# Group 51- Given a large set of unsorted integers, find the maximum and minimum element using divide and conquer approach.

IHM2016501[1] IHM2016005[2] BIM2016004[3] IIM2016006[4] RIT2015050[5]

***Abstract*— We worked on an efficient algorithm to find the maximum and minimum elements in a large set of unsorted integers using divide and conquer.We have successfully reduced the complexity of the algorithm when compared to the Brute force approach.The result obtained are the minimum and maximum elements of the given large set of unsorted integers.**

## I. INTRODUCTION AND LITERATURE SURVEY

Divide and conquer is an algorithm design paradigm based on multi-branched recursion. A divide and conquer algorithm works by recursively breaking down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly. The solutions to the sub-problems are then combined to give a solution to the original problem.

So,here in the question we are given a large unsorted integers .We noe store these integers in an array of size $n$ and we have to write an efficient algorithm in-order to find the the maximum and minimum element in it .

## II. ALGORITHM DESIGN

This part of the report can be further divided into following sections.

### *A. Input Format*

In first few lines of the code we use $rand()$ function to generate a number randomly and assign it to the variable $n$ which defines the size of the array.We then fill this array $a[]$ with randomly generated numbers.So,we now have a array $a[]$of size $num$ filled randomly generated numbers.

- Also the array $a[]$and variables max,min are declared globally.

### *B. Algorithm*

**Algorithm 1** MaxMin

```
1:  Input Two integers i and j
2:  Initialization max1,min1,mid
3:  procedure MAXMIN(i, j)
4:      if i==j then
5:          max = a[i]
6:          min = a[i]
7:      else
8:          if i==j-1 then
9:              if a[i] < a[j] then
10:                 max = a[j]
11:                 min = a[i]
12:             else
13:                 max = a[j]
14:                 min = a[i]
15:         else
16:             mid = (i + j)/2
17:             maxmin(i, mid)
18:             max1 = max
19:             min1 = min
20:             maxmin(mid + 1, j)
21:             if max < max1 then
22:                 max = max1
23:             if min > min1 then
24:                 min = min1
```

**Algorithm 2** Main Function

```
Initialize i, n
procedure MAIN                          ▷ main function
    for i ← 0 to n do
        scan(a)
      max = a[0] min = a[0]
    maxmin(0, n − 1);
```

### C. Output Format

As mentioned in the question we are given large set of unsorted integers, we are finding the maximum and minimum element using divide and conquer approach.We will output the maximum and minimum element.

### D. Description

- **Divide** the problem into a number of subproblems .There must be base case (to stop recursion).
- **Conquer** (solve) each subproblem recursively.
- **Combine** (merge) solutions to subproblems into a solution to the original problem
- Split the array in half which is done with the help of MAXMIN() function.
- Find the MAX and MIN of both halves
- Compare the 2 MAX 's and compare the 2 MIN 's to get overall MAX and MIN.

## III. ANALYSIS

### A. Analysis

*1) Worst Case Analysis :* For the worst case analysis we will mainly focus on the number of comparisons and ignore all other operations because the number of comparisons is the facto which is to be reduced to obtain a efficient solution.

- $t_{worst}$(n) = 1 for n$\leq$2 (as we just need to compare once and there is no need to recursively call the function.
- $t_{worst}$(n) = 2 *($t_{worst}$(n/2))+2 for n>2 (as we need to recursively call the function)
- If we consider the stage of the initial array as $level$ 0 then to solve the question we will have to divide the array in $k$ levels where $k = log_2(n/2)$.

$\therefore t_{worst}(n) \propto 2^k t_{worst}(n/2^k)$+$\sum_{n=1}^{k} 2^n$

- But we already know that $(n/2^k)$ =2.
- Therefore,he above equation reduces as $t_{worst}(n) \propto 3 * (2^k) - 2$

$\implies t_{worst}(n) \propto 3 * (n/2) - 2$

So,Worst case time complexity is in the order of n i.e,$O(n)$

*2) Best Case Analysis:* The algorithm will have to go through all the above mentioned procedure and it doesn't depend on the input.i.e, Whatever the input is we have to divide it and find the minimum and maximum elements in each part of it irrespective of the input.So,we can conclude that $t_{best} = t_{worst}$.

$\implies t_{best}(n) \propto 3 * (n/2) - 2$

So,Best case time complexity is in the order of n i.e,$O(n)$

*3) Average Case Analysis :* We know that $t_{best} \leq t_{avg} \leq t_{worst}$. But we also know that

$t_{best} = t_{worst}$.

$\implies t_{best} = t_{avg} = t_{worst}$

$\implies t_{avg}(n) \propto 3 * (n/2) - 2$

So,Average case time complexity is $O(n)$

### B. Comparison with Brute Force Approach

- In the normal Brute force method we compare each and every element with the other to find minimun and maximum elements.
- So to find maximum element we need $n - 1$ comparisons.
- So to find minimum element we need one less comparison than we needed to find the maximum element i.e, $n - 2$ comparisons.
- So in total we need $2 * n - 3$ comparisons.
- So, $t_{bruteforce} \propto (2 * n - 3)$
- But we have already proved that $t_{divideandconquer} \propto 1.5 * (n) - 2$.

- Hence,the algorithm we have followed is efficient.

## IV. EXPERIMENTALSTUDY

The best way to study an algorithm is by graphs and profiling.

*1) Graphs-Time-Complexity:* From the graph it is clear that our algorithm takes same amount of time in all the cases i.e, best case and also in average and worst cases.We have obtained a linear graph i.e, it is proportional to $O(n)$ in all the cases.We can also see that the lower bound of it occurs when size of the array(n) is in the form of $2^k$ (k - positive integer).As though the time taken increases with the size of the set of given unsorted integers the rate at which it increases is not constant sometimes the slope is more and sometimes its less but the lower bound occurs when size of the set of given unsorted integer is in the form of $2^k$ (k - positive integer).(Here, the number of instructions taken is directly proportional to the time taken,We also consider that basic instructions take unit amount of time).

**Graphs-Time-Complexity**

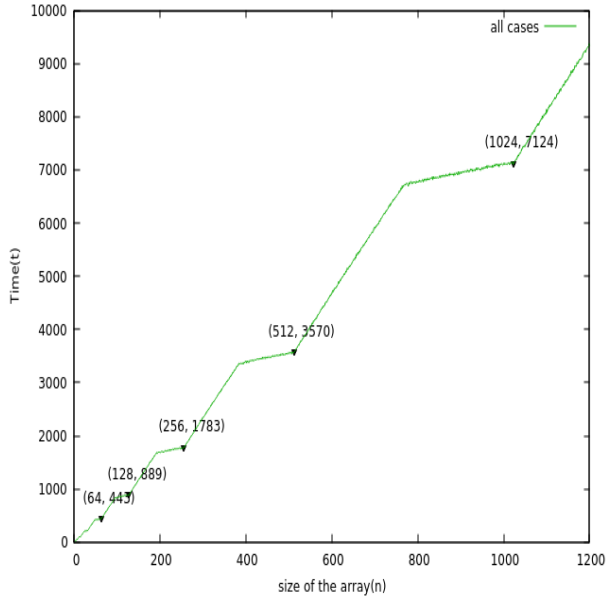


Fig. 1. size of the array(n) vs Time(t)

*2) Profiling:* $No$ of $Computations$

| $n$ | $t_{avg}$ | $t_{best}$ | $t_{worst}$ |
|---|---|---|---|
| 2 | 7 | 7 | 7 |
| 4 | 21 | 21 | 21 |
| 10 | 73 | 73 | 73 |
| 39 | 301 | 301 | 301 |
| 70 | 513 | 513 | 513 |
| 140 | 1033 | 1033 | 1033 |
| 250 | 1773 | 1773 | 1773 |
| 1200 | 9273 | 9273 | 9273 |

TABLE I
SIZE OF THE ARRAY(N) VS TIME(T)($t_{avg}, t_{best}, t_{worst}$)

- We can also see that time increases as the value of n increases and $t_{worst} \propto n$,$tavg \propto n$,$t_{best} \propto n$.

## V. DISCUSSIONS

### A. Understanding the Use Of File Handling and generation of random numbers

In this context the file handling is used to generate the random test cases through which analysis and experimentation of algorithm becomes easier for handling different test cases. We have used $srand()$ and $rand()$ to generate the random values to fill the matrix with the help of $< time.h >$ header file.

## VI. CONCLUSION

- While plotting the graphs we have taken the upper limit of the size of the array to 1200 as it is sufficient enough to Analise the graph and draw conclusions.
- It is show clearly in the analysis that we have optimized the algorithm when compared to the brute force method.
- By making the proper analysis of the algorithm and optimizing the code we can conclude that the order of time taken by the algorithm is not dependent on the input .So we conclude that $t_{worst} = t_{best} = t_{avg}$

## REFERENCES

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.
[2] Introduction to Algorithms English By Thomas H. Cormen , By Charles E. Leiserson , Ronald L. Rivest ,Clifford Stein(3rd edition)