DAA C-2 Assignment-1

Frequency of x in Sorted array using Divide and conquer algorithm.

Sarthak IIT2019114
Kaustubh Gangwar IIT2019115
Harsh Singh IIT2019116

Indian Institute of Information Technology
Allahabad Uttar Pradesh 211015

March 22, 2021

Overview

Introduction

- Count number of occurrences (or frequency) of x in a given sorted array arr[] using divide and conquer algorithm.
- Divide and Conquer is an algorithm design paradigm in which we recursively break down a problem into two or more sub-problems of the same or related type, until these subproblems become simple enough to be solved directly.
- The solutions to the sub-problems are then combined to give a solution to the original problem.
- Binary search is a search algorithm that finds the position of a target value within a sorted array.
- Binary search compares the target value to the middle element of the array. If they are not equal, the half in which the target cannot lie is eliminated and the search continues on the remaining half, again taking the middle element to compare to the target value, and repeating this until the target value is found.

Algorithm

Algorithm Description

Since the array is sorted, all occurrences of x will be adjacent.
So, the steps used in this algorithm are :
1) Use Binary search to get index of the first occurrence of x in arr[]. Let the index of the first occurrence be i.
2) Use Binary search to get index of the last occurrence of x in arr[]. Let the index of the last occurrence be j.
3) Return (j – i + 1); //And This will be the frequency of x.

Algorithm

Algorithm Analysis

As the given array is already sorted so we will find the first occurrence of x in arr[] using the function search() by passing the array arr[] , n i.e. size of array , x and 1.
If the returned value is -1 that means the element is not present in array and we will stop our algorithm here and if it is not -1 then we will find the last occurrence of x in array arr[] by using search() function again but this time we will pass 0 instead of 1 in flg value to get the last occurrence of x.
So, after getting the first and last occurrence of x in arr[] , the number of occurrence of x will be last - first + 1.

Algorithm

Pseudo Code

```
PSEUDO CODE for main() :
int first = search(arr, n, x, 1);
if (first != -1)
int last = search(arr, n, x, 0);
int count = last - first + 1;
cout<<"Element "<<x<<" occurs "<<count<<" times.";
else
cout<<"Element not found in the array.";
```

Algorithm

```
int low = 0;
int high = N - 1;
int result = -1;
while (low <= high) :
        int mid = (low + high)/2;
        if (x == arr[mid])
                result = mid;
                if (flg)
                        high = mid - 1;
                else
                        low = mid + 1;
        else if (x < arr[mid])
                high = mid - 1;
        else
                low = mid + 1;
```

TIME COMPLEXITY

While Finding the First and Last Occurrence of x if the complexity of the entire Algorithm is T(n) where n signifies number of elements, we are actually reducing the list to half i.e T(n/2) in every step of the loop along with some constant time (O(1) for reference) needed to find middle index and do comparisons.
Hence Time Complexity of Binary Search :

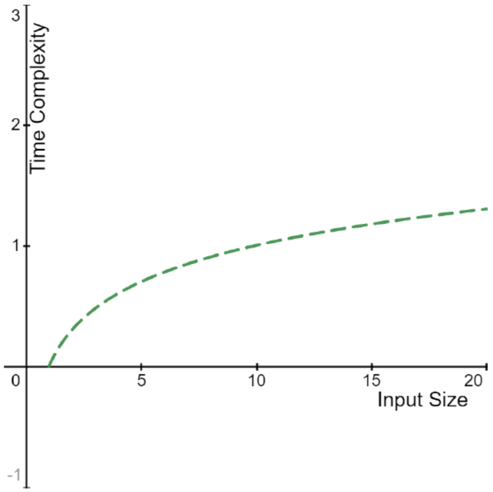
$$\mathrm{T(n)} = \mathrm{T(n/2)} + \mathrm{O(1)}$$

Let say after k divisions, the size of search space becomes 1.
Therefore size of search space = $n/2^k = 1$
Applying log function on both sides:

$$\mathrm{k} = log(n)$$

Hence the complexity of finding freq of x is $log(n)$

Figure



SPACE COMPLEXITY

Space required for searching is O(1) because we just use some variables which come under O(1) or constant space complexity.
Only Space required is for storing the array arr[] of size n which will be O(n).

References

- Introduction to Algorithms by TH Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein.
- Binary Search : https://en.wikipedia.org/wiki/Binarysearchalgorithm
- Divide and Conquer : https://www.geeksforgeeks.org/divide-and-conquer/

The End