Team Reference Document

Heltion

2024 年 10 月 15 日

目录

1 Template **1**

2 Data **3**

3 DP **32**

4 Geometry **33**

5 Graph **42**

6 Math **58**

7 String **82**

8 Basic **91**

1 Template

1.1 .clang-format

```
BasedOnStyle: Google
IndentWidth: 4
ColumnLimit: 160
```

1.2 debug.cpp

```
#include <bits/stdc++.h>

using namespace std;

template <class T, size_t size = tuple_size<T>::value>
string to_debug(T, string s = "") requires(not ranges::range<T>);

template<class T>
concept check = requires(T x, ostream &os) {
  os << x;
};

template<check T>
string to_debug(T x) {
  return static_cast<ostringstream>(ostringstream() << x).str();
}

string to_debug(ranges::range auto x, string s = "") requires(not is_same_v<
    decltype(x), string>) {
  for (auto xi : x) {
    s += ",␣" + to_debug(xi);
  }
  return "[␣" + s.substr(s.empty() ? 0 : 2) + "␣]";
}

template <class T, size_t size>
string to_debug(T x, string s) requires(not ranges::range<T>) {
  [&]<size_t... I>(index_sequence<I...>) {
    ((s += ",␣" + to_debug(get<I>(x))), ...);
  }(make_index_sequence<size>());
  return "{" + s.substr(s.empty() ? 0 : 2) + "}";
}

#define debug(...) [](auto... $){ ((cout << to_debug($) << "␣"), ...); cout
    << endl; }("[", #__VA_ARGS__, "]:", __VA_ARGS__)
```

1.3 gen.py

```
from random import *
n = 10000
s = 'qwertyuiopasdfghjklzxcvbnm'
for i in range(n):
    print(choice(s), end = '')
print()
print(randint(0, 1), randint(1, n))
```

1.4 head.cpp

```
#pragma GCC optimize("Ofast", "inline", "unroll-loops")
#include <bits/stdc++.h>
using namespace std;
#define rep(i, a, n) for (int i = a; i <= n; i++)
#define per(i, a, n) for (int i = a; i >= n; i--)
#define pb push_back
#define eb emplace_back
#define all(x) (x).begin(), (x).end()
```

```
#define bit(x) (1ll << (x))
#define fi first
#define se second
#define SZ(x) ((int)(x).size())
using VI = vector<int>;
using PII = pair<int, int>;
using ll = long long;
using ull = unsigned long long;
using db = double;
using ldb = long double;
mt19937_64 rng(chrono::steady_clock::now().time_since_epoch().count());
// head

#ifdef DEBUG
#include "debug.cpp"
#else
#define debug(...) 42
#endif

void solve() {}
int main() {
    cin.tie(nullptr)->sync_with_stdio(false);
    cout << fixed << setprecision(16);
    int tt = 1;
    cin >> tt;
    while (tt--) {
        solve();
    }
}
```

1.5 head-apiadu.cpp

```
#include <bits/stdc++.h>
using namespace std;
#define rep(i,a,n) for (int i=a;i<n;i++)
#define per(i,a,n) for (int i=n-1;i>=a;i--)
#define pb push_back
#define eb emplace_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define fi first
#define se second
#define SZ(x) ((int)(x).size())
typedef vector<int> VI;
typedef basic_string<int> BI;
typedef long long ll;
typedef pair<int,int> PII;
typedef double db;
mt19937 mrand(random_device{}());
const ll mod=1000000007;
int rnd(int x) { return mrand() % x;}
ll powmod(ll a,ll b) {ll res=1;a%=mod; assert(b>=0); for(;b;b>>=1){if(b&1)
    res=res*a%mod;a=a*a%mod;}return res;}
ll gcd(ll a,ll b) { return b?gcd(b,a%b):a;}
// head
```

1.6 Makefile

```
%: %.cpp
    g++-13 $< -o $@ -std=gnu++20 -g -O2 -Wall -Wextra -DDEBUG -
        D_GLIBCXX_DEBUG -D_GLIBCXX_DEBUG_PEDANTIC
```

1.7 pai.py

```
import os
tt = 0
while True:
    os.system('python␣gen.py␣>␣A.in')
    os.system('./a␣<␣A.in␣>␣a.out')
    os.system('./b␣<␣A.in␣>␣b.out')
    # diff for linux or macos, fc for windows
    if os.system('diff␣a.out␣b.out'):
        print("WA")
        exit(0)
    else:
        tt += 1
        print("AC:", tt)
```

1.8 settings.json

```
{
    "editor.formatOnSave": true,
    "files.autoSave": "afterDelay",
    "files.autoSaveDelay": 350,
    "C_Cpp.default.cppStandard": "gnu++20"
}
```

2 Data

2.1 01trie.cpp

```
struct node {
    int son[2];
    int end;
    int sz;
} seg[maxn << 2];
int root, tot;
int n, m;

void insert(ll x) {
    int cnt = root;
    for (int i = 62; i >= 0; i--) {
        int w = (x >> i) & 1;
        if (seg[cnt].son[w] == 0) seg[cnt].son[w] = ++tot;
        cnt = seg[cnt].son[w];
        seg[cnt].sz++;
    }
    seg[cnt].end++;
}

ll query(ll x, ll k) {
    ll res = 0;
    int cnt = root;
    for (int i = 62; i >= 0; i--) {
        int w = (x >> i) & 1;
        if (seg[seg[cnt].son[w]].sz >= k) cnt = seg[cnt].son[w];
        else {
            k -= seg[seg[cnt].son[w]].sz;
            cnt = seg[cnt].son[abs(w - 1)];
            res += bit(i);
        }
    }
    return res;
}
```

2.2 2dtree(bqi343).cpp

```
const int SZ = 1.1e5;
template <class T>
struct node {
    T val = 0;
    node<T>* c[2];
    node() { c[0] = c[1] = NULL; }
    void upd(int ind, T v, int L = 0, int R = SZ - 1) {  // add v
        if (L == ind && R == ind) {
            val += v;
            return;
        }
        int M = (L + R) / 2;
        if (ind <= M) {
            if (!c[0]) c[0] = new node();
            c[0]->upd(ind, v, L, M);
        } else {
            if (!c[1]) c[1] = new node();
            c[1]->upd(ind, v, M + 1, R);
        }
        val = 0;
        rep(i, 0, 1) if (c[i]) val += c[i]->val;
    }
    T query(int lo, int hi, int L = 0, int R = SZ - 1) {  // query sum of
        segment
        if (hi < L || R < lo) return 0;
        if (lo <= L && R <= hi) return val;
        int M = (L + R) / 2;
        T res = 0;
        if (c[0]) res += c[0]->query(lo, hi, L, M);
        if (c[1]) res += c[1]->query(lo, hi, M + 1, R);
        return res;
    }
    void UPD(int ind, node* c0, node* c1, int L = 0, int R = SZ - 1) {  //
        for 2D segtree
        if (L != R) {
            int M = (L + R) / 2;
            if (ind <= M) {
                if (!c[0]) c[0] = new node();
                c[0]->UPD(ind, c0 ? c0->c[0] : NULL, c1 ? c1->c[0] : NULL, L
                    , M);
            } else {
                if (!c[1]) c[1] = new node();
                c[1]->UPD(ind, c0 ? c0->c[1] : NULL, c1 ? c1->c[1] : NULL, M
                     + 1, R);
            }
        }
```

```
        val = (c0 ? c0->val : 0) + (c1 ? c1->val : 0);
    }
};

/**
 * Description: BIT of SegTrees. $x\in (0,SZ), y\in [0,SZ)$.
 * Memory: O(N\log^2 N)
 * Source: USACO Mowing the Field
 * Verification:
 * USACO Mowing the Field
 * http://www.usaco.org/index.php?page=viewproblem2&cpid=722 (13/15, 15/15
     and 1857ms with BumpAllocator)
 */

#include "../1D Range Queries (9.2)/SparseSeg (9.2).h"

template <class T>
struct BITseg {
    node<T> seg[SZ];
    BITseg() { fill(seg, seg + SZ, node<T>()); }
    void upd(int x, int y, int v) {  // add v
        for (; x < SZ; x += x & -x) seg[x].upd(y, v);
    }
    T query(int x, int yl, int yr) {
        T res = 0;
        for (; x; x -= x & -x) res += seg[x].query(yl, yr);
        return res;
    }
    T query(int xl, int xr, int yl, int yr) {  // query sum of rectangle
        return query(xr, yl, yr) - query(xl - 1, yl, yr);
    }
};

/**
 * Description: SegTree of SegTrees. $x,y\in [0,SZ).$
 * Memory: O(N\log^2 N)
 * Source: USACO Mowing the Field
 * Verification:
 * http://www.usaco.org/index.php?page=viewproblem2&cpid=722 (9/15 w/
     BumpAllocator)
 * http://www.usaco.org/index.php?page=viewproblem2&cpid=601 (4238 ms, 2907
     ms w/ BumpAllocator)
 */

#include "../1D Range Queries (9.2)/SparseSeg (9.2).h"

template <class T>
struct Node {
    node<T> seg;
    Node* c[2];
    Node() { c[0] = c[1] = NULL; }
    void upd(int x, int y, T v, int L = 0, int R = SZ - 1) {  // add v
        if (L == x && R == x) {
            seg.upd(y, v);
            return;
        }
        int M = (L + R) / 2;
        if (x <= M) {
            if (!c[0]) c[0] = new Node();
            c[0]->upd(x, y, v, L, M);
        } else {
            if (!c[1]) c[1] = new Node();
            c[1]->upd(x, y, v, M + 1, R);
        }
        seg.upd(y, v);  // only for addition
        // seg.UPD(y,c[0]?&c[0]->seg:NULL,c[1]?&c[1]->seg:NULL);
    }
    T query(int x1, int x2, int y1, int y2, int L = 0, int R = SZ - 1) {  //
         query sum of rectangle
        if (x1 <= L && R <= x2) return seg.query(y1, y2);
        if (x2 < L || R < x1) return 0;
        int M = (L + R) / 2;
        T res = 0;
        if (c[0]) res += c[0]->query(x1, x2, y1, y2, L, M);
        if (c[1]) res += c[1]->query(x1, x2, y1, y2, M + 1, R);
        return res;
    }
};
```

2.3 cdq.cpp

```
int ans[maxn], lev[maxn];
array<int, 5> v[maxn], tmp[maxn];

struct BIT {
        ...
```

```
} c;

void solve(int l, int r) {
    if (l >= r) return;
    int mid = (l + r) / 2;
    solve(l, mid), solve(mid + 1, r);
    int i = l, j = mid + 1;
    int piv = l;
    while (i <= mid || j <= r) {
        if (i <= mid && (j > r || mp(v[i][1], v[i][2]) <= mp(v[j][1], v[j
            ][2]))) {
            c.modify(v[i][2], v[i][3]);
            tmp[piv++] = v[i++];
        } else {
            v[j][4] += c.query(v[j][2]);
            tmp[piv++] = v[j++];
        }
    }
    rep(i, l, mid) c.modify(v[i][2], -v[i][3]);
    rep(i, l, r) v[i] = tmp[i];
}

void solve() {
    cin >> n >> k;
    c.resize(k);
    rep(i, 1, n) {
        int s, c, m;
        cin >> s >> c >> m;
        v[i] = {s, c, m, 1, 0};
    }
    v[0][0] = -1;
    sort(v + 1, v + n + 1);
    int cnt = 0;
    rep(i, 1, n) {
        if (v[i][0] == v[cnt][0] && v[i][1] == v[cnt][1] && v[i][2] == v[cnt
            ][2]) v[cnt][3]++;
        else v[++cnt] = v[i];
    }
    solve(1, cnt);
    rep(i, 1, cnt) {
        ans[v[i][4] + v[i][3] - 1] += v[i][3];
    }
    rep(i, 0, n - 1) cout << ans[i] << '\n';
}
```

2.4 compact.cpp

```
namespace compact {
const int LOGN=18;
int l[N],r[N],tot,p[N][20],n;
map<int,int> cv;
int lca(int u,int v) {
    if (dep[u]>dep[v]) swap(u,v);
    per(i,LOGN-1,0) if (dep[p[v][i]]>=dep[u]) v=p[v][i];
    if (u==v) return u;
    per(i,LOGN-1,0) if (p[v][i]!=p[u][i]) u=p[u][i],v=p[v][i];
    return p[u][0];
}
void dfs(int u,int f) {
    l[u]=++tot; dep[u]=dep[f]+1; p[u][0]=f;
    vec[dep[u]].pb(u);
    for (auto v:vE[u]) {
        if (v==f) continue;
        dfs(v,u);
    }
    r[u]=tot;
}
void build(int _n) {
    n=_n; tot=0;
    dfs(1,0);
    rep(j,1,LOGN-1) rep(i,1,n) p[i][j]=p[p[i][j-1]][j-1];
}

bool cmp(int u,int v) { return l[u]<l[v]; }
vector<PII> compact(VI v) {
    int m=SZ(v);
    vector<PII> E;
    sort(all(v),cmp);
    rep(i,0,m-2) {
        int w=lca(v[i],v[i+1]);
        v.pb(w);
    }
    v.pb(0);
    v.pb(1);
    sort(all(v),cmp);
    v.erase(unique(all(v)),v.end());
```

```
    cv.clear();
    per(i,SZ(v)-1,1) {
        int u=v[i];
        while (1) {
            auto it=cv.lower_bound(l[u]);
            if (it==cv.end()||it->fi>r[u]) break;
            E.pb(mp(u,v[it->se]));
            cv.erase(it);
        }
        cv[l[u]]=i;
    }
    return E;
}
};
```

2.5 dominator.cpp

```
void solve(int u, int S) {
  int best = -1, cnt = S + 1;
  auto find_best = [&](auto &find_best, int u, int par) -> void {
    sz[u] = 1, sdom[u] = 0;
    for (auto v : e[u]) {
      if (v == par || del[v]) continue;
      find_best(find_best, v, u);
      sz[u] += sz[v];
      sdom[u] = max(sdom[u], sz[v]);
    }
    sdom[u] = max(sdom[u], S - sz[u]);
    if (sdom[u] < cnt) {
      cnt = sdom[u], best = u;
    }
  };
  find_best(find_best, u, 0);
  int id1 = tot++, dep1 = 0;
  int id2, dep2;
  auto dfs = [&](auto &dfs, int u, int par, int dep) -> void {
    dep1 = max(dep1, dep);
    dep2 = max(dep2, dep);
    Q[u].pb({id1, 1, dep});
    Q[u].pb({id2, -1, dep});
    for (auto v : e[u]) {
      if (v == par || del[v]) continue;
      dfs(dfs, v, u, dep + 1);
    }
  };
  Q[best].pb({id1, 1, 0});
  for (auto v : e[best]) {
    if (del[v]) continue;
    id2 = tot++, dep2 = 0;
    dfs(dfs, v, best, 1);
    fenw[id2] = BIT<ll>(dep2 + 1);
  }
  fenw[id1] = BIT<ll>(dep1 + 1);
  del[best] = 1;
  for (auto v : e[best]) {
    if (!del[v]) solve(v, sz[v]);
  }
}
```

2.6 dsu.cpp

```
class dsu {
 public:
  vector<int> fa;
  vector<ll> dist;
  int n;

  dsu(int _n) : n(_n) {
    fa.resize(n);
    dist.assign(n, 0);
    iota(fa.begin(), fa.end(), 0);
  }

  int find(int x) {
    if (fa[x] == x) return x;
    int par = fa[x];
    fa[x] = find(fa[x]);
    dist[x] += dist[par];
    return fa[x];
  }

  void unite(int x, int y, ll v) {
    int px = find(x);
    int py = find(y);
    fa[py] = px;
    dist[py] = dist[x] - dist[y] - v;
```

```
  }
};
```

2.7 dsu-on-tree.cpp

```
void dfs(int x, int fa) {
    hs[x] = -1, w[x] = 1;
    l[x] = ++tot;
    id[tot] = x;
    for (auto y : g[x]) if (y != fa) {
            dfs(y, x);
            w[x] += w[y];
            if (hs[x] == -1 || w[y] > w[hs[x]])
                hs[x] = y;
        }
    r[x] = tot;
}

void dsu(int x, int fa, int keep) {
    for (auto y : g[x]) {
        if (y != hs[x] && y != fa) {
            dsu(y, x, 0);
        }
    }
    if (hs[x] != -1) dsu(hs[x], x, 1);

    for (auto y : g[x]) {
        if (y != hs[x] && y != fa) {
            for (int i = l[y]; i <= r[y]; i++) {

            }
        }
    }
    // add current node

    ans[x] = cnt;

    if (!keep) {
        // clear
    }
}
```

2.8 fenwick.cpp

```
template <typename T>
struct BIT {
  vector<T> fenw;
  int n, pw;

  BIT(int n_ = 0) : n(n_) {
    fenw.assign(n + 1, 0);
    pw = bit_floor(unsigned(n));
  }

  void Modify(int x, T v) {
    if (x <= 0) return;  // assert(0 <= x && x < n);
    while (x <= n) {     // x < n
      fenw[x] += v;
      x += (x & -x);  // x |= x + 1;
    }
  }

  T Query(int x) {
    // assert(0 <= x && x <= n);
    T v{};
    while (x > 0) {
      v += fenw[x];   // fenw[x - 1];
      x -= (x & -x);  // x &= x - 1;
    }
    return v;
  }

  // Returns the length of the longest prefix with sum <= c
  int MaxPrefix(T c) {
    T v{};
    int at = 0;
    for (int i = 20; i >= 0; i--) {
      if (at + bit(i) <= n && v + fenw[at + bit(i)] <= c) {
        v += fenw[at + bit(i)];
        at += bit(i);
      }
    }
    /**
     * for (int len = pw; len > 0; len >>= 1) {
     *   if (at + len <= n) {
     *     auto nv = v;
```

```
      *     nv += fenw[at + len - 1];
      *     if (!(c < nv)) {
      *       v = nv;
      *       at += len;
      *     }
      *   }
      * }
      * assert(0 <= at && at <= n);
      */
    return at;
  }
};
```

2.9 fenwick2d.cpp

```
template <typename T>
class fenwick2d {
 public:
  vector<vector<T>> fenw;
  int n, m;

  fenwick2d(int _n, int _m) : n(_n), m(_m) {
    fenw.resize(n);
    for (int i = 0; i < n; i++) {
      fenw[i].resize(m);
    }
  }

  inline void modify(int i, int j, T v) {
    int x = i;
    while (x < n) {
      int y = j;
      while (y < m) {
        fenw[x][y] += v;
        y |= (y + 1);
      }
      x |= (x + 1);
    }
  }

  inline T get(int i, int j) {
    T v{};
    int x = i;
    while (x >= 0) {
      int y = j;
      while (y >= 0) {
        v += fenw[x][y];
        y = (y & (y + 1)) - 1;
      }
      x = (x & (x + 1)) - 1;
    }
    return v;
  }
};

struct node {
  int a = ...;  // don't forget to set default value

  inline void operator+=(node &other) { ... }
};
```

2.10 hash-table.cpp

```
struct Hash_table {
    static const int V = 1000003;
    int fst[V], nxt[V];
    int ctm, ptm[V], T;
    int val[V];
    ll key[V];
    void init() {T = 0, ctm++;}
    void insert(ll k, int v) {
        int s = k % V;
        if (ptm[s] != ctm) ptm[s] = ctm, fst[s] = -1;
        for (int i = fst[s]; i != -1; i = nxt[i]) if (key[i] == k) {
                return;
            }
        nxt[T] = fst[s], fst[s] = T, key[T] = k, val[T] = v;
        T++;
    }
    int query(ll k) {
        int s = k % V;
        if (ptm[s] != ctm) return -1;
        for (int i = fst[s]; i != -1; i = nxt[i]) {
            if (key[i] == k) return val[i];
        }
        return -1;
```

```
    }
};
```

2.11 hilbertOrder.cpp

```
template <class M, bool val_on_edge = false>
class Mo_Tree {
private:
    static inline int64_t hilbertOrder(int x, int y, int pow, int rotate) {
        if (pow == 0)
            return 0;
        int hpow = 1 << (pow - 1);
        int seg = (x < hpow) ? ((y < hpow) ? 0 : 3) : ((y < hpow) ? 1 : 2);
        seg = (seg + rotate) & 3;
        const int rotateDelta[4] = {3, 0, 0, 1};
        int nx = x & (x ^ hpow), ny = y & (y ^ hpow);
        int nrot = (rotate + rotateDelta[seg]) & 3;
        int64_t subSquareSize = int64_t(1) << (2 * pow - 2);
        int64_t ordd = seg * subSquareSize;
        int64_t add = hilbertOrder(nx, ny, pow - 1, nrot);
        ordd += (seg == 1 || seg == 2) ? add : (subSquareSize - add - 1);
        return ordd;
    }

    struct query {
        int l, r, q_indx, lca;
        int64_t ord;
    };

    int q;
    vector<int> depth, flat, st, en, node_freq;
    vector<vector<int>> anc;
    vector<M> par_v;
    vector<query> Q;

    void add_node(int node) {
        if (par_v[node] < N) {
            aux[par_v[node]]++;
            w.reset(par_v[node]);
        }
    }

    void remove_node(int node) {
        if (par_v[node] < N) {
            if (--aux[par_v[node]] == 0) {
                w.set(par_v[node]);
            }
        }
    }

    long long calc(...) {
        return w._Find_first();
    }

    void add(int ind) {
        node_freq[flat[ind]]++;
        if (node_freq[flat[ind]] == 1) {
            add_node(flat[ind]);
        } else {
            remove_node(flat[ind]);
        }
    }

    void remove(int ind) {
        node_freq[flat[ind]]--;
        if (node_freq[flat[ind]] == 1) {
            add_node(flat[ind]);
        } else {
            remove_node(flat[ind]);
        }
    }

public:
    Mo_Tree(vector<vector<int>> &g, const vector<M> &v) {
        q = 0;
        this->par_v = v;
        int n = (int)g.size() - 1;
        depth.resize(n + 1), st.resize(n + 1);
        en.resize(n + 1), node_freq.resize(n + 1);
        anc.assign(n + 1, vector<int>(__lg(n) + 2, 0));

        auto dfs = [&](auto &dfs, int node, int p, int d) -> void {
            anc[node][0] = p;
            depth[node] = d;
            st[node] = flat.size();
            flat.push_back(node);
```

```
            for (auto it : g[node]) {
                if (it != p) {
                    dfs(dfs, it, node, d + 1);
                }
            }
            en[node] = flat.size();
            flat.push_back(node);
        };
        dfs(dfs, 1, 0, 0);

        for (int j = 1; j <= __lg(n); ++j) {
            for (int i = 1; i <= n; ++i) {
                anc[i][j] = anc[anc[i][j - 1]][j - 1];
            }
        }
    }

    int kth_anc(int node, int k) {
        int ret = node;
        for (int bit = (int)anc[ret].size() - 1; ~bit; --bit) {
            if (k & (1 << bit)) {
                ret = anc[ret][bit];
            }
        }
        return ret;
    }

    int LCA(int u, int v) {
        if (depth[u] < depth[v]) {
            swap(u, v);
        }
        u = kth_anc(u, depth[u] - depth[v]);
        if (u == v) {
            return u;
        }
        for (int bit = anc[0].size() - 1; ~bit; --bit) {
            if (anc[u][bit] != anc[v][bit]) {
                u = anc[u][bit];
                v = anc[v][bit];
            }
        }
        return anc[u][0];
    }


    template <class... T>
    void add_query(int u, int v, T &...x) {
        if (st[u] > st[v]) {
            swap(u, v);
        }
        int lca = LCA(u, v), l, r, q_lca = -1;
        if (lca == u || lca == v) {
            l = st[u] + val_on_edge, r = st[v];
        } else {
            l = en[u], r = st[v], q_lca = lca;
        }
        Q.push_back({l, r, q++, q_lca, hilbertOrder(l, r, __lg(flat.size())
            + 1, 0), x...});
    }

    vector<int> Mo() {
        vector<int> ans(q);
        sort(Q.begin(), Q.end(), [&](const query &a, const query &b) {
            return a.ord < b.ord;
        });
        int l = 0, r = -1;
        for (auto [L, R, q_indx, lca, ord] : Q) {
            while (l > L)
                add(--l);
            while (r < R)
                add(++r);
            while (r > R)
                remove(r--);
            while (l < L)
                remove(l++);

            if (~lca && !val_on_edge)
                add_node(lca);
            ans[q_indx] = calc();
            if (~lca && !val_on_edge)
                remove_node(lca);
        }
        return ans;
    }
};
```

2.12 HLD.cpp

```cpp
struct HLD {
    int n;
    std::vector<int> siz, top, dep, parent, in, out, seq;
    std::vector<std::vector<int>> adj;
    int cur;

    HLD() {}
    HLD(int n) {
        init(n);
    }
    void init(int n) {
        this->n = n;
        siz.resize(n);
        top.resize(n);
        dep.resize(n);
        parent.resize(n);
        in.resize(n);
        out.resize(n);
        seq.resize(n);
        cur = 0;
        adj.assign(n, {});
    }
    void addEdge(int u, int v) {
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    void work(int root = 0) {
        top[root] = root;
        dep[root] = 0;
        parent[root] = -1;
        dfs1(root);
        dfs2(root);
    }
    void dfs1(int u) {
        if (parent[u] != -1) {
            adj[u].erase(std::find(adj[u].begin(), adj[u].end(), parent[u]))
                ;
        }

        siz[u] = 1;
        for (auto &v : adj[u]) {
            parent[v] = u;
            dep[v] = dep[u] + 1;
            dfs1(v);
            siz[u] += siz[v];
            if (siz[v] > siz[adj[u][0]]) {
                std::swap(v, adj[u][0]);
            }
        }
    }
    void dfs2(int u) {
        in[u] = cur++;
        seq[in[u]] = u;
        for (auto v : adj[u]) {
            top[v] = v == adj[u][0] ? top[u] : v;
            dfs2(v);
        }
        out[u] = cur;
    }
    int lca(int u, int v) {
        while (top[u] != top[v]) {
            if (dep[top[u]] > dep[top[v]]) {
                u = parent[top[u]];
            } else {
                v = parent[top[v]];
            }
        }
        return dep[u] < dep[v] ? u : v;
    }

    int dist(int u, int v) {
        return dep[u] + dep[v] - 2 * dep[lca(u, v)];
    }

    int jump(int u, int k) {
        if (dep[u] < k) {
            return -1;
        }

        int d = dep[u] - k;

        while (dep[top[u]] > d) {
            u = parent[top[u]];
        }

```

```
        return seq[in[u] - dep[u] + d];
    }

    bool isAncester(int u, int v) {
        return in[u] <= in[v] && in[v] < out[u];
    }

    int rootedParent(int u, int v) {
        std::swap(u, v);
        if (u == v) {
            return u;
        }
        if (!isAncester(u, v)) {
            return parent[u];
        }
        auto it = std::upper_bound(adj[u].begin(), adj[u].end(), v, [&](int
            x, int y) {
            return in[x] < in[y];
        }) - 1;
        return *it;
    }

    int rootedSize(int u, int v) {
        if (u == v) {
            return n;
        }
        if (!isAncester(v, u)) {
            return siz[v];
        }
        return n - siz[rootedParent(u, v)];
    }

    int rootedLca(int a, int b, int c) {
        return lca(a, b) ^ lca(b, c) ^ lca(c, a);
    }
};
```

2.13 kdtree.cpp

```
namespace kd {
const int K = 2, N = 2.1e5;
template <typename T>
using P = array<T, K>;
template <typename T>
struct node {
  P<T> pt, mx, mn;
  ll val, sum;
  node *l, *r, *p;
  int id;
  node(const P<T> &_pt = P<T>(), ll _val = 0, int _id = 0)
      : pt(_pt), val(_val), sum(_val), id(_id) {
    mx = mn = pt;
    p = l = r = nullptr;
  }
};
node<ll> *ptr[N];
template <typename T>
void pull(node<T> *u) {
  if (not u) return;
  u->sum = u->val;
  rep(i, 0, K - 1) u->mx[i] = u->mn[i] = u->pt[i];
  if (u->l) {
    u->sum += u->l->sum;
    u->l->p = u;
  }
  if (u->r) {
    u->sum += u->r->sum;
    u->r->p = u;
  }
  rep(i, 0, K - 1) {
    if (u->l) {
      u->mx[i] = max(u->mx[i], u->l->mx[i]);
      u->mn[i] = min(u->mn[i], u->l->mn[i]);
    }
    if (u->r) {
      u->mx[i] = max(u->mx[i], u->r->mx[i]);
      u->mn[i] = min(u->mn[i], u->r->mn[i]);
    }
  }
}

template <typename T>
node<T> *build(vector<node<T>> &a, int l, int r, int d = 0) {
  if (d == K) d = 0;
  if (l >= r) {
    return nullptr;
```

```
  } else {
    int md = (l + r) >> 1;
    nth_element(a.begin() + l, a.begin() + md, a.begin() + r,
                [&](node<T> &x, node<T> &y) { return x.pt[d] < y.pt[d]; });
    node<T> *p = new node<T>(a[md]);
    ptr[p->id] = p;
    p->l = build(a, l, md, d + 1);
    p->r = build(a, md + 1, r, d + 1);
    pull(p);
    return p;
  }
}

template <typename T>
node<T> *search(node<T> *u, P<T> p, int d = 0) {
  if (d == K) d = 0;
  if (not u) return nullptr;
  if (u->pt == p) return u;
  if (p[d] < u->pt[d]) {
    return search(u->l, p, d + 1);
  } else if (p[d] > u->pt[d]) {
    return search(u->r, p, d + 1);
  } else {
    auto tmp = search(u->l, p, d + 1);
    if (tmp) return tmp;
    return search(u->r, p, d + 1);
  }
}

template <typename T>
void modify(node<T> *u, ll v) {
  if (not u) return;
  u->val = v;
  for (auto cur = u; cur; cur = cur->p) {
    pull(cur);
  }
}

template <typename T>
bool inside(node<T> *nd, P<T> p, ll c) {
  int cc = 0;
  if (nd->mx[0] * p[0] + nd->mx[1] * p[1] >= c) cc++;
  if (nd->mn[0] * p[0] + nd->mn[1] * p[1] >= c) cc++;
  if (nd->mx[0] * p[0] + nd->mn[1] * p[1] >= c) cc++;
  if (nd->mn[0] * p[0] + nd->mx[1] * p[1] >= c) cc++;
  return cc == 0;
}

template <typename T>
bool outside(node<T> *nd, P<T> p, ll c) {
  int cc = 0;
  if (nd->mx[0] * p[0] + nd->mx[1] * p[1] >= c) cc++;
  if (nd->mn[0] * p[0] + nd->mn[1] * p[1] >= c) cc++;
  if (nd->mx[0] * p[0] + nd->mn[1] * p[1] >= c) cc++;
  if (nd->mn[0] * p[0] + nd->mx[1] * p[1] >= c) cc++;
  return cc == 4;
}

template <typename T>
ll query(node<T> *u, P<T> p, ll c) {
  if (inside(u, p, c)) return u->sum;
  if (outside(u, p, c)) return 0;
  ll s = 0;
  if (u->pt[0] * p[0] + u->pt[1] * p[1] < c) {
    s += u->val;
  }
  if (u->l) s += query(u->l, p, c);
  if (u->r) s += query(u->r, p, c);
  return s;
}

template <typename T>
T eval_min(node<T> *nd,
           P<T> p) {  // 通过估价函数进行启发式搜索，根据当前结果对搜索剪枝
  if (not nd) return numeric_limits<T>::max() / 4;
  ll s = 0;
  rep(i, 0, K - 1) {
    if (p[i] <= nd->mn[i]) s += nd->mn[i] - p[i];
    if (p[i] >= nd->mx[i]) s += p[i] - nd->mx[i];
  }
  return s;
}

template <typename T>
ll mindist(node<T> *u, P<T> p) {
  ll s = numeric_limits<T>::max() / 4;
```

```
134   if (u->pt != p) {
135     s = min(s, abs(u->pt[0] - p[0]) + abs(u->pt[1] - p[1]));
136   }
137   ll best1 = eval_min(u->l, p), best2 = eval_min(u->r, p);
138   if (best1 < best2) {
139     if (u->l) s = min(s, mindist(u->l, p));
140     if (u->r and best2 < s) s = min(s, mindist(u->r, p));
141     return s;
142   } else {
143     if (u->r) s = min(s, mindist(u->r, p));
144     if (u->l and best1 < s) s = min(s, mindist(u->l, p));
145     return s;
146   }
147 }
148
149 template <typename T>
150 T eval_max(node<T> *nd,
151            P<T> p) {  // 通过估价函数进行启发式搜索, 根据当前结果对搜索剪枝
152   if (not nd) return 0;
153   ll s = 0;
154   rep(i, 0, K - 1) s += max(abs(nd->mx[i] - p[i]), abs(nd->mn[i] - p[i]));
155   return s;
156 }
157
158 template <typename T>
159 ll maxdist(node<T> *u, P<T> p) {
160   ll s = 0;
161   if (u->pt != p) {
162     s = max(s, abs(u->pt[0] - p[0]) + abs(u->pt[1] - p[1]));
163   }
164   ll best1 = eval_max(u->l, p), best2 = eval_max(u->r, p);
165   if (best1 > best2) {
166     if (u->l) s = max(s, maxdist(u->l, p));
167     if (u->r and best2 > s) s = max(s, maxdist(u->r, p));
168     return s;
169   } else {
170     if (u->r) s = max(s, maxdist(u->r, p));
171     if (u->l and best1 > s) s = max(s, maxdist(u->l, p));
172     return s;
173   }
174 }
175 }  // namespace kd
```

2.14 LCT.cpp

```
1  namespace linkCutTree {
2  struct node {
3      node *child[2], *parent, *max;
4      int id;
5      ll sum, val, sz, weight, rev;
6      node(ll val, ll weight, int id) : child {nullptr, nullptr}, parent(
           nullptr), max(this), sum(val), val(val), sz(weight), weight(weight),
            id(id), rev(false) {}
7  };
8
9  bool isRoot(node *p) { return p->parent == nullptr || (p->parent->child[0]
       != p && p->parent->child[1] != p); }
10
11 int side(node *p) { return p->parent->child[1] == p; }
12
13 ll sum(node *p) { return p == nullptr ? 0 : p->sum; }
14
15 ll sz(node *p) { return p == nullptr ? 0 : p->sz; }
16
17 node *max(node *p) { return p == nullptr ? nullptr : p->max; }
18
19 node *max(node *p, node *q) {
20     if (p == nullptr)
21         return q;
22     if (q == nullptr)
23         return p;
24     return p->weight > q->weight ? p : q;
25 }
26
27 void reverse(node *p) {
28     if (p == nullptr)
29         return;
30     swap(p->child[0], p->child[1]);
31     p->rev ^= 1;
32 }
33
34 void push(node *p) {
35     if (p->rev == 0)
36         return;
37     p->rev = 0;
38     reverse(p->child[0]);
39     reverse(p->child[1]);
```

```
}

void pull(node *p) {
    p->sum = sum(p->child[0]) + sum(p->child[1]) + p->val;
    p->max = max(max(max(p->child[0]), max(p->child[1])), p);
    p->sz = p->weight + sz(p->child[0]) + sz(p->child[1]);
}

void connect(node *p, node *q, int side) {
    q->child[side] = p;
    if (p != nullptr)
        p->parent = q;
}

void rotate(node *p) {
    auto q = p->parent;
    int dir = side(p) ^ 1;
    connect(p->child[dir], q, dir ^ 1);
    if (!isRoot(q))
        connect(p, q->parent, side(q));
    else
        p->parent = q->parent;
    connect(q, p, dir);
    pull(q);
}

void splay(node *p) {
    vector<node *> stk;
    for (auto i = p; !isRoot(i); i = i->parent)
        stk.push_back(i->parent);
    while (!stk.empty()) {
        push(stk.back());
        stk.pop_back();
    }
    push(p);
    while (!isRoot(p)) {
        auto q = p->parent;
        if (!isRoot(q))
            rotate(side(p) == side(q) ? q : p);
        rotate(p);
    }
    pull(p);
}

node *access(node *p) {
    node *j = nullptr;
    for (node *i = p; i != nullptr; j = i, i = i->parent) {
        splay(i);
        i->val -= sum(j);
        i->val += sum(i->child[1]);
        i->child[1] = j;
        pull(i);
    }
    splay(p);
    return j;
}

void makeRoot(node *p) {
    access(p);
    reverse(p);
}

void link(node *p, node *q) {
    makeRoot(p);
    access(q);
    p->parent = q;
    q->val += sum(p);
}

void cut(node *p, node *q) {
    makeRoot(p);
    access(q);
    p->parent = q->child[0] = nullptr;
}

node *pathMax(node *p, node *q) {
    makeRoot(p);
    access(q);
    return max(q);
}

ll pathSize(node *p, node *q) {
    makeRoot(p);
    access(q);
    return sz(q);
}
```

```
ll rootedSum(node *p) {
    makeRoot(p);
    return sum(p);
}

ll getSubtree(node *rt, node *v) {
    makeRoot(rt);
    access(v);
    return v->val;
}

bool connected(node *p, node *q) {
    access(p);
    access(q);
    return p->parent != nullptr;
}

void fix(node *p, ll v) {
    access(p);
    push(p);
    // modify ...
    p->val += v;
    pull(p);
}

node *lca(node *z,node *x,node *y) {
    makeRoot(z);
    access(x);
    return access(y);
}
}  // namespace linkCutTree
using namespace linkCutTree;
```

2.15 lichao-tree.cpp

```
struct Line {
  i64 k, b;
  i64 operator()(i64 x) const { return k * x + b; }
};
template <i64 L, i64 R>
struct Segments {
  struct Node {
    optional<Line> s;
    Node *l, *r;
  };
  Node *root;
  Segments() : root(nullptr) {}
  void add(i64 l, i64 r, i64 k, i64 b) {
    auto rec = [&](auto &rec, Node *&p, i64 tl, i64 tr, Line s) -> void {
      if (p == nullptr) p = new Node();
      i64 tm = midpoint(tl, tr);
      if (tl >= l and tr <= r) {
        if (not p->s) return p->s = s, void();
        auto t = p->s.value();
        if (t(tl) >= s(tl)) {
          if (t(tr) >= s(tr)) return;
          if (t(tm) >= s(tm)) return rec(rec, p->r, tm + 1, tr, s);
          return p->s = s, rec(rec, p->l, tl, tm, t);
        }
        if (t(tr) <= s(tr)) return p->s = s, void();
        if (t(tm) <= s(tm)) return p->s = s, rec(rec, p->r, tm + 1, tr, t);
        return rec(rec, p->l, tl, tm, s);
      }
      if (l <= tm) rec(rec, p->l, tl, tm, s);
      if (r > tm) rec(rec, p->r, tm + 1, tr, s);
    };
    rec(rec, root, L, R, {k, b});
  }
  optional<i64> get(i64 x) {
    optional<i64> res = {};
    auto rec = [&](auto &rec, Node *p, i64 tl, i64 tr) -> void {
      if (p == nullptr) return;
      i64 tm = midpoint(tl, tr);
      if (p->s) {
        i64 y = p->s.value()(x);
        if (not res or res.value() < y) res = y;
      }
      if (x <= tm)
        rec(rec, p->l, tl, tm);
      else
        rec(rec, p->r, tm + 1, tr);
    };
    rec(rec, root, L, R);
    return res;
  }
```

```
};
```

2.16 Mo.cpp

```
/**
 * #define K(x) pii(x.first/blk, x.second ^ -(x.first/blk & 1))
 *      iota(all(s), 0);
 *      sort(all(s), [&](int s, int t){ return K(Q[s]) < K(Q[t]); });
 */

VI Mo(const vector<array<int, 3>> &Q) {
    const int blk = 350;
    vector<int> s(SZ(Q)), res = s;
    iota(all(s), 0);
    sort(all(s), [&](int i, int j) {
        int u = Q[i][0] / blk, v = Q[j][0] / blk;
        return u == v ? u % 2 ? Q[i][1] > Q[j][1] : Q[i][1] < Q[j][1] : u <
            v;
    });
    int L = 1, R = 0;
    for (int qi : s) {
        while (R < Q[qi][1]) R++, add(R);
        while (L > Q[qi][0]) L--, add(L);
        while (R > Q[qi][1]) del(R), R--;
        while (L < Q[qi][0]) del(L), L++;
        res[qi] = calc(Q[qi][2]);
    }
    return res;
}
```

2.17 moTree.cpp

```
/**
 * Author: Simon Lindholm
 * Date: 2019-12-28
 * License: CC0
 * Source: https://github.com/hoke-t/tamu-kactl/blob/master/content/data-
     structures/MoQueries.h
 * Description: Answer interval or tree path queries by finding an
     approximate TSP through the queries,
 * and moving from one query to the next by adding/removing points at the
     ends.
 * If values are on tree edges, change \texttt{step} to add/remove the edge
     $(a, c)$ and remove the initial \texttt{add} call (but keep \texttt{in
     }).
 * Time: O(N \sqrt Q)
 * Status: stress-tested
 */

void add(int ind, int end) { ... } // add a [ ind ] (end = 0 or 1)
void del(int ind, int end) { ... } // remove a [ ind ]
int calc() { ... } // compute current answer
vi mo(vector<pii> Q) {
  int L = 0, R = 0, blk = 350; //  N/sqrt (Q)
  vi s(sz(Q)), res = s;
#define K(x) pii(x.first/blk, x.second ^ -(x.first/blk & 1))
  iota(all(s), 0);
  sort(all(s), [&](int s, int t) { return K(Q[s]) < K(Q[t]); });
  for (int qi : s) {
    pii q = Q[qi];
    while (L > q.first) add(--L, 0);
    while (R < q.second) add(R++, 1);
    while (L < q.first) del(L++, 0);
    while (R > q.second) del(--R, 1);
    res[qi] = calc();
  }
  return res;
}
vi moTree(vector<array<int, 2>> Q, vector<vi>& ed, int root = 0) {
  int N = sz(ed), pos[2] = {}, blk = 350; //  N/sqrt (Q)
  vi s(sz(Q)), res = s, I(N), L(N), R(N), in(N), par(N);
  add(0, 0), in[0] = 1;
  auto dfs = [&](int x, int p, int dep, auto & f) -> void {
    par[x] = p;
    L[x] = N;
    if (dep) I[x] = N++;
    for (int y : ed[x]) if (y != p) f(y, x, !dep, f);
    if (!dep) I[x] = N++;
    R[x] = N;
  };
  dfs(root, -1, 0, dfs);
#define K(x) pii(I[x[0]] / blk, I[x[1]] ^ -(I[x[0]] / blk & 1))
  iota(all(s), 0);
  sort(all(s), [&](int s, int t) { return K(Q[s]) < K(Q[t]); });
  for (int qi : s) rep(end, 0, 2) {
```

```
    int &a = pos[end], b = Q[qi][end], i = 0;
#define step(c) { if (in[c]) { del(a, end); in[a] = 0; } \
else { add(c, end); in[c] = 1; } a = c; }
    while (!(L[b] <= L[a] && R[a] <= R[b]))
      I[i++] = b, b = par[b];
    while (a != b) step(par[a]);
    while (i--) step(I[i]);
    if (end) res[qi] = calc();
  }
  return res;
}
```

2.18 MSTMo.cpp

```
#include <bits/stdc++.h>
using namespace std;
#define rep(i,a,n) for (int i=a;i<n;i++)
#define per(i,a,n) for (int i=n-1;i>=a;i--)
#define pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define fi first
#define se second
#define SZ(x) ((int)(x).size())
typedef vector<int> VI;
typedef long long ll;
typedef pair<int,int> PII;
typedef double db;
mt19937 mrand(random_device{}());
const ll mod=1000000007;
int rnd(int x) { return mrand() % x;}
ll powmod(ll a,ll b) {ll res=1;a%=mod; assert(b>=0); for(;b;b>>=1){if(b&1)
    res=res*a%mod;a=a*a%mod;}return res;}
ll gcd(ll a,ll b) { return b?gcd(b,a%b):a;}
// head

const int N=1010000;
int a[N];
namespace Mo {
  int Q,l[N],r[N],f[N],l0,r0,ans[N],n;
  VI ne[N];
  struct point {
    int x, y, o;
    point(int a, int b, int c): x(a), y(b), o(c) {}
  };
  inline bool operator<(const point &a, const point &b) {
    if (a.x != b.x) return a.x > b.x;
    else return a.y < b.y;
  }
  vector<point> p;
  struct edge {
    int s, t, d;
    edge(const point &a, const point &b): s(a.o), t(b.o),
      d(abs(a.x - b.x) + abs(a.y - b.y)) {}
  };
  inline bool operator<(const edge &a, const edge &b) {return a.d < b.d;}
  vector<edge> e;
  int g[N],z[N];
  int cc,cnt[101000];
  void addedge() {
    sort(all(p));
      memset(g,0,sizeof(g));
      z[0]=N;
    rep(i,0,SZ(p)) z[i+1]=p[i].x-p[i].y;
    rep(i,0,SZ(p)) {
          int k = 0, t = p[i].x + p[i].y;
          for (int j = t; j; j -= j & -j)
              if (z[g[j]] < z[k]) k = g[j];
          if (k) e.pb(edge(p[i], p[k - 1]));
          k = z[i + 1];
          for (int j = t; j <N; j += j & -j)
              if (k < z[g[j]]) g[j] = i + 1;
      }
  }
  void updata(int i, bool j,bool k=0) {
    // j=1 insert  j=0 delete
    // k=0 left k=1 right
    if (j==1) {
      cnt[a[i]]++;
      if (cnt[a[i]]%2==0) cc++;
    } else {
      if (cnt[a[i]]%2==0) cc--;
      cnt[a[i]]--;
    }
  }
  void init(int l,int r) {
```

```
    for (int i=l;i<=r;i++) {
      cnt[a[i]]++;
      if (cnt[a[i]]%2==0) cc++;
    }
  }
  inline int query() {
    return cc;
  }
  int find(int x) { if (f[x] != x) f[x] = find(f[x]); return f[x];}
  void dfs(int i,int p) {
    int l1 = l[i], r1 = r[i];
    per(j,l1,l0) updata(j,1,0);
    rep(j,r0+1,r1+1) updata(j,1,1);
    rep(j,l0,l1) updata(j,0,0);
    per(j,r1+1,r0+1) updata(j,0,1);
    ans[i]=query();l0=l1;r0=r1;
    rep(j,0,SZ(ne[i])) if (ne[i][j]!=p) dfs(ne[i][j],i);
  }
  void solve() {
    p.clear();e.clear();
    rep(i,1,Q+1) ans[i]=0;
    rep(i,1,Q+1) p.pb(point(l[i],r[i],i));
    addedge();
    rep(i,0,SZ(p)) p[i].y =n-p[i].y+1;
    addedge();
    rep(i,0,SZ(p)) {
      int j =n-p[i].x+1;
      p[i].x = p[i].y; p[i].y = j;
    }
    addedge();
    rep(i,0,SZ(p)) p[i].x=n-p[i].x+1;
    addedge();
    sort(all(e));
    rep(i,1,Q+1) ne[i].clear(),f[i]=i;
    rep(i,0,SZ(e)) {
      int j=e[i].s,k=e[i].t;
      if (find(j)!=find(k)) f[f[j]]=f[k],ne[j].pb(k),ne[k].pb(j);
    }
    l0=l[1];r0=r[1];
    init(l0,r0);
    dfs(1,0);
  }
}
```

```

int main() {
  scanf("%d",&Mo::n);
  for (int i=1;i<=Mo::n;i++) scanf("%d",a+i);
  scanf("%d",&Mo::Q);
  rep(i,1,Mo::Q+1) scanf("%d%d",&Mo::l[i],&Mo::r[i]);
  Mo::solve();
  rep(i,1,Mo::Q+1) printf("%d\n",Mo::ans[i]);
}
```

2.19 psegt.cpp

```
struct node {
  node *l, *r;
  ll val, sz, add;
};

void pull(node *u) {
  u->sz = 0, u->val = 0;
  if (u->l) u->sz += u->l->sz, u->val += u->l->val;
  if (u->r) u->sz += u->r->sz, u->val += u->r->val;
}

void push(node *u) {
  if (u->add) {
    if (u->l) {
      node *p = new node();
      *p = *u->l;
      u->l = p;
      p->add += u->add;
      p->val += p->sz * u->add;
    }
    if (u->r) {
      node *p = new node();
      *p = *u->r;
      u->r = p;
      p->add += u->add;
      p->val += p->sz * u->add;
    }
    u->add = 0;
  }
}

```

```
node *build(int l, int r) {
  node *p = new node();
  p->add = 0;
  if (l == r) {
    p->l = p->r = nullptr;
    p->val = a[l];
    p->sz = 1;
  } else {
    int mid = (l + r) >> 1;
    p->l = build(l, mid);
    p->r = build(mid + 1, r);
    pull(p);
  }
  return p;
}

ll query(node *v, int l, int r, int ql, int qr) {
  if (ql == l && qr == r) {
    return v->val;
  } else {
    push(v);
    int mid = (l + r) >> 1;
    if (qr <= mid)
      return query(v->l, l, mid, ql, qr);
    else if (ql > mid)
      return query(v->r, mid + 1, r, ql, qr);
    else
      return query(v->l, l, mid, ql, mid) +
             query(v->r, mid + 1, r, mid + 1, qr);
  }
}

node *modify(node *v, int l, int r, int ql, int qr, ll x) {
  if (ql == l && qr == r) {
    node *p = new node();
    *p = *v;
    p->add += x;
    p->val += p->sz * x;
    return p;
  } else {
    push(v);
    int mid = (l + r) >> 1;
    node *p = new node();
    *p = *v;
    if (qr <= mid)
      p->l = modify(v->l, l, mid, ql, qr, x);
    else if (ql > mid)
      p->r = modify(v->r, mid + 1, r, ql, qr, x);
    else
      p->l = modify(v->l, l, mid, ql, mid, x),
      p->r = modify(v->r, mid + 1, r, mid + 1, qr, x);
    pull(p);
    return p;
  }
}
```

2.20 rollbackMo.cpp

```
VI rollbackMo(const vector<array<int, 3>> &Q) {
    const int blk = 350;
    vector<VI> s(SZ(Q));
    vector<int> BF, res(SZ(Q));
    for (int i = 0; i < SZ(Q); i++) {
        int u = Q[i][0] / blk, v = Q[i][1] / blk;
        if (u == v) BF.push_back(i);
        else s[u].push_back(i);
    }
    for (int i = 0; i < SZ(Q); i++)
        sort(all(s[i]), [&](int i, int j) { return Q[i][1] < Q[j][1]; });
    for (int qi : BF) {
        for (int i = Q[qi][0]; i <= Q[qi][1]; i++)
            add(i);
        res[qi] = calc(Q[qi][2]);
        for (int i = Q[qi][0]; i <= Q[qi][1]; i++)
            del(i);
    }
    for (const auto &v : s) {
        if (v.empty()) continue;
        int next_blk = (Q[v.back()][0] / blk + 1) * blk;
        int L = next_blk, R = next_blk - 1;
        for (int qi : v) {
            while (R < Q[qi][1]) R++, add(R);
            while (L > Q[qi][0]) L--, add(L);
            res[qi] = calc(Q[qi][2]);
            while (L < next_blk) del(L), L++;
        }
```

```
        for (int i = next_blk; i <= R; i++)
            del(i);
    }
    return res;
}
```

2.21 segtree.cpp

```
struct info {
    ll sum;
    int sz;
    friend info operator+(const info &a, const info &b) {
        return {(a.sum + b.sum) % mod, a.sz + b.sz};
    }
};

struct tag {
    ll add, mul;
    friend tag operator+(const tag &a, const tag &b) {
        tag res = {(a.add * b.mul + b.add) % mod, a.mul * b.mul % mod};
        return res;
    }
};
info operator+(const info &a, const tag &b) {
    return {(a.sum * b.mul + a.sz * b.add) % mod, a.sz};
}

struct node {
    info val;
    tag t;
} seg[maxn << 2];

void update(int id) {
    seg[id].val = seg[id * 2].val + seg[id * 2 + 1].val;
}
void settag(int id, tag t) {
    seg[id].val = seg[id].val + t;
    seg[id].t = seg[id].t + t;
}
void pushdown(int id) {
    if (seg[id].t.mul == 1 and seg[id].t.add == 0) return;
    settag(id * 2, seg[id].t);
    settag(id * 2 + 1, seg[id].t);
    seg[id].t.mul = 1;
    seg[id].t.add = 0;
}
void build(int l, int r, int id) {
    seg[id].t = {0, 1};
    if (l == r) {
        seg[id].val = {a[l], 1};
    } else {
        int mid = (l + r) >> 1;
        build(l, mid, id * 2);
        build(mid + 1, r, id * 2 + 1);
        update(id);
    }
}
void change(int l, int r, int id, int ql, int qr, tag t) {
    if (l == ql && r == qr) {
        settag(id, t);
    } else {
        int mid = (l + r) >> 1;
        pushdown(id);
        if (qr <= mid) {
            change(l, mid, id * 2, ql, qr, t);
        } else if (ql > mid) {
            change(mid + 1, r, id * 2 + 1, ql, qr, t);
        } else {
            change(l, mid, id * 2, ql, mid, t);
            change(mid + 1, r, id * 2 + 1, mid + 1, qr, t);
        }
        update(id);
    }
}
info query(int l, int r, int id, int ql, int qr) {
    if (l == ql && r == qr) {
        return seg[id].val;
    } else {
        int mid = (l + r) >> 1;
        pushdown(id);
        if (qr <= mid)
            return query(l, mid, id * 2, ql, qr);
        else if (ql > mid)
            return query(mid + 1, r, id * 2 + 1, ql, qr);
        else
            return query(l, mid, id * 2, ql, mid) +
```

```
                    query(mid + 1, r, id * 2 + 1, mid + 1, qr);
    }
}
ll search(int l, int r, int id, int ql, int qr, int d) {
    if (ql == l && qr == r) {
        int mid = (l + r) / 2;
        // if (l != r) pushdown(id); ...
        if (seg[id].val < d)
            return -1;
        else {
            if (l == r)
                return l;
            else if (seg[id * 2].val >= d)
                return search(l, mid, id * 2, ql, mid, d);
            else
                return search(mid + 1, r, id * 2 + 1, mid + 1, qr, d);
        }
    } else {
        int mid = (l + r) >> 1;
        // pushdown(id); ...
        if (qr <= mid)
            return search(l, mid, id * 2, ql, qr, d);
        else if (ql > mid)
            return search(mid + 1, r, id * 2 + 1, ql, qr, d);
        else {
            int tmp = search(l, mid, id * 2, ql, mid, d);
            if (tmp != -1)
                return tmp;
            else
                return search(mid + 1, r, id * 2 + 1, mid + 1, qr, d);
        }
    }
}
```

2.22 segtreefast.cpp

```
/**
 * Author: Lucian Bicsi
 * Description: Very fast and quick segment tree.
 * Only useful for easy invariants. 0-indexed.
 * Range queries are half-open.
 */
#pragma once

struct SegmTree {
  vector<int> T; int n;
  SegmTree(int n) : T(2 * n, (int)2e9), n(n) {}

  void Update(int pos, int val) {
    for (T[pos += n] = val; pos > 1; pos /= 2)
      T[pos / 2] = min(T[pos], T[pos ^ 1]);
  }

  int Query(int b, int e) {
    int res = (int)2e9;
    for (b += n, e += n; b < e; b /= 2, e /= 2) {
      if (b % 2) res = min(res, T[b++]);
      if (e % 2) res = min(res, T[--e]);
    }
    return res;
  }
};
```

2.23 SparseTable.cpp

```
template <typename T, class F = function<T(const T&, const T&)>>
class SparseTable {
public:
  int n;
  vector<vector<T>> mat;
  F func;

  SparseTable(const vector<T>& a, const F& f) : func(f) {
    n = static_cast<int>(a.size());
    int max_log = 32 - __builtin_clz(n);
    mat.resize(max_log);
    mat[0] = a;
    for (int j = 1; j < max_log; j++) {
      mat[j].resize(n - (1 << j) + 1);
      for (int i = 0; i <= n - (1 << j); i++) {
        mat[j][i] = func(mat[j - 1][i], mat[j - 1][i + (1 << (j - 1))]);
      }
    }
  }

  T get(int from, int to) const {
```

```
    assert(0 <= from && from <= to && to <= n - 1);
    int lg = 32 - __builtin_clz(to - from + 1) - 1;
    return func(mat[lg][from], mat[lg][to - (1 << lg) + 1]);
  }
};
```

2.24 SparseTable2D.cpp

```
// lg[1] = 0;
// rep(i, 2, N - 1) {
//     lg[i] = lg[i / 2] + 1;
// }
// int k = log2(r - l + 1); very slow!!!
// int k = __lg(r - l + 1);
// int k = lg[r - l + 1];
// int k = 32 - __builtin_clz(r - l + 1) - 1;
vector<vector<int>> sparse[12];

int query(int x, int y, int d) {
  int k = __lg(d);
  int s = d - bit(k);
  return min({sparse[k][x][y], sparse[k][x + s][y], sparse[k][x][y + s],
      sparse[k][x + s][y + s]});
}

void build() {
  rep(i, 1, n) rep(j, 1, m) sparse[0][i][j] = mat[i][j];
  rep(k, 1, 11) rep(i, 1, n) rep(j, 1, m) {
    int d = bit(k - 1);
    if (i + d > n || j + d > m) continue;
    sparse[k][i][j] = min({sparse[k - 1][i][j], sparse[k - 1][i + d][j],
        sparse[k - 1][i][j + d], sparse[k - 1][i + d][j + d]});
  }
}
```

2.25 treap.cpp

```
/**
 *    author:  tourist
 *    created: 07.10.2022 20:32:03
**/
#include <bits/stdc++.h>

using namespace std;

#ifdef LOCAL
#include "algo/debug.h"
#else
#define debug(...) 42
#endif

mt19937_64 rng(chrono::steady_clock::now().time_since_epoch().count());

class node {
public:
  int id;
  node* l;
  node* r;
  node* p;
  bool rev;
  int sz;
  // declare extra variables:
  long long P;
  long long add;
  long long x;

  node(int _id, long long _x) {
    id = _id;
    l = r = p = nullptr;
    rev = false;
    sz = 1;
    // init extra variables:
    P = rng();
    add = 0;
    x = _x;
  }

  // push everything else:
  void push_stuff() {
    if (add != 0) {
      if (l != nullptr) {
        l->unsafe_apply(add);
      }
      if (r != nullptr) {
        r->unsafe_apply(add);
      }
```

```
      add = 0;
    }
  }

  void unsafe_reverse() {
    push_stuff();
    rev ^= 1;
    swap(l, r);
    pull();
  }

  // apply changes:
  void unsafe_apply(long long delta) {
    add += delta;
    x += delta;
  }

  void push() {
    if (rev) {
      if (l != nullptr) {
        l->unsafe_reverse();
      }
      if (r != nullptr) {
        r->unsafe_reverse();
      }
      rev = 0;
    }
    push_stuff();
  }

  void pull() {
    sz = 1;
    if (l != nullptr) {
      l->p = this;
      sz += l->sz;
    }
    if (r != nullptr) {
      r->p = this;
      sz += r->sz;
    }
  }
};

void debug_node(node* v, string pref = "") {
#ifdef LOCAL
  if (v != nullptr) {
    debug_node(v->r, pref + "␣");
    cerr << pref << "-" << "␣" << v->id << '\n';
    debug_node(v->l, pref + "␣");
  } else {
    cerr << pref << "-" << "␣" << "nullptr" << '\n';
  }
#endif
}

namespace treap {

pair<node*, int> find(node* v, const function<int(node*)> &go_to) {
  // go_to returns: 0 -- found; -1 -- go left; 1 -- go right
  // find returns the last vertex on the descent and its go_to
  if (v == nullptr) {
    return {nullptr, 0};
  }
  int dir;
  while (true) {
    v->push();
    dir = go_to(v);
    if (dir == 0) {
      break;
    }
    node* u = (dir == -1 ? v->l : v->r);
    if (u == nullptr) {
      break;
    }
    v = u;
  }
  return {v, dir};
}

node* get_leftmost(node* v) {
  return find(v, [&](node*) { return -1; }).first;
}

node* get_rightmost(node* v) {
  return find(v, [&](node*) { return 1; }).first;
}
```

```
136
137 node* get_kth(node* v, int k) { // 0-indexed
138   pair<node*, int> p = find(v, [&](node * u) {
139     if (u->l != nullptr) {
140       if (u->l->sz > k) {
141         return -1;
142       }
143       k -= u->l->sz;
144     }
145     if (k == 0) {
146       return 0;
147     }
148     k--;
149     return 1;
150   });
151   return (p.second == 0 ? p.first : nullptr);
152 }
153
154 int get_pos(node* v) { // 0-indexed
155   int k = (v->l != nullptr ? v->l->sz : 0);
156   while (v->p != nullptr) {
157     if (v == v->p->r) {
158       k++;
159       if (v->p->l != nullptr) {
160         k += v->p->l->sz;
161       }
162     }
163     v = v->p;
164   }
165   return k;
166 }
167
168 node* get_root(node* v) {
169   while (v->p != nullptr) {
170     v = v->p;
171   }
172   return v;
173 }
174
175 pair<node*, node*> split(node* v, const function<bool(node*)> &is_right) {
176   if (v == nullptr) {
177     return {nullptr, nullptr};
178   }
179   v->push();
180   if (is_right(v)) {
181     pair<node*, node*> p = split(v->l, is_right);
182     if (p.first != nullptr) {
183       p.first->p = nullptr;
184     }
185     v->l = p.second;
186     v->pull();
187     return {p.first, v};
188   } else {
189     pair<node*, node*> p = split(v->r, is_right);
190     v->r = p.first;
191     if (p.second != nullptr) {
192       p.second->p = nullptr;
193     }
194     v->pull();
195     return {v, p.second};
196   }
197 }
198
199 pair<node*, node*> split_cnt(node* v, int k) {
200   if (v == nullptr) {
201     return {nullptr, nullptr};
202   }
203   v->push();
204   int left_and_me = (v->l != nullptr ? v->l->sz : 0) + 1;
205   if (k < left_and_me) {
206     pair<node*, node*> p = split_cnt(v->l, k);
207     if (p.first != nullptr) {
208       p.first->p = nullptr;
209     }
210     v->l = p.second;
211     v->pull();
212     return {p.first, v};
213   } else {
214     pair<node*, node*> p = split_cnt(v->r, k - left_and_me);
215     v->r = p.first;
216     if (p.second != nullptr) {
217       p.second->p = nullptr;
218     }
219     v->pull();
220     return {v, p.second};
221   }
```

```
}

node* merge(node* v, node* u) {
  if (v == nullptr) {
    return u;
  }
  if (u == nullptr) {
    return v;
  }
  if (v->P > u->P) {
//    if (rng() % (v->sz + u->sz) < (unsigned int) v->sz) {
    v->push();
    v->r = merge(v->r, u);
    v->pull();
    return v;
  } else {
    u->push();
    u->l = merge(v, u->l);
    u->pull();
    return u;
  }
}

int count_left(node* v, const function<bool(node*)> &is_right) {
  if (v == nullptr) {
    return 0;
  }
  v->push();
  if (is_right(v)) {
    return count_left(v->l, is_right);
  }
  return (v->l != nullptr ? v->l->sz : 0) + 1 + count_left(v->r, is_right);
}

int count_less(node* v, long long val) {
  int res = 0;
  while (v != nullptr) {
    v->push();
    if (v->x >= val) {
      v = v->l;
    } else {
      res += (v->l != nullptr ? v->l->sz : 0) + 1;
      v = v->r;
    }
  }
  return res;
}

node* add(node* r, node* v, const function<bool(node*)> &go_left) {
  pair<node*, node*> p = split(r, go_left);
  return merge(p.first, merge(v, p.second));
}

node* remove(node* v) { // returns the new root
  v->push();
  node* x = v->l;
  node* y = v->r;
  node* p = v->p;
  v->l = v->r = v->p = nullptr;
  v->push();
  v->pull(); // now v might be reusable...
  node* z = merge(x, y);
  if (p == nullptr) {
    if (z != nullptr) {
      z->p = nullptr;
    }
    return z;
  }
  if (p->l == v) {
    p->l = z;
  }
  if (p->r == v) {
    p->r = z;
  }
  while (true) {
    p->push();
    p->pull();
    if (p->p == nullptr) {
      break;
    }
    p = p->p;
  }
  return p;
}

node* next(node* v) {
```

```
    if (v->r == nullptr) {
      while (v->p != nullptr && v->p->r == v) {
        v = v->p;
      }
      return v->p;
    }
    v->push();
    v = v->r;
    while (v->l != nullptr) {
      v->push();
      v = v->l;
    }
    return v;
}

node* prev(node* v) {
    if (v->l == nullptr) {
      while (v->p != nullptr && v->p->l == v) {
        v = v->p;
      }
      return v->p;
    }
    v->push();
    v = v->l;
    while (v->r != nullptr) {
      v->push();
      v = v->r;
    }
    return v;
}

int get_size(node* v) {
    return (v != nullptr ? v->sz : 0);
}

template<typename... T>
void Apply(node* v, T... args) {
    v->unsafe_apply(args...);
}

void reverse(node* v) {
    v->unsafe_reverse();
}


// extra of mine
long long lower(node* u, long long x) {
    if (u == nullptr)
      return numeric_limits<long long>::min();
    else if (x <= u->x)
      return lower(u->l, x);
    else
      return max(u->x, lower(u->r, x));
}

long long upper(node* u, long long x) {
    if (u == nullptr)
      return numeric_limits<long long>::max();
    else if (u->x <= x)
      return upper(u->r, x);
    else
      return min(u->x, upper(u->l, x));
}

}  // namespace treap

using namespace treap;

int n;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    node* root = nullptr;
    cin >> n;
    for (int i = 1; i <= n; i++) {
      int op;
      long long x;
      cin >> op >> x;
      switch (op) {
        case 1: {
          root = add(root, new node(x, x), [&](node * u) {
            return x < u->x;
          });
          break;
        }
        case 2: {
```

```
        auto [pt, w] = find(root, [&](node * u) {
          if (x < u->x) return -1;
          else if (x == u->x) return 0;
          else return 1;
        });
        assert(w == 0);
        root = remove(pt);
        break;
      }
      case 3: {
        cout << count_less(root, x) + 1 << '\n';
        break;
      }
      case 4: {
        cout << get_kth(root, x - 1)->x << '\n';
        break;
      }
      case 5: {
        cout << lower(root, x) << '\n';
        break;
      }
      case 6: {
        cout << upper(root, x) << '\n';
        break;
      }
    }
  }
}
```

2.26 UnionFindRollback.cpp

```
vector<int> fa(n);
iota(all(fa), 0);
vector<int> sz(n, 1);
vector<pair<int, int>> ops;

auto Get = [&](int i) {
    while (i != fa[i]) {
        i = fa[i];
    }
    return i;
};
auto Unite = [&](int i, int j) {
    i = Get(i), j = Get(j);
    if (i == j) {
        return;
    }
    if (sz[i] > sz[j]) {
        swap(i, j);
    }
    ops.emplace_back(i, fa[i]);
    fa[i] = j;
    ops.emplace_back(~j, sz[j]);
    sz[j] += sz[i];
};
auto RollBack = [&](int T) {
    while (SZ(ops) > T) {
        auto [i, j] = ops.back();
        ops.pop_back();
        if (i >= 0) {
            fa[i] = j;
        } else {
            sz[~i] = j;
        }
    }
};

ll ans = 0;
auto Dfs = [&](auto &&Dfs, int l, int r) -> void {
    if (l == r) {
        for (auto [x, y] : g[l]) {
            x = Get(x);
            y = Get(y);
            ans += 1ll * sz[x] * sz[y];
        }
    } else {
        int mid = midpoint(l, r);
        {
            int save = SZ(ops);
            for (int i = mid + 1; i <= r; i++) {
                for (auto [x, y] : g[i]) {
                    Unite(x, y);
                }
            }
            Dfs(Dfs, l, mid);
            RollBack(save);
```

```
        }
        {
            int save = SZ(ops);
            for (int i = l; i <= mid; i++) {
                for (auto [x, y] : g[i]) {
                    Unite(x, y);
                }
            }
            Dfs(Dfs, mid + 1, r);
            RollBack(save);
        }
    }
};
Dfs(Dfs, 0, n - 1);
```

2.27 树哈希.cpp

```
basic_string<int> e[maxn];
ull hashv[maxn];
ull seed1, seed2, seed3, seed4;

ull f(ull x) { return x * x * x * seed1 + x * seed2; }
ull h(ull x) { return f(x) ^ ((x & seed3) >> 31) ^ ((x & seed4) << 31); }

void dfs1(int u, int fa) {
    hashv[u] = 1;
    for (auto v : e[u]) if (v != fa) {
            dfs1(v, u);
            hashv[u] += h(hashv[v]);
        }
}

void dfs2(int u, int fa, ull fv) {
// for each root
    hashv[u] += fv;
    for (auto v : e[u]) if (v != fa) {
            dfs2(v, u, h(hashv[u] - h(hashv[v])));
        }
}

void solve() {
    seed1 = rng(), seed2 = rng();
    seed3 = rng(), seed4 = rng();
    cin >> n;
    rep(i, 2, n) {
        int u, v;
        cin >> u >> v;
        e[u].pb(v);
        e[v].pb(u);
    }
    dfs1(1, 0);
    sort(hashv + 1, hashv + n + 1);
    n = unique(hashv + 1, hashv + n + 1) - hashv - 1;
    cout << n << '\n';
}
```

2.28 树链剖分 segtree.cpp

```
int n, m, a[N];
vector<int> e[N];
int l[N], r[N], idx[N];
int sz[N], hs[N], tot, top[N], dep[N], fa[N];

struct info {
    int maxv, sum;
};

info operator + (const info &l, const info &r) {
    return (info){max(l.maxv, r.maxv), l.sum + r.sum};
}

struct node {
    info val;
} seg[N * 4];

// [l, r]

void update(int id) {
    seg[id].val = seg[id * 2].val + seg[id * 2 + 1].val;
}

void build(int id, int l, int r) {
    if (l == r) {
        // l号点， DFS序中第l个点
        seg[id].val = {a[idx[l]], a[idx[l]]};
    } else {
```

```
        int mid = (l + r) / 2;
        build(id * 2, l, mid);
        build(id * 2 + 1, mid + 1, r);
        update(id);
    }
}

void change(int id, int l, int r, int pos, int val) {
    if (l == r) {
        seg[id].val = {val, val};
    } else {
        int mid = (l + r) / 2;
        if (pos <= mid) change(id * 2, l, mid, pos, val);
        else change(id * 2 + 1, mid + 1, r, pos, val);
        update(id);
    }
}

info query(int id, int l, int r, int ql, int qr) {
    if (l == ql && r == qr) return seg[id].val;
    int mid = (l + r) / 2;
    if (qr <= mid) return query(id * 2, l, mid, ql, qr);
    else if (ql > mid) return query(id * 2 + 1, mid + 1, r, ql,qr);
    else {
        return query(id * 2, l, mid, ql, mid) +
            query(id * 2 + 1, mid + 1, r, mid + 1, qr);
    }
}

// 第一遍DFS，子树大小，重儿子，父亲，深度
void dfs1(int u,int f) {
    sz[u] = 1;
    hs[u] = -1;
    fa[u] = f;
    dep[u] = dep[f] + 1;
    for (auto v : e[u]) {
        if (v == f) continue;
        dfs1(v, u);
        sz[u] += sz[v];
        if (hs[u] == -1 || sz[v] > sz[hs[u]])
            hs[u] = v;
    }
}

// 第二遍DFS，每个点DFS序，重链上的链头的元素。
void dfs2(int u, int t) {
    top[u] = t;
    l[u] = ++tot;
    idx[tot] = u;
    if (hs[u] != -1) {
        dfs2(hs[u], t);
    }
    for (auto v : e[u]) {
        if (v != fa[u] && v != hs[u]) {
            dfs2(v, v);
        }
    }
    r[u] = tot;
}

int LCA(int u, int v) {
    while (top[u] != top[v]) {
        if (dep[top[u]] < dep[top[v]]) v = fa[top[v]];
        else u = fa[top[u]];
    }
    if (dep[u] < dep[v]) return u;
    else return v;
}

info query(int u,int v) {
    info ans{(int)-1e9, 0};
    while (top[u] != top[v]) {
        if (dep[top[u]] < dep[top[v]]) {
            ans = ans + query(1, 1, n, l[top[v]], l[v]);
            v = fa[top[v]];
        } else {
            ans = ans + query(1, 1, n, l[top[u]], l[u]);
            u = fa[top[u]];
        }
    }
    if (dep[u] <= dep[v]) ans = ans + query(1, 1, n, l[u], l[v]);
    else ans = ans + query(1, 1, n, l[v], l[u]);
    return ans;
}
```

2.29 笛卡尔树.cpp

```
int a[maxn], l[maxn], r[maxn], root;
int ans[maxn], tot;

void build() {
    stack<int> stk;
    for (int i = 1; i <= n; i++) {
        int last = 0;
        while (!stk.empty() && a[stk.top()] > a[i]) {
            last = stk.top();
            stk.pop();
        }
        if (stk.empty())
            root = i;
        else
            r[stk.top()] = i;
        l[i] = last;
        stk.push(i);
    }
}

void dfs(int c, int L, int R) {
    ans[c] = ++tot;
    if (l[c]) dfs(l[c], L, c - 1);
    if (r[c]) dfs(r[c], c + 1, R);
}
```

2.30 线段树合并.cpp

```
struct node {
  int sz, sum;
  node *l, *r;
  node() : sz(0), sum(0), l(nullptr), r(nullptr) {}
} pool[N * 20], *cur = pool;

node *newnode() {
  return cur++;
}

void upd(node *rt) {
  if (not rt) return;
  rt->sum = rt->sz > 0;
  if (rt->l) rt->sum += rt->l->sum;
  if (rt->r) rt->sum += rt->r->sum;
}

node *modify(node *rt, int l, int r, int pos, int d) {
  if (not rt) rt = newnode();
  if (l == r) {
    rt->sz += d;
    upd(rt);
    return rt;
  } else {
    int md = (l + r) >> 1;
    if (pos <= md)
      rt->l = modify(rt->l, l, md, pos, d);
    else
      rt->r = modify(rt->r, md + 1, r, pos, d);
    upd(rt);
    return rt;
  }
}

node *merge(node *u, node *v, int l, int r) {
  if (not u) return v;
  if (not v) return u;
  if (l == r) {
    u->sz += v->sz;
    upd(u);
    return u;
  } else {
    int md = (l + r) >> 1;
    u->l = merge(u->l, v->l, l, md);
    u->r = merge(u->r, v->r, md + 1, r);
    upd(u);
    return u;
  }
}

ll query(node *rt, int l, int r) {
  if (not rt) return 0;
  return rt->sum;
}

pair<node *, node *> split(node *rt, int l, int r, int ql, int qr) {
  if (not rt) return {nullptr, nullptr};
```

```
  if (ql == l && qr == r) {
    return {nullptr, rt};
  } else {
    int md = (l + r) >> 1;
    if (qr <= md) {
      auto [p1, p2] = split(rt->l, l, md, ql, qr);
      rt->l = p1;
      upd(rt);
      if (not p2) return {rt, nullptr};
      node *u = newnode();
      u->l = p2;
      upd(u);
      return {rt, u};
    } else if (ql > md) {
      auto [p1, p2] = split(rt->r, md + 1, r, ql, qr);
      rt->r = p1;
      upd(rt);
      if (not p2) return {rt, nullptr};
      node *u = newnode();
      u->r = p2;
      upd(u);
      return {rt, u};
    } else {
      auto [p1, p2] = split(rt->l, l, md, ql, md);
      auto [p3, p4] = split(rt->r, md + 1, r, md + 1, qr);
      rt->l = p1, rt->r = p3;
      upd(rt);
      if (not p2 and not p4) return {rt, nullptr};
      node *u = newnode();
      u->l = p2, u->r = p4;
      upd(u);
      return {rt, u};
    }
  }
}
```

3 DP

3.1 Convex hull optimization.cpp

```
array<ll, 3> a[maxn];
int q[maxn];
ll ans[maxn];

ll X(int p) {
    return 2ll * a[p][0];
}
ll Y(int p) {
    return a[p][0] * a[p][0] + a[p][1];
}
ldb slope(int x, int y) {
    return (ldb)(Y(y) - Y(x)) / (X(y) - X(x));
}
void solve() {
    cin >> n;
    int head = 1, rear = 0;
    rep(i, 1, n) {
        cin >> a[i][0] >> a[i][1];
        a[i][2] = i;
    }
    sort(a + 1, a + n + 1);

    rep(i, 1, n) {
        while (head < rear && slope(q[rear], i) <= slope(q[rear], q[rear -
            1])) rear--;
        q[++rear] = i;
    }
    rep(i, 1, n) {
        ll k = -a[i][0];
        while (head < rear && slope(q[head], q[head + 1]) <= k) head++;
        ans[a[i][2]] = (a[i][0] + a[q[head]][0]) * (a[i][0] + a[q[head]][0])
             + a[i][1] + a[q[head]][1];
    }
    rep(i, 1, n) cout << ans[i] << '\n';
}
```

3.2 DivideAndConquerDP.cpp

```
ll w[N][N],sum[N][N],opt[N],dp[805][N];

ll calc(int i,int j) { return sum[j][j]-sum[j][i]-sum[i][j]+sum[i][i]; }

void rec(int d,int l,int r,int optl,int optr) {
    if (l>r) return;
    int md=(l+r)>>1;
```

```
    rep(i,optl,optr) if (dp[d-1][i]+calc(i,md)<dp[d][md]) {
        dp[d][md]=dp[d-1][i]+calc(i,md);
        opt[md]=i;
    }
    rec(d,l,md-1,optl,opt[md]);
    rec(d,md+1,r,opt[md],optr);
}
```

3.3 有依赖决策单调.cpp

```
pair<int, int> stk[N];
auto calc = [&](int i, int j) { ... } // dp[j] -> dp[i]
int h = 0, t = 0;
stk[t++] = {1, 0}; // {left, opt}

for (int i = 1; i <= n; i++) {
    if (h < t && stk[h].first < i) stk[h].first++;
    if (h + 1 < t && stk[h].first >= stk[h + 1].first) ++h;
    dp[i] = calc(i, stk[h].second);
    while (h < t && calc(stk[t - 1].first, stk[t - 1].second) >= calc(stk[t
        - 1].first, i))
        --t;
    if (h < t) {
        int l = stk[t - 1].first, r = n + 1;
        while (l + 1 < r) {
            int md = (l + r) >> 1;
            if (calc(md, stk[t - 1].second) < calc(md, i)) l = md; else r =
                md;
        }
        if (r <= n) stk[t++] = {r, i};
    } else stk[t++] = {i, i};
}
```

4 Geometry

4.1 1 (1).cpp

```
typedef double db;
const db EPS = 1e-9;

inline int sign(db a) { return a < -EPS ? -1 : a > EPS; }

inline int cmp(db a, db b) { return sign(a - b); }

struct P {
    db x, y;
    P() {}
    P(db _x, db _y) : x(_x), y(_y) {}
    P operator+(P p) { return {x + p.x, y + p.y}; }
    P operator-(P p) { return {x - p.x, y - p.y}; }
    P operator*(db d) { return {x * d, y * d}; }
    P operator/(db d) { return {x / d, y / d}; }

    bool operator<(P p) const {
        int c = cmp(x, p.x);
        if (c) return c == -1;
        return cmp(y, p.y) == -1;
    }

    bool operator==(P o) const {
        return cmp(x, o.x) == 0 && cmp(y, o.y) == 0;
    }

    db dot(P p) { return x * p.x + y * p.y; }
    db det(P p) { return x * p.y - y * p.x; }

    db distTo(P p) { return (*this - p).abs(); }
    db alpha() { return atan2(y, x); }
    void read() { cin >> x >> y; }
    void write() {cout << "(" << x << "," << y << ")" << endl;}
    db abs() { return sqrt(abs2());}
    db abs2() { return x * x + y * y; }
    P rot90() { return P(-y, x);}
    P unit() { return *this / abs(); }
    int quad() const { return sign(y) == 1 || (sign(y) == 0 && sign(x) >= 0)
        ; }
    P rot(db an) { return {x * cos(an) - y * sin(an), x * sin(an) + y * cos(
        an)}; }
};

#define cross(p1,p2,p3) ((p2.x-p1.x)*(p3.y-p1.y)-(p3.x-p1.x)*(p2.y-p1.y))
#define crossOp(p1,p2,p3) sign(cross(p1,p2,p3))

// 直线 p1p2, q1q2 是否恰有一个交点
bool chkLL(P p1, P p2, P q1, P q2) {
```

```
    db a1 = cross(q1, q2, p1), a2 = -cross(q1, q2, p2);
    return sign(a1 + a2) != 0;
}

// 求直线 p1p2, q1q2 的交点
P isLL(P p1, P p2, P q1, P q2) {
    db a1 = cross(q1, q2, p1), a2 = -cross(q1, q2, p2);
    return (p1 * a2 + p2 * a1) / (a1 + a2);
}

// 判断区间 [l1, r1], [l2, r2] 是否相交
bool intersect(db l1, db r1, db l2, db r2) {
    if (l1 > r1) swap(l1, r1); if (l2 > r2) swap(l2, r2);
    return !( cmp(r1, l2) == -1 || cmp(r2, l1) == -1 );
}

// 线段 p1p2, q1q2 相交
bool isSS(P p1, P p2, P q1, P q2) {
    return intersect(p1.x, p2.x, q1.x, q2.x) && intersect(p1.y, p2.y, q1.y, q2.y) &&
           crossOp(p1, p2, q1) * crossOp(p1, p2, q2) <= 0 && crossOp(q1, q2, p1)
           * crossOp(q1, q2, p2) <= 0;
}

// 线段 p1p2, q1q2 严格相交
bool isSS_strict(P p1, P p2, P q1, P q2) {
    return crossOp(p1, p2, q1) * crossOp(p1, p2, q2) < 0 && crossOp(q1, q2, p1)
           * crossOp(q1, q2, p2) < 0;
}

// m 在 a 和 b 之间
bool isMiddle(db a, db m, db b) {
    /*if (a > b) swap(a, b);
    return cmp(a, m) <= 0 && cmp(m, b) <= 0;*/
    return sign(a - m) == 0 || sign(b - m) == 0 || (a < m != b < m);
}

bool isMiddle(P a, P m, P b) {
    return isMiddle(a.x, m.x, b.x) && isMiddle(a.y, m.y, b.y);
}

// 点 p 在线段 p1p2 上
bool onSeg(P p1, P p2, P q) {
    return crossOp(p1, p2, q) == 0 && isMiddle(p1, q, p2);
}
// q1q2 和 p1p2 的交点 在 p1p2 上?

// 点 p 严格在 p1p2 上
bool onSeg_strict(P p1, P p2, P q) {
    return crossOp(p1, p2, q) == 0 && sign((q - p1).dot(p1 - p2)) * sign((q
        - p2).dot(p1 - p2)) < 0;
}

// 求 q 到 直线p1p2 的投影（垂足）    : p1 != p2
P proj(P p1, P p2, P q) {
    P dir = p2 - p1;
    return p1 + dir * (dir.dot(q - p1) / dir.abs2());
}

// 求 q 以 直线p1p2 为轴的反射
P reflect(P p1, P p2, P q) {
    return proj(p1, p2, q) * 2 - q;
}

// 求 q 到 线段p1p2 的最小距离
db nearest(P p1, P p2, P q) {
    if (p1 == p2) return p1.distTo(q);
    P h = proj(p1, p2, q);
    if (isMiddle(p1, h, p2))
        return q.distTo(h);
    return min(p1.distTo(q), p2.distTo(q));
}

// 求 线段p1p2 与 线段q1q2 的距离
db disSS(P p1, P p2, P q1, P q2) {
    if (isSS(p1, p2, q1, q2)) return 0;
    return min(min(nearest(p1, p2, q1), nearest(p1, p2, q2)), min(nearest(q1
        , q2, p1), nearest(q1, q2, p2)));
}

// 极角排序
sort(p, p + n, [&](P a, P b) {
    int qa = a.quad(), qb = b.quad();
    if (qa != qb) return qa < qb;
```

```
    else return sign(a.det(b)) > 0;
});
```

4.2 1 (2).cpp

```
db area(vector<P> ps){
    db ret = 0; rep(i,0,ps.size()) ret += ps[i].det(ps[(i+1)%ps.size()]);
    return ret/2;
}

int contain(vector<P> ps, P p){ //2:inside,1:on_seg,0:outside
    int n = ps.size(), ret = 0;
    rep(i,0,n){
        P u=ps[i],v=ps[(i+1)%n];
        if(onSeg(u,v,p)) return 1;
        if(cmp(u.y,v.y)<=0) swap(u,v);
        if(cmp(p.y,u.y) > 0 || cmp(p.y,v.y) <= 0) continue;
        ret ^= crossOp(p,u,v) > 0;
    }
    return ret*2;
}

vector<P> convexHull(vector<P> ps) {
    int n = ps.size(); if(n <= 1) return ps;
    sort(ps.begin(), ps.end());
    vector<P> qs(n * 2); int k = 0;
    for (int i = 0; i < n; qs[k++] = ps[i++])
        while (k > 1 && crossOp(qs[k - 2], qs[k - 1], ps[i]) <= 0) --k;
    for (int i = n - 2, t = k; i >= 0; qs[k++] = ps[i--])
        while (k > t && crossOp(qs[k - 2], qs[k - 1], ps[i]) <= 0) --k;
    qs.resize(k - 1);
    return qs;
}

vector<P> convexHullNonStrict(vector<P> ps) {
    //caution: need to unique the Ps first
    int n = ps.size(); if(n <= 1) return ps;
    sort(ps.begin(), ps.end());
    vector<P> qs(n * 2); int k = 0;
    for (int i = 0; i < n; qs[k++] = ps[i++])
        while (k > 1 && crossOp(qs[k - 2], qs[k - 1], ps[i]) < 0) --k;
    for (int i = n - 2, t = k; i >= 0; qs[k++] = ps[i--])
        while (k > t && crossOp(qs[k - 2], qs[k - 1], ps[i]) < 0) --k;
    qs.resize(k - 1);
    return qs;
}

db convexDiameter(vector<P> ps){
    int n = ps.size(); if(n <= 1) return 0;
    int is = 0, js = 0; rep(k,1,n) is = ps[k]<ps[is]?k:is, js = ps[js] < ps[
        k]?k:js;
    int i = is, j = js;
    db ret = ps[i].distTo(ps[j]);
    do{
        if((ps[(i+1)%n]-ps[i]).det(ps[(j+1)%n]-ps[j]) >= 0)
            (++j)%=n;
        else
            (++i)%=n;
        ret = max(ret,ps[i].distTo(ps[j]));
    }while(i!=is || j!=js);
    return ret;
}

vector<P> convexCut(const vector<P>&ps, P q1, P q2) {
    vector<P> qs;
    int n = ps.size();
    rep(i,0,n){
        P p1 = ps[i], p2 = ps[(i+1)%n];
        int d1 = crossOp(q1,q2,p1), d2 = crossOp(q1,q2,p2);
        if(d1 >= 0) qs.push_back(p1);
        if(d1 * d2 < 0) qs.push_back(isLL(p1,p2,q1,q2));
    }
    return qs;
}

void reorderPolygon(vector<P> &ps) {
    size_t pos = 0;
    for(size_t i = 1; i < ps.size(); i++){
        if(ps[i].y < ps[pos].y || (ps[i].y == ps[pos].y && ps[i].x < ps[pos
            ].x))
            pos = i;
    }
    rotate(ps.begin(), ps.begin() + pos, ps.end());
}

vector<P> minkowski(vector<P> p, vector<P> q){
```

```
    if(p.empty()) return q;
    // the first vertex must be the lowest
    reorderPolygon(p);
    reorderPolygon(q);
    // must ensure cyclic indexing
    p.push_back(p[0]);
    p.push_back(p[1]);
    q.push_back(q[0]);
    q.push_back(q[1]);
    // main part
    vector<P> result;
    size_t i = 0, j = 0;
    while(i < p.size() - 2 || j < q.size() - 2){
        result.push_back(p[i] + q[j]);
        auto cross = (p[i + 1] - p[i]).det(q[j + 1] - q[j]);
        if(cross >= 0 && i < SZ(p) - 2)
            ++i;
        if(cross <= 0 && j < SZ(q) - 2)
            ++j;
    }
    return result;
}

bool convexContain(const vector<P> &l, P p, bool strict = true) {
    int a = 1, b = l.size() - 1, r = !strict;
    if (l.size() < 3) return r && onSeg(l[0], l.back(), p);
    if (crossOp(l[0], l[a], l[b]) > 0) swap(a, b);
    if (crossOp(l[0], l[a], p) >= r || crossOp(l[0], l[b], p)<= -r)
        return false;
    while (abs(a - b) > 1) {
        int c = (a + b) / 2;
        (crossOp(l[0], l[c], p) > 0 ? b : a) = c;
    }
    return sign(cross(l[a], l[b], p)) < r;
}
```

4.3 1 (3).cpp

```
int type(P o1,db r1,P o2,db r2){
    db d = o1.distTo(o2);
    if(cmp(d,r1+r2) == 1) return 4;
    if(cmp(d,r1+r2) == 0) return 3;
    if(cmp(d,abs(r1-r2)) == 1) return 2;
    if(cmp(d,abs(r1-r2)) == 0) return 1;
    return 0;
}

vector<P> isCL(P o,db r,P p1,P p2){
    if (cmp(abs((o-p1).det(p2-p1)/p1.distTo(p2)),r)>0) return {};
    db x = (p1-o).dot(p2-p1), y = (p2-p1).abs2(), d = x * x - y * ((p1-o).
        abs2() - r*r);
    d = max(d,(db)0.0); P m = p1 - (p2-p1)*(x/y), dr = (p2-p1)*(sqrt(d)/y);
    return {m-dr,m+dr}; //along dir: p1->p2
}

vector<P> isCC(P o1, db r1, P o2, db r2) { //need to check whether two
    circles are the same
    db d = o1.distTo(o2);
    if (cmp(d, r1 + r2) == 1) return {};
    if (cmp(d,abs(r1-r2))==-1) return {};
    d = min(d, r1 + r2);
    db y = (r1 * r1 + d * d - r2 * r2) / (2 * d), x = sqrt(r1 * r1 - y * y);
    P dr = (o2 - o1).unit();
    P q1 = o1 + dr * y, q2 = dr.rot90() * x;
    return {q1-q2,q1+q2};//along circle 1
}

// extanCC, intanCC : -r2, tanCP : r2 = 0
vector<pair<P, P>> tanCC(P o1, db r1, P o2, db r2) {
    P d = o2 - o1;
    db dr = r1 - r2, d2 = d.abs2(), h2 = d2 - dr * dr;
    if (sign(d2) == 0|| sign(h2) < 0) return {};
    h2 = max((db)0.0, h2);
    vector<pair<P, P>> ret;
    for (db sign : {-1, 1}) {
        P v = (d * dr + d.rot90() * sqrt(h2) * sign) / d2;
        ret.push_back({o1 + v * r1, o2 + v * r2});
    }
    if (sign(h2) == 0) ret.pop_back();
    return ret;
}

db rad(P p1,P p2){
    return atan2l(p1.det(p2),p1.dot(p2));
}
```

```
db areaCT(db r, P p1, P p2){
    vector<P> is = isCL(P(0,0),r,p1,p2);
    if(is.empty()) return r*r*rad(p1,p2)/2;
    bool b1 = cmp(p1.abs2(),r*r) == 1, b2 = cmp(p2.abs2(), r*r) == 1;
    if(b1 && b2){
        P md=(is[0]+is[1])/2;
        if(sign((p1-md).dot(p2-md)) <= 0)
            return r*r*(rad(p1,is[0]) + rad(is[1],p2))/2 + is[0].det(is[1])
                /2;
        else return r*r*rad(p1,p2)/2;
    }
    if(b1) return (r*r*rad(p1,is[0]) + is[0].det(p2))/2;
    if(b2) return (p1.det(is[1]) + r*r*rad(is[1],p2))/2;
    return p1.det(p2)/2;
}

P inCenter(P A, P B, P C) {
    double a = (B - C).abs(), b = (C - A).abs(), c = (A - B).abs();
    return (A * a + B * b + C * c) / (a + b + c);
}

P circumCenter(P a, P b, P c) {
    P bb = b - a, cc = c - a;
    double db = bb.abs2(), dc = cc.abs2(), d = 2 * bb.det(cc);
    return a - P(bb.y * dc - cc.y * db, cc.x * db - bb.x * dc) / d;
}

P othroCenter(P a, P b, P c) {
    P ba = b - a, ca = c - a, bc = b - c;
    double Y = ba.y * ca.y * bc.y,
    A = ca.x * ba.y - ba.x * ca.y,
    x0 = (Y + ca.x * ba.y * b.x - ba.x * ca.y * c.x) / A,
    y0 = -ba.x * (x0 - c.x) / ba.y + ca.y;
    return {x0, y0};
}

pair<P,db> min_circle(vector<P> ps){
    random_shuffle(ps.begin(), ps.end());
    int n = ps.size();
    P o = ps[0]; db r = 0;
    rep(i,1,n) if(o.distTo(ps[i]) > r + EPS){
        o = ps[i], r = 0;
        rep(j,0,i) if(o.distTo(ps[j]) > r + EPS){
            o = (ps[i] + ps[j]) / 2; r = o.distTo(ps[i]);
            rep(k,0,j) if(o.distTo(ps[k]) > r + EPS){
                o = circumCenter(ps[i],ps[j],ps[k]);
                r = o.distTo(ps[i]);
            }
        }
    }
    return {o,r};
}
```

4.4 all.cpp

```
typedef double db;
const db EPS = 1e-9;

inline int sign(db a) { return a < -EPS ? -1 : a > EPS; }

inline int cmp(db a, db b){ return sign(a-b); }

struct P {
    db x, y;
    P() {}
    P(db _x, db _y) : x(_x), y(_y) {}
    P operator+(P p) { return {x + p.x, y + p.y}; }
    P operator-(P p) { return {x - p.x, y - p.y}; }
    P operator*(db d) { return {x * d, y * d}; }
    P operator/(db d) { return {x / d, y / d}; }

    bool operator<(P p) const {
        int c = cmp(x, p.x);
        if (c) return c == -1;
        return cmp(y, p.y) == -1;
    }

    bool operator==(P o) const{
        return cmp(x,o.x) == 0 && cmp(y,o.y) == 0;
    }

    db dot(P p) { return x * p.x + y * p.y; }
    db det(P p) { return x * p.y - y * p.x; }

    db distTo(P p) { return (*this-p).abs(); }
    db alpha() { return atan2(y, x); }
```

```
    void read() { cin>>x>>y; }
    void write() {cout<<"("<<x<<","<<y<<")"<<endl;}
    db abs() { return sqrt(abs2());}
    db abs2() { return x * x + y * y; }
    P rot90() { return P(-y,x);}
    P unit() { return *this/abs(); }
    int quad() const { return sign(y) == 1 || (sign(y) == 0 && sign(x) >= 0)
        ; }
    P rot(db an){ return {x*cos(an)-y*sin(an),x*sin(an) + y*cos(an)}; }
};

struct L{ //ps[0] -> ps[1]
    P ps[2];
    P& operator[](int i) { return ps[i]; }
    P dir() { return ps[1] - ps[0]; }
    bool include(P p) { return sign((ps[1] - ps[0]).det(p - ps[0])) > 0; }
    L push(){ // push eps outward
        const double eps = 1e-6;
        P delta = (ps[1] - ps[0]).rot90().unit() * eps;
        return {{ps[0] - delta, ps[1] - delta}};
    }
};

#define cross(p1,p2,p3) ((p2.x-p1.x)*(p3.y-p1.y)-(p3.x-p1.x)*(p2.y-p1.y))
#define crossOp(p1,p2,p3) sign(cross(p1,p2,p3))

bool chkLL(P p1, P p2, P q1, P q2) {
    db a1 = cross(q1, q2, p1), a2 = -cross(q1, q2, p2);
    return sign(a1+a2) != 0;
}

P isLL(P p1, P p2, P q1, P q2) {
    db a1 = cross(q1, q2, p1), a2 = -cross(q1, q2, p2);
    return (p1 * a2 + p2 * a1) / (a1 + a2);
}

P isLL(L l1,L l2){ return isLL(l1[0],l1[1],l2[0],l2[1]); }

bool intersect(db l1,db r1,db l2,db r2){
    if(l1>r1) swap(l1,r1); if(l2>r2) swap(l2,r2);
    return !( cmp(r1,l2) == -1 || cmp(r2,l1) == -1 );
}

bool isSS(P p1, P p2, P q1, P q2){
    return intersect(p1.x,p2.x,q1.x,q2.x) && intersect(p1.y,p2.y,q1.y,q2.y)
        &&
    crossOp(p1,p2,q1) * crossOp(p1,p2,q2) <= 0 && crossOp(q1,q2,p1)
            * crossOp(q1,q2,p2) <= 0;
}

bool isSS_strict(P p1, P p2, P q1, P q2){
    return crossOp(p1,p2,q1) * crossOp(p1,p2,q2) < 0 && crossOp(q1,q2,p1)
            * crossOp(q1,q2,p2) < 0;
}

bool isMiddle(db a, db m, db b) {
    return sign(a - m) == 0 || sign(b - m) == 0 || (a < m != b < m);
}

bool isMiddle(P a, P m, P b) {
    return isMiddle(a.x, m.x, b.x) && isMiddle(a.y, m.y, b.y);
}

bool onSeg(P p1, P p2, P q){
    return crossOp(p1,p2,q) == 0 && isMiddle(p1, q, p2);
}

bool onSeg_strict(P p1, P p2, P q){
    return crossOp(p1,p2,q) == 0 && sign((q-p1).dot(p1-p2)) * sign((q-p2).
        dot(p1-p2)) < 0;
}

P proj(P p1, P p2, P q) {
    P dir = p2 - p1;
    return p1 + dir * (dir.dot(q - p1) / dir.abs2());
}

P reflect(P p1, P p2, P q){
    return proj(p1,p2,q) * 2 - q;
}

db nearest(P p1,P p2,P q){
    P h = proj(p1,p2,q);
    if(isMiddle(p1,h,p2))
        return q.distTo(h);
    return min(p1.distTo(q),p2.distTo(q));
```

```
}

db disSS(P p1, P p2, P q1, P q2){
    if(isSS(p1,p2,q1,q2)) return 0;
    return min(min(nearest(p1,p2,q1),nearest(p1,p2,q2)), min(nearest(q1,q2,
        p1),nearest(q1,q2,p2)));
}

db rad(P p1,P p2){
    return atan2l(p1.det(p2),p1.dot(p2));
}

db incircle(P p1, P p2, P p3){
    db A = p1.distTo(p2);
    db B = p2.distTo(p3);
    db C = p3.distTo(p1);
    return sqrtl(A*B*C/(A+B+C));
}

//polygon

db area(vector<P> ps){
    db ret = 0; rep(i,0,ps.size()) ret += ps[i].det(ps[(i+1)%ps.size()]);
    return ret/2;
}

int contain(vector<P> ps, P p){ //2:inside,1:on_seg,0:outside
    int n = ps.size(), ret = 0;
    rep(i,0,n){
        P u=ps[i],v=ps[(i+1)%n];
        if(onSeg(u,v,p)) return 1;
        if(cmp(u.y,v.y)<=0) swap(u,v);
        if(cmp(p.y,u.y) >0 || cmp(p.y,v.y) <= 0) continue;
        ret ^= crossOp(p,u,v) > 0;
    }
    return ret*2;
}

vector<P> convexHull(vector<P> ps) {
    int n = ps.size(); if(n <= 1) return ps;
    sort(ps.begin(), ps.end());
    vector<P> qs(n * 2); int k = 0;
    for (int i = 0; i < n; qs[k++] = ps[i++])
        while (k > 1 && crossOp(qs[k - 2], qs[k - 1], ps[i]) <= 0) --k;
    for (int i = n - 2, t = k; i >= 0; qs[k++] = ps[i--])
        while (k > t && crossOp(qs[k - 2], qs[k - 1], ps[i]) <= 0) --k;
    qs.resize(k - 1);
    return qs;
}

vector<P> convexHullNonStrict(vector<P> ps) {
    //caution: need to unique the Ps first
    int n = ps.size(); if(n <= 1) return ps;
    sort(ps.begin(), ps.end());
    vector<P> qs(n * 2); int k = 0;
    for (int i = 0; i < n; qs[k++] = ps[i++])
        while (k > 1 && crossOp(qs[k - 2], qs[k - 1], ps[i]) < 0) --k;
    for (int i = n - 2, t = k; i >= 0; qs[k++] = ps[i--])
        while (k > t && crossOp(qs[k - 2], qs[k - 1], ps[i]) < 0) --k;
    qs.resize(k - 1);
    return qs;
}

db convexDiameter(vector<P> ps){
    int n = ps.size(); if(n <= 1) return 0;
    int is = 0, js = 0; rep(k,1,n) is = ps[k]<ps[is]?k:is, js = ps[js] < ps[
        k]?k:js;
    int i = is, j = js;
    db ret = ps[i].distTo(ps[j]);
    do{
        if((ps[(i+1)%n]-ps[i]).det(ps[(j+1)%n]-ps[j]) >= 0)
            (++j)%=n;
        else
            (++i)%=n;
        ret = max(ret,ps[i].distTo(ps[j]));
    }while(i!=is || j!=js);
    return ret;
}

vector<P> convexCut(const vector<P>&ps, P q1, P q2) {
    vector<P> qs;
    int n = ps.size();
    rep(i,0,n){
        P p1 = ps[i], p2 = ps[(i+1)%n];
        int d1 = crossOp(q1,q2,p1), d2 = crossOp(q1,q2,p2);
        if(d1 >= 0) qs.pb(p1);
```

```
        if(d1 * d2 < 0) qs.pb(isLL(p1,p2,q1,q2));
    }
    return qs;
}

//min_dist

db min_dist(vector<P>&ps,int l,int r){
    if(r-l<=5){
        db ret = 1e100;
        rep(i,l,r) rep(j,l,i) ret = min(ret,ps[i].distTo(ps[j]));
        return ret;
    }
    int m = (l+r)>>1;
    db ret = min(min_dist(ps,l,m),min_dist(ps,m,r));
    vector<P> qs; rep(i,l,r) if(abs(ps[i].x-ps[m].x)<= ret) qs.pb(ps[i]);
    sort(qs.begin(), qs.end(),[](P a,P b) -> bool {return a.y<b.y; });
    rep(i,1,qs.size()) for(int j=i-1;j>=0&&qs[j].y>=qs[i].y-ret;--j)
        ret = min(ret,qs[i].distTo(qs[j]));
    return ret;
}

int type(P o1,db r1,P o2,db r2){
    db d = o1.distTo(o2);
    if(cmp(d,r1+r2) == 1) return 4;
    if(cmp(d,r1+r2) == 0) return 3;
    if(cmp(d,abs(r1-r2)) == 1) return 2;
    if(cmp(d,abs(r1-r2)) == 0) return 1;
    return 0;
}

vector<P> isCL(P o,db r,P p1,P p2){
    db x = (p1-o).dot(p2-p1), y = (p2-p1).abs2(), d = x * x - y * ((p1-o).
        abs2() - r*r);
    if(sign(d) < 0) return {};
    d = max(d,0.0); P m = p1 - (p2-p1)*(x/y), dr = (p2-p1)*(sqrt(d)/y);
    return {m-dr,m+dr}; //along dir: p1->p2
}

vector<P> isCC(P o1, db r1, P o2, db r2) { //need to check whether two
    circles are the same
    db d = o1.distTo(o2);
    if (cmp(d, r1 + r2) == 1) return {};
    d = min(d, r1 + r2);
    db y = (r1 * r1 + d * d - r2 * r2) / (2 * d), x = sqrt(r1 * r1 - y * y);
    P dr = (o2 - o1).unit();
    P q1 = o1 + dr * y, q2 = dr.rot90() * x;
    return {q1-q2,q1+q2};//along circle 1
}

vector<P> tanCP(P o, db r, P p) {
    db x = (p - o).abs2(), d = x - r * r;
    if (sign(d) <= 0) return {}; // on circle => no tangent
    P q1 = o + (p - o) * (r * r / x);
    P q2 = (p - o).rot90() * (r * sqrt(d) / x);
    return {q1-q2,q1+q2}; //counter clock-wise
}


vector<L> extanCC(P o1, db r1, P o2, db r2) {
    vector<L> ret;
    if (cmp(r1, r2) == 0) {
        P dr = (o2 - o1).unit().rot90() * r1;
        ret.pb({{o1 + dr, o2 + dr}}), ret.pb({{o1 - dr, o2 - dr}});
    } else {
        P p = (o2 * r1 - o1 * r2) / (r1 - r2);
        vector<P> ps = tanCP(o1, r1, p), qs = tanCP(o2, r2, p);
        rep(i,0,min(ps.size(),qs.size())) ret.pb({{ps[i], qs[i]}}); //c1
            counter-clock wise
    }
    return ret;
}

vector<L> intanCC(P o1, db r1, P o2, db r2) {
    vector<L> ret;
    P p = (o1 * r2 + o2 * r1) / (r1 + r2);
    vector<P> ps = tanCP(o1,r1,p), qs = tanCP(o2,r2,p);
    rep(i,0,min(ps.size(),qs.size())) ret.pb({{ps[i], qs[i]}}); //c1 counter
        -clock wise
    return ret;
}

db areaCT(db r, P p1, P p2){
    vector<P> is = isCL(P(0,0),r,p1,p2);
    if(is.empty()) return r*r*rad(p1,p2)/2;
    bool b1 = cmp(p1.abs2(),r*r) == 1, b2 = cmp(p2.abs2(), r*r) == 1;
```

```
    if(b1 && b2){
        if(sign((p1-is[0]).dot(p2-is[0])) <= 0 &&
            sign((p1-is[0]).dot(p2-is[0])) <= 0)
        return r*r*(rad(p1,is[0]) + rad(is[1],p2))/2 + is[0].det(is[1])/2;
        else return r*r*rad(p1,p2)/2;
    }
    if(b1) return (r*r*rad(p1,is[0]) + is[0].det(p2))/2;
    if(b2) return (p1.det(is[1]) + r*r*rad(is[1],p2))/2;
    return p1.det(p2)/2;
}

bool parallel(L l0, L l1) { return sign( l0.dir().det( l1.dir() ) ) == 0; }

bool sameDir(L l0, L l1) { return parallel(l0, l1) && sign(l0.dir().dot(l1.
    dir()) ) == 1; }

bool cmp (P a,  P b) {
    if (a.quad() != b.quad()) {
        return a.quad() < b.quad();
    } else {
        return sign( a.det(b) ) > 0;
    }
}

bool operator < (L l0, L l1) {
    if (sameDir(l0, l1)) {
        return l1.include(l0[0]);
    } else {
        return cmp( l0.dir(), l1.dir() );
    }
}

bool check(L u, L v, L w) {
    return w.include(isLL(u,v));
}

vector<P> halfPlaneIS(vector<L> &l) {
    sort(l.begin(), l.end());
    deque<L> q;
    for (int i = 0; i < (int)l.size(); ++i) {
        if (i && sameDir(l[i], l[i - 1])) continue;
        while (q.size() > 1 && !check(q[q.size() - 2], q[q.size() - 1], l[i
            ])) q.pop_back();
        while (q.size() > 1 && !check(q[1], q[0], l[i])) q.pop_front();
        q.push_back(l[i]);
    }
    while (q.size() > 2 && !check(q[q.size() - 2], q[q.size() - 1], q[0])) q
        .pop_back();
    while (q.size() > 2 && !check(q[1], q[0], q[q.size() - 1])) q.pop_front
        ();
    vector<P> ret;
    for (int i = 0; i < (int)q.size(); ++i) ret.push_back(isLL(q[i], q[(i +
        1) % q.size()]));
    return ret;
}

P inCenter(P A, P B, P C) {
    double a = (B - C).abs(), b = (C - A).abs(), c = (A - B).abs();
    return (A * a + B * b + C * c) / (a + b + c);
}

P circumCenter(P a, P b, P c) {
    P bb = b - a, cc = c - a;
    double db = bb.abs2(), dc = cc.abs2(), d = 2 * bb.det(cc);
    return a - P(bb.y * dc - cc.y * db, cc.x * db - bb.x * dc) / d;
}

P othroCenter(P a, P b, P c) {
    P ba = b - a, ca = c - a, bc = b - c;
    double Y = ba.y * ca.y * bc.y,
    A = ca.x * ba.y - ba.x * ca.y,
    x0 = (Y + ca.x * ba.y * b.x - ba.x * ca.y * c.x) / A,
    y0 = -ba.x * (x0 - c.x) / ba.y + ca.y;
    return {x0, y0};
}
```

4.5 圆面积并.cpp

```
db intergal(db x,db y,db r,db L,db R){
    return r*r*(R-L) + x*r*(sinl(R) - sinl(L)) + y*r*(-cosl(R) + cosl(L));
}

db calc_area_circle(P c,db r,db L,db R){
    return intergal(c.x,c.y,r,L,R) / 2;
}

```

```
db norm(db x){
    while(x < 0) x += 2 * PI;
    while(x > 2 * PI) x -= 2 * PI;
    return x;
}

P cs[N]; db rs[N];

void work(){
    vector<int> cand = {};
    rep(i,0,m){
        bool ok = 1;
        rep(j,0,m) if(i!=j){
            if(rs[j] > rs[i] + EPS && rs[i] + cs[i].distTo(cs[j]) <= rs[j] +
                 EPS){
                ok = 0; break;
            }
            if(cs[i] == cs[j] && cmp(rs[i],rs[j]) == 0 && j < i){
                ok = 0; break;
            }
        }
        if(ok) cand.pb(i);
    }

    rep(i,0,cand.size()) cs[i] = cs[cand[i]], rs[i] = rs[cand[i]];
    m = cand.size();

    db area = 0;

    //work
    rep(i,0,m){
        vector<pair<db,int>> ev = {{0,0},{2*PI,0}};

        int cur = 0;

        rep(j,0,m) if(j!=i){
            auto ret = isCC(cs[i],rs[i],cs[j],rs[j]);
            if(!ret.empty()){
                db l = (ret[0] - cs[i]).alpha();
                db r = (ret[1] - cs[i]).alpha();
                l = norm(l); r = norm(r);
                ev.pb({l,1});ev.pb({r,-1});
                if(l > r) ++cur;
            }
        }

        sort(ev.begin(), ev.end());
        rep(j,0,ev.size() - 1){
            cur += ev[j].se;
            if(cur == 0){
                area += calc_area_circle(cs[i],rs[i],ev[j].fi,ev[j+1].fi);
            }
        }
    }
}
```

5 Graph

5.1 bellmanford.cpp

```
vector<PII> e[N];

template <typename T>
void add(int u, int v, T w) {
    e[u].eb(v, w);
}

template <typename T>
vector<T> bellmanford(vector<pair<int, T>> *g, int start) {
    // assert(0 <= start && start < g.n);
    // maybe use inf = numeric_limits<T>::max() / 4
    const T inf = numeric_limits<T>::max() / 4;
    vector<T> dist(n, inf);
    dist[start] = 0;
    int cnt = 0;
    while (true) {
        bool upd = 0;
        cnt++;
        for (int i = 0; i < n; i++) {
            for (auto [to, cost] : e[i]) {
                if (dist[to] > dist[i] + cost) {
                    upd = 1;
                    dist[to] = dist[i] + cost;
                }
            }
```

```
        }
        if (!upd || cnt == n) {
            break;
        }
    }
    return dist;
    // returns inf if there's no path
}
```

5.2 BlockCutTree.cpp

```
struct BlockCutTree {
    int n;
    std::vector<std::vector<int>> adj;
    std::vector<int> dfn, low, stk;
    int cnt, cur;
    std::vector<std::pair<int, int>> edges;

    BlockCutTree() {}
    BlockCutTree(int n) {
        init(n);
    }

    void init(int n) {
        this->n = n;
        adj.assign(n, {});
        dfn.assign(n, -1);
        low.resize(n);
        stk.clear();
        cnt = cur = 0;
        edges.clear();
    }

    void addEdge(int u, int v) {
        adj[u].push_back(v);
        adj[v].push_back(u);
    }

    void dfs(int x) {
        stk.push_back(x);
        dfn[x] = low[x] = cur++;

        for (auto y : adj[x]) {
            if (dfn[y] == -1) {
                dfs(y);
                low[x] = std::min(low[x], low[y]);
                if (low[y] == dfn[x]) {
                    int v;
                    do {
                        v = stk.back();
                        stk.pop_back();
                        edges.emplace_back(n + cnt, v);
                    } while (v != y);
                    edges.emplace_back(x, n + cnt);
                    cnt++;
                }
            } else {
                low[x] = std::min(low[x], dfn[y]);
            }
        }
    }

    std::pair<int, std::vector<std::pair<int, int>>> work() {
        for (int i = 0; i < n; i++) {
            if (dfn[i] == -1) {
                stk.clear();
                dfs(i);
            }
        }
        return {cnt, edges};
    }
};
```

5.3 boruvka.cpp

```
/**
 * while component > 1:
 *     for each component:
 *         find select[i]
 *     for each component:
 *         if select[i] != i:
 *             merge(i, select[i])
 *             component--
 */

ll ans = 0, cnt = n;
```

```
while (cnt > 1) {
    fill(select + 1, select + n + 1, -1);
    vector<int> cand;
    for (int i = 1; i <= n; i++) {
        cand.push_back(col[i]);
    }
    ranges::sort(cand);
    cand.erase(unique(all(cand)), cand.end());

    for (auto id : cand) {
        for (auto x : S[id]) remove(x);
        for (auto x : S[id]) {
            auto [opt, w] = get(x);
            if (select[id] == -1 || w < mn[id]) {
                select[id] = opt, mn[id] = w;
            }
        }
        for (auto x : S[id]) insert(x);
    }

    for (int i = 1; i <= n; i++) if (col[i] == i) {
        int j = col[select[i]];
        if (i == j) continue;
        ans += mn[i];
        merge(i, j);
        cnt--;
    }
}
```

5.4 dijfast.cpp

```
vector<PII> e[N];

template <typename T>
void add(int u, int v, T w) {
    e[u].eb(v, w);
}

template <typename T>
vector<T> dijkstra(vector<pair<int, T>> *g, int start) {
    // assert(0 <= start && start < g.n);
    // maybe use inf = numeric_limits<T>::max() / 4
    vector<T> dist(n, numeric_limits<T>::max());
    priority_queue<pair<T, int>, vector<pair<T, int>>, greater<pair<T, int
        >>> s;
    dist[start] = 0;
    s.emplace(dist[start], start);
    while (!s.empty()) {
        T expected = s.top().first;
        int i = s.top().second;
        s.pop();
        if (dist[i] != expected) {
            continue;
        }
        for (auto [to, cost] : g[i]) {
            if (dist[i] + cost < dist[to]) {
                dist[to] = dist[i] + cost;
                s.emplace(dist[to], to);
            }
        }
    }
    return dist;
    // returns numeric_limits<T>::max() if there's no path
}
```

5.5 dijkstra.cpp

```
vector<PII> e[N];

template <typename T>
void add(int u, int v, T w) {
    e[u].eb(v, w);
}

template <typename T>
vector<T> dijkstra(vector<pair<int, T>> *g, int start) {
    // assert(0 <= start && start < g.n);
    // maybe use inf = numeric_limits<T>::max() / 4
    const T inf = numeric_limits<T>::max();
    vector<T> dist(n, inf);
    vector<int> was(n, 0);
    dist[start] = 0;
    while (true) {
        int cur = -1;
        for (int i = 0; i < n; i++) {
            if (was[i] || dist[i] == inf) continue;
```

```
            if (cur == -1 || dist[i] < dist[cur]) {
                cur = i;
            }
        }
        if (cur == -1 || dist[cur] == inf) {
            break;
        }
        was[cur] = 1;
        for (auto [to, cost] : g[cur]) {
            dist[to] = min(dist[to], dist[cur] + cost);
        }
    }
    return dist;
    // returns inf if there's no path
}
```

5.6 dinic.cpp

```
template<typename T>
struct FlowGraph {
    static const int V = 1015;
    static const int E = 100015;
    int s, t, vtot;
    int head[V], etot;
    int dis[V], cur[V];
    struct edge {
        int v, nxt;
        T f;
    } e[E * 2];
    void addedge(int u, int v, T f) {
        e[etot] = {v, head[u], f};
        head[u] = etot++;
        e[etot] = {u, head[v], 0};
        head[v] = etot++;
    }
    bool bfs() {
        for (int i = 1; i <= vtot; i++) {
            dis[i] = 0;
            cur[i] = head[i];
        }
        queue<int> q;
        q.push(s); dis[s] = 1;
        while (!q.empty()) {
            int u = q.front(); q.pop();
            for (int i = head[u]; i != -1; i = e[i].nxt) {
                if (e[i].f && !dis[e[i].v]) {
                    int v = e[i].v;
                    dis[v] = dis[u] + 1;
                    if (v == t) return true;
                    q.push(v);
                }
            }
        }
        return false;
    }
    T dfs(int u, T m) {
        if (u == t) return m;
        T flow = 0;
        for (int i = cur[u]; i != -1; cur[u] = i = e[i].nxt) {
            if (e[i].f && dis[e[i].v] == dis[u] + 1) {
                T f = dfs(e[i].v, min(m, e[i].f));
                e[i].f -= f;
                e[i ^ 1].f += f;
                m -= f;
                flow += f;
                if (!m) break;
            }
        }
        if (!flow) dis[u] = -1;
        return flow;
    }
    T dinic() {
        T flow = 0;
        while (bfs()) flow += dfs(s, numeric_limits<T>::max());
        return flow;
    }
    void init(int _s, int _t, int _vtot) {
        s = _s;
        t = _t;
        vtot = _vtot;
        etot = 0;
        for (int i = 1; i <= vtot; i++) head[i] = -1;
    }
};
```

5.7 dinic-tourist.cpp

```
template <typename T>
class flow_graph {
 public:
  static constexpr T eps = (T)1e-9;

  struct edge {
    int from;
    int to;
    T c;
    T f;
  };

  vector<vector<int>> g;
  vector<edge> edges;
  int n;
  int st;
  int fin;
  T flow;

  flow_graph(int _n, int _st, int _fin) : n(_n), st(_st), fin(_fin) {
    assert(0 <= st && st < n && 0 <= fin && fin < n && st != fin);
    g.resize(n);
    flow = 0;
  }

  void clear_flow() {
    for (const edge &e : edges) {
      e.f = 0;
    }
    flow = 0;
  }

  int add(int from, int to, T forward_cap, T backward_cap) {
    assert(0 <= from && from < n && 0 <= to && to < n);
    int id = (int)edges.size();
    g[from].push_back(id);
    edges.push_back({from, to, forward_cap, 0});
    g[to].push_back(id + 1);
    edges.push_back({to, from, backward_cap, 0});
    return id;
  }
};

template <typename T>
class dinic {
 public:
  flow_graph<T> &g;

  vector<int> ptr;
  vector<int> d;
  vector<int> q;

  dinic(flow_graph<T> &_g) : g(_g) {
    ptr.resize(g.n);
    d.resize(g.n);
    q.resize(g.n);
  }

  bool expath() {
    fill(d.begin(), d.end(), -1);
    q[0] = g.fin;
    d[g.fin] = 0;
    int beg = 0, end = 1;
    while (beg < end) {
      int i = q[beg++];
      for (int id : g.g[i]) {
        const auto &e = g.edges[id];
        const auto &back = g.edges[id ^ 1];
        if (back.c - back.f > g.eps && d[e.to] == -1) {
          d[e.to] = d[i] + 1;
          if (e.to == g.st) {
            return true;
          }
          q[end++] = e.to;
        }
      }
    }
    return false;
  }

  T dfs(int v, T w) {
    if (v == g.fin) {
      return w;
    }
    int &j = ptr[v];
```

```
    while (j >= 0) {
      int id = g.g[v][j];
      const auto &e = g.edges[id];
      if (e.c - e.f > g.eps && d[e.to] == d[v] - 1) {
        T t = dfs(e.to, min(e.c - e.f, w));
        if (t > g.eps) {
          g.edges[id].f += t;
          g.edges[id ^ 1].f -= t;
          return t;
        }
      }
      j--;
    }
    return 0;
  }

  T max_flow() {
    while (expath()) {
      for (int i = 0; i < g.n; i++) {
        ptr[i] = (int)g.g[i].size() - 1;
      }
      T big_add = 0;
      while (true) {
        T add = dfs(g.st, numeric_limits<T>::max());
        if (add <= g.eps) {
          break;
        }
        big_add += add;
      }
      if (big_add <= g.eps) {
        break;
      }
      g.flow += big_add;
    }
    return g.flow;
  }

  vector<bool> min_cut() {
    max_flow();
    vector<bool> ret(g.n);
    for (int i = 0; i < g.n; i++) {
      ret[i] = (d[i] != -1);
    }
    return ret;
  }
};
```

5.8 eulerian-digraph.cpp

```
optional<vector<int>> eulerian_path(int n, const vector<PII> &E) {
  vector<int> res;
  if (E.empty()) return res;
  vector<VI> adj(n + 1);
  vector<int> in(n + 1);
  for (int i = 0; i < ssize(E); i++) {
    auto [u, v] = E[i];
    adj[u].push_back(i);
    in[v] += 1;
  }

  int s = -1, hi = 0, lo = 0;
  for (int i = 1; i <= n; i++) {
    if (SZ(adj[i]) == in[i]) continue;
    if (abs(SZ(adj[i]) - in[i]) > 1) return {};
    if (SZ(adj[i]) > in[i]) {
      hi++, s = i;
    } else {
      lo++;
    }
  }
  if (!(hi == 0 && lo == 0) && !(hi == 1 && lo == 1)) {
    return {};
  }
  for (int i = 1; s == -1 && i <= n; i++)
    if (!adj[i].empty()) s = i;

  auto Dfs = [&](auto &Dfs, int u) -> void {
    while (!adj[u].empty()) {
      auto id = adj[u].back();
      adj[u].pop_back();
      int v = E[id].second;
      Dfs(Dfs, v);
      res.push_back(v);
    }
  };
  Dfs(Dfs, s);
```

```
  if (SZ(res) != SZ(E)) return {};
  ranges::reverse(res);
  return res;
}
```

5.9 eulerian-undigraph.cpp

```
optional<vector<int>> eulerian_path(int n, const vector<PII> &E) {
  vector<int> res;
  if (E.empty()) return res;
  vector<VI> adj(n + 1);
  for (int i = 0; i < ssize(E); i++) {
    auto [u, v] = E[i];
    adj[u].push_back(i);
    adj[v].push_back(i);
  }

  int s = -1, odd = 0;
  for (int i = 1; i <= n; i++) {
    if (ssize(adj[i]) % 2 == 0) continue;
    if (++odd > 2) return {};
    s = i;
  }
  for (int i = 1; s == -1 && i <= n; i++)
    if (!adj[i].empty()) s = i;

  vector<int> vis(ssize(E));
  auto Dfs = [&](auto &Dfs, int u) -> void {
    while (!adj[u].empty()) {
      auto id = adj[u].back();
      adj[u].pop_back();
      if (vis[id]) continue;
      vis[id] = 1;
      int v = u ^ E[id].fi ^ E[id].se;
      Dfs(Dfs, v);
      res.push_back(v);
    }
  };
  Dfs(Dfs, s);
  if (SZ(res) != SZ(E)) return {};
  ranges::reverse(res);
  return res;
}
```

5.10 hungarian.cpp

```
vector<vector<int>> e(SZ(rloc));
vector<int> match(SZ(cloc), -1), vis(SZ(rloc));
for (auto [u, v] : E) {
  e[u].push_back(v);
}
auto find = [&](auto&& find, int x) -> bool {
  vis[x] = 1;
  for (auto y : e[x]) {
    if (match[y] == -1 || (!vis[match[y]] && find(find, match[y]))) {
      match[y] = x;
      return true;
    }
  }
  return false;
};
auto DFSMatching = [&]() {
  int res = 0;
  rep(i, 0, SZ(rloc)) {
    fill(all(vis), 0);
    if (find(find, i)) res++;
  }
  return res;
};
```

5.11 KM.cpp

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;

// L <= R, 左边完全匹配
// 最小权完备匹配

// 带权匹配：使得该二分图的权值和最大（或最小）的匹配。
// 最大匹配：使得该二分图边数最多的匹配。
// 完备匹配：使得点数较小的点集中每个点都被匹配的匹配。
// 完美匹配：所有点都被匹配的匹配。
// 定理1：最大匹配数 = 最小点覆盖数（Konig 定理）
// 定理2：最大匹配数 = 最大独立数
// 定理3：最小路径覆盖数 = 顶点数 - 最大匹配数

// 二分图的最小点覆盖
```

```
// 定义：在二分图中，求最少的点集，使得每一条边至少都有端点在这个点集中。
// 二分图的最小点覆盖 = 二分图的最大匹配

// 二分图的最少边覆盖
// 定义：在二分图中，求最少的边，使得他们覆盖所有的点，并且每一个点只被一条边覆盖。
// 二分图的最少边覆盖 = 点数 - 二分图的最大匹配

// 二分图的最大独立集
// 定义：在二分图中，选最多的点，使得任意两个点之间没有直接边连接。
// 二分图的最大独立集 = 点数 - 二分图的最大匹配

template<class T>
pair<T, vector<int>> hungarian(const vector<vector<T>> &a) {
    if (a.empty()) return {0, {}};
    int n = a.size() + 1, m = a[0].size() + 1;
    vector<T> u(n), v(m); // 顶标
    vector<int> p(m), ans(n - 1);
    for (int i = 1; i < n; i++) {
        p[0] = i;
        int j0 = 0;
        vector<T> dist(m, numeric_limits<T>::max());
        vector<int> pre(m, -1);
        vector<bool> done(m + 1);
        do { // dijkstra
            done[j0] = true;
            int i0 = p[j0], j1;
            T delta = numeric_limits<T>::max();
            for (int j = 1; j < m; j++) if (!done[j]) {
                auto cur = a[i0 - 1][j - 1] - u[i0] - v[j];
                if (cur < dist[j]) dist[j] = cur, pre[j] = j0;
                if (dist[j] < delta) delta = dist[j], j1 = j;
            }
            for (int j = 0; j < m; j++) {
                if (done[j]) u[p[j]] += delta, v[j] -= delta;
                else dist[j] -= delta;
            }
            j0 = j1;
        } while (p[j0]);
        while (j0) { // update alternating path
            int j1 = pre[j0];
            p[j0] = p[j1], j0 = j1;
        }
    }
    for (int j = 1; j < m; j++) {
        if (p[j]) ans[p[j] - 1] = j - 1;
    }
    return {-v[0], ans}; // min cost
}

int L, R, m;
int main() {
    scanf("%d%d%d", &L, &R, &m);
    R = max(L, R);
    auto a = vector<vector<ll>>(L, vector<ll>(R, 0));
    for (int i = 0; i < m; i++) {
        int u, v, w;
        scanf("%d%d%d", &u, &v, &w);
        --u; --v;
        a[u][v] = -w;
    }
    auto [val, ans] = hungarian(a);
    printf("%lld\n", -val);
    for (int i = 0; i < L; i++) {
        if (a[i][ans[i]] >= 0) ans[i] = -1;
        printf("%d%c", ans[i] + 1, "␣\n"[i == L - 1]);
    }
}
```

5.12 kosaraju.cpp

```
vector<int> e[maxn], erev[maxn];
vector<int> c, out;
vector<vector<int>> scc;
int vis[maxn];
void dfs(int u) {
    vis[u] = 1;
    for (auto v : e[u]) if (!vis[v]) dfs(v);
    out.pb(u);
}
void dfs_rev(int u) {
    vis[u] = 1;
    for (auto v : erev[u]) if (!vis[v]) dfs_rev(v);
    c.pb(u);
}
void solve() {
```

```
    cin >> n >> m;
    rep(i, 1, m) {
        int u, v;
        cin >> u >> v;
        e[u].pb(v);
        erev[v].pb(u);
    }
    rep(i, 1, n) if (!vis[i]) dfs(i);
    fill(vis + 1, vis + n + 1, 0);
    reverse(all(out));
    for (auto v : out) if (!vis[v]) {
            c.clear();
            dfs_rev(v);
            scc.pb(c);
        }
}
```

5.13 kruskal 重构树.cpp

```
/**
 * 构建后是一颗二叉树，如果按最小生成树建立的话是大根堆。
 * 性质：原图中两个点间所有路径上的边最大权值的最小值=最小生成树上两点简单路径的边最大权值
 * =kruskal重构树上两点LCA的权值。
 * 重构树中代表原树中的点的节点全是叶子节点，其余节点都代表了一条边的边权。
 * 利用这个性质可以找到点P的简单路径上边权最大值小于lim深度最小的节点。
 * 要求最小权值最大值，可以建最大生成树的重构树从而达到一样的效果。
 */

vector<tuple<ll, ll, ll>> E;
rep(i, 1, m) {
    int u, v, w;
    cin >> u >> v >> w;
    E.emplace_back(w, u, v);
}
ranges::sort(E);
for (auto [w, u, v] : E) {
    u = find(u), v = find(v);
    if (u == v) continue;
    int p = ++idx;
    lim[p] = w;
    fa[u] = p, fa[v] = p;
    e[p].push_back(u);
    e[u].push_back(p);
    e[p].push_back(v);
    e[v].push_back(p);
}
```

5.14 MCMF.cpp

```
template<typename T>
struct MinCostGraph {
    static const int V = 20100;
    static const int E = 201000;
    int s, t, vtot;
    int head[V], etot;
    T dis[V], flow, cost;
    int pre[V];
    bool vis[V];

    struct edge {
        int v, nxt;
        T f, c;
    } e[E * 2];
    void addedge(int u,int v, T f, T c, T f2 = 0){
        e[etot]= {v, head[u], f, c}; head[u] = etot++;
        e[etot]= {u, head[v], f2, -c}; head[v] = etot++;
    }

    bool spfa() {
        T inf = numeric_limits<T>::max() / 2;
        for (int i = 1; i <= vtot; i++) {
            dis[i] = inf;
            vis[i] = false;
            pre[i] = -1;
        }
        dis[s] = 0;
        vis[s] = true;
        queue<int> q;
        q.push(s);
        while (!q.empty()) {
            int u = q.front();
            for (int i = head[u]; ~i; i = e[i].nxt) {
                int v = e[i].v;
                if (e[i].f && dis[v] > dis[u] + e[i].c) {
                    dis[v] = dis[u] + e[i].c;
```

```
                    pre[v] = i;
                    if (!vis[v]) {
                        vis[v] = 1;
                        q.push(v);
                    }
                }
            }
            q.pop();
            vis[u] = false;
        }
        return dis[t] != inf;
    }

    void augment() {
        int u = t;
        T f = numeric_limits<T>::max();
        while (~pre[u]) {
            f = min(f, e[pre[u]].f);
            u = e[pre[u] ^ 1].v;
        }
        flow += f;
        cost += f * dis[t];
        u = t;
        while (~pre[u]) {
            e[pre[u]].f -= f;
            e[pre[u] ^ 1].f += f;
            u = e[pre[u] ^ 1].v;
        }
    }

    pair<T, T> solve() {
        flow = 0;
        cost = 0;
        while (spfa()) augment();
        return {flow, cost};
    }
    void init(int s_, int t_, int vtot_) {
        s = s_;
        t = t_;
        vtot = vtot_;
        etot = 0;
        for (int i = 1; i <= vtot; i++) head[i] = -1;
    }
};
```

5.15 MCMFfast.cpp

```
template <typename flow_t = int, typename cost_t = long long>
struct MCMF_SSPA {
    int N;
    vector<vector<int>> adj;
    struct edge_t {
        int dest;
        flow_t cap;
        cost_t cost;
    };
    vector<edge_t> edges;

    vector<char> seen;
    vector<cost_t> pi;
    vector<int> prv;

    explicit MCMF_SSPA(int N_) : N(N_), adj(N), pi(N, 0), prv(N) {}

    void addEdge(int from, int to, flow_t cap, cost_t cost) {
        assert(cap >= 0);
        int e = int(edges.size());
        edges.emplace_back(edge_t{to, cap, cost});
        edges.emplace_back(edge_t{from, 0, -cost});
        adj[from].push_back(e);
        adj[to].push_back(e+1);
    }

    const cost_t INF_COST = numeric_limits<cost_t>::max() / 4;
    const flow_t INF_FLOW = numeric_limits<flow_t>::max() / 4;
    vector<cost_t> dist;
    __gnu_pbds::priority_queue<pair<cost_t, int>> q;
    vector<typename decltype(q)::point_iterator> its;
    void path(int s) {
        dist.assign(N, INF_COST);
        dist[s] = 0;

        its.assign(N, q.end());
        its[s] = q.push({0, s});

        while (!q.empty()) {
```

```
            int i = q.top().second; q.pop();
            cost_t d = dist[i];
            for (int e : adj[i]) {
                if (edges[e].cap) {
                    int j = edges[e].dest;
                    cost_t nd = d + edges[e].cost;
                    if (nd < dist[j]) {
                        dist[j] = nd;
                        prv[j] = e;
                        if (its[j] == q.end()) {
                            its[j] = q.push({-(dist[j] - pi[j]), j});
                        } else {
                            q.modify(its[j], {-(dist[j] - pi[j]), j});
                        }
                    }
                }
            }
        }

        swap(pi, dist);
    }

    vector<pair<flow_t, cost_t>> maxflow(int s, int t) {
        assert(s != t);
        flow_t totFlow = 0; cost_t totCost = 0;
        vector<pair<flow_t, cost_t>> res;
        while (path(s), pi[t] < INF_COST) {
            flow_t curFlow = numeric_limits<flow_t>::max();
            for (int cur = t; cur != s; ) {
                int e = prv[cur];
                int nxt = edges[e^1].dest;
                curFlow = min(curFlow, edges[e].cap);
                cur = nxt;
            }
            totFlow += curFlow;
            totCost += pi[t] * curFlow;
            for (int cur = t; cur != s; ) {
                int e = prv[cur];
                int nxt = edges[e^1].dest;
                edges[e].cap -= curFlow;
                edges[e^1].cap += curFlow;
                cur = nxt;
            }

            res.emplace_back(totFlow, totCost);
        }
        return res;
    }
};
```

5.16 MCMFfull.cpp

```
template <typename T, typename C>
class MCMF {
 public:
  static constexpr T eps = (T) 1e-9;

  struct edge {
    int from;
    int to;
    T c;
    T f;
    C cost;
  };

  int n;
  vector<vector<int>> g;
  vector<edge> edges;
  vector<C> d;
  vector<C> pot;
  __gnu_pbds::priority_queue<pair<C, int>> q;
  vector<typename decltype(q)::point_iterator> its;
  vector<int> pe;
  const C INF_C = numeric_limits<C>::max() / 2;

  explicit MCMF(int n_) : n(n_), g(n), d(n), pot(n, 0), its(n), pe(n) {}

  int add(int from, int to, T forward_cap, T backward_cap, C edge_cost) {
    assert(0 <= from && from < n && 0 <= to && to < n);
    assert(forward_cap >= 0 && backward_cap >= 0);
    int id = static_cast<int>(edges.size());
    g[from].push_back(id);
    edges.push_back({from, to, forward_cap, 0, edge_cost});
    g[to].push_back(id + 1);
    edges.push_back({to, from, backward_cap, 0, -edge_cost});
    return id;
```

```
  }

  void expath(int st) {
    fill(d.begin(), d.end(), INF_C);
    q.clear();
    fill(its.begin(), its.end(), q.end());
    its[st] = q.push({pot[st], st});
    d[st] = 0;
    while (!q.empty()) {
      int i = q.top().second;
      q.pop();
      its[i] = q.end();
      for (int id : g[i]) {
        const edge &e = edges[id];
        int j = e.to;
        if (e.c - e.f > eps && d[i] + e.cost < d[j]) {
          d[j] = d[i] + e.cost;
          pe[j] = id;
          if (its[j] == q.end()) {
            its[j] = q.push({pot[j] - d[j], j});
          } else {
            q.modify(its[j], {pot[j] - d[j], j});
          }
        }
      }
    }
    swap(d, pot);
  }

  pair<T, C> calc(int st, int fin) {  // max_flow_min_cost
    T flow = 0;
    C cost = 0;
    bool ok = true;
    for (auto& e : edges) {
      if (e.c - e.f > eps && e.cost + pot[e.from] - pot[e.to] < 0) {
        ok = false;
        break;
      }
    }
    if (ok) {
      expath(st);
    } else {
      vector<int> deg(n, 0);
      for (int i = 0; i < n; i++) {
        for (int eid : g[i]) {
          auto& e = edges[eid];
          if (e.c - e.f > eps) {
            deg[e.to] += 1;
          }
        }
      }
      vector<int> que;
      for (int i = 0; i < n; i++) {
        if (deg[i] == 0) {
          que.push_back(i);
        }
      }
      for (int b = 0; b < (int) que.size(); b++) {
        for (int eid : g[que[b]]) {
          auto& e = edges[eid];
          if (e.c - e.f > eps) {
            deg[e.to] -= 1;
            if (deg[e.to] == 0) {
              que.push_back(e.to);
            }
          }
        }
      }
      fill(pot.begin(), pot.end(), INF_C);
      pot[st] = 0;
      if (static_cast<int>(que.size()) == n) {
        for (int v : que) {
          if (pot[v] < INF_C) {
            for (int eid : g[v]) {
              auto& e = edges[eid];
              if (e.c - e.f > eps) {
                if (pot[v] + e.cost < pot[e.to]) {
                  pot[e.to] = pot[v] + e.cost;
                  pe[e.to] = eid;
                }
              }
            }
          }
        }
      } else {
        que.assign(1, st);
```

```
        vector<bool> in_queue(n, false);
        in_queue[st] = true;
        for (int b = 0; b < (int) que.size(); b++) {
          int i = que[b];
          in_queue[i] = false;
          for (int id : g[i]) {
            const edge &e = edges[id];
            if (e.c - e.f > eps && pot[i] + e.cost < pot[e.to]) {
              pot[e.to] = pot[i] + e.cost;
              pe[e.to] = id;
              if (!in_queue[e.to]) {
                que.push_back(e.to);
                in_queue[e.to] = true;
              }
            }
          }
        }
      }
    }
    // debug(pot[fin]);
    while (pot[fin] < INF_C) {  // < 0
      T push = numeric_limits<T>::max();
      int v = fin;
      while (v != st) {
        const edge &e = edges[pe[v]];
        push = min(push, e.c - e.f);
        v = e.from;
      }
      v = fin;
      while (v != st) {
        edge &e = edges[pe[v]];
        e.f += push;
        edge &back = edges[pe[v] ^ 1];
        back.f -= push;
        v = e.from;
      }
      flow += push;
      cost += push * pot[fin];
      expath(st);
    }
    return {flow, cost};
  }
};
```

5.17 prim.cpp

```
vector<PII> e[N];

template <typename T>
void add(int u, int v, T w) {
    e[u].eb(v, w);
}

template <typename T>
T prim(vector<pair<int, T>> *g, int start) {
    const T inf = numeric_limits<T>::max() / 4;
    T res = 0;
    vector<T> dist(n, inf);
    dist[start] = 0;
    priority_queue<pair<T, int>, vector<pair<T, int>>, greater<pair<T, int
        >>> s;
    s.emplace(dist[start], start);
    vector<int> was(n, 0);
    while (!s.empty()) {
        T expected = s.top().first;
        int i = s.top().second;
        s.pop();
        if (dist[i] != expected || was[i]) {
            continue;
        }
        was[i] = 1;
        res += expected;
        for (auto [to, cost] : g[i]) {
            if (cost < dist[to]) {
                dist[to] = cost;
                s.emplace(dist[to], to);
            }
        }
    }
    return res;
}
```

5.18 PushRelabel.cpp

```
/**
 * Author: Simon Lindholm
 * Date: 2015-02-24
 * License: CC0
```

```
 * Source: Wikipedia, tinyKACTL
 * Description: Push-relabel using the highest label selection rule and the
     gap heuristic. Quite fast in practice.
 *  To obtain the actual flow, look at positive values only.
 * Time: $O(V^2\sqrt E)$
 * Status: Tested on Kattis and SPOJ, and stress-tested
 */
#pragma once

struct PushRelabel {
  typedef vector<int> vi;
  struct Edge {
    int dest, back;
    ll f, c;
  };
  vector<vector<Edge>> g;
  vector<ll> ec;
  vector<Edge*> cur;
  vector<vi> hs; vi H;
  PushRelabel(int n) : g(n), ec(n), cur(n), hs(2*n), H(n) {}

  void addEdge(int s, int t, ll cap, ll rcap=0) {
    if (s == t) return;
    g[s].push_back({t, SZ(g[t]), 0, cap});
    g[t].push_back({s, SZ(g[s])-1, 0, rcap});
  }

  void addFlow(Edge& e, ll f) {
    Edge &back = g[e.dest][e.back];
    if (!ec[e.dest] && f) hs[H[e.dest]].push_back(e.dest);
    e.f += f; e.c -= f; ec[e.dest] += f;
    back.f -= f; back.c += f; ec[back.dest] -= f;
  }
  ll calc(int s, int t) {
    int v = SZ(g); H[s] = v; ec[t] = 1;
    vi co(2*v); co[0] = v-1;
    rep(i,0,v-1) cur[i] = g[i].data();
    for (Edge& e : g[s]) addFlow(e, e.c);

    for (int hi = 0;;) {
      while (hs[hi].empty()) if (!hi--) return -ec[s];
      int u = hs[hi].back(); hs[hi].pop_back();
      while (ec[u] > 0)  // discharge u
        if (cur[u] == g[u].data() + SZ(g[u])) {
          H[u] = 1e9;
          for (Edge& e : g[u]) if (e.c && H[u] > H[e.dest]+1)
            H[u] = H[e.dest]+1, cur[u] = &e;
          if (++co[H[u]], !--co[hi] && hi < v)
            rep(i,0,v-1) if (hi < H[i] && H[i] < v)
              --co[H[i]], H[i] = v + 1;
          hi = H[u];
        } else if (cur[u]->c && H[u] == H[cur[u]->dest]+1)
          addFlow(*cur[u], min(ec[u], cur[u]->c));
        else ++cur[u];
    }
  }
  bool leftOfMinCut(int a) { return H[a] >= SZ(g); }
};
```

5.19 tarjan 割点.cpp

```
vector<int> g[maxn], ans;
stack<int> stk;
int dfn[maxn], cut[maxn], low[maxn], idx;

void dfs(int x, int f) {
    low[x] = dfn[x] = ++idx;
    stk.push(x);
    int ch = 0;
    for (auto y : g[x]) {
        if (!dfn[y]) {
            ch++;
            dfs(y, x);
            low[x] = min(low[x], low[y]);
            if (low[y] >= dfn[x]) cut[x] = 1;
        } else {
            if (y != f) low[x] = min(low[x], dfn[y]);
        }
    }
    if (x == 1 && ch <= 1) cut[x] = 0;
    if (cut[x]) ans.pb(x);
}
```

5.20 tarjan 割边.cpp

```
vector<PII> g[maxn];
```

```
stack<int> stk;
int dfn[maxn], ins[maxn], low[maxn];
int idx, tot;
VI ans;
void dfs(int x, int f) {
    low[x] = dfn[x] = ++idx;
    stk.push(x);
    ins[x] = 1;
    for (auto [y, id] : g[x]) {
        if (!dfn[y]) {
            dfs(y, id);
            low[x] = min(low[x], low[y]);
        } else {
            if (ins[y] && id != f) low[x] = min(low[x], dfn[y]);
        }
    }
    if (low[x] >= dfn[x]) {
        ++tot;
        while (true) {
            int cur = stk.top();
            stk.pop();
            ins[cur] = 0;
            if (cur == x) break;
        }
        if (f != 0) ans.pb(f);
    }
}
```

5.21 tarjan 强连通分量.cpp

```
vector<int> g[maxn];
stack<int> stk;
int dfn[maxn], ins[maxn], low[maxn], belong[maxn];
int idx, tot;

void dfs(int x) {
    low[x] = dfn[x] = ++idx;
    ins[x] = 1;
    stk.push(x);
    for (auto y : g[x]) {
        if (!dfn[y]) {
            dfs(y);
            low[x] = min(low[x], low[y]);
        } else {
            if (ins[y]) low[x] = min(low[x], dfn[y]);
        }
    }
    if (low[x] >= dfn[x]) {
        ++tot;
        while (true) {
            int cur = stk.top(); stk.pop();
            ins[cur] = 0;
            belong[cur] = tot;
            if (cur == x) break;
        }
    }
}
```

5.22 tarjan 点双.cpp

```
vector<int> g[maxn];
stack<int> stk;
int dfn[maxn], low[maxn], idx, tot, cut[maxn];
vector<int> bcc[maxn];

void dfs(int x, int f) {
    low[x] = dfn[x] = ++idx;
    stk.push(x);
    int ch = 0;
    for (auto y : g[x]) {
        if (!dfn[y]) {
            ch++;
            dfs(y, x);
            low[x] = min(low[x], low[y]);
            if (low[y] >= dfn[x]) {
                cut[x] = 1;
                ++tot;
                bcc[tot].pb(x);
                while (true) {
                    int cur = stk.top();
                    stk.pop();
                    bcc[tot].pb(cur);
                    if (cur == y) break;
                }
            }
        } else {
```

```
            if (y != f) low[x] = min(low[x], dfn[y]);
        }
    }
    if (x == 1 && ch <= 1) cut[x] = 0;
}
```

5.23 tarjan 边双.cpp

```
vector<PII> g[maxn];
stack<int> stk;
int dfn[maxn], low[maxn], idx, tot, belong[maxn];
vector<int> bcc[maxn];

void dfs(int x, int f) {
    low[x] = dfn[x] = ++idx;
    stk.push(x);
    for (auto [y, id] : g[x]) {
        if (!dfn[y]) {
            dfs(y, id);
            low[x] = min(low[x], low[y]);
        } else {
            if (id != f) low[x] = min(low[x], dfn[y]);
        }
    }
    if (low[x] >= dfn[x]) {
        ++tot;
        while (true) {
            int cur = stk.top();
            stk.pop();
            belong[cur] = tot;
            bcc[tot].pb(cur);
            if (cur == x) break;
        }
    }
}
```

5.24 twosat.cpp

```
class twosat {
public:
    digraph<int> g;
    int n;

    twosat(int _n) : g(digraph<int>(2 * _n)), n(_n) {
    }

    // (v[x] == value_x)
    inline void add(int x, int value_x) {
        assert(0 <= x && x < n);
        assert(0 <= value_x && value_x <= 1);
        g.add(2 * x + (value_x ^ 1), 2 * x + value_x);
    }

    // (v[x] == value_x || v[y] == value_y)
    inline void add(int x, int value_x, int y, int value_y) {
        assert(0 <= x && x < n && 0 <= y && y < n);
        assert(0 <= value_x && value_x <= 1 && 0 <= value_y && value_y <= 1)
            ;
        g.add(2 * x + (value_x ^ 1), 2 * y + value_y);
        g.add(2 * y + (value_y ^ 1), 2 * x + value_x);
    }

    inline vector<int> solve() {
        int cnt;
        vector<int> c = find_scc(g, cnt);
        vector<int> res(n);
        for (int i = 0; i < n; i++) {
            if (c[2 * i] == c[2 * i + 1]) {
                return vector<int>();
            }
            res[i] = (c[2 * i] < c[2 * i + 1]);
        }
        return res;
    }
};
```

5.25 差分约束系统.cpp

```
/**
    Description:
    求解方程组 x_u - x_v <= w_i, 求出的x_i为满足条件的最大值
     转化为x_u <= x_v + w_i
    问题等价于求最短路（bellmanford或Floyd）
    即加一条有向边add(u, v, w), dist[v] = min(dist[v], dist[u] + w)
    求最小值（满足条件情况下尽量小）等价于求(-x_i)最大（或者转化为求最长路）
     求非负解只需要添加超级节点S，S向各个点连边（S + 0 <= x_i），再设dist[S]
```

```
          = 0
 */
void solve() {
    cin >> n >> m;
    vector<int> dist(n, 0);
    vector<vector<PII>> g(n);
    rep(i, 0, m - 1) {
        int u, v, w;
        cin >> u >> v >> w;
        u--, v--;
        g[u].eb(v, -w);
    }
    bool ok = 1;
    rep(i, 1, n) {
        bool upd = 0;
        rep(u, 0, n - 1) {
            for (auto [v, w] : g[u]) {
                if (dist[v] < dist[u] + w) {
                    dist[v] = dist[u] + w;
                    upd = 1;
                }
            }
        }
        if (!upd) break;
        // 仍然有约束未满足
        if (i == n && upd) ok = 0;
    }
    if (!ok) {
        return cout << -1 << '\n', void();
    }
    rep(i, 0, n - 1) {
        cout << dist[i] << "␣\n"[i == n - 1];
    }
}
```

6 Math

6.1 binom.cpp

```
vector<Mint> fact(1, 1);
vector<Mint> inv_fact(1, 1);

Mint C(int n, int k) {
    if (k < 0 || k > n) {
        return 0;
    }
    while ((int)fact.size() < n + 1) {
        fact.push_back(fact.back() * (int)fact.size());
        inv_fact.push_back(1 / fact.back());
    }
    return fact[n] * inv_fact[k] * inv_fact[n - k];
}

const int mod = 1000000007;
const int T = 1000000;
ll fact[] = {};
ll powmod(ll a, ll b) {
    ll ret = 1;
    for (; b; b >>= 1) {
        if (b & 1) ret = ret * a % mod;
        a = a * a %mod;
    }
    return ret;
}
ll fac(int n) {
    ll v = fact[n / T];
    for (int i = n / T * T + 1; i <= n; i++)
        v = v * i % mod;
    return v;
}
ll binom(int n, int m) {
    if (m < 0 || m > n) return 0;
    return fac(n) * powmod(fac(m) * fac(n - m) % mod, mod - 2) % mod;
}
```

6.2 bsgs.cpp

```
int bsgs(int a, int b, int m) { // a^x=b(mod m)
    int res = m + 1;
    int t = sqrt(m) + 2;
    ll d = powmod(a, t, m);
    ll cnt = 1;
    //map<int,int> p;
    hs.init();
    for (int i = 1; i <= t; i++) {
```

```
        cnt = cnt * d % m;
        //if (!p.count(cnt)) p[cnt] = i;
        if (hs.query(cnt) == -1) hs.insert(cnt, i);
    }
    cnt = b;
    for (int i = 1; i <= t; i++) {
        cnt = cnt * a % m;
        //if (p.count(cnt)) res = min(res, p[cnt] * t - i);
        int tmp = hs.query(cnt);
        if (tmp != -1) res = min(res, tmp * t - i);
    }
    if (res >= m) res = -1;
    return res;
}
```

6.3 cantor.cpp

```
ll fac[maxn], A[maxn], w[maxn];
void init(int n) {
    fac[0] = 1;
    rep(i, 1, n) fac[i] = fac[i - 1] * i % mod;
}
ll cantor(int w[], int n) {
    ll ans = 1;
    for (int i = 1; i <= n; i++) { // can optimize by BIT
        for (int j = i + 1; j <= n; j++) {
            if (w[i] > w[j]) A[i]++;
        }
    }
    for (int i = 1; i < n; i++) {
        ans += A[i] * fac[n - i];
    }
    return ans;
}

void decanter(ll x, int n) { // x->rank n->length
    x--;
    vector<int> rest(n, 0);
    iota(rest.begin(), rest.end(), 1); // rest->1,2,3,4...
    for (int i = 1; i <= n; i++) {
        A[i] = x / fac[n - i];
        x %= fac[n - i];
    }
    for (int i = 1; i <= n; i++) {
        w[i] = rest[A[i]];
        rest.erase(lower_bound(rest.begin(), rest.end(), w[i]));
    }
}
```

6.4 EXCRT modequ exgcd.cpp

```
ll exgcd(ll a, ll b, ll &x, ll &y) {
    if (b == 0) {
        x = 1, y = 0;
        return a;
    }
    ll d = exgcd(b, a % b, y, x);
    y -= (a / b) * x;
    return d;
}

// 求 a * x = b (mod m) 的解
ll modequ(ll a, ll b, ll m) {
    ll x, y;
    ll d = exgcd(a, m, x, y);
    if (b % d != 0) return -1;
    m /= d; a /= d; b /= d;
    x = x * b % m;
    if (x < 0) x += m;
    return x;
}

void merge(ll &a, ll &b, ll c, ll d) {
    if (a == -1 || b == -1) return;
    ll x, y;
    ll g = exgcd(b, d, x, y);
    if ((c - a) % g != 0) {
        a = -1, b = -1;
        return;
    }
    d /= g;
    ll t = ((c - a) / g) % d * x % d;
    if (t < 0) t += d;
    a = b * t + a;
    b = b * d;
}
```

6.5 factor.cpp

```
namespace Factor {
    const int N=1010000;
    ll C,fac[10010],n,mut,a[1001000];
    int T,cnt,i,l,prime[N],p[N],psize,_cnt;
    ll _e[100],_pr[100];
    vector<ll> d;
    inline ll mul(ll a,ll b,ll p) {
        if (p<=1000000000) return a*b%p;
        else if (p<=100000000000000ll) return (((a*(b>>20)%p)<<20)+(a*(b
            &((1<<20)-1))))%p;
        else {
            ll d=(ll)floor(a*(long double)b/p+0.5);
            ll ret=(a*b-d*p)%p;
            if (ret<0) ret+=p;
            return ret;
        }
    }
    void prime_table(){
        int i,j,tot,t1;
        for (i=1;i<=psize;i++) p[i]=i;
        for (i=2,tot=0;i<=psize;i++){
            if (p[i]==i) prime[++tot]=i;
            for (j=1;j<=tot && (t1=prime[j]*i)<=psize;j++){
                p[t1]=prime[j];
                if (i%prime[j]==0) break;
            }
        }
    }
    void init(int ps) {
        psize=ps;
        prime_table();
    }
    ll powl(ll a,ll n,ll p) {
        ll ans=1;
        for (;n;n>>=1) {
            if (n&1) ans=mul(ans,a,p);
            a=mul(a,a,p);
        }
        return ans;
    }
    bool witness(ll a,ll n) {
        int t=0;
        ll u=n-1;
        for (;~u&1;u>>=1) t++;
        ll x=powl(a,u,n),_x=0;
        for (;t;t--) {
            _x=mul(x,x,n);
            if (_x==1 && x!=1 && x!=n-1) return 1;
            x=_x;
        }
        return _x!=1;
    }
    bool miller(ll n) {
        if (n<2) return 0;
        if (n<=psize) return p[n]==n;
        if (~n&1) return 0;
        for (int j=0;j<=7;j++) if (witness(rng()%(n-1)+1,n)) return 0;
        return 1;
    }
    ll gcd(ll a,ll b) {
        ll ret=1;
        while (a!=0) {
            if ((~a&1) && (~b&1)) ret<<=1,a>>=1,b>>=1;
            else if (~a&1) a>>=1; else if (~b&1) b>>=1;
            else {
                if (a<b) swap(a,b);
                a-=b;
            }
        }
        return ret*b;
    }
    ll rho(ll n) {
        while (1) {
            ll X=rng()%n,Y,Z,T=1,*lY=a,*lX=lY;
            int tmp=20;
            C=rng()%10+3;
            X=mul(X,X,n)+C;*(lY++)=X;lX++;
            Y=mul(X,X,n)+C;*(lY++)=Y;
            for(;X!=Y;) {
                ll t=X-Y+n;
                Z=mul(T,t,n);
                if(Z==0) return gcd(T,n);
                tmp--;
                if (tmp==0) {
                    tmp=20;
```

```
                    Z=gcd(Z,n);
                    if (Z!=1 && Z!=n) return Z;
                }
                T=Z;
                Y=*(lY++)=mul(Y,Y,n)+C;
                Y=*(lY++)=mul(Y,Y,n)+C;
                X=*(lX++);
            }
        }
    }
    void _factor(ll n) {
        for (int i=0;i<cnt;i++) {
            if (n%fac[i]==0) n/=fac[i],fac[cnt++]=fac[i];}
        if (n<=psize) {
            for (;n!=1;n/=p[n]) fac[cnt++]=p[n];
            return;
        }
        if (miller(n)) fac[cnt++]=n;
        else {
            ll x=rho(n);
            _factor(x);_factor(n/x);
        }
    }
    void dfs(ll x,int dep) {
        if (dep==_cnt) d.pb(x);
        else {
            dfs(x,dep+1);
            for (int i=1;i<=_e[dep];i++) dfs(x*=_pr[dep],dep+1);
        }
    }
    void norm() {
        sort(fac,fac+cnt);
        _cnt=0;
        rep(i,0,cnt-1) if (i==0||fac[i]!=fac[i-1]) _pr[_cnt]=fac[i],_e[_cnt
            ++]=1;
            else _e[_cnt-1]++;
    }
    vector<ll> getd() {
        d.clear();
        dfs(1,0);
        return d;
    }
    vector<ll> factor(ll n) {
        cnt=0;
        _factor(n);
        norm();
        return getd();
    }
    vector<PLL> factorG(ll n) {
        cnt=0;
        _factor(n);
        norm();
        vector<PLL> d;
        rep(i,0,_cnt-1) d.pb(mp(_pr[i],_e[i]));
        return d;
    }
    bool is_primitive(ll a,ll p) {
        assert(miller(p));
        vector<PLL> D=factorG(p-1);
        rep(i,0,SZ(D)-1) if (powl(a,(p-1)/D[i].fi,p)==1) return 0;
        return 1;
    }
    ll phi(ll n) {
        auto d=factorG(n);
        for (auto p:d) n=n/p.fi*(p.fi-1);
        return n;
    }
}
```

6.6 fft.cpp

```
namespace fft {
  typedef double dbl;

  struct num {
    dbl x, y;
    num() { x = y = 0; }
    num(dbl x, dbl y) : x(x), y(y) { }
  };

  inline num operator+(num a, num b) { return num(a.x + b.x, a.y + b.y); }
  inline num operator-(num a, num b) { return num(a.x - b.x, a.y - b.y); }
  inline num operator*(num a, num b) { return num(a.x * b.x - a.y * b.y, a.x
       * b.y + a.y * b.x); }
  inline num conj(num a) { return num(a.x, -a.y); }

```

```
  int base = 1;
  vector<num> roots = {{0, 0}, {1, 0}};
  vector<int> rev = {0, 1};

  const dbl PI = acosl(-1.0);

  void ensure_base(int nbase) {
    if (nbase <= base) {
      return;
    }
    rev.resize(1 << nbase);
    for (int i = 0; i < (1 << nbase); i++) {
      rev[i] = (rev[i >> 1] >> 1) + ((i & 1) << (nbase - 1));
    }
    roots.resize(1 << nbase);
    while (base < nbase) {
      dbl angle = 2 * PI / (1 << (base + 1));
//      num z(cos(angle), sin(angle));
      for (int i = 1 << (base - 1); i < (1 << base); i++) {
        roots[i << 1] = roots[i];
//        roots[(i << 1) + 1] = roots[i] * z;
        dbl angle_i = angle * (2 * i + 1 - (1 << base));
        roots[(i << 1) + 1] = num(cos(angle_i), sin(angle_i));
      }
      base++;
    }
  }

  void fft(vector<num> &a, int n = -1) {
    if (n == -1) {
      n = a.size();
    }
    assert((n & (n - 1)) == 0);
    int zeros = __builtin_ctz(n);
    ensure_base(zeros);
    int shift = base - zeros;
    for (int i = 0; i < n; i++) {
      if (i < (rev[i] >> shift)) {
        swap(a[i], a[rev[i] >> shift]);
      }
    }
/*    for (int k = 1; k < n; k <<= 1) {
      for (int i = 0; i < n; i += 2 * k) {
        for (int j = 0; j < k; j++) {
          num z = a[i + j + k] * roots[j + k];
          a[i + j + k] = a[i + j] - z;
          a[i + j] = a[i + j] + z;
        }
      }
    }*/
    for (int len = 1; len < n; len <<= 1) {
      for (int i = 0; i < n; i += 2 * len) {
        for (int j = i, k = i + len; j < i + len; j++, k++) {
          num z = a[k] * roots[k - i];
          a[k] = a[j] - z;
          a[j] = a[j] + z;
        }
      }
    }
  }

  vector<num> fa, fb;

  vector<long long> multiply(vector<int> &a, vector<int> &b) {
    int need = a.size() + b.size() - 1;
    int nbase = 0;
    while ((1 << nbase) < need) nbase++;
    ensure_base(nbase);
    int sz = 1 << nbase;
    if (sz > (int) fa.size()) {
      fa.resize(sz);
    }
    for (int i = 0; i < sz; i++) {
      int x = (i < (int) a.size() ? a[i] : 0);
      int y = (i < (int) b.size() ? b[i] : 0);
      fa[i] = num(x, y);
    }
    fft(fa, sz);
    num r(0, -0.25 / sz);
    for (int i = 0; i <= (sz >> 1); i++) {
      int j = (sz - i) & (sz - 1);
      num z = (fa[j] * fa[j] - conj(fa[i] * fa[i])) * r;
      if (i != j) {
        fa[j] = (fa[i] * fa[i] - conj(fa[j] * fa[j])) * r;
      }
      fa[i] = z;
```

```
    }
    fft(fa, sz);
    vector<long long> res(need);
    for (int i = 0; i < need; i++) {
      res[i] = fa[i].x + 0.5;
    }
    return res;
  }

  vector<int> multiply_mod(vector<int> &a, vector<int> &b, int m, int eq =
      0) {
    int need = a.size() + b.size() - 1;
    int nbase = 0;
    while ((1 << nbase) < need) nbase++;
    ensure_base(nbase);
    int sz = 1 << nbase;
    if (sz > (int) fa.size()) {
      fa.resize(sz);
    }
    for (int i = 0; i < (int) a.size(); i++) {
      int x = (a[i] % m + m) % m;
      fa[i] = num(x & ((1 << 15) - 1), x >> 15);
    }
    fill(fa.begin() + a.size(), fa.begin() + sz, num {0, 0});
    fft(fa, sz);
    if (eq) {
      copy(fa.begin(), fa.begin() + sz, fb.begin());
    } else {
      if (sz > (int) fb.size()) {
        fb.resize(sz);
      }
      for (int i = 0; i < (int) b.size(); i++) {
        int x = (b[i] % m + m) % m;
        fb[i] = num(x & ((1 << 15) - 1), x >> 15);
      }
      fill(fb.begin() + b.size(), fb.begin() + sz, num {0, 0});
      fft(fb, sz);
    }
    dbl ratio = 0.25 / sz;
    num r2(0, -1);
    num r3(ratio, 0);
    num r4(0, -ratio);
    num r5(0, 1);
    for (int i = 0; i <= (sz >> 1); i++) {
      int j = (sz - i) & (sz - 1);
      num a1 = (fa[i] + conj(fa[j]));
      num a2 = (fa[i] - conj(fa[j])) * r2;
      num b1 = (fb[i] + conj(fb[j])) * r3;
      num b2 = (fb[i] - conj(fb[j])) * r4;
      if (i != j) {
        num c1 = (fa[j] + conj(fa[i]));
        num c2 = (fa[j] - conj(fa[i])) * r2;
        num d1 = (fb[j] + conj(fb[i])) * r3;
        num d2 = (fb[j] - conj(fb[i])) * r4;
        fa[i] = c1 * d1 + c2 * d2 * r5;
        fb[i] = c1 * d2 + c2 * d1;
      }
      fa[j] = a1 * b1 + a2 * b2 * r5;
      fb[j] = a1 * b2 + a2 * b1;
    }
    fft(fa, sz);
    fft(fb, sz);
    vector<int> res(need);
    for (int i = 0; i < need; i++) {
      long long aa = fa[i].x + 0.5;
      long long bb = fb[i].x + 0.5;
      long long cc = fa[i].y + 0.5;
      res[i] = (aa + ((bb % m) << 15) + ((cc % m) << 30)) % m;
    }
    return res;
  }

  vector<int> square_mod(vector<int> &a, int m) {
    return multiply_mod(a, a, m, 1);
  }
  // fft::multiply uses dbl, outputs vector<long long> of rounded values
  // fft::multiply_mod might work for res.size() up to 2^21
  // typedef long double dbl;          =>        up to 2^25 (but takes a lot
      of memory)
};
```

6.7 fftfast.cpp

```
// FFT_MAXN = 2^k
// fft_init() to precalc FFT_MAXN-th roots

```

```
typedef long double db;
const int FFT_MAXN = 262144;
const int N = 3.1e5;
const db pi = acosl(-1.);
struct cp {
    db a, b;
    cp operator+(const cp &y) const { return (cp){a + y.a, b + y.b}; }
    cp operator-(const cp &y) const { return (cp){a - y.a, b - y.b}; }
    cp operator*(const cp &y) const { return (cp){a * y.a - b * y.b, a * y.b + b * y.a}; }
    cp operator!() const { return (cp){a, -b}; };
} nw[FFT_MAXN + 1];
int bitrev[FFT_MAXN];
void dft(cp *a, int n, int flag = 1) {
    int d = 0;
    while ((1 << d) * n != FFT_MAXN) d++;
    rep(i, 0, n - 1) if (i < (bitrev[i] >> d)) swap(a[i], a[bitrev[i] >> d]);
    for (int l = 2; l <= n; l <<= 1) {
        int del = FFT_MAXN / l * flag;
        for (int i = 0; i < n; i += l) {
            cp *le = a + i, *ri = a + i + (l >> 1), *w = flag == 1 ? nw : nw + FFT_MAXN;
            rep(k, 0, l / 2 - 1) {
                cp ne = *ri * *w;
                *ri = *le - ne, *le = *le + ne;
                le++, ri++, w += del;
            }
        }
    }
    if (flag != 1) rep(i, 0, n - 1) a[i].a /= n, a[i].b /= n;
}
void fft_init() {
    int L = 0;
    while ((1 << L) != FFT_MAXN) L++;
    bitrev[0] = 0;
    rep(i, 1, FFT_MAXN - 1) bitrev[i] = bitrev[i >> 1] >> 1 | ((i & 1) << (L - 1));
    nw[0] = nw[FFT_MAXN] = (cp){1, 0};
    rep(i, 0, FFT_MAXN)
    nw[i] = (cp){cosl(2 * pi / FFT_MAXN * i), sinl(2 * pi / FFT_MAXN * i)};
        // very slow
}

void convo(db *a, int n, db *b, int m, db *c) {
    static cp f[FFT_MAXN >> 1], g[FFT_MAXN >> 1], t[FFT_MAXN >> 1];
    int N = 2;
    while (N <= n + m) N <<= 1;
    rep(i, 0, N - 1) if (i & 1) {
        f[i >> 1].b = (i <= n) ? a[i] : 0.0;
        g[i >> 1].b = (i <= m) ? b[i] : 0.0;
    }
    else {
        f[i >> 1].a = (i <= n) ? a[i] : 0.0;
        g[i >> 1].a = (i <= m) ? b[i] : 0.0;
    }
    dft(f, N >> 1);
    dft(g, N >> 1);
    int del = FFT_MAXN / (N >> 1);
    cp qua = (cp){0, 0.25}, one = (cp){1, 0}, four = (cp){4, 0}, *w = nw;
    rep(i, 0, N / 2 - 1) {
        int j = i ? (N >> 1) - i : 0;
        t[i] = (four * !(f[j] * g[j]) - (!f[j] - f[i]) * (!g[j] - g[i]) * (one + *w)) * qua;
        w += del;
    }
    dft(t, N >> 1, -1);
    rep(i, 0, n + m) c[i] = (i & 1) ? t[i >> 1].a : t[i >> 1].b;
}

int mo;
void mul(int *a, int *b, int n) {  // n<=N, 0<=a[i],b[i]<mo
    static cp f[N], g[N], t[N], r[N];
    int nn = 2;
    while (nn <= n + n) nn <<= 1;
    rep(i, 0, nn - 1) {
        f[i] = (i <= n) ? (cp){(db)(a[i] >> 15), (db)(a[i] & 32767)} : (cp){0, 0};
        g[i] = (i <= n) ? (cp){(db)(b[i] >> 15), (db)(b[i] & 32767)} : (cp){0, 0};
    }
    swap(n, nn);
    dft(f, n, 1);
    dft(g, n, 1);
    rep(i, 0, n - 1) {
        int j = i ? n - i : 0;
```

```
        t[i] = ((f[i] + !f[j]) * (!g[j] - g[i]) + (!f[j] - f[i]) * (g[i] + !g[j])) * (cp){0, 0.25};
        r[i] = (!f[j] - f[i]) * (!g[j] - g[i]) * (cp){-0.25, 0} + (cp){0, 0.25} * (f[i] + !f[j]) * (g[i] + !g[j]);
    }
    dft(t, n, -1);
    dft(r, n, -1);
    rep(i, 0, n - 1)
    a[i] = ((ll(t[i].a + 0.5) % mo << 15) + ll(r[i].a + 0.5) + (ll(r[i].b + 0.5) % mo << 30)) % mo;
}
```

6.8 fftnew.cpp

```
namespace fft {

typedef double dbl;

struct num {
  dbl x, y;
  num() { x = y = 0; }
  num(dbl x_, dbl y_) : x(x_), y(y_) {}
};

inline num operator+(num a, num b) { return num(a.x + b.x, a.y + b.y); }
inline num operator-(num a, num b) { return num(a.x - b.x, a.y - b.y); }
inline num operator*(num a, num b) { return num(a.x * b.x - a.y * b.y, a.x * b.y + a.y * b.x); }
inline num conj(num a) { return num(a.x, -a.y); }

int base = 1;
vector<num> roots = {{0, 0}, {1, 0}};
vector<int> rev = {0, 1};

const dbl PI = static_cast<dbl>(acosl(-1.0));

void ensure_base(int nbase) {
  if (nbase <= base) {
    return;
  }
  rev.resize(1 << nbase);
  for (int i = 0; i < (1 << nbase); i++) {
    rev[i] = (rev[i >> 1] >> 1) + ((i & 1) << (nbase - 1));
  }
  roots.resize(1 << nbase);
  while (base < nbase) {
    dbl angle = 2 * PI / (1 << (base + 1));
//      num z(cos(angle), sin(angle));
    for (int i = 1 << (base - 1); i < (1 << base); i++) {
      roots[i << 1] = roots[i];
//        roots[(i << 1) + 1] = roots[i] * z;
      dbl angle_i = angle * (2 * i + 1 - (1 << base));
      roots[(i << 1) + 1] = num(cos(angle_i), sin(angle_i));
    }
    base++;
  }
}

void fft(vector<num>& a, int n = -1) {
  if (n == -1) {
    n = (int) a.size();
  }
  assert((n & (n - 1)) == 0);
  int zeros = __builtin_ctz(n);
  ensure_base(zeros);
  int shift = base - zeros;
  for (int i = 0; i < n; i++) {
    if (i < (rev[i] >> shift)) {
      swap(a[i], a[rev[i] >> shift]);
    }
  }
  for (int k = 1; k < n; k <<= 1) {
    for (int i = 0; i < n; i += 2 * k) {
      for (int j = 0; j < k; j++) {
        num z = a[i + j + k] * roots[j + k];
        a[i + j + k] = a[i + j] - z;
        a[i + j] = a[i + j] + z;
      }
    }
  }
}

vector<num> fa, fb;

vector<int64_t> square(const vector<int>& a) {
  if (a.empty()) {
```

```
    return {};
  }
  int need = (int) a.size() + (int) a.size() - 1;
  int nbase = 1;
  while ((1 << nbase) < need) nbase++;
  ensure_base(nbase);
  int sz = 1 << nbase;
  if ((sz >> 1) > (int) fa.size()) {
    fa.resize(sz >> 1);
  }
  for (int i = 0; i < (sz >> 1); i++) {
    int x = (2 * i < (int) a.size() ? a[2 * i] : 0);
    int y = (2 * i + 1 < (int) a.size() ? a[2 * i + 1] : 0);
    fa[i] = num(x, y);
  }
  fft(fa, sz >> 1);
  num r(1.0 / (sz >> 1), 0.0);
  for (int i = 0; i <= (sz >> 2); i++) {
    int j = ((sz >> 1) - i) & ((sz >> 1) - 1);
    num fe = (fa[i] + conj(fa[j])) * num(0.5, 0);
    num fo = (fa[i] - conj(fa[j])) * num(0, -0.5);
    num aux = fe * fe + fo * fo * roots[(sz >> 1) + i] * roots[(sz >> 1) + i
        ];
    num tmp = fe * fo;
    fa[i] = r * (conj(aux) + num(0, 2) * conj(tmp));
    fa[j] = r * (aux + num(0, 2) * tmp);
  }
  fft(fa, sz >> 1);
  vector<int64_t> res(need);
  for (int i = 0; i < need; i++) {
    res[i] = llround(i % 2 == 0 ? fa[i >> 1].x : fa[i >> 1].y);
  }
  return res;
}

vector<int64_t> multiply(const vector<int>& a, const vector<int>& b) {
  if (a.empty() || b.empty()) {
    return {};
  }
  if (a == b) {
    return square(a);
  }
  int need = (int) a.size() + (int) b.size() - 1;
  int nbase = 1;
  while ((1 << nbase) < need) nbase++;
  ensure_base(nbase);
  int sz = 1 << nbase;
  if (sz > (int) fa.size()) {
    fa.resize(sz);
  }
  for (int i = 0; i < sz; i++) {
    int x = (i < (int) a.size() ? a[i] : 0);
    int y = (i < (int) b.size() ? b[i] : 0);
    fa[i] = num(x, y);
  }
  fft(fa, sz);
  num r(0, -0.25 / (sz >> 1));
  for (int i = 0; i <= (sz >> 1); i++) {
    int j = (sz - i) & (sz - 1);
    num z = (fa[j] * fa[j] - conj(fa[i] * fa[i])) * r;
    fa[j] = (fa[i] * fa[i] - conj(fa[j] * fa[j])) * r;
    fa[i] = z;
  }
  for (int i = 0; i < (sz >> 1); i++) {
    num A0 = (fa[i] + fa[i + (sz >> 1)]) * num(0.5, 0);
    num A1 = (fa[i] - fa[i + (sz >> 1)]) * num(0.5, 0) * roots[(sz >> 1) + i
        ];
    fa[i] = A0 + A1 * num(0, 1);
  }
  fft(fa, sz >> 1);
  vector<int64_t> res(need);
  for (int i = 0; i < need; i++) {
    res[i] = llround(i % 2 == 0 ? fa[i >> 1].x : fa[i >> 1].y);
  }
  return res;
}

vector<int> multiply_mod(const vector<int>& a, const vector<int>& b, int m)
    {
  if (a.empty() || b.empty()) {
    return {};
  }
  int eq = (a.size() == b.size() && a == b);
  int need = (int) a.size() + (int) b.size() - 1;
  int nbase = 0;
  while ((1 << nbase) < need) nbase++;
```

```
  ensure_base(nbase);
  int sz = 1 << nbase;
  if (sz > (int) fa.size()) {
    fa.resize(sz);
  }
  for (int i = 0; i < (int) a.size(); i++) {
    int x = (a[i] % m + m) % m;
    fa[i] = num(x & ((1 << 15) - 1), x >> 15);
  }
  fill(fa.begin() + a.size(), fa.begin() + sz, num {0, 0});
  fft(fa, sz);
  if (sz > (int) fb.size()) {
    fb.resize(sz);
  }
  if (eq) {
    copy(fa.begin(), fa.begin() + sz, fb.begin());
  } else {
    for (int i = 0; i < (int) b.size(); i++) {
      int x = (b[i] % m + m) % m;
      fb[i] = num(x & ((1 << 15) - 1), x >> 15);
    }
    fill(fb.begin() + b.size(), fb.begin() + sz, num {0, 0});
    fft(fb, sz);
  }
  dbl ratio = 0.25 / sz;
  num r2(0, -1);
  num r3(ratio, 0);
  num r4(0, -ratio);
  num r5(0, 1);
  for (int i = 0; i <= (sz >> 1); i++) {
    int j = (sz - i) & (sz - 1);
    num a1 = (fa[i] + conj(fa[j]));
    num a2 = (fa[i] - conj(fa[j])) * r2;
    num b1 = (fb[i] + conj(fb[j])) * r3;
    num b2 = (fb[i] - conj(fb[j])) * r4;
    if (i != j) {
      num c1 = (fa[j] + conj(fa[i]));
      num c2 = (fa[j] - conj(fa[i])) * r2;
      num d1 = (fb[j] + conj(fb[i])) * r3;
      num d2 = (fb[j] - conj(fb[i])) * r4;
      fa[i] = c1 * d1 + c2 * d2 * r5;
      fb[i] = c1 * d2 + c2 * d1;
    }
    fa[j] = a1 * b1 + a2 * b2 * r5;
    fb[j] = a1 * b2 + a2 * b1;
  }
  fft(fa, sz);
  fft(fb, sz);
  vector<int> res(need);
  for (int i = 0; i < need; i++) {
    int64_t aa = llround(fa[i].x);
    int64_t bb = llround(fb[i].x);
    int64_t cc = llround(fa[i].y);
    res[i] = static_cast<int>((aa + ((bb % m) << 15) + ((cc % m) << 30)) % m
        );
  }
  return res;
}

}  // namespace fft
```

6.9 FST.cpp

```
void fst(VI &a,bool inv) {
    for (int n=SZ(a),step=1;step<n;step*=2) {
        for (int i=0;i<n;i+=2*step) rep(j,i,i+step-1) {
            int &u=a[j],&v=a[j+step];
            tie(u,v)=
            inv?PII(v-u,u):PII(v,u+v); // AND
            inv?PII(v,u-v):PII(u+v,u); // OR
            PII(u+v,u-v); // XOR
        }
    }
    if (inv) for (auto &x : a) x/=SZ(a); // XOR only
}
VI conv(VI a,VI b) {
    fst(a,0),fst(b,0);
    rep(i,0,SZ(a)-1) a[i]=a[i]*b[i];
    fst(a,1); return a;
}
```

6.10 FWT.cpp

```
ll f[maxn], g[maxn], h[maxn];
int main() {
    for (int i = 0; i < n; i++) {
```

```
        for (int j = 0; j < bit(n); j++) {
            if ((j & bit(i)) == 0) {
                f[j] += f[j + bit(i)];
                g[j] += g[j + bit(i)];
            }
        }
    }
    for (int i = 0; i < bit(n); i++) {
        f[i] %= mod;
        g[i] %= mod;
        h[i] = f[i] * g[i] % mod;
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < bit(n); j++) {
            if ((j & bit(i)) == 0)
                h[j] -= h[j + bit(i)];
        }
    }
    for (int i = 0; i < bit(n); i++) {
        h[i] %= mod;
        if (h[i] < 0) h[i] += mod;
    }

    ll ans = 0;
    rep(i, 0, bit(n) - 1) ans ^= h[i];
    cout << ans << '\n';
}
```

6.11 gauss(合数).cpp

```
void gauss(int n) {
    int ans = 1;
    //rep(i,1,n) rep(j,1,n) p[i][j]%=mod;
    for (int i = 1; i <= n; i++) {
        for (int j = i + 1; j <= n; j++) {
            int x = i, y = j;
            while (p[x][i]) {
                int t = p[y][i] / p[x][i];
                for (int k = i; k <= n; k++)
                    p[y][k] = (p[y][k] - p[x][k] * t) % mod;
                swap(x, y);
            }
            if (x == i) {
                for (int k = i; k <= n; k++) swap(p[i][k], p[j][k]);
                ans = -ans;
            }
        }
    }
}
```

6.12 gauss.cpp

```
ll f[N][N];
ll v[N], a[N];
void gauss() {
    for (int i = 1; i <= n; i++) {
        for (int j = i; j <= n; j++) {
            if (f[j][i] > f[i][i]) {
                swap(v[i], v[j]);
                for (int k = 1; k <= n; k++)
                    swap(f[j][k], f[i][k]);
            }
        }
        for (int j = i + 1; j <= n; j++) {
            if (f[j][i]) {
                int delta = f[j][i] * fpow(f[i][i], mod - 2) % mod;
                for (int k = i; k <= n; k++) {
                    f[j][k] -= f[i][k] * delta % mod;
                    if (f[j][k] < 0)
                        f[j][k] += mod;
                }
                v[j] -= v[i] * delta % mod;
                if (v[j] < mod)
                    v[j] += mod;
            }
        }
    }
    for (int j = n; j > 0; j--) {
        for (int k = j + 1; k <= n; k++) {
            v[j] -= f[j][k] * a[k] % mod;
            if (v[j] < 0)
                v[j] += mod;
        }
        a[j] = v[j] * fpow(f[j][j], mod - 2) % mod;
    }
}
```

6.13 linearbasis.cpp

```
struct linear_base {
    ll w[64];
    ll zero = 0;
    ll tot = -1;
    void clear() {
        rep(i, 0, 63) w[i] = 0;
        zero = 0;
        tot = -1;
    }
    void insert(ll x) {
        for (int i = 62; i >= 0; i--) {
            if (x & bit(i))
                if (!w[i]) {w[i] = x; return;}
                else x ^= w[i];
        }
        zero++;
    }
    void build() {
        rep(i, 0, 63) rep(j, 0, i - 1) {
            if (w[i]&bit(j)) w[i] ^= w[j];
        }
        for (int i = 0; i <= 62; i++) {
            if (w[i] != 0) w[++tot] = w[i];
        }
    }
    ll qmax() {
        ll res = 0;
        for (int i = 62; i >= 0; i--) {
            res = max(res, res ^ w[i]);
        }
        return res;
    }
    bool check(ll x) {
        for (int i = 62; i >= 0; i--) {
            if (x & bit(i))
                if (!w[i]) return false;
                else x ^= w[i];
        }
        return true;
    }
    ll query(ll k) {
        ll res = 0;
        // if (zero) k-=1;
        // if (k >= bit(tot)) return -1;
        for (int i = tot; i >= 0; i--) {
            if (k & bit(i)) {
                res = max(res, res ^ w[i]);
            } else {
                res = min(res, res ^ w[i]);
            }
        }
        return res;
    }
};
```

6.14 lucas.cpp

```
ll fac[maxn], fnv[maxn];

ll binom(ll a, ll b) {
    if (b > a || b < 0) return 0;
    return fac[a] * fnv[a - b] % p * fnv[b] % p;
}

ll lucas(ll a, ll b, ll p) {
    ll ans = 1;
    while (a > 0 || b > 0) {
        ans = (ans * binom(a % p, b % p)) % p;
        a /= p, b /= p;
    }
    return ans;
}

int main() {
    cin >> p >> T;
    fac[0] = 1;
    rep(i, 1, p - 1) fac[i] = fac[i - 1] * i % p;
    fnv[p - 1] = powmod(fac[p - 1], p - 2, p);
    per(i, p - 2, 0) fnv[i] = fnv[i + 1] * (i + 1) % p;
    assert(fnv[0] == 1);
}
```

6.15 mathdiv.cpp

```
ll floor_div(ll x, ll y) {
```

```
    assert(y != 0);
    if (y < 0) {
        y = -y;
        x = -x;
    }
    if (x >= 0) return x / y;
    return (x + 1) / y - 1;
}
ll ceil_div(ll x, ll y) {
    assert(y != 0);
    if (y < 0) {
        y = -y;
        x = -x;
    }
    if (x <= 0) return x / y;
    return (x - 1) / y + 1;
}
```

6.16 matrix.cpp

```
template <typename T>
vector<vector<T>> operator*(const vector<vector<T>>& a, const vector<vector<
    T>>& b) {
  if (a.empty() || b.empty()) {
    return {{}};
  }
  vector<vector<T>> c(a.size(), vector<T>(b[0].size()));
  for (int i = 0; i < static_cast<int>(c.size()); i++) {
    for (int j = 0; j < static_cast<int>(c[0].size()); j++) {
      c[i][j] = 0;
      for (int k = 0; k < static_cast<int>(b.size()); k++) {
        c[i][j] += a[i][k] * b[k][j];
      }
    }
  }
  return c;
}

template <typename T>
vector<vector<T>>& operator*=(vector<vector<T>>& a, const vector<vector<T>>&
     b) {
  return a = a * b;
}

template <typename T, typename U>
vector<vector<T>> power(const vector<vector<T>>& a, const U& b) {
  assert(b >= 0);
  vector<U> binary;
  U bb = b;
  while (bb > 0) {
    binary.push_back(bb & 1);
    bb >>= 1;
  }
  vector<vector<T>> res(a.size(), vector<T>(a.size()));
  for (int i = 0; i < static_cast<int>(a.size()); i++) {
    res[i][i] = 1;
  }
  for (int j = (int)binary.size() - 1; j >= 0; j--) {
    res *= res;
    if (binary[j] == 1) {
      res *= a;
    }
  }
  return res;
}
```

6.17 matrixfast.cpp

```
Description: Basic operations on square matrices.
Usage: Matrix<int, 3> A;
A.d = {{{{1, 2, 3}}, {{4, 5, 6}}, {{7, 8, 9}}}};
vector<int> vec = {1, 2, 3};
vec = (A^N) * vec;

template<class T, int N> struct Matrix {
    typedef Matrix M;
    array<array<T, N>, N> d{};
    M operator*(const M& m) const {
        M a;
        rep(i, 0, N) rep(j, 0, N)
        rep(k, 0, N) a.d[i][j] += d[i][k] * m.d[k][j];
        return a;
    }
    vector<T> operator*(const vector<T>& vec) const {
        vector<T> ret(N);
        rep(i, 0, N) rep(j, 0, N) ret[i] += d[i][j] * vec[j];
```

```
        return ret;
    }
    M operator^(ll p) const {
        assert(p >= 0);
        M a, b(*this);
        rep(i, 0, N) a.d[i][i] = 1;
        while (p) {
            if (p & 1) a = a * b;
            b = b * b;
            p >>= 1;
        }
        return a;
    }
};
```

6.18 MillerRabbin pollard modmul.cpp

```
/*ModMulLL.h
Description: Calculate a•b mod c (or a
b mod c) for 0   a, b   c   7.2•10^18
Time: O (1) for modmul, O (log b) for modpow*/
/*ull modmul(ull a, ull b, ull M) {
    ll ret = a * b - M * ull(1.L / M * a * b);
    return ret + M * (ret < 0) - M * (ret >= (ll)M);
}
ull modpow(ull b, ull e, ull mod) {
    ull ans = 1;
    for (; e; b = modmul(b, b, mod), e /= 2)
        if (e & 1) ans = modmul(ans, b, mod);
    return ans;
}*/
ll modmul(ll a, ll b, ll m) {
    a %= m, b %= m;
    ll d = ((ldb)a * b / m);
    d = a * b - d * m;
    if (d >= m) d -= m;
    if (d < 0) d += m;
    return d;
}
ll modpow(ll a, ll b, ll p) {
    ll ans = 1;
    while (b) {
        if (b & 1) ans = modmul(ans, a, p);
        a = modmul(a, a, p); b >>= 1;
    } return ans;
}
/*MillerRabin.h
Description: Deterministic Miller-Rabin primality test. Guaranteed to
work for numbers up to 7 • 1018; for larger numbers, use Python and extend A
     randomly.
Time: 7 times the complexity of a^b mod c.*/
bool isPrime(ll n) {
    if (n < 2 || n % 6 % 4 != 1) return (n | 1) == 3;
    ll A[] = {2, 325, 9375, 28178, 450775, 9780504, 1795265022},
               s = __builtin_ctzll(n - 1), d = n >> s;
    for (ll a : A) { // ^ count trailing zeroes
        ll p = modpow(a % n, d, n), i = s;
        while (p != 1 && p != n - 1 && a % n && i--)
            p = modmul(p, p, n);
        if (p != n - 1 && i != s) return 0;
    }
    return 1;
}
/*Factor.h
Description: Pollard-rho randomized factorization algorithm. Returns
prime factors of a number, in arbitrary order (e.g. 2299 -> {11, 19, 11}).
Time: O(n^1/4), less for numbers with small factors.*/
ll pollard(ll n) {
    auto f = [n](ll x) { return modmul(x, x, n) + 1; };
    ll x = 0, y = 0, t = 30, prd = 2, i = 1, q;
    while (t++ % 40 || __gcd(prd, n) == 1) {
        if (x == y) x = ++i, y = f(x);
        if ((q = modmul(prd, max(x, y) - min(x, y), n))) prd = q;
        x = f(x), y = f(f(y));
    }
    return __gcd(prd, n);
}
vector<ll> factor(ll n) {
    if (n == 1) return {};
    if (isPrime(n)) return {n};
    ll x = pollard(n);
    auto l = factor(x), r = factor(n / x);
    l.insert(l.end(), all(r));
    return l;
}
```

6.19 ntt(polynomial).cpp

```cpp
#include<bits/stdc++.h>
using namespace std;

const int mod = 998244353;

inline void add(int &x, int y) {
  x += y;
  if (x >= mod) {
    x -= mod;
  }
}

inline void sub(int &x, int y) {
  x -= y;
  if (x < 0) {
    x += mod;
  }
}

inline int mul(int x, int y) {
  return (long long) x * y % mod;
}

inline int power(int x, int y) {
  int res = 1;
  for (; y; y >>= 1, x = mul(x, x)) {
    if (y & 1) {
      res = mul(res, x);
    }
  }
  return res;
}

inline int inv(int a) {
  a %= mod;
  if (a < 0) {
    a += mod;
  }
  int b = mod, u = 0, v = 1;
  while (a) {
    int t = b / a;
    b -= t * a;
    swap(a, b);
    u -= t * v;
    swap(u, v);
  }
  if (u < 0) {
    u += mod;
  }
  return u;
}

namespace ntt {
int base = 1, root = -1, max_base = -1;
vector<int> rev = {0, 1}, roots = {0, 1};

void init() {
  int temp = mod - 1;
  max_base = 0;
  while (temp % 2 == 0) {
    temp >>= 1;
    ++max_base;
  }
  root = 2;
  while (true) {
    if (power(root, 1 << max_base) == 1 && power(root, 1 << (max_base - 1))
        != 1) {
      break;
    }
    ++root;
  }
}

void ensure_base(int nbase) {
  if (max_base == -1) {
    init();
  }
  if (nbase <= base) {
    return;
  }
  assert(nbase <= max_base);
  rev.resize(1 << nbase);
  for (int i = 0; i < 1 << nbase; ++i) {
    rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (nbase - 1));
  }
```

```
  roots.resize(1 << nbase);
  while (base < nbase) {
    int z = power(root, 1 << (max_base - 1 - base));
    for (int i = 1 << (base - 1); i < 1 << base; ++i) {
      roots[i << 1] = roots[i];
      roots[i << 1 | 1] = mul(roots[i], z);
    }
    ++base;
  }
}

void dft(vector<int> &a) {
  int n = a.size(), zeros = __builtin_ctz(n);
  ensure_base(zeros);
  int shift = base - zeros;
  for (int i = 0; i < n; ++i) {
    if (i < rev[i] >> shift) {
      swap(a[i], a[rev[i] >> shift]);
    }
  }
  for (int i = 1; i < n; i <<= 1) {
    for (int j = 0; j < n; j += i << 1) {
      for (int k = 0; k < i; ++k) {
        int x = a[j + k], y = mul(a[j + k + i], roots[i + k]);
        a[j + k] = (x + y) % mod;
        a[j + k + i] = (x + mod - y) % mod;
      }
    }
  }
}

vector<int> multiply(vector<int> a, vector<int> b) {
  int need = a.size() + b.size() - 1, nbase = 0;
  while (1 << nbase < need) {
    ++nbase;
  }
  ensure_base(nbase);
  int sz = 1 << nbase;
  a.resize(sz);
  b.resize(sz);
  bool equal = a == b;
  dft(a);
  if (equal) {
    b = a;
  } else {
    dft(b);
  }
  int inv_sz = inv(sz);
  for (int i = 0; i < sz; ++i) {
    a[i] = mul(mul(a[i], b[i]), inv_sz);
  }
  reverse(a.begin() + 1, a.end());
  dft(a);
  a.resize(need);
  return a;
}

vector<int> inverse_new(const vector<int> &a) {
  assert(!a.empty());
  int n = (int) a.size();
  vector<int> b = {inv(a[0])};
  while ((int) b.size() < n) {
    vector<int> x(a.begin(), a.begin() + min(a.size(), b.size() << 1));
    x.resize(b.size() << 1);
    b.resize(b.size() << 1);
    vector<int> c = b;
    // NTT<T>::fft(c);
    // NTT<T>::fft(x);
    dft(c);
    dft(x);
    // Modular<T> inv = 1 / static_cast<Modular<T>>((int) x.size());
    int inv_sz = inv((int)x.size());
    for (int i = 0; i < (int) x.size(); i++) {
      // x[i] *= c[i] * inv;
      x[i] = mul(x[i], mul(c[i], inv_sz));
    }
    reverse(x.begin() + 1, x.end());
    // NTT<T>::fft(x);
    dft(x);
    rotate(x.begin(), x.begin() + (x.size() >> 1), x.end());
    fill(x.begin() + (x.size() >> 1), x.end(), 0);
    // NTT<T>::fft(x);
    dft(x);
    for (int i = 0; i < (int) x.size(); i++) {
      // x[i] *= c[i] * inv;
      x[i] = mul(x[i], mul(c[i], inv_sz));
```

```
    }
    reverse(x.begin() + 1, x.end());
    // NTT<T>::fft(x);
    dft(x);
    for (int i = 0; i < ((int) x.size() >> 1); i++) {
      // b[i + ((int) x.size() >> 1)] = -x[i];
      int t = 0; sub(t, x[i]);
      b[i + ((int) x.size() >> 1)] = t;
    }
  }
  b.resize(n);
  return b;
}

vector<int> inverse(vector<int> a) {
  int n = a.size(), m = (n + 1) >> 1;
  if (n == 1) {
    return vector<int>(1, inv(a[0]));
  } else {
    vector<int> b = inverse(vector<int>(a.begin(), a.begin() + m));
    int need = n << 1, nbase = 0;
    while (1 << nbase < need) {
      ++nbase;
    }
    ensure_base(nbase);
    int sz = 1 << nbase;
    a.resize(sz);
    b.resize(sz);
    dft(a);
    dft(b);
    int inv_sz = inv(sz);
    for (int i = 0; i < sz; ++i) {
      a[i] = mul(mul(mod + 2 - mul(a[i], b[i]), b[i]), inv_sz);
    }
    reverse(a.begin() + 1, a.end());
    dft(a);
    a.resize(n);
    return a;
  }
}
}

using ntt::multiply;
using ntt::inverse;

vector<int>& operator += (vector<int> &a, const vector<int> &b) {
  if (a.size() < b.size()) {
    a.resize(b.size());
  }
  for (int i = 0; i < b.size(); ++i) {
    add(a[i], b[i]);
  }
  return a;
}

vector<int> operator + (const vector<int> &a, const vector<int> &b) {
  vector<int> c = a;
  return c += b;
}

vector<int>& operator -= (vector<int> &a, const vector<int> &b) {
  if (a.size() < b.size()) {
    a.resize(b.size());
  }
  for (int i = 0; i < b.size(); ++i) {
    sub(a[i], b[i]);
  }
  return a;
}

vector<int> operator - (const vector<int> &a, const vector<int> &b) {
  vector<int> c = a;
  return c -= b;
}

vector<int>& operator *= (vector<int> &a, const vector<int> &b) {
  if (min(a.size(), b.size()) < 128) {
    vector<int> c = a;
    a.assign(a.size() + b.size() - 1, 0);
    for (int i = 0; i < c.size(); ++i) {
      for (int j = 0; j < b.size(); ++j) {
        add(a[i + j], mul(c[i], b[j]));
      }
    }
  } else {
    a = multiply(a, b);
```

```
  }
  return a;
}

vector<int> operator * (const vector<int> &a, const vector<int> &b) {
  vector<int> c = a;
  return c *= b;
}

vector<int>& operator /= (vector<int> &a, const vector<int> &b) {
  int n = a.size(), m = b.size();
  if (n < m) {
    a.clear();
  } else {
    vector<int> c = b;
    reverse(a.begin(), a.end());
    reverse(c.begin(), c.end());
    c.resize(n - m + 1);
    a *= inverse(c);
    a.erase(a.begin() + n - m + 1, a.end());
    reverse(a.begin(), a.end());
  }
  return a;
}

vector<int> operator / (const vector<int> &a, const vector<int> &b) {
  vector<int> c = a;
  return c /= b;
}

vector<int>& operator %= (vector<int> &a, const vector<int> &b) {
  int n = a.size(), m = b.size();
  if (n >= m) {
    vector<int> c = (a / b) * b;
    a.resize(m - 1);
    for (int i = 0; i < m - 1; ++i) {
      sub(a[i], c[i]);
    }
  }
  return a;
}

vector<int> operator % (const vector<int> &a, const vector<int> &b) {
  vector<int> c = a;
  return c %= b;
}

vector<int> derivative(const vector<int> &a) {
  int n = a.size();
  vector<int> b(n - 1);
  for (int i = 1; i < n; ++i) {
    b[i - 1] = mul(a[i], i);
  }
  return b;
}

vector<int> primitive(const vector<int> &a) {
  int n = a.size();
  vector<int> b(n + 1), invs(n + 1);
  for (int i = 1; i <= n; ++i) {
    invs[i] = i == 1 ? 1 : mul(mod - mod / i, invs[mod % i]);
    b[i] = mul(a[i - 1], invs[i]);
  }
  return b;
}

vector<int> logarithm(const vector<int> &a) {
  vector<int> b = primitive(derivative(a) * inverse(a));
  b.resize(a.size());
  return b;
}

vector<int> exponent(const vector<int> &a) {
  vector<int> b(1, 1);
  while (b.size() < a.size()) {
    vector<int> c(a.begin(), a.begin() + min(a.size(), b.size() << 1));
    add(c[0], 1);
    vector<int> old_b = b;
    b.resize(b.size() << 1);
    c -= logarithm(b);
    c *= old_b;
    for (int i = b.size() >> 1; i < b.size(); ++i) {
      b[i] = c[i];
    }
  }
  b.resize(a.size());
```

```
  return b;
}

vector<int> power(vector<int> a, int m) {
  int n = a.size(), p = -1;
  vector<int> b(n);
  for (int i = 0; i < n; ++i) {
    if (a[i]) {
      p = i;
      break;
    }
  }
  if (p == -1) {
    b[0] = !m;
    return b;
  }
  if ((long long) m * p >= n) {
    return b;
  }
  int mu = power(a[p], m), di = inv(a[p]);
  vector<int> c(n - m * p);
  for (int i = 0; i < n - m * p; ++i) {
    c[i] = mul(a[i + p], di);
  }
  c = logarithm(c);
  for (int i = 0; i < n - m * p; ++i) {
    c[i] = mul(c[i], m);
  }
  c = exponent(c);
  for (int i = 0; i < n - m * p; ++i) {
    b[i + m * p] = mul(c[i], mu);
  }
  return b;
}

vector<int> sqrt(const vector<int> &a) {
  vector<int> b(1, 1);
  while (b.size() < a.size()) {
    vector<int> c(a.begin(), a.begin() + min(a.size(), b.size() << 1));
    vector<int> old_b = b;
    b.resize(b.size() << 1);
    c *= inverse(b);
    for (int i = b.size() >> 1; i < b.size(); ++i) {
      b[i] = mul(c[i], (mod + 1) >> 1);
    }
  }
  b.resize(a.size());
  return b;
}

vector<int> multiply_all(int l, int r, vector<vector<int>> &all) {
  if (l > r) {
    return vector<int>();
  } else if (l == r) {
    return all[l];
  } else {
    int y = (l + r) >> 1;
    return multiply_all(l, y, all) * multiply_all(y + 1, r, all);
  }
}

vector<int> evaluate(const vector<int> &f, const vector<int> &x) {
  int n = x.size();
  if (!n) {
    return vector<int>();
  }
  vector<vector<int>> up(n * 2);
  for (int i = 0; i < n; ++i) {
    up[i + n] = vector<int> {(mod - x[i]) % mod, 1};
  }
  for (int i = n - 1; i; --i) {
    up[i] = up[i << 1] * up[i << 1 | 1];
  }
  vector<vector<int>> down(n * 2);
  down[1] = f % up[1];
  for (int i = 2; i < n * 2; ++i) {
    down[i] = down[i >> 1] % up[i];
  }
  vector<int> y(n);
  for (int i = 0; i < n; ++i) {
    y[i] = down[i + n][0];
  }
  return y;
}

vector<int> interpolate(const vector<int> &x, const vector<int> &y) {
```

```
  int n = x.size();
  vector<vector<int>> up(n * 2);
  for (int i = 0; i < n; ++i) {
    up[i + n] = vector<int> {(mod - x[i]) % mod, 1};
  }
  for (int i = n - 1; i; --i) {
    up[i] = up[i << 1] * up[i << 1 | 1];
  }
  vector<int> a = evaluate(derivative(up[1]), x);
  for (int i = 0; i < n; ++i) {
    a[i] = mul(y[i], inv(a[i]));
  }
  vector<vector<int>> down(n * 2);
  for (int i = 0; i < n; ++i) {
    down[i + n] = vector<int>(1, a[i]);
  }
  for (int i = n - 1; i; --i) {
    down[i] = down[i << 1] * up[i << 1 | 1] + down[i << 1 | 1] * up[i << 1];
  }
  return down[1];
}

int main() {

}
```

6.20 PrimitiveRoot.cpp

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

int pf[1010];

ll powmod(ll a, ll b, ll mod) {
  ll res = 1;
  a %= mod;
  for (; b; b >>= 1) {
    if (b & 1) res = res * a % mod;
    a = a * a % mod;
  }
  return res;
}

int main() {
  int T;
  scanf("%d", &T);
  for (int tc = 0; tc < T; tc++) {
    int p;
    scanf("%d", &p);
    int m = p - 1, t = 0;
    for (int i = 2; i * i <= m; i++) {
      if (m % i == 0) {
        pf[t++] = i;
        while (m % i == 0) m /= i;
      }
    }
    if (m != 1) pf[t++] = m;
    for (int g = 1; g < p; g++) {
      bool valid = true;
      for (int i = 0; i < t; i++)
        if (powmod(g, (p - 1) / pf[i], p) == 1) {
          valid = false;
          break;
        }
      if (valid) {
        printf("%d\n", g);
        break;
      }
    }
  }
}
```

6.21 simplex.cpp

```
/**
 * Author: Stanford
 * Source: Stanford Notebook
 * License: MIT
 * Description: Solves a general linear maximization problem: maximize $c^T
     x$ subject to $Ax \le b$, $x \ge 0$.
 * Returns -inf if there is no solution, inf if there are arbitrarily good
     solutions, or the maximum value of $c^T x$ otherwise.
 * The input vector is set to an optimal $x$ (or in the unbounded case, an
     arbitrary solution fulfilling the constraints).
 * Numerical stability is not guaranteed. For better performance, define
```

```
    variables such that $x = 0$ is viable.
 * Usage:
 * vvd A = {{1,-1}, {-1,1}, {-1,-2}};
 * vd b = {1,1,-4}, c = {-1,-1}, x;
 * T val = LPSolver(A, b, c).solve(x);
 * Time: O(NM * \#pivots), where a pivot may be e.g. an edge relaxation. O(2^n) in the general case.
 * Status: seems to work?
 */

typedef long double T; // long double, Rational, double + mod<P>...
typedef vector<T> vd;
typedef vector<vd> vvd;

const T eps = 1e-8, inf = 1/.0;
#define MP make_pair
#define ltj(X) if(s == -1 || MP(X[j],N[j]) < MP(X[s],N[s])) s=j

struct LPSolver {
    int m, n;
    vector<int> N, B;
    vvd D;

    LPSolver(const vvd& A, const vd& b, const vd& c) :
        m(b.size()), n(c.size()), N(n+1), B(m), D(m+2, vd(n+2)) {
            for(int i = 0; i < m; i++){
                for(int j = 0; j < n; j++){
                    D[i][j] = A[i][j];
                }
            }
            for(int i = 0; i < m; i++){
                B[i] = n+i; D[i][n] = -1; D[i][n+1] = b[i];
            }
            for(int j = 0; j < n; j++){
                N[j] = j; D[m][j] = -c[j];
            }
            N[n] = -1; D[m+1][n] = 1;
        }

    void pivot(int r, int s) {
        T *a = D[r].data(), inv = 1 / a[s];
        for(int i = 0; i < m+2; i++){
            if (i != r && abs(D[i][s]) > eps) {
                T *b = D[i].data(), inv2 = b[s] * inv;
                for(int j = 0; j < n+2; j++){
                    b[j] -= a[j] * inv2;
                }
                b[s] = a[s] * inv2;
            }
        }
        for(int j = 0; j < n+2; j++){
            if (j != s) D[r][j] *= inv;
        }
        for(int i = 0; i < m+2; i++){
            if (i != r) D[i][s] *= -inv;
        }
        D[r][s] = inv;
        swap(B[r], N[s]);
    }

    bool simplex(int phase) {
        int x = m + phase - 1;
        for (;;) {
            int s = -1;
            for(int j = 0; j < n+1; j++){
                if (N[j] != -phase) ltj(D[x]);
            }
            if (D[x][s] >= -eps) return true;
            int r = -1;
            for(int i = 0; i < m; i++){
                if (D[i][s] <= eps) continue;
                if (r == -1 || MP(D[i][n+1] / D[i][s], B[i])
                        < MP(D[r][n+1] / D[r][s], B[r])) r = i;
            }
            if (r == -1) return false;
            pivot(r, s);
        }
    }

    T solve(vd &x) {
        int r = 0;
        for(int i = 1; i < m; i++){
            if (D[i][n+1] < D[r][n+1]) r = i;
        }
        if (D[r][n+1] < -eps) {
            pivot(r, n);
```

```
            if (!simplex(2) || D[m+1][n+1] < -eps) return -inf;
            for(int i = 0; i < m; i++) if (B[i] == -1) {
                int s = 0;
                for(int j = 1; j < n+1; j++){
                    ltj(D[i]);
                }
                pivot(i, s);
            }
        }
        bool ok = simplex(1); x = vd(n);
        for(int i = 0; i < m; i++){
            if (B[i] < n) x[B[i]] = D[i][n+1];
        }
        return ok ? D[m][n+1] : inf;
    }
};
```

6.22 区间互质.cpp

```
int p[N / 5], num;
void prime(int n) {
    num = 0;
    for (int i = 2; i * i <= n; i++) {
        if ((n % i) == 0) {
            p[++num] = i;
            while ((n % i) == 0) n /= i;
        }
    }
    if (n > 1) p[++num] = n;
}
ll solve(ll r, int k) {
    prime(k);
    ll res = 0;
    for (int i = 1; i < (1 << num); i++) {
        int k = 0;
        ll div = 1;
        for (int j = 1; j <= num; j++) {
            if (i & (1 << (j - 1))) {
                k++;
                div *= p[j];
            }
        }
        if (k % 2)
            res += r / div;
        else
            res -= r / div;
    }
    return r - res;
}
ll que(ll L, ll R, ll k) {
    return solve(R, k) - solve(L - 1, k);
}
```

6.23 幂转下降幂 (求幂和).cpp

```
ll comb[N][N];
ll s[maxn], inv[maxn], p;
// 1^k+2^k+...+n^k
void solve() {
    cin >> k >> n >> p;
    rep(i, 0, k + 1) {
        comb[i][0] = comb[i][i] = 1;
        rep(j, 1, i - 1) {
            comb[i][j] = (comb[i - 1][j - 1] + comb[i - 1][j]) % p;
        }
    }
    inv[1] = 1;
    rep(i, 2, k + 1) inv[i] = (p - p / i) * inv[p % i] % p;
    assert(inv[k] * k % p == 1);

    ll pw = 1;
    // (k+1)*S[k]=(n+1)^(k+1)-[0-k-1](k+1,j)*S[j]-1
    rep(i, 0, k) {
        pw = pw * (n + 1) % p;
        s[i] = (pw - 1 + p) % p;
        rep(j, 0, i - 1) {
            s[i] = (s[i] - comb[i + 1][j] * s[j] % p + p) % p;
        }
        s[i] = s[i] * inv[i + 1] % p;
    }
    cout << s[k] << '\n';
}
```

6.24 扩展欧拉定理.cpp

```
// mod {min(b, b % phi + phi)}
```

```
ll calc(ll p) {
    if (p == 1) return 0;
    int phi = p, q = p;
    for (int i = 2; i * i <= p; i++) {
        if (q % i == 0) {
            phi = phi / i * (i - 1);
            while (q % i == 0) q /= i;
        }
    }
    if (q != 1) phi = phi / q * (q - 1);
    return powmod(2, calc(phi) + phi, p);
}
```

6.25 拉格朗日插值.cpp

```
// k阶多项式(需要k+1个点)
// 求在点n上的值
// O(k)
ll lagrange(ll n, int k) {
    vector<ll> x(k + 5), y(k + 5);
    rep(i, 1, k + 1) {
        x[i] = i;
        // y[i]=(y[i-1]+powmod(i,k-1,mod))%mod;
    }
    if (n <= k + 1) return y[n];

    vector<ll> fac(k + 5);
    fac[0] = 1;
    ll coe = 1;
    rep(i, 1, k + 4) fac[i] = fac[i - 1] * i % mod;
    rep(i, 1, k + 1) coe = coe * (n - i + mod) % mod;
    ll ans = 0;
    rep(i, 1, k + 1) {
        ll sgn = (((k + 1 - i) % 2) ? -1 : 1);
        ll f1 = powmod(fac[i - 1] * fac[k + 1 - i] % mod, mod - 2, mod);
        ll f2 = powmod(n - i, mod - 2, mod);
        ans += sgn * coe * f1 % mod * f2 % mod * y[i] % mod;
        ans = (ans + mod) % mod;
    }
    return ans;
}
```

6.26 整除分块.cpp

```
void solve() {
    u64 ans = 0;
    cin >> n;
    for (ll l = 1; l <= n; l++) {
        ll d = n / l, r = n / d;
        ans += (l + r) * (r - l + 1) / 2 * d;
        l = r;
    }
}
```

6.27 枚举子集.cpp

```
void solve() {
    f[0] = 1;
    for (int i = 1; i < (1ll << n); i++) {
        int t = i;
        ll res = 0;
        while (true) {
            if (t == 0) break;
            t = (t - 1)&i;
            res = (res + f[t]) % mod;
        }
        f[i] = res * i;
    }
}
```

6.28 枚举超集.cpp

```
void solve() {
    for (int i = 1; i < (1ll << n); i++) {
        int t = i;
        while (true) {
            t = (t + 1) | i;
            if (t == bit(n) - 1) break;
        }
    }
}
```

6.29 狄利克雷卷积.cpp

```
const int N = 1010000;
```

```
int p[N], pr[N / 5], n, tot;
unsigned int A, B, C, mu[N], f[N], g[N];

inline unsigned int rng61() {
    A ^= A << 16;
    A ^= A >> 5;
    A ^= A << 1;
    unsigned int t = A;
    A = B;
    B = C;
    C ^= t ^ A;
    return C;
}

int main() {
    scanf("%d%u%u%u", &n, &A, &B, &C);
    for (int i = 1; i <= n; i++)
        f[i] = rng61();

    p[1] = 1; mu[1] = 1;
    for (int i = 2; i <= n; i++) {
        if (!p[i]) p[i] = i, mu[i] = (uint)-1, pr[++tot] = i;
        for (int j = 1; j <= tot && pr[j] * i <= n; j++) {
            p[i * pr[j]] = pr[j];
            if (p[i] == pr[j]) {
                mu[i * pr[j]] = 0;
                break;
            } else {
                mu[i * pr[j]] = (uint)-mu[i];
            }
        }
    }
    for (int d1 = 1; d1 <= n; d1++)
        for (int d2 = 1; d1 * d2 <= n; d2++)
            g[d1 * d2] += f[d1] * mu[d2];
    uint ans = 0;
    for (int i = 1; i <= n; i++) ans ^= g[i];
    printf("%u\n", ans);
}
```

6.30 线性筛常见积性函数.cpp

```
const int N = 20010000;
int p[N], pr[N / 5], n, pe[N], tot;
uint f[N], a, b, ans;

void compute(int n, function<void(int)> calcpe) {
    ans = 0;
    f[1] = 1;
    for (int i = 2; i <= n; i++) {
        if (i == pe[i])
            calcpe(i);
        else
            f[i] = f[pe[i]] * f[i / pe[i]];
    }
    for (uint i = 1; i <= n; i++) {
        ans ^= (a * i * f[i] + b);
    }
    printf("%u\n", ans);
}

int main() {
    scanf("%d%u%u", &n, &a, &b);
    p[1] = 1;
    for (int i = 2; i <= n; i++) {
        if (!p[i]) p[i] = i, pe[i] = i, pr[++tot] = i;
        for (int j = 1; j <= tot && pr[j] * i <= n; j++) {
            p[i * pr[j]] = pr[j];
            if (p[i] == pr[j]) {
                pe[i * pr[j]] = pe[i] * pr[j];
                break;
            } else {
                pe[i * pr[j]] = pr[j];
            }
        }
    }
    // 因子个数，因子和，欧拉函数，莫比乌斯函数
    compute(n, [&](int x) {
        f[x] = f[x / p[x]] + 1;
    });

    compute(n, [&](int x) {
        f[x] = f[x / p[x]] + x;
    });

    compute(n, [&](int x) {
```

```
        f[x] = x / p[x] * (p[x] - 1);
    });

    compute(n, [&](int x) {
        f[x] = x == p[x] ? -1 : 0;
    });
}
```

6.31 莫比乌斯反演 gcd 常见结论.cpp

```
// u * 1 = e, phi * 1 = id, phi = id * u
const int N = 10100000, M = 10000000;
int p[N], pr[N / 5], n, tot;
int mu[N], smu[N];

int main() {
    p[1] = 1; mu[1] = 1;
    for (int i = 2; i <= M; i++) {
        if (!p[i]) p[i] = i, mu[i] = -1, pr[++tot] = i;
        for (int j = 1; j <= tot && pr[j] * i <= M; j++) {
            p[i * pr[j]] = pr[j];
            if (p[i] == pr[j]) {
                mu[i * pr[j]] = 0;
                break;
            } else {
                mu[i * pr[j]] = -mu[i];
            }
        }
    }
    for (int i = 1; i <= M; i++)
        smu[i] = smu[i - 1] + mu[i];
    int T;
    scanf("%d", &T);
    for (int tc = 0; tc < T; tc++) {
        int n, m;
        scanf("%d%d", &n, &m);
        if (n > m) swap(n, m);
        ll ans = 0;
        for (int l = 1; l <= n; l++) {
            int n1 = n / l, m1 = m / l;
            int r = min(n / n1, m / m1);
            // l ... r
            ans += 1ll * (smu[r] - smu[l - 1]) * n1 * m1;
            l = r;
        }
        printf("%lld\n", ans);
    }
}
```

7 String

7.1 ACAM.cpp

```
const int AC_SIGMA = 26, AC_V = 26, AC_N = 810000;
struct AC_automaton {
    struct node {
        node *go[AC_V], *fail, *f;
// declare extra variables:
    } pool[AC_N], *cur, *root, *q[AC_N];
    node* newnode() {
        node *p = cur++;
// init extra variables:
        return p;
    }
// CALL init() and CHECK all const variables:
    void init() { cur = pool; root = newnode(); }
    node* append(node *p, int w) {
        if (!p->go[w]) p->go[w] = newnode(), p->go[w]->f = p;
        return p->go[w];
    }
    void build() {
        int t = 0;
        q[t++] = root;
        root->fail = root;
        rep(i, 0, AC_SIGMA - 1) if (root->go[i]) {
            q[t++] = root->go[i];
            root->go[i]->fail = root;
        } else {
            root->go[i] = root;
        }
        rep(i, 1, t - 1) {
            node *u = q[i];
            rep(j, 0, AC_SIGMA - 1) if (u->go[j]) {
                u->go[j]->fail = u->fail->go[j];
                q[t++] = u->go[j];
```

```
            } else {
                u->go[j] = u->fail->go[j];
            }
        }
    }
} ac;
typedef AC_automaton::node ACnode;

const int M = 2, N = 2.1e5;
struct node {
    node *son[M], *go[M], *fail;
    int cnt, vis, ins;
} pool[N], *cur = pool, *q[N], *root;

node *newnode() { return cur++; }
int t, n;

void build() {
    t = 0;
    q[t++] = root;
    for (int i = 0; i < t; i++) {
        node *u = q[i];
        for (int j = 0; j < M; j++) {
            if (u->son[j]) {
                u->go[j] = u->son[j];
                if (u != root)
                    u->go[j]->fail = u->fail->go[j];
                else
                    u->go[j]->fail = root;
                q[t++] = u->son[j];
            } else {
                if (u != root)
                    u->go[j] = u->fail->go[j];
                else
                    u->go[j] = root;
            }
        }
    }
}

void insert(string &s) {
    node *cur = root;
    for (auto c : s) {
        int w = c - '0';
        if (!cur->son[w]) {
            cur->son[w] = newnode();
        }
        cur = cur->son[w];
    }
    cur->cnt = 1;
}
```

7.2 hash61.cpp

```
struct hash61 {
  static const uint64_t md = (1LL << 61) - 1;
  static uint64_t step;
  static vector<uint64_t> pw;

  uint64_t addmod(uint64_t a, uint64_t b) const {
    a += b;
    if (a >= md) a -= md;
    return a;
  }

  uint64_t submod(uint64_t a, uint64_t b) const {
    a += md - b;
    if (a >= md) a -= md;
    return a;
  }

  uint64_t mulmod(uint64_t a, uint64_t b) const {
    uint64_t l1 = (uint32_t) a, h1 = a >> 32, l2 = (uint32_t) b, h2 = b >>
        32;
    uint64_t l = l1 * l2, m = l1 * h2 + l2 * h1, h = h1 * h2;
    uint64_t ret = (l & md) + (l >> 61) + (h << 3) + (m >> 29) + (m << 35 >>
         3) + 1;
    ret = (ret & md) + (ret >> 61);
    ret = (ret & md) + (ret >> 61);
    return ret - 1;
  }

  void ensure_pw(int sz) {
    int cur = (int) pw.size();
    if (cur < sz) {
      pw.resize(sz);
```

```
      for (int i = cur; i < sz; i++) {
        pw[i] = mulmod(pw[i - 1], step);
      }
    }
  }

  vector<uint64_t> pref;
  int n;

  template<typename T>
  hash61(const T& s) {
    n = (int) s.size();
    ensure_pw(n + 1);
    pref.resize(n + 1);
    pref[0] = 1;
    for (int i = 0; i < n; i++) {
      pref[i + 1] = addmod(mulmod(pref[i], step), s[i]);
    }
  }

  inline uint64_t operator()(const int from, const int to) const {
    assert(0 <= from && from <= to && to <= n - 1);
    return submod(pref[to + 1], mulmod(pref[from], pw[to - from + 1]));
  }
};

uint64_t hash61::step = (md >> 2) + rng() % (md >> 1);
vector<uint64_t> hash61::pw = vector<uint64_t>(1, 1);
```

7.3 kmp.cpp

```
template <typename T>
vector<int> kmp_table(int n, const T &s) {
  vector<int> p(n, 0);
  int k = 0;
  for (int i = 1; i < n; i++) {
    while (k > 0 && !(s[i] == s[k])) {
      k = p[k - 1];
    }
    if (s[i] == s[k]) {
      k++;
    }
    p[i] = k;
  }
  return p;
}

template <typename T>
vector<int> kmp_table(const T &s) {
  return kmp_table((int) s.size(), s);
}

template <typename T>
vector<int> kmp_search(int n, const T &s, int m, const T &w, const vector<
    int> &p) {
  assert(n >= 1 && (int) p.size() == n);
  vector<int> res;
  int k = 0;
  for (int i = 0; i < m; i++) {
    while (k > 0 && (k == n || !(w[i] == s[k]))) {
      k = p[k - 1];
    }
    if (w[i] == s[k]) {
      k++;
    }
    if (k == n) {
      res.push_back(i - n + 1);
    }
  }
  return res;
  // returns 0-indexed positions of occurrences of s in w
}

template <typename T>
vector<int> kmp_search(const T &s, const T &w, const vector<int> &p) {
  return kmp_search((int) s.size(), s, (int) w.size(), w, p);
}
```

7.4 manacherfast.cpp

```
template <typename T>
vector<int> manacher(int n, const T &s) {
  if (n == 0) {
    return vector<int>();
  }
  vector<int> res(2 * n - 1, 0);
```

```
  int l = -1, r = -1;
  for (int z = 0; z < 2 * n - 1; z++) {
    int i = (z + 1) >> 1;
    int j = z >> 1;
    int p = (i >= r ? 0 : min(r - i, res[2 * (l + r) - z]));
    while (j + p + 1 < n && i - p - 1 >= 0) {
      if (!(s[j + p + 1] == s[i - p - 1])) {
        break;
      }
      p++;
    }
    if (j + p > r) {
      l = i - p;
      r = j + p;
    }
    res[z] = p;
  }
  return res;
  // res[2 * i] = odd radius in position i
  // res[2 * i + 1] = even radius between positions i and i + 1
  // s = "abaa" -> res = {0, 0, 1, 0, 0, 1, 0}
  // in other words, for every z from 0 to 2 * n - 2:
  // calculate i = (z + 1) >> 1 and j = z >> 1
  // now there is a palindrome from i - res[z] to j + res[z]
  // (watch out for i > j and res[z] = 0)
}

template <typename T>
vector<int> manacher(const T &s) {
  return manacher((int) s.size(), s);
}
```

7.5 MinRotation.cpp

```
Description: Finds the lexicographically smallest rotation of a string.
Usage: rotate(v.begin(), v.begin() + minRotation(v), v.end());
Time: O (N)

int minRotation(string s) {
    int a = 0, N = sz(s); s += s;
    rep(b, 0, N) rep(k, 0, N) {
        if (a + k == b || s[a + k] < s[b + k]) {b += max(0, k - 1); break;}
        if (s[a + k] > s[b + k]) { a = b; break; }
    }
    return a;
}
```

7.6 PAM.cpp

```
struct PAM {
    struct T {
        array<int, 10> tr;
        int fail, len, tag;
        T() : fail(0), len(0), tag(0) {
            tr.fill(0);
        }
    };
    vector<T> t;
    vector<int> stk;
    int newnode(int len) {
        t.emplace_back();
        t.back().len = len;
        return (int)t.size() - 1;
    }
    PAM() : t(2) {
        t[0].fail = 1, t[0].len = 0;
        t[1].fail = 0, t[1].len = -1;
        stk.push_back(-1);
    }
    int getfail(int v) {
        while (stk.end()[-2 - t[v].len] != stk.back()) {
            v = t[v].fail;
        }
        return v;
    }
    int insert(int lst, int c, int td) {
        stk.emplace_back(c);
        int x = getfail(lst);
        if (!t[x].tr[c]) {
            int u = newnode(t[x].len + 2);
            t[u].fail = t[getfail(t[x].fail)].tr[c];
            t[x].tr[c] = u;
        }
        t[t[x].tr[c]].tag += td;
        return t[x].tr[c];
    }
```

```
    int build(int n) {
        int ans = 0;
        for (int i = (int)t.size() - 1; i > 1; i--) {
            t[t[i].fail].tag += t[i].tag;
            if (t[i].len > n) {
                continue;
            }
            ans = (ans + 1ll * t[i].tag * t[i].tag % M * t[i].len) % M;
        }
        return ans;
    }
};
```

7.7 rollingHash.cpp

```
typedef pair<int,int> hashv;
const ll mod1=1000000007;
const ll mod2=1000000009;

// prefixSum trick for high dimensions

hashv operator + (hashv a,hashv b) {
    int c1=a.fi+b.fi,c2=a.se+b.se;
    if (c1>=mod1) c1-=mod1;
    if (c2>=mod2) c2-=mod2;
    return mp(c1,c2);
}

hashv operator - (hashv a,hashv b) {
    int c1=a.fi-b.fi,c2=a.se-b.se;
    if (c1<0) c1+=mod1;
    if (c2<0) c2+=mod2;
    return mp(c1,c2);
}

hashv operator * (hashv a,hashv b) {
    return mp(1ll*a.fi*b.fi%mod1,1ll*a.se*b.se%mod2);
}
```

7.8 SA.cpp

```
#include <bits/stdc++.h>
using namespace std;

const int N = 101000;
char s[N];
int sa[N], rk[N], ht[N], n;
// 0-based sa 表示第i大的为哪个, rk 表示第i个后缀第几大
// ht表示 lcp(sa[i], sa[i-1])
void buildSA(char *s, int *sa, int *rk, int *ht, int n, int m = 128) {
    static int x[N], y[N], c[N];
    s[n] = 0;
    for (int i = 0; i < m; i++) c[i] = 0;
    for (int i = 0; i < n; i++) c[x[i] = s[i]]++;
    for (int i = 1; i < m; i++) c[i] += c[i - 1];
    for (int i = n - 1; i >= 0; i--) sa[--c[x[i]]] = i;
    for (int k = 1; k < n; k <<= 1) {
        int p=0;
        for (int i = n - 1; i >= n - k; i--) y[p++] = i;
        for (int i = 0; i < n; i++) if (sa[i] >= k) y[p++] = sa[i] - k;
        for (int i = 0; i < m; i++) c[i] = 0;
        for (int i = 0; i < n; i++) c[x[y[i]]]++;
        for (int i = 1; i < m; i++) c[i] += c[i - 1];
        for (int i = n - 1; i >= 0; i--) sa[--c[x[y[i]]]] = y[i];
        swap(x, y);
        p = 1; x[sa[0]] = 0; y[n] = -1;
        for (int i = 1; i < n; i++) {
            if (y[sa[i - 1]] == y[sa[i]] && y[sa[i - 1] + k] == y[sa[i] + k
                ])
                x[sa[i]] = p - 1;
            else
                x[sa[i]] = p++;
        }
        if (p == n) break;
        m = p;
    }
    for (int i = 0; i < n; i++) rk[sa[i]] = i;
    int k = 0;
    for (int i = 0; i < n; i++) {
        k = max(k - 1, 0);
        if (rk[i] == 0) continue;
        int j = sa[rk[i] - 1];
        while (s[i + k] == s[j + k]) k++;
        ht[rk[i]] = k;
    }
}
```

```
45
46  int LCP(int u, int v) {
47      if (u == v) return n - u;
48      if (rk[u] > rk[v]) swap(u, v);
49      // RMQ(ht, rk[u] + 1, rk[v])
50  }
51
52  int main() {
53      scanf("%s", s);
54      n = strlen(s);
55      buildSA(s, sa, rk, ht, n);
56      for (int i = 0; i < n; i++) printf("%d␣", sa[i] + 1); puts("");
57      for (int i = 1; i < n; i++) printf("%d␣", ht[i]); puts("");
58  }
```

7.9 SAfast.cpp

```
1   template <typename T>
2   vector<int> suffix_array(int n, const T &s, int char_bound) {
3     vector<int> a(n);
4     if (n == 0) {
5       return a;
6     }
7     if (char_bound != -1) {
8       vector<int> aux(char_bound, 0);
9       for (int i = 0; i < n; i++) {
10        aux[s[i]]++;
11      }
12      int sum = 0;
13      for (int i = 0; i < char_bound; i++) {
14        int add = aux[i];
15        aux[i] = sum;
16        sum += add;
17      }
18      for (int i = 0; i < n; i++) {
19        a[aux[s[i]]++] = i;
20      }
21    } else {
22      iota(a.begin(), a.end(), 0);
23      sort(a.begin(), a.end(), [&s](int i, int j) { return s[i] < s[j]; });
24    }
25    vector<int> sorted_by_second(n);
26    vector<int> ptr_group(n);
27    vector<int> new_group(n);
28    vector<int> group(n);
29    group[a[0]] = 0;
30    for (int i = 1; i < n; i++) {
31      group[a[i]] = group[a[i - 1]] + (!(s[a[i]] == s[a[i - 1]]));
32    }
33    int cnt = group[a[n - 1]] + 1;
34    int step = 1;
35    while (cnt < n) {
36      int at = 0;
37      for (int i = n - step; i < n; i++) {
38        sorted_by_second[at++] = i;
39      }
40      for (int i = 0; i < n; i++) {
41        if (a[i] - step >= 0) {
42          sorted_by_second[at++] = a[i] - step;
43        }
44      }
45      for (int i = n - 1; i >= 0; i--) {
46        ptr_group[group[a[i]]] = i;
47      }
48      for (int i = 0; i < n; i++) {
49        int x = sorted_by_second[i];
50        a[ptr_group[group[x]]++] = x;
51      }
52      new_group[a[0]] = 0;
53      for (int i = 1; i < n; i++) {
54        if (group[a[i]] != group[a[i - 1]]) {
55          new_group[a[i]] = new_group[a[i - 1]] + 1;
56        } else {
57          int pre = (a[i - 1] + step >= n ? -1 : group[a[i - 1] + step]);
58          int cur = (a[i] + step >= n ? -1 : group[a[i] + step]);
59          new_group[a[i]] = new_group[a[i - 1]] + (pre != cur);
60        }
61      }
62      swap(group, new_group);
63      cnt = group[a[n - 1]] + 1;
64      step <<= 1;
65    }
66    return a;
67  }
68
69  template <typename T>
```

```cpp
vector<int> suffix_array(const T &s, int char_bound) {
  return suffix_array((int) s.size(), s, char_bound);
}

template <typename T>
vector<int> build_lcp(int n, const T &s, const vector<int> &sa) {
  assert((int) sa.size() == n);
  vector<int> pos(n);
  for (int i = 0; i < n; i++) {
    pos[sa[i]] = i;
  }
  vector<int> lcp(max(n - 1, 0));
  int k = 0;
  for (int i = 0; i < n; i++) {
    k = max(k - 1, 0);
    if (pos[i] == n - 1) {
      k = 0;
    } else {
      int j = sa[pos[i] + 1];
      while (i + k < n && j + k < n && s[i + k] == s[j + k]) {
        k++;
      }
      lcp[pos[i]] = k;
    }
  }
  return lcp;
}

template <typename T>
vector<int> build_lcp(const T &s, const vector<int> &sa) {
  return build_lcp((int) s.size(), s, sa);
}
```

7.10 SAM.cpp

```cpp
struct SAM {
    static constexpr int ALPHABET_SIZE = 26;
    struct Node {
        int len;
        int link;
        std::array<int, ALPHABET_SIZE> next;
        Node() : len{}, link{}, next{} {}
    };
    std::vector<Node> t;
    SAM() {
        init();
    }
    void init() {
        t.assign(2, Node());
        t[0].next.fill(1);
        t[0].len = -1;
    }
    int newNode() {
        t.emplace_back();
        return t.size() - 1;
    }
    int extend(int p, int c) {
        if (t[p].next[c]) {
            int q = t[p].next[c];
            if (t[q].len == t[p].len + 1) {
                return q;
            }
            int r = newNode();
            t[r].len = t[p].len + 1;
            t[r].link = t[q].link;
            t[r].next = t[q].next;
            t[q].link = r;
            while (t[p].next[c] == q) {
                t[p].next[c] = r;
                p = t[p].link;
            }
            return r;
        }
        int cur = newNode();
        t[cur].len = t[p].len + 1;
        while (!t[p].next[c]) {
            t[p].next[c] = cur;
            p = t[p].link;
        }
        t[cur].link = extend(p, c);
        return cur;
    }
};
```

7.11 SA-IS.cpp

```cpp
/*
 * Time Complexity: Suffix Array: O(N + Character_Set_Size) time and space //
 128 --- ASCII
 *                  LCP: O(N) time and space
 * Usage:
 *       1. Suffix Array (returns s.size() elements, NOT considering 0-length/empty suffix)
 *             auto sa = suffix_array(s); // s is the input string with ASCII characters
 *             auto sa_wide_char = suffix_array(s, LIM); // LIM = max(s[i]) + 2,
 s is the string with arbitary big characters.
 *       2. LCP:
 *            auto lcp = LCP(s, suffix_array(s)); // returns s.size() elements,
 where lcp[i]=LCP(sa[i], sa[i+1])
 * Status: Tested (DMOJ: ccc03s4, SPOJ: SARRAY (100pts), Yosupo's: Suffix Array
 & Number of Substrings, CodeForces EDU
 */
// Based on: Rickypon, https://judge.yosupo.jp/submission/10105
void induced_sort(const std::vector<int>& vec, int val_range,
                  std::vector<int>& SA, const std::vector<bool>& sl,
                  const std::vector<int>& lms_idx) {
    std::vector<int> l(val_range, 0), r(val_range, 0);
    for (int c : vec) {
        if (c + 1 < val_range) ++l[c + 1];
        ++r[c];
    }
    std::partial_sum(l.begin(), l.end(), l.begin());
    std::partial_sum(r.begin(), r.end(), r.begin());
    std::fill(SA.begin(), SA.end(), -1);
    for (int i = (int)lms_idx.size() - 1; i >= 0; --i)
        SA[--r[vec[lms_idx[i]]]] = lms_idx[i];
    for (int i : SA)
        if (i >= 1 && sl[i - 1]) SA[l[vec[i - 1]]++] = i - 1;
    std::fill(r.begin(), r.end(), 0);
    for (int c : vec) ++r[c];
    std::partial_sum(r.begin(), r.end(), r.begin());
    for (int k = (int)SA.size() - 1, i = SA[k]; k >= 1; --k, i = SA[k])
        if (i >= 1 && !sl[i - 1]) {
            SA[--r[vec[i - 1]]] = i - 1;
        }
}

std::vector<int> SA_IS(const std::vector<int>& vec, int val_range) {
    const int n = vec.size();
    std::vector<int> SA(n), lms_idx;
    std::vector<bool> sl(n);
    sl[n - 1] = false;
    for (int i = n - 2; i >= 0; --i) {
        sl[i] = (vec[i] > vec[i + 1] || (vec[i] == vec[i + 1] && sl[i + 1]));
        if (sl[i] && !sl[i + 1]) lms_idx.push_back(i + 1);
    }
    std::reverse(lms_idx.begin(), lms_idx.end());
    induced_sort(vec, val_range, SA, sl, lms_idx);
    std::vector<int> new_lms_idx(lms_idx.size()), lms_vec(lms_idx.size());
    for (int i = 0, k = 0; i < n; ++i)
        if (!sl[SA[i]] && SA[i] >= 1 && sl[SA[i] - 1]) {
            new_lms_idx[k++] = SA[i];
        }
    int cur = 0;
    SA[n - 1] = cur;
    for (size_t k = 1; k < new_lms_idx.size(); ++k) {
        int i = new_lms_idx[k - 1], j = new_lms_idx[k];
        if (vec[i] != vec[j]) {
            SA[j] = ++cur;
            continue;
        }
        bool flag = false;
        for (int a = i + 1, b = j + 1;; ++a, ++b) {
            if (vec[a] != vec[b]) {
                flag = true;
                break;
            }
            if ((!sl[a] && sl[a - 1]) || (!sl[b] && sl[b - 1])) {
                flag = !((!sl[a] && sl[a - 1]) && (!sl[b] && sl[b - 1]));
                break;
            }
        }
        SA[j] = (flag ? ++cur : cur);
    }
    for (size_t i = 0; i < lms_idx.size(); ++i) lms_vec[i] = SA[lms_idx[i]];
```

```
    if (cur + 1 < (int)lms_idx.size()) {
        auto lms_SA = SA_IS(lms_vec, cur + 1);
        for (size_t i = 0; i < lms_idx.size(); ++i) {
            new_lms_idx[i] = lms_idx[lms_SA[i]];
        }
    }
    induced_sort(vec, val_range, SA, sl, new_lms_idx);
    return SA;
}

std::vector<int> suffix_array(const std::string& s, const char first = 'a',
                          const char last = 'z') {
    std::vector<int> vec(s.size() + 1);
    std::copy(std::begin(s), std::end(s), std::begin(vec));
    for (auto& x : vec) x -= (int)first - 1;
    vec.back() = 0;
    auto ret = SA_IS(vec, (int)last - (int)first + 2);
    ret.erase(ret.begin());
    return ret;
}
// Author: https://codeforces.com/blog/entry/12796?#comment-175287
// Uses kasai's algorithm linear in time and space
std::vector<int> LCP(const std::string& s, const std::vector<int>& sa) {
    int n = s.size(), k = 0;
    std::vector<int> lcp(n), rank(n);
    for (int i = 0; i < n; i++) rank[sa[i]] = i;
    for (int i = 0; i < n; i++, k ? k-- : 0) {
        if (rank[i] == n - 1) {
            k = 0;
            continue;
        }
        int j = sa[rank[i] + 1];
        while (i + k < n && j + k < n && s[i + k] == s[j + k]) k++;
        lcp[rank[i]] = k;
    }
    lcp[n - 1] = 0;
    return lcp;
}

template <typename T, class F = function<T(const T&, const T&)>>
class SparseTable {
 public:
  int n;
  vector<vector<T>> mat;
  F func;

  SparseTable(const vector<T>& a, const F& f) : func(f) {
    n = static_cast<int>(a.size());
    int max_log = 32 - __builtin_clz(n);
    mat.resize(max_log);
    mat[0] = a;
    for (int j = 1; j < max_log; j++) {
      mat[j].resize(n - (1 << j) + 1);
      for (int i = 0; i <= n - (1 << j); i++) {
        mat[j][i] = func(mat[j - 1][i], mat[j - 1][i + (1 << (j - 1))]);
      }
    }
  }

  T get(int from, int to) const {
    assert(0 <= from && from <= to && to <= n - 1);
    int lg = 32 - __builtin_clz(to - from + 1) - 1;
    return func(mat[lg][from], mat[lg][to - (1 << lg) + 1]);
  }
};
```

7.12 Z.cpp

```
template <typename T>
vector<int> z_function(int n, const T &s) {
  vector<int> z(n, n);
  int l = 0, r = 0;
  for (int i = 1; i < n; i++) {
    z[i] = (i > r ? 0 : min(r - i + 1, z[i - l]));
    while (i + z[i] < n && s[z[i]] == s[i + z[i]]) {
      z[i]++;
    }
    if (i + z[i] - 1 > r) {
      l = i;
      r = i + z[i] - 1;
    }
  }
  return z;
}

template <typename T>
```

```cpp
vector<int> z_function(const T &s) {
  return z_function((int) s.size(), s);
}
```

8 Basic

8.1 AST.py

```python
class Solution:
    def calculate(self, s: str) -> int:
        sign = ['+', '-', '*', '/', '(', ')']
        v = []
        num = ''
        for c in s:
            if c in sign:
                if num:
                    v.append(num)
                    num = ''
                if c == '-' and (not v or v[-1] == '('):
                    v.append('0')
                v.append(c)
            elif c.isnumeric():
                num += c
        if num:
            v.append(num)

        stk0 = []
        stk1 = []
        for e in v:
            if e.isnumeric():
                stk0.append(e)
            elif e in ['+', '-']:
                while stk1 and stk1[-1] in ['*', '/', '+', '-']:
                    stk0.append(stk1.pop())
                stk1.append(e)
            elif e in ['*', '/', '(']:
                stk1.append(e)
            else:
                while stk1 and stk1[-1] != '(':
                    stk0.append(stk1.pop())
                stk1.pop()
        while stk1:
            stk0.append(stk1.pop())

        res = []
        for e in stk0:
            if e.isnumeric():
                res.append(int(e))
            else:
                v = res.pop()
                u = res.pop()
                if e == '+':
                    res.append(u + v)
                if e == '-':
                    res.append(u - v)
                if e == '*':
                    res.append(u * v)
                if e == '/':
                    res.append(u // v)
        return res[0]
```

8.2 bitset.cpp

```cpp
template <int len = 1>
void solve(int n) {
    if (n > len) {
        solve<std::min(len*2, MAXLEN)>(n);
        return;
    }
    // solution using bitset<len>
}

struct Bitset {
    vector<ull> b;
    int n;
    Bitset(int x = 0) {
        n = x;
        b.resize((n + 63) / 64, 0);
    }

    int get(int x) {
        return (b[x >> 6] >> (x & 63)) & 1;
    }

    void set(int x, int y) {
```

```
        b[x >> 6] |= 1ULL << (x & 63);
        if (!y) b[x >> 6] ^= 1ULL << (x & 63);
    }

    Bitset &operator&=(const Bitset &another) {
        rep(i, 0, min(SZ(b), SZ(another.b)) - 1) {
            b[i] &= another.b[i];
        }
        return (*this);
    }

    Bitset operator&(const Bitset &another)const {
        return (Bitset(*this) &= another);
    }

    Bitset &operator|=(const Bitset &another) {
        rep(i, 0, min(SZ(b), SZ(another.b)) - 1) {
            b[i] |= another.b[i];
        }
        return (*this);
    }

    Bitset operator|(const Bitset &another)const {
        return (Bitset(*this) |= another);
    }

    Bitset &operator^=(const Bitset &another) {
        rep(i, 0, min(SZ(b), SZ(another.b)) - 1) {
            b[i] ^= another.b[i];
        }
        return (*this);
    }

    Bitset operator^(const Bitset &another)const {
        return (Bitset(*this) ^= another);
    }

    Bitset &operator>>=(int x) {
        if (x & 63) {
            rep(i, 0, SZ(b) - 2) {
                b[i] >>= (x & 63);
                b[i] ^= (b[i + 1] << (64 - (x & 63)));
            }
            b.back() >>= (x & 63);
        }

        x >>= 6;
        rep(i, 0, SZ(b) - 1) {
            if (i + x < SZ(b)) b[i] = b[i + x];
            else b[i] = 0;
        }
        return (*this);
    }

    Bitset operator>>(int x)const {
        return (Bitset(*this) >>= x);
    }

    Bitset &operator<<=(int x) {
        if (x & 63) {
            for (int i = SZ(b) - 1; i >= 1; i--) {
                b[i] <<= (x & 63);
                b[i] ^= b[i - 1] >> (64 - (x & 63));
            }
            b[0] <<= x & 63;
        }

        x >>= 6;
        for (int i = SZ(b) - 1; i >= 0; i--) {
            if (i - x >= 0) b[i] = b[i - x];
            else b[i] = 0;
        }
        return (*this);
    }

    Bitset operator<<(int x)const {
        return (Bitset(*this) <<= x);
    }
};
```

8.3 fastIO.cpp

```
static struct FastInput {
  static constexpr int BUF_SIZE = 1 << 20;
  char buf[BUF_SIZE];
  size_t chars_read = 0;
```

```
  size_t buf_pos = 0;
  FILE *in = stdin;
  char cur = 0;

  inline char get_char() {
    if (buf_pos >= chars_read) {
      chars_read = fread(buf, 1, BUF_SIZE, in);
      buf_pos = 0;
      buf[0] = (chars_read == 0 ? -1 : buf[0]);
    }
    return cur = buf[buf_pos++];
  }

  template <typename T>
  inline void tie(T) {}

  inline explicit operator bool() {
    return cur != -1;
  }

  inline static bool is_blank(char c) {
    return c <= '␣';
  }

  inline bool skip_blanks() {
    while (is_blank(cur) && cur != -1) {
      get_char();
    }
    return cur != -1;
  }

  inline FastInput& operator>>(char& c) {
    skip_blanks();
    c = cur;
    get_char();
    return *this;
  }

  inline FastInput& operator>>(string& s) {
    if (skip_blanks()) {
      s.clear();
      do {
        s += cur;
      } while (!is_blank(get_char()));
    }
    return *this;
  }

  template <typename T>
  inline FastInput& read_integer(T& n) {
    // unsafe, doesn't check that characters are actually digits
    n = 0;
    if (skip_blanks()) {
      int sign = +1;
      if (cur == '-') {
        sign = -1;
        get_char();
      }
      do {
        n += n + (n << 3) + cur - '0';
      } while (!is_blank(get_char()));
      n *= sign;
    }
    return *this;
  }

  template <typename T>
  inline typename enable_if<is_integral<T>::value, FastInput&>::type
      operator>>(T& n) {
    return read_integer(n);
  }

  #if !defined(_WIN32) || defined(_WIN64)
  inline FastInput& operator>>(__int128& n) {
    return read_integer(n);
  }
  #endif

  template <typename T>
  inline typename enable_if<is_floating_point<T>::value, FastInput&>::type
      operator>>(T& n) {
    // not sure if really fast, for compatibility only
    n = 0;
    if (skip_blanks()) {
      string s;
      (*this) >> s;
```

```
      sscanf(s.c_str(), "%lf", &n);
    }
    return *this;
  }
} fast_input;

#define cin fast_input

static struct FastOutput {
  static constexpr int BUF_SIZE = 1 << 20;
  char buf[BUF_SIZE];
  size_t buf_pos = 0;
  static constexpr int TMP_SIZE = 1 << 20;
  char tmp[TMP_SIZE];
  FILE *out = stdout;

  inline void put_char(char c) {
    buf[buf_pos++] = c;
    if (buf_pos == BUF_SIZE) {
      fwrite(buf, 1, buf_pos, out);
      buf_pos = 0;
    }
  }

  ~FastOutput() {
    fwrite(buf, 1, buf_pos, out);
  }

  inline FastOutput& operator<<(char c) {
    put_char(c);
    return *this;
  }

  inline FastOutput& operator<<(const char* s) {
    while (*s) {
      put_char(*s++);
    }
    return *this;
  }

  inline FastOutput& operator<<(const string& s) {
    for (int i = 0; i < (int) s.size(); i++) {
      put_char(s[i]);
    }
    return *this;
  }

  template <typename T>
  inline char* integer_to_string(T n) {
    // beware of TMP_SIZE
    char* p = tmp + TMP_SIZE - 1;
    if (n == 0) {
      *--p = '0';
    } else {
      bool is_negative = false;
      if (n < 0) {
        is_negative = true;
        n = -n;
      }
      while (n > 0) {
        *--p = (char) ('0' + n % 10);
        n /= 10;
      }
      if (is_negative) {
        *--p = '-';
      }
    }
    return p;
  }

  template <typename T>
  inline typename enable_if<is_integral<T>::value, char*>::type stringify(T
      n) {
    return integer_to_string(n);
  }

  #if !defined(_WIN32) || defined(_WIN64)
  inline char* stringify(__int128 n) {
    return integer_to_string(n);
  }
  #endif

  template <typename T>
  inline typename enable_if<is_floating_point<T>::value, char*>::type
      stringify(T n) {
    sprintf(tmp, "%.17f", n);
```

```
    return tmp;
  }

  template <typename T>
  inline FastOutput& operator<<(const T& n) {
    auto p = stringify(n);
    for (; *p != 0; p++) {
      put_char(*p);
    }
    return *this;
  }
} fast_output;

#define cout fast_output
```

8.4 FastMod.cpp

```
Description: Compute a % b about 5 times faster than usual, where b is
constant but not known at compile time. Returns a value congruent to a
(mod b) in the range [0, 2b).

typedef unsigned long long ull;
struct FastMod {
    ull b, m;
    FastMod(ull b) : b(b), m(-1ULL / b) {}
    ull reduce(ull a) { // a % b + (0 or b)
        return a - (ull)((__uint128_t(m) * a) >> 64) * b;
    }
};
```

8.5 intervalContainer.cpp

```
Description: Add and remove intervals from a set of disjoint intervals.
Will merge the added interval with any overlapping intervals in the set when
adding. Intervals are [inclusive, exclusive).
Time: O (log N)

set<pii>::iterator addInterval(set<pii>& is, int L, int R) {
    if (L == R) return is.end();
    auto it = is.lower_bound({L, R}), before = it;
    while (it != is.end() && it->first <= R) {
        R = max(R, it->second);
        before = it = is.erase(it);
    }
    if (it != is.begin() && (--it)->second >= L) {
        L = min(L, it->first);
        R = max(R, it->second);
        is.erase(it);
    }
    return is.insert(before, {L, R});
}
void removeInterval(set<pii>& is, int L, int R) {
    if (L == R) return;
    auto it = addInterval(is, L, R);
    auto r2 = it->second;
    if (it->first == L) is.erase(it);
    else (int&)it->second = L;
    if (R != r2) is.emplace(R, r2);
}
```

8.6 lineContainer.cpp

```
/**
 * Author: Simon Lindholm
 * Date: 2017-04-20
 * License: CC0
 * Source: own work
 * Description: Container where you can add lines of the form kx+m, and
     query maximum values at points x.
 *  Useful for dynamic programming (``convex hull trick'').
 * Time: O(\log N)
 * Status: stress-tested
 */
#pragma once

struct Line {
    mutable ll k, m, p;
    bool operator<(const Line& o) const { return k < o.k; }
    bool operator<(ll x) const { return p < x; }
};

struct LineContainer : multiset<Line, less<>> {
    // (for doubles, use inf = 1/.0, div(a,b) = a/b)
    static const ll inf = LLONG_MAX;
    ll div(ll a, ll b) { // floored division
        return a / b - ((a ^ b) < 0 && a % b); }
```

```
    bool isect(iterator x, iterator y) {
        if (y == end()) return x->p = inf, 0;
        if (x->k == y->k) x->p = x->m > y->m ? inf : -inf;
        else x->p = div(y->m - x->m, x->k - y->k);
        return x->p >= y->p;
    }
    void add(ll k, ll m) {
        auto z = insert({k, m, 0}), y = z++, x = y;
        while (isect(y, z)) z = erase(z);
        if (x != begin() && isect(--x, y)) isect(x, y = erase(y));
        while ((y = x) != begin() && (--x)->p >= y->p)
            isect(x, erase(y));
    }
    ll query(ll x) {
        assert(!empty());
        auto l = *lower_bound(x);
        return l.k * x + l.m;
    }
};
```

8.7 mint.cpp

```
template<int MOD, int RT> struct mint {
    static const int mod = MOD;
    static constexpr mint rt() { return RT; } // primitive root for FFT
    int v; explicit operator int() const { return v; } // explicit -> don't
        silently convert to int
    mint():v(0) {}
    mint(ll _v) { v = int((-MOD < _v && _v < MOD) ? _v : _v % MOD);
        if (v < 0) v += MOD; }
    bool operator==(const mint& o) const {
        return v == o.v; }
    friend bool operator!=(const mint& a, const mint& b) {
        return !(a == b); }
    friend bool operator<(const mint& a, const mint& b) {
        return a.v < b.v; }

    mint& operator+=(const mint& o) {
        if ((v += o.v) >= MOD) v -= MOD;
        return *this; }
    mint& operator-=(const mint& o) {
        if ((v -= o.v) < 0) v += MOD;
        return *this; }
    mint& operator*=(const mint& o) {
        v = int((ll)v*o.v%MOD); return *this; }
    mint& operator/=(const mint& o) { return (*this) *= inv(o); }
    friend mint pow(mint a, ll p) {
        mint ans = 1; assert(p >= 0);
        for (; p; p /= 2, a *= a) if (p&1) ans *= a;
        return ans; }
    friend mint inv(const mint& a) { assert(a.v != 0);
        return pow(a,MOD-2); }

    mint operator-() const { return mint(-v); }
    mint& operator++() { return *this += 1; }
    mint& operator--() { return *this -= 1; }
    friend mint operator+(mint a, const mint& b) { return a += b; }
    friend mint operator-(mint a, const mint& b) { return a -= b; }
    friend mint operator*(mint a, const mint& b) { return a *= b; }
    friend mint operator/(mint a, const mint& b) { return a /= b; }
};

const int MOD=998244353;
using mi = mint<MOD,5>; // 5 is primitive root for both common mods

namespace simp {
    vector<mi> fac,ifac,invn;
    void check(int x) {
        if (fac.empty()) {
            fac={mi(1),mi(1)};
            ifac={mi(1),mi(1)};
            invn={mi(0),mi(1)};
        }
        while (SZ(fac)<=x) {
            int n=SZ(fac),m=SZ(fac)*2;
            fac.resize(m);
            ifac.resize(m);
            invn.resize(m);
            for (int i=n;i<m;i++) {
                fac[i]=fac[i-1]*mi(i);
                invn[i]=mi(MOD-MOD/i)*invn[MOD%i];
                ifac[i]=ifac[i-1]*invn[i];
            }
        }
    }
    mi gfac(int x) {
```

```
        check(x); return fac[x];
    }
    mi ginv(int x) {
        check(x); return invn[x];
    }
    mi gifac(int x) {
        check(x); return ifac[x];
    }
    mi binom(int n,int m) {
        if (m < 0 || m > n) return mi(0);
        return gfac(n)*gifac(m)*gifac(n - m);
    }
}
```

8.8 pbds.cpp

```
#include <bits/extc++.h>
using namespace __gnu_cxx;
using namespace __gnu_pbds;

#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
#include<ext/pb_ds/hash_policy.hpp>
#include<ext/pb_ds/trie_policy.hpp>
#include<ext/pb_ds/priority_queue.hpp>

pairing_heap_tag: 配对堆
thin_heap_tag: 斐波那契堆
binomial_heap_tag: 二项堆
binary_heap_tag: 二叉堆

__gnu_pbds::priority_queue<PII, greater<PII>, pairing_heap_tag> q;
__gnu_pbds::priority_queue<PII, greater<PII>, pairing_heap_tag>::
    point_iterator its[N];

its[v] = q.push({dis[v], v});
q.modify(its[v], {dis[v], v});

可以将两个优先队列中的元素合并（无任何约束）
使用方法为a.join(b)
此时优先队列b内所有元素就被合并进优先队列a中，且优先队列b被清空

cc_hash_table<string, int> mp1拉链法

gp_hash_table<string, int> mp2查探法
```

8.9 simu.cpp

```
db rnd(db l, db r) {
  static uniform_real_distribution<db> u(0, 1);
  static default_random_engine e(rng());
  return l + (r - l) * u(e);  // u(rng);
}

db eval(pair<db, db> x) { ... }

void simulate_anneal() {
  pair<db, db> cur(rnd(0, 10000), rnd(0, 10000));
  for (double k = 10000; k > 1e-5; k *= 0.99) {
    // [start, end, step]
    pair<db, db> nxt(cur.fi + rnd(-k, k), cur.se + rnd(-k, k));
    db delta = eval(nxt) - eval(cur);
    if (exp(-delta / k) > rnd(0, 1)) {
      cur = nxt;
    }
  }
}

/**
 * https://codeforces.com/gym/104813/submission/234982955
 * The 9th CCPC (Harbin) 2023
 * Author: QwertyPi
 */
LD Prob() {
  static uniform_real_distribution<> dist(0.0, 1.0);
  return dist(rng);
}
LD Sigma(LD x) { return 1 / (1 + exp(-x)); }

LD overall_max_score = 0;
for (int main_loop = 0; main_loop < 5; main_loop++) {
  vector<LD> e(n, (LD)1 / n);
  for (int tr = 0; tr < 1000; tr++) {
    vector<LD> ne(n);
    for (int i = 0; i < n; i++) {
      ne[i] = Prob();
    }
```

```
    LD s = accumulate(all(ne), 0.0L);
    for (int i = 0; i < n; i++) {
      ne[i] /= s;
    }
    if (eval(ne) > eval(e)) e = ne;
  }
  LD t = (LD)0.0002;
  LD max_score = 0;
  const LD depr = 0.999995;
  const int tries = 2E6;
  const int loop = 1E5;

  LD score_old = eval(e);
  for (int tr = 0; tr < tries; tr++) {
#ifdef LOCAL
    if (tr % loop == loop - 1) {
      cout << fixed << setprecision(10) << "current␣score␣=␣" << max_score
           << ",␣t␣=␣" << t << '\n';
    }
#endif
    int x = rng() % n, y = rng() % n;
    if (e[x] < t || x == y) {
      t *= depr;
      continue;
    }
    e[x] -= t;
    e[y] += t;
    LD score_new = eval(e);
    if (score_new > score_old) {  // ok
      ;
    } else {  // revert
      e[x] += t;
      e[y] -= t;
    }
    score_old = score_new;
    max_score = max(max_score, score_new);
    t *= depr;
  }
  overall_max_score = max(overall_max_score, max_score);
#ifdef LOCAL
  cout << "Loop␣#" << main_loop << ":␣" << max_score << '\n';
#endif
}
```

8.10 sort.cpp

```
void merge_sort(int q[], int l, int r) {
    if (l >= r) return;
    int mid = l + r >> 1;
    merge_sort(q, l, mid);
    merge_sort(q, mid + 1, r);

    int k = 0, i = l, j = mid + 1;
    while (i <= mid && j <= r)
        if (q[i] <= q[j])
            tmp[k++] = q[i++];
        else
            tmp[k++] = q[j++];

    while (i <= mid)
        tmp[k++] = q[i++];
    while (j <= r)
        tmp[k++] = q[j++];

    for (i = l, j = 0; i <= r; i++, j++) q[i] = tmp[j];
}

void quick_sort(int q[], int l, int r) {
    if (l >= r) return;
    int i = l - 1, j = r + 1, x = q[l + r >> 1];
    while (i < j) {
        do i ++ ; while (q[i] < x);
        do j -- ; while (q[j] > x);
        if (i < j) swap(q[i], q[j]);
    }
    quick_sort(q, l, j), quick_sort(q, j + 1, r);
}

template<class T>
void radixsort(T *a, ll n) {
    int base = 0;
    rep(i, 1, n) sa[i] = i;
    rep(k, 1, 5) {
        rep(i, 0, 255) c[i] = 0;
        rep(i, 1, n) c[(a[i] >> base) & 255]++;
        rep(i, 1, 255) c[i] += c[i - 1];
        per(i, n, 1) {
            rk[sa[i]] = c[(a[sa[i]] >> base) & 255]--;
```

```
        }
        rep(i, 1, n) sa[rk[i]] = i;
        base += 7;
    }
}
```

8.11 高精度.cpp

```
vector<int> add(vector<int> &A, vector<int> &B) {
    if (A.size() < B.size()) return add(B, A);
    vector<int> C;
    int t = 0;
    for (int i = 0; i < A.size(); i ++ ) {
        t += A[i];
        if (i < B.size()) t += B[i];
        C.push_back(t % 10);
        t /= 10;
    }
    if (t) C.push_back(t);
    return C;
}

vector<int> sub(vector<int> &A, vector<int> &B) {
    vector<int> C;
    for (int i = 0, t = 0; i < A.size(); i ++ ) {
        t = A[i] - t;
        if (i < B.size()) t -= B[i];
        C.push_back((t + 10) % 10);
        if (t < 0) t = 1;
        else t = 0;
    }
    while (C.size() > 1 && C.back() == 0) C.pop_back();
    return C;
}

vector<int> mul(vector<int> &A, int b) {
    vector<int> C;
    int t = 0;
    for (int i = 0; i < A.size() || t; i ++ ) {
        if (i < A.size()) t += A[i] * b;
        C.push_back(t % 10);
        t /= 10;
    }
    while (C.size() > 1 && C.back() == 0) C.pop_back();
    return C;
}

vector<int> div(vector<int> &A, int b, int &r) {
    vector<int> C;
    r = 0;
    for (int i = A.size() - 1; i >= 0; i -- ) {
        r = r * 10 + A[i];
        C.push_back(r / b);
        r %= b;
    }
    reverse(C.begin(), C.end());
    while (C.size() > 1 && C.back() == 0) C.pop_back();
    return C;
}
```