Finding Intersection of Two Segments

Table of Contents

You are given two segments AB and CD, described as pairs of their endpoints. Each segment can be a single point if its endpoints are the same. You have to find the intersection of these segments, which can be empty (if the segments don't intersect), a single point or a segment (if the given segments overlap).

Solution

We can find the intersection point of segments in the same way as the intersection of lines: reconstruct line equations from the segments' endpoints and check whether they are parallel.

If the lines are not parallel, we need to find their point of intersection and check whether it belongs to both segments (to do this it's sufficient to verify that the intersection point belongs to each segment projected on

X and Y axes). In this case the answer will be either "no intersection" or the single point of lines' intersection.

The case of parallel lines is slightly more complicated (the case of one or more segments being a single point also belongs here). In this case we need to check that both segments belong to the same line. If they don't, the answer is "no intersection". If they do, the answer is the intersection of the segments belonging to the same line, which is obtained by ordering the endpoints of both segments in the increasing order of certain coordinate and taking the rightmost of left endpoints and the leftmost of right endpoints.

If both segments are single points, these points have to be identical, and it makes sense to perform this check separately.

In the beginning of the algorithm let's add a bounding box check - it is necessary for the case when the segments belong to the same line, and (being a lightweight check) it allows the algorithm to work faster on average on random tests.

Implementation

Here is the implementation, including all helper functions for lines and segments processing.

The main function `intersect` returns true if the segments have a non-empty intersection, and stores endpoints of the intersection segment in arguments `left`

and `right`. If the answer is a single point, the values written to `left` and `right` will be the same.

```
const double EPS = 1E-9;

struct pt {
    double x, y;

    bool operator<(const pt& p) const
    {
        return x < p.x - EPS || (abs(x - p.x)
    }
};

struct line {
    double a, b, c;

    line() {}
    line(pt p, pt q)
    {
        a = p.y - q.y;
        b = q.x - p.x;
        c = -a * p.x - b * p.y;
        norm();
    }

    void norm()
    {
        double z = sqrt(a * a + b * b);
        if (abs(z) > EPS)
            a /= z, b /= z, c /= z;
    }
```

```
    double dist(pt p) const { return a * p.x +
};

double det(double a, double b, double c, doubl
{
    return a * d - b * c;
}

inline bool betw(double l, double r, double x)
{
    return min(l, r) <= x + EPS && x <= max(l,
}

inline bool intersect_1d(double a, double b, d
{
    if (a > b)
        swap(a, b);
    if (c > d)
        swap(c, d);
    return max(a, c) <= min(b, d) + EPS;
}

bool intersect(pt a, pt b, pt c, pt d, pt& lef
{
    if (!intersect_1d(a.x, b.x, c.x, d.x) || !
        return false;
    line m(a, b);
    line n(c, d);
    double zn = det(m.a, m.b, n.a, n.b);
    if (abs(zn) < EPS) {
        if (abs(m.dist(c)) > EPS || abs(n.dist
            return false;
```

```
        if (b < a)
            swap(a, b);
        if (d < c)
            swap(c, d);
        left = max(a, c);
        right = min(b, d);
        return true;
    } else {
        left.x = right.x = -det(m.c, m.b, n.c,
        left.y = right.y = -det(m.a, m.c, n.a,
        return betw(a.x, b.x, left.x) && betw(
               betw(c.x, d.x, left.x) && betw(
    }
}
```