

```

186 Search for a pair of intersecting segments (sweep line algorithm):
187 //O(n log(n))
188 const double EPS = 1E-9;
189
190 struct pt {
191     double x, y;
192 };
193
194 struct seg {
195     pt p, q;
196     int id;
197
198     double get_y(double x) const {
199         if (abs(p.x - q.x) < EPS)
200             return p.y;
201         return p.y + (q.y - p.y) * (x - p.x) / (q.x - p.x);
202     }
203 };
204
205 bool intersect1d(double l1, double r1, double l2, double r2) {
206     if (l1 > r1)
207         swap(l1, r1);
208     if (l2 > r2)
209         swap(l2, r2);
210     return max(l1, l2) ≤ min(r1, r2) + EPS;
211 }
212
213 int vec(const pt& a, const pt& b, const pt& c) {
214     double s = (b.x - a.x) * (c.y - a.y) - (b.y - a.y) * (c.x - a.x);
215     return abs(s) < EPS ? 0 : s > 0 ? +1 : -1;
216 }
217
218 bool intersect(const seg& a, const seg& b)
219 {
220     return intersect1d(a.p.x, a.q.x, b.p.x, b.q.x) &&
221         intersect1d(a.p.y, a.q.y, b.p.y, b.q.y) &&
222         vec(a.p, a.q, b.p) * vec(a.p, a.q, b.q) ≤ 0 &&
223         vec(b.p, b.q, a.p) * vec(b.p, b.q, a.q) ≤ 0;
224 }
225
226 bool operator<(const seg& a, const seg& b)
227 {
228     double x = max(min(a.p.x, a.q.x), min(b.p.x, b.q.x));
229     return a.get_y(x) < b.get_y(x) - EPS;
230 }
231
232 struct event {
233     double x;
234     int tp, id;
235
236     event() {}
237     event(double x, int tp, int id) : x(x), tp(tp), id(id) {}
238
239     bool operator<(const event& e) const {
240         if (abs(x - e.x) > EPS)
241             return x < e.x;
242         return tp > e.tp;
243     }
244 };

```