**Sweepline Algorithm Summary**

**Main Idea of Sweepline Algorithm**

Sweep an imaginary line L across the plain, while

1. Maintaining the status of L, the status of L only changes at certain discrete points.
2. Fulfilling an invariant. When the sweepline encounters an event point the status is updated in such a way that the invariant is guaranteed to hold after the event point has been processed.

**Prove of correctness**

1. Prove that the status can’t change between two consecutive event points.
2. Prove that the invariant holds before and after an event point is processed.

**Depth of Intervals**

Given a set S of n intervals, compute the depth of S.

**Depth:** The depth of S is the maximum number of intervals passing over any point.

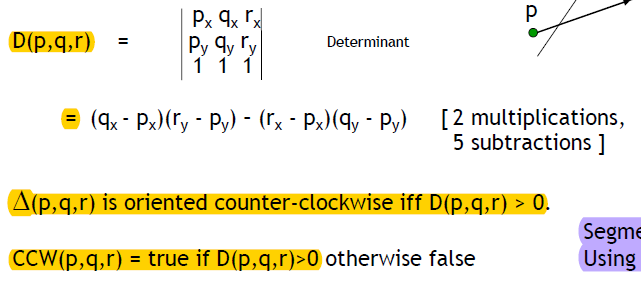
**Event points**: The points where a change of depth may occur, in this case is the endpoints of the intervals (both left endpoint and right endpoint). Event points are the place where changes happen.

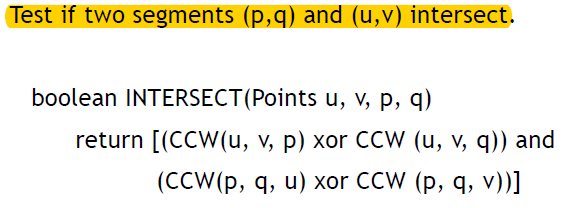
**Sweep-line Status**: The information stored with the Sweepline, in this case is the current depth.

**Theorem**: The depth of a set of n intervals in 1D can be computed in O(n\*logn) time using sweepline algorithm.

**Segment Intersection**

**Testing Intersections: Using CCW function (determinant)**

****

****

**Event Points:** Discrete points where sweep line status needs to be updated. (End points of the segments plus the intersection points).

**Sweep line status**: Store information along sweep line.

**Maintain invariant**: At any point in time, to the left of sweepline everything has been properly processed. (The order of the segments along the sweep line, no intersection among segments encountered by the sweepline).

**Structure:** Balanced binary search trees, each update can be done in O(logn) time.

**Running time of Detection of intersections:** O(n\*logn)

**Running time of Intersection Reporting**: O(n\*logn + h\*logn), where h is the size of the output.

**Convex Hulls and the sweepline technique**

**Convex:** A subset S of the plane is convex if for every pair of points p, q in S the straight line segment pq is completely contained in S.

**Convex Hull**: Convex Hull of a point set is the smallest convex set containing S.

**Theorem**: The intersection of two convex sets is a convex set, while the union of two convex sets is not a convex set.

**Finding the convex hull using the sweepline algorithm:**

**Event points:**  every points

**Invariant**: Correct upper convex hull of points to the left of L.

**Running time:** O(n\*logn)

**Closest pair using a sweepline approach with time complexity O(n\*logn)**

Given a set S of n points in the plane, the closest pair in S can be computed in O(n\*logn) time.

The sweepline algorithm utilities two sweeplines. (two parallel vertical sweep-lines)

Status Structure: A balanced binary search tree T storing all the points in S between these two parallel lines, ordered from top to bottom.

**Visibility**

Using a radial sweep line.

Initially we need to find all segments intersecting the ray, sort them with respect to distance from q.

Event points: Endpoints of the segments.

Status: The order of the segment along the ray

Invariant: The segments seen so far, and the order of the segments intersecting the ray.

Structure: Binary Search Tree: Insert, delete, query = O(logn)

**Running time of the algorithm**: O(n\*logn)