Carleton University

COMP 2402

Plane Sweep

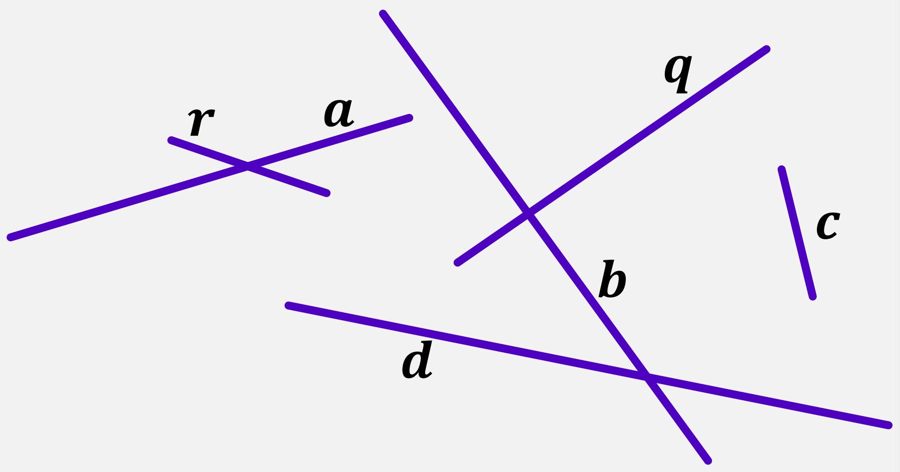
Alina Shaikhet

Introduction

The **plane sweep** (or **sweep line**) algorithm is a basic computational geometry algorithm for finding intersecting line segments.

Input: A set $\boldsymbol{S}$ of $\boldsymbol{n}$ line segments

Output: All pairs $\boldsymbol{s}, \boldsymbol{t} \in \boldsymbol{S}$ such that $\boldsymbol{s}$ intersects $\boldsymbol{t}$



$$(\boldsymbol{a}, \boldsymbol{r}), (\boldsymbol{b}, \boldsymbol{d}), (\boldsymbol{b}, \boldsymbol{q})$$

Plane sweep algorithm can be altered to solve many related computational-geometry problems, such as finding intersecting polygons.

Naïve algorithm

for each pair $s, t \in \binom{S}{2}$:
    if $(s$ intersects $t)$ then add (s, t) to the output

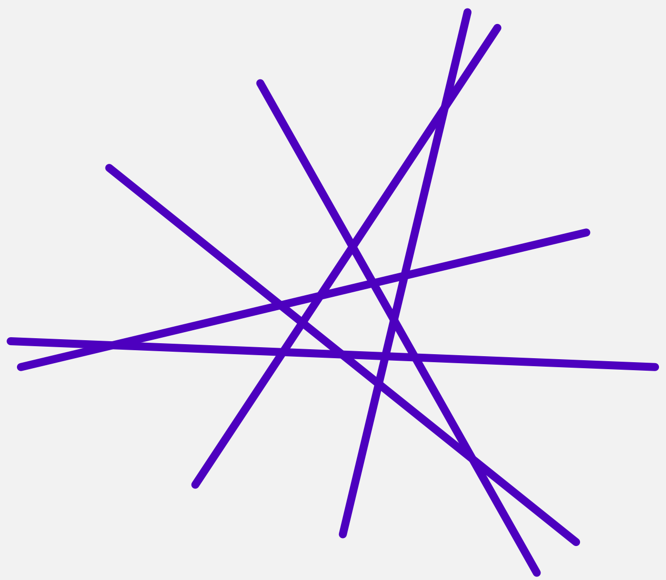
Running time is proportional to $\binom{n}{2} = \frac{n!}{2!\,(n-2)!} = \frac{n(n-1)}{2} = O(n^2)$

Can we do better?

Should we try to do better?

In the worst case every pair in S might intersect.

In this case the size of the output is $\binom{n}{2} = \Omega(n^2)$



Output-Sensitive Algorithms

The lower-bound on the size of the output is $\boldsymbol{\Omega}(\boldsymbol{n}^2)$.

However, in many cases, the number of intersecting pairs is much smaller than $\binom{\boldsymbol{n}}{\boldsymbol{2}}$

An **output-sensitive algorithm** is an algorithm whose running-time is sensitive to the number $\boldsymbol{k}$ of intersecting pairs.

The **Bentley–Ottmann plane-sweep algorithm** runs in time $\boldsymbol{O}\big((\boldsymbol{n}+\boldsymbol{k})\log \boldsymbol{n}\big)$, where $\boldsymbol{k}$ is the number of intersecting pairs of segments

This is much faster when $\boldsymbol{k} \ll \binom{\boldsymbol{n}}{\boldsymbol{2}}$

Bentley-Ottmann Plane Sweep Algorithm

1979

The plane sweep algorithm runs a simulation, in which a vertical line (the **sweep line**) moves from left to right across the plane, intersecting the input line segments in sequence as it moves.

The sweep line "pauses" at the two events:

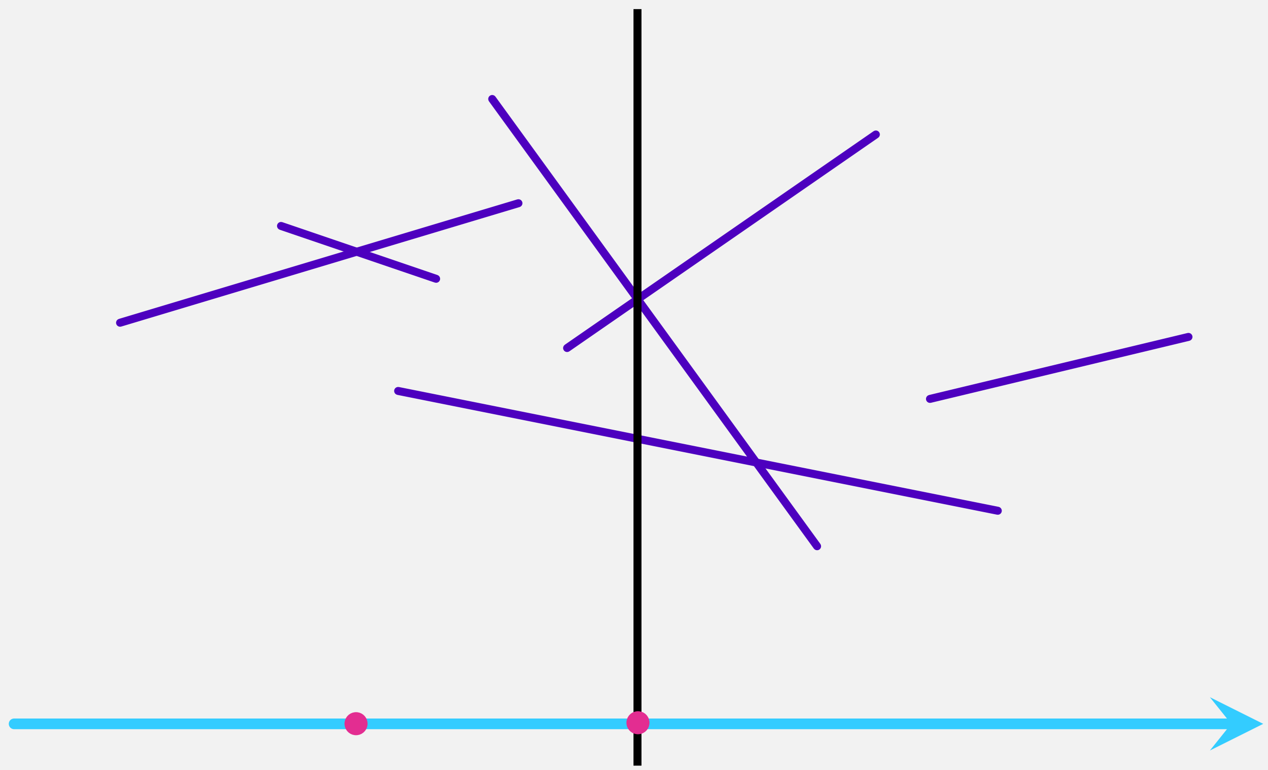
- the endpoints of segments ← **endpoint events**
- the intersection points ← **intersection events**

During intersection events, we record the intersecting pairs

For simplification we assume:

- No two segment endpoints or crossings have the same x-coordinate (no segment is vertical)
- No three segments intersect at a single point

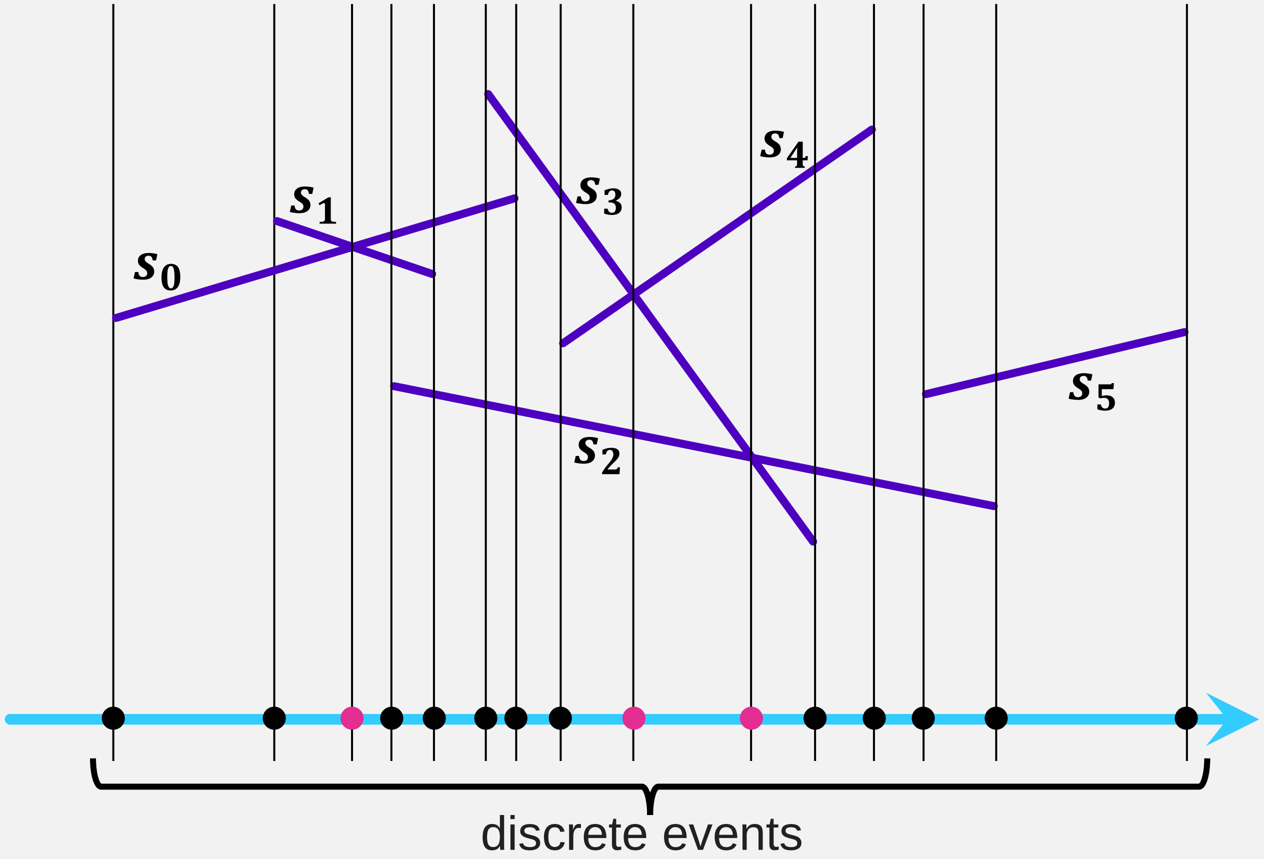
Bentley-Ottmann Plane Sweep Algorithm



There are two types of events that may happen during this simulation:

- Endpoint events.
  These events are easy to predict, as the endpoints are known from the input to the algorithm.
- The remaining events occur when the sweep line sweeps across an intersection of two segments s and t.
  Notice: just prior to the intersection event, the points of intersection of the sweep line with s and t are adjacent in the vertical ordering of the intersection points.

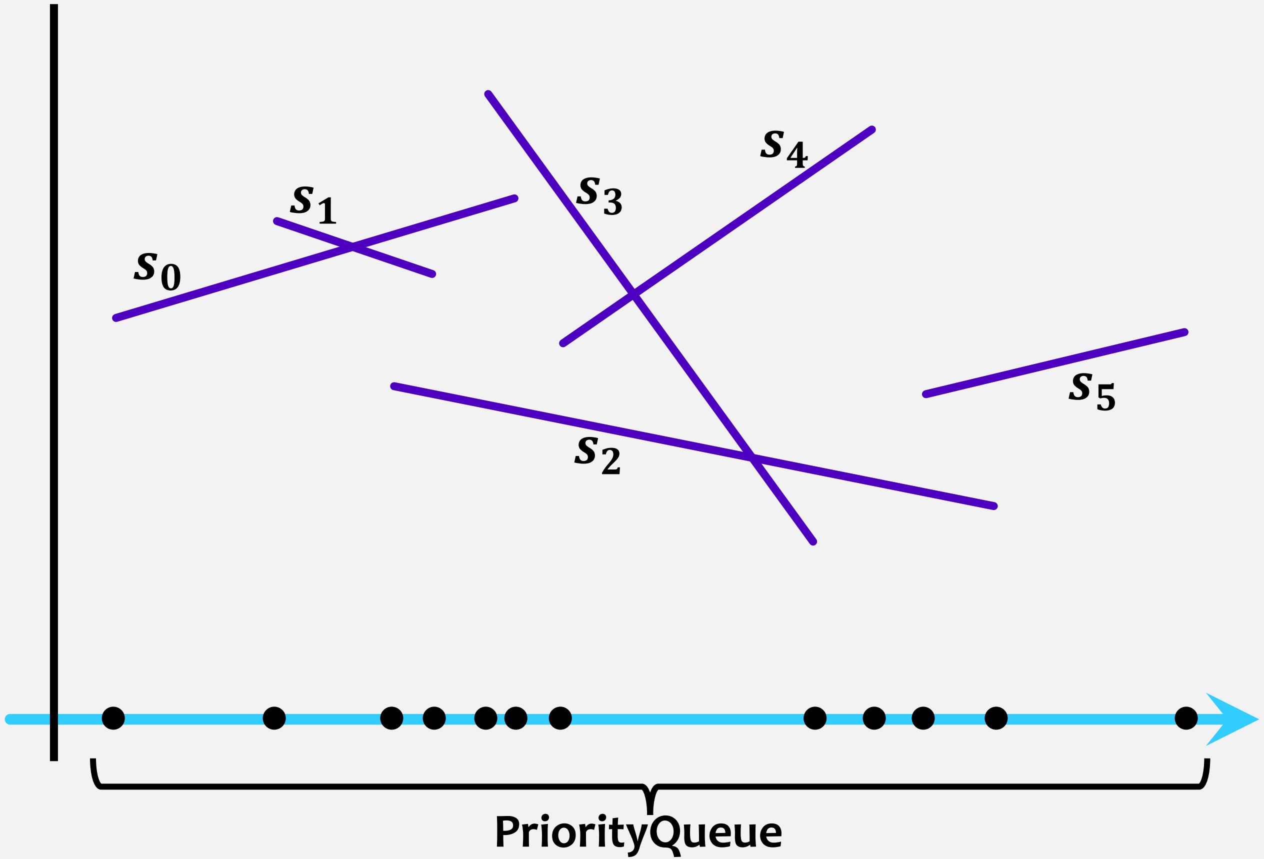
Bentley-Ottmann Plane Sweep Algorithm



The algorithm maintains two data structures:

- The **sweep-line status** is a **SortedSet** that stores the segments that currently intersect the sweep line, ordered from top to bottom (y-coordinate)
- The **event queue** is a **PriorityQueue** that stores events (segment endpoints and intersections) ordered from left to right (x-coordinate)

Bentley-Ottmann Plane Sweep Algorithm



1. **Initialize the sweep-line status:**

   a self-balancing binary search tree of the segments that cross the sweep line, ordered by the y-coordinates of the crossing points.

   Initially, **SortedSet** is empty.

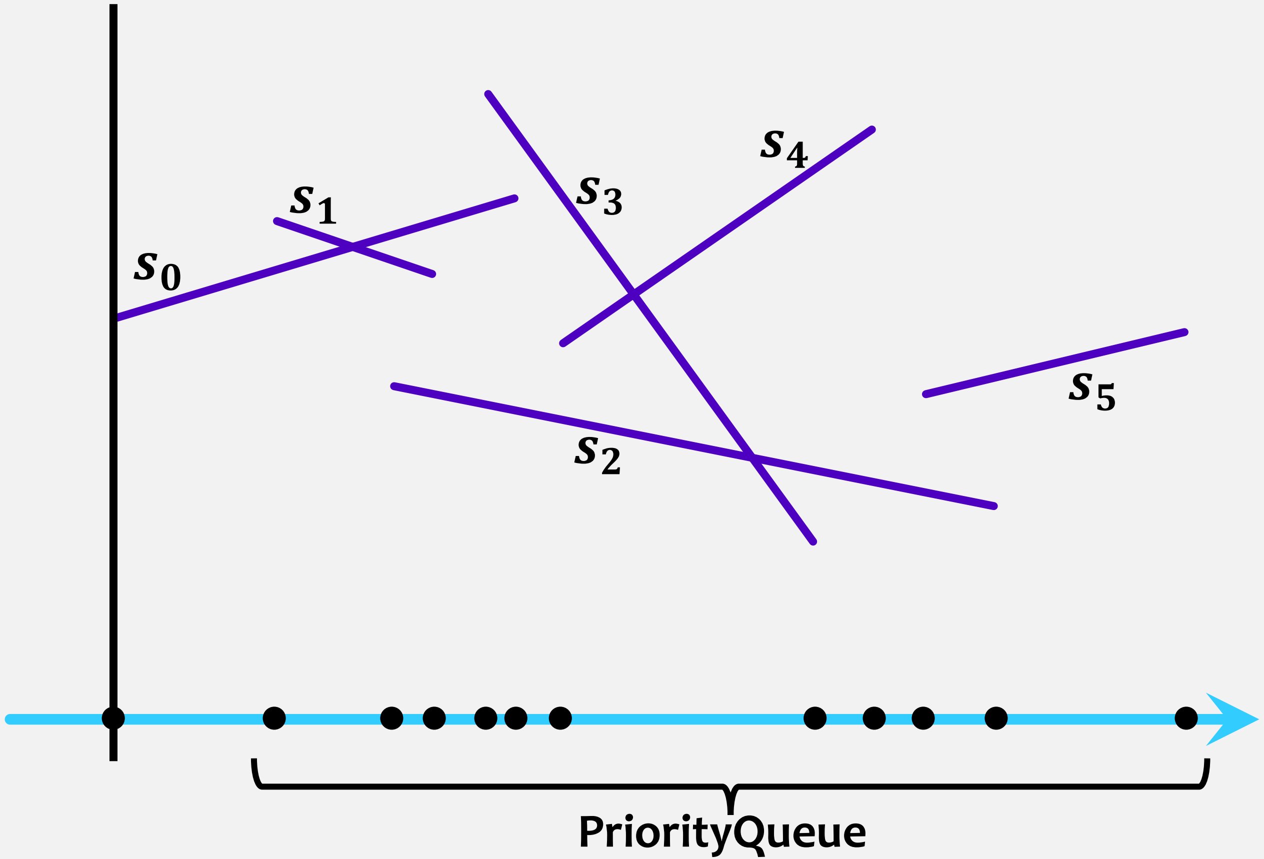
2. **Initialize an event queue:**

   Initially, **PriorityQueue** contains an event for each of the endpoints of the input segments ($2n$ events total)

3. While the **PriorityQueue** is nonempty, find and remove the event with the min x-coordinate.

Bentley-Ottmann Plane Sweep Algorithm

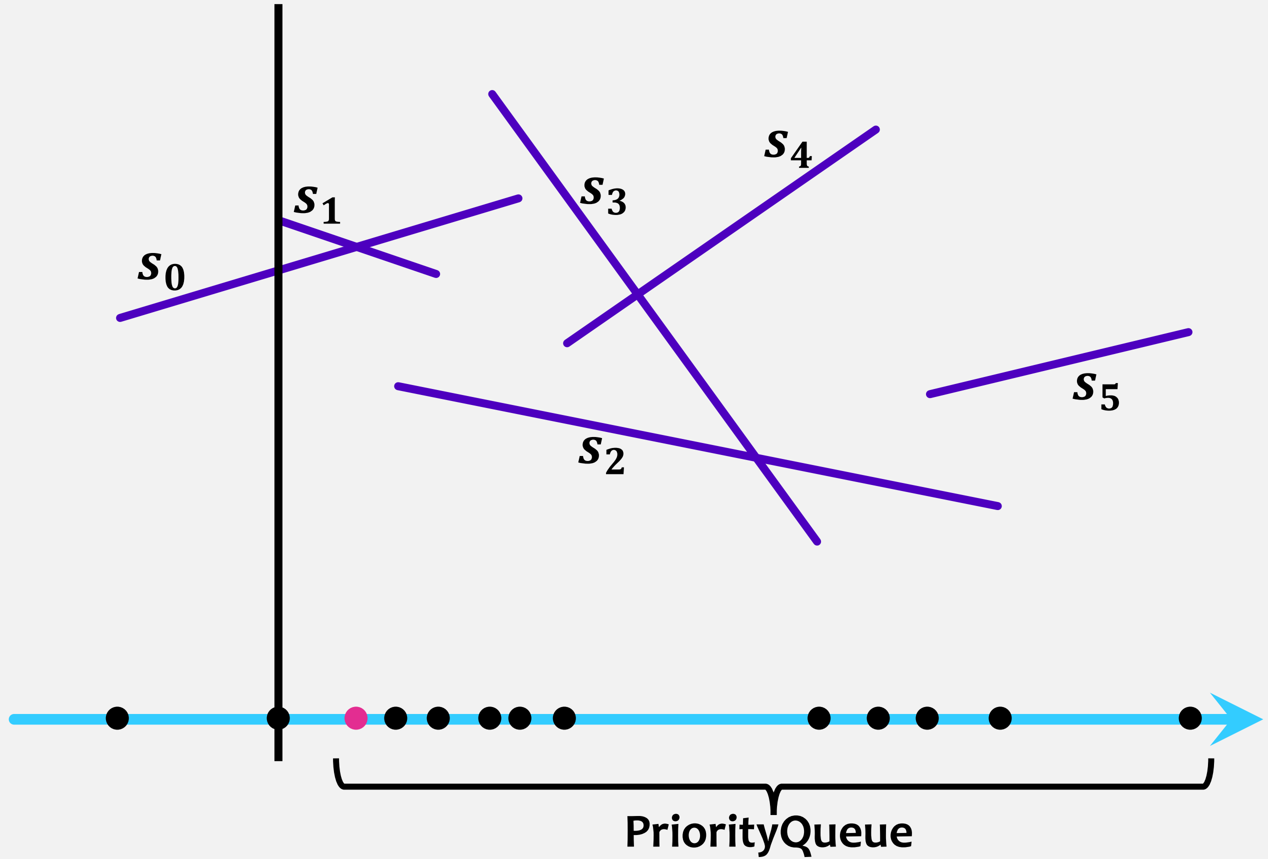
sweep-line status: $\langle s_0 \rangle$



Bentley-Ottmann Plane Sweep Algorithm

sweep-line status: $\langle s_0 \rangle$

$\langle s_1, s_0 \rangle$

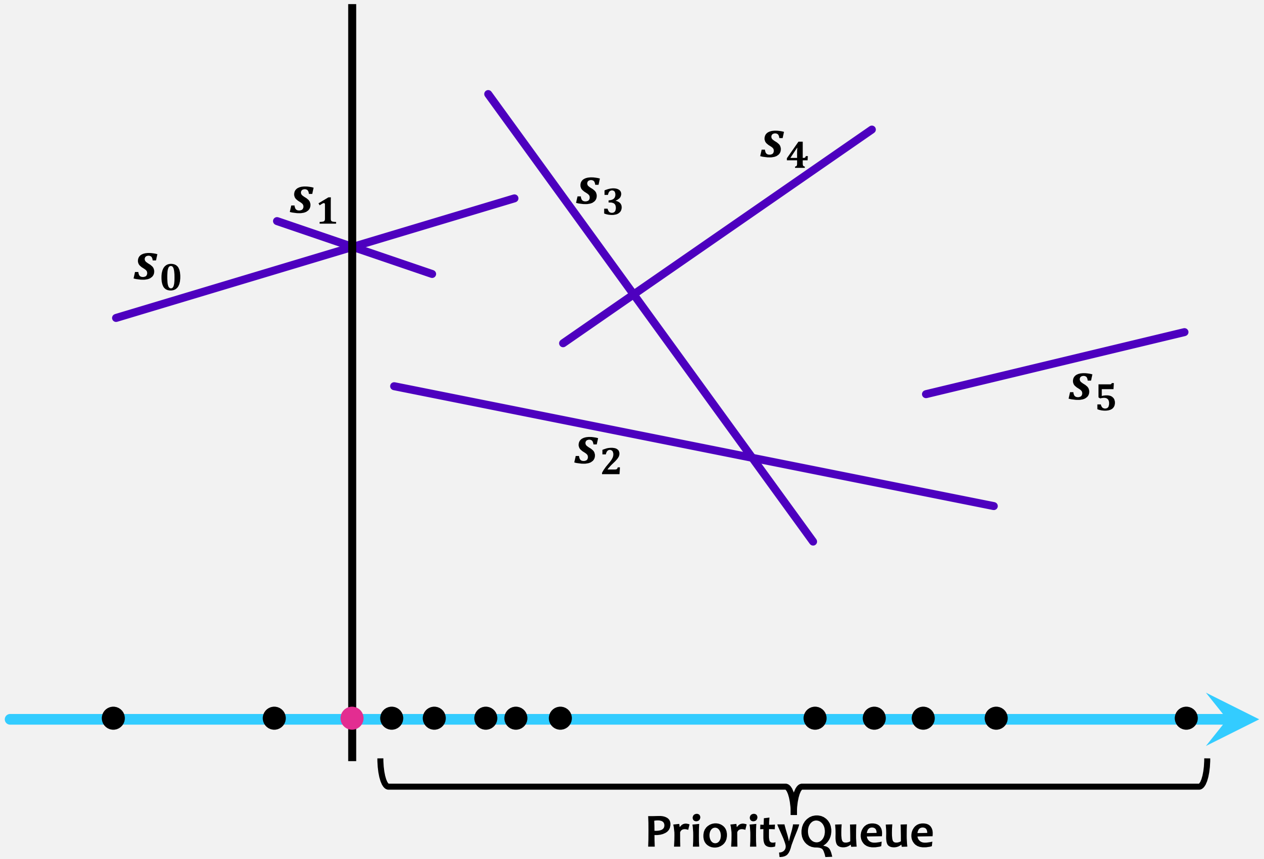


Bentley-Ottmann Plane Sweep Algorithm

sweep-line status: $\langle s_0 \rangle$

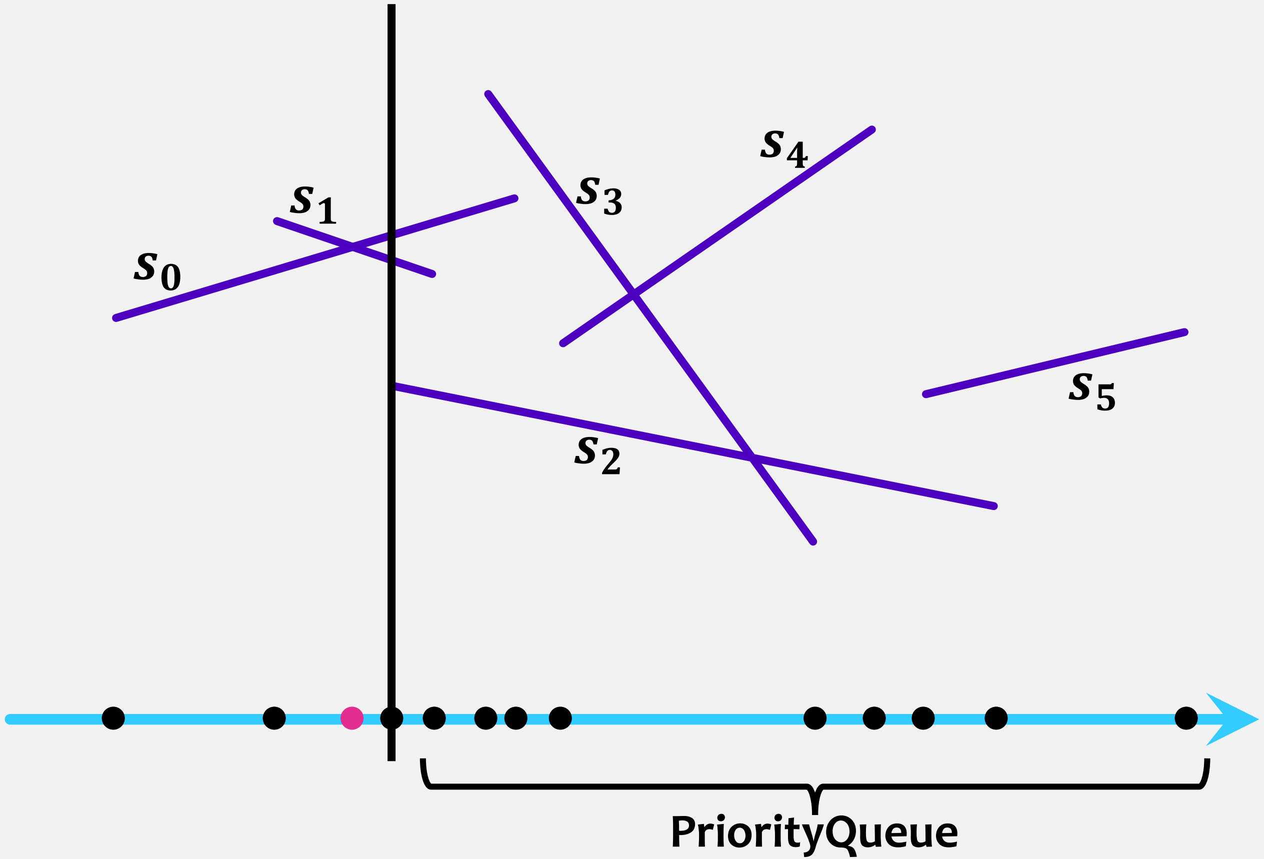
$\langle s_1, s_0 \rangle$

$\langle s_0, s_1 \rangle$

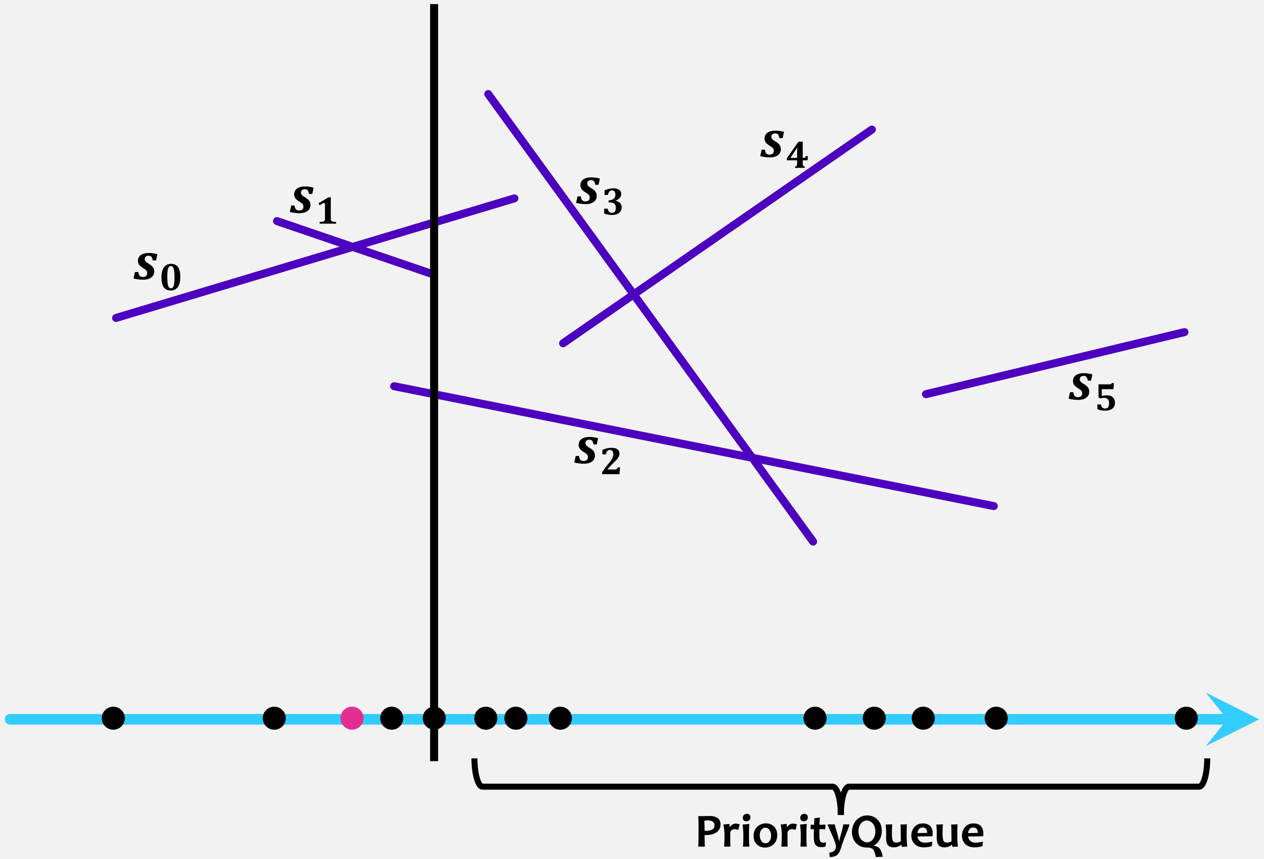


Bentley-Ottmann Plane Sweep Algorithm

sweep-line status: $\langle s_0 \rangle$
$\langle s_1, s_0 \rangle$
$\langle s_0, s_1 \rangle$
$\langle s_0, s_1, s_2 \rangle$



Bentley-Ottmann Plane Sweep Algorithm



sweep-line status: $\langle s_0 \rangle$

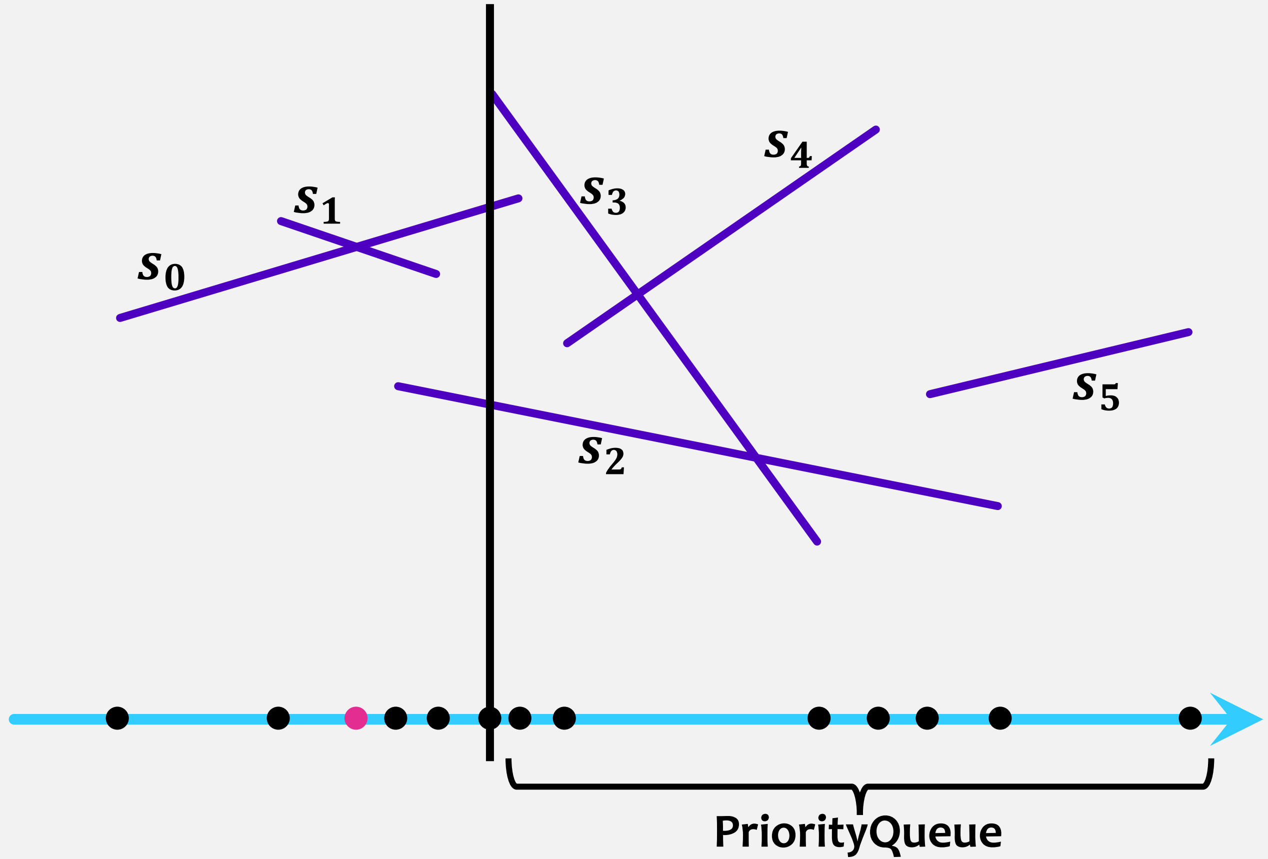
$\langle s_1, s_0 \rangle$

$\langle s_0, s_1 \rangle$

$\langle s_0, s_1, s_2 \rangle$

$\langle \boldsymbol{s_0}, \boldsymbol{s_2} \rangle$

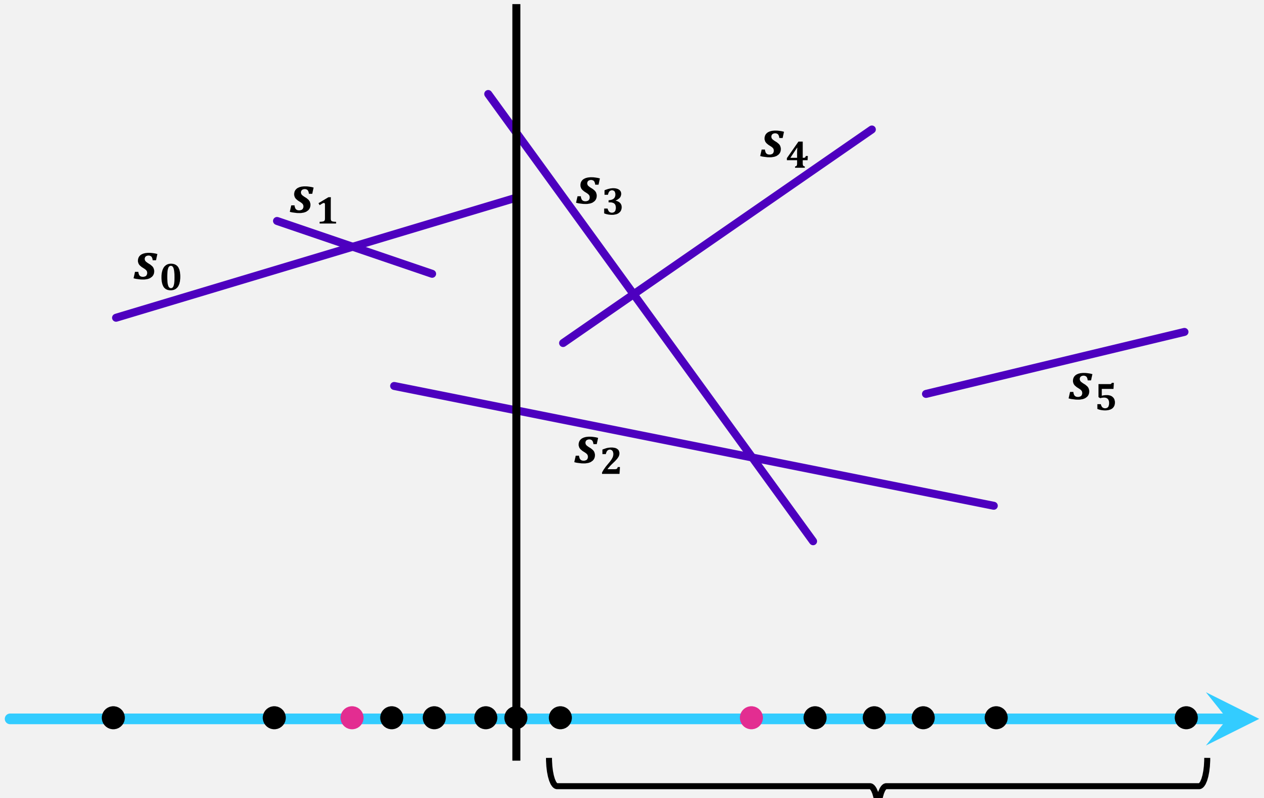
Bentley-Ottmann Plane Sweep Algorithm



sweep-line status:

$\langle s_0 \rangle$
$\langle s_1, s_0 \rangle$
$\langle s_0, s_1 \rangle$
$\langle s_0, s_1, s_2 \rangle$
$\langle s_0, s_2 \rangle$
$\langle \boldsymbol{s_3}, \boldsymbol{s_0}, \boldsymbol{s_2} \rangle$

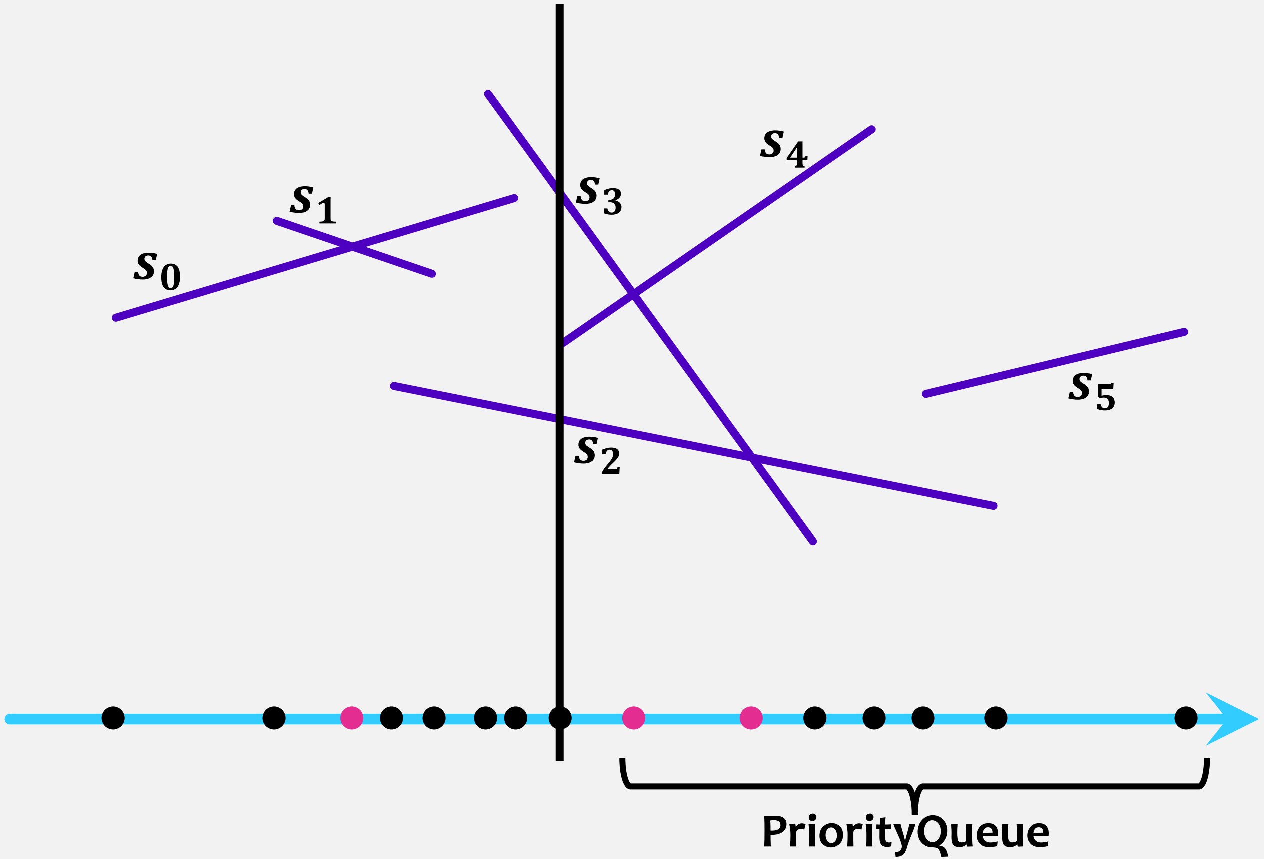
Bentley-Ottmann Plane Sweep Algorithm



sweep-line status:

$\langle s_0 \rangle$
$\langle s_1, s_0 \rangle$
$\langle s_0, s_1 \rangle$
$\langle s_0, s_1, s_2 \rangle$
$\langle s_0, s_2 \rangle$
$\langle s_3, s_0, s_2 \rangle$
$\langle s_3, s_2 \rangle$

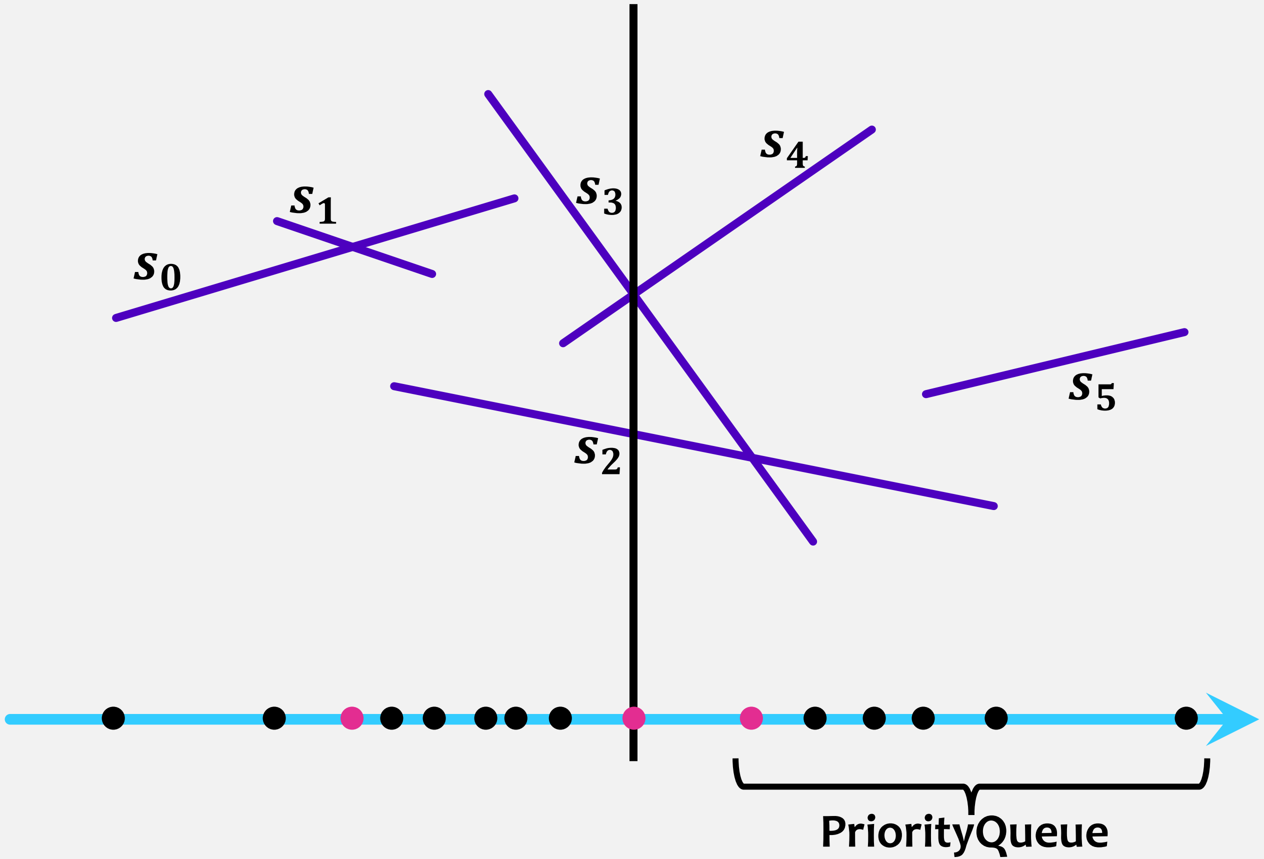
Bentley-Ottmann Plane Sweep Algorithm



sweep-line status:

$\langle s_0 \rangle$
$\langle s_1, s_0 \rangle$
$\langle s_0, s_1 \rangle$
$\langle s_0, s_1, s_2 \rangle$
$\langle s_0, s_2 \rangle$
$\langle s_3, s_0, s_2 \rangle$
$\langle s_3, s_2 \rangle$
$\boldsymbol{\langle s_3, s_4, s_2 \rangle}$

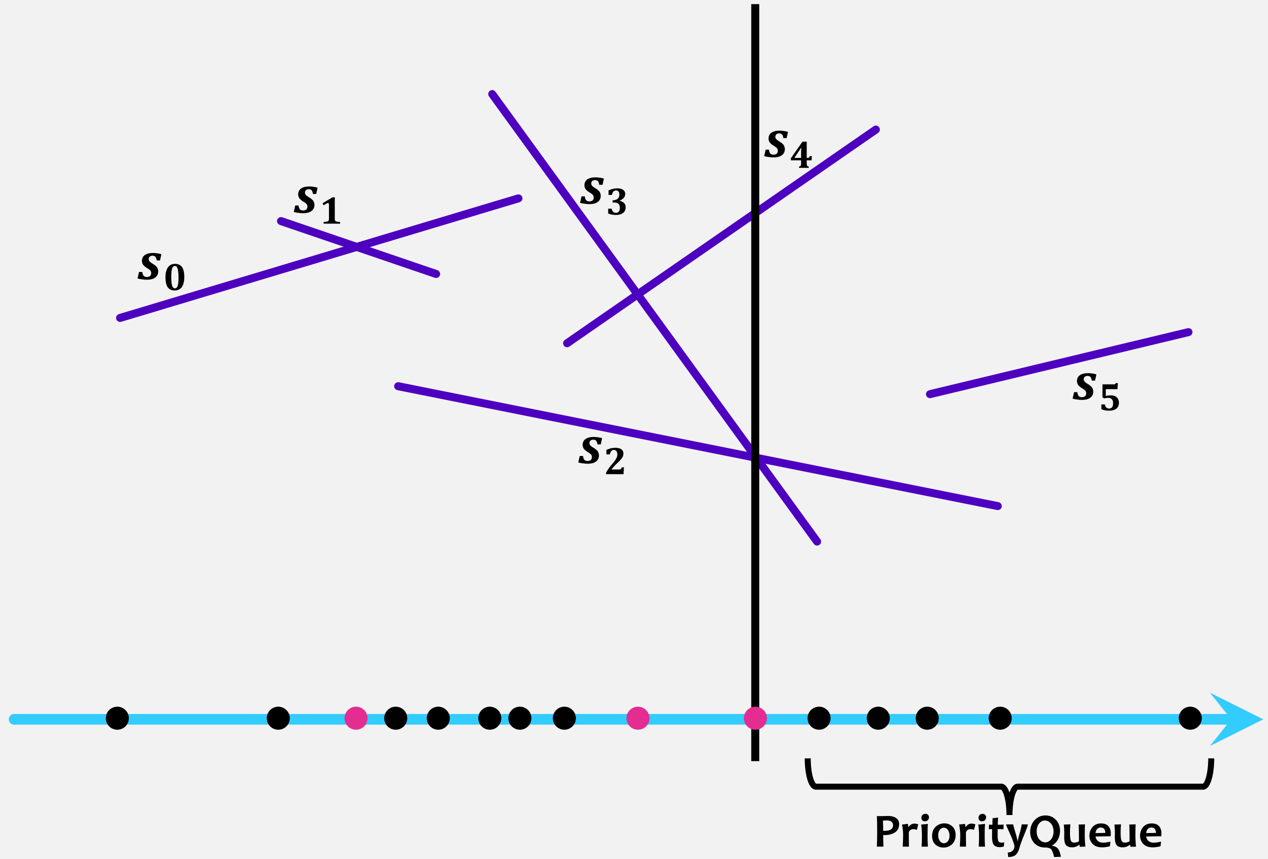
Bentley-Ottmann Plane Sweep Algorithm



sweep-line status:

$\langle s_0 \rangle$
$\langle s_1, s_0 \rangle$
$\langle s_0, s_1 \rangle$
$\langle s_0, s_1, s_2 \rangle$
$\langle s_0, s_2 \rangle$
$\langle s_3, s_0, s_2 \rangle$
$\langle s_3, s_2 \rangle$
$\langle s_3, s_4, s_2 \rangle$
$\langle \boldsymbol{s_4, s_3, s_2} \rangle$

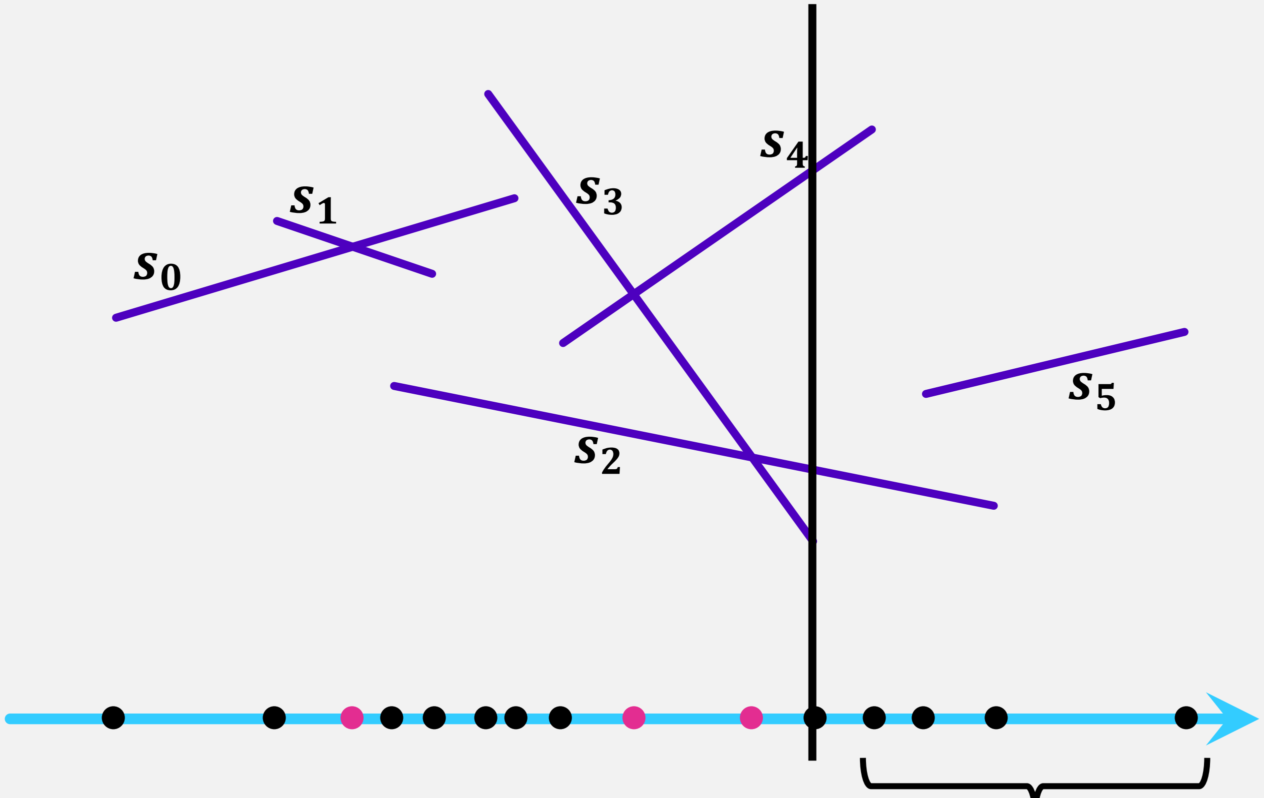
Bentley-Ottmann Plane Sweep Algorithm



sweep-line status:

$\langle s_0 \rangle$
$\langle s_1, s_0 \rangle$
$\langle s_0, s_1 \rangle$
$\langle s_0, s_1, s_2 \rangle$
$\langle s_0, s_2 \rangle$
$\langle s_3, s_0, s_2 \rangle$
$\langle s_3, s_2 \rangle$
$\langle s_3, s_4, s_2 \rangle$
$\langle s_4, s_3, s_2 \rangle$
$\boldsymbol{\langle s_4, s_2, s_3 \rangle}$

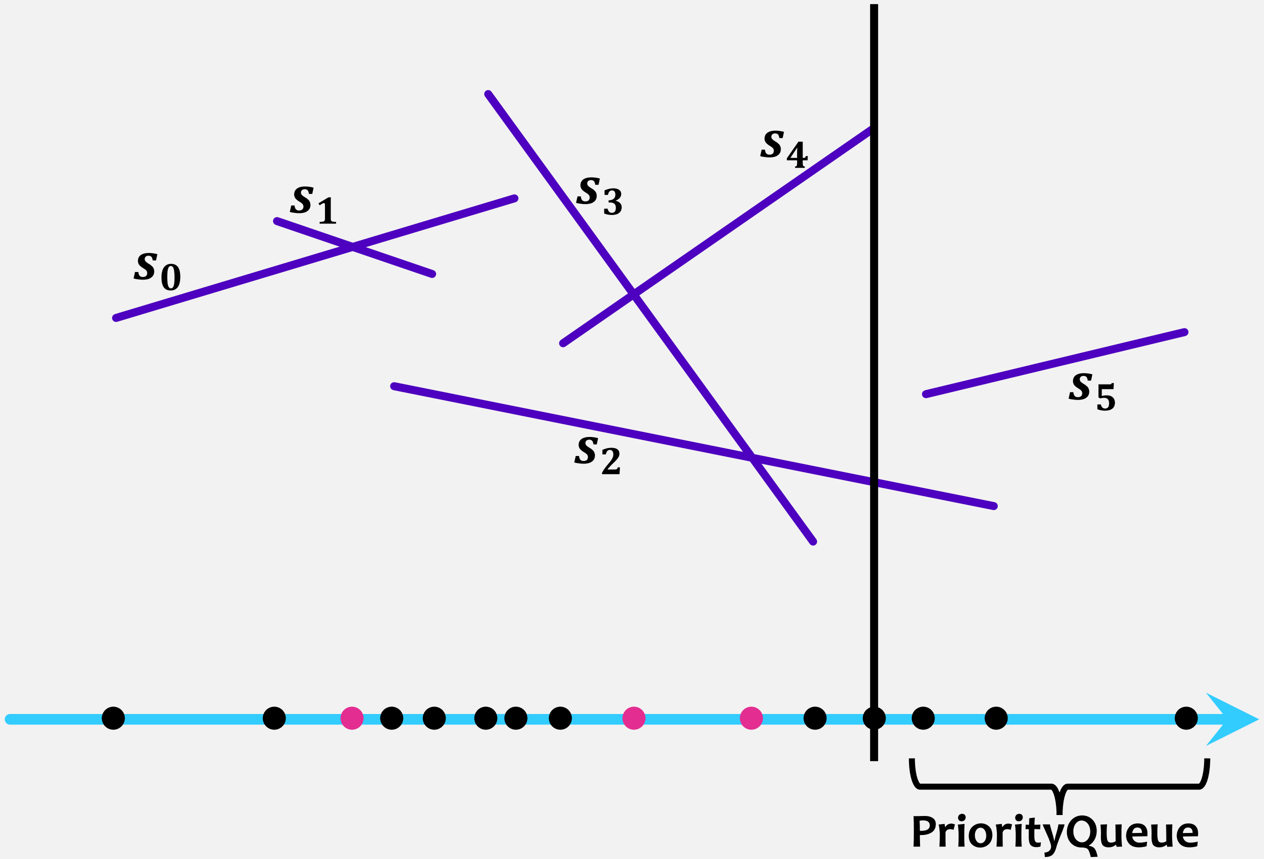
Bentley-Ottmann Plane Sweep Algorithm



sweep-line status:

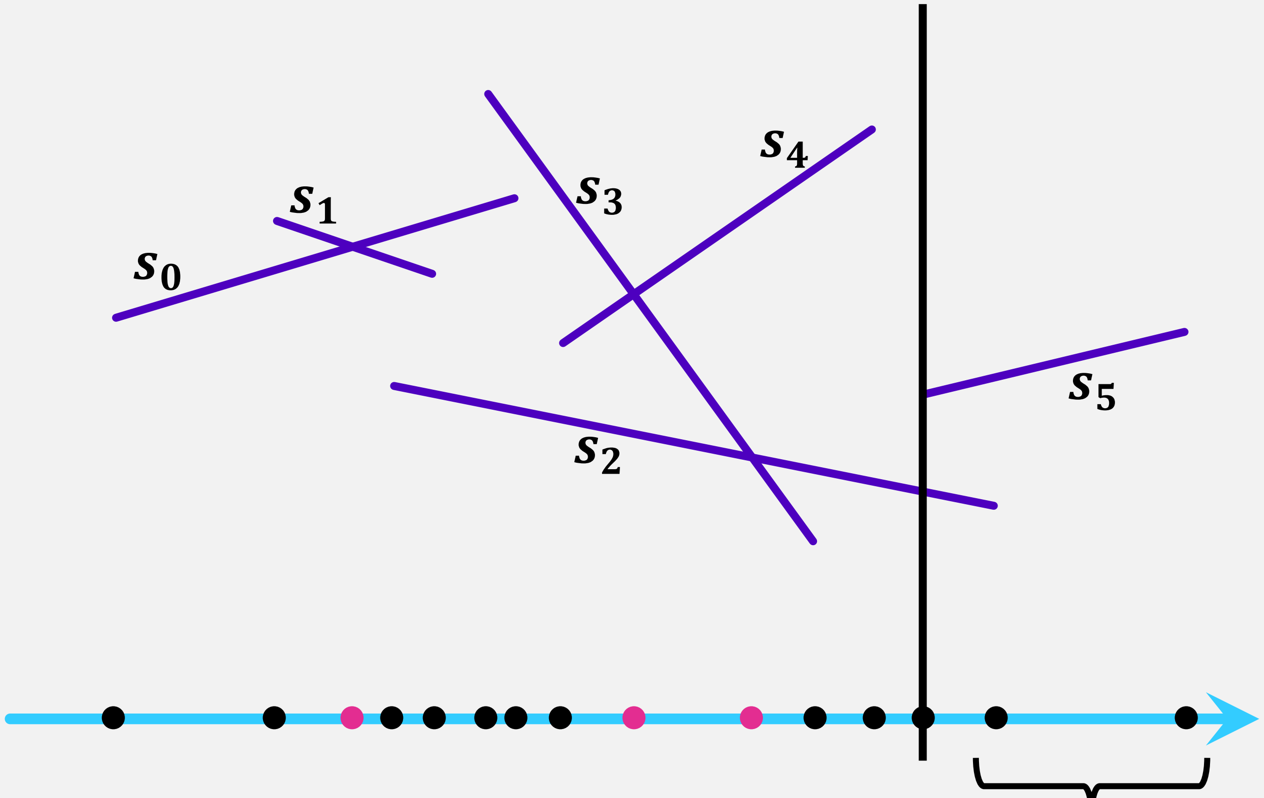
$\langle s_0 \rangle$
$\langle s_1, s_0 \rangle$
$\langle s_0, s_1 \rangle$
$\langle s_0, s_1, s_2 \rangle$
$\langle s_0, s_2 \rangle$
$\langle s_3, s_0, s_2 \rangle$
$\langle s_3, s_2 \rangle$
$\langle s_3, s_4, s_2 \rangle$
$\langle s_4, s_3, s_2 \rangle$
$\langle s_4, s_2, s_3 \rangle$
$\langle \boldsymbol{s_4, s_2} \rangle$

Bentley-Ottmann Plane Sweep Algorithm



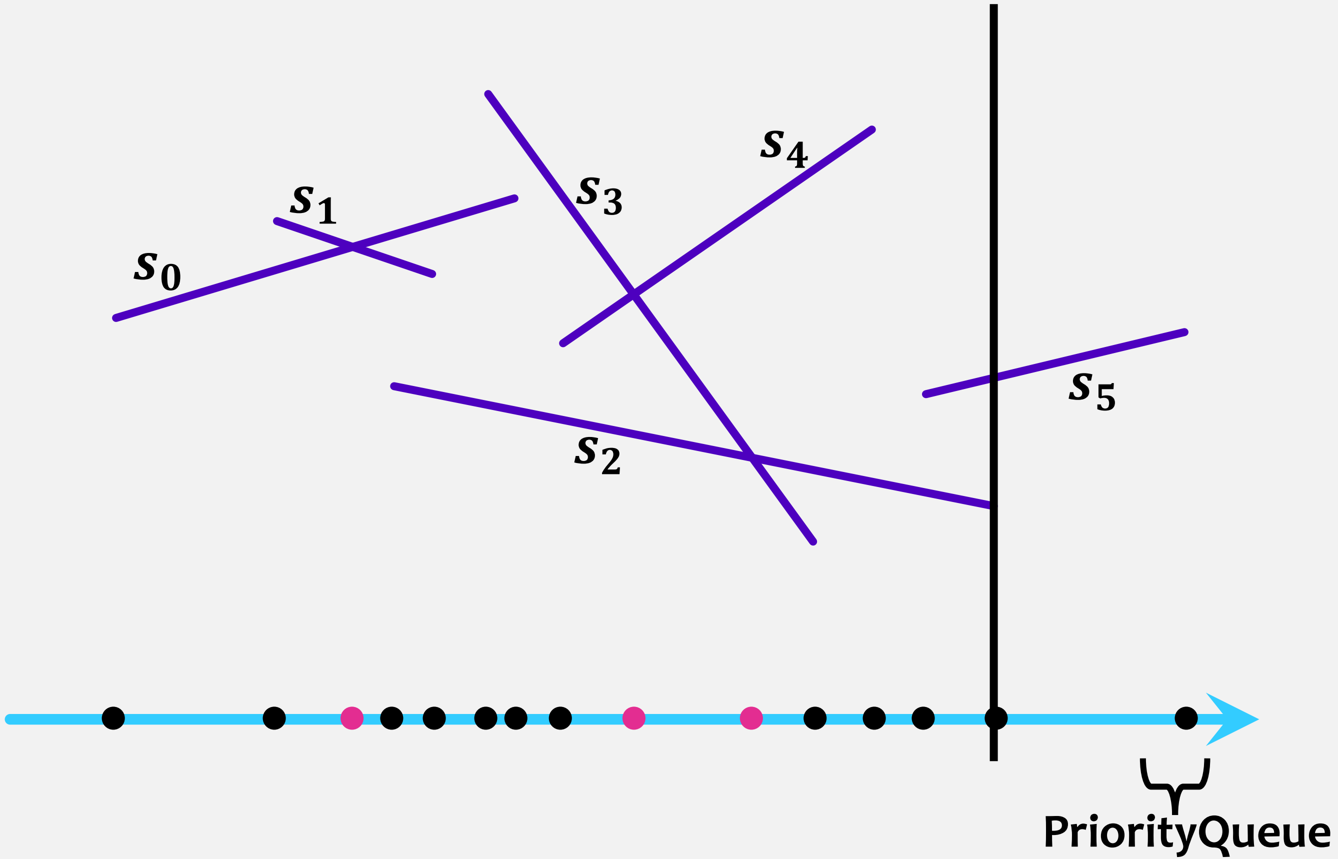
sweep-line status:
$\langle s_0 \rangle$
$\langle s_1, s_0 \rangle$
$\langle s_0, s_1 \rangle$
$\langle s_0, s_1, s_2 \rangle$
$\langle s_0, s_2 \rangle$
$\langle s_3, s_0, s_2 \rangle$
$\langle s_3, s_2 \rangle$
$\langle s_3, s_4, s_2 \rangle$
$\langle s_4, s_3, s_2 \rangle$
$\langle s_4, s_2, s_3 \rangle$
$\langle s_4, s_2 \rangle$
$\langle \boldsymbol{s_2} \rangle$

Bentley-Ottmann Plane Sweep Algorithm



sweep-line status:
$\langle s_0 \rangle$
$\langle s_1, s_0 \rangle$
$\langle s_0, s_1 \rangle$
$\langle s_0, s_1, s_2 \rangle$
$\langle s_0, s_2 \rangle$
$\langle s_3, s_0, s_2 \rangle$
$\langle s_3, s_2 \rangle$
$\langle s_3, s_4, s_2 \rangle$
$\langle s_4, s_3, s_2 \rangle$
$\langle s_4, s_2, s_3 \rangle$
$\langle s_4, s_2 \rangle$
$\langle s_2 \rangle$
$\langle s_5, s_2 \rangle$

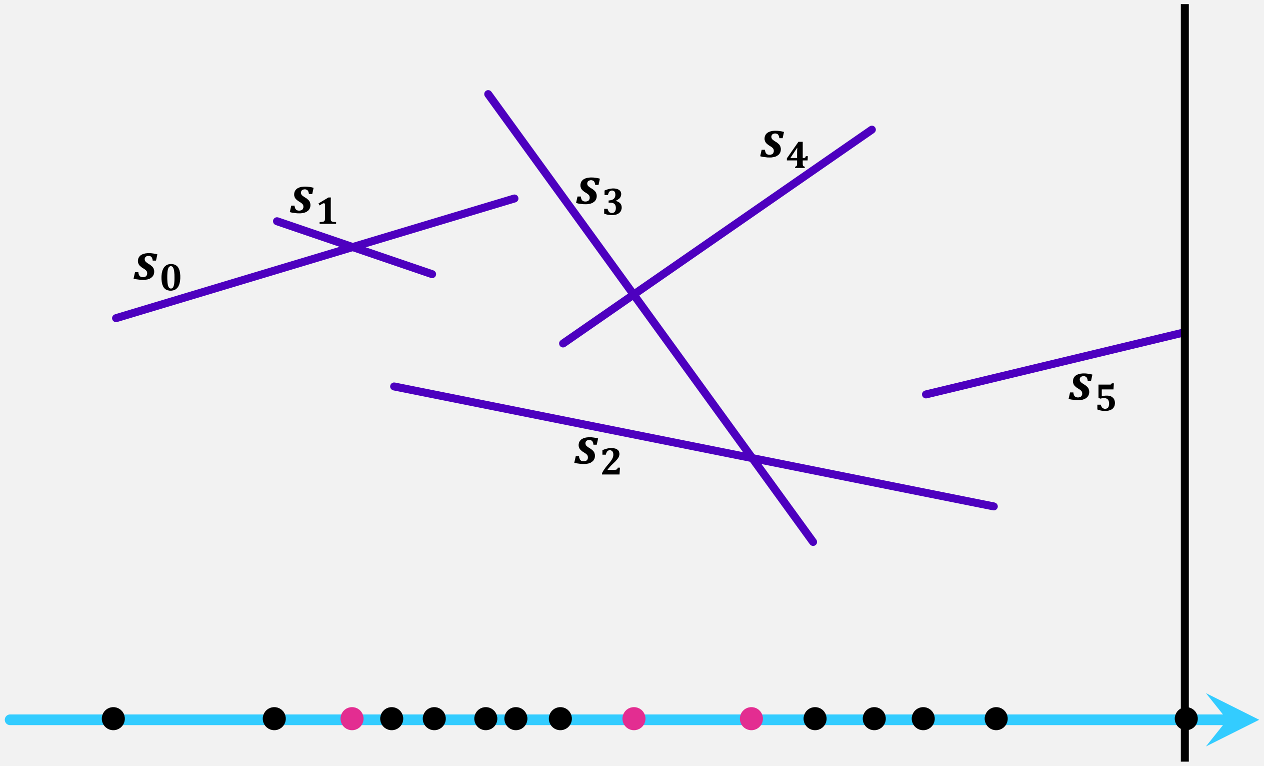
Bentley-Ottmann Plane Sweep Algorithm



sweep-line status:

$\langle s_0 \rangle$
$\langle s_1, s_0 \rangle$
$\langle s_0, s_1 \rangle$
$\langle s_0, s_1, s_2 \rangle$
$\langle s_0, s_2 \rangle$
$\langle s_3, s_0, s_2 \rangle$
$\langle s_3, s_2 \rangle$
$\langle s_3, s_4, s_2 \rangle$
$\langle s_4, s_3, s_2 \rangle$
$\langle s_4, s_2, s_3 \rangle$
$\langle s_4, s_2 \rangle$
$\langle s_2 \rangle$
$\langle s_5, s_2 \rangle$
$\langle \boldsymbol{s_5} \rangle$

Bentley-Ottmann Plane Sweep Algorithm



sweep-line status:

$\langle s_0 \rangle$
$\langle s_1, s_0 \rangle$
$\langle s_0, s_1 \rangle$
$\langle s_0, s_1, s_2 \rangle$
$\langle s_0, s_2 \rangle$
$\langle s_3, s_0, s_2 \rangle$
$\langle s_3, s_2 \rangle$
$\langle s_3, s_4, s_2 \rangle$
$\langle s_4, s_3, s_2 \rangle$
$\langle s_4, s_2, s_3 \rangle$
$\langle s_4, s_2 \rangle$
$\langle s_2 \rangle$
$\langle s_5, s_2 \rangle$
$\langle s_5 \rangle$
$\langle \quad \rangle$

Processing endpoint events

- For the **left** endpoint of a segment s:
    - Add s to the sweep line status
    - Check if s intersects the segment **above** or **below** it and add a crossing event to the event queue if necessary

- For the **right** endpoints of a segment s:
    - Remove s from the sweep line status
    - Check if the element **above** and **below** s cross and add a crossing event to the event queue if necessary

Processing crossing events

To process a crossing event where $\boldsymbol{s}$ and $\boldsymbol{t}$ cross:

- Switch the order of $\boldsymbol{s}$ and $\boldsymbol{t}$ in the sweep line status
- Check if $\boldsymbol{s}$ or $\boldsymbol{t}$ intersects the new elements **above** and **below** them in the sweep line and add crossing events to the event queue if necessary

Correctness

The Plane Sweep Algorithm is correct because any pair $\boldsymbol{s}$ and $\boldsymbol{t}$ that crosses will eventually become adjacent in the sweep-line status structure.

When $\boldsymbol{s}$ and $\boldsymbol{t}$ become adjacent, their crossing event is added to the event queue.

Analysis

We process $\boldsymbol{2n + k}$ events

Each event requires

- Adding an element to the event queue: $\boldsymbol{O}(\log \boldsymbol{n})$
- Getting an element from the event queue: $\boldsymbol{O}(\log \boldsymbol{n})$
- Searching the sweep-line status: $\boldsymbol{O}(\log \boldsymbol{n})$

Total running time is therefore $(\boldsymbol{2n + k})\boldsymbol{O}(\log \boldsymbol{n}) = \boldsymbol{O}\big((\boldsymbol{n + k}) \log \boldsymbol{n}\big)$

Summary

The **Bentley–Ottmann Plane-Sweep Algorithm** can compute all pairs of intersecting segments in $\boldsymbol{O}\big((\boldsymbol{n}+\boldsymbol{k})\log \boldsymbol{n}\big)$ time, where $\boldsymbol{k}$ is the number of pairs of segments that intersect.

Plane-sweep algorithms can solve many other problems:

- Given any set of objects, determine if any pair in the set intersect: $\boldsymbol{O}(\boldsymbol{n}\log \boldsymbol{n})$ time
- Find the closest pair of points among $\boldsymbol{n}$ points: $\boldsymbol{O}(\boldsymbol{n}\log \boldsymbol{n})$ time
- A data structure for the planar point location problem: $\boldsymbol{O}(\boldsymbol{n}\log \boldsymbol{n})$ space and $\boldsymbol{O}(\log \boldsymbol{n})$ query time