Shinnippori

Standard Code Library

November 6, 2017

Contents

2D Geometry

Basic Operations

```
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];
    L() {}
    L(P p1,P p2) { ps[0]=p1; ps[1]=p2; }
  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};
}
```

Delaunay Triangulation

```
const int N = int(1e5) + 10;

int sign(ll x) { return x < 0 ? -1 : x > 0;}

struct P3{
    ll x,y,z;
    P3 operator-(P3 o){ return {x-o.x,y-o.y,z-o.z}; }
    ll operator*(P3 o){ return x*o.x+y*o.y+z*o.z; }
    P3 operator^(P3 o){ return
      ↪ {y*o.z-z*o.y,z*o.x-x*o.z,x*o.y-y*o.x}; }
};

vector<int> edges[N];

ll sqr(ll x) { return x*x; }

struct P {
    ll x, y;
    P operator-(P p) { return {x - p.x, y - p.y}; }
    bool operator<(P o) const {
        return x != o.x ? x < o.x : y < o.y;
    }
    ll det(P p) { return x * p.y - y * p.x; }
    P3 get() { return {x,y,x*x+y*y}; }
    ll distTo2(P o) { return sqr(x-o.x) + sqr(y-o.y); }
};

void addEdge(int u,int v){
    edges[u].pb(v); edges[v].pb(u);
}

#define cross(p1,p2,p3)
  ↪ ((p2.x-p1.x)*(p3.y-p1.y)-(p3.x-p1.x)*(p2.y-p1.y))
inline int crossOp(P p1,P p2,P p3){ return
  ↪ sign(cross(p1,p2,p3)); }

bool crsSS(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;
}

int inCircle(P a, P b, P c, P d) {
    b = b - a; c = c - a; d = d - a;
    if (b.det(c) < 0) swap(b,c);
    P3 pb = b.get(), pc = c.get(), pd = d.get();
    P3 o = pb ^ pc;
    return sign(pd * o);
}

int n;
P ps[N];

inline int crossOp(int i,int j,int k){ return
  ↪ crossOp(ps[i],ps[j],ps[k]); }
inline int dist2(int i,int j){ return ps[i].distTo2(ps[j]); }
inline int inCircle(int a,int b,int c,int d){ return
  ↪ inCircle(ps[a],ps[b],ps[c],ps[d]); }
inline bool crsSS(int a,int b,int c,int d){ return
  ↪ crsSS(ps[a],ps[b],ps[c],ps[d]); }

void construct(int l, int r) {//[l,r)
    if (r - l <= 3) {
        rep(i,l,r) rep(j,l+1,r) addEdge(i,j); return;
    }
    int m = (l + r) / 2; construct(l, m); construct(m, r);

    //find the common tangent
    int pl = l, pr = r - 1;
    for (; ; ) {
        int next = -1;
        for (int p : edges[pl]) {
            int op = crossOp(pr, pl, p);
            if (op > 0 || (op == 0 && dist2(p, pr) < dist2(pl,
              ↪ pr))) {
                next = p;
                break;
            }
        }
        if (next != -1)
            pl = next;
        else {
            next = -1;
            for (int p : edges[pr]) {
                int op = crossOp(pr, pl, p);
                if (op > 0 || (op == 0 && dist2(p, pl) <
                  ↪ dist2(pr, pl))) {
                    next = p;
                    break;
                }
            }
            if (next != -1)
                pr = next;
            else
                break;
        }
    }

    //merge
    addEdge(pl,pr);
    for (; ; ) {
        int next = -1;
        bool which = 0;
        for (int p : edges[pl]) {
            if (crossOp(pr, pl, p) < 0 && (next == -1 ||
              ↪ inCircle(next, pl, pr, p) == -1))
                next = p;
        }
        for (int p : edges[pr]) {
            if (crossOp(pl, pr, p) > 0 && (next == -1 ||
              ↪ inCircle(next, pl, pr, p) == -1)) {
                next = p;
                which = 1;
            }
        }
        if (next == -1)
            break;
        if (!which) {//pl
            vector<int> nEdges;
            for (int p : edges[pl]) {
                if (!crsSS(next, pr, pl, p))
                    nEdges.pb(p);
            }
            edges[pl] = nEdges;
            addEdge(pr,next);
            pl = next;
        } else {//pr
            vector<int> nEdges;
            for (int p : edges[pr]) {
                if (!crsSS(next, pl, pr, p))
                    nEdges.pb(p);
            }
            edges[pr] = nEdges;
            addEdge(pl,next);
            pr = next;
        }
    }
}
```

Minimum Circle Cover

```
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};
}
```

Circle Union

```
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 calc_X(db r,db x,db t){
  return 1.0/2*r*(r*t*x+(3*r*r/4+x*x+r*x*cosl(t))*sinl(t) +
    ↪ 1.0/12 * r*r*sinl(3*t));
}

db calc_Y(db r,db y,db t){
  return 1.0/24 * r * (-3*(3*r*r+4*y*y)*cosl(t) + r *
    ↪ (r*cosl(3*t) - 6 * y * (-2*t + sinl(2*t))));
}

P calc_wc_circle(P c,db r,db L,db R){
  return {calc_X(r,c.x,R) - calc_X(r,c.x,L), calc_Y(r,c.y,R) -
    ↪ calc_Y(r,c.y,L)};
}

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;
  P wc = 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);
        wc = wc +
          ↪ calc_wc_circle(cs[i],rs[i],ev[j].fi,ev[j+1].fi);
      }
    }
  }
}
```

Fast Convex Hull Operations

```
struct CH{
  int n;

  vector<P> ps, lower, upper;

  P operator[](int i){return ps[i];}

  int find(vector<P>&vec, P dir){
    int l=0,r=vec.size();
    while(l+5<r){
      int L = (l*2+r)/3, R = (l+r*2)/3;
      if(vec[L].dot(dir) > vec[R].dot(dir))
        r=R;
      else
        l=L;
    }
    int ret = l; rep(k,l+1,r) if(vec[k].dot(dir) >
      ↪ vec[ret].dot(dir)) ret = k;
    return ret;
  }

  /*
  ps[0] must be the smallest one!
  */

  void init(vector<P> _ps){
    ps = _ps; n = ps.size();

    rotate(ps.begin(),min_element(ps.begin(),
      ↪ ps.end()),ps.end());

    int at = max_element(ps.begin(), ps.end()) - ps.begin();

    lower = vector<P>(ps.begin(),ps.begin() + at + 1);

    upper = vector<P>(ps.begin()+at,ps.end());
      ↪ upper.pb(ps[0]);
  }

  int findFarest(P dir){
    if(dir.y > 0 || dir.y==0 && dir.x > 0){
      return ( (int)lower.size() -1 + find(upper,dir)) % n;
    } else {
      return find(lower,dir);
    }
  }

  P get(int l,int r,P p1,P p2){
    int sl = crossOp(p1,p2,ps[l%n]);
    while(l+1<r){
      int m = (l+r)>>1;
      if(crossOp(p1,p2,ps[m%n]) == sl)
        l = m;
      else
        r = m;
    }

    return isLL(p1,p2,ps[l%n],ps[(l+1)%n]);
  }

  vector<P> getIS(P p1,P p2){
    int X = findFarest((p2-p1).rot90());
    int Y = findFarest((p1-p2).rot90());
    if(X > Y) swap(X,Y);
    if(crossOp(p1,p2,ps[X]) * crossOp(p1,p2,ps[Y]) < 0){
      return {get(X,Y,p1,p2),get(Y,X+n,p1,p2)};
    } else {
      return {};
    }
  }

  void update_tangent(P p, int id, int&a,int&b){
    if(crossOp(p,ps[a],ps[id]) > 0) a = id;
    if(crossOp(p,ps[b],ps[id]) < 0) b = id;
  }

  void binary_search(int l,int r,P p,int&a,int&b){
    if(l==r) return;
    update_tangent(p,l%n,a,b);
    int sl = crossOp(p,ps[l%n],ps[(l+1)%n]);
    while(l+1<r){
      int m = l+r>>1;
      if(crossOp(p,ps[m%n],ps[(m+1)%n]) == sl)
        l=m;
      else
        r=m;
    }
    update_tangent(p,r%n,a,b);
  }
```

```
  bool contain(P p){
    if(p.x < lower[0].x || p.x > lower.back().x) return 0;
    int id = lower_bound(lower.begin(),
      ↪ lower.end(),(P){p.x,-INF}) - lower.begin();
    if(lower[id].x == p.x){
      if(lower[id].y > p.y) return 0;
    } else {
      if(crossOp(lower[id-1],lower[id],p) < 0) return 0;
    }
    id = lower_bound(upper.begin(),
      ↪ upper.end(),(P){p.x,INF},greater<P>()) -
      ↪ upper.begin();
    if(upper[id].x == p.x){
      if(upper[id].y < p.y) return 0;
    } else {
      if(crossOp(upper[id-1],upper[id],p) < 0) return 0;
    }
    return 1;
  }

  bool get_tangent(P p,int&a,int&b){ // b->a
    if(contain(p)) return 0;
    a=b=0;
    int id = lower_bound(lower.begin(), lower.end(),p) -
      ↪ lower.begin();
    binary_search(0,id,p,a,b);
    binary_search(id,lower.size(),p,a,b);
    id = lower_bound(upper.begin(),
      ↪ upper.end(),p,greater<P>()) - upper.begin();
    binary_search((int)lower.size() - 1, (int) lower.size() -
      ↪ 1 + id,p,a,b);
    binary_search((int) lower.size() - 1 + id,(int)
      ↪ lower.size() - 1 + upper.size(),p,a,b);
    return 1;
  }
};

int main(){
  return 0;
}
```

Dynamic Convex Hull With Queries

```
const double eps=1e-9,inf=1e30;
const ll Inf=10000100000000000011;
struct point {
  int x;ll y;double k;
  point(){}
  point(int x):x(x),y(0),k(0){}
  point(int x,ll y):x(x),y(y),k(0){}
  point(int x,ll y,double k):x(x),y(y),k(k){}
};
typedef set<point>::iterator ite;
bool operator <(const point &a,const point &b) {
  if (cw==0) return a.x<b.x; else return a.k<b.k;
}
inline double getk(ite a,ite b) {return
  ↪ 1.*(b->y-a->y)/(b->x-a->x);}
inline double getk(ite a,point b) {return
  ↪ 1.*(b.y-a->y)/(b.x-a->x);}
struct hull {
  set<point> hul;
  void insert(int x,ll y) {
    cw=0;
    point q=point(x,y);
    ite pr,nt,ppr,nnt,Pr;
    if (hul.size()&&x>=hul.begin()->x&&x<=hul.rbegin()->x) {
      pr=hul.lower_bound(point(x));
      if (pr->x==x) {
        if (pr->y>y) {
          ll &p=const_cast<ll&>(pr->y);
          p=y;
        }
      } else {
        --pr;
        if (getk(pr,q)>=pr->k) return;
        else pr=hul.insert(q).first;
      }
    } else pr=hul.insert(q).first;
    Pr=pr;--pr;
    if (Pr!=hul.begin()) while (1) {
      if (pr==hul.begin()) break;
      --(ppr=pr);
      if (getk(ppr,q)<=ppr->k) hul.erase(pr); else break;
      pr=ppr;
    }
    nt=Pr;++nt;
    if (nt!=hul.end()) while (1) {
      ++(nnt=nt);
      if (nnt==hul.end()) break;
      if (getk(nnt,q)>=nt->k) hul.erase(nt); else break;
      nt=nnt;
    }
    nnt=Pr;nt=nnt++;
    double &p=const_cast<double&>(nt->k);
    p=(nnt==hul.end()?inf:getk(nt,nnt));
    ppr=nt;pr=ppr--;
    if (pr!=hul.begin()) {
      double &p=const_cast<double&>(ppr->k);
      p=getk(ppr,pr);
    }
  }
  ll query(int k) {
    if (hul.empty()) return Inf;
    cw=1;
    ite pr=hul.lower_bound(point(0,0,-k));
    return 1ll*k*pr->x+pr->y;
  }
};
```

3D Geometry

Basic Operations

```
db sqr(db x){ return x*x; }

struct P3{
    db x,y,z;
    P3 operator+(P3 o){ return {x+o.x,y+o.y,z+o.z}; }
    P3 operator-(P3 o){ return {x-o.x,y-o.y,z-o.z}; }
    db operator*(P3 o){ return x*o.x+y*o.y+z*o.z; }
    P3 operator^(P3 o){ return
      ↪ {y*o.z-z*o.y,z*o.x-x*o.z,x*o.y-y*o.x}; }
    P3 operator*(db o){ return {x*o,y*o,z*o}; }
    P3 operator/(db o){ return {x/o,y/o,z/o}; }

    db abs2(){ return sqr(x) + sqr(y) + sqr(z); }
    db abs(){ return sqrt(abs2()); }

    P3 norm(){ return *this / abs(); }
    bool operator<(P3 o){
        if(cmp(x,o.x) != 0) return x < o.x;
        if(cmp(y,o.y) != 0) return y < o.y;
        return cmp(z,o.z) == -1;
    }
    bool operator==(P3 o){
        return cmp(x,o.x) == 0 && cmp(y,o.y) == 0 &&
          ↪ cmp(z,o.z) == 0;
    }
    void read(){
        cin>>x>>y>>z;
    }
    void print(){
        //printf("%lf,%lf,%lf\n",x,y,z);
    }
};

typedef vector<P3> VP;
typedef vector<VP> VVP;

db r;

db Acos(db x) {
    return acos(max(-(db)1,min(x,(db)1)));
}

db dist(P3 a,P3 b){// qiumian juli
    db r=Acos(a*b);
    return r;
}

vector<db> solve(db a,db b,db c) {
    // return cos(t)*a+sin(t)*b <= c
    // a=r*cos(th) b=r*sin(th)
    db r=sqrt(a*a+b*b);
    db th=atan2(b,a);
    // r*cos(t-th) <= c
    if (cmp(c,-r)==-1)  return {0}; // c < -r
    else if (cmp(r,c) <= 0) return {1}; // r <= c
    else {
        db tr=pi-Acos(c/r);
        assert(tr < pi);
        return {th+pi-tr,th+pi+tr};
    }
}

P3 rnd;

vector<db> jiao(P3 a,P3 b){
    if (cmp(dist(a,b),2*r)>0) return {0};
    P3 rd=a*cos(r); P3 z=a.norm(); P3 y=(z^rnd).norm(); P3
      ↪ x=(y^z).norm();
```

```
    // (rd+x*cos(t)+y*sin(t))*b >= cos(r)
    vector<db> ret =
      ↪ solve(-(x*b*sin(r)),-(y*b*sin(r)),-(cos(r)-rd*b));
    //return
      ↪ solve(-(x*b*sin(r)),-(y*b*sin(r)),-(cos(r)-rd*b));
    return ret;
}

db norm(db x){ //[0,2pi)
    while(x < 0) x+= 2*pi;
    while(x >= 2*pi) x-= 2*pi;
    return x;
}

db disLP(P3 p1,P3 p2,P3 q){
    return ((p2-p1)^(q-p1)).abs() / (p2-p1).abs();
}

db disLL(P3 p1,P3 p2,P3 q1,P3 q2){
    P3 o = (p2-p1) ^ (q2-q1); if(o.abs() <= EPS) return
      ↪ disLP(p1,p2,q1);
    return fabs(o.norm() * (p1-p2));
}

VP isFL(P3 p,P3 o,P3 q1,P3 q2){
    db a = (q2-p)*o, b = (q1-p)*o;
    db d = a - b;
    if(fabs(d) < EPS) return {};
    return {(q1*a-q2*b)/d};
}

VP isFF(P3 p1,P3 o1,P3 p2,P3 o2){
    P3 e = o1 ^ o2, v = o1 ^ e;
    db d = o2 * v; if(fabs(d) < EPS) return {};
    P3 q = p1 + v * (o2 * (p2-p1) / d);
    return {q,q+e};
}

int main(){
    return 0;
}
```

Convex Hull in 3D

```
db Volume(P3 a,P3 b,P3 c,P3 d){
    return ((b-a)^(c-a))*(d-a);
}

db rand_db(){
    return 1.0 * rand() / RAND_MAX;
}

typedef vector<P3> VP;
typedef vector<VP> VVP;

namespace CH3{
    VVP ret;
    set<pair<int,int> > eg;
    int n;
    VP p,q;

    void wrap(int a,int b){
        if (eg.find({a,b})==eg.end()){
            int c=-1;
            for (int i=0;i<n;i++)if (i!=a && i!=b){
                if (c==-1 || Volume(q[c],q[a],q[b],q[i])>0)
                    c=i;
            }
            if (c!=-1){
                ret.pb({p[a],p[b],p[c]});

                  ↪ eg.insert({a,b});eg.insert({b,c});eg.insert({c,a});
                wrap(c,b);wrap(a,c);
            }
        }
    };

    VVP convexHull3d(VP _p){
        p = q = _p; n = p.size();
        ret.clear(); eg.clear();
        for(auto&i:q) i = i +
          ↪ (P3){rand_db()*1e-4,rand_db()*1e-4,rand_db()*1e-4};
        for (int i=1;i<n;i++)if
          ↪ (q[i].x<q[0].x)swap(p[0],p[i]),swap(q[0],q[i]);
        for (int i=2;i<n;i++)if (
            (q[i].x-q[0].x)*(q[1].y-q[0].y)>
            (q[i].y-q[0].y)*(q[1].x-q[0].x))
              ↪ swap(q[1],q[i]),swap(p[1],p[i]);
        wrap(0,1);
        return ret;
    }
}

VVP unite_tri(VVP pss){
    VVP ret;
    map<P3,VP> by_norm;
    for(auto ps:pss){
        P3 o = ((ps[1] - ps[0]) ^ (ps[2] - ps[0])).norm();
        for(auto i : ps) by_norm[o].pb(i);
    }
    for(auto it:by_norm) ret.pb(convexHull2D(it.se,it.fi));
    return ret;
}

pair<P3,P3> get_face(VP ps){
    return mp(ps[0],((ps[1]-ps[0])^(ps[2]-ps[0])).norm());
}

int main(){
    return 0;
}
```

Convex Cut in 3D

```
VP convexHull2D(VP ps,P3 o){
    P3 x = {rand(),rand(),rand()}; x = x.norm();

    x = (x ^ o).norm(); P3 y = (x ^ o).norm();

    P3 vec = o.norm() * (ps[0] * o);

    vector<P> qs; for(auto p:ps) qs.pb({p*x,p*y}); qs =
      ↪ convexHull(qs);

    ps = {}; for(auto p : qs) ps.pb(x*p.x + y*p.y + vec);
    return ps;
}

VVP convexCut(VVP pss, P3 p, P3 o){ // keep o*(x-p) >= 0
    VVP ret; VP sec;
    for(auto ps : pss){
        int n = ps.size();
        VP qs; bool dif = 0;
        rep(i,0,n){
            int d1 = sign(o*(ps[i]-p));
            int d2 = sign(o*(ps[(i+1)%n]-p));
            if(d1 >= 0) qs.pb(ps[i]);
            if(d1 * d2 < 0){
                P3 q = isFL(p,o,ps[i],ps[(i+1)%n])[0];
                qs.pb(q);
                sec.pb(q);
            }
            if(d1 == 0) sec.pb(ps[i]);
            else dif = 1;
            dif |= o * ((ps[(i+1)%n] - ps[i]) ^
              ↪ (ps[(i+2)%n]-ps[i])) < -EPS;
        }
        if(qs.size() > 0 && dif) ret.pb(qs);
    }
    if(sec.size() > 0) ret.pb(convexHull2D(sec,o));
    return ret;
}

db vol(VVP pss){
    P3 p = pss[0][0];
    db V = 0;
    for(auto ps : pss){
        rep(i,2,ps.size())
            V += fabs(Volume(p,ps[0],ps[i-1],ps[i]));
    }
    return V/6;
}

VVP init(db INF) {
    VVP pss(6,VP(4));
    pss[0][0] = pss[1][0] = pss[2][0] = {-INF, -INF, -INF};
    pss[0][3] = pss[1][1] = pss[5][2] = {-INF, -INF, INF};
    pss[0][1] = pss[2][3] = pss[4][2] = {-INF, INF, -INF};
    pss[0][2] = pss[5][3] = pss[4][1] = {-INF, INF, INF};
    pss[1][3] = pss[2][1] = pss[3][2] = {INF, -INF, -INF};
    pss[1][2] = pss[5][1] = pss[3][3] = {INF, -INF, INF};
    pss[2][2] = pss[4][3] = pss[3][1] = {INF, INF, -INF};
    pss[5][0] = pss[4][0] = pss[3][0] = {INF, INF, INF};
    return pss;
}
```

Circumscribed Sphere of 4 points

```
//  ,
int nouter;
P3 outer[4], res;
db radius;
void ball() {
  P3 q[3];
  db m[3][3], sol[3], L[3], det;
  int i, j;
  res.x = res.y = res.z = radius = 0;
  for (i = 0; i < 3; ++i)
    q[i] = outer[i + 1] - outer[0], sol[i] = q[i] * q[i];
  for (i = 0; i < 3; ++i)
    for (j = 0; j < 3; ++j)
      m[i][j] = (q[i] * q[j]) * 2;

  det = m[0][0] * m[1][1] * m[2][2] + m[0][1] * m[1][2] *
    ↪ m[2][0] + m[0][2]
      * m[2][1] * m[1][0] - m[0][2] * m[1][1] * m[2][0] -
        ↪ m[0][1]
      * m[1][0] * m[2][2] - m[0][0] * m[1][2] * m[2][1];

  if (fabs(det) < EPS)
    return;

  for (j = 0; j < 3; ++j) {
    for (i = 0; i < 3; ++i)
      m[i][j] = sol[i];
    L[j] = (m[0][0] * m[1][1] * m[2][2] + m[0][1] * m[1][2] *
      ↪ m[2][0]
        + m[0][2] * m[2][1] * m[1][0] - m[0][2] * m[1][1] *
          ↪ m[2][0]
        - m[0][1] * m[1][0] * m[2][2] - m[0][0] * m[1][2] *
          ↪ m[2][1])
        / det;
    for (i = 0; i < 3; ++i)
      m[i][j] = (q[i] * q[j]) * 2;
  }
  res = outer[0];
  for (i = 0; i < 3; ++i)
    res = res + q[i] * L[i];

  radius = (res - outer[0]).abs();
}
```

Data Structure

Cartesian Tree

```
int stk[N],top,a[N],l[N],r[N];
void build() {
  int top=0;
  rep(i,1,n+1) {
    int k=top;
    while (k>0&&a[stk[k-1]]>a[i]) --k;
    if (k) r[stk[k-1]]=i;
    if (k<top) l[i]=stk[k];
    stk[k++]=i;
    top=k;
  }
  rep(i,1,n+1) {
    if (l[i]) add(i,l[i]);
    if (r[i]) add(i,r[i]);
  }
}
```

dsu on a tree

```
//nlogn * Time(INSERT)
void insert(data x);
void erase(data x);
void addall(int u,int del){
  if(del==1)insert(val[u]);
  else erase(val[u]);
  for (int i=g[u];~i;i=e[i].next)if(e[i].v!=pre[u])
    addall(e[i].v,del);
}
void dfs(int u,int keep=0){
  int big=-1;
  for (int i=g[u];~i;i=e[i].next)if(e[i].v!=pre[u])
    if(big==-1 || sz[e[i].v]>sz[big])big=e[i].v;

  for (int i=g[u];~i;i=e[i].next)if(e[i].v!=pre[u] &&
    ↪ e[i].v!=big){
    dfs(e[i].v,0);
  }
  if(big!=-1)dfs(big,1);
  insert(val[u]);
  for (int i=g[u];~i;i=e[i].next)if(e[i].v!=pre[u] &&
    ↪ e[i].v!=big)
    addall(e[i].v,1);
  solve_query(u);//now DS has all u's subtree
  if(!keep)addall(u,-1);//or: clear the whole ds
}
```

Link Cut Tree

```
struct node {
  node *s[2],*f,*minv;
  int val,d,id;
  bool rev;
  bool isr() { return !f||(f->s[0]!=this && f->s[1]!=this);}
  bool dir() { return f->s[1]==this;}
  void setc(node *c,int d) { s[d]=c;if (c) c->f=this;}
  void push() {
    if (rev) { swap(s[0],s[1]); rep(i,0,2) if (s[i])
      ↪ s[i]->rev^=1;} rev=0;
  }
  void upd() {
    minv=this; val=d;
    rep(i,0,2) if (s[i]&&s[i]->val>val)
      ↪ val=s[i]->val,minv=s[i]->minv;

  }
}pool[N],*cur;
stack<node*> sta;
void rot(node *x) {
  node *p=x->f;bool d=x->dir();
  if (!p->isr()) p->f->setc(x,p->dir()); else x->f=p->f;
  p->setc(x->s[!d],d);x->setc(p,!d);
  p->upd();
}
void splay(node *x) {
  node *q=x;
  while (1) { sta.push(q);if (q->isr()) break; q=q->f; }
  while (!sta.empty()) sta.top()->push(),sta.pop();
  while (!x->isr()) {
    if (x->f->isr()) rot(x);
    else if (x->isr()==x->f->isr()) rot(x->f),rot(x);
    else rot(x),rot(x);
  }
  x->upd();
}
node *expose(node *x) {
  node *q=NULL;
  for (;x;x=x->f) splay(x),x->s[1]=q,(q=x)->upd();
  return q;
}
void evert(node *x) { expose(x); splay(x); x->rev^=1;
  ↪ x->push();}
void expose(node *x,node *y) { evert(x); expose(y); splay(x);}
void link(node *x,node *y) { evert(x); evert(y);
  ↪ x->setc(y,1);}
void cut(node *x,node *y) { expose(x,y); x->s[1]=y->f=NULL;}
```

Treap

```
const int N=101000;
struct node {
  int wt;
  node *s[2];
  void push() {
  }
  void upd() {
  }
}pool[N],*cur=pool,*rt;
node *newnode(int w) {
  node *q=cur++;
  q->wt=(rand()<<15)+rand();
  return q;
}
#define SIZE(a) ((a)?a->sz:0)
void merge(node *&p,node *l,node *r) {
  if (!l||!r) p=l?l:r;
  else if (l->wt<r->wt) {
    l->push();
    merge(l->s[1],l->s[1],r);
    (p=l)->upd();
  } else {
    r->push();
    merge(r->s[0],l,r->s[0]);
    (p=r)->upd();
  }
}
void split(node *p,node *&l,node *&r,int x) {
  if (x==0) l=0,r=p;
  else if (x==SIZE(p)) l=p,r=0;
```

```
  else {
    p->push();
    if (SIZE(p->s[0])>=x)
      ↪ r=p,split(p->s[0],l,r->s[0],x),r->upd();
    else
      ↪ l=p,split(p->s[1],l->s[1],r,x-SIZE(p->s[0])-1),l->upd();
  }
}
```

K-d Tree

```
struct node {
  node *l,*r;
  ll mx[2],mn[2],d[2];
  int mv,fg,v,id;
  node *mp;
}pool[N],*cur;

bool operator < (const node &a,const node &b) {
  return a.d[D]<b.d[D];
}

void assign(node *p,ll x,ll y,int id,int v) {
  p->l=p->r=0;
  p->d[0]=x; p->d[1]=y;
  rep(i,0,2) p->mn[i]=p->mx[i]=p->d[i];
  p->mv=p->v=v; p->id=id; p->fg=0;
  p->mp=p;
}
struct kdtree{
  node nd[N],*rt;
  ll px,py,qx,qy;
  void updb(node *p){
    node *l=p->l,*r=p->r;
    rep(i,0,2) {
      p->mn[i]=p->mx[i]=p->d[i];
      if(l) p->mn[i]=min(p->mn[i],l->mn[i]),
        p->mx[i]=max(p->mx[i],l->mx[i]);
      if(r) p->mn[i]=min(p->mn[i],r->mn[i]),
        p->mx[i]=max(p->mx[i],r->mx[i]);
    }
  }
  void upd(node *p) {
    p->mv=p->v; p->mp=p;
    if (p->l&&p->l->mv<p->mv) p->mv=p->l->mv,p->mp=p->l->mp;
    if (p->r&&p->r->mv<p->mv) p->mv=p->r->mv,p->mp=p->r->mp;
  }
  node* build(node *p,int l,int r,int D){
    if (l>r) return 0;
    int md=(l+r)>>1;
    ::D=D; nth_element(p+l,p+md,p+r+1);
    node *q=nd+md;
    *q=p[md];
    q->l=build(p,l,md-1,D^1);
    q->r=build(p,md+1,r,D^1);
    updb(q);
    upd(q);
    return q;
  }
  void build(node *p,int n) {
    rt=build(p,0,n-1,0);
  }
  void setf(node *p,int v) {
    p->mv+=v,p->v+=v,p->fg+=v;
  }
  void push(node *p) {
    if (p->fg) {
      if (p->l) setf(p->l,p->fg);
      if (p->r) setf(p->r,p->fg);
      p->fg=0;
    }
  }
  void setf(node *p,int v,int d) {
    if (d==0) p->mv+=v,p->v+=v,p->fg+=v;
    else if (d==1) push(p),p->v=v,upd(p);
    else push(p),p->v+=v,upd(p);
  }
  void modify(node *p,int v,int d=0) {
    if (!p) return;
    if (p->mx[0]<px||p->mn[0]>qx||p->mx[1]<py||p->mn[1]>qy)
      ↪ return;
    if
      ↪ (p->mn[0]>=px&&p->mx[0]<=qx&&p->mn[1]>=py&&p->mx[1]<=qy)
      ↪ {
      setf(p,v,d);
      return;
    }
    if (p->d[0]>=px&&p->d[0]<=qx&&p->d[1]>=py&&p->d[1]<=qy)
      ↪ setf(p,v,d==0?2:1);
    push(p);
    modify(p->l,v,d);
    modify(p->r,v,d);
    upd(p);
  }
  void modify(ll x1,ll y1,ll x2,ll y2,int v,int d=0) {
    px=x1; py=y1; qx=x2; qy=y2;
    modify(rt,v,d);
  }
}T;
```

Leftist Heap

```
struct node{
       int l,r;
       int dis;
}t[N];
int merge(int a,int b){
     if(b==0)return a;
     if(a==0)return b;
     if(key[a]<key[b])swap(a,b);
     t[a].r=merge(t[a].r,b);
     if(t[t[a].r].dis>t[t[a].l].dis)swap(t[a].l,t[a].r);
     t[a].dis=t[t[a].r].dis+1;
     return a;
}
```

zkw Tree

```
class Seg {
  int data[N << 1];
 public:
  void clear() {
    memset(data, 0x3f, sizeof data);
  }
  void modify(int p, int v) {
    data[p += N] = v;
    while (p >>= 1) data[p] = std::min(data[p << 1], data[p <<
      ↪ 1 | 1]);
  }
  int query(int l, int r) {
    int res = INF;
    for (l = l + N - 1, r = r + N + 1; l ^ r ^ 1; l >>= 1, r
      ↪ >>= 1) {
      if (!(l & 1)) res = std::min(res, data[l + 1]);
      if (r & 1) res = std::min(res, data[r - 1]);
    }
    return res;
  }
} seg;
```

Segment Tree Beats

```
#include <cctype>
#include <cstdio>
#include <iostream>
#include <algorithm>

typedef long long i64;

const int N = 100000 + 10, INF = 2000000000 + 10;

inline int nextInt() {
  char ch;
  while (ch = getchar(), ch != '-' && !isdigit(ch)) {}
  bool sig = false;
  if (ch == '-') sig = true, ch = getchar();
  int res = ch - '0';
  while (isdigit(ch = getchar())) res = 10 * res + ch - '0';
  return sig ? -res : res;
}

int n;

struct Info {
  int min, cnt, se;
  int size;
  i64 sum;
  int add, madd;
  Info() {
    min = se = INF;
    cnt = size = sum = add = madd = 0;
  }
  explicit Info(int x) {
    size = cnt = 1;
    sum = min = x;
    se = INF;
    add = madd = 0;
```

```
  }
  inline Info& operator+= (int rhs) {
    if (!rhs) return *this;
    if (min < INF) min += rhs;
    if (se < INF) se += rhs;
    sum += (i64)rhs * size;
    add += rhs;
    return *this;
  }
  inline Info& operator^= (int rhs) {
    if (!rhs) return *this;
    if (min < INF) min += rhs;
    sum += (i64)rhs * cnt;
    madd += rhs;
    return *this;
  }
  friend inline Info operator+ (const Info &lhs, const Info
    ↪ &rhs) {
    Info res;
    res.min = std::min(lhs.min, rhs.min);
    if (lhs.min == res.min) res.cnt += lhs.cnt; else res.se =
      ↪ std::min(res.se, lhs.min);
    if (rhs.min == res.min) res.cnt += rhs.cnt; else res.se =
      ↪ std::min(res.se, rhs.min);
    res.se = std::min(res.se, std::min(lhs.se, rhs.se));
    res.size = lhs.size + rhs.size;
    res.sum = lhs.sum + rhs.sum;
    return res;
  }
  inline Info& operator+= (const Info &rhs) { return *this =
    ↪ *this + rhs; }
};

struct Node {
  Info a, c[2];
} tree[2 * N];

inline Node operator+ (const Node &lhs, const Node &rhs) {
  Node res;
  res.a = lhs.a + rhs.a;
  res.c[1] = lhs.c[1] + rhs.c[1];
  if (lhs.a.min == res.a.min) res.c[0] += lhs.c[0]; else
    ↪ res.c[1] += lhs.c[0];
  if (rhs.a.min == res.a.min) res.c[0] += rhs.c[0]; else
    ↪ res.c[1] += rhs.c[0];
  return res;
}

inline int pos(int l, int r) { return (l + r) | (l != r); }

void release(Node &id, Node &lch, Node &rch) {
  bool l0 = lch.a.min <= rch.a.min, r0 = rch.a.min <=
    ↪ lch.a.min;
  bool l1 = lch.c[0].min <= rch.c[0].min, r1 = rch.c[0].min <=
    ↪ lch.c[0].min;
  if (id.a.add) {
    lch.a += id.a.add;
    rch.a += id.a.add;
    id.a.add = 0;
  }
  if (id.a.madd) {
    if (l0) lch.a ^= id.a.madd;
    if (r0) rch.a ^= id.a.madd;
    id.a.madd = 0;
  }
  if (id.c[0].add || id.c[1].add) {
    lch.c[1] += id.c[1].add;
    rch.c[1] += id.c[1].add;
    if (l0) lch.c[0] += id.c[0].add; else lch.c[0] +=
      ↪ id.c[1].add;
    if (r0) rch.c[0] += id.c[0].add; else rch.c[0] +=
      ↪ id.c[1].add;
    id.c[0].add = id.c[1].add = 0;
  }
  if (id.c[0].madd) {
    if (l0 && !r0) {
      lch.c[0] ^= id.c[0].madd;
    } else if (!l0 && r0) {
      rch.c[0] ^= id.c[0].madd;
    } else {
      if (l1) lch.c[0] ^= id.c[0].madd;
      if (r1) rch.c[0] ^= id.c[0].madd;
    }
    id.c[0].madd = 0;
  }
  if (id.c[1].madd) {
    if (l0 && !r0) {
      int t = std::min(std::min(lch.c[1].min, rch.c[0].min),
        ↪ rch.c[1].min);
      if (lch.c[1].min == t) lch.c[1] ^= id.c[1].madd;
      if (rch.c[0].min == t) rch.c[0] ^= id.c[1].madd;
      if (rch.c[1].min == t) rch.c[1] ^= id.c[1].madd;
    } else if (!l0 && r0) {
      int t = std::min(std::min(lch.c[0].min, lch.c[1].min),
        ↪ rch.c[1].min);
      if (lch.c[0].min == t) lch.c[0] ^= id.c[1].madd;
      if (lch.c[1].min == t) lch.c[1] ^= id.c[1].madd;
      if (rch.c[1].min == t) rch.c[1] ^= id.c[1].madd;
    } else {
      int t = std::min(lch.c[1].min, rch.c[1].min);
      if (lch.c[1].min == t) lch.c[1] ^= id.c[1].madd;
      if (rch.c[1].min == t) rch.c[1] ^= id.c[1].madd;
    }
    id.c[1].madd = 0;
  }
}

void build(int l, int r) {
  int id = pos(l, r);
  if (l == r) {
    tree[id].a = Info(nextInt());
    tree[id].c[0] = Info(0);
    return;
  }
  int mid = (l + r) / 2;
  build(l, mid);
  build(mid + 1, r);
  tree[id] = tree[pos(l, mid)] + tree[pos(mid + 1, r)];
}

void aPlus(int l, int r, int p, int q, int v) {
  int id = pos(l, r);
  if (p <= l && r <= q) {
    tree[id].a += v;
    return;
  }
  int mid = (l + r) / 2, lch = pos(l, mid), rch = pos(mid + 1,
    ↪ r);
  release(tree[id], tree[lch], tree[rch]);
  if (p <= mid) aPlus(l, mid, p, q, v);
  if (q > mid) aPlus(mid + 1, r, p, q, v);
  tree[id] = tree[lch] + tree[rch];
}

void aMax(int l, int r, int p, int q, int v) {
  int id = pos(l, r);
  if (p <= l && r <= q) {
    if (v <= tree[id].a.min) return;
    if (v < tree[id].a.se) {
      v -= tree[id].a.min;
      tree[id].a ^= v;
      tree[id].c[0] += v;
      return;
    }
  }
  int mid = (l + r) / 2, lch = pos(l, mid), rch = pos(mid + 1,
    ↪ r);
  release(tree[id], tree[lch], tree[rch]);
  if (p <= mid) aMax(l, mid, p, q, v);
  if (q > mid) aMax(mid + 1, r, p, q, v);
  tree[id] = tree[lch] + tree[rch];
}

void cPlus(int l, int r, int p, int q, int v) {
  int id = pos(l, r);
  if (p <= l && r <= q) {
    tree[id].c[0] += v;
    tree[id].c[1] += v;
    return;
  }
  int mid = (l + r) / 2, lch = pos(l, mid), rch = pos(mid + 1,
    ↪ r);
  release(tree[id], tree[lch], tree[rch]);
  if (p <= mid) cPlus(l, mid, p, q, v);
  if (q > mid) cPlus(mid + 1, r, p, q, v);
  tree[id] = tree[lch] + tree[rch];
}

void cMax(int l, int r, int p, int q, int v) {
  int id = pos(l, r);
  if (p <= l && r <= q) {
    Info c = tree[id].c[0] + tree[id].c[1];
    if (v <= c.min) return;
    if (v < c.se) {
      v -= c.min;
      if (tree[id].c[0].min == c.min) tree[id].c[0] ^= v;
      if (tree[id].c[1].min == c.min) tree[id].c[1] ^= v;
      return;
    }
  }
  int mid = (l + r) / 2, lch = pos(l, mid), rch = pos(mid + 1,
    ↪ r);
  release(tree[id], tree[lch], tree[rch]);
  if (p <= mid) cMax(l, mid, p, q, v);
  if (q > mid) cMax(mid + 1, r, p, q, v);
```

```
  tree[id] = tree[lch] + tree[rch];
}

i64 query(int l, int r, int p, int q) {
  int id = pos(l, r);
  if (p <= l && r <= q) return tree[id].a.sum -
    ↪ tree[id].c[0].sum - tree[id].c[1].sum;
  int mid = (l + r) / 2, lch = pos(l, mid), rch = pos(mid + 1,
    ↪ r);
  release(tree[id], tree[lch], tree[rch]);
  i64 res = 0;
  if (p <= mid) res += query(l, mid, p, q);
  if (q > mid) res += query(mid + 1, r, p, q);
  tree[id] = tree[lch] + tree[rch];
  return res;
}

int main() {
  n = nextInt();
  int m = nextInt();
  build(1, n);
  while (m--) {
    int op = nextInt(), l = nextInt(), r = nextInt();
    // both c and d can be negative
    if (op == 1) { // a[i] += c
      int c = nextInt();
      aPlus(1, n, l, r, c);
      cPlus(1, n, l, r, c);
      cMax(1, n, l, r, 0);
    } else if (op == 2) { // a[i] = max(a[i], d)
      int d = nextInt();
      aMax(1, n, l, r, d);
    } else { // query historical minimum
      std::cout << query(1, n, l, r) << std::endl;
    }
  }
  return 0;
}
```

Graph

2-SAT

```
namespace SAT2 {
  const int N=2200000;
  VI e[N];
  int n,cnt,dfn[N],low[N],st[N],bel[N],top,ind;
  bool ins[N];
  void init(int ct) {
    n=ct;
    cnt=0;top=0;ind=0;
    rep(i,0,n) e[i].clear();
  }
  void add(int u,int v) { e[u].pb(v);}
  void tarjan(int u) {
    dfn[u]=low[u]=++ind;
    ins[u]=1;
    st[++top]=u;
    rep(i,0,SZ(e[u])) {
      int v=e[u][i];
      if (!dfn[v]) tarjan(v),low[u]=min(low[u],low[v]);
      else if (ins[v]) low[u]=min(low[u],low[v]);
    }
    if (dfn[u]==low[u]) {
      ++cnt;
      while (1) {
        bel[st[top]]=cnt;
        ins[st[top]]=0;
        if (st[top--]==u) break;
      }
    }
  }
  void solve() {
    rep(i,0,n) dfn[i]=0;
    rep(i,0,n) if (!dfn[i]) tarjan(i);
    // bel i>=bel i' ->i'
  }
}
```

Blossom Algorithm

```
// vertices 1~n, chd[x]=0 or y (x match y)
int n;
vector<int> g[N];
int chd[N],nex[N],fl[N],fa[N];
int gf(int x){return fa[x]==x?x:fa[x]=gf(fa[x]);}
void un(int x,int y){x=gf(x),y=gf(y);fa[x]=y;}
int qu[N],p,q;
int lca(int u,int v){
  static int t=0,x[N];
  t++;
  for(;; swap(u,v) )
    if(u){
      u=gf(u);
      if(x[u]==t)return u;
      x[u]=t;
      u= chd[u] ? nex[chd[u]] : 0;
    }
}
void lk(int a,int x){
  while(a!=x){
    int b=chd[a],c=nex[b];
    if(gf(c)!=x)nex[c]=b;
    if(fl[b]==2)fl[qu[q++]=b]=1;
    if(fl[c]==2)fl[qu[q++]=c]=1;
    un(a,b);un(b,c);
    a=c;
  }
}
void find(int rt){
  rep(i,1,n+1)nex[i]=fl[i]=0,fa[i]=i;
  p=q=0;qu[q++]=rt;fl[rt]=1;
  while(p!=q){
    int u=qu[p++];
    rep(j,0,g[u].size()){
      int v=g[u][j];
      if(gf(v)==gf(u) || fl[v]==2 || v==chd[u])continue;
      if(fl[v]==1){
        int x=lca(u,v);
        if(gf(u)!=x)nex[u]=v;
        if(gf(v)!=x)nex[v]=u;
        lk(u,x);
        lk(v,x);
      }else if(!chd[v]){
        nex[v]=u;
        while(v){
          u=nex[v];
          int t=chd[u];
          chd[v]=u;chd[u]=v;
          v=t;
        }
        return;
      }else{
        nex[v]=u;
        fl[v]=2;
        fl[qu[q++]=chd[v]]=1;
      }
    }
  }
}
void work(){
  memset(chd,0,sizeof(chd));
  rep(i,1,n+1)if(!chd[i])find(i);
}
```

Cactus

```
void docycle(int u,int v) {
  cyc.clear();
  while (1) {
    cyc.pb(v);
    if (v==u) break;
    v=p[v];
  }
  reverse(cyc.begin(),cyc.end());
  rep(l,1,SZ(cyc)) {
    v=cyc[l];
    c[u].pb(mp(v,mp(l,SZ(cyc)-l)));
    sz[u]+=sz[v];
  }
}
void dfs(int u,int f) {
  p[u]=f;
  dfn[u]=low[u]=++cnt;
  sz[u]=1;
  bool ff=0;
  rep(i,0,SZ(e[u])) {
    int v=e[u][i];
    if (v==f&&!ff) { ff=1; continue;}
    if (!dfn[v]) {
      dfs(v,u);
      if (low[v]>dfn[u]) {
        t[u].pb(v);
        sz[u]+=sz[v];
      }
    } else cc[u].pb(v);
    low[u]=min(low[u],low[v]);
  }
  rep(i,0,SZ(cc[u])) {
    int v=cc[u][i];
```

```
    if (dfn[v]>dfn[u]) docycle(u,v);
  }
}
```

Clique

```
#define TWOL(x) (1ll<<(x))
void BronK(int S,ll P,ll X) { // 0, TWOL(n)-1, 0
  if (P==0&&X==0) r=max(r,S);
  if (P==0) return;
  int u=__builtin_ctzll(P|X);
  ll c=P&~G[u];
  while (c) {
    int v=__builtin_ctzll(c);
    BronK(S+1,P&G[v],X&G[v]);
    P^=TWOL(v); X|=TWOL(v); c^=TWOL(v);
  }
}
```

Directed MST

```
#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

const int N = 100000 + 10;
const int M = 100000 + 10;
struct Cost;
vector<Cost*> csts;

struct Cost {
  int c;
  Cost*a, *b; //a-b
  int id;
  int nUsed;

  bool operator<(Cost o) const { return c < o.c; }
  Cost(int c, int id) {
    this->c = c;this->id = id;a = b = 0;nUsed = 0;
    csts.push_back(this);
  }

  Cost(Cost*a, Cost*b) {
    this->a = a;this->b = b;id = -1;c = a->c - b->c;nUsed = 0;
    csts.push_back(this);
  }

  void push() {
    if (id == -1) {
      a->nUsed += nUsed;
      b->nUsed -= nUsed;
    }
  }

  void useIt() {++nUsed;}
};

struct edge {
  int u, v;
  Cost* cost;
  edge() {
  }
  edge(int u, int v, int c, int id) :
      u(u), v(v) {
    cost = new Cost(c, id);
  }
} e[M];

int pre[N], hash1[N], vis[N];
Cost* In[N];

bool better(Cost*a, Cost*b) { //a better than b?
  if (a == 0 || b == 0)
    return b == 0;
  return a->c < b->c;
}
int Directed_MST(int root, int n, int m) {
  int ret = 0;
  while (true) {
    rep(i,0,n) In[i] = 0;
    rep(i,0,m) {
      int u = e[i].u; int v = e[i].v;
      if (better(e[i].cost, In[v]) && u != v) {
        pre[v] = u;
        In[v] = e[i].cost;
      }
    }
    rep(i,0,n) {
      if (i == root) continue;
      if (In[i] == 0) return -1;
    }
    int cntnode = 0;
    memset(hash1, -1, sizeof(int) * n);
    memset(vis, -1, sizeof(int) * n);

    rep(i,0,n) if (i != root) {
      ret += In[i]->c;
      In[i]->useIt();
      int v = i;
      while (vis[v] != i && hash1[v] == -1 && v != root) {
        vis[v] = i;
        v = pre[v];
      }
      if (v != root && hash1[v] == -1) {
        for (int u = pre[v]; u != v; u = pre[u])
          hash1[u] = cntnode;
        hash1[v] = cntnode++;
      }
    }
    if (cntnode == 0)
      break;
    rep(i,0,n)
      if (hash1[i] == -1)
        hash1[i] = cntnode++;
    rep(i,0,m) {
      int v = e[i].v;
      e[i].u = hash1[e[i].u];
      e[i].v = hash1[e[i].v];
      if (e[i].u != e[i].v) {
        e[i].cost = new Cost(e[i].cost, In[v]);
      }
    }
    n = cntnode;
    root = hash1[root];
  }
  return ret;
}
int n, m;
int main() {
  scanf("%d %d", &n, &m);
  int mm = 0;
  rep(i,0,m) {
    int a, b, c;
    scanf("%d%d%d", &a, &b, &c);
    a--, b--;
    e[mm++] = edge(a, b, c, i + 1);
  }
  int ans = Directed_MST(0, n, mm);
  if (ans == -1)
    puts("-1");
  else {
    cout << ans << endl;
    per(i,0,csts.size()) csts[i]->push();
    vector<int> lst;
    rep(i,0,csts.size()) {
      Cost*c = csts[i];
      if (c->id != -1 && c->c > 0 && c->nUsed > 0) {
        lst.push_back(c->id);
      }
    }
    sort(lst.begin(), lst.end());
    rep(i,0,lst.size()) {
      cout << lst[i] << " ";
    }
    cout << endl;
  }
  return 0;
}
```

Dominator Tree

```
//vertices 1~ n
vector<int> g[N],gt[N],dom[N];
int dfn[N],ind,id[N];
int pre[N],idom[N],semi[N];
int f[N],best[N];
int get(int x){
  if(x==f[x])return x;
  int y=get(f[x]);
  if(semi[best[x]]>semi[best[f[x]]])best[x]=best[f[x]];
  return f[x]=y;
}
void dfs(int u){
  id[dfn[u]=++ind]=u;
  rep(j,0,g[u].size()){
    int v=g[u][j];
```

```
    if(!dfn[v]){
      dfs(v);
      pre[dfn[v]]=dfn[u];
    }
  }
}
void tarjan(){
  per(j,2,ind+1){
    int u=id[j];
    rep(i,0,gt[u].size()){
      int v=dfn[gt[u][i]];
      if(!v)continue;
      get(v);
      if(semi[best[v]]<semi[j])semi[j]=semi[best[v]];
    }
    dom[semi[j]].pb(j);
    int x=f[j]=pre[j];
    rep(i,0,dom[x].size()){
      int z=dom[x][i];
      get(z);
      if(semi[best[z]]<x)idom[z]=best[z];
      else idom[z]=x;
    }
    dom[x].clear();
  }
  rep(i,2,ind+1){
    if(semi[i]!=idom[i])idom[i]=idom[idom[i]];
    dom[id[idom[i]]].pb(id[i]);
  }
}
void solve(){
  rep(i,1,n+1)g[i].clear(),gt[i].clear();
  while(m--){
    cin>>u>>v;
    g[u].pb(v);
    gt[v].pb(u);
  }
  ind=0;
  rep(i,1,n+1){
    dom[i].clear();
    dfn[i]=0;
    f[i]=best[i]=semi[i]=i;
  }
  int rt=1;
  dfs(rt);
  tarjan();
}
```

Euler Tour

```
struct edge{
  int v,nex;
}e[E*2];int g[V],etot;
int flag[V];
vector<int> ans;
void ae(int u,int v){
  e[etot].v=v;e[etot].nex=g[u];g[u]=etot++;
}
void dfs(int u){
  for (int i=g[u];~i;i=g[u]){
    while(~i && flag[i>>1])g[u]=i=e[i].nex;
    if(i==-1)break;
    flag[i>>1]=1;
    dfs(e[i].v);
    ans.pb(i);
  }
}
bool solve(){
  memset(g,-1,sizeof(g));
  rep(i,1,m+1){
    scanf("%d%d",&x,&y);
    st=x;
    ae(x,y);
    ae(y,x);
    d[x]++;
    d[y]++;
  }
  rep(i,1,n+1)if(d[i]&1) return 0;
  dfs(st);
  if(ans.size()<m) return 0;
  reverse(ans.begin(),ans.end());
  return 1;
}
```

HLDoT

```
int q[N],hs[N],hv[N],dep[N],id[N],l[N],r[N],bel[N],s[N],f[N];
void dfs(int u,int f) {
  id[l[u]=++tot]=u;
  dep[u]=dep[f]+1;
  if (hv[u]) dfs(hv[u],u);
  rep(j,0,SZ(e[u])) if (e[u][j]!=f&&e[u][j]!=hv[u])
    dfs(e[u][j],u);
  r[u]=tot;
}
void HLDoT(int rt) {
  int t=1;
  q[0]=rt;
  rep(i,0,n) {
    int u=q[i];
    rep(j,0,SZ(e[u])) if (e[u][j]!=f[u])
      f[e[u][j]]=u,dep[q[t++]=e[u][j]]=dep[u]+1;
  }
  per(i,0,n) {
    int u=q[i],p=f[u];
    s[u]++,s[p]+=s[u];
    if (!l[u]) l[u]=1;
    if (hs[p]<s[u]) hs[p]=s[u],hv[p]=u,l[p]=l[u]+1;
  }
  rep(i,0,n) {
    int u=q[i];
    if (!bel[u]) bel[u]=u;
    if (hv[u]) bel[hv[u]]=bel[u];
  }
  dfs(rt,0);
}
```

KM

```
namespace KM { // maximum
  const int N=110;
  int a[N][N],dx[N],dy[N],g[N],f[N],b[N],slack[N],n,mat[N];
  bool hungary(int x) {
    if (!x) return 1;
    f[x]=1;
    rep(i,1,n+1) {
      if (g[i]) continue;
      int t=dx[x]+dy[i]-a[x][i];
      if (!t) {
        g[i]=1;
        if (hungary(b[i])) {
          b[i]=x;
          return 1;
        }
      } else slack[i]=min(slack[i],t);
    }
    return 0;
  }
  void solve() {
    clr(dy);
    rep(i,1,n+1) {
      dx[i]=a[i][1];
      rep(j,1,n+1) dx[i]=max(dx[i],a[i][j]);
    }
    rep(i,1,n+1) {
      memset(slack,0x20,sizeof(slack));
      clr(f);clr(g);
      while (!hungary(i)) {
        int d=inf;
        rep(j,1,n+1) if (!g[j]) d=min(d,slack[j]);
        rep(j,1,n+1) {
          if (f[j]) dx[j]-=d;
          if (g[j]) dy[j]+=d;
        }
        clr(f);clr(g);
      }
    }
    rep(i,1,n+1) mat[b[i]]=i;
  }
  void init(int v) {
    n=v;
    clr(a);clr(b);
  }
}
```

Naive Maxflow

```
int s,t,vtot;
struct edge{int v,ne;ll f;}e[E*2];int g[V],et;
void ae(int u,int v,ll f){
  e[et]={v,g[u],f};g[u]=et++;
  e[et]={u,g[v],0};g[v]=et++;
}
int vis[V],ti;
ll aug(int u,ll m){
  if(u==t)return m;
  vis[u]=ti;
```

```
  for (int i=g[u];~i;i=e[i].ne)if(e[i].f && vis[e[i].v]!=ti){
    ll f=aug(e[i].v,min(m,e[i].f));
    if(f){
      e[i].f-=f; e[i^1].f+=f;
      return f;
    }
  }
  return 0;
}
ll maxflow(){
  ll su=0,d;
  while(ti++,d=aug(s,inf))su+=d;
  return su;
}
void init(){
  rep(i,1,vtot+1)g[i]=-1;
  et=0;
}
```

Dinic

```
int s,t,vtot;
struct edge{int v,ne;ll f;}e[E*2];int g[V],et;
void ae(int u,int v,ll f){
  e[et]={v,g[u],f};g[u]=et++;
  e[et]={u,g[v],0};g[v]=et++;
}
int d[V],cu[V];
bool lb(){
  rep(i,1,vtot+1)d[i]=0,cu[i]=g[i];
  static int qu[V];
  int p=0,q=0;
  qu[q++]=s,d[s]=1;
  while(p!=q){
    int u=qu[p++];
    for (int i=g[u];~i;i=e[i].ne)if(e[i].f && !d[e[i].v]){
      d[e[i].v]=d[u]+1;
      if(e[i].v==t)return 1;
      qu[q++]=e[i].v;
    }
  }return 0;
}
ll aug(int u,ll m){
  if(u==t)return m;
  ll su=0,f;
  for (int i=cu[u];~i;cu[u]=i=e[i].ne)if(e[i].f &&
    ↪ d[e[i].v]==d[u]+1){
    f=aug(e[i].v,min(m,e[i].f));
    e[i].f-=f; e[i^1].f+=f;
    m-=f; su+=f;
    if(!m)break;
  }
  if(!su)d[u]=-1;
  return su;
}
ll dinic(){
  ll su=0;
  while(lb())su+=aug(s,inf);
  return su;
}
void init(){
  rep(i,1,vtot+1)g[i]=-1;
  et=0;
}
```

ISAP

```
class Flow {
  int adj[N], to[E], next[E], cap[E], cnt;
  int h[N], gap[N], s, t;
  int dfs(int a, int df) {
    if (a == t) return df;
    int res = 0;
    for (int i = adj[a]; i; i = next[i]) {
      int b = to[i];
      if (cap[i] && h[a] == h[b] + 1) {
        int f = dfs(b, std::min(df - res, cap[i]));
        cap[i] -= f;
        cap[i ^ 1] += f;
        res += f;
      }
      if (res == df) return res;
    }
    if (--gap[h[a]] == 0) h[s] = t + 1;
    ++gap[++h[a]];
    return res;
  }
 public:
  inline void clear(int _s, int _t) {
    cnt = 2, s = _s, t = _t;
    memset(adj, 0, sizeof adj);
  }
  inline void link(int a, int b, int c) {
    to[cnt] = b, next[cnt] = adj[a], cap[cnt] = c, adj[a] =
      ↪ cnt++;
    to[cnt] = a, next[cnt] = adj[b], cap[cnt] = 0, adj[b] =
      ↪ cnt++;
  }
  int flow() {
    memset(h, 0, sizeof h);
    memset(gap, 0, sizeof gap);
    int res = 0;
    while (h[s] < t + 1) res += dfs(s, INF);
    return res;
  }
};
```

Min Cost Flow SPFA

```
int s,t,vtot;
struct edge{int v,ne;ll f,c;}e[E*2];int g[V],et;
void ae(int u,int v,ll f,ll c){
  e[et]={v,g[u],f,c}; g[u]=et++;
  e[et]={u,g[v],0,-c};g[v]=et++;
}
int to[V];
ll d[V];
bool spfa(){
  static int qu[V],in[V];
  int p=0,q=0;
  qu[q++]=s;in[s]=1;
  rep(i,1,vtot+1) d[i]=infc;
  d[s]=0;
  while(p!=q){
    int u=qu[p++]; if(p==V)p=0;
    in[u]=0;
    for (int i=g[u];~i;i=e[i].ne){
      int v=e[i].v;
      if(e[i].f && d[v]>d[u]+e[i].c){
        d[v]=d[u]+e[i].c;
        to[v]=i;
        if(!in[v]){
          in[v]=1;
          qu[q++]=v; if(q==V)q=0;
        }
      }
    }
  }
  return d[t]<infc;
}
ll mcmf(){
  ll co=0,flo=0;
  while(spfa()){
    // if(d[t]>0)break;
    ll f=inff;
    for (int u=t;u!=s;u=e[to[u]^1].v)
      f=min(f,e[to[u]].f);
    flo+=f;
    co+=f*d[t];
    for (int u=t;u!=s;u=e[to[u]^1].v)
      e[to[u]].f-=f, e[to[u]^1].f+=f;
  }
  return co;
}
void init(){
  rep(i,1,vtot+1)g[i]=-1;
  et=0;
}
```

Primal Dual

```
class MCMF {
  int arc[N], adj[N], to[E], next[E], cap[E], cost[E], cnt;
  int s, t, cur;
  int dist[N];
  std::pair<int, int> heap[E];
  bool dijkstra() {
    std::fill(dist, dist + t + 1, INF);
    int top = 1;
    for (heap[0] = std::make_pair(dist[s] = 0, s); top;) {
      std::pop_heap(heap, heap + top);
      std::pair<int, int> info = heap[--top];
      int a = info.second;
      if (-info.first > dist[a]) continue;
      for (int i = adj[a]; i; i = next[i]) {
        int b = to[i], c = cost[i];
        if (cap[i] && dist[a] + c < dist[b]) {
```

```
          heap[top++] = std::make_pair(-(dist[b] = dist[a] +
            ↪ c), b);
          std::push_heap(heap, heap + top);
        }
      }
    }
    if (dist[t] == INF) return false;
    for (int a = 0; a <= t; ++a)
      for (int i = adj[a]; i; i = next[i])
        cost[i] -= dist[to[i]] - dist[a];
    cur += dist[t];
    return true;
  }
  int tag[N], tot;
  int dfs(int a, int df) {
    if (a == t) return df;
    tag[a] = tot;
    int res = 0;
    for (int &i = arc[a]; i; i = next[i]) {
      int b = to[i];
      if (cap[i] && !cost[i] && tag[b] != tot) {
        int f = dfs(b, std::min(df - res, cap[i]));
        cap[i] -= f;
        cap[i ^ 1] += f;
        res += f;
      }
      if (res == df) break;
    }
    return res;
  }
 public:
  inline void clear(int _s, int _t) {
    cnt = 2, s = _s, t = _t;
    memset(adj, 0, sizeof adj);
  }
  inline void link(int a, int b, int c, int d) {
    to[cnt] = b, next[cnt] = adj[a], cap[cnt] = c, cost[cnt] =
      ↪ d, adj[a] = cnt++;
    to[cnt] = a, next[cnt] = adj[b], cap[cnt] = 0, cost[cnt] =
      ↪ -d, adj[b] = cnt++;
  }
  std::pair<int, int> flow() {
    std::pair<int, int> res(0, 0);
    for (cur = 0; dijkstra();) {
      std::copy(adj, adj + t + 1, arc);
      do {
        ++tot;
        int f = dfs(s, INF);
        if (!f) break;
        res.first += f;
        res.second += cur * f;
      } while (1);
    }
    return res;
  }
};
```

Stoer Wagner

```
// min cut of undirected graph O(n^3)
// vertices 1~n, g[x][y]=g[y][x]=weight
ll g[N][N],deg[N];
bool vis[N],off[N];
int n;
ll work(){
  memset(g,0,sizeof(g));
  while(m--){
    cin>>x>>y>>z;
    g[x][y]=g[y][x]+=z;
  }
  memset(off,0,sizeof(off));
  ll ans=inf;
  per(num,2,n+1){
    memset(vis,0,sizeof(vis));
    memset(deg,0,sizeof(deg));
    int s,t;
    per(i,1,num+1){
      int u=0;
      rep(x,1,n+1)if(!off[x] && !vis[x] && deg[x]>=deg[u])u=x;
      vis[u]=1;
      rep(x,1,n+1)deg[x]+=g[x][u];
      if(i==2)s=u;
      if(i==1)t=u;
    }
    ans=min(ans,deg[t]);
    rep(u,1,n+1)if(u!=s && u!=t)g[u][t]=g[t][u]+=g[u][s];
    off[s]=1;
  }
  return ans;
}
```

BCC

```
struct edge{int v,ne;}e[E*6];int et=0,g[V];
int g2[V*2];int ndtot;
void ae(int u,int v,int *h=g){
  e[et].v=v;e[et].ne=h[u];h[u]=et++;
}
int in[V],dfn[V]={0},low[V],stk[V],tmp[V]={0},top=0;
int ind=0;
void mark(int l,int r){
  int m=++ndtot;
  for (int i=l;i<=r;i++){
    int u=e[stk[i]<<1].v,v=e[stk[i]<<1|1].v;
    if(tmp[u]!=m)ae(m,u,g2),ae(u,m,g2);
    if(tmp[v]!=m)ae(m,v,g2),ae(v,m,g2);
    tmp[u]=tmp[v]=m;
  }
}
void dfs(int u,int pr){
  dfn[u]=low[u]=++ind;
  in[u]=1;
  for (int i=g[u];~i;i=e[i].ne)if(i!=pr){
    if(!dfn[e[i].v]){
      stk[++top]=i>>1;
      dfs(e[i].v,i^1);
      low[u]=min(low[u],low[e[i].v]);
      if(low[e[i].v]>=dfn[u]){
        int to=top;
        while(stk[top]!=i>>1)top--;
        top--;
        mark(top+1,to);
      }
    }else if(in[e[i].v]){
      stk[++top]=i>>1;
      low[u]=min(low[u],dfn[e[i].v]);
    }
  }
  in[u]=0;
}
```

K-th Shortest

```
#define for_each(it, v) for (vector<Edge*>::iterator it =
  ↪ (v).begin(); it != (v).end(); ++it)
const int MAX_N = 10000;
const int MAX_M = 50000;
const int MAX_K = 10000;
const int INF = 1000000000;

struct Edge
{
  int from, to;
  int weight;
};
struct HeapNode
{
  Edge* edge;
  int depth;
  HeapNode* child[4];
  //child[0..1] for heap G
  //child[2..3] for heap out edge
};

int n, m, k, s, t;
Edge* edge[MAX_M];
int dist[MAX_N];
Edge* prev[MAX_N];
vector<Edge*> graph[MAX_N];
vector<Edge*> graphR[MAX_N];
HeapNode* nullNode;
HeapNode* heapTop[MAX_N];

HeapNode* createHeap(HeapNode* curNode, HeapNode* newNode)
{
  if (curNode == nullNode)
    return newNode;
  HeapNode* rootNode = new HeapNode;
  memcpy(rootNode, curNode, sizeof(HeapNode));
  if (newNode->edge->weight < curNode->edge->weight)
  {
    rootNode->edge = newNode->edge;
    rootNode->child[2] = newNode->child[2];
    rootNode->child[3] = newNode->child[3];
    newNode->edge = curNode->edge;
    newNode->child[2] = curNode->child[2];
    newNode->child[3] = curNode->child[3];
  }
  if (rootNode->child[0]->depth < rootNode->child[1]->depth)
```

```
    rootNode->child[0] = createHeap(rootNode->child[0],
      ↪ newNode);
  else
    rootNode->child[1] = createHeap(rootNode->child[1],
      ↪ newNode);
  rootNode->depth = max(rootNode->child[0]->depth,
    ↪ rootNode->child[1]->depth) + 1;
  return rootNode;
}
bool heapNodeMoreThan(HeapNode* node1, HeapNode* node2)
{
  return node1->edge->weight > node2->edge->weight;
}
int main()
{
  scanf("%d%d%d", &n, &m, &k);
  scanf("%d%d", &s, &t);
  s--, t--;
  while (m--)
  {
    Edge* newEdge = new Edge;
    int i, j, w;
    scanf("%d%d%d", &i, &j, &w);
    i--, j--;
    newEdge->from = i;
    newEdge->to = j;
    newEdge->weight = w;
    graph[i].push_back(newEdge);
    graphR[j].push_back(newEdge);
  }

  //Dijkstra
  queue<int> dfsOrder;

  memset(dist, -1, sizeof(dist));
  typedef pair<int, pair<int, Edge*> > DijkstraQueueItem;
  priority_queue<DijkstraQueueItem, vector<DijkstraQueueItem>,
    ↪ greater<DijkstraQueueItem> > dq;
  dq.push(make_pair(0, make_pair(t, (Edge*) NULL)));
  while (!dq.empty())
  {
    int d = dq.top().first;
    int i = dq.top().second.first;
    Edge* edge = dq.top().second.second;
    dq.pop();
    if (dist[i] != -1)
      continue;
    dist[i] = d;
    prev[i] = edge;
    dfsOrder.push(i);
    for_each(it, graphR[i])
      dq.push(make_pair(d + (*it)->weight,
        ↪ make_pair((*it)->from, *it)));
  }

  //Create edge heap
  nullNode = new HeapNode;
  nullNode->depth = 0;
  nullNode->edge = new Edge;
  nullNode->edge->weight = INF;
  fill(nullNode->child, nullNode->child + 4, nullNode);

  while (!dfsOrder.empty())
  {
    int i = dfsOrder.front();
    dfsOrder.pop();

    if (prev[i] == NULL)
      heapTop[i] = nullNode;
    else
      heapTop[i] = heapTop[prev[i]->to];

    vector<HeapNode*> heapNodeList;
    for_each(it, graph[i])
    {
      int j = (*it)->to;
      if (dist[j] == -1)
        continue;
      (*it)->weight += dist[j] - dist[i];
      if (prev[i] != *it)
      {
        HeapNode* curNode = new HeapNode;
        fill(curNode->child, curNode->child + 4, nullNode);
        curNode->depth = 1;
        curNode->edge = *it;
        heapNodeList.push_back(curNode);
      }
    }

    if (!heapNodeList.empty()) //Create heap out
    {
      make_heap(heapNodeList.begin(), heapNodeList.end(),
        ↪ heapNodeMoreThan);
      int size = heapNodeList.size();
      for (int p = 0; p < size; p++)
      {
        heapNodeList[p]->child[2] = 2 * p + 1 < size ?
          ↪ heapNodeList[2 * p + 1] : nullNode;
        heapNodeList[p]->child[3] = 2 * p + 2 < size ?
          ↪ heapNodeList[2 * p + 2] : nullNode;
      }
      heapTop[i] = createHeap(heapTop[i],
        ↪ heapNodeList.front());
    }
  }

  //Walk on DAG
  typedef pair<long long, HeapNode*> DAGQueueItem;
  priority_queue<DAGQueueItem, vector<DAGQueueItem>,
    ↪ greater<DAGQueueItem> > aq;
  if (dist[s] == -1)
    printf("NO\n");
  else
  {
    printf("%d\n", dist[s]);
    if (heapTop[s] != nullNode)
      aq.push(make_pair(dist[s] + heapTop[s]->edge->weight,
        ↪ heapTop[s]));
  }
  k--;
  while (k--)
  {
    if (aq.empty())
    {
      printf("NO\n");
      continue;
    }
    long long d = aq.top().first;
    HeapNode* curNode = aq.top().second;
    aq.pop();
    printf("%I64d\n", d);
    if (heapTop[curNode->edge->to] != nullNode)
      aq.push(make_pair(d +
        ↪ heapTop[curNode->edge->to]->edge->weight,
        ↪ heapTop[curNode->edge->to]));
    for (int i = 0; i < 4; i++)
      if (curNode->child[i] != nullNode)
        aq.push(make_pair(d - curNode->edge->weight +
          ↪ curNode->child[i]->edge->weight,
          ↪ curNode->child[i]));
  }

  return 0;
}
```

Math

$(ax+b)$ div c

```
ll Get(ll a,ll b,ll c,ll n) { // (ax+b)/c sum i=0..n-1
    ll res=0;
    res+=(b/c)*n;b%=c;
    res+=(a/c)*n*(n-1)/2;a%=c;
    if (a*n+b<c) return res;
    else return res+Get(c,(a*n+b)%c,a,(a*n+b)/c);
}
```

FFT EXP

```
const int mo=998244353,g=3;
int qp(int a,int b){
  int ans=1;
  do{if(b&1)ans=1ll*ans*a%mo;a=1ll*a*a%mo;}while(b>>=1);
  return ans;
}
int w[2][N+1];
void dft(int *a,int n,bool v){//0 <= a[i] < mo
  int j=0;
  rep(i,0,n){
    if(i>j)swap(a[i],a[j]);
    for (int l=n>>1;(j^=l)<l;l>>=1);
  }
  for (int i=2;i<=n;i<<=1)
    for (int j=0,s=n/i;j<n;j+=i)
      rep(l,0,i>>1){
        int t=1ll*a[j+l+(i>>1)]*w[v][s*l]%mo;
        a[j+l+(i>>1)]=(a[j+l]+mo-t)%mo;
        a[j+l]=(a[j+l]+t)%mo;
      }
  if(v){
    int y=qp(n,mo-2);
    rep(i,0,n)a[i]=1ll*a[i]*y%mo;
```

```
  }
}
void init(int n){
  int ww=qp(g,(mo-1)/n);
  w[0][0]=1;
  rep(i,1,n+1)w[0][i]=1ll*w[0][i-1]*ww%mo;
  rep(i,0,n+1)w[1][i]=w[0][n-i];
}
void mul(int *a,int *b,int n){
  static int x[N];
  rep(i,0,2*n)x[i]=b[i];
  init(2*n);
  dft(x,2*n,0);
  dft(a,2*n,0);
  rep(i,0,2*n)a[i]=1ll*x[i]*a[i]%mo;
  dft(a,2*n,1);
  rep(i,n,2*n)a[i]=0;
}
void inv(int *a,int n,int *b){
  static int x[N];
  b[0]=qp(a[0],mo-2);b[1]=0;
  for (int m=2;m<=n;m<<=1){
    rep(i,0,m) x[i]=a[i],x[i+m]=b[i+m]=0;
    init(2*m);
    dft(x,2*m,0);
    dft(b,2*m,0);
    rep(i,0,2*m) b[i]=1ll*b[i]*(2-1ll*x[i]*b[i]%mo+mo)%mo;
    dft(b,2*m,1);
    rep(i,m,2*m)b[i]=0;
  }
}
void Ln(int *a,int n){
  static int x[N];
  a[0]=1;inv(a,n,x);
  rep(i,0,n-1)a[i]=1ll*(i+1)*a[i+1]%mo;
  mul(a,x,n);
  per(i,1,n)a[i]=1ll*a[i-1]*qp(i,mo-2)%mo;
  a[0]=0;
}
void Exp(int *a,int n,int *r){
  static int x[N];
  r[0]=1;r[1]=0;
  for (int m=2;m<=n;m<<=1){
    rep(i,0,m)x[i]=r[i];
    Ln(x,m);
    rep(i,0,m)x[i]=(a[i]-x[i]+mo)%mo;
    x[0]=(x[0]+1)%mo;
    rep(i,m,2*m) r[i]=x[i]=0;
    mul(r,x,m);
    rep(i,m,2*m)r[i]=0;
  }
}
```

FFT Multi Point

```
// qs[1..qtot] query point
// f[0..fdeg] f(x)
// suan()
// ans[1..qtot] answer
void convo(int*a,int n,int*b,int m,int*c){
  int k=2; while(k<=n+m)k<<=1;
  static int x[N],y[N];
  rep(i,0,k)x[i]=y[i]=0;
  rep(i,0,n+1)x[i]=(a[i]%mo+mo)%mo;
  rep(i,0,m+1)y[i]=(b[i]%mo+mo)%mo;
  mul(x,y,k);
  rep(i,0,n+m+1)c[i]=x[i];
}
void inv(int*a,int n,int*b,int m){
  static int x[N],g[N],y[N],z[N];
  int k=2; while(k<m)k<<=1;
  n=min(n,m);
  rep(i,0,n)x[i]=a[i];
  rep(i,n,k)x[i]=0;
  g[0]=qp(a[0],mo-2);
  for (int l=1;l<k;l<<=1){
    convo(g,l-1,g,l-1,y);
    convo(y,2*l-2,x,2*l-1,z);
    rep(i,0,2*l)g[i]=((2ll*(i<l?g[i]:0)-z[i])%mo+mo)%mo;
  }
  rep(i,0,m)b[i]=g[i];
}
void dv(int *r,int &rdeg,int *a,int n,int *b,int
  ↪ m){//dega=n,degb=m
  static int p[N],q[N],c[N];
  if(n<m){
    rep(i,0,n+1)r[i]=a[i];
    rdeg=n;
    return;
  } //if(min(n-m,m)<=20){...
  int k=1; while(k<=n)k<<=1;
  rep(i,0,n+1)p[i]=a[n-i];
  rep(i,0,m+1)q[i]=b[m-i];
  inv(q,m+1,c,k);
  convo(c,k-1,p,n,c);
  reverse(c,c+n-m+1);
  convo(c,n-m,b,m,c);
  rep(i,n+1,2*k)c[i]=0;
  rep(i,0,m) r[i]=((a[i]-c[i])%mo+mo)%mo;
  for (rdeg=m-1;rdeg && !r[rdeg];rdeg--);
}
int buf[N*20],*qt[N],qd[N],to=0;
int qs[N],qtot=0;
int ans[N];
int f[N],fdeg;
void bq(int l,int r,int x){
  if(l==r){
    qt[x]=buf+to;
    qd[x]=1;
    qt[x][0]=-qs[l];
    qt[x][1]=1;
    to+=qd[x]+1;
  }else{
    int mid=l+r>>1;
    bq(l,mid,x<<1);
    bq(mid+1,r,x<<1|1);
    qt[x]=buf+to;
    convo(qt[x<<1],qd[x<<1],qt[x<<1|1],qd[x<<1|1],qt[x]);
    qd[x]=qd[x<<1]+qd[x<<1|1];
    to+=qd[x]+1;
  }
}
void work(int l,int r,int x,int *f,int fdeg){
  if(l==r) ans[l]=f[0];
  else{
    int mid=l+r>>1;
    int *s1=buf+to,sdeg;
    dv(s1,sdeg,f,fdeg,qt[x<<1],qd[x<<1]);
    to+=sdeg+1;
    work(l,mid,x<<1,s1,sdeg);

    dv(s1,sdeg,f,fdeg,qt[x<<1|1],qd[x<<1|1]);
    to+=sdeg+1;
    work(mid+1,r,x<<1|1,s1,sdeg);
  }
}
void suan(){
  to=0;
  if(qtot){
    bq(1,qtot,1);
    work(1,qtot,1,f,fdeg);
  }
}
```

FFT MYY double

```
cp operator!()const{return (cp){a,-b};};

void mul(db *a,db *b,int n){// n<=N
  init(n>>1);
  static cp f[N],g[N],t[N];
  rep(i,0,n){
    if(i&1)f[i>>1].b=a[i], g[i>>1].b=b[i];
    else f[i>>1].a=a[i], g[i>>1].a=b[i];
  }
  dft(f,n>>1,0);dft(g,n>>1,0);

  rep(i,0,n>>1){
    int j=i?(n>>1)-i:0;
    t[i]=( (cp){4,0}*!(f[j]*g[j]) -
      ↪ (!f[j]-f[i])*(!g[j]-g[i])*((cp){1,0}+w[0][i]) ) *
      ↪ (cp){0,0.25};
  }
  dft(t,n>>1,1);
  rep(i,0,n)a[i]=(i&1)?t[i>>1].a:t[i>>1].b;
}
```

FFT MYY MOD

```
cp operator!()const{return (cp){a,-b};};

void mul(int *a,int *b,int n){// n<=N, 0<=a[i],b[i]<mo
  init(n);
  static cp f[N],g[N],t[N],r[N];
  rep(i,0,n){
    f[i]=(cp){a[i]>>15,a[i]&32767};
    g[i]=(cp){b[i]>>15,b[i]&32767};
  }
  dft(f,n,0);dft(g,n,0);
```

```
  rep(i,0,n){
    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)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;
}
```

FFT Naive

```
const db pi = acosl(-1.0);
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};}
}w[2][N+1];
void dft(cp *a,int n,bool v){
  int j=0;
  rep(i,0,n){
    if(i>j)swap(a[i],a[j]);
    for (int l=n>>1;(j^=l)<l;l>>=1);
  }
  for (int i=2;i<=n;i<<=1)
    for (int j=0,s=n/i;j<n;j+=i)
      rep(l,0,i>>1){
        cp t=a[j+l+(i>>1)]*w[v][s*l];
        a[j+l+(i>>1)]=a[j+l]-t;
        a[j+l]=a[j+l]+t;
      }
  if(v)rep(i,0,n)a[i].a/=n,a[i].b/=n;
}
void init(int n){
  rep(i,0,n+1)w[0][i]=(cp){cosl(2*pi/n*i),sinl(2*pi/n*i)};
  rep(i,0,n+1)w[1][i]=w[0][n-i];
}
void mul(cp *a,cp *b,int n){// n<=N
  init(n);
  dft(a,n,0);
  dft(b,n,0);
  rep(i,0,n)a[i]=a[i]*b[i];
  dft(a,n,1);
}
```

Pell Equation

```
import math
len=10010
m=[0 for i in range(0,len)]
d=[0 for i in range(0,len)]
a=[0 for i in range(0,len)]
p=[0 for i in range(0,len)]
q=[0 for i in range(0,len)]
def solve(i,md,re):
  a[0]=int(math.sqrt(i))
  m[0]=0
  d[0]=1
  if (a[0]*a[0]==i): return -1,-1
  pr=1
  while a[pr-1]!=2*a[0]:
    m[pr]=d[pr-1]*a[pr-1]-m[pr-1]
    d[pr]=(i-m[pr]*m[pr])/d[pr-1]
    a[pr]=(a[0]+m[pr])/d[pr]
    pr+=1
  p[0]=1
  q[0]=0
  p[1]=a[0]
  q[1]=1
  it=1
  while 1:
    if (p[it]*p[it]-i*q[it]*q[it]==1): return p[it],q[it]
    it+=1
    p[it]=a[(it-2)%(pr-1)+1]*p[it-1]+p[it-2]
    q[it]=a[(it-2)%(pr-1)+1]*q[it-1]+q[it-2]
while (1):
  s=raw_input()
  if (s=="0"): break
  P,A=map(int,s.split())
  if (P==3):
    res=solve(8*A,2,1)
    if (res[0]==-1): print "Impossible!"
    else: print (res[0]-1)/2,res[1]
  if (P==5):
    res=solve(24*A,6,5)
    if (res[0]==-1 or res[0]%6!=5): print "Impossible!"
    else: print (res[0]+1)/6,res[1]
  if (P==6):
    res=solve(8*A,4,3)
    if (res[0]==-1 or res[0]%4!=3): print "Impossible!"
    else: print (res[0]+1)/4,res[1]
```

Pollard-Rho

```
typedef pair<ll,ll> PLL;
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<=100000000000011) 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(rand()%(n-1)+1,n))
      ↪ return 0;
    return 1;
  }
  ll rho(ll n) {
    for (;;) {
      ll X=rand()%n,Y,Z,T=1,*lY=a,*lX=lY;
      int tmp=20;
      C=rand()%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);
  }
}
}
```

Simplex

```
namespace simplex {
  const db eps=1e-6;
  const int N=110,M=410;
  int n,m;
  int Left[M],Down[N],idx[N],va[N];
  db a[M][N],b[M],c[N],v;
  void init(int p,int q) {
    n=p; m=q;
    rep(i,1,m+1) rep(j,1,n+1) a[i][j]=0;
    rep(j,1,m+1) b[j]=0; rep(i,1,n+1) c[i]=0;
    rep(i,1,n+1) idx[i]=0;
    v=0;
  }
  void pivot(int x,int y) {
    swap(Left[x],Down[y]);
    db k=a[x][y];
    a[x][y]=1; b[x]/=k;
    int t=0;
    rep(j,1,n+1) {
      a[x][j]/=k;
      if (abs(a[x][j])>eps) va[++t]=j;
    }
    rep(i,1,m+1) if(i!=x&&abs(a[i][y])>eps) {
      k=a[i][y];
      a[i][y]=0;
      b[i]-=k*b[x];
      rep(j,1,t+1) a[i][va[j]]-=k*a[x][va[j]];
    }
    k=c[y];
    c[y]=0;
    v+=k*b[x];
    rep(j,1,t+1) c[va[j]]-=k*a[x][va[j]];
  }
  int solve() {
    rep(i,1,n+1) Down[i]=i;
    rep(i,1,m+1) Left[i]=n+i;
    while(1) {
      int x=0;
      rep(i,1,m+1) if (b[i]<-eps&&(x==0||b[i]<b[x])) x=i;
      if(x==0) break;
      int y=0;
      rep(j,1,n+1) if (a[x][j]<-eps) if
        ↪ (y==0||a[x][j]<a[x][y]) y=j;
      if(y==0) { puts("Infeasible"); return -1; } //Infeasible
      pivot(x,y);
    }
    while(1) {
      int y=0;
      rep(i,1,n+1) if (c[i]>eps&&(y==0||c[i]>c[y])) y=i;
      if(y==0) break;
      int x=0;
      rep(j,1,m+1) if (a[j][y]>eps) if
        ↪ (x==0||b[j]/a[j][y]<b[x]/a[x][y]) x=j;
      if(x==0) { puts("Unbounded"); return -2; } // Unbounded
      pivot(x,y);
    }
    printf("%.10lf ",v);
    rep(i,1,m+1) if(Left[i]<=n) idx[Left[i]]=i;
    rep(i,1,n+1) printf("%.10lf ",b[idx[i]]);
    puts("");
    return 1;
  }
}
```

Simpson

```
double F(double x) {
}
double simpson(double a,double b) {
  double c=a+(b-a)/2;
  return (F(a)+F(b)+4*F(c))*(b-a)/6;
}
double rsimpson(double a,double b,double A){
  double c=a+(b-a)/2;
  double L=simpson(a,c),R=simpson(c,b);
  if (fabs(A-L-R)<=eps) return L+R+(A-L-R)/15;
    return rsimpson(a,c,L)+rsimpson(c,b,R);
}
```

Quadratic Residue

```
void mul(ll& a1,ll& b1,ll a2,ll b2,ll w,ll p) {
  ll t1=(a1*a2+b1*b2%p*w),t2=(a1*b2+a2*b1);
  a1=t1%p,b1=t2%p;
}
int Pow(ll a,ll w,ll b,ll p) {
  ll res1=1,res2=0,c1=a,c2=1;
  for (;b;b>>=1) { if (b&1)
    ↪ mul(res1,res2,c1,c2,w,p);mul(c1,c2,c1,c2,w,p);}
  return res1;
}
int solve(ll n,int p) { // x^2=n(mod p)
  ll a,r=0;n%=p;
  if (p==2) return -1;
  if (n==0) return 0;
  if (powmod(n,p/2,p)!=1) return -1;
  do a=rand()%(p-1)+1;while ((powmod(a*a-n,p/2,p)-1)%p==0);
  r=Pow(a,(a*a-n)%p,(p+1)/2,p);
  if (r<0) r+=p;
  assert((r*r-n)%p==0);
  return r;
}
```

FWT

```
void dft(int l,int r) {
  if (l==r) return;
  int md=(l+r)>>1,len=(r-l+1)>>1;
  dft(l,md);dft(md+1,r);
  rep(i,0,len) {
    int x1=a[l+i],x2=a[l+len+i];
    a[l+i]=(x1-x2)%mod;
    a[l+len+i]=(x1+x2)%mod;
  }
}
void idft(int l,int r) {
  if (l==r) return;
  int md=(l+r)>>1,len=(r-l+1)>>1;
  rep(i,0,len) {
    int x1=a[l+i],x2=a[l+len+i];
    a[l+i]=(x1+x2)*inv2%mod;
    a[l+len+i]=(x2-x1)*inv2%mod;
  }
  idft(l,md);idft(md+1,r);
}
```

Linear Recurrence

```
namespace linear_seq {
  const int N=10010;
  ll res[N],base[N],_c[N],_md[N];

  vector<int> Md;
  void mul(ll *a,ll *b,int k) {
    rep(i,0,k+k) _c[i]=0;
    rep(i,0,k) if (a[i]) rep(j,0,k)
      ↪ _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
    for (int i=k+k-1;i>=k;i--) if (_c[i])
      rep(j,0,SZ(Md))
        ↪ _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%mod;
    rep(i,0,k) a[i]=_c[i];
  }
  int solve(ll n,VI a,VI b) { // a   b   b[n+1]=a[0]*b[n]+...
    ll ans=0,pnt=0;
    int k=SZ(a);
    assert(SZ(a)==SZ(b));
    rep(i,0,k) _md[k-1-i]=-a[i];_md[k]=1;
    Md.clear();
    rep(i,0,k) if (_md[i]!=0) Md.push_back(i);
    rep(i,0,k) res[i]=base[i]=0;
    res[0]=1;
```

```
    while ((1ll<<pnt)<=n) pnt++;
    for (int p=pnt;p>=0;p--) {
      mul(res,res,k);
      if ((n>>p)&1) {
        for (int i=k-1;i>=0;i--) res[i+1]=res[i];res[0]=0;
        rep(j,0,SZ(Md))
          ↪ res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%mod;
      }
    }
    rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
    if (ans<0) ans+=mod;
    return ans;
  }
  VI BM(VI s) {
    VI C(1,1),B(1,1);
    int L=0,m=1,b=1;
    rep(n,0,SZ(s)) {
      ll d=0;
      rep(i,0,L+1) d=(d+(ll)C[i]*s[n-i])%mod;
      if (d==0) ++m;
      else if (2*L<=n) {
        VI T=C;
        ll c=mod-d*powmod(b,mod-2)%mod;
        while (SZ(C)<SZ(B)+m) C.pb(0);
        rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
        L=n+1-L; B=T; b=d; m=1;
      } else {
        ll c=mod-d*powmod(b,mod-2)%mod;
        while (SZ(C)<SZ(B)+m) C.pb(0);
        rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
        ++m;
      }
    }
    return C;
  }
  int gao(VI a,ll n) {
    VI c=BM(a);
    c.erase(c.begin());
    rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;
    return solve(n,c,VI(a.begin(),a.begin()+SZ(c)));
  }
};
```

Poly Sum

```
namespace polysum {
  const int D=101000;
  ll a[D],f[D],g[D],p[D],p1[D],p2[D],b[D],h[D][2],C[D];
  ll calcn(int d,ll *a,ll n) {
    if (n<=d) return a[n];
    p1[0]=p2[0]=1;
    rep(i,0,d+1) {
      ll t=(n-i+mod)%mod;
      p1[i+1]=p1[i]*t%mod;
    }
    rep(i,0,d+1) {
      ll t=(n-d+i+mod)%mod;
      p2[i+1]=p2[i]*t%mod;
    }
    ll ans=0;
    rep(i,0,d+1) {
      ll t=g[i]*g[d-i]%mod*p1[i]%mod*p2[d-i]%mod*a[i]%mod;
      if ((d-i)&1) ans=(ans-t+mod)%mod;
      else ans=(ans+t)%mod;
    }
    return ans;
  }
  void init(int M) {
    f[0]=f[1]=g[0]=g[1]=1;
    rep(i,2,M+5) f[i]=f[i-1]*i%mod;
    g[M+4]=powmod(f[M+4],mod-2);
    per(i,1,M+4) g[i]=g[i+1]*(i+1)%mod;
  }
  ll polysum(ll n,ll *a,ll m) { // a[0].. a[m]
    ↪ \sum_{i=0}^{n-1} a[i]
    a[m+1]=calcn(m,a,m+1);
    rep(i,1,m+2) a[i]=(a[i-1]+a[i])%mod;
    return calcn(m+1,a,n-1);
  }
  ll qpolysum(ll R,ll n,ll *a,ll m) { // a[0].. a[m]
    ↪ \sum_{i=0}^{n-1} a[i]*R^i
    if (R==1) return polysum(n,a,m);
    a[m+1]=calcn(m,a,m+1);
    ll r=powmod(R,mod-2),p3=0,p4=0,c,ans;
    h[0][0]=0;h[0][1]=1;
    rep(i,1,m+2) {
      h[i][0]=(h[i-1][0]+a[i-1])*r%mod;
      h[i][1]=h[i-1][1]*r%mod;
    }
    rep(i,0,m+2) {
      ll t=g[i]*g[m+1-i]%mod;
      if (i&1) p3=((p3-h[i][0]*t)%mod+mod)%mod,
        p4=((p4-h[i][1]*t)%mod+mod)%mod;
      else p3=(p3+h[i][0]*t)%mod,p4=(p4+h[i][1]*t)%mod;
    }
    c=powmod(p4,mod-2)*(mod-p3)%mod;
    rep(i,0,m+2) h[i][0]=(h[i][0]+h[i][1]*c)%mod;
    rep(i,0,m+2) C[i]=h[i][0];
    ans=(calcn(m,C,n)*powmod(R,n)-c)%mod;
    if (ans<0) ans+=mod;
    return ans;
  }
}
```

Misc

Multiplication Modulo

```
i64 mul(i64 a, i64 b, i64 mod) {
  a = (a % mod + mod) % mod;
  b = (b % mod + mod) % mod;
  return (a * b - (i64)((f80)a / mod * b + .5L) * mod + mod) %
    ↪ mod;
}
```

1D/1D Dynamic Programming

```
//dp: for i = 1 to n : f[i]= min_{0<=j<i} {f[j]+w(j,i)}
ll f[N];
ll val(int j,int i); //let val(j,i) = f[j]+w(j,i)
void solve(){
  static int vo[N],po[N];
  int p=1,q=0;
  f[0]=0;
  rep(i,0,n){
    while(p<=q && val(i,po[q])<=val(vo[q],po[q]))q--;//
    if(p>q)po[++q]=i+1,vo[q]=i;
    else{
      int l=po[q],r=n;
      while(l<=r){
        int mid=l+r>>1;
        if(val(i,mid)<=val(vo[q],mid))r=mid-1;//
        else l=mid+1;
      }
      if(l<=n) po[++q]=l,vo[q]=i;
    }
    while(p<q && i+1>=po[p+1])p++;
    f[i+1]=val(vo[p],i+1);
  }
}
```

Checker

```
declare -i n=1
while (true)
do
  ./dtmk
  ./my <1.in >my.out
  ./force <1.in >for.out
  if diff my.out for.out
  then
    echo right $n
    n=n+1
  else
    exit
  fi
done
```

STL

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    vector<int> u(9,0); //#number,#init
    make_heap(u.begin(), u.end());
    u.push_back(1); push_heap(u.begin(), u.end()); // last one
      ↪ is inserted
    pop_heap(u.begin(), u.end()); u.pop_back(); // max removed
      ↪ to last
    string S(10,'?');
    S.substr(3,2);//start,length
    S.find("123") != string::npos; // if didn't find
}
```

vimrc

```
set nu ci ai si mouse=a
nmap<F9> : w <cr> : !g++ % -o %< -Wall -Wextra -O2 <cr> :
  ↪ !./%< <cr>
nmap<F8> : w <cr> : !g++ % -o %< -Wall -Wextra -g <cr> : !gdb
  ↪ %< <cr>
nmap<C-F11> : w <cr> : !gedit % <cr>
```

zeller

```
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);
}
int getId(int y, int m, int d) {
  if (m < 3) {y --; m += 12;}
  return 365 * y + y / 4 - y / 100 + y / 400 + (153 * (m - 3)
    ↪ + 2) / 5 + d - 307;
}
```

Dancing Links

```
//exact covering
int n,m;//position num 1~m
int pos[N][M]; // 1<=i<=n, pos[i][0..k-1] : position that the
  ↪ ith block covers. pos[i][k]==0
int l[M],r[M],u[M],d[M],c[M],s[M];
int ndtot;
int cu[M];
int ox[M];
int answer[N];
void del(int x){
  l[r[x]]=l[x];
  r[l[x]]=r[x];
  for (int y=d[x];y!=x;y=d[y]){
    for (int z=r[y];z!=y;z=r[z]){
      u[d[z]]=u[z];
      d[u[z]]=d[z];
    }
  }
}
void res(int x){
  for (int y=u[x];y!=x;y=u[y]){
    for (int z=l[y];z!=y;z=l[z]){
      u[d[z]]=z;
      d[u[z]]=z;
    }
  }
  l[r[x]]=x;
  r[l[x]]=x;
}
int dfs(int dep){
  int p=r[0];
  for (int x=r[0];x;x=r[x])if(s[x]<s[p])p=x;
  int tmp;
  if(p==0)return dep;
  del(p);
  for (int x=d[p];x!=p;x=d[x]){
    answer[dep]=ox[x];
    for (int y=r[x];y!=x;y=r[y])del(c[y]);
    if(tmp=dfs(dep+1))return tmp;
    for (int y=l[x];y!=x;y=l[y])res(c[y]);
  }
  res(p);
  return 0;
}

void work(){
  ndtot=m;
  rep(i,0,m)r[i]=i+1,l[i+1]=i;
  r[m]=0;l[0]=m;
  rep(i,1,m+1)cu[i]=i;
  rep(i,1,n+1){
    ndtot++;
    ox[ndtot]=i;
    s[c[ndtot]=pos[i][0]]++;
    int st=ndtot,la=st;
    for (int j=1;pos[i][j];j++){
      ndtot++;ox[ndtot]=i;s[c[ndtot]=pos[i][j]]++;
      r[la]=ndtot;
      l[ndtot]=la;
      la=ndtot;
    }
    r[la]=st;l[st]=la;
    for (int j=0,x=st;pos[i][j];j++,x++){
      d[cu[pos[i][j]]]=x;
      u[x]=cu[pos[i][j]];
      cu[pos[i][j]]=x;
    }
  }
  rep(i,1,m+1)u[i]=cu[i],d[cu[i]]=i;
  int tmp=dfs(0);
  if(!tmp)printf("No solution\n");
  else{
    //answer[0..tmp-1]
  }
}
```

Surface Walk

```
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;
}
```

Schreier-Sims Algorithm

```
struct Perm {
  int a[N];
  Perm() {
    for (int i = 1; i <= n; ++i) a[i] = i;
  }
  friend Perm operator* (const Perm &lhs, const Perm &rhs) {
    static Perm res;
    for (int i = 1; i <= n; ++i) res.a[i] = lhs.a[rhs.a[i]];
    return res;
  }
  friend Perm inv(const Perm &cur) {
    static Perm res;
    for (int i = 1; i <= n; ++i) res.a[cur.a[i]] = i;
    return res;
  }
};

class Group {
  bool flag[N];
  Perm w[N];
  std::vector<Perm> x;
 public:
  void clear(int p) {
    memset(flag, 0, sizeof flag);
    for (int i = 1; i <= n; ++i) w[i] = Perm();
    flag[p] = true;
    x.clear();
  }
  friend bool check(const Perm&, int);
  friend void insert(const Perm&, int);
  friend void updateX(const Perm&, int);
} g[N];

bool check(const Perm &cur, int k) {
  if (!k) return true;
  int t = cur.a[k];
  return g[k].flag[t] ? check(g[k].w[t] * cur, k - 1) : false;
}

void updateX(const Perm&, int);

void insert(const Perm &cur, int k) {
  if (check(cur, k)) return;
  g[k].x.push_back(cur);
```

```
  for (int i = 1; i <= n; ++i) if (g[k].flag[i]) updateX(cur *
    ↪ inv(g[k].w[i]), k);
}

void updateX(const Perm &cur, int k) {
  int t = cur.a[k];
  if (g[k].flag[t]) {
    insert(g[k].w[t] * cur, k - 1);
  } else {
    g[k].w[t] = inv(cur);
    g[k].flag[t] = true;
    for (int i = 0; i < g[k].x.size(); ++i) updateX(g[k].x[i]
      ↪ * cur, k);
  }
}
```

Java Example

```
import java.io.BufferedInputStream;
import java.math.BigInteger;
import java.util.*;

public class Main {
    static final int N = 7000;
    static final long MOD = 998244353;
    static final BigInteger bigIntegerMOD =
      ↪ BigInteger.valueOf(MOD), TWO = BigInteger.valueOf(2);

    public static BigInteger calcRoot(BigInteger n, int k,
      ↪ BigInteger r) {
        BigInteger l = BigInteger.ONE;
        while (l.compareTo(r) < 0) {
            BigInteger mid =
              ↪ l.add(r).add(BigInteger.ONE).divide(TWO);
            if (mid.pow(k).compareTo(n) <= 0) l = mid; else r
              ↪ = mid.subtract(BigInteger.ONE);
        }
        return l;
    }

    static long power(long base, long exp) {
        long res = 1;
        for (; exp > 0; exp >>= 1) {
            if ((exp & 1) == 1) res = res * base % MOD;
            base = base * base % MOD;
        }
        return res;
    }

    static int[] moe = new int[N];
    static long[] fact = new long[N], inv = new long[N];

    @SuppressWarnings("unchecked")

    static HashMap<Integer, Long>[] hashMap = new HashMap[N];

    static void preprocess() {
        boolean[] flag = new boolean[N];
        ArrayList<Integer> prime = new ArrayList<Integer>();
        moe[1] = 1;
        for (int i = 2; i < N; ++i) {
            if (!flag[i]) {
                prime.add(i);
                moe[i] = -1;
            }
            for (int j : prime) {
                if (i * j >= N) break;
                flag[i * j] = true;
                if (i % j == 0) {
                    moe[i * j] = 0;
                    break;
                }
                moe[i * j] = -moe[i];
            }
        }
        fact[0] = 1;
        for (int i = 1; i < N; ++i) fact[i] = fact[i - 1] * i
          ↪ % MOD;
        inv[N - 1] = power(fact[N - 1], MOD - 2);
        for (int i = N - 2; i >= 0; --i) inv[i] = inv[i + 1] *
          ↪ (i + 1L) % MOD;
        for (int i = 0; i < N; ++i) hashMap[i] = new
          ↪ HashMap<Integer, Long>();
    }

    static long[] cur = new long[N], pre = new long[N], suf =
      ↪ new long[N];

    public static long calcPowerSum(int n, int k) {
        if (hashMap[k].containsKey(n)) return
          ↪ hashMap[k].get(n);
        final int m = k + 2;
        cur[0] = 0;
        for (int i = 1; i <= n && i <= m; ++i) cur[i] = (cur[i
          ↪ - 1] + power(i, k)) % MOD;
        if (n <= m) return cur[n];
        pre[0] = suf[m + 1] = 1;
        for (int i = 1; i <= m; ++i) pre[i] =
          ↪ (int)((long)pre[i - 1] * (n - i + MOD) % MOD);
        for (int i = m; i > 0; --i) suf[i] = (int)((long)suf[i
          ↪ + 1] * (n - i + MOD) % MOD);
        long res = 0;
        for (int i = 1; i <= m; ++i) {
            long val = pre[i - 1] * suf[i + 1] % MOD * inv[i -
              ↪ 1] % MOD * inv[m - i] % MOD;
            if (((m - i) & 1) == 1) val = MOD - val;
            res = (res + cur[i] * val) % MOD;
        }
        hashMap[k].put(n, res);
        return res;
    }

    public static void main(String[] args) {
        preprocess();
        Scanner scanner = new Scanner(new
          ↪ BufferedInputStream(System.in));
        for (int tcase = scanner.nextInt(); tcase-- > 0;) {
            BigInteger n = scanner.nextBigInteger(), cur = n;
            long ans =
              ↪ calcPowerSum(n.mod(bigIntegerMOD).intValue(),
              ↪ 1);
            for (int i = 2; i < N; ++i) {
                cur = calcRoot(n, i, cur);
                if (cur.compareTo(BigInteger.ONE) <= 0) break;
                int val = cur.mod(bigIntegerMOD).intValue();
                for (int j = 1; j * j <= i; ++j) {
                    if (i % j != 0) continue;
                    ans = (ans + moe[j] * calcPowerSum(val,
                      ↪ j)) % MOD;
                    int k = i / j;
                    if (k != j) ans = (ans + moe[k] *
                      ↪ calcPowerSum(val, k)) % MOD;
                }
            }
            System.out.println((ans + MOD - 1) % MOD);
        }
    }
}
```

String Algorithms

AC Automaton

```
// root is 0
// full children version
const int N=100005;
int ch[N][26],fail[N],lab[N],ndtot;

void add(char *s){  //s[0..] ending with '\0'
  int p=0,i;
  for (i=0;s[i];i++){
    if(!ch[p][s[i]-'a'])ch[p][s[i]-'a']=++ndtot;
    p=ch[p][s[i]-'a'];
  }
  lab[p]++;
}
void build(){
  static int qu[N];
  int p=0,q=0;
  qu[q++]=0;
  while(p!=q){
    int u=qu[p++];
    lab[u]+=lab[fail[u]];
    rep(i,0,26)
      if(ch[u][i]){
        qu[q++]=ch[u][i];
        fail[ch[u][i]]=u? ch[fail[u]][i] : 0;
      }else ch[u][i]=ch[fail[u]][i];
  }
}
```

KMP

```
// getpre: str: s[1..n] ans: p[1..n]
p[1]=0;
int j=0;
rep(i,2,n+1){
  while(j && s[j+1]!=s[i])j=p[j];
  if(s[j+1]==s[i])j++;
  p[i]=j;
```

```
}

// matching: short str: s[1..n] (p[1..n]), long str: t[1..m]
// f[i]==n => t[i-n+1..i]=s
int j=0;
rep(i,1,m+1){
  while(j && s[j+1]!=t[i])j=p[j];
  if(s[j+1]==t[i])j++;
  f[i]=j;
  if(j==n)j=p[j];
}
```

Lyndon

```
//s[1..n], return starting position in [1,n]
const int N=1111111;
int minsuf(char *s,int n){
  int las;
  int i,j,k;
  i=1;
  while(i<=n){
    j=i;
    k=i+1;
    while(k<=n && s[j]<=s[k]){
      if(s[j]<s[k])j=i;
      else j++;
      k++;
    }
    while(i<=j){
      las=i; //ans+=s[i..i+k-j-1]
      i+=k-j;
    }
  }
  return las;
}
int maxsuf(char *s,int n){
  char t[N];
  rep(i,1,n+1)t[i]=127-s[i];
  t[++n]=127;
  return minsuf(t,n);
}
int minrot(char *s,int n){//smallest starting pos
  char t[2*N];
  rep(i,1,n+1)t[i]=t[i+n]=s[i];
  n=n*2+1;
  t[n]=127;
  return minsuf(t,n);
}
```

Manacher

```
// str[1..n]
// f[i*2] = len of maxpal center at str[i]
// f[i*2+1] = len of maxpal center at str[i+0.5]
const int N = 100005;
int f[N*2];
void manacher(char* str,int n){
  static char s[N*2];
  rep(i,0,n+1) s[i*2]=str[i], s[i*2+1]='#';
  s[0]='!'; s[2*n+2]='?';
  int a=0,p=0;
  rep(i,1,2*n+2){
    int h=0;
    if(i<=a+p)h=min(f[2*a-i],a+p-i);
    while(s[i+h+1]==s[i-h-1])h++;
    f[i]=h;
    if(i+h>a+p)a=i,p=h;
  }
}
```

Palindromic Tree

```
struct node {
  int ch[26];
  int len,pre;
}nd[N];
int ndtot,last,cnt[N][2];
char s[N],t[N];
ll ret;
void init() {
  nd[1].len=-1; nd[2].len=0;
  nd[1].pre=nd[2].pre=1;
  ndtot=2;
}
void gao(char *s,int ty) {
  int n=strlen(s);
  last=2;
  rep(pos,0,n) {
    int p=last,c=s[pos]-'A';
    for (;s[pos]!=s[pos-nd[p].len-1];p=nd[p].pre);
    if (nd[p].ch[c]) cnt[last=nd[p].ch[c]][ty]++;
    else {
      int np=++ndtot;
      cnt[nd[p].ch[c]=last=np][ty]++;
      nd[np].len=nd[p].len+2;
      if (nd[np].len==1) nd[np].pre=2;
      else {
        for
          ↪ (p=nd[p].pre;s[pos]!=s[pos-nd[p].len-1];p=nd[p].pre);
        nd[np].pre=nd[p].ch[c];
      }
    }
  }
}
```

Suffix Array

```
void buildSA(char *s,int *sa,int *rk,int *h,int n,int m=128) {
  static int X[N],Y[N],c[N];
  int *x=X,*y=Y;
  rep(i,0,m) c[i]=0;
  rep(i,0,n) c[x[i]=s[i]]++;
  rep(i,1,m) c[i]+=c[i-1];
  per(i,0,n) sa[--c[x[i]]]=i;
  for (int k=1;k<n;k<<=1) {
    int p=0;
    per(i,n-k,n) y[p++]=i;
    rep(i,0,n) if (sa[i]>=k) y[p++]=sa[i]-k;
    rep(i,0,m) c[i]=0;
    rep(i,0,n) c[x[y[i]]]++;
    rep(i,1,m) c[i]+=c[i-1];
    per(i,0,n) sa[--c[x[y[i]]]]=y[i];
    swap(x,y);
    p=1; x[sa[0]]=0; y[n]=-1;
    rep(i,1,n) x[sa[i]]=y[sa[i-1]]==y[sa[i]]&&
    y[sa[i-1]+k]==y[sa[i]+k]?p-1:p++;
    if (p==n) break;
    m=p;
  }
  rep(i,0,n)rk[sa[i]]=i;
  x[n]=-1;
  int l=0;
  rep(i,0,n){
    if(l)l--;
    if(rk[i]) {
      for (int j=sa[rk[i]-1]; max(j,i)+l<n &&
        ↪ s[j+l]==s[i+l];l++);
    }
    h[rk[i]]=l;
  }
}
```

Dictionary of Basic Factors

```
struct Info {
  int first, second, last;
  Info() {}
  Info(int x, int y, int z): first(x), second(y), last(z) {}
  void update(int t) {
    last = t;
    if (!first) first = t;
    if (!second || second == first) second = t;
  }
  bool check(int t) {
    if (second == first) return t == first;
    int d = second - first;
    if ((t - first) % d) return false;
    return first <= t && t <= last;
  }
  inline Info& operator+= (int rhs) {
    first += rhs;
    second += rhs;
    last += rhs;
    return *this;
  }
  inline Info& operator-= (int rhs) {
    first -= rhs;
    second -= rhs;
    last -= rhs;
    return *this;
  }
};

class DBF {
  int sa[S][N], rank[S][N], st[S][N], ed[S][N], k;
```

```
  bool comp(int a, int b) {
    if (rank[k - 1][a] != rank[k - 1][b]) return rank[k -
      ↪ 1][a] < rank[k - 1][b];
    a += (1 << (k - 1)), b += (1 << (k - 1));
    return a <= n && b <= n ? (rank[k - 1][a] < rank[k -
      ↪ 1][b]) : (a > b);
  }
 public:
  void build() {
    static int sum[N];
    memset(sum, 0, sizeof sum);
    for (int i = 1; i <= n; ++i) ++sum[rank[0][i] = s[i]];
    for (int i = 1; i < 256; ++i) sum[i] += sum[i - 1];
    for (int i = n; i > 0; --i) sa[0][sum[rank[0][i]]--] = i;
    for (int i = 1; i <= n; ++i) rank[0][sa[0][i]] =
      ↪ rank[0][sa[0][i - 1]] + (s[sa[0][i - 1]] !=
      ↪ s[sa[0][i]]);
    for (k = 1; k < S; ++k) {
      int gap = 1 << (k - 1);
      static int temp[N];
      for (int i = 1; i <= n && i <= gap; ++i) temp[i] = n - i
        ↪ + 1;
      for (int i = 1, tot = std::min(gap, n); i <= n; ++i) if
        ↪ (sa[k - 1][i] - gap > 0) temp[++tot] = sa[k - 1][i]
        ↪ - gap;
      memset(sum, 0, sizeof sum);
      for (int i = 1; i <= n; ++i) ++sum[rank[k - 1][i]];
      for (int i = 1; i <= n; ++i) sum[i] += sum[i - 1];
      for (int i = n; i > 0; --i) sa[k][sum[rank[k -
        ↪ 1][temp[i]]]--] = temp[i];
      for (int i = 1; i <= n; ++i) rank[k][sa[k][i]] =
        ↪ rank[k][sa[k][i - 1]] + comp(sa[k][i - 1],
        ↪ sa[k][i]);
    }
    for (int i = 0; i < S; ++i) {
      for (int j = 1; j <= n; ++j) {
        int t = sa[i][j];
        if (!st[i][rank[i][t]]) st[i][rank[i][t]] = j;
      }
      for (int j = n; j > 0; --j) {
        int t = sa[i][j];
        if (!ed[i][rank[i][t]]) ed[i][rank[i][t]] = j;
      }
    }
  }
  Info ipm(int p, int q, int l, int r) { // find all l <= i <=
    ↪ r such that s[i..i+(1<<p)-1] = s[q..q+(1<<p)-1]
    Info res(0, 0, 0);
    int u = st[p][rank[p][q]], v = ed[p][rank[p][q]];
    int t = std::lower_bound(sa[p] + u, sa[p] + v + 1, l) -
      ↪ sa[p];
    if (u <= t && t <= v && sa[p][t] <= r)
      ↪ res.update(sa[p][t]);
    if (u <= t + 1 && t + 1 <= v && sa[p][t + 1] <= r)
      ↪ res.update(sa[p][t + 1]);
    t = std::upper_bound(sa[p] + u, sa[p] + v + 1, r) - sa[p]
      ↪ - 1;
    if (u <= t && t <= v && sa[p][t] >= l)
      ↪ res.update(sa[p][t]);
    return res;
  }
} dbf;

Info reverse(const Info &info, int t) {
  int d = info.second - info.first;
  Info res;
  res.first = t - info.last + 1;
  res.last = t - info.first + 1;
  res.second = std::min(res.first + d, res.last);
  return res;
}

int merge(Info a, Info b) {
  if (b.second == b.last) std::swap(a, b);
  if (a.second == a.last) {
    if (b.check(a.second)) return a.second;
    return b.check(a.first) ? a.first : 0;
  } else {
    assert(a.second - a.first == b.second - b.first);
    return std::max(a.first, b.first) <= std::min(a.last,
      ↪ b.last) ? std::min(a.last, b.last) : 0;
  }
}

int query(int l, int r) { // range border query
  int k = 32 - __builtin_clz(r - l + 1);
  for (int i = k; i > 0; --i) {
    int t = 1 << (i - 1);
    Info p = dbf.ipm(i - 1, l, std::max(l + 1, r - 2 * t + 2),
      ↪ r - t + 1), q = dbf.ipm(i - 1, r - t + 1, l,
      ↪ std::min(r - t, l + t - 1));
    if (!p.first || !q.first) continue;
    p += t - 1;
    p = reverse(p, r);
    q -= l - 1;
    int res = merge(p, q);
    if (res) return res + t - 1;
  }
  return 0;
}
```

Suffix Automaton

```
const int N=201000;
struct node {
  node *go[4],*p;
  vector<node*> son;
  int val,cnt;
}pool[N],*cur=pool,*rt;
char s[N];
int n;

node* newnode() {
  node *p=cur++;
  p->val=p->cnt=0; p->p=0;
  clr(p->go);
  p->son.clear();
  return p;
}
node *append(node *p,int w) {
  node *np=newnode();np->val=p->val+1;
  for (;p&&!p->go[w];p=p->p) p->go[w]=np;
  if (!p) np->p=rt;
  else {
    node *q=p->go[w];
    if (q->val==p->val+1) np->p=q;
    else {
      node *nq=newnode();
      nq->val=p->val+1;
      memcpy(nq->go,q->go,sizeof(q->go));
      nq->p=q->p;
      np->p=q->p=nq;
      for (;p&&p->go[w]==q;p=p->p) p->go[w]=nq;
    }
  }
  return np;
}
void init() {
  cur=pool;
  rt=newnode();
  node *np=rt;
  per(i,1,n+1) {
    np=append(np,s[i]-'a');
    np->cnt=i;
  }
  for (node *p=pool;p!=cur;p++) if (p->p) p->p->son.pb(p);
}
void dfs(node *p,int sl) {
  int sr=p->val;
  rep(i,0,SZ(p->son)) dfs(p->son[i],sr+1);
}
```

Z Algorithm

```
// getz: str s[1..n]
// ans: z[1..n], z[i]=lcp(s[1..],s[i..])
s[n+1]='?';
z[1]=n;
int a=0;
rep(i,2,n+1){
  z[i]=0;
  if(a && a+z[a]-1>=i)z[i]=min(a+z[a]-i,z[i-a+1]);
  for(;s[1+z[i]]==s[i+z[i]];z[i]++);
  if(!a || i+z[i]>a+z[a])a=i;
}

// zmatch: short str: s[1..n],(z[1..n]) long str: t[1..m]
// ans: f[1..m], f[i]=lcp(s[1..],t[i..])
s[n+1]='?';t[m+1]='!';
int a=0;
rep(i,1,m+1){
  f[i]=0;
  if(a && a+f[a]-1>=i)f[i]=min(a+f[a]-i,z[i-a+1]);
  for (;s[1+f[i]]==t[i+f[i]];f[i]++);
  if(!a || i+f[i]>a+f[a])a=i;
}
```

Appendices

Integral Table

$\int \frac{1}{1+x^2}dx = \tan^{-1}x \quad \int \frac{1}{a^2+x^2}dx = \frac{1}{a}\tan^{-1}\frac{x}{a}$

$\int \frac{x}{a^2+x^2}dx = \frac{1}{2}\ln|a^2+x^2| \quad \int \frac{x^2}{a^2+x^2}dx = x - a\tan^{-1}\frac{x}{a}$

$\int \sqrt{x^2 \pm a^2}dx = \frac{1}{2}x\sqrt{x^2 \pm a^2} \pm \frac{1}{2}a^2 \ln\left|x + \sqrt{x^2 \pm a^2}\right|$

$\int \sqrt{a^2 - x^2}dx = \frac{1}{2}x\sqrt{a^2 - x^2} + \frac{1}{2}a^2\tan^{-1}\frac{x}{\sqrt{a^2-x^2}}$

$\int \frac{x^2}{\sqrt{x^2 \pm a^2}}dx = \frac{1}{2}x\sqrt{x^2 \pm a^2} \mp \frac{1}{2}a^2 \ln\left|x + \sqrt{x^2 \pm a^2}\right|$

$\int \frac{1}{\sqrt{x^2 \pm a^2}}dx = \ln\left|x + \sqrt{x^2 \pm a^2}\right|$

$\int \frac{1}{\sqrt{a^2 - x^2}}dx = \sin^{-1}\frac{x}{a} \quad \int \frac{x}{\sqrt{x^2 \pm a^2}}dx = \sqrt{x^2 \pm a^2} \quad \int \frac{x}{\sqrt{a^2-x^2}}dx = -\sqrt{a^2 - x^2}$

$\int \sqrt{ax^2 + bx + c}dx = \frac{b+2ax}{4a}\sqrt{ax^2 + bx + c} + \frac{4ac-b^2}{8a^{3/2}} \ln\left|2ax + b + 2\sqrt{a(ax^2 + bx + c)}\right|$

$\int x^n e^{ax}\,\mathrm{d}x = \frac{x^n e^{ax}}{a} - \frac{n}{a}\int x^{n-1}e^{ax}\,\mathrm{d}x$

$\int \sin^2 axdx = \frac{x}{2} - \frac{1}{4a}\sin 2ax \quad \int \sin^3 axdx = -\frac{3\cos ax}{4a} + \frac{\cos 3ax}{12a}$

$\int \cos^2 axdx = \frac{x}{2} + \frac{\sin 2ax}{4a} \quad \int \cos^3 axdx = \frac{3\sin ax}{4a} + \frac{\sin 3ax}{12a}$

$\int \tan axdx = -\frac{1}{a}\ln\cos ax \quad \int \tan^2 axdx = -x + \frac{1}{a}\tan ax$

$\int x\cos axdx = \frac{1}{a^2}\cos ax + \frac{x}{a}\sin ax \quad \int x^2\cos axdx = \frac{2x\cos ax}{a^2} + \frac{a^2x^2-2}{a^3}\sin ax$

$\int x\sin axdx = -\frac{x\cos ax}{a} + \frac{\sin ax}{a^2} \quad \int x^2\sin axdx = \frac{2-a^2x^2}{a^3}\cos ax + \frac{2x\sin ax}{a^2}$

Triangle Equations

$\sin(a \pm b) = \sin a\cos b \pm \cos a\sin b \quad \cos(a \pm b) = \cos a\cos b \mp \sin a\sin b$

$\tan(a \pm b) = \frac{\tan(a)\pm\tan(b)}{1\mp\tan(a)\tan(b)} \quad \tan(a) \pm \tan(b) = \frac{\sin(a\pm b)}{\cos(a)\cos(b)}$

$\sin(a) + \sin(b) = 2\sin(\frac{a+b}{2})\cos(\frac{a-b}{2}) \quad \sin(a) - \sin(b) = 2\cos(\frac{a+b}{2})\sin(\frac{a-b}{2})$

$\cos(a) + \cos(b) = 2\cos(\frac{a+b}{2})\cos(\frac{a-b}{2}) \quad \cos(a) - \cos(b) = -2\sin(\frac{a+b}{2})\sin(\frac{a-b}{2})$

$\sin(na) = n\cos^{n-1}a\sin a - \binom{n}{3}\cos^{n-3}a\sin^3 a + \binom{n}{5}\cos^{n-5}a\sin^5 a - \ldots$

$\cos(na) = \cos^n a - \binom{n}{2}\cos^{n-2}a\sin^2 a + \binom{n}{4}\cos^{n-4}a\sin^4 a - \ldots$

Chordal Graph

1. Clique Number $\leq$ Color Number , They are equal in Chordal Graph.
2. Let $next(v)$ be the earliest point in $N(v)$. To check whether $v \cup N(v)$ is a maximal clique, we only have to check whether there exists an w with $Next(w) = v$ and $|N(v)| + 1 \leq |N(w)|$.
3. Minimum Color : Following Perfect Elimination Sequence, from back to front to color each node with the smallest possible color.
4. Maximum Independent Set : Following Perfect Elimination Sequence, from front to back, choose if you can.
5. Maximum Independent Set = Minimum Clique Cover , Minimum Clique Cover : If the Maximum Independent Set is $\{p_1, p_2, \ldots, p_t\}$, Then $\{p_1 \cup N(p_1), \ldots, p_t \cup N(p_t)\}$ is the Minimum Clique Cover.

Maximum Cardinality Search

- Follow the order from n to 1 to label all the vertices, (the vertex with label i is the i-th vertex in the perfect elimination sequence).
- Let L_i denote that how many neighbors of i are labeled vertices, each time we choose an unlabeled vertex with largest L_i.

Constant Table

n	$\log_{10} n$	$n!$	$nC(n/2)$	LCM$(1,\ldots,n)$	P_n	B_n
2	0.30	2	2	2	2	2
3	0.48	6	3	6	3	5
4	0.60	24	6	12	5	15
5	0.70	120	10	60	7	52
6	0.78	720	20	60	11	203
7	0.85	5040	35	420	15	877
8	0.90	40320	70	840	22	4140
9	0.95	362880	126	2520	30	21147
10	1	3628800	252	2520	42	115975
11		39916800	462	27720	56	678570
12		479001600	924	27720	77	4213597
15			6435	360360	176	1382958545
20			184756	232792560	627	
25			5200300		1958	
30			155117520		5604	
40					37338	
50					204226	
70					4087968	
100					190569292	


FACEU