**1)Write a python program for merge sort.**

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=[38,27,43,3,9,82,10]

sorted\_arr=merge\_sort(arr)

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

**OUTPUT:** Sorted array is: [3, 9, 10, 27, 38, 43, 82]

**2)Write a python program for quicksort.**

def quick\_sort(arr):

if len(arr)<=1:

return arr

else:

pivot=arr[0]

less\_than\_pivot=[x for x in arr[1:]if x<=pivot]

greater\_than\_pivot=[x for x in arr[1:]if x>pivot]

return quick\_sort(less\_than\_pivot)+[pivot]+quick\_sort(greater\_than\_pivot)

my\_list=[29,10,14,37,13]

sorted\_list=quick\_sort(my\_list)

print(sorted\_list)

**OUTPUT**: [10, 13, 14, 29, 37]

**3) Write a python program for Binary Search**.

def binary\_search(arr,target):

low=0

high=len(arr)-1

while low<=high:

mid=(low+high)//2

if arr[mid]==target:

return mid

elif arr[mid]<target:

low=mid+1

else:

high=mid-1

return -1

arr=[1,3,5,7,9]

target=5

result=binary\_search(arr,target)

if result !=-1:

print(f"Element {target} found at index {result}")

else:

print("Element not found")

**OUTPUT:** Element 5 found at index 2

**4) Write a python program for Strassen’s matrix multiplication.**

import numpy as np

def strassen(A,B):

if len(A)==1:

return A\*B

mid=len(A)//2

A11=A[:mid,:mid]

A12=A[:mid,mid:]

A21=A[mid:,:mid]

A22=A[mid:,mid:]

B11=B[:mid,:mid]

B12=B[:mid,mid:]

B21=B[mid:,:mid]

B22=B[mid:,mid:]

M1=strassen(A11+A22,B11+B22)

M2=strassen(A21+A22,B11)

M3=strassen(A11,B12-B22)

M4=strassen(A22,B21-B11)

M5=strassen(A11+A12,B22)

M6=strassen(A21-A11,B11+B12)

M7=strassen(A12-A22,B21+B22)

C11=M1+M4-M5+M7

C12=M3+M5

C21=M2+M4

C22=M1-M2+M3+M6

C=np.zeros((len(A),len(B)))

C[:mid,:mid]=C11

C[:mid,mid:]=C12

C[mid:,:mid]=C21

C[mid:,mid:]=C22

return C

A=np.array([[1,2],[3,4]])

B=np.array([[5,6],[7,8]])

result=strassen(A,B)

print(result)

**OUTPUT:**

[[19. 22.]

[43. 50.]]

**5) Write a python program Karatsuba algorithm for multiplication.**

def karatsuba(x,y):

if x<10 or y<10:

return x\*y

max\_len=max(len(str(x)),len(str(y)))

half\_len=max\_len//2

x\_high,x\_low=divmod(x,10\*\*half\_len)

y\_high,y\_low=divmod(y,10\*\*half\_len)

z0=karatsuba(x\_low,y\_low)

z1=karatsuba(x\_low+x\_high,y\_low+y\_high)

z2=karatsuba(x\_high,y\_high)

return(z2\*10\*\*(2\*half\_len))+((z1-z2-z0)\*10\*\*half\_len)+z0

result=karatsuba(1234,5678)

print(f"The product of 1234 and 5678 is:{result}")

**OUTPUT:** The product of 1234 and 5678 is:7006652

**6) Write a python program for Closest pair of points using divide and conquer.**

import math

def distance(p1,p2):

return math.sqrt((p1[0]-p2[0])\*\*2+(p1[1]-p2[1])\*\*2)

def closest\_pair(points):

def closest\_util(px,py):

if len(px)<=3:

return min((distance(px[i],px[j]),(px[i],px[j])) for i in range(len(px)) for j in range(i+1,len(px)))

mid=len(px)//2

mid\_point=px[mid]

pyl=[p for p in py if p[0]<=mid\_point[0]]

pyr=[p for p in py if p[0]>mid\_point[0]]

d1,pair1=closest\_util(px[:mid],pyl)

d2,pair2=closest\_util(px[mid:],pyr)

d=min(d1,d2)

closest\_pair=pair1 if d1<d2 else pair2

strip=[p for p in py if abs(p[0]-mid\_point[0])<d]

for i in range(len(strip)):

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

if (strip[j][1]-strip[i][1])<d:

d=distance(strip[i],strip[j])

closest\_pair=(strip[i],strip[j])

return d,closest\_pair

points.sort(key=lambda x:x[0])

py=sorted(points,key=lambda x:x[1])

return closest\_util(points,py)

points = [(0, 0), (1, 1), (2, 2), (3, 3), (0, 1)]

result = closest\_pair(points)

print(f"The closest pair of points is: {result[1]} with a distance of {result[0]}")

**OUTPUT:** The closest pair of points is: ((0, 0), (0, 1)) with a distance of 1.0

**7) Write a python program for Median of medians.**

def median\_of\_medians(arr, k):

if len(arr) < 10:

return sorted(arr)[k]

sublists = [arr[i:i + 5] for i in range(0, len(arr), 5)]

medians = [sorted(sublist)[len(sublist) // 2] for sublist in sublists]

pivot=median\_of\_medians(medians,len(medians)//2)

low=[x for x in arr if x<pivot]

high=[x for x in arr if x>pivot]

k\_index=len(low)

if k<k\_index:

return median\_of\_medians(low,k)

elif k>k\_index:

return median\_of\_medians(high,k-k\_index-1)

else:

return pivot

arr=[3,6,2,7,5,1,4,8]

k=3

print(median\_of\_medians(arr,k))

**8) Write a python program for Meet in middle technique.**

**9)Write a python program for finding maximum and minimum .**

numbers=[10,20,4,45,99]

maximum\_value=max(numbers)

minimum\_value=min(numbers)

print(f"The maximum value is:{maximum\_value}")

print(f"The minimum value is:{minimum\_value}")

**OUTPUT:**

The maximum value is:99

The minimum value is:4