ACM ICPC Reference

University of São Paulo

April 10, 2018

Contents

1. workspaces/teclado 2. .vimrc and .bashrc 3. temp.cpp

```
syntax on
colo evening
set ai si noet ts=4 sw=4 sta sm nu rnu so=7 t_Co=8
imap {<CR> {<CR>}<Esc>O
```

```
import hashlib,sys,string
m = hashlib.md5()
for line in sys.stdin.readlines():
▹   safe = line
▹   line = "".join(line.split())
▹   trim = line
▹   if line.find("//") != -1:
▹   ▹   line = line[:line.find("//")]
▹   m.update(line.encode('utf-8'))
▹   hash = m.hexdigest()[:4]
▹   if trim.endswith("$"):
▹   ▹   hash = "@" + hash + "@" # ignore this
▹   ▹   m = hashlib.md5()
▹   print("%s %s"%(hash,safe), end='')
```

1 Geometry

1.1 Base

```
d41d // typedef double cood; cood eps = 1e-8; // risky: XXX, untested: TODO
00a0 const double pi = acos(-1.);
f2a3 template<typename T> inline T sq(T x) { return x*x; }
7be9 struct vec {
46aa ▹   cood x, y;
2216 ▹   vec () : x(0), y(0) {} vec (cood a, cood b) : x(a), y(b) {}
b76e ▹   inline vec operator - (vec o) { return {x - o.x, y - o.y}; }
6156 ▹   inline vec operator + (vec o) { return {x + o.x, y + o.y}; }
ec28 ▹   inline vec operator * (cood o) { return {x * o, y * o}; }
9949 ▹   inline vec operator / (cood o) { return {x / o, y / o}; }
414e ▹   inline cood operator ˆ (vec o) { return x * o.y - y * o.x; }
9ea2 ▹   inline cood operator * (vec o) { return x * o.x + y * o.y; }
aa02 ▹   inline cood cross (vec a, vec b) { return ((*this)-a) ˆ ((*this)-b); } // |(this)a||(this)b|sen(angle)
b6c2 ▹   inline cood inner (vec a, vec b) { return ((*this)-a) * ((*this)-b); } // |(this)a||(this)b|cos(angle)
85ac ▹   inline double angle (vec a, vec b) { return atan2(cross(a,b),inner(a,b)); } // ccw angle from (this)a to
     (this)b in range [-pi,pi]
6860 ▹   inline int ccw (vec a, vec b) { cood o = cross(a,b); return (eps < o) - (o < -eps); } // this is to the
     (1 left, 0 over, -1 right) of ab
b102 ▹   inline int dir (vec a, vec b) { cood o = inner(a,b); return (eps < o) - (o < -eps); } // a(this) is to
     the (1 same, 0 none, -1 opposite) direction of ab
09b5 ▹   inline cood sq (vec o = vec()) { return inner(o,o); }
3350 ▹   inline double nr (vec o = vec()) { return sqrt(sq(o)); } //$
4e72 ▹   inline vec operator ˜ () { return (*this)/nr(); }
117a ▹   inline vec proj (vec a, vec b) { return a + (b-a)*(a.inner((*this),b) / a.sq(b)); } // projects this onto
     line ab
08dc ▹   inline vec rotate (double a) { return vec(cos(a) * x - sin(a) * y, sin(a) * x + cos(a) * y); } // ccw by
     a radians
2d08 ▹   inline vec rot90 () { return vec(-y,x); } // rotate(pi/2)$
2810 ▹   bool in_seg (vec a, vec b) { return ccw(a,b) == 0 && dir(a,b) <= 0; } // tips included
f04f ▹   double dist2_lin (vec a, vec b) { return a.sq(b) <= eps ? sq(a) : double(::sq(cross(a,b)))/a.sq(b); } //
     see cir.has_inter_lin
4499 ▹   double dist2_seg (vec a, vec b) { return a.dir((*this),b) == (b.dir((*this),a)) ? dist2_lin(a,b) :
     min(sq(a),sq(b)); }
b520 ▹   inline bool operator == (const vec & o) const { return abs(x-o.x) <= eps && abs(y-o.y) <= eps; }
97b1 ▹   inline bool operator < (const vec & o) const { return (abs(x-o.x)>eps)?(x < o.x):(y > o.y); } // lex
     compare (inc x, dec y)
97b1 ▹   // full ccw angle strict compare beginning upwards (this+(0,1)) around (*this)
97b1 ▹   // incresing distance on ties, this is the first
3154 ▹   bool compare (vec a, vec b) {
3834 ▹   ▹   if ((*this < a) != (*this < b)) return *this < b;
c0fb ▹   ▹   int o = ccw(a,b); return o?o>0:((a == *this && !(a == b)) || a.dir(*this,b) < 0);
f0a2 ▹   }
b4bd }; //$
```

```
bafe struct lin { // line
932b ▹ vec p; cood c; // p*(x,y) = c
96eb ▹ lin () {} lin (vec a, cood b) : p(a), c(b) {}
33f9 ▹ lin (vec s, vec t) : p((s-t).rot90()), c(p*s) {}
41f6 ▹ inline lin parll (vec v) { return lin(p,v*p); }
c53d ▹ inline lin perp () { return lin(p.rot90(),c); }
2c29 ▹ vec inter (lin o) { if (vec(0,0).ccw(p,o.p) == 0) throw 1; cood d = (p^o.p); return vec((c*o.p.y -
     p.y*o.c)/d,(o.c*p.x - o.p.x*c)/d); }
b449 ▹ bool contains (vec v) { return abs(p*v - c) <= eps; }
ed12 ▹ vec at_x (cood x) { return vec(x,(c-p.x*x)/p.y); }
bdef ▹ vec at_y (cood y) { return vec((c-y*p.y)/p.x,y); }
709e ▹ double sign_dist (vec v) { return double(p*v - c)/p.nr(); }
5f32 }; //$
3236 struct cir { // circle
5eb6 ▹ vec c; cood r;
957c ▹ cir () {} cir (vec v, cood d) : c(v), r(d) {}
70a5 ▹ cir (vec u, vec v, vec w) { // XXX untreated degenerates
9b1a ▹ ▹ vec mv = (u+v)/2; lin s(mv, mv+(v-u).rot90());
71b3 ▹ ▹ vec mw = (u+w)/2; lin t(mw, mw+(w-u).rot90());
5974 ▹ ▹ c = s.inter(t); r = c.nr(u);
e0bc ▹ }//$
9e54 ▹ inline bool contains (vec w) { return c.sq(w) <= sq(r) + eps; } // border included
9f05 ▹ inline bool border (vec w) { return abs(c.sq(w) - sq(r)) <= eps; }
2582 ▹ inline bool has_inter (cir o) { return c.sq(o.c) <= sq(r + o.r) + eps; } // borders included
b6a1 ▹ inline bool has_border_inter (cir o) { return has_inter(o) && c.sq(o.c) + eps >= sq(r - o.r); }
e4c3 ▹ inline bool has_inter_lin (vec a, vec b) { return a.sq(b) <= eps ? contains(a) : sq(c.cross(a,b)) <=
     sq(r)*a.sq(b) + eps; } // borders included XXX overflow
cc53 ▹ inline bool has_inter_seg (vec a, vec b) { return has_inter_lin(a,b) && (contains(a) || contains(b) ||
     a.dir(c,b)*b.dir(c,a) != -1); } // borders and tips included XXX overflow
e62f ▹ inline double arc_area (vec a, vec b) { return c.angle(a,b)*r*r/2; } // smallest arc, ccw positive
224c ▹ inline double arc_len (vec a, vec b) { return c.angle(a,b)*r; } // smallest arc, ccw positive$
771f ▹ pair<vec,vec> tan (vec v) { // XXX low precision
0d5a ▹ ▹ if (contains(v) && !border(v)) throw 0;
7976 ▹ ▹ cood d2 = c.sq(v); double s = sqrt(d2 - r*r); s = (s==s)?s:0;
19f9 ▹ ▹ double al = atan2(r,s); vec t = (˜(c-v));
a1b3 ▹ ▹ return pair<vec,vec>(v + t.rotate(al)*s, v + t.rotate(-al)*s);
9230 ▹ }//$
c56f ▹ pair<vec,vec> border_inter (cir o) {
513c ▹ ▹ if (!has_border_inter(o) || o.c == (*this).c) throw 0;
1455 ▹ ▹ double a = (sq(r) + o.c.sq(c) - sq(o.r))/(2*o.c.nr(c));
366b ▹ ▹ vec v = (o.c - c)/o.c.nr(c); vec m = c + v * a;
1d79 ▹ ▹ double h = sqrt(sq(r) - sq(a)); h = h!=h?0:h;
288b ▹ ▹ return pair<vec,vec>(m + v.rot90()*h, m - v.rot90()*h);
3fd4 ▹ }//$
5182 ▹ pair<vec,vec> border_inter_lin (vec a, vec b) { // first is closest to a than second
89ee ▹ ▹ if (a.sq(b) <= eps) { if (border(a)) return pair<vec,vec>(a,a); throw 0; }
4ffc ▹ ▹ if (a.dir(b,c) == -1) swap(a,b);
bbb1 ▹ ▹ if (!has_inter_lin(a,b)) throw 0;
9e88 ▹ ▹ double d2 = c.dist2_lin(a,b); vec p = (b-a)/a.nr(b);
cbeb ▹ ▹ double h = sqrt(r*r - d2); h = h!=h?0:h;
07fe ▹ ▹ double y = sqrt(c.sq(a) - d2); y = y!=y?0:y;
c5ab ▹ ▹ return pair<vec,vec>(a + p*(y-h), a + p*(y+h));
8976 ▹ }//$
be35 ▹ double triang_inter (vec a, vec b) { // ccw oriented, this with (c,a,b)
87cb ▹ ▹ if (c.sq(a) > c.sq(b)) return -triang_inter(b,a);
8464 ▹ ▹ if (contains(b)) return c.cross(a,b)/2;
7900 ▹ ▹ if (!has_inter_seg(a,b)) return arc_area(a,b);
6159 ▹ ▹ pair<vec,vec> itr = border_inter_lin(b,a); // order important
b186 ▹ ▹ if (contains(a)) return c.cross(a,itr.first)/2 + arc_area(itr.first,b);
6426 ▹ ▹ return arc_area(a,itr.second) + c.cross(itr.second,itr.first)/2 + arc_area(itr.first,b);
916b ▹ }
42ef }; //$
a71b bool inter_seg (vec a, vec b, vec c, vec d) {
7ff4 ▹ if (a.in_seg(c, d) || b.in_seg(c, d) || c.in_seg(a, b) || d.in_seg(a, b)) return true;
49df ▹ return (c.ccw(a, b) * d.ccw(a, b) == -1 && a.ccw(c, d) * b.ccw(c, d) == -1);
6cc5 }
e074 double dist2_seg (vec a, vec b, vec c, vec d){return inter_seg(a,b,c,d)?0.:min({ a.dist2_seg(c,d),
     b.dist2_seg(c,d), c.dist2_seg(a,b), d.dist2_seg(a,b) });}
```

1.2 Advanced

```
484c cir min_spanning_circle (vec * v, int n) { // n
02d0 ▹ srand(time(NULL)); random_shuffle(v, v+n); cir c(vec(), 0); int i,j,k;
81a2 ▹ for (i = 0; i < n; i++) if (!c.contains(v[i]))
1d61 ▹ ▹ for (c = cir(v[i],0), j = 0; j < i; j++) if (!c.contains(v[j]))
69a5 ▹ ▹ ▹ for (c = cir((v[i] + v[j])/2,v[i].nr(v[j])/2), k = 0; k < j; k++) if (!c.contains(v[k]))
47f4 ▹ ▹ ▹ ▹ ▹ c = cir(v[i],v[j],v[k]);
3242 ▹ return c;
2c43 }//$
d45c int convex_hull (vec * v, int n, int border_in) { // nlg | border_in (should border points stay?)
6414 ▹ swap(v[0], *min_element(v,v+n)); int s, i;
2239 ▹ sort(v+1, v+n, [&v] (vec a, vec b) { int o = b.ccw(v[0], a); return (o?o==1:v[0].sq(a)<v[0].sq(b)); });
72b3 ▹ if (border_in) {
404c ▹ ▹ for (s = n-1; s > 1 && v[s].ccw(v[s-1],v[0]) == 0; s--);
b41d ▹ ▹ reverse(v+s, v+n);
9998 ▹ }
7bbb ▹ for (i = s = 0; i < n; i++) if (!s || !(v[s-1] == v[i])) {
caa7 ▹ ▹ for (; s >= 2 && v[s-1].ccw(v[s-2],v[i]) >= border_in; s--);
62fb ▹ ▹ swap(v[s++],v[i]);
a0cd ▹ }
f648 ▹ return s;
847b }//$
79b9 int monotone_chain (vec * v, int n, int border_in) { // nlg | border_in (should border points stay?)
8814 ▹ vector<vec> r; sort(v, v+n); n = unique(v, v+n) - v;
10ad ▹ for (int i = 0; i < n; r.pb(v[i++])) while (r.size() >= 2 && r[r.size()-2].ccw(r.back(),v[i]) <=
     -border_in) r.pop_back();
2ac3 ▹ r.pop_back(); unsigned int s = r.size();
fd1f ▹ for (int i = n-1; i >= 0; r.pb(v[i--])) while (r.size() >= s+2 && r[r.size()-2].ccw(r.back(),v[i]) <=
     -border_in) r.pop_back();
fa34 ▹ return copy(r.begin(), r.end() - (r.size() > 1), v) - v;
42e7 }//$
f80f double polygon_inter (vec * p, int n, cir c) { // signed area
aedd ▹ return inner_product(p, p+n-1, p+1, c.triang_inter(p[n-1],p[0]), std::plus<double>(), [&c] (vec a, vec b)
     { return c.triang_inter(a,b); });
f00d }//$
3214 int polygon_pos (vec * p, int n, vec v) { // lg | p should be simple (-1 out, 0 border, 1 in)
0858 ▹ int in = -1; // it's a good idea to randomly rotate the points in the double case, numerically safer
7574 ▹ for (int i = 0; i < n; i++) {
b3c1 ▹ ▹ vec a = p[i], b = p[i?i-1:n-1]; if (a.x > b.x) swap(a,b);
ec10 ▹ ▹ if (a.x + eps <= v.x && v.x < b.x + eps) { in *= v.ccw(a,b); }
9cc1 ▹ ▹ else if (v.in_seg(a,b)) { return 0; }
71bf ▹ }
b9cd ▹ return in;
d218 }//$
271f int polygon_pos_convex (vec * p, int n, vec v) { // lg(n) | (-1 out, 0 border, 1 in) TODO
0b37 ▹ if (v.sq(p[0]) <= eps) return 0;
37ed ▹ if (n <= 1) { return 0; } if (n == 2) { return v.in_seg(p[0],p[1])?0:-1; }
c73f ▹ if (v.ccw(p[0],p[1]) < 0 || v.ccw(p[0],p[n-1]) > 0) return -1;
5d39 ▹ int di = lower_bound(p+1,p+n-1,v, [&p](vec a,vec v) { return v.ccw(p[0],a) > 0; }) - p;
a43b ▹ if (di == 1) return v.ccw(p[1],p[2]) >= 0?0:-1;
e357 ▹ return v.ccw(p[di-1],p[di]);
657e }//$
d41d // v is the pointset, w is auxiliary with size at least equal to v's
bf98 cood closest_pair (vec * v, vec * w, int l, int r, bool sorted = 0) { // nlg | r is exclusive TODO (AC on
     cf, no test)
2cb9 ▹ if (l + 1 >= r) return inf;
ee44 ▹ if (!sorted) sort(v+l,v+r,[](vec a, vec b){ return a.x < b.x; });
1734 ▹ int m = (l+r)/2; cood x = v[m].x;
065c ▹ cood res = min(closest_pair(v,w,l,m,1),closest_pair(v,w,m,r,1));
8c2f ▹ merge(v+l,v+m,v+m,v+r,w+l,[](vec a, vec b){ return a.y < b.y; });
a422 ▹ for (int i = l, s = l; i < r; i++) if (sq((v[i] = w[i]).x - x) < res) {
e49d ▹ ▹ for (int j = s-1; j >= l && sq(w[i].y - w[j].y) < res; j--)
ff85 ▹ ▹ ▹ res = min(res, w[i].sq(w[j]));
c098 ▹ ▹ w[s++] = v[i];
1d0c ▹ }
185f ▹ return res;
0fe5 }//$
ac2e double union_area (cir * v, int n) { // n^2lg | XXX joins equal circles TODO (AC on szkopul, no tests)
```

```
6608 ▹ struct I { vec v; int i; } c[2*(n+4)];
f89e ▹ srand(time(NULL)); cood res = 0; vector<bool> usd(n);
7692 ▹ cood lim = 1./0.; for (int i = 0; i < n; i++) lim = min(lim, v[i].c.y - v[i].r - 1);
a056 ▹ for (int i = 0, ss = 0; i < n; i++, ss = 0) {
6e6f ▹ ▹ vec fp = v[i].c + vec(0,v[i].r).rotate(rand()); // rotation avoids corner on cnt initialization
179e ▹ ▹ int cnt = 0, eq = 0;
dc3c ▹ ▹ for (int j = 0; j < n; j++) {
5bac ▹ ▹ ▹ cnt += (usd[j] = v[j].contains(fp));
8c46 ▹ ▹ ▹ if (!v[i].has_border_inter(v[j])) continue;
0bc3 ▹ ▹ ▹ if (v[i].c == v[j].c) eq++;
367c ▹ ▹ ▹ else {
ceb8 ▹ ▹ ▹ ▹ pair<vec,vec> r = v[i].border_inter(v[j]);
f633 ▹ ▹ ▹ ▹ c[ss++] = {r.first, j}; c[ss++] = {r.second, j};
97f8 ▹ ▹ ▹ }
b78e ▹ ▹ }
eda8 ▹ ▹ vec d = vec(v[i].r,0); for (int k = 0; k < 4; k++, d = d.rot90()) c[ss++] = {v[i].c + d, i};
10fb ▹ ▹ int md = partition(c,c+ss,[v,i,fp](I a){return a.v.ccw(v[i].c,fp) > 0;}) - c;
c791 ▹ ▹ sort(c,c+md,[v,i](I a,I b){return a.v.ccw(v[i].c,b.v) < 0;});
7b38 ▹ ▹ sort(c+md,c+ss,[v,i](I a,I b){return a.v.ccw(v[i].c,b.v) < 0;});
50c1 ▹ ▹ for (int j = 0; j < ss; j++) {
6a44 ▹ ▹ ▹ if (c[j].i != i) { cnt -= usd[c[j].i]; usd[c[j].i] = !usd[c[j].i]; cnt += usd[c[j].i]; }
fd4b ▹ ▹ ▹ vec a = c[j].v, b = c[(j+1)%ss].v;
9349 ▹ ▹ ▹ cood cir = abs(v[i].arc_area(a,b) - v[i].c.cross(a,b)/2), tra = abs((b.x-a.x)*(a.y+b.y-2*lim)/2);
fc4b ▹ ▹ ▹ cood loc = (a.x<b.x)?cir-tra:tra+cir; res += (cnt==eq)?loc/eq:0;
8621 ▹ ▹ }
e0d3 ▹ }
bb62 ▹ return res;
c3ac }//$
4ede pii antipodal (vec * p, int n, vec v) { // lg(n) | extreme segments relative to direction v TODO
4ede ▹ // po: closest to dir, ne: furthest from dir
b196 ▹ bool sw = ((p[1]-p[0])*v < 0);
7136 ▹ if (sw) v = vec(0,0) - v; // lower_bound returns the first such that lambda is false
62d0 ▹ int md = lower_bound(p+1, p+n, v, [p] (vec & a, vec v) { return (a-p[0])*v > eps; }) - p; // chain separation
e770 ▹ int po = lower_bound(p, p+md-1, v, [p,n] (vec & a, vec v) { return (p[(&a+1-p)%n]-a)*v > eps; }) - p; // positive
e804 ▹ int ne = (lower_bound(p+md, p+n, v, [p,n] (vec & a, vec v) { return (p[(&a+1-p)%n]-a)*v <= eps; }) - p)%n; // negative
9cc3 ▹ if (sw) swap(po,ne);
2bf4 ▹ return pii(po,ne);
eeb0 }//$
34e2 int mink_sum (vec * a, int n, vec * b, int m, vec * r) { // (n+m) | a[0]+b[0] should belong to sum, doesn't create new border points TODO
4f7e ▹ if (!n || !m) { return 0; } int i, j, s; r[0] = a[0] + b[0];
b6c1 ▹ for (i = 0, j = 0, s = 1; i < n || j < m; s++) {
1036 ▹ ▹ if (i >= n) j++;
a814 ▹ ▹ else if (j >= m) i++;
8fef ▹ ▹ else {
4524 ▹ ▹ ▹ int o = (a[(i+1)%n]+b[j%m]).ccw(r[s-1],a[i%n]+b[(j+1)%m]);
fef4 ▹ ▹ ▹ j += (o >= 0); i += (o <= 0);
39e2 ▹ ▹ }
1433 ▹ ▹ r[s] = a[i%n] + b[j%m];
0b44 ▹ }
dc7e ▹ return s-1;
3a46 }//$
9e65 int inter_convex (vec * p, int n, vec * q, int m, vec * r) { // (n+m) | XXX
b106 ▹ int a = 0, b = 0, aa = 0, ba = 0, inflag = 0, s = 0;
079f ▹ while ((aa < n || ba < m) && aa < n+n && ba < m+m) {
67e5 ▹ ▹ vec p1 = p[a], p2 = p[(a+1)%n], q1 = q[b], q2 = q[(b+1)%m];
d8ba ▹ ▹ vec A = p2 - p1, B = q2 - q1;
538a ▹ ▹ int cross = vec(0,0).ccw(A,B), ha = p1.ccw(p2,q2), hb = q1.ccw(q2,p2);
b030 ▹ ▹ if (cross == 0 && p2.ccw(p1,q1) == 0 && A*B < -eps) {
a6f1 ▹ ▹ ▹ if (q1.in_seg(p1,p2)) r[s++] = q1;
1d07 ▹ ▹ ▹ if (q2.in_seg(p1,p2)) r[s++] = q2;
c97f ▹ ▹ ▹ if (p1.in_seg(q1,q2)) r[s++] = p1;
ab44 ▹ ▹ ▹ if (p2.in_seg(q1,q2)) r[s++] = p2;
42a9 ▹ ▹ ▹ if (s < 2) return s;
359f ▹ ▹ ▹ inflag = 1; break;
463c ▹ ▹ } else if (cross != 0 && inter_seg(p1,p2,q1,q2)) {
```

```
32e2 ▹ ▹ ▹ if (inflag == 0) aa = ba = 0;
db5d ▹ ▹ ▹ r[s++] = lin(p1,p2).inter(lin(q1,q2));
53ab ▹ ▹ ▹ inflag = (hb > 0) ? 1 : -1;
74c9 ▹ ▹ }
8f3d ▹ ▹ if (cross == 0 && hb < 0 && ha < 0) return s;
e020 ▹ ▹ bool t = cross == 0 && hb == 0 && ha == 0;
531b ▹ ▹ if (t ? (inflag == 1) : (cross >= 0) ? (ha <= 0) : (hb > 0)) {
79e9 ▹ ▹ ▹ if (inflag == -1) r[s++] = q2;
d590 ▹ ▹ ▹ ba++; b++; b %= m;
0d16 ▹ ▹ } else {
997e ▹ ▹ ▹ if (inflag == 1) r[s++] = p2;
ced6 ▹ ▹ ▹ aa++; a++; a %= n;
1018 ▹ ▹ }
b45e ▹ }
20c0 ▹ if (inflag == 0) {
0313 ▹ ▹ if (polygon_pos_convex(q,m,p[0]) >= 0) { copy(p, p+n, r); return n; }
3fdb ▹ ▹ if (polygon_pos_convex(p,n,q[0]) >= 0) { copy(q, q+m, r); return m; }
8a67 ▹ }
42c5 ▹ s = unique(r, r+s) - r;
97ad ▹ if (s > 1 && r[0] == r[s-1]) s--;
8ede ▹ return s;
7316 }//$
03ae bool isear (vec * p, int n, int i, int prev[], int next[]) { // aux to triangulate
0dbc ▹ vec a = p[prev[i]], b = p[next[i]];
0cac ▹ if (b.ccw(a,p[i]) <= 0) return false;
1da0 ▹ for (int j = 0; j < n; j++) {
907a ▹ ▹ if (j == prev[i] || j == next[i]) continue;
94d7 ▹ ▹ if (p[j].ccw(a,p[i]) >= 0 && p[j].ccw(p[i],b) >= 0 && p[j].ccw(b,a) >= 0) return false;
03c1 ▹ ▹ int k = (j+1)%n;
0198 ▹ ▹ if (k == prev[i] || k == next[i]) continue;
3e37 ▹ ▹ if (inter_seg(p[j],p[k],a,b)) return false;
ff02 ▹ }
b36b ▹ return true;
3cde }
8c27 int triangulate (vec * p, int n, bool ear[], int prev[], int next[], int tri[][3]) { // O(n^2) | n >= 3
b177 ▹ int s = 0, i = 0;
d9fd ▹ for (int i = 0, prv = n-1; i < n; i++) { prev[i] = prv; prv = i; next[i] = (i+1)%n; ear[i] =
    isear(p,n,i,prev,next); }
0c93 ▹ for (int lef = n; lef > 3; lef--, i = next[i]) {
fd01 ▹ ▹ while (!ear[i]) i = next[i];
5afb ▹ ▹ tri[s][0] = prev[i]; tri[s][1] = i; tri[s][2] = next[i]; s++; // tri[i][0],i,tri[i][1] inserted
21a0 ▹ ▹ int c_prev = prev[i], c_next = next[i];
1639 ▹ ▹ next[c_prev] = c_next; prev[c_next] = c_prev;
4b6a ▹ ▹ ear[c_prev] = isear(p,n,c_prev,prev,next); ear[c_next] = isear(p,n,c_next,prev,next);
34ef ▹ }
172c ▹ tri[s][0] = next[next[i]]; tri[s][1] = i; tri[s][2] = next[i]; s++; // tri[i][0],i,tri[i][1] inserted
d3aa ▹ return s;
da3e }
```

1.3 3D

```
f61c const double pi = acos(-1);
f61c // typedef double cood; cood eps = 1e-6; // risky: XXX, untested: TODO
1402 struct pnt { // TODO it's not tested at all :)
c77e ▹ cood x, y, z;
46f7 ▹ pnt () : x(0), y(0), z(0) {} pnt (cood a, cood b, cood c) : x(a), y(b), z(c) {}
a570 ▹ inline pnt operator - (pnt o) { return pnt(x - o.x, y - o.y, z - o.z); }
f033 ▹ inline pnt operator + (pnt o) { return pnt(x + o.x, y + o.y, z + o.z); }
b62d ▹ inline pnt operator * (cood o) { return pnt(x*o, y*o, z*o); }
05ae ▹ inline pnt operator / (cood o) { return pnt(x/o, y/o, z/o); }
11be ▹ inline cood operator * (pnt o) { return x*o.x + y*o.y + z*o.z; } // inner: |this||o|*cos(ang)
1527 ▹ inline pnt operator ^ (pnt o) { return pnt(y*o.z - z*o.y, z*o.x - x*o.z, x*o.y - y*o.x); } // cross:
    oriented normal to the plane containing the two vectors, has norm |this||o|*sin(ang)
2194 ▹ inline cood operator () (pnt a, pnt b) { return (*this)*(a^b); } // mixed: positive on the right-hand
    rule (thumb=this,index=a,mid=b)
2194
a475 ▹ inline cood inner (pnt a, pnt b) { return (a-(*this))*(b-(*this)); }
0f48 ▹ inline pnt cross (pnt a, pnt b) { return (a-(*this))^(b-(*this)); } // its norm is twice area of triangle
```

```
af7c ▹ inline cood mixed (pnt a, pnt b, pnt c) { return (a-(*this))(b-(*this),c-(*this)); } // 6 times the
    oriented area of thetahedra
af7c
97f8 ▹ inline cood sq (pnt o = pnt()) { return inner(o,o); }
8f77 ▹ inline double nr (pnt o = pnt()) { return sqrt(sq(o)); }
f892 ▹ inline pnt operator ˜ () { return (*this)/nr(); }
f892
3be9 ▹ inline bool in_seg (pnt a, pnt b) { return cross(a,b).sq() <= eps && inner(a,b) <= eps; } // tips included
ed92 ▹ inline bool in_tri (pnt a, pnt b, pnt c) { return abs(mixed(a,b,c)) <= eps && cross(a,b)*cross(b,c) >=
    -eps && cross(a,b)*cross(c,a) >= -eps; } // border included$
d41d
7c26 ▹ inline pnt proj (pnt a, pnt b) { return a + (b-a)*a.inner(b,(*this))/a.sq(b); }
8091 ▹ inline pnt proj (pnt a, pnt b, pnt c) { pnt n = a.cross(b,c); return (*this) - n*(n*((*this)-a))/n.sq(); }
8091
4ec5 ▹ inline double dist2_lin (pnt a, pnt b) { return cross(a,b).sq()/a.sq(b); }
6652 ▹ inline double dist2_seg (pnt a, pnt b) { return a.inner(b,(*this))*b.inner(a,(*this)) <= eps ?
    min(sq(a),sq(b)) : dist2_lin(a,b); }
de18 ▹ inline double dist_pln (pnt a, pnt b, pnt c) { return abs((˜a.cross(b,c))*((*this)-a)); }
c64c ▹ inline double dist2_tri (pnt a, pnt b, pnt c) { pnt p = proj(a,b,c); return p.in_tri(a,b,c) ? sq(p) :
    min({ dist2_seg(a,b), dist2_seg(b,c), dist2_seg(c,a) }); }
8dba };
bd9a inline cood area (pnt a, pnt b, pnt c) { return abs(a.cross(b,c).nr()) / 2; }
f7e9 inline cood vol (pnt a, pnt b, pnt c, pnt d) { return abs(a.mixed(b,c,d)) / 6; } // thetahedra
f2e0 pnt inter_lin_pln (pnt s, pnt t, pnt a, pnt b, pnt c) { pnt n = a.cross(b,c); return s +
    (t-s)*(n*(a-s))/(n*(t-s)); } //$
fabc struct sph { // TODO it's also not tested at all
b698 ▹ pnt c; cood r;
021e ▹ sph () : c(), r(0) {} sph (pnt a, cood b) : c(a), r(b) {}
35ac ▹ inline pnt operator () (cood lat, cood lon) { return c + pnt(cos(lat)*cos(lon), sin(lon), sin(lat))*r; }
    // (1,0,0) is (0,0). z is height.
5e05 ▹ inline double area_hull (double h) { return 2.*pi*r*h; }
1fb9 ▹ inline double vol_hull (double h) { return pi*h/6 * (3.*r*r + h*h); }
f2bb };
```

2 Graphs

2.1 Dinic

```
d41d //typedef int num; const int N = ; const int M = * 2; const num eps = 0;
582d struct dinic {
43e6 ▹ int hd[N], seen[N], qu[N], lv[N], ei[N], to[M], nx[M]; num fl[M], cp[M]; int en = 2; int tempo = 0;
c364 ▹ bool bfs(int s, int t) {
7e88 ▹ ▹ seen[t] = ++tempo; lv[t] = 0; int ql = 0, qr = 0; qu[qr++] = t;
21ca ▹ ▹ while(ql != qr) {
d50e ▹ ▹ ▹ t = qu[ql++]; ei[t] = hd[t]; if(s == t) return true;
b9fb ▹ ▹ ▹ for(int e = hd[t]; e; e = nx[e]) if(seen[to[e]] != tempo && cp[e ˆ 1] - fl[e ˆ 1] > eps) {
3bcd ▹ ▹ ▹ ▹ seen[to[e]] = tempo;
21ae ▹ ▹ ▹ ▹ lv[to[e]] = lv[t] + 1;
4d73 ▹ ▹ ▹ ▹ qu[qr++] = to[e];
82ea ▹ ▹ ▹ }
1312 ▹ ▹ }
72ae ▹ ▹ return false;
87db ▹ }
c08d ▹ num dfs(int s, int t, num f) {
747c ▹ ▹ if(s == t) return f;
bac0 ▹ ▹ for(int &e = ei[s]; e; e = nx[e]) if(ei[to[e]] && seen[to[e]] == tempo && cp[e] - fl[e] > eps &&
    lv[to[e]] == lv[s] - 1)
d7b6 ▹ ▹ ▹ if(num rf = dfs(to[e], t, min(f, cp[e] - fl[e]))) {
b79e ▹ ▹ ▹ ▹ fl[e] += rf;
e6cd ▹ ▹ ▹ ▹ fl[e ˆ 1] -= rf;
d77a ▹ ▹ ▹ ▹ return rf;
ff34 ▹ ▹ ▹ }
1f5d ▹ ▹ return 0;
edb8 ▹ }
edb8 ▹ // public $
de22 ▹ num max_flow(int s, int t) {
f240 ▹ ▹ num fl = 0;
da71 ▹ ▹ while (bfs(s, t)) for(num f; (f = dfs(s, t, numeric_limits<num>::max())); fl += f);
91c9 ▷ ▷ return fl;
c7e1 ▷ }
9df3 ▷ void add_edge(int a, int b, num c, num rc=0) {
4de4 ▷ ▷ to[en] = b; nx[en] = hd[a]; fl[en] = 0; cp[en] = c; hd[a] = en++;
7c19 ▷ ▷ to[en] = a; nx[en] = hd[b]; fl[en] = 0; cp[en] = rc; hd[b] = en++;
e0af ▷ }
1dac ▷ void reset_flow() { memset(fl, 0, sizeof(num) * en); }
578f ▷ void init(int n=N) { en = 2; memset(hd, 0, sizeof(int) * n); } // resets all
1ab8 };
```

2.2 MinCost MaxFlow

```
d41d //typedef int val; // type of flow
d41d //typedef int num; // type of cost
d41d //const int N = , M = * 2; const num eps = 0;
1854 struct mcmf {
4266 ▷ int es[N], to[M], nx[M], en = 2, pai[N], seen[N], tempo, qu[N];
aa14 ▷ val fl[M], cp[M], flow; num cs[M], d[N], tot;
a0c9 ▷ val spfa(int s, int t) {
1bc7 ▷ ▷ tempo++; int a = 0, b = 0;
9b58 ▷ ▷ for(int i = 0; i < N; i++) d[i] = numeric_limits<num>::max();
d373 ▷ ▷ d[s] = 0; qu[b++] = s; seen[s] = tempo;
e936 ▷ ▷ while(a != b) {
0d58 ▷ ▷ ▷ int u = qu[a++]; if(a == N) a = 0; seen[u] = 0;
1d1c ▷ ▷ ▷ for(int e = es[u]; e; e = nx[e]) if(cp[e] - fl[e] > val(0) && d[u] + cs[e] < d[to[e]] - eps) {
1b7b ▷ ▷ ▷ ▷ d[to[e]] = d[u] + cs[e]; pai[to[e]] = e ^ 1;
2327 ▷ ▷ ▷ ▷ if(seen[to[e]] < tempo) { seen[to[e]] = tempo; qu[b++] = to[e]; if(b == N) b = 0; }
9ee9 ▷ ▷ ▷ }
5c4b ▷ ▷ }
9c6c ▷ ▷ if(d[t] == numeric_limits<num>::max()) return false;
e606 ▷ ▷ val mx = numeric_limits<val>::max();
1475 ▷ ▷ for(int u = t; u != s; u = to[pai[u]])
dfd0 ▷ ▷ ▷ mx = min(mx, cp[pai[u] ^ 1] - fl[pai[u] ^ 1]);
538e ▷ ▷ tot += d[t] * val(mx);
3659 ▷ ▷ for(int u = t; u != s; u = to[pai[u]])
24b6 ▷ ▷ ▷ fl[pai[u]] -= mx, fl[pai[u] ^ 1] += mx;
0594 ▷ ▷ return mx;
58b2 ▷ }
58b2 ▷ // public $
8662 ▷ num min_cost(int s, int t) {
59d8 ▷ ▷ tot = 0; flow = 0;
cb91 ▷ ▷ while(val a = spfa(s, t)) flow += a;
7fe2 ▷ ▷ return tot;
a0f1 ▷ }
0f48 ▷ void add_edge(int u, int v, val c, num s) {
eca8 ▷ ▷ fl[en] = 0; cp[en] = c; to[en] = v; nx[en] = es[u]; cs[en] = s; es[u] = en++;
20b8 ▷ ▷ fl[en] = 0; cp[en] = 0; to[en] = u; nx[en] = es[v]; cs[en] = -s; es[v] = en++;
c98e ▷ }
3a9d ▷ void reset_flow() { memset(fl, 0, sizeof(val) * en); }
f6de ▷ void init(int n) { en = 2; memset(es, 0, sizeof(int) * n); } // XXX must be called
3ef5 };
```

2.3 Cycle Cancelling

```
d41d //typedef int val; // type of flow
d41d //typedef int num; // type of cost
d41d //const int N = ; const int M = * 2; const val eps = 0;
afb2 struct cycle_cancel {
5141 ▷ int hd[N], seen[N], qu[N], lv[N], ei[N], to[M], nx[M], ct[N], pai[N]; val fl[M], cp[M], flow; num cs[M],
     d[N], tot; int en = 2, n; int tempo = 0;
5179 ▷ bool bfs(int s, int t) {
feea ▷ ▷ seen[t] = ++tempo; lv[t] = 0; int ql = 0, qr = 0; qu[qr++] = t;
0484 ▷ ▷ while(ql != qr) {
a6bf ▷ ▷ ▷ t = qu[ql++]; ei[t] = hd[t]; if(s == t) return true;
6c0b ▷ ▷ ▷ for(int e = hd[t]; e; e = nx[e]) if(seen[to[e]] != tempo && cp[e ^ 1] - fl[e ^ 1] > eps) {
e600 ▷ ▷ ▷ ▷ seen[to[e]] = tempo;
bfe8 ▷ ▷ ▷ ▷ lv[to[e]] = lv[t] + 1;
ef44 ▹ ▹ ▹ ▹ qu[qr++] = to[e];
a3ce ▹ ▹ ▹ }
910f ▹ ▹ }
fd44 ▹ ▹ return false;
3f5a ▹ }
e145 ▹ val dfs(int s, int t, val f) {
8e37 ▹ ▹ if(s == t) return f;
c5f3 ▹ ▹ for(int &e = ei[s]; e; e = nx[e]) if(ei[to[e]] && seen[to[e]] == tempo && cp[e] - fl[e] > eps &&
    lv[to[e]] == lv[s] - 1)
bc35 ▹ ▹ ▹ if(val rf = dfs(to[e], t, min(f, cp[e] - fl[e]))) {
2d62 ▹ ▹ ▹ ▹ fl[e] += rf;
25b3 ▹ ▹ ▹ ▹ fl[e ^ 1] -= rf;
39a1 ▹ ▹ ▹ ▹ return rf;
aff4 ▹ ▹ ▹ }
71e1 ▹ ▹ return 0;
0404 ▹ }
92d6 ▹ bool spfa() {
5126 ▹ ▹ tempo++; int a = 0, b = 0, u;
54b8 ▹ ▹ for(int i = 0; i < n; i++) { d[i] = 0; qu[b++] = i; seen[i] = tempo; ct[i] = 0; }
973a ▹ ▹ while(a != b) {
5804 ▹ ▹ ▹ u = qu[a++]; if(a == N) a = 0; seen[u] = 0;
238d ▹ ▹ ▹ if(ct[u]++ >= n + 1) { a--; break; }
7085 ▹ ▹ ▹ for(int e = hd[u]; e; e = nx[e]) if(cp[e] - fl[e] > val(0) && d[u] + cs[e] < d[to[e]] - eps) {
73bb ▹ ▹ ▹ ▹ d[to[e]] = d[u] + cs[e]; pai[to[e]] = e ^ 1;
e89b ▹ ▹ ▹ ▹ if(seen[to[e]] < tempo) { seen[to[e]] = tempo; qu[b++] = to[e]; if(b == N) b = 0; }
6d87 ▹ ▹ ▹ }
5b16 ▹ ▹ }
4689 ▹ ▹ if(a == b) return false;
72bc ▹ ▹ val mn = numeric_limits<val>::max();
ff79 ▹ ▹ tempo++;
1a94 ▹ ▹ for(; seen[u] != tempo; u = to[pai[u]]) seen[u] = tempo;
c7cd ▹ ▹ for(int v = u; seen[v] != tempo + 1; v = to[pai[v]]) {
a2a0 ▹ ▹ ▹ seen[v] = tempo + 1;
ab99 ▹ ▹ ▹ mn = min(mn, cp[pai[v] ^ 1] - fl[pai[v] ^ 1]);
9be8 ▹ ▹ }
ab37 ▹ ▹ for(int v = u; seen[v] == tempo + 1; v = to[pai[v]]) {
5999 ▹ ▹ ▹ seen[v] = 0;
f906 ▹ ▹ ▹ fl[pai[v]] -= mn;
0d8f ▹ ▹ ▹ fl[pai[v] ^ 1] += mn;
f849 ▹ ▹ }
a941 ▹ ▹ return true;
3e77 ▹ }
cf9e ▹ val max_flow(int s, int t) {
23c8 ▹ ▹ val fl = 0;
6abd ▹ ▹ while (bfs(s, t)) for(val f; (f = dfs(s, t, numeric_limits<val>::max())); fl += f);
850b ▹ ▹ return fl;
a1c8 ▹ }
a1c8 ▹ // public $
8662 ▹ num min_cost(int s, int t) {
ea9f ▹ ▹ flow = max_flow(s, t);
a931 ▹ ▹ while(spfa());
fabb ▹ ▹ tot = 0;
2896 ▹ ▹ for(int i = 2; i < en; i++)
46d1 ▹ ▹ ▹ if(fl[i] > 0)
3625 ▹ ▹ ▹ ▹ tot += fl[i] * cs[i];
48fb ▹ ▹ return tot;
776d ▹ }
5ce3 ▹ void reset_flow() { memset(fl, 0, sizeof(val) * en); }
21e1 ▹ void add_edge(int u, int v, val c, num s) {
b047 ▹ ▹ fl[en] = 0; cp[en] = c; to[en] = v; nx[en] = hd[u]; cs[en] = s; hd[u] = en++;
142f ▹ ▹ fl[en] = 0; cp[en] = 0; to[en] = u; nx[en] = hd[v]; cs[en] = -s; hd[v] = en++;
afa6 ▹ }
ff5d ▹ void init(int n) { this->n = n; en = 2; memset(hd, 0, sizeof(int) * n); } // XXX must be called
8e87 };
```

2.4 Hungarian

```
d41d //const int N = ; typedef ll num; const num eps = ;
```

```
d41d // Solves minimum perfect matching in an n by n bipartite graph with edge costs in c
d41d // y and z will be such that y[i] + z[j] <= c[i][j] and sum of y and z is maximum
55ad struct hungarian {
dc2a ▹ int n, MA[N], MB[N], PB[N], mn[N], st[N], sn; bool S[N], T[N];
6ce0 ▹ num c[N][N], d[N], y[N], z[N];
196f ▹ bool increase(int b) {
31f4 ▹ ▹ for (int a = PB[b];;) {
6a28 ▹ ▹ ▹ int n_b = MA[a];
1cfa ▹ ▹ ▹ MB[b] = a; MA[a] = b;
688f ▹ ▹ ▹ if(n_b == -1) break;
f76a ▹ ▹ ▹ b = n_b; a = PB[b];
4f2b ▹ ▹ }
a67a ▹ ▹ return true;
c5e8 ▹ }
2078 ▹ bool visit(int a) {
4a81 ▹ ▹ S[a] = true;
15b8 ▹ ▹ for(int b = 0; b < n; b++) {
f511 ▹ ▹ ▹ if(T[b]) continue;
82fc ▹ ▹ ▹ if(c[a][b] - y[a] - z[b] < d[b] - eps) { d[b] = c[a][b] - y[a] - z[b]; mn[b] = a; }
4d01 ▹ ▹ ▹ if(c[a][b] - eps <= y[a] + z[b]) {
b5b2 ▹ ▹ ▹ ▹ T[b] = true; PB[b] = a; st[sn++] = b;
d070 ▹ ▹ ▹ ▹ if(MB[b] == -1) return increase(b);
8149 ▹ ▹ ▹ }
4153 ▹ ▹ }
7dda ▹ ▹ return false;
289e ▹ }
1eb5 ▹ bool update_dual() {
1710 ▹ ▹ int mb = -1, b; num e;
9d8d ▹ ▹ for(b = 0; b < n; b++) if(!T[b] && (mb == -1 || d[b] < d[mb])) mb = b;
7267 ▹ ▹ for(e = d[mb], b = 0; b < n; b++)
91c1 ▹ ▹ ▹ if(T[b]) z[b] -= e;
3b28 ▹ ▹ ▹ else d[b] -= e;
f5f6 ▹ ▹ for(int a = 0; a < n; a++)
f6e6 ▹ ▹ ▹ if(S[a]) y[a] += e;
1817 ▹ ▹ PB[mb] = mn[mb];
7397 ▹ ▹ if(MB[mb] == -1) return increase(mb);
f789 ▹ ▹ st[sn++] = mb; T[mb] = true;
637d ▹ ▹ return false;
e26f ▹ }
3557 ▹ void find_path() {
da0c ▹ ▹ int a; for(a = 0; MA[a] != -1; a++);
eb0d ▹ ▹ memset(S, 0, sizeof S); memset(T, 0, sizeof T);
469d ▹ ▹ for(int i = 0; i < N; i++) d[i] = numeric_limits<num>::max();
bfae ▹ ▹ sn = 0; if(visit(a)) return;
7b65 ▹ ▹ while(true) {
67db ▹ ▹ ▹ if(sn) { if(visit(MB[st[--sn]])) break; }
d1da ▹ ▹ ▹ else if(update_dual()) break;
a220 ▹ ▹ }
bd96 ▹ }
a439 ▹ void reset_all() {
0e20 ▹ ▹ for(int i = 0; i < n; i++) { y[i] = *min_element(c[i], c[i] + n); z[i] = 0; }
b269 ▹ ▹ for(int i = 0; i < n; i++) MA[i] = MB[i] = -1;
023e ▹ }
023e ▹ // public $
957f ▹ num min_match() { // set n and c then call this function
5591 ▹ ▹ reset_all(); num all = 0;
1061 ▹ ▹ for(int i = 0; i < n; i++) find_path();
459d ▹ ▹ for(int a = 0; a < n; a++) all += c[a][MA[a]];
1b77 ▹ ▹ return all;
bb00 ▹ }
9f6a };
```

3 Structures

3.1 Ordered Set

```
7747 #include <ext/pb_ds/assoc_container.hpp>
0702 #include <ext/pb_ds/tree_policy.hpp>
```

```
9969 using namespace __gnu_pbds;
b9a9 template <typename tA, typename tB=null_type> using ord_set = tree<tA, tB, less<tA>, rb_tree_tag,
    tree_order_statistics_node_update>;
b9a9 // map: tA -> tB com comparador less<tA>
b9a9 // pode usar como um map normalmente
b9a9 // s.find_by_order(k) :: retorna iterador para o k-esimo elemento (0-index) (ou s.end())
b9a9 // s.order_of_key(x) :: retorna a qtd de elementos estritamente menores que x
```

3.2 Treap

```
d41d //const int N = ; typedef int num;
5463 num X[N]; int en = 1, Y[N], sz[N], L[N], R[N];
56f4 void calc (int u) { // update node given children info
0807 ▹ sz[u] = sz[L[u]] + 1 + sz[R[u]];
0807 ▹ // code here, no recursion
c13f }
abed void unlaze (int u) {
ff04 ▹ if(!u) return;
ff04 ▹ // code here, no recursion
e17d }
1422 void split_val(int u, num x, int &l, int &r) { // l gets <= x, r gets > x
38dc ▹ unlaze(u); if(!u) return (void) (l = r = 0);
a614 ▹ if(X[u] <= x) { split_val(R[u], x, l, r); R[u] = l; l = u; }
966c ▹ else { split_val(L[u], x, l, r); L[u] = r; r = u; }
accd ▹ calc(u);
1524 }
1808 void split_sz(int u, int s, int &l, int &r) { // l gets first s, r gets remaining
bd76 ▹ unlaze(u); if(!u) return (void) (l = r = 0);
9ab9 ▹ if(sz[L[u]] < s) { split_sz(R[u], s - sz[L[u]] - 1, l, r); R[u] = l; l = u; }
0e9f ▹ else { split_sz(L[u], s, l, r); L[u] = r; r = u; }
dedb ▹ calc(u);
3419 }
a655 int merge(int l, int r) { // els on l <= els on r
0fc5 ▹ unlaze(l); unlaze(r); if(!l || !r) return l + r; int u;
72c6 ▹ if(Y[l] > Y[r]) { R[l] = merge(R[l], r); u = l; }
834a ▹ else { L[r] = merge(l, L[r]); u = r; }
295a ▹ calc(u); return u;
8772 }
ff63 void init(int n=N-1) { // XXX call before using other funcs
6d10 ▹ for(int i = en = 1; i <= n; i++) { Y[i] = i; sz[i] = 1; L[i] = R[i] = 0; }
2bd3 ▹ random_shuffle(Y + 1, Y + n + 1);
7d26 }
```

3.3 Envelope

```
d41d // typedef ll num; const num eps = 0;
d41d // XXX double: indicates operations specific to integers, not precision related
d79f template<typename line> struct envelope {
1a27 ▹ deque<line> q; num lo,hi; envelope (num _lo, num _hi) : lo(_lo), hi(_hi) {}
14d6 ▹ void push_front (line l) { // amort. O(inter) | l is best at lo or never
ba5b ▹ ▹ if (q.size() && q[0](lo) < l(lo)) return;
52c8 ▹ ▹ for (num x; q.size(); q.pop_front()) {
1f0a ▹ ▹ ▹ x = (q.size()<=1?hi:q[0].inter(q[1],lo,hi)-1); // XXX double (-1)
a656 ▹ ▹ ▹ if (l(x) > q[0](x)) break;
5f7f ▹ ▹ }
9d65 ▹ ▹ q.push_front(l);
92ce ▹ }
9132 ▹ void push_back (line l) { // amort. O(inter) | l is best at hi or never
0e36 ▹ ▹ if (q.size() && q[q.size()-1](hi) <= l(hi)) return;
f8c9 ▹ ▹ for (num x; q.size(); q.pop_back()) {
0e00 ▹ ▹ ▹ x = (q.size()<=1?lo:q[q.size()-2].inter(q[q.size()-1],lo,hi));
9314 ▹ ▹ ▹ if (l(x) >= q[q.size()-1](x)) break;
3e5a ▹ ▹ }
737f ▹ ▹ q.push_back(l);
52f9 ▹ }
d569 ▹ void pop_front (num _lo) { for (lo=_lo; q.size()>1 && q[0](lo) > q[1](lo); q.pop_front()); } // amort.
    O(n)
abdb ▹ void pop_back (num _hi) { for (hi=_hi; q.size()>1 && q[q.size()-2](hi) <= q[q.size()-1](hi);
    q.pop_back()); } // amort. O(n)
1eb0 ▹ line get (num x) { // O(lg(R))
0f0b ▹ ▹ int lo, hi, md; for (lo = 0, hi = q.size()-1, md = (lo+hi)/2; lo < hi; md = (lo+hi)/2)
05f2 ▹ ▹ ▹ if (q[md](x) > q[md+1](x)) { lo = md+1; }
e930 ▹ ▹ ▹ else { hi = md; }
463a ▹ ▹ return q[lo];
e806 ▹ }
4e77 };
b770 struct line { // inter = O(1)
7e6b ▹ num a,b; num operator () (num x) const { return a*x+b; }
966b ▹ num inter (line o, num lo, num hi) { return
    abs(o.a-a)<=eps?((b<o.b)?hi+1:lo):min(hi+1,max(lo,(o.b-b-(o.b-b<0)*(a-o.a-1))/(a-o.a) + 1)); }
e972 };
d59b struct generic_line { // inter = O(lg(R))
1ff6 ▹ num a,b; num operator () (num x) const { return a*x+b; }
96e0 ▹ num inter (generic_line o, num lo, num hi) { // first point where o strictly beats this
1431 ▹ ▹ for (num md = lo+((++hi)-lo)/2; lo < hi; md = lo+(hi-lo)/2) { // XXX double
e0c5 ▹ ▹ ▹ if ((*this)(md)<=o(md)) { lo = md+1; } // XXX double
1388 ▹ ▹ ▹ else { hi = md; }
00af ▹ ▹ }
a762 ▹ ▹ return lo;
7348 ▹ }
ae48 };
522c template<typename line> struct full_envelope { // XXX ties are broken arbitrarily
f3ef ▹ vector<envelope<line> > v; full_envelope(envelope<line> c) : v({c}) {} // v.reserve(30);
a356 ▹ void add (line l) { // amort. O(lg(n)*inter)
8448 ▹ ▹ envelope<line> cur(v.back().lo,v.back().hi); cur.push_back(l);
48c7 ▹ ▹ while (!v.empty() && v.back().q.size() <= cur.q.size()) {
1787 ▹ ▹ ▹ deque<line> aux; swap(aux,cur.q); int i = 0, j = 0;
68b1 ▹ ▹ ▹ for (; i < aux.size(); i++) {
3c24 ▹ ▹ ▹ ▹ for (; j < v.back().q.size() && v.back().q[j](cur.hi) > aux[i](cur.hi); j++)
9c65 ▹ ▹ ▹ ▹ ▹ cur.push_back(v.back().q[j]);
f48b ▹ ▹ ▹ ▹ cur.push_back(aux[i]);
af00 ▹ ▹ ▹ }
322f ▹ ▹ ▹ for (; j < v.back().q.size(); j++) cur.push_back(v.back().q[j]);
f4fc ▹ ▹ ▹ v.pop_back();
59ed ▹ ▹ }
888a ▹ ▹ v.push_back(cur);
9701 ▹ }
0fa9 ▹ line get (num x) { // O(lg(n)lg(R)) | pop_back/pop_front can optimize
dcb8 ▹ ▹ line a = v[0].get(x);
291d ▹ ▹ for (int i = 1; i < (int) v.size(); i++) {
9a87 ▹ ▹ ▹ line b = v[i].get(x);
a55f ▹ ▹ ▹ if (b(x)<a(x)) a = b;
bfb8 ▹ ▹ }
dec3 ▹ ▹ return a;
dfea ▹ }
79bc };
```

3.4 Centroid

```
0eca vector<int> adj[N]; int cn_sz[N], n;
526e vector<int> cn_chld[N]; int cn_dep[N], cn_dist[20][N]; // removable
100f void cn_setdist (int u, int p, int depth, int dist) { // removable
43aa ▹ cn_dist[depth][u] = dist;
c47d ▹ for (int v : adj[u]) if (p != v && cn_sz[v] != -1) // sz = -1 marks processed centroid (not dominated)
9376 ▹ ▹ cn_setdist(v, u, depth, dist+1);
d78e }
7066 int cn_getsz (int u, int p) {
6ca8 ▹ cn_sz[u] = 1;
414b ▹ for (int v : adj[u]) if (p != v && cn_sz[v] != -1)
76bf ▹ ▹ cn_sz[u] += cn_getsz(v,u);
c1a2 ▹ return cn_sz[u];
dd96 }
cc54 int cn_build (int u, int depth) {
3d11 ▹ int siz = cn_getsz(u,u); int w = u;
d3a8 ▹ do {
```

```
de25 ▹ ▹ u = w;
fa35 ▹ ▹ for (int v : adj[u]) if (cn_sz[v] != -1 && cn_sz[v] < cn_sz[u] && cn_sz[v] + cn_sz[v] >= siz)
1568 ▹ ▹ ▹ w = v;
83e1 ▹ } while (u != w); // u becomes current centroid root
ba98 ▹ cn_setdist(u,u,depth,0); // removable, here you can iterate over all dominated tree
f972 ▹ cn_sz[u] = -1; cn_dep[u] = depth;
da49 ▹ for (int v : adj[u]) if (cn_sz[v] != -1) {
a15a ▹ ▹ int w = cn_build(v, depth+1);
f456 ▹ ▹ cn_chld[u].pb(w); // removable
5009 ▹ }
1bf7 ▹ return u;
ec99 }
```

3.5 Link Cut Tree

```
d41d //const int N = ; typedef int num;
8db1 int en = 1, p[N], sz[N], pp[N]; bool lzswp[N];
8424 int C[N][2]; // {left, right} children
e2ac inline void calc(int u) { // update node given children info
25fc ▹ sz[u] = sz[C[u][0]] + 1 + sz[C[u][1]];
25fc ▹ // code here, no recursion
1109 }
dd21 inline void unlaze(int u) {
798a ▹ if(!u) return;
1c71 ▹ if(lzswp[u]) {
f8fc ▹ ▹ swap(C[u][0], C[u][1]);
7afd ▹ ▹ if(C[u][0]) lzswp[C[u][0]] ^= 1;
b9bb ▹ ▹ if(C[u][1]) lzswp[C[u][1]] ^= 1;
72da ▹ ▹ lzswp[u] = 0;
80e0 ▹ }
d1c4 }
1c50 int rotate(int u, int dir) { // pulls C[u][dir] up to u and returns it
db32 ▹ int v = C[u][dir];
6a10 ▹ swap(pp[v], pp[u]);
e106 ▹ C[u][dir] = C[v][!dir];
eb5a ▹ if(C[u][dir]) p[C[u][dir]] = u;
93fa ▹ C[v][!dir] = u; p[v] = p[u];
7926 ▹ if(p[v]) C[p[v]][C[p[v]][1] == u] = v;
49c3 ▹ p[u] = v; calc(u); calc(v);
eca8 ▹ return v;
72a2 }
e98d void unlz_back(int u) { if(!u) return; unlz_back(p[u]); unlaze(u); }
21be void splay(int u) { // pulls node u to root
d6a2 ▹ unlz_back(u);
9f2a ▹ while(p[u]) {
11b2 ▹ ▹ int v = p[u], w = p[p[u]];
a646 ▹ ▹ int du = (C[v][1] == u);
8d24 ▹ ▹ if(!w) { rotate(v, du); assert(!p[u]); }
1df2 ▹ ▹ else {
852a ▹ ▹ ▹ int dv = (C[w][1] == v);
bb73 ▹ ▹ ▹ if(du == dv) { rotate(w, dv); assert(C[v][du] == u); rotate(v, du); }
b1a7 ▹ ▹ ▹ else { rotate(v, du); assert(C[w][dv] == u); rotate(w, dv); }
c672 ▹ ▹ }
33c4 ▹ }
1ca7 }
89ef int find_sz(int u, int s) { // returns s-th node (0-index)
3a1d ▹ unlaze(u);
79a4 ▹ while(sz[C[u][0]] != s) {
afec ▹ ▹ if(sz[C[u][0]] < s) { s -= sz[C[u][0]] + 1; u = C[u][1]; }
48fe ▹ ▹ else u = C[u][0];
11ef ▹ ▹ unlaze(u);
a09d ▹ }
fb1c ▹ splay(u); return u;
cd18 }
13c3 int new_node() {
1b91 ▹ int i = en++; assert(i < N);
7dc1 ▹ pp[i] = C[i][0] = C[i][1] = p[i] = 0;
81cf ▹ lzswp[i] = 0; sz[i] = 1; return i;
8f65 }
9e4d int access(int u) {
d1ad ▷ if(!u) return u;
a275 ▷ splay(u);
2fca ▷ if(int v = C[u][1]) { p[v] = 0; pp[v] = u; C[u][1] = 0; }
308e ▷ calc(u);
7698 ▷ while(pp[u]) {
ad32 ▷ ▷ int w = pp[u]; splay(w);
6390 ▷ ▷ if(int v = C[w][1]) { p[v] = 0; pp[v] = w; }
b36d ▷ ▷ C[w][1] = u; p[u] = w; pp[u] = 0; calc(w); splay(u);
52c1 ▷ }
2425 ▷ return u;
8eab }
a948 int find_root(int u) { // root o u's tree
6b08 ▷ access(u);
0c6b ▷ while(C[u][0]) { unlaze(u = C[u][0]); }
6f5e ▷ access(u); return u;
345c }
cfa2 int get_parent(int u) { // u's parent, rootify might change it
1492 ▷ access(u);
8f02 ▷ if(!C[u][0]) return pp[u];
60bb ▷ unlaze(u = C[u][0]);
64fa ▷ while(C[u][1]) unlaze(u = C[u][1]);
c6ef ▷ access(u); return u;
ea38 }
7ae4 void link(int u, int v) { // adds edge from u to v, v must be root
ae22 ▷ if(find_root(u) == find_root(v)) return;
52d3 ▷ access(u); access(v);
bdf4 ▷ assert(C[v][0] == 0 && pp[v] == 0 && sz[v] == 1); // v must be root
b77b ▷ C[u][1] = v; p[v] = u; calc(u);
82b3 }
82b3 // XXX cut + rootify require get_parent, cut unlinks u from parent, rootify makes u root
4cfe void cut(int u) { access(u); assert(C[u][0]); p[C[u][0]] = 0; C[u][0] = 0; calc(u); }
12fd void rootify(int u) { access(u); lzswp[u] = 1; access(u); }
7992 void init() { en = 1; } // XXX initialize
```

3.6 Splay Tree

```
d41d //const int N = ;
d41d //typedef int num;
d41d
576f int en = 1;
af8d int p[N], sz[N];
ce7e int C[N][2]; // {left, right} children
f778 num X[N];
f778
f778 // atualize os valores associados aos nos que podem ser calculados a partir dos filhos
be20 void calc(int u) {
842c ▷ sz[u] = sz[C[u][0]] + 1 + sz[C[u][1]];
7bd0 }
7bd0
7bd0 // Puxa o filho dir de u para ficar em sua posicao e o retorna
b067 int rotate(int u, int dir) {
caea ▷ int v = C[u][dir];
414f ▷ C[u][dir] = C[v][!dir];
3e86 ▷ if(C[u][dir]) p[C[u][dir]] = u;
9ee5 ▷ C[v][!dir] = u;
4535 ▷ p[v] = p[u];
d1e0 ▷ if(p[v]) C[p[v]][C[p[v]][1] == u] = v;
6f9a ▷ p[u] = v;
cacf ▷ calc(u);
72d9 ▷ calc(v);
f3dd ▷ return v;
f0a5 }
f0a5
f0a5 // Traz o no u a raiz
ab99 void splay(int u) {
9867 ▷ while(p[u]) {
```

```
e330 ▹ ▹ int v = p[u], w = p[p[u]];
e7b1 ▹ ▹ int du = C[v][1] == u;
ffbb ▹ ▹ if(!w)
75c5 ▹ ▹ ▹ rotate(v, du);
daae ▹ ▹ else {
7d5e ▹ ▹ ▹ int dv = (C[w][1] == v);
b213 ▹ ▹ ▹ if(du == dv) {
a184 ▹ ▹ ▹ ▹ rotate(w, dv);
baa9 ▹ ▹ ▹ ▹ rotate(v, du);
7f27 ▹ ▹ ▹ } else {
5414 ▹ ▹ ▹ ▹ rotate(v, du);
058f ▹ ▹ ▹ ▹ rotate(w, dv);
e8ea ▹ ▹ ▹ }
d76f ▹ ▹ }
1864 ▹ }
5d3d }
5d3d
5d3d // retorna um no com valor x, ou outro no se n foi encontrado (n eh floor nem ceiling)
51ad int find_val(int u, num x) {
f645 ▹ int v = u;
077b ▹ while(u && X[u] != x) {
a87c ▹ ▹ v = u;
be11 ▹ ▹ if(x < X[u]) u = C[u][0];
2d7a ▹ ▹ else u = C[u][1];
ce7d ▹ }
b518 ▹ if(!u) u = v;
3571 ▹ splay(u);
76c5 ▹ return u;
dbd9 }
dbd9
dbd9 // retorna o s-esimo no (0-indexed)
bdd6 int find_sz(int u, int s) {
7b32 ▹ while(sz[C[u][0]] != s) {
5583 ▹ ▹ if(sz[C[u][0]] < s) {
369e ▹ ▹ ▹ s -= sz[C[u][0]] + 1;
b7c9 ▹ ▹ ▹ u = C[u][1];
7235 ▹ ▹ } else u = C[u][0];
d8f8 ▹ }
64fb ▹ splay(u);
ff92 ▹ return u;
2c2c }
2c2c
2c2c // junte duas splays, assume que elementos l <= elementos r
8987 int merge(int l, int r) {
93fa ▹ if(!l || !r) return l + r;
0d23 ▹ while(C[l][1]) l = C[l][1];
d8b7 ▹ splay(l);
50b9 ▹ assert(!C[l][1]);
091d ▹ C[l][1] = r;
aa15 ▹ p[r] = l;
841d ▹ calc(l);
6afb ▹ return l;
38a2 }
38a2
38a2 // Adiciona no x a splay u e retorna x
db32 int add(int u, int x) {
31a8 ▹ int v = 0;
37fb ▹ while(u) v = u, u = C[u][X[x] >= X[u]];
f035 ▹ if(v) { C[v][X[x] >= X[v]] = x; p[x] = v; }
b54d ▹ splay(x);
b185 ▹ return x;
1a16 }
1a16
1a16 // chame isso 1 vez no inicio
0240 void init() {
a02a ▹ en = 1;
196f }
196f
196f // Cria um novo no
```

```
b218 int new_node(num val) {
119b ▹ int i = en++;
a656 ▹ assert(i < N);
0fed ▹ C[i][0] = C[i][1] = p[i] = 0;
d691 ▹ sz[i] = 1;
cafa ▹ X[i] = val;
1ce7 ▹ return i;
30e1 }
```

4 Strings

4.1 Suffix Tree

```
4623 namespace sf {
4623 // const int NS = ; const int N = * 2;
9635 int cn, cd, ns, en = 1, lst;
9291 string S[NS]; int si = -1;
64ca vector<int> sufn[N]; // sufn[si][i] no do sufixo S[si][i...]
e9c3 struct node {
fb38 ▹ int l, r, si, p, suf;
98b3 ▹ map<char, int> adj;
5edc ▹ node() : l(0), r(-1), suf(0), p(0) {}
b72f ▹ node(int L, int R, int S, int P) : l(L), r(R), si(S), p(P) {}
997f ▹ inline int len() { return r - l + 1; }
fe05 ▹ inline int operator[](int i) { return S[si][l + i]; }
d9e6 ▹ inline int& operator()(char c) { return adj[c]; }
fcde } t[N];
174b inline int new_node(int L, int R, int S, int P) { t[en] = node(L, R, S, P); return en++; }
a45b void add_string(string s) {
74e6 ▹ s += '$'; S[++si] = s; sufn[si].resize(s.size() + 1); cn = cd = 0;
8a0c ▹ int i = 0; const int n = s.size();
8d8f ▹ for(int j = 0; j < n; j++)
7613 ▹ ▹ for(; i <= j; i++) {
bf53 ▹ ▹ ▹ if(cd == t[cn].len() && t[cn](s[j])) { cn = t[cn](s[j]); cd = 0; }
f05f ▹ ▹ ▹ if(cd < t[cn].len() && t[cn][cd] == s[j]) {
1dd6 ▹ ▹ ▹ ▹ cd++;
7fd5 ▹ ▹ ▹ ▹ if(j < s.size() - 1) break;
90fc ▹ ▹ ▹ ▹ else {
a8ed ▹ ▹ ▹ ▹ ▹ if(i) t[lst].suf = cn;
b914 ▹ ▹ ▹ ▹ ▹ for(; i <= j; i++) { sufn[si][i] = cn; cn = t[cn].suf; }
0f76 ▹ ▹ ▹ ▹ }
b338 ▹ ▹ ▹ } else if(cd == t[cn].len()) {
f90b ▹ ▹ ▹ ▹ sufn[si][i] = en;
5483 ▹ ▹ ▹ ▹ if(i) t[lst].suf = en; lst = en;
872f ▹ ▹ ▹ ▹ t[cn](s[j]) = new_node(j, n - 1, si, cn);
0499 ▹ ▹ ▹ ▹ cn = t[cn].suf; cd = t[cn].len();
56f0 ▹ ▹ ▹ } else {
ac3e ▹ ▹ ▹ ▹ int mid = new_node(t[cn].l, t[cn].l + cd - 1, t[cn].si, t[cn].p);
9372 ▹ ▹ ▹ ▹ t[t[cn].p](t[cn][0]) = mid;
e5dd ▹ ▹ ▹ ▹ if(ns) t[ns].suf = mid;
fce9 ▹ ▹ ▹ ▹ if(i) t[lst].suf = en; lst = en;
58b7 ▹ ▹ ▹ ▹ sufn[si][i] = en;
87a2 ▹ ▹ ▹ ▹ t[mid](s[j]) = new_node(j, n - 1, si, mid);
fed5 ▹ ▹ ▹ ▹ t[mid](t[cn][cd]) = cn;
7846 ▹ ▹ ▹ ▹ t[cn].p = mid; t[cn].l += cd; cn = t[mid].p;
da79 ▹ ▹ ▹ ▹ int g = cn? j - cd : i + 1; cn = t[cn].suf;
5116 ▹ ▹ ▹ ▹ while(g < j && g + t[t[cn](S[si][g])].len() <= j) {
d170 ▹ ▹ ▹ ▹ ▹ cn = t[cn](S[si][g]); g += t[cn].len();
5ba7 ▹ ▹ ▹ ▹ }
3819 ▹ ▹ ▹ ▹ if(g == j) { ns = 0; t[mid].suf = cn; cd = t[cn].len(); }
ccde ▹ ▹ ▹ ▹ else { ns = mid; cn = t[cn](S[si][g]); cd = j - g; }
839f ▹ ▹ ▹ }
f189 ▹ ▹ }
aea5 ▹ }
30c2 };
```

4.2 Z-function

```
2a61 void Z(char s[], int n, int z[]) { // z[i] = |lcp(s,s[i..n])|
24d4 ▹ for(int i = 1, m = -1; i < n; i++) {
d138 ▹ ▹ z[i] = (m != -1 && m + z[m] >= i)?min(m + z[m] - i, z[i - m]):0;
171a ▹ ▹ while (i + z[i] < n && s[i + z[i]] == s[z[i]]) z[i]++;
7021 ▹ ▹ if (m == -1 || i + z[i] > m + z[m]) m = i;
9eed ▹ }
ea97 }
```

4.3 Manacher

```
d41d // max odd pali cent on i: s[i - M[2 * i] / 2..i + M[2 * i] / 2]
d41d // max even pali cent on [i,i+1]: s[i + 1 - (M[2*i+1] + 1) / 2..i + (M[2*i+1] + 1) / 2] (if M[2*i+1] != 0)
2a2b void manacher(char s[], int n, char t[], int M[]) { // t and M should have size 2*n
3035 ▹ for(int i = 0; i < n; i++) t[2 * i] = s[i];
297d ▹ for(int i = 0; i < n - 1; i++) t[2 * i + 1] = 1; // XXX s should not contain 1
48ed ▹ n = 2 * n - 1;
13c6 ▹ for(int i = 0, m = -1; i < n; i++) {
af7b ▹ ▹ M[i] = 0;
0a95 ▹ ▹ if (m != -1 && m + M[m] >= i) M[i] = min(m + M[m] - i, M[2 * m - i]);
5af1 ▹ ▹ for (; i + M[i] + 1 < n && i - M[i] - 1 >= 0 && t[i + M[i] + 1] == t[i - M[i] - 1]; M[i]++);
0b2a ▹ ▹ if (m == -1 || i + M[i] > m + M[m]) m = i;
017a ▹ }
a03f }
```

5 Math

5.1 FFT

```
5f83 typedef complex<double> cpx; const double pi = acos(-1.0);
5f83 // DFT if type = 1, IDFT if type = -1
5f83 // If you are multiplying, remember to let EACH vector with n >= sum of degrees of both polys
5f83 // n is required to be a power of 2
e4be void FFT(cpx v[], cpx ans[], int n, int type, int p[]) { // p[n]
0679 ▹ assert(!(n & (n - 1))); int i, sz, o; p[0] = 0;
2ec6 ▹ for(i = 1; i < n; i++) p[i] = (p[i >> 1] >> 1) | ((i & 1)? (n >> 1) : 0); // repetition can be avoided
92f5 ▹ for(i = 0; i < n; i++) ans[i] = v[p[i]];
fe06 ▹ for(sz = 1; sz < n; sz <<= 1) {
4caa ▹ ▹ const cpx wn(cos(type * pi / sz), sin(type * pi / sz));
490e ▹ ▹ for(o = 0; o < n; o += (sz << 1)) {
1381 ▹ ▹ ▹ cpx w = 1;
841c ▹ ▹ ▹ for(i = 0; i < sz; i++) {
e92b ▹ ▹ ▹ ▹ const cpx u = ans[o + i], t = w * ans[o + sz + i];
ec53 ▹ ▹ ▹ ▹ ans[o + i] = u + t;
3b83 ▹ ▹ ▹ ▹ ans[o + i + sz] = u - t;
d519 ▹ ▹ ▹ ▹ w *= wn;
54ec ▹ ▹ ▹ }
2972 ▹ ▹ }
9541 ▹ }
eb52 ▹ if(type == -1) for(i = 0; i < n; i++) ans[i] /= n;
d336 }
```

5.2 Discrete FFT

```
c9bc inline num s_mod (ll x, ll p) {
cbc9 ▹ if (x >= p) return x-p;
c6fd ▹ else if (x < 0) return x += p;
fc49 ▹ return x;
d655 }
e402 num fexp (ll x, int e, num p) {
ed8f ▹ ll r = 1;
1bf1 ▹ for (; e; x = (x*x)%p, e >>= 1) if (e&1) r = (r*x)%p;
e879 ▹ return r;
9c45 }
f727 void rou (int n, int p, num w[]) { // w[i] = (n-th root of unity of p)^i
061a ▹ w[0] = 1; bool ok = 0;
c7eb ▹ for (num i = 2; !ok && i < p; i++) {
0b59 ▹ ▹ ok = 1;
124e ▹ ▹ for (ll j = 2; ok && j*j <= p-1; j++)
c86f ▹ ▹ ▹ if ((p-1)%j == 0)
9f65 ▹ ▹ ▹ ▹ ok = !(fexp(i,j,p) == 1 || fexp(i,(p-1)/j,p) == 1);
c069 ▹ ▹ if (ok) w[1] = fexp(i,(p-1)/n,p);
26e5 ▹ }
322d ▹ assert(ok);
5041 ▹ for (int i = 2; i <= n; i++)
cd82 ▹ ▹ w[i] = (ll(w[i-1])*w[1])%p;
bd4f }
5978 void fft_finite (num v[], num ans[], int n, int type, num p, int pr[], num w[]) { // pr[n], w[n]
d4a7 ▹ assert(!(n & (n-1)));
199d ▹ rou(n,p,w); ll invn = fexp(n,p-2,p); // repetition can be avoided
9855 ▹ if (type == -1) reverse(w, w+n+1);
3d96 ▹ pr[0] = 0;
275b ▹ for (int i = 1; i < n; i++) pr[i] = ((pr[i>>1] >> 1) | ((i&1)?(n>>1):0)); // repetition can be avoided
099c ▹ for (int i = 0; i < n; i++) ans[i] = v[pr[i]];
c9dd ▹ for (int sz = 1; sz < n; sz <<= 1) {
68a7 ▹ ▹ for (int o = 0; o < n; o += (sz<<1)) {
60fb ▹ ▹ ▹ for (int i = 0; i < sz; i++) {
6b17 ▹ ▹ ▹ ▹ const num u = ans[o+i], t = (w[(n/sz/2)*i]*ans[o+sz+i])%p;
e1ee ▹ ▹ ▹ ▹ ans[o+i] = s_mod(u+t,p);
005a ▹ ▹ ▹ ▹ ans[o+i+sz] = s_mod(u-t,p);
aa10 ▹ ▹ ▹ }
1184 ▹ ▹ }
e419 ▹ }
7400 ▹ if(type == -1) for(int i = 0; i < n; i++) ans[i] = (ans[i]*invn)%p;
39d1 }
39d1
```

5.3 Linear System Solver

```
d41d //const int N = ;
d41d
46cc double a[N][N];
686b double ans[N];
686b
686b // sum(a[i][j] * x_j) = a[i][n] para 0 <= i < n
686b // guarda a resposta em ans e retorna o determinante de a
ab71 double solve(int n) {
0eb7 ▹ double det = 1;
03e6 ▹ for(int i = 0; i < n; i++) {
fe06 ▹ ▹ int mx = i;
0cc0 ▹ ▹ for(int j = i + 1; j < n; j++)
e71d ▹ ▹ ▹ if(abs(a[j][i]) > abs(a[mx][i]))
b49a ▹ ▹ ▹ ▹ mx = j;
2853 ▹ ▹ if(i != mx) {
4775 ▹ ▹ ▹ swap_ranges(a[i], a[i] + n + 1, a[mx]);
0289 ▹ ▹ ▹ det = -det;
e4b3 ▹ ▹ }
1104 ▹ ▹ if(abs(a[i][i]) < 1e-6); // singular matrix
07d4 ▹ ▹ det *= a[i][i];
badf ▹ ▹ for(int j = i + 1; j < n; j++) {
1b15 ▹ ▹ ▹ for(int k = i + 1; k <= n; k++)
65e2 ▹ ▹ ▹ ▹ a[j][k] -= (a[j][i] / a[i][i]) * a[i][k];
90b0 ▹ ▹ ▹ a[j][i] = 0;
0fb3 ▹ ▹ }
674e ▹ }
e3e5 ▹ for(int i = n - 1; i >= 0; i--) {
89c0 ▹ ▹ ans[i] = a[i][n];
8f17 ▹ ▹ for(int j = i + 1; j < n; j++)
3594 ▹ ▹ ▹ ans[i] -= a[i][j] * ans[j];
1416 ▹ ▹ ans[i] /= a[i][i];
6890 ▹ }
1b75 ▹ return det;
285a }
```

5.4 Simplex

```
d41d //typedef long double dbl;
bec0 const dbl eps = 1e-6;
bec0 //const int N = , M = ;
bec0
2c35 struct simplex {
57af ▹ int X[N], Y[M];
5c3e ▹ dbl A[M][N], b[M], c[N];
2cac ▹ dbl ans;
9021 ▹ int n, m;
1b1b ▹ dbl sol[N];
1b1b
501a ▹ void pivot(int x,int y){
25f5 ▹ ▹ swap(X[y], Y[x]);
515a ▹ ▹ b[x] /= A[x][y];
1507 ▹ ▹ for(int i = 0; i < n; i++)
47a3 ▹ ▹ ▹ if(i != y)
6add ▹ ▹ ▹ ▹ A[x][i] /= A[x][y];
3670 ▹ ▹ A[x][y] = 1. / A[x][y];
1208 ▹ ▹ for(int i = 0; i < m; i++)
1284 ▹ ▹ ▹ if(i != x && abs(A[i][y]) > eps) {
d094 ▹ ▹ ▹ ▹ b[i] -= A[i][y] * b[x];
0223 ▹ ▹ ▹ ▹ for(int j = 0; j < n; j++)
d2c4 ▹ ▹ ▹ ▹ ▹ if(j != y)
34c9 ▹ ▹ ▹ ▹ ▹ ▹ A[i][j] -= A[i][y] * A[x][j];
0b1a ▹ ▹ ▹ ▹ A[i][y] = -A[i][y] * A[x][y];
b6a8 ▹ ▹ ▹ }
89a6 ▹ ▹ ans += c[y] * b[x];
bfd7 ▹ ▹ for(int i = 0; i < n; i++)
27b7 ▹ ▹ ▹ if(i != y)
1121 ▹ ▹ ▹ ▹ c[i] -= c[y] * A[x][i];
1d1c ▹ ▹ c[y] = -c[y] * A[x][y];
ba5d ▹ }
ba5d
ba5d ▹ // maximiza sum(x[i] * c[i])
ba5d ▹ // sujeito a
ba5d ▹ //  sum(a[i][j] * x[j]) <= b[i] para 0 <= i < m (Ax <= b)
ba5d ▹ //  x[i] >= 0 para 0 <= i < n (x >= 0)
ba5d ▹ // (n variaveis, m restricoes)
ba5d ▹ // guarda a resposta em ans e retorna o valor otimo
8c98 ▹ dbl solve(int n, int m) {
df25 ▹ ▹ this->n = n; this->m = m;
b879 ▹ ▹ ans = 0.;
32f5 ▹ ▹ for(int i = 0; i < n; i++) X[i] = i;
42a0 ▹ ▹ for(int i = 0; i < m; i++) Y[i] = i + n;
f798 ▹ ▹ while(true) {
25cf ▹ ▹ ▹ int x = min_element(b, b + m) - b;
3b85 ▹ ▹ ▹ if(b[x] >= -eps)
048a ▹ ▹ ▹ ▹ break;
1626 ▹ ▹ ▹ int y = find_if(A[x], A[x] + n, [](dbl d) { return d < -eps; }) - A[x];
f625 ▹ ▹ ▹ if(y == n) throw 1; // no solution
09e2 ▹ ▹ ▹ pivot(x, y);
1aa1 ▹ ▹ }
aed8 ▹ ▹ while(true) {
2ed6 ▹ ▹ ▹ int y = max_element(c, c + n) - c;
da50 ▹ ▹ ▹ if(c[y] <= eps) break;
7c1e ▹ ▹ ▹ int x = -1;
35f1 ▹ ▹ ▹ dbl mn = 1. / 0.;
6fe9 ▹ ▹ ▹ for(int i = 0; i < m; i++)
ccd4 ▹ ▹ ▹ ▹ if(A[i][y] > eps && b[i] / A[i][y] < mn)
e45b ▹ ▹ ▹ ▹ ▹ mn = b[i] / A[i][y], x = i;
4e67 ▹ ▹ ▹ if(x == -1) throw 2; // unbounded
cb5c ▹ ▹ ▹ pivot(x, y);
1dd1 ▹ ▹ }
07c8 ▹ ▹ memset(sol, 0, sizeof(dbl) * n);
b315 ▹ ▹ for(int i = 0; i < m; i++)
6862 ▹ ▹ ▹ if(Y[i] < n)
7074 ▹ ▹ ▹ ▹ sol[Y[i]] = b[i];
c948 ▹ ▹  return ans;
e1f8 ▹  }
2062 };
```

5.5 Zeta

```
d41d // To calculate c[i] = sum (a[j] * b[k]) st j | k == i
d41d // Use c = itf(tf(a) * tf(b)), where * is element by element multiplication
d41d
d41d // Common transformations and inverses:
d41d // OR - (a, b) => (a, a + b) | (a, b) => (a, b - a)
d41d // AND - (a, b) => (a + b, b) | (a, b) => (a - b, b)
d41d // XOR - (a, b) => (a + b, a - b) | (a, b) => ((a + b) / 2, (a - b) / 2)
d41d
d41d //typedef ll num;
d41d
d41d // Transform a inplace (OR), initially l = 0, r = 2^n - 1
10ea void tf(num a[], int l, int r) {
eb81 ▹  if(l == r) return;
011c ▹  int m = (l + r) / 2;
a731 ▹  tf(a, l, m);
4695 ▹  tf(a, m + 1, r);
7c0f ▹  for(int i = l; i <= m; i++)
8bfd ▹ ▹  a[m + 1 + (i - l)] += a[i];
cc36 }
cc36
cc36 // Inverse transforms a inplace (OR), initially l = 0, r = 2^n - 1
bf63 void itf(num a[], int l, int r) {
91f1 ▹  if(l == r) return;
60b5 ▹  int m = (l + r) / 2;
0137 ▹  for(int i = l; i <= m; i++)
2488 ▹ ▹  a[m + 1 + (i - l)] -= a[i];
2726 ▹  itf(a, l, m);
a933 ▹  itf(a, m + 1, r);
dd54 }
```

5.6 Zeta Disjoint Or

```
d41d //const int K = ;
d41d //typedef ll num;
d41d
d41d // overwrites b such that b[i] = sum (a[j]) such that (j | i) == i and popcount(j) = k
a6e5 void tf(int k, num a[], num b[], int l, int r) {
9108 ▹  if(l == r) return (void) (b[l] = a[l] * (__builtin_popcount(l) == k));
9461 ▹  int m = (l + r) / 2;
2a2c ▹  tf(k, a, b, l, m);
bc25 ▹  tf(k, a, b, m + 1, r);
eed5 ▹  for(int i = l; i <= m; i++)
85a8 ▹ ▹  b[m + 1 + (i - l)] += b[i];
dd92 }
dd92
dd92 // Ranked mobius transform (transform above for all k)
1545 void tf(int k, num a[], num b[K+1][1 << K]) {
25f9 ▹  for(int i = 0; i <= k; i++)
7c00 ▹ ▹  tf(i, a, b[i], 0, (1 << k) - 1);
28f0 }
28f0
28f0 // Convolutes two transforms. c[j][i] = sum(a[g][i] * b[k - g][i]) for 0 <= g <= j
7d72 void conv(int k, num a[K+1][1 << K], num b[K+1][1 << K], num c[K+1][1 << K]) {
bcb2 ▹  for(int j = 0; j <= k; j++)
5dbc ▹ ▹  for(int i = 0; i < (1 << k); i++) {
fee2 ▹ ▹ ▹  c[j][i] = 0;
14cc ▹ ▹ ▹  for(int g = 0; g <= j; g++)
3f8d ▹ ▹ ▹ ▹  c[j][i] += a[g][i] * b[j - g][i];
e57d ▹ ▹  }
b86a }
b86a
```

```
b86a // Inverse of ranked mobius transform for k
e172 void itf(num a[], int l, int r) {
98bf ▹ if(l == r) return;
bbab ▹ int m = (l + r) / 2;
6fa1 ▹ for(int i = l; i <= m; i++)
cf6c ▹ ▹ a[m + 1 + (i - l)] -= a[i];
81b1 ▹ itf(a, l, m);
888f ▹ itf(a, m + 1, r);
69a2 }
69a2
69a2 // Inverse of ranked mobius transform for all k
d320 void itf(int k, num a[K+1][1 << K], num b[]) {
7dc7 ▹ for(int j = 0; j <= k; j++) {
33a6 ▹ ▹ itf(a[j], 0, (1 << k) - 1);
def6 ▹ ▹ for(int i = 0; i < (1 << k); i++)
8bbf ▹ ▹ ▹ if(__builtin_popcount(i) == j)
3acd ▹ ▹ ▹ ▹ b[i] = a[j][i];
791b ▹ }
c710 }
c710
c710 // use when you want to calculate c[i] = sum (a[j] * b[k]) such that (j | k) == i and (j & k) = 0
c710 // example use (if the size of a and b is (1 << k))
c710 // tf(k, a, a_);
c710 // tf(k, b, b_);
c710 // conv(k, a_, b_, ans);
c710 // itf(k, ans, c);
c710 // the answer will now be stored in c
```

5.7 Miller-Rabin

```
a288 llu llrand() { llu a = rand(); a<<= 32; a+= rand(); return a;}
67b7 int is_probably_prime(llu n) {
61d5    if (n <= 1) return 0;
2ecf    if (n <= 3) return 1;
a093    llu s = 0, d = n - 1;
0127    while (d % 2 == 0) {
028a       d/= 2; s++;
1c22    }
6cab    for (int k = 0; k < 64; k++) {
fc88       llu a = (llrand() % (n - 3)) + 2;
9d61       llu x = exp_mod(a, d, n);
e9cb       if (x != 1 && x != n-1) {
6e13          for (int r = 1; r < s; r++) {
1479             x = mul_mod(x, x, n);
569b             if (x == 1)
7ee2                return 0;
74f4             if (x == n-1)
344f                break;
429d          }
c1fc          if (x != n-1)
85bd             return 0;
03b9       }
abcb    }
8fad    return 1;
78e3 }
```

5.8 Pollard-Rho

```
295a llu rho(llu n) {
dd00 ▹ llu d, c = rand() % n, x = rand() % n, xx = x;
77b5 ▹ if (n % 2 == 0)
d711 ▹ ▹ return 2;
410c ▹ do {
6200 ▹ ▹ x = (mul_mod(x, x, n) + c) % n;
72a6 ▹ ▹ xx = (mul_mod(xx, xx, n) + c) % n;
7ba8 ▹ ▹ xx = (mul_mod(xx, xx, n) + c) % n;
bf50 ▹ ▹ d = gcd(val_abs(x - xx), n);
07a4 ▹  } while (d == 1);
4ae0 ▹  return d;
0884 }
b528 map <llu,int> F;
6ac2 void factor(llu n) {
3fa3 ▹  if (n == 1)
aa26 ▹  ▹  return;
d6b5 ▹  if (is_probably_prime(n)) {
780e ▹  ▹  F[n]++;
7609 ▹  ▹  return;
1f13 ▹  }
6468 ▹  llu d = rho(n);
0bcb ▹  factor(d);
79c1 ▹  factor(n/d);
838b }
```

6 Old Solutions

6.1 Ceiling Function

```
2b74 #include <bits/stdc++.h>
916e using namespace std;
1c98 #define fst first
a520 #define snd second
2029 typedef long long ll;
d15c typedef pair<int, int> pii;
7821 #define pb push_back
0426 #define for_tests(t, tt) int t; scanf("%d", &t); for(int tt = 1; tt <= t; tt++)
9c0f const ll modn = 1000000007;
7e89 inline ll mod(ll x) { return x % modn; }
7e89
2822 const int N = 112345;
31df int L[N], R[N], v[N];
89a2 int en = 1;
89a2
f9d0 int add(int r, int x) {
6a54    if(r == 0) {
5b4f       r = en++;
b4e3       v[r] = x;
d4ea       return r;
30fe    }
2e10    if(x < v[r])
666c       L[r] = add(L[r], x);
9fec    else
5ade       R[r] = add(R[r], x);
f8a3    return r;
976e }
976e
ed06 string get_str(int r) {
a87f    if(r == 0) return "";
b676    return "(" + get_str(L[r]) + "," + get_str(R[r]) + ")";
ee93 }
ee93
b16c string s[112345];
b16c
ff26 int main() {
0b99    int n, k, i, j, x;
6a5f    scanf("%d %d", &n, &k);
c285    for(i = 0; i < n; i++) {
e01a       int root = 0;
53d4       for(j = 0; j < k; j++) {
dd11          scanf("%d", &x);
c369          root = add(root, x);
71fa       }
b98a       s[i] = get_str(root);
9beb    }
0997    sort(s, s + n);
7743    printf("%d\n", int(unique(s, s + n) - s));
459a }
```

6.2 Secret Chamber at Mount Rushmore

```
2b74 #include <bits/stdc++.h>
916e using namespace std;
1c98 #define fst first
a520 #define snd second
2029 typedef long long ll;
d15c typedef pair<int, int> pii;
7821 #define pb push_back
0426 #define for_tests(t, tt) int t; scanf("%d", &t); for(int tt = 1; tt <= t; tt++)
9c0f const ll modn = 1000000007;
7e89 inline ll mod(ll x) { return x % modn; }
7e89
8139 char adj[256][256];
0057 char seen[256];
0057
50bb void go(char p, char u) {
6cda    if(seen[u] == p) return;
32a9    seen[u] = p;
ce1c    adj[p][u] = 1;
d57c    for(int v = 'a'; v <= 'z'; v++)
d982       if(adj[u][v])
6738          go(p, v);
eec4 }
eec4
f984 char s[1123], t[1123];
f984
b1c8 int main() {
7cfa    int i, m, n, j;
05cf    scanf("%d %d", &m, &n);
adb3    for(i = 0; i < m; i++) {
6fcd       char a, b;
8cd6       scanf(" %c %c", &a, &b);
1747       adj[a][b] = 1;
108e    }
2015    for(i = 'a'; i <= 'z'; i++)
0ec7       go(i, i);
628b    for(i =0 ; i < n; i++) {
c020       scanf("%s %s", s, t);
48c3       if(strlen(s) != strlen(t)) { puts("no"); continue; }
a036       for(j = 0; s[j]; j++)
f0dd          if(!adj[s[j]][t[j]])
c9fe             break;
3484       if(s[j]) puts("no");
ea0d       else puts("yes");
9c17    }
3ad5 }
```

6.3 Need for Speed

```
2b74 #include <bits/stdc++.h>
916e using namespace std;
1c98 #define fst first
a520 #define snd second
2029 typedef long long ll;
d15c typedef pair<int, int> pii;
7821 #define pb push_back
0426 #define for_tests(t, tt) int t; scanf("%d", &t); for(int tt = 1; tt <= t; tt++)
9c0f const ll modn = 1000000007;
7e89 inline ll mod(ll x) { return x % modn; }
7e89
ec23 const int N = 1123;
8d0d int d[N], s[N];
8d0d
d8cc int main() {
```

```
5db9    int n, t, i;
3570    scanf("%d %d", &n, &t);
ae94    long double l = -2e7, r = 1502;
1894    for(i = 0; i < n; i++) scanf("%d %d", &d[i], &s[i]);
3f4d    for(int x = 0; x < 200; x++) {
31f6       long double c = (l + r) / 2;
efdb       long double tot = 0;
77c2       for(i = 0; i < n; i++) {
37ac          long double ss = s[i] - c;
1bef          if(ss <= 0) break;
8772          tot += d[i] / ss;
2164       }
b481       if(tot >= t || i < n) r = c;
462e       else l = c;
9b63    }
45a9    printf("%.10f\n", -double(l));
45a9
5987 }
```

6.4 Amalgamated Artichokes

```
2b74 #include <bits/stdc++.h>
916e using namespace std;
916e
901b int main() {
f999    int p, a, b, c, d, n;
a0b1    scanf("%d %d %d %d %d %d", &p, &a, &b, &c, &d, &n);
7055    double mx = -1. / 0.;
2874    double ans = 0;
bd33    for(int i = 1; i <= n; i++) {
4602       double x = p * (sin(a * i + b) + cos(c * i + d) + 2);
2f35       mx = max(mx, x);
6b60       ans = max(ans, mx - x);
8efe    }
1a50    printf("%.10f\n", ans);
1895 }
```

6.5 Low Power

```
2b74 #include <bits/stdc++.h>
916e using namespace std;
916e
368b typedef long long ll;
8ba8 typedef pair<ll, ll> pii;
bd85 #define pb push_back
bd85
9eb0 const int N = 1e6+7;
9eb0
9142 int n, k;
280f ll a[N];
280f
5e6b bool solve (ll d) {
0c1c ▹  ll s = 0, m = n;
4956 ▹  for (int i = 0; m && i < 2*n*k - 1; i++) {
ec6a ▹  ▹  if (a[i+1] - a[i] <= d) {
53c8 ▹  ▹  ▹  m--;
cd3c ▹  ▹  ▹  i++;
4493 ▹  ▹  ▹  s += 2*(k-1);
4dd3 ▹  ▹  } else if (!s) return 0;
e3eb ▹  ▹  else s--;
61ba ▹  }
73ae ▹  return 1;
9c22 }
9c22
e597 int main () {
181e ▹  scanf("%d %d",&n, &k);
181e
a5b5 ▷ for (int i = 0; i < 2*n*k; i++)
b449 ▷ ▷  scanf("%lld", &a[i]);
7e4f ▷ sort(a, a+2*n*k);
7e4f
8062 ▷ ll lo = 0, hi = 1e9+2;
3253 ▷ while (lo < hi) {
1ff9 ▷ ▷  ll md = (lo+hi)/2;
57ac ▷ ▷  if (solve(md)) hi = md;
6a3b ▷ ▷  else lo = md+1;
fa61 ▷ }
fa61
7221 ▷ printf("%lld\n", lo);
56de }
```

7 Anotações

7.1 Intersecção de Matróides

Sejam $M_1 = (E, I_1)$ e $M_2 = (E, I_2)$ matróides. Então $\max\limits_{S \in I_1 \cap I_2} |S| = \min\limits_{U \subseteq E} r_1(U) + r_2(E \setminus U)$.

7.2 Möebius

Se $F(n) = \sum\limits_{d|n} f(d)$, então $f(n) = \sum\limits_{d|n} \mu(d) F(n/d)$.

7.3 Burnside

Seja $A\colon GX \to X$ uma ação. Defina:

- $w :=$ número de órbitas em X.
- $S_x := \{g \in G \mid g \cdot x = x\}$
- $F_g := \{x \in X \mid g \cdot x = x\}$

  Então $w = \frac{1}{|G|} \sum\limits_{x \in X} |S_x| = \frac{1}{|G|} \sum\limits_{g \in G} |F_g|$.

7.4 Landau

Existe um torneio com graus de saída $d_1 \leq d_2 \leq \ldots \leq d_n$ sse:

- $d_1 + d_2 + \ldots + d_n = \binom{n}{2}$
- $d_1 + d_2 + \ldots + d_k \geq \binom{k}{2} \quad \forall 1 \leq k \leq n$.

Para construir, fazemos 1 apontar para $2, 3, \ldots, d_1 + 1$ e seguimos recursivamente.

7.5 Erdös-Gallai

Existe um grafo simples com graus $d_1 \geq d_2 \geq \ldots \geq d_n$ sse:

- $d_1 + d_2 + \ldots + d_n$ é par
- $\sum\limits_{i=1}^{k} d_i \leq k(k-1) + \sum\limits_{i=k+1}^{n} \min(d_i, k) \quad \forall 1 \leq k \leq n$.

Para construir, ligamos 1 com $2, 3, \ldots, d_1 + 1$ e seguimos recursivamente.

7.6 Gambler's Ruin

Em um jogo no qual ganhamos cada aposta com probabilidade p e perdemos com probabilidade $q := 1 - p$, paramos quando ganhamos B ou perdemos A. Então $Prob(\text{ganhar B}) = \frac{1-(p/q)^B}{1-(p/q)^{A+B}}$.

7.7 Extra

- $Fib(x + y) = Fib(x + 1)Fib(y) + Fib(x)Fib(y - 1)$