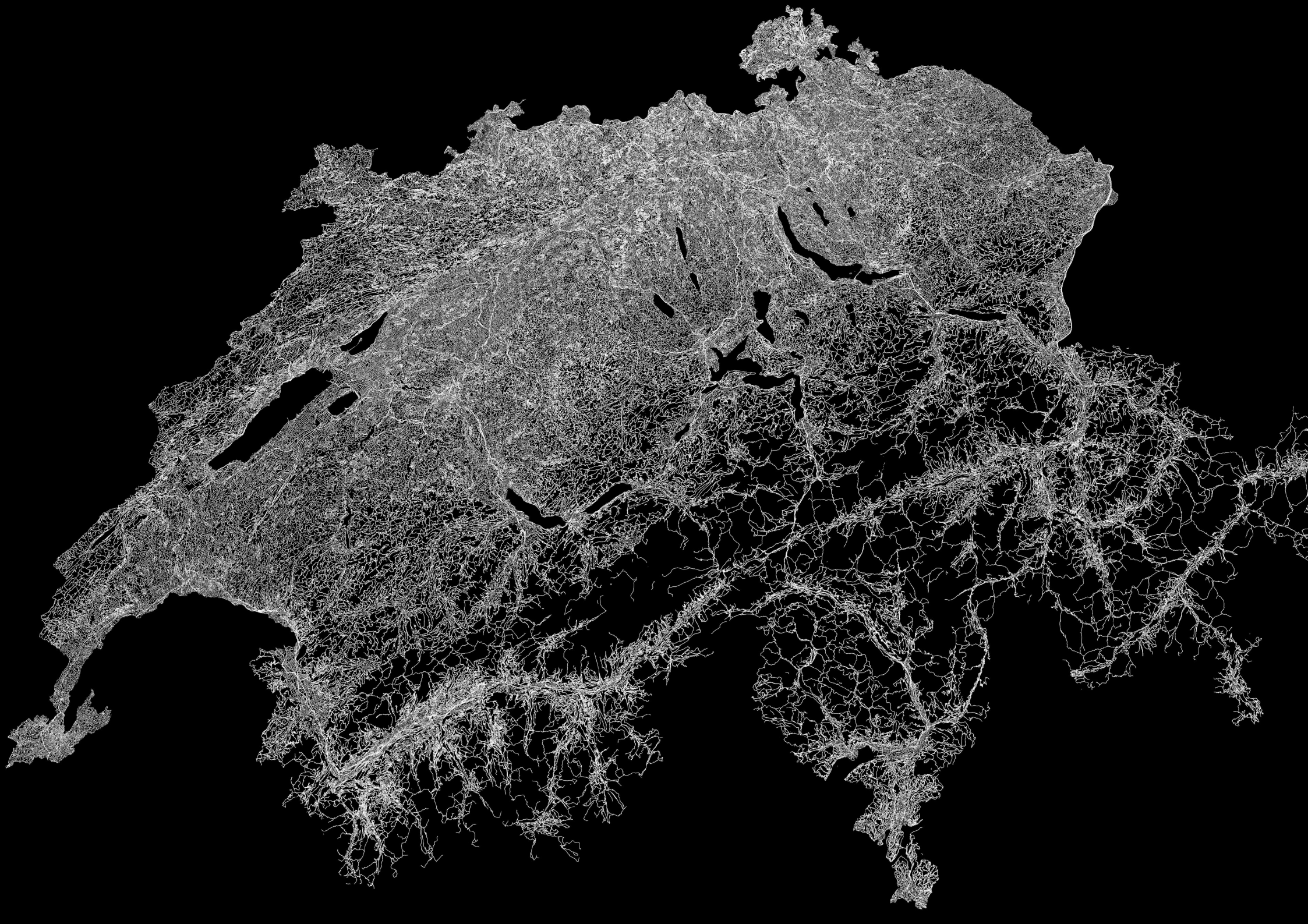


INTERSECTING LINE SEGMENTS

Rodrigo Silveira

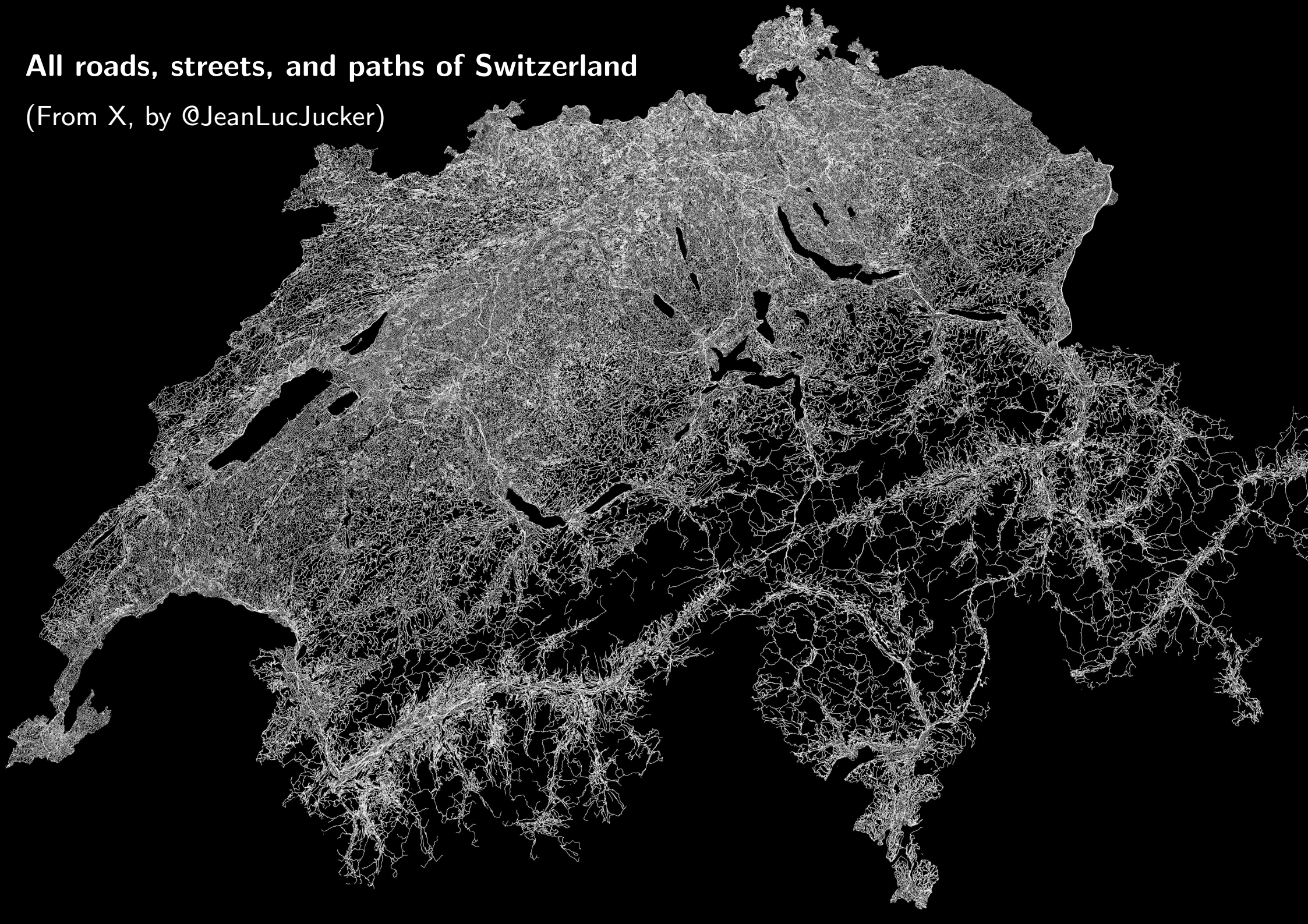
Based on slides by Vera Sacristán

Universitat Politècnica de Catalunya



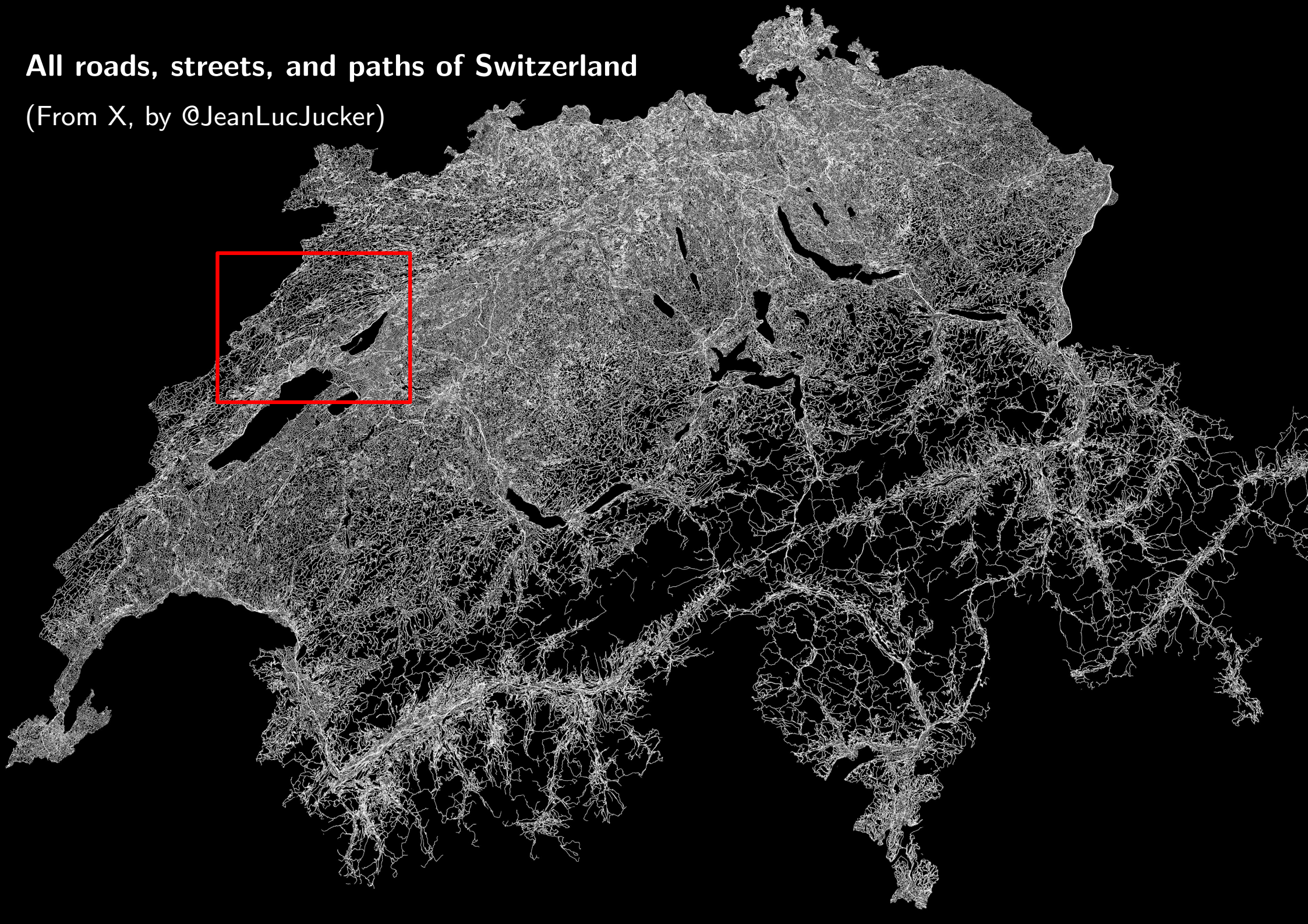
All roads, streets, and paths of Switzerland

(From X, by @JeanLucJucker)



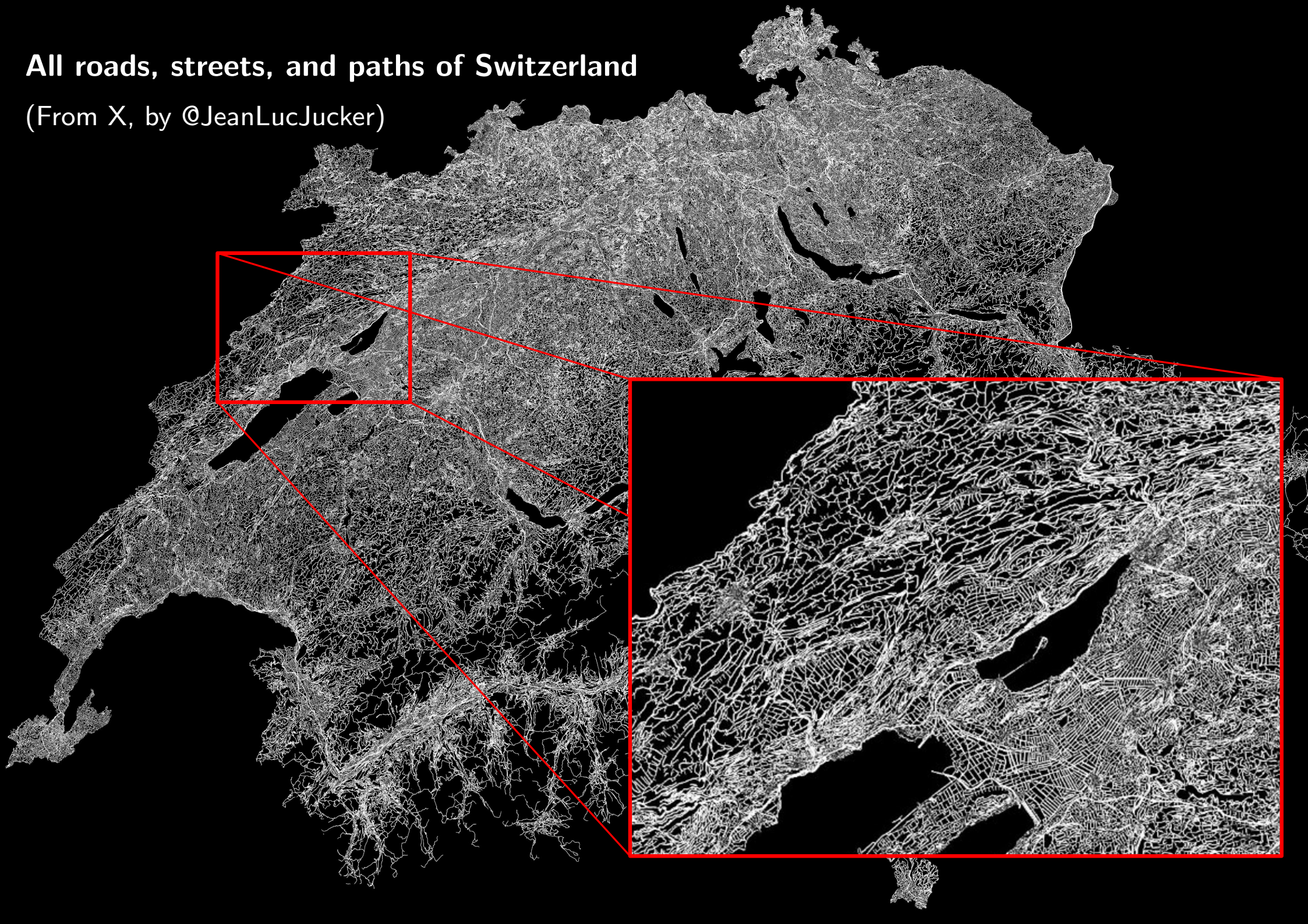
All roads, streets, and paths of Switzerland

(From X, by @JeanLucJucker)



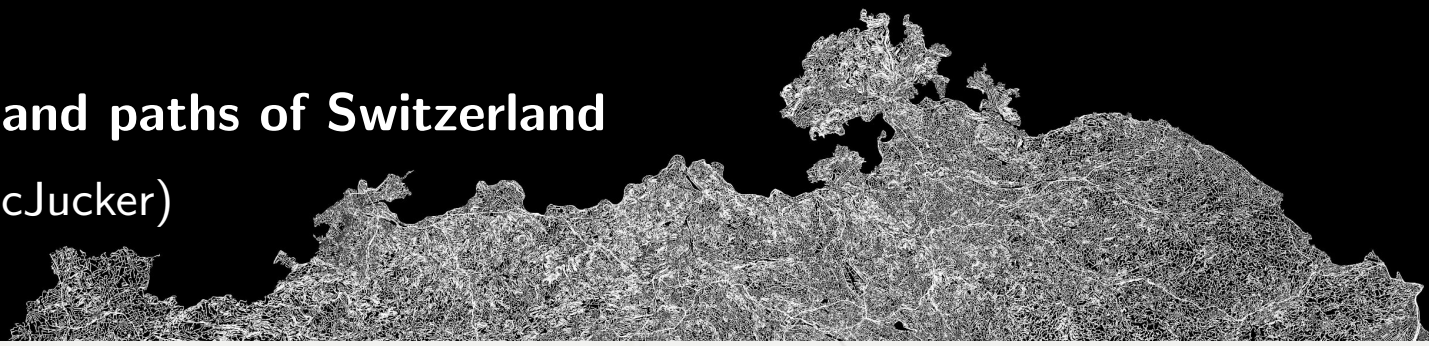
All roads, streets, and paths of Switzerland

(From X, by @JeanLucJucker)



All roads, streets, and paths of Switzerland

(From X, by @JeanLucJucker)



SWISS WATERWAYS

Swiss waterways have an approximate combined length of 74'700 km: That is almost twice the circumference of the entire earth. However, they are not all of equal importance: To define stream size based on network hierarchy, the Strahler number is used as a numerical measure of a stream's branching complexity. The more streams branch into a stream, the higher its Strahler order. This map displays streams of order 3 to 9 and their relation to human settlements.

Data from [Bundesamt für Umwelt](#)



INTERSECTING LINE SEGMENTS

INTERSECTING LINE SEGMENTS

Problem

Input: n line segments in the plane, $s_i = (p_i, q_i)$, $i = 1 \dots n$.

Output: the $k = O(n^2)$ intersections of line segment pairs, (x, y, i, j) .

INTERSECTING LINE SEGMENTS

Problem

Input: n line segments in the plane, $s_i = (p_i, q_i)$, $i = 1 \dots n$.

Output: the $k = O(n^2)$ intersections of line segment pairs, (x, y, i, j) .

Some applications

Geographic information systems

Detecting the intersections among the elements of the different layers of information (cities, roads, services, ...)

Realistic visualization

Eliminating the hidden portions of a scene

INTERSECTING LINE SEGMENTS

Problem

Input: n line segments in the plane, $s_i = (p_i, q_i)$, $i = 1 \dots n$.

Output: the $k = O(n^2)$ intersections of line segment pairs, (x, y, i, j) .

Brute force solution

Check the intersection of the $\binom{n}{2}$ pairs of line segments. This algorithm runs in $\Theta(n^2)$ time.

INTERSECTING LINE SEGMENTS

Problem

Input: n line segments in the plane, $s_i = (p_i, q_i)$, $i = 1 \dots n$.

Output: the $k = O(n^2)$ intersections of line segment pairs, (x, y, i, j) .

Brute force solution

Check the intersection of the $\binom{n}{2}$ pairs of line segments. This algorithm runs in $\Theta(n^2)$ time.

Problem complexity

The problem has complexity $\Omega(n^2)$, because there exist line segment configurations with $\binom{n}{2}$ intersections.

INTERSECTING LINE SEGMENTS

Problem

Input: n line segments in the plane, $s_i = (p_i, q_i)$, $i = 1 \dots n$.

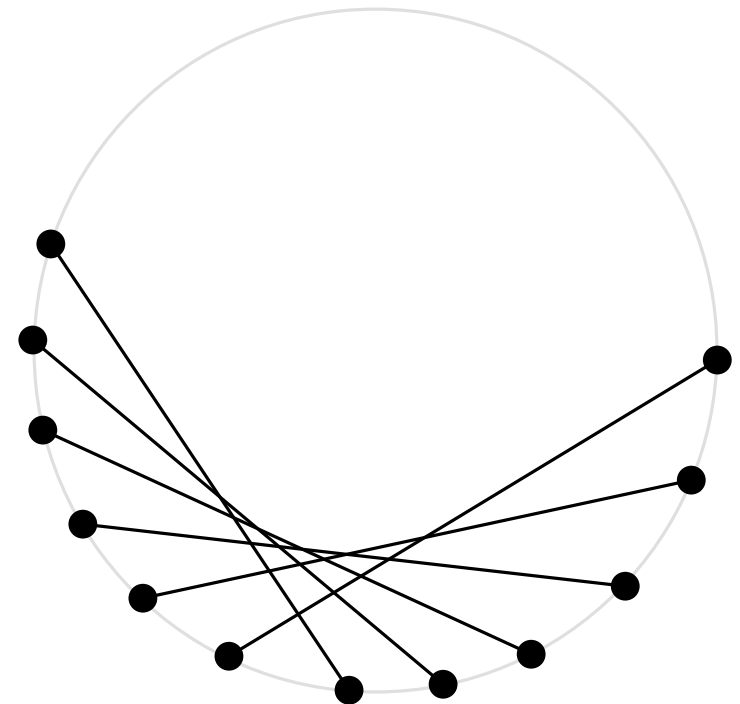
Output: the $k = O(n^2)$ intersections of line segment pairs, (x, y, i, j) .

Brute force solution

Check the intersection of the $\binom{n}{2}$ pairs of line segments. This algorithm runs in $\Theta(n^2)$ time.

Problem complexity

The problem has complexity $\Omega(n^2)$, because there exist line segment configurations with $\binom{n}{2}$ intersections.



INTERSECTING LINE SEGMENTS

Problem

Input: n line segments in the plane, $s_i = (p_i, q_i)$, $i = 1 \dots n$.

Output: the $k = O(n^2)$ intersections of line segment pairs, (x, y, i, j) .

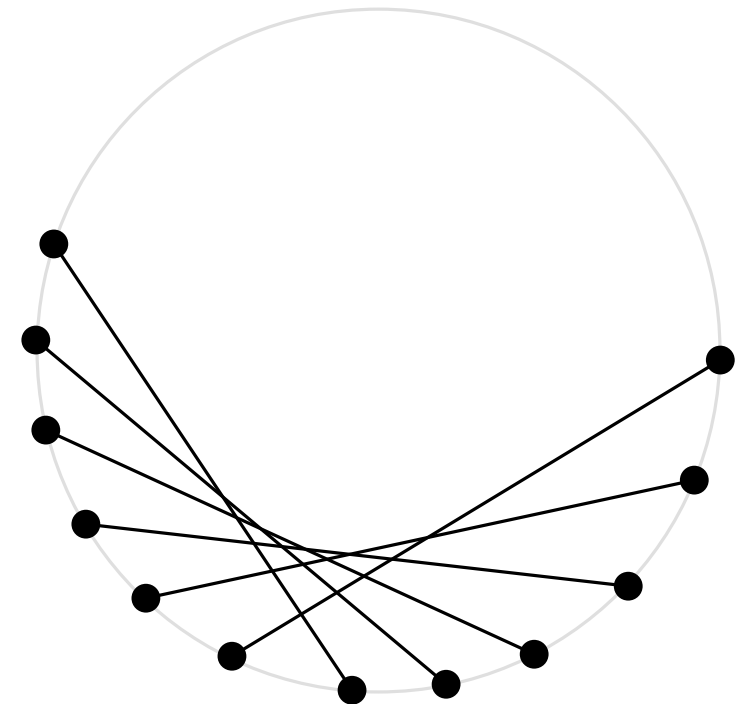
Brute force solution

Check the intersection of the $\binom{n}{2}$ pairs of line segments. This algorithm runs in $\Theta(n^2)$ time.

Problem complexity

The problem has complexity $\Omega(n^2)$, because there exist line segment configurations with $\binom{n}{2}$ intersections.

Nevertheless, there exist sets of n line segments with a number of intersections substantially smaller than $\binom{n}{2}$.



INTERSECTING LINE SEGMENTS

Problem

Input: n line segments in the plane, $s_i = (p_i, q_i)$, $i = 1 \dots n$.

Output: the $k = O(n^2)$ intersections of line segment pairs, (x, y, i, j) .

Brute force solution

Check the intersection of the $\binom{n}{2}$ pairs of line segments. This algorithm runs in $\Theta(n^2)$ time.

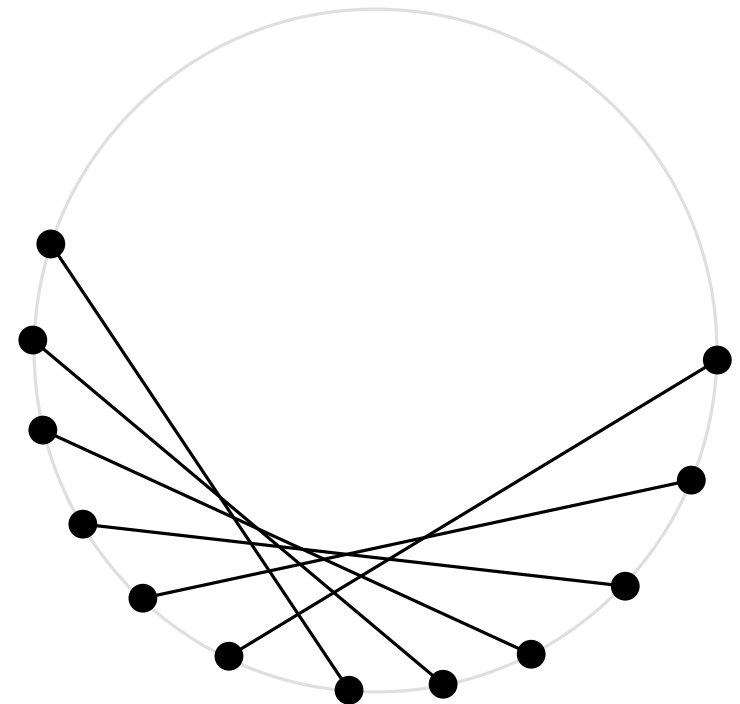
Problem complexity

The problem has complexity $\Omega(n^2)$, because there exist line segment configurations with $\binom{n}{2}$ intersections.

Nevertheless, there exist sets of n line segments with a number of intersections substantially smaller than $\binom{n}{2}$.

Our goal: output-sensitive algorithm

Algorithm whose running time depends on the number of intersections to be reported.



INTERSECTING LINE SEGMENTS

Problem

Input: n line segments in the plane, $s_i = (p_i, q_i)$, $i = 1 \dots n$.

Output: the $k = O(n^2)$ intersections of line segment pairs, (x, y, i, j) .

INTERSECTING LINE SEGMENTS

Problem

Input: n line segments in the plane, $s_i = (p_i, q_i)$, $i = 1 \dots n$.

Output: the $k = O(n^2)$ intersections of line segment pairs, (x, y, i, j) .

Observation 1

If two line segments have disjoint projections onto a given line, then they are disjoint.

INTERSECTING LINE SEGMENTS

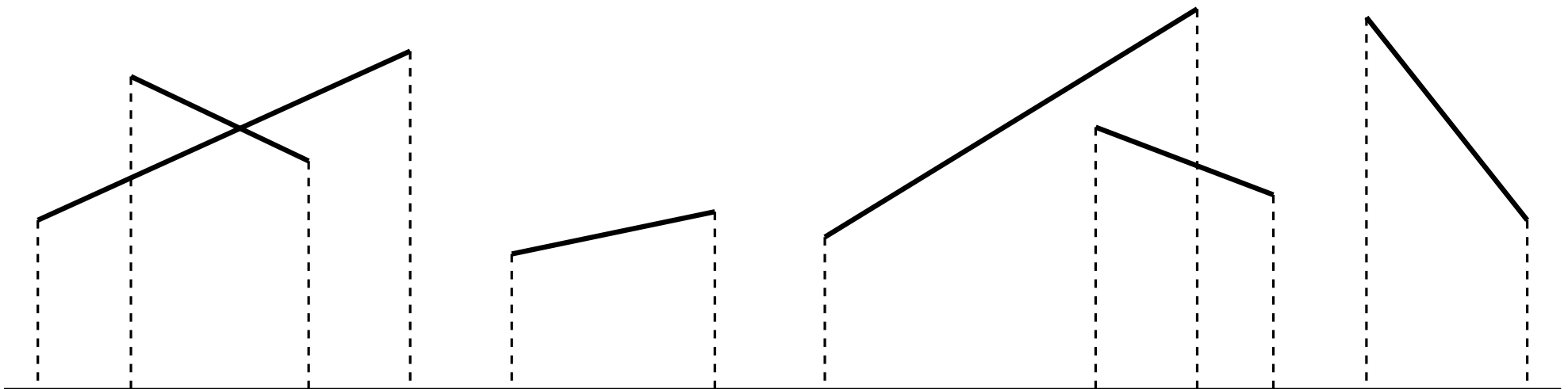
Problem

Input: n line segments in the plane, $s_i = (p_i, q_i)$, $i = 1 \dots n$.

Output: the $k = O(n^2)$ intersections of line segment pairs, (x, y, i, j) .

Observation 1

If two line segments have disjoint projections onto a given line, then they are disjoint.



INTERSECTING LINE SEGMENTS

Problem

Input: n line segments in the plane, $s_i = (p_i, q_i)$, $i = 1 \dots n$.

Output: the $k = O(n^2)$ intersections of line segment pairs, (x, y, i, j) .

Observation 1

If two line segments have disjoint projections onto a given line, then they are disjoint.

Observation 2

When sweeping a set of line segments with a line, two intersecting line segments need to be consecutive in the sweeping line right before their intersection point.

INTERSECTING LINE SEGMENTS

Problem

Input: n line segments in the plane, $s_i = (p_i, q_i)$, $i = 1 \dots n$.

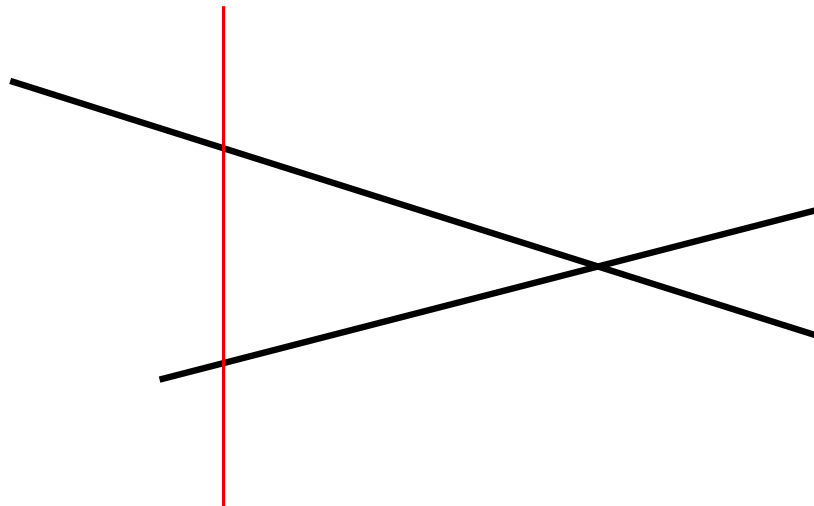
Output: the $k = O(n^2)$ intersections of line segment pairs, (x, y, i, j) .

Observation 1

If two line segments have disjoint projections onto a given line, then they are disjoint.

Observation 2

When sweeping a set of line segments with a line, two intersecting line segments need to be consecutive in the sweeping line right before their intersection point.



INTERSECTING LINE SEGMENTS

Problem

Input: n line segments in the plane, $s_i = (p_i, q_i)$, $i = 1 \dots n$.

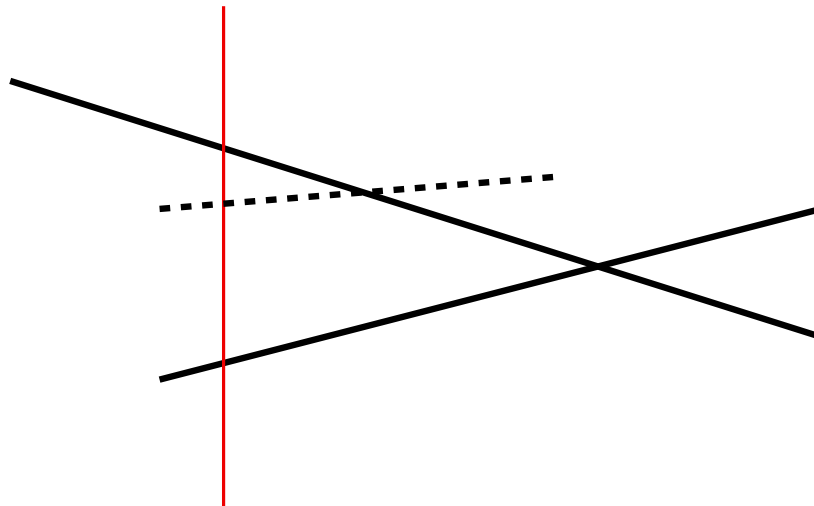
Output: the $k = O(n^2)$ intersections of line segment pairs, (x, y, i, j) .

Observation 1

If two line segments have disjoint projections onto a given line, then they are disjoint.

Observation 2

When sweeping a set of line segments with a line, two intersecting line segments need to be consecutive in the sweeping line right before their intersection point.



INTERSECTING LINE SEGMENTS

Problem

Input: n line segments in the plane, $s_i = (p_i, q_i)$, $i = 1 \dots n$.

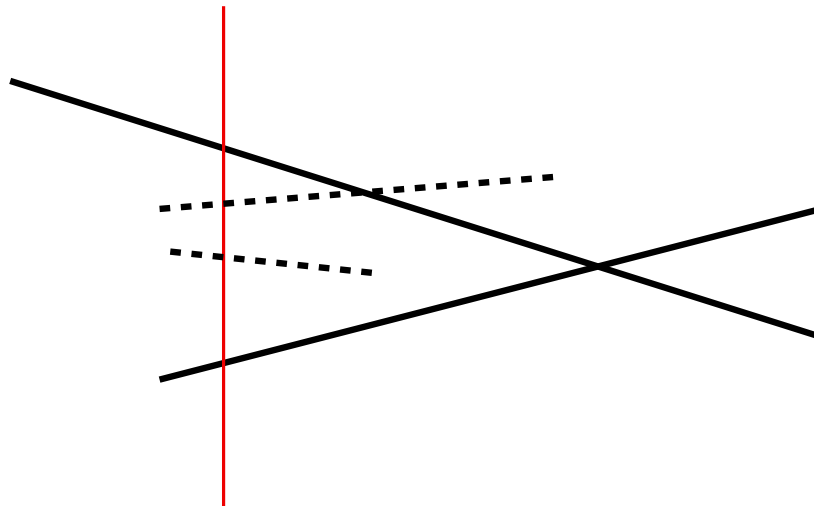
Output: the $k = O(n^2)$ intersections of line segment pairs, (x, y, i, j) .

Observation 1

If two line segments have disjoint projections onto a given line, then they are disjoint.

Observation 2

When sweeping a set of line segments with a line, two intersecting line segments need to be consecutive in the sweeping line right before their intersection point.



INTERSECTING LINE SEGMENTS

Problem

Input: n line segments in the plane, $s_i = (p_i, q_i)$, $i = 1 \dots n$.

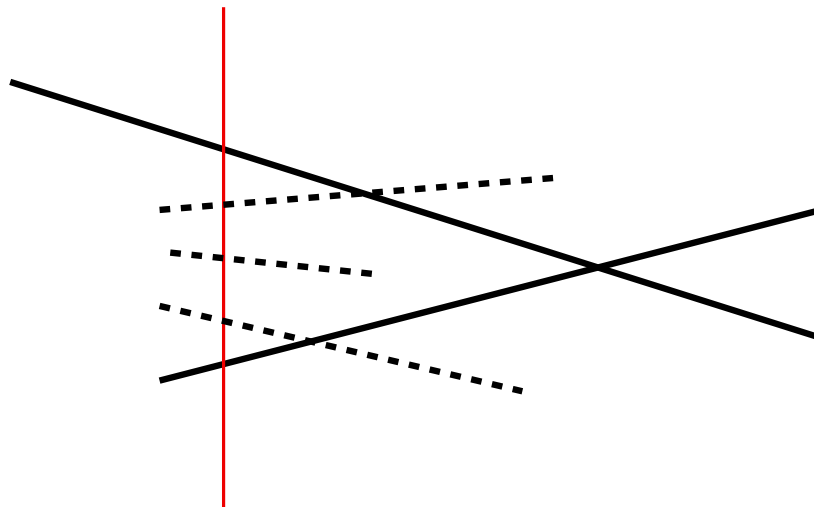
Output: the $k = O(n^2)$ intersections of line segment pairs, (x, y, i, j) .

Observation 1

If two line segments have disjoint projections onto a given line, then they are disjoint.

Observation 2

When sweeping a set of line segments with a line, two intersecting line segments need to be consecutive in the sweeping line right before their intersection point.



INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm

Sweep-line algorithm

INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm

Sweep-line algorithm

A straight line (vertical, in this case) scans the scene (the line segments) and allows detecting and constructing the desired elements (intersections), leaving the problem solved behind it. The sweeping process is discretized.

INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm

Sweep-line algorithm

A straight line (vertical, in this case) scans the scene (the line segments) and allows detecting and constructing the desired elements (intersections), leaving the problem solved behind it. The sweeping process is discretized.

Essential elements of a sweep line algorithm:

- Sweep line

Data structure storing the information of the portion of the scene intersected by the sweep line. It stays updated at all times. In our problem, it will contain the information of the line segments intersected by the sweep line, sorted by y -coordinate.

- Events queue

Priority queue keeping the information of the locations where the sweep line changes and needs to be updated. In our problem, the events will be the endpoints and the intersection points of the line segments, sorted by their x -coordinate. Notice that not all events are known in advance!

- Output data structure

INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm

Sweep-line algorithm

Hypothesis (to be eliminated later on)

1. There are no repeated abscissae, i.e.: there are no vertical line segments, and no two endpoints of two line segments, no two intersection points of line segments, no endpoint and intersection point, lie in the same vertical line.
2. At each intersection point, only two line segments intersect.

INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm

Sweep-line algorithm

Hypothesis (to be eliminated later on)

1. There are no repeated abscissae, i.e.: there are no vertical line segments, and no two endpoints of two line segments, no two intersection points of line segments, no endpoint and intersection point, lie in the same vertical line.
2. At each intersection point, only two line segments intersect.

Sweep line (L)

Stabbed line segments, in vertical order.

INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm

Sweep-line algorithm

Hypothesis (to be eliminated later on)

1. There are no repeated abscissae, i.e.: there are no vertical line segments, and no two endpoints of two line segments, no two intersection points of line segments, no endpoint and intersection point, lie in the same vertical line.
2. At each intersection point, only two line segments intersect.

Sweep line (L)

Stabbed line segments, in vertical order.

Events queue (E)

- All endpoints of the line segments (known a priori).
- All intersection points of line segments (found on the fly).

INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm

Input: n line segments in the plane, $s_i = (p_i, q_i)$, $i = 1 \dots n$.

Output: the $k = O(n^2)$ intersections of line segment pairs, (x, y, i, j) .

INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm

Input: n line segments in the plane, $s_i = (p_i, q_i)$, $i = 1 \dots n$.

Output: the $k = O(n^2)$ intersections of line segment pairs, (x, y, i, j) .

Initialization:

- Insert the $2n$ start/endpoint events into E .
- Line L starts empty.

INTERSECTING LINE SEGMENTS

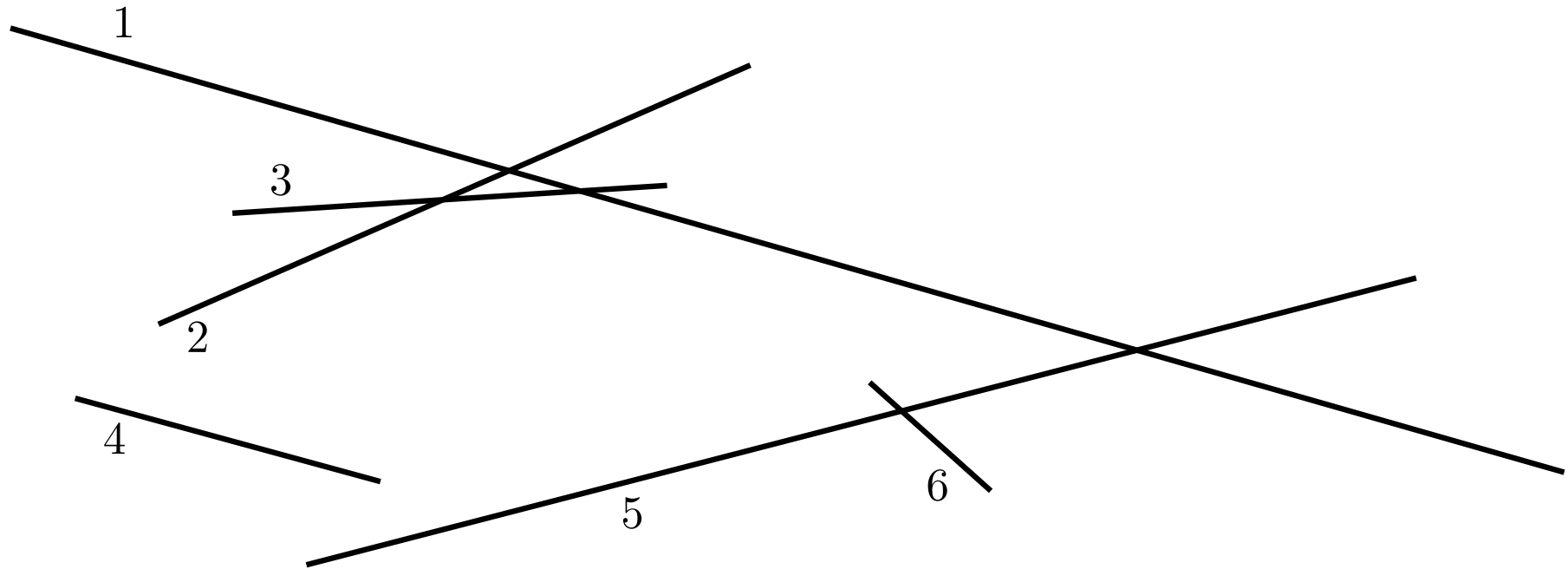
Advance

While $E \neq \emptyset$ do:

1. $p = \min E$
2. If $p = \text{start}(s)$, then:
 - Insert s in L
 - If s^- and s intersect to the right of p , insert their intersection point in E and report it (if needed). Do the same for s^+ .
3. If $p = \text{end}(s)$, then:
 - If s^- and s^+ intersect to the right of p , insert their intersection point in E and report it (if needed).
 - Delete s from L
4. If $p = s_1 \cap s_2$ with $s_1 <_L s_2$, then:
 - If s_1^- and s_2 intersect to the right of p , insert their intersection point in E and report it (if needed). Do the same for s_2^+ and s_1 .
 - Transpose s_1 and s_2 in L
5. Delete p from E

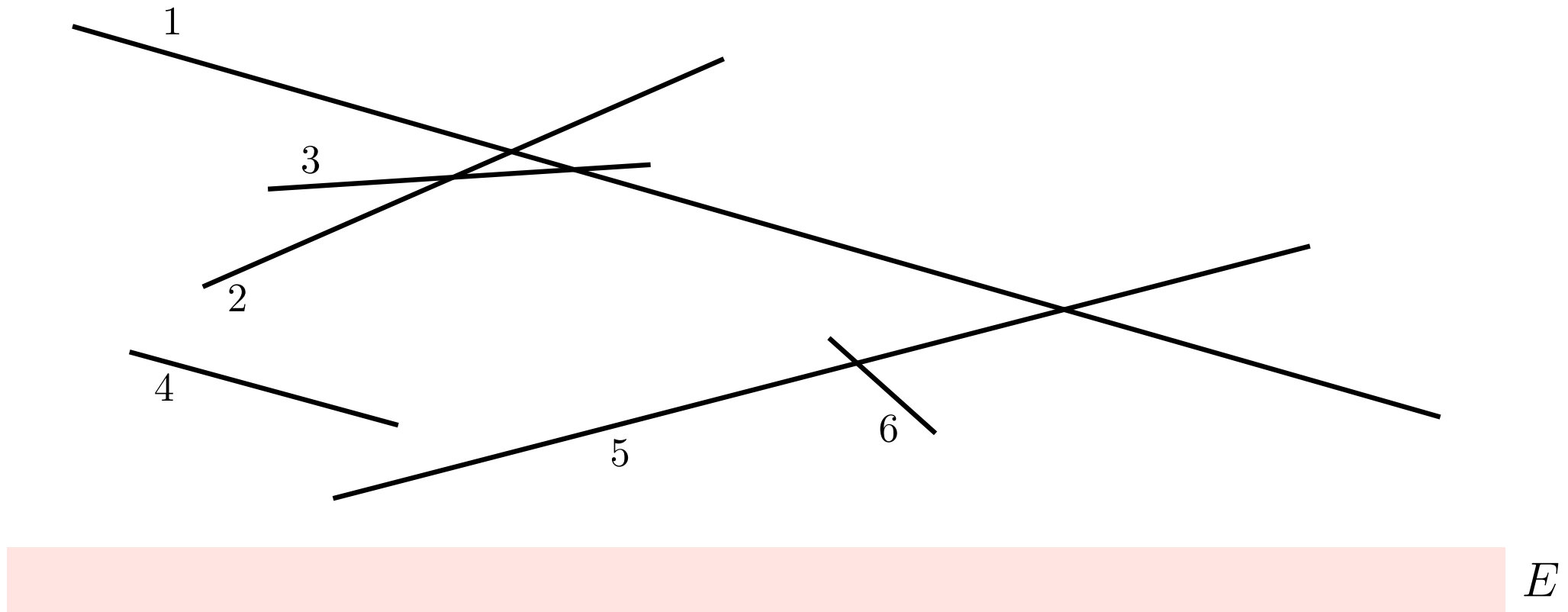
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



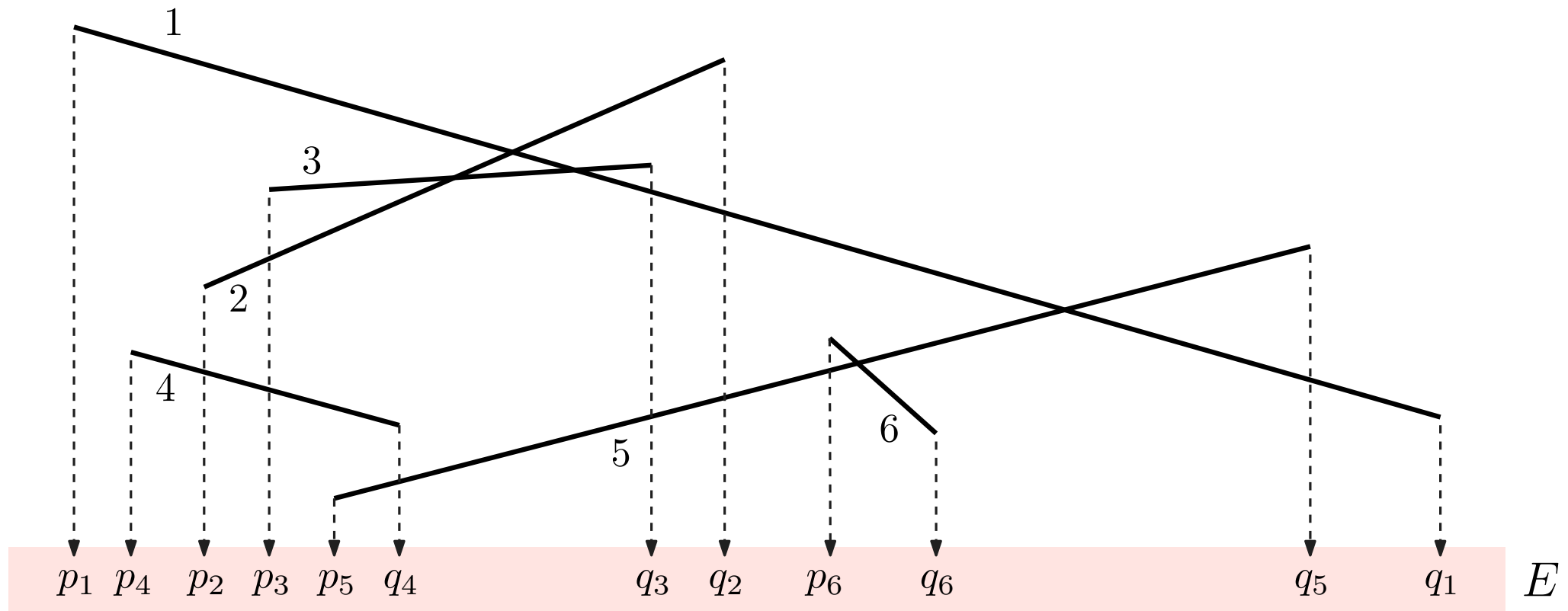
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



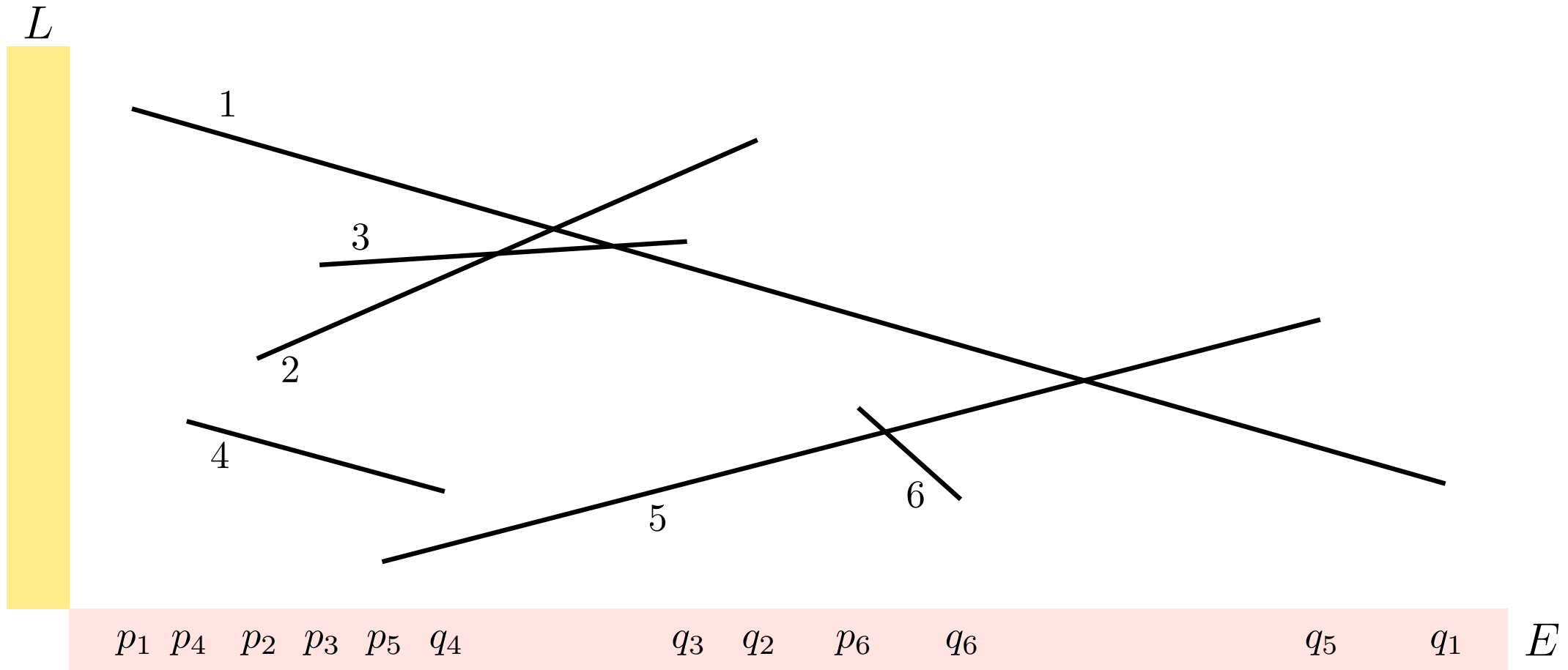
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



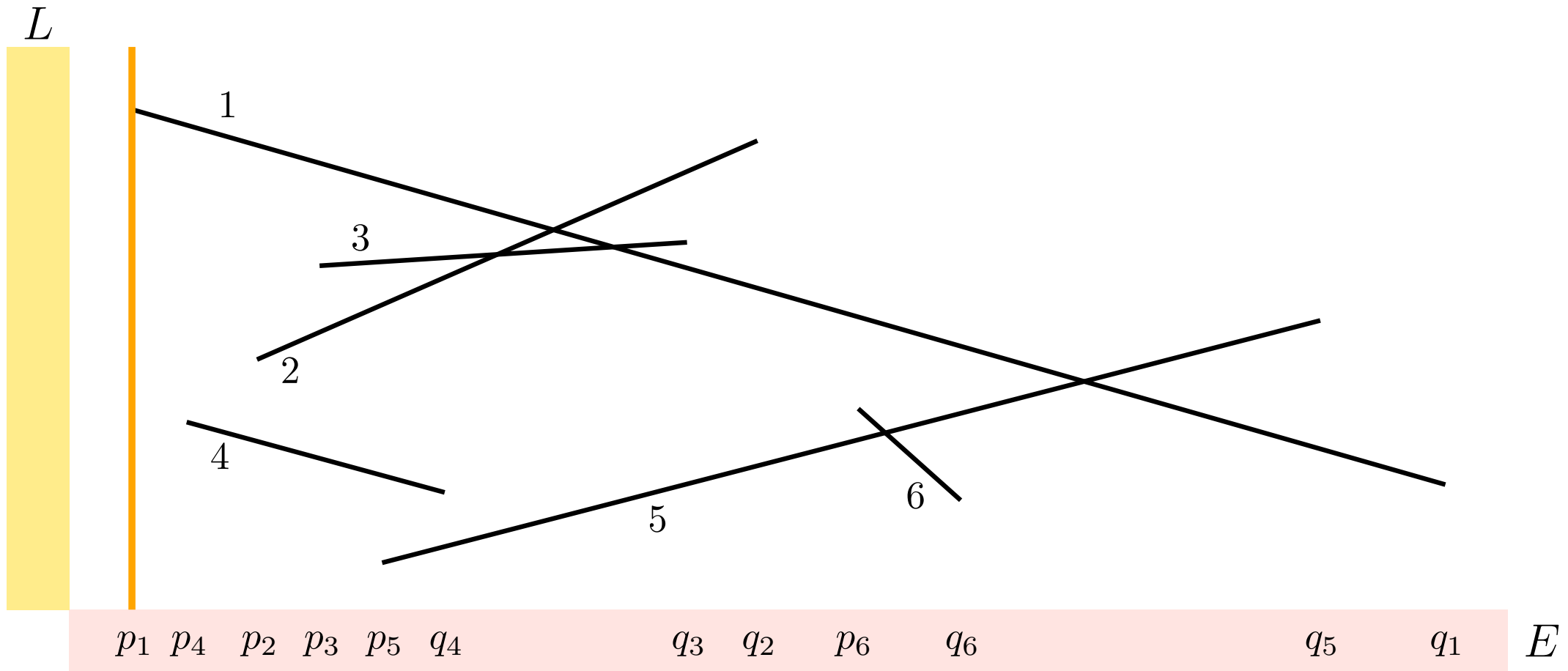
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



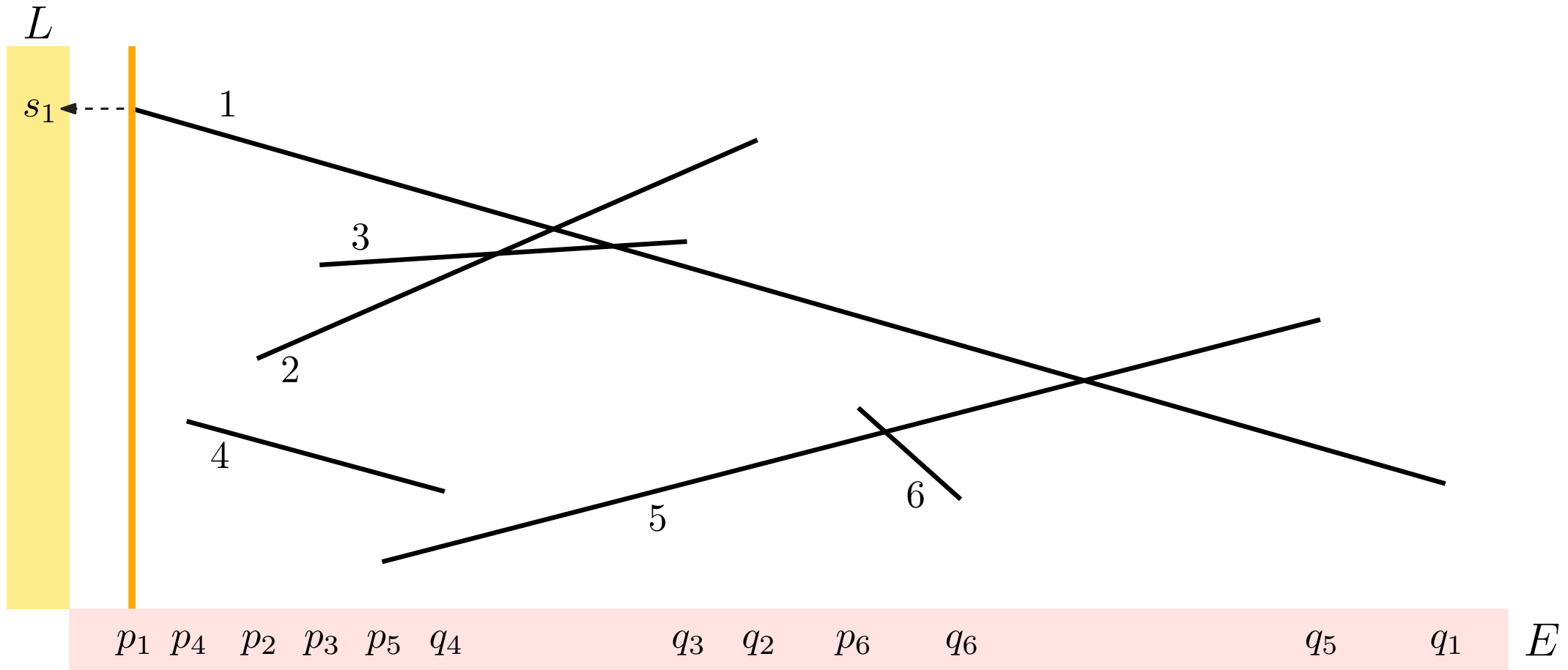
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



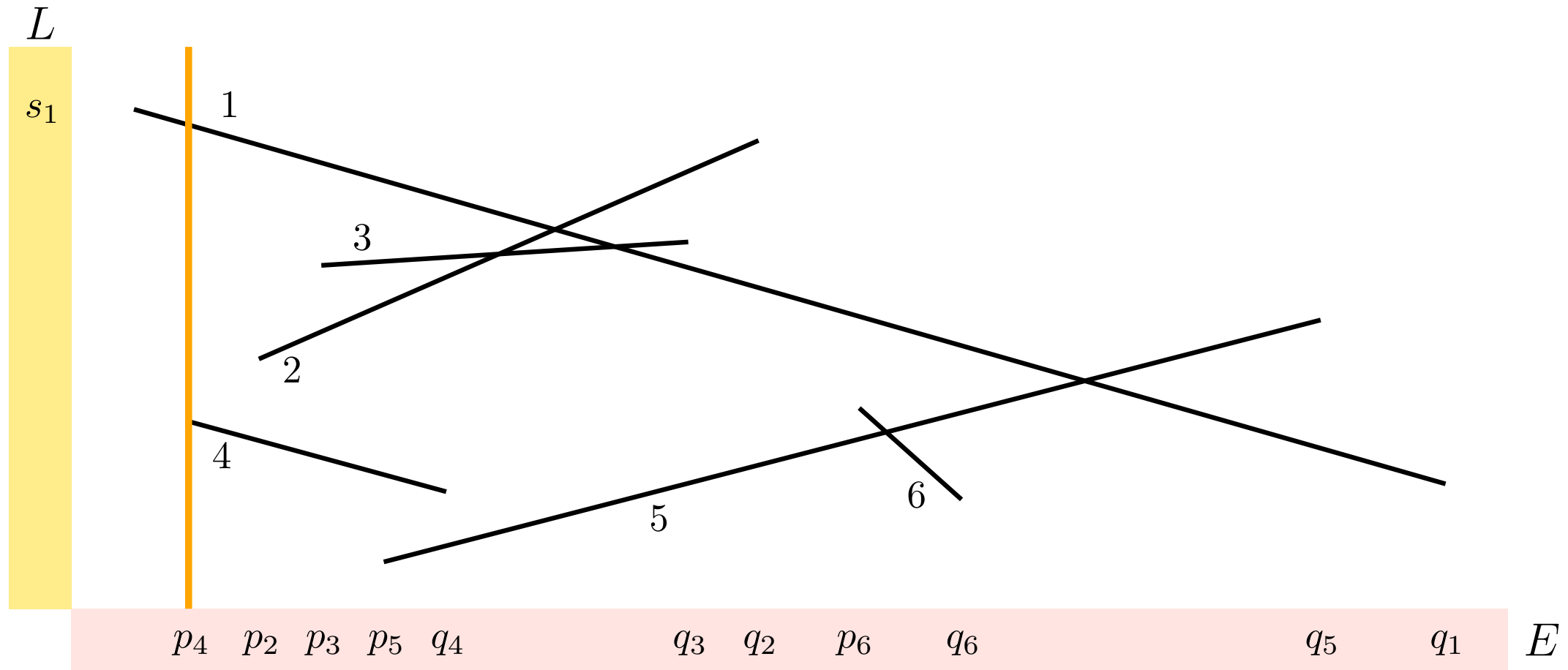
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



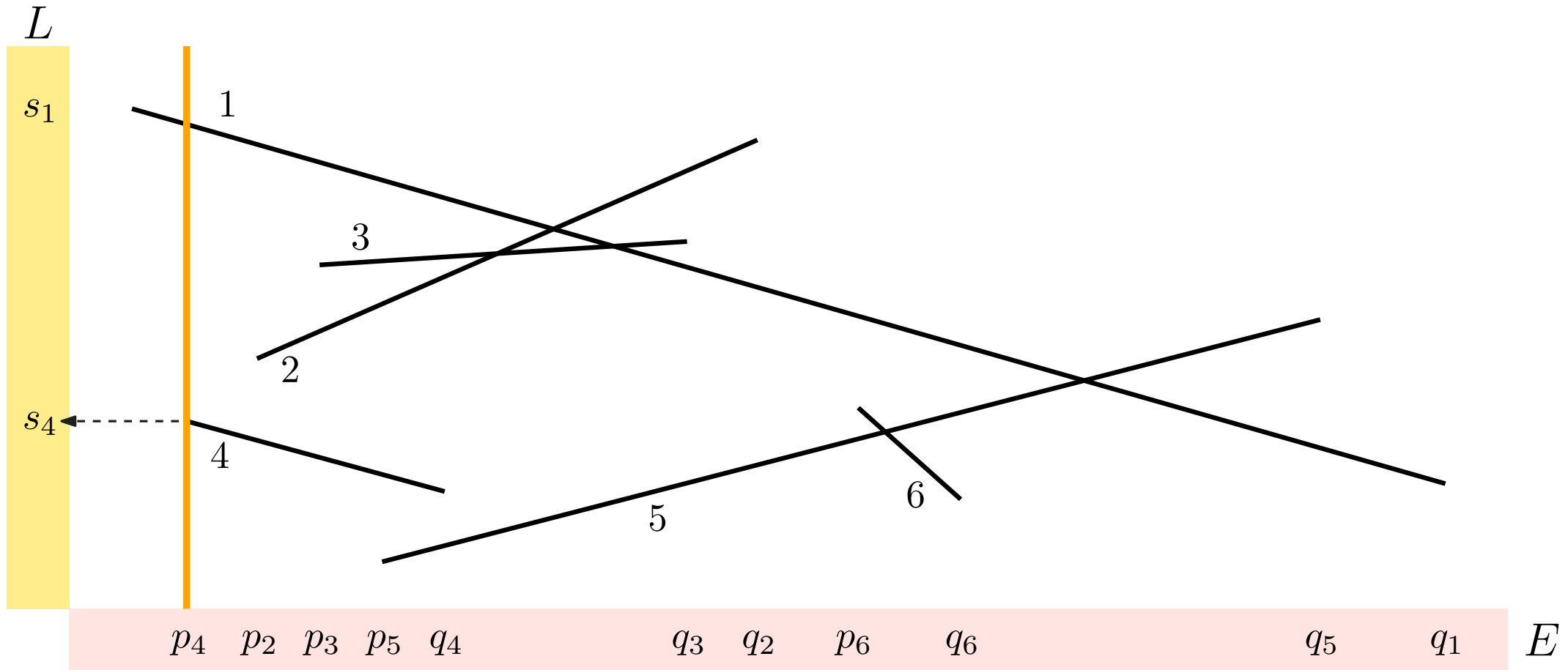
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



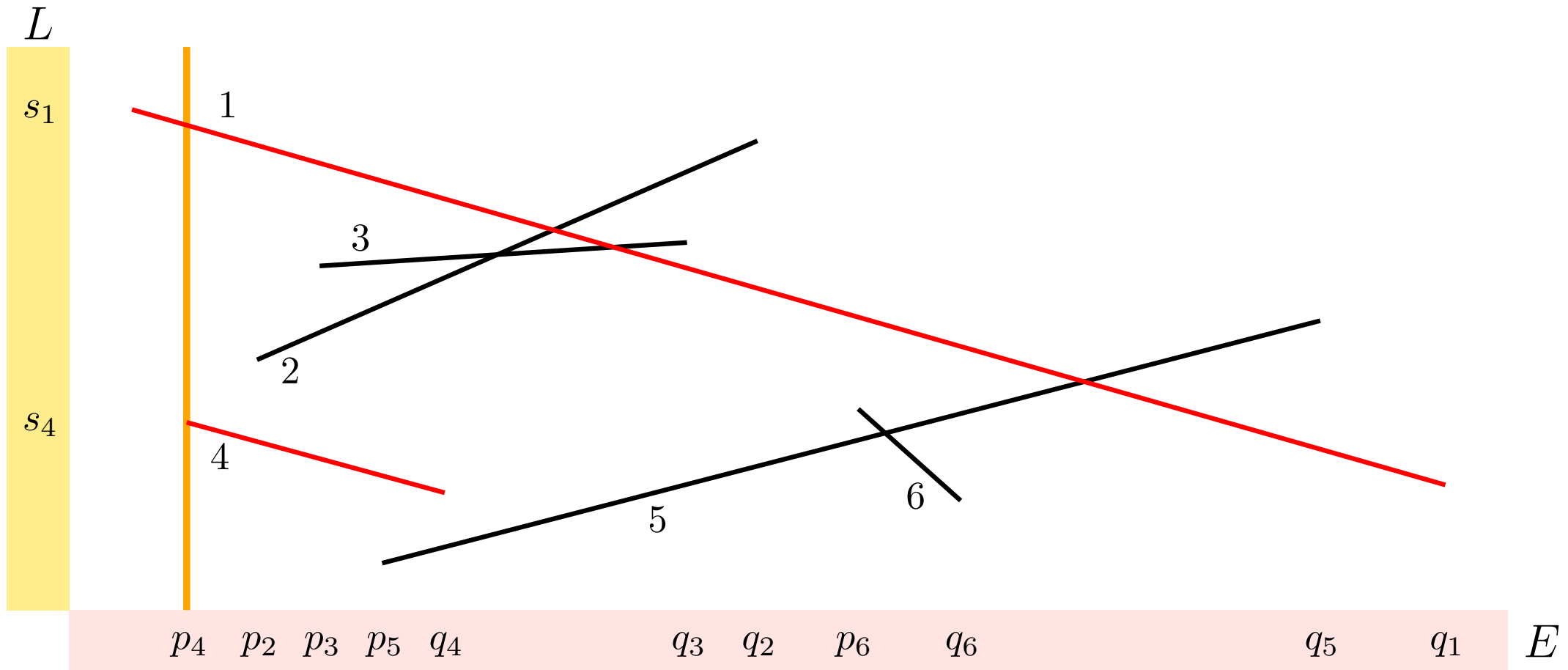
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



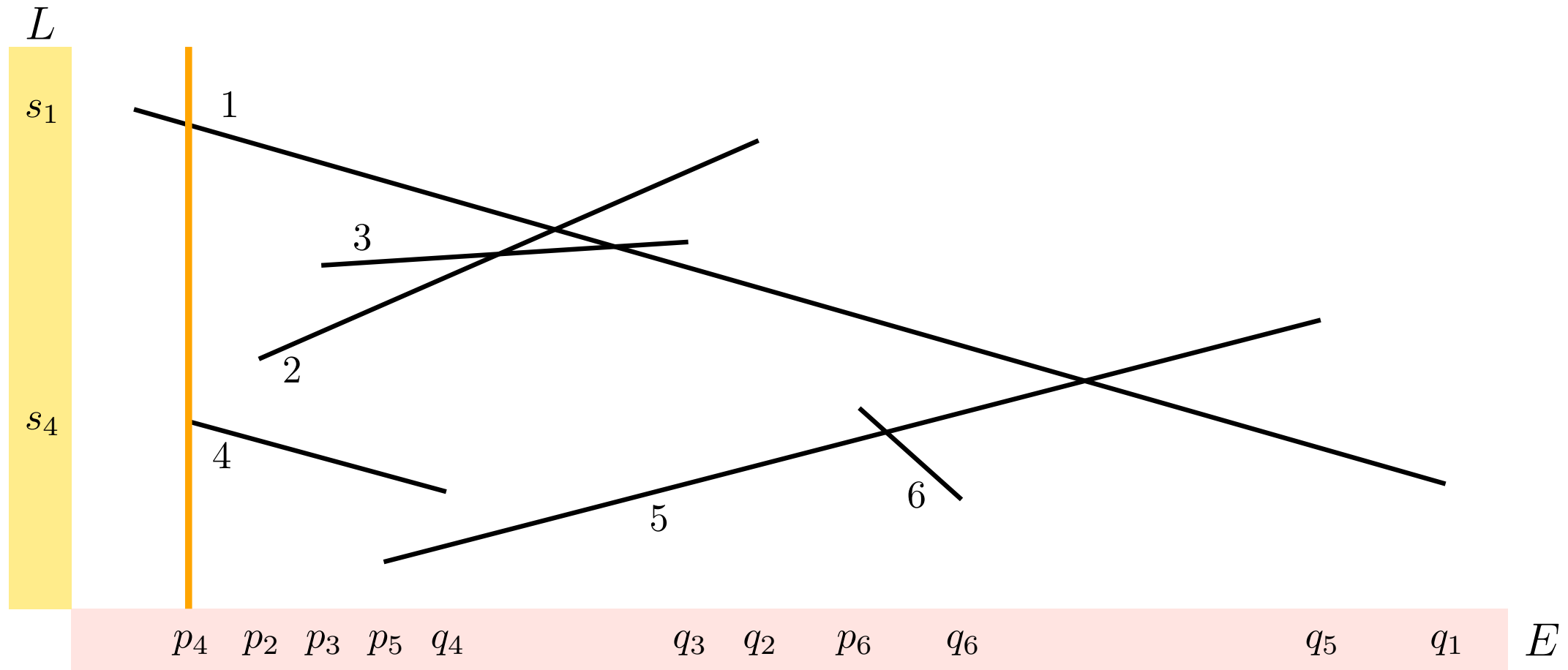
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



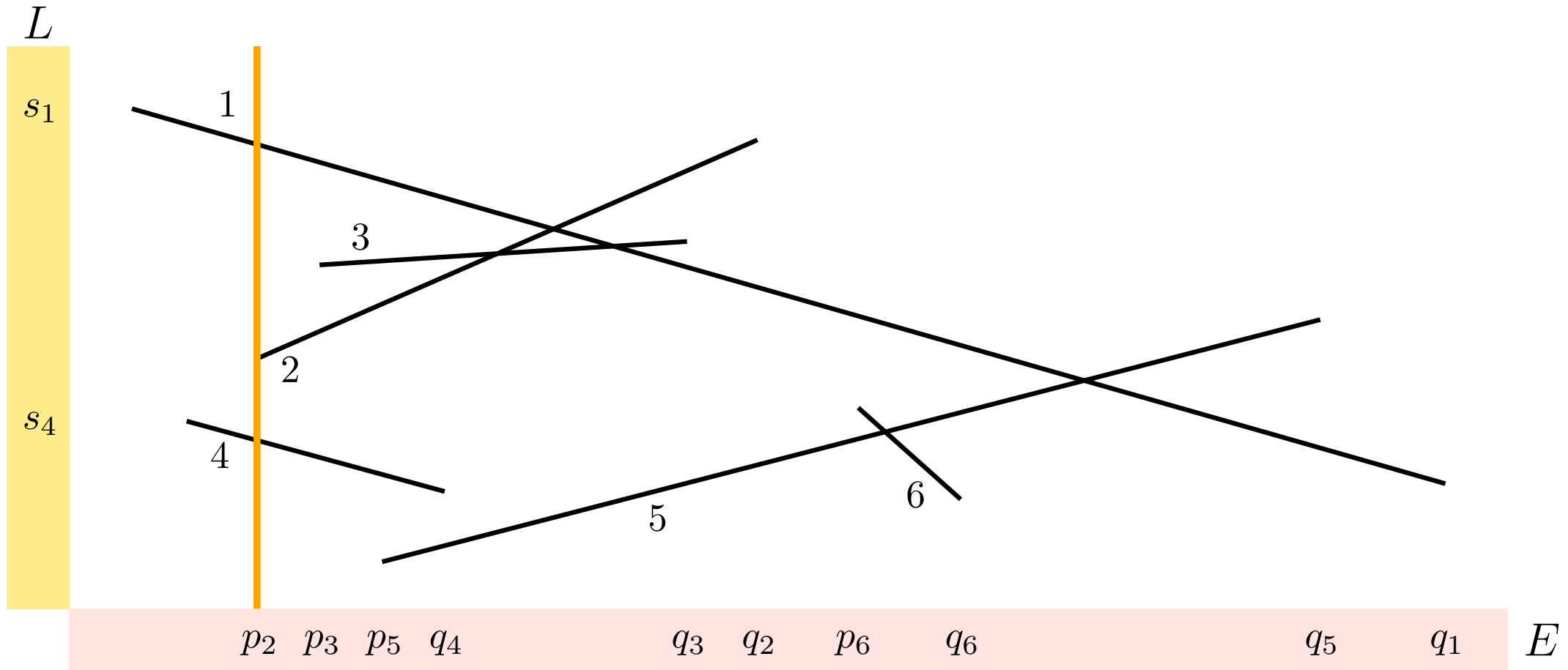
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



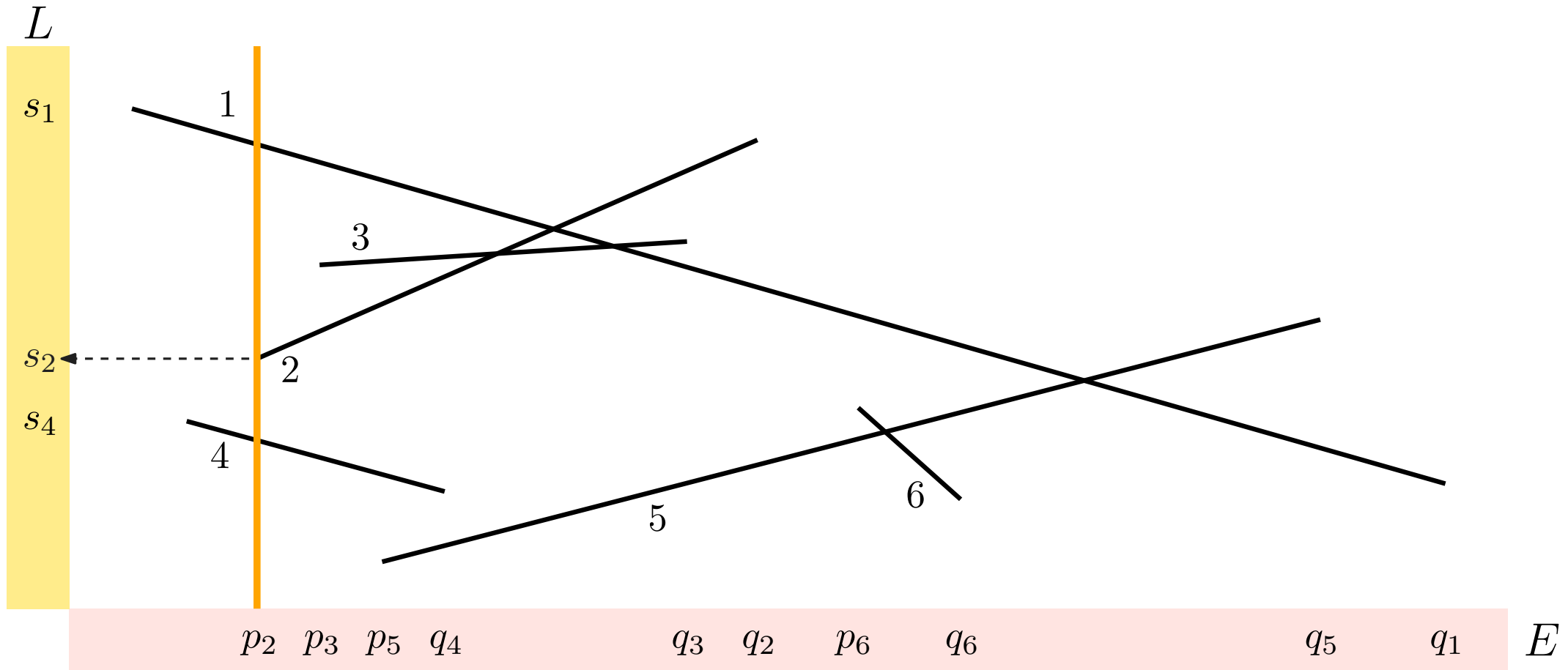
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



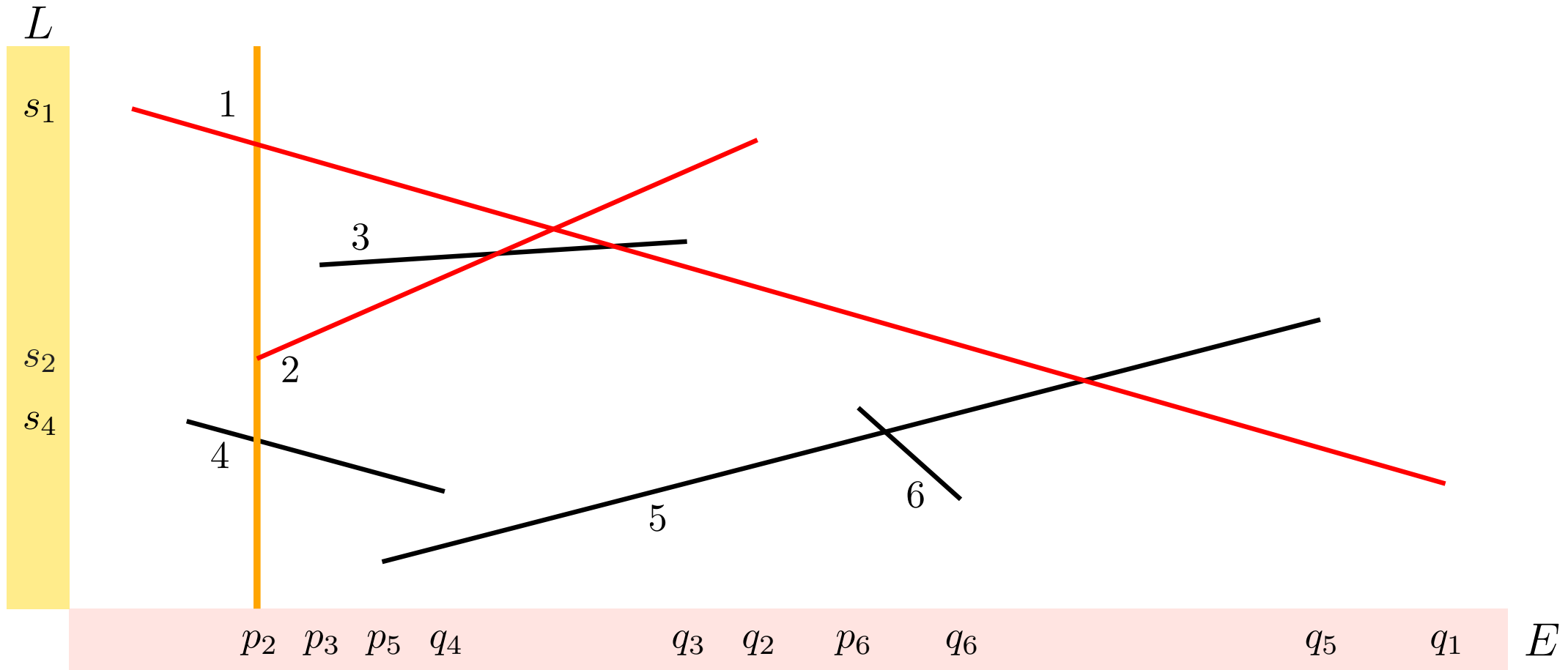
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



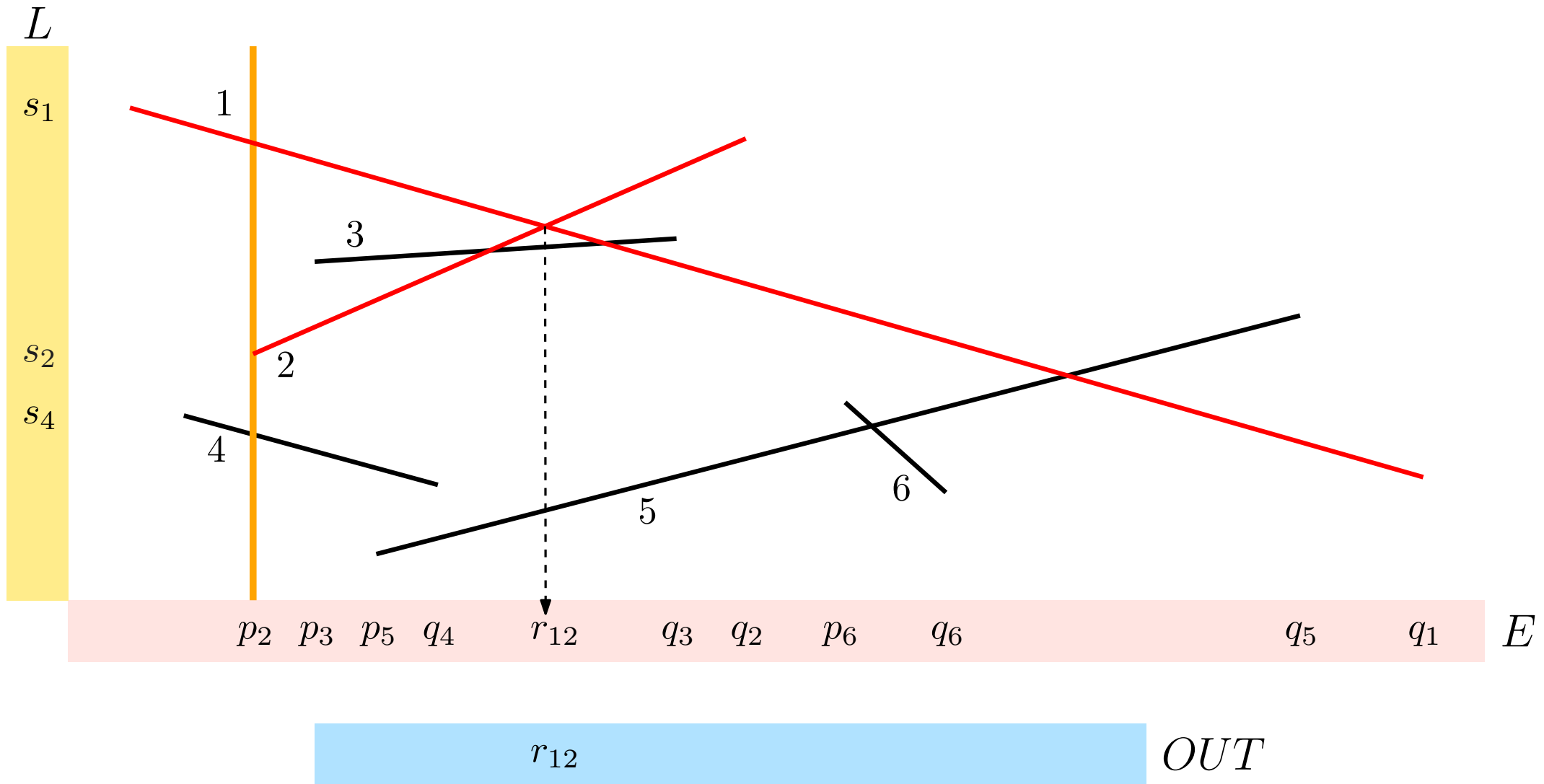
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



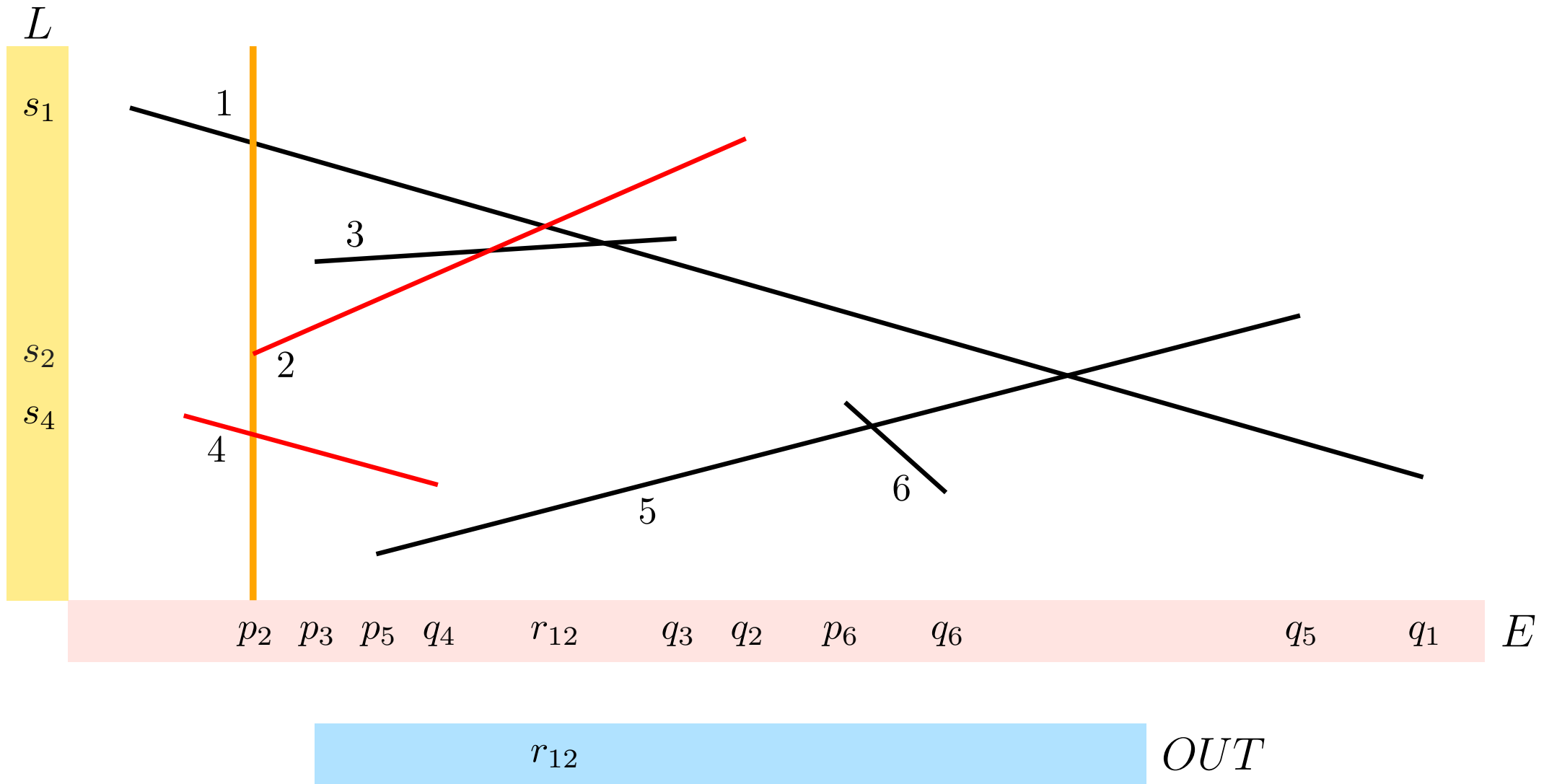
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



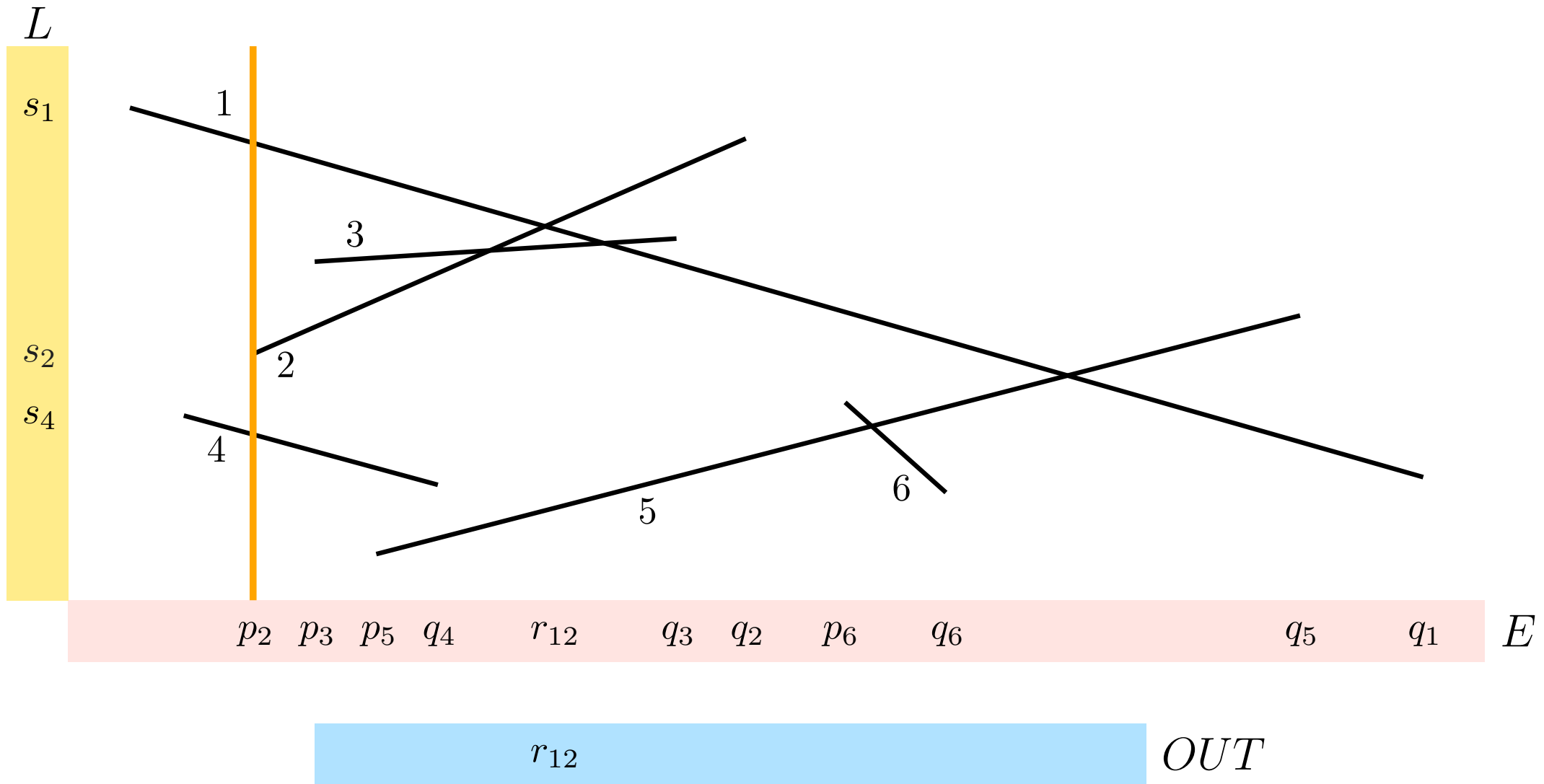
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



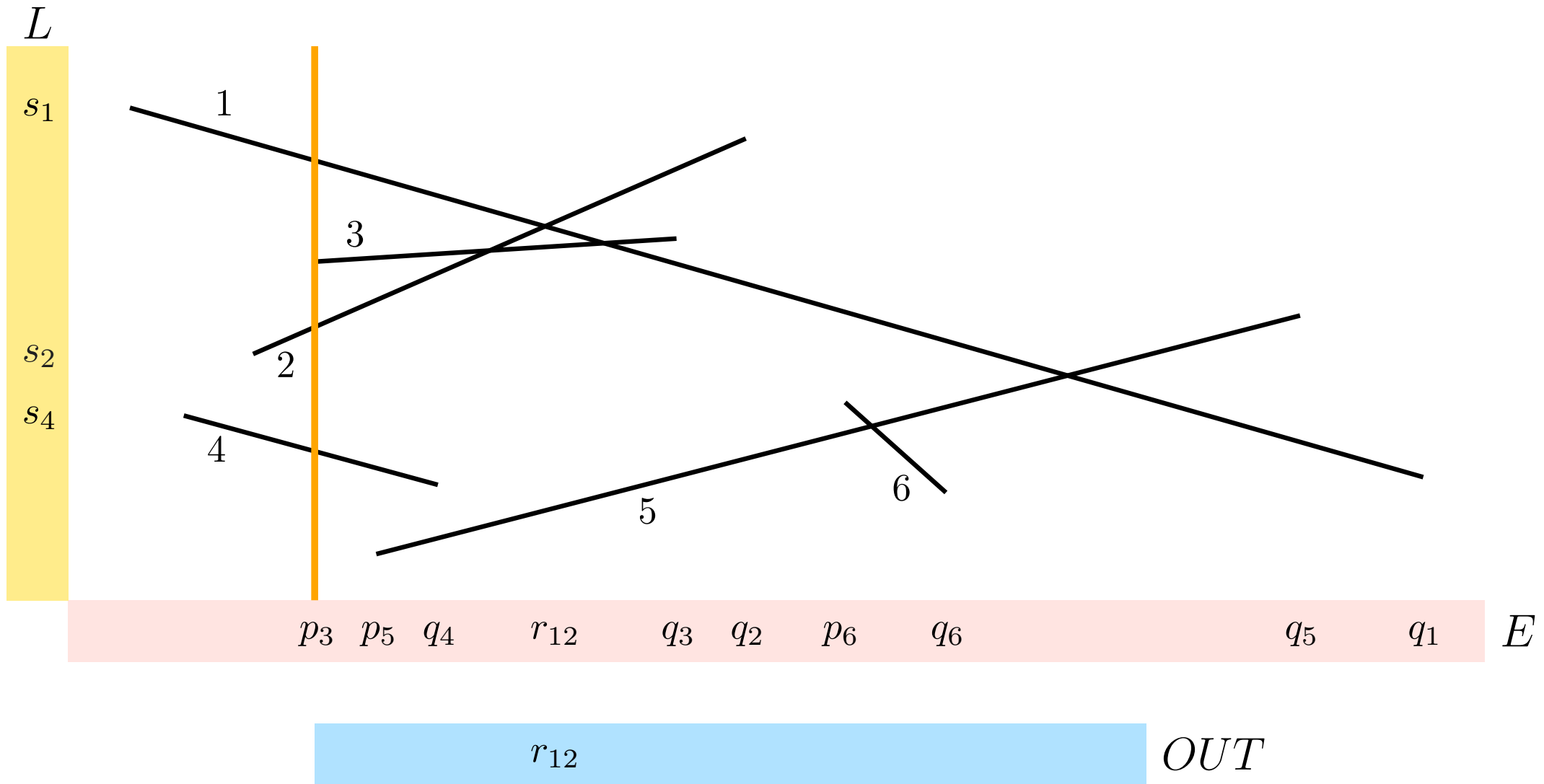
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



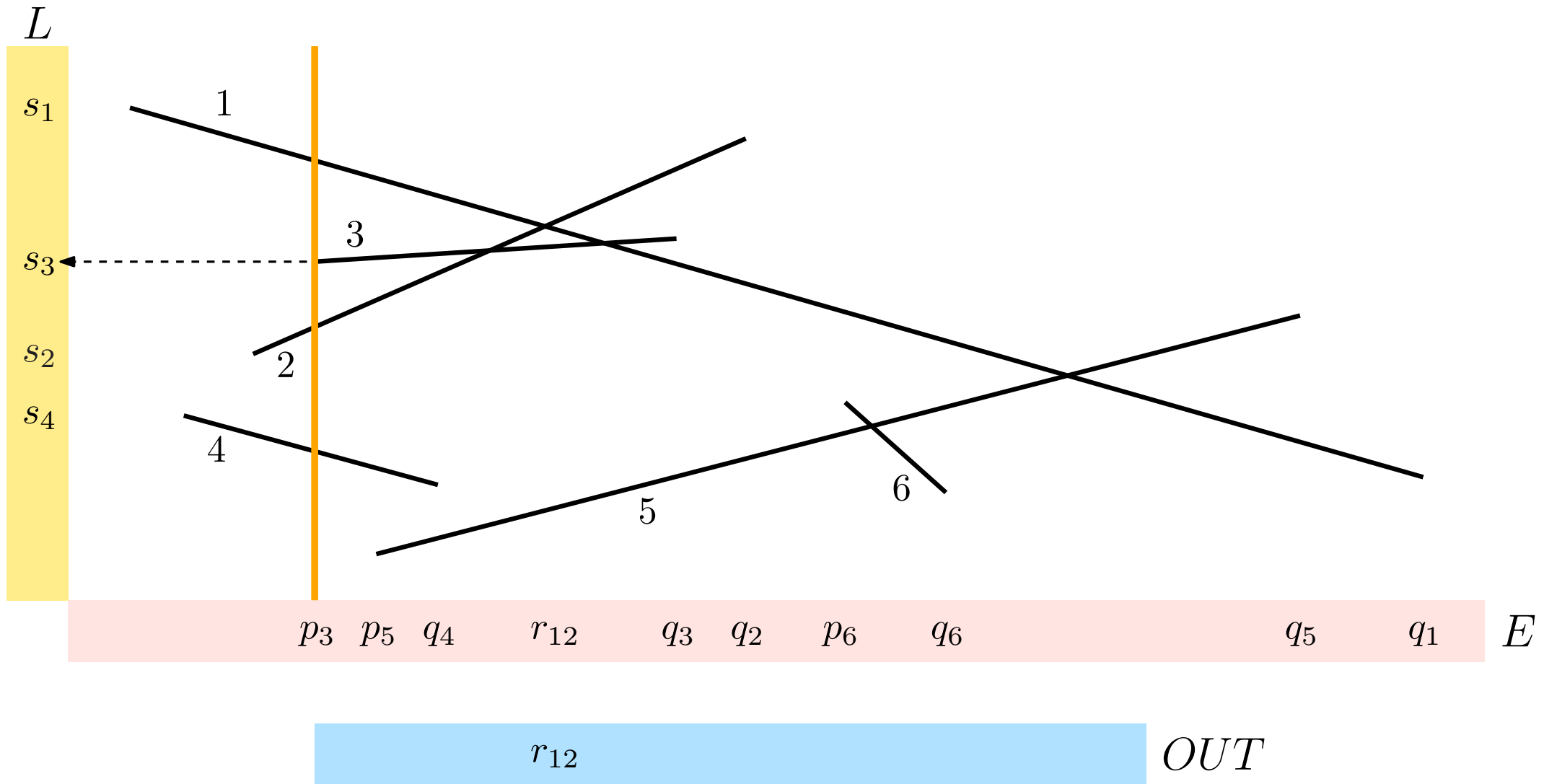
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



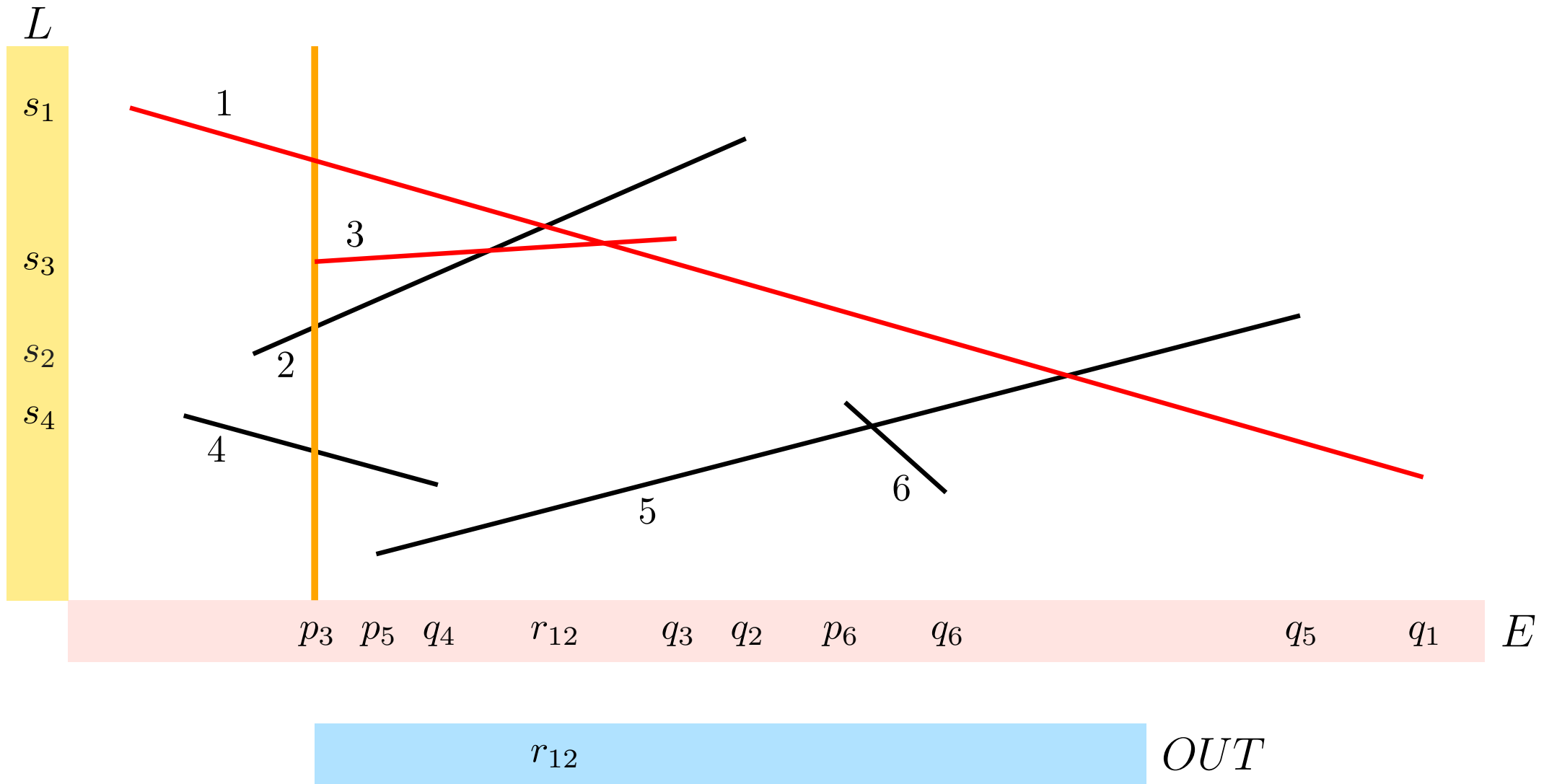
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



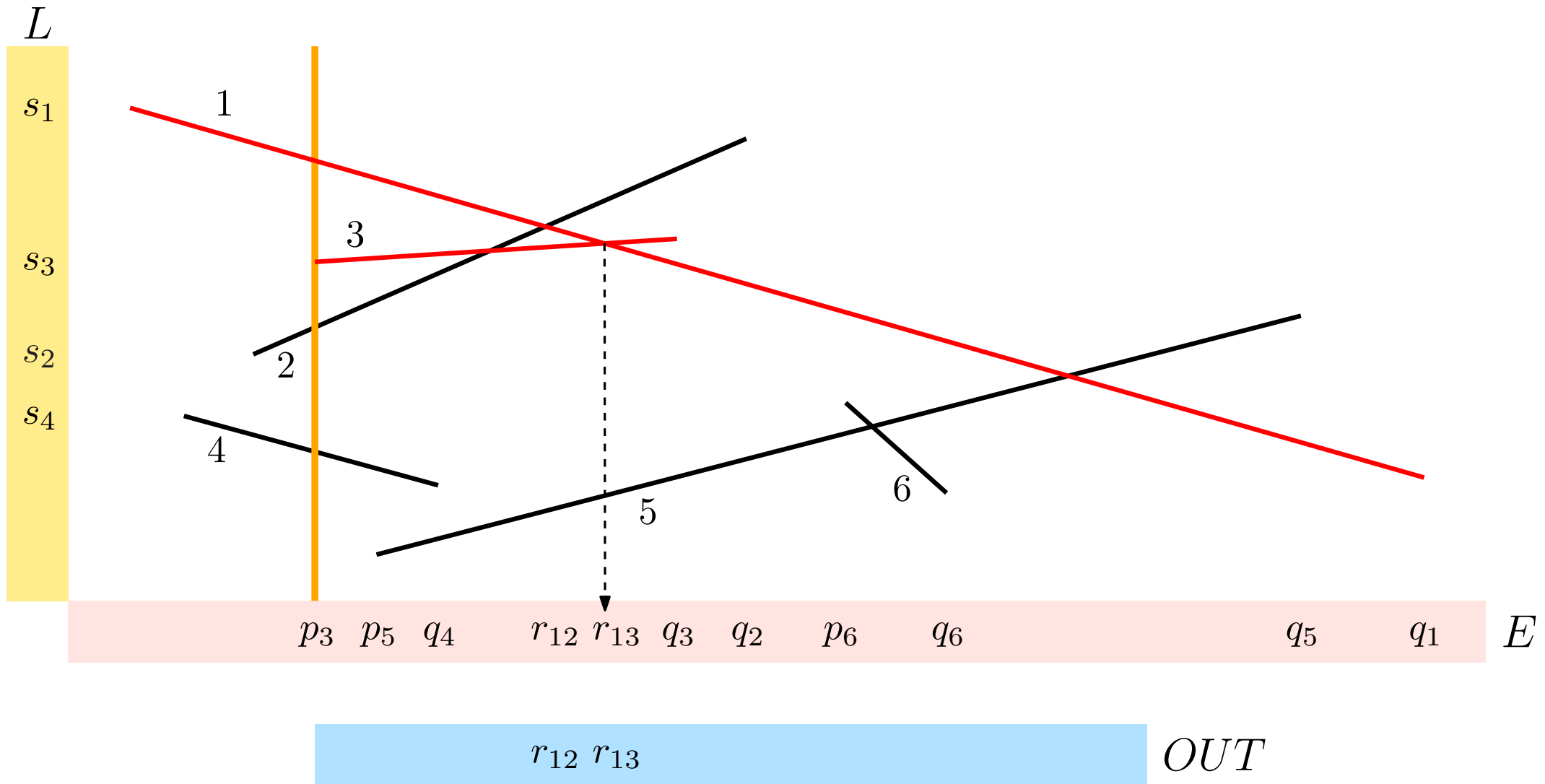
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



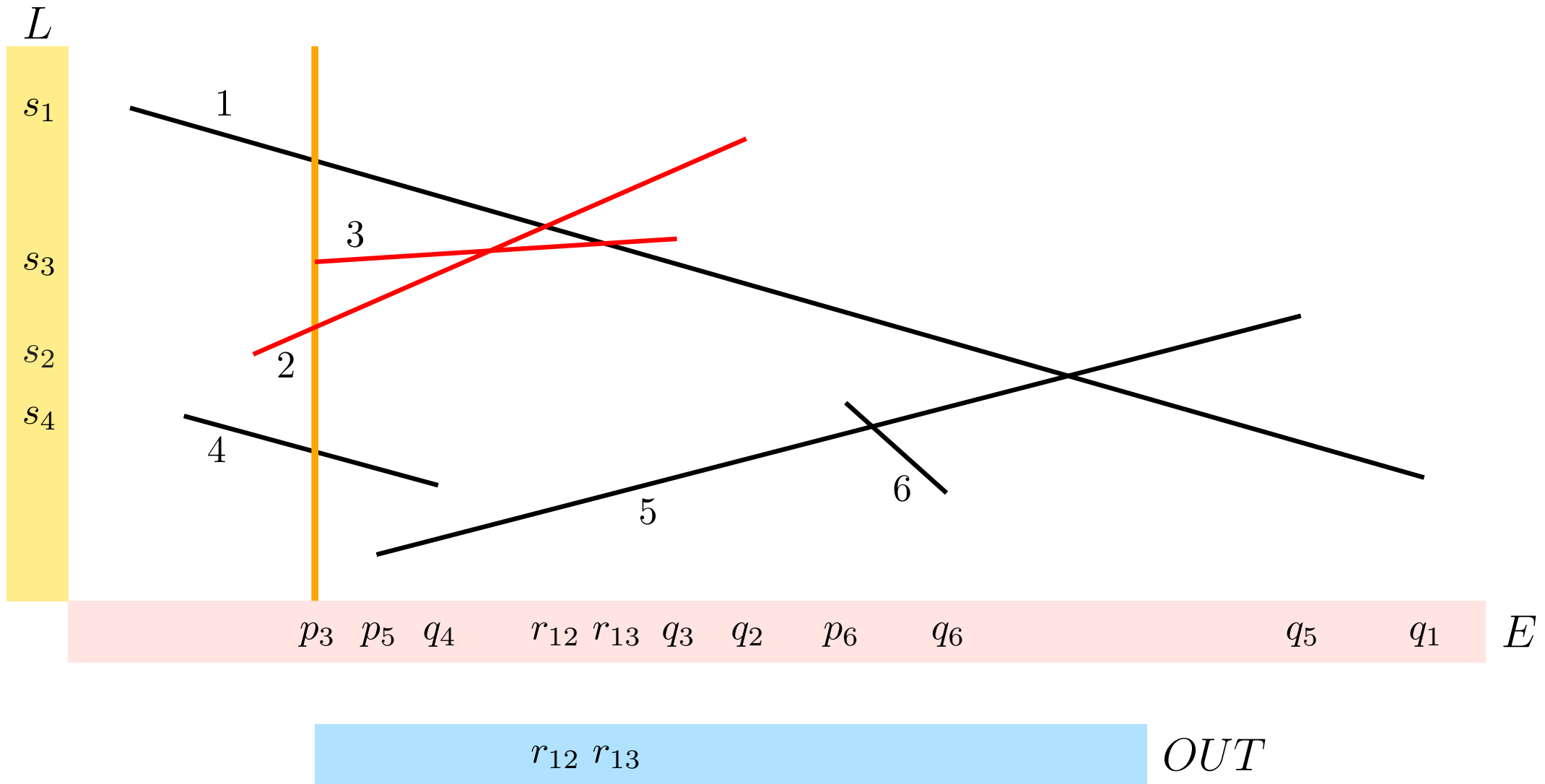
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



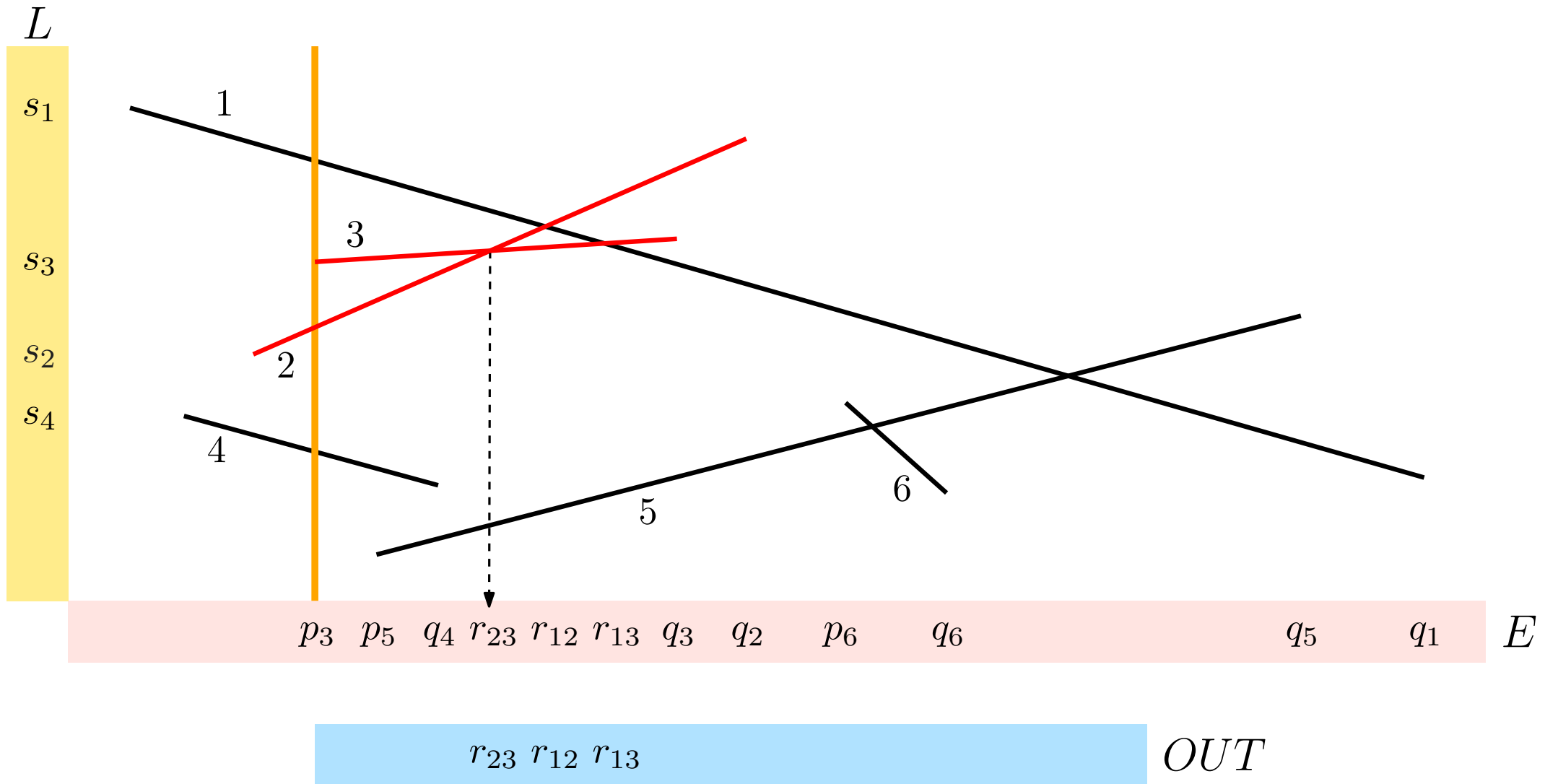
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



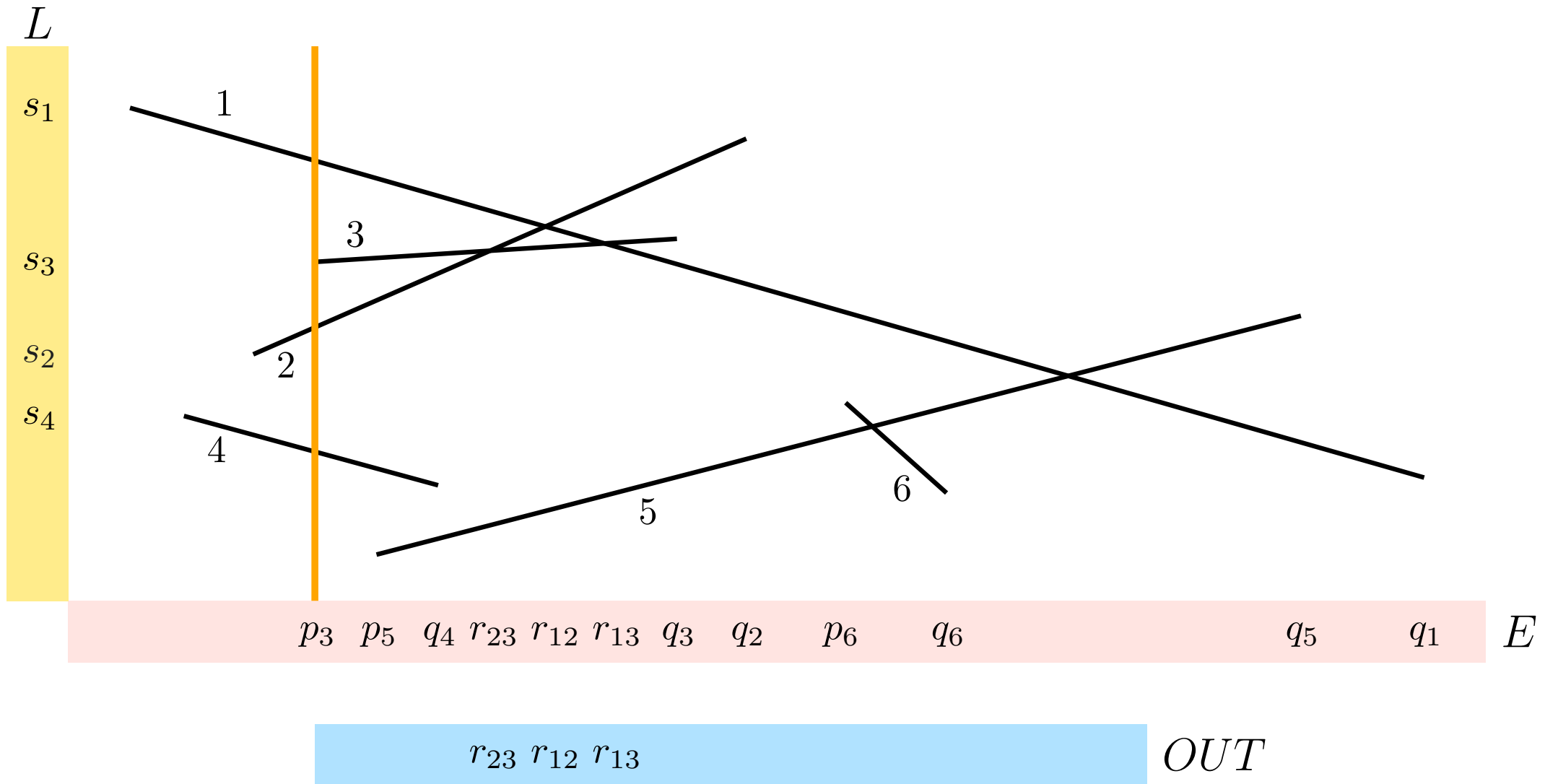
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



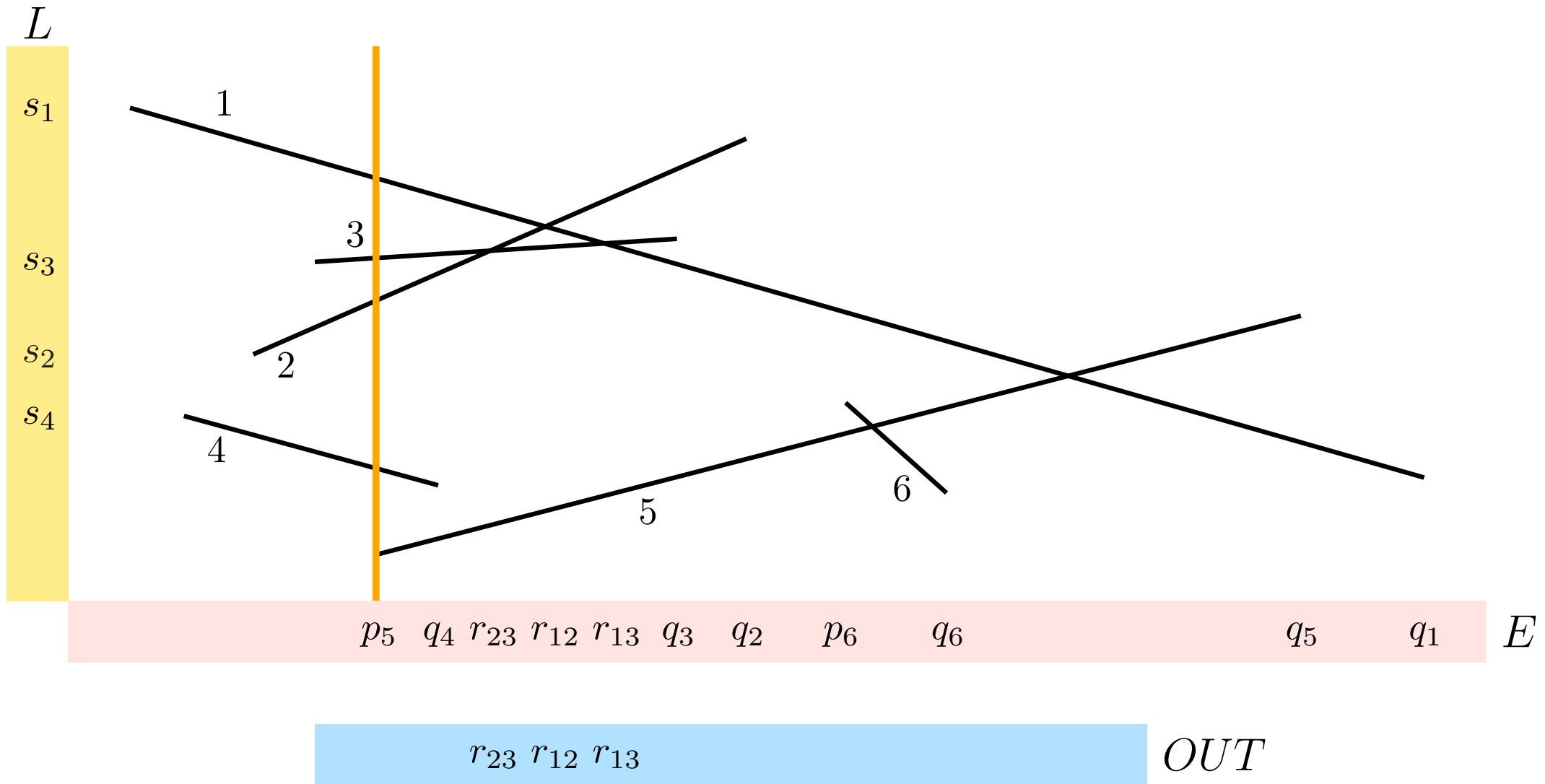
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



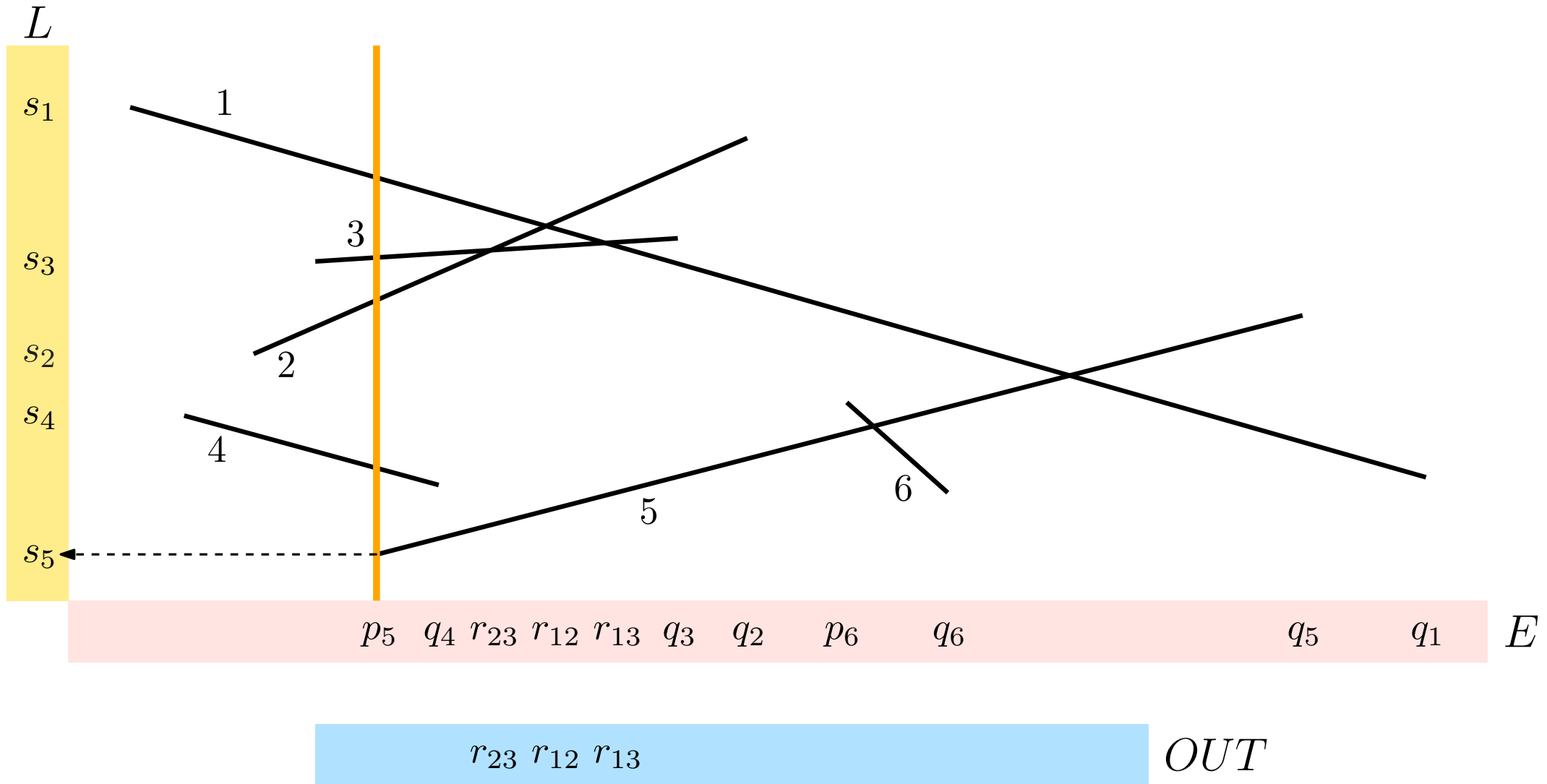
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



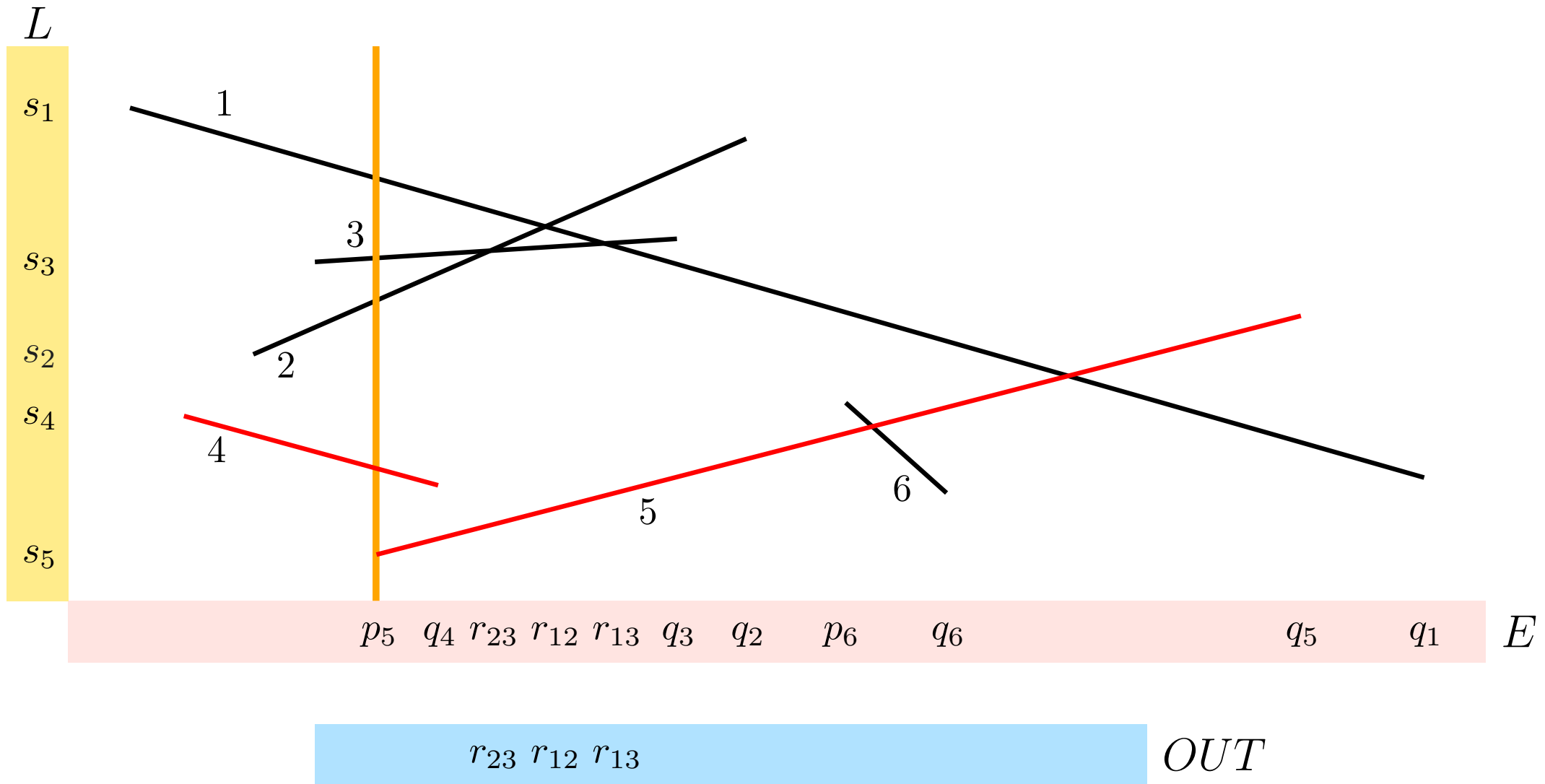
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



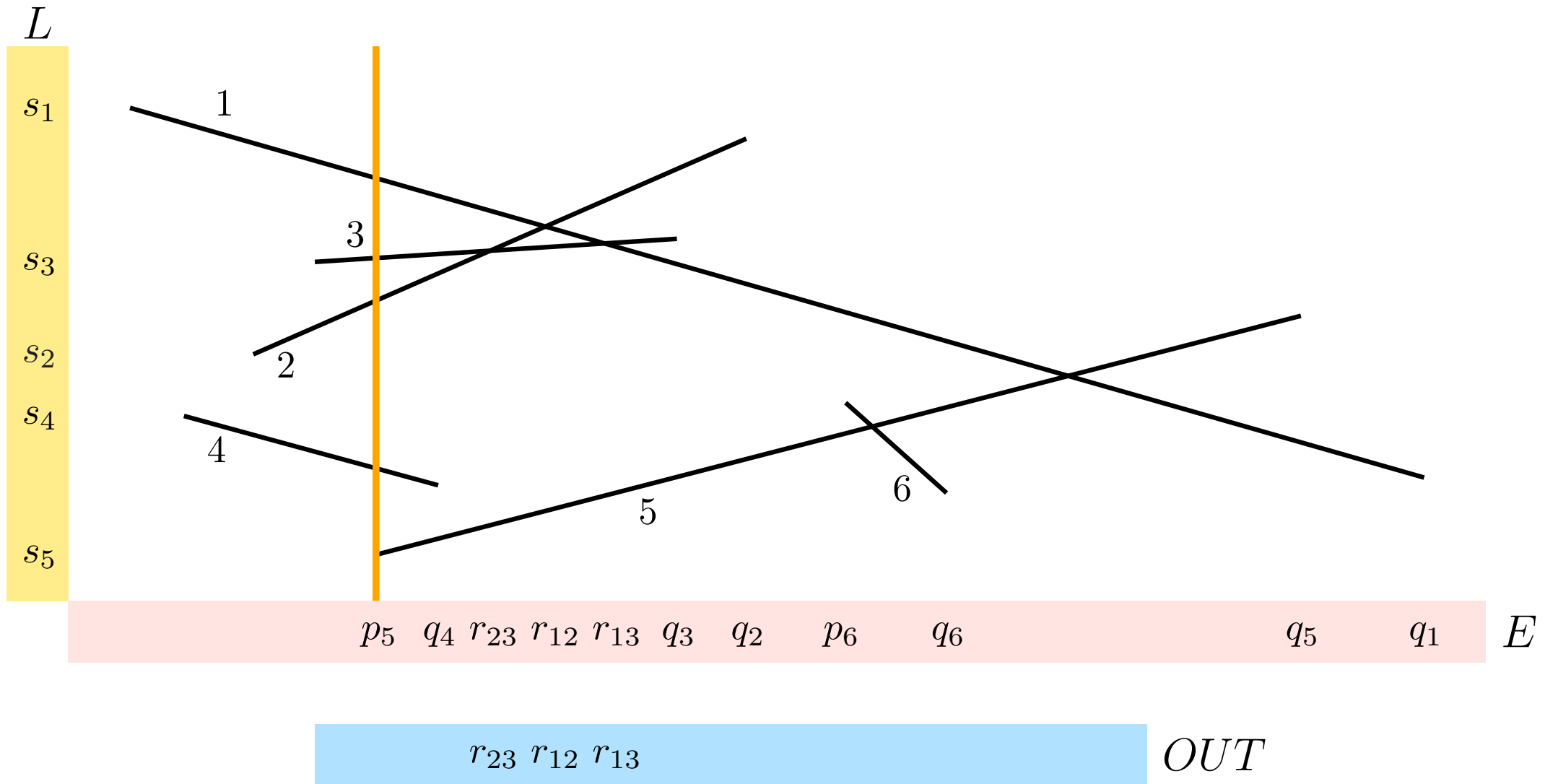
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



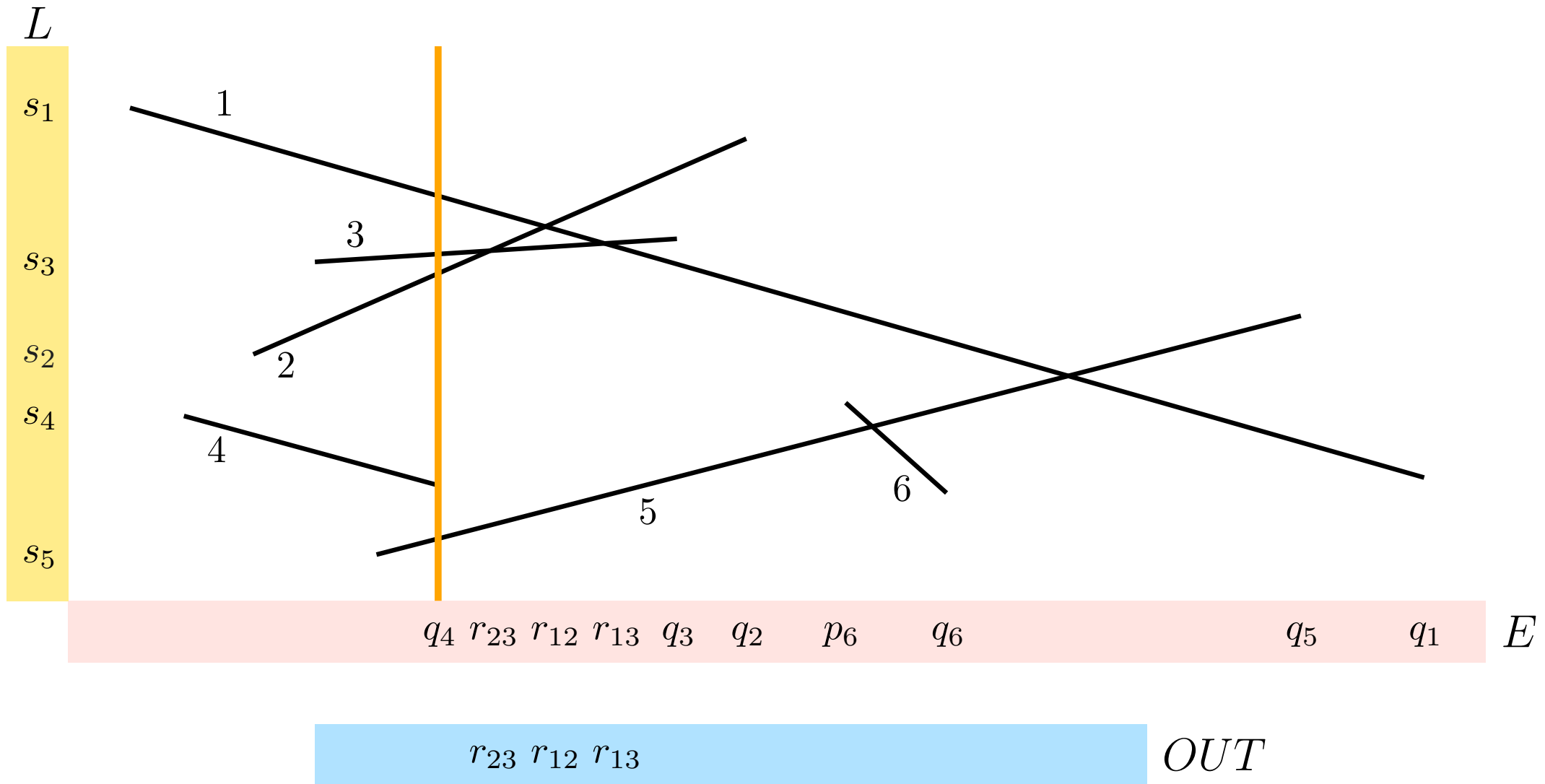
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



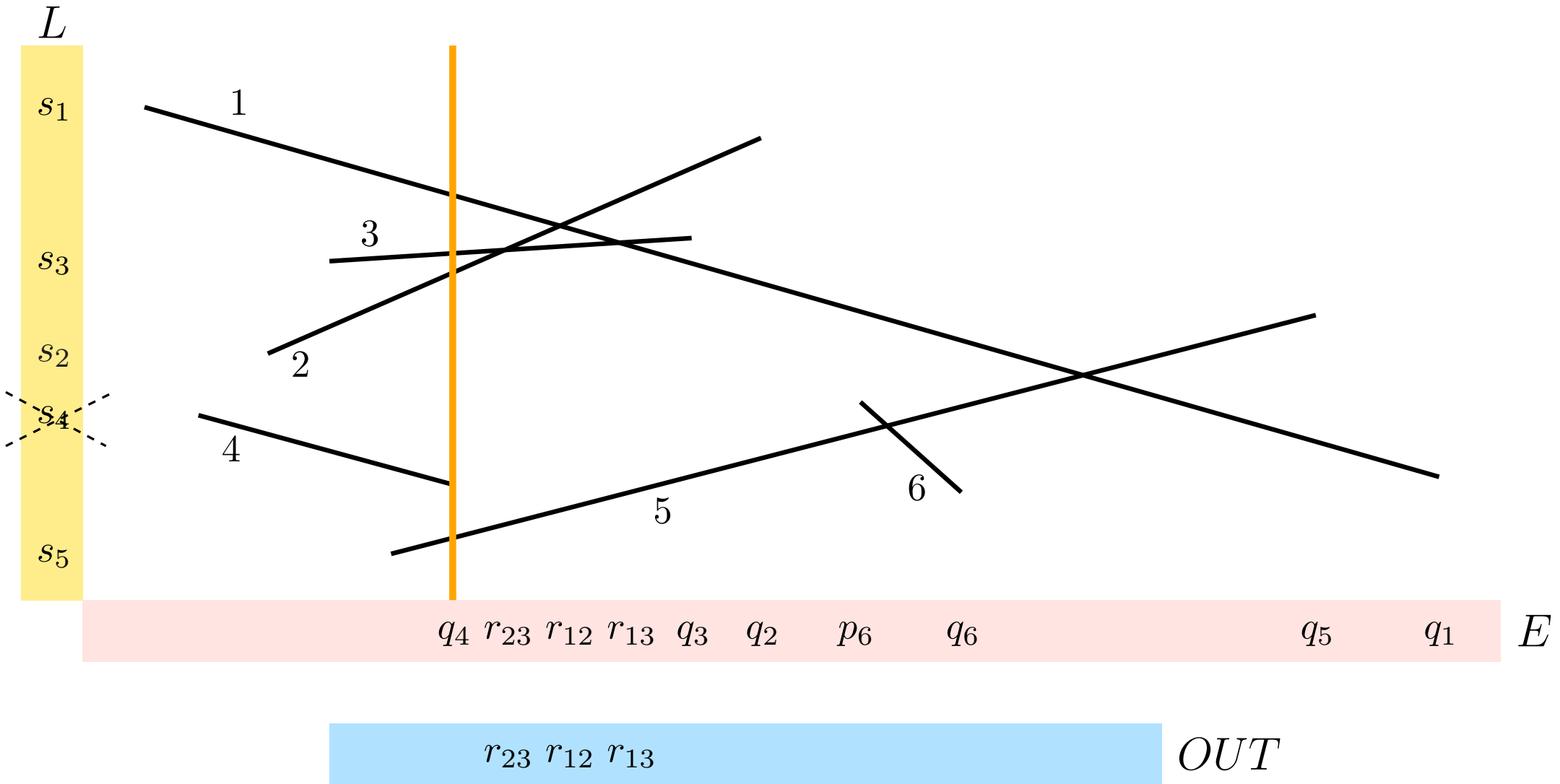
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



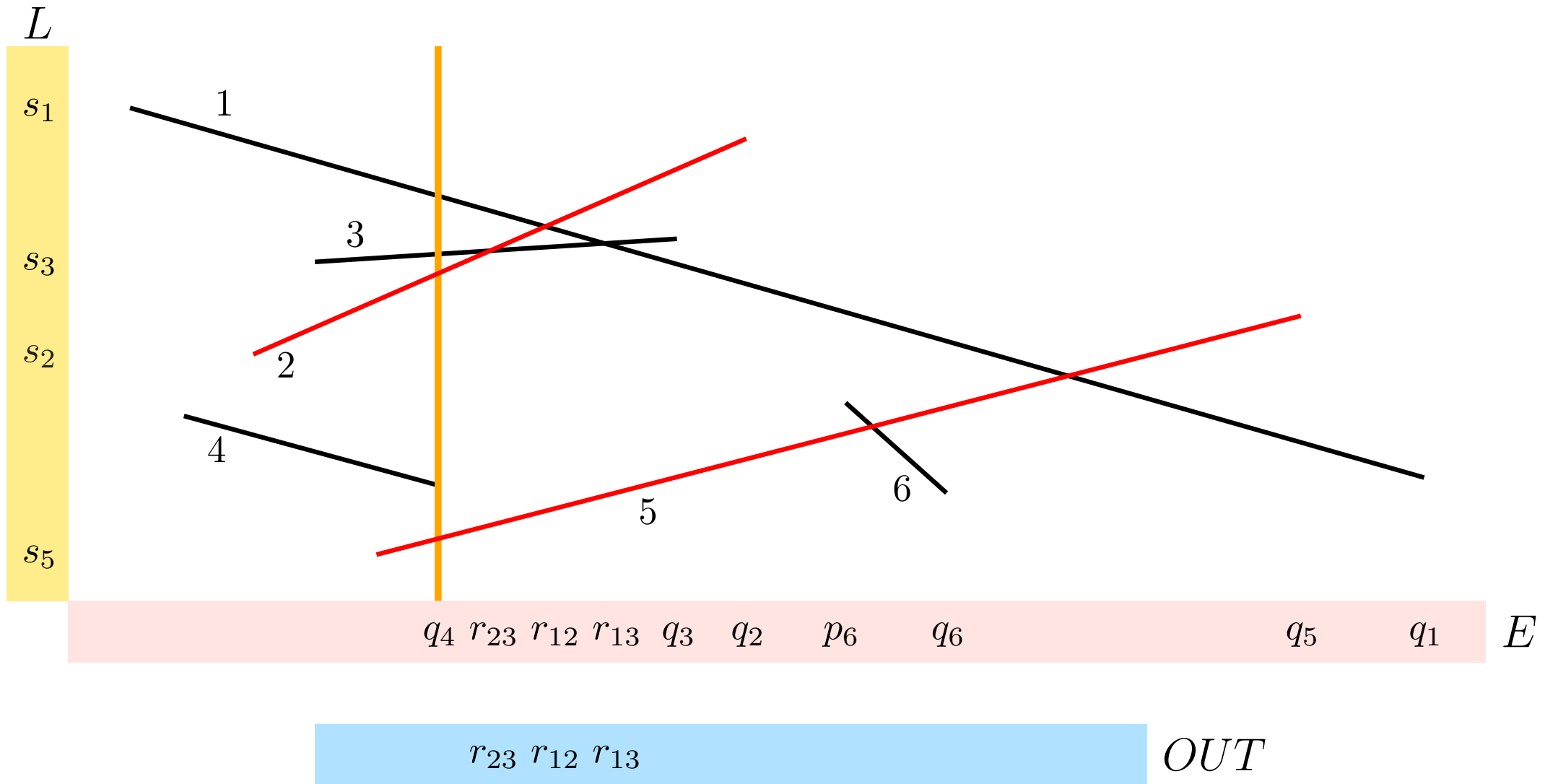
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



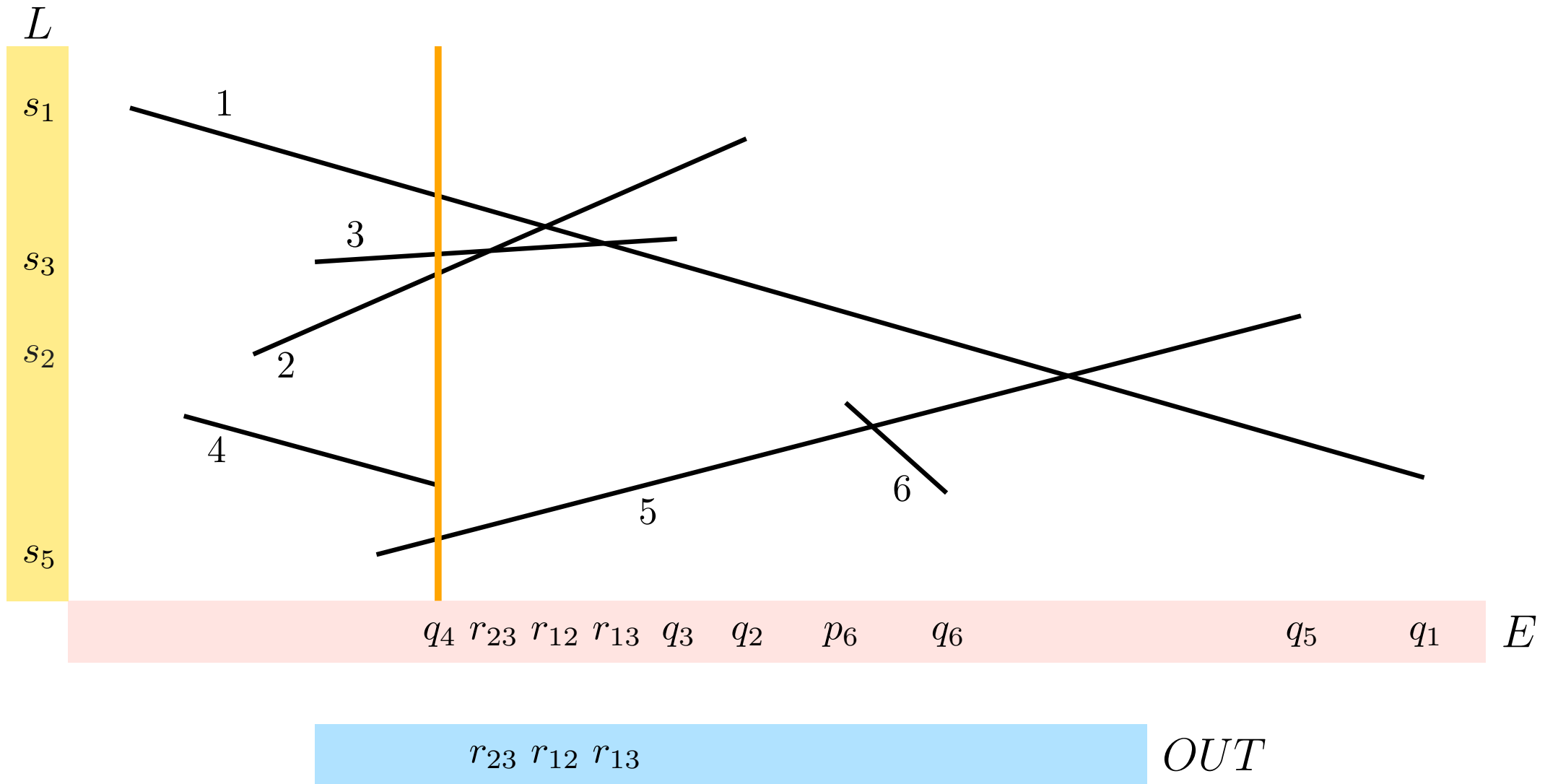
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



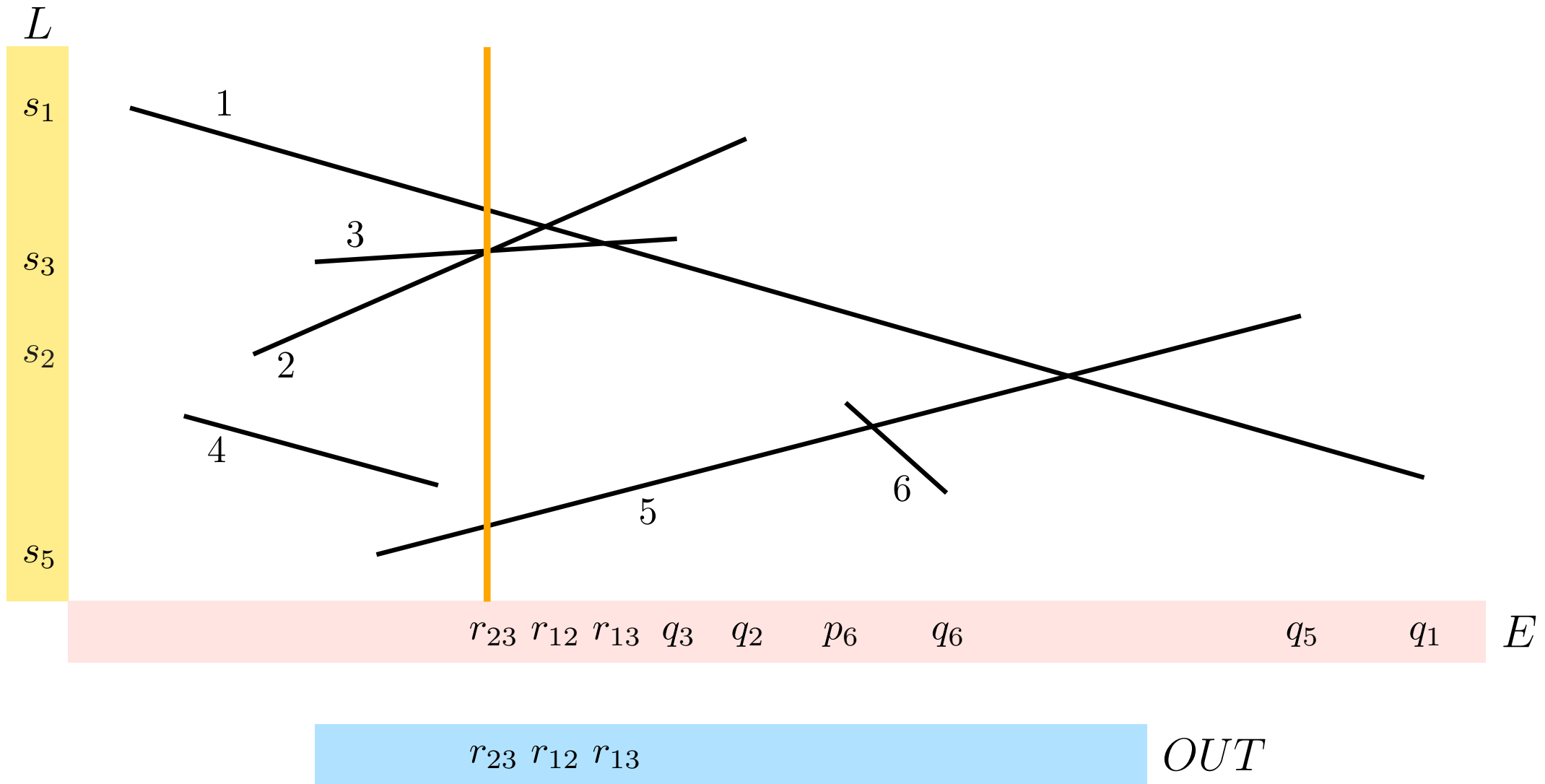
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



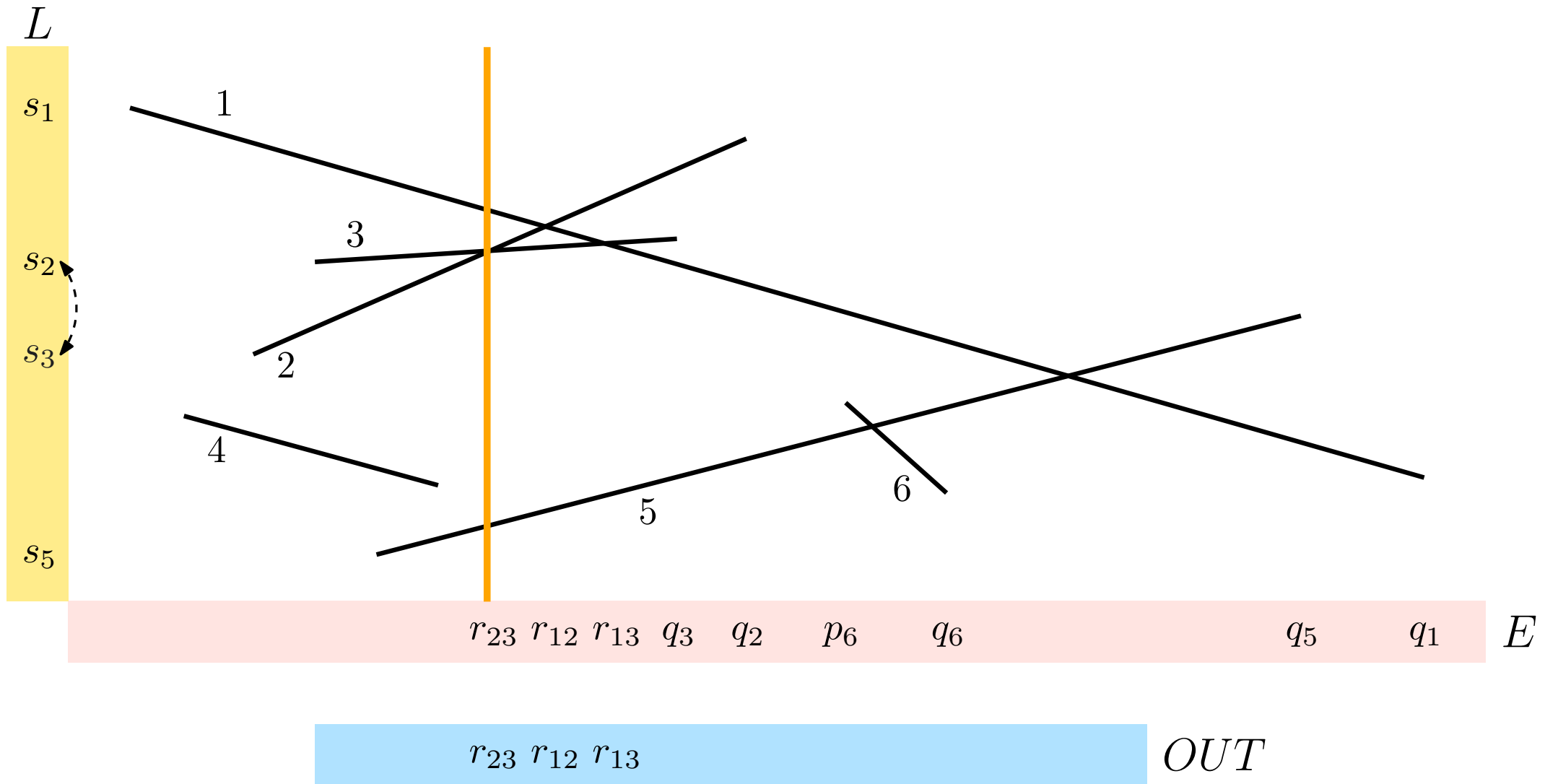
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



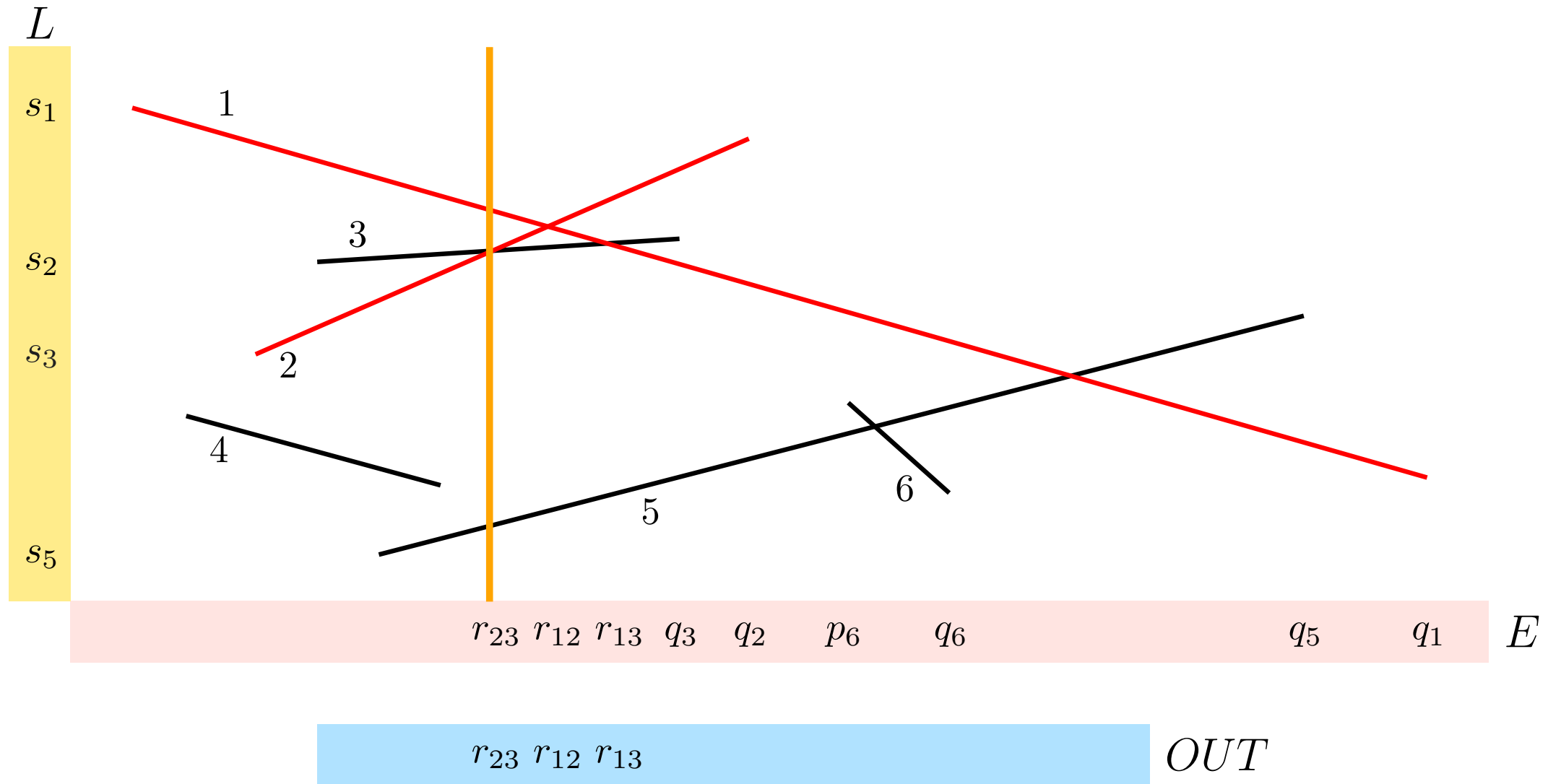
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



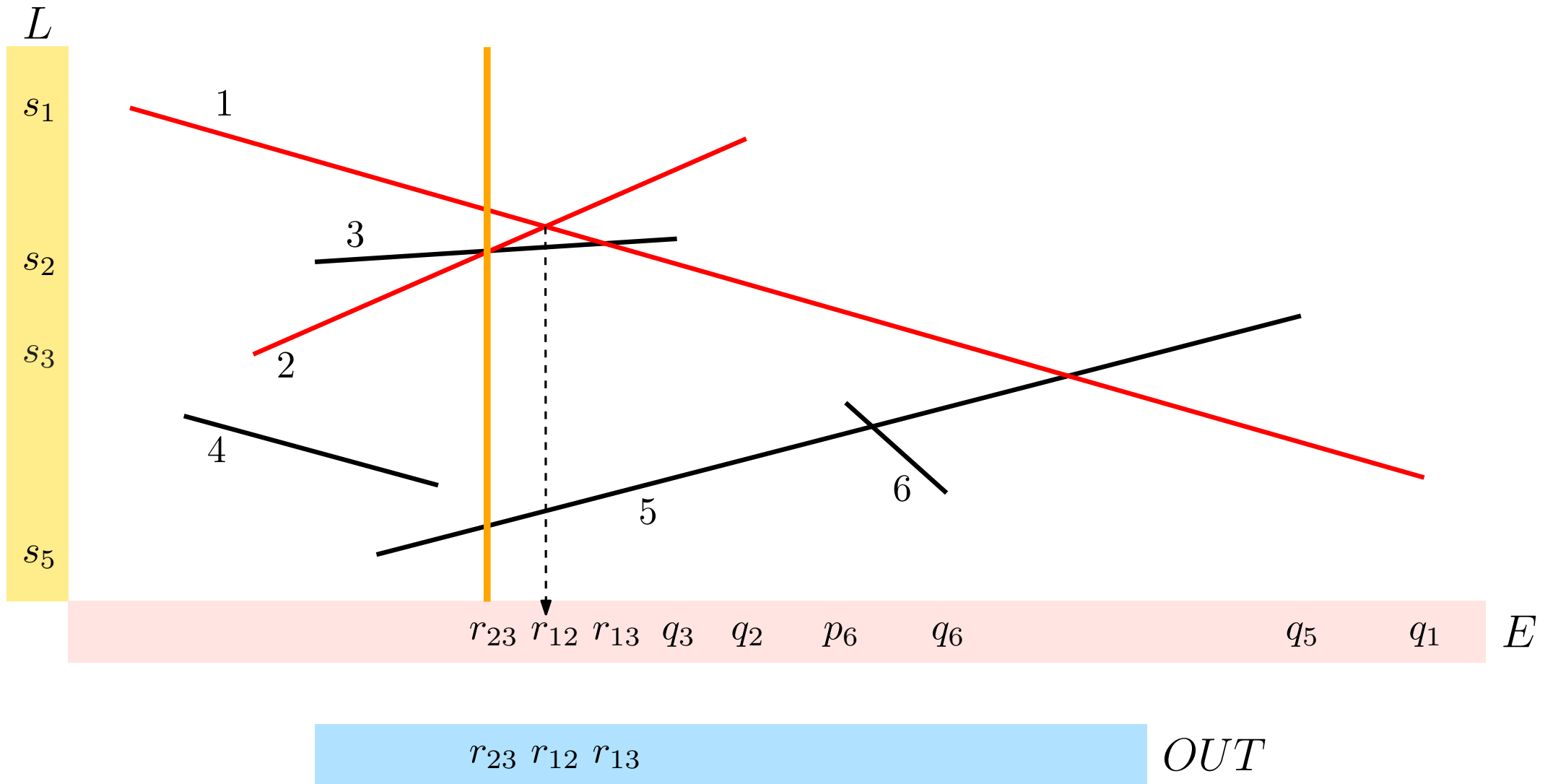
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



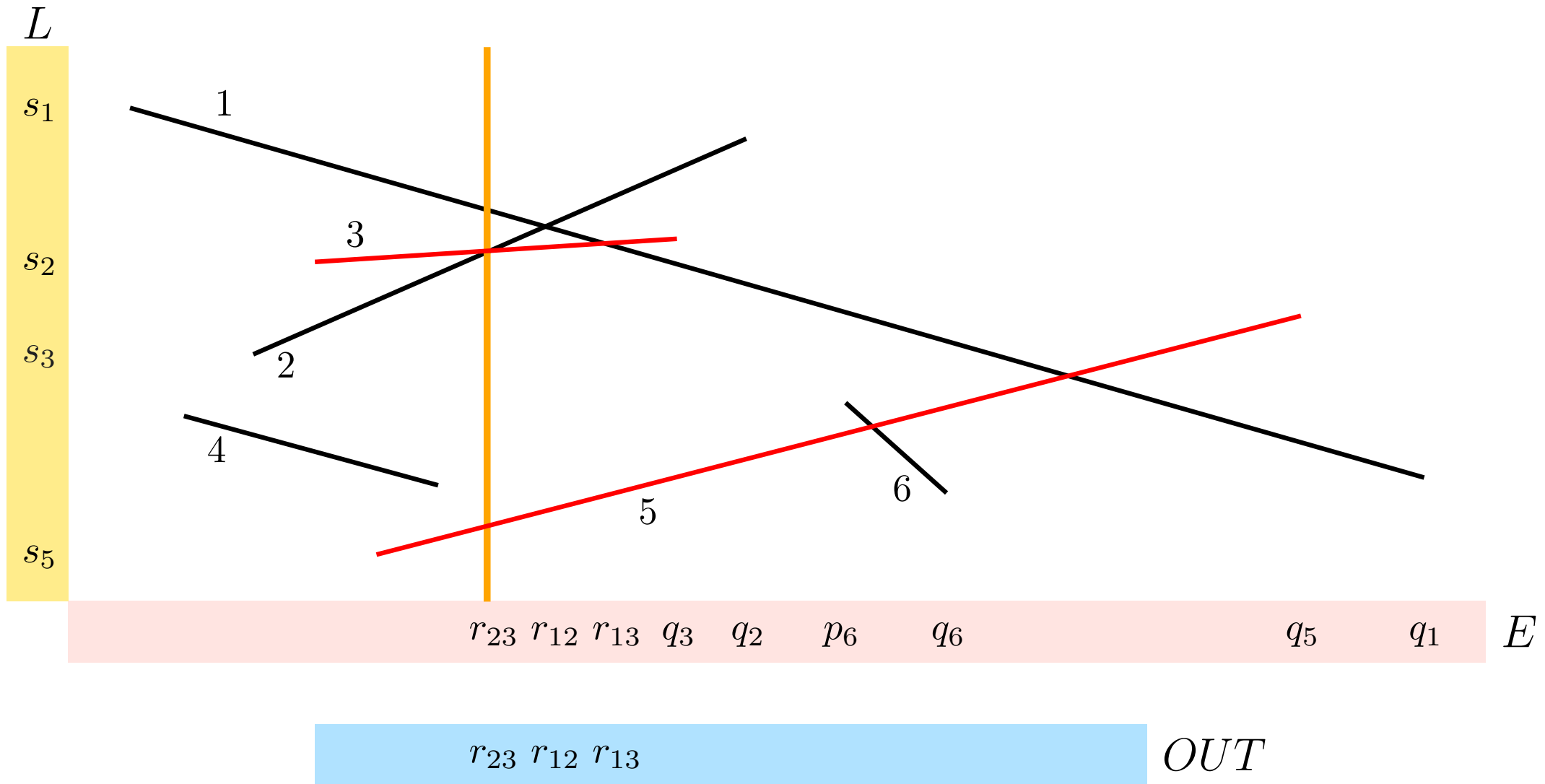
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



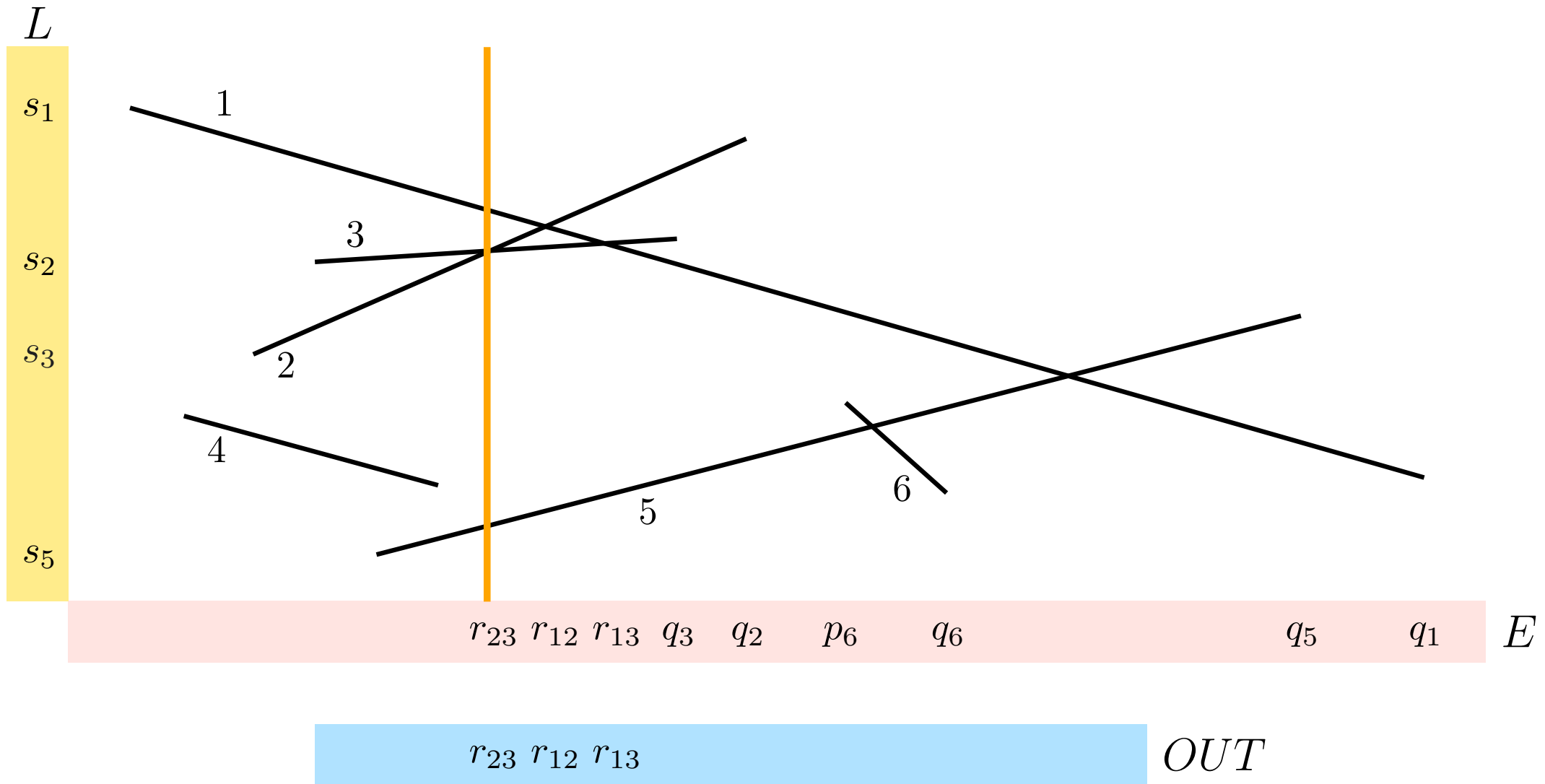
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



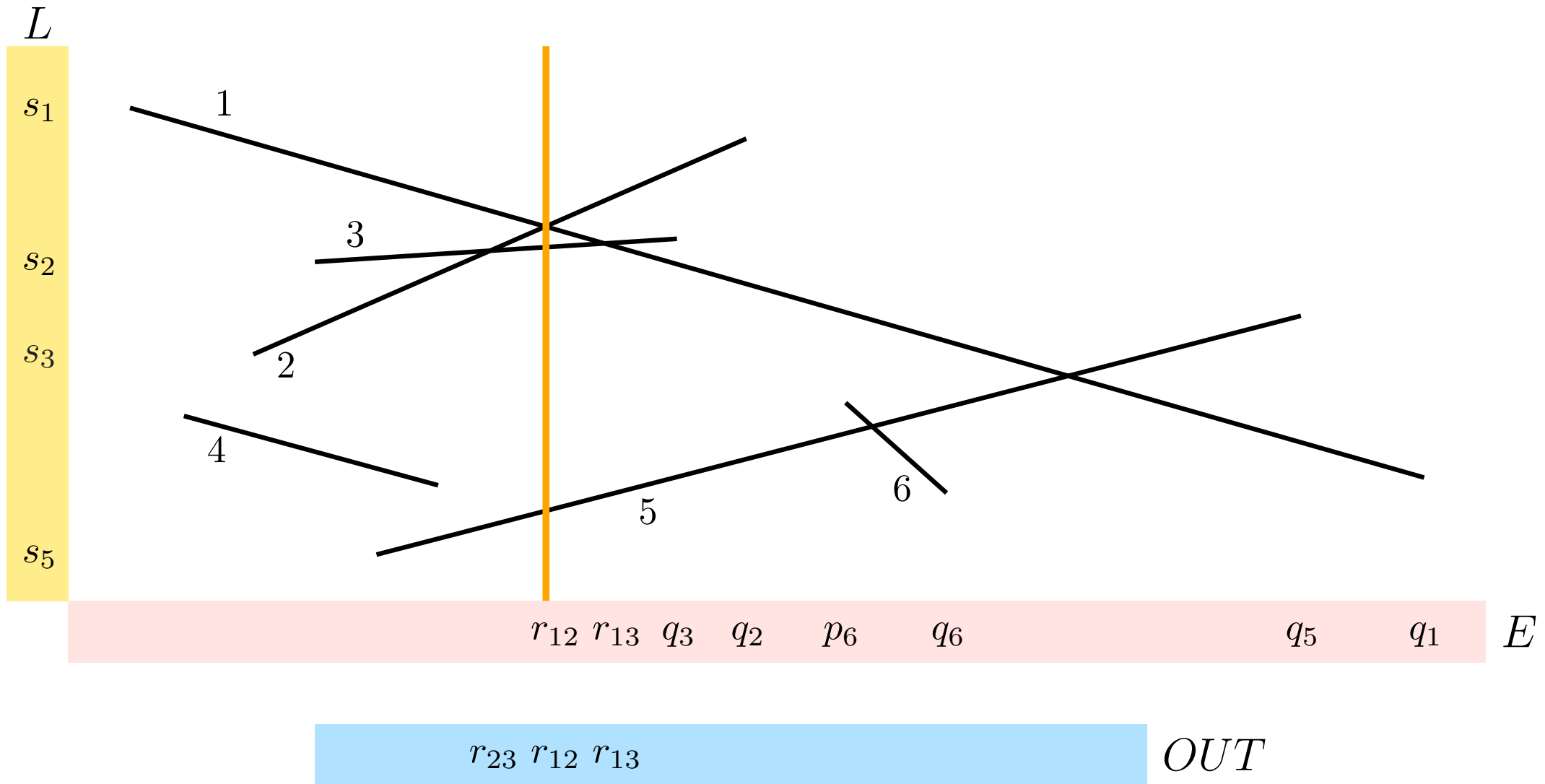
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



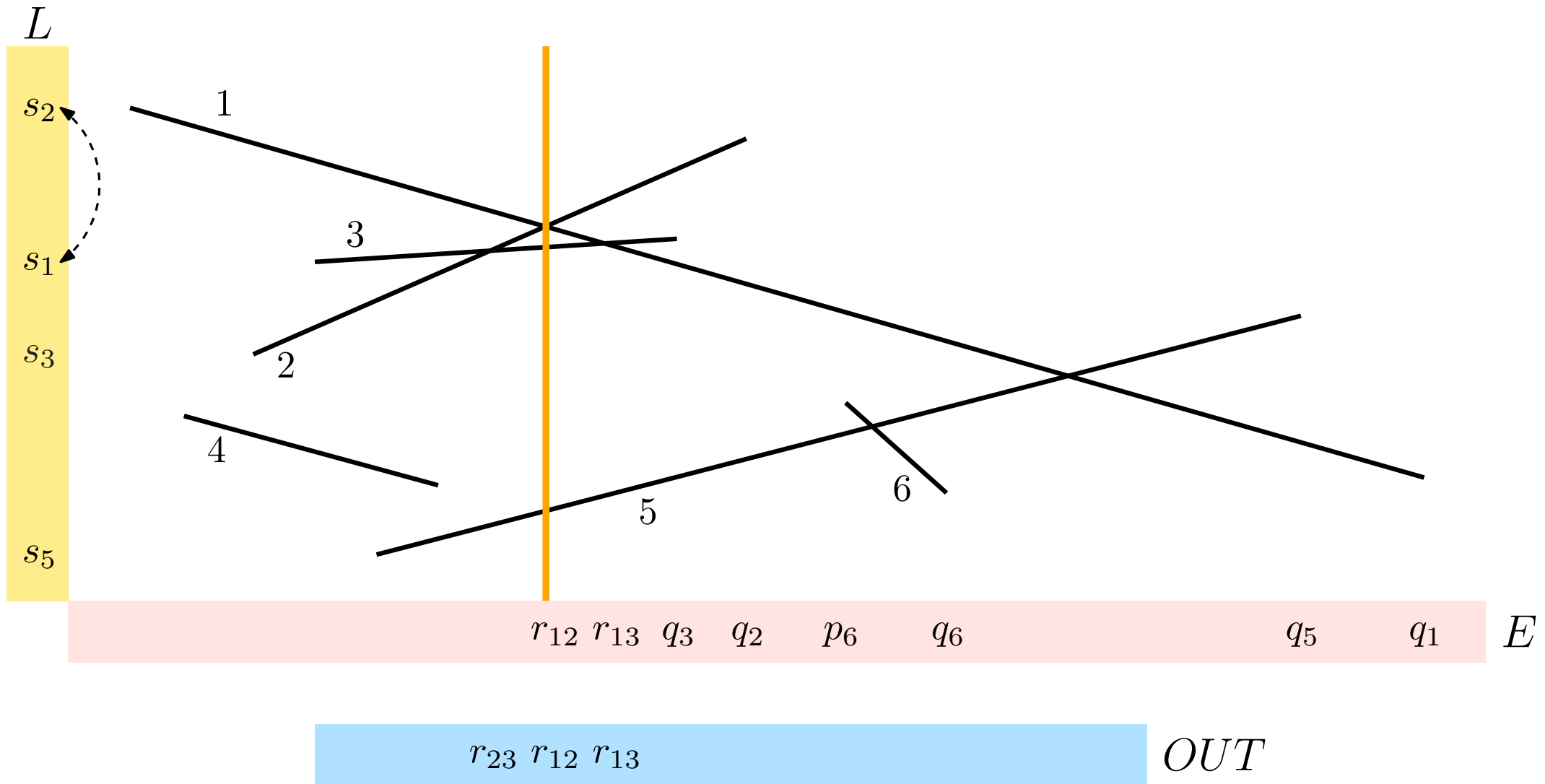
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



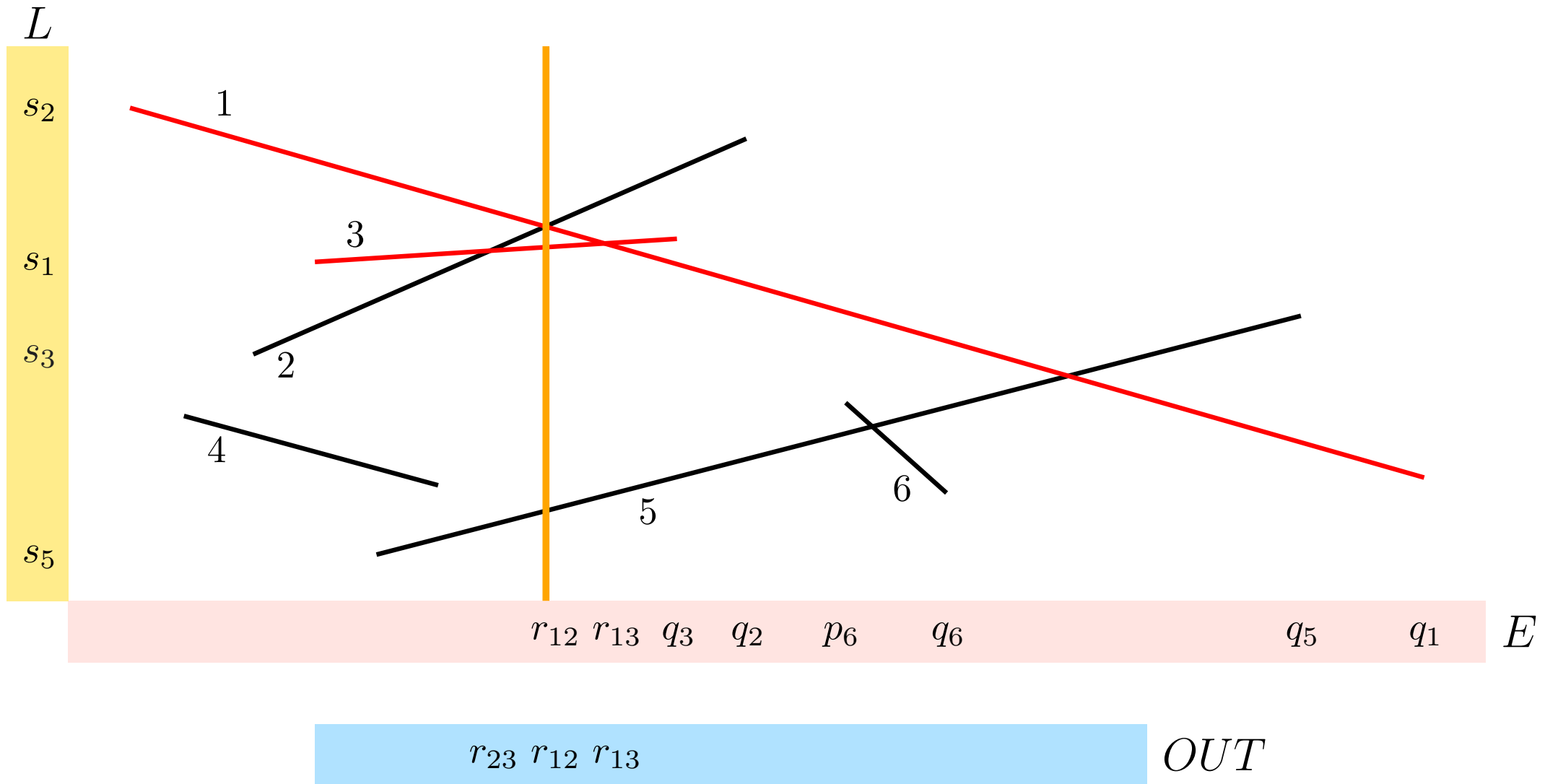
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



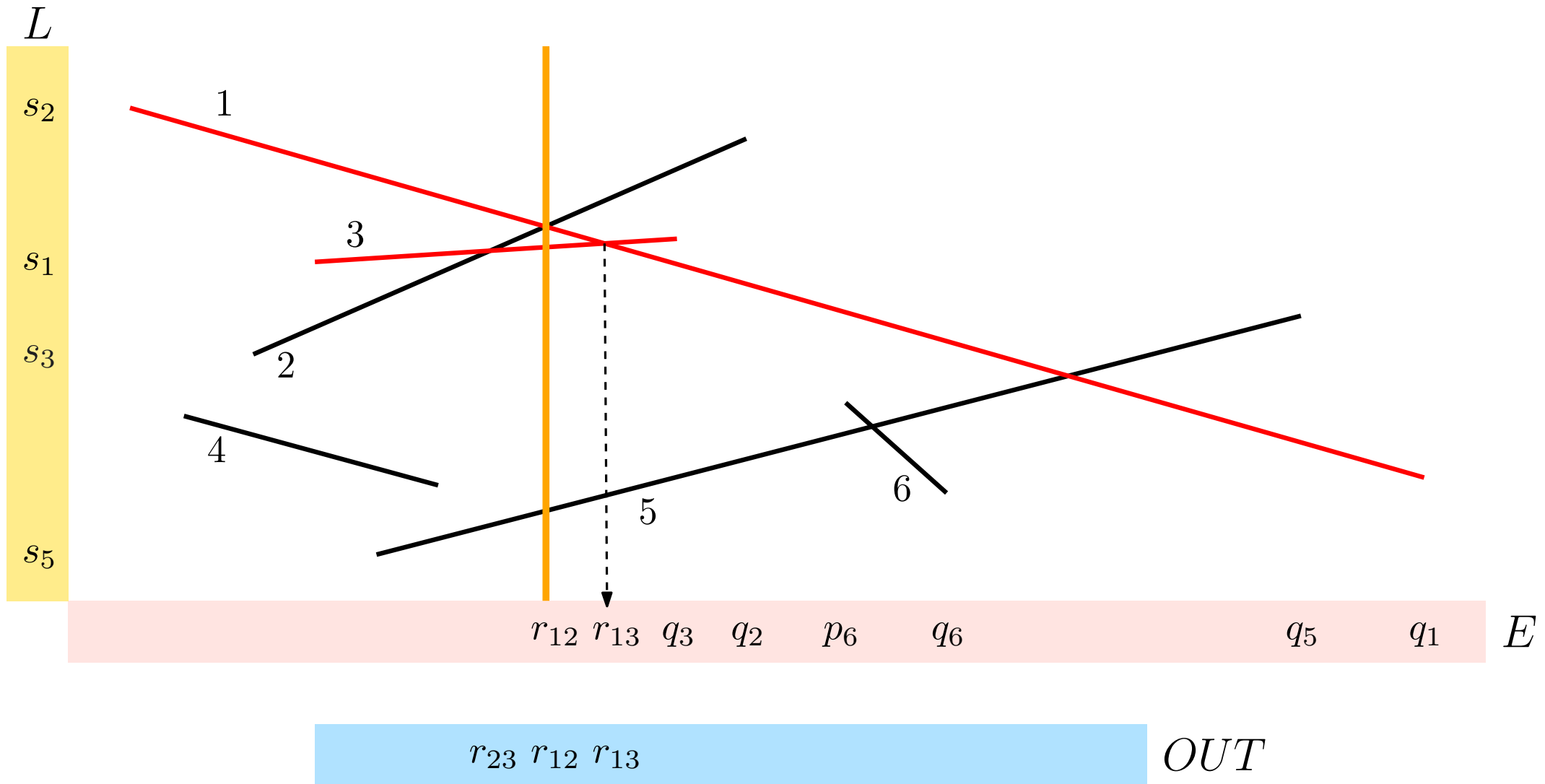
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



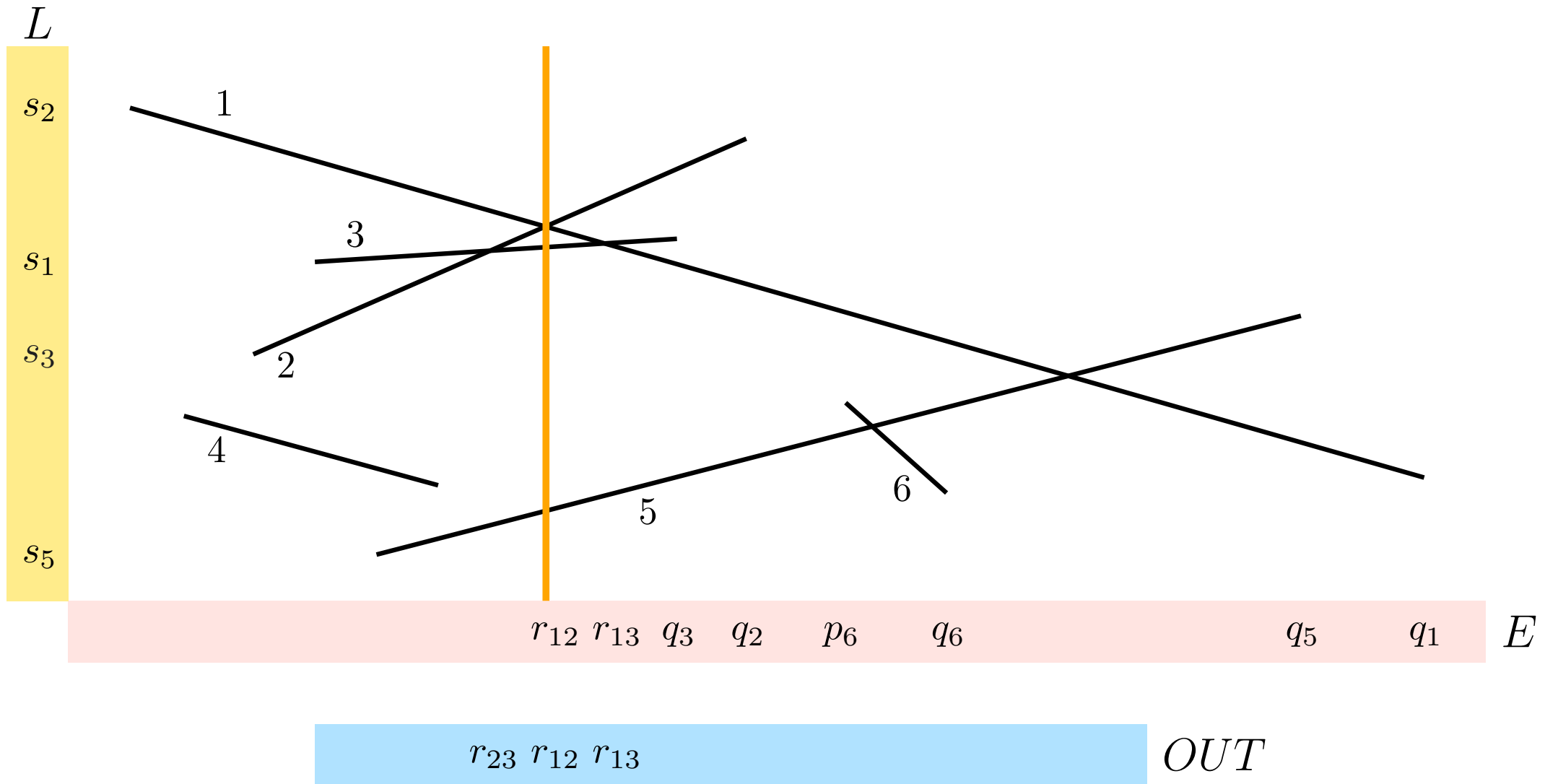
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



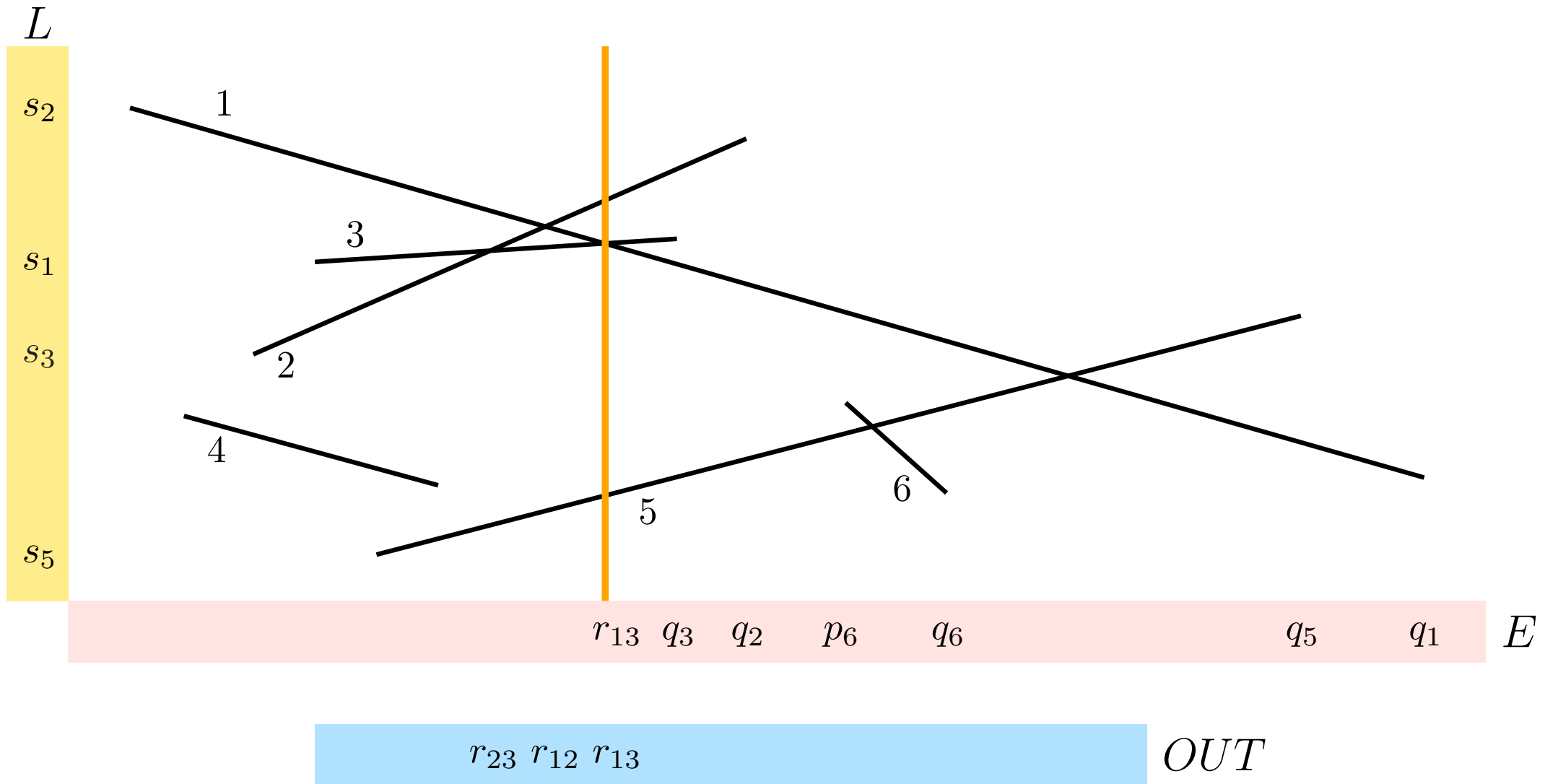
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



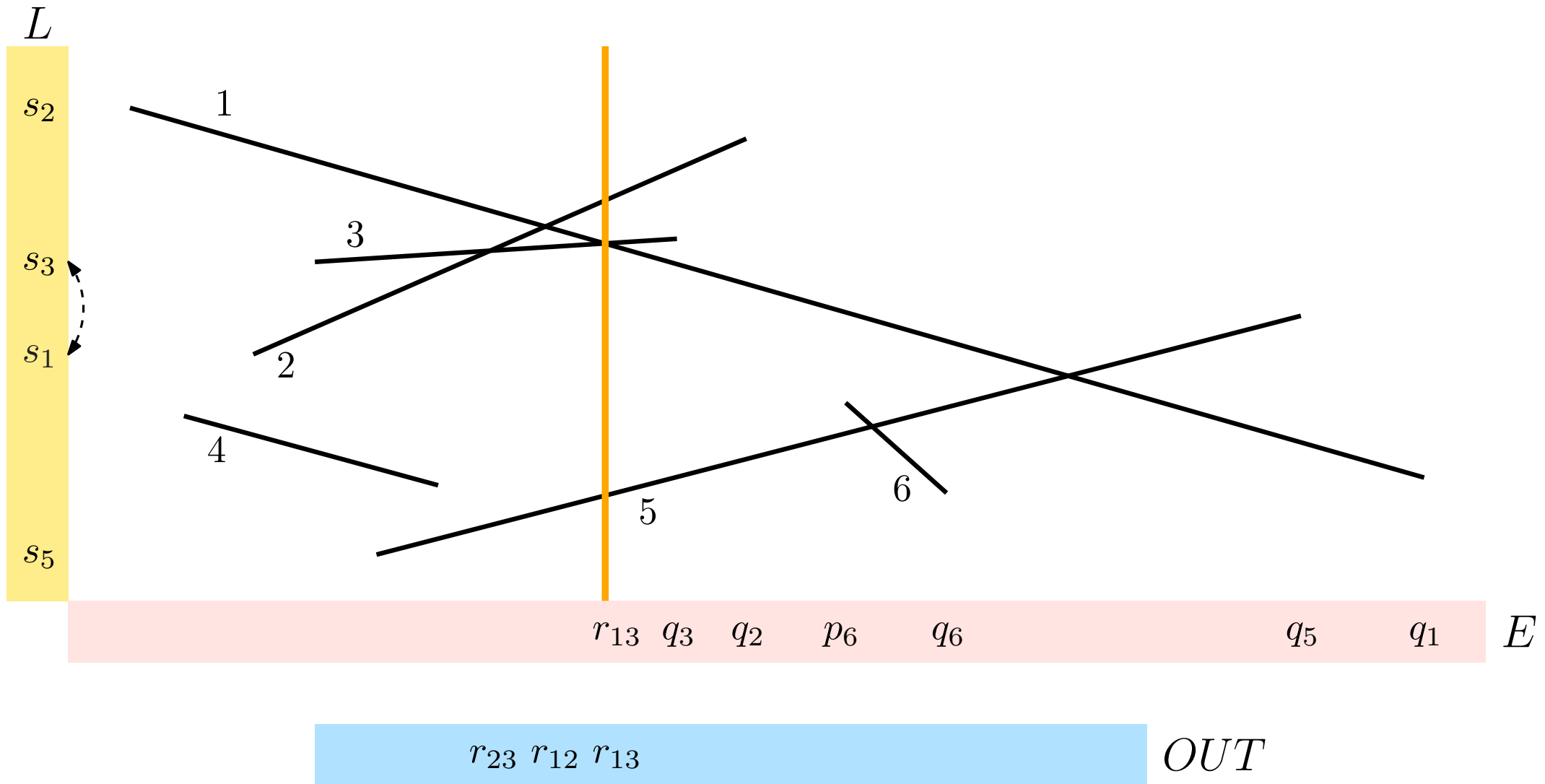
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



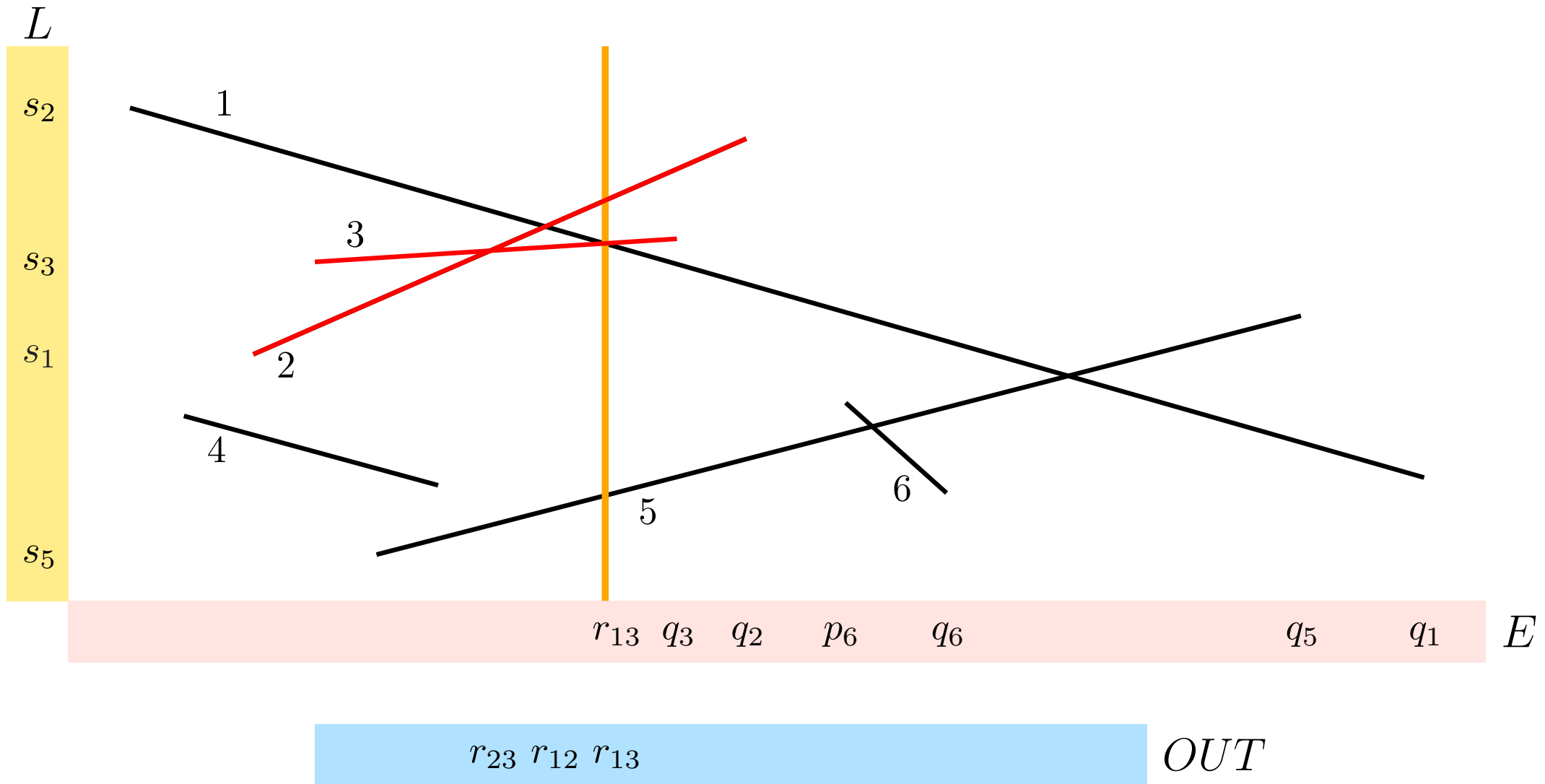
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



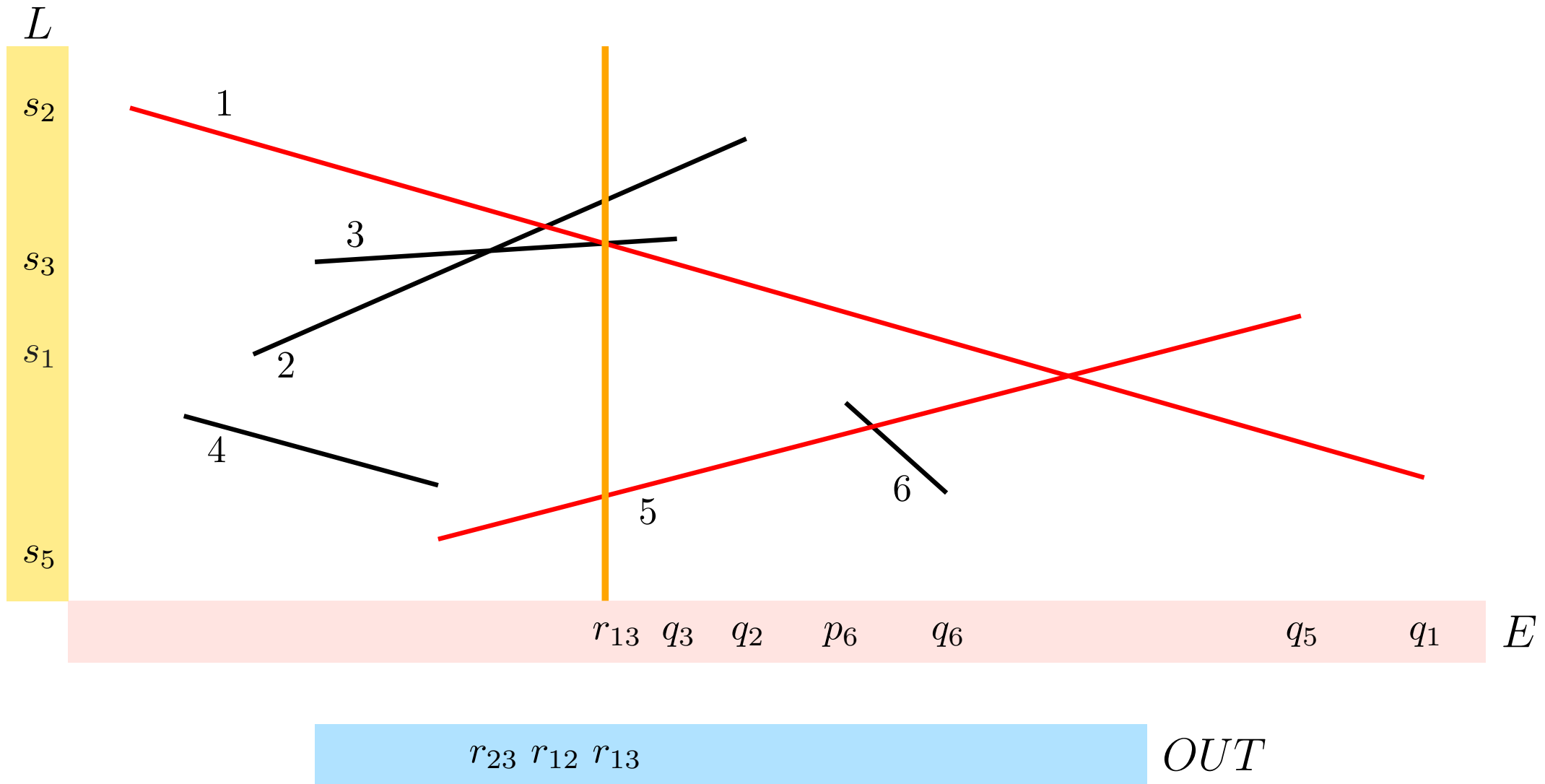
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



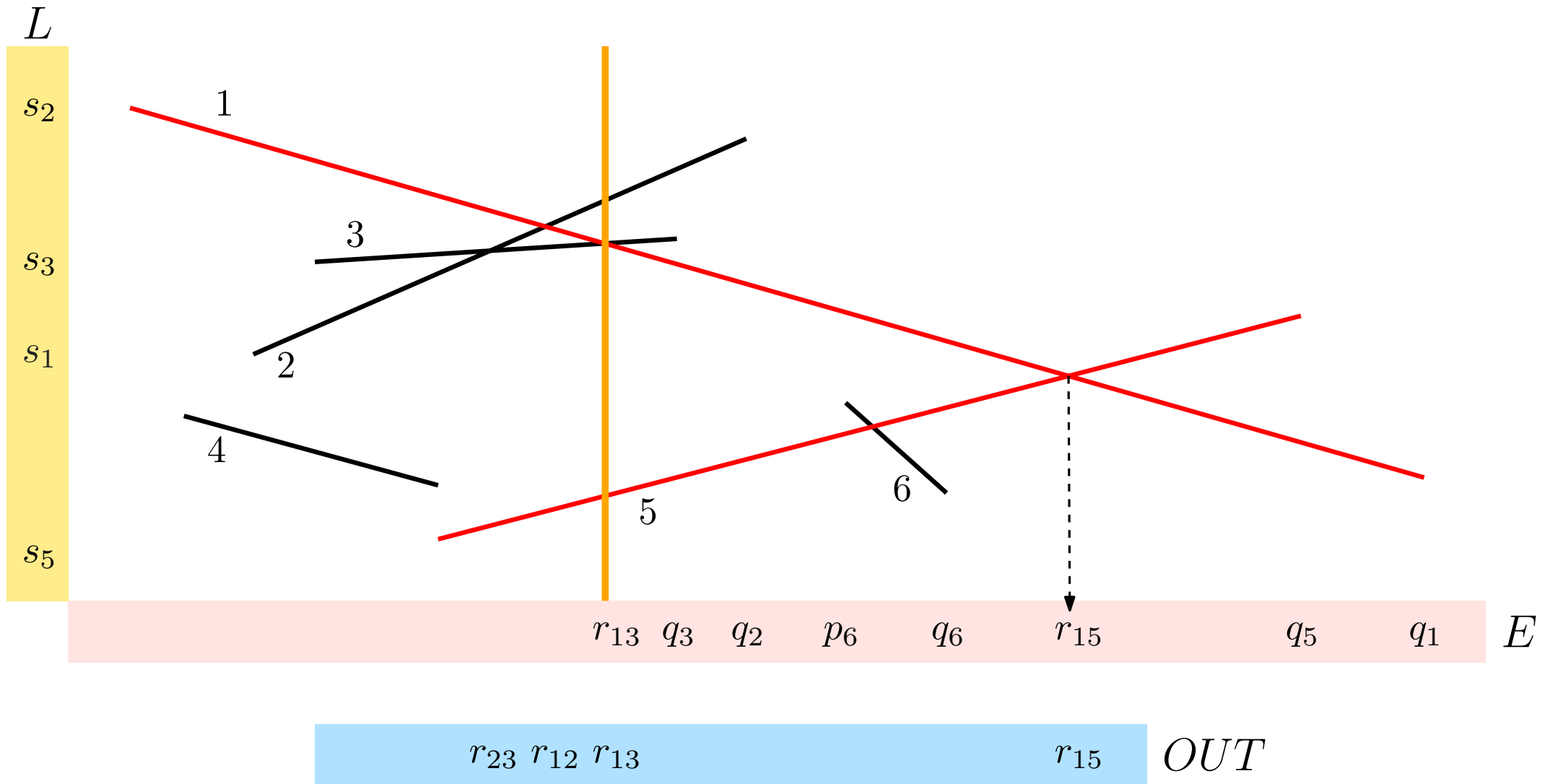
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



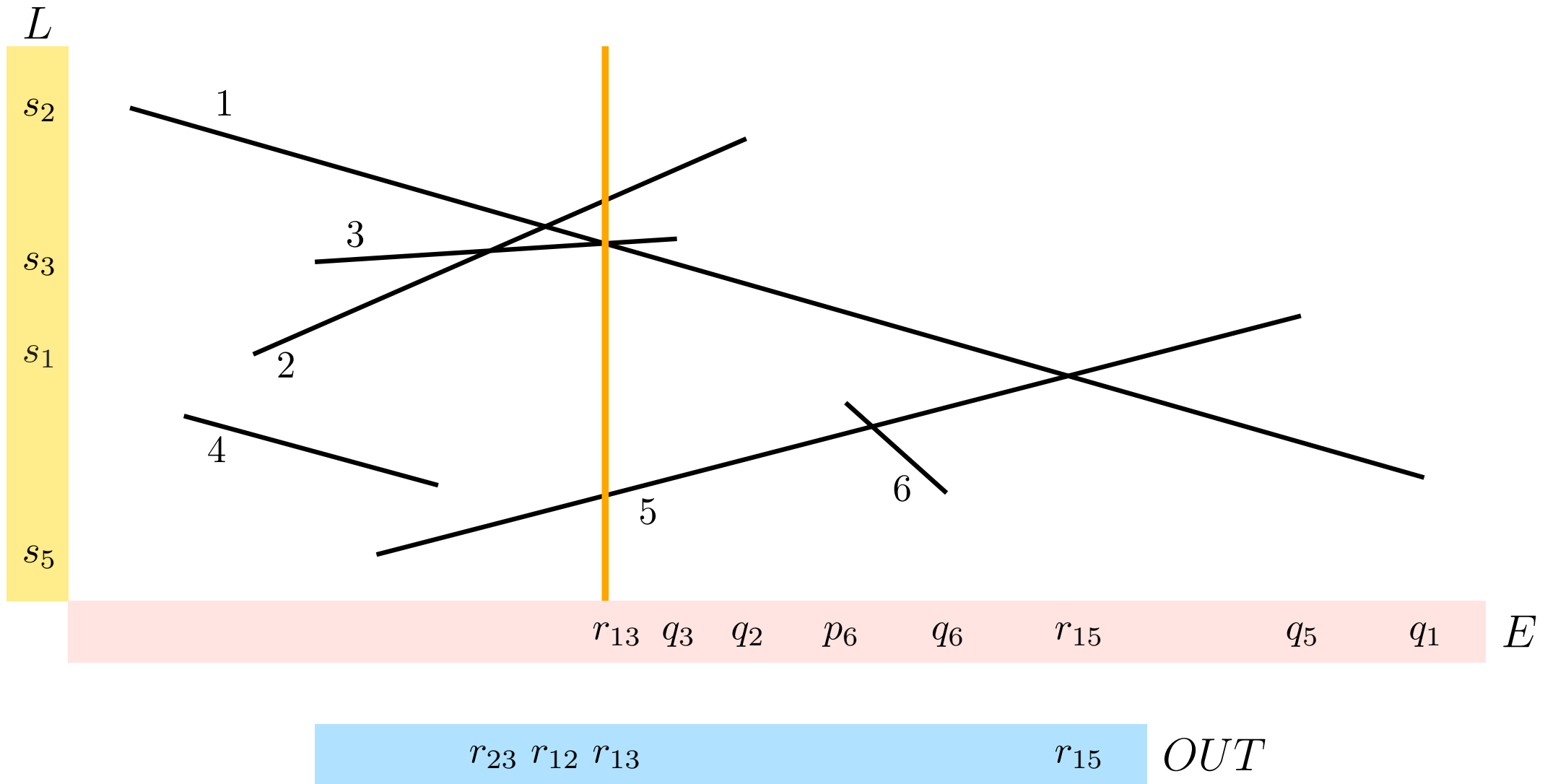
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



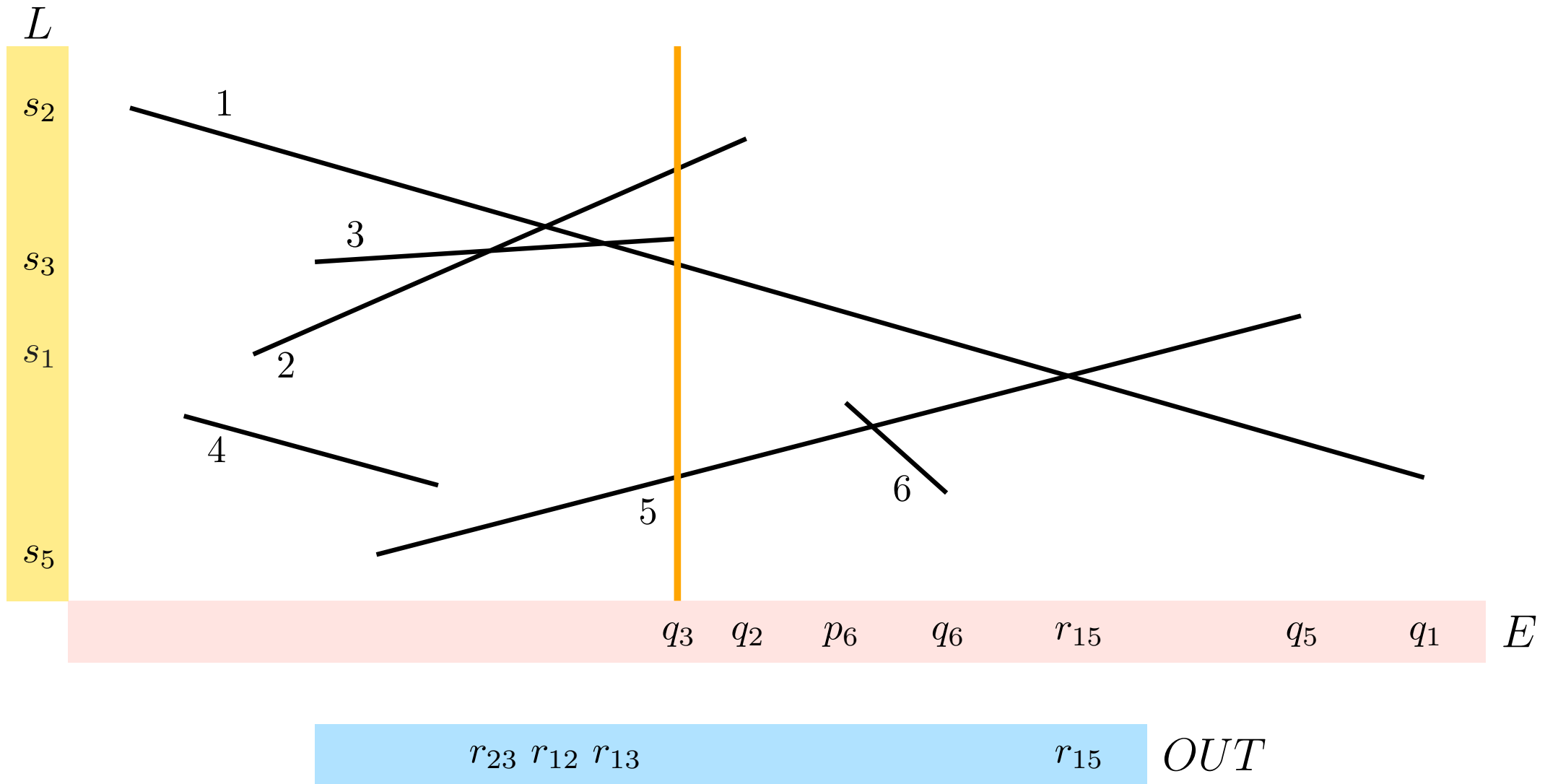
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



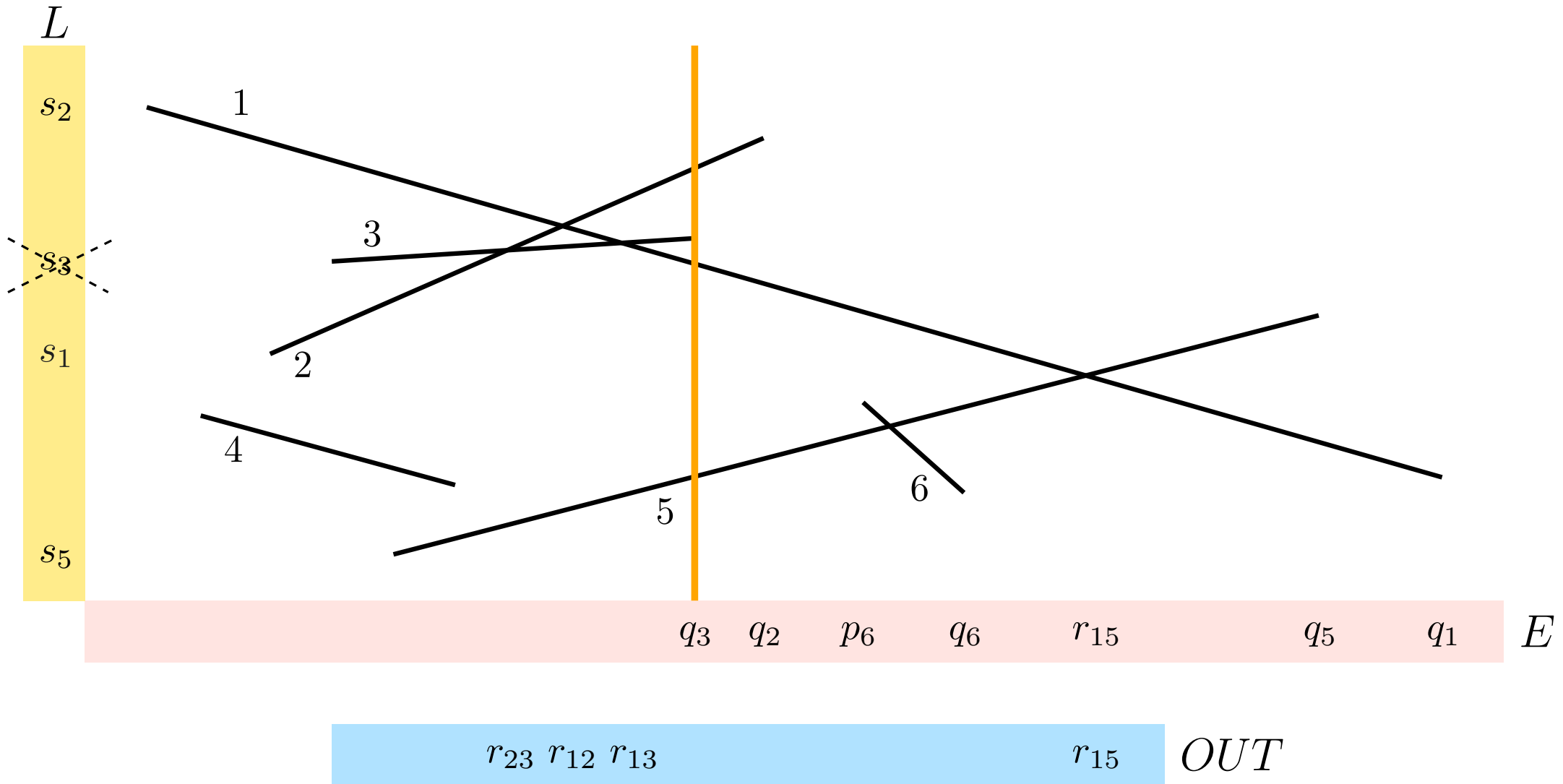
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



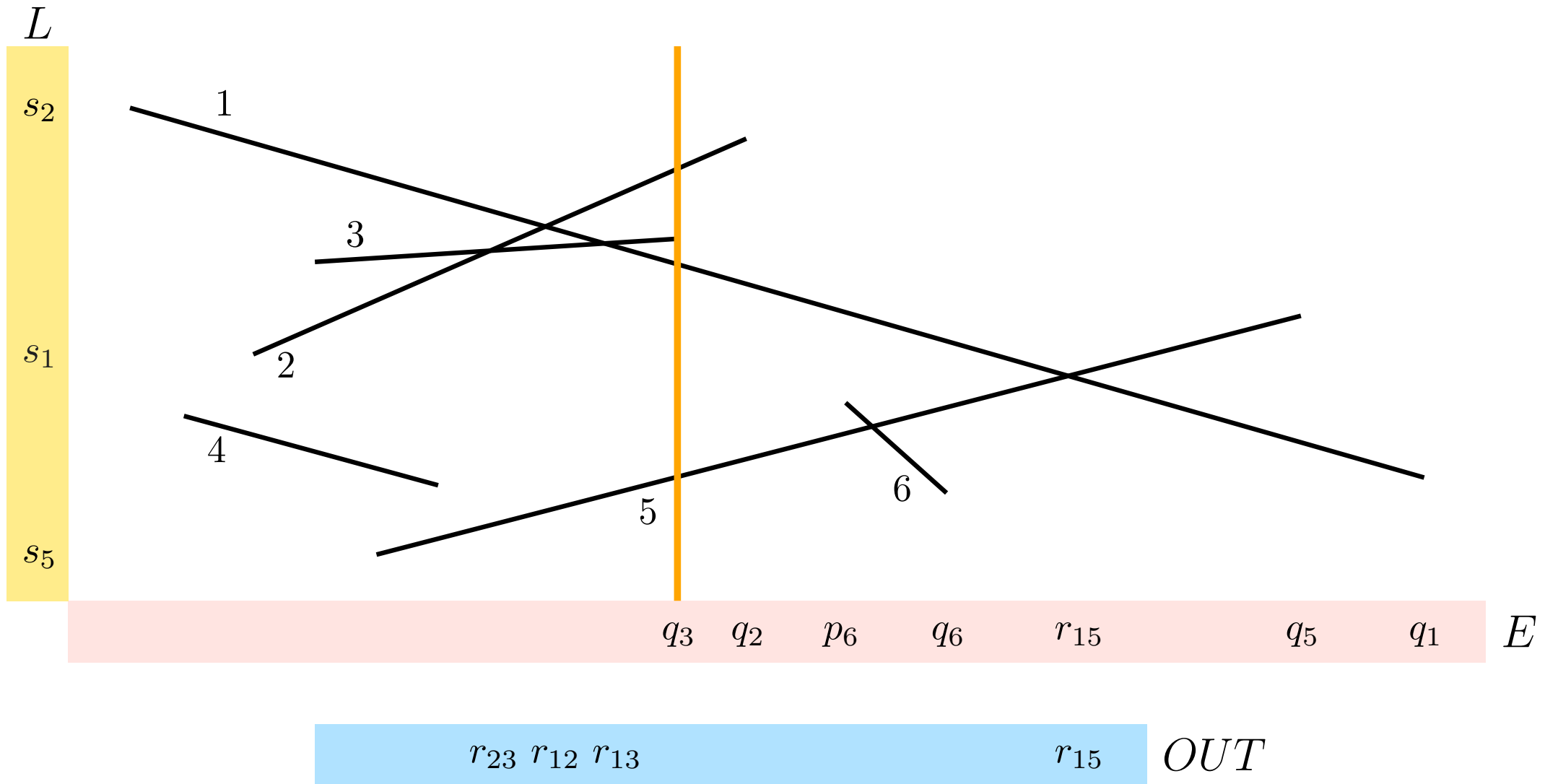
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



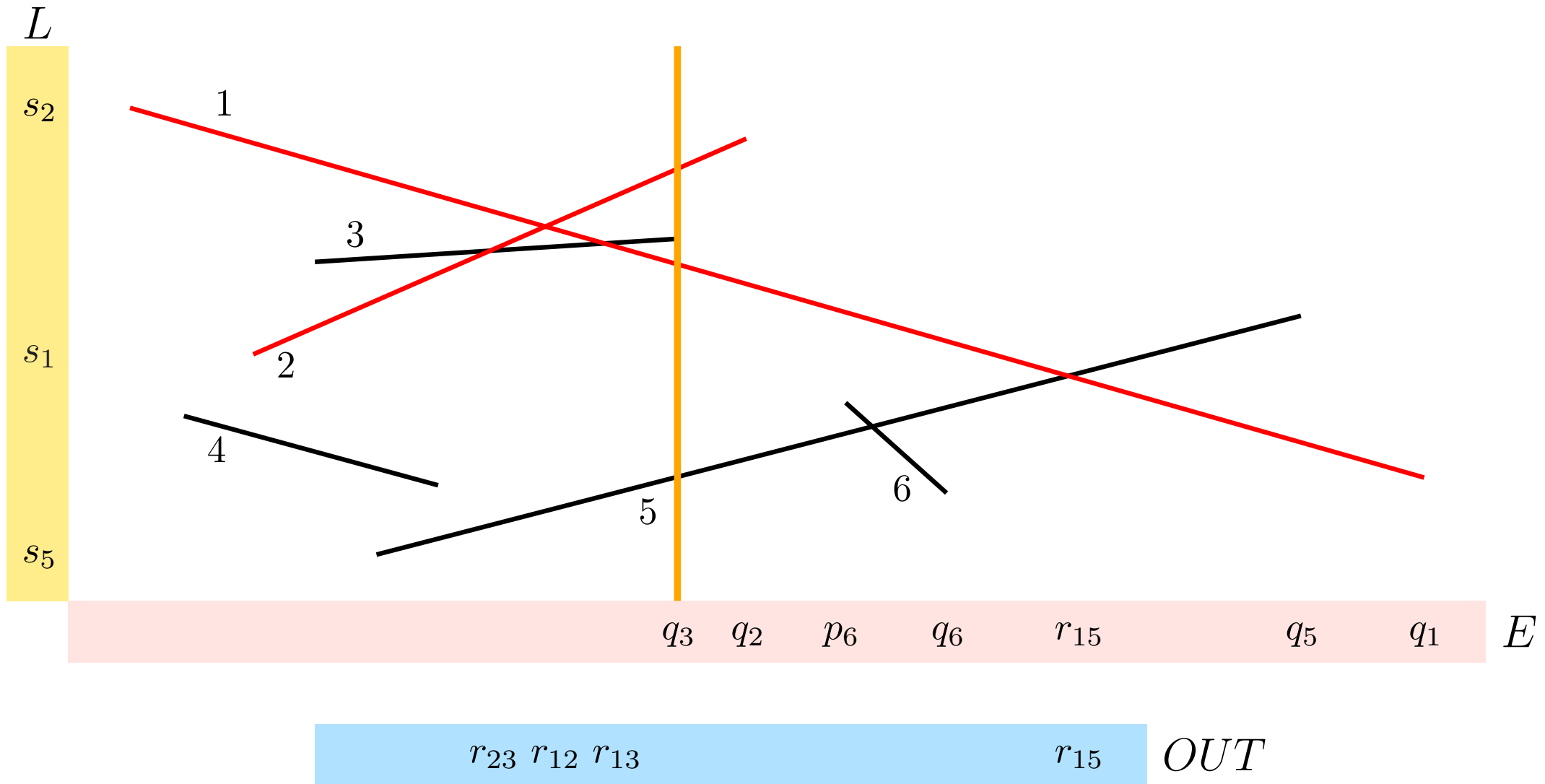
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



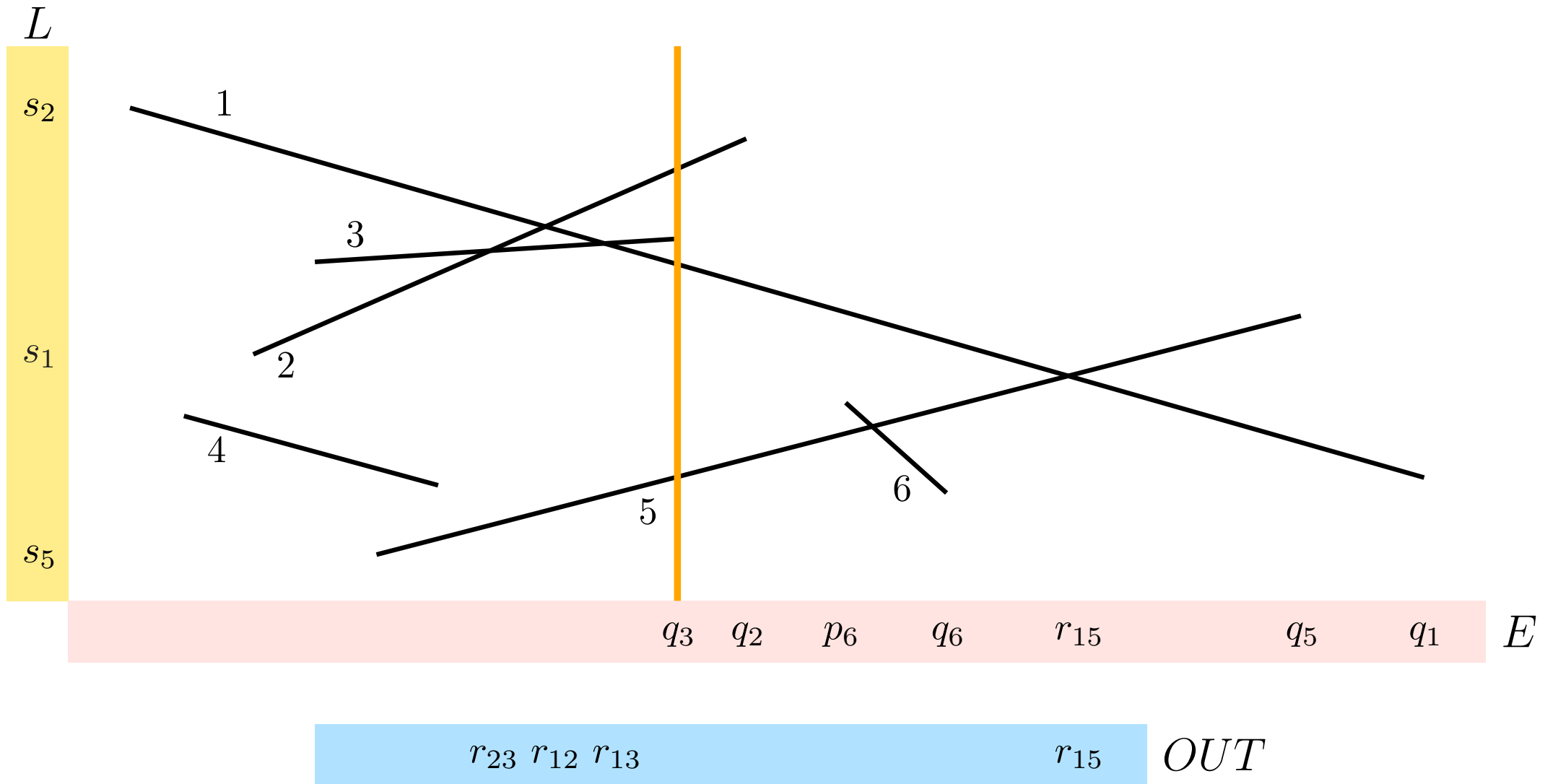
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



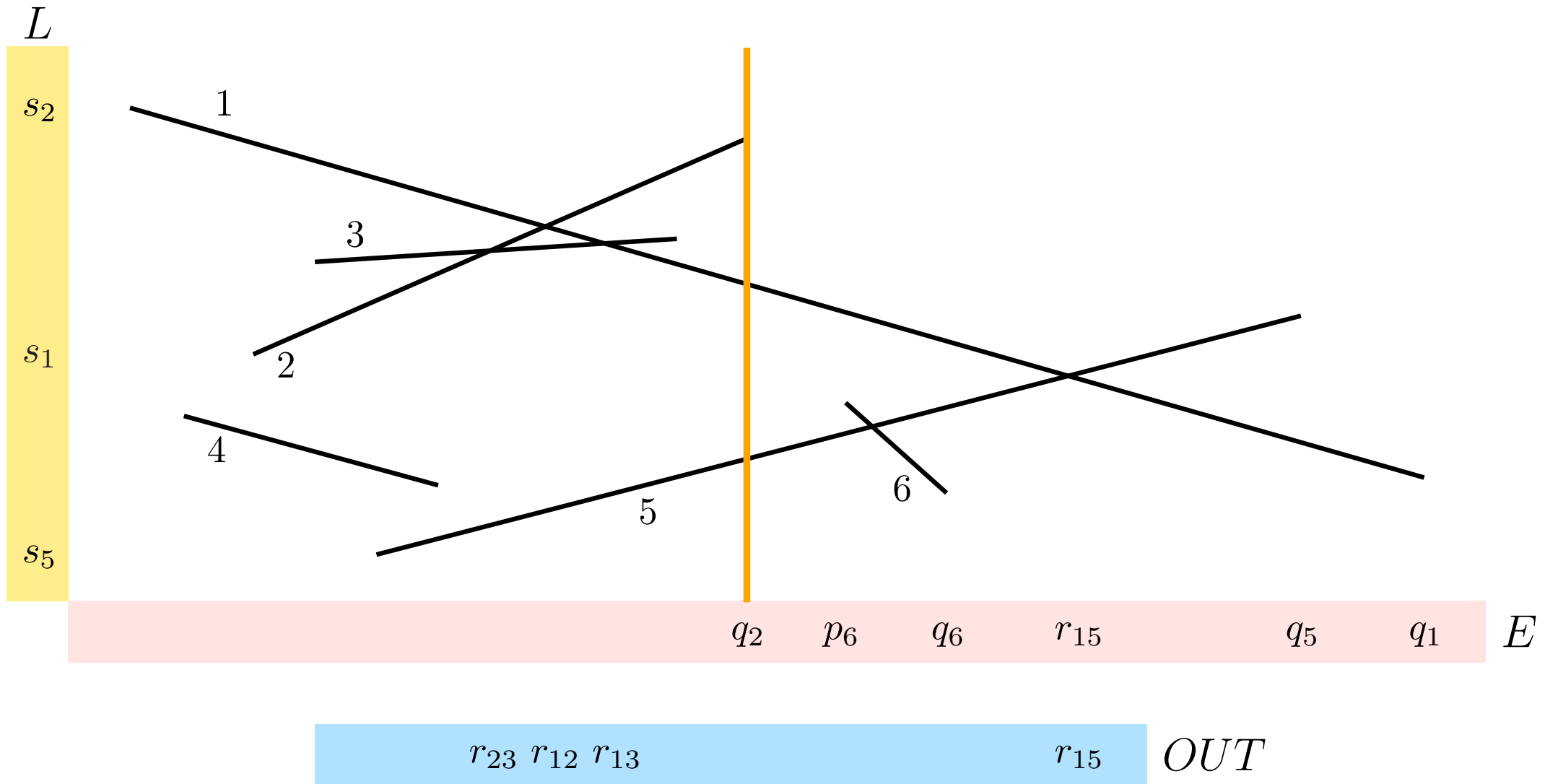
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



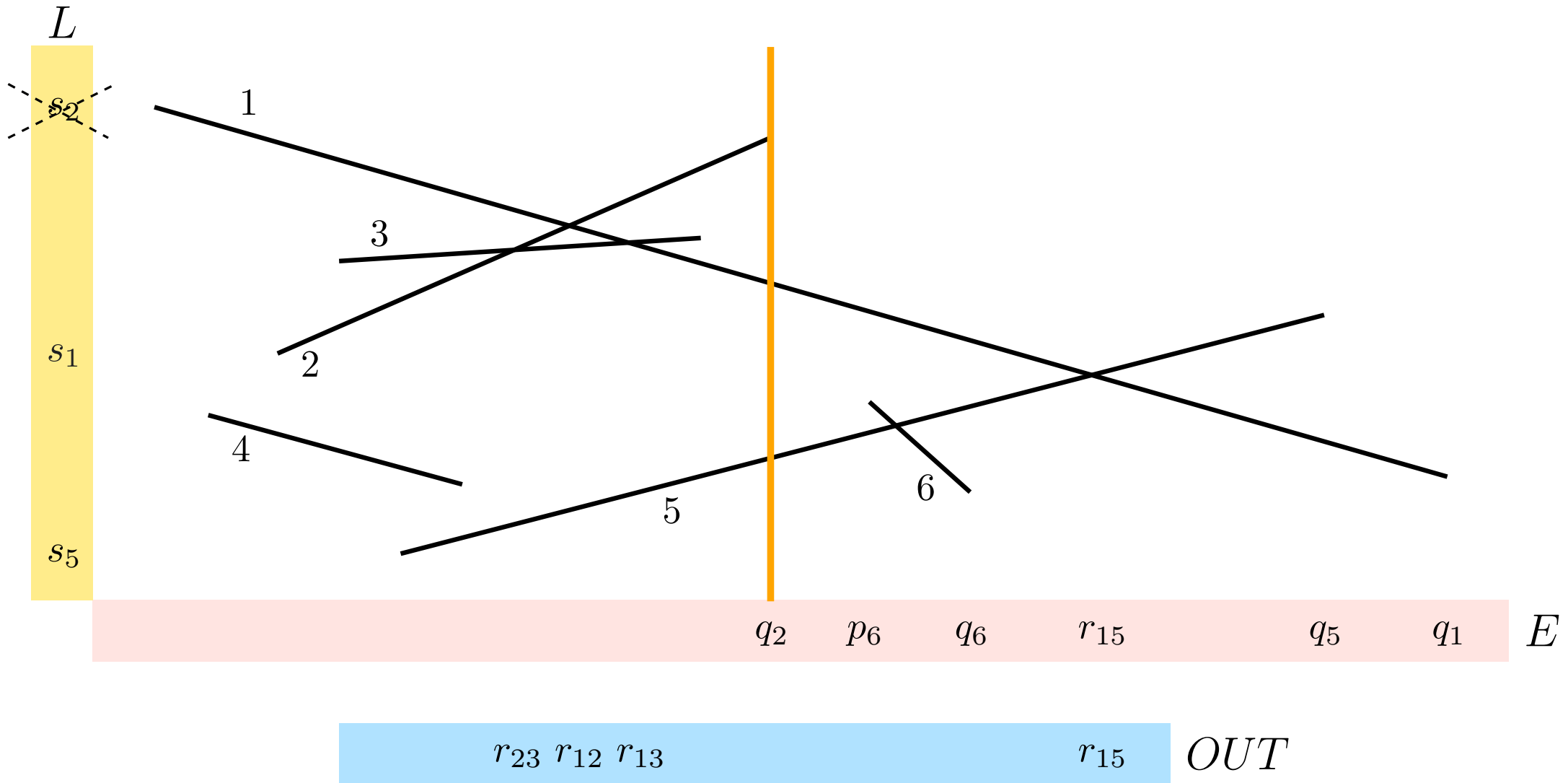
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



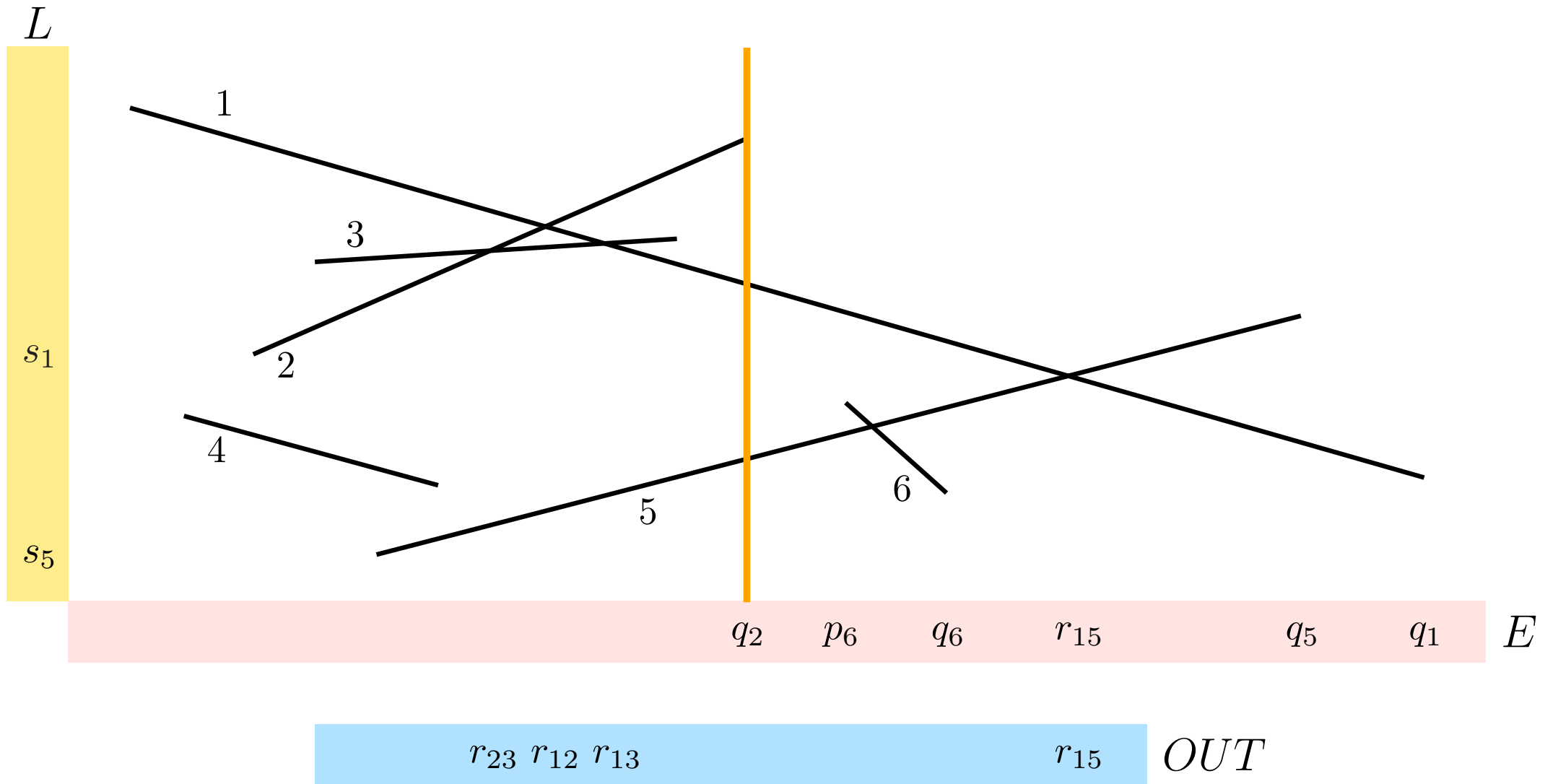
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



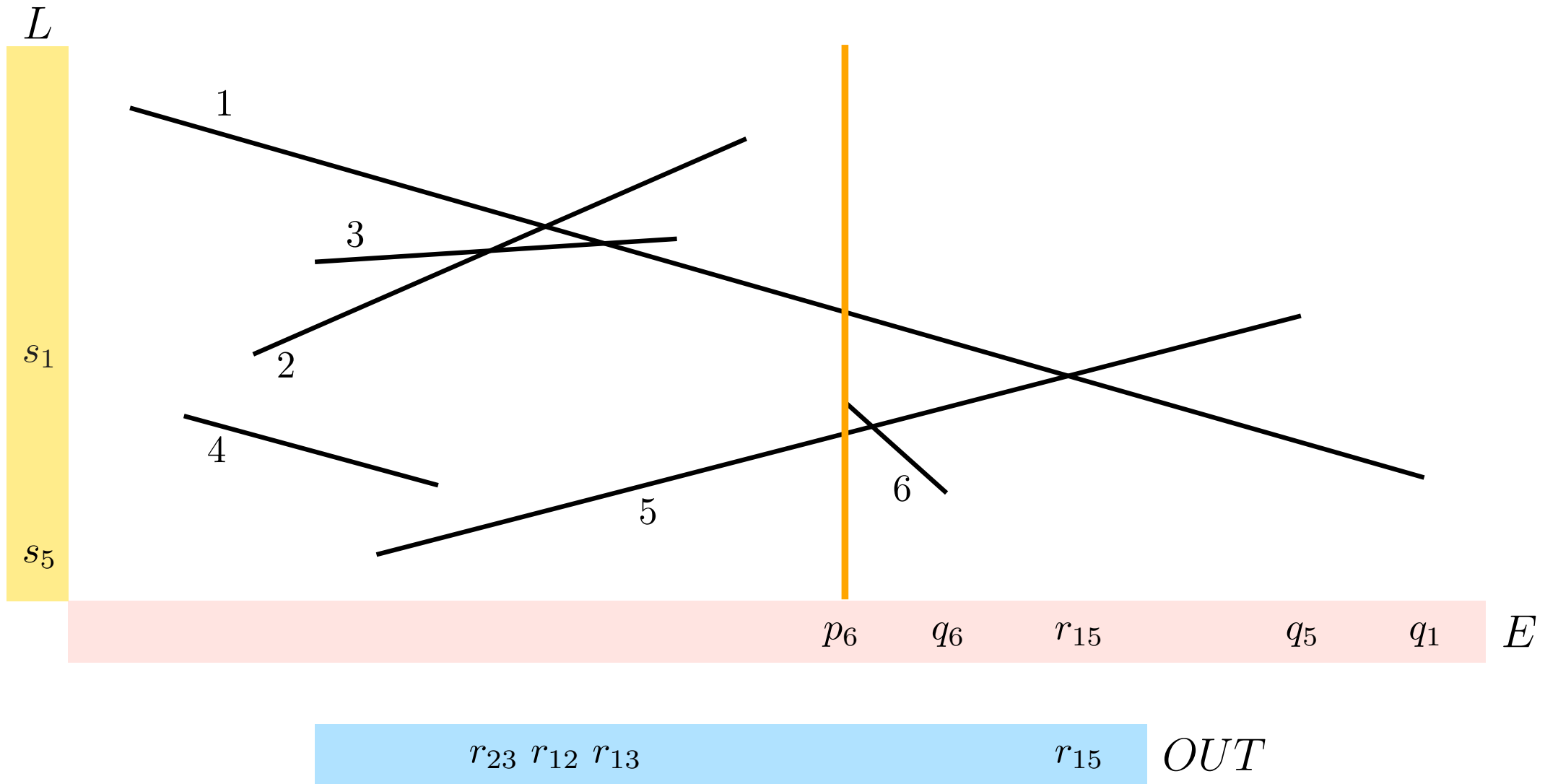
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



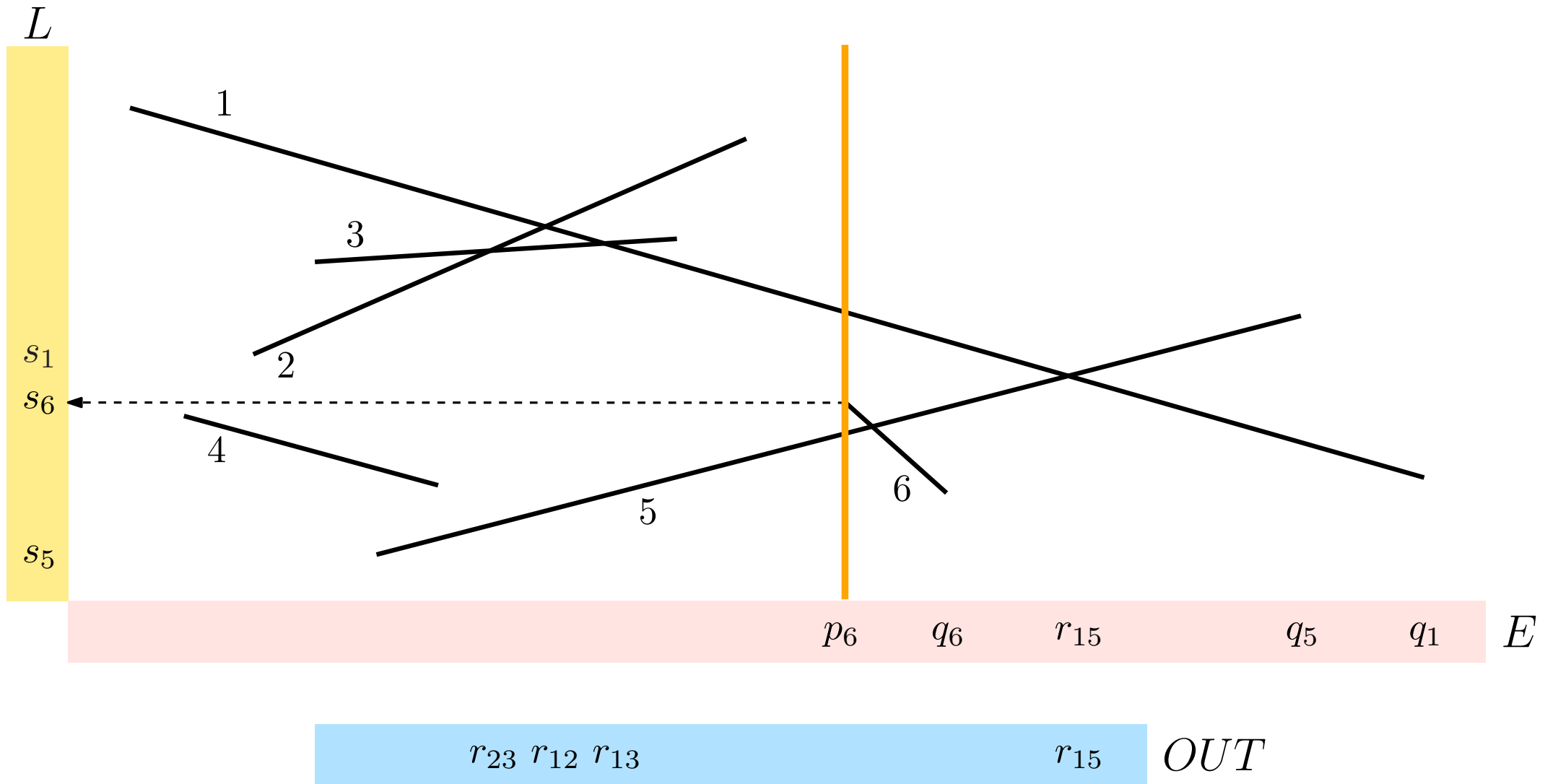
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



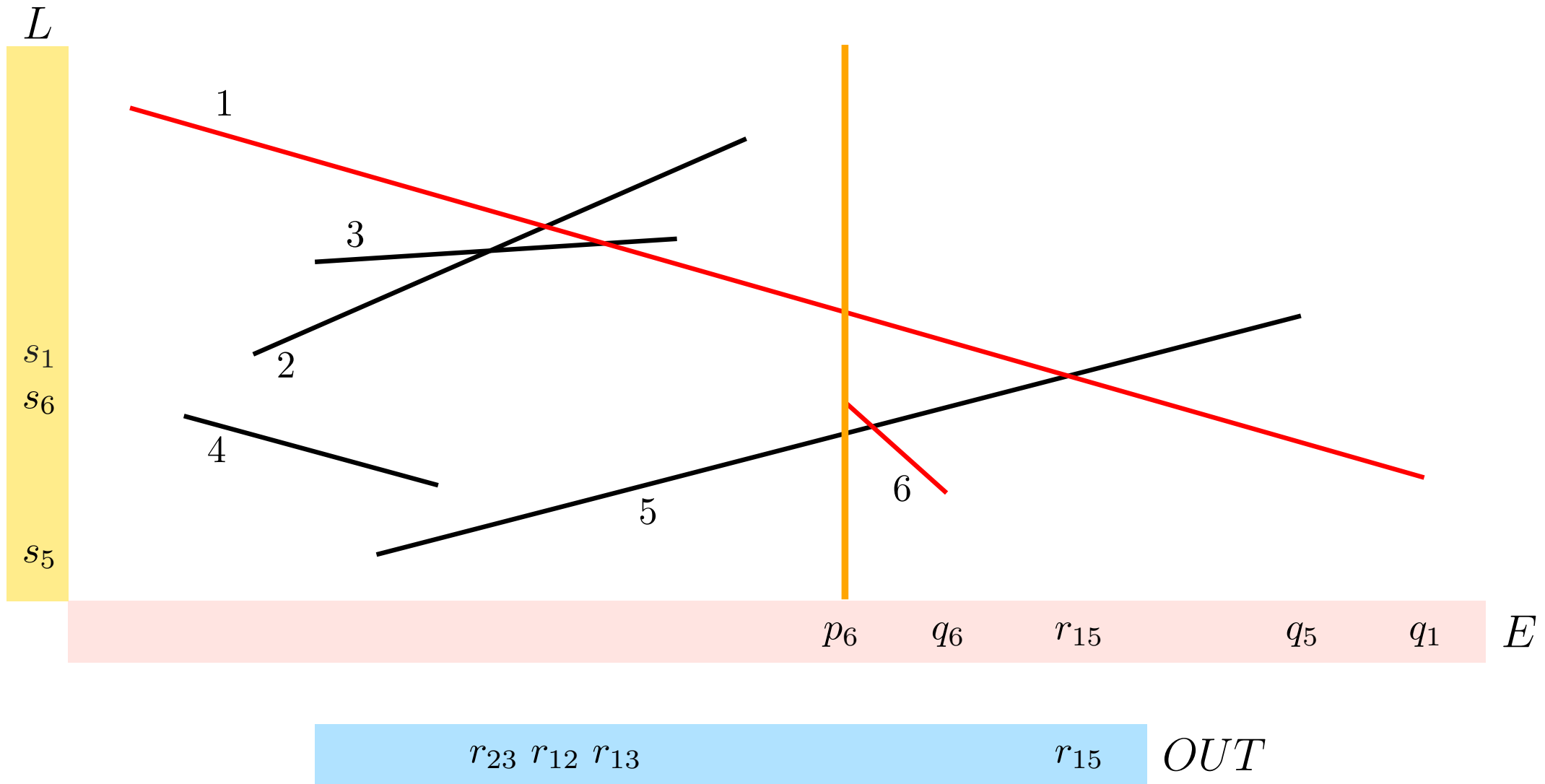
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



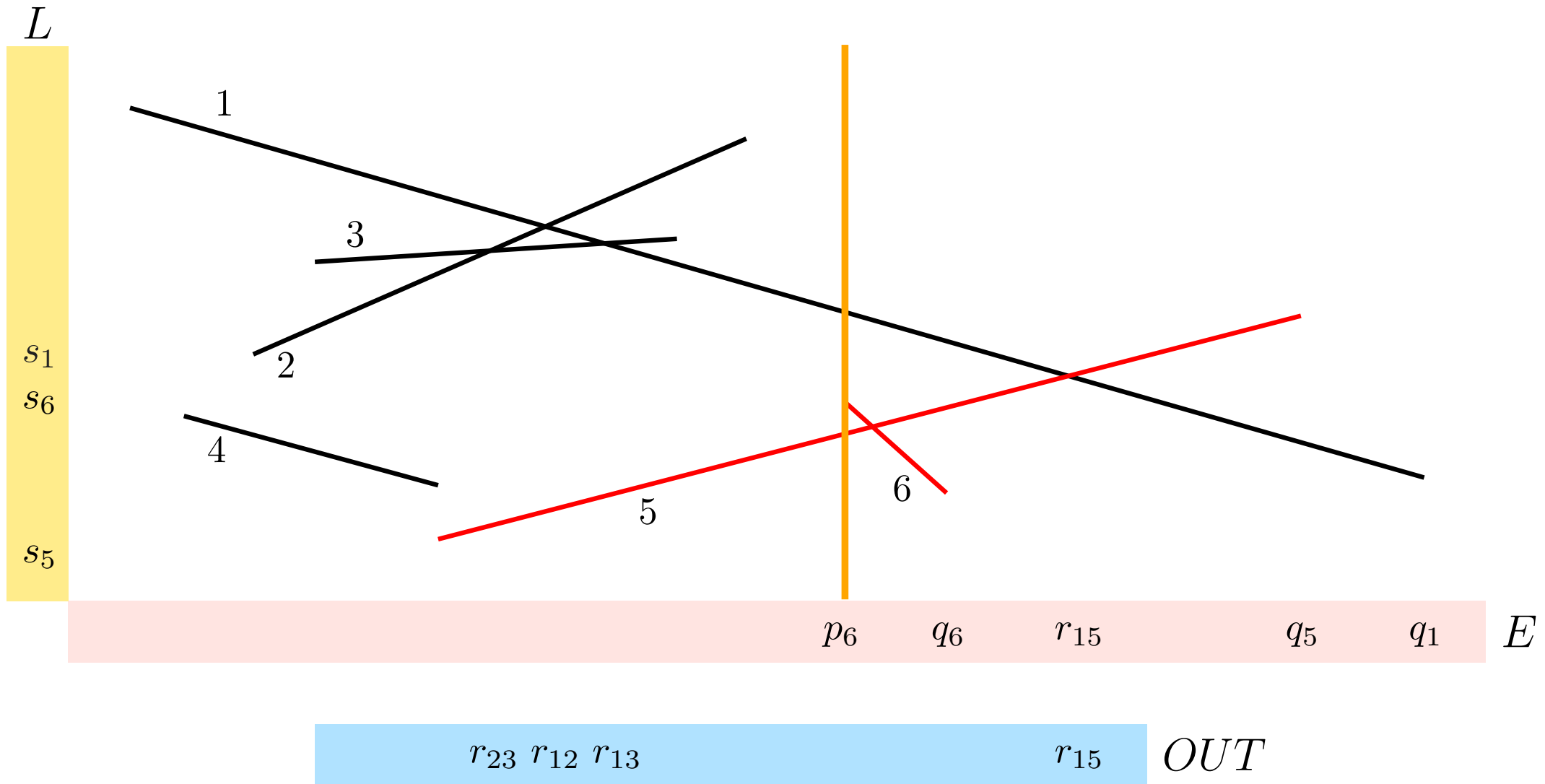
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



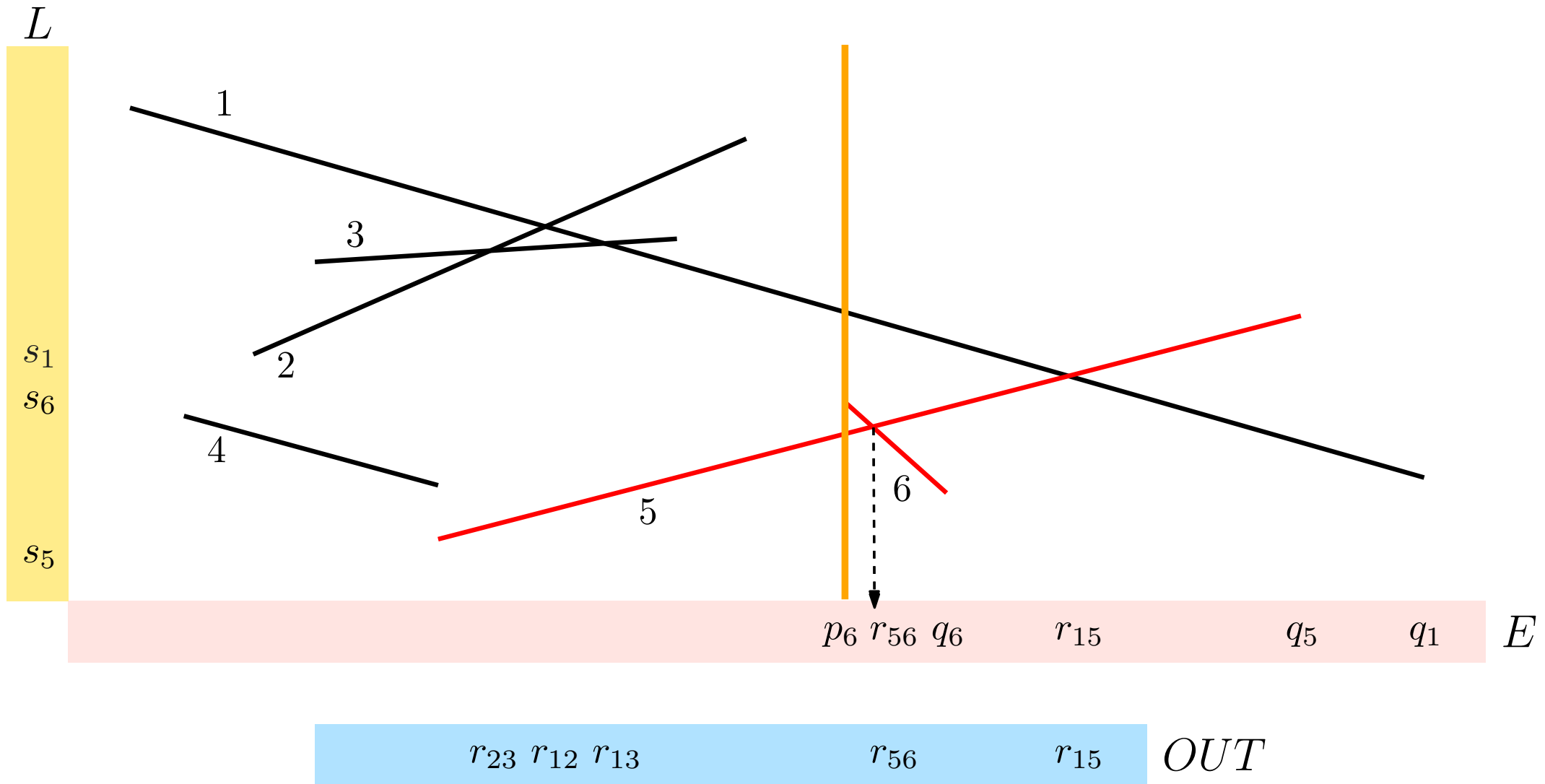
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



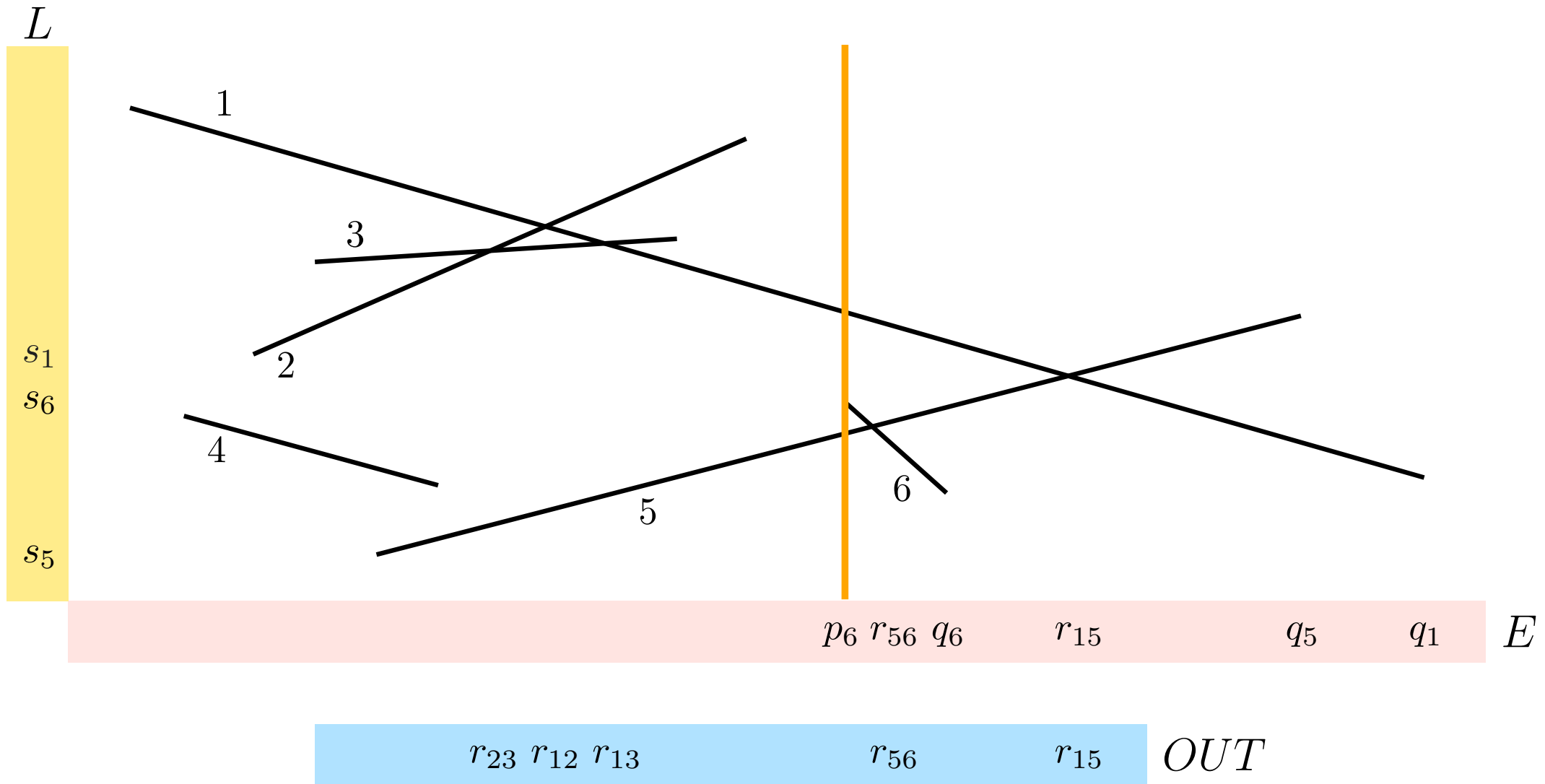
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



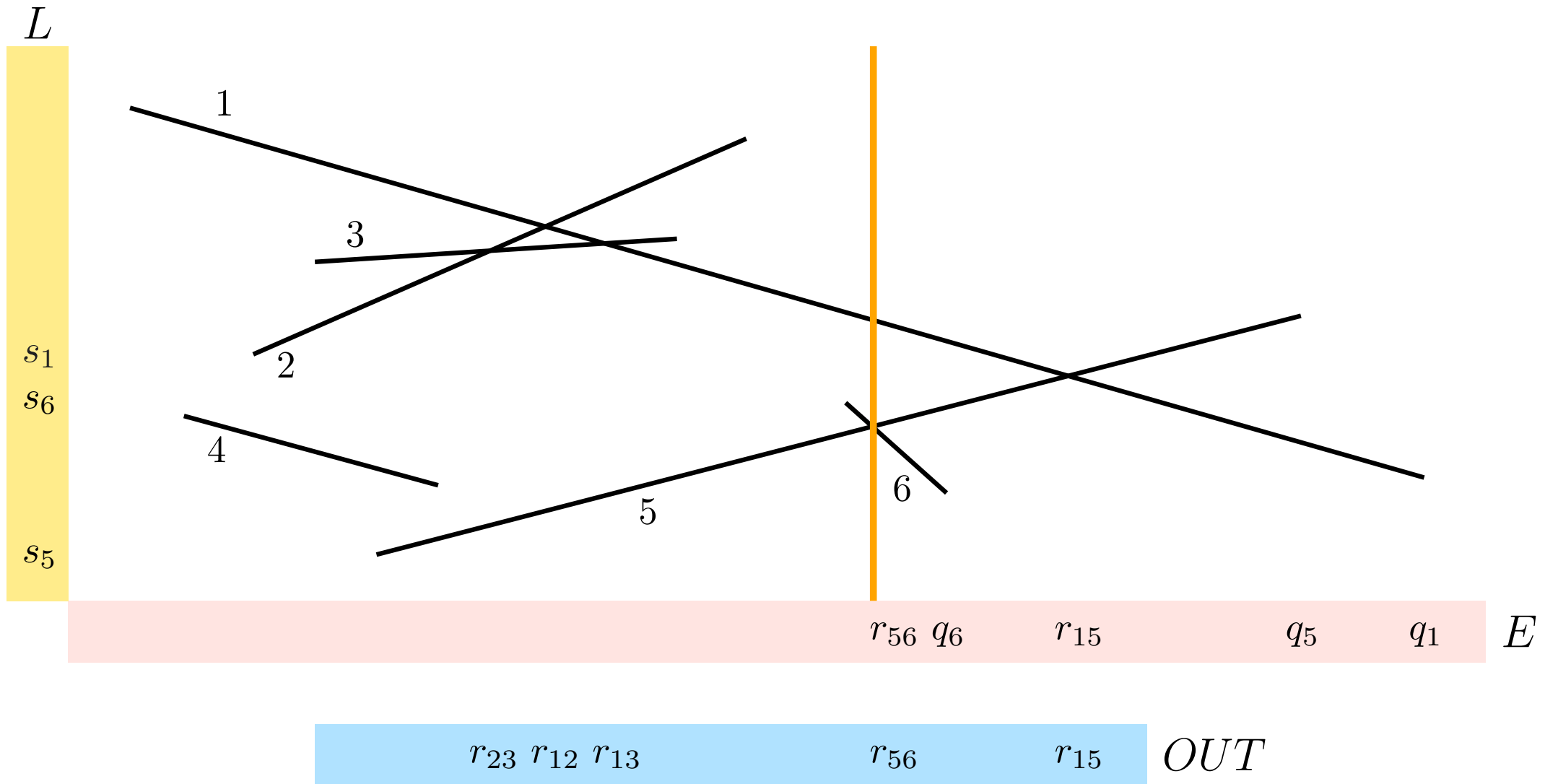
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



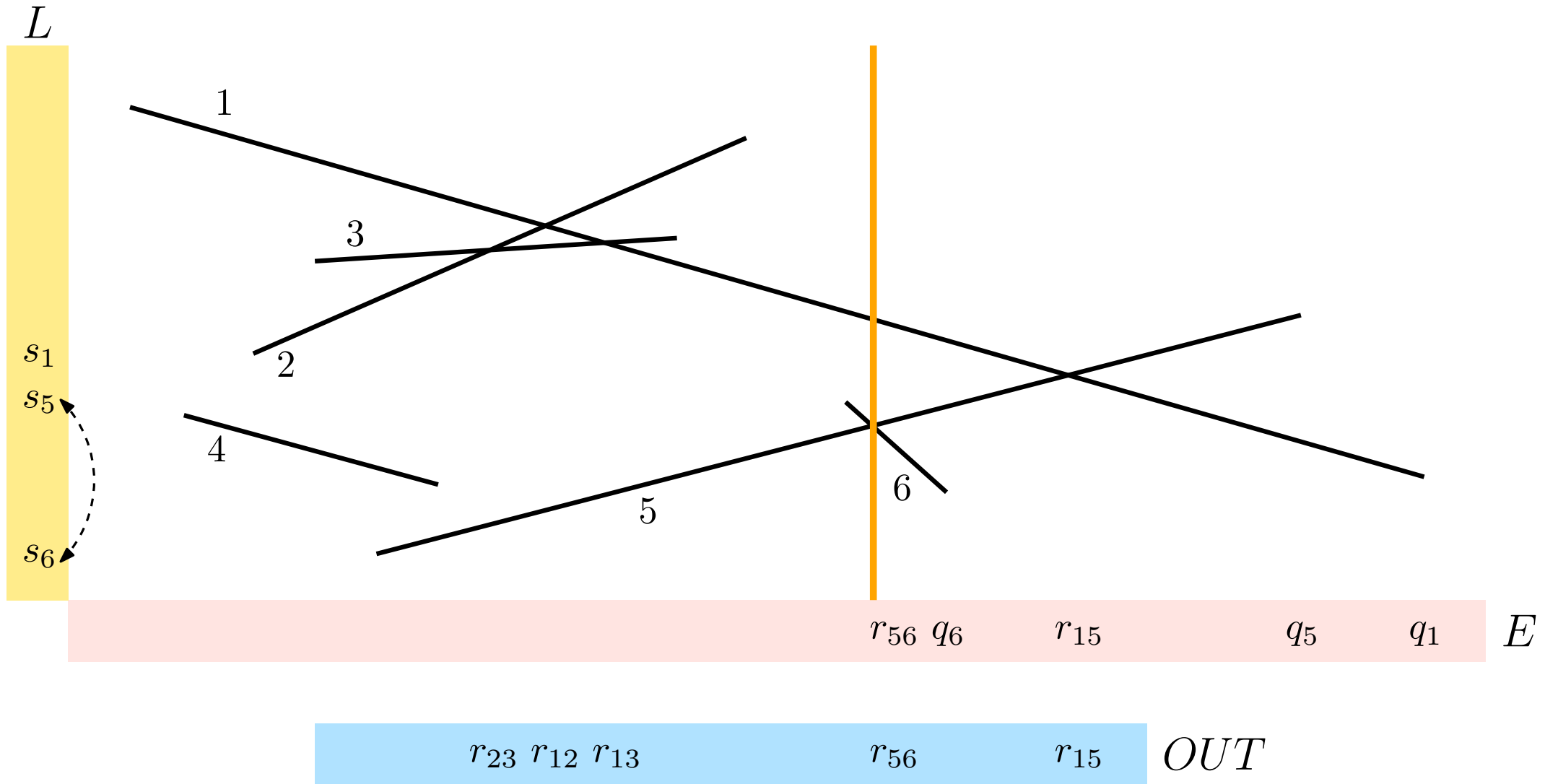
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



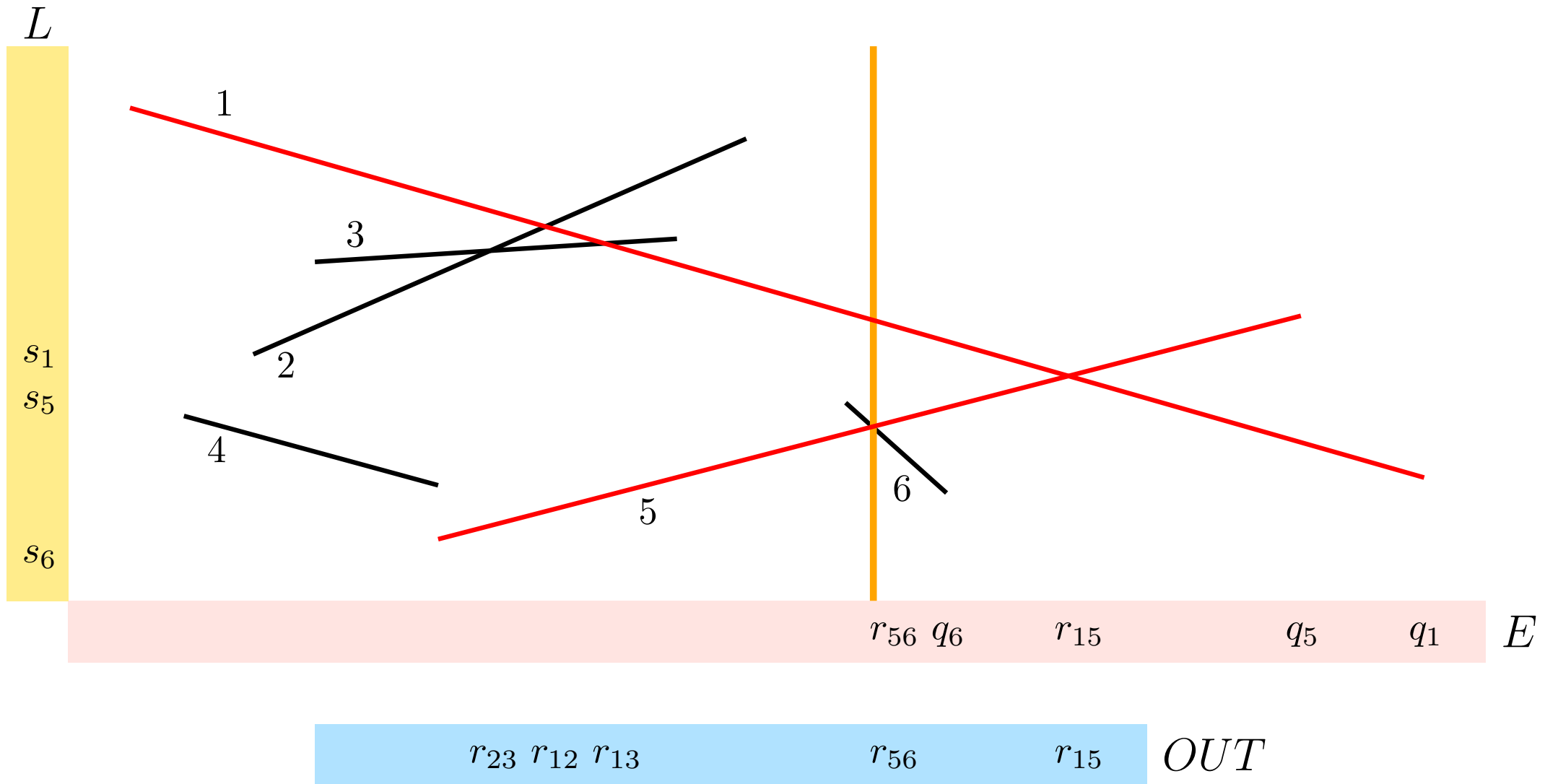
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



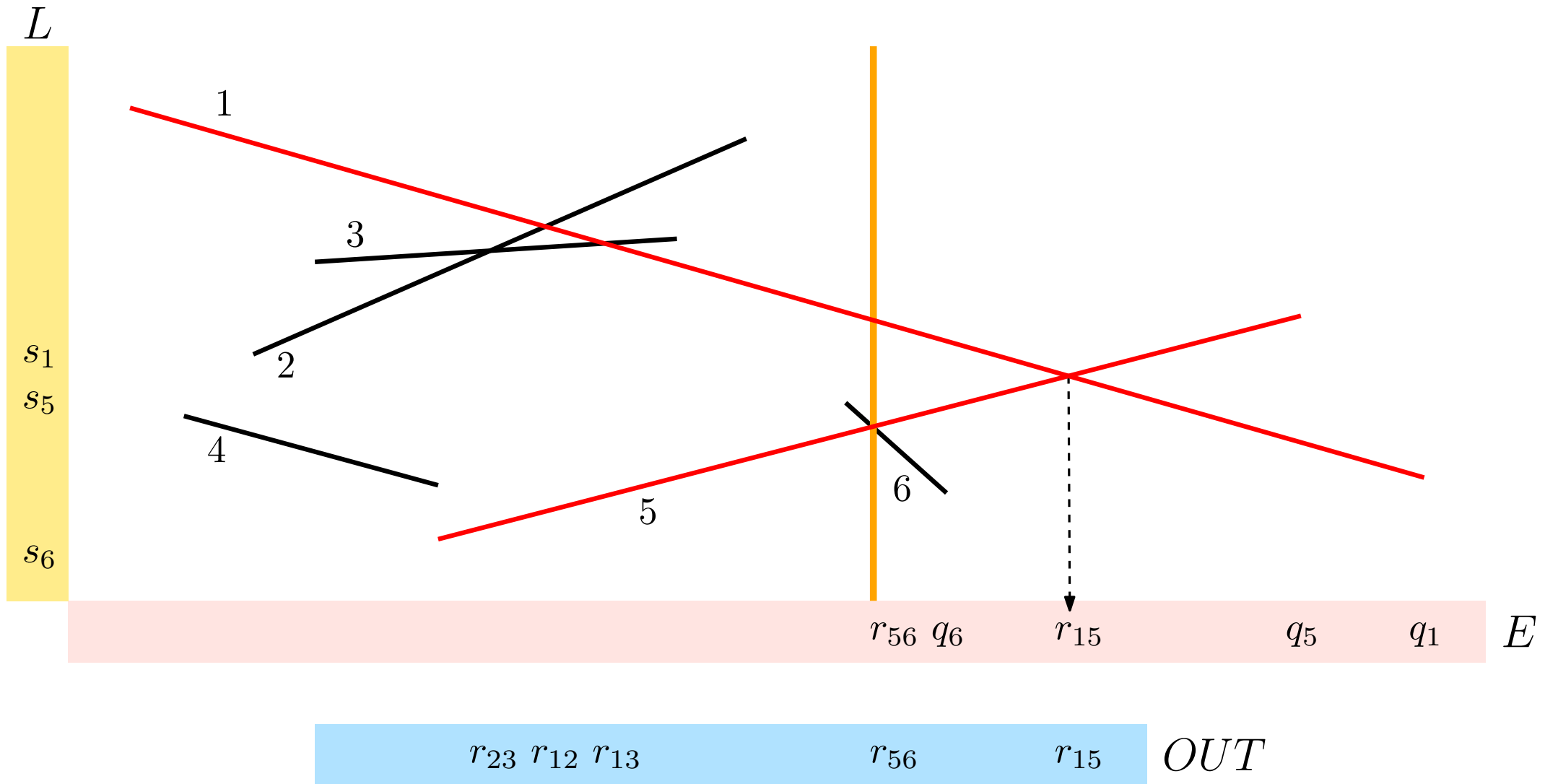
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



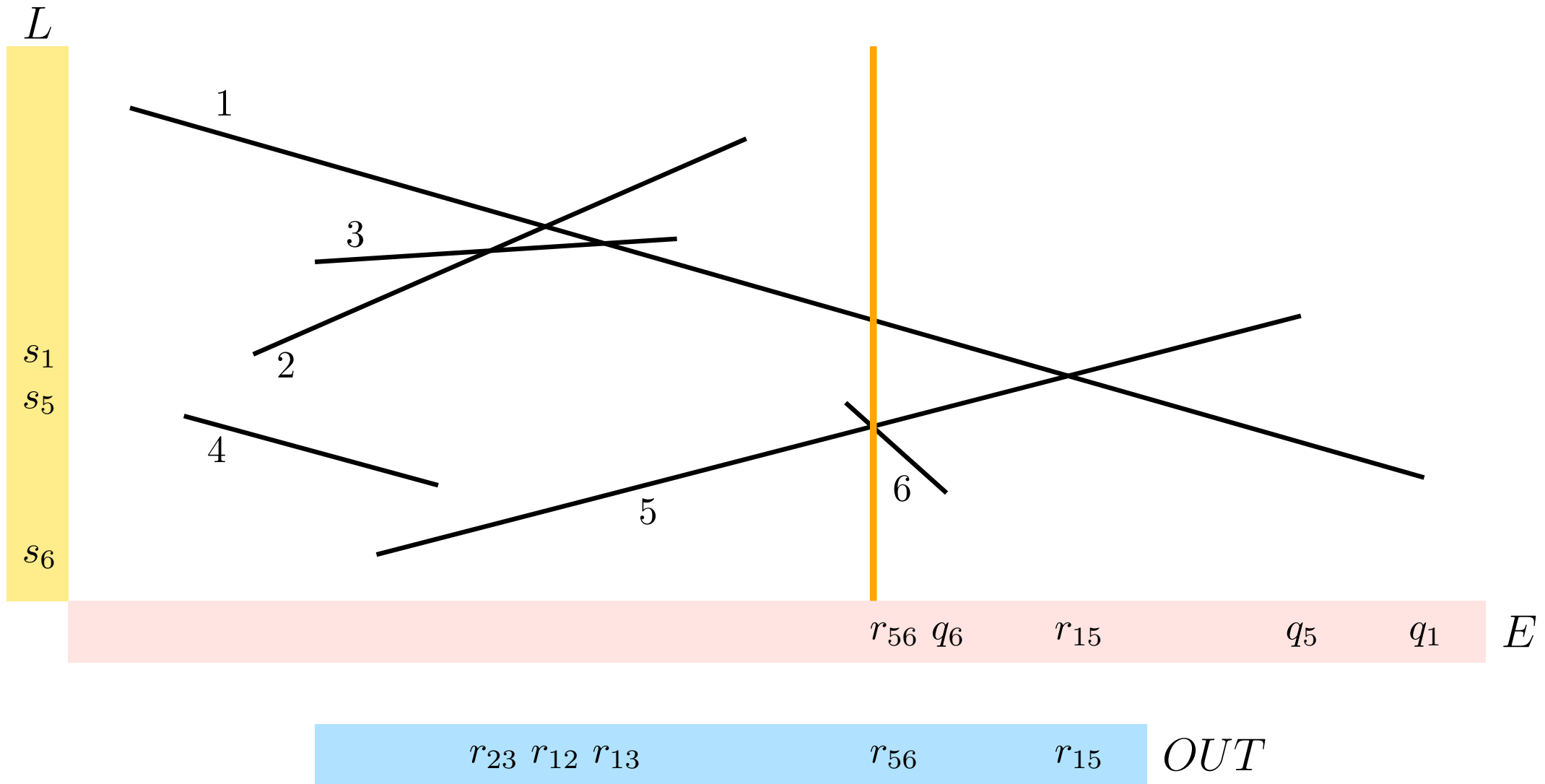
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



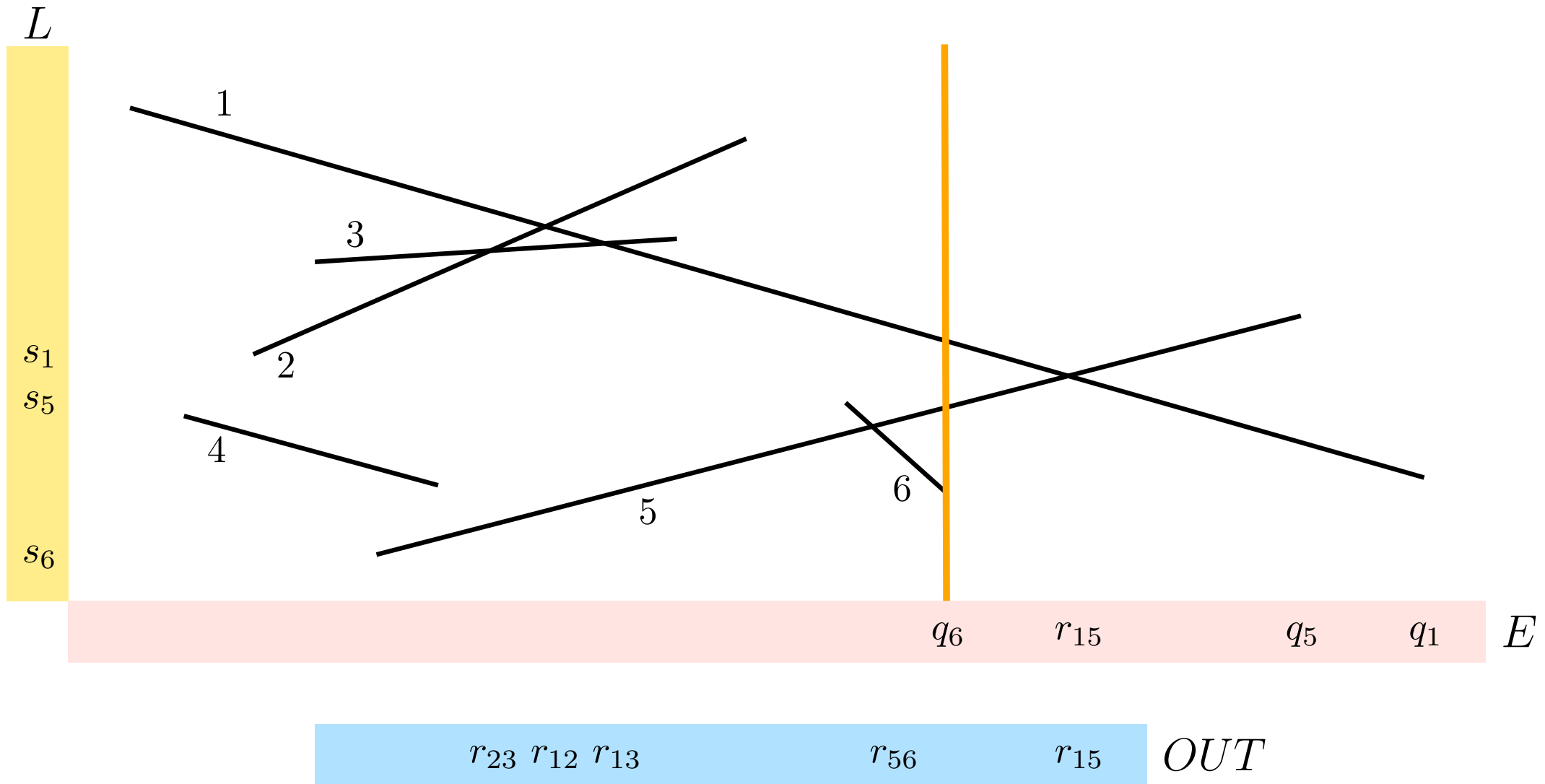
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



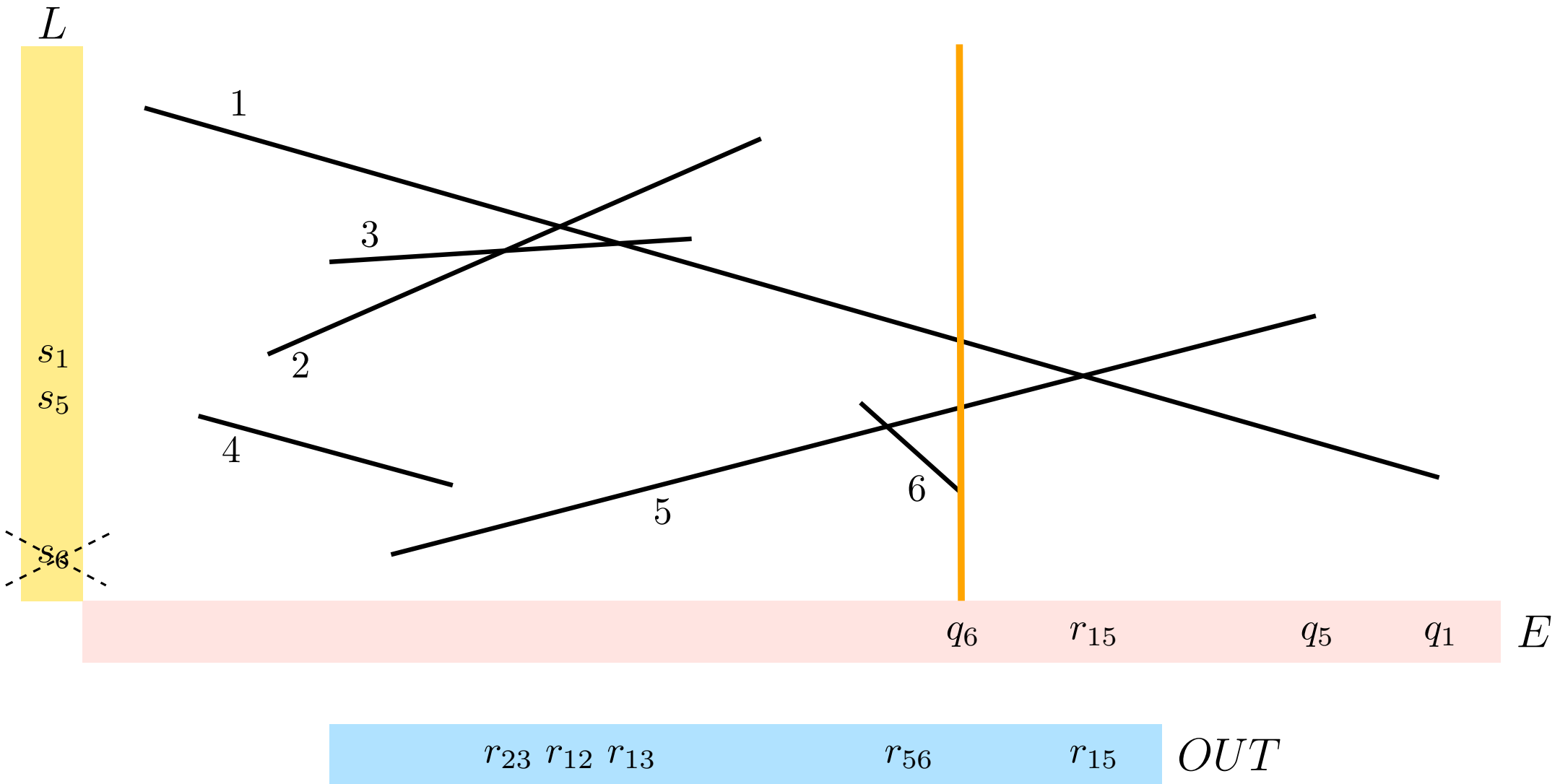
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



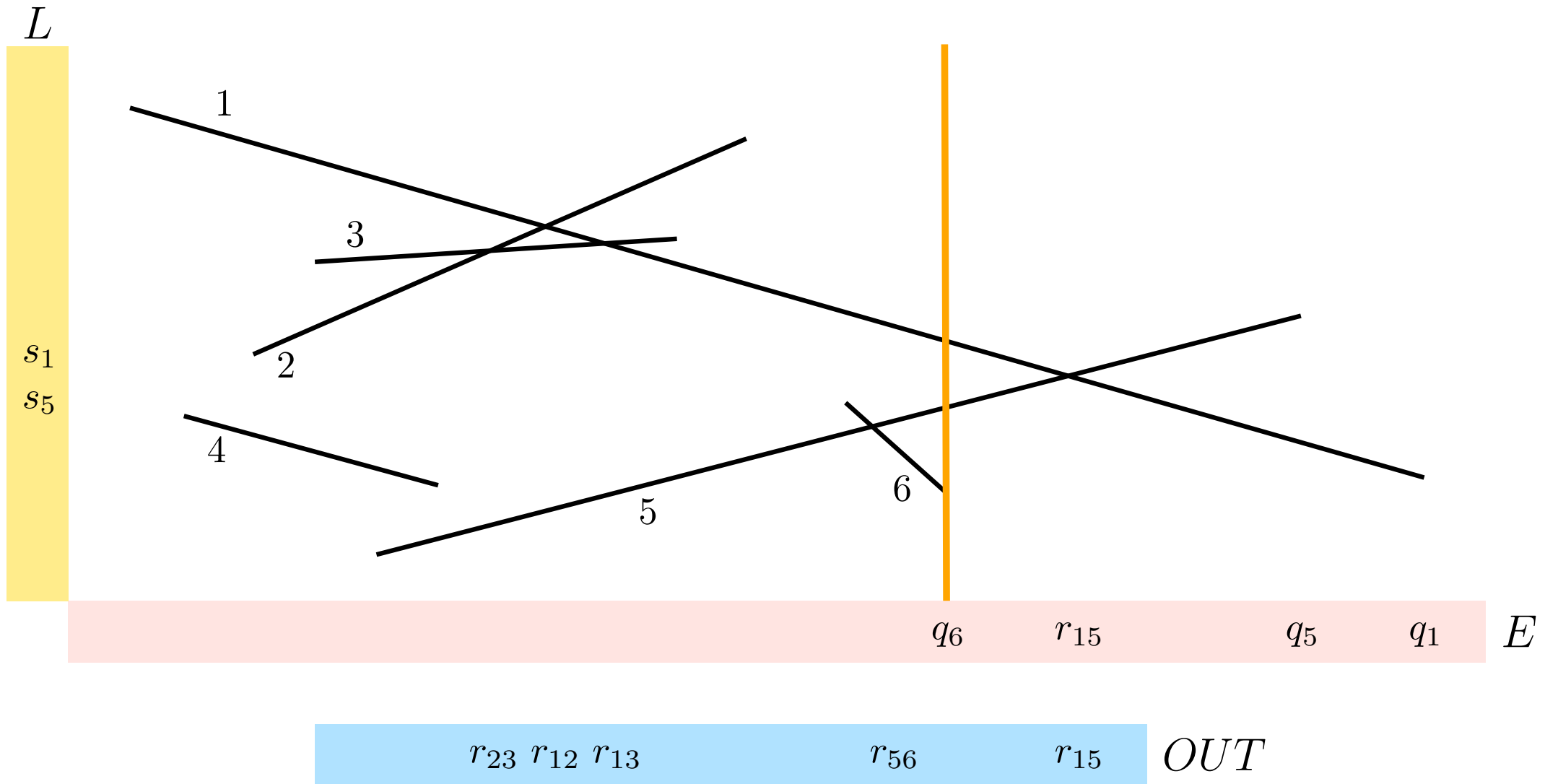
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



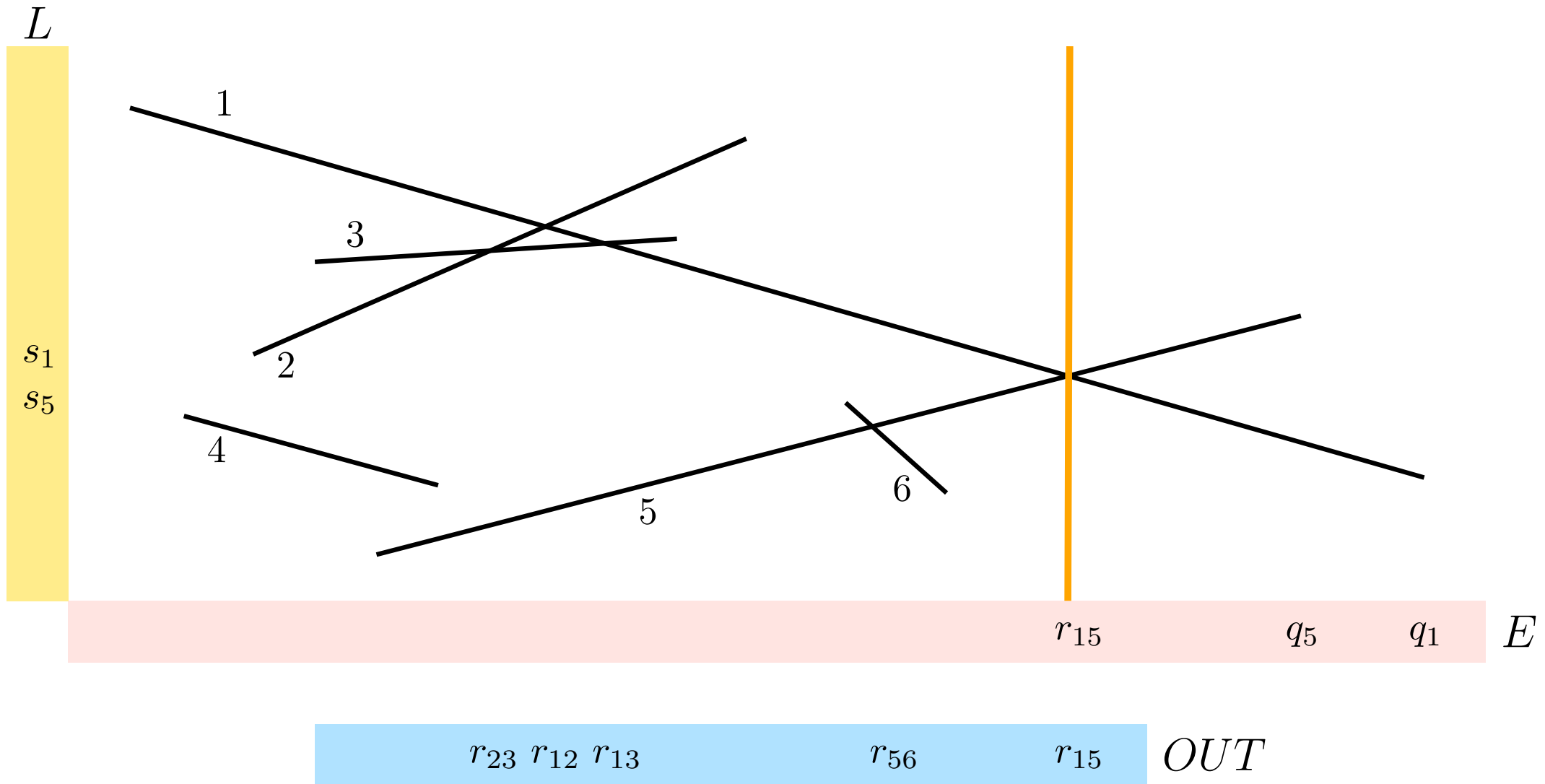
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



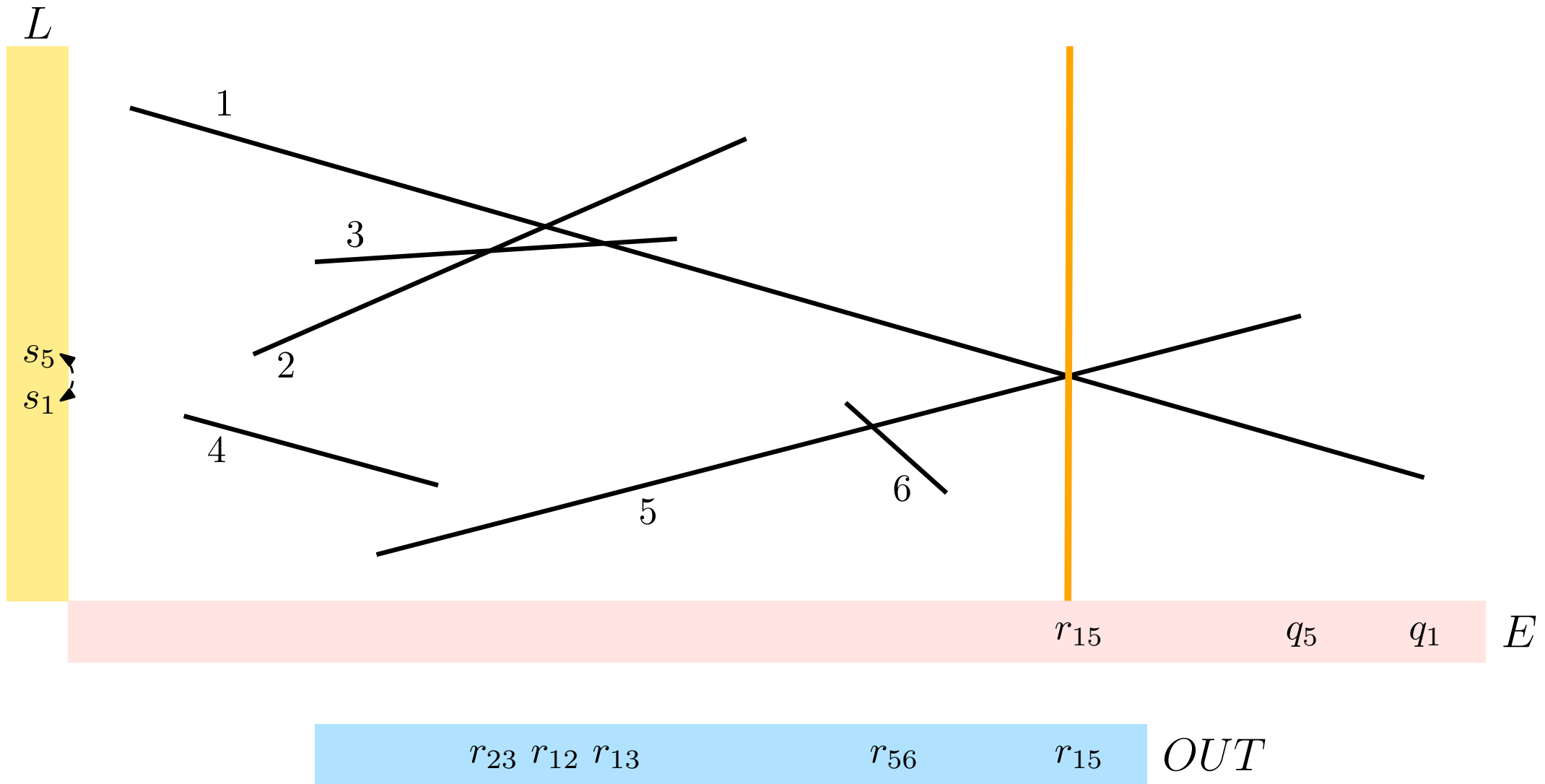
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



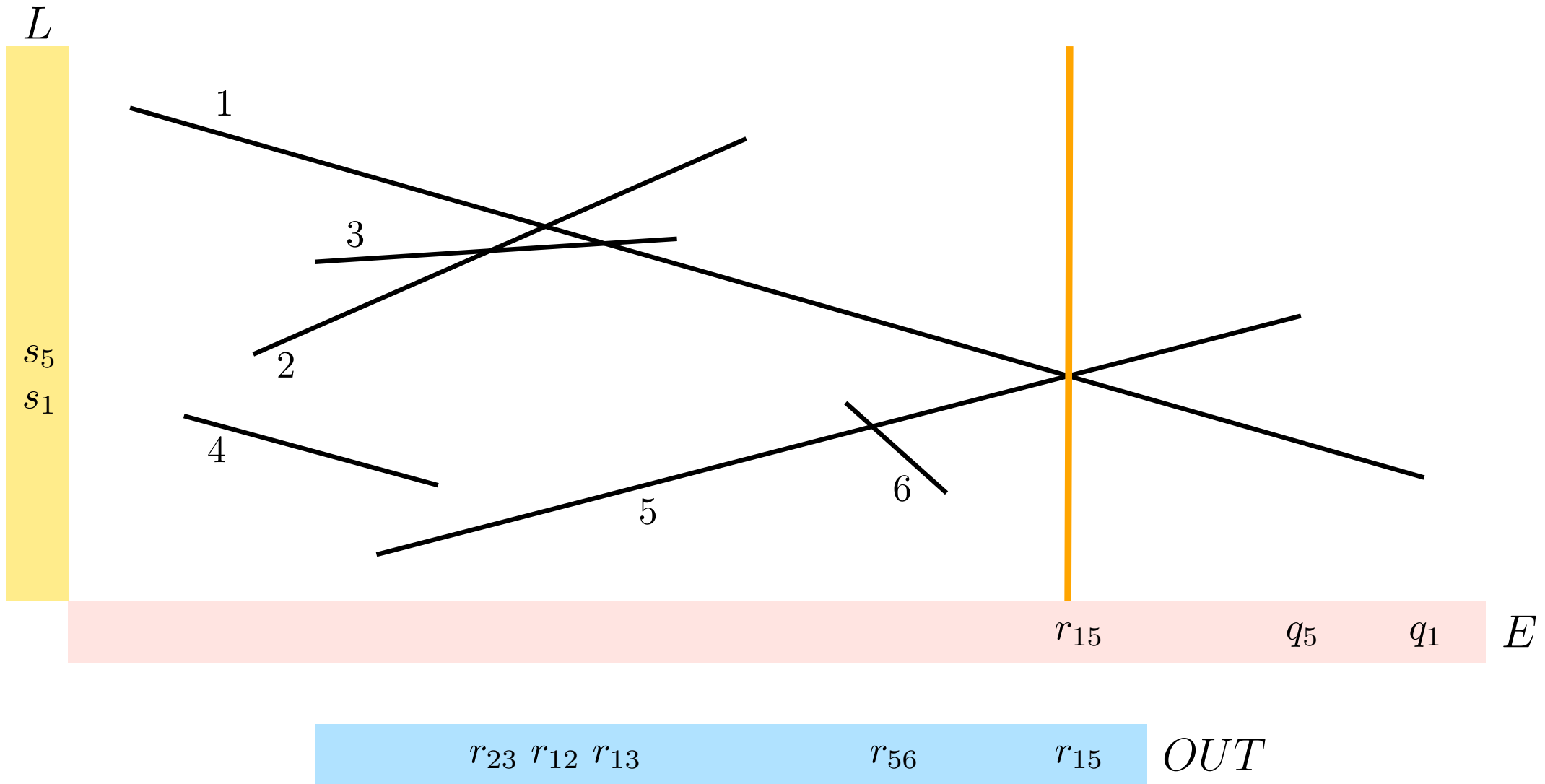
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



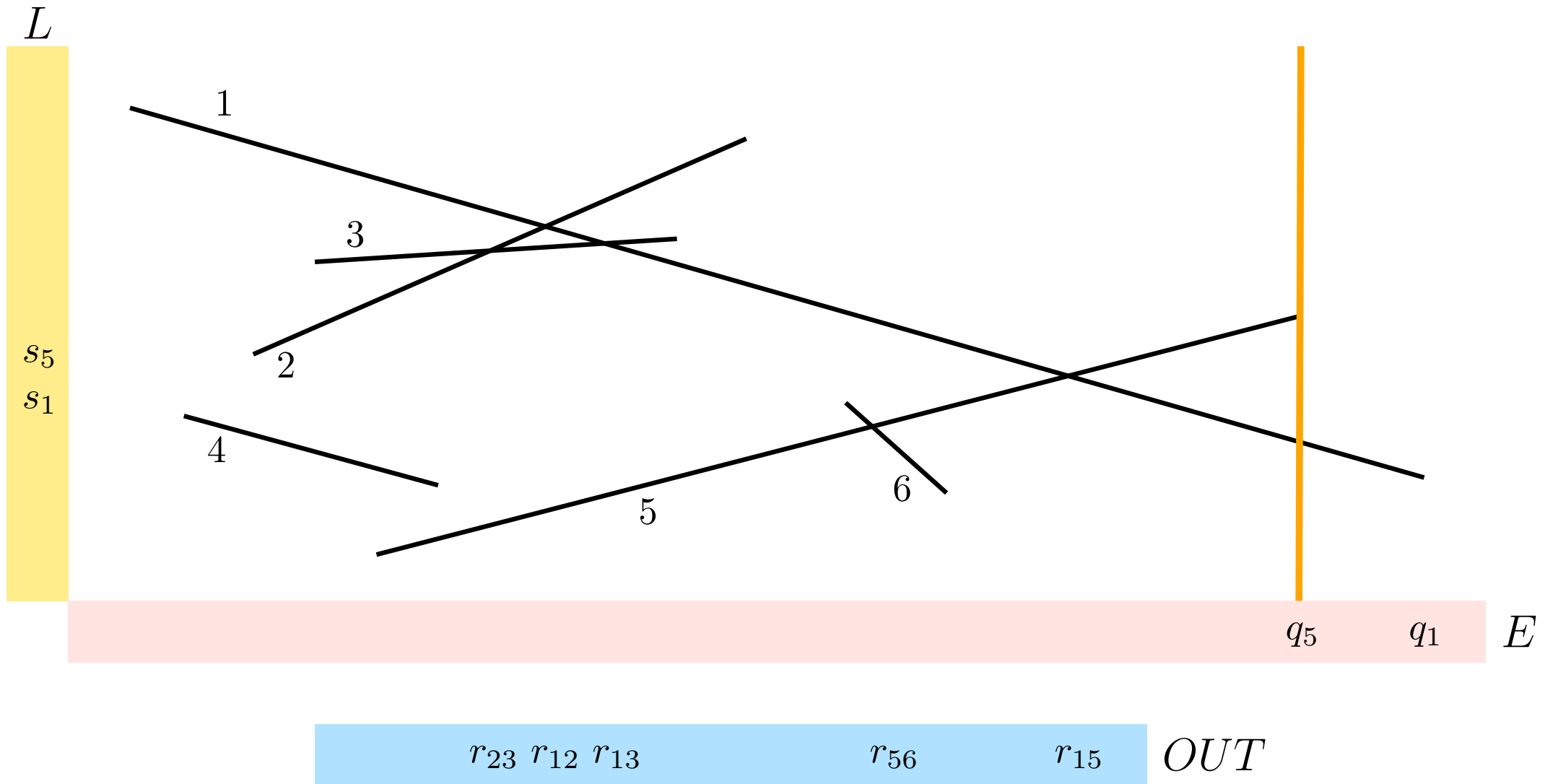
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



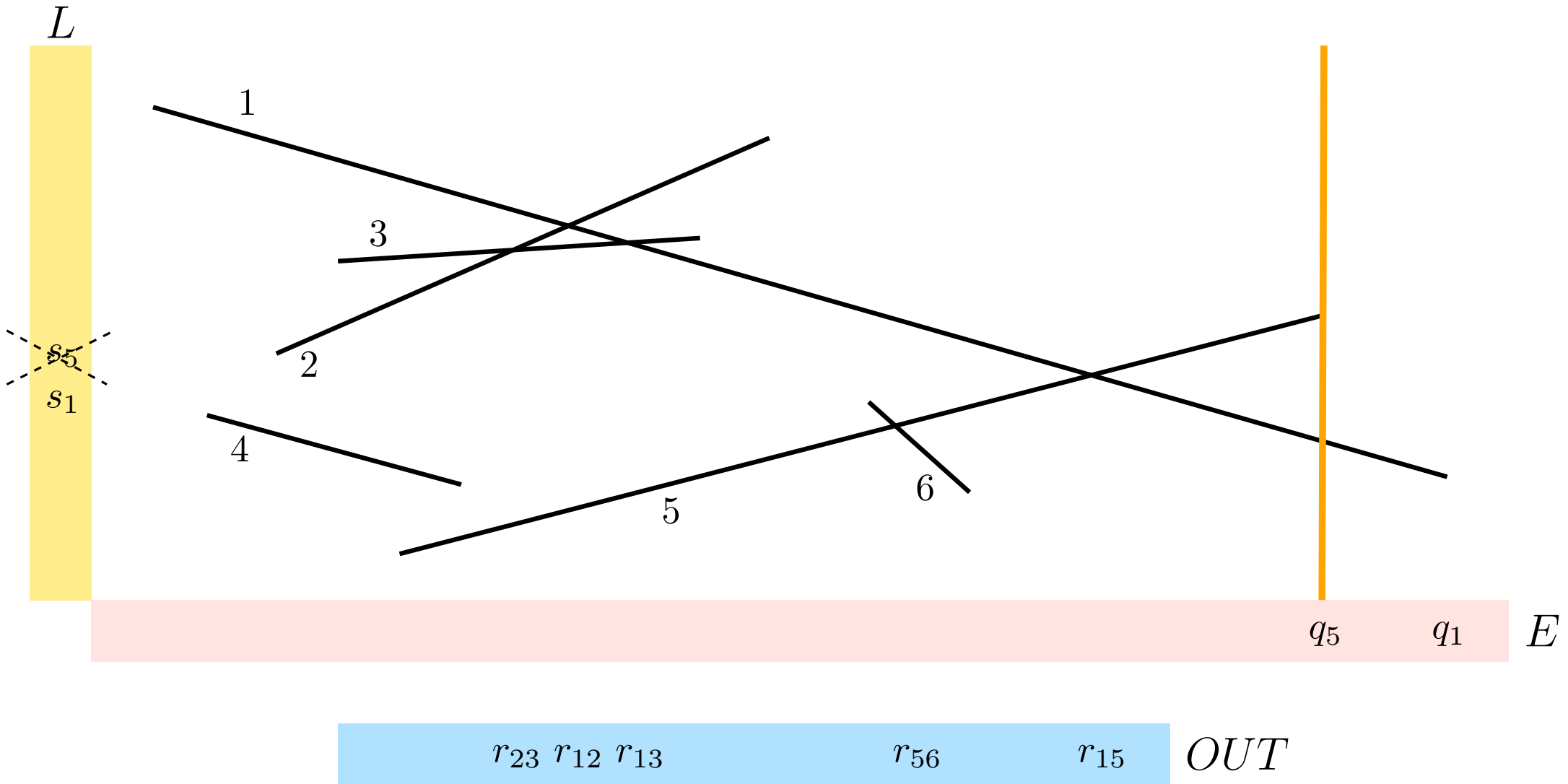
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



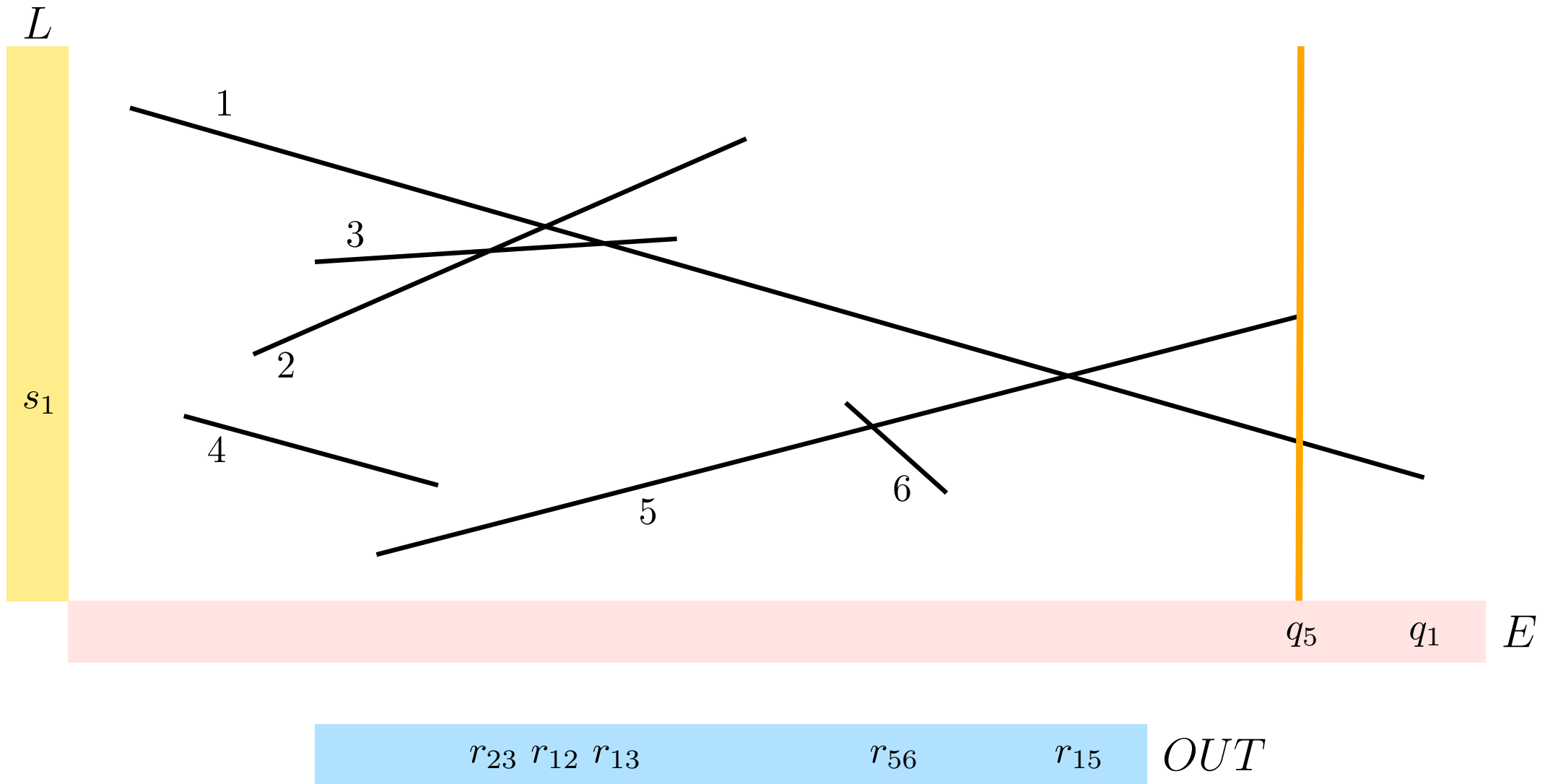
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



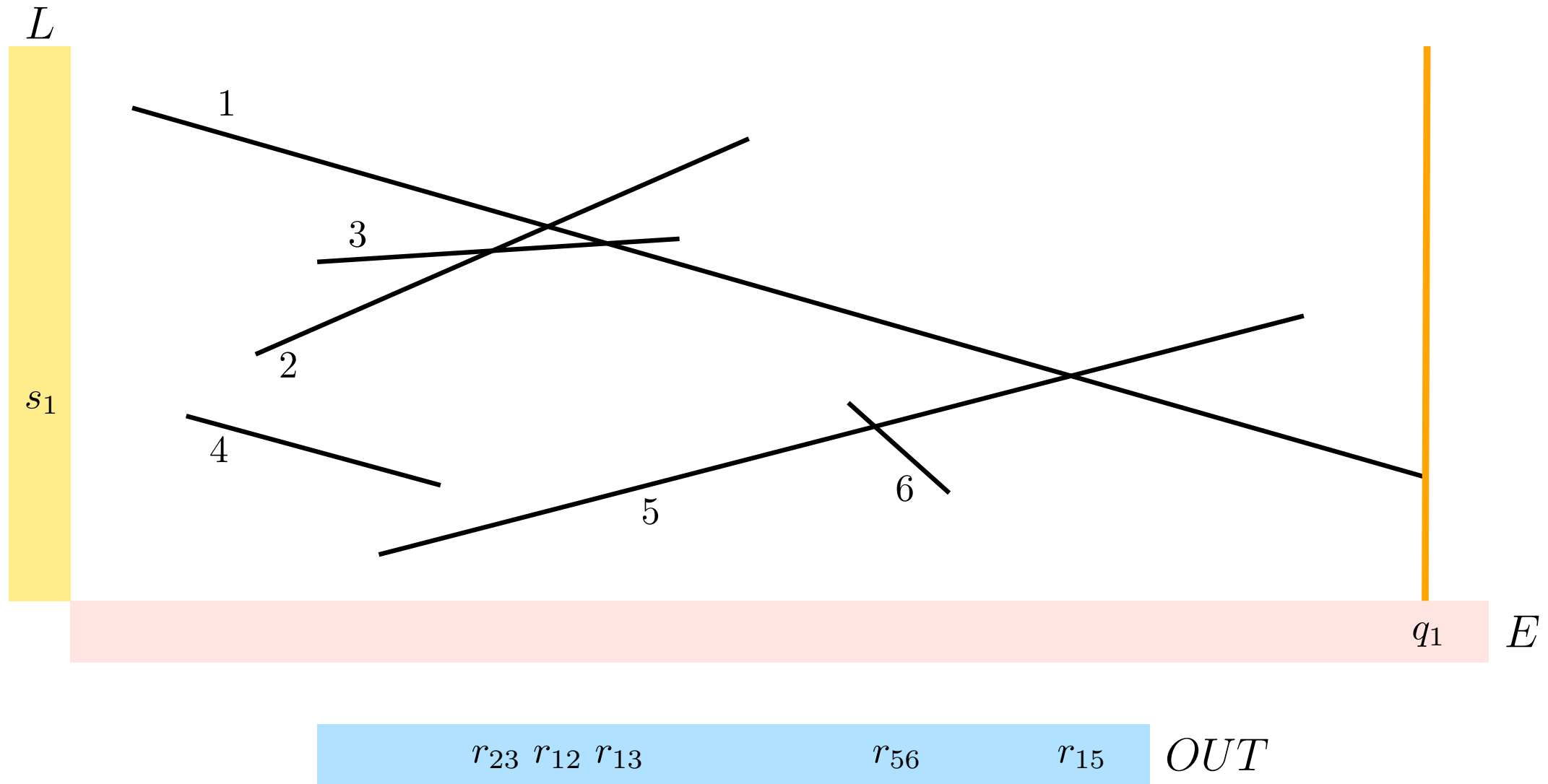
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



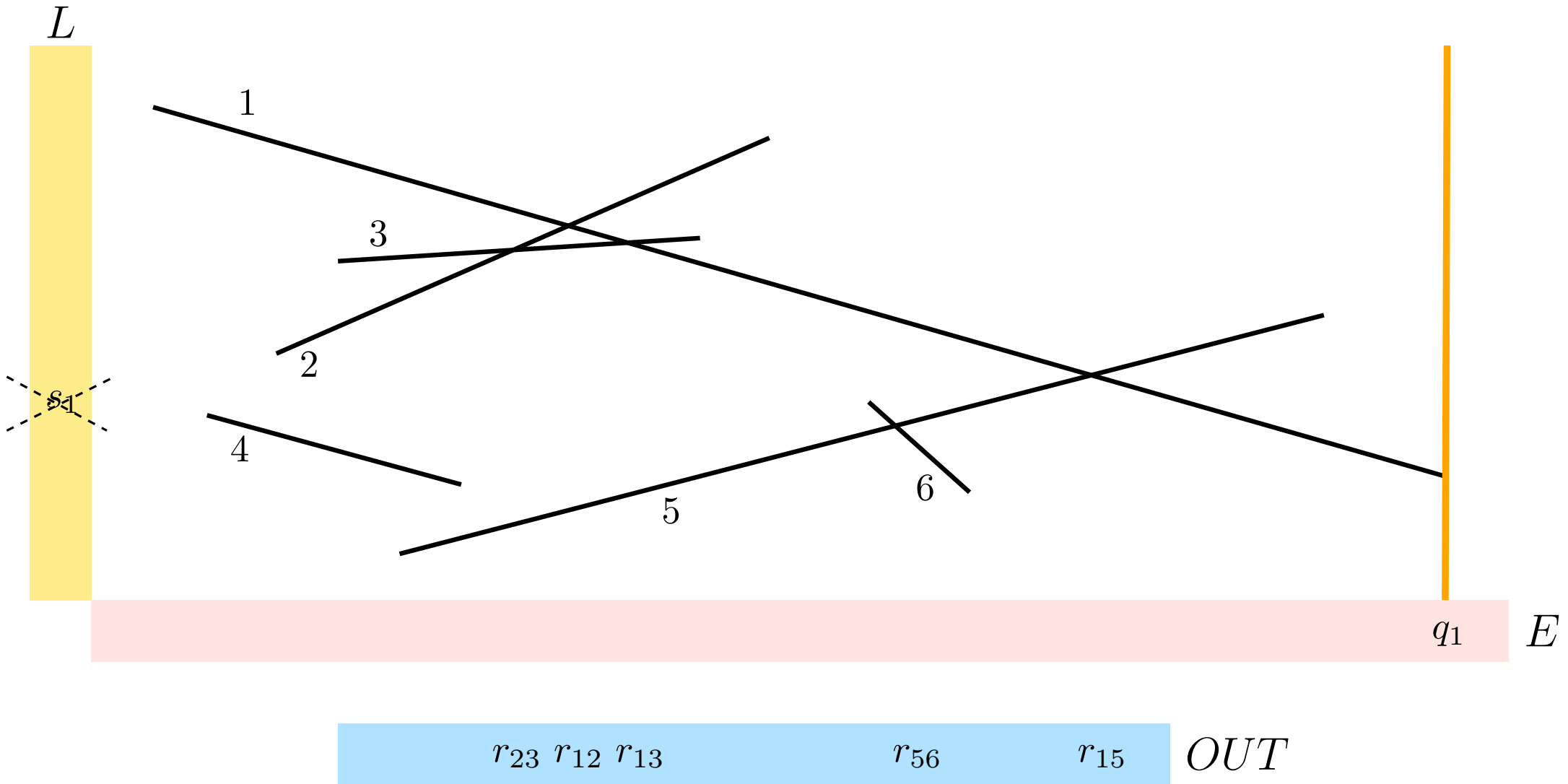
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



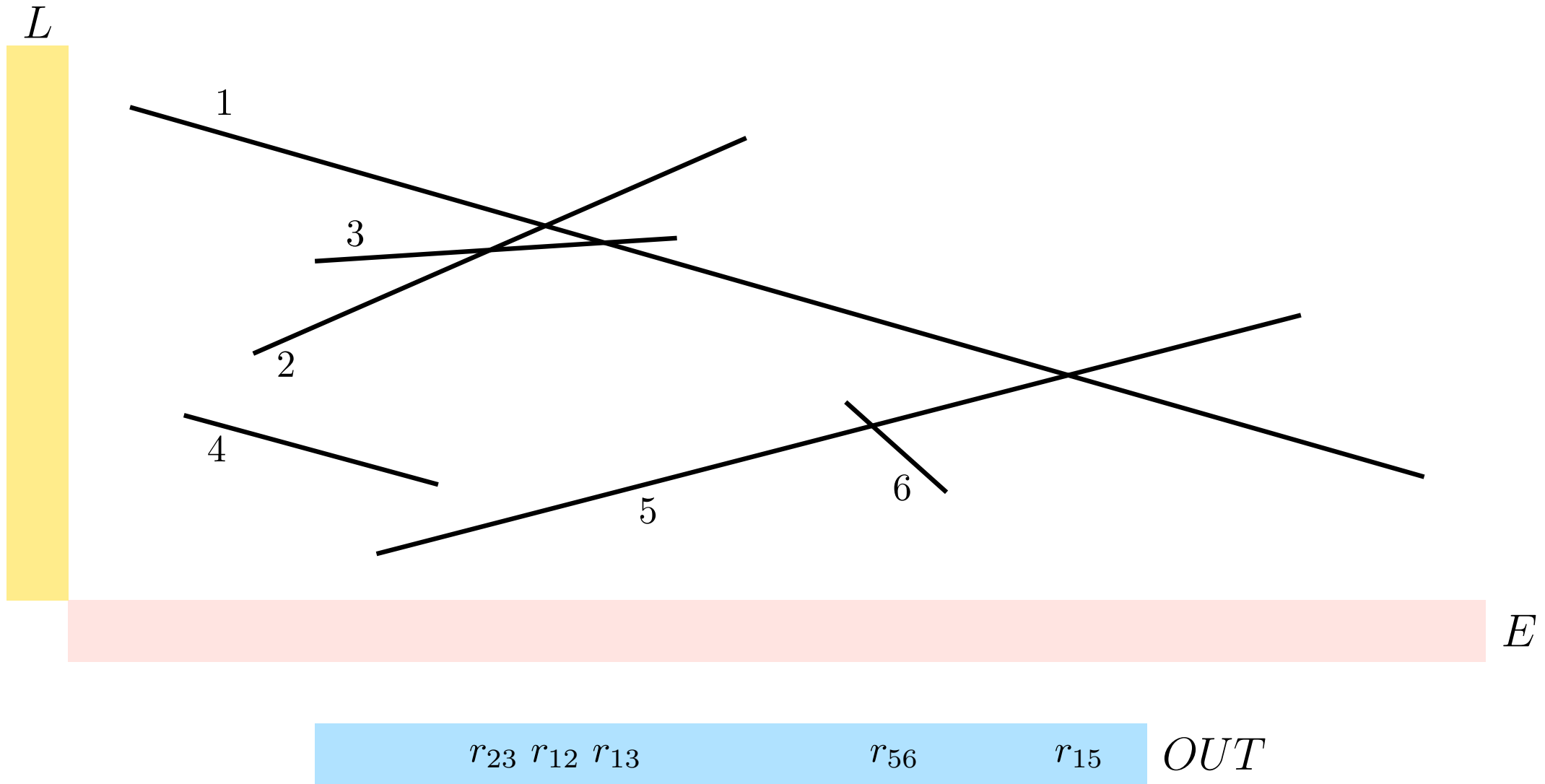
INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



INTERSECTING LINE SEGMENTS

Bentley-Ottman's algorithm: simulation



INTERSECTING LINE SEGMENTS

Bentley-Ottman's Algorithm

INTERSECTING LINE SEGMENTS

Bentley-Ottman's Algorithm

Correctness

- The algorithm finds all intersections (due to Observation 2).
- The algorithm does not find any false intersection (all intersections reported exist).

INTERSECTING LINE SEGMENTS

Bentley-Ottman's Algorithm

Correctness

- The algorithm finds all intersections (due to Observation 2).
- The algorithm does not find any false intersection (all intersections reported exist).

Dealing with degenerate cases

- In order to deal with input data containing more than one point sharing the same abscissa, the event queue E must store the points in lexicographical order (and not only by abscissae).
- The algorithm can trivially detect whether two or more line segments intersect in more than one point (i.e., intersect in a line segment), since it stops at their endpoints.
- A slight modification also allows to deal with input data in which three or more line segments intersect at the same point: in this case, the algorithm inverts their order in the sweep line at the intersection point event.

INTERSECTING LINE SEGMENTS

Bentley-Ottman's Algorithm

Data structures

INTERSECTING LINE SEGMENTS

Bentley-Ottman's Algorithm

Data structures

Sweep line, L :

Keeps the total order of the stabbed line segments and supports:

- $\text{insert}(s)$
- $\text{delete}(s)$
- $\text{transpose}(s_1, s_2)$
- $\text{previous}(s)$
- $\text{next}(s)$

A balanced binary tree allows to perform each of these operations in $O(\log n)$ time.

INTERSECTING LINE SEGMENTS

Bentley-Ottman's Algorithm

Data structures

Events queue, E :

Keeps the total order of the events and supports:

- minimum (report and extract)
- insert(p)
- memberQ(p)

A priority queue (possibly implemented with a balanced binary tree) allows to perform each of these operations in $O(\log n)$ time.

INTERSECTING LINE SEGMENTS

Bentley-Ottman's Algorithm

Complexity (time)

INTERSECTING LINE SEGMENTS

Bentley-Ottman's Algorithm

Complexity (time)

Initialization (insert initial events): $O(n \log n)$ time

Advance (performed $2n + k$ times):

Step 1 (find next event): $O(\log n)$ time

Steps 2, 3, or 4 (process event): $O(\log n)$ time

Step 5 (delete event): $O(\log n)$ time

Overall running time: $O((n + k) \log n)$

INTERSECTING LINE SEGMENTS

Bentley-Ottman's Algorithm

Complexity (time)

Initialization (insert initial events): $O(n \log n)$ time

Advance (performed $2n + k$ times):

Step 1 (find next event): $O(\log n)$ time

Steps 2, 3, or 4 (process event): $O(\log n)$ time

Step 5 (delete event): $O(\log n)$ time

Overall running time: $O((n + k) \log n)$

The previous counting corresponds to the non-degenerate case.

When each intersection point, v_i , may correspond to more than two intersecting line segments, the total running time of the advance step of the algorithm is $O((\sum_{i=1}^k \text{degree}(v_i)) \log n)$. However, considering the points v_i as vertices of the graph induced by all segments and their intersections, we obtain:

$$\sum_{i=1}^k \text{degree}(v_i) \leq 2e = O(e) = O(v) = O(2n + k) = O(n + k).$$

INTERSECTING LINE SEGMENTS

Bentley-Ottman's Algorithm

Complexity (space)

INTERSECTING LINE SEGMENTS

Bentley-Ottman's Algorithm

Complexity (space)

At each step of the algorithm, the sweep line stores at most n line segments.

INTERSECTING LINE SEGMENTS

Bentley-Ottman's Algorithm

Complexity (space)

At each step of the algorithm, the sweep line stores at most n line segments.

However, the events queue may, at some point, contain *all* $O(n^2)$ intersection points!

INTERSECTING LINE SEGMENTS

Bentley-Ottman's Algorithm

Complexity (space)

At each step of the algorithm, the sweep line stores at most n line segments.

However, the events queue may, at some point, contain *all* $O(n^2)$ intersection points!

Quadratic space is a lot!

INTERSECTING LINE SEGMENTS

Bentley-Ottman's Algorithm

Complexity (space)

At each step of the algorithm, the sweep line stores at most n line segments.

However, the events queue may, at some point, contain *all* $O(n^2)$ intersection points!

Quadratic space is a lot!

This can be fixed: it is enough to slightly modify the algorithm so that the events queue, at each step of the algorithm, contains at most $n - 1$ intersection events.

How? Make sure that at each step, E only stores intersection points of line segments **adjacent** in L . Intersection points are deleted from E as soon as the intersecting segments stop being adjacent.

INTERSECTING LINE SEGMENTS

The decision problem

INTERSECTING LINE SEGMENTS

The decision problem

Input: n line segments in the plane, $s_i = (p_i, q_i)$, $i = 1 \dots n$.

Output: there is / there is not a pair of intersecting line segments, and report a witness.

INTERSECTING LINE SEGMENTS

The decision problem

Input: n line segments in the plane, $s_i = (p_i, q_i)$, $i = 1 \dots n$.

Output: there is / there is not a pair of intersecting line segments, and report a witness.

Solution

Bentley-Ottman's algorithm solves this problem in $O(n \log n)$ time.

INTERSECTING LINE SEGMENTS

The decision problem

Input: n line segments in the plane, $s_i = (p_i, q_i)$, $i = 1 \dots n$.

Output: there is / there is not a pair of intersecting line segments, and report a witness.

Solution

Bentley-Ottman's algorithm solves this problem in $O(n \log n)$ time.

Lower bound

The decision problem has complexity $\Omega(n \log n)$.

INTERSECTING LINE SEGMENTS

The decision problem

Input: n line segments in the plane, $s_i = (p_i, q_i)$, $i = 1 \dots n$.

Output: there is / there is not a pair of intersecting line segments, and report a witness.

Solution

Bentley-Ottman's algorithm solves this problem in $O(n \log n)$ time.

Lower bound

The decision problem has complexity $\Omega(n \log n)$.

Proof: by reduction from unicity of integers.

Given $x_1, \dots, x_n \in \mathbb{N}$, compute $p_i = (x_i, 0)$, $q_i = (x_i, 1)$ and $s_i = (p_i, q_i)$.

There exists a pair of intersecting line segments if and only if there exist duplicate numbers in the original set.

If you don't like degeneracies, consider the following points:

$p_i = (x_i - \frac{1}{2i}, 0)$ and $q_i = (x_i + \frac{1}{2i}, 1)$.

INTERSECTING LINE SEGMENTS

The problem of reporting all intersections

INTERSECTING LINE SEGMENTS

The problem of reporting all intersections

Corollary

The problem of reporting all intersection has complexity $\Omega(k + n \log n)$, because

- Reporting requires $\Omega(k)$ time
- Deciding requires $\Omega(n \log n)$ time

INTERSECTING LINE SEGMENTS

The problem of reporting all intersections

Corollary

The problem of reporting all intersection has complexity $\Omega(k + n \log n)$, because

- Reporting requires $\Omega(k)$ time
- Deciding requires $\Omega(n \log n)$ time

Optimal algorithm

An algorithm by Chazelle and Edelsbrunner solves this problem in $\Theta(k + n \log n)$ time.

INTERSECTING LINE SEGMENTS

The problem of reporting all intersections

Corollary

The problem of reporting all intersection has complexity $\Omega(k + n \log n)$, because

- Reporting requires $\Omega(k)$ time
- Deciding requires $\Omega(n \log n)$ time

Optimal algorithm

An algorithm by Chazelle and Edelsbrunner solves this problem in $\Theta(k + n \log n)$ time.

Consequences

- Deciding whether a polygon is simple can be solved in $O(n \log n)$ time.
- Deciding whether two simple polygons intersect can be solved in $O(n \log n)$ time.

INTERSECTING LINE SEGMENTS

FURTHER READING

- M. de Berg, O. Cheong, M. van Kreveld, M. Overmars: *Computational Geometry: Algorithms and Applications*, Springer.
- F. Preparata, M. Shamos, *Computational Geometry: An introduction*, Springer.
- J-D. Boissonnat, M. Yvinec, *Algorithmic Geometry*, Cambridge University Press.