ACM ICPC Reference

University of Notre Dame

October 23, 2019

Contents

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

```
#!/bin/bash
while IFS=$'\n' read -r line; do
    trim=$(echo "$line" | tr -d "[:space:]")
    md5=$(echo -n "${trim%%\/\/*}" | md5sum)
    md5=${md5:0:4}
    [ "${trim:~0}" == "$" ] && md5="@$md5@"
    echo "$md5 $line"
done
```

1 Geometry

1.1 Base

```
d41d // typedef double cood; cood eps = 1e-8; // risky: XXX, untested: TODO
00a0 const double pi = acos(-1.);
ccb5 template<typename T> inline T sq(T x) { return x*x; }
87bc struct vec {
b86a     cood x, y;
6e4f     vec () : x(0), y(0) {} vec (cood a, cood b) : x(a), y(b) {}
741a     inline vec operator - (vec o) { return {x - o.x, y - o.y}; }
ff7e     inline vec operator + (vec o) { return {x + o.x, y + o.y}; }
b6dd     inline vec operator * (cood o) { return {x * o, y * o}; }
2711     inline vec operator / (cood o) { return {x / o, y / o}; }
6ac9     inline cood operator ^ (vec o) { return x * o.y - y * o.x; }
83dd     inline cood operator * (vec o) { return x * o.x + y * o.y; }
46ef     inline cood cross (vec a, vec b) { return ((*this)-a) ^ ((*this)-b); } // |(this)a||(this)b|sen(angle)
cbad     inline cood inner (vec a, vec b) { return ((*this)-a) * ((*this)-b); } // |(this)a||(this)b|cos(angle)
cddd     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]
e4d3     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
2e1f     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
5d26     inline cood sq (vec o = vec()) { return inner(o,o); }
e7cf     inline double nr (vec o = vec()) { return sqrt(sq(o)); } //$
4e72     inline vec operator ~ () { return (*this)/nr(); }
f149     inline vec proj (vec a, vec b) { return a + (b-a)*(a.inner((*this),b) / a.sq(b)); } // projects this onto line ab
1664     inline vec rotate (double a) { return vec(cos(a) * x - sin(a) * y, sin(a) * x + cos(a) * y); } // ccw by a radians
3206     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
5e56     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
8831     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)); }
436b     inline bool operator == (const vec & o) const { return abs(x-o.x) <= eps && abs(y-o.y) <= eps; }
5522     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)
d41d     // full ccw angle strict compare beginning upwards (this+(0,1)) around (*this)
d41d     // incresing distance on ties, this is the first
69ad     bool compare (vec a, vec b) {
a482         if ((*this < a) != (*this < b)) return *this < b;
bdb1         int o = ccw(a,b); return o?o>0:((a == *this && !(a == b)) || a.dir(*this,b) < 0);
cbb1     }
2145 }; //$
bafe struct lin { // line
6143     vec p; cood c; // p*(x,y) = c
1105     lin () {} lin (vec a, cood b) : p(a), c(b) {}
d036     lin (vec s, vec t) : p((s-t).rot90()), c(p*s) {}
5c8b     inline lin parll (vec v) { return lin(p,v*p); }
1263     inline lin perp () { return lin(p.rot90(),c); }
```

```
3838 ▹ 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); }
1375 ▹ bool contains (vec v) { return abs(p*v - c) <= eps; }
eda5 ▹ vec at_x (cood x) { return vec(x,(c-p.x*x)/p.y); }
c0fb ▹ vec at_y (cood y) { return vec((c-y*p.y)/p.x,y); }
e1ef ▹ double sign_dist (vec v) { return double(p*v - c)/p.nr(); }
2145 }; //$
3236 struct cir { // circle
b6d3 ▹ vec c; cood r;
126a ▹ cir () {} cir (vec v, cood d) : c(v), r(d) {}
c118 ▹ cir (vec u, vec v, vec w) { // XXX untreated degenerates
0fb6 ▹ ▹ vec mv = (u+v)/2; lin s(mv, mv+(v-u).rot90());
bf5f ▹ ▹ vec mw = (u+w)/2; lin t(mw, mw+(w-u).rot90());
a0c4 ▹ ▹ c = s.inter(t); r = c.nr(u);
cbb1 ▹ }//$
9e54 ▹ inline bool contains (vec w) { return c.sq(w) <= sq(r) + eps; } // border included
0549 ▹ inline bool border (vec w) { return abs(c.sq(w) - sq(r)) <= eps; }
1cd6 ▹ inline bool has_inter (cir o) { return c.sq(o.c) <= sq(r + o.r) + eps; } // borders included
376d ▹ inline bool has_border_inter (cir o) { return has_inter(o) && c.sq(o.c) + eps >= sq(r - o.r); }
8ab4 ▹ 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
9bf7 ▹ 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
7abe ▹ inline double arc_area (vec a, vec b) { return c.angle(a,b)*r*r/2; } // smallest arc, ccw positive
f967 ▹ 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
84ec ▹ ▹ if (contains(v) && !border(v)) throw 0;
2894 ▹ ▹ cood d2 = c.sq(v); double s = sqrt(d2 - r*r); s = (s==s)?s:0;
0f70 ▹ ▹ double al = atan2(r,s); vec t = (~(c-v));
3a69 ▹ ▹ return pair<vec,vec>(v + t.rotate(al)*s, v + t.rotate(-al)*s);
cbb1 ▹ }//$
c56f ▹ pair<vec,vec> border_inter (cir o) {
c4d4 ▹ ▹ if (!has_border_inter(o) || o.c == (*this).c) throw 0;
2b40 ▹ ▹ double a = (sq(r) + o.c.sq(c) - sq(o.r))/(2*o.c.nr(c));
b647 ▹ ▹ vec v = (o.c - c)/o.c.nr(c); vec m = c + v * a;
65b9 ▹ ▹ double h = sqrt(sq(r) - sq(a)); h = h!=h?0:h;
440c ▹ ▹ return pair<vec,vec>(m + v.rot90()*h, m - v.rot90()*h);
cbb1 ▹ }//$
5182 ▹ pair<vec,vec> border_inter_lin (vec a, vec b) { // first is closest to a than second
c6e7 ▹ ▹ if (a.sq(b) <= eps) { if (border(a)) return pair<vec,vec>(a,a); throw 0; }
40f6 ▹ ▹ if (a.dir(b,c) == -1) swap(a,b);
45ab ▹ ▹ if (!has_inter_lin(a,b)) throw 0;
5cb6 ▹ ▹ double d2 = c.dist2_lin(a,b); vec p = (b-a)/a.nr(b);
0aca ▹ ▹ double h = sqrt(r*r - d2); h = h!=h?0:h;
ddf2 ▹ ▹ double y = sqrt(c.sq(a) - d2); y = y!=y?0:y;
5539 ▹ ▹ return pair<vec,vec>(a + p*(y-h), a + p*(y+h));
cbb1 ▹ }//$
be35 ▹ double triang_inter (vec a, vec b) { // ccw oriented, this with (c,a,b)
53ba ▹ ▹ if (c.sq(a) > c.sq(b)) return -triang_inter(b,a);
148a ▹ ▹ if (contains(b)) return c.cross(a,b)/2;
7434 ▹ ▹ if (!has_inter_seg(a,b)) return arc_area(a,b);
773a ▹ ▹ pair<vec,vec> itr = border_inter_lin(b,a); // order important
12a9 ▹ ▹ if (contains(a)) return c.cross(a,itr.first)/2 + arc_area(itr.first,b);
c2f4 ▹ ▹ return arc_area(a,itr.second) + c.cross(itr.second,itr.first)/2 + arc_area(itr.first,b);
cbb1 ▹ }
2145 }; //$
a71b bool inter_seg (vec a, vec b, vec c, vec d) {
2397 ▹ if (a.in_seg(c, d) || b.in_seg(c, d) || c.in_seg(a, b) || d.in_seg(a, b)) return true;
bbbd ▹ return (c.ccw(a, b) * d.ccw(a, b) == -1 && a.ccw(c, d) * b.ccw(c, d) == -1);
cbb1 }
e0fd 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
f1ea ▹ srand(time(NULL)); random_shuffle(v, v+n); cir c(vec(), 0); int i,j,k;
b11a ▹ for (i = 0; i < n; i++) if (!c.contains(v[i]))
```

```
e5b6 ▹ ▹ for (c = cir(v[i],0), j = 0; j < i; j++) if (!c.contains(v[j]))
a47c ▹ ▹ ▹ for (c = cir((v[i] + v[j])/2,v[i].nr(v[j])/2), k = 0; k < j; k++) if (!c.contains(v[k]))
3dd3 ▹ ▹ ▹ ▹ ▹ c = cir(v[i],v[j],v[k]);
807f ▹ return c;
cbb1 }//$
d45c int convex_hull (vec * v, int n, int border_in) { // nlg | border_in (should border points stay?)
4f17 ▹ swap(v[0], *min_element(v,v+n)); int s, i;
f37e ▹ 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)); });
a69c ▹ if (border_in) {
9492 ▹ ▹ for (s = n-1; s > 1 && v[s].ccw(v[s-1],v[0]) == 0; s--);
0bb0 ▹ ▹ reverse(v+s, v+n);
cbb1 ▹ }
c497 ▹ for (i = s = 0; i < n; i++) if (!s || !(v[s-1] == v[i])) {
cea9 ▹ ▹ for (; s >= 2 && v[s-1].ccw(v[s-2],v[i]) >= border_in; s--);
ceca ▹ ▹ swap(v[s++],v[i]);
cbb1 ▹ }
0478 ▹ return s;
cbb1 }//$
79b9 int monotone_chain (vec * v, int n, int border_in) { // nlg | border_in (should border points stay?)
5031 ▹ vector<vec> r; sort(v, v+n); n = unique(v, v+n) - v;
d885 ▹ 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();
dd80 ▹ r.pop_back(); unsigned int s = r.size();
c19d ▹ 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();
a255 ▹ return copy(r.begin(), r.end() - (r.size() > 1), v) - v;
cbb1 }//$
f80f double polygon_inter (vec * p, int n, cir c) { // signed area
2eae ▹ 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); });
cbb1 }//$
3214 int polygon_pos (vec * p, int n, vec v) { // lg | p should be simple (-1 out, 0 border, 1 in)
6c2a ▹ int in = -1; // it's a good idea to randomly rotate the points in the double case, numerically safer
6033 ▹ for (int i = 0; i < n; i++) {
2bca ▹ ▹ vec a = p[i], b = p[i?i-1:n-1]; if (a.x > b.x) swap(a,b);
c9e9 ▹ ▹ if (a.x + eps <= v.x && v.x < b.x + eps) { in *= v.ccw(a,b); }
c3b1 ▹ ▹ else if (v.in_seg(a,b)) { return 0; }
cbb1 ▹ }
091d ▹ return in;
cbb1 }//$
271f int polygon_pos_convex (vec * p, int n, vec v) { // lg(n) | (-1 out, 0 border, 1 in) TODO
a868 ▹ if (v.sq(p[0]) <= eps) return 0;
088f ▹ if (n <= 1) { return 0; } if (n == 2) { return v.in_seg(p[0],p[1])?0:-1; }
2ceb ▹ if (v.ccw(p[0],p[1]) < 0 || v.ccw(p[0],p[n-1]) > 0) return -1;
fcfd ▹ int di = lower_bound(p+1,p+n-1,v, [&p](vec a,vec v) { return v.ccw(p[0],a) > 0; }) - p;
adf3 ▹ if (di == 1) return v.ccw(p[1],p[2]) >= 0?0:-1;
cfa4 ▹ return v.ccw(p[di-1],p[di]);
cbb1 }//$
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)
91d7 ▹ if (l + 1 >= r) return inf;
900b ▹ if (!sorted) sort(v+l,v+r,[](vec a, vec b){ return a.x < b.x; });
89cd ▹ int m = (l+r)/2; cood x = v[m].x;
1a44 ▹ cood res = min(closest_pair(v,w,l,m,1),closest_pair(v,w,m,r,1));
d046 ▹ merge(v+l,v+m,v+m,v+r,w+l,[](vec a, vec b){ return a.y < b.y; });
2dd0 ▹ for (int i = l, s = l; i < r; i++) if (sq((v[i] = w[i]).x - x) < res) {
ad96 ▹ ▹ for (int j = s-1; j >= l && sq(w[i].y - w[j].y) < res; j--)
c3b1 ▹ ▹ ▹ res = min(res, w[i].sq(w[j]));
1991 ▹ ▹ w[s++] = v[i];
cbb1 ▹ }
b505 ▹ return res;
cbb1 }//$
ac2e double union_area (cir * v, int n) { // n^2lg | XXX joins equal circles TODO (AC on szkopul, no tests)
c765 ▹ struct I { vec v; int i; } c[2*(n+4)];
cf66 ▹ srand(time(NULL)); cood res = 0; vector<bool> usd(n);
dd83 ▹ cood lim = 1./0.; for (int i = 0; i < n; i++) lim = min(lim, v[i].c.y - v[i].r - 1);
0b02 ▹ for (int i = 0, ss = 0; i < n; i++, ss = 0) {
dc37 ▹ ▹ vec fp = v[i].c + vec(0,v[i].r).rotate(rand()); // rotation avoids corner on cnt initialization
```

```
6e87 ▹ ▹ int cnt = 0, eq = 0;
578e ▹ ▹ for (int j = 0; j < n; j++) {
df48 ▹ ▹ ▹ cnt += (usd[j] = v[j].contains(fp));
2311 ▹ ▹ ▹ if (!v[i].has_border_inter(v[j])) continue;
8daa ▹ ▹ ▹ if (v[i].c == v[j].c) eq++;
4e6b ▹ ▹ ▹ else {
e59e ▹ ▹ ▹ ▹ pair<vec,vec> r = v[i].border_inter(v[j]);
0782 ▹ ▹ ▹ ▹ c[ss++] = {r.first, j}; c[ss++] = {r.second, j};
cbb1 ▹ ▹ ▹ }
cbb1 ▹ ▹ }
d21b ▹ ▹ vec d = vec(v[i].r,0); for (int k = 0; k < 4; k++, d = d.rot90()) c[ss++] = {v[i].c + d, i};
85d3 ▹ ▹ int md = partition(c,c+ss,[v,i,fp](I a){return a.v.ccw(v[i].c,fp) > 0;}) - c;
19c7 ▹ ▹ sort(c,c+md,[v,i](I a,I b){return a.v.ccw(v[i].c,b.v) < 0;});
7430 ▹ ▹ sort(c+md,c+ss,[v,i](I a,I b){return a.v.ccw(v[i].c,b.v) < 0;});
56cd ▹ ▹ for (int j = 0; j < ss; j++) {
2b5e ▹ ▹ ▹ if (c[j].i != i) { cnt -= usd[c[j].i]; usd[c[j].i] = !usd[c[j].i]; cnt += usd[c[j].i]; }
b115 ▹ ▹ ▹ vec a = c[j].v, b = c[(j+1)%ss].v;
7c4a ▹ ▹ ▹ 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);
e20b ▹ ▹ ▹ cood loc = (a.x<b.x)?cir-tra:tra+cir; res += (cnt==eq)?loc/eq:0;
cbb1 ▹ ▹ }
cbb1 ▹ }
b505 ▹ return res;
cbb1 }//$
4ede pii antipodal (vec * p, int n, vec v) { // lg(n) | extreme segments relative to direction v TODO
d41d ▹ // po: closest to dir, ne: furthest from dir
3bd9 ▹ bool sw = ((p[1]-p[0])*v < 0);
d189 ▹ if (sw) v = vec(0,0) - v; // lower_bound returns the first such that lambda is false
0303 ▹ int md = lower_bound(p+1, p+n, v, [p] (vec & a, vec v) { return (a-p[0])*v > eps; }) - p; // chain separation
25f1 ▹ 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
9dc9 ▹ 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
5703 ▹ if (sw) swap(po,ne);
ef0b ▹ return pii(po,ne);
cbb1 }//$
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
8d81 ▹ if (!n || !m) { return 0; } int i, j, s; r[0] = a[0] + b[0];
de54 ▹ for (i = 0, j = 0, s = 1; i < n || j < m; s++) {
1ab0 ▹ ▹ if (i >= n) j++;
1dc4 ▹ ▹ else if (j >= m) i++;
4e6b ▹ ▹ else {
4f09 ▹ ▹ ▹ int o = (a[(i+1)%n]+b[j%m]).ccw(r[s-1],a[i%n]+b[(j+1)%m]);
e43c ▹ ▹ ▹ j += (o >= 0); i += (o <= 0);
cbb1 ▹ ▹ }
f5b4 ▹ ▹ r[s] = a[i%n] + b[j%m];
cbb1 ▹ }
162b ▹ return s-1;
cbb1 }//$
9e65 int inter_convex (vec * p, int n, vec * q, int m, vec * r) { // (n+m) | XXX
2d76 ▹ int a = 0, b = 0, aa = 0, ba = 0, inflag = 0, s = 0;
2a6c ▹ while ((aa < n || ba < m) && aa < n+n && ba < m+m) {
b977 ▹ ▹ vec p1 = p[a], p2 = p[(a+1)%n], q1 = q[b], q2 = q[(b+1)%m];
35b2 ▹ ▹ vec A = p2 - p1, B = q2 - q1;
1479 ▹ ▹ int cross = vec(0,0).ccw(A,B), ha = p1.ccw(p2,q2), hb = q1.ccw(q2,p2);
c6e0 ▹ ▹ if (cross == 0 && p2.ccw(p1,q1) == 0 && A*B < -eps) {
507b ▹ ▹ ▹ if (q1.in_seg(p1,p2)) r[s++] = q1;
5e83 ▹ ▹ ▹ if (q2.in_seg(p1,p2)) r[s++] = q2;
ce58 ▹ ▹ ▹ if (p1.in_seg(q1,q2)) r[s++] = p1;
526a ▹ ▹ ▹ if (p2.in_seg(q1,q2)) r[s++] = p2;
7b25 ▹ ▹ ▹ if (s < 2) return s;
e2a8 ▹ ▹ ▹ inflag = 1; break;
5e6d ▹ ▹ } else if (cross != 0 && inter_seg(p1,p2,q1,q2)) {
f420 ▹ ▹ ▹ if (inflag == 0) aa = ba = 0;
2b81 ▹ ▹ ▹ r[s++] = lin(p1,p2).inter(lin(q1,q2));
37fd ▹ ▹ ▹ inflag = (hb > 0) ? 1 : -1;
cbb1 ▹ ▹ }
5499 ▹ ▹ if (cross == 0 && hb < 0 && ha < 0) return s;
```

```
0872 		bool t = cross == 0 && hb == 0 && ha == 0;
c0ec 		if (t ? (inflag == 1) : (cross >= 0) ? (ha <= 0) : (hb > 0)) {
9873 			if (inflag == -1) r[s++] = q2;
1146 			ba++; b++; b %= m;
9d97 		} else {
5c98 			if (inflag == 1) r[s++] = p2;
5ecb 			aa++; a++; a %= n;
cbb1 		}
cbb1 	}
c1b2 	if (inflag == 0) {
3880 		if (polygon_pos_convex(q,m,p[0]) >= 0) { copy(p, p+n, r); return n; }
115c 		if (polygon_pos_convex(p,n,q[0]) >= 0) { copy(q, q+m, r); return m; }
cbb1 	}
fc37 	s = unique(r, r+s) - r;
2629 	if (s > 1 && r[0] == r[s-1]) s--;
0478 	return s;
cbb1 }//$
03ae bool isear (vec * p, int n, int i, int prev[], int next[]) { // aux to triangulate
7630 	vec a = p[prev[i]], b = p[next[i]];
2d9f 	if (b.ccw(a,p[i]) <= 0) return false;
578e 	for (int j = 0; j < n; j++) {
97eb 		if (j == prev[i] || j == next[i]) continue;
0ef9 		if (p[j].ccw(a,p[i]) >= 0 && p[j].ccw(p[i],b) >= 0 && p[j].ccw(b,a) >= 0) return false;
0639 		int k = (j+1)%n;
2898 		if (k == prev[i] || k == next[i]) continue;
a537 		if (inter_seg(p[j],p[k],a,b)) return false;
cbb1 	}
8a6c 	return true;
cbb1 }
1851 int triangulate (vec * p, int n, bool ear[], int prev[], int next[], int tri[][3]) { // O(n^2) | n >= 3
d14e 	int s = 0, i = 0;
78d0 	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); }
6b3b 	for (int lef = n; lef > 3; lef--, i = next[i]) {
ced7 		while (!ear[i]) i = next[i];
e7a9 		tri[s][0] = prev[i]; tri[s][1] = i; tri[s][2] = next[i]; s++; // tri[i][0],i,tri[i][1] inserted
e0c0 		int c_prev = prev[i], c_next = next[i];
c354 		next[c_prev] = c_next; prev[c_next] = c_prev;
84b6 		ear[c_prev] = isear(p,n,c_prev,prev,next); ear[c_next] = isear(p,n,c_next,prev,next);
cbb1 	}
bc1d 	tri[s][0] = next[next[i]]; tri[s][1] = i; tri[s][2] = next[i]; s++; // tri[i][0],i,tri[i][1] inserted
0478 	return s;
cbb1 }
```

1.3 3D

```
f61c const double pi = acos(-1);
d41d // typedef double cood; cood eps = 1e-6; // risky: XXX, untested: TODO
3f73 struct pnt { // TODO it's not tested at all :)
5e43 	cood x, y, z;
cf2f 	pnt () : x(0), y(0), z(0) {} pnt (cood a, cood b, cood c) : x(a), y(b), z(c) {}
4e90 	inline pnt operator - (pnt o) { return pnt(x - o.x, y - o.y, z - o.z); }
2b18 	inline pnt operator + (pnt o) { return pnt(x + o.x, y + o.y, z + o.z); }
7470 	inline pnt operator * (cood o) { return pnt(x*o, y*o, z*o); }
8194 	inline pnt operator / (cood o) { return pnt(x/o, y/o, z/o); }
a269 	inline cood operator * (pnt o) { return x*o.x + y*o.y + z*o.z; } // inner: |this||o|*cos(ang)
079c 	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)
a2ea 	inline cood operator () (pnt a, pnt b) { return (*this)*(a^b); } // mixed: positive on the right-hand
    rule (thumb=this,index=a,mid=b)
d41d
f500 	inline cood inner (pnt a, pnt b) { return (a-(*this))*(b-(*this)); }
4114 	inline pnt cross (pnt a, pnt b) { return (a-(*this))^(b-(*this)); } // its norm is twice area of triangle
fa90 	inline cood mixed (pnt a, pnt b, pnt c) { return (a-(*this))(b-(*this),c-(*this)); } // 6 times the
    oriented area of thetahedra
d41d
4f78 	inline cood sq (pnt o = pnt()) { return inner(o,o); }
113b 	inline double nr (pnt o = pnt()) { return sqrt(sq(o)); }
```

```
6edf     inline pnt operator ~ () { return (*this)/nr(); }
d41d
11c0     inline bool in_seg (pnt a, pnt b) { return cross(a,b).sq() <= eps && inner(a,b) <= eps; } // tips included
a6b7     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); }
3a26     inline pnt proj (pnt a, pnt b, pnt c) { pnt n = a.cross(b,c); return (*this) - n*(n*((*this)-a))/n.sq(); }
d41d
8fbb     inline double dist2_lin (pnt a, pnt b) { return cross(a,b).sq()/a.sq(b); }
1880     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); }
39c1     inline double dist_pln (pnt a, pnt b, pnt c) { return abs((~a.cross(b,c))*((*this)-a)); }
5bc2     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) }); }
2145 };
eb48 inline cood area (pnt a, pnt b, pnt c) { return abs(a.cross(b,c).nr()) / 2; }
a6c7 inline cood vol (pnt a, pnt b, pnt c, pnt d) { return abs(a.mixed(b,c,d)) / 6; } // thetahedra
084a 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
af42     pnt c; cood r;
390f     sph () : c(), r(0) {} sph (pnt a, cood b) : c(a), r(b) {}
baaf     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.
171a     inline double area_hull (double h) { return 2.*pi*r*h; }
60a4     inline double vol_hull (double h) { return pi*h/6 * (3.*r*r + h*h); }
2145 };
```

2 Graphs

2.1 Dinic

```
d41d //typedef int num; const int N = ; const int M = * 2; const num eps = 0;
582d struct dinic {
9740     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;
1233     bool bfs(int s, int t) {
5ff9         seen[t] = ++tempo; lv[t] = 0; int ql = 0, qr = 0; qu[qr++] = t;
a872         while(ql != qr) {
036d             t = qu[ql++]; ei[t] = hd[t]; if(s == t) return true;
7e80             for(int e = hd[t]; e; e = nx[e]) if(seen[to[e]] != tempo && cp[e ^ 1] - fl[e ^ 1] > eps) {
a74c                 seen[to[e]] = tempo;
de5c                 lv[to[e]] = lv[t] + 1;
f0ff                 qu[qr++] = to[e];
cbb1             }
cbb1         }
d1fe         return false;
cbb1     }
a444     num dfs(int s, int t, num f) {
f449         if(s == t) return f;
d4ad         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)
7004             if(num rf = dfs(to[e], t, min(f, cp[e] - fl[e]))) {
805c                 fl[e] += rf;
5226                 fl[e ^ 1] -= rf;
2cb7                 return rf;
cbb1             }
bb30         return 0;
cbb1     }
d41d     // public $
de22     num max_flow(int s, int t) {
6cb2         num fl = 0;
1c5e         while (bfs(s, t)) for(num f; (f = dfs(s, t, numeric_limits<num>::max())); fl += f);
e508         return fl;
cbb1     }
5a3f     void add_edge(int a, int b, num c, num rc=0) {
d03a         to[en] = b; nx[en] = hd[a]; fl[en] = 0; cp[en] = c; hd[a] = en++;
```

```
2f94 ▹ ▹ to[en] = a; nx[en] = hd[b]; fl[en] = 0; cp[en] = rc; hd[b] = en++;
cbb1 ▹ }
7415 ▹ void reset_flow() { memset(fl, 0, sizeof(num) * en); }
ae0a ▹ void init(int n=N) { en = 2; memset(hd, 0, sizeof(int) * n); } // resets all
2145 };
```

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 {
b6db ▹ int es[N], to[M], nx[M], en = 2, pai[N], seen[N], tempo, qu[N];
ef55 ▹ val fl[M], cp[M], flow; num cs[M], d[N], tot;
d0cc ▹ val spfa(int s, int t) {
09b0 ▹ ▹ tempo++; int a = 0, b = 0;
e0c6 ▹ ▹ for(int i = 0; i < N; i++) d[i] = numeric_limits<num>::max();
68ad ▹ ▹ d[s] = 0; qu[b++] = s; seen[s] = tempo;
9841 ▹ ▹ while(a != b) {
32d9 ▹ ▹ ▹ int u = qu[a++]; if(a == N) a = 0; seen[u] = 0;
a86f ▹ ▹ ▹ for(int e = es[u]; e; e = nx[e]) if(cp[e] - fl[e] > val(0) && d[u] + cs[e] < d[to[e]] - eps) {
a694 ▹ ▹ ▹ ▹ d[to[e]] = d[u] + cs[e]; pai[to[e]] = e ^ 1;
1889 ▹ ▹ ▹ ▹ if(seen[to[e]] < tempo) { seen[to[e]] = tempo; qu[b++] = to[e]; if(b == N) b = 0; }
cbb1 ▹ ▹ ▹ }
cbb1 ▹ ▹ }
8e2a ▹ ▹ if(d[t] == numeric_limits<num>::max()) return false;
91fe ▹ ▹ val mx = numeric_limits<val>::max();
285a ▹ ▹ for(int u = t; u != s; u = to[pai[u]])
7039 ▹ ▹ ▹ mx = min(mx, cp[pai[u] ^ 1] - fl[pai[u] ^ 1]);
6de0 ▹ ▹ tot += d[t] * val(mx);
285a ▹ ▹ for(int u = t; u != s; u = to[pai[u]])
4c48 ▹ ▹ ▹ fl[pai[u]] -= mx, fl[pai[u] ^ 1] += mx;
b9aa ▹ ▹ return mx;
cbb1 ▹ }
d41d ▹ // public $
8662 ▹ num min_cost(int s, int t) {
3b69 ▹ ▹ tot = 0; flow = 0;
e66e ▹ ▹ while(val a = spfa(s, t)) flow += a;
126a ▹ ▹ return tot;
cbb1 ▹ }
457a ▹ void add_edge(int u, int v, val c, num s) {
1d08 ▹ ▹ fl[en] = 0; cp[en] = c; to[en] = v; nx[en] = es[u]; cs[en] = s; es[u] = en++;
8015 ▹ ▹ fl[en] = 0; cp[en] = 0; to[en] = u; nx[en] = es[v]; cs[en] = -s; es[v] = en++;
cbb1 ▹ }
8537 ▹ void reset_flow() { memset(fl, 0, sizeof(val) * en); }
451f ▹ void init(int n) { en = 2; memset(es, 0, sizeof(int) * n); } // XXX must be called
2145 };
```

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 {
2c47 ▹ 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;
1233 ▹ bool bfs(int s, int t) {
5ff9 ▹ ▹ seen[t] = ++tempo; lv[t] = 0; int ql = 0, qr = 0; qu[qr++] = t;
a872 ▹ ▹ while(ql != qr) {
036d ▹ ▹ ▹ t = qu[ql++]; ei[t] = hd[t]; if(s == t) return true;
7e80 ▹ ▹ ▹ for(int e = hd[t]; e; e = nx[e]) if(seen[to[e]] != tempo && cp[e ^ 1] - fl[e ^ 1] > eps) {
a74c ▹ ▹ ▹ ▹ seen[to[e]] = tempo;
de5c ▹ ▹ ▹ ▹ lv[to[e]] = lv[t] + 1;
f0ff ▹ ▹ ▹ ▹ qu[qr++] = to[e];
cbb1 ▹ ▹ ▹ }
cbb1 ▹ ▹ }
d1fe ▹ ▹ return false;
```

```
cbb1 ▹ }
e4d9 ▹ val dfs(int s, int t, val f) {
f449 ▹ ▹ if(s == t) return f;
d4ad ▹ ▹ 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)
9fe1 ▹ ▹ ▹ if(val rf = dfs(to[e], t, min(f, cp[e] - fl[e]))) {
805c ▹ ▹ ▹ ▹ fl[e] += rf;
5226 ▹ ▹ ▹ ▹ fl[e ^ 1] -= rf;
2cb7 ▹ ▹ ▹ ▹ return rf;
cbb1 ▹ ▹ ▹ }
bb30 ▹ ▹ return 0;
cbb1 ▹ }
5cbe ▹ bool spfa() {
a019 ▹ ▹ tempo++; int a = 0, b = 0, u;
99a4 ▹ ▹ for(int i = 0; i < n; i++) { d[i] = 0; qu[b++] = i; seen[i] = tempo; ct[i] = 0; }
9841 ▹ ▹ while(a != b) {
b492 ▹ ▹ ▹ u = qu[a++]; if(a == N) a = 0; seen[u] = 0;
d627 ▹ ▹ ▹ if(ct[u]++ >= n + 1) { a--; break; }
ccce ▹ ▹ ▹ for(int e = hd[u]; e; e = nx[e]) if(cp[e] - fl[e] > val(0) && d[u] + cs[e] < d[to[e]] - eps) {
a694 ▹ ▹ ▹ ▹ d[to[e]] = d[u] + cs[e]; pai[to[e]] = e ^ 1;
1889 ▹ ▹ ▹ ▹ if(seen[to[e]] < tempo) { seen[to[e]] = tempo; qu[b++] = to[e]; if(b == N) b = 0; }
cbb1 ▹ ▹ ▹ }
cbb1 ▹ ▹ }
5c28 ▹ ▹ if(a == b) return false;
02be ▹ ▹ val mn = numeric_limits<val>::max();
1fd8 ▹ ▹ tempo++;
695a ▹ ▹ for(; seen[u] != tempo; u = to[pai[u]]) seen[u] = tempo;
e539 ▹ ▹ for(int v = u; seen[v] != tempo + 1; v = to[pai[v]]) {
ff98 ▹ ▹ ▹ seen[v] = tempo + 1;
3225 ▹ ▹ ▹ mn = min(mn, cp[pai[v] ^ 1] - fl[pai[v] ^ 1]);
cbb1 ▹ ▹ }
c141 ▹ ▹ for(int v = u; seen[v] == tempo + 1; v = to[pai[v]]) {
7618 ▹ ▹ ▹ seen[v] = 0;
60f1 ▹ ▹ ▹ fl[pai[v]] -= mn;
0329 ▹ ▹ ▹ fl[pai[v] ^ 1] += mn;
cbb1 ▹ ▹ }
8a6c ▹ ▹ return true;
cbb1 ▹ }
2b0e ▹ val max_flow(int s, int t) {
e7a0 ▹ ▹ val fl = 0;
036d ▹ ▹ while (bfs(s, t)) for(val f; (f = dfs(s, t, numeric_limits<val>::max())); fl += f);
e508 ▹ ▹ return fl;
cbb1 ▹ }
d41d ▹ // public $
8662 ▹ num min_cost(int s, int t) {
94a7 ▹ ▹ flow = max_flow(s, t);
6c9f ▹ ▹ while(spfa());
ed25 ▹ ▹ tot = 0;
112e ▹ ▹ for(int i = 2; i < en; i++)
b951 ▹ ▹ ▹ if(fl[i] > 0)
dae8 ▹ ▹ ▹ ▹ tot += fl[i] * cs[i];
126a ▹ ▹ return tot;
cbb1 ▹ }
8537 ▹ void reset_flow() { memset(fl, 0, sizeof(val) * en); }
457a ▹ void add_edge(int u, int v, val c, num s) {
d321 ▹ ▹ fl[en] = 0; cp[en] = c; to[en] = v; nx[en] = hd[u]; cs[en] = s; hd[u] = en++;
f081 ▹ ▹ fl[en] = 0; cp[en] = 0; to[en] = u; nx[en] = hd[v]; cs[en] = -s; hd[v] = en++;
cbb1 ▹ }
bfc4 ▹ void init(int n) { this->n = n; en = 2; memset(hd, 0, sizeof(int) * n); } // XXX must be called
2145 };
```

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 {
2f6a ▹ int n, MA[N], MB[N], PB[N], mn[N], st[N], sn; bool S[N], T[N];
```

```
6cc1     num c[N][N], d[N], y[N], z[N];
cd49     bool increase(int b) {
03dd         for (int a = PB[b];;) {
9ae2             int n_b = MA[a];
1ba8             MB[b] = a; MA[a] = b;
8f2f             if(n_b == -1) break;
5af0             b = n_b; a = PB[b];
cbb1         }
8a6c         return true;
cbb1     }
3a3b     bool visit(int a) {
cdb1         S[a] = true;
f580         for(int b = 0; b < n; b++) {
367c             if(T[b]) continue;
e782             if(c[a][b] - y[a] - z[b] < d[b] - eps) { d[b] = c[a][b] - y[a] - z[b]; mn[b] = a; }
3f25             if(c[a][b] - eps <= y[a] + z[b]) {
b46d                 T[b] = true; PB[b] = a; st[sn++] = b;
f8ab                 if(MB[b] == -1) return increase(b);
cbb1             }
cbb1         }
d1fe         return false;
cbb1     }
415c     bool update_dual() {
2f63         int mb = -1, b; num e;
f135         for(b = 0; b < n; b++) if(!T[b] && (mb == -1 || d[b] < d[mb])) mb = b;
04ff         for(e = d[mb], b = 0; b < n; b++)
3c42             if(T[b]) z[b] -= e;
6435             else d[b] -= e;
a915         for(int a = 0; a < n; a++)
cbbc             if(S[a]) y[a] += e;
eabc         PB[mb] = mn[mb];
7dcf         if(MB[mb] == -1) return increase(mb);
e309         st[sn++] = mb; T[mb] = true;
d1fe         return false;
cbb1     }
c4db     void find_path() {
2cc3         int a; for(a = 0; MA[a] != -1; a++);
0351         memset(S, 0, sizeof S); memset(T, 0, sizeof T);
e0c6         for(int i = 0; i < N; i++) d[i] = numeric_limits<num>::max();
7160         sn = 0; if(visit(a)) return;
6679         while(true) {
1f3f             if(sn) { if(visit(MB[st[--sn]])) break; }
6656             else if(update_dual()) break;
cbb1         }
cbb1     }
7e1e     void reset_all() {
52b4         for(int i = 0; i < n; i++) { y[i] = *min_element(c[i], c[i] + n); z[i] = 0; }
e517         for(int i = 0; i < n; i++) MA[i] = MB[i] = -1;
cbb1     }
d41d     // public $
957f     num min_match() { // set n and c then call this function
b989         reset_all(); num all = 0;
c13f         for(int i = 0; i < n; i++) find_path();
fe8e         for(int a = 0; a < n; a++) all += c[a][MA[a]];
64a8         return all;
cbb1     }
2145 };
```

3 Structures

3.1 Ordered Set

```
7747 #include <ext/pb_ds/assoc_container.hpp>
30f4 #include <ext/pb_ds/tree_policy.hpp>
0d73 using namespace __gnu_pbds;
4519 template <typename tA, typename tB=null_type> using ord_set = tree<tA, tB, less<tA>, rb_tree_tag,
     tree_order_statistics_node_update>;
```

```
d41d // map: tA -> tB com comparador less<tA>
d41d // pode usar como um map normalmente
d41d // s.find_by_order(k) :: retorna iterador para o k-esimo elemento (0-index) (ou s.end())
d41d // 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];
8b25 void calc (int u) { // update node given children info
d4c7 ▹ sz[u] = sz[L[u]] + 1 + sz[R[u]];
d41d ▹ // code here, no recursion
cbb1 }
234f void unlaze (int u) {
e39f ▹ if(!u) return;
d41d ▹ // code here, no recursion
cbb1 }
ee5e void split_val(int u, num x, int &l, int &r) { // l gets <= x, r gets > x
754f ▹ unlaze(u); if(!u) return (void) (l = r = 0);
4bc1 ▹ if(X[u] <= x) { split_val(R[u], x, l, r); R[u] = l; l = u; }
81a7 ▹ else { split_val(L[u], x, l, r); L[u] = r; r = u; }
aaa8 ▹ calc(u);
cbb1 }
9374 void split_sz(int u, int s, int &l, int &r) { // l gets first s, r gets remaining
754f ▹ unlaze(u); if(!u) return (void) (l = r = 0);
e06d ▹ if(sz[L[u]] < s) { split_sz(R[u], s - sz[L[u]] - 1, l, r); R[u] = l; l = u; }
f524 ▹ else { split_sz(L[u], s, l, r); L[u] = r; r = u; }
aaa8 ▹ calc(u);
cbb1 }
c870 int merge(int l, int r) { // els on l <= els on r
67f0 ▹ unlaze(l); unlaze(r); if(!l || !r) return l + r; int u;
7801 ▹ if(Y[l] > Y[r]) { R[l] = merge(R[l], r); u = l; }
ae90 ▹ else { L[r] = merge(l, L[r]); u = r; }
0ffd ▹ calc(u); return u;
cbb1 }
500b void init(int n=N-1) { // XXX call before using other funcs
7d1c ▹ for(int i = en = 1; i <= n; i++) { Y[i] = i; sz[i] = 1; L[i] = R[i] = 0; }
8c5a ▹ random_shuffle(Y + 1, Y + n + 1);
cbb1 }
```

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 {
5e0f ▹ deque<line> q; num lo,hi; envelope (num _lo, num _hi) : lo(_lo), hi(_hi) {}
01ca ▹ void push_front (line l) { // amort. O(inter) | l is best at lo or never
a86b ▹ ▹ if (q.size() && q[0](lo) < l(lo)) return;
89b8 ▹ ▹ for (num x; q.size(); q.pop_front()) {
cc18 ▹ ▹ ▹ x = (q.size()<=1?hi:q[0].inter(q[1],lo,hi)-1); // XXX double (-1)
4202 ▹ ▹ ▹ if (l(x) > q[0](x)) break;
cbb1 ▹ ▹ }
45bc ▹ ▹ q.push_front(l);
cbb1 ▹ }
f644 ▹ void push_back (line l) { // amort. O(inter) | l is best at hi or never
0334 ▹ ▹ if (q.size() && q[q.size()-1](hi) <= l(hi)) return;
b71c ▹ ▹ for (num x; q.size(); q.pop_back()) {
4e80 ▹ ▹ ▹ x = (q.size()<=1?lo:q[q.size()-2].inter(q[q.size()-1],lo,hi));
1747 ▹ ▹ ▹ if (l(x) >= q[q.size()-1](x)) break;
cbb1 ▹ ▹ }
5e56 ▹ ▹ q.push_back(l);
cbb1 ▹ }
e732 ▹ void pop_front (num _lo) { for (lo=_lo; q.size()>1 && q[0](lo) > q[1](lo); q.pop_front()); } // amort. O(n)
218a ▹ 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)
7155 ▹ line get (num x) { // O(lg(R))
```

```
e32f         int lo, hi, md; for (lo = 0, hi = q.size()-1, md = (lo+hi)/2; lo < hi; md = (lo+hi)/2)
c1fb             if (q[md](x) > q[md+1](x)) { lo = md+1; }
b029             else { hi = md; }
adf9         return q[lo];
cbb1     }
2145 };
b3a6 struct line { // inter = O(1)
7bd4     num a,b; num operator () (num x) const { return a*x+b; }
2417     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)); }
2145 };
16ed struct generic_line { // inter = O(lg(R))
7bd4     num a,b; num operator () (num x) const { return a*x+b; }
3cfe     num inter (generic_line o, num lo, num hi) { // first point where o strictly beats this
ca4f         for (num md = lo+((++hi)-lo)/2; lo < hi; md = lo+(hi-lo)/2) { // XXX double
760b             if ((*this)(md)<=o(md)) { lo = md+1; } // XXX double
b029             else { hi = md; }
cbb1         }
2532         return lo;
cbb1     }
2145 };
11a2 template<typename line> struct full_envelope { // XXX ties are broken arbitrarily
85c9     vector<envelope<line> > v; full_envelope(envelope<line> c) : v({c}) {} // v.reserve(30);
6aed     void add (line l) { // amort. O(lg(n)*inter)
8cca         envelope<line> cur(v.back().lo,v.back().hi); cur.push_back(l);
bb4a         while (!v.empty() && v.back().q.size() <= cur.q.size()) {
ce29             deque<line> aux; swap(aux,cur.q); int i = 0, j = 0;
31d2             for (; i < aux.size(); i++) {
542d                 for (; j < v.back().q.size() && v.back().q[j](cur.hi) > aux[i](cur.hi); j++)
0015                     cur.push_back(v.back().q[j]);
70a1                 cur.push_back(aux[i]);
cbb1             }
a0e7             for (; j < v.back().q.size(); j++) cur.push_back(v.back().q[j]);
deff             v.pop_back();
cbb1         }
026e         v.push_back(cur);
cbb1     }
7155     line get (num x) { // O(lg(n)lg(R)) | pop_back/pop_front can optimize
9351         line a = v[0].get(x);
ad67         for (int i = 1; i < (int) v.size(); i++) {
bcbe             line b = v[i].get(x);
ad0f             if (b(x)<a(x)) a = b;
cbb1         }
3f53         return a;
cbb1     }
2145 };
```

3.4 Centroid

```
0eca vector<int> adj[N]; int cn_sz[N], n;
c864 vector<int> cn_chld[N]; int cn_dep[N], cn_dist[20][N]; // removable
ace4 void cn_setdist (int u, int p, int depth, int dist) { // removable
989e     cn_dist[depth][u] = dist;
59dd     for (int v : adj[u]) if (p != v && cn_sz[v] != -1) // sz = -1 marks processed centroid (not dominated)
4ce5         cn_setdist(v, u, depth, dist+1);
cbb1 }
e897 int cn_getsz (int u, int p) {
08c9     cn_sz[u] = 1;
59dd     for (int v : adj[u]) if (p != v && cn_sz[v] != -1)
b2f6         cn_sz[u] += cn_getsz(v,u);
37a9     return cn_sz[u];
cbb1 }
912c int cn_build (int u, int depth) {
28a0     int siz = cn_getsz(u,u); int w = u;
0168     do {
9847         u = w;
a786         for (int v : adj[u]) if (cn_sz[v] != -1 && cn_sz[v] < cn_sz[u] && cn_sz[v] + cn_sz[v] >= siz)
9a13             w = v;
```

```
06ba ▹ } while (u != w); // u becomes current centroid root
094e ▹ cn_setdist(u,u,depth,0); // removable, here you can iterate over all dominated tree
32c2 ▹ cn_sz[u] = -1; cn_dep[u] = depth;
5cff ▹ for (int v : adj[u]) if (cn_sz[v] != -1) {
1df5 ▹ ▹ int w = cn_build(v, depth+1);
2e31 ▹ ▹ cn_chld[u].pb(w); // removable
cbb1 ▹ }
03f4 ▹ return u;
cbb1 }
```

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];
c7d4 int C[N][2]; // {left, right} children
fc41 inline void calc(int u) { // update node given children info
5665 ▹ sz[u] = sz[C[u][0]] + 1 + sz[C[u][1]];
d41d ▹ // code here, no recursion
cbb1 }
93d8 inline void unlaze(int u) {
e39f ▹ if(!u) return;
a2c4 ▹ if(lzswp[u]) {
3550 ▹ ▹ swap(C[u][0], C[u][1]);
20b7 ▹ ▹ if(C[u][0]) lzswp[C[u][0]] ^= 1;
8917 ▹ ▹ if(C[u][1]) lzswp[C[u][1]] ^= 1;
53e1 ▹ ▹ lzswp[u] = 0;
cbb1 ▹ }
cbb1 }
0584 int rotate(int u, int dir) { // pulls C[u][dir] up to u and returns it
05db ▹ int v = C[u][dir];
5b77 ▹ swap(pp[v], pp[u]);
2116 ▹ C[u][dir] = C[v][!dir];
6c8a ▹ if(C[u][dir]) p[C[u][dir]] = u;
ed1d ▹ C[v][!dir] = u; p[v] = p[u];
b9c1 ▹ if(p[v]) C[p[v]][C[p[v]][1] == u] = v;
6967 ▹ p[u] = v; calc(u); calc(v);
6dc7 ▹ return v;
cbb1 }
3ca5 void unlz_back(int u) { if(!u) return; unlz_back(p[u]); unlaze(u); }
81a1 void splay(int u) { // pulls node u to root
c46d ▹ unlz_back(u);
bdd0 ▹ while(p[u]) {
2a84 ▹ ▹ int v = p[u], w = p[p[u]];
c76a ▹ ▹ int du = (C[v][1] == u);
448e ▹ ▹ if(!w) { rotate(v, du); assert(!p[u]); }
4e6b ▹ ▹ else {
d499 ▹ ▹ ▹ int dv = (C[w][1] == v);
4780 ▹ ▹ ▹ if(du == dv) { rotate(w, dv); assert(C[v][du] == u); rotate(v, du); }
e576 ▹ ▹ ▹ else { rotate(v, du); assert(C[w][dv] == u); rotate(w, dv); }
cbb1 ▹ ▹ }
cbb1 ▹ }
cbb1 }
a7c2 int find_sz(int u, int s) { // returns s-th node (0-index)
d9d5 ▹ unlaze(u);
3939 ▹ while(sz[C[u][0]] != s) {
da07 ▹ ▹ if(sz[C[u][0]] < s) { s -= sz[C[u][0]] + 1; u = C[u][1]; }
afa2 ▹ ▹ else u = C[u][0];
d9d5 ▹ ▹ unlaze(u);
cbb1 ▹ }
49a4 ▹ splay(u); return u;
cbb1 }
498d int new_node() {
a2cc ▹ int i = en++; assert(i < N);
bea5 ▹ pp[i] = C[i][0] = C[i][1] = p[i] = 0;
0db4 ▹ lzswp[i] = 0; sz[i] = 1; return i;
cbb1 }
c538 int access(int u) {
10c3 ▹ if(!u) return u;
```

```
6d13 ▹ splay(u);
f206 ▹ if(int v = C[u][1]) { p[v] = 0; pp[v] = u; C[u][1] = 0; }
aaa8 ▹ calc(u);
566b ▹ while(pp[u]) {
0068 ▹ ▹ int w = pp[u]; splay(w);
33f4 ▹ ▹ if(int v = C[w][1]) { p[v] = 0; pp[v] = w; }
db1d ▹ ▹ C[w][1] = u; p[u] = w; pp[u] = 0; calc(w); splay(u);
cbb1 ▹ }
03f4 ▹ return u;
cbb1 }
0782 int find_root(int u) { // root o u's tree
29bf ▹ access(u);
3980 ▹ while(C[u][0]) { unlaze(u = C[u][0]); }
c607 ▹ access(u); return u;
cbb1 }
4d88 int get_parent(int u) { // u's parent, rootify might change it
29bf ▹ access(u);
c6f1 ▹ if(!C[u][0]) return pp[u];
e123 ▹ unlaze(u = C[u][0]);
323c ▹ while(C[u][1]) unlaze(u = C[u][1]);
c607 ▹ access(u); return u;
cbb1 }
c63a void link(int u, int v) { // adds edge from u to v, v must be root
961c ▹ if(find_root(u) == find_root(v)) return;
78b9 ▹ access(u); access(v);
612a ▹ assert(C[v][0] == 0 && pp[v] == 0 && sz[v] == 1); // v must be root
8e1a ▹ C[u][1] = v; p[v] = u; calc(u);
cbb1 }
d41d // XXX cut + rootify require get_parent, cut unlinks u from parent, rootify makes u root
e166 void cut(int u) { access(u); assert(C[u][0]); p[C[u][0]] = 0; C[u][0] = 0; calc(u); }
1cea void rootify(int u) { access(u); lzswp[u] = 1; access(u); }
b59a void init() { en = 1; } // XXX initialize
```

3.6 Splay Tree

```
d41d //const int N = ;
d41d //typedef int num;
d41d
576f int en = 1;
37e4 int p[N], sz[N];
c7d4 int C[N][2]; // {left, right} children
abac num X[N];
d41d
d41d // atualize os valores associados aos nos que podem ser calculados a partir dos filhos
8b25 void calc(int u) {
5665 ▹ sz[u] = sz[C[u][0]] + 1 + sz[C[u][1]];
cbb1 }
d41d
d41d // Puxa o filho dir de u para ficar em sua posicao e o retorna
0584 int rotate(int u, int dir) {
05db ▹ int v = C[u][dir];
2116 ▹ C[u][dir] = C[v][!dir];
6c8a ▹ if(C[u][dir]) p[C[u][dir]] = u;
0928 ▹ C[v][!dir] = u;
c0a7 ▹ p[v] = p[u];
b9c1 ▹ if(p[v]) C[p[v]][C[p[v]][1] == u] = v;
136e ▹ p[u] = v;
aaa8 ▹ calc(u);
b6b0 ▹ calc(v);
6dc7 ▹ return v;
cbb1 }
d41d
d41d // Traz o no u a raiz
81a1 void splay(int u) {
bdd0 ▹ while(p[u]) {
2a84 ▹ ▹ int v = p[u], w = p[p[u]];
1a8a ▹ ▹ int du = C[v][1] == u;
e764 ▹ ▹ if(!w)
```

```
76c8 			rotate(v, du);
4e6b 		else {
d499 			int dv = (C[w][1] == v);
9b57 			if(du == dv) {
6c72 				rotate(w, dv);
76c8 				rotate(v, du);
9d97 			} else {
76c8 				rotate(v, du);
6c72 				rotate(w, dv);
cbb1 			}
cbb1 		}
cbb1 	}
cbb1 }
d41d
d41d // retorna um no com valor x, ou outro no se n foi encontrado (n eh floor nem ceiling)
8975 int find_val(int u, num x) {
93fe 	int v = u;
9a3d 	while(u && X[u] != x) {
766a 		v = u;
1b5b 		if(x < X[u]) u = C[u][0];
a73d 		else u = C[u][1];
cbb1 	}
3418 	if(!u) u = v;
6d13 	splay(u);
03f4 	return u;
cbb1 }
d41d
d41d // retorna o s-esimo no (0-indexed)
a7c2 int find_sz(int u, int s) {
3939 	while(sz[C[u][0]] != s) {
7ef0 		if(sz[C[u][0]] < s) {
2777 			s -= sz[C[u][0]] + 1;
6bdb 			u = C[u][1];
66d9 		} else u = C[u][0];
cbb1 	}
6d13 	splay(u);
03f4 	return u;
cbb1 }
d41d
d41d // junte duas splays, assume que elementos l <= elementos r
c870 int merge(int l, int r) {
db1b 	if(!l || !r) return l + r;
45ba 	while(C[l][1]) l = C[l][1];
bab4 	splay(l);
0258 	assert(!C[l][1]);
e3ec 	C[l][1] = r;
924c 	p[r] = l;
f046 	calc(l);
792f 	return l;
cbb1 }
d41d
d41d // Adiciona no x a splay u e retorna x
684a int add(int u, int x) {
e29c 	int v = 0;
9d2d 	while(u) v = u, u = C[u][X[x] >= X[u]];
f257 	if(v) { C[v][X[x] >= X[v]] = x; p[x] = v; }
0b6f 	splay(x);
ea56 	return x;
cbb1 }
d41d
d41d // chame isso 1 vez no inicio
ca2f void init() {
0cee 	en = 1;
cbb1 }
d41d
d41d // Cria um novo no
3e8b int new_node(num val) {
cecb 	int i = en++;
9c38 	assert(i < N);
```

```
9029     C[i][0] = C[i][1] = p[i] = 0;
02c8     sz[i] = 1;
4281     X[i] = val;
d9a5     return i;
cbb1 }
```

4 Strings

4.1 Suffix Tree

```
4623 namespace sf {
d41d // const int NS = ; const int N = * 2;
1506 int cn, cd, ns, en = 1, lst;
f48b string S[NS]; int si = -1;
08ad vector<int> sufn[N]; // sufn[si][i] no do sufixo S[si][i...]
3c9e struct node {
a322     int l, r, si, p, suf;
d3ca     map<char, int> adj;
499b     node() : l(0), r(-1), suf(0), p(0) {}
2a9f     node(int L, int R, int S, int P) : l(L), r(R), si(S), p(P) {}
a577     inline int len() { return r - l + 1; }
48b2     inline int operator[](int i) { return S[si][l + i]; }
9eae     inline int& operator()(char c) { return adj[c]; }
fbe2 } t[N];
ea71 inline int new_node(int L, int R, int S, int P) { t[en] = node(L, R, S, P); return en++; }
e33b void add_string(string s) {
9a02     s += '$'; S[++si] = s; sufn[si].resize(s.size() + 1); cn = cd = 0;
c5eb     int i = 0; const int n = s.size();
f90a     for(int j = 0; j < n; j++)
fb3e         for(; i <= j; i++) {
8d90             if(cd == t[cn].len() && t[cn](s[j])) { cn = t[cn](s[j]); cd = 0; }
465b             if(cd < t[cn].len() && t[cn][cd] == s[j]) {
c4d2                 cd++;
ce02                 if(j < s.size() - 1) break;
4e6b                 else {
aafd                     if(i) t[lst].suf = cn;
ac68                     for(; i <= j; i++) { sufn[si][i] = cn; cn = t[cn].suf; }
cbb1                 }
7ced             } else if(cd == t[cn].len()) {
0a2a                 sufn[si][i] = en;
0467                 if(i) t[lst].suf = en; lst = en;
aff4                 t[cn](s[j]) = new_node(j, n - 1, si, cn);
02c2                 cn = t[cn].suf; cd = t[cn].len();
9d97             } else {
f287                 int mid = new_node(t[cn].l, t[cn].l + cd - 1, t[cn].si, t[cn].p);
12ed                 t[t[cn].p](t[cn][0]) = mid;
5201                 if(ns) t[ns].suf = mid;
0467                 if(i) t[lst].suf = en; lst = en;
0a2a                 sufn[si][i] = en;
cb00                 t[mid](s[j]) = new_node(j, n - 1, si, mid);
7bfa                 t[mid](t[cn][cd]) = cn;
07fe                 t[cn].p = mid; t[cn].l += cd; cn = t[mid].p;
5967                 int g = cn? j - cd : i + 1; cn = t[cn].suf;
c197                 while(g < j && g + t[t[cn](S[si][g])].len() <= j) {
6fea                     cn = t[cn](S[si][g]); g += t[cn].len();
cbb1                 }
71c3                 if(g == j) { ns = 0; t[mid].suf = cn; cd = t[cn].len(); }
f90d                 else { ns = mid; cn = t[cn](S[si][g]); cd = j - g; }
cbb1             }
cbb1         }
cbb1     }
2145 };
```

4.2 Z-function

```
2a61 void Z(char s[], int n, int z[]) { // z[i] = |lcp(s,s[i..n])|
```

```
fc15 ▹ for(int i = 1, m = -1; i < n; i++) {
d69b ▹ ▹ z[i] = (m != -1 && m + z[m] >= i)?min(m + z[m] - i, z[i - m]):0;
8a63 ▹ ▹ while (i + z[i] < n && s[i + z[i]] == s[z[i]]) z[i]++;
bbe8 ▹ ▹ if (m == -1 || i + z[i] > m + z[m]) m = i;
cbb1 ▹ }
cbb1 }
```

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
1df6 ▹ for(int i = 0; i < n; i++) t[2 * i] = s[i];
9866 ▹ for(int i = 0; i < n - 1; i++) t[2 * i + 1] = 1; // XXX s should not contain 1
5d62 ▹ n = 2 * n - 1;
9038 ▹ for(int i = 0, m = -1; i < n; i++) {
b39f ▹ ▹ M[i] = 0;
e618 ▹ ▹ if (m != -1 && m + M[m] >= i) M[i] = min(m + M[m] - i, M[2 * m - i]);
09a9 ▹ ▹ for (; i + M[i] + 1 < n && i - M[i] - 1 >= 0 && t[i + M[i] + 1] == t[i - M[i] - 1]; M[i]++);
086c ▹ ▹ if (m == -1 || i + M[i] > m + M[m]) m = i;
cbb1 ▹ }
cbb1 }
```

5 Math

5.1 FFT

```
5f83 typedef complex<double> cpx; const double pi = acos(-1.0);
d41d // DFT if type = 1, IDFT if type = -1
d41d // If you are multiplying, remember to let EACH vector with n >= sum of degrees of both polys
d41d // n is required to be a power of 2
d822 void FFT(cpx v[], cpx ans[], int n, int type, int p[]) { // p[n]
e228 ▹ assert(!(n & (n - 1))); int i, sz, o; p[0] = 0;
6be6 ▹ for(i = 1; i < n; i++) p[i] = (p[i >> 1] >> 1) | ((i & 1)? (n >> 1) : 0); // repetition can be avoided
d48c ▹ for(i = 0; i < n; i++) ans[i] = v[p[i]];
abc7 ▹ for(sz = 1; sz < n; sz <<= 1) {
3d36 ▹ ▹ const cpx wn(cos(type * pi / sz), sin(type * pi / sz));
1728 ▹ ▹ for(o = 0; o < n; o += (sz << 1)) {
dfb7 ▹ ▹ ▹ cpx w = 1;
c854 ▹ ▹ ▹ for(i = 0; i < sz; i++) {
d1db ▹ ▹ ▹ ▹ const cpx u = ans[o + i], t = w * ans[o + sz + i];
3f57 ▹ ▹ ▹ ▹ ans[o + i] = u + t;
e817 ▹ ▹ ▹ ▹ ans[o + i + sz] = u - t;
d2cd ▹ ▹ ▹ ▹ w *= wn;
cbb1 ▹ ▹ ▹ }
cbb1 ▹ ▹ }
cbb1 ▹ }
cde2 ▹ if(type == -1) for(i = 0; i < n; i++) ans[i] /= n;
cbb1 }
```

5.2 Discrete FFT

```
c9bc inline num s_mod (ll x, ll p) {
02ae ▹ if (x >= p) return x-p;
6d8b ▹ else if (x < 0) return x += p;
ea56 ▹ return x;
cbb1 }
6554 num fexp (ll x, int e, num p) {
ef50 ▹ ll r = 1;
6244 ▹ for (; e; x = (x*x)%p, e >>= 1) if (e&1) r = (r*x)%p;
4c1f ▹ return r;
cbb1 }
55a7 void rou (int n, int p, num w[]) { // w[i] = (n-th root of unity of p)^i
df57 ▹ w[0] = 1; bool ok = 0;
c238 ▹ for (num i = 2; !ok && i < p; i++) {
```

```
1145 		ok = 1;
4d9f 		for (ll j = 2; ok && j*j <= p-1; j++)
155e 			if ((p-1)%j == 0)
8ee4 				ok = !(fexp(i,j,p) == 1 || fexp(i,(p-1)/j,p) == 1);
8fe3 		if (ok) w[1] = fexp(i,(p-1)/n,p);
cbb1 	}
1862 	assert(ok);
4dd2 	for (int i = 2; i <= n; i++)
6580 		w[i] = (ll(w[i-1])*w[1])%p;
cbb1 }
03fd void fft_finite (num v[], num ans[], int n, int type, num p, int pr[], num w[]) { // pr[n], w[n]
4794 	assert(!(n & (n-1)));
13c0 	rou(n,p,w); ll invn = fexp(n,p-2,p); // repetition can be avoided
b3e7 	if (type == -1) reverse(w, w+n+1);
4fc1 	pr[0] = 0;
a8cf 	for (int i = 1; i < n; i++) pr[i] = ((pr[i>>1] >> 1) | ((i&1)?(n>>1):0)); // repetition can be avoided
b514 	for (int i = 0; i < n; i++) ans[i] = v[pr[i]];
f5fd 	for (int sz = 1; sz < n; sz <<= 1) {
849c 		for (int o = 0; o < n; o += (sz<<1)) {
8873 			for (int i = 0; i < sz; i++) {
7a0c 				const num u = ans[o+i], t = (w[(n/sz/2)*i]*ans[o+sz+i])%p;
b7a6 				ans[o+i] = s_mod(u+t,p);
8881 				ans[o+i+sz] = s_mod(u-t,p);
cbb1 			}
cbb1 		}
cbb1 	}
f8e2 	if(type == -1) for(int i = 0; i < n; i++) ans[i] = (ans[i]*invn)%p;
cbb1 }
d41d
```

5.3 Linear System Solver

```
d41d //const int N = ;
d41d
46cc double a[N][N];
3793 double ans[N];
d41d
d41d // sum(a[i][j] * x_j) = a[i][n] para 0 <= i < n
d41d // guarda a resposta em ans e retorna o determinante de a
c42a double solve(int n) {
f99b 	double det = 1;
6033 	for(int i = 0; i < n; i++) {
0268 		int mx = i;
197a 		for(int j = i + 1; j < n; j++)
b83d 			if(abs(a[j][i]) > abs(a[mx][i]))
672f 				mx = j;
28c6 		if(i != mx) {
e83f 			swap_ranges(a[i], a[i] + n + 1, a[mx]);
0143 			det = -det;
cbb1 		}
997e 		if(abs(a[i][i]) < 1e-6); // singular matrix
2f40 		det *= a[i][i];
94fe 		for(int j = i + 1; j < n; j++) {
12fe 			for(int k = i + 1; k <= n; k++)
ea32 				a[j][k] -= (a[j][i] / a[i][i]) * a[i][k];
efbc 			a[j][i] = 0;
cbb1 		}
cbb1 	}
45bd 	for(int i = n - 1; i >= 0; i--) {
7634 		ans[i] = a[i][n];
197a 		for(int j = i + 1; j < n; j++)
9b00 			ans[i] -= a[i][j] * ans[j];
35e5 		ans[i] /= a[i][i];
cbb1 	}
7a32 	return det;
cbb1 }
```

5.4 Simplex

```
d41d //typedef long double dbl;
bec0 const dbl eps = 1e-6;
d41d //const int N = , M = ;
d41d
79ee struct simplex {
0643     int X[N], Y[M];
6b50     dbl A[M][N], b[M], c[N];
e268     dbl ans;
14e0     int n, m;
a00d     dbl sol[N];
d41d
c511     void pivot(int x,int y){
eb91         swap(X[y], Y[x]);
c057         b[x] /= A[x][y];
8300         for(int i = 0; i < n; i++)
7f61             if(i != y)
d311                 A[x][i] /= A[x][y];
3fa2         A[x][y] = 1. / A[x][y];
94f7         for(int i = 0; i < m; i++)
a325             if(i != x && abs(A[i][y]) > eps) {
6856                 b[i] -= A[i][y] * b[x];
f90a                 for(int j = 0; j < n; j++)
6739                     if(j != y)
8c78                         A[i][j] -= A[i][y] * A[x][j];
e112                 A[i][y] = -A[i][y] * A[x][y];
cbb1             }
8c7e         ans += c[y] * b[x];
8300         for(int i = 0; i < n; i++)
7f61             if(i != y)
bec1                 c[i] -= c[y] * A[x][i];
0997         c[y] = -c[y] * A[x][y];
cbb1     }
d41d
d41d     // maximiza sum(x[i] * c[i])
d41d     // sujeito a
d41d     //   sum(a[i][j] * x[j]) <= b[i] para 0 <= i < m (Ax <= b)
d41d     //   x[i] >= 0 para 0 <= i < n (x >= 0)
d41d     // (n variaveis, m restricoes)
d41d     // guarda a resposta em ans e retorna o valor otimo
59d9     dbl solve(int n, int m) {
1f59         this->n = n; this->m = m;
f1bf         ans = 0.;
b1c6         for(int i = 0; i < n; i++) X[i] = i;
3e36         for(int i = 0; i < m; i++) Y[i] = i + n;
6679         while(true) {
ee39             int x = min_element(b, b + m) - b;
988b             if(b[x] >= -eps)
c2be                 break;
49a2             int y = find_if(A[x], A[x] + n, [](dbl d) { return d < -eps; }) - A[x];
6f8c             if(y == n) throw 1; // no solution
7fb4             pivot(x, y);
cbb1         }
6679         while(true) {
f802             int y = max_element(c, c + n) - c;
b7b6             if(c[y] <= eps) break;
d6b5             int x = -1;
06d7             dbl mn = 1. / 0.;
94f7             for(int i = 0; i < m; i++)
5877                 if(A[i][y] > eps && b[i] / A[i][y] < mn)
832b                     mn = b[i] / A[i][y], x = i;
ff22             if(x == -1) throw 2; // unbounded
7fb4             pivot(x, y);
cbb1         }
d094         memset(sol, 0, sizeof(dbl) * n);
94f7         for(int i = 0; i < m; i++)
cff4             if(Y[i] < n)
09d7                 sol[Y[i]] = b[i];
```

```
ba75         return ans;
cbb1     }
2145 };
```

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) {
8ce4     if(l == r) return;
ee49     int m = (l + r) / 2;
b34b     tf(a, l, m);
ba7b     tf(a, m + 1, r);
1e28     for(int i = l; i <= m; i++)
95f6         a[m + 1 + (i - l)] += a[i];
cbb1 }
d41d
d41d // Inverse transforms a inplace (OR), initially l = 0, r = 2^n - 1
0772 void itf(num a[], int l, int r) {
8ce4     if(l == r) return;
ee49     int m = (l + r) / 2;
1e28     for(int i = l; i <= m; i++)
27fa         a[m + 1 + (i - l)] -= a[i];
5001     itf(a, l, m);
ebd6     itf(a, m + 1, r);
cbb1 }
```

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) {
9fae     if(l == r) return (void) (b[l] = a[l] * (__builtin_popcount(l) == k));
ee49     int m = (l + r) / 2;
6d7f     tf(k, a, b, l, m);
33d9     tf(k, a, b, m + 1, r);
1e28     for(int i = l; i <= m; i++)
a168         b[m + 1 + (i - l)] += b[i];
cbb1 }
d41d
d41d // Ranked mobius transform (transform above for all k)
29b2 void tf(int k, num a[], num b[K+1][1 << K]) {
2380     for(int i = 0; i <= k; i++)
0b85         tf(i, a, b[i], 0, (1 << k) - 1);
cbb1 }
d41d
d41d // Convolutes two transforms. c[j][i] = sum(a[g][i] * b[k - g][i]) for 0 <= g <= j
f78d void conv(int k, num a[K+1][1 << K], num b[K+1][1 << K], num c[K+1][1 << K]) {
55df     for(int j = 0; j <= k; j++)
9ccb         for(int i = 0; i < (1 << k); i++) {
5b1b             c[j][i] = 0;
06b2             for(int g = 0; g <= j; g++)
b8c7                 c[j][i] += a[g][i] * b[j - g][i];
cbb1         }
cbb1 }
d41d
```

```
d41d // Inverse of ranked mobius transform for k
0772 void itf(num a[], int l, int r) {
8ce4     if(l == r) return;
ee49     int m = (l + r) / 2;
1e28     for(int i = l; i <= m; i++)
27fa         a[m + 1 + (i - l)] -= a[i];
5001     itf(a, l, m);
ebd6     itf(a, m + 1, r);
cbb1 }
d41d
d41d // Inverse of ranked mobius transform for all k
dde8 void itf(int k, num a[K+1][1 << K], num b[]) {
85c1     for(int j = 0; j <= k; j++) {
2341         itf(a[j], 0, (1 << k) - 1);
252d         for(int i = 0; i < (1 << k); i++)
8363             if(__builtin_popcount(i) == j)
e738                 b[i] = a[j][i];
cbb1     }
cbb1 }
d41d
d41d // use when you want to calculate c[i] = sum (a[j] * b[k]) such that (j | k) == i and (j & k) = 0
d41d // example use (if the size of a and b is (1 << k))
d41d // tf(k, a, a_);
d41d // tf(k, b, b_);
d41d // conv(k, a_, b_, ans);
d41d // itf(k, ans, c);
d41d // the answer will now be stored in c
```

5.7 Miller-Rabin

```
a288 llu llrand() { llu a = rand(); a<<= 32; a+= rand(); return a;}
0a9c int is_probably_prime(llu n) {
8dbf    if (n <= 1) return 0;
2373    if (n <= 3) return 1;
7de1    llu s = 0, d = n - 1;
66b4    while (d % 2 == 0) {
90f4       d/= 2; s++;
cbb1    }
6b3a    for (int k = 0; k < 64; k++) {
12c0       llu a = (llrand() % (n - 3)) + 2;
dc17       llu x = exp_mod(a, d, n);
1181       if (x != 1 && x != n-1) {
f0ea          for (int r = 1; r < s; r++) {
708d             x = mul_mod(x, x, n);
61d9             if (x == 1)
bb30                return 0;
68b2             if (x == n-1)
c2be                break;
cbb1          }
34bc          if (x != n-1)
bb30             return 0;
cbb1       }
cbb1    }
6a55    return 1;
cbb1 }
```

5.8 Pollard-Rho

```
295a llu rho(llu n) {
6214     llu d, c = rand() % n, x = rand() % n, xx = x;
28f6     if (n % 2 == 0)
18b9         return 2;
0168     do {
23db         x = (mul_mod(x, x, n) + c) % n;
a315         xx = (mul_mod(xx, xx, n) + c) % n;
a315         xx = (mul_mod(xx, xx, n) + c) % n;
864b         d = gcd(val_abs(x - xx), n);
```

```
4d51    } while (d == 1);
be24    return d;
cbb1 }
4722 map <llu,int> F;
6b0e void factor(llu n) {
e7f6    if (n == 1)
505b        return;
1b21    if (is_probably_prime(n)) {
d64d        F[n]++;
505b        return;
cbb1    }
3462    llu d = rho(n);
3ad0    factor(d);
d6f6    factor(n/d);
cbb1 }
```

6 Old Solutions

6.1 Ceiling Function

```
2b74 #include <bits/stdc++.h>
ca41 using namespace std;
35b1 #define fst first
6507 #define snd second
ad11 typedef long long ll;
ff0b typedef pair<int, int> pii;
efe1 #define pb push_back
924e #define for_tests(t, tt) int t; scanf("%d", &t); for(int tt = 1; tt <= t; tt++)
5a83 const ll modn = 1000000007;
cbba inline ll mod(ll x) { return x % modn; }
d41d
c2b3 const int N = 112345;
3606 int L[N], R[N], v[N];
576f int en = 1;
d41d
7296 int add(int r, int x) {
9dc6    if(r == 0) {
d6f0       r = en++;
b557       v[r] = x;
4c1f       return r;
cbb1    }
8b66    if(x < v[r])
dba8       L[r] = add(L[r], x);
2954    else
bb73       R[r] = add(R[r], x);
4c1f    return r;
cbb1 }
d41d
9c15 string get_str(int r) {
239b    if(r == 0) return "";
2ec6    return "(" + get_str(L[r]) + "," + get_str(R[r]) + ")";
cbb1 }
d41d
0114 string s[112345];
d41d
e8d7 int main() {
762a    int n, k, i, j, x;
e459    scanf("%d %d", &n, &k);
3f35    for(i = 0; i < n; i++) {
caa4       int root = 0;
fb39       for(j = 0; j < k; j++) {
e456          scanf("%d", &x);
95b4          root = add(root, x);
cbb1       }
6c08       s[i] = get_str(root);
cbb1    }
60eb    sort(s, s + n);
```

```
b8ba    printf("%d\n", int(unique(s, s + n) - s));
cbb1 }
```

6.2 Secret Chamber at Mount Rushmore

```
2b74 #include <bits/stdc++.h>
ca41 using namespace std;
35b1 #define fst first
6507 #define snd second
ad11 typedef long long ll;
ff0b typedef pair<int, int> pii;
efe1 #define pb push_back
924e #define for_tests(t, tt) int t; scanf("%d", &t); for(int tt = 1; tt <= t; tt++)
5a83 const ll modn = 1000000007;
cbba inline ll mod(ll x) { return x % modn; }
d41d
a8d9 char adj[256][256];
38f1 char seen[256];
d41d
4674 void go(char p, char u) {
2ac0    if(seen[u] == p) return;
0886    seen[u] = p;
6c58    adj[p][u] = 1;
9d57    for(int v = 'a'; v <= 'z'; v++)
111a       if(adj[u][v])
1ac3          go(p, v);
cbb1 }
d41d
aba0 char s[1123], t[1123];
d41d
e8d7 int main() {
94c3    int i, m, n, j;
7676    scanf("%d %d", &m, &n);
cc5c    for(i = 0; i < m; i++) {
37b2       char a, b;
64f2       scanf(" %c %c", &a, &b);
d7dd       adj[a][b] = 1;
cbb1    }
419d    for(i = 'a'; i <= 'z'; i++)
c6b2       go(i, i);
3f35    for(i =0 ; i < n; i++) {
96b1       scanf("%s %s", s, t);
abae       if(strlen(s) != strlen(t)) { puts("no"); continue; }
41d3       for(j = 0; s[j]; j++)
a92d          if(!adj[s[j]][t[j]])
c2be             break;
85c9       if(s[j]) puts("no");
b8ef       else puts("yes");
cbb1    }
cbb1 }
```

6.3 Need for Speed

```
2b74 #include <bits/stdc++.h>
ca41 using namespace std;
35b1 #define fst first
6507 #define snd second
ad11 typedef long long ll;
ff0b typedef pair<int, int> pii;
efe1 #define pb push_back
924e #define for_tests(t, tt) int t; scanf("%d", &t); for(int tt = 1; tt <= t; tt++)
5a83 const ll modn = 1000000007;
cbba inline ll mod(ll x) { return x % modn; }
d41d
c49e const int N = 1123;
5319 int d[N], s[N];
d41d
```

```
e8d7 int main() {
a1d4    int n, t, i;
2f9a    scanf("%d %d", &n, &t);
a2ed    long double l = -2e7, r = 1502;
e668    for(i = 0; i < n; i++) scanf("%d %d", &d[i], &s[i]);
6c58    for(int x = 0; x < 200; x++) {
8591       long double c = (l + r) / 2;
cb4e       long double tot = 0;
3f35       for(i = 0; i < n; i++) {
31b2          long double ss = s[i] - c;
05c3          if(ss <= 0) break;
f7ab          tot += d[i] / ss;
cbb1       }
e2ae       if(tot >= t || i < n) r = c;
0399       else l = c;
cbb1    }
e11b    printf("%.10f\n", -double(l));
d41d
cbb1 }
```

6.4 Amalgamated Artichokes

```
2b74 #include <bits/stdc++.h>
ca41 using namespace std;
d41d
e8d7 int main() {
79eb    int p, a, b, c, d, n;
1a4f    scanf("%d %d %d %d %d %d", &p, &a, &b, &c, &d, &n);
97e4    double mx = -1. / 0.;
e9cc    double ans = 0;
78ac    for(int i = 1; i <= n; i++) {
3ab3       double x = p * (sin(a * i + b) + cos(c * i + d) + 2);
314d       mx = max(mx, x);
c387       ans = max(ans, mx - x);
cbb1    }
3ccd    printf("%.10f\n", ans);
cbb1 }
```

6.5 Low Power

```
2b74 #include <bits/stdc++.h>
ca41 using namespace std;
d41d
ad11 typedef long long ll;
70cf typedef pair<ll, ll> pii;
efe1 #define pb push_back
d41d
4d42 const int N = 1e6+7;
d41d
dca6 int n, k;
5e7a ll a[N];
d41d
91a0 bool solve (ll d) {
d179 ▹ ll s = 0, m = n;
22f9 ▹ for (int i = 0; m && i < 2*n*k - 1; i++) {
a7bf ▹ ▹ if (a[i+1] - a[i] <= d) {
76b4 ▹ ▹ ▹ m--;
0dd5 ▹ ▹ ▹ i++;
29bf ▹ ▹ ▹ s += 2*(k-1);
b579 ▹ ▹ } else if (!s) return 0;
5e99 ▹ ▹ else s--;
cbb1 ▹ }
6a55 ▹ return 1;
cbb1 }
d41d
e8d7 int main () {
e459 ▹ scanf("%d %d",&n, &k);
```

```
d41d
8575 ▹ for (int i = 0; i < 2*n*k; i++)
a20e ▹ ▹ scanf("%lld", &a[i]);
b8a4 ▹ sort(a, a+2*n*k);
d41d
99d3 ▹ ll lo = 0, hi = 1e9+2;
6e78 ▹ while (lo < hi) {
fda7 ▹ ▹ ll md = (lo+hi)/2;
ccff ▹ ▹ if (solve(md)) hi = md;
e47d ▹ ▹ else lo = md+1;
cbb1 ▹ }
d41d
161f ▹ printf("%lld\n", lo);
cbb1 }
```

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 $A: GX \to X$ 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

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