Intersections

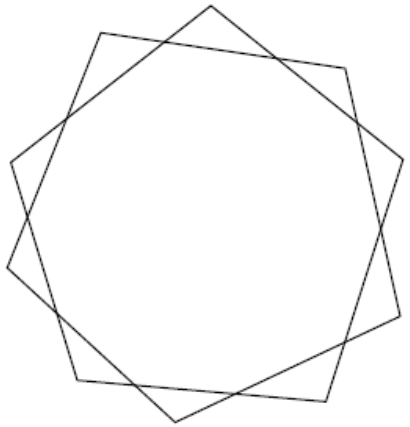
- Intersection of convex polygons
- Detection of line segments intersection
- Intersections of orthogonal line segments
- The Bentley-Ottmann algorithm

 16.10.2021

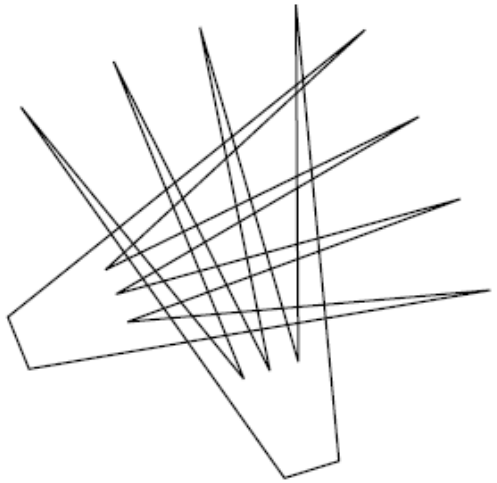
Intersection of Polygons

Lower bounds:

- Convex polygons with n and m vertices: $\Omega(n+m)$.



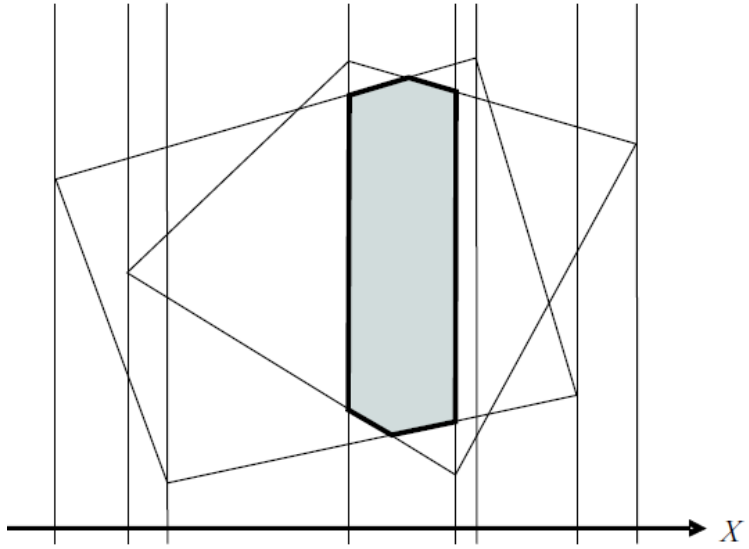
- Non-convex polygons: $\Omega(n^2)$.



Intersection of Convex Polygons

The algorithm:

1. Sort the vertices of the both polygons by x-coordinate in time $O(n+m)$ using their convexity.
2. Draw fictitious vertical lines through all the vertices which form $n+m-1$ strips. Intersection of a strip with a polygon is a **trapezoid** (maybe, degenerate).
3. Scan the strips from left to right:
   a. Calculate the both trapezoids: $O(1)$.
   b. Calculate their intersection: $O(1)$.
   c. Merge this intersection with those found in the previous strips: $O(1)$.



Complexity: $O(n+m)$.

Intersection of Line Segments

Lower bounds

Problem (INTERSECTION DETECTION). *Given n line segments, determine whether any two of them intersect.*

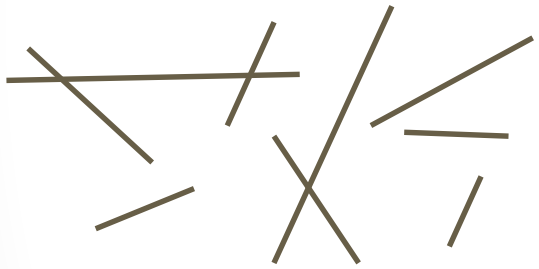
$$\Rightarrow \quad \Omega(n \log n)$$

Problem (ITEMS IDENTITY, $\Omega(n \log n)$). *Given n real numbers $x_1, \dots, x_n$. Are they different*?

Intersection of Line Segments

Lower bound

Problem (INTERSECTION OF LINE SEGMENTS). *Given n line segments on the plane. Find all their intersections.*



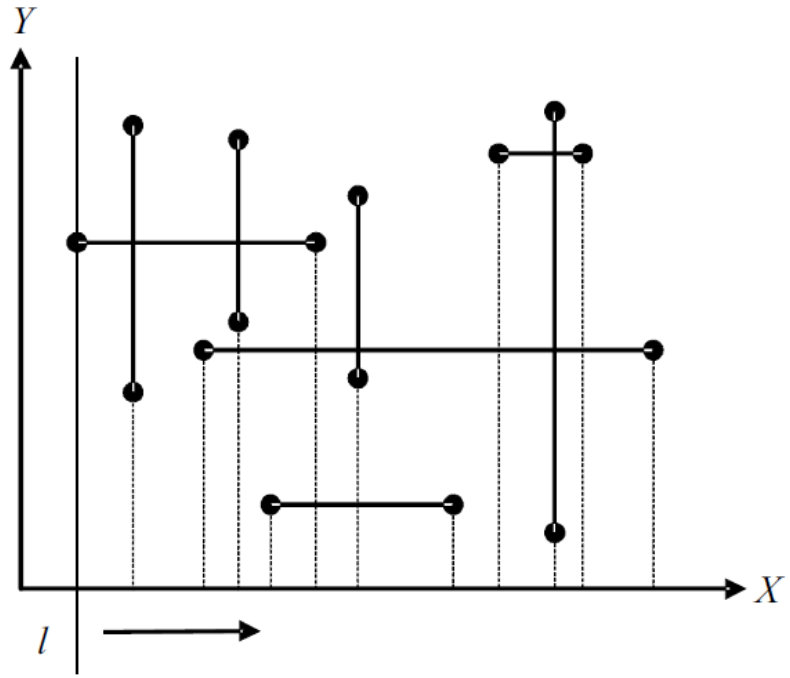
k intersections

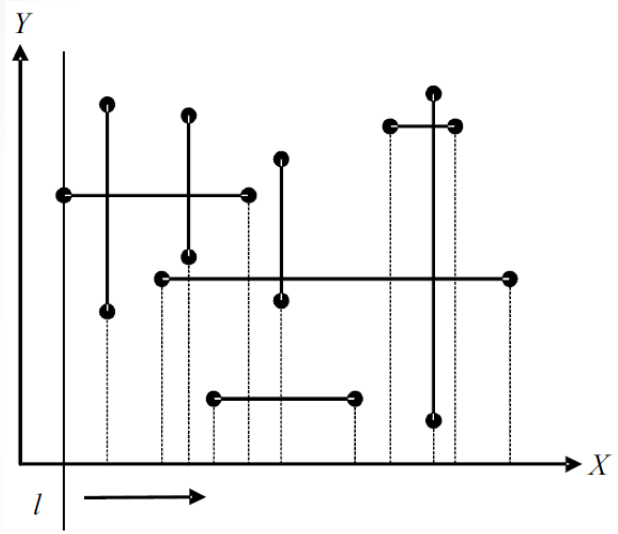
$\Rightarrow \quad \Omega(n \log n + k)$

Problem (DETECTION OF INTERSECTION, $\Omega(n \log n)$). *Given n line segments, determine whether any two of them intersect.*

Intersection of Orthogonal Line Segments

Problem. *Given n line segments on the plane each parallel to a coordinate axis (X or Y). Find all their intersections.*





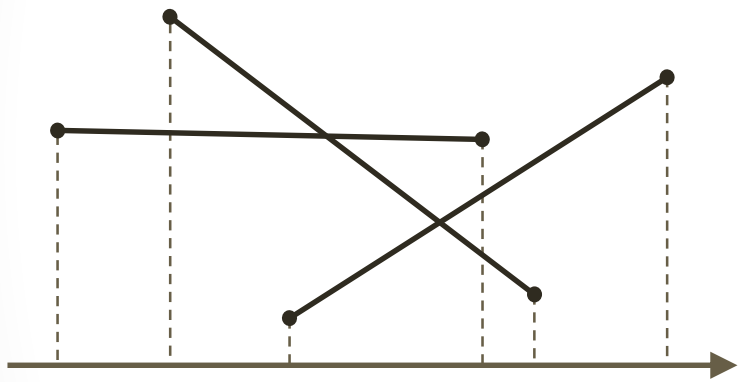
Complexity $\theta(n \log n + k)$

The algorithm:

1. Sort all the endpoints by the x-coordinate.
2. Apply plane sweeping by a vertical line l iterating on the endpoints from left to right:
   a. If a **left endpoint** of a horizontal line is detected, then insert this line to the *dictionary* $\mathcal{L}$ regarding its y-coordinate: $O(\log n)$.
   b. If a **right endpoint** is detected, then remove this segment from $\mathcal{L}$: $O(\log n)$.
   c. If a vertical segment is detected, then select from $\mathcal{L}$ the horizontal segments whose y-coordinates are covered by the y-interval specified by the vertical segment: $O(\log n)$ for each vertical segment, $O(1)$ for each intersection point ($O(k)$ in total).

Intersection of Arbitrary Segments

Problem. *Given n segments on the plane, find all their intersections.*



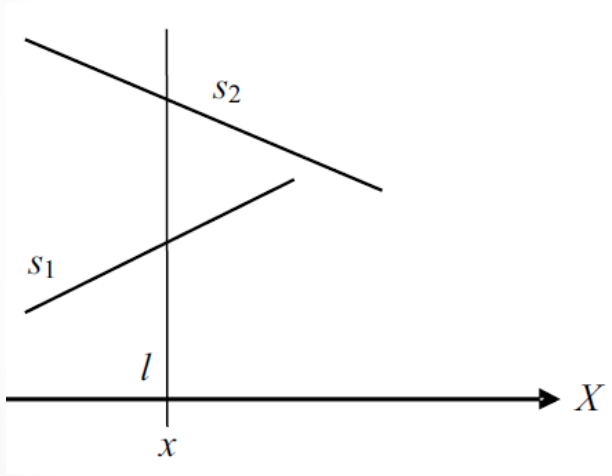
The algorithm:

1. Sort all the endpoints by the x-coordinate and put them to the *priority queue* $\mathcal{E}$.
2. Apply plane sweeping by a vertical line l:
   a. If a **left endpoint** is detected, insert the segment to the *dictionary* $\mathcal{L}$ regarding the y-coordinate of its left endpoint.
   b. If a **right endpoint** – remove the segment from $\mathcal{L}$.

If some segments intersect, then, sooner or later, they become neighbors in $\mathcal{L}$ while l passes through them.

- Insert the intersection point in $\mathcal{E}$.
- When l is passing through this point swap the intersecting segments in $\mathcal{L}$.

Intersection of Arbitrary Segments



Dynamic segment comparator

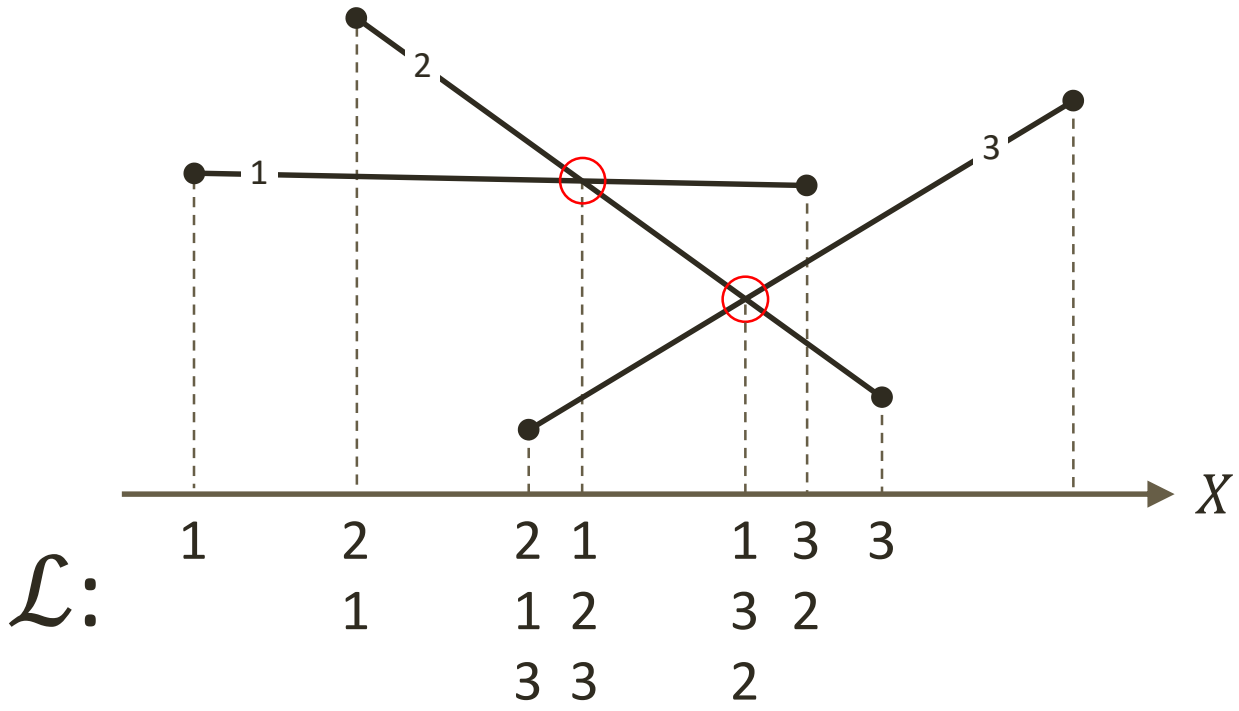
```
double xl; // current x-coordinate of the sweep-line

// Calculates y-coordinate of the intersection point
// of segment seg and the sweep-line
double yseg (int seg, double x) const;

// The dynamic comparator
struct compseg
{
    bool operator()(int s0, int s1) const
    {
        return (yseg(s0, xl) < yseg(s1, xl));
    }
};

// Dictionary L
std::set<int, compseg> L;
```

Intersection of Arbitrary Segments



The Bentley-Ottmann Algorithm

```
LineIntersection(S, Q)
 1  Отсортировать 2n концов отрезков множества S
    и поместить их в очередь E
 2  while E ≠ ∅ do
 3      p ← Pop(E)
 4      if p – левый конец отрезка s then
 5          Insert(L, s)
 6          s1 ← Above(s, L)
 7          s2 ← Below(s, L)
 8          if s1 пересекает s справа от p then
 9              Push(A, (s1, s))
10          if s2 пересекает s справа от p
11              Push(A, (s2, s))
12      else if p – правый конец отрезка s then
13          s1 ← Above(s, L)
14          s2 ← Below(s, L)
15          if s1 пересекает s2 справа от p
16              Push(A, (s1, s2))
17          Delete(L, s)
18      else          ▷ p – точка пересечения
19          Пусть s1 и s2 – отрезки, пересекающиеся в p,
            причем s1 = Above(s2, L) слева от p
20          s3 ← Above(s1, L)
21          s4 ← Below(s2, L)
22          if s3 пересекает s2 справа от p
23              Push(A, (s3, s2))
24          if s4 пересекает s1 справа от p
25              Push(A, (s4, s1))
26          Поменять s1 и s2 местами в L
27      while A ≠ ∅ do
28          (s1, s2) ← Pop(A)
29          Пусть p – точка пересечения отрезков s1 и s2
30          if p ∉ E then
31              Push(Q, (s1, s2))
32              Insert(E, p)
```

Complexity $O\big((n+k)\log n\big)$.

Detection of Intersection

```
DetectLineIntersection(S)
 1  Отсортировать 2n концов отрезков множества S
    и поместить их в вектор M
 2  for i ← 1 to 2n do
 3      p ← M[i]
 4      if p – левый конец отрезка s then
 5          Insert(L, s)
 6          s1 ← Above(s, L)
 7          s2 ← Below(s, L)
 8          if s1 пересекает s then
 9              return TRUE
10          if s2 пересекает s then
11              return TRUE
12      else if p – правый конец отрезка s then
13          s1 ← Above(s, L)
14          s2 ← Below(s, L)
15          if s1 пересекает s2 then
16              return TRUE
17          Delete(L, s)
18  return FALSE
```

Complexity $\theta(n \log n)$.

Questions?