1. Selection Sort

```
def selection(arr):
    n = len(arr)
    for i in range(n):
        mini = i
        for j in range(i+1, n):
            if arr[j] < arr[mini]:
                mini = j

        arr[i], arr[mini] = arr[mini], arr[i]

    return arr
arr = [64, 25, 12, 22, 11]
s= selection(arr)
print("Sorted array:", s)
```

2. Bubble Sort

```
def sort(arr):
    n = len(arr)
    for i in range(n):
        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
arr=[5,4,3,2,1]
res = sort(arr)
print(res)
```

3. Insertion sort

```
def sorting(A):
  for i in range(0,len(A)):
    key = A[i]
    j = i - 1
    while j >= 0 and A[j] > key:
      A[j + 1] = A[j]
      j = j - 1
    A[j + 1] = key
  return A

A=[5,1,3,2,4]
res = sorting(A)
print(res)
```

4. Sequential Search

```
def linear(num,t):
   n = len(num)
   for i in range(n):
      if(num[i]==t):
         print(i)
         break
   else:
      print("not found")

num = [5,6,2,1,3]
t = 3
linear(num,t)
```

5.Brute-Force String Matching

```
def check(text, pattern):

   for i in range(len(text)-len(pattern)):

     j = 0

   while j < len(pattern) and text[i + j] == pattern[j]:

     j = j + 1

   if j == len(pattern):

     return i

   return -1


text = "savee tha"

pattern = "tha"

res = check(text , pattern)

print(res)
```

6.Closest-Pair

```
def brute(points):

   mini= float('inf')

   closest= None


   for i in range(len(points)):

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

         p1 = points[i]
```

```
            p2 = points[j]
            dis = distance(p1, p2)

            if dis < mini:
                mini = dis
                closest = (p1, p2)

    return closest, mini

def distance(p1, p2):
    return ((p1[0] - p2[0]) ** 2 + (p1[1] - p2[1]) ** 2) ** 0.5

point = [(0, 0), (1, 1), (4, 5), (3, 1)]
res , value = brute(point)
print(res)
print(value)
```

7.Convex-Hull Problems

8.Exhaustive Search

```
from itertools import combinations

class Item:
    def __init__(self, value, weight):
        self.value = value
        self.weight = weight

    def __repr__(self):
        return f"(Value: {self.value}, Weight: {self.weight})"

def exhaustive(items, maxw):
    n = len(items)
    value = 0
    combination = []
    for r in range(n + 1):
        for c in combinations(items, r):
            tweight = sum(item.weight for item in c)
            tvalue = sum(item.value for item in c)
            if tweight <= maxw and tvalue > value:
                value = tvalue
                combination = c

    return combination, value

items = [
    Item(value=60, weight=10),
    Item(value=100, weight=20),
    Item(value=120, weight=30)
]

maxw = 50
combination, value = exhaustive(items, maxw)

print("Best combination of items:")
for item in combination:
    print(item)
print(f"Total value: {value}")
```