**Experiment No:2**

**Title:-** Write efficient algorithm to find minimum and maximum of given numbers.

**Theory:**

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]

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)

Analysis

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

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)**

if y – x ≤ 1 then

return (max(numbers[x], numbers[y]), min((numbers[x], numbers[y]))

else

(max1, min1):= maxmin(x, ⌊((x + y)/2)⌋)

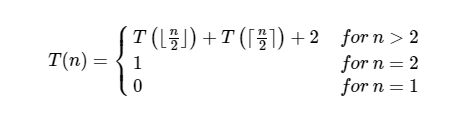
(max2, min2):= maxmin(⌊((x + y)/2) + 1)⌋,y)

return (max(max1, max2), min(min1, min2))

**Analysis**

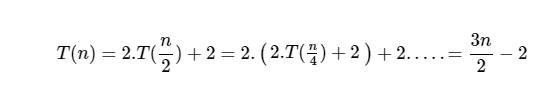
Let ***T(n)*** be the number of comparisons made by Max−Min(x,y)Max−Min(x,y), where the number of elements n=y−x+1

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



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

So,



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)**.