1. **Linear search(sequential search**

def linear\_search(arr, target):

for i in range(len(arr)):

if arr[i] == target:

return i

return -1

arr = [4, 2, 7, 1, 9]

target = 7

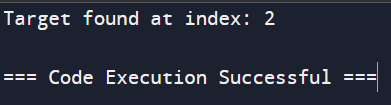
result = linear\_search(arr, target)

if result != -1:

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

else:

print("Target not found in the array.")



1. **Merg sort**

arr = [38, 27, 43, 3, 9, 82, 10]

n = len(arr)

width = 1

while width < n:

for i in range(0, n, 2\*width):

left = arr[i:i+width]

right = arr[i+width:i+2\*width]

l, r = 0, 0

for j in range(i, min(i+2\*width, n)):

if r >= len(right) or (l < len(left) and left[l] <= right[r]):

arr[j] = left[l]

l += 1

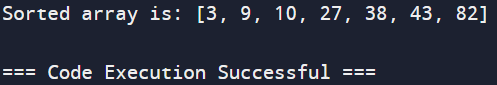
else:

arr[j] = right[r]

r += 1

width \*= 2

print("Sorted array is:", arr)



1. **String matching**

def naive\_string\_matching(text, pattern):

n = len(text)

m = len(pattern)

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

match = True

for j in range(m):

if text[i + j] != pattern[j]:

match = False

break

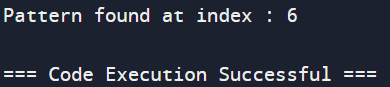
if match:

print(f"Pattern found at index {i}")

text = "hello world"

pattern = "world"

naive\_string\_matching(text, pattern)



1. **Convex hull**

points = [(0, 3), (1, 1), (2, 2), (4, 4), (0, 0), (1, 2), (3, 1), (3, 3)]

points.sort()

def cross(o, a, b):

return (a[0] - o[0]) \* (b[1] - o[1]) - (a[1] - o[1]) \* (b[0] - o[0])

lower = []

for p in points:

while len(lower) >= 2 and cross(lower[-2], lower[-1], p) <= 0:

lower.pop()

lower.append(p)

upper = []

for p in reversed(points):

while len(upper) >= 2 and cross(upper[-2], upper[-1], p) <= 0:

upper.pop()

upper.append(p)

convex\_hull = lower[:-1] + upper[:-1]

print(convex\_hull)

