RAJSHAHI UNIVERSITY OF ENGINEERING & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Lab Report 2

Course Code: *CSE 2202*

Course Title: *Sessional Based on CSE 2201.*

Submitted By:	**Submitted To:**
Name : Shanjid Hasan Nishat	Dr. Md. Ali Hossain
Roll No : 1803172	Associate Professor,
Section : 'C'	Dept. of Computer Science and
Date of Submission: 06/02/ 2021	Engineering.
	Rajshahi University of
	Engineering & Technology.

Problem Statement: Comparison of Straight forward and recursive algorithms for finding maximum and minimum.

Description and Algorithm:

Straight forward method means the basic method to solve a problem. In this method, the maximum and minimum number can be found separately. To find the maximum and minimum numbers, the following straightforward algorithm can is used:

```
Maximum-Minimum (numbers[])
max := numbers[1]
min := numbers[1]

for i = 2 to n do
   if numbers[i] > max then
      max := numbers[i]
   if numbers[i] < min then
      min := numbers[i]
return (max, min)
```

In this algorithm, number of comparisons for n number of inputs is **2n -2**.

By **Divide and Conquer** approach, the array is divided into two halves. Then using recursive approach maximum and minimum numbers in each half are found. Then, return the maximum of two maximum of each half and the minimum of two minimum of each half and thus the final minimum and maximum are found. To find the maximum and minimum numbers, the following divide and conquer algorithm can is used:

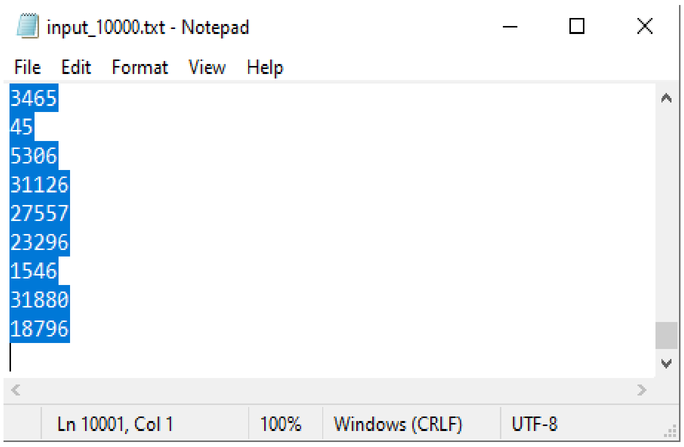
```
Maximum-Minimum (x, y)
if y – x ≤ 1 then
   return (max(numbers[x], numbers[y]), min((numbers[x],
numbers[y]))
else
   (max1, min1):= Maximum-Minimum(x, ⌊((x + y)/2)⌋)
   (max2, min2):= Maximum-Minimum(⌊((x + y)/2) + 1)⌋,y)
return (max(max1, max2), min(min1, min2))
```

In this algorithm, number of comparisons for n number of inputs where, $\boldsymbol{n = 2^k}$ is $\boldsymbol{\frac{3n}{2} - 2}$.

Code:

```
#include<bits//stdc++.h>
using namespace std;
long long arr[100010],len = 0;
long long cnt1=0,cnt2=0;
void find_min_max(long long &mini,long long &maxi)
{cnt1++;
    for(long long i = 0 ; i < len ; i++)
    {cnt1+=2;
        cnt1++;
        if(arr[i]<mini)
        {cnt1++;
            mini = arr[i];
        }
        cnt1++;
        if(arr[i]>maxi)
        {cnt1++;
            maxi = arr[i];
        }
    }
}
void divide_and_conquer_min_max(long long i, long long n, long long
&mini, long long &maxi)
{
    cnt2++;
    if(n-i == 0)
    {
        cnt2+=2;
        mini = min(arr[i],mini);
        maxi = max(arr[i],maxi);
        return;
    }
    else if(n-i == 1)
    {
        cnt2+=2;
        mini = min(mini,min(arr[i],arr[n]));
        maxi = max(maxi,max(arr[i],arr[n]));
        return;
    }
    cnt2+=2;
    divide_and_conquer_min_max(i,(i+n)/2,mini,maxi);
    divide_and_conquer_min_max(((i+n)/2+1),n,mini,maxi);
}
void readFile(string fname)
{
    long long x,i=0;
    ifstream inFile;
    inFile.open(fname);
    if (!inFile)
    {
        cout << "Cannot open file.\n";
        exit(1);
    }
    while (inFile >> x)
    {
        arr[i++] = x;
    }inFile.close();
    len = i;
}
int main()
{
    long long a,b,data[100][8],i=0;
        cnt1=0;
        cnt2=0;
        readFile("input_10000.txt");
        data[i][0] = len;
        a = LLONG_MAX;
        b = LLONG_MIN;
        find_min_max(a,b);
        data[i][1] = a;
        data[i][2] = b;
        data[i][5] = cnt1;

        a = LLONG_MAX;
        b = LLONG_MIN;
        divide_and_conquer_min_max(0,len-1,a,b);
        data[i][3] = a;
        data[i][4] = b;
        data[i++][6] = cnt2;

        cnt1=0;
        cnt2=0;
        readFile("input_20000.txt");
        data[i][0] = len;
        a = LLONG_MAX;
        b = LLONG_MIN;
        find_min_max(a,b);
        data[i][1] = a;
        data[i][2] = b;
        data[i][5] = cnt1;

        a = LLONG_MAX;
        b = LLONG_MIN;
        divide_and_conquer_min_max(0,len-1,a,b);
        data[i][3] = a;
        data[i][4] = b;
        data[i++][6] = cnt2;

        cnt1=0;
        cnt2=0;
        readFile("input_30000.txt");
        data[i][0] = len;
        a = LLONG_MAX;
        b = LLONG_MIN;
        find_min_max(a,b);
        data[i][1] = a;
        data[i][2] = b;
        data[i][5] = cnt1;

        a = LLONG_MAX;
        b = LLONG_MIN;
        divide_and_conquer_min_max(0,len-1,a,b);
        data[i][3] = a;
        data[i][4] = b;
        data[i++][6] = cnt2;
        cnt1=0;
        cnt2=0;
        readFile("input_40000.txt");
        data[i][0] = len;
        a = LLONG_MAX;
        b = LLONG_MIN;
        find_min_max(a,b);
        data[i][1] = a;
        data[i][2] = b;
        data[i][5] = cnt1;

        a = LLONG_MAX;
        b = LLONG_MIN;
        divide_and_conquer_min_max(0,len-1,a,b);
        data[i][3] = a;
        data[i][4] = b;
        data[i++][6] = cnt2;

        cnt1=0;
        cnt2=0;
        readFile("input_50000.txt");
        data[i][0] = len;
        a = LLONG_MAX;
        b = LLONG_MIN;
        find_min_max(a,b);
        data[i][1] = a;
        data[i][2] = b;
        data[i][5] = cnt1;

        a = LLONG_MAX;
        b = LLONG_MIN;
        divide_and_conquer_min_max(0,len-1,a,b);
        data[i][3] = a;
        data[i][4] = b;
        data[i++][6] = cnt2;

    cout<<"Data\tNormal_Minimum\t  Normal_Maximum\t  DAC_Minimum\t
DAC_Maximum\t  Normal_Steps\t  DAC_Steps\n\n";
    for(long long j = 0 ; j < i ; j++)
    {
        cout<<data[j][0]<<"\t\t"<<data[j][1]<<"\t\t"<<data[j][2]<<"
\t\t"<<data[j][3]<<"\t\t"<<data[j][4]<<"\t\t"<<data[j][5]<<"\t\t"<<data
[j][6]<<"\n\n";
    }
    return 0;
}
```

Input:

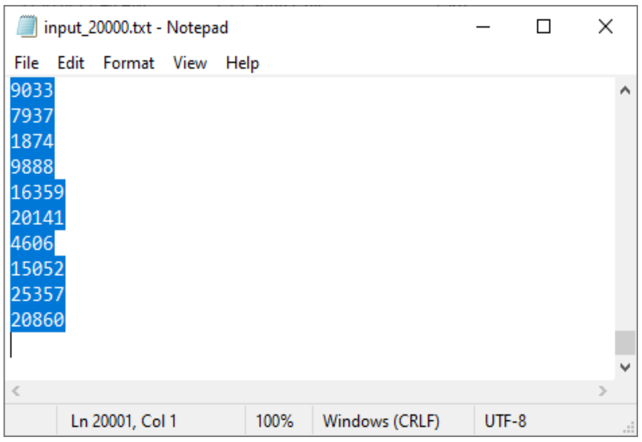


input_10000.txt - Notepad

File Edit Format View Help

```
3465
45
5306
31126
27557
23296
1546
31880
18796
```

Ln 10001, Col 1 100% Windows (CRLF) UTF-8

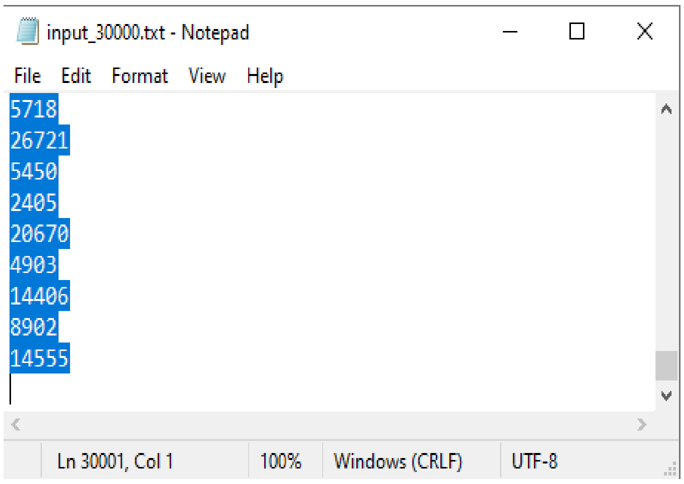


input_20000.txt - Notepad

File Edit Format View Help

```
9033
7937
1874
9888
16359
20141
4606
15052
25357
20860
```

Ln 20001, Col 1 100% Windows (CRLF) UTF-8

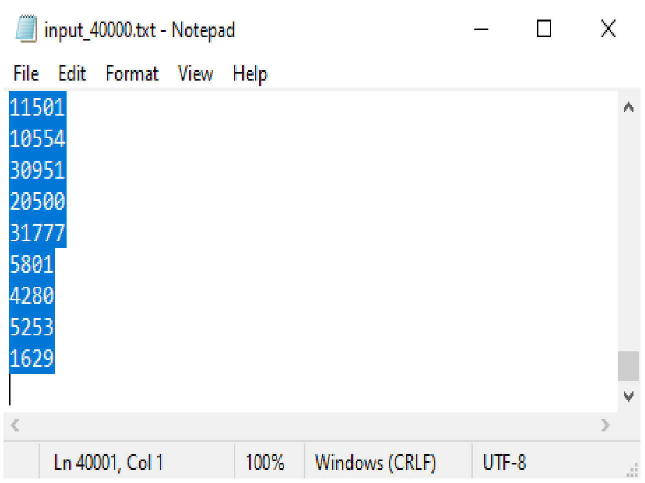


input_30000.txt - Notepad

File Edit Format View Help

```
5718
26721
5450
2405
20670
4903
14406
8902
14555
```

Ln 30001, Col 1 100% Windows (CRLF) UTF-8

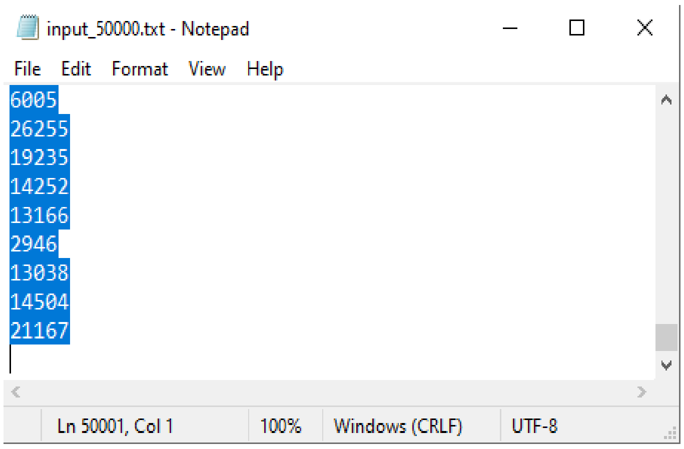


input_40000.txt - Notepad

File Edit Format View Help

```
11501
10554
30951
20500
31777
5801
4280
5253
1629
```

Ln 40001, Col 1 100% Windows (CRLF) UTF-8



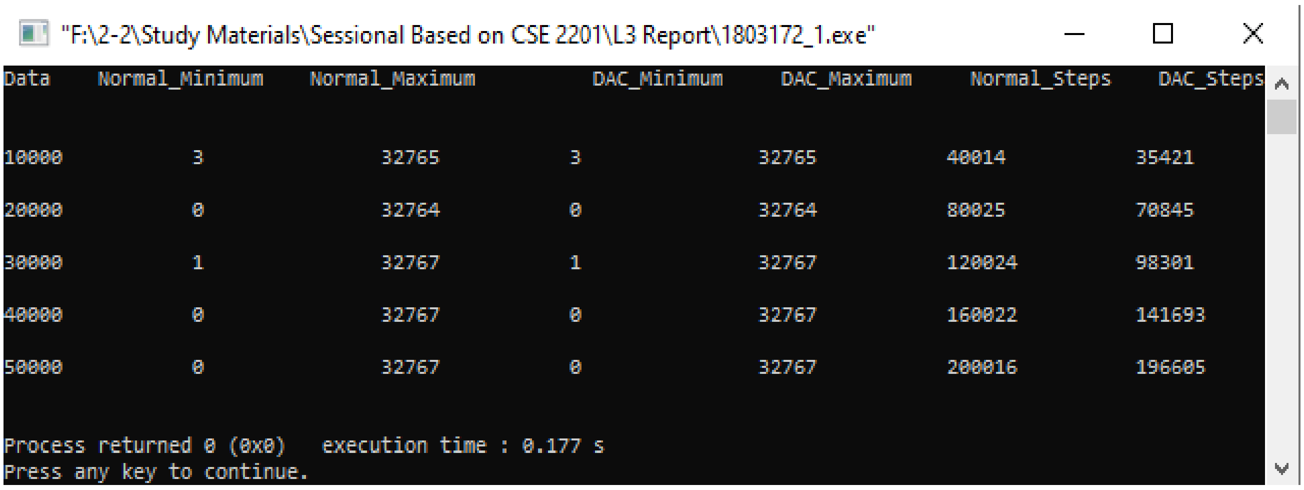
input_50000.txt - Notepad

File Edit Format View Help

```
6005
26255
19235
14252
13166
2946
13038
14504
21167
```

Ln 50001, Col 1 100% Windows (CRLF) UTF-8

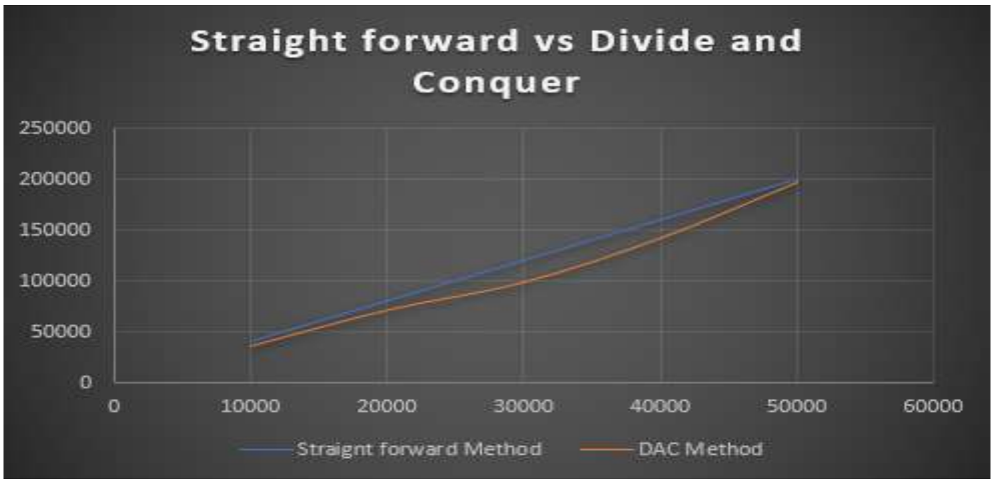
Output:

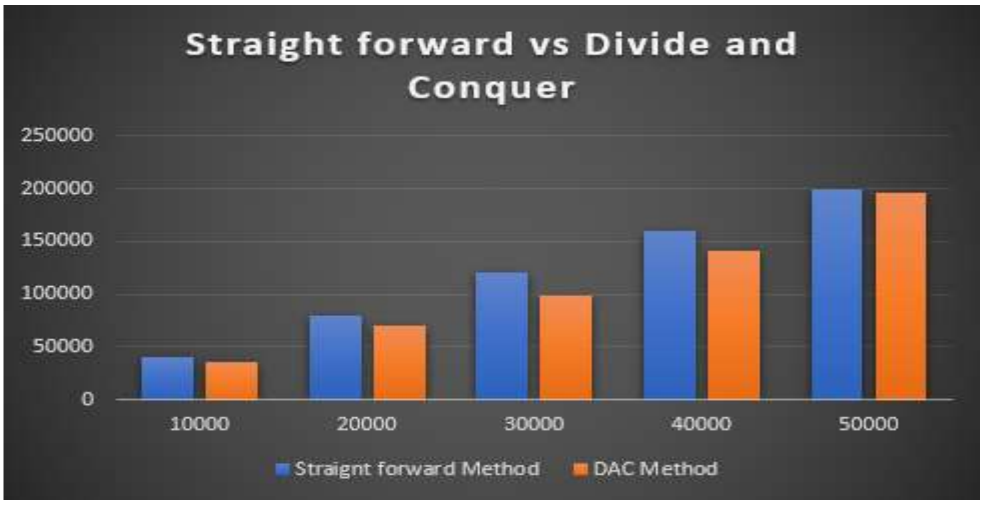


"F:\2-2\Study Materials\Sessional Based on CSE 2201\L3 Report\1803172_1.exe"

```
Data     Normal_Minimum     Normal_Maximum          DAC_Minimum       DAC_Maximum       Normal_Steps      DAC_Steps


10000           3               32765           3               32765           40014           35421

20000           0               32764           0               32764           80025           70845

30000           1               32767           1               32767           120024          98301

40000           0               32767           0               32767           160022          141693

50000           0               32767           0               32767           200016          196605


Process returned 0 (0x0)   execution time : 0.177 s
Press any key to continue.
```

Graph:





Discussion and conclusion:

In this problem we have seen the comparison of recursive and non-recursive method for finding minimum of maximum from a given array. Here, we see from the algorithm that the divide and conquer method takes less comparison than the straight forward algorithm which we can also notice in the two graphs. The line of divide and conquer method is always below the line of straight forward method. However, using the asymptotic notation time complexity of the both methods are represented by **O(n)**.