Algorithms and Data Structures

CSE 311

Fall 2020

Group 1 - Section 1

Assignment 3

Ahmed Moustafa AbdElfatah Mohamed Ramadan

Personal Email: geekahmed1@gmail.com

Problem 6: Searching Problem

Binary Search:

Algorithm Pseudo-Code:

```
Binary-Search-Iterative(L,V):
    left = 0
    right = L.size - 1
    while left <= right:
        mid = left + right / 2
        if L[mid] == V : return mid
        if L[mid] < V : left = mid + 1
        if L[mid] > V : right = mid - 1
    return Null
Binary-Search-Recursive(L,V, left, right):
    if left <= right:
        mid = left + right / 2
        if L[mid] == V : return mid
        if L[mid] > V : return Binary-Search-Recursive(L,V, left, mid - 1)
        return Binary-Search-Recursive(L,V, mid + 1, right)
    return Null
```

Analysis:

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.

We basically ignore half of the elements just after one comparison.

1. Compare v with the middle element.
2. If v matches with the middle element, we return the mid index.
3. Else If v is greater than the mid element, then v can only lie in the right half subarray after the mid element. So we recur for the right half.
4. Else (v is smaller) recur for the left half.

The time complexity of Binary Search can be written as: $T(n) = T(n/2) + c$

Using Master Theorem or Recurrence tree, we find that the running time is $O(\lg n)$

Java Implementation

Here you could find the implementation using JAVA:

https://drive.google.com/file/d/1ALpqwc3o9GeewwELlZfe6sxt7h7p8w3k/view?usp=sharing