1. Selection Sort

def selection\_sort(arr):

n = len(arr)

for i in range(n):

min\_idx = i

for j in range(i+1, n):

if arr[j] < arr[min\_idx]:

min\_idx = j

arr[i], arr[min\_idx] = arr[min\_idx], arr[i]

return arr

# Example Usage

arr = [64, 25, 12, 22, 11]

sorted\_arr = selection\_sort(arr)

print("Sorted array is:", sorted\_arr)

1. Bubble Sort

def bubble\_sort(arr):

n = len(arr)

for i in range(n-1):

for j in range(0, n-i-1):

if arr[j] > arr[j+1]:

arr[j], arr[j+1] = arr[j+1], arr[j]

return arr

1. Insertion sort

def insertion\_sort(arr):

for i in range(1, len(arr)):

key = arr[i]

j = i - 1

while j >= 0 and key < arr[j]:

arr[j + 1] = arr[j]

j -= 1

arr[j + 1] = key

# Example Usage

arr = [12, 11, 13, 5, 6]

insertion\_sort(arr)

print("Sorted array is:", arr)

1. Sequential Search

def sequential\_search(arr, target):

for i in range(len(arr)):

if arr[i] == target:

return i

return -1

# Example

arr = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]

target = 5

result = sequential\_search(arr, target)

print(f"Target {target} found at index: {result}")

Brute-Force StringBrute-Force String Matching Matching

5.Brute-Force String Matching

5.brute force string matching

def brute\_force\_string\_match(text, pattern):

n = len(text)

m = len(pattern)

for i in range(n - m + 1):

j = 0

while j < m and text[i + j] == pattern[j]:

j += 1

if j == m:

return i

return -1

# Test the function

text = "hello world"

pattern = "world"

result = brute\_force\_string\_match(text, pattern)

print("Pattern found at index:", result)

6.Closest-Pair6.Closest-Pair

6.Closest-Pair

import math

def closest\_pair(points):

def distance(p1, p2):

return math.sqrt((p1[0] - p2[0])\*\*2 + (p1[1] - p2[1])\*\*2)

min\_distance = float('inf')

closest = (None, None)

for i in range(len(points)):

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

dist = distance(points[i], points[j])

if dist < min\_distance:

min\_distance = dist

closest = (points[i], points[j])

return closest

# Example Usage

points = [(1, 2), (3, 4), (5, 6), (7, 8)]

print(closest\_pair(points))

7.Convex-Hull Problems

from scipy.spatial import ConvexHull

import matplotlib.pyplot as plt

import numpy as np

points = np.random.rand(30, 2)

hull = ConvexHull(points)

plt.plot(points[:,0], points[:,1], 'o')

for simplex in hull.simplices:

plt.plot(points[simplex, 0], points[simplex, 1], 'k-')

plt.show()

8. Exhaustive Search

def exhaustive\_search(data, target):

for i in range(len(data)):

if data[i] == target:

return i

return -1

data = [4, 7, 2, 1, 9, 5]

target = 9

result = exhaustive\_search(data, target)

print(f"Target found at index: {result}")