Algorithms & Adv. Data Structures

**Lab 3: Iterative Algorithm Design**

Alhagie Boye Sukhbir Singh Vamsi Sudersanam

**Introduction**

In this lab, we will tackle the computational geometry problem of determining whether a given point lies inside a polygon. This problem has practical applications in various fields such as computer graphics, geographic information systems (GIS), and robotics

**1. Problem:** *Determine if a Point is Located Inside a Polygon*

**Input:**

* A sequence <p1, p2, …, pn> of n ≥ 3 2D points. Each point is a pair of x and y coordinates. The points correspond to the vertices of a simple (non-intersecting) polygon. The polygon is connected by line segments between each adjacent pair of points, including a line segment from the last point to the first point.
* The x and y coordinates for a single point distinct from the vertex points.

**Output:** A Boolean value indicating whether the point is located inside the polygon.

**A screenshot of a diagram

Description automatically generated2. Decision Rule and Illustrations**

To determine whether a point (X,Y) lies inside a polygon, we employ the Ray Casting algorithm (also known as the Even-Odd Rule). The decision process follows these criteria (**See figure 1**):

a. Cast a horizontal ray from the test point P to infinity (or a sufficiently large x-coordinate)

b. Count the number of intersections (N) between this ray and the polygon's edges

c. Apply the following decision rule:

* If **N is odd or point lies exactly on a polygon edge** → Point lies **INSIDE** the polygon.
* If **N is even** → Point lies **OUTSIDE** the polygon

Fig 1: Ray Casting Algorithm Diagram

**Proof of The Ray Casting Algorithm**

An intuitive explanation of why it works is that every time we cross a border, we change "country" (inside-outside, or outside-inside), but the last "country" we land on is surely *outside* (since the inside of the polygon is finite, while the ray continues towards infinity) . So, if we crossed an odd number of borders we were surely inside, otherwise we were outside.

we can follow the ray backward to see it better: starting from outside, only an odd number of crossing can give an *inside*: outside-inside, outside-inside-outside-inside, and so on (the - represents the crossing of a border).

**High-Level Pseudocode - vamsi**

**Justification of Correctness for Ray-Casting Algorithm – Sukhbir**

1. How it works:

* The algorithm casts an imaginary ray from the point in question and counts how many times it crosses the polygon’s edges.
  + Odd number of crossings means the point is inside the polygon.
  + Even number of crossings means the point is outside the polygon.

1. Why it works:

* Crossing edges: Each time the ray crosses an edge of the polygon, the point moves from inside to outside or vice versa. If it crosses an odd number of times, the point started outside and is now inside. If even, it either never entered or crossed back out.

1. Convex Vs Concave Polygons:

* Convex Polygons:

**Algorithm Implementation - vamsi**

**Worst-Case Run Time Analysis - alhagie**

**Test Cases and Expected vs. Actual Results - vamsi**

**Benchmarking Results and Analysis - alhagie**

**Appendix: Source Code and Test Cases**