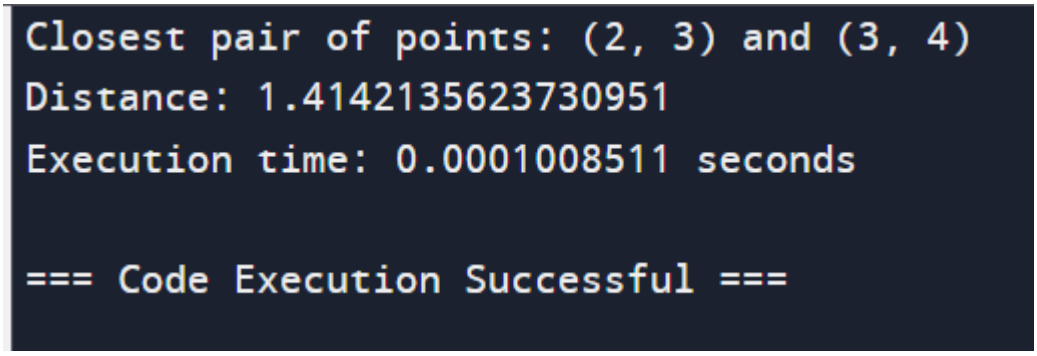
85. Closest pair of points using divide and conquer

PROGRAM:-

```
import math
import time

def distance(point1, point2):
    return math.sqrt((point1[0] - point2[0])**2 + (point1[1] - point2[1])**2)

def closest_pair_dc(points):
    def closest_pair_rec(points_sorted_x, points_sorted_y):
        n = len(points_sorted_x)
        if n <= 3:
            return brute_force_closest_pair(points_sorted_x)

        mid = n // 2
        mid_point = points_sorted_x[mid]

        left_x = points_sorted_x[:mid]
        right_x = points_sorted_x[mid:]

        midpoint = points_sorted_x[mid][0]
        left_y = list(filter(lambda x: x[0] <= midpoint, points_sorted_y))
        right_y = list(filter(lambda x: x[0] > midpoint, points_sorted_y))

        (p1_left, p2_left, dist_left) = closest_pair_rec(left_x, left_y)
        (p1_right, p2_right, dist_right) = closest_pair_rec(right_x, right_y)

        if dist_left < dist_right:
            min_dist = dist_left
            min_pair = (p1_left, p2_left)
        else:
            min_dist = dist_right
            min_pair = (p1_right, p2_right)

        (p3, p4, dist_split) = closest_split_pair(points_sorted_x, points_sorted_y, min_dist, min_pair)

        if min_dist <= dist_split:
            return min_pair[0], min_pair[1], min_dist
        else:
            return p3, p4, dist_split

    def brute_force_closest_pair(points):
        min_dist = float('inf')
        p1, p2 = 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
```

```
                p1, p2 = points[i], points[j]
        return p1, p2, min_dist

    def closest_split_pair(points_sorted_x, points_sorted_y, delta, best_pair):
        n = len(points_sorted_x)
        mid_x = points_sorted_x[n // 2][0]

        sy = [p for p in points_sorted_y if mid_x - delta <= p[0] <= mid_x + delta]

        best = delta
        ln_sy = len(sy)
        for i in range(ln_sy - 1):
            for j in range(i + 1, min(i + 7, ln_sy)):
                p, q = sy[i], sy[j]
                dst = distance(p, q)
                if dst < best:
                    best_pair = (p, q)
                    best = dst
        return best_pair[0], best_pair[1], best

    points_sorted_x = sorted(points, key=lambda x: x[0])
    points_sorted_y = sorted(points, key=lambda x: x[1])
    return closest_pair_rec(points_sorted_x, points_sorted_y)

def find_closest_pair_time(points):
    start_time = time.time()  # Start time measurement

    result = closest_pair_dc(points)  # Perform closest pair of points

    end_time = time.time()  # End time measurement
    elapsed_time = end_time - start_time

    return result, elapsed_time

# Example usage
points = [(2, 3), (12, 30), (40, 50), (5, 1), (12, 10), (3, 4)]
result, execution_time = find_closest_pair_time(points)

print(f"Closest pair of points: {result[0]} and {result[1]}")
print(f"Distance: {result[2]}")
print(f"Execution time: {execution_time:.10f} seconds")
```

OUTPUT:-



TIME COMPLEXITY:- $O(n\log n)$