

ACM ICPC Reference

University of Notre Dame

October 24, 2019

Contents

1	vimrc	2
2	hashify.sh	2
3	STL	2
4	Geometry	2
4.1	Base	2
4.2	Advanced	4
4.3	3D	6
5	Graphs	7
5.1	Dinic	7
5.2	MinCost MaxFlow	8
5.3	Cycle Cancellation	8
6	Structures	10
6.1	Ordered Set	10
6.2	Treap	10
6.3	Envelope	10
6.4	Centroid	11
6.5	Splay Tree	12
7	Strings	14
7.1	Z-function	14
8	Math	14
8.1	Linear System Solver	14
8.2	Simplex	14
9	Number Theory	15
9.1	Extended Euclidean Algorithm	15
9.2	Miller-Rabin	16
10	Notes	16
10.1	Modular Multiplicative Inverse	16
10.2	Chinese Remainder Theorem	16
10.3	Euler's Totient Function	16
10.4	Möebius	16
10.5	Burnside	16
10.6	Landau	17
10.7	Erdős-Gallai	17
10.8	Gambler's Ruin	17
10.9	Extra	17

1 vimrc

```

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

```

2 hashify.sh

```

#!/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

```

3 STL

?

4 Geometry

4.1 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>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); }
elif > 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) });}
```

4.2 Advanced

```
484c cir min_spanning_circle (vec * v, int n) { // n
flea ▶ 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) { // nlq | 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) { // nlq | 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-1)>0?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) { // nlq | 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 = 1; i < r; i++) if (sq((v[i] = w[i]).x - x) < res) {
ad96 ▶ ▶ for (int j = s-1; j >= 1 && 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 szkopol, 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 > > > }

```

4.3 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 thetetrahedra
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))/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; } // tetrahedra
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 };

```

5 Graphs

5.1 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 > };

```

5.2 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 > };

```

5.3 Cycle Cancellation

```

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 > > }
dife > > 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 };
```

6 Structures

6.1 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
```

6.2 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 }
```

6.3 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(1);
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(1);
cbb1 > > }
e732 > void pop_front (num _lo) { for (lo=_lo; q.size()>1 && q[0](lo) > q[1](lo); q.pop_front()); } // amort.
0(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(1);
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 > > };

```

6.4 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 }

```

6.5 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 }

```

7 Strings

7.1 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 }
```

8 Math

8.1 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 }
```

8.2 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 };

```

9 Number Theory

9.1 Extended Euclidean Algorithm

```

c25f int egcd(int a, int b, int& x, int& y) { // a*x + b*y = gcd(a, b) [Bezout's Theorem]
8273 > if (b == 0) return x = 1, y = 0, a;
98d1 > int xx, yy;
0c0d > int g = egcd(b, a % b, xx, yy);
512d > x = yy;

```



```

a9d0 ▸ y = xx - (a / b) * yy;
96b5 ▸ return g;
cbb1 }

```

9.2 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 }

```

10 Notes

10.1 Modular Multiplicative Inverse

- If $\gcd(a, m) = 1$, then let $ax + my = \gcd(a, m) = 1$ (Bezout's Theorem). Then $ax \equiv 1 \pmod{m}$.
- If $\gcd(a, m) = 1$, then $a \cdot a^{\phi(m)-1} \equiv 1 \pmod{m}$ (Euler's Theorem).
- If m is prime, then $\phi(m) = m - 1$, so $a \cdot a^{m-2} \equiv 1 \pmod{m}$.

10.2 Chinese Remainder Theorem

We are given $N = n_1 n_2 \cdots n_k$ where n_i are pairwise coprime. We are also given $x_1 \cdots x_k$ such that $x \equiv x_i \pmod{n_i}$. Let $N_i = N/n_i$. There exists M_i and m_i such that $M_i N_i + m_i n_i = 1$ (Bezout). Then, there is only one solution x , given by:

$$x = \sum_{i=1}^k a_i M_i N_i$$

10.3 Euler's Totient Function

Positive integers up to a given integer n that are relatively prime to n . $\varphi(n) = n \prod_{p|n} \left(1 - \frac{1}{p}\right)$ where the product is over the distinct prime numbers dividing n .

10.4 Möebius

If $F(n) = \sum_{d|n} f(d)$, then $f(n) = \sum_{d|n} \mu(d) F(n/d)$.

10.5 Burnside

Let $A: GX \rightarrow X$ be an action. Define:

- $w :=$ number of orbits in X .

- $S_x := \{g \in G \mid g \cdot x = x\}$
- $F_g := \{x \in X \mid g \cdot x = x\}$

Then $w = \frac{1}{|G|} \sum_{x \in X} |S_x| = \frac{1}{|G|} \sum_{g \in G} |F_g|$.

10.6 Landau

There is a tournament with outdegrees $d_1 \leq d_2 \leq \dots \leq d_n$ iff:

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

In order to build it, let 1 point to $2, 3, \dots, d_1 + 1$ and repeat recursively.

10.7 Erdős-Gallai

There is a simple graph with degrees $d_1 \geq d_2 \geq \dots \geq d_n$ iff:

- $d_1 + d_2 + \dots + d_n$ is even
- $\sum_{i=1}^k d_i \leq k(k-1) + \sum_{i=k+1}^n \min(d_i, k) \quad \forall 1 \leq k \leq n.$

In order to build it, connect 1 with $2, 3, \dots, d_1 + 1$ and repeat recursively.

10.8 Gambler's Ruin

In a game in which we win a coin with probability p and lose a coin with probability $q := 1 - p$, the game stops when we win B or lose A coins. Then $\text{Prob}(\text{win B}) = \frac{1-(p/q)^B}{1-(p/q)^{A+B}}$.

10.9 Extra

- $\text{Fib}(x+y) = \text{Fib}(x+1)\text{Fib}(y) + \text{Fib}(x)\text{Fib}(y-1)$