Lab 3: Implement and analyze Binary Search using Divide and Conquer approach.

Theory: A divide-and-conquer algorithm recursively breaks down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly. The solutions to the sub-problems are then combined to give a solution to the original problem.

Algorithm for Binary Search

```
bSearch(A, arrayStart, arrayEnd,key)
{
        flag = 0;
        if (arrayStart<=arrayEnd)
        {
                m=(arrayStart+arrayEnd)/2;
                if (A[m]==key)
                        flag = m;
                else if (a < A[m])
                        return bSearch(A, arrayStart, m-1, key);
                else
                        return bSearch(A, m+1, arrayEnd, key);
        }
        return flag;
}
```

Analysis:

Time Complexity: O ($\log_2 n$)

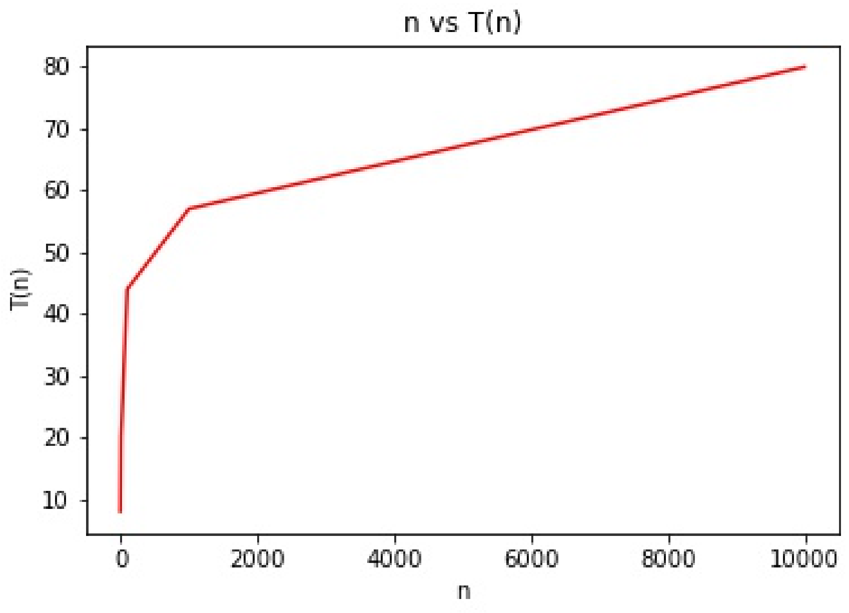
Space Complexity: O (n)

Source Code

```
#include <iostream>
#define MAX 30000
int t = 0;
using namespace std;
void bSearch(int A[], int l, int r, int key)
{
    int flag = -1, m;t++;
    t++;
    if(l<=r)
    {
        m = (l+r)/2;t++;
        t++;
        if(key == A[m])
            {
                flag = m;t++;
            }
        else if(key < A[m])
        {
            t++;
            t++;
            return bSearch(A, l, m-1, key);
        }
        else
        {
            t++;
            return bSearch(A, m+1, r, key);
        }
    }
    t++;
    if(flag == -1)
        cout<<"Search Unsuccessfull"<<endl;
    else
    {
        cout<<"Search Successfull. Element found at index
"<<flag<<endl;t++;
    }
}
```

```
int main()
{
    int A[MAX], n, i, key;
    cout<<"How many elements? ";
    cin>>n;
    for(i=0; i<n; i++)
        A[i] = rand();
    cout<<n<<" elements generated"<<endl;
    cout<<"Key ? ";
    cin>>key;
    bSearch(A,0,n-1,key);
    cout<<"T(n) = "<<t<<endl;
    return 0;
}
```

n	1	10	100	1000	10000
T(n)	8	20	44	57	80



Conclusion: Hence, in this lab, we successfully implemented the divide and conquer based binary searched algorithm. We also analyzed the time and space complexity of this algorithm.