Exp-2.11

Title:

Convex Hull using Brute Force Approach

Aim:

To implement a brute force algorithm to find the convex hull of a set of 2D points. The convex hull points are returned in counter-clockwise order.

Procedure:

1. Read the input list of 2D points.
2. Define an orientation function to determine the relative position of points.
3. For each pair of points (p1, p2), check if all other points lie on one side of the line or are collinear.
4. If yes, include both points in the hull set (including collinear points on hull edges).
5. Sort the hull points in counter-clockwise order around their centroid.
6. Print the hull points.

Algorithm:

Closest Pair:

1. For each pair of points (p1, p2):
    - Initialize flags for points on left or right side of line.
    - For each other point p:
        - Determine orientation wrt line (p1, p2).
        - Update left or right flags accordingly.
        - If points found on both sides, discard this line.
2. If not discarded, points (p1, p2) are part of hull.
3. After collecting hull points, compute centroid.
4. Sort hull points by angle relative to centroid to get CCW order.
5. Return sorted hull points.

Input:

A list of points: [(1,1), (4,6), (8,1), (0,0), (3,3)]

Output:

Convex hull points in counter clockwise order:
[(0, 0), (1, 1), (8, 1), (4, 6)]

Program:

```
import math
def orientation(p, q, r):
    return (q[0] - p[0]) * (r[1] - q[1]) - (q[1] - p[1]) * (r[0] - q[0])
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_from_centroid(point):
        return math.atan2(point[1] - cy, point[0] - cx)

    hull_list.sort(key=angle_from_centroid)
    return hull_list

n = int(input("Enter number of points: "))
points = []
for i in range(n):
    x, y = map(float, input(f"Enter x y for point {i + 1}, separated by space:
").split())
    points.append((x, y))

hull = convexHull(points)
print("Convex Hull:", hull)
```

Performance Analysis:

Time Complexity: $O(n^3)$

Space Complexity: O(n)

Program Output:

exp2.11.py - C:/Users/KABILAN/Documents/daa docs/exp2.11.py (3.11.8)

File Edit Format Run Options Window Help

```
import math

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

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_from_centroid(point):
        return math.atan2(point[1] - cy, point[0] - cx)

    hull_list.sort(key=angle_from_centroid)
    return hull_list

n = int(input("Enter number of points: "))
points = []
for i in range(n):
    x, y = map(float, input(f"Enter x y for point {i + 1}, separated by space: ").split())
    points.append((x, y))

hull = convexHull(points)
print("Convex Hull:", hull)
```

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/exp2.11.py
Enter number of points: 4
Enter x y for point 1, separated by space: 1 1
Enter x y for point 2, separated by space: 4 6
Enter x y for point 3, separated by space: 8 1
Enter x y for point 4, separated by space: 0 0
Convex Hull: [(0.0, 0.0), (8.0, 1.0), (4.0, 6.0)]
>>>
```

Ln: 11 Col: 0

Result:

The program computes the convex hull and outputs the points in counter clockwise order as requested.