**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SORTING TECHNIQUES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**SELECTION SORT:**

def selection\_sort(arr):

n = len(arr)

for i in range(n):

min\_index = i

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

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

min\_index = j

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

input\_str = input("Enter a list of numbers: ")

arr = [int(x) for x in input\_str.split()]

selection\_sort(arr)

print(arr)

**TIME COMPLEXITY = O(n2)**

**QUICK SORT:**

def quick\_sort(arr):  
 if len(arr) <= 1:  
 return arr  
 pivot = arr[0]  
 left = []  
 right = []  
 for i in arr[1:]:  
 if i < pivot:  
 left.append(i)  
 else:  
 right.append(i)  
 return quick\_sort(left) + [pivot] + quick\_sort(right)  
user\_input = input("Enter a list of numbers: ")  
arr = [int(x) for x in user\_input.split()]  
sorted\_list = quick\_sort(arr)  
print("Sorted list:", sorted\_list)

**TIME COMPLEXITY = O(n2)**

**MERGE SORT:**

def merge\_sort(arr):

if len(arr) <= 1:

return arr

mid = len(arr) // 2

left = merge\_sort(arr[:mid])

right = merge\_sort(arr[mid:])

return merge(left, right)

def merge(left, right):

result = []

i, j = 0, 0

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

if left[i] <= right[j]:

result.append(left[i])

i += 1

else:

result.append(right[j])

j += 1

result.extend(left[i:])

result.extend(right[j:])

return result

arr = list(map(int, input("Enter array elements: ").split()))

sorted\_arr = merge\_sort(arr)

print(sorted\_arr)

**TIME COMPLEXITY = O(n log n)**

**INVERSION:**

def inversions(arr):

count = 0

n = len(arr)

for i in range(n):

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

if arr[i] > arr[j]:

count += 1

return count

arr = list(map(int, input("Enter array elements: ").split()))

inversion\_count = inversions(arr)

print("Number of inversions:", inversion\_count)