Exp-2.10

Title:

Closest Pair of Points and Convex Hull Using Brute Force

Aim:

To implement brute force algorithms to find the closest pair of points and compute the convex hull for a given set of 2D points, including handling collinear points on hull edges.

Procedure:

1. Read the list of points.
2. Define a function to calculate Euclidean distance between two points.
3. Iterate over all pairs of points to find the pair with the minimum distance.
4. Return and print the closest pair and their distance.

Algorithm:

Closest Pair:

1. Initialize minimum distance as infinity and closest pair as empty.
2. For each pair (i, j) in points:
    - Calculate distance.
    - Update minimum and closest pair if smaller distance found.
3. Return the closest pair and minimum distance.

Convex Hull:

1. For each pair of points (p1, p2):
    - For every other point, determine orientation with line (p1, p2).
    - Track if points lie on both sides of the line.
    - If points lie only on one side or are collinear, the segment forms part of the hull.
2. Collect all such boundary points including collinear ones.
3. Sort hull points in counterclockwise order for output.

Input:

4

1 2

4 5

7 8

3 1

Output:

Closest pair: (1, 2) - (3, 1)

Minimum distance: 1.4142135623730951

Program:

```
import math

def distance(p1, p2):
    return math.sqrt((p1[0] - p2[0])**2 + (p1[1] - p2[1])**2)

def orientation(p, q, r):
    return (q[0] - p[0]) * (r[1] - q[1]) - (q[1] - p[1]) * (r[0] - q[0])

def closestPair(points):
    min_dist = float('inf')
    closest_pair = (None, None)
    n = len(points)
    for i in range(n):
        for j in range(i + 1, n):
            d = distance(points[i], points[j])
            if d < min_dist:
```

```
                min_dist = d
                closest_pair = (points[i], points[j])
    return closest_pair, min_dist
def convexHull(points):
    n = len(points)
    hull_points = set()

    for i in range(n):
        for j in range(i + 1, n):
            p1, p2 = points[i], points[j]
            left_side = right_side = False

            for k in range(n):
                if k == i or k == j:
                    continue
                o = orientation(p1, p2, points[k])
                if o > 0:
                    left_side = True
                elif o < 0:
                    right_side = True
                if left_side and right_side:
                    break

            if not (left_side and right_side):
                hull_points.add(p1)
                hull_points.add(p2)
```

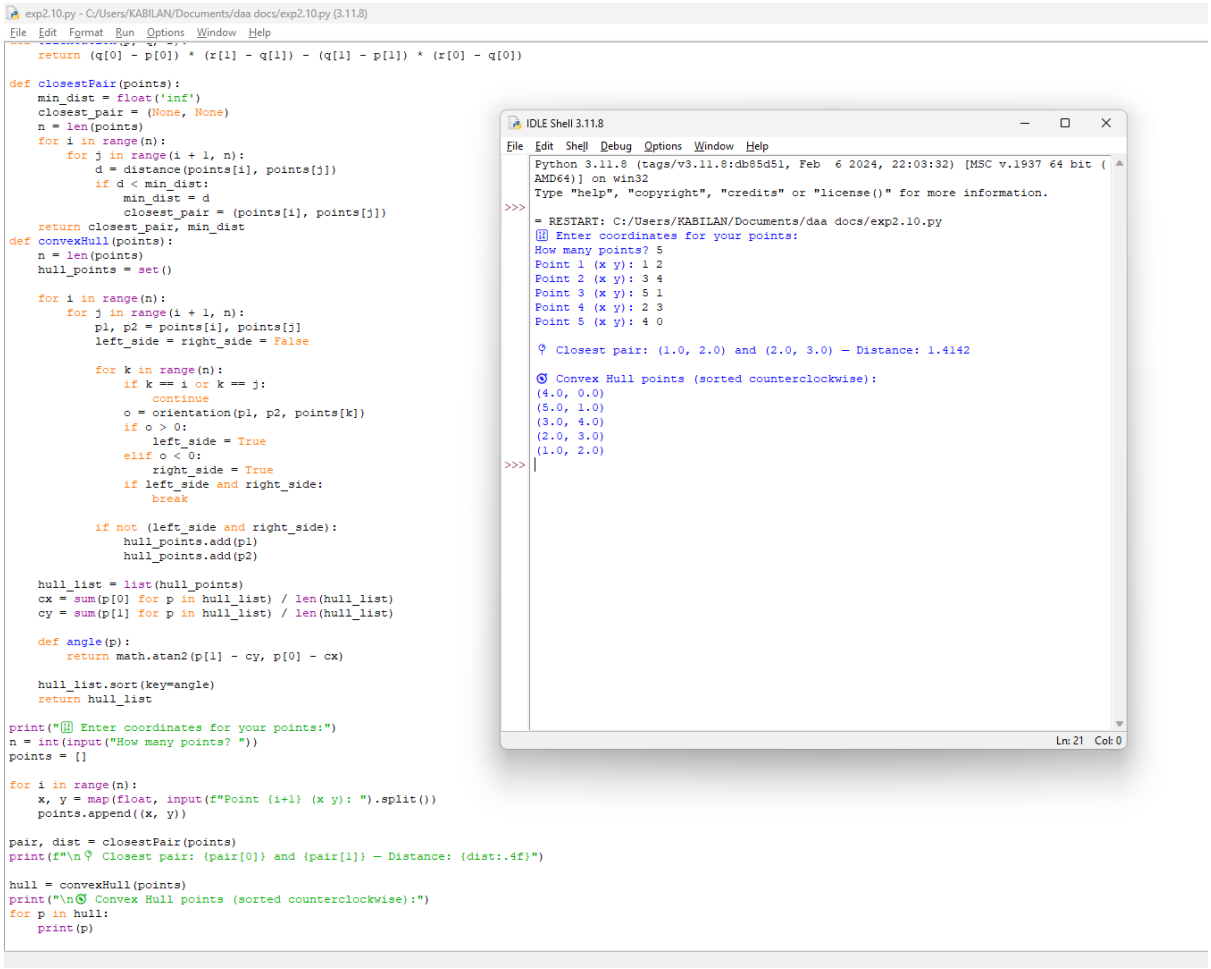
```
    hull_list = list(hull_points)
    cx = sum(p[0] for p in hull_list) / len(hull_list)
    cy = sum(p[1] for p in hull_list) / len(hull_list)

    def angle(p):
        return math.atan2(p[1] - cy, p[0] - cx)

    hull_list.sort(key=angle)
    return hull_list

print("🔢 Enter coordinates for your points:")
n = int(input("How many points? "))
points = []

for i in range(n):
    x, y = map(float, input(f"Point {i+1} (x y): ").split())
    points.append((x, y))

pair, dist = closestPair(points)
print(f"\n📍 Closest pair: {pair[0]} and {pair[1]} — Distance: {dist:.4f}")

hull = convexHull(points)
print("\n Convex Hull points (sorted counterclockwise):")
for p in hull:
    print(p)
```

Performance Analysis:

Closest pair brute force is $O(n^2)$

Convex hull brute force is $O(n^3)$

Program Output:



```
exp2.10.py - C:/Users/KABILAN/Documents/daa docs/exp2.10.py (3.11.8)
File  Edit  Format  Run  Options  Window  Help
    return (q[0] - p[0]) * (r[1] - q[1]) - (q[1] - p[1]) * (r[0] - q[0])

def closestPair(points):
    min_dist = float('inf')
    closest_pair = (None, None)
    n = len(points)
    for i in range(n):
        for j in range(i + 1, n):
            d = distance(points[i], points[j])
            if d < min_dist:
                min_dist = d
                closest_pair = (points[i], points[j])
    return closest_pair, min_dist
def convexHull(points):
    n = len(points)
    hull_points = set()

    for i in range(n):
        for j in range(i + 1, n):
            p1, p2 = points[i], points[j]
            left_side = right_side = False

            for k in range(n):
                if k == i or k == j:
                    continue
                o = orientation(p1, p2, points[k])
                if o > 0:
                    left_side = True
                elif o < 0:
                    right_side = True
                if left_side and right_side:
                    break

            if not (left_side and right_side):
                hull_points.add(p1)
                hull_points.add(p2)

    hull_list = list(hull_points)
    cx = sum(p[0] for p in hull_list) / len(hull_list)
    cy = sum(p[1] for p in hull_list) / len(hull_list)

    def angle(p):
        return math.atan2(p[1] - cy, p[0] - cx)

    hull_list.sort(key=angle)
    return hull_list

print("🅷 Enter coordinates for your points:")
n = int(input("How many points? "))
points = []

for i in range(n):
    x, y = map(float, input(f"Point {i+1} (x y): ").split())
    points.append((x, y))

pair, dist = closestPair(points)
print(f"\n📍 Closest pair: {pair[0]} and {pair[1]} — Distance: {dist:.4f}")

hull = convexHull(points)
print("\n🔄 Convex Hull points (sorted counterclockwise):")
for p in hull:
    print(p)
```

```
IDLE Shell 3.11.8
File  Edit  Shell  Debug  Options  Window  Help
Python 3.11.8 (tags/v3.11.8:db85d51, Feb  6 2024, 22:03:32) [MSC v.1937 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/KABILAN/Documents/daa docs/exp2.10.py
🅷 Enter coordinates for your points:
How many points? 5
Point 1 (x y): 1 2
Point 2 (x y): 3 4
Point 3 (x y): 5 1
Point 4 (x y): 2 3
Point 5 (x y): 4 0

📍 Closest pair: (1.0, 2.0) and (2.0, 3.0) — Distance: 1.4142

🔄 Convex Hull points (sorted counterclockwise):
(4.0, 0.0)
(5.0, 1.0)
(3.0, 4.0)
(2.0, 3.0)
(1.0, 2.0)
>>>
Ln: 21  Col: 0
```

Result:

Both closest pair and convex hull algorithms executed successfully for the sample input.