Experiment No: 1

BINARY SEARCH USING DIVIDE AND CONQUER

Aim: To implement Binary Search using Divide and Conquer.

Theory:

Divide and Conquer:

In divide and conquer approach, the problem in hand, is divided into smaller sub-problems and then each problem is solved independently. When we keep on dividing the sub problems into even smaller sub-problems, we may eventually reach a stage where no more division is possible. Those "atomic" smallest possible sub-problem (fractions) are solved. The solution of all sub-problems is finally merged in order to obtain the solution of an original problem.

Here are the steps involved:

1. Divide: Divide the given problem into sub-problems using recursion.
2. Conquer: Solve the smaller sub-problems recursively. If the sub problem is small enough, then solve it directly.
3. Combine: Combine the solutions of the sub-problems that are part of the recursive process to solve the actual problem.

Binary Search

Binary search is a fast search algorithm with run-time complexity of O(log n). This search algorithm works on the principle of divide and conquer. For this algorithm to work properly, the data collection should be in the sorted form.

Binary search looks for a particular item by comparing the middle most item of the collection. If a match occurs, then the index of item is returned. If the middle item is greater than the item, then the item is searched in the sub-array to the left of the middle item. Otherwise, the item is searched for in the sub-array to the right of the middle item. This process continues on the sub-array as well until the size of the sub array reduces to zero.

Algorithm(Iterative)

```
int BinarySearch(int arr[], int size, int item)
{
int low=0, high=size-1, mid;
while(low<=high)
{
        mid = (low+high)/2;
        if(item>arr[mid])
                low = mid+1;
        else if(item<arr[mid])
                high = mid-1;
        else
                return mid;
}
return -1;
}
```

Algorithm(Recursive)

```
int RBinarySearch(int arr[], int key,int low, int high)
{
int mid;
if(low>high)
        return -1;
mid=(low+high)/2
if(x==a[mid])
{
        return (mid);
}
else if(x<a[mid])
{
        RBinarySearch(arr, key, low, mid-1);
}
else
{
                RBinarySearch(arr, key, mid+1, high);
}

}
}
```

PROGRAM OUTPUT:

1) **Binary Search using Iteration**
2) **Binary Search using Recursion**

Conclusion: Binary Search using Divide and Conquer was studied and implemented successfully.