# Northern University, Nowshera

## Application of Information and Communication Technologies (AICT)

**(Week 12) Lecture 23 & 24**

**Objectives:** **Learning objectives of this lecture are**

- **Searching**
- **Linear Search**
- **Binary Search**

**Text Book & Resources:** Introduction to Computers 6th International Edition, Peter, N. McGraw-Hill

# Searching

Searching is a process of locating a particular element present in a given set of elements. The element may be a record, a table, or a file.

Searching Algorithms are designed to check for an element or retrieve an element from any data structure where it is stored.

A search algorithm is an algorithm that accepts an argument 'a' and tries to find an element whose value is 'a'. It is possible that the search for a particular element in a set is unsuccessful if that element does not exist. There are number of techniques available for searching.

**Why do we need searching algorithms?**

We often need to find one particular item of data amongst many hundreds, thousands, millions or more. For example, you might need to find someone's phone number on your phone, or a particular business's address in the country.

This is why searching algorithms are important. Without them you would have to look at each item of data – each phone number or business address – individually, to see whether it is what you are looking for. In a large set of data, it will take a long time to do this. Instead, a searching algorithm can be used to help find the item of data you are looking for.

# Northern University, Nowshera

**Application of Information and Communication Technologies (AICT)**

## Linear Search

Linear search is used on a collections of items. It relies on the technique of traversing a list from start to end by exploring properties of all the elements that are found on the way.

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.

For example, consider an array of integers of size. You should find and print the position of all the elements with value. Here, the linear search is based on the idea of matching each element from the beginning of the list to the end of the list with the integer, and then printing the position of the element if the condition is `True'.

The time complexity of the linear search is O(n) because each element in an array is compared only once.

Linear search is rarely used practically because other search algorithms such as the binary search algorithm and hash tables allow significantly faster searching comparison to linear search.

**Example**

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

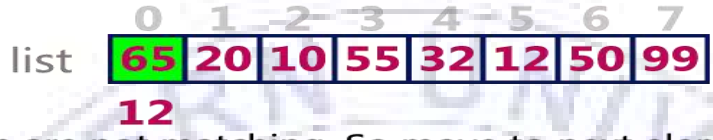
# Northern University, Nowshera

## Application of Information and Communication Technologies (AICT)


search element 12

**Step 1:**

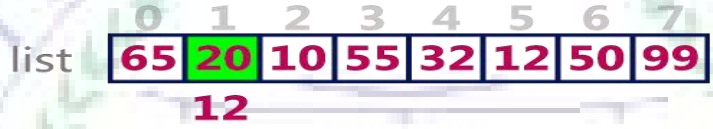
search element (12) is compared with first element (65)




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

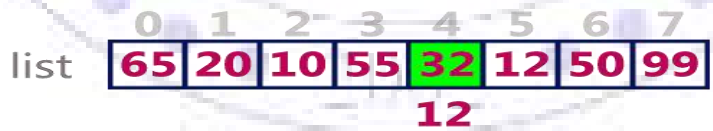
**Step 4:**

search element (12) is compared with next element (55)


Both are not matching. So move to next element

**Step 5:**

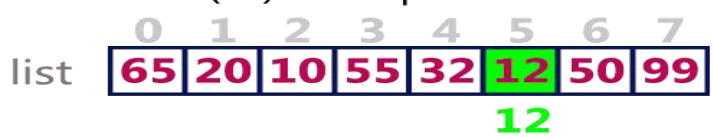
search element (12) is compared with next element (32)




Both are not matching. So move to next element

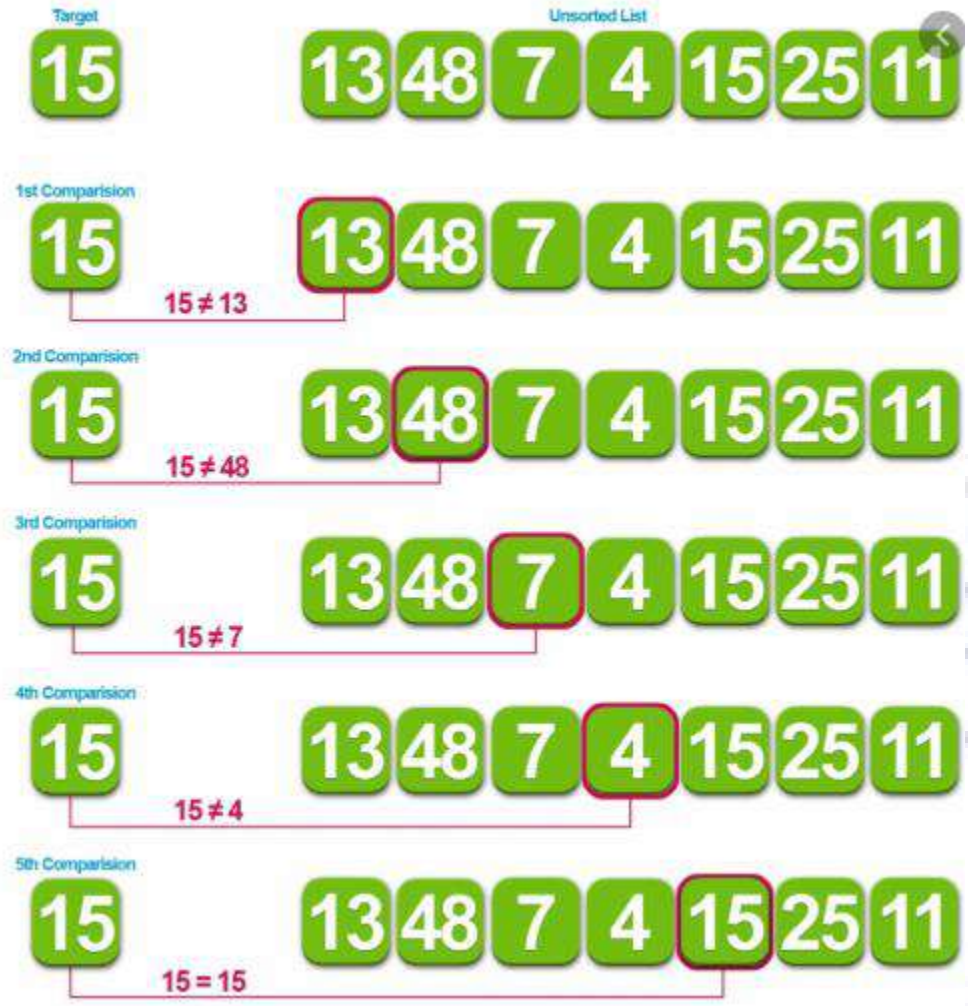
**Step 6:**

search element (12) is compared with next element (12)




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

**Another Example:**




Algorithm:

1. Take a value to be searched, from user
2. Set a variable 'i' to first index of array
3. Compare value with the element of list at index 'i'
4. If value found display message and goto step 8
5. Increment 'i' by one
6. If i less than n and goto step 3
7. Display not found
8. Stop

# Northern University, Nowshera

## Application of Information and Communication Technologies (AICT)

Pseudo code:

1. Start
2. Input x
3. i := 0
4. if x = a[i] ,
   display: found at index i goto 7
5. i++
6. if i < n goto 4
   display: value not found
7. stop

Value to search is 66

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 56 | 43 | 23 | 66 | 44 |
| 56 | 43 | 23 | 66 | 44 |
| 56 | 43 | 23 | 66 | 44 |
| 56 | 43 | 23 | 66 | 44 |

Web:
http://www.cs.armstrong.edu/liang/animation/web/LinearSearch.html

https://yongdanielliang.github.io/animation/web/LinearSearchNew.html

# Northern University, Nowshera

**Application of Information and Communication Technologies (AICT)**

## Binary Search

Binary search is the most popular Search algorithm. It is efficient and also one of the most commonly used techniques that is used to solve problems.

If all the names in the world are written down together in order and you want to search for the position of a specific name, binary search will accomplish this in a maximum of 35 iterations.

Binary search works only on a sorted set of elements. To use binary search on a collection, the collection must first be sorted.

When binary search is used to perform operations on a sorted set, the number of iterations can always be reduced on the basis of the value that is being searched.
Binary search looks for a particular item by comparing the middle most item of the collection. If a match occurs, then the index of item is returned. If the middle item is greater than the item, then the item is searched in the sub-array to the left of the middle item. Otherwise, the item is searched for in the sub-array to the right of the middle item. This process continues on the sub-array as well until the size of the sub array reduces to zero.

Binary search is an efficient algorithm for finding an item from a sorted list of items. It works by repeatedly dividing in half the portion of the list that could contain the item, until you've narrowed down the possible locations to just one.

Search a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise narrow it to the upper half. Repeatedly check until the value is found or the interval is empty.

Binary search is a fast search algorithm with run-time complexity of O(log n). This search algorithm works on the principle of divide and conquer. For this algorithm to work properly, the data collection should be in the sorted form.

# Northern University, Nowshera

## Application of Information and Communication Technologies (AICT)

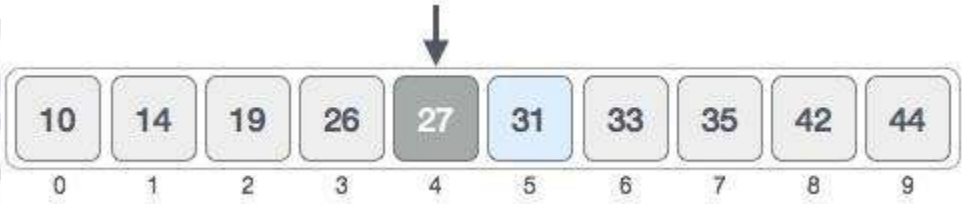
### How Binary Search Works?

For a binary search to work, it is mandatory for the target array to be sorted. We shall learn the process of binary search with a pictorial example. The following is our sorted array and let us assume that we need to search the location of value 31 using binary search.

First, we shall determine half of the array by using this formula:

mid = low + (high - low) / 2

Here it is, 0 + (9 - 0 ) / 2 = 4 (integer value of 4.5). So, 4 is the mid of the array.




Now we compare the value stored at location 4, with the value being searched, i.e. 31. We find that the value at location 4 is 27, which is not a match. As the value is greater than 27 and we have a sorted array, so we also know that the target value must be in the upper portion of the array.


We change our low to mid + 1 and find the new mid value again.

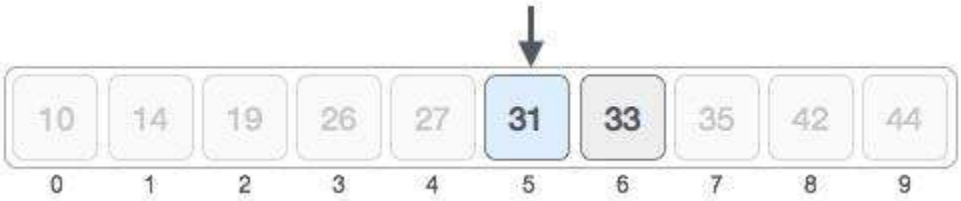
low = mid + 1

mid = low + (high - low) / 2

Our new mid is 7 now. We compare the value stored at location 7 with our target value 31.

The value stored at location 7 is not a match, rather it is more than what we are looking for. So, the value must be in the lower part from this location.

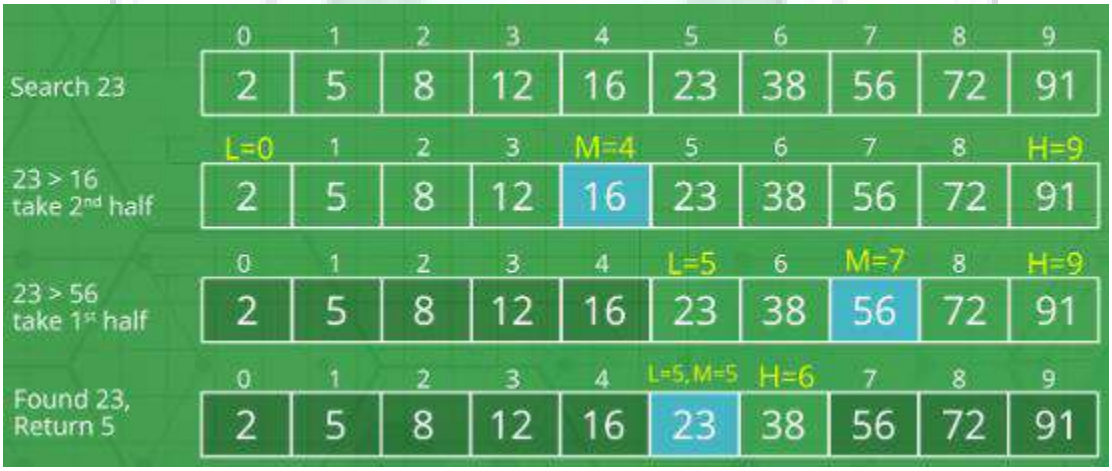

Hence, we calculate the mid again. This time it is 5.




We compare the value stored at location 5 with our target value. We find that it is a match.


We conclude that the target value 31 is stored at location 5.

**Another Example:**



# Northern University, Nowshera

## Application of Information and Communication Technologies (AICT)

Web:

http://www.cs.armstrong.edu/liang/animation/web/BinarySearch.html

https://yongdanielliang.github.io/animation/web/BinarySearchNew.html

## Algorithm

1. Take a value to be searched, from user
2. Set var low at start and high at end.
3. If low is greater from high display not found and go to step 9.
4. Calculate midpoint.
5. Compare value with value at mid.
6. If value found display message and go to step 9.
7. If value is greater from mid than set low equal to mid + 1 and go to step 3.
8. If value is less from mid than set high equal to mid – 1 and go to step 3.
9. Stop.

## Pseudo Code

1. Start
2. Input x
3. low := 0, high := n-1
4. if high < low
    i. display: value not found go to 9
5. mid := low + high/2
6. if x = a[mid]
    i. display: found at index i go to 9
7. if x > a[mid]
    i. low := mid + 1 go to 4
8. if x < a[mid]
    i. high := mid - 1 go to 4
9. stop

# Northern University, Nowshera

## Application of Information and Communication Technologies (AICT)

**Comparison between Binary Search and Linear Search:**

- Binary Search requires the input data to be sorted; Linear Search doesn't.
- Binary Search requires an *ordering* comparison; Linear Search only requires equality comparisons.
- Binary Search has complexity O(log n); Linear search has complexity O(n) as discussed earlier.
- Binary Search requires random access to the data; Linear Search only requires sequential access.

**Video Link:**

https://www.youtube.com/playlist?list=PLeby327DX1chatsrI4xy_7Wb1bpRCwUM0