Linear search

Implementation:

```
int linear_search(int arr[],int n,int x)
{
   int i,index=-1;
   for(i=0; i<n; i++)
   {
      if(arr[i]==x)
      {
         index = i;
         break;
      }
   }
   return index;
}
```

Analysis:

5	3	10	15	1

let consider this array having 5 elements that means n=5. We want to search the value x = 1

Step 1: i=0 and i<n which is true

Comparing x with arr[i] value which is 5 and 5 is not equal to 1.

5	3	10	15	1

Go to next step.

Step 2: i=1 and i<n which is true

Comparing x with arr[i] value which is 3 and 3 is not equal to 1.

5	3	10	15	1

Go to next step

Step 3: i=2 and i<n which is true

Comparing x with arr[i] value which is 10 and 10 is not equal to 1.

5	3	10	15	1

Go to next step

Step 4: i=3 and i<n which is true

Comparing x with arr[i] value which is 15 and 15 is not equal to 1.

5	3	10	15	1

Go to next step

Step 5: i=4 and i<n which is true

Comparing x with arr[i] value which is 1 and 1 is equal to 1.

5	3	10	15	1

We found the value and it is in the last position that is n-1.

After checking the condition if the value is found then it will break and return the index. But in this case the value we want to search that is in the last positon or last index of array which is 4.

For that reason, this loop will be executed for 5 times as n=5.

Worst Case:

If there are n elements and the value either exist in the last positon n-1 or not exist, the loop will run for n times.

Therefore, the complexity would be O(n).

Best Case:

5	3	10	15	1

If x = 5 which is in the begging of the array, the loop will run for only 1 time.

Therefore, for best case, the complexity would be O(1).

Average Case:

We know average case = $\dfrac{\text{All possible case time}}{\text{No. of cases}}$ (Till n)

$$= \frac{1+2+3\ldots\ldots+n}{n}$$

$$= \frac{\frac{n(n+1)}{2}}{n}$$

$$= \frac{n+1}{2}$$

Ignoring the constant co-efficient, we can say that the complexity in average case of linear search is O(n).