2.9 FINDING CLOSEST PAIR OF POINTS IN 2D

Question:

Write a program that finds the closest pair of points in a set of 2D points using the brute force approach.

AIM

To find the closest pair of points and their minimum Euclidean distance from a given set of 2D points using the brute force method.

ALGORITHM

1. Start and Read the list of 2D points.
2. Initialize min_distance as infinity and closest_pair as None.
3. For each point p1 in the list
4. For each other point p2 after p1:
5. Calculate the Euclidean distance between p1 and p2.
6. If this distance is less than min_distance:- Update min_distance with the new distance.
7. Update closest_pair with (p1, p2).
8. Return closest_pair and min_distance.

PROGRAM

```
import math

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

def closest_pair(points):
    min_dist = float('inf')
    pair = ()
    for i in range(len(points)):
        for j in range(i+1, len(points)):
            dist = euclidean(points[i], points[j])
            if dist < min_dist:
                min_dist = dist
                pair = (points[i], points[j])
    return pair, min_dist

def run_closest_pair():
    raw = input("Enter points as x,y separated by space (e.g. 1,2 4,5): ").split()
    points = [tuple(map(float, p.split(','))) for p in raw]
    pair, dist = closest_pair(points)
    print(f"Closest pair: {pair[0]} - {pair[1]} Minimum distance: {dist}")
run_closest_pair()
```

Input:

Points = [(1, 2), (4, 5), (7, 8), (3, 1)]

Output:

```
Enter points as x,y separated by space (e.g. 1,2 4,5): 1,2 4,5 7,8 3,1
Closest pair: (1.0, 2.0) - (3.0, 1.0) Minimum distance: 2.23606797749979
>>>
```

RESULT:

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

PERFORMANCE ANALYSIS:

· Time Complexity: $O(n^2)$

· Space Complexity: O(1)