Line Segment Intersection

1389-2

Problem

Line segment intersection problem:

Given two sets of line segments, compute all intersections between a segment from one set and a segment from the other.

⋆ We consider the segments to be closed.

Simplified version:

Given a set S of n closed segments in the plane, report all intersection points among the segments in S.

Problem

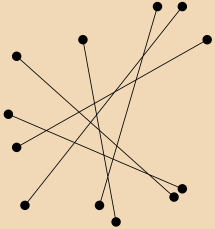
Line segment intersection problem:

Given two sets of line segments, compute all intersections between a segment from one set and a segment from the other.

⋆ We consider the segments to be closed.

Simplified version:

Given a set S of n closed segments in the plane, report all intersection points among the segments in S.

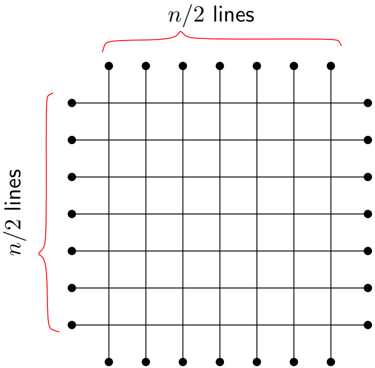


1st algorithm

- The brute-force algorithm clearly requires $\mathcal{O}(n^2)$ time.
- In a sense this is optimal: when each pair of segments intersects any algorithm must take $\Omega(n^2)$ time, because it has to report all intersections.

1st algorithm

- The brute-force algorithm clearly requires $\mathcal{O}(n^2)$ time.
- In a sense this is optimal: when each pair of segments intersects any algorithm must take $\Omega(n^2)$ time, because it has to report all intersections.



Output sensitive algorithm

Definition:

An algorithm whose running time depends not only on the number of segments in the input, but also on the number of intersection points.

In our case:

We want an algorithm that runs faster when the number of intersections is sub-quadratic.

Output sensitive algorithm

Definition:

An algorithm whose running time depends not only on the number of segments in the input, but also on the number of intersection points.

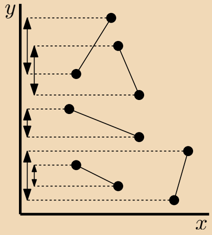
In our case:

We want an algorithm that runs faster when the number of intersections is sub-quadratic.

Plane sweep algorithm

y-intervals

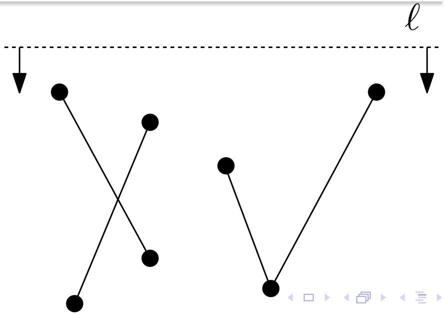
- Define the y-interval of a segment to be its orthogonal projection onto the y-axis.
- When the y-intervals of a pair of segments do not overlap then they cannot intersect.
- To find segments whose y-intervals overlap we use a **Plane sweep algorithm.**



Plane sweep algorithm

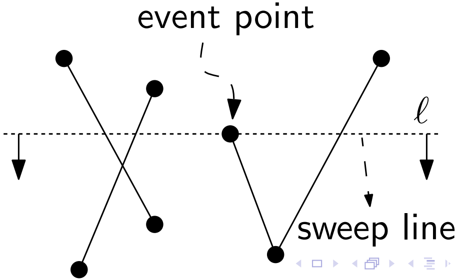
Plane sweep algorithm

- We imagine sweeping a line ℓ downwards over the plane, starting from a position above all segments.
- While we sweep the imaginary line, we keep track of all segments intersecting it so that we can find the pairs we need.
- The **status** of the sweep line is the set of segments intersecting it.



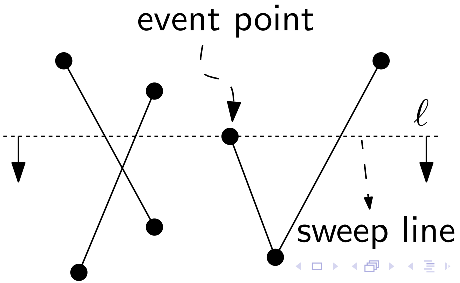
Plane sweep algorithm

- We imagine sweeping a line ℓ downwards over the plane, starting from a position above all segments.
- While we sweep the imaginary line, we keep track of all segments intersecting it so that we can find the pairs we need.
- The **status** of the sweep line is the set of segments intersecting it.



Plane sweep algorithm

- We imagine sweeping a line ℓ downwards over the plane, starting from a position above all segments.
- While we sweep the imaginary line, we keep track of all segments intersecting it so that we can find the pairs we need.
- The **status** of the sweep line is the set of segments intersecting it.



Plane sweep algorithm

when the sweep line reaches an event point:

- If the event point is the upper endpoint of a segment, then a new segment starts intersecting the sweep line and must be added to the status.
- If the event point is a lower endpoint, a segment stops intersecting the sweep line and must be deleted from the status.
- If the algorithm test pairs of segments for which there is a horizontal line that intersects both segments. (still quadratic).

Plane sweep algorithm

New algorithm:

- Order the segments from left to right as they intersect the sweep line.
- Test adjacent segments in the horizontal ordering for intersection.
- To maintain the sorted list, we need to take care of new event points.

Plane sweep algorithm

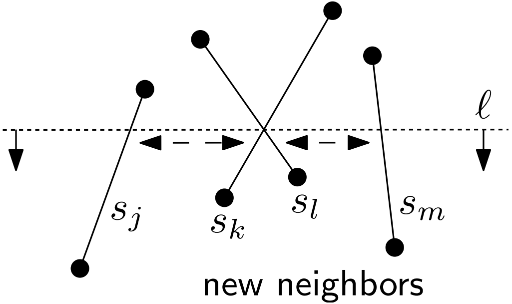
New algorithm:

- Order the segments from left to right as they intersect the sweep line.
- Test adjacent segments in the horizontal ordering for intersection.
- To maintain the sorted list, we need to take care of new event points.

Plane sweep algorithm

New algorithm:

- Order the segments from left to right as they intersect the sweep line.
- Test adjacent segments in the horizontal ordering for intersection.
- To maintain the sorted list, we need to take care of new event points.

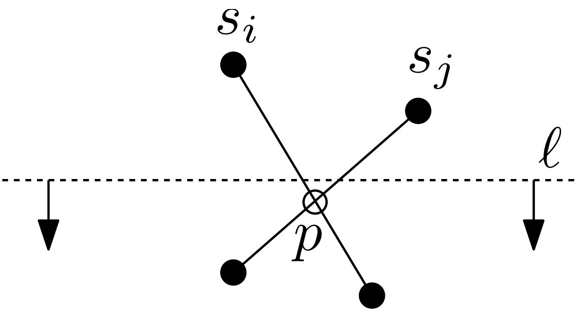


Do we still find all intersections?

Lemma 2.1 Let s_i and s_j be two non-horizontal segments whose interiors intersect in a single point p, and assume there is no third segment passing through p. Then there is an event point above p where s_i and s_j become adjacent and are tested for intersection.

Do we still find all intersections?

Lemma 2.1 Let s_i and s_j be two non-horizontal segments whose interiors intersect in a single point p, and assume there is no third segment passing through p. Then there is an event point above p where s_i and s_j become adjacent and are tested for intersection.



Plane sweep algorithm

Handling event points:

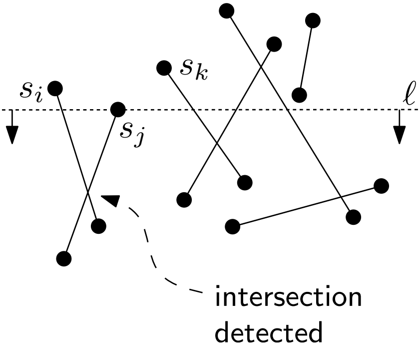
The event point is the upper endpoint of a segment:

- Insert the new segment in the sorted list.
- Check for intersection between the new segment and the segment before and after it in the sorted list.

Handling event points:

The event point is the upper endpoint of a segment:

- Insert the new segment in the sorted list.
- Check for intersection between the new segment and the segment before and after it in the sorted list.



Plane sweep algorithm

Handling event points:

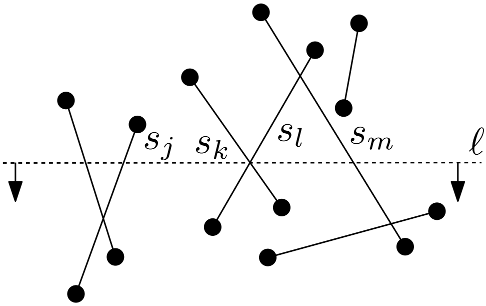
The event point is an intersection:

- Change the order of intersected segments in the sorted list.
- For each intersected segment, check for intersection between the segment and the new neighbor in the sorted list.

Handling event points:

The event point is an intersection:

- Change the order of intersected segments in the sorted list.
- For each intersected segment, check for intersection between the segment and the new neighbor in the sorted list.



Plane sweep algorithm

Handling event points:

The event point is a lower endpoint:

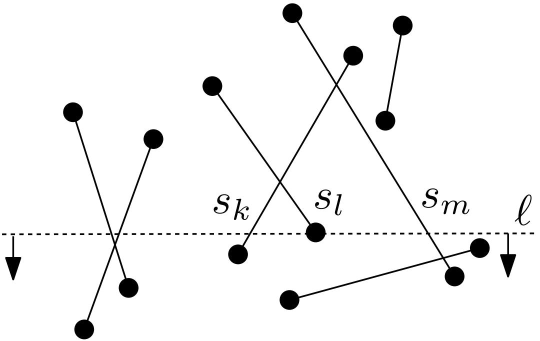
- Remove the segments from the sorted list.
- check for intersection between the neighboring segments.

Plane sweep algorithm

Handling event points:

The event point is a lower endpoint:

- Remove the segments from the sorted list.
- check for intersection between the neighboring segments.



Plane sweep algorithm

A data structure for handling event:

We need an event queue $\mathcal{Q}$ such that:

1. find and removes the next event that will occur from $\mathcal{Q}$. If two event points have the same y-coordinate, then the one with smaller x-coordinate will be returned.
2. Insert an event point in $\mathcal{Q}$. An insertion must be able to check whether an event is already present in $\mathcal{Q}$.

A data structure for handling event:

We need an event queue $\mathcal{Q}$ such that:

1. find and removes the next event that will occur from $\mathcal{Q}$. If two event points have the same y-coordinate, then the one with smaller x-coordinate will be returned.
2. Insert an event point in $\mathcal{Q}$. An insertion must be able to check whether an event is already present in $\mathcal{Q}$.

Plane sweep algorithm

Implementation of the event queue:

1. Define an order $\prec$ on event points: $p \prec q$ if and only if $p_y > q_y$ holds or $p_y = q_y$ and $p_x < q_x$ holds.
2. We store the event points in a balanced binary search tree, ordered according to $\prec$.
3. Fetching the next event and inserting an event and testing whether a given event is already present in $\mathcal{Q}$ take $\mathcal{O}(\log m)$ time, where m is the number of events in $\mathcal{Q}$.

Plane sweep algorithm

Implementation of the event queue:

1. Define an order $\prec$ on event points: $p \prec q$ if and only if $p_y > q_y$ holds or $p_y = q_y$ and $p_x < q_x$ holds.
2. We store the event points in a balanced binary search tree, ordered according to $\prec$.
3. Fetching the next event and inserting an event and testing whether a given event is already present in Q take $\mathcal{O}(\log m)$ time, where m is the number of events in Q.

Plane sweep algorithm

Implementation of the event queue:

1. Define an order $\prec$ on event points: $p \prec q$ if and only if $p_y > q_y$ holds or $p_y = q_y$ and $p_x < q_x$ holds.
2. We store the event points in a balanced binary search tree, ordered according to $\prec$.
3. Fetching the next event and inserting an event and testing whether a given event is already present in $\mathcal{Q}$ take $\mathcal{O}(\log m)$ time, where m is the number of events in $\mathcal{Q}$.

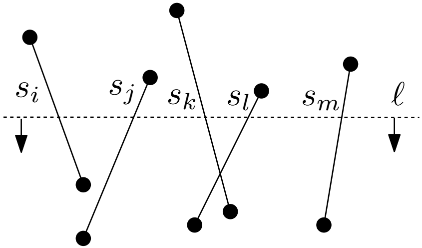
Plane sweep algorithm

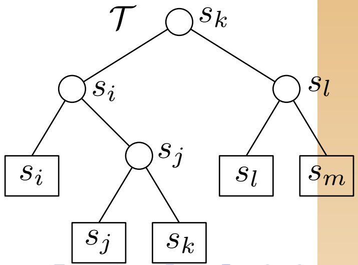
To maintain the sorted list of segments (status of the algorithm):

1. The status structure must be dynamic: segments must be inserted into or deleted from the structure.
2. We use a balanced binary search tree as status structure.
3. At each internal node, we store the segment from the rightmost leaf in its left subtree.

To maintain the sorted list of segments (status of the algorithm):

1. The status structure must be dynamic: segments must be inserted into or deleted from the structure.
2. We use a balanced binary search tree as status structure.
3. At each internal node, we store the segment from the rightmost leaf in its left subtree.

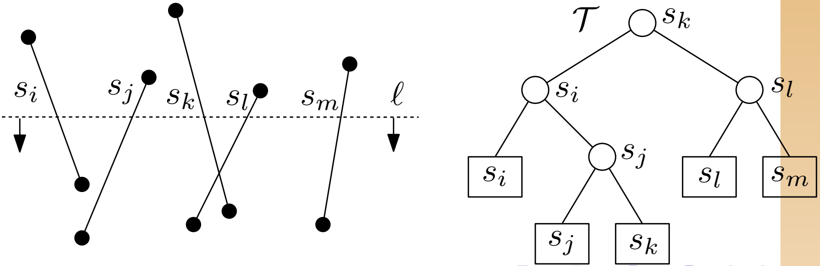




Plane sweep algorithm

To maintain the sorted list of segments (status of the algorithm):

1. The status structure must be dynamic: segments must be inserted into or deleted from the structure.
2. We use a balanced binary search tree as status structure.
3. At each internal node, we store the segment from the rightmost leaf in its left subtree.



Plane sweep algorithm

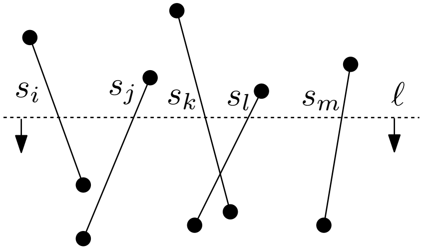
To search in $\mathcal{T}$ for the segment immediately to the left of some point p:

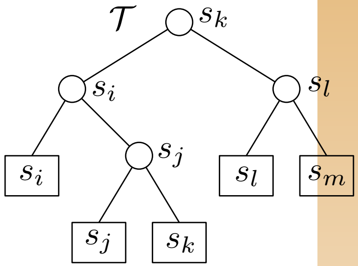
1. Traverse the tree until you meet a leaf.
2. This leaf, or the leaf immediately to the left of it, stores the segment we are searching for.
3. Therefore each update and neighbor search operation takes $\mathcal{O}(\log n)$ time.

Plane sweep algorithm

To search in $\mathcal{T}$ for the segment immediately to the left of some point p:

1. Traverse the tree until you meet a leaf.
2. This leaf, or the leaf immediately to the left of it, stores the segment we are searching for.
3. Therefore each update and neighbor search operation takes $\mathcal{O}(\log n)$ time.





Plane sweep algorithm

Algorithm FINDINTERSECTIONS(S)
Input: A set S of line segments in the plane.
Output: The set of intersection points among the segments in S, with for each intersection point the segments that contain it.

1. Initialize an empty event queue $\mathcal{Q}$. Next, insert the segment endpoints into $\mathcal{Q}$; when an upper endpoint is inserted, the corresponding segment should be stored with it.
2. Initialize an empty status structure $\mathcal{T}$.
3. **while** $\mathcal{Q}$ is not empty
4. Determine the next event point p in $\mathcal{Q}$ and delete it.
5. HANDLEEVENTPOINT(p)

Plane sweep algorithm

Algorithm HANDLEEVENTPOINT(p)

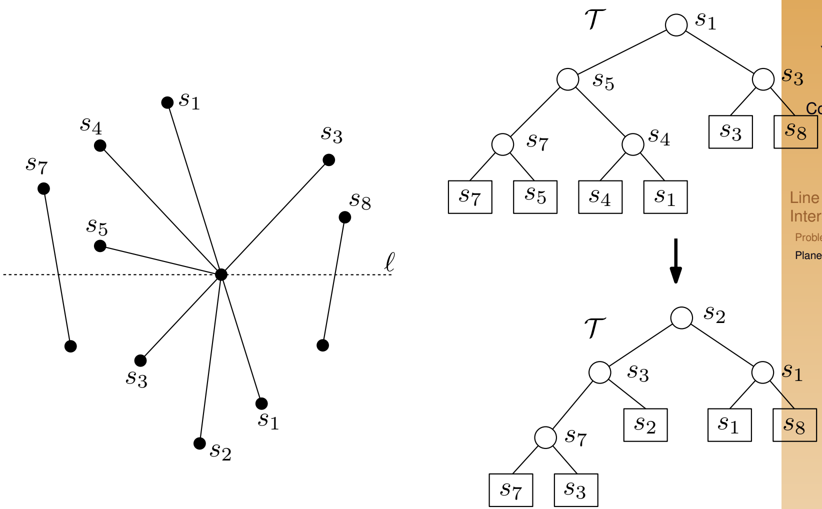
1. $U(p) \leftarrow$segments whose upper endpoint is p;
2. Find all segments stored in $\mathcal{T}$ that contain p;
   $L(p) \leftarrow$segments found whose lower endpoint is p;
   $C(p) \leftarrow$segments found that contain p in their interior.
3. **if** $|L(p) \cup U(p) \cup C(p)| > 1$
4.   **then** Report p as an intersection, together with
         $L(p), U(p)$, and $C(p)$.
5. Delete the segments in $L(p) \cup C(p)$ from $\mathcal{T}$.
6. Insert the segments in $U(p) \cup C(p)$ into $\mathcal{T}$.
7. **if** $U(p) \cup C(p) == \emptyset$
8.   **then** s_l and $s_r \leftarrow$the left and right neighbors of p in $\mathcal{T}$.
9.   FINDNEWEVENT(s_l, s_r, p)
10.   **else** $s' \leftarrow$the leftmost segment of $U(p) \cup C(p)$ in $\mathcal{T}$.
          $s_l \leftarrow$the left neighbor of s' in $\mathcal{T}$.
11.   FINDNEWEVENT(s_l, s', p)
12. $s'' \leftarrow$the rightmost segment of $U(p) \cup C(p)$ in $\mathcal{T}$.
13. $s_r \leftarrow$the right neighbor of s'' in $\mathcal{T}$.
14.   FINDNEWEVENT(s'', s_r, p)

Plane sweep algorithm

Algorithm FINDNEWEVENT(s_l, s_r, p)

1. **if** s_l and s_r intersect below the sweep line, or on it and to the right of the current event point p, and the intersection is not yet present as an event in $\mathcal{Q}$
2. **then** Insert the intersection point as an event into $\mathcal{Q}$.

Plane sweep algorithm



Plane sweep algorithm

Lemma 2.2

Algorithm FINDINTERSECTIONS computes all intersection points and the segments that contain it correctly.

Plane sweep algorithm

Lemma 2.3

The running time of Algorithm FINDINTERSECTIONS for a set S of n line segments in the plane is $\mathcal{O}(n \log n + I \log n)$, where I is the number of intersection points of segments in S.

Plane sweep algorithm

Theorem 2.4

Let S be a set of n line segments in the plane. All intersection points in S, with for each intersection point the segments involved in it, can be reported in $\mathcal{O}(n \log n + I \log n)$ time and $\mathcal{O}(n)$ space, where I is the number of intersection points.

THE END.