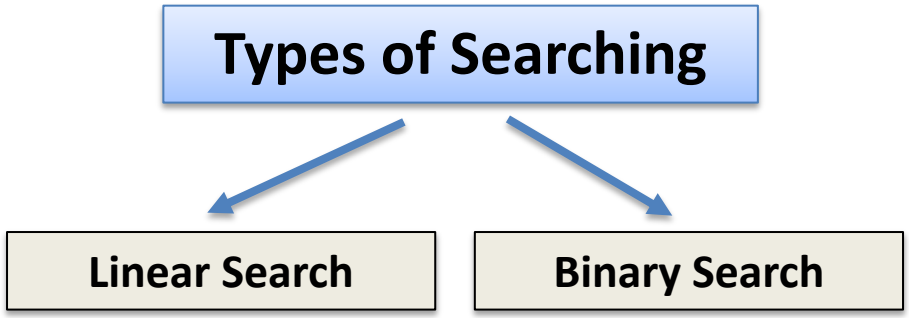
# Searching

Finding a particular element and its location in a data structure is known as searching. The search is said to be successful if element is present in the list otherwise search is said to be unsuccessful.

**For example**, checking whether **5** is present or not in the following list of numbers:

| 12 | 4 | 45 | 5 | 56 | 23 |
|---|---|---|---|---|---|

In above case **5** is present, hence searching is successful.



# Linear Search

This is the simplest method of searching.
In this method, the element to be found is sequentially searched in the list. This method can be applied to a sorted or unsorted list. In linear search, searching starts from first element and continues until element is found or end of list is reached.

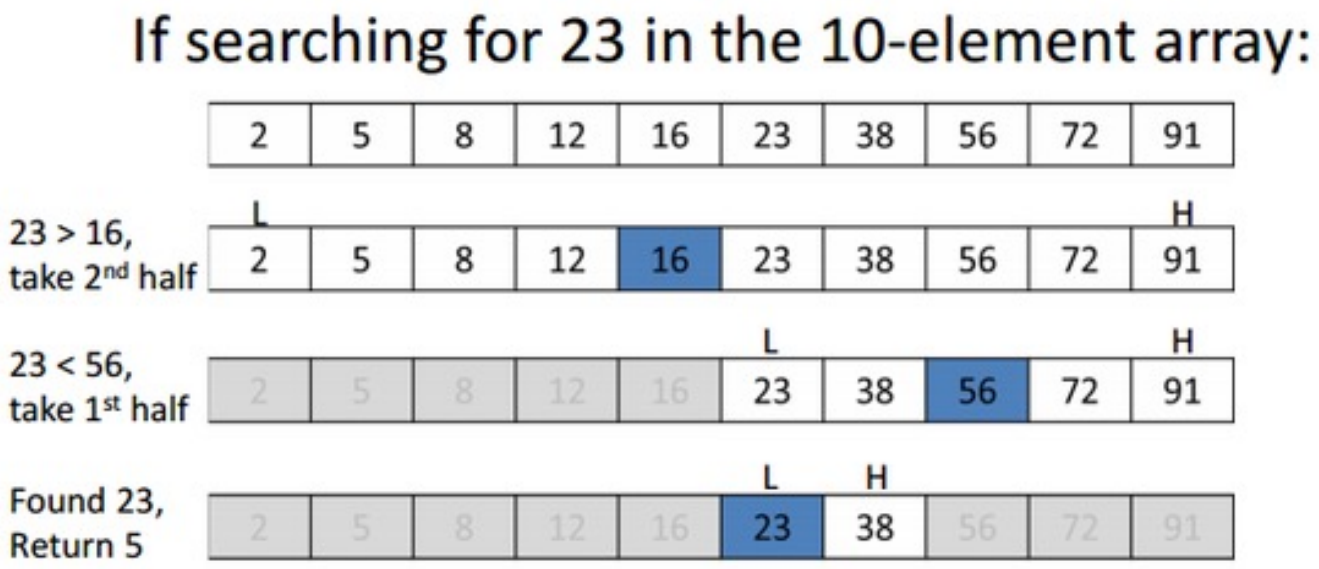
# Algo for Linear Search

Here **arr** is the array, **length** is the number of elements stored in the array, **value** will have the value to be searched in the array, and **flag** will be used to find value present or not, then the algorithm is:

1. Set flag=true.
2. Read value.
3. Repeat for i = 0 to length-1
    a. Check if value = arr[i], then:
    b. Print value is present in array at index i.
    c. Set flag=false.
    d. Break the loop.
4. Check if flag = true, then:
    a. Print value is not present in array.
5. Exit

# Binary Search

Binary search is very fast and efficient method of searching.
This method can be applied to a sorted collection of values only.



# Binary Search

In this method, the element is searched at middle element of the list. If its middle element of the list matches with the element to be searched, then search is said to successful. Otherwise, the list is divided into two halves: one from first element to the middle element (first half) and another from middle element to the last element. Because all elements of list are sorted, therefore all elements in first half are smaller than the middle element and all elements in second half are greater than the middle element. If the element to be searched is smaller than the middle element then searching for the element will be done in first half, otherwise searching for the element will be done in second half. This process of comparing element to be searched with middle element and dividing list into two halves is repeated until the element is found or division of two half parts gives one element.

# Algo for Binary Search

Here **arr** is the sorted array, **length** is the total number of elements stored in the array, **lower** will have the index of first element**, upper** will have the index of last element, **mid** will have the index of the middle element of array, **value** will have the number to be searched in the array, and **flag** will be used to find value present or not, , then the algorithm is:

1. Set flag=true.
2. Set lower = 0
3. Set upper = length – 1.
4. Read value.
5. Repeat while lower ≤ upper
    a. Set mid = (lower + upper) / 2.
    b. Check if value = arr[mid], then:
        i. Print value is present in array at index mid.
        ii. Set flag=false.
        iii. Break the loop.
    c. Otherwise, check if value < arr[mid], then:
        a. Set upper = mid – 1.
    d. Otherwise,
        a. Set lower = mid + 1.
6. Check if flag = true, then:
    a. Print value is not present in array.
7. Exit