RAJSHAHI UNIVERSITY OF ENGINEERING AND TECHNOLOGY

LAB REPORT – 02

COURSE NAME: SESSIONAL BASED ON CSE-2201
COURSE CODE: CSE-2102

SUBMITTED TO:
DR. MD. ALI HOSSAIN
ASSOCIATE PROFESSOR
Department of Computer Science & Engineering
Rajshahi University of Engineering & Technology

SUBMITTED BY:
SRABONTI DEB
Roll-1803163
Section - C
Department of Computer Science & Engineering
Rajshahi University of Engineering & Technology

SUBMISSION DATE – 3 FEBRUARY, 2021

Problem Statement: Comparison of Straight forward and recursive max-min algorithms.

Theory: A recursive algorithm is an algorithm which calls itself with "smaller (or simpler)" input values, and which obtains the result for the current input by applying simple operations to the returned value for the smaller (or simpler) input.

Recursive maximum-minimum algorithm:

```
Algorithm MaxMin(i,j,max,min)
//a[1:n] is a global array.Parameters i and j are integers,
//1<=i<=j<=n. The effect is to set max and min to the
//largest and smallest values in a[i:j],respectively.
{
   if(i=j) then max:=min :=a[i];//Sma||(P)
   else if (i=j-1) then // Another case of Sma||(P)
        {
              if(a[i]<a[j]) then
                   {
                     max:=a[j]; min:=a[i];
                   }
              else
                   {
                    max:=a[i]; min:=a[j];
                   }
        }
    else
        {//IF P is not small,divide P into subproblems.
          //Find where to split the set.
             mid:= [(i+j)/2];
          //Solve the subproblems.
             MaxMin(i,mid,max,min);
             MaxMin(mid+1,j,max1,min1);
         //Combine  the solutions.
             if(max<max1) then max:=max1;
             if(min>min1)then min:=min1;
```

Straight forward maximum-minimum algorithm:

```
Algorithm StraightMaxMin(a,n,max,min)
//Set max to the maximum and min to the minimum of a[1:n].
{
    max:=min:=a[1];
    for i:=2 to n do
```

```
       {
       if(a[i]>max) then max:=a[i];
       if(a[i]<min) then min:=a[i];
       }
}
```

Implemented code for recursive algorithm:

```
#include<bits/stdc++.h>
using namespace std;
vector<int>ans;
long long val=0;
void max_min(vector<int>&v, int idx1, int idx2, int& minimum, int& maximum)
{
   val++;
   if(idx1==idx2)
   {
      val++;
      if(maximum<ans[idx1])
      {
         maximum=ans[idx1];
         val++;
      }
      val++;
      if(minimum>ans[idx2])
      {
         minimum=ans[idx2];
         val++;
      }
      return;
   }
   val++;
   if(idx2-idx1==1)
   {
      val++;
      if(ans[idx1]<ans[idx2])
      {
         val++;
         if(minimum>ans[idx1])
         {
            minimum=ans[idx1];
            val++;
         }
         val++;
         if (maximum <ans[idx2])
```

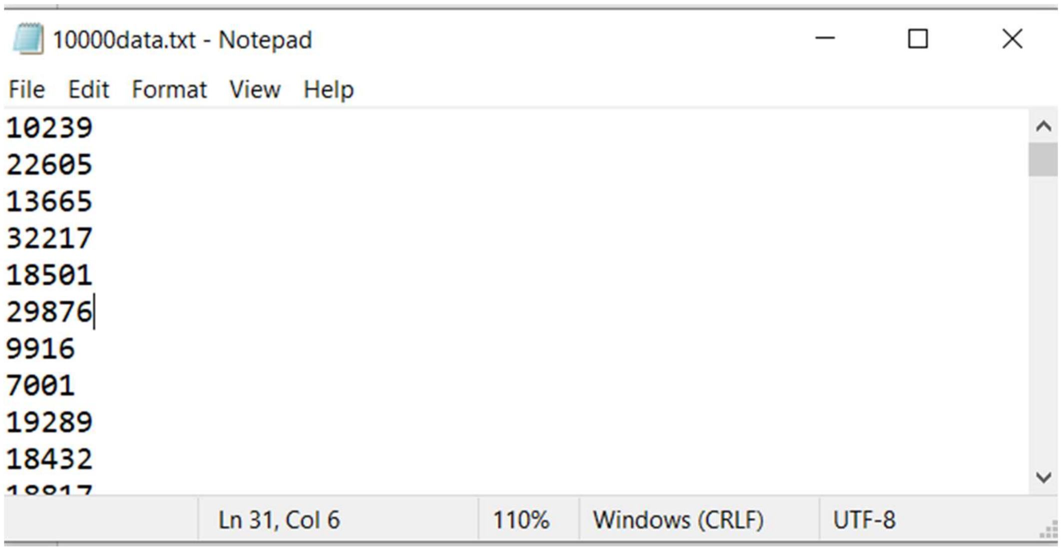
```
            {
               maximum =ans[idx2];
               val++;
            }
         }
         else
         {
            val++;
            if (minimum>ans[idx2])
            {
               minimum=ans[idx2];
               val++;
            }
            val++;
            if (maximum <ans[idx1])
            {
               maximum =ans[idx1];

            }
         }
         return;
      }
      val+=3;
      int mid=(idx1+idx2)/2;
      max_min(ans,idx1,mid,minimum, maximum);
      max_min(ans,mid + 1,idx2,minimum,maximum);
   }
   int main()
   {
      int p,k=0,minimum,maximum;
      ifstream inFile;
      inFile.open("10000data.txt");
      if (!inFile)
      {
         cout << "Cannot open file"<<endl;
         exit(1);
      }
      while(inFile>>p)
      {
         ans.push_back(p);
         k++;
      }
      cout<<endl;
      inFile.close();
      maximum=INT_MIN, minimum=INT_MAX;
      max_min(ans,0,k-1, minimum,maximum);
```

```
cout<<"minimum "<<minimum<<endl;
cout<<"maximum "<<maximum<<endl;
cout<<val<<endl;
val=0;
k=0;
ans.clear();
inFile.open("20000data.txt");
if (!inFile)
{
   cout << "Cannot open file"<<endl;
   exit(1);
}
while(inFile>>p)
{
   ans.push_back(p);
   k++;
}
cout<<endl;
inFile.close();
maximum=INT_MIN, minimum=INT_MAX;
max_min(ans,0,k-1, minimum,maximum);
cout<<"minimum "<<minimum<<endl;
cout<<"maximum "<<maximum<<endl;
cout<<val<<endl;
val=0;
k=0;
ans.clear();
inFile.open("30000data.txt");
if (!inFile)
{
   cout << "Cannot open file"<<endl;
   exit(1);
}
while(inFile>>p)
{
   ans.push_back(p);
   k++;
}
cout<<endl;
inFile.close();
maximum=INT_MIN, minimum=INT_MAX;
max_min(ans,0,k-1, minimum,maximum);
cout<<"minimum "<<minimum<<endl;
cout<<"maximum "<<maximum<<endl;
cout<<val<<endl;
val=0;
```

```
    k=0;
    ans.clear();
    inFile.open("40000data.txt");
    if (!inFile)
    {
       cout << "Cannot open file"<<endl;
       exit(1);
    }
    while(inFile>>p)
    {
       ans.push_back(p);
       k++;
    }
    cout<<endl;
    inFile.close();
    maximum=INT_MIN, minimum=INT_MAX;
    max_min(ans,0,k-1, minimum,maximum);
    cout<<"minimum "<<minimum<<endl;
    cout<<"maximum "<<maximum<<endl;
    cout<<val<<endl;
    val=0;
    k=0;
    ans.clear();
    inFile.open("50000data.txt");
    if (!inFile)
    {
       cout << "Cannot open file"<<endl;
       exit(1);
    }
    while(inFile>>p)
    {
       ans.push_back(p);
       k++;
    }
    cout<<endl;
    inFile.close();
    maximum=INT_MIN, minimum=INT_MAX;
    max_min(ans,0,k-1, minimum,maximum);
    cout<<"minimum "<<minimum<<endl;
    cout<<"maximum "<<maximum<<endl;
    cout<<val<<endl;
    val=0;
    k=0;
    ans.clear();
    return 0;
}
```

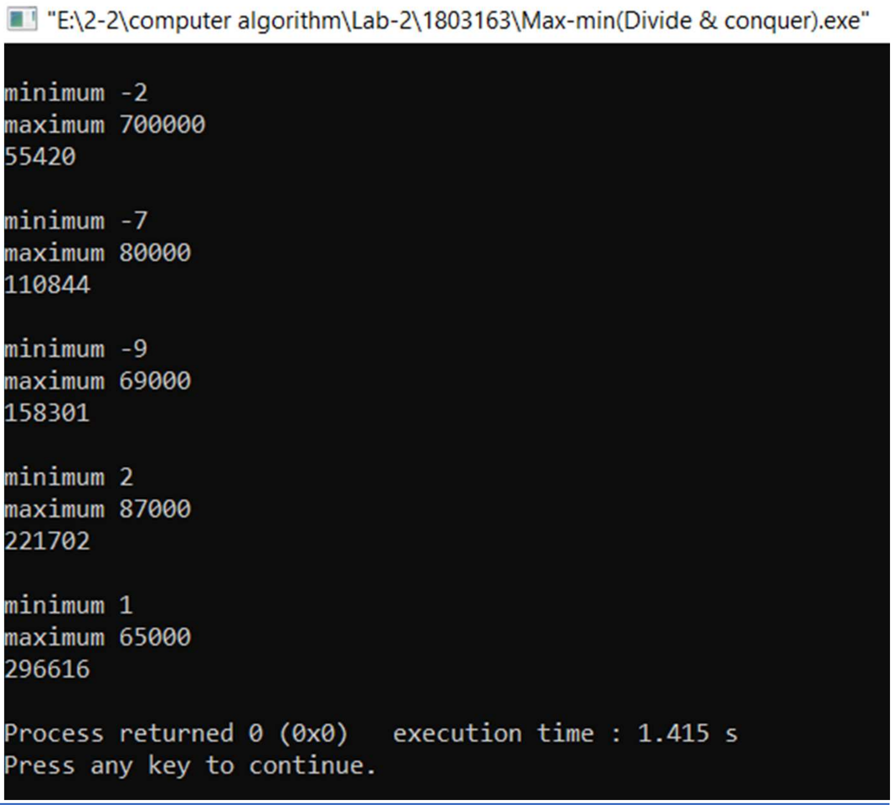
Sample input and output:

- **Input:**



```
10239
22605
13665
32217
18501
29876
9916
7001
19289
18432
```

- **Output:**


"E:\2-2\computer algorithm\Lab-2\1803163\Max-min(Divide & conquer).exe"

```
minimum -2
maximum 700000
55420

minimum -7
maximum 80000
110844

minimum -9
maximum 69000
158301

minimum 2
maximum 87000
221702

minimum 1
maximum 65000
296616

Process returned 0 (0x0)   execution time : 1.415 s
Press any key to continue.
```

Implemented code for straight forward algorithm:

```
#include<bits/stdc++.h>
using namespace std;
vector<int>ans;
long long val=0;
int main()
{
    int p,k=0,i;
    ifstream inFile;
    inFile.open("10000data.txt");
    if (!inFile)
    {
        cout<<"Cannot open file."<<endl;;
        exit(1);
    }
    while(inFile>>p)
    {
        ans.push_back(p);
    }
    cout<<endl;
    inFile.close();
    val+=2;
    int mn=ans[0],mx=ans[0];
    val++;
    for(i=1; i<ans.size(); i++)
    {
        val++;
        if(ans[i]<mn)
        {
            mn=ans[i];
            val++;
        }
        val++;
        if(ans[i]>mx)
        {
            mx=ans[i];
            val++;
        }
        val+=2;
    }
    val++;
    cout<<"minimum "<<mn<<endl;
    cout<<"maximum "<<mx<<endl;
    cout<<val<<endl;
```

```
val=0;
ans.clear();
inFile.open("20000data.txt");
if (!inFile)
{
   cout<<"Cannot open file."<<endl;;
   exit(1);
}
while(inFile>>p)
{
   ans.push_back(p);
}
cout<<endl;
inFile.close();
val+=2;
mn=ans[0],mx=ans[0];
val++;
for(i=1; i<ans.size(); i++)
{
   val+=2;
   if(ans[i]<mn)
   {
      mn=ans[i];
   }
   if(ans[i]>mx)
   {
      mx=ans[i];
   }
   val+=4;
}
val++;
cout<<"minimum "<<mn<<endl;
cout<<"maximum "<<mx<<endl;
cout<<val<<endl;
val=0;
ans.clear();
inFile.open("30000data.txt");
if (!inFile)
{
   cout<<"Cannot open file."<<endl;;
   exit(1);
}
while(inFile>>p)
{
   ans.push_back(p);
}
```

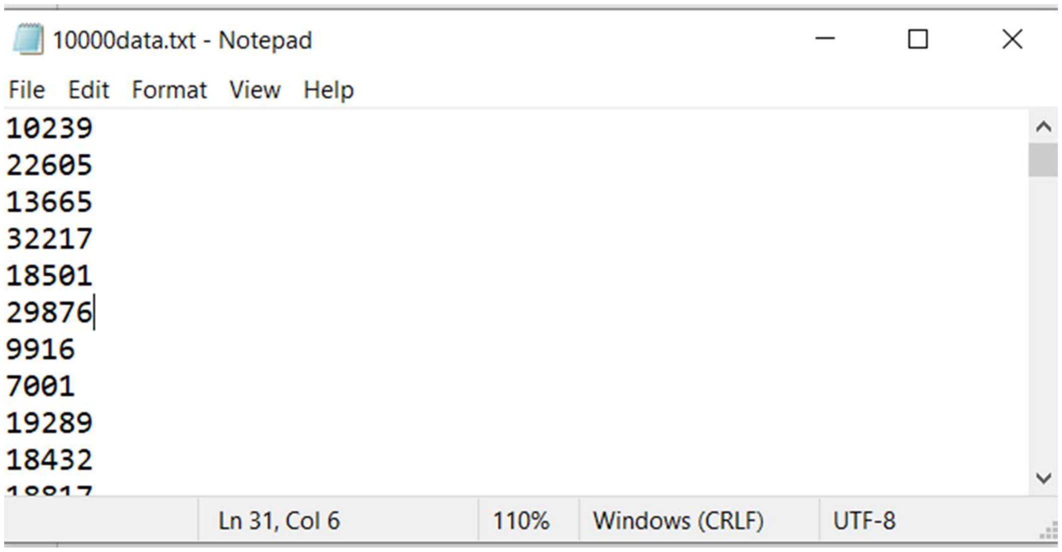
```
cout<<endl;
inFile.close();
val+=2;
mn=ans[0],mx=ans[0];
val++;
for(i=1; i<ans.size(); i++)
{
   val+=2;
   if(ans[i]<mn)
   {
      mn=ans[i];
   }
   if(ans[i]>mx)
   {
      mx=ans[i];
   }
   val+=4;
}
val++;
cout<<"minimum "<<mn<<endl;
cout<<"maximum "<<mx<<endl;
cout<<val<<endl;
val=0;
ans.clear();
inFile.open("40000data.txt");
if (!inFile)
{
   cout<<"Cannot open file."<<endl;;
   exit(1);
}
while(inFile>>p)
{
   ans.push_back(p);
}
cout<<endl;
inFile.close();
val+=2;
mn=ans[0],mx=ans[0];
val++;
for(i=1; i<ans.size(); i++)
{
   val+=2;
   if(ans[i]<mn)
   {
      mn=ans[i];
   }
```

```
    if(ans[i]>mx)
    {
        mx=ans[i];
    }
    val+=4;
}
val++;
cout<<"minimum "<<mn<<endl;
cout<<"maximum "<<mx<<endl;
cout<<val<<endl;
val=0;
ans.clear();
inFile.open("50000data.txt");
if (!inFile)
{
    cout<<"Cannot open file."<<endl;;
    exit(1);
}
while(inFile>>p)
{
    ans.push_back(p);
}
cout<<endl;
inFile.close();
val+=2;
mn=ans[0],mx=ans[0];
val++;
for(i=1; i<ans.size(); i++)
{
    val+=2;
    if(ans[i]<mn)
    {
        mn=ans[i];
    }
    if(ans[i]>mx)
    {
        mx=ans[i];
    }
    val+=4;
}
val++;
cout<<"minimum "<<mn<<endl;
cout<<"maximum "<<mx<<endl;
cout<<val<<endl;
val=0;
ans.clear();
```

```
    return 0;
}
```

Sample input and output:

- **Input:**



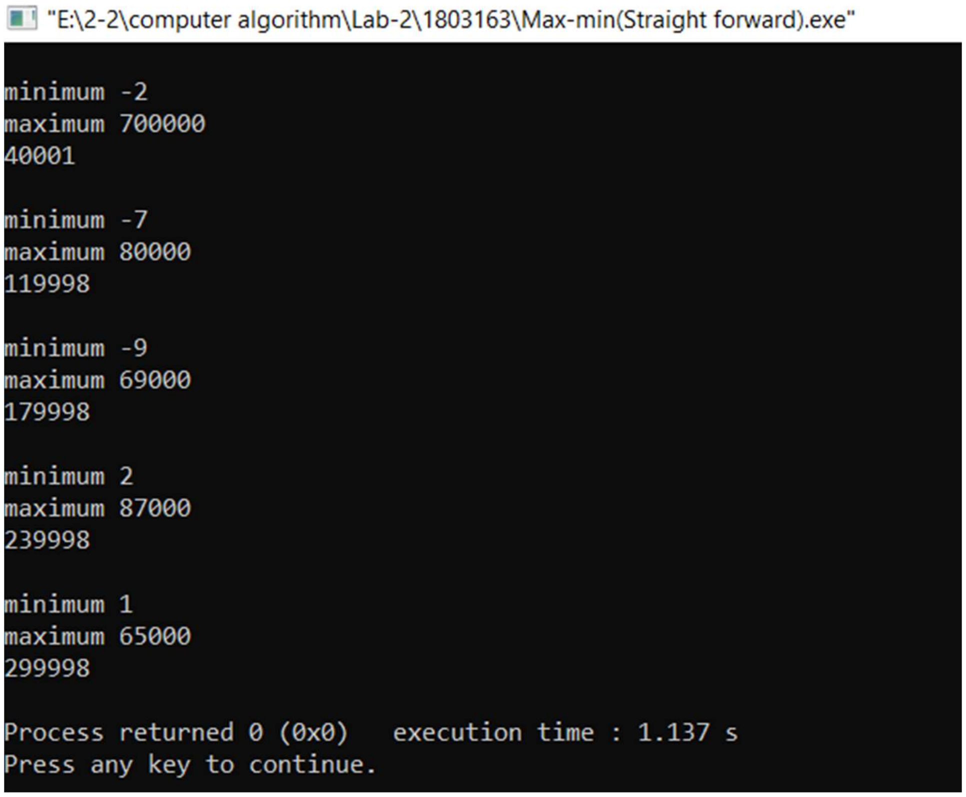
10000data.txt - Notepad

File Edit Format View Help

```
10239
22605
13665
32217
18501
29876
9916
7001
19289
18432
```

Ln 31, Col 6 | 110% | Windows (CRLF) | UTF-8

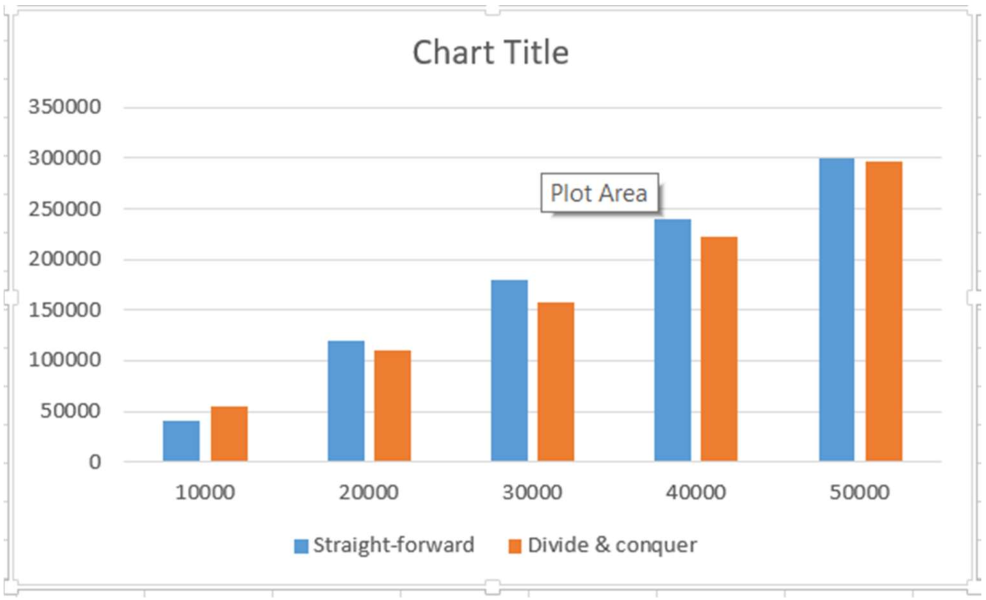
- **Output:**



"E:\2-2\computer algorithm\Lab-2\1803163\Max-min(Straight forward).exe"

```
minimum -2
maximum 700000
40001

minimum -7
maximum 80000
119998

minimum -9
maximum 69000
179998

minimum 2
maximum 87000
239998

minimum 1
maximum 65000
299998

Process returned 0 (0x0)   execution time : 1.137 s
Press any key to continue.
```

Graph:



Discussion & Conclusion:

Here we see that the number of counting steps for divide & conquer algorithm is less than those of straight forward algorithm.For large value of inputs,divide & conquer algorithm shows less time complexity.So,for generating maximum-minimum number,divide & conquer algorithm is better than straight forward algorithm.