Check weather an integer repeated more than n/2 times in sorted array

DAA ASSIGNMENT-4 , GROUP 1

Harsh Mahajan
IIB2019001

Pradhuman Singh Baid
IIB2019002

Tauhid Alam
BIM2015003

***Abstract*—This Paper contains the algorithm to find if a given integer x appears more than n/2 times in a sorted array of n integers. Divide and conquer approach is taken to solve it.**

I. Introduction

Divide and conquer is the way of solving a problem by dividing the problem into smaller subproblems.

It is a technique in which the main problem is divided into smaller subproblems and then solving the smaller subproblems and combining them to find the solution of the problem.

In this paper, we have to find if a given integer x appears more than n/2 times in a sorted array of n integers

II. Algorithmic Design

In the problem we need to find weather a given integer x in appearing more than n/2 times in a sorted array or not.Efficiently this problem can be solved using binary search which is a divide and conquer algorithm.

First We divided the array in two parts and checked the middle element. If this element is greater than or equal the integer x then we would focus on the left sub-array otherwise right sub-array.

Now, The size of focused sub-array is reduced to half of original array.

And then we will recursively apply this approach on the focused array until its size is reduced to 1.

When the size of resultant array is '1' it means that we are on the index where the integer x appeared 1st time. Let name that index as 'left'.

Now,we need to check weather it appeared more than n/2 times or not.For that we would add n/2 to 'left' and name the result 'right'.As it is provided that array is sorted and if 'x' appeared more than n/2 times then it must also present at index 'right'.

And finally we will print the result according to its presence on index 'right'.

III. Pseudo Code

```
Array ← sorted array of size n
target ← Target integer 'x'

solve function
    long long int mid ← middleEl
    long long int left ← 0
    long long int mid ← n
    while left <right
        mid = left + (left+right)/2
        if (target <= Array[mid])
            right = mid
        else
            left = mid + 1


    right = left + n/2
    if(Array[right]=target)
        PRINT "YES"
    else
        PRINT NO


main function
    call solve function
end procedure
```

IV. Complexity Analysis

A. Time Complexity

In this algorithm at each iteration the array is divided by 2. Suppose length of array is n and after K iteration the length of array becomes 1 and then we get

$n/2^k = 1$

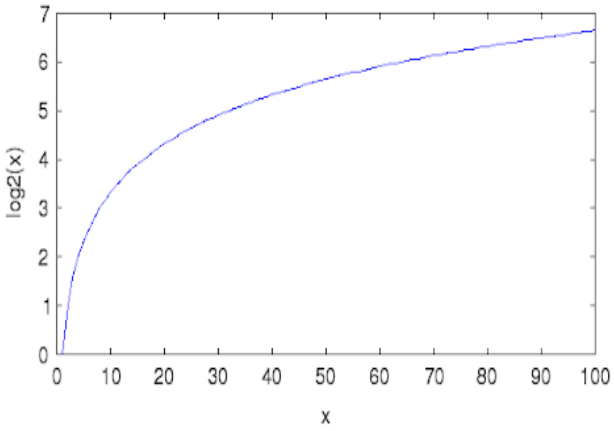
$n = 2^k$

$log(n) = log(2^k)$

$log(n) = klog(2)$

$k = log(n)$

Therefore, Time complexity is O(nlog(n))



B. Space Complexity

In this method, the space complexity would be O(n) as we have used an array of size n and some variables of constant space.

Therefore, Space complexity is O(n)

V. Conclusion

Through this assignment we conclude that fastest way to find an element is using divide and conquer alogrithm.

VI. REFERENCES

https://en.wikipedia.org/wiki/Divide-and-conquer$_a lgorithm$
https://www.geeksforgeeks.org/divide-and-conquer-algorithm-i