**DAY-04**

**Q01.**

import math

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

min\_dist, closest\_pair = float('inf'), None

for i in range(len(points)):

for j in range(i + 1, len(points)):

dist = math.dist(points[i], points[j])

if dist < min\_dist: min\_dist, closest\_pair = dist, (points[i], points[j])

print("Closest pair:", closest\_pair, "Minimum distance:", min\_dist)

**output:**

Closest pair: ((1, 2), (3, 1)) Minimum distance: 2.23606797749979

**Q02.**

points = [(10, 0), (11, 5), (5, 3), (9, 3.5), (15, 3), (12.5, 7), (6, 6.5), (7.5, 4.5)]

hull = set()

for i in range(len(points)):

for j in range(i + 1, len(points)):

left = right = 0

for k in range(len(points)):

if k == i or k == j: continue

cross = (points[j][0] - points[i][0]) \* (points[k][1] - points[i][1]) - (points[j][1] - points[i][1]) \* (points[k][0] - points[i][0])

left += cross > 0; right += cross < 0

if not left or not right: hull.update([points[i], points[j]])

print(hull)

**output:**

{(6, 6.5), (10, 0), (15, 3), (5, 3), (12.5, 7)}

**Q03.**

points = [(1, 1), (4, 6), (8, 1), (0, 0), (3, 3)]

hull = set()

for i in range(len(points)):

for j in range(i + 1, len(points)):

left = right = 0

for k in range(len(points)):

if k == i or k == j: continue

cross = (points[j][0] - points[i][0]) \* (points[k][1] - points[i][1]) - (points[j][1] - points[i][1]) \* (points[k][0] - points[i][0])

left += cross > 0; right += cross < 0

if not left or not right: hull.update([points[i], points[j]])

hull = list(hull)

hull.sort(key=lambda p: (p[0], p[1]))

print("Convex Hull:", hull)

**output:**

Convex Hull: [(0, 0), (4, 6), (8, 1)]

**Q04.**

**Q05.**

import itertools

cost\_matrix = [[3, 10, 7], [8, 5, 12], [4, 6, 9]]

min\_cost, best\_assignment = float('inf'), []

for perm in itertools.permutations(range(len(cost\_matrix))):

total\_cost = sum(cost\_matrix[i][perm[i]] for i in range(len(perm)))

if total\_cost < min\_cost: min\_cost, best\_assignment = total\_cost, perm

print("Optimal Assignment:", [(f'worker {i+1}', f'task {best\_assignment[i]+1}') for i in range(len(best\_assignment))], "\nTotal Cost:", min\_cost)

**output:**

Optimal Assignment: [('worker 1', 'task 3'), ('worker 2', 'task 2'),

('worker 3', 'task 1')]

Total Cost: 16

**Q06.**

import itertools

weights, values, capacity = [2, 3, 1], [4, 5, 3], 4

max\_value, best\_combination = 0, []

for i in range(len(weights) + 1):

for comb in itertools.combinations(range(len(weights)), i):

if sum(weights[j] for j in comb) <= capacity:

total\_value = sum(values[j] for j in comb)

if total\_value > max\_value: max\_value, best\_combination = total\_value, comb

print("Optimal Selection:", best\_combination, "\nTotal Value:", max\_value)

**output:**

Optimal Selection: (1, 2)

Total Value: 8