Check if two segments intersect

Table of Contents

You are given two segments (a, b) and (c, d). You have to check if they intersect. Of course, you may find their intersection and check if it isn't empty, but this can't be done in integers for segments with integer coordinates. The approach described here can work in integers.

Algorithm

Firstly, consider the case when the segments are part of the same line. In this case it is sufficient to check if their projections on Ox and Oy intersect. In the other case a and b must not lie on the same side of line (c, d), and c and d must not lie on the same side of line (a, b). It can be checked with a couple of cross products.

Implementation

The given algorithm is implemented for integer points. Of course, it can be easily modified to work with doubles.

```
struct pt {
    long long x, y;
    pt() {}
    pt(long long _x, long long _y) : x(_x), y(_y) {}
    pt operator-(const pt& p) const { return pt(x - p.x, y - p.y); }
    long long cross(const pt& p) const { return x * p.y - y * p.x; }
    long long cross(const pt& a, const pt& b) const { return (a - *this).cross(b - *this); }
};

int sgn(const long long& x) { return x >= 0 ? x ? 1 : 0 : -1; }

bool inter1(long long a, long long b, long long c, long long d) {
    if (a > b)
        swap(a, b);
    if (c > d)
        swap(c, d);
    return max(a, c) <= min(b, d);
}

bool check_inter(const pt& a, const pt& b, const pt& c, const pt& d) {
    if (c.cross(a, d) == 0 && c.cross(b, d) == 0)
        return inter1(a.x, b.x, c.x, d.x) && inter1(a.y, b.y, c.y, d.y);
    return sgn(a.cross(b, c)) != sgn(a.cross(b, d)) &&
           sgn(c.cross(d, a)) != sgn(c.cross(d, b));
}
```