..... 4
Permutation.hpp ..... 4
modint
BarrettReduction.hpp ..... 4
modint.hpp ..... 4
FPS
FFT.hpp ..... 5
FFT_fast.hpp ..... 5
graph
UnionFindMonoid.hpp ..... 6
fast_graph.hpp ..... 6
graphALL.hpp ..... 6
graph/tree
flow
燃やす埋める.md ..... 7
string
KMP.hpp ..... 8
Manacher.hpp ..... 8
RollingHash.hpp ..... 8
SuffixArray.hpp ..... 8
Zalgorithm.hpp ..... 8
algorithm
geometry
DynamicConvexPolygon.hpp ..... 9
Geometry2D.hpp ..... 9
memo
Primes.md ..... 9

# 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: 586639

```cpp
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
const ll INF = LLONG_MAX / 4;
const ll LINF = 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; }

#define fi first
#define se second
#define eb emplace_back
#define pb push_back

void solve() {}

signed main() {
   cin.tie(0)->sync_with_stdio(0);
   int t = 1;
   cin >> t;
   while(t--) { solve(); }
}
```

# 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

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

## Rangetree.hpp

md5: 31d0a1

```
// rangeset(merge-sort tree) 静态结构，高效查询但不支持更新
//  一组点集上，查询矩形区域
//  RangeSet rs(x); x为vector<pair<>>;
//  rs.count(x1,x2,y1,y2), [x1,x2) * [y1,y2)
using ll = int64_t;
using pll = pair<ll, ll>;
const LINF = 0x1fffffffffffffff;

struct RangeSet {
   vector<vector<ll>> data;
   vector<ll> x;
   ll siz_x = 1;
   RangeSet(vector<pll> a) {
      sort(a.begin(), a.end());
      if(a.size()) x.push_back(a[0].first);
      for(auto& i : a)
         if(x.back() != i.first) x.push_back(i.first);
      while(x.size() > siz_x) siz_x *= 2;
      while(x.size() < siz_x) x.push_back(LINF);
      data.resize(siz_x * 2);
      ll at = 0;
      for(auto& i : a) {
         if(x[at] != i.first) at++;
         data[siz_x + at].push_back(i.second);
      }
      for(ll i = siz_x; --i;) {
         data[i].resize(data[i * 2].size() + data[i * 2 + 1].size());
         merge(data[i * 2].begin(), data[i * 2].end(), data[i * 2 + 1].begin(), data[i *
2 + 1].end(), data[i].begin());
      }
   }
   ll count(ll x1, ll x2, ll y1, ll y2) const {  // [x1, x2) × [y1, y2)
      ll l = lower_bound(x.begin(), x.end(), x1) - x.begin(), r = lower_bound(x.begin(),
x.end(), x2) - x.begin();
      l += siz_x;
      r += siz_x;
      ll ans = 0;
      for(; l < r; l /= 2, r /= 2) {
         if(l & 1) {
            ans += lower_bound(data[l].begin(), data[l].end(), y2) -
lower_bound(data[l].begin(), data[l].end(), y1);
            l++;
         }
         if(r & 1) {
            r--;
            ans += lower_bound(data[r].begin(), data[r].end(), y2) -
lower_bound(data[r].begin(), data[r].end(), y1);
         }
      }
      return ans;
   }
};
```

## ST.hpp

md5: 007a88

```
namespace ds {
template<typename T> struct ST {
   vector<vector<T>> table;
   vector<int> Lg;
   int n;
   static T default_func(const T& t1, const T& t2) { return max(t1, t2); }
   function<T(const T&, const T&)> op = default_func;
   ST() {
      table.clear();
      Lg = vector<int>{0, 0};
      n = 0;
   }
   ST(vector<T> v, function<T(const T&, const T&)> func = default_func) {
      Lg = vector<int>{0, 0};
      init(v, func);
   }
   void init(vector<T> v, function<T(const T&, const T&)> func) {
      op = func;
      table.clear();
      n = v.size();
      if((int)Lg.size() <= n) {
         int tmp = Lg.size();
         Lg.resize(n + 1);
         for(int i = tmp; i <= n; i++) Lg[i] = Lg[i >> 1] + 1;
      }
      int l = Lg[n];
      table.push_back(v);
      for(int i = 1, tmp; i <= l; i++) {
         table.push_back(vector<int>(tmp = n - (1 << i) + 1));
         for(int j = 0; j < tmp; j++) table[i][j] = op(table[i - 1][j], table[i - 1][j +
(1 << (i - 1))]);
      }
   }
   T query(int l, int r) {
      int u = Lg[r - l + 1];
      return op(table[u][l], table[u][r - (1 << u) + 1]);
   }
};
}  // namespace ds
using ds::ST;
```

## SWAG.hpp

md5: 980cd3

```
// 滑动窗口！
// 用法：需要自定义 using T = ;
//       inline T f(T a, T b) {}; 聚合函数
//       const T def_value = -INF; 默认空窗口的数据
//       函数实例：a+b,min(a,b),max(a,b),gcd(a,b)
struct SWAG {
   using T = <#ll #>;
   inline T f(T a, T b) const { return <#gcd(a, b) #>; }
   const T def_value = <# - LINF #>;
   ll size = 0;
   vector<T> fold_l, r;
   T fold_r;
   T get() const {
      if(r.empty()) {
         if(fold_l.empty()) return def_value;
         return fold_l.back();
      }
      if(fold_l.empty()) return fold_r;
      return f(fold_l.back(), fold_r);
   }
   void push(T a) {
      if(r.empty()) fold_r = a;
      else fold_r = f(fold_r, a);
      r.push_back(a);
      size++;
   }
   void pop() {
      if(fold_l.empty()) {
         for(ll i = size; i--;) fold_l.push_back(r[i]);
         r.clear();
         for(ll i = 1; i < size; i++) fold_l[i] = f(fold_l[i], fold_l[i - 1]);
      }
      fold_l.pop_back();
      size--;
   }
};
```

## Segtree-easy.hpp

md5: dc45c7

```
// easy version

struct SegmentTree {
   ll size = 1;
   vector<ll> data;
   SegmentTree(ll n) {
      while(size < n) size *= 2;
      data.assign(size * 2, -LINF);
   }
   void update(ll at) {
      while(at /= 2) data[at] = max(data[at * 2], data[at * 2 + 1]);
   }
   void set(ll at, ll val) {
      at += size;
      data[at] = val;
      update(at);
   }
   ll get(ll l, ll r) {
      ll ans = -LINF;
      l += size;
      r += size;
      for(; l < r; l /= 2, r /= 2) {
         if(l & 1) chmax(ans, data[l++]);
         if(r & 1) chmax(ans, data[--r]);
      }
      return ans;
   }
};
```

# math

## BinaryGCD.hpp

md5: f3ab31

```cpp
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

```cpp
// 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;
}
```

## Permutation.hpp

md5: 9c154e

```cpp
// 置换类： 组合，逆置换，幂运算，0-based
// 直接初始化: Permutation p = {}, q(vector<>{});
// p.inv() 求逆置换
// p * q 置换组合，p *= q 就地组合;
// p.pow(k) 置换的幂
// Permutation::identity 生成恒等置换:
using ll = long long;

struct Permutation {  // 0-indexed
   vector<ll> perm;
   Permutation(initializer_list<ll>&& perm) : perm(perm) {}
   Permutation(vector<ll>&& perm) : perm(perm) {}
   template<class T> Permutation(const initializer_list<T>& perm) : perm(perm.begin(),
perm.end()) {}
   template<class T> Permutation(const vector<T>& perm) : perm(perm.begin(), perm.end())
{}
   static Permutation identity(ll n) {
      vector<ll> ans(n);
      iota(ans.begin(), ans.end(), 0);
      return ans;
   }
   Permutation inv() const {
      vector<ll> ans(perm.size());
      for(ll i = 0; i < perm.size(); i++) ans[perm[i]] = i;
      return ans;
   }
   Permutation operator*(const Permutation& x) const {
      vector<ll> ans(perm.size());
      for(ll i = 0; i < perm.size(); i++) ans[i] = perm[x.perm[i]];
      return ans;
   }
   Permutation& operator*=(const Permutation& x) {
      vector<ll> ans(perm.size());
      for(ll i = 0; i < perm.size(); i++) ans[i] = perm[x.perm[i]];
      perm.swap(ans);
      return *this;
   }
   Permutation pow(ll x) const {
      Permutation ans = identity(perm.size()), cnt = *this;
      while(x) {
         if(x & 1) ans *= cnt;
         x /= 2;
         cnt *= cnt;
      }
      return ans;
   }
};
```

# modint

## BarrettReduction.hpp

md5: 2ca7f3

```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: 81b530

```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(); }
   // 下面四行 虚数
   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

## UnionFindMonoid.hpp

md5: 1120a6

```cpp
// 带权并查集
template<class S, auto op, auto e> class UnionFindMonoid {
   static_assert(std::is_convertible_v<decltype(op), std::function<S(S, S)>>, "op must
work as S(S, S)");
   static_assert(std::is_convertible_v<decltype(e), std::function<S()>>, "e must work as
S()");
   int _n;
   std::vector<int> parent_or_size;
   std::vector<S> data;

   public:
   UnionFindMonoid(int n = 0) : _n(n), parent_or_size(n, -1), data(n, e()) {}
   UnionFindMonoid(const std::vector<S>& a) : _n((int)a.size()), parent_or_size(a.size(),
-1), data(a) {}

   int leader(int a) {
      assert(0 <= a && a < _n);
      while(true) {
         const int b = parent_or_size[a];
         if(b < 0) return a;
         const int c = parent_or_size[b];
         if(c < 0) return b;
         a = parent_or_size[a] = c;  // path halving
      }
   }
   bool is_leader(int a) const {
      assert(0 <= a && a < _n);
      return parent_or_size[a] < 0;
   }
   int merge(int a, int b) {
      assert(0 <= a && a < _n);
      assert(0 <= b && b < _n);
      a = leader(a);
      b = leader(b);
      if(a == b) return a;
      if(-parent_or_size[a] < -parent_or_size[b]) swap(a, b);
      parent_or_size[a] += parent_or_size[b];
      parent_or_size[b] = a;
      data[a] = op(data[a], data[b]);
      return a;
   }
   bool same(int a, int b) {
      assert(0 <= a && a < _n);
      assert(0 <= b && b < _n);
      return leader(a) == leader(b);
   }
   int size(int a) {
      assert(0 <= a && a < _n);
      return -parent_or_size[leader(a)];
   }
   S get(int a) {
      assert(0 <= a && a < _n);
      return data[leader(a)];
   }
   void set(int a, S x) {
      assert(0 <= a && a < _n);
      data[leader(a)] = x;
   }
};
```

## fast_graph.hpp

md5: 7c9c15

```cpp
// CSR压缩邻接表，静态模型
//
class fast_graph {
   using I = vector<int>::iterator;
   struct edges {
      I l, r;
      bool empty() const { return l == r; }
      size_t size() const { return r - l; }
      auto begin() const { return l; }
      auto end() const { return r; }
      int operator[](size_t i) const { return l[i]; }
   };

   vector<int> mem;
   vector<edges> g;

   public:
   fast_graph() {}
   fast_graph(int n, vector<pair<int, int>> edge) : g(n), mem(edge.size() * 2) {
      for(auto [a, b] : edge) {
         g[a].r++;
         g[b].r++;
      }
      I sum = mem.begin();
      for(int i = 0; i < n; i++) { tie(g[i].l, g[i].r, sum) = tuple{sum, sum, sum +
(g[i].r - g[i].l)}; }
      for(auto [a, b] : edge) {
         *g[a].r++ = b;
         *g[b].r++ = a;
      }
   }
   bool empty() const { return g.empty(); }
   size_t size() const { return g.size(); }
   auto begin() const { return g.begin(); }
   auto end() const { return g.end(); }
   const auto& operator[](size_t i) const { return g[i]; }
};
```

## graphALL.hpp

md5: d03b78

```cpp
// 勿忘，图基础算法模板大全！！！！
struct WeightedEdge {
   ll to, cost;
   WeightedEdge() {}
   WeightedEdge(ll to, ll cost) : to(to), cost(cost) {}
   operator ll() const { return to; }
};
struct WeightedGraph {
   using E = WeightedEdge;
   vector<vector<E>> g;
   WeightedGraph() {}
   WeightedGraph(ll n) : g(n) {}
   vector<E>& operator[](ll at) { return g[at]; }
   operator vector<vector<E>>&() { return g; }
   auto begin() const { return g.cbegin(); }
```

```
    auto end() const { return g.cend(); }
    ll size() const { return g.size(); }
    const vector<E>& operator[](ll at) const { return g[at]; }
    operator const vector<vector<E>>&() const { return g; }
    void resize(ll a) { g.resize(a); }
    void add_edge(ll a, ll b, ll cost) {
        g[a].emplace_back(b, cost);
        g[b].emplace_back(a, cost);
    }
    void add_directed_edge(ll from, ll to, ll cost) { g[from].emplace_back(to, cost); }
    template<ll start_index = 1, bool directed = false> void input_graph(ll m) {
        while(m--) {
            ll a, b, c;
            scanf("%lld%lld%lld", &a, &b, &c);
            a -= start_index;
            b -= start_index;
            g[a].emplace_back(b, c);
            if(!directed) g[b].emplace_back(a, c);
        }
    }
    template<ll start_index = 1> void input_tree(ll n) { input_graph<start_index>(n - 1);
}
};

struct UnWeightedEdge {
    ll to;
    static constexpr ll cost = 1;
    UnWeightedEdge() {}
    UnWeightedEdge(ll to) : to(to) {}
    operator ll() const { return to; }
};
struct UnWeightedGraph {
    using E = UnWeightedEdge;
    vector<vector<E>> g;
    UnWeightedGraph() {}
    UnWeightedGraph(ll n) : g(n) {}
    vector<E>& operator[](ll at) { return g[at]; }
    operator vector<vector<E>>&() { return g; }
    auto begin() const { return g.cbegin(); }
    auto end() const { return g.cend(); }
    ll size() const { return g.size(); }
    const vector<E>& operator[](ll at) const { return g[at]; }
    operator const vector<vector<E>>&() const { return g; }
    void resize(ll a) { g.resize(a); }
    void add_edge(ll a, ll b) {
        g[a].emplace_back(b);
        g[b].emplace_back(a);
    }
    void add_directed_edge(ll from, ll to) { g[from].emplace_back(to); }
    template<ll start_index = 1, bool directed = false> void input_graph(ll m) {
        while(m--) {
            ll a, b;
            scanf("%lld%lld", &a, &b);
            a -= start_index;
            b -= start_index;
            g[a].emplace_back(b);
            if(!directed) g[b].emplace_back(a);
        }
    }
    template<ll start_index = 1> void input_tree(ll n) { input_graph<start_index>(n - 1);
}
};
```

```
template<class Graph> vector<ll> Dijkstra(const Graph& g, ll start) {
    vector<ll> cost(g.size(), LINF);
    vector<bool> used(g.size());
    pq<pll> q;
    cost[start] = 0;
    q.emplace(0, start);
    while(q.size()) {
        ll at = q.top().second;
        q.pop();
        if(used[at]) continue;
        used[at] = 1;
        each(i, g[at]) if(chmin(cost[i], cost[at] + i.cost)) q.emplace(cost[i], i);
    }
    return cost;
}

vector<ll> BFS(const UnWeightedGraph& g, ll start) {
    vector<ll> cost(g.size(), LINF);
    queue<ll> q;
    cost[start] = 0;
    q.push(start);
    while(q.size()) {
        ll at = q.front();
        q.pop();
        each(i, g[at]) if(chmin(cost[i], cost[at] + i.cost)) q.push(i);
    }
    return cost;
}

template<class Graph> vector<vector<ll>> WarshallFloyd(const Graph& g) {
    ll n = g.size();
    vector<vector<ll>> cost(n, vector<ll>(n, LINF));
    queue<ll> q;
    for(ll i = 0; i < n; i++) cost[i][i] = 0;
    for(ll i = 0; i < n; i++)
        for(auto& j : g[i]) cost[i][j] = j.cost;
    for(ll k = 0; k < n; k++)
        for(ll i = 0; i < n; i++)
            for(ll j = 0; j < n; j++) chmin(cost[i][j], cost[i][k] + cost[k][j]);
    return cost;
}
```

## graph/tree

## flow

### 燃やす埋める.md

| 原始约束条件 | 网络流形式的约束（变形后） |
| --- | --- |
| 当 x = 0 时，z 不可获得（或失去 z） | 在网络中添加边 (x, T, z)，表示若 x 取 0，则 z 与汇点 T 相连（流量受限） |
| 当 x = 0 时，z 可获得 | 视为无条件获得 z；在网络中添加边 (S, x, z)，即从源点 S 到 x 再到 z |
| 当 x = 1 时，z 不可获得（或失去 z） | 在网络中添加边 (S, x, z)，表示若 x 取 1，则 S 经 x 控制 z 的流动 |

| 原始约束条件 | 网络流形式的约束（变形后） |
| --- | --- |
| 当 x = 1 时，z 可获得 | 视为无条件获得 z；在网络中添加边 (x, T, z)，即 x 通向汇点 T 并激活 z |
| 当 x, y, … 全为 0 时，z 可获得 | 视为无条件获得 z；引入辅助节点 w，并添加边：(S, w, z)、(w, x, ∞)、(w, y, ∞)，表示 w 由多个变量控制，均为 0 时可达 |
| 当 x, y, … 全为 1 时，z 可获得 | 视为无条件获得 z；引入辅助节点 w，并添加边：(w, T, z)、(x, w, ∞)、(y, w, ∞)，表示 w 仅在所有变量取 1 时与汇点 T 连通 |

# string

## KMP.hpp

md5: 886c63

```cpp
// 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.hpp

md5: 5882fb

```cpp
// 各位置での回文半径を求める
// aaabaaa -> 1214121
// 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
// $a$a$a$b$a$a$a$ -> 123432181234321
auto manacher(string s) {
   ll n = sz(s), i = 0, j = 0;
   vector<ll> r(n);
   while(i < n) {
      while(i >= j && i + j < n && s[i - j] == s[i + j]) j++;
      r[i] = j;
      ll k = 1;
      while(i >= k && i + k < n && k + r[i - k] < j) {
         r[i + k] = r[i - k];
         k++;
      }
      i += k, j -= k;
   }
   return r;
}
```

## RollingHash.hpp

md5: adb8d3

```cpp
// using u64 = uint64_t;
const u64 mod = INF;
u64 add(u64 a, u64 b) {
   a += b;
   if(a >= mod) a -= mod;
   return a;
}
u64 mul(u64 a, u64 b) {
   auto c = (__uint128_t)a * b;
   return add(c >> 61, c & mod);
}
random_device rnd;
const u64 r = ((u64)rnd() << 32 | rnd()) % mod;
struct RH {
   ll n;
   vector<u64> hs, pw;
   RH(string s) : n(sz(s)), hs(n + 1), pw(n + 1, 1) {
      rep(i, 0, n) {
         pw[i + 1] = mul(pw[i], r);
         hs[i + 1] = add(mul(hs[i], r), s[i]);
      }
   }
   u64 get(ll l, ll r) const { return add(hs[r], mod - mul(hs[l], pw[r - l])); }
};
```

## SuffixArray.hpp

md5: 1d70ce

```cpp
// 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.hpp

md5: b20b04

```cpp
// Z[i] := LCP(s, s[i:])
// abacaba -> 7010301
auto Z(string s) {
   ll n = sz(s), l = -1, r = -1;
   vector<ll> z(n, n);
   rep(i, 1, n) {
```

```
      ll& x = z[i] = i < r ? min(r - i, z[i - l]) : 0;
      while(i + x < n && s[i + x] == s[x]) x++;
      if(i + x > r) l = i, r = i + x;
   }
   return z;
}
```

# algorithm

# geometry

## DynamicConvexPolygon.hpp

md5: b07234

```
struct Polygon {
   map<ll, ll> lower{{0, 0}}, upper{{0, 0}};
   ll size() const {
      return lower.size() + upper.size() - (*lower.begin() == *upper.begin()) -
(*lower.rbegin() == *upper.rbegin());
   }
   void add(ll x, ll y) {
      {
         auto can_erase = [&](map<ll, ll>::iterator p) -> bool {
            if(p == lower.begin()) return 0;
            if(next(p) == lower.end()) return 0;
            return cross(*next(p) - *p, *prev(p) - *p) <= 0;
         };
         auto [p, f] = lower.try_emplace(x, y);
         chmin(p->second, y);
         if(can_erase(p)) {
            lower.erase(p);
            return;
         }
         while(p != lower.begin() && can_erase(prev(p))) lower.erase(prev(p));
         while(next(p) != lower.end() && can_erase(next(p))) lower.erase(next(p));
      }
      {
         auto can_erase = [&](map<ll, ll>::iterator p) -> bool {
            if(p == upper.begin()) return 0;
            if(next(p) == upper.end()) return 0;
            return cross(*prev(p) - *p, *next(p) - *p) <= 0;
         };
         auto [p, f] = upper.try_emplace(x, y);
         chmax(p->second, y);
         if(can_erase(p)) {
            upper.erase(p);
            return;
         }
         while(p != upper.begin() && can_erase(prev(p))) upper.erase(prev(p));
         while(next(p) != upper.end() && can_erase(next(p))) upper.erase(next(p));
      }
   }
};
```

## Geometry2D.hpp

md5: 295c9e

```
template<class T> using P = pair<T, T>;
#define x first
#define y second
template<class T> P<T> operator+(const P<T>& a, const P<T>& b) { return {a.x + b.x, a.y +
b.y}; }
template<class T> P<T> operator-(const P<T>& a, const P<T>& b) { return {a.x - b.x, a.y -
b.y}; }
template<class T> P<T> operator-(const P<T>& a) { return {-a.x, -a.y}; }
template<class T, class U> P<T> operator*(const P<T>& a, const U& b) { return {a.x * b,
a.y * b}; }
template<class T, class U> P<T> operator/(const P<T>& a, const U& b) { return {a.x / b,
a.y / b}; }
template<class T> P<T>& operator+=(P<T>& a, const P<T>& b) { return a = a + b; }
template<class T> P<T>& operator-=(P<T>& a, const P<T>& b) { return a = a - b; }
template<class T, class U> P<T>& operator*=(P<T>& a, const U& b) { return a = a * b; }
template<class T, class U> P<T>& operator/=(P<T>& a, const U& b) { return a = a / b; }
template<class T> P<T> rotate(const P<T>& a) { return {-a.y, a.x}; }  // 90 degree ccw
template<class T> T dot(const P<T>& a, const P<T>& b) { return a.x * b.x + a.y * b.y; }
template<class T> T cross(const P<T>& a, const P<T>& b) { return dot(rotate(a), b); }
template<class T> T square(const P<T>& a) { return dot(a, a); }
template<class T> ld abs(const P<T>& a) { return hypotl(a.x, a.y); }
template<class T> T gcd(const P<T>& a) { return gcd(a.x, a.y); }
template<class T> P<T> normalize(P<T> a) {
   if(a == P<T>{}) return a;
   a /= gcd(a);
   if(a < P<T>{}) a = -a;
   return a;
}
```

# memo

## Primes.md

**素数个数**

| n | 10^2 | 10^3 | 10^4 | 10^5 | 10^6 | 10^7 | 10^8 | 10^9 | 10^10 |
|---|---|---|---|---|---|---|---|---|---|
| π(n) | 25 | 168 | 1229 | 9592 | 78498 | 664579 | 5760000 | 50800000 | 455000000 |

**高度合成数**

| ≤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 |

| ≤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 |

**素数阶乘**

| n | 2 | 3 | 5 | 7 | 11 | 13 | 17 | 19 | 23 | 29 |
|---|---|---|---|---|---|---|---|---|---|---|
| n# | 2 | 6 | 30 | 210 | 2310 | 30030 | 510510 | 9702000 | 223000000 | 6470000000 |

**阶乘**

| 4! | 5! | 6! | 7! | 8! | 9! | 10! | 11! | 12! | 13! |
|---|---|---|---|---|---|---|---|---|---|
| 24 | 120 | 720 | 5040 | 40320 | 362880 | 3628800 | 39916800 | 479001600 | 6227020800 |