1. 140. 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?

   Given 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).

   output: P3, P4, P6, P5, P7, P1

code:

```
import math

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

def closest_pair_brute_force(points):
    min_distance = float('inf')
    closest_pair = None

    for i in range(len(points)):
        for j in range(i + 1, len(points)):
            p1, p2 = points[i], points[j]
            distance = euclidean_distance(p1, p2)
            if distance < min_distance:
                min_distance = distance
                closest_pair = (p1, p2)

    return closest_pair, min_distance

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

closest_pair, min_distance = closest_pair_brute_force(points)
print(f"Closest pair: {closest_pair[0]} - {closest_pair[1]}")
print(f"Minimum distance: {min_distance}")
```

output:

```
PS C:\Users\karth>
PS C:\Users\karth> & C:/Users/karth/AppData/Local/Programs/Python/Python312/python.exe c:/Users/karth/OneDrive/Documents/Orig
Closest pair: (1, 2) - (3, 1)
Minimum distance: 2.23606797749979
PS C:\Users\karth>
```

Time complexity:

F(n)=o(nlogn)