**DSP**

**Implement linear search and compute space and time complexity by using asyn [assempting notation]**

def linearsearch(list1,key):

for i in range(len(list1)):

if key==list1[i]:

print("key element is found")

break

else:

print("element is not found")

list1=[13,24,35,46,57,68,79]

print(list1)

key=int(input("enter the key element to search:"))

linearsearch(list1,key)

**out put:**

[13, 24, 35, 46, 57, 68, 79]

enter the key element to search:24

element is not found

key element is found

**2. Implement bubble sort, insertion sort, selection sort and compute space and time complexity plot graph using asymptomatic notation**

list1=[10,15,4,23,2]

print("unsorted list:",list)

for j in range(len(list1)-1):

for i in range(len(list1)-1-j):

if list1[i]>list1[i+1]:

list1[i],list1[i+1]=list1[i+1],list1[i]

print(list1)

else:

print(list1)

print()

print("sorted list:",list1)

**output:**

unsorted list: <class 'list'>

[10, 15, 4, 23, 2]

[10, 4, 15, 23, 2]

[10, 4, 15, 23, 2]

[10, 4, 15, 2, 23]

[4, 10, 15, 2, 23]

[4, 10, 15, 2, 23]

[4, 10, 2, 15, 23]

[4, 10, 2, 15, 23]

[4, 2, 10, 15, 23]

[2, 4, 10, 15, 23]

sorted list: [2, 4, 10, 15, 23]

**selection sort.**

def selection\_sort(arr):

for i in range(len(arr)):

temp=i

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

if arr[temp]<arr[j]:

temp=j

arr[i],arr[temp]=arr[temp],arr[i]

return (arr)

input\_arr=[43,61,2,35,18,9,18,34,4]

print(selection\_sort(input\_arr))

**out put:**

[61, 43, 18, 9, 18, 34, 4, 35, 2]

**Insertion sort**:

def insertionsort(my\_list):

for index in range(1,len(my\_list)):

cur\_element=my\_list[index]

pos=index

while cur\_element<my\_list[pos-1]and pos>0:

my\_list[pos]=my\_list[pos-1]

pos=pos-1

my\_list[pos]=cur\_element

list1=[10,4,25,1,5]

insertionsort(list1)

print(list1)

**output:**

**[1, 4, 5, 10, 25]**