

```

246 set<seg> s;
247 vector<set<seg>::iterator> where;
248
249 set<seg>::iterator prev(set<seg>::iterator it) {
250     return it == s.begin() ? s.end() : --it;
251 }
252
253 set<seg>::iterator next(set<seg>::iterator it) {
254     return ++it;
255 }
256
257 //use this
258 pair<int, int> solve(const vector<seg>& a) {
259     int n = (int)a.size();
260     vector<event> e;
261     for (int i = 0; i < n; ++i) {
262         e.push_back(event(min(a[i].p.x, a[i].q.x), +1, i));
263         e.push_back(event(max(a[i].p.x, a[i].q.x), -1, i));
264     }
265     sort(e.begin(), e.end());
266
267     s.clear();
268     where.resize(a.size());
269     for (size_t i = 0; i < e.size(); ++i) {
270         int id = e[i].id;
271         if (e[i].tp == +1) {
272             set<seg>::iterator nxt = s.lower_bound(a[id]), prv = prev(nxt);
273             if (nxt != s.end() && intersect(*nxt, a[id]))
274                 return make_pair(nxt->id, id);
275             if (prv != s.end() && intersect(*prv, a[id]))
276                 return make_pair(prv->id, id);
277             where[id] = s.insert(nxt, a[id]);
278         } else {
279             set<seg>::iterator nxt = next(where[id]), prv = prev(where[id]);
280             if (nxt != s.end() && prv != s.end() && intersect(*nxt, *prv))
281                 return make_pair(prv->id, nxt->id);
282             s.erase(where[id]);
283         }
284     }
285
286     return make_pair(-1, -1);
287 }

```