➢ Linear Search: -

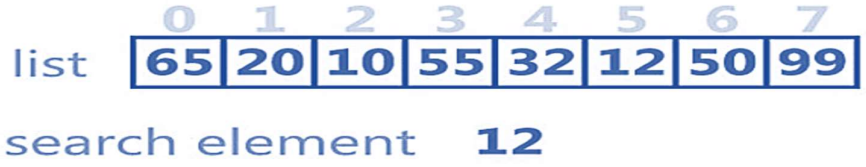
Linear search is a very simple search algorithm. In this type of search, a sequential search is made over all items one by one. Every item is checked and if a match is found then that particular item is returned, otherwise the search continues till the end of the data collection.

Linear search is implemented using following steps...

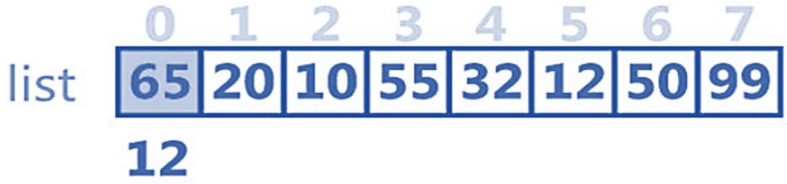
- **Step 1** - Read the search element from the user.
- **Step 2** - Compare the search element with the first element in the list.
- **Step 3** - If both are matched, then display "Given element is found!!!" and terminate the function
- **Step 4** - If both are not matched, then compare search element with the next element in the list.
- **Step 5** - Repeat steps 3 and 4 until search element is compared with last element in the list.
- **Step 6** - If last element in the list also doesn't match, then display "Element is not found!!!" and terminate the function.

Example: -

Consider the following list of elements and the element to be searched...

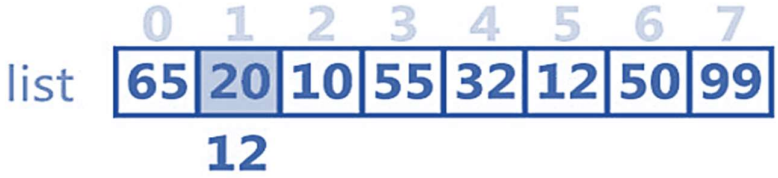


Step 1: Search element (12) is compared with first element (65)



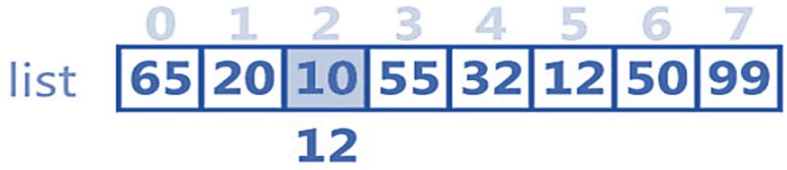
Both are not matching. So, move to next element

Step 2: Search element (12) is compared with next element (20)



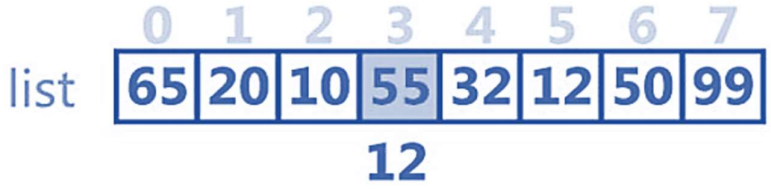
Both are not matching. So, move to next element

Step 3: Search element (12) is compared with next element (10)



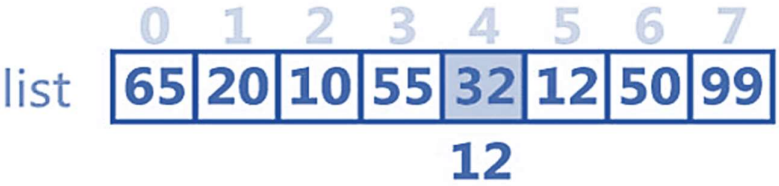
Both are not matching. So, move to next element

Step 4: Search element (12) is compared with next element (55)



Both are not matching. So, move to next element

Step 5: Search element (12) is compared with next element (32)



Both are not matching. So, move to next element

Step 6: Search element (12) is compared with next element (12)

Both are matching. So, we stop comparing and display element found at index 5

➢ Binary Search: -

Binary search algorithm finds a given element in a list of elements with O (log n) time complexity where n is total number of elements in the list. The binary search algorithm can be used with only a sorted list of elements. That means the binary search is used only with a list of elements that are already arranged in an order. The binary search cannot be used for a list of elements arranged in random order

Binary search is implemented using following steps...

- **Step 1** - Read the search element from the user.
- **Step 2** - Find the middle element in the sorted list.
- **Step 3** - Compare the search element with the middle element in the sorted list.
- **Step 4** - If both are matched, then display "Given element is found!!!" and terminate the function.
- **Step 5** - If both are not matched, then check whether the search element is smaller or larger than the middle element.
- **Step 6** - If the search element is smaller than middle element, repeat steps 2, 3, 4 and 5 for the left sublist of the middle element.

- **Step 7** - If the search element is larger than middle element, repeat steps 2, 3, 4 and 5 for the right sublist of the middle element.
- **Step 8** - Repeat the same process until we find the search element in the list or until sublist contains only one element.
- **Step 9** - If that element also doesn't match with the search element, then display "Element is not found in the list!!!" and terminate the function.

Example: -
Consider the following list of elements and the element to be searched...

Step 1: Search element (12) is compared with middle element (50)

Both are not matching. And 12 is smaller than 50. So, we search only in the left sublist (i.e.,10, 12, 20, 32)

Step 2: Search element (12) is compared with middle element (12)

Both are matching. So, the result is “Element found at index 1”

➢ Interpolation Search: -

Interpolation search is an improved variant of binary search. This search algorithm works on the probing position of the required value. For this algorithm to work properly, the data collection should be in a sorted form and equally distributed.
Binary search has a huge advantage of time complexity over linear search. Linear search has worst-case complexity of O(n) whereas binary search has O (log n).
There are cases where the location of target data may be known in advance. For example, in case of a telephone directory, if we want to search the telephone number of Morpheus. Here, linear search and even binary search will seem slow as we can directly jump to memory space where the names start from 'M' are stored.

Algorithm: -
As it is an improvisation of the existing BST algorithm, we are mentioning the steps to search the 'target' data value index, using position probing –

- **Step 1** − Start searching data from middle of the list.
- **Step 2** − If it is a match, return the index of the item, and exit.
- **Step 3** − If it is not a match, probe position.

- **Step 4** − Divide the list using probing formula and find the new middle.
- **Step 5** − If data is greater than middle, search in higher sub-list.
- **Step 6** − If data is smaller than middle, search in lower sub-list.
- **Step 7** − Repeat until match.

Positioning in Binary Search: -

Position Probing in Interpolation Search: -