Binary Search

Implementation:

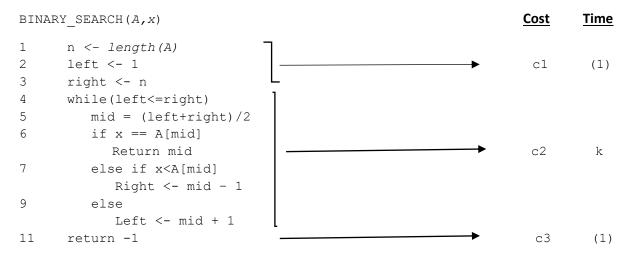
```
#include<stdio.h>
int binarysearch(int arr[],int n,int x)
  int i,left,right,mid;
  left = 0;
  right = n-1;
  while(left<=right)
    mid = (left+right)/2;
    if(x==arr[mid])
    {
       return mid;
    else if(x<arr[mid])
       right = mid-1;
    }
    else
       left = mid+1;
  }
  return -1;
int main()
  int n,i,res,x;
 scanf("%d",&n); //how many values in the list?
  int arr[n];
  for(i=0; i<n; i++)
    scanf("%d",&arr[i]);
  scanf("%d",&x); //value that want to search
  res = binarysearch(arr,n,x);
if(res==-1)
```

```
{
    printf("Value not found in the list\n");
}
else
{
    printf("Value found\nThe position of the value is %d\n",res);
}
return 0;
```

Analysis:

We start analyzing the Insertion Sort procedure with the time "cost" of each statement and the number of times each statement is executed.

pseudo code of binary search:



The running time of the algorithm is the sum of running times for each statement executed; a statement that takes c_i steps to execute and executes n times will contribute $c_i * n$ times to the total running time.

Considering the algorithm of binary search,

$$T(n)=c1+c2*k+c3$$
As $c1$, $c2$ and $c3$ are some constant then this equation can be written as, $T(n)=c*k+c'$
Imagine an array of having 16 elements as follows:

1 2 3 4 5 6 7 8 9 10 11 12 13 14	15 1	15	15	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
----------------------------------	------	----	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	--

The value we want to search is 16 which is in the last position of the array.

According to the algorithm,

left = 0 right = 16

Entering the while loop the n will be divided as follows to find the value 16:

Mid = 8 (left + right)/2

Now left = mid+1 = 9 and again, calculating the mid = 12 which is not equal to our desired value.

Again, Left = mid+1 = 13 and mid = 14 which is also not the searched value.

Again, left = mid+1 = 15; now mid = 15 which is also not the searched value.

Lastly, left = mid+1 = 16; now mid is also 16, which is the actual value we are looking for.

Therefore, we can say that if n = 16 we need 4 division of the array.
$$n \to \frac{n}{2} \to \frac{n}{2*2} \to \frac{n}{2*2*2} \to \frac{n}{2*2*2} \to \frac{n}{2*2*2*2}$$

If we continue for k times then n will be divided for, $\frac{n}{2^k}$ times.

Let,

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

$$2^{k} = 1$$

$$k = \log_2 n$$

Therefore, $T(n) = c * \log_2 n + c'$

Time Complexity:

Worst Case: The time function we get from the analysis is $T(n) = c * \log_2 n + c'$

So the complexity of binary search in worst case is O(log n)

Best Case:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

n = 16

x = 8 (search value)

In 1st iteration n will be divided for $\frac{n}{2}$ times.

Now mid = 8 which is the same as our search value. We need only one iteration to find the value.

So the complexity of binary search in best case is O(1)