

Practical No. :- 01

Implement binary search algorithm and compute its time complexity.

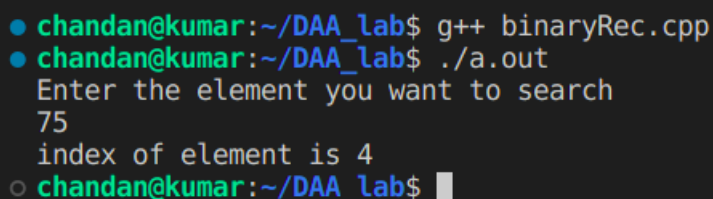
Code->

```
#include <bits/stdc++.h>
using namespace std;
int binarySearch(int arr[], int x, int low, int high)
{
    if (low <= high)
    {
        int mid = (low + high) / 2;
        if (arr[mid] == x)
            return mid;
        else if (x > arr[mid])
        {
            return binarySearch(arr, x, mid + 1, high);
        }
        else
        {
            return binarySearch(arr, x, low, mid - 1);
        }
    }
    return -1;
}

int main()
{
    int arr[] = {12, 32, 45, 67, 75, 79, 90};
    int size = sizeof(arr) / sizeof(arr[0]);
    cout<<"Enter the element you want to search"<<endl;
    int x;
    cin>>x;
    cout << "index of element is " << binarySearch(arr, x, 0, size - 1) << endl;

    return 0;
}
```

output:-



```
● chandan@kumar:~/DAA_lab$ g++ binaryRec.cpp
● chandan@kumar:~/DAA_lab$ ./a.out
Enter the element you want to search
75
index of element is 4
○ chandan@kumar:~/DAA_lab$
```

Analysis of algorithm:-

Binary Search [Recursive analysis]

After the first call length of array becomes $\frac{n}{2}$

After the second call length of array becomes $\frac{n}{2} \times \frac{1}{2} = \frac{n}{2^2}$

⋮

After k^{th} call length of array becomes $\frac{n}{2^k}$

$k \uparrow$ then value of $\frac{n}{2^k}$ becomes 1

$$\frac{n}{2^k} = 1$$

$$2^k = n$$

Taking log on both side

$$\log_2 2^k = \log_2 n$$

$$k \log_2 2 = \log_2 n$$

$$[\log_a a = 1]$$

$$k = \log_2 n$$

so the time complexity = $O(\log_2 n)$