1)selection sort

def selectionsort(arr):

n=len(arr)

for i in range(0,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

print("selection sort:",selectionsort([]))

print("selection sort:",selectionsort([5]))

print("selection sort:",selectionsort([3,3,3,3]))

print("selection sort:",selectionsort([-4,-1,-33,-2]))

2)

def bubblesort(arr):

n=len(arr)

for i in range(n):

swapped=False

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

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

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

swapped=True

if not swapped:

break

return arr

print("bubble sort:",bubblesort([64,25,12,22,11]))

print("bubblesort:",bubblesort([13,29,10,14,37]))

print("bubblesort:",bubblesort([3,5,2,1,4]))

print("bubblesort:",bubblesort([1,2,3,4,5]))

print("bubblesort:",bubblesort([5,4,3,2,1]))

3)

def insertion\_sort(arr):

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

key = arr[i]

j = i - 1

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

arr[j + 1] = arr[j]

j -= 1

arr[j + 1] = key

return arr

input\_array\_1 = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3]

output\_array\_1 = insertion\_sort(input\_array\_1)

print("Sorted Output 1:", output\_array\_1)

4)

def merge\_sort(arr):

if len(arr) > 1:

mid = len(arr) // 2

left\_half = arr[:mid]

right\_half = arr[mid:]

merge\_sort(left\_half)

merge\_sort(right\_half)

i = j = k = 0

while i < len(left\_half) and j < len(right\_half):

if left\_half[i] < right\_half[j]:

arr[k] = left\_half[i]

i += 1

else:

arr[k] = right\_half[j]

j += 1

k += 1

while i < len(left\_half):

arr[k] = left\_half[i]

i += 1

k += 1

while j < len(right\_half):

arr[k] = right\_half[j]

j += 1

k += 1

return arr

arr = [12, 11, 13, 5, 6, 7]

sorted\_arr = merge\_sort(arr)

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

5)

def print\_board(board):

for row in board:

print(" ".join(row))

def solve\_n\_queens(n):

def is\_safe(board, row, col):

for i in range(row):

if board[i][col] == 'Q':

return False

if col - (row - i) >= 0 and board[i][col - (row - i)] == 'Q':

return False

if col + (row - i) < n and board[i][col + (row - i)] == 'Q':

return False

return True

def solve(board, row):

if row == n:

print\_board(board)

print()

return

for col in range(n):

if is\_safe(board, row, col):

board[row][col] = 'Q'

solve(board, row + 1)

board[row][col] = '.'

board = [['.' for \_ in range(n)] for \_ in range(n)]

solve(board, 0)

print("Solutions for 4-Queens:")

solve\_n\_queens(4)