|  |  |
| --- | --- |
| **Data Structures & Algorithms**  Diploma in IT / ISF  Year 2 (2017/18) Semester 4 | **Week 7** |
| **2 Hours** |
| **Tutorial 7 – Searching** | |

1. Given an ***unsorted array*** of numbers below,

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 48 | 25 | 95 | 76 | 57 | 12 | 33 | 88 | 63 | 82 |

1. How many comparisons are required to search for 57?

Ans: 5

1. How many comparisons are required to search for 35?

Ans: 10

1. Given a ***sorted array*** of numbers below,

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 12 | 25 | 33 | 48 | 57 | 63 | 76 | 82 | 88 | 95 |

1. How many comparisons are required to search for 57 using sequential search ?

Ans: 5

1. How many comparisons are required to search for 35 using sequential search ?

Ans: 4

1. How many comparisons are required to search for 57 using binary search ?

Ans: 1

1. How many comparisons are required to search for 35 using binary search ?

Log2^10 = 4

1. Write a ***recursive*** sequential search function to search for a target in a sorted array of integer numbers. The function header is given below.

**int search (int dataArray[], int n, int start, int target){**

**// base case**

**If (start > n-1) // item not found**

**Return -1;**

**Else{**

**If (dataArray[start] == target) //item found**

**Return start;**

**Else{**

**If (dataArray[start] > target) // item not found**

**Return -1;**

**Else{**

**Return Search(dataArray,n,start+1, target);**

**}**

**}**

**}**

**}**

Write a ***recursive*** binary search function to search for a target in a sorted array of integer numbers. The function header is given below.

**int binarySearch (int dataArray[], int first, int last, int target){**

**if (first > last) //base case**

**return -1;**

**else{**

**int mid = (first + last) / 2;**

**if (array[mid] == target)**

**return mid;**

**else**

**if (target < array[mid])**

**return binarySearch(array, first, mid-1,target)**

**else  
 return binarySearch(array,mid+1,last,target)  
}**

**}**

1. Discuss and compare the time efficiency between **sequential search** and **binary search** algorithm.

Sequential search requires going through and comparing the elements one after another in an array until a match is found.

The average case involves the match corresponding to the middle element, in which case n/2 number of comparisons are required.

The worst case involves either the math occurs at the last element, or no match is found, in which case n comparisons are required.

Therefore, time complexity of sequential search in both average and worst case is O(n).

Binary Search involves narrowing the scope of the problem by half with every iteration/recursive step, i.e. originally there are n elements to deal with, then after first iteration n/2 elements will remain, followed by n/4 and so on.

Therefore, complexity of binary search is O(log2n).