1 DataStructure

1.1 ST

```
struct ST{  // [0,n)
    static const int N = 101010; int a[20][N] , lg[N] , (*f)(int,int);
    void Build(int *v,int n,int (*_f)(int,int)){
        f = _f;rep(i,0,n) a[0][i] = v[i]; rep(i,2,n+1) lg[i] = lg[i>>1]+1;
        rep(i,1,lg[n]+1) rep(j,0,n-(1<<i)+1) a[i][j]=f(a[i-1][j],a[i-1][j+(1<<i>>1)]);
    }
    int rmq(int x,int y){
        if(x>y) swap(x,y); int i=lg[y-x+1];
        return f(a[i][x] , a[i][y+1-(1<<i)]);
    }
};
```

1.2 cartesian_tree

```
// time complexity O(N), main property:
// 1.in-order traversal of the tree results in the original sequence.
//   that is, id in left subtree less than the id of root.
// 2.heap property, the parent of node has a smaller value than the
//   node itself.
void solve() { top = 0;
   rep(i, 1, n + 1) {
      int k = top; while (k > 0 && a[sta[k]].v > a[i].v) --k;
      if (k > 0) {
         fa[a[i].id] = a[sta[k]].id; rs[a[sta[k]].id] = a[i].id;
      }
      if (k < top) {
         fa[a[sta[k + 1]].id] = a[i].id; ls[a[i].id] = a[sta[k + 1]].id;
      }
      sta[top = ++k] = i;
   }
}
```

1.3 easy_segment_tree

```
void build() {  // build the tree
   for (int i = 0; i < n; ++i) t[n + i] = init_value[i];
   for (int i = n - 1; i > 0; --i) t[i] = t[i<<1] + t[i<<1|1];
}
void modify(int p, int value) {  // set value at position p
   for (t[p += n] = value; p > 1; p >>= 1) t[p>>1] = t[p] + t[p^1];
}
int query(int l, int r) {  // sum on interval [l, r)
   int res = 0;
   for (l += n, r += n; l < r; l >>= 1, r >>= 1) {
      if (l&1) res += t[l++]; if (r&1) res += t[--r];
   }  return res;
}
```

1.4 kd_tree

```
const int N = 2e5 + 7; const int D = 2; const double SCALE = 0.75;
struct Point { int c, id, x[D]; } buf[N];
int d, ctr;
int cmp(const Point &a, const Point &b) { return a.x[ctr] < b.x[ctr]; }
struct Node {
   int depth, size; Node *ch[2], *p; Point val, maxv, minv;
   void set(Node *t, int d) { ch[d] = t; t->p = this; }
   bool dir() { return this == p->ch[1]; }
   bool balanced() {
      return (double) max(ch[0]->size, ch[1]->size) <= (double) size * SCALE;
   }
   void update() {
      size = ch[0]->size + ch[1]->size + 1;
      for (int i = 0; i < d; ++i) {
         maxv.x[i] = max(val.x[i], max(ch[0]->maxv.x[i], ch[1]->maxv.x[i]));
         minv.x[i] = min(val.x[i], min(ch[0]->minv.x[i], ch[1]->minv.x[i]));
      }
   }
} pool[N], *cur, *null;
Node* newNode(Point p, int depth) {
   Node *t = cur++; t->size = 1; t->depth = depth;
   t->val = t->maxv = t->minv = p; t->ch[0] = t->ch[1] = t->p = null;
   return t;
}
struct KDTree {
   Node *root;
   Node *build(Point *a, int l, int r, int depth) {
      if (l > r) return null;
      ctr = depth; int mid = (l + r) >> 1;
      nth_element(a + l, a + mid, a + r + 1, cmp);
      Node *t = newNode(a[mid], depth);
      t->set(build(a, l, mid - 1, (depth + 1) % d), 0);
      t->set(build(a, mid + 1, r, (depth + 1) % d), 1);
      t->update();
      return t;
   }
   void dfs(Node *t, Point *vec, int &tot) {
      if (t == null) return;
      vec[tot++] = t->val;
      dfs(t->ch[0], vec, tot); dfs(t->ch[1], vec, tot);
   }
   void rebuild(Node *t) {
      Node *p = t->p; int tot = 0; dfs(t, buf, tot);
      Node *u = build(buf, 0, tot - 1, t->depth);
      p->set(u, t->dir());
      for (; p != null; p = p->p) p->update();
      if (t == root) root = u;
   }
   void insert(Point p) {
      if (root == null) { root = newNode(p, 0); return; }
      Node *cur = root, *last = null; int dir = 0;
      for (; cur != null;) {
         last = cur;
```

```
      dir = (p.x[cur->depth] > cur->val.x[cur->depth]);
      cur = cur->ch[dir];
    }
    Node *t = newNode(p, (last->depth + 1) % d), *bad = null;
    last->set(t, dir);
    for (; t != null; t = t->p) {
      t->update(); if (!t->balanced()) bad = t;
    }
    if (bad != null) rebuild(bad);
  }
  ll calcEval(Point u, Node *t, int d) {
    ll l = t->minv.x[d], r = t->maxv.x[d], x = u.x[d];
    if (x >= l && x <= r) return 0LL;
    ll ret = min(abs(x - l), abs(x - r));
    return ret * ret;
  }
  Point R; pair<ll, int> ans;
  void updateAns(Point u, Point p) {
    ll dis = 0;
    rep(i, 0, d) dis += 1ll * (u.x[i] - p.x[i]) * (u.x[i] - p.x[i]);
    if (u.c <= p.c && mp(dis, u.id) < ans) R = u, ans = mp(dis, u.id);
  }
  void query(Node *t, Point p) {
    if (t == null) return; updateAns(t->val, p);
    ll evalLeft = calcEval(p, t->ch[0], t->depth);
    ll evalRight = calcEval(p, t->ch[1], t->depth);
    if (evalLeft <= evalRight) {
      if (ans.fi > evalLeft) query(t->ch[0], p);
      if (ans.fi > evalRight) query(t->ch[1], p);
    } else {
      if (ans.fi > evalRight) query(t->ch[1], p);
      if (ans.fi > evalLeft) query(t->ch[0], p);
    }
  }
  void query(Point p) { ans = mp(1e18, INT_MAX); query(root, p); }
} kd;
void initNull(int _d) {
  d = _d; cur = pool; null = cur++; null->size = 0;
  for (int i = 0; i < d; ++i)
    null->maxv.x[i] = INT_MIN, null->minv.x[i] = INT_MAX;
}
```

1.5 lc_segment_tree

```
// vector<int> V include data point and query point
vector<int> V;
struct Line {
  int id; long double k, b;
  Line() { k = b = 0, id = 0; }
  Line(long double _k, long double _b){ k = _k, b = _b, id = 0;}
  long double getf(int x) const { return k * x + b; }
} nd[N << 2];
struct SegTree {
#define ls ((t)<<1)
#define rs ((t)<<1|1)
  bool less(const Line &p, const Line &q, int x) {
    if (!p.id) return true; if (!q.id) return false;
    return p.getf(x) < q.getf(x);
  }
  void upd(int t, int l, int r, Line line) {
    if (!nd[t].id) nd[t] = line;
    if (less(nd[t], line, V[l])) swap(nd[t], line);
    if (l == r || nd[t].k == line.k) return ;
    int z = (l + r) >> 1;
    long double ix = (nd[t].b - line.b) / ( -nd[t].k + line.k);
    if (ix < V[l] || ix > V[r]) return ;
    if (ix <= V[z]) upd(ls, l, z, nd[t]), nd[t] = line;
    else upd(rs, z + 1, r, line);
  }
  void upd2(int t, int l, int r, int L, int R, Line line) {
    if (L <= l && r <= R) { upd(t, l, r, line); return ; }
    int z = (l + r) >> 1;
    if (L <= z) upd2(ls, l, z, L, R, line);
    if (R > z) upd2(rs, z + 1, r, L, R, line);
  }
  Line qry(int t, int l, int r, int x) {
    if (l == r) return nd[t];
    int z = (l + r) >> 1;
    Line ret;
    if (x <= V[z]) ret = qry(ls, l, z, x);
    else ret = qry(rs, z + 1, r, x);
    if (less(ret, nd[t], x)) ret = nd[t];
    return ret;
  }
} seg;
```

1.6 lct

```
// 1.remember initialize all values of node 0, and node 0 must not be used.
// 2.before addEdge, weight of node should be initialized.
// 3.update weight of one node x: splay(x), [update operation], up(x).
struct LCT {
  bool rt[N], rev[N]; int n, fa[N], que[N], ch[N][2];
  void init(int _n) {
    n = _n; rep(i, 0, n) { rt[i] = true, rev[i] = false;
      fa[i] = ch[i][0] = ch[i][1] = 0; }
  }
  void reverse(int x) { rev[x] = !rev[x], swap(ch[x][0], ch[x][1]); }
  void up(int x) { }
  void down(int x){ if(rev[x]) rev[x] = 0,reverse(ch[x][0]),reverse(ch[x][1]); }
  void rotate(int x) {
    int y = fa[x], k = (ch[y][0] == x);
    ch[y][!k] = ch[x][k];
    fa[ch[x][k]] = y, fa[x] = fa[y];
    fa[ch[x][k] = y] = x;
    if (rt[y]) rt[y] = false, rt[x] = true;
    else ch[fa[x]][ch[fa[x]][1] == y] = x;
    up(y);
  }
  void update(int x) {
```

```
      int top = 0; que[top++] = x;
      while (!rt[x]) x = fa[x], que[top++] = x;
      while (top) down(que[--top]);
   }
   void splay(int x) {
      update(x);
      while (!rt[x]) {
         int y = fa[x], z = fa[y];
         if (!rt[y]) (ch[z][1] == y) == (ch[y][1] == x) ? rotate(y) : rotate(x);
         rotate(x);
      }  up(x);
   }
   void access(int x) { // x~root be a preferred path, like heavy chain
      for (int y = 0; x; y = x, x = fa[x]) {
         splay(x); rt[ch[x][1]] = true; rt[ch[x][1] = y] = false; up(x);
      }
   }
   int getRoot(int x) {
      access(x), splay(x); while (ch[x][0]) x = ch[x][0]; return x;
   }
   void makeRoot(int x) { access(x), splay(x), reverse(x); }
   // be sure x,y not in one tree
   void addEdge(int x, int y) { makeRoot(x), fa[x] = y; }
   void cut(int x) { // delete edge between(x, parent_x)
      access(x), splay(x);
      if (ch[x][0]) fa[ch[x][0]] = fa[x], rt[ch[x][0]] = true;
      fa[x] = ch[x][0] = 0;
   }
   void delEdge(int x, int y) { makeRoot(y), cut(x); }
   void delNode(int x) {
      splay(x); rep(i, 0, 2) {
         if (!ch[x][i]) continue;
         fa[ch[x][i]] = fa[x], rt[ch[x][i]] = true;
         fa[x] = ch[x][i] = 0;
      }
   }
} lct;
```

1.7 persistent_treap

```
struct Node {
   int ls, rs, siz; void clr() { ls = rs = 0, siz = 1; }
} a[N];
struct Treap {
   int tot; void init() { tot = 0, srand(time(NULL)); }
   inline int _rand() { return (((ll) rand() << 14) ^ ((ll) rand()));}
   inline int newnode() { ++tot, a[tot].clr(); return tot; }
   void up(int t) { a[t].siz = 1 + a[a[t].ls].siz + a[a[t].rs].siz;}
   int merge(int t1, int t2) {
      if (!t1 || !t2) {
         if (!t1 && !t2) return 0;
         int nt = newnode(); a[nt] = a[t1 | t2]; return nt;
      }
      int nt = newnode();
      if (_rand() % (a[t1].siz + a[t2].siz) < a[t1].siz) {
         a[nt] = a[t1]; a[nt].rs = merge(a[t1].rs, t2);
      } else { a[nt] = a[t2]; a[nt].ls = merge(t1, a[t2].ls); }
      up(nt); return nt;
   }
   void split(int t, int k, int &lt, int &rt) {
      if (!t) { lt = rt = 0; return ; }
      int nt = newnode(); a[nt] = a[t];
      if (a[a[t].ls].siz >= k) {
         rt = nt; split(a[t].ls, k, lt, a[nt].ls);
      } else {
         lt = nt; split(a[t].rs, k - 1 - a[a[t].ls].siz, a[nt].rs, rt);
      }  up(nt);
   }
} T;
```

1.8 splay_tree

```
struct SplayTree { // set T.L = 0 before use
   bool rev[N]; int rt, L, fa[N], val[N], siz[N], ch[N][2];
   inline int newnode() {
      ++L;siz[L]=1,fa[L]=rev[L]=ch[L][0]=ch[L][1]=0;return L;
   }
   void up(int x) {siz[x] = 1 + siz[ch[x][0]] + siz[ch[x][1]];}
   void down(int x) {if(rev[x])rev[x]=0,flip(ch[x][0]),flip(ch[x][1]);}
   void flip(int x) { rev[x] ^= 1, swap(ch[x][0], ch[x][1]); }
   void rot(int x) {
      int y = fa[x]; down(y), down(x);
      if (rt == y) rt = x; int f = ch[y][0] == x;
      ch[y][!f] = ch[x][f]; if (ch[x][f]) fa[ch[x][f]] = y;
      fa[x] = fa[y]; if (fa[y]) ch[fa[y]][ch[fa[y]][1] == y] = x;
      ch[x][f] = y; fa[y] = x; up(y);
   }
   void splay(int x, int g = 0) {
      if (fa[x] == g) down(x);
      else { while (fa[x] != g) rot(x); up(x); }
   }
   int build(int l, int r, int p) {
      if (l > r) return 0; int m = (l + r) >> 1, x = newnode();
      val[x] = m - 1, fa[x] = p; ch[x][0] = build(l, m - 1, x);
      ch[x][1] = build(m + 1, r, x); up(x); return x;
   }
   int getk(int k) {
      int x = rt;
      while (x) { down(x);
         if (k == siz[ch[x][0]] + 1) break;
         if (k <= siz[ch[x][0]]) x = ch[x][0];
         else k -= siz[ch[x][0]] + 1, x = ch[x][1];
      }  return x;
   }
} T;
```

1.9 stern_brocot_tree

```
/*  1. Initialize two values L and H to 0/1 and 1/0, respectively.
```

```
    2. Until q is found, repeat the following steps:
        Let L = a/b and H = c/d; compute the mediant M = (a + c)/(b + d).
        If M is less than q, then q is in the open interval (M,H);
         replace L by M and continue.
        If M is greater than q, then q is in the open interval (L,M);
         replace H by M and continue.
        In the remaining case, q = M; terminate the search algorithm. */
namespace SBT {
   const int INF = 1e9 + 7;
   typedef __int128 T;
   typedef pair<T, T> V; //V=[double | long double | fraction]
   inline int cmp(const V &a, const V &b) {
      T x = a.fi * b.se - a.se * b.fi; return (x > 0) - (x < 0); }
   inline bool in(const V &a, const V &b, const V &c) {
      return 0 <= cmp(c, a) && cmp(c, b) < 0; }
   pii operator+(const pii &a, const pii &b) {
      return mp(a.fi + b.fi, a.se + b.se); }
   pii operator*(const pii &a, int x) { return mp(a.fi * x, a.se * x);}
   void search(V v, int MAXB, pii &lo, pii &hi, int f) {
      V x;
      int l = 0, r = f > 0 ? (hi.se ? (MAXB - lo.se) / hi.se : INF) :
         (lo.se ? (MAXB - hi.se) / lo.se : INF);
      while (l + 1 < r) {
         int z = (l + r) >> 1;
         x = f > 0 ? lo + hi * z : lo * z + hi;
         f * cmp(x, v) <= 0 ? l = z : r = z;
      }
      x = f > 0 ? lo + hi * r : lo * r + hi;
      r = f * cmp(x, v) <= 0 ? r : l;
      f > 0 ? lo = lo + hi * r : hi = lo * r + hi;
   }
   pii solve(V v, int MAXB) { // find ROUND_HALF_UP(a / b) = v, b <= MAXB
      V L = mp(v.fi * 10 - 5, v.se * 10);
      V R = mp(v.fi * 10 + 5, v.se * 10);
      pii lo(0, 1), hi(1, 0);
      while (true) {
         V m = mp(lo.fi + hi.fi, lo.se + hi.se);
         if (in(L, R, m)) return mp(m.fi, m.se);
         search(v, MAXB, lo, hi, 1);search(v, MAXB, lo, hi, -1);
         if (in(L, R, lo)) return lo;if (in(L, R, hi)) return hi;
      }
      return mp(-1, -1);
   }
};
```

1.10 treap

```
ll v[N]; int tot, root, l[N], r[N], hr[N], s[N]
void init() { srand (time(NULl));tot = root = 0; }
void Lr(int &x) {
   int y; y = r[x], r[x] = l[y], l[y] = x;
   s[y] = s[x], s[x] = 1 + s[l[x]] + s[r[x]]; x = y; }
void Rr(int &x) {
   int y; y = l[x], l[x] = r[y], r[y] = x;
   s[y] = s[x], s[x] = 1 + s[l[x]] + s[r[x]]; x = y; }
void insert(int &k, ll key) {
   if (k == 0) {
      ++tot, v[tot] = key, s[tot] = 1; hr[k = tot] = (int) rand();
      return;
   }  ++s[k];
   if (key <= v[k]) { insert(l[k], key); if (hr[l[k]] > hr[k]) Rr(k);
   } else { insert(r[k], key); if (hr[r[k]] > hr[k]) Lr(k); }
}
ll del(int &k, ll key) {
   ll res; --s[k];
   if (key == v[k] || (l[k] == 0 && key < v[k]) || (r[k] == 0 && key >= v[k])) {
      res = v[k]; if (l[k] == 0 || r[k] == 0) { k = l[k] + r[k]; return res; }
      v[k] = del(l[k], key + 1); return res;
   }
   if (key <= v[k]) return del(l[k], key); if (key > v[k]) return del(r[k], key);
   return 0;
}
bool find(int &k, ll key) {
   if (k == 0) return false; if (key > v[k]) return find(r[k], key);
   return v[k] == key || find(l[k], key);
}
int rank(int &k, ll key) {
   if (k == 0) return 1; if (key <= v[k]) return rank(l[k], key);
   return s[l[k]] + 1 + rank(r[k], key);
}
ll select(int &k, int t) {
   if (t == s[l[k]] + 1) return v[k];
   if (t <= s[l[k]]) return select(l[k], t);
   return select(r[k], t - 1 - s[l[k]]);
}
```

2 Geometry

2.1 DelaunayTriangulation

```
typedef __int128 T;
const int N = 1e4 + 10 , NODE = N * 20;
const T eps = 0 , inf = 1e8;// be careful with inf
int sgn(T x){return (x>eps)-(x<-eps);}
struct P{
    T x,y;int id;P(){} P(T x,T y,int id=0):x(x),y(y),id(id){}
    P operator - (const P&b) const {return P(x-b.x,y-b.y);}
    T operator * (const P&b) const {return x*b.x+y*b.y;}
    T operator / (const P&b) const {return x*b.y-y*b.x;}
    void out() {cout << "(" << (int)x <<"," << (int)y << ")" << endl;}
};
T norm(P a){return a*a;}
T cross(P a,P b,P p){return (b-a)/(p-a);}
// be careful with integer limitation
bool inCir(P a,P b,P c,P p){
    T A = (b - p) / (c - p) * (norm(a) - norm(p));
    T B = (c - p) / (a - p) * (norm(b) - norm(p));
    T C = (a - p) / (b - p) * (norm(c) - norm(p));
    return sgn(A + B + C) > 0;
```

```
}
struct Tri;
struct Edge{
    Tri* tri;int side;
    Edge(Tri* tri=NULL,int side=0):tri(tri),side(side){}
};
struct Tri{
    P p[3];Edge edge[3];Tri*son[3];Tri(){}
    Tri(P p0,P p1,P p2){p[0]=p0,p[1]=p1,p[2]=p2;rep(i,0,3) son[i]=NULL;}
    bool has_son() const {return son[0];}
    bool contains(P q) const {
        rep(i,0,3) if(sgn(cross(p[i],p[(i+1)%3],q))<0) return false;
        return true;
    }
}tri_pool[NODE],*pt;
void set_edge(Edge a,Edge b){
    if(a.tri) a.tri->edge[a.side]=b;
    if(b.tri) b.tri->edge[b.side]=a;
}
Tri* root;
// use bfs to handle in-line case
Tri* find(P p){
    Tri*c=root;
    while(c->has_son()) rep(i,0,3)
        if(c->son[i]->contains(p))
        {c=c->son[i];break;}
    return c;
}
void flip(Tri*x,int px){
    Tri* y=x->edge[px].tri;int py=x->edge[px].side;
    if(!y||!inCir(x->p[0],x->p[1],x->p[2],y->p[py])) return;
    Tri*s[2];
    s[0]=new(pt++) Tri(x->p[(px+1)%3],y->p[py],x->p[px]);
    s[1]=new(pt++) Tri(y->p[(py+1)%3],x->p[px],y->p[py]);
    set_edge(Edge(s[0],0),Edge(s[1],0));
    set_edge(Edge(s[0],1),x->edge[(px+2)%3]),set_edge(Edge(s[0],2),y->edge[(py+1)%3]);
    set_edge(Edge(s[1],1),y->edge[(py+2)%3]),set_edge(Edge(s[1],2),x->edge[(px+1)%3]);
    rep(i,0,2) x->son[i]=y->son[i]=s[i];
    x->son[2]=y->son[2]=0;
    rep(i,0,2) flip(s[i],1),flip(s[i],2);
}
void add(P p){
    Tri*c=find(p),*s[3];
    rep(i,0,3) c->son[i]=s[i]=new(pt++) Tri(c->p[i],c->p[(i+1)%3],p);
    rep(i,0,3) set_edge(Edge(s[i],0),Edge(s[(i+1)%3],1));
    rep(i,0,3) set_edge(Edge(s[i],2),c->edge[(i+2)%3]);
    rep(i,0,3) flip(s[i],2);
}
void init(P*p,int n){
    pt=tri_pool;
    root=new(pt++) Tri(P(-inf,-inf),P(inf,-inf),P(0,inf));
    random_shuffle(p,p+n);
    rep(i,0,n) add(p[i]);
}
```

2.2 Geo2D

```
typedef db T;
const db eps = 1e-9 , pi = acosl(-1.);
int sgn(T x){return (x>eps)-(x<-eps);}
struct P{
  T x,y; P(){} P(T x,T y):x(x),y(y){}
  P operator - (const P&b) const {return P(x-b.x,y-b.y);}
  P operator + (const P&b) const {return P(x+b.x,y+b.y);}
  T operator * (const P&b) const {return x*b.x+y*b.y;}
  T operator / (const P&b) const {return x*b.y-y*b.x;}
  P operator * (const T&k) const {return P(x*k,y*k);}
  P operator / (const T&k) const {return P(x/k,y/k);}
  bool operator < (const P&b) const {return sgn(x-b.x)?x<b.x:y<b.y;}
  bool operator == (const P&b) const{return !sgn(x-b.x)&&!sgn(y-b.y);}
  P rot90(){return P(-y,x);}
  db arg() const {return atan2(y,x);}
};
T norm(P a){return a*a;}
T abs(P a) {return sqrtl(norm(a));}
P proj(P p,P a,P b){return (b-a)*((p-a)*(b-a)/norm(b-a))+a;}
P reflect(P p,P a,P b){return proj(p,a,b)*2-p;}
T cross(P o,P a,P b){return (a-o)/(b-o);}
int crossOp(P o,P a,P b){return sgn(cross(o,a,b));}
db rad(P p1,P p2){return atan2l(p1/p2,p1*p2);}
bool onPS(P p,P s,P t){return sgn((t-s)/(p-s))==0&&sgn((p-s)*(p-t))<=0;}
bool order(const P&a,const P&b){ return a.arg() < b.arg();}
struct L{ P s,t;L(){} L(P s,P t):s(s),t(t){}};
P insLL(L a,L b){ // line x line
  P s = a.s - b.s , v = a.t - a.s , w = b.t - b.s;
  db k1 = s / w , k2 = w / v;
  if(sgn(k2) == 0) return abs(b.s - a.s) < abs(b.t - a.s) ? b.s : b.t;
  return a.s + v * (k1 / k2);
}
bool isSSr(const L&a,const L&b){ // seg x seg restrict
  T c1=(a.t-a.s)/(b.s-a.s) , c2=(a.t-a.s)/(b.t-a.s),
    c3=(b.t-b.s)/(a.s-b.s) , c4=(b.t-b.s)/(a.t-b.s);
  return sgn(c1) * sgn(c2) < 0 && sgn(c3) * sgn(c4) < 0;
}
bool isSS(L a,L b){ // seg x seg , replace x->y to accelerate
  T c1=(a.t-a.s)/(b.s-a.s),c2=(a.t-a.s)/(b.t-a.s);
  T c3=(b.t-b.s)/(a.s-b.s),c4=(b.t-b.s)/(a.t-b.s);
  return sgn(c1) * sgn(c2) <= 0 && sgn(c3) * sgn(c4) <= 0 &&
    sgn(max(a.s.x,a.t.x) - min(b.s.x,b.t.x)) >= 0 &&
    sgn(max(b.s.x,b.t.x) - min(a.s.x,a.t.x)) >= 0 &&
    sgn(max(a.s.y,a.t.y) - min(b.s.y,b.t.y)) >= 0 &&
    sgn(max(b.s.y,b.t.y) - min(a.s.y,a.t.y)) >= 0;
}
bool inRegion(T a,T p,T b) {return sgn(a-p)==0||sgn(b-p)==0||(a<p!=b<p);}
bool inRec(P p,L a){ // p in Rectangle
  return inRegion(a.s.x,p.x,a.t.x) && inRegion(a.s.y,p.y,a.t.y);
}
db disPL(P p,L a){return fabs((a.t-a.s)/(p-a.s)) / abs(a.t-a.s);}
db disPS(P p,L a){ // p x seg dis
  if(sgn((a.t-a.s)*(p-a.s)) == -1) return abs(p-a.s);
```

```
  if(sgn((a.s-a.t)*(p-a.t)) == -1) return abs(p-a.t);
  return disPL(p,a);
}
db disSS(L a,L b){ // seg x seg dis
  if(isSS(a,b)) return 0;
  return min(min(disPS(a.s,b),disPS(a.t,b)),min(disPS(b.s,a),disPS(b.t,a)));
}
typedef vector<P> polygon;
polygon convex(polygon A){ // counter-clockwise , <= : <=180 , < : <180
  int n=sz(A),m=0;
  polygon B;B.resize(n+1);
  sort(all(A));
  rep(i,0,n){
    while(m > 1 && sgn((B[m-1]-B[m-2])/(A[i]-B[m-2]))<=0) --m;
    B[m++]=A[i];
  }
  per(i,0,n-1){
    while(m > 1 && sgn((B[m-1]-B[m-2])/(A[i]-B[m-2]))<=0) --m;
    B[m++]=A[i];
  }
  B.resize(m);
  if(sz(B) > 1) B.pop_back();
  return B;
}
T area(polygon A) { // multiple 2 with integer type
  T res=0;
  rep(i,0,sz(A)) res+=A[i]/(A[(i+1)%sz(A)]);
  return fabs(res) / 2;
}
bool isconvex(polygon A){ // counter-clockwise
  bool ok=1;int n=sz(A);
  rep(i,0,2) A.pb(A[i]);
  rep(i,0,n) ok&=((A[i+1]-A[i])/(A[i+2]-A[i]))>=0;
  return ok;
}
int inPpolygon(P p,polygon A){ // -1 : on , 0 : out , 1 : in
  int res=0;
  rep(i,0,sz(A)){
    P u=A[i],v=A[(i+1)%sz(A)];
    if(onPS(p,u,v)) return -1;
    T cross = sgn((v-u)/(p-u)) , d1 = sgn(u.y-p.y) , d2 = sgn(v.y-p.y);
    if(cross > 0 && d1 <= 0 && d2 > 0) ++res;
    if(cross < 0 && d2 <= 0 && d1 > 0) --res;
  }
  return res != 0;
}
T diameter(polygon A) { // longest distance
  int n=sz(A);if(n <= 1) return 0;
  int l=0,r=0;rep(i,1,n) (A[i]<A[l])&&(l=i),(A[r]<A[i])&&(r=i);
  db res=abs(A[l]-A[r]);int i=l,j=r;
  do (++((A[(i+1)%n]-A[i])/(A[(j+1)%n]-A[j])>=0?j:i))%=n,
    res=max(res,abs(A[i]-A[j]));
  while(i!=l||j!=r);
  return res;
}
```

```
polygon convexCut(polygon A,P s,P t){ // counter-clockwise , left hand of st
  int n=sz(A);
  polygon B;
  rep(i,0,n){
    P u=A[i],v=A[(i+1)%n];
    int d1 = sgn((t-s)/(u-s)) , d2 = sgn((t-s)/(v-s));
    if(d1 >= 0) B.pb(u);
    if(d1 * d2 < 0) B.pb(insLL(L(u,v),L(s,t)));
  }
  return B;
}
namespace NearestPoints{
  T solve(int l,int r,vector<P>&p){
    if(l == r) return 1e100;
    int m=(l+r)>>1;
    T Xm = p[m].x , lim = min(solve(l,m,p) , solve(m+1,r,p));
    inplace_merge(p.begin()+l,p.begin()+m+1,p.begin()+r+1,[&](P a,P b){return a.y<b.y;})
    ;
    vector<P> V;
    rep(i,l,r+1) if(fabs(p[i].x - Xm) <= lim) V.pb(p[i]);
    rep(i,0,sz(V)) rep(j,i+1,sz(V)){
      if(fabs(V[j].y - V[i].y) >= lim) break;
      T dis = abs(V[i]-V[j]);
      lim = min(lim,dis);
    }
    return lim;
  }
  T solve(vector<P> A){
    sort(all(A),[&](P a,P b){return a.x<b.x;});
    return solve(0,sz(A)-1,A);
  }
}
struct C{P o;T r;C(){} C(P o,T r):o(o),r(r){}
  bool operator == (const C&b) const {return o==b.o&&sgn(r-b.r)==0;}};
vector<P> insCL(C c,L a){
  db x = (a.s-c.o)*(a.t-a.s) , y = norm(a.t-a.s);
  db d = x * x - y * (norm(a.s-c.o) - c.r*c.r);
  vector<P> res;
  if(sgn(d) < 0) return res;
  d = max(d,0.);
  P mid = a.s - (a.t - a.s) * (x / y);
  P del = (a.t - a.s) * (sqrt(d) / y);
  return {mid - del,mid + del}; // dir : a.s -> a.t
}
vector<P> insCC(C a,C b){
  vector<P> res;
  T x = norm(a.o - b.o);
  if(sgn(x)==0) return res;
  T  y = ((a.r * a.r - b.r * b.r) / x + 1) / 2 ,
     d = a.r * a.r / x - y * y;
  if(sgn(d) < 0) return res;
  d = max(d,0.);
  P mid = (b.o - a.o) * y + a.o ,
    del = ((b.o - a.o) * sqrt(d)).rot90();
  return {mid - del , mid + del};// counter-clockwise along a
```

```
}
vector<P> tanCP(C c,P p){
  db x = norm(p - c.o) , d = x - c.r * c.r;
  vector<P> res;
  if(sgn(d) < 0) return res;
  d = max(d,0.);
  P mid = c.o + (p - c.o) * (c.r * c.r / x) ,
    del = ((p-c.o)*(c.r*sqrt(d)/x)).rot90();
  return {mid - del ,mid + del}; // counter-clockwise
}
vector<pair<P,P> > tanCC(C c1,C c2){// need to unique
  vector<pair<P,P> > res;
  // extan
  if(!sgn(c1.r-c2.r)){
    P dir = c2.o-c1.o;
    dir = (dir*(c1.r/abs(dir))).rot90();
    res.pb({c1.o+dir,c2.o+dir});
    res.pb({c1.o-dir,c2.o-dir});
  } else {
    P p = (c2.o * c1.r - c1.o * c2.r) / (c1.r - c2.r);
    vector<P> ps = tanCP(c1 , p) , qs = tanCP(c2 , p);
    rep(i,0,min(sz(ps),sz(qs))) res.pb({ps[i],qs[i]});
  }
  // intan
  P p = (c1.o * c2.r + c2.o * c1.r) / (c1.r + c2.r);
  vector<P> ps = tanCP(c1 , p) , qs = tanCP(c2 , p);
  rep(i,0,min(sz(ps),sz(qs))) res.pb({ps[i],qs[i]});
  return res;
}
db areaCT(db r,P s,P t) { // need divide 2
  vector<P> p = insCL(C(P(0,0),r),L(s,t));
  if(!sz(p)) return r*r*rad(s,t);
  bool b1 = sgn(norm(s)-r*r) == 1 , b2 = sgn(norm(t)-r*r) == 1;
  if(b1 && b2) {
    if(sgn((s-p[0])*(t-p[0])) <= 0 && sgn((s-p[1])*(t-p[1])) <= 0)
      return r*r*(rad(s,p[0]) + rad(p[1],t)) + (p[0]/p[1]);
    else return r*r*rad(s,t);
  } else if(b1) return r*r*rad(s,p[0])+(p[0]/t);
  else if(b2) return r*r*rad(p[1],t)+(s/p[1]);
  return (s/t);
}
P inC(P A,P B,P C){
  db a = abs(B - C) , b = abs(C - A) , c = abs(A - B);
  return (A * a + B * b + C * c) / (a + b + c);
}
P outC(P A,P B,P C){
  P b = B - A , c = C - A;
  db dB = norm(b) , dC = norm(c) , d = b / c * 2;
  return A - P(b.y * dC - c.y * dB , c.x * dB - b.x * dC) / d;
}
P othroC(P A,P B,P C){
  P b = B - A , c = C - A;
  db Y = b.y * c.y * (B - C).y,
     a = c / b,
     xx = (Y + c.x * b.y * B.x - b.x * c.y * C.x) / a,
     yy = -b.x * (xx - C.x) / b.y + c.y;
  return P(xx , yy);
}
C Mincir(P *p,int n){
  random_shuffle(p , p + n);
  P o = p[0];db r = 0;
  rep(i,1,n) {
    if(sgn(abs(o-p[i])-r) <= 0) continue;
    o = p[i] , r = 0;
    rep(j,0,i) {
      if(sgn(abs(o-p[j])-r) <= 0) continue;
      o = (p[i] + p[j]) / 2 , r = abs(o-p[j]);
      rep(k,0,j) {
        if(sgn(abs(o-p[k])-r) <= 0) continue;
        o = outC(p[i],p[j],p[k]) , r = abs(o-p[k]);
      }}}
  return C(o,r);
}
namespace CircleIntersection{
  struct E{
    P p;T ang;int delta;
    E(){} E(P p,T ang,int delta):p(p),ang(ang),delta(delta){}
    bool operator < (const E&b) const {return ang<b.ang;}
  };
  bool overlap(C a,C b) {return sgn(a.r-b.r-abs(a.o-b.o))>=0;}
  void solve(C *c,int n,T *ans) {
    memset(ans , 0 , sizeof(T) * (n + 1));
    rep(i,0,n) {
      int cnt=1;
      vector<E> evt;
      rep(j,0,i) if(c[i]==c[j]) cnt++;
      rep(j,0,n) if(j!=i&&!(c[i]==c[j])&&overlap(c[j],c[i])) cnt++;
      rep(j,0,n) if(j!=i){
        vector<P> pts=insCC(c[i],c[j]);
        if(sz(pts)) {
          T a[2];
          rep(j,0,2) a[j]=(pts[j]-c[i].o).arg();
          evt.pb(E(pts[0],a[0],1));
          evt.pb(E(pts[1],a[1],-1));
          cnt += a[0] > a[1];
        }
      }
      if(!sz(evt)) ans[cnt] += pi*c[i].r*c[i].r;
      else{
        sort(all(evt));
        evt.pb(evt.front());
        rep(j,0,sz(evt)-1) {
          cnt+=evt[j].delta;
          ans[cnt] += evt[j].p / evt[j+1].p / 2;
          db ang = evt[j + 1].ang - evt[j].ang;
          if(ang < 0) ang += pi * 2;
          ans[cnt] += ang * c[i].r * c[i].r / 2 - sin(ang) * c[i].r * c[i].r / 2;
        }}}}}
namespace ConvecIntersection{
  const int N = 1005;
```

```
struct Rec {
  P d[10];int dn;// d[dn] = d[0]
  P operator [] (const int&n) {return d[n];}
}r[N];
typedef pair<db,int> pdi;
int n;pdi res[1000005];
db getLoc(P a,P b,P p){
  if(sgn(b.x - a.x)) return (p.x - a.x) / (b.x - a.x);
  return (p.y - a.y) / (b.y - a.y);
}
db work() {
  db rt=0;
  rep(i,0,n) rep(j,0,r[i].dn){
    int sz=0;
    res[sz++] = pdi(0,0);res[sz++] = pdi(1,0);
    rep(t,0,n) {
      if(t == i) continue;
      rep(g,0,r[t].dn) {
        int du = sgn((r[i][j+1] - r[i][j]) / (r[t][g] - r[i][j]));
        int dv = sgn((r[i][j+1] - r[i][j]) / (r[t][g+1] - r[i][j]));
        if(!du && !dv) {
          if(sgn((r[i][j+1] - r[i][j]) * (r[t][g+1] - r[t][g])) < 0 || i < t){
            res[sz++] = pdi(getLoc(r[i][j] , r[i][j+1] , r[t][g]) , 1);
            res[sz++] = pdi(getLoc(r[i][j] , r[i][j+1] , r[t][g+1]) , -1);
          }} else {
          db s1 = (r[i][j] - r[t][g]) / (r[t][g+1] - r[t][g]);
          db s2 = (r[t][g+1] - r[t][g]) / (r[i][j+1] - r[t][g]);
          if(du >= 0 && dv < 0) res[sz++] = pdi(s1 / (s1 + s2) , 1);
          else if(du < 0 && dv >= 0) res[sz++] = pdi(s1 / (s1 + s2) , -1);
        }}}
    sort(res , res + sz);
    int cnt = 0; --sz;
    rep(t,0,sz) {
      cnt += res[t].se;
      if(cnt == 0 && sgn(res[t].fi - res[t+1].fi)) {
        db a = res[t].fi;
        if(a < 0) a = 0; if(a > 1) break;
        db b = res[t+1].fi;
        if(b < 0) continue; if(b > 1) b = 1;
        rt += ((r[i][j+1] - r[i][j]) * a + r[i][j]) / ((r[i][j+1]-r[i][j]) * b + r[i][
        j]);
      }}}
  return rt / 2;}}
```

2.3 Geo3D

```
// didn't vertify
typedef double T;
const T eps = 1e-8;
int sgn(T x){return (x>eps)-(x<-eps);}
struct P{
  T x,y,z;P(){} P(T x,T y,T z):x(x),y(y),z(z){}
  P operator - (const P&b) const {return P(x-b.x,y-b.y,z-b.z);}
  P operator + (const P&b) const {return P(x+b.x,y+b.y,z+b.z);}
  P operator * (const T&k) const {return P(x*k,y*k,z*k);}
  P operator / (const T&k) const {return P(x/k,y/k,z/k);}
  T operator * (const P&b) const {return x*b.x+y*b.y+z*b.z;}
  P operator / (const P&b) const {return P(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);}
  bool operator < (const P&b) const {return tie(x,y,z)<tie(b.x,b.y,b.z);}
  bool operator == (const P&b) const {return sgn(x-b.x)==0&&sgn(y-b.y)==0&&sgn(z-b.z)
  ==0;}
  T len() const {return sqrtl(x*x+y*y+z*z);}
};
T norm(P a){return a*a;}
P outC(P A,P B,P C){
  T d = 2 * norm((A-B)/(B-C));
  if(sgn(d) == 0) {
    if((B-A).len()<(C-A).len()) swap(B,C);
    if((B-A).len()<(C-B).len()) swap(A,C);
    return (A+B)/2;
  }
  T a = (A-B)*(A-C)*norm(B-C);
  T b = (B-C)*(B-A)*norm(C-A);
  T c = (C-A)*(C-B)*norm(A-B);
  return (A*a+B*b+C*c)/d;
}
T mix(P a,P b,P c){return a / b * c;}
T area(P a,P b,P c){return ((b - a) / (c - a)).len();}
namespace Convex{
  typedef tuple<int,int,int> F;
  const int N = 1010;
  int mark[N][N] , n , cnt;
  vector<F> face;// (p[a]-p[b])/(p[c]-p[b]) inward ?
  void build(vector<P> p){
    sort(all(p));p.erase(unique(all(p)),p.end());
    n = sz(p);
    random_shuffle(all(p));
    face.clear();
    auto volume = [&](int a,int b,int c,int d)
    {return mix(p[b]-p[a],p[c]-p[a],p[d]-p[a]);};
    auto insert = [&](int a,int b,int c)
    {face.pb(make_tuple(a,b,c));};
    auto find = [&](){
      rep(i,2,n){
        P dir = (p[0] - p[i]) / (p[1] - p[i]);
        if(dir == P(0 , 0 , 0)) continue;
        swap(p[i] , p[2]);
        rep(j,i+1,n) if(sgn(volume(0,1,2,j))) {
          swap(p[j],p[3]);
          insert(0,1,2);
          insert(0,2,1);
          return 1;
        }
      }
      return 0;
    };
    auto add = [&](int d){
      vector<F> tmp;
      int a, b, c;
      cnt++;
```

```
      for(auto f : face){
        tie(a , b , c) = f;
        if (sgn(volume(d, a, b, c)) < 0)
          mark[a][b] = mark[b][a] = mark[b][c] = mark[c][b]
            = mark[c][a] = mark[a][c] = cnt;
        else tmp.pb(f);
      }
      face = tmp;
      for(auto f : tmp){
        tie(a , b , c) = f;
        if (mark[a][b] == cnt) insert(b,a,d);
        if (mark[b][c] == cnt) insert(c,b,d);
        if (mark[c][a] == cnt) insert(a,c,d);
      }
    };
    if(find()){
      rep(i,0,n) memset(mark[i],0,sizeof(int)*n);
      cnt = 0;
      rep(i,3,n) add(i);
    }}}
struct L{ P a,b; L(){} L(P a,P b):a(a),b(b){}};
struct PL{
  P a,b,c;
  PL(){} PL(P a,P b,P c):a(a),b(b),c(c){}
  P pvec() {return (b-a)/(c-a);}
  T area() {return pvec().len();}
};
T dis(P a,P b){return (b-a).len();}
bool PonL(P a,L l) {return sgn(((l.b-l.a)/(a-l.a)).len())==0;}
bool PonS(P a,L l){
  return PonL(a,l) &&
    sgn((l.a.x-a.x)*(l.b.x-a.x)) <= 0 &&
    sgn((l.a.y-a.y)*(l.b.y-a.y)) <= 0 &&
    sgn((l.a.z-a.z)*(l.b.z-a.z)) <= 0;
}
bool PonPL(P a,PL s) {return sgn(s.pvec()*(a-s.a))==0;}
bool PonTri(P a,PL s){
  return sgn(s.area()-PL(a,s.a,s.b).area()-PL(a,s.b,s.c).area()-PL(a,s.c,s.a).area())
  ==0;
}
bool PonPL(vector<P> p){ // distinct points
  int n=sz(p);
  if(n<4) return true;
  int c=-1;rep(i,2,n) if(!PonL(p[0],L(p[1],p[i]))) {c=i;break;}
  if(c==-1) return true;
  PL s(p[0],p[1],p[c]);
  rep(i,2,n) if(!PonPL(p[i],s)) return false;
  return true;
}
bool same_side(P a,P b,L l){ // coplanar, sgn(pvec()) to prove precision
  return sgn(PL(l.a,l.b,a).pvec()*PL(l.a,l.b,b).pvec()) > 0;
}
bool same_side(P a,P b,PL s){
  return sgn((s.pvec()*(a-s.a))*(s.pvec()*(b-s.a))) > 0;
}
bool opposite_side(P a,P b,L l){ // coplanar, sgn(pvec()) to prove precision
  return sgn(PL(l.a,l.b,a).pvec()*PL(l.a,l.b,b).pvec()) < 0;
}
bool opposite_side(P a,P b,PL s){
  return sgn((s.pvec()*(a-s.a))*(s.pvec()*(b-s.a))) < 0;
}
bool isSSr(L u,L v){
  return PonPL(u.a,PL(u.b,v.a,v.b)) &&
    opposite_side(u.a,u.b,v) && opposite_side(v.a,v.b,u);
}
bool isSS(L u,L v){
  if(!PonPL(u.a,PL(u.b,v.a,v.b))) return false;
  if(!PonL(u.a,v) || !PonL(u.b,v)) return isSSr(u,v);
  return PonS(u.a,v) || PonS(u.b,v) || PonS(v.a,u) || PonS(v.b,u);
}
bool isSTri(L l,PL s){ // can't coplanar
  return !same_side(l.a,l.b,s) && !same_side(s.a,s.b,PL(l.a,l.b,s.c)) &&
    !same_side(s.b,s.c,PL(l.a,l.b,s.a)) && !same_side(s.c,s.a,PL(l.a,l.b,s.b));
}
bool isSTrir(L l,PL s){
  return opposite_side(l.a,l.b,s) && opposite_side(s.a,s.b,PL(l.a,l.b,s.c)) &&
    opposite_side(s.b,s.c,PL(l.a,l.b,s.a)) && opposite_side(s.c,s.a,PL(l.a,l.b,s.b));
}
// | parallel , ^ perpendicular , & intersection
bool operator | (L a,L b) {return sgn(((a.b-a.a)/(b.b-b.a)).len())==0;}
bool operator ^ (L a,L b) {return sgn((a.b-a.a)*(b.b-b.a))==0;}
bool operator | (PL a,PL b) {return sgn((a.pvec()/b.pvec()).len())==0;}
bool operator ^ (PL a,PL b) {return sgn(a.pvec()*b.pvec())==0;}
bool operator | (L l,PL s) {return sgn(((l.b-l.a)/s.pvec()).len())==0;}
bool operator ^ (L l,PL s) {return sgn((l.b-l.a)*s.pvec())==0;}
P operator & (L a,L b) { // can't parallel
  P s = a.a - b.a , v = a.b - a.a , w = b.b - b.a;
  db k = (s / w) * (w / v) / ((w / v) * (w / v));
  return a.a + v * k;
}
P operator & (L l,PL s){ // can't parallel
  return l.a+(l.b-l.a)*((s.pvec()*(s.a-l.a))/(s.pvec()*(l.b-l.a)));
}
L operator & (PL s,PL t){ // can't parallel
  P a=L(s.a,(L(s.a,s.b)|t)?s.c:s.b)&t;
  return L(a,a+s.pvec()/t.pvec());
}
P PtoS(P a,L l){
  P b=l.a,c=l.b;
  rep(i,0,50) {
    P d=(b+c)*0.5,e=(d+c)*0.5;
    if(dis(a,d)<=dis(a,e)) b=e;
    else c=d;
  }
  return b;
}
// - distance , + projection
T operator - (P a,L l) {return ((a-l.a)/(l.b-l.a)).len() / dis(l.a,l.b);}
P operator + (P a,L l) {P s=l.a,d=l.b-l.a;return s+d*((a-s)*d/(d*d));}
T operator - (P a,PL s) {return fabs((a-s.a)*s.pvec()/s.pvec().len());}
```

```
P operator + (P a,PL s) {P d=s.pvec();return a+d*((s.a-a)*d/(d*d));}
T operator - (L u,L v) {P t=(u.b-u.a)/(v.b-v.a);return fabs((v.a-u.a)*t/t.len());}
P operator + (L a,L b) {return a & b;}
db angle(P a,P b) {return acos(max(-1.,min(1.,a*b/a.len()/b.len())));}
db angle(PL a,PL b) {return angle(a.pvec(),b.pvec());}
db angle(L l,PL s) {return asin(max(-1.,min(1.,(l.b-l.a)*s.pvec()/(l.b-l.a).len()/s.pvec
().len())));}
struct SP{
  P o;T r;
  T dis(P a,P b){ return angle(a-o,b-o)*r;}
  P to(T lng,T lat){ return P(cos(lng)*cos(lat)*r,sin(lng)*cos(lat)*r,sin(lat)*r);}
};
vector<P> operator & (L l,SP sp){
  if(sgn((sp.o-l)-sp.r)>0) return vector<P>();
  P s=l.a,d=l.b-l.a;
  T A=d*d,B=(s-sp.o)*d*2,C=(s-sp.o)*(s-sp.o)-sp.r*sp.r;
  T delta=sqrt(max(0.,B*B-4*A*C)),k1=(-B-delta)/(2*A),k2=(-B+delta)/(2*A);
  return {s+d*k1,s+d*k2};
}
```

2.4 HalfPlantIns

```
namespace HIP {
  static const double eps = 1e-8;
  static const int N = 450005;
  int dcmp(double x) {if (fabs(x) < eps) return 0;return x < 0 ? -1 : 1;}
  struct Point{ double x,y;Point() {}
    Point(double x, double y) : x(x), y(y) {}
  }p[N*2];
  struct Seg{ Point s,e;double r;
    void getr(){r=atan2(e.y-s.y,e.x-s.x);}
  } seg[N], Q[N];
  int n; void init() { n = 0; }
  void add_seg(double xa, double ya, double xb, double yb) {
    seg[n].s = Point(xa, ya); seg[n].e = Point(xb, yb);
    seg[n].getr(); n++;
  }
  double xmul(Point o,Point a,Point b)
  {return (a.x-o.x)*(b.y-o.y)-(b.x-o.x)*(a.y-o.y);}
  Point GetIns(Seg s1,Seg s2){
    double u=xmul(s1.s,s1.e,s2.s),v=xmul(s1.e,s1.s,s2.e);
    return Point((s2.s.x*v+s2.e.x*u)/(u+v),(s2.s.y*v+s2.e.y*u)/(u+v));
  }
  bool cmp(Seg s1,Seg s2){
    if(dcmp(s1.r-s2.r)>0) return true;
    else if(dcmp(s1.r-s2.r)==0&&dcmp(xmul(s2.s, s2.e, s1.e))>=0) return true;
    return false;
  }
  double Getarea(Point *p,int n){
    double area=0;
    for(int i=1;i<n-1;i++)area+=xmul(p[0],p[i],p[i+1]);
    return fabs(area)/2.0;
  }
  double HPI(){
    sort(seg,seg+n,cmp);
    int tmp=1;
    for(int i=1;i<n;i++) if(dcmp(seg[i].r-seg[tmp-1].r)) seg[tmp++]=seg[i];
    n=tmp; Q[0]=seg[0];Q[1]=seg[1];
    int h=0,r=1;
    for(int i=2;i<n;i++){
      while(h<r&&dcmp(xmul(seg[i].s,seg[i].e,GetIns(Q[r],Q[r-1])))<0) r--;
      while(h<r&&dcmp(xmul(seg[i].s,seg[i].e,GetIns(Q[h],Q[h+1])))<0) h++;
      Q[++r]=seg[i];
    }
    while(h<r&&dcmp(xmul(Q[h].s,Q[h].e,GetIns(Q[r],Q[r-1])))<0) r--;
    while(h<r&&dcmp(xmul(Q[r].s,Q[r].e,GetIns(Q[h],Q[h+1])))<0) h++;
    if (h==r) return 0.0;
    int m=0;
    for(int i=h;i<r;i++) p[m++]=GetIns(Q[i],Q[i+1]);
    if(r>h+1) p[m++]=GetIns(Q[h],Q[r]);
    return Getarea(p, m);
  }
}
```

2.5 Hull

```
typedef complex<ll> P;
typedef map<ll,P> hull;
#define X real()
#define Y imag()
ll cross(const P&a,const P&b){return (conj(a)*b).Y;}
struct Hull{
    hull h1,h2;
    bool in(hull&h,ll x,ll y){
        if(!sz(h)) return false;
        if(x < h.begin()->se.X || x > h.rbegin()->se.X) return false;
        auto l = h.lower_bound(x);
        if(x == l->se.X) return y <= l->se.Y;
        auto r = l--;
        return cross(r->se - l->se , P(x,y) - l->se) <= 0;
    }
    void ins(hull&h,ll x,ll y){
        if(in(h , x , y)) return;
        P p(x,y);h[x] = p;
        auto LL = h.find(x) , RR = LL , L = LL , R = L;
        if(L != h.begin()) for(--LL;(L = LL) != h.begin();){
            --(LL = L);
            if(cross(p - LL->se , L->se - LL->se) <= 0) h.erase(L);
            else break;
        }
        if(*R != *h.rbegin()) for(++RR;*(R = RR) != *h.rbegin();){
            ++(RR = R);
            if(cross(p - RR->se , R->se - RR->se) >= 0) h.erase(R);
            else break;
        }
    }
    void ins(ll x,ll y){ ins(h1,x,y);ins(h2,x,-y); }
    bool in(ll x,ll y){ return in(h1,x,y) && in(h2,x,-y); }
};
```

2.6 经纬度求球面最短距离

```
//lati 为纬度为经度 longi 为半径R
double Dist(double lati1,double longi1,double lati2,double longi2,double R)
{
  double pi=acos(-1.0);
  lati1*=pi/180,longi1*=pi/180,lati2*=pi/180,longi2*=pi/180;
  double x1=cos(lati1)*sin(longi1),y1=cos(lati1)*cos(longi1),z1=sin(lati1);
  double x2=cos(lati2)*sin(longi2),y2=cos(lati2)*cos(longi2),z2=sin(lati2);
  double theta=acos(x1*x2+y1*y2+z1*z2);
  return(R*theta);
}
```

2.7 长方体表面两点最短距离

```
int r;
void turn(int i, int j, int x, int y, int z,int x0, int y0, int L, int W, int H) {
  if (z==0) { int R = x*x+y*y; if (R<r) r=R;
  } else {
    if(i>=0 && i< 2) turn(i+1, j, x0+L+z, y, x0+L-x, x0+L, y0, H, W, L);
    if(j>=0 && j< 2) turn(i, j+1, x, y0+W+z, y0+W-y, x0, y0+W, L, H, W);
    if(i<=0 && i>-2) turn(i-1, j, x0-z, y, x-x0, x0-H, y0, H, W, L);
    if(j<=0 && j>-2) turn(i, j-1, x, y0-z, y-y0, x0, y0-H, L, H, W);
  }
}
int main(){
  int L, H, W, x1, y1, z1, x2, y2, z2;
  cin >> L >> W >> H >> x1 >> y1 >> z1 >> x2 >> y2 >> z2;
  if (z1!=0 && z1!=H) if (y1==0 || y1==W)
    swap(y1,z1), std::swap(y2,z2), std::swap(W,H);
  else swap(x1,z1), std::swap(x2,z2), std::swap(L,H);
  if (z1==H) z1=0, z2=H-z2;
  r=0x3fffffff;
  turn(0,0,x2-x1,y2-y1,z2,-x1,-y1,L,W,H);
  cout<<r<<endl;
}
```

3 Graph

3.1 ALLMincut

```
struct Stoer_Wagner {
  static const int N=506, INF=(1<<30);
  int G[N][N], W[N], Merge[N], S, T, minCut, n;bool vs[N];
  void init(int _n) {n=_n;memset(G, 0, sizeof G);}
  void BFS() {
    memset(vs, 0, sizeof vs);memset(W, 0, sizeof W);
    S=T=-1;int Max, id;
    rep(cnt,0,n) {
      Max=-INF;
      for(int i=0; i<n; i++) if(!Merge[i] && !vs[i] && W[i]>Max) Max=W[i], id=i;
      if(id==T) return;
      S=T, T=id; minCut=Max; vs[id]=1;
      rep(i,0,n) {
        if(Merge[i] || vs[i]) continue; W[i]+=G[id][i];
      }}}
  int StoerWagner() {
    memset(Merge, 0, sizeof Merge);
    int ans=INF;
    for(int cnt=1; cnt<n; cnt++) {
      BFS();
      if(minCut<ans) ans=minCut; if(ans==0) return ans;
      Merge[T]=1;
      for(int i=0; i<n; i++) {
        if(Merge[i]) continue;
        G[S][i]+=G[T][i]; G[i][S]+=G[i][T];
      }}
    return ans;
  }};
```

3.2 BCC

```
// key contains the id of edges, _ starts from 0
namespace BCC{
  const int N = 202020;
  vi key , bcc[N]; int dfn[N] , low[N] , id[N] , st[N] , _st , _;
  void dfs(int c,int dep,vector<pii> g[]){
    int cc=0;st[_st++]=c; dfn[c]=low[c]=dep;
    for(auto e:g[c]){
      int t=e.fi;
      if(!dfn[t]){
        dfs(t,dep+1,g); low[c]=min(low[c],low[t]);
        if(low[t]>dfn[c]) key.pb(e.se);
      } else if(dfn[t] != dfn[c] - 1 || cc++)
        low[c] = min(low[c] , dfn[t]);
    }
    if(low[c]==dfn[c])
    { do{id[st[--_st]]=_;}while(st[_st]!=c); _++; }
  }
  int solve(int n,vector<pii> g[]){
    fill_n(dfn,n,_=0);fill_n(low,n,_st=0);fill_n(bcc,n,key=vi());
    rep(i,0,n) if(!dfn[i]) dfs(i,1,g);
    rep(i,0,n) for(auto j:g[i]) if(id[i]!=id[j.fi])
      bcc[id[i]].pb(id[j.fi]);
    return _;
  }
};
```

3.3 DCC

```
// key is cuts
// dcc is edges , i->n+j , i(points) , j(bcc_block)
// be care of isolated point
namespace DCC{
  const int N = 202020;
  vi key , dcc[N]; int dfn[N] , low[N] , st[N] , _st , _;
  void dfs(int c,int dep,const vi g[]){
```

```
    int cc=0,out=1<dep;st[_st++]=c; dfn[c]=low[c]=dep;
    for(auto t:g[c])
      if(!dfn[t]){
        dfs(t,dep+1,g); low[c]=min(low[c],low[t]);
        if(low[t]>=dfn[c]){
          if(++out==2) key.pb(c);
          while(st[--_st]!=t) dcc[st[_st]].pb(_);
          dcc[c].pb(_);dcc[t].pb(_++);
        }
      } else if(dfn[t] != dfn[c] - 1 || cc++)
        low[c] = min(low[c] , dfn[t]);
  }
  int solve(int n,const vi g[]){// n is size of points
    fill_n(dfn,n,_=0); fill_n(low,n,_st=0); fill_n(dcc,n,key=vi());
    rep(i,0,n) if(!dfn[i]) dfs(i,1,g);
    rep(i,0,n) if(sz(dcc[i]) == 0) dcc[i].pb(_++);
    return _;
  }
}
```

3.4 DMST

```
struct edge{int u,v,d;};
struct DMST{
  static const int N = 55 , M = N * N , inf = 2e9;
  edge e[M];int n,m,vis[N],pre[N],id[N];int in[N];
  void ini(int n) {this->n=n,m=0;}
  void addedge(int u,int v,int d) {e[m++]=edge({u,v,d});}
  int run(int root){
    int ans=0;
    while(1){
      rep(i,0,n) in[i]=inf;
      rep(i,0,m){
        int u=e[i].u , v=e[i].v;
        if(e[i].d < in[v] && u != v){
          in[v] = e[i].d , pre[v] = u;
        }}
      rep(i,0,n) {
        if(i == root) continue; if(in[i] == inf) return -1;
      }
      int cnt = 0;in[root] = 0;
      memset(id,-1,sizeof(*id)*n);
      memset(vis,-1,sizeof(*vis)*n);
      rep(i,0,n){
        ans+=in[i];int v=i;
        while(vis[v]!=i && id[v]==-1 && v!=root){
          vis[v] = i;v = pre[v];
        }
        if(v != root && id[v] == -1) {
          for(int u=pre[v];u != v;u = pre[u]) id[u] = cnt;
          id[v] = cnt++;
        }}
      if(cnt == 0) break;
      rep(i,0,n) if(id[i] == -1) id[i] = cnt++;
      rep(i,0,m) {
        int v=e[i].v;
        e[i].u = id[e[i].u]; e[i].v = id[e[i].v];
        if(e[i].u != e[i].v) e[i].d -= in[v];
      }
      n = cnt;root = id[root];
    }
    return ans;
  }}tree;
```

3.5 Dinic

```
template<class T>
struct Dinic{ // [0,n) init!!
  const static int N = 10101 , M = N * 10;
  int s , t , n , h[N] , cur[N] , level[N] , q[N] , e , ne[M] , to[M];
  T cap[M] , flow;
  void liu(int u,int v,T w){ to[e] = v;ne[e] = h[u];cap[e] = w;h[u] = e++;}
  void link(int u,int v,T w){ liu(u , v , w);liu(v , u , 0);}
  void ini(int _n = N) { fill(h , h + (n=_n) , -1);e = 0;}
  bool bfs(){
    int L = 0 , R = 0;
    fill(level , level + n , -1); level[q[R++] = s] = 0;
    while(L < R && level[t] == -1){
      int c = q[L++];
      for(int k=h[c];~k;k=ne[k]) if(cap[k] > 0 && level[to[k]] == -1)
          level[q[R++] = to[k]] = level[c] + 1;
    }
    return ~level[t];
  }
  T dfs(int c,T mx){
    if(c == t) return mx;
    T ret = 0;
    for(int &k = cur[c];~k;k = ne[k]){
      if(level[to[k]] == level[c] + 1 && cap[k] > 0){
        T flow = dfs(to[k] , min(mx , cap[k]));
        ret += flow;cap[k] -= flow , cap[k^1] += flow;mx -= flow;
        if(!mx) return ret;
      }
    }
    level[c] = -1;
    return ret;
  }
  T run(int _s,int _t){
    s = _s , t = _t; flow = 0;
    while(bfs()){
      copy(h , h + n , cur); flow += dfs(s,~0U>>1);
    }
    return flow;
  }
};
```

3.6 KM

```
// init!! , id starts from 0
```

```
template<class T>
struct KM {
  static const int N = 505;
  static const T inf = ~0U>>2;
  int n, m, left[N], pre[N], used[N];
  T g[N][N], Lx[N], Ly[N], slack[N];
  void ini(int _n, int _m)
  { n = _n , m = _m; rep(i,0,n) rep(j,0,m) g[i][j] = -inf; }
  void go(int now) {
    rep(i,0,m+1) used[i]=0,slack[i]=inf; left[m] = now;int u,v;
    for(u=m;~left[u];u=v){
      used[u] = 1; T d = inf;
      rep(i,0,m) if(!used[i]){
        T tmp = Lx[left[u]] + Ly[i] - g[left[u]][i];
        if(tmp < slack[i]) slack[i] = tmp, pre[i] = u;
        if(slack[i] < d) d = slack[v=i];
      }
      rep(i,0,m+1) if(used[i]) Lx[left[i]] -= d , Ly[i] += d;
      else slack[i] -= d;
    }
    for(;u!=m;left[u]=left[pre[u]],u=pre[u]);
  }
  T run() {
    fill_n(Lx,n,0);fill_n(Ly,m,0); fill_n(left,m,-1);
    rep(i,0,n) go(i);
    T ans = 0; rep(i,0,n) ans += Lx[i]; rep(i,0,m) ans += Ly[i];
    return -ans;
  }
};
```

3.7 MinCostMaxFlow

```
// [0,n) , init!! , inf modify
template<class U,class V>
struct MCMF{
  static const int N = 204 , M = 101010;
  int h[N] , ing[N] , pre[N] , to[M] , ne[M] , e , s , t , n;
  U cap[M];V dis[N] , cost[M];
  void ini(int _n = N){ fill(h , h + (n=_n) , -1);e = 0;}
  void liu(int u,int v,U c,V w){ to[e] = v;ne[e] = h[u];cap[e] = c;cost[e] = w;h[u] = e
  ++;}
  void link(int u,int v,U c,V w){ liu(u,v,c,w);liu(v,u,0,-w); }
  bool spfa(){
    queue<int> Q; fill(dis,dis+n,inf); ing[s] = true , dis[s] = 0;
    Q.push(s);
    while(!Q.empty()){
      int c = Q.front();Q.pop();ing[c] = false;
      for(int k=h[c];~k;k=ne[k]){
        int v = to[k];
        if(cap[k] <= 0) continue;
        if(dis[c] + cost[k] < dis[v]){
          dis[v] = dis[c] + cost[k]; pre[v] = k;
          if(!ing[v]) Q.push(v) , ing[v] = true;
        }
      }
    }
    return dis[t] != inf;
  }
  U flow;V mincost;
  pair<U,V> run(int _s,int _t){
    s = _s , t = _t; flow = mincost = 0;
    while(spfa()){
      U pl = inf;int p , k;
      for(p=t;p!=s;p=to[k^1]) pl = min(pl , cap[k=pre[p]]);
      for(p=t;p!=s;p=to[k^1]){
        k = pre[p]; cap[k] -= pl; cap[k^1] += pl;
      }
      mincost += pl * dis[t]; flow += pl;
    }
    return make_pair(flow , mincost);
  }
};
```

3.8 SCC

```
namespace SCC{ // _ starts from 0
  const int N = 100050;
  int dfn[N],low[N],id[N],st[N],_st,_,cc;
  void dfs(int c,vi g[]){
    dfn[c]=low[c]=++cc; st[_st++]=c;
    for(auto t:g[c]) if(!dfn[t])
        dfs(t,g),low[c]=min(low[c],low[t]);
      else if(!id[t]) low[c]=min(low[c],dfn[t]);
    if(low[c]==dfn[c])
    { ++_; do{id[st[--_st]]=_;}while(st[_st]!=c); }
  }
  vi ng[N];
  int solve(int n,vi g[]){
    fill_n(dfn,n,cc=0); fill_n(low,n,_st=0); fill_n(id,n,_=0);
    rep(i,0,n) if(!dfn[i]) dfs(i,g); rep(i,0,n) --id[i];
    fill_n(ng,_,vi());
    rep(i,0,n) for(auto j:g[i]) if(id[i]!=id[j]) ng[id[i]].pb(id[j]);
    return _;
  }
}
```

3.9 TwoSat

```
struct TwoSat {
  static const int N = 1e5 + 10;
  int dfn[N],low[N],id[N],st[N],_st,_,cc,n,mark[N]; vi g[N];
  void ini(int tot) {
    n = tot * 2; rep(i,0,tot) g[i<<1].clear(),g[i<<1|1].clear();
  }
  void addedge(int u,int v,int pu,int pv) {
    u=u<<1|pu,v=v<<1|pv; g[u].pb(v);g[v^1].pb(u^1);
  }
  void dfs(int c,vi g[]){
    dfn[c]=low[c]=++cc; st[_st++]=c;
```

```
    for(auto t:g[c]) if(!dfn[t])
        dfs(t,g),low[c]=min(low[c],low[t]);
      else if(!id[t]) low[c]=min(low[c],dfn[t]);
    if(low[c]==dfn[c])
    { ++_; do{id[st[--_st]]=_;}while(st[_st]!=c); }
  }
  bool solve(){
    fill_n(dfn,n,cc=0); fill_n(low,n,_st=0); fill_n(id,n,_=0);
    rep(i,0,n) if(!dfn[i]) dfs(i,g);
    for(int i=0;i<n;i+=2) if(id[i]==id[i+1]) return false;
    //else mark[i>>1]=id[i]>id[i+1];
    return true;
  }
}sat;
```

3.10 cuttree

```
const int N = 1005;
Dinic<int> gao; int n, m;
struct Edge {
  int u, v, w; Edge() {}
  Edge(int u, int v, int w) : u(u), v(v), w(w) {}
  void read() { scanf("%d%d%d", &u, &v, &w); }
  bool operator < (const Edge& c) const
  { return w > c.w; }
} e[N * 10], E[N];
int en;int ans; int id[N], S[N], T[N];
void solve(int l, int r) {
  if (l == r) return;
  gao.ini(n + 1);
  for (int i = 0; i < m; i++) {
    gao.link(e[i].u, e[i].v, e[i].w);
    gao.link(e[i].v, e[i].u, e[i].w);
  }
  int tmp = gao.run(id[l], id[r], n + 1);
  ans += tmp;
  E[en++] = Edge(id[l], id[r], tmp);
  int sn = 0, tn = 0;
  for (int i = l; i <= r; i++) if (gao.level[id[i]] == -1) T[tn++] = id[i];
    else S[sn++] = id[i];
  for (int i = 0; i < sn; i++) id[i + l] = S[i];
  for (int i = 0; i < tn; i++) id[i + l + sn] = T[i];
  solve(l, l + sn - 1); solve(l + sn, l + sn + tn - 1);
}
int main() {
  while (~scanf("%d%d", &n, &m)) {
    for (int i = 1; i <= n; i++) id[i] = i;
    for (int i = 0; i < m; i++) e[i].read();
    en = ans = 0;
    solve(1, n); sort(E, E + en);
    int ans = 1;
    for (int i = 1; i < en; i++) if (E[i].w != E[i - 1].w) ans++;
    printf("%d\n", ans);
  } return 0;
}
```

3.11 hopcrof_karp

```
// Time O(V^0.5 * E)
// Left size - n, numbered from 1 to n
// Right size - m, numbered from 1 to m
struct Hopcroft {
  int n, m, que[N], dw[N << 1], mat[N << 1]; vector<int> e[N];
  void init(int _n, int _m) { n = _n, m = _m;
    rep(i, 1, n + 1) e[i].clear(); rep(i, 1, n + m + 1) mat[i] = 0;
  }
  void addEdge(int u, int v) { e[u].pb(n + v); }
  int bfs() { int flag = 0, qh = 0, qt = 0;
    rep(i, 1, n + m + 1) dw[i] = 0;
    rep(i, 1, n + 1) if (!mat[i]) que[qt++] = i;
    while (qh < qt) {
      int u = que[qh++];
      rep(i, 0, sz(e[u])) {
        int v = e[u][i]; if (dw[v]) continue;dw[v] = dw[u] + 1;
        if (!mat[v]) flag = true;
        else { dw[mat[v]] = dw[u] + 2; que[qt++] = mat[v]; }
      }
    }
    return flag;
  }
  int dfs(int u) {
    rep(i, 0, sz(e[u])) {
      int v = e[u][i]; if (dw[v] != dw[u] + 1) continue;
      dw[v] = 0;
      if (!mat[v] || dfs(mat[v])) {
        mat[u] = v, mat[v] = u; return true;
      }
    }
    return false;
  }
  int run() { int ret = 0;
    while (bfs()) rep(i, 1, n + 1) if (!mat[i]) ret += dfs(i);
    return ret;
  }
} G;
```

3.12 max_clique_fastest

```
const int N = 130;
typedef bool BB[N];
struct Maxclique {
  const BB *e; int pk, lv; db Tlimit;
  struct ve {int i, d; ve(int i): i(i),d(0) {}}; //ve : Vertex
  struct sc {int a, b; sc(     ): a(0),b(0) {}}; //sc : StepCount
  typedef vector<ve> ves; ves V;                 //ves: Vertices
  typedef vector<int> cc; cc  Q, QMAX;           //cc : ColorClass
  vector<cc> C; vector<sc> S;
  Maxclique(BB *conn, int sz, const db tt = 0.025): pk(0), lv(1), Tlimit(tt) {
```

```
    rep(i, 0, sz) V.pb(ve(i)); e = conn;
    C.resize(sz + 1); S.resize(sz + 1);
  }
  static bool desc_deg(const ve &a, const ve &b) { return a.d > b.d; }
  void ini_col(ves &v) { per(i, 0, sz(v)) v[i].d = min(i, v[0].d) + 1; }
  void set_deg(ves &v) { rep(i, 0, sz(v)){v[i].d = 0; rep(j, 0, sz(v)) v[i].d += e[v[i].
  i][v[j].i]; } }
  void deg_sort(ves &R) { set_deg(R); sort(all(R), desc_deg); }
  bool cut1(int pi , cc  &va) { rep(i, 0, sz(va)) if (e[pi][va[i]]) return true; return
  false; }
  void cut2(ves &va, ves &vb) { rep(i, 0, sz(va) - 1) if (e[va.back().i][va[i].i]) vb.pb
  (va[i].i); }
  void co_sort(ves &R) {
    int j = 0, maxno = 1, min_k = max(sz(QMAX) - sz(Q) + 1, 1);
    rep(i, 1, 3) C[i].clear();
    rep(i, 0, sz(R)) {
      int pi = R[i].i, k = 1;
      while (cut1(pi, C[k])) k++;
      if (k > maxno) C[(maxno = k) + 1].clear(); C[k].pb(pi);
      if (k < min_k) R[j++].i = pi;
    }
    if (j > 0) R[j - 1].d = 0;
    rep(k, min_k, maxno + 1) rep(i, 0, sz(C[k]))
      R[j].i = C[k][i], R[j++].d = k;
  }
  void exp_dyn(ves &R) {   // expand_dyn
    S[lv].a += S[lv - 1].a - S[lv].b;
    S[lv].b  = S[lv - 1].a;
    for (; sz(R); Q.pop_back(), R.pop_back()) {
      if (sz(Q) + R.back().d <= sz(QMAX)) return;
      Q.pb(R.back().i);
      ves Rp; cut2(R, Rp);
      if (sz(Rp)) {
        if ((db) S[lv].a / ++pk < Tlimit) deg_sort(Rp);
        co_sort(Rp); S[lv++].a++; exp_dyn(Rp); --lv;
      } else if (sz(Q) > sz(QMAX)) QMAX = Q;
    }
  }
  void mcqdyn(int *mxc, int &sz) { // mcqdyn(int maxclique, int &siz)
    set_deg(V); sort(all(V), desc_deg);
    ini_col(V); rep(i, 0, sz(V) + 1) S[i].a = S[i].b = 0;
    exp_dyn(V); per(i, 0, sz(QMAX)) mxc[i] = QMAX[i];
    sz = sz(QMAX);
  }
};
```

3.13 带花树匹配

```
const int MAXN = 45;
struct GraphMatch {
  int Next[MAXN] , spouse[MAXN] , parent[MAXN];
  int findb(int a) {return parent[a] == a ? a : parent[a] = findb(parent[a]);}
  void together(int a,int b) {parent[findb(a)]=findb(b);}
  vi E[MAXN]; int N , Q[MAXN] , bot , mark[MAXN] , visited[MAXN];
  int findLCA(int x,int y) {
    static int Ts = 0; Ts++;
    while (1) {
      if (x!=-1) {
        x = findb(x);
        if (visited[x] == Ts) return x;
        visited[x] = Ts;
        if (spouse[x] != -1) x = Next[spouse[x]]; else x = -1;
      }
      swap(x,y);
    }}
  void goup(int a,int p){
    while (a != p){
      int b = spouse[a], c = Next[b];
      if (findb(c) != p) Next[c] = b;
      if (mark[b] == 2) mark[Q[bot++] = b] = 1;
      if (mark[c] == 2) mark[Q[bot++] = c] = 1;
      together(a, b); together(b, c); a = c;
    }}
  void findaugment(int s){
    rep(i,0,N)
    { Next[i] = visited[i] = -1; parent[i] = i;mark[i] = 0; }
    Q[0] = s; bot = 1; mark[s] = 1;
    for (int head = 0; spouse[s] == -1 && head < bot; head++){
      int x = Q[head];
      rep(i,0,sz(E[x])) {
        int y = E[x][i];
        if (spouse[x] != y && findb(x) != findb(y) && mark[y] != 2){
          if (mark[y] == 1){
            int p = findLCA(x, y);
            if (findb(x) != p) Next[x] = y;
            if (findb(y) != p) Next[y] = x;
            goup(x, p); goup(y, p);
          } else if (spouse[y] == -1){
            Next[y] = x;
            for (int j = y; j != -1;){
              int k = Next[j]; int l = spouse[k];
              spouse[j] = k; spouse[k] = j; j = l;
            }
            break;
          }else{
            Next[y] = x; mark[Q[bot++] = spouse[y]] = 1;mark[y] = 2;
          }}}}}
  void init(int n) {N = n;rep(i,0,N) E[i].clear();}
  void addEdge(int a, int b) {E[a].pb(b);E[b].pb(a);}
  int maxMatch() {
    int ret = 0;
    rep(i,0,N) spouse[i]=-1;
    rep(i,0,N) if(spouse[i]==-1) findaugment(i);
    rep(i,0,N) if(spouse[i]!=-1) ++ret;
    return ret;
  }};
```

4 Math

4.1 ArithmeticProgressionXor

```
bool get(ll l,ll d,ll P,ll n){ bool res=0;
  res ^= (l / P) & n & 1;l %= P;
  res ^= (d / P) & (n >> 1) & 1;d %= P;
  if(d * n + l < P) return res;
  else return res ^ get((d * n + l) % P , P , d , (d * n + l) / P);
}
ll Xor(ll l,ll r,ll d){
  ll n=(r-l)/d+1,res=0;int u=r?63-__builtin_clzll(r):0;
  rep(i,0,u+1) if(get(l,d,1ll<<i,n)) res|=1ll<<i;
  return res;
}
```

4.2 FFT

```
namespace FFT{ // len = 2^x >= max(sz(a),sz(b))*2
  void fft(vir *F,int len,int o){
    int j = 0 , k , h;
    rep(i,0,len-1){ if(i < j) swap(F[i] , F[j]);
      for(k=len;j>=(k>>=1);j&=~k);  j|=k;
    }
    for(h=1;h<len;h<<=1){ vir wn(cos(pi*o/h),sin(pi*o/h));
      for(j=0;j<len;j+=h<<1){
        vir w(1.);
        for(k=j;k<j+h;++k,w=w*wn){
          vir b = w*F[k+h]; F[k+h] = F[k]-b , F[k] = F[k]+b;
        }}}
    if(o == -1) rep(i,0,len) F[i] = F[i]/(db)len;
  }
  void mult(vir *a,vir *b,int len){
    fft(a,len,1);fft(b,len,1);rep(i,0,len) a[i] = a[i] * b[i];
    fft(a,len,-1);
  }}
```

4.3 FFTmod

```
const db pi = acos(-1);
namespace FFT{
  const int N = 1<<18;
  struct vir {
    double r, i;
    vir() {r = i = 0;} vir(db r, db i) : r(r),i(i){}
    vir operator+(const vir &p) const {return vir(r+p.r,i+p.i);}
    vir operator-(const vir &p) const {return vir(r-p.r,i-p.i);}
    vir operator*(const vir &p) const {return vir(r*p.r-i*p.i,r*p.i+i*p.r);}
  };
  inline vir conj(const vir &p) {return vir(p.r, -p.i);}
  vir w[N], A[N], B[N], dfa[N], dfb[N], dfc[N], dfd[N];
  int L,n,bitrev[N];
  void init(int len) {
    L = 0;while(1<<L<=len) ++L;
    n=1<<L;
    rep(i,0,n) bitrev[i] = (bitrev[i>>1]>>1)|((i&1)<<(L-1));
    rep(i,0,n) w[i] = vir(cos(2*pi*i/n), sin(2*pi*i/n));
  }
  void fft(vir *a, const int &n) {
    rep(i,0,n) if (i < bitrev[i]) swap(a[i], a[bitrev[i]]);
    for (int i=2,d=n>>1;i<=n;i<<=1,d>>=1)
      for (int j=0;j<n;j+=i) {
        vir *l=a+j,*r=a+j+(i>>1),*p=w;
        for (int k=0;k<(i>>1);++k) {
          vir tmp=(*r)*(*p);
          *r=*l-tmp,*l=*l+tmp;
          ++l,++r,p+=d;
        }}}
  vi mul(const vi &a, const vi &b) {
    if((sz(a)<=100 && sz(b)<=100) || min(sz(a),sz(b))<=5){
      vi res(sz(a)+sz(b)-1,0);
      rep(i,0,sz(a)) rep(j,0,sz(b)) pp(res[i+j],1ll*a[i]*b[j]%P);
      return res;
    }
    init(sz(a)+sz(b));
    rep(i,0,n) A[i]=B[i]=vir(0,0);
    rep(i,0,sz(a)) A[i]=vir(a[i]&32767,a[i]>>15);
    rep(i,0,sz(b)) B[i]=vir(b[i]&32767,b[i]>>15);
    fft(A,n),fft(B,n);
    rep(i,0,n) {
      int j=(n-i)&(n-1);
      static vir da, db, dc, dd;
      da = (A[i] + conj(A[j])) * vir(0.5, 0);
      db = (A[i] - conj(A[j])) * vir(0, -0.5);
      dc = (B[i] + conj(B[j])) * vir(0.5, 0);
      dd = (B[i] - conj(B[j])) * vir(0, -0.5);
      dfa[j] = da * dc, dfb[j] = da * dd;
      dfc[j] = db * dc, dfd[j] = db * dd;
    }
    rep(i,0,n) {
      A[i]=dfa[i]+dfb[i]*vir(0, 1);
      B[i]=dfc[i]+dfd[i]*vir(0, 1);
    }
    fft(A,n),fft(B,n); vi ret(n,0);
    rep(i,0,n) {
      ll da = (ll) (A[i].r / n + 0.5) % P;
      ll db = (ll) (A[i].i / n + 0.5) % P;
      ll dc = (ll) (B[i].r / n + 0.5) % P;
      ll dd = (ll) (B[i].i / n + 0.5) % P;
      pp(ret[i], (da + ((db + dc) << 15) + (dd << 30)) % P);
    }
    return ret;
  }}
```

4.4 FWT

```
const int P = 1e9 + 7; // P is prime
int inv(int x){return x==1?x:P-ll(P/x)*inv(P%x)%P;}
```

```
void FWT(int *a,int len,int o){ // o : 1(+) , -1(-)
  int inv2=inv(2);
  for(int k=0;1<<k<len;++k) rep(i,0,len) if(~i>>k&1){
    int j=i^(1<<k),x,y;
    x=(a[i]+a[j])%P , y=(a[i]-a[j]+P)%P; // xor
    if(o==-1) x=ll(x)*inv2%P , y=ll(y)*inv2%P;
    //x=(a[i]+a[j])%P , y=a[j]; // and
    //if(o==-1) x=(a[i]-a[j])%P;
    //x=a[i] , y=(a[i]+a[j])%P; // or
    //if(o==-1) y=(a[j]-a[i])%P;
    a[i]=x,a[j]=y;
  }}
```

4.5 NTT

```
namespace NTT { // init!! , G is root , B is base , len = 2^x
  const int N = (1<<19) , P = (479 << 21) + 1, G = 3 , B = 2;
  int w[2][N],rev[N];
  ll Pow(ll x, ll t){ll r=1;for(;t;t>>=1,x=x*x%P)if(t&1)r=r*x%P;return r;}
  void ini(){
    ll t = Pow(G,(P-1)/N);
    w[0][0] = w[1][0] = 1;
    rep(i,1,N) w[0][i] = t*w[0][i-1]%P;
    rep(i,1,N) w[1][i] = w[0][N-i];
    rep(i,0,N) for(int j=1;j<N;j*=B) (rev[i]<<=1)|=(i/j)%B;
  }
  void ntt(int *a,int n,int o){
    int tt = N/n;
    rep(i,0,n)
    { int j = rev[i]/tt; if(i<j) swap(a[i],a[j]); }
    for(int i=1;i<n;i<<=1)
      for(int j=0,t=N/(i+i);j<n;j+=i+i)
        for(int k=j,l=0;k<j+i;++k,l+=t){
          int b = (ll)a[k+i]*w[o][l]%P;
          a[k+i] = a[k]-b; if(a[k+i]<0) a[k+i]+=P;
          a[k] = a[k]+b; if(a[k]>=P) a[k]-=P;
        }
    if(o == 1)
    { ll inv = Pow(n,P-2); rep(i,0,n) a[i] = a[i]*inv%P;}
  }
  void mult(int *a,int *b,int len){
    ntt(a,len,0);ntt(b,len,0);rep(i,0,len) a[i] = (ll)a[i]*b[i]%P;
    ntt(a,len,1);
  }
}
```

4.6 NimProduct

```
typedef unsigned int ui;
namespace NimProduct { // Build first
  const int N = 1 << (1 << 3) , M = 1 << 3;
  ui sg[N][N] , pw[5];
  inline ui Product(ui x,ui y){
    if(x < y) swap(x , y);
    if(x < N && ~sg[x][y]) return sg[x][y];
    int base = 4; while(base > 0 && x < pw[base]) --base;
    ui M = 1 << (1 << base);
    ui a = x >> (1 << base) , b = x & (M - 1);
    ui c = y >> (1 << base) , d = y & (M - 1);
    ui c1 = Product(a, c),c2 = Product(a^b, c^d),c3 = Product(b, d);
    ui res = ((c2^c3) << (1<<base)) | (Product(c1,M>>1)^c3);
    if(x < N) sg[x][y] = res; return res;
  }
  void Build(){
    rep(i,0,5) pw[i] = 1<<(1<<i); memset(sg , -1 , sizeof(sg));
    rep(i,0,N) sg[1][i]=sg[i][1]=i,sg[i][0]=sg[0][i]=0;
    rep(i,0,N) rep(j,0,N) Product(i,j);
  }
}
```

4.7 Simpson

```
namespace Simpson {
  const double eps = 1e-8;
  inline double F(double x) { //F(x) = ? }
  inline double simpson(double fa, double fb, double fc, double a, double c) { return (
  fa + 4 * fb + fc) * (c - a) / 6; }
  double asr(double a, double b, double c, double esp, double A, double fa, double fb,
  double fc) {
    double ab = (a + b) / 2, bc = (b + c) / 2;
    double fab = F(ab), fbc = F(bc);
    double L = simpson(fa, fab, fc, a, b), R = simpson(fb, fbc, fc, b, c);
    if (fabs(L + R - A) <= 15 * eps) return L + R + (L + R - A) / 15.0;
    return asr(a, ab, b, esp / 2, L, fa, fab, fb) + asr(b, bc, c, esp / 2, R, fb, fbc,
    fc);
  }
  //f(a, c)
  double asr(double a, double c, double eps) {
    double b = (a + c) / 2;
    double fa = F(a), fb = F(b), fc = F(c);
    return asr(a, b, c, eps, simpson(fa, fb, fc, a, c), fa, fb, fc);
  }
}
```

4.8 guass_linear_base

```
// b[i] : the base of i-th bit
// time : O(n * B + B * B)
struct GuassLB { // !!!! : use before ini() !
  static const int B = 64; ll b[B] , sz;
  void ini()  { fill_n(b, B, 0); }
  void add(ll a) {
    per(i, 0, B) if((a >> i) & 1) {
      if (b[i]) a ^= b[i]; else {
        b[i] = a;
        per(j, 0, i) if(b[j] && (b[i] & pw(j))) b[i] ^= b[j];
        per(j, i + 1, B) if (b[j] & pw(i)) b[j] ^= b[i];
        break;
```

```
    }}}
  ll kth(ll k) { vi v; rep(i, 0, B) if (b[i]) v.pb(i);
    ll r = 0, mask = pw(sz(v)) - 1 - k;
    rep(i,0,sz(v)) if(mask & pw(i)) r ^= b[v[i]];
    return r;
  }}gs;
```

4.9 linear_recurrence

```
// a_{m} = \sum_{j=0}^{m-1}a_{j}*c_{j} O(m^2lgn)
ll linear_recurrence(ll n,int m,ll*a,ll*c,int P){
  vector<ll> v(m,0),u(m<<1,0); v[0]=1;
  for(ll x=0,W=n?1ll<<(63-__builtin_clzll(n)):0;W;W>>=1,x<<=1){
    fill(all(u),0); int b = !!(n & W); if(b) x++;
    if(x<m) u[x]=1;
    else{
      rep(i,0,m) rep(j,0,m) (u[i+b+j]+=v[i]*v[j])%=P;
      per(i,m,2*m) rep(j,0,m) (u[i-m+j]+=c[j]*u[i])%=P;
    }
    copy(u.begin(),u.begin()+m,v.begin());
  }
  ll ans=0; rep(i,0,m) (ans+=v[i]*a[i])%=P; return ans;
}
```

4.10 number_theory

```
namespace number_theory{
  int Phi(int n){
    int res(n);
    for(int i=2;i*i<=n;++i) if(n%i==0){
      res=res/i*(i-1); while(n%i==0) n/=i;
    }
    if(n>1) res=res/n*(n-1); return res;
  }
  ll Pow(ll x,ll t,ll P){ll r=1;for(;t;t>>=1,x=x*x%P)if(t&1)r=r*x%P;return r;}
  void exgcd(ll a,ll b,ll&x,ll&y)
  { if(!b) x = 1 , y = 0; else exgcd(b,a%b,y,x) , y -= a/b*x; }
  const int P = 1e9 + 7;
  int inverse(int x){return x == 1 ? 1 : P - ll(P/x)*inverse(P%x)%P;}
  ll inverse(ll a,ll b){ll x,y;exgcd(a,b,x,y);return (x%b+b)%b;}
  // minimal non-negative x satisfied a^x%m = r, if not exist return -1
#include <unordered_map>
  ll log(ll a, ll r, ll m){
    if(r >= m) return -1;
    for(ll i=0,x=1%m;i<50;++i,x=x*a%m) if(x==r) return i;
    ll c=0,g=1,x=1,lcof=1;
    for(;(g=__gcd(a,m))!=1;r/=g,m/=g,lcof=lcof*(a/g)%m,++c)
      if(r%g) return -1;
    r=r*inverse(lcof,m)%m;
    std::unordered_map<ll,ll> u;
    g = Phi(m), u[x] = 0;
    int q = sqrt(g) + 2;
    g = Pow(a, g - q % g, m);
    rep(i,1,q){
      x=x*a%m; u.insert(mp(x,i));
      if(x == r) return c + i;
    }
    rep(i,1,q){
      r=r*g%m; auto t = u.find(r);
      if(t != u.end()) return c + i * q + t->se;
    }
    return -1;
  }
  // x = r[i] (mod m[i]) (0 <= i < n)
  // m[] pairwise co-prime
  ll crt(int n, ll *r, ll *m) {
    ll M = 1 , ans = 0;
    rep(i,0,n) M *= m[i];
    rep(i,0,n) {
      ll t=M/m[i],invt=inverse(t,m[i]);
      (ans += t*invt*r[i]%M) %= M;
    }
    return ans;
  }
  // x = r[i] (mod m[i]) (0 <= i < n)
  // return -1 when solution doesn't exist
  ll congruences(int n, ll *r, ll *m){
    ll ans = r[0], LCM = m[0];
    rep(i,1,n){
      ll g = __gcd(LCM , m[i]) , x = inverse(LCM , m[i]);
      if ((r[i] - ans) % g) return -1;
      ll tmp = ((r[i] - ans) / g * x) % (m[i] / g);
      (ans += LCM * tmp) %= (LCM / g * m[i]);
      LCM = LCM / g * m[i];
    }
    return (ans+LCM)%LCM;
  }}
```

4.11 pie

```
namespace e { int digits[1000];
    string get(int len) {
        rep(i,1,1000) digits[i] = 1;
        digits[1] = 2; string res=""; int d , carry;
        rep(i,0,len) {
            res+=char(digits[1]+'0'); digits[1] = carry = 0;
            per(i,2,1000) {
                d = digits[i] * 10 + carry;
                digits[i] = d % i; carry = d / i;
            }
            digits[1] = carry;
        }
        return res;
    }}
namespace pi { int a = 10000,b,c,d,e,f[14*700+1],g;
    string get(int len) { // len = 4 * k
        string res="";
        c = len / 4 * 14;
        while(b-c) f[b++] = a/5;
```

```
        for(;d = 0,g = c*2;c -= 14,printf("%.4d",e+d/a),e = d%a)
            for(b = c;d += f[b]*a , f[b] = d%--g , d /= g-- , --b ;d *= b);
        return res;
    }}
```

4.12 prinum

```
ll f[340000],g[340000],n,k;//f[i] means pi(n/i),g[i] means pi(i)
ll dp(ll n){
  ll i,j,m,res=0;
  for(m=1;m*m<=n;++m)f[m]=n/m-1;
  for(i=1;i<=m;++i)g[i]=i-1;
  for(i=2;i<=m;++i){ if(g[i]==g[i-1])continue;
    for(j=1;j<=min(m-1,n/i/i);++j){
      if(i*j<m)f[j]-=f[i*j]-g[i-1];
      else f[j]-=g[n/i/j]-g[i-1];
    }
    for(j=m;j>=i*i;--j)g[j]-=g[j/i]-g[i-1];
  }
  return f[1];
}
```

4.13 simplex

```
const double EPS = 1e-8;
const double DINF = 1e15;
struct Simplex {
  int n, m, B[MAXN], N[MAXN];
  db v, ans[MAXN], b[MAXN], c[MAXN], A[MAXN][MAXN];
  /* n - variables, m - equations
   * max f(x) = cx
   * s.t. Ax <= b, x >= 0    */
  void init(int _n, int _m) { n = _n, m = _m, v = 0;
    rep(i, 1, n + 1) N[i] = i; rep(i, 1, m + 1) B[i] = i + n;
  }
  inline int sgn(db x) { return (x > EPS) - (x < -EPS); }
  void pivot(int l, int e) {
    db tmp = A[l][e];
    b[l] /= tmp, A[l][e] = 1 / tmp;
    rep(i, 1, n + 1) if (i != e) A[l][i] /= tmp;
    rep(i, 1, m + 1) if (i != l) {
      b[i] -= A[i][e] * b[l];
      rep(j, 1, n + 1) A[i][j] -= (j != e) * A[i][e] * A[l][j];
      A[i][e] = - A[i][e] / tmp;
    }
    rep(i, 1, n + 1) c[i] -= (i != e) * c[e] * A[l][i];
    v += b[l] * c[e];c[e] *= -A[l][e];swap(B[l], N[e]);
  }
  db run() {
    while (1) {
      int r, l, e = -1; db delt = -DINF;
      rep(j, 1, n + 1) if (sgn(c[j]) > 0) {
        db tmp = DINF;
        rep(i, 1, m + 1) if (sgn(A[i][j]) > 0 && b[i] / A[i][j] < tmp)
          r = i, tmp = b[i] / A[i][j];
        if (delt < tmp * c[j]) l = r, e = j, delt = tmp * c[j];
      }
      if (e == -1) break; pivot(l, e);
    }
    rep(i, 1, n + 1) rep(j, 1, m + 1)
      if (B[j] == i) { ans[i] = (j <= m ? b[j] : 0); break; }
    return v;
  }
} sp;
```

5 Others

5.1 FastMul vimrc zeller

```
inline ll mul(ll a,ll b){return (a*b-ll((long double)a*b/P+0.5)*P+P)%P;}
set nu ai ci si mouse=a ts=4 sts=4 sw=4
nmap<F2> : vs %<.in <CR>
nmap<F7> : !g++ -o %< %<.cpp -std=c++11 <CR>
nmap<F8> : !./%< < %<.in <CR>
nmap<F9> : make %< <CR>
nmap<F3> : !gedit %<.cpp <CR>
int zeller(int y,int m,int d) {
  if(m <= 2) y--,m+=12;int c=y/100;y%=100;
  int w=((c>>2))-(c<<1)+y+(y>>2)+(13*(m+1)/5)+d-1)%7;
  if(w<0) w+=7; return w;
}
```

5.2 expression_parse

```
#define ass(x) {if(!(x)) throw 1;}
string t; string::iterator p; using val = int;
int pri(char ch){
  if(ch == '*') return 3; if(ch == '+') return 2;
  if(ch == '-') return 2; if(ch == '$') return -1;
}
val cal(char ch,val a,val b){ if(ch == '+') return a + b; }
val next();
val expr(char pre,val x){
  if(pri(pre) < pri(*p))
  {auto op = *p++; return expr(pre , cal(op , x , expr(op , next())));}
  else return x;
}
val next(){
  if(*p == '('){
    p++;val res = expr('(' , next());ass(*p++ == ')');return res;
  } else if(isdigit(*p)){
    val x = 0;
    while(isdigit(*p))
    { ass(*p == '0' || *p == '1');x = x * 2 + *p++ - '0'; }
    return x;
  } else if(*p == '-')
  { p++; return -next(); }
```

```
}
bool check(){ // s$
  try{ if(count(all(t) , '=') != 1) throw 1;
    p = t.begin(); int res = expr('$' , next());
    return res == 1;
  } catch(...){ return 0; }
}
```

6 String

6.1 ACAutomaton

```
struct Trie{  // [0,L) , N-1 is virtual , 0 is rt , init!!
    static const int N = 101010 , M = 26;
    int ne[N][M] , fail[N] , fa[N] , rt , L;
    void ini(){ fill_n(ne[fail[0] = N-1],M,0);L = 0;rt = newnode();}
    int newnode(){ fill_n(ne[L],M,0); return L++; }
    void add(char *s){
        int p = rt;
        for(int i=0;s[i];++i){
            int c = s[i] - 'a';// modify
            if(!ne[p][c]) ne[p][c] = newnode() , fa[L-1] = p;
            p = ne[p][c];
        }
    }
    void Build(){
        vi v;v.pb(rt);
        rep(i,0,sz(v)){
            int c = v[i];
            rep(i,0,M) ne[c][i] ?
                v.pb(ne[c][i]) , fail[ne[c][i]] = ne[fail[c]][i] :
                ne[c][i] = ne[fail[c]][i];
        }}};
```

6.2 DoublingArray

```
namespace Doubling{
  static const int N = 101010;
  int t[N] , wa[N] , wb[N] , sa[N] , h[N];
  void sort(int *x,int *y,int n,int m){
    rep(i,0,m) t[i] = 0;rep(i,0,n) t[x[y[i]]]++;rep(i,1,m) t[i] += t[i-1];
    per(i,0,n) sa[--t[x[y[i]]]] = y[i];
  }
  bool cmp(int *x,int a,int b,int d)
  { return x[a] == x[b] && x[a+d] == x[b+d];}
  void da(int *s,int n,int m){
    int *x=wa,*y=wb;
    rep(i,0,n) x[i] = s[i] , y[i] = i;
    sort(x , y , n , m);
    for(int j=1,p=1;p<n;m=p,j<<=1){
      p = 0;rep(i,n-j,n) y[p++] = i;
      rep(i,0,n) if(sa[i] >= j) y[p++] = sa[i] - j;
      sort(x , y , n , m); swap(x , y);p = 1;x[sa[0]] = 0;
      rep(i,1,n) x[sa[i]] = cmp(y,sa[i],sa[i-1],j)?p-1:p++;
    }
  }
  void cal_h(int *s,int n,int *rk){
    int j,k=0;
    for(int i=1;i<=n;++i) rk[sa[i]] = i;
    for(int i=0;i<n;h[rk[i++]] = k)
      for(k&&--k,j=sa[rk[i]-1];s[i+k]==s[j+k];++k);
  }
}
struct DA{ // [0,n] , in[n] = 0 , n load
  static const int N = 101010;
  int p[18][N] , rk[N] , in[N] , Log[N] , n;
  void Build(){
    Doubling::da(in,n+1,300); Doubling::cal_h(in,n,rk);
    Log[0] = -1;for(int i=1;i<=n;++i) Log[i] = Log[i-1] + (i==(i&(-i)));
    for(int i=1;i<=n;++i) p[0][i] = Doubling::h[i];
    for(int j=1;1<<j<=n;++j){
      int lim = n+1-(1<<j);
      for(int i=1;i<=lim;++i)
        p[j][i] = min(p[j-1][i] , p[j-1][i+(1<<j>>1)]);
    }
  }
  int lcp(int a,int b){
    a = rk[a] , b = rk[b]; if(a > b) swap(a , b);++a;
    int t = Log[b-a+1]; return min(p[t][a] , p[t][b-(1<<t)+1]);
  }
};
```

6.3 Exkmp

```
void exkmp(char *s,int *z,char *t,int *p){
  int lens = strlen(s); int lent = strlen(t); p[0]=0;
  for(int i=0,x=0,y=0;i<lens;++i){
    z[i] = i <= y ? min(y-i,p[i-x]) : 0;
    while(i + z[i] < lens && z[i] < lent && s[i + z[i]] == t[z[i]]) ++z[i];
    if(y <= i + z[i]) x = i, y = i + z[i];
  }}
void Exkmp(){scanf("%s%s",s,t);exkmp(t+1,nt+1,t,nt);exkmp(s,ns,t,nt);}
```

6.4 Kmp

```
void kmp(char *s,int *ns,char *t,int *nt){
  int lens = strlen(s);int lent = strlen(t);nt[0] = -1;
  for(int i=0,j=-1;i<lens;++i){
    while(j >= 0 && s[i] != t[j + 1]) j = nt[j];
    if(s[i] == t[j + 1]) ++j;
    ns[i] = j;if(j + 1 == lent) j = nt[j];
  }}
void KMP(){scanf("%s%s",s,t);kmp(t+1,nt+1,t,nt);kmp(s,ns,t,nt);}
```

6.5 Manacher

```
// length of pa is two size of str
// pa[i<<1] : odd string
// pa[i<<1|1] : even string
void Manacher(char *s,int n,int *pa){ pa[0] = 1;
  for(int i=1,j=0;i<(n<<1)-1;++i){
    int p = i >> 1 , q = i - p , r = ((j + 1)>>1) + pa[j] - 1;
    pa[i] = r < q ? 0 : min(r - q + 1 , pa[(j<<1) - i]);
    while(0 <= p - pa[i] && q + pa[i] < n && s[p - pa[i]] == s[q + pa[i]])
      pa[i]++;
    if(q + pa[i] - 1 > r) j = i;
  }}
```

6.6 PalindromicTree

```
struct Palindromic_Tree{ // [0,p) , 0(even) and 1(odd) is virtual , init!!
  static const int N = 101010 , M = 26;
  int ne[N][M] , fail[N] , len[N] , S[N] , last , n , p;
  int newnode(int l)
  { fill(ne[p] , ne[p] + M , 0);len[p] = l;return p++; }
  void ini()
  { p = 0;newnode(0);newnode(-1);S[n = last = 0] = -1;fail[0] = 1; }
  int get_fail(int x)
  { while(S[n - len[x] - 1] != S[n]) x = fail[x]; return x; }
  void add(int c){ S[++n] = c; int cur = get_fail(last);
    if(!ne[cur][c]){ int now = newnode(len[cur] + 2);
      fail[now] = ne[get_fail(fail[cur])][c]; ne[cur][c] = now;
    }
    last = ne[cur][c];
  } };
```

6.7 SuffixAutomaton

```
struct SAM{// [0,L] , 0 is virtual , 1 is rt , init!!
  static const int N = 101010 , M = 26;
  int par[N] , l[N] , ne[N][M]; int rt , last , L;
  void add(int c){ int p = last , np = ++L;
    fill(ne[np] , ne[np] + M , 0);
    l[np] = l[p] + 1; last = np;
    while(p && !ne[p][c]) ne[p][c] = np , p = par[p];
    if(!p) par[np] = rt;
    else{ int q = ne[p][c];
      if(l[q] == l[p] + 1) par[np] = q;
      else{ int nq = ++L;
        l[nq] = l[p] + 1;copy(ne[q] , ne[q] + M , ne[nq]);
        par[nq] = par[q]; par[q] = par[np] = nq;
        while(p && ne[p][c] == q) ne[p][c] = nq , p = par[p];
      }}}
  void ini()
  { rt = last = L = 1;fill(ne[rt], ne[rt] + M, 0);l[0] = -1; }
};
```

6.8 SuffixTree

```
// init!! , go[0] is virtual , add 0 in the end of string
const int N = 101010 , C = 27 , inf = ~0U>>1; int pos,S[N];
struct SuffixTree{
  struct Node{ int l , r , du; Node *fail, *go[C], *fa;
    Node(int l=-1,int r=inf) : l(l),r(r)
    { fail = fa = NULL; du = 0; memset(go,0,sizeof(go)); }
    Node* link(Node*t){int c=S[t->l];du+=!go[c];go[c]=t;t->fa=this;return t;}
    int len(){return min(r,pos+1)-l;}
  }pool[N<<2],*pl,*rt,*p,*pre;
  int L,R; ll size; queue<Node*> leaves;
  void ini(){
    pos=-1;pl=pool;rt=p=new(pl++) Node(-1,-1);pre=NULL;
    L=R=0;size = 0; while(sz(leaves)) leaves.pop();
  }
  void jump(Node*u){ if(pre) pre->fail = u;pre = u;}
  bool walk(Node*u){
    int len=u->len(); if(R >= len) return L+=len,R-=len,p=u,true;
    return false;
  }
  void extend(int c){
    S[++pos] = c; pre = NULL;
    for(;;){
      int ch = S[L = R ? L : pos];
      if(p->go[ch]){
        Node*q = p->go[ch];
        if(walk(q)) continue;
        if(S[q->l + R] == c){ ++R; jump(p); break; }
        Node *s = new(pl++) Node(q->l,q->l+R);
        leaves.push(s->link(new(pl++) Node(pos)));
        q->l += R; p->link(s)->link(q);
        jump(s);
      }
      else leaves.push(p->link(new(pl++) Node(pos))) , jump(p);
      if(p == rt && !R) break;
      else if(p == rt) L = pos - --R;
      else p = p->fail ? p->fail : rt;
    }
    size += sz(leaves);
  }
  void eraseUp(Node*&u)
  {size -= u->len();u->fa->go[S[u->l]] = NULL;--((u=u->fa)->du);}
  void erase(){
    Node*u = leaves.front(); leaves.pop();
    while(!u->du && u != p) eraseUp(u);
    if(u == p){
      if(!p->du && !R){
        L = pos - (R = p->len()) + 1;
        p = p->fa; eraseUp(u);
      }
      if(R && !p->go[S[L]]){
        Node *leaf = new(pl++) Node(L);
        leaves.push(p->link(leaf));
        size += leaf->len();
        if(p == rt && R) L = pos - --R + 1;
```

```
      else p = p->fail ? p->fail : rt;
    }}}
  int stop , ord[N<<1] , rk[N];
  void dfs(Node*u)
  {ord[u - pool] = stop++;rep(i,0,C) if(u->go[i]) dfs(u->go[i]);}
  void getrk(){
    stop = 0;dfs(rt);
    for(int i=0;sz(leaves);++i)
      rk[i] = ord[leaves.front() - pool] , leaves.pop();
  }};
```

6.9 最小表示法

```
int solve(char *str) { int i, j, l, n = strlen(str); i = 0; j = 1;
  while (i < n && j < n) {
    for (l = 0; l < n; l++)
      if (str[(i + l) % n] != str[(j + l) % n]) break;
    if (l >= n) break;
    if (str[(i + l) % n] > str[(j + l) % n]) i = i + l + 1;
    else j = j + l + 1;
    if (i == j) j = i + 1;
  }
  if (i < j) return i; return j;
}
```

7 Tree

7.1 Centroid

```
namespace Centriod { // id starts from 1
  const int N = 101010; int vis[N],sz[N];
  void dfssz(int c,int fa,int Sz,int &rt){ sz[c] = 1;
    for(auto t : g[c]) if(!vis[t]&&t!=fa) dfssz(t,c,Sz,rt) , sz[c]+=sz[t];
    if(!rt && sz[c]*2>Sz) rt=c;
  }
  void dfs(int c){ int rt=0;dfssz(c,0,0,rt);dfssz(c,0,sz[c],rt=0);
    // cal something
    vis[rt] = true; for(auto t : g[rt]) if(!vis[t]) dfs(t);
  }};
```

7.2 HeavyChain

```
struct HeavyChain{ // id starts with 1
  static const int N = 100005 , inf = ~0U>>1;
  int sz[N] , wson[N] , top[N] , dep[N] , id[N] , _ , par[N] , who[N];
  void dfs(int c,int fa,vi g[]){
    sz[c]=1;dep[c]=dep[par[c]=fa]+1;int &s=wson[c]=top[c]=0;
    for(auto t:g[c]) if(t!=fa)
      dfs(t,c,g),sz[c]+=sz[t],(sz[t]>=sz[s])&&(s=t);
  }
  void dfs2(int c,int fa,vi g[]){
    who[id[c]=++_]=c;int s=wson[c];
    if(!top[c]) top[c]=c;
    if(s) top[s]=top[c],dfs2(s,c,g);
    for(auto t:g[c]) if(t!=fa&&t!=s) dfs2(t,c,g);
  }
  void Query(int a,int b){// info in points
    int fa=top[a],fb=top[b];
    while(fa!=fb){
      if(dep[fa]<dep[fb]) swap(a,b) , swap(fa,fb);
      // Cal id[fa] .. id[a]
      a = par[fa];fa = top[a];
    }
    if(dep[a]<dep[b]) swap(a,b);
    // Cal id[b] .. id[a]
  }
  void Build(vi g[]){ dfs(1,0,g);_=0;dfs2(1,0,g);}
}hc;
```

7.3 LCARMQ

```
struct LCARMQ{ // N is 2 size of tree , id of nodes start from 1
  static const int N = 101010 << 1;
  int a[20][N] , lft[N] , dep[N] , lg[N] , L;
  int rmin(int x,int y){return dep[x] < dep[y] ? x : y;}
  void add(int x){ a[0][L++] = x;}
  void dfs(int c,int fa,const vi g[]){
    lft[c]=L;add(c);
    for(auto t : g[c]) if(t!=fa) dep[t]=dep[c]+1,dfs(t,c,g),add(c);
  }
  void Build(const vi g[]){
    L = 0;dfs(1,0,g);
    rep(i,2,L) lg[i]=lg[i>>1]+1;
    rep(i,1,20){ int lim = L+1-(1<<i);
      rep(j,0,lim) a[i][j] = rmin(a[i-1][j] , a[i-1][j+(1<<i>>1)]);
    }}
  int lca(int x,int y){
    x = lft[x] , y = lft[y];if(x > y) swap(x , y);
    int i = lg[y-x+1];return rmin(a[i][x] , a[i][y+1-(1<<i)]);
  }};
```

7.4 LongChain

```
struct LongChain{ // id starts with 1
  static const int N = 100005 , inf = ~0U>>1;
  int wson[N] , top[N] , dep[N] , lg[N];
  int jump[N][20] , id[N] , who[N] , rwho[N] , _;
  void dfs(int c,int fa,vi g[]){
    dep[c]=1;int &s=wson[c]=top[c]=0;
    jump[c][0]=fa;rep(i,1,20) jump[c][i]=jump[jump[c][i-1]][i-1];
    for(auto t:g[c]) if(t!=fa)
      dfs(t,c,g),dep[c]=max(dep[t]+1,dep[c]),(dep[t]>=dep[s])&&(s=t);
  }
  void dfs2(int c,int fa,int rc,vi g[]){
    if(!top[c]) top[c]=c,rc=c;
    who[id[c]=++_]=c;rwho[_]=rc;
    int s=wson[c];
```

```
    if(s) top[s]=top[c],dfs2(s,c,jump[rc][0],g);
    for(auto t:g[c]) if(t!=fa&&t!=s) dfs2(t,c,t,g);
  }
  void Build(vi g[])
  {dfs(1,0,g);_=0;dfs2(1,0,1,g); rep(i,2,N) lg[i]=lg[i>>1]+1;}
  int kth_par(int x,int k){ // kth_par should exist
    if(k==0) return x;
    int j0=1<<lg[k];int p0=jump[x][lg[k]];
    int j1=k-j0;int del=id[p0]-id[top[p0]];
    if(del>=j1) return who[id[p0]-j1];
    else return rwho[id[top[p0]]+j1-del];
  }}hc;
```

7.5 QuerySubtree

```
namespace QuerySubtree{ // id starts with 1
  static const int N = 100005 , inf = ~0U>>1;
  int sz[N] , wson[N] , par[N];
  void dfs(int c,int fa,vi g[]){
    sz[c]=1;par[c]=fa;int &s=wson[c]=0;
    for(auto t:g[c]) if(t!=fa)
      dfs(t,c,g),sz[c]+=sz[t],(sz[t]>=sz[s])&&(s=t);
  }
  void solve(int c,int fa,bool iswson,vi g[]){
    for(auto t : g[c]) if(t != wson[c]) solve(t , c , false , g);
    if(wson[c]) solve(wson[c] , c , true , g);
    for(auto t : g[c]) if(t != wson[c]){
      // query // add
    }
    if(!iswson) ;// del
  }
  void solve(vi g[]){ dfs(1,0,g);solve(1,0,false,g); }
}
```

7.6 VTree

```
namespace Vtree{// some nodes remain
  const int N = 101010;
  int tp[N] , _; vi g[N];// nodes sorted in dfs order
  void solve(vi&v,LCARMQ&R){
    _ = 0; vi del;del.pb(tp[_++] = v[0]);
    rep(i,1,sz(v)){
      int lca = R.lca(tp[_-1] , v[i]);
      vi l;while(_ > 0 && R.dep[lca] < R.dep[tp[_-1]]) l.pb(tp[--_]);
      if(_ == 0 || lca != tp[_-1]) del.pb(tp[_++] = lca);
      l.pb(tp[_-1]);del.pb(tp[_++] = v[i]);
      rep(i,1,sz(l)) g[l[i]].pb(l[i-1]);
    }
    rep(i,0,_-1) g[tp[i]].pb(tp[i+1]);
    // dfs()
    for(auto t : del){
      // Cal()
      g[t].clear();
    }}}
```

7.7 tree_modui

```
int B; // block size, B = sqrt(n)
int dfs(int u, int p) { static int blocks = 0;
  dep[u] = dep[p] + 1; dfn[u] = ++timeStamp; int size = 0;
  rep(i, 0, sz(e[u])) {
    int v = e[u][i]; if (v == p) continue;
    size += dfs(v, u);
    if (size >= B)
    { ++blocks; while (size--) bid[sta[top--]] = blocks; }
  }
  sta[++top] = u;
  if (u == 1)  // 1 is root
  {++blocks; while (top) bid[sta[top--]] = blocks;}
  return size + 1;
}
void flip(int u) {//change status of u,if vis[u] remove else insert}
void gao(int u, int v) {
  while (u != v) { if (dep[u] < dep[v]) swap(u, v);
    flip(u); u = f[u][0]; // direct father
  }}
struct Query {
  int u, v, id, bid; // dfn[u] < dfn[v], bid = bid[u]
  bool operator<(const Query&q) const { // order(bid, dfn[v])
    if (bid != q.bid) return bid < q.bid;
    return dfn[v] < dfn[q.v];
  }} q[M]; // M operations
int main() {
  sort(q, q + m);int u = 1, v = 1, lca = 1;flip(lca);
  rep(i, 0, m) {
    flip(lca);gao(u, q[i].u), gao(v, q[i].v);
    u = q[i].u, v = q[i].v;flip(lca = LCA(u, v));
    ans[q[i].id] = nowAns;
  }
  return 0; }
```

素数表
十亿以下 122420729, 163227661, 217636919, 290182597, 386910137, 515880193, 687840301, 917120411
十亿以上 1222827239, 1610612741, 3221225473ll, 4294967291ll
NTT(g=3):【167772161】【469762049】【998244353】【1004535809】【2281701377】
卡特兰数 $F_n=\sum_{i=0}^{n-1}f_i*f_{n-i-1}\|F_0=1\|F_{n+1}=\frac{2(2n+1)}{n+2}*F_n\|F_n=C_{(2n)}^{(n)}-C_{(2n)}^{(n+1)}*\frac{C_{(2n)}^{(n)}}{n+1}$
威尔逊定理 $p\epsilon\mathbb{P}\rightarrow((p-1)!+1)\equiv 0\,mod\,p$ **欧拉公式**：简单多面体 $V-E+F=2$
三角形外接圆半径 $R=\frac{abc}{4S}$ **海伦公式** $p=\frac{a+b+c}{2}\|S=\sqrt{s(s-a)(s-b)(s-c)}$
$(p-1)!\equiv -1(modP)$

1	2
$sin(a\pm b)=sin(a)cos(b)\pm cos(a)*sin(b)$	$cos(a\pm b)=cos(a)cos(b)\mp sin(a)*sin(b)$
$tan(a\pm b)=\frac{tan(a)\pm tan(b)}{1\mp tan(a)tan(b)}$	$tan(a)\pm tan(b)=\frac{sin(a\pm b)}{cos(a)cos(b)}$
$sin(a)\pm sin(b)=2sin(\frac{a\pm b}{2})cos(\frac{a\mp b}{2})$	$cos(a)+cos(b)=2cos(\frac{a+b}{2})cos(\frac{a-b}{2})$
$cos(a)-cos(b)=-2sin(\frac{a+b}{2})sin(\frac{a-b}{2})$	$\int x^ne^{ax}dx=\frac{x^ne^{ax}}{a}-\frac{n}{a}\int x^{n-1}e^{ax}dx$
$\int\frac{1}{1+x^2}dx=tan^{-1}x$	$\int\frac{1}{a^2+x^2}dx=\frac{1}{a}\,tan^{-1}\,\frac{x}{a}$
$\int\frac{x}{a^2+x^2}dx=\frac{1}{2}\,ln\|a^2+x^2\|$	$\int\frac{x^2}{a^2+x^2}dx=x-atan^{-1}\,\frac{x}{a}$
$\int\frac{1}{\sqrt{x^2\pm a^2}}dx=ln\|x+\sqrt{x^2\pm a^2}\|$	$\int\frac{1}{\sqrt{a^2-x^2}}dx=sin^{-1}\,\frac{x}{a}$
$\int\frac{x}{\sqrt{x^2\pm a^2}}dx=\sqrt{x^2\pm a^2}$	$\int\frac{x}{\sqrt{a^2-x^2}}dx=-\sqrt{a^2-x^2}$
$\int sin^2axdx=\frac{x}{2}-\frac{1}{4a}\,sin2ax$	$\int sin^3axdx=-\frac{3cosax}{4a}+\frac{cos3ax}{12a}$
$\int cos^2axdx=\frac{x}{2}+\frac{sin2ax}{4a}$	$\int cos^3axdx=\frac{3sinax}{4a}+\frac{sin3ax}{12a}$
$\int tan(ax)dx=-\frac{1}{a}\,lncosax$	$\int tan^2axdx=-x+\frac{1}{a}\,tan(ax)$
$\int xcos(ax)dx=\frac{1}{a^2}\,cos(ax)+\frac{x}{a}\,sin(ax)$	$\int x^2cos(ax)dx=\frac{2xcos(ax)}{a^2}+\frac{a^2x^2-2}{a^3}\,sin(ax)$
$\int xsin(ax)dx=-\frac{xcos(ax)}{a}+\frac{sin(ax)}{a^2}$	$\int x^2sin(ax)dx=\frac{2-a^2x^2}{a^3}\,cos(ax)+\frac{2xsin(ax)}{a^2}$

1
$sin(na)=ncos^{n-1}asina-\binom{n}{3}cos^{n-3}asin^3a+\binom{n}{5}cos^{n-5}asin^5a-\ldots$
$cos(nb)=cos^na-\binom{n}{2}cos^{n-2}asin^2a+\binom{n}{4}cos^{n-4}asin^4a-\ldots$
$\int\frac{x^2}{\sqrt{x^2\pm a^2}}=\frac{1}{2}\,x\sqrt{x^2\pm a^2}\mp\frac{1}{2}\,a^2ln\lvert x+\sqrt{x^2\pm a^2}\rvert$
$\int\sqrt{x^2\pm a^2}=\frac{1}{2}\,x\sqrt{x^2\pm a^2}\pm\frac{1}{2}\,a^2ln\lvert x+\sqrt{x^2\pm a^2}\rvert$
$\int\sqrt{a^2-x^2}dx=\frac{1}{2}\,x\sqrt{a^2-x^2}+\frac{1}{2}\,a^2tan^{-1}\,\frac{x}{\sqrt{a^2-x^2}}$
$\int\sqrt{ax^2+bx+c}dx=\frac{b+2ax}{4a}\,\sqrt{ax^2+bx+c}+\frac{4ac-b^2}{8a^{3/2}}\,ln\lvert 2ax+b+2\sqrt{a(ax^2+bx+c)}\rvert$

- **组合**
  - 第 i 中有 m_i 个，n种中挑出k个的组合数 $x_i^j=x_{i-1}^j+x_i^{j-1}-x_{i-1}^{j-m_i-1}$
- **重心**：
  - 半径为`r`，圆心角为 `a` 的扇形重心与圆心的距离为 $\frac{4rsin(\frac{a}{2})}{3a}$
  - 半径为`r`，圆心角为 `a` 的圆弧重心与圆心的距离为$\frac{4rsin^3(\frac{a}{2})}{3(a-sin(a))}$
- **Stirling 数**
  - **第一类**：n个元素的项目分作k个环排列的方法数目
    $s_n^k=(-1)^{n+k}|s_n^k|$；$|s_n^0|=0$；$|s_1^1|=1$；$|s_n^k|=|s_{n-1}^{k-1}|+(n-1)*|s_{n-1}^k|$
  - **第二类**：n个元素的集定义k个等价类的方法数
    $S_n^1=S_n^n=1$；$s_n^k=S_{n-1}^{k-1}+k*S_{n-1}^k$

```
int days(int y, int m, int d) {
    if (m < 3)  { y--; m += 12}
    return 365 * y + y / 4 - y / 100 + y / 400 + (153 * m + 2) / 5 + d;
}
int upper(int a,int b) {
  return sgn(a)*sgn(b)<0 ? -lower(a*sgn(a), b*sgn(b)) : a/b + (a % b != 0);  }
int lower(int a,int b) {
  return sgn(a) * sgn(b) < 0 ? -upper(a * sgn(a), b * sgn(b)) : a / b; }
```