ICPC Notebook

pedroteosousa

Contents

1 Geometry

1.1 Integer Basic

```cpp
#define int long long

bool zero(int x) {
        return x == 0;
}

// CORNER: point = (0, 0)
struct point {
        int x, y;

        point(int x=0, int y=0): x(x), y(y) {}

        point operator+(point rhs) { return point(x+rhs.x, y+rhs.y); }
        point operator-(point rhs) { return point(x-rhs.x, y-rhs.y); }
        int operator*(point rhs) { return x*rhs.x + y*rhs.y; }
        int operator^(point rhs) { return x*rhs.y - y*rhs.x; }

        int norm2() { return *this * *this; }

        using tup = tuple<int, int>;

        bool operator<(const point& rhs) const {
                return tup{x, y} < tup{rhs.x, rhs.y};
        }

        bool operator==(const point& rhs) const {
                return tup{x, y} == tup{rhs.x, rhs.y};
        }
};

// angular comparison in [0, 2pi)
// smallest is (1, 0)
// CORNER: a || b == (0, 0)
bool ang_cmp(point a, point b) {
        auto quad = [](point p) -> bool {
                // 0 if ang in [0, pi), 1 if in [pi, 2pi)
                return p.y < 0 || (p.y == 0 && p.x < 0);
        };
        using tup = tuple<bool, int>;
        return tup{quad(a), 0} < tup{quad(b), a^b};
}

int dist2(point p, point q) { // squared distance
    return (p - q)*(p - q);
}
```

```
int area2(point a, point b, point c) { // two times signed area of triangle abc
        return (b - a) ^ (c - a);
}

bool left(point a, point b, point c) {
        return area2(a, b, c) > 0; // counterclockwise
}

bool right(point a, point b, point c) {
        return area2(a, b, c) < 0; // clockwise
}

bool collinear(point a, point b, point c) {
        return zero(area2(a,b,c));
}

// CORNER: a || b == (0, 0)
int parallel(point a, point b) {
        if((a ^ b) != 0) return 0;
        return (a.x>0) == (b.x>0) && (a.y > 0) == (b.y > 0) ? 1 : -1;
}

// CORNER: a == b
struct segment {
        point a, b;

        segment(point a=point(), point b=point()): a(a), b(b) {}

        point v() { return b - a; }
};

bool contains(segment r, point p) {
        return r.a==p || r.b==p || parallel(r.a-p,r.b-p) == -1;
}

bool intersects(segment r, segment s) {
        if(contains(r, s.a) || contains(r, s.b) || contains(s, r.a) || contains(s, r.b)) return 1;
        return left(r.a,r.b,s.a) != left(r.a,r.b,s.b) &&
                left(s.a, s.b, r.a) != left(s.a, s.b, r.b);
}

bool parallel(segment r, segment s) {
        return parallel(r.v(), s.v());
}
```