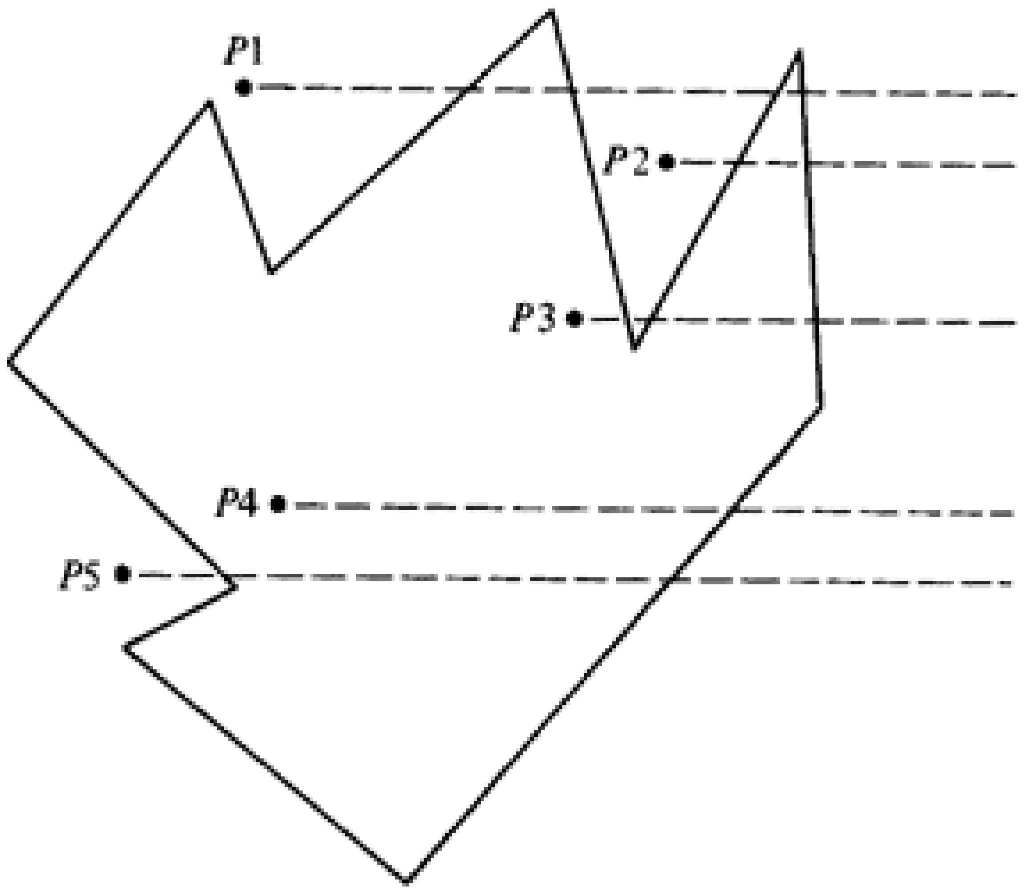
Module – 5
Geometric Algorithm

The Geometry and its features (Points, lines, and Polygons)

- How does the computer determine whether a particular candidate point in the rectangle is inside or outside
- the polygon of interest?
  - An effective way to deal with this problem is the point-polygon method.

Point-Polygon Method



Step-1: Rays from P1, P2 and P5 intersect the polygon an even number of times
Step-2: P1, P2, and P5 are not within the polygon.
Step-3: Rays from P3 and P4 intersect the polygon an odd number of times
Step-4: P3 and P4 are within the polygon,

Point Polygon Method

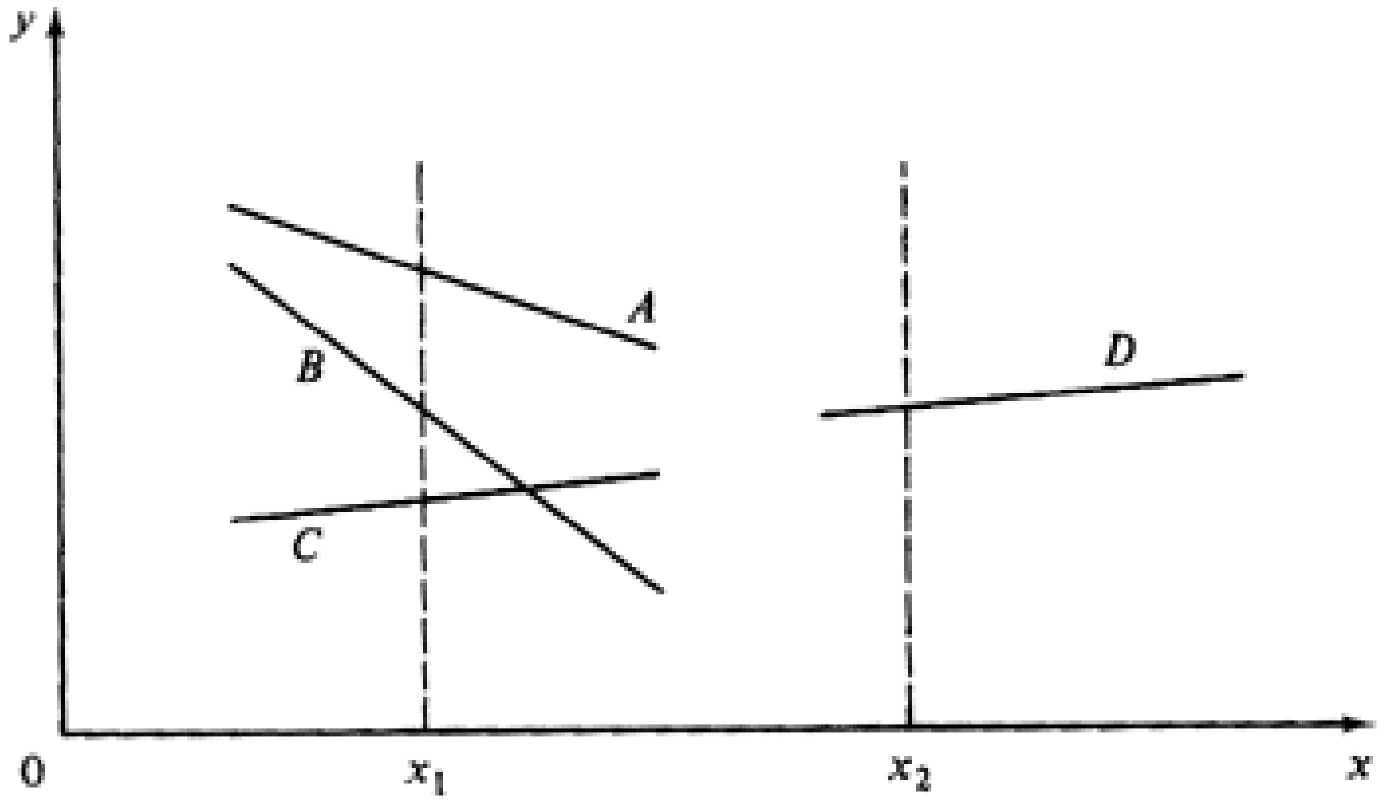
- ***"Given a point (x, y) and a polygon specified by the n clockwise ordered vertices (x1, y1), (x2, y2), . . . , (xn, yn), is the point (x, y) contained within the polygon?"***
- The basic idea of the method is to extend a ray in any direction from the point in question; if the ray intersects the sides of the polygon an odd (even) number of times, the point is (is not) within the polygon.
- The method "works" because if the point is outside the polygon,
  - the ray will exit the polygon each time that it enters it, since it must end up outside the polygon;
  - the number of exits equaling the number of entries implies that the total number of intersections must be an even number.
  - If, on the other hand, the point is within the polygon,
  - then the number of exits e must be one greater than the number of entrances (e - 1),
  - because the initial exit is not paired with an entrance;
  - the total number of intersections 2e - I is clearly an odd number.

Intersection of line segment

- ***Suppose now that we are given n straight-line segments on a plane with each segment specified by its two end points.***
- An interesting question then is: ***Do any of these line segments intersect?***
- The straightforward approach requires O***(n^2) time:***
  - there are ***$n(n - 1)/2$*** possible pairs of line segments which must be checked for intersections, and checking for the intersection of any given pair of line segments requires constant time.
  - Once again, however, the most straightforward approach is not the most efficient one.
- An algorithm that requires ***O(n log2 n)*** time to answer the problem:
  - "Is there at least one intersection?
  - " In the x-y coordinate system we shall call the left (right) end point of a given line segment that end point of the segment with the smaller (larger) value of the x-coordinate.
  - Suppose now that we begin "sweeping" a vertical line along the x-axis.
  - Two line segments S and T are then said to be comparable at $\boldsymbol{x = x_1}$ if they are both intersected by the vertical line at $\boldsymbol{x = x_1}$ The segments are consecutive if S is immediately above or immediately below T;
  - in the former case, ***S is above (T, x1),*** in the latter ***S is below (T, x1)***.

Note: No line segment will be above and/or below a line segment T for some or all values of x.

Relation between line segments



A is above B (The co-ordinate (B,x_1)
C is below B (The co-ordinate (C,x_1)
D is above the point x_2

Algorithm: Sweep-line

STEP 1: Sort the x-coordinates of the 2n end points of the line segments from left to right.

Begin processing the 2n end points at the leftmost end point.

STEP 2: Let xi denote the x-coordinate of the end point currently being processed and X the line segment corresponding to that end point.

1. If this is the left end point of X:

a. Add X to the sorted list of current line segments, at its appropriate place according to the y-coordinate of the end point.

b. If Y = above (X, xi), check to determine if Y and X intersect. If they do, stop and return " Y and X."

If they do not, continue. c. If Z = below (X, xi),

check to determine if Z and X intersect.

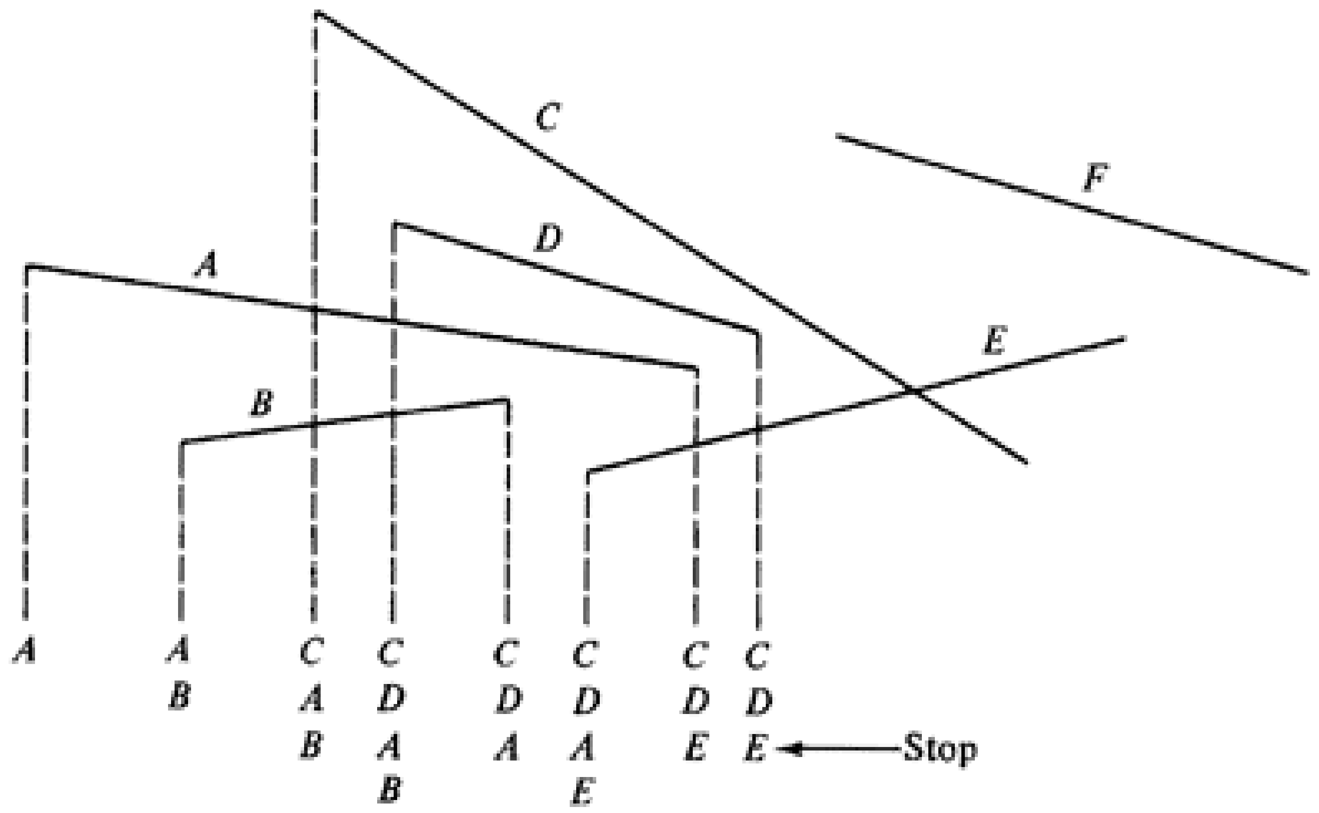
If they do, stop and return "Z and X." If they do not, go to Step 3.

2. If this is the right end point of X: If Y = above (X, xi) and Z = below (X, xi), check whether Y and Z intersect.

If they do, stop and return " Y and Z." If they do not, delete X from the list of current line segments.

STEP 3: If the last end point processed was the rightmost end point, stop and return "no intersection." If not, go to the next end point (to the right) and return to Step 2.

Algorithm: Sweep – Line

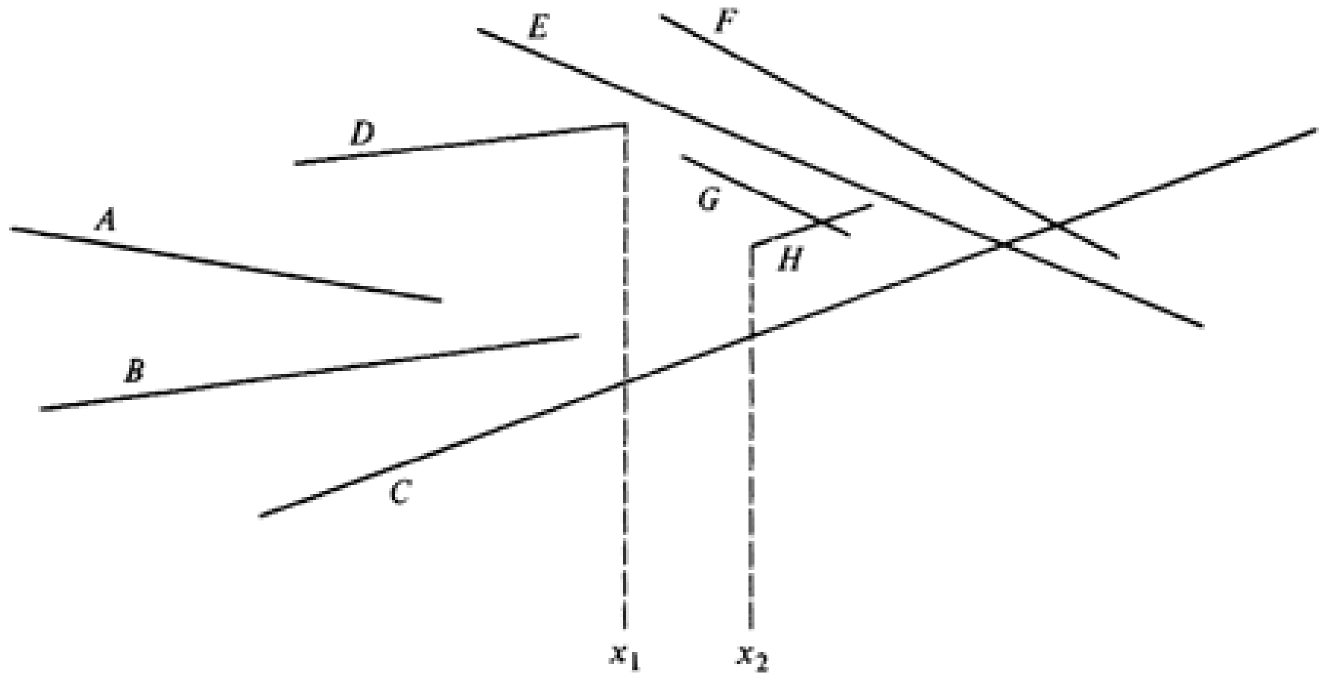


The sorted list of line segments at all end points.
The algorithm stops at the point indicated, since it discovers there the intersection of C with E.

Computational Complexity

- **Step 1** requires O(n log n) time for sorting the 2n end points.
  - [It actually requires 0(2n log n), but remember that in the theory of computational complexity 0(2n) is equivalent to O(n).]
  - The addition (or deletion) of line segments to (or from) the sorted list of current line segments
- **Step 2** requires O(log n) time, since the sorted list can contain at most n line segments at any time.
  - Checking whether two line segments intersect requires constant time (i.e., is independent of n).
  - Since in the worse case, when there is no intersection,
  - Step 2 must be repeated 2n times,
  - Step 2 also consumes a total time which is O(n log n).
- **Step 3** is trivial and will be repeated 2n times in the worst case,
- **The Algorithm is O(n log n).**

The intersection points



1. Detect first the intersection of C with E at the point X1.
2. The intersection of C with E is not the leftmost one.
3. If the algorithm is allowed to proceed, it will detect at x2, the intersection of G with H.
4. The algorithm cannot detect intersection of F with C (even if allowed to proceed)

Applications

1. Given the coordinates of an incident requiring dispatching of a service unit, in which geographical or administrative subdivision did this incident occur?
2. Which of an agency's available service units is the closest one
   (in terms of travel distance or travel time) to a request for assistance?
3. Which ambulance zones have areas in common with a particular police precinct(the area within the walls or perceived boundaries of particular building or place)?
4. Which city blocks lie closest to each one of a new set of voting centers,
   so that voters can be assigned to the voting precinct most
   convenient to them?
5. Which ZIP-code areas in a city contain segments of a particular highway or railroad?
6. Given that a refuse incinerator is to be installed at a particular location
   and that it has a certain geometrically described pattern of smoke dispersal, which voting districts will be affected by the pollutants?