2.10 FINDING CLOSEST PAIR OF POINTS IN 2D

Question:

Write a program to find the closest pair of points in a given set using the brute force approach. Analyze the time complexity of your implementation. Define a function to calculate the Euclidean distance between two points. Implement a function to find the closest pair of points using the brute force method. Test your program with a sample set of points and verify the correctness of your results. Analyze the time complexity of your implementation. Write a brute-force algorithm to solve the convex hull problem for the following set S of points? P1 (10,0)P2 (11,5)P3 (5, 3)P4 (9, 3.5)P5 (15, 3)P6 (12.5, 7)P7 (6, 6.5)P8 (7.5, 4.5).How do you modify your brute force algorithm to handle multiple points that are lying on the sameline?

AIM

To determine the closest pair of points and the minimum Euclidean distance between them using a brute force approach.

ALGORITHM

1. Start
2. Define a function to calculate the Euclidean distance.
3. Read the list of points.
4. Initialize min_distance = ∞ and closest_pair = None.
5. Compare each pair of points:
6. If the distance between them is smaller than min_distance:
7. Update min_distance and closest_pair.
8. Return closest_pair and min_distance
9. End

PROGRAM

```
def orientation(p, q, r):
    val = (q[1] - p[1]) * (r[0] - q[0]) - \
          (q[0] - p[0]) * (r[1] - q[1])
    return 0 if val == 0 else (1 if val > 0 else 2)

def convex_hull(points):
    n = len(points)
    if n < 3:
        return []

    hull = []
    for i in range(n):
        for j in range(i+1, n):
            left = right = False
            for k in range(n):
                if k == i or k == j:
                    continue
                o = orientation(points[i], points[j], points[k])
                if o == 1:
                    left = True
                elif o == 2:
                    right = True
            if not (left and right):
                if points[i] not in hull:
                    hull.append(points[i])
                if points[j] not in hull:
                    hull.append(points[j])
    return hull

def run_convex_hull():
    raw = input("Enter points as x,y separated by space: ").split()
    points = [tuple(map(float, p.split(','))) for p in raw]
    hull = convex_hull(points)
    print("Convex hull points:")
    for p in hull:
        print(p)
run_convex_hull()
```

Input:

P1 (10,0) P2 (11,5) P3 (5, 3) P4 (9, 3.5) P5 (15, 3) P6 (12.5, 7) P7 (6, 6.5) P8 (7.5, 4.5)

Output:

```
Enter points as x,y separated by space: 10,0 11,5 5,3 9,3.5 15,3 12.5,7 6,6.5 7.5,4.5
Convex hull points:
(10.0, 0.0)
(5.0, 3.0)
(15.0, 3.0)
(6.0, 6.5)
(12.5, 7.0)
>>>
```

RESULT:

Thus the program is successfully executed and the output is verified.

PERFORMANCE ANALYSIS:

· Time Complexity: $O(n^2)$

· Space Complexity: O(1)