EXPERIMENT NO 01
****

==
Author : Atharva Paliwal
Roll No : 40 [5B]
Date : 11-August-2020
==

****
AIM : Write a program to find Minimum and Maximum element in the given array using Min-Max Algorithm based on Divide and Conquer Strategy.

****

CODES

****

```
INF = float('inf')
import time
import random
def  findMinAndMax1(numbers,n):
    max= numbers[0]
    min= numbers[0]
    for i in range(0,n):
       if numbers[i] > max:
          max= numbers[i]
       if numbers[i] < min:
          min= numbers[i]
    return (min,max)

def findMinAndMax2(A, left, right, min, max):
       if left == right:
              if min > A[right]:
                     min = A[right]
              if max < A[left]:
                     max = A[left]

              return min, max
       if right - left == 1:

              if A[left] < A[right]:
                     if min > A[left]:
                            min = A[left]

                     if max < A[right]:
                            max = A[right]

              else:
                     if min > A[right]:
                            min = A[right]
```

```
                    if max < A[left]:
                          max = A[left]
              return min, max
        mid = (left + right) // 2
        min, max = findMinAndMax2(A, left, mid, min, max)
        min, max = findMinAndMax2(A, mid + 1, right, min, max)
        return min, max







def minmax(A):
    (min2, max2) = (INF, -INF)

    #naive
    start_time1 = time.time()
    (min1, max1) = findMinAndMax1(A,len(A))
    end_time1 = time.time()
    print("The minimum element in the "+str(len(A))+" list is with naive", min1)
    print("The maximum element in the "+str(len(A))+"  list is with naive", max1)
    print("time taken was for naive:=",end_time1-start_time1)
    print("")
    print("")
    #d and c
    start_time2 = time.time()
    (min2,max2) = findMinAndMax2(A, 0, len(A) - 1, min2, max2)
    end_time2 = time.time()
    print("The minimum element in the "+str(len(A))+"  list is with divide and 
conquer", min2)
    print("The maximum element in the "+str(len(A))+" list is with divide and 
conquer", max2)
    print("time taken was for naive:=",end_time2-start_time2)
    print("")
    print("")
    return(len(A),start_time1-end_time1,start_time2-end_time2)


#for 100 elements
A =random.sample(range(1,100000),100)
n1,time11,time12=minmax(A)

#for 1000 elements
A =random.sample(range(1,10000000),1000)
n2,time21,time22=minmax(A)

#for 10000 elements
A =random.sample(range(1,1000000),10000)
n3,time31,time32=minmax(A)
```

```
#for 100000
A =random.sample(range(1,10000000),100000)
n3,time41,time42=minmax(A)
*************************************************************************************

*************************************************************************************
```

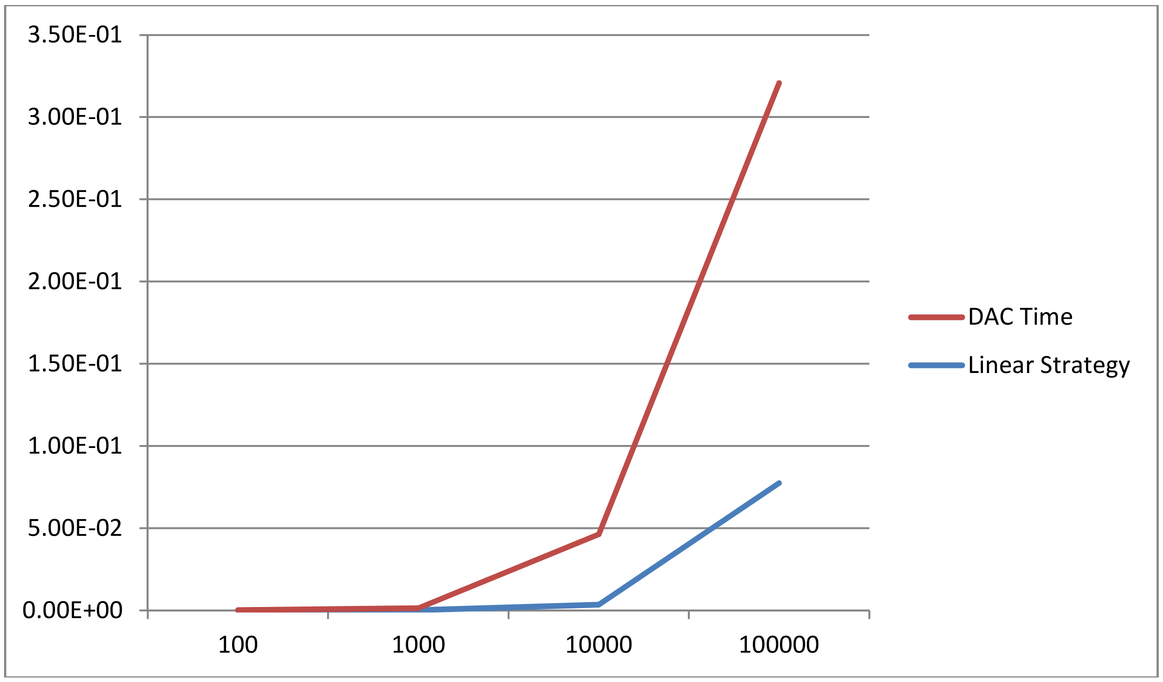
TABULAR ANALYSIS

****

NUMBER OF INPUTS	MIN-MAX TIME (SIMPLE LINEAR STRATEGY)	MIN-MAX (DAC) TIME
100	3.2901763916015625e-05	0.00016951560974121094
1000	0.0003266334533691406	0.0010440349578857422
10000	0.0034902095794677734	0.04285717010498047
100000	0.07744646072387695	0.2432863712310791

****

GRAPH



***********************************INFERENCE*************************************

We have learnt about divide and conquer programming. We have seen that DAC takes less time to execute a particular array than normal min-max algo. In DAC we have used recursive approach/top down approach.

****

******END******