Name: Chirag Rana
SE COMPS
2018130043

Experiment 4: (Min-Max Using Divide and Conquer)

Aim: Find minimum and Maximum using Divide and Conquer strategy.

Theory:

Many algorithms are recursive in nature to solve a given problem recursively dealing with sub-problems.

In **divide and conquer approach**, a problem is divided into smaller problems, then the smaller problems are solved independently, and finally the solutions of smaller problems are combined into a solution for the large problem.

Generally, divide-and-conquer algorithms have three parts −

- **Divide the problem** into a number of sub-problems that are smaller instances of the same problem.
- **Conquer the sub-problems** by solving them recursively. If they are small enough, solve the sub-problems as base cases.
- **Combine the solutions** to the sub-problems into the solution for the original problem.

Pros and cons of Divide and Conquer Approach

Divide and conquer approach supports parallelism as sub-problems are independent. Hence, an algorithm, which is designed using this technique, can run on the multiprocessor system or in different machines simultaneously.

In this approach, most of the algorithms are designed using recursion, hence memory management is very high. For recursive function stack is used, where function state needs to be stored.

Application of Divide and Conquer Approach

Following are some problems, which are solved using divide and conquer approach.

- Finding the maximum and minimum of a sequence of numbers
- Strassen’s matrix multiplication
- Merge sort
- Binary search

Explain the problem:

The Max-Min Problem in algorithm analysis is finding the maximum and minimum value in an array.

Solution

To find the maximum and minimum numbers in a given array ***numbers[]*** of size **n**, the following algorithm can be used. First we are representing the **naive method** and then we will present **divide and conquer approach**.

Naïve Method

Naïve method is a basic method to solve any 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 be used.

Algorithm: Max-Min-Element (numbers[])

```
max = numbers[1]              1 time
min = numbers[1]              1 time

for i = 2 to n do             n times
   if numbers[i] > max then   n-1 times
      max = numbers[i]        n-1 times
   if numbers[i] < min then   n-1 times
      min = numbers[i]        n-1 times
return (max, min)             1 time
```

Analysis

The number of comparison in Naive method is **2n - 2**.

Time Compexity: O(n)

The number of comparisons can be reduced using the divide and conquer approach.

Following is the technique.

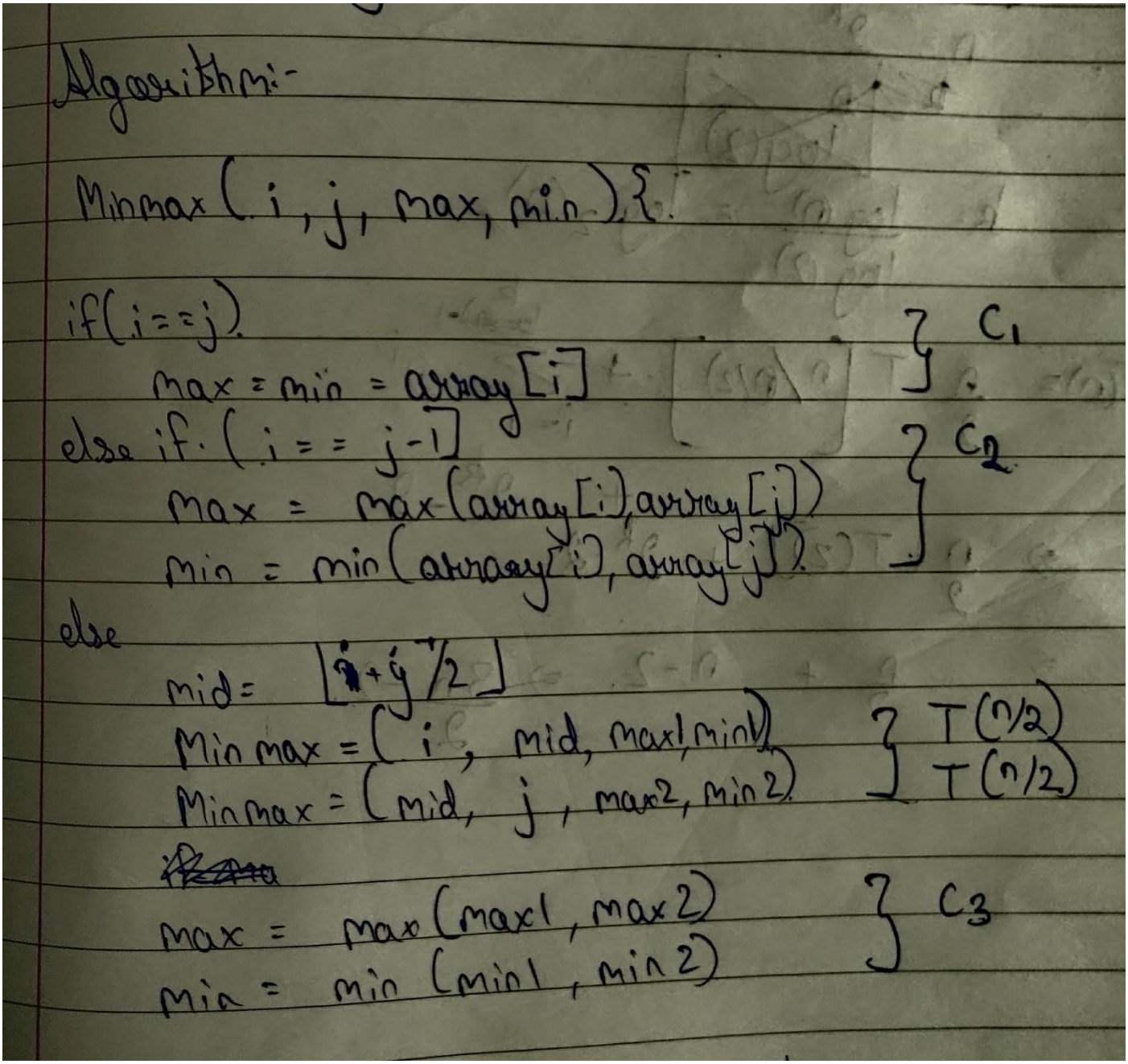
Divide and Conquer Approach

In this approach, the array is divided into two halves. Then using recursive approach maximum and minimum numbers in each halves are found. Later, return the maximum of two maxima of each half and the minimum of two minima of each half.

In this given problem, the number of elements in an array is y−x+1y−x+1, where **y** is greater than or equal to **x**.

Max−Min(x,y)Max−Min(x,y) will return the maximum and minimum values of an array numbers[x...y]numbers[x...y].

Algorithm: Max - Min(x, y)



Analysis

Let ***T(n)*** be the number of comparisons made by Max−Min(x,y)Max−Min(x,y), where the number of elements n=y−x+1n=y−x+1.

If ***T(n)*** represents the numbers, then the recurrence relation can be represented as

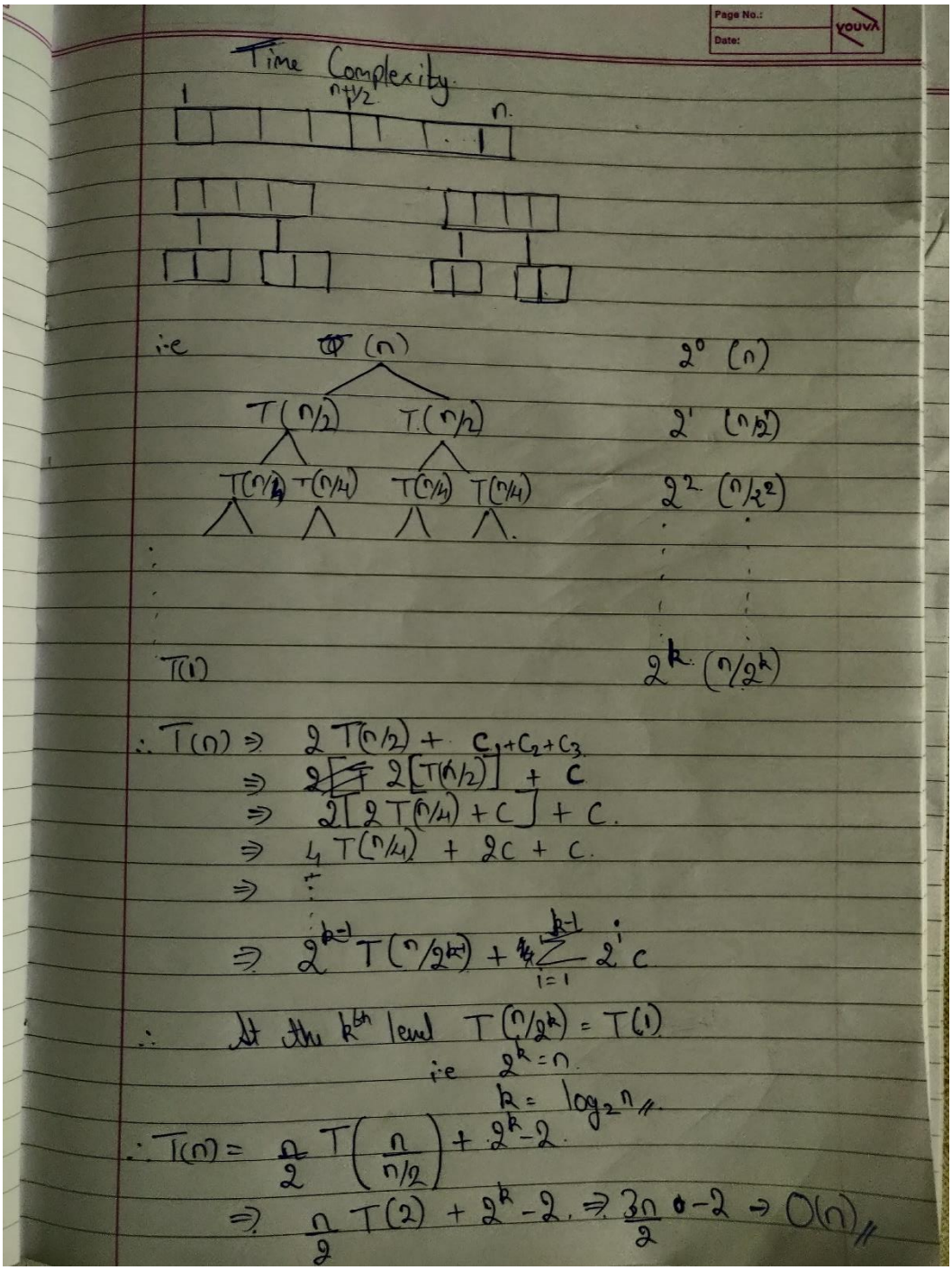
$$T(n) = \begin{cases} T\left(\lfloor \frac{n}{2} \rfloor\right) + T\left(\lceil \frac{n}{2} \rceil\right) + 2 & for\ n > 2 \\ 1 & for\ n = 2 \\ 0 & for\ n = 1 \end{cases}$$

Let us assume that ***n*** is in the form of power of **2**. Hence, **n = 2^k** where **k** is height of the recursion tree.

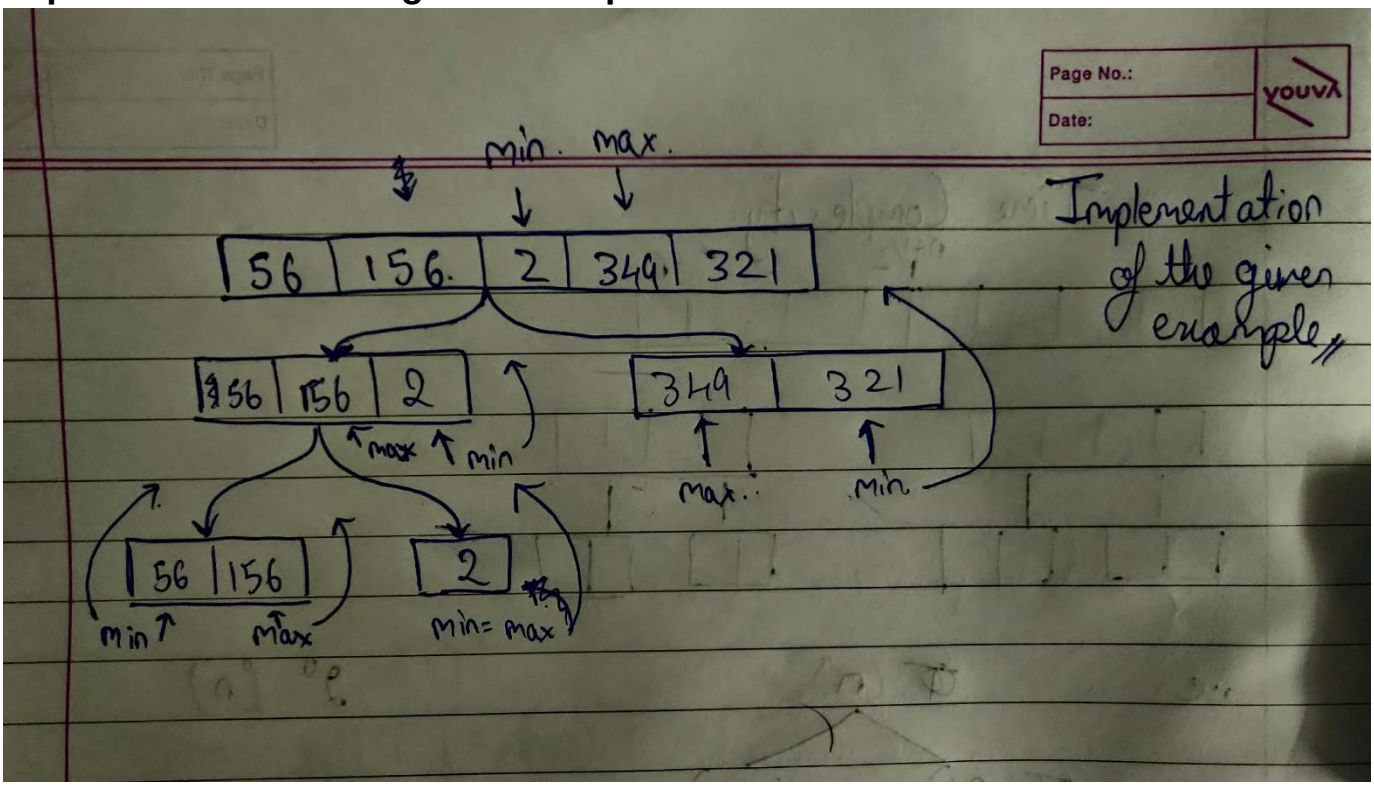
So,

$$T(n) = 2.T(\frac{n}{2}) + 2 = 2.\left(2.T(\frac{n}{4}) + 2\right) + 2.... = \frac{3n}{2} - 2$$

Compared to Naïve method, in divide and conquer approach, the number of comparisons is less. However, using the asymptotic notation both of the approaches are represented by **O(n)**.



Implementation of the given example:



Code:

```
#include <stdio.h>
#include<stdlib.h>
void maxmin(int i,int j,int*max,int*min,int arr[])
{
   for(int k = i;k <= j; k++){
      printf("%d ",arr[k]);
   }
   printf("\n");
   int max1,max2,min1,min2;
        int mid;
        if(i==j)
                *max=*min=arr[i];
        else if(i==j-1)
        {
                if(arr[i] < arr[j])
                        {
                                *max=arr[j];
                                *min=arr[i];
                        }
                        else
                        {
                                *max=arr[i];
                                *min=arr[j];
```

```
                }
        }
        else
        {
                mid=(i+j)/2;
                maxmin(i,mid,&max1,&min1,arr);
                maxmin(mid+1,j,&max2,&min2,arr);
                if(max1>max2)
                        *max = max1;
                else
                        *max = max2;
                if(min1 < min2)
                        *min = min1;
                else
                        *min = min2;
        }
        printf("minval: %d \n",(*min));
        printf("maxval: %d \n",(*max));
    printf("\n");

}
int main()
{
        int max,min,arr[20],size;
        printf("Enter the size of an array: ");
        scanf("%d",&size);
        printf("Enter the elements of an array: ");
        for(int i=0;i<size;i++)
       scanf("%d",&arr[i]);
    max=arr[0];
        min=arr[0];

        maxmin(0,size-1,&max,&min,arr);
        printf("The minimum element of an array is %d\n",min );
        printf("The maximum element of an array is %d\n",max );
}
```

Output:
For the input array: 56, 156, 2, 349, 321.

```
Enter the size of an array: 5
Enter the elements of an array: 56 156 2 349 321
56 156 2 349 321
56 156 2
56 156
minval: 56
maxval: 156

2
minval: 2
maxval: 2

minval: 2
maxval: 156

349 321
minval: 321
maxval: 349

minval: 2
maxval: 349

The minimum element of an array is 2
The maximum element of an array is 349

Process returned 0 (0x0)   execution time : 23.445 s
Press any key to continue.
```

Conclusion:

1] I implemented Min-max algorithm for finding out the minimum and maximum value and understood how divide and conquer method is better than the usual Naïve Bayes method.