DESIGN AND ANALYSIS OF ALGORITHM PRACTICAL-1

NAME: VEDANT BHUTADA

ROLL: 69

BATCH: A4

AIM: Write a program read the file and find Minimum and Maximum using Brute force approach. Also propose and implement an improved algorithm based on Divide and Conquer Strategy. Also calculate the time taken by both the algotihm.

Compute the time complexity of i) Brute force algorithm ii) Divide and Conquer based algorithm iii) Plot a graph to show the time comparison between algorithms

```
import pandas as pd
def brute(path):
  if(path=="data500.csv"):
    df=pd.read_csv("data500.csv")
    data=df['numbers'].tolist()
  if(path=="test1.csv"):
    df=pd.read_csv("test1.csv")
    data=df['Open'].tolist()
  if(path=="test2.csv"):
    df=pd.read_csv("test2.csv")
    data=df['fare_amount'].tolist()
  if(path=="test3.csv"):
    df=pd.read_csv("test3.csv")
    data=df['volume'].tolist()
  if(path=="test4.csv"):
    df=pd.read_csv("test4.csv")
    data=df['val'].tolist()
  max=data[0]
  min=data[0]

  for i in data:
    if i>max:
      max=i
    if i<min:
      min=i
  print("MIN number is:",max)
  print("Max number is:",min)
print("Max Min for test1 file:")
brute("test1.csv")
print("Max Min for data500 file:")
brute("data500.csv")
print("Max Min for test3 file:")
brute("test3.csv")
print("Max Min for test2 file:")
brute("test2.csv")
print("Max Min for test4 file:")
brute("test4.csv")
```

```
Max Min for test1 file:
MIN number is: 40.58
Max number is: 34.54
Max Min for data500 file:
MIN number is: 998
Max number is: 4
Max Min for test3 file:
MIN number is: 268336455.0
Max number is: 78029.0
Max Min for test2 file:
MIN number is: 180.0
Max number is: -3.0
Max Min for test4 file:
MIN number is: 496858.598
Max number is: 0.001
```

```
def time(methode,path):
  import time
  start=time.perf_counter()
  if(methode=="brute"):
    if(path=="data500.csv"):
      brute("data500.csv")
    if(path=="test1.csv"):
```

```
      brute("test1.csv")
    if(path=="test3.csv"):
      brute("test3.csv")
    if(path=="test2.csv"):
      brute("test2.csv")
    if(path=="test4.csv"):
      brute("test4.csv")
  if(methode=="div_con"):
    if(path=="data500.csv"):
      div_con("data500.csv")
    if(path=="test1.csv"):
      div_con("test1.csv")
    if(path=="test3.csv"):
      div_con("test3.csv")
    if(path=="test2.csv"):
      div_con("test2.csv")
    if(path=="test4.csv"):
      div_con("test4.csv")
  end= time.perf_counter()
  timetaken=end-start
  print("Time taken is: ",timetaken)
  return timetaken
```

```
def returning():
  x=time("brute","data500.csv")
  x1=time("brute","test1.csv")
  x2=time("brute","test3.csv")
  x3=time("brute","test2.csv")
  x4=time("brute","test4.csv")
  y=time("div_con","data500.csv")
  y1=time("div_con","test1.csv")
  y2=time("div_con","test3.csv")
  y3=time("div_con","test2.csv")
  y4=time("div_con","test4.csv")
  return x,x1,x2,x3,x4,y,y1,y2,y3,y4
```

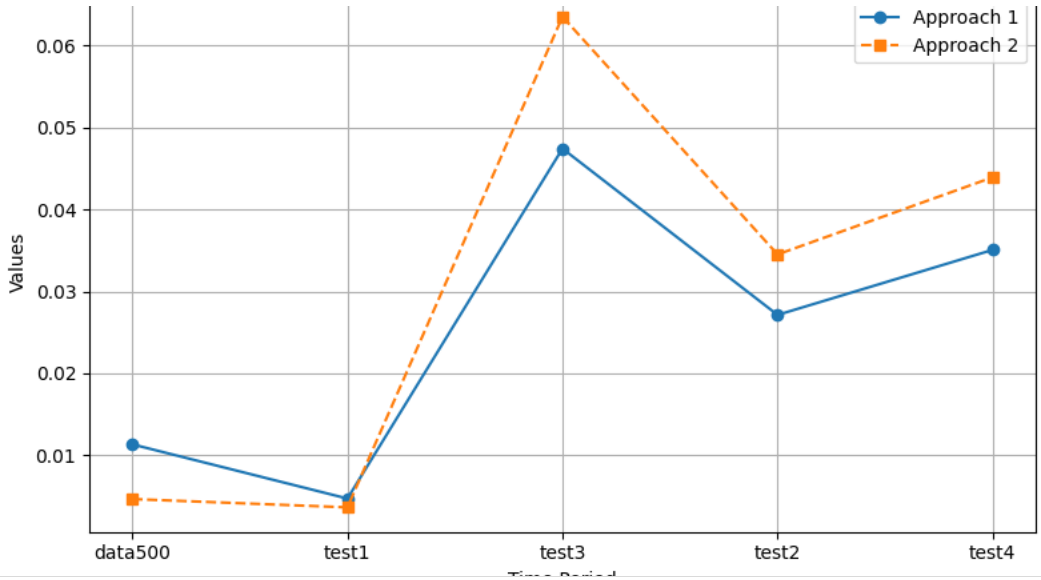
```
def MinMaxDAC(A, i, j):
    if(i == j):
        return (A[i], A[i])
    if((j - i) == 1):
        if (A[i] <= A[j]):
            return (A[i], A[j])
        else:
            return (A[j], A[i])
    else :
        mid = (i + j) / 2
        mid=round(mid)
        LMin, LMax = MinMaxDAC(A, i, mid)
        RMin, RMax = MinMaxDAC(A, mid + 1, j)
        if(LMax > RMax):
            max = LMax
        else :
            max = RMax
        if(LMin < RMin):
            min = LMin
        else :
            min = RMin
        return (min, max)

def div_con(path):
  if(path=="data500.csv"):
    df=pd.read_csv("data500.csv")
    data=df['numbers'].tolist()
  if(path=="test1.csv"):
    df=pd.read_csv("test1.csv")
    data=df['Open'].tolist()
  if(path=="test3.csv"):
    df=pd.read_csv("test3.csv")
    data=df['low'].tolist()
  if(path=="test2.csv"):
```

```
    df=pd.read_csv("test2.csv")
    data=df['fare_amount'].tolist()
  if(path=="test4.csv"):
    df=pd.read_csv("test4.csv")
    data=df['val'].tolist()
  num=len(data)
  min,max=MinMaxDAC(data,1,num-1)
  print(max)
  print(min)

div_con("data500.csv")
div_con("test1.csv")
div_con("test3.csv")
div_con("test2.csv")
div_con("test4.csv")
```

```
998
4
40.58
34.54
335.06
1.61
180.0
-3.0
496858.598
0.001
```

```
import matplotlib.pyplot as plt

x,x1,x2,x3,x4,y,y1,y2,y3,y4=returning()

approach1_values = [x,x1,x2,x3,x4]
approach2_values = [y,y1,y2,y3,y4]

x_labels = ['data500', 'test1','test3','test2','test4']

plt.figure(figsize=(8, 5))
plt.plot(x_labels, approach1_values, marker='o', label='Approach 1', linestyle='-')
plt.plot(x_labels, approach2_values, marker='s', label='Approach 2', linestyle='--')

plt.xlabel('Time Period')
plt.ylabel('Values')
plt.title('Comparison of Two Approaches')

plt.legend()

plt.grid(True)
plt.tight_layout()
plt.show()
```

```
MIN number is: 998
Max number is: 4
Time taken is:  0.01133883600050467
MIN number is: 40.58
Max number is: 34.54
Time taken is:  0.004726056999970751
MIN number is: 268336455.0
Max number is: 78029.0
Time taken is:  0.047410869000486855
MIN number is: 180.0
Max number is: -3.0
Time taken is:  0.027126709999720333
MIN number is: 496858.598
Max number is: 0.001
Time taken is:  0.03505934700024227
998
4
Time taken is:  0.004665186000238464
40.58
34.54
Time taken is:  0.0036537400001179776
335.06
1.61
Time taken is:  0.06350181899961171
180.0
-3.0
Time taken is:  0.034504027999901155
496858.598
0.001
Time taken is:  0.04391020299954107
```



✓ 1s completed at 10:07 PM

CONCLUSION: The brute force approach provides accurate results but may not be the most efficient for large datasets due to its time complexity.
The divide and conquer approach is a more efficient way to find the minimum and maximum values in a dataset, especially for large datasets. It takes advantage of the divide-and-conquer paradigm to reduce the number of comparisons required.
In practice, for large datasets, the divide and conquer approach is recommended as it offers better performance.