**2105A41131**

**DAY:7**

**SELECTION SORT:**

**TIME COMPLEXITY: O(n square)**

**Code:**

def selection\_sort(arr):

length=len(arr)

for i in range(length):

min\_index=i

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

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

min\_index=j

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

arr=[14,8,2,19,69,1]

selection\_sort(arr)

print("the sorted elements are:")

print(arr)

**QUICK SORT:**

* It is based on divide and conquer method.

**TIME COMPLEXITY:**

**BEST CASE : O(Nlog(N))**

**WORST CASE: O(N square)**

**Code:**

def quicksort(list):

if len(list)<=1:

return list

else:

pivot=list[0]

left\_list=[i for i in list if i<pivot]

right\_list=[i for i in list if i>pivot]

return quicksort(left\_list)+[pivot]+quicksort(right\_list)

list=list(map(int,input().split()))

sorted\_list=quicksort(list)

print(sorted\_list)

**MERGE SORT:**

**CODE:**

def merge\_sort(list):

if len(list)<=1:

return list

else:

mid=len(list)//2

left=list[mid]

right=list[mid]

merge\_sort(left)

merge\_sort(right)

i=j=k=0

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

if left[i]<right[j]:

list[k]=list[i]

i+=1

k+=1

else:

list[k]=list[j]

j+=1

k+=1

while i<len(left):

list[k]=left[i]

i+=1

k+=1

while j<len(right):

list[k]=right[j]

j+=1

k+=1

list=list(map(int,input().split()))

sorted\_array=merge\_sort(list)

print(sorted\_array)

**TIME COMPLEXITY:**

**BEST CASE:O(logN)**

**WORST CASE:O(N)**

**INVERSION:**

Ex. [1,20,3,4,5]

Ans: {20,3}, {20,4}, {20,5}, {3,4} These are the various inversions of the above example.

**Code:**

arr=[1,20,4,3,5]

n=len(arr)

inv\_count=0

for i in range(n-1):

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

if arr[i]>arr[j]:

inv\_count+=1

print("the inversion elements are:")

print(inv\_count)