CS603: Geometric Algorithms

Course Project Report

Anshika Raman	Roll No: 210050014
Kanad Shende	Roll No: 210050078

April 20, 2025

Submitted as part of the CS603 Course Project

Department of Computer Science and Engineering
IIT Bombay

Question 1: Geometric Median in $\mathbb{R}^2$

Objective

Given a finite set of points $S = \{p_1, p_2, \ldots, p_n\} \subset \mathbb{R}^2$, we aim to find a point $q \in \mathbb{R}^2$ that minimizes the sum of Euclidean distances from q to each p_i. That is, find

$$q^* = \arg\min_{q \in \mathbb{R}^2} \sum_{i=1}^{n} \|q - p_i\|.$$

This point is known as the **geometric median**.

Algorithm Description

The geometric median has no closed-form solution for $n > 2$, but can be computed using **Weiszfeld's algorithm**, an iterative method defined as follows.

Initialization

Let the initial guess $q^{(0)}$ be the centroid:

$$q^{(0)} = \frac{1}{n} \sum_{i=1}^{n} p_i.$$

Weiszfeld Update Rule

For each iteration $t \geq 0$, we update:

$$q^{(t+1)} = \frac{\sum_{i=1}^{n} \frac{p_i}{\|q^{(t)} - p_i\|}}{\sum_{i=1}^{n} \frac{1}{\|q^{(t)} - p_i\|}}, \quad \text{if } \|q^{(t)} - p_i\| > \varepsilon \text{ for all } i,$$

where $\varepsilon > 0$ is a small threshold to avoid division by zero.

Stopping Criterion

The iteration terminates when:

$$\|q^{(t+1)} - q^{(t)}\| < \varepsilon \quad \text{or} \quad \exists i \text{ such that } \|q^{(t)} - p_i\| < \varepsilon.$$

In the latter case, the algorithm directly returns p_i as the geometric median.

Algorithm (Pseudocode)

Algorithm 1 Weiszfeld's Algorithm for Geometric Median

Input: Set of points $S = \{p_1, p_2, \ldots, p_n\} \subset \mathbb{R}^2$
Output: Geometric median q^*
Initialize $q \leftarrow \frac{1}{n} \sum_{i=1}^{n} p_i$
for $k = 1$ to max_iter **do**
    numerator $\leftarrow (0, 0)$;   denominator $\leftarrow 0$
    for $i = 1$ to n **do**
        $d_i \leftarrow \|q - p_i\|$
        if $d_i < \varepsilon$ **then**
            return p_i
        end if
        weight $\leftarrow 1/d_i$
        numerator $\leftarrow$ numerator $+$weight $\cdot\, p_i$
        denominator $\leftarrow$ denominator $+$weight
    end for
    $q_{\text{new}} \leftarrow$ numerator / denominator
    if $\|q_{\text{new}} - q\| < \varepsilon$ **then**
        return q_{new}
    end if
    $q \leftarrow q_{\text{new}}$
end for
return q

Cases Handled

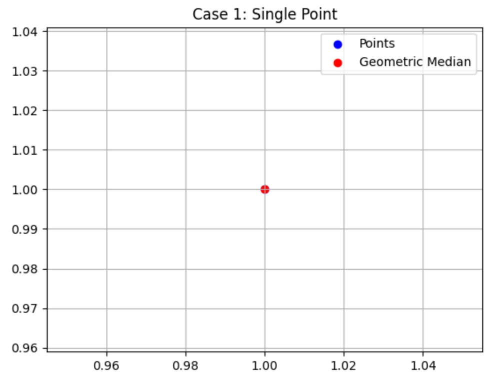
- **Single Point:** The input point itself is returned as the median.
- **All Points Identical:** Same behavior as single-point case.
- **Collinear Points:** The algorithm naturally handles such configurations.
- **Points Too Close:** If any p_i is too close to q, it is returned directly to avoid instability.

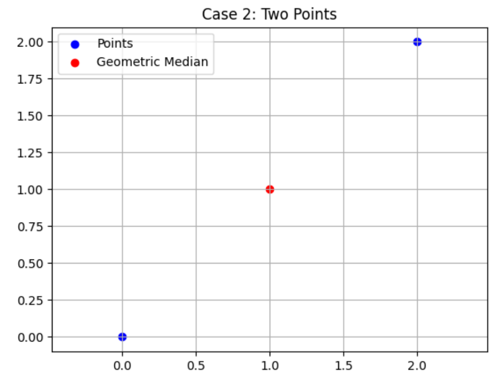
Test Cases Evaluated

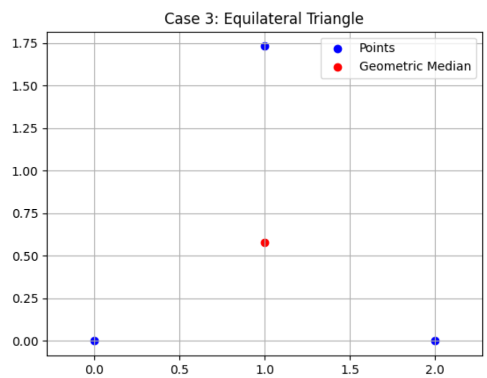
We used the following parameter values for all experiments:

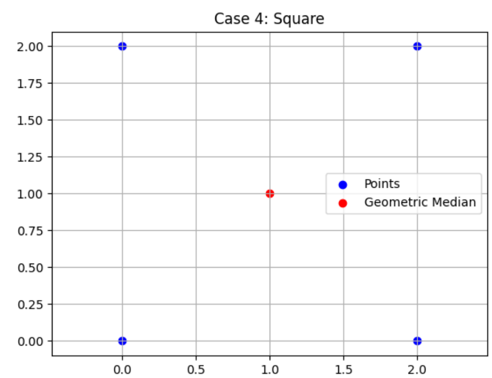
$$\varepsilon = 10^{-7}, \quad \text{max\_iter} = 1000$$

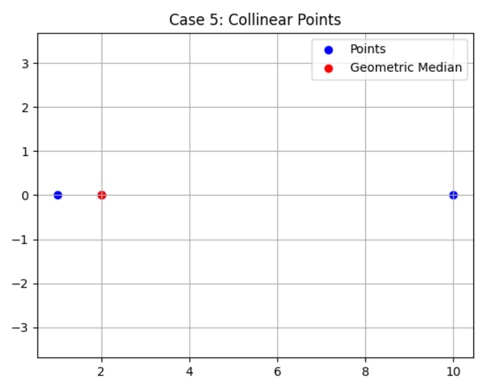
The algorithm was evaluated on the following scenarios:

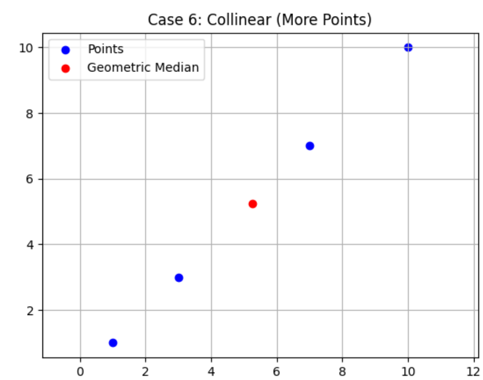

Case 1: Single Point
Points
Geometric Median
1.04
1.03
1.02
1.01
1.00
0.99
0.98
0.97
0.96
0.96
0.98
1.00
1.02
1.04

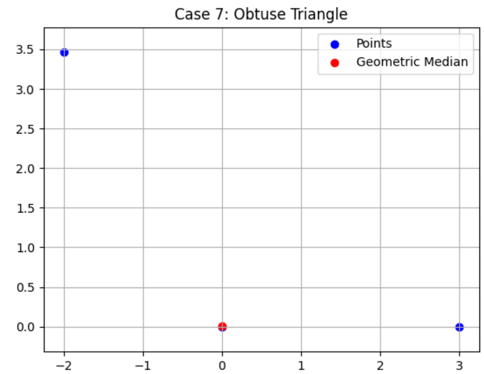

Case 2: Two Points
Points
Geometric Median
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.5
1.0
1.5
2.0

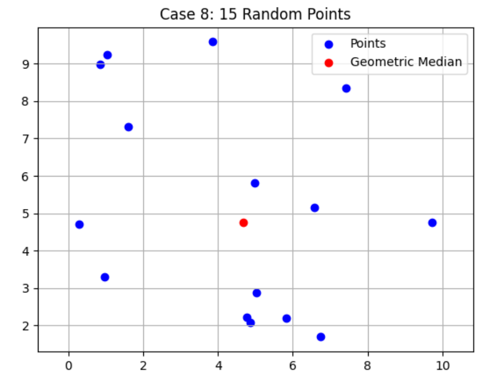

Case 3: Equilateral Triangle
Points
Geometric Median
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.5
1.0
1.5
2.0

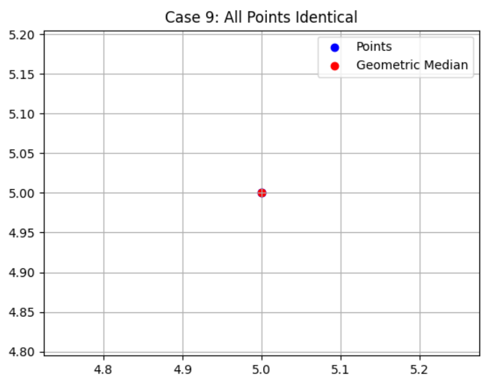

Case 4: Square
Points
Geometric Median
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.5
1.0
1.5
2.0

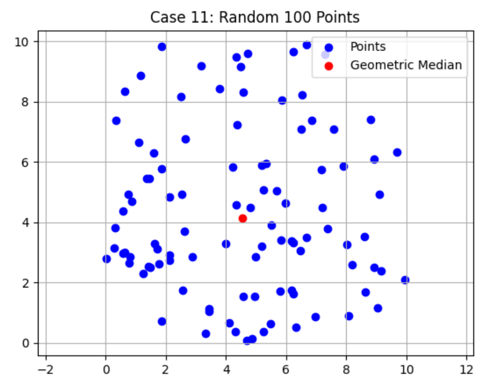

Case 5: Collinear Points
Points
Geometric Median
3
2
1
0
−1
−2
−3
2
4
6
8
10


Case 6: Collinear (More Points)
Points
Geometric Median
10
8
6
4
2
0
2
4
6
8
10
12









Question 2: Minimum Enclosing Disk in $\mathbb{R}^2$

Objective

Given a finite set of points $S = \{p_1, p_2, \ldots, p_n\} \subset \mathbb{R}^2$, the goal is to find the smallest disk $D = (O, r)$—with center O and radius r—such that every point in S lies within or on the boundary of D, i.e.,

$$\|p_i - O\| \leq r \quad \text{for all } i.$$

This problem is central in computational geometry and has applications in clustering, collision detection, and bounding region estimation.

Algorithm Description

We adopt Welzl's randomized algorithm, a powerful recursive method that computes the minimum enclosing circle in expected linear time. The algorithm proceeds incrementally, randomly ordering the points and maintaining a small set R (of at most 3 points) that define the boundary of the current minimal enclosing disk.

The recursive function `welzl`(P, R, n) takes:

- a set P of n points (from which we build the solution),
- a set R of points that must lie on the boundary of the final circle.

The recursion obeys the following structure:

1. If $n = 0$ or $|R| = 3$, compute the trivial circle through points in R.
2. Else, remove a point p from P, compute the minimum enclosing circle D of the remaining points.
3. If p lies within D, return D. Otherwise, include p in R and recurse.

A unique feature of this algorithm is that the recursion depth is implicitly limited since no more than 3 points are ever added to R—a circle is uniquely determined by three points in general position.

Trivial Circle Construction

The function to compute the trivial enclosing circle handles cases based on the number of points in R:

- For 0 or 1 point, the circle is trivial (zero radius).
- For 2 points, the circle is centered at the midpoint with radius half the distance.
- For 3 points, we compute the *circumcircle* of the triangle they form. This involves solving for the intersection of perpendicular bisectors of any two sides.

The circumcenter (O_x, O_y) for points A, B, C is computed using:

$$D = 2 \cdot (x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)),$$

$$O_x = \frac{(x_1^2 + y_1^2)(y_2 - y_3) + (x_2^2 + y_2^2)(y_3 - y_1) + (x_3^2 + y_3^2)(y_1 - y_2)}{D},$$

$$O_y = \frac{(x_1^2 + y_1^2)(x_3 - x_2) + (x_2^2 + y_2^2)(x_1 - x_3) + (x_3^2 + y_3^2)(x_2 - x_1)}{D}.$$

The radius r is then $\|O - A\|$.

To ensure robustness, the implementation checks if the determinant D is close to zero (i.e., degenerate or collinear input), and falls back on simpler constructions using 2-point circles when necessary.

Algorithm (Pseudocode)

Algorithm 2 Welzl's Minimum Enclosing Circle

```
function Welzl(P, R, n)
    if n = 0 or |R| = 3 then
        return TrivialCircle(R)
    end if
    p ← P[n − 1]
    D ← Welzl(P, R, n − 1)
    if p inside D then
        return D
    else
        return Welzl(P, R ∪ {p}, n − 1)
    end if
end function
```

Before the recursion begins, the point set P is randomly shuffled. This ensures the expected linear performance by avoiding worst-case insertion sequences. The main entry point is:

$$\texttt{minimum\_enclosing\_circle}(P) = \texttt{Welzl}(P_{\text{shuffled}}, \emptyset, n)$$

Performance and Complexity

Welzl's algorithm achieves an expected time complexity of $\mathcal{O}(n)$ due to the randomized incremental construction. On average, the number of recursive calls that add to R is bounded by a constant because:

- The probability of a point lying on the boundary decreases as more points are processed.
- Only those violating the current disk are added to R.

Since each valid enclosing circle depends on at most 3 boundary points and the recursion depth grows only when necessary, the algorithm maintains near-linear behavior across all practical inputs.

The implementation also uses geometric predicates with an ε threshold to handle floating-point precision errors, ensuring robustness in edge cases such as collinearity or duplicated points.

Test Cases Evaluated

We used the following parameter values for all experiments:

$$\varepsilon = 10^{-7}$$

The algorithm was evaluated on the following scenarios:

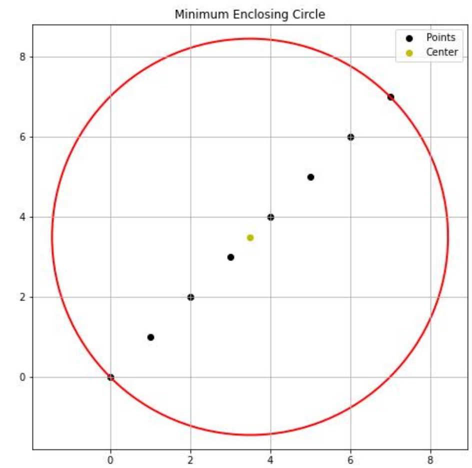


Figure 1: Sample input file 1

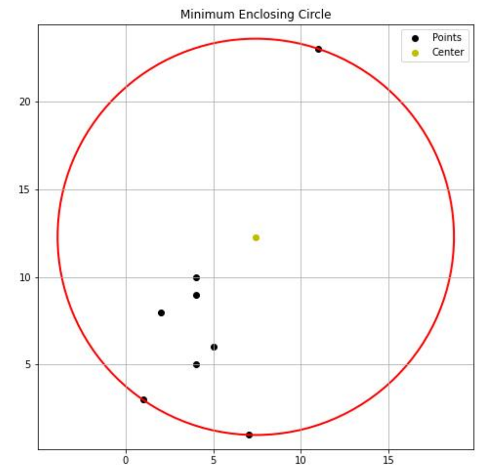


Figure 2: Sample input from file 2

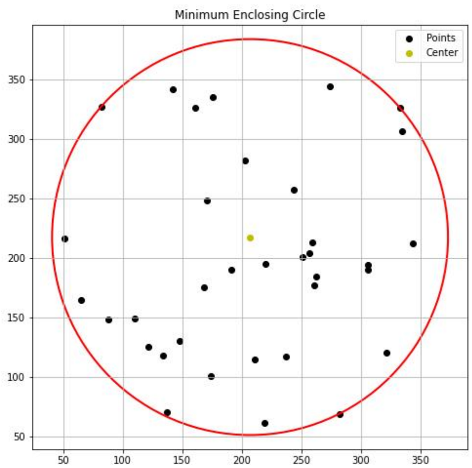


Figure 3: Random 35 points

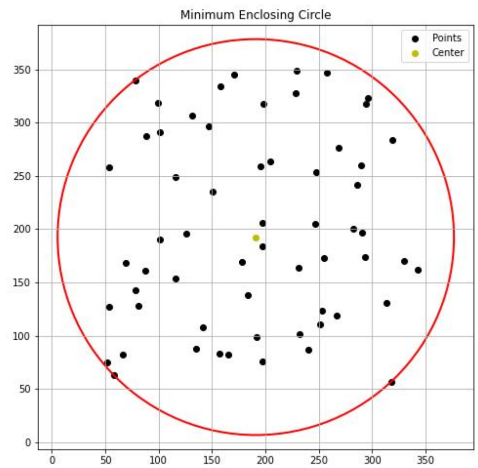


Figure 4: Random 60 points

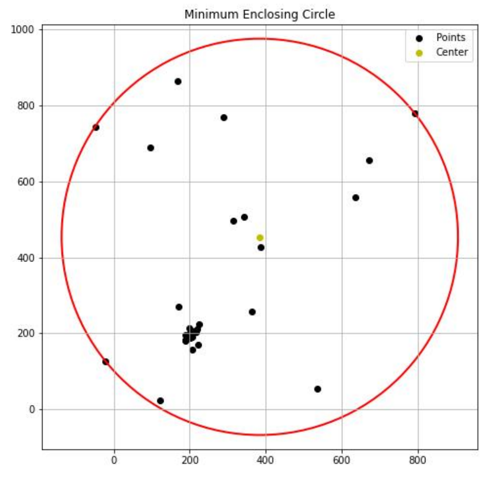


Figure 5: 30 points with uneven distribution

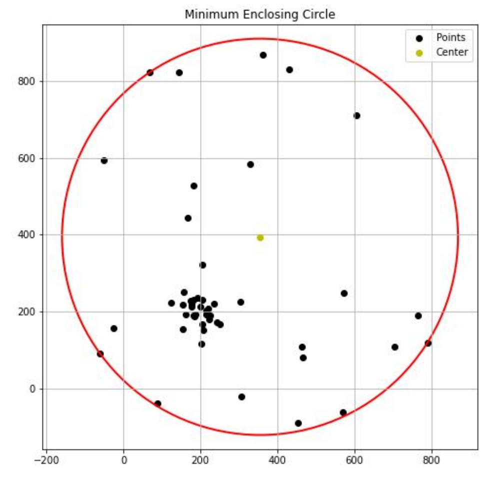


Figure 6: 50 points with uneven distribution