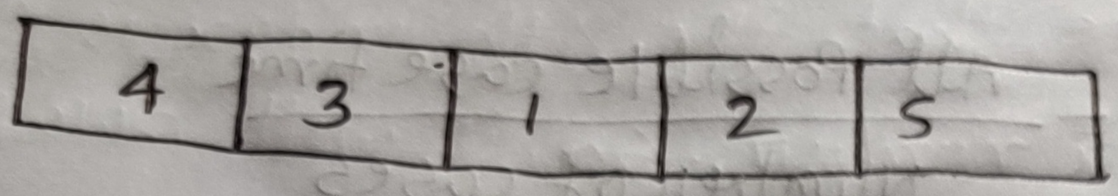
Linear search

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

Analysis



Here we have five array elements. So $n = 5$. From the array we want to search $x = 5$.

Let $i = 0$, which is less than n and start the loop. It will check every element in every loop. After checking the element, if it is not the desire loop then it will break and return the index. We search the value 5 which is on the 4th index of the array. So the loop will continue 5 times. Then it will find the value 5.

worst case

If the array has n elements and the value is n in the array on it is in the last Position $n-1$. Then the loops will run for n times. So the complexity would be $O(n)$.

Average case

$$\text{Average case} = \frac{\text{All Possible case time}}{\text{Number of cases}}$$

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

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

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

Ignoring the constants co-efficient, the complexity of average case is $O(n)$

Best case

5	3	1	2	4

If $n=5$, which is in the 1st index of array. The loop will run for 1 time.

So, the best case of complexity is $O(1)$.