Design And Analysis Of Algorithms- ASSIGNMENT 7
Group-2

ADITYA RAJ(IIT2017005)
MAYANK MRINAL(IIT2017006)
MOHIT PAHUJA(IIT2017007)
ARASHPREET SINGH(IIT2017008)

***Abstract*— This document is complete design and analysis of algorithm of searching an element 'x' using divide and conquer and comparing it with linear search.**

I. INTRODUCTION

In this problem we have search element x using divide and conquer strategy.We have used the binary search approach. First we find the middle element , then we check whether it is equal to that element or less or greater.

This report further contains::
II. Algorithm Design.
III.Algorithm Analysis
IV.Experimental Study.
V.Conclusions.
VI.References.

II. ALGORITHM DESIGN

A. Algorithm 1

Binary search is the most popular Search algorithm.It is efficient and also one of the most commonly used techniques that is used to solve problems.

Binary search works only on a sorted set of elements. To use binary search on a collection, the collection must first be sorted.

When binary search is used to perform operations on a sorted set, the number of iterations can always be reduced on the basis of the value that is being searched.
Search a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise narrow it to the upper half. Repeatedly check until the value is found or the interval is empty.

Algorithm 1

```
int t←0
int binarySearch(int arr[], int l, int r, int x)
{
if r≥ l then
    t++
  int mid←l + (r - l) / 2
  t←t+4
  if arr[mid]==x then
      t++
    return mid
    else
    end
    if arr[mid]¿x then
      t++
    return binarySearch(arr, l, mid - 1, x)
    else
    end
    return binarySearch(arr, mid + 1, r, x)
  else
  end
  return -1
}
int main()
{
int arr,x
int result←binarySearch(int arr, 0,n-1, x)
t++
print←result
}
```

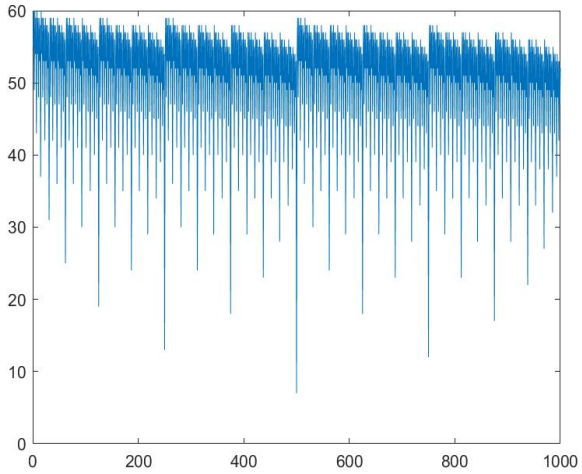
III. ALGORITHM ANALYSIS

A. Time Complexity

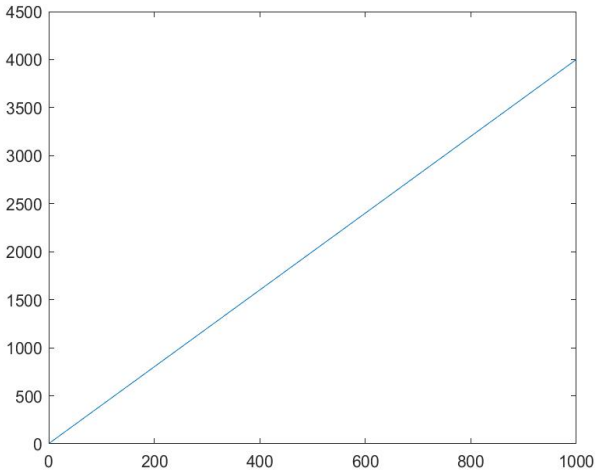
- *We have calculated the time consumed in each step in terms of number of computations,as given in the above pseudo code.*
- *Thus for a general value of x to be searched from an array of length N , we have the worst case time complexity using this algorithm is of order $O(logN)$ of base 2.*
- *The best case time complexity will be $O(1)$ because the searched element may be the middle element.*
- *Also the time complexity in linear search will be of order $O(N)$ for all the cases.*

B. Space Complexity

- *Since the number of memory location depends upon the size of the array lets say N therefore the space complexity will be of order $O(N)$.*



Thus the graph depicts, our calculation, using Binary Search, where x axis is the element to be searched in an array of element [1-1000] and y axis is for time.



The graph depicts, our calculation, using Linear Search, where x axis is the element to be searched in an array of element [1-1000] and y axis is for time.

IV. EXPERIMENTAL STUDY

The integrated Development environment of C++ is used for processing the algorithm and graphical analysis is done using MATLAB plot function.

V. CONCLUSIONS

In this document we conclude that our algorithm has a non-polynomial with worst case of order $O(logN)$ where N is the size of the given array.
The space complexity is of the order $O(N)$(where N is SIZE of the array)
. Also the graph depicts that the divide and conquer algorithm is more optimised than the linear search algorithm.

VI. REFERENCES

1.[CLR96] Thomas H. Cormen, Charles E. Leiserson, and Ronald L Rivest. Introduction to algorithms. The MIT press, 2nd edition, 1996.
2.[DW96] Nell Dale and Henry M. Walker. Abstract data types: specifications, implementations, and applications. D. C. Heath and Company, Lexington, MA, USA, 1996.