ETH Zurich
Institut für Theoretische Informatik
Prof. Dr. Angelika Steger
Prof. Dr. Emo Welzl
Prof. Dr. Peter Widmayer

# Algorithms Lab

## Exercise – *Hit?*

A laser source shot Phileas Photon into some direction. Will he stop at an obstacle or travel to infinity?

The scenery is considered as seen from above, such that obstacle walls appear as line segments and the trajectory of Phileas is described by a ray.

**Input** The input file consists of several test cases. Each of them starts with a line containing one integer $n$ ($1 \leq n \leq 20'000$). The following line describes the ray along which Phileas travels. It contains integers `x y a b` where $(x, y)$ are the coordinates of the starting point of the ray and $(a, b) \neq (x, y)$ is another point on the ray. The following $n$ lines describe an obstacle line segment each. The $i$-th of these lines contains four integers `r s t u`, where $(r, s)$ and $(t, u)$ are the endpoints of the $i$-th segment. All the above coordinates are integers that are in absolute value smaller than $2^{51}$. In particular, you cannot represent them using a 32-bit integer data type in general. All numbers on a single line are separated by a single space. The input is terminated by a single line containing `0` (i.e., an empty testcase).

**Output** The output for each testcase appears on a separate line. The line consists of the word `yes`, if the ray hits an obstacle[1] and `no`, otherwise.

**Sample Input**

```
1
0 0 1 1
2 0 1 2
1
1 1 0 0
0 -2 -1 0
2
0 0 1 1
-1 -2 -1 0
2 0 2 1
2
0 1 1125899906842623 1125899906842623
1 2 1 3
1125899906842621 1125899906842620 1125899906842621 1125899906842621
3
1125899906842623 1125899906842623 0 1
1125899906842621 1125899906842620 1125899906842621 1125899906842621
1 2 1 3
-1 0 -1 1
0
```

[1] The obstacles segments are relatively closed, that is, both endpoints are included into consideration.

**Sample Output**

```
yes
yes
no
no
yes
```

## S 1: A very straightforward solution

**Time:** $O(n)$, **Space:** $O(1)$

This solution just goes through all the obstacles and checks whether any of them intersects with the ray.

```
#include <CGAL/Exact_predicates_exact_constructions_kernel.h>

typedef CGAL::Exact_predicates_exact_constructions_kernel K;
typedef K::Segment_2 S;
typedef K::Ray_2 R;
typedef K::Point_2 P;

bool solve(R ray, std::vector<S>& obstacles) {
  for (auto o: obstacles) {
    bool intersect = CGAL::do_intersect(ray, o);
    if (intersect) {
      return true;
    }
  }
  return false;
}
```