## Experiment No: 1

***Title:*** Implementation of Binary search algorithm using Divide & Conquer method.

##### Theory/Description:

Binary search can be performed on a sorted array. In this approach, the index of an element **x** is determined if the element belongs to the list of elements. If the array is unsorted, linear search is used to determine the position.

In this algorithm, we want to find whether element **x** belongs to a set of numbers stored in an array ***numbers[]***. Where ***l*** and ***r*** represent the left and right index of a sub-array in which searching operation should be performed.

**Algorithm: Binary-Search (numbers[], x, l, r)**

if l = r then return l else

m := ⌊(l + r) / 2⌋

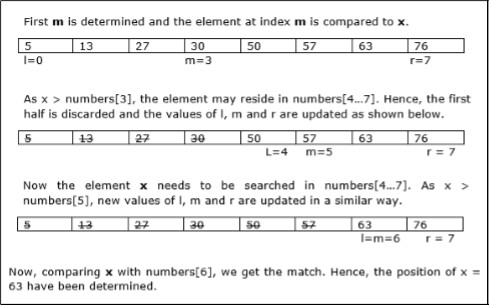
if x ≤ numbers[m] then

return Binary-Search(numbers[], x, l, m) else

return Binary-Search(numbers[], x, m+1, r)

**Example :**

In this example, we are going to search element 63.



*Program:*

*#include <stdio.h>*

*int main()*

*{*

*int c, first, last, middle, n, search, array[100];*

*printf("Enter number of elements\n"); scanf("%d",&n);*

*printf("Enter %d integers\n", n); for (c = 0; c < n; c++)*

*scanf("%d",&array[c]);*

*printf("Enter value to find\n"); scanf("%d", &search);*

*first = 0; last = n - 1;*

*middle = (first+last)/2;*

*while (first <= last) {*

*if (array[middle] < search) first = middle + 1;*

*else if (array[middle] == search) {*

*printf("%d found at location %d.\n", search, middle+1); break;*

*}*

*else*

*last = middle - 1; middle = (first + last)/2;*

*}*

*if (first > last)*

*printf("Not found! %d isn't present in the list.\n", search); return 0;*

*}*

***Conclusion( must include analysis of program):***

#### Analysis : Linear search runs in *O(n)* time. Whereas binary search produces the result in *O(log n)* time. Let T(n) be the number of comparisons in worst-case in an array of nelements.

Hence,T(n)={0 fn=1

T(n/2)+1 otherwise

Using this recurrence relation T(n)=(logn) Therefore, binary search uses O(logn)time.