Unit 5

Searching, Sorting and Hashing

Searching:

- **"Searching** is a process of finding a particular record, which can be a single element or a small chunk, within a huge amount of data."
- The data can be in various forms: arrays, linked lists, trees, heaps, and graphs etc.

- **Searching Algorithms in Data Structures:**

  Searching algorithms are essential tools in computer science **used to locate specific items** within a collection of data.

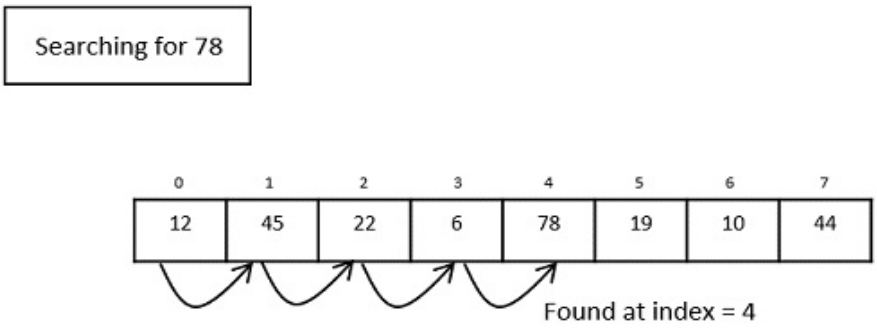
- Various searching techniques can be applied on the data structures to retrieve certain data. A search operation is said to be successful only if it returns the desired element or data; otherwise, the searching method is unsuccessful.
- There are two categories these searching techniques fall into. They are −
  Sequential/ Linear Searching
  Interval/Binary Searching

1)Linear /Sequential Searching :

As the name suggests, the sequential searching operation traverses through each element of the data sequentially to look for the desired data. **The data need not be in a sorted manner** for this type of search.Time Complexity O(n)
Example − Linear Search

Unit 5

Searching, Sorting and Hashing



Linear Search Algorithm

The algorithm for linear search is relatively simple. The procedure starts at the very first index of the input array to be searched.

Step 1 − Start from the 0th index of the input array, compare the key value with the value present in the 0th index.

Step 2 − If the value matches with the key, return the position at which the value was found.

Step 3 − If the value does not match with the key, compare the next element in the array.

Step 4 − Repeat Step 3 until there is a match found. Return the position at which the match was found.

Step 5 − If it is an unsuccessful search, print that the element is not present in the array and exit the program.

Pseudocode

```
procedure linear_search (list, value)
   for each item in the list
      if match item == value
         return the item's location
```

```
    end if
  end for
end procedure
```

Analysis

Linear search traverses through every element sequentially therefore, the best case is when the element is found in the very first iteration. The best-case time complexity would be **O(1)**.

However, the worst case of the linear search method would be an unsuccessful search that does not find the key value in the array, it performs n iterations. Therefore, the worst-case time complexity of the linear search algorithm would be **O(n)**.

Linear Search	
7 Advantages	**5 Disadvantages**
Simplicity and ease of implementation	Worst case time complexity is O(N)
Best case time complexity of O(1)	Not suitable for Sorted data

Average case is same as Worst Case: Constant expectation	Not suitable for frequent searches
No additional memory space required	Not suitable for data structures with non-sequential access
Best searching algorithm for data with < 512 elements	Not suitable for very large data
Works on any data structure that allows sequential access	-
Can handle frequently changing data	-

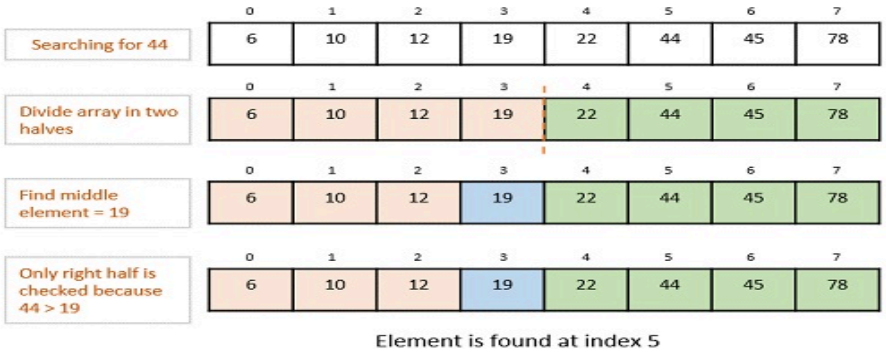
2) Interval Searching/Binary Searching:

Unlike sequential searching, the interval searching operation **requires the data to be in a sorted manner.** This method usually searches the data in intervals; it could be done by either dividing the data into multiple sub-parts or jumping through the indices to search for an element.Time Complexity is O(log n)

Example − Binary Search, Jump Search etc.

Unit 5

Searching, Sorting and Hashing



Binary Search Algorithm

Binary Search algorithm is an interval searching method that performs the searching in intervals only. The input taken by the binary search algorithm must always be in a sorted array since it divides the array into subarrays based on the greater or lower values. The algorithm follows the procedure below −

Step 1 − Select the middle item in the array and compare it with the key value to be searched. If it is matched, return the position of the median.

Step 2 − If it does not match the key value, check if the key value is either greater than or less than the median value.

Step 3 − If the key is greater, perform the search in the right sub-array; but if the key is lower than the median value, perform the search in the left sub-array.

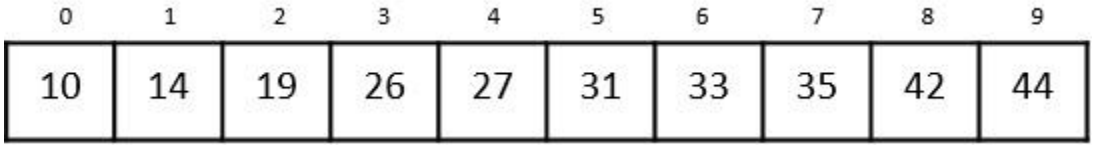
Step 4 − Repeat Steps 1, 2 and 3 iteratively, until the size of sub-array becomes 1.

Step 5 − If the key value does not exist in the array, then the algorithm returns an unsuccessful search.

Example:

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

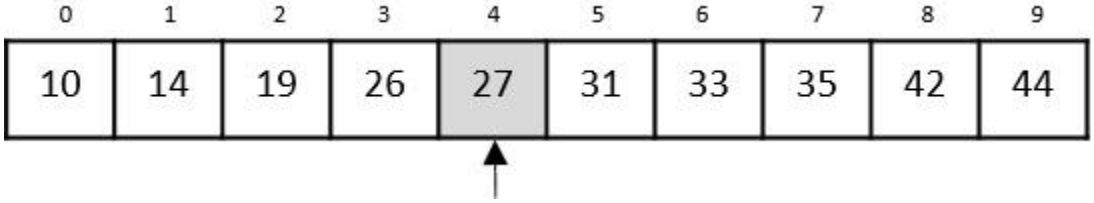
our sorted array and let us assume that we need to search the location of value 31 using binary search.



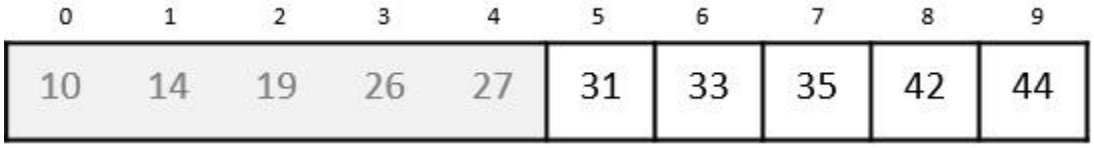
First, we shall determine half of the array by using this formula −

```
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.



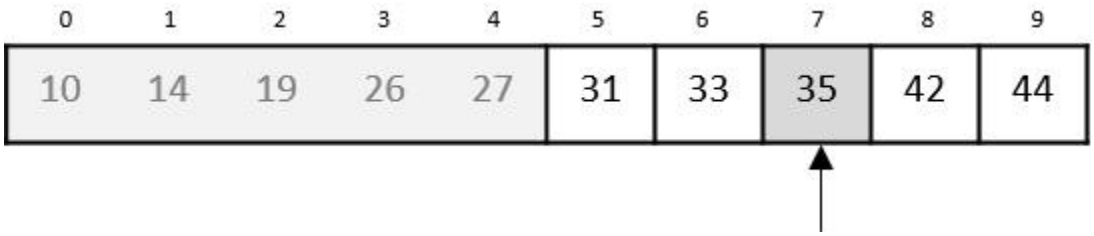
Now we compare the value stored at location 4, with the value being searched, i.e. 31. We find that the value at location 4 is 27, which is not a match. As the value is greater than 27 and we have a sorted array, so we also know that the target value must be in the upper portion of the array.



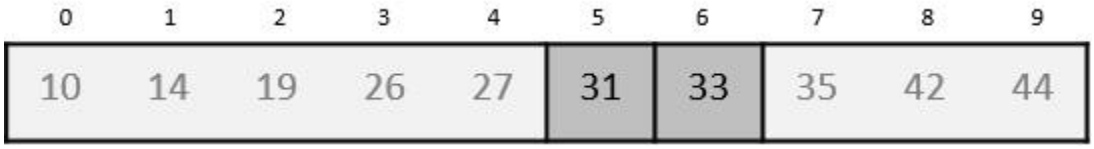
We change our low to mid + 1 and find the new mid value again.

```
low = mid + 1
mid = low + (high - low) / 2
```

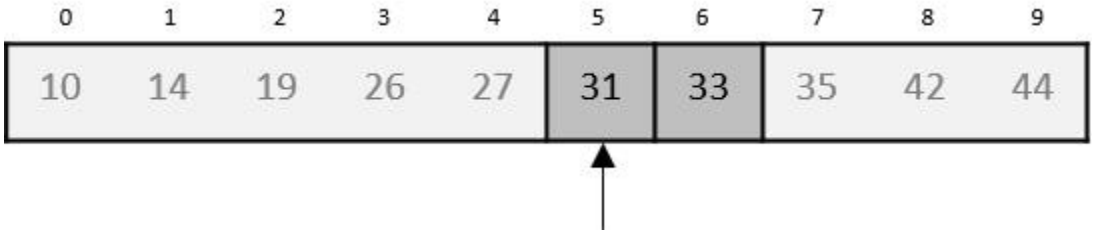
Our new mid is 7 now. We compare the value stored at location 7 with our target value 31.



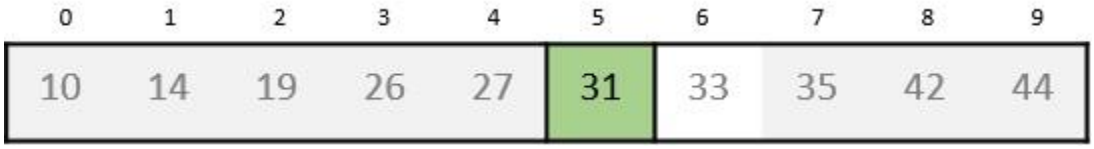
The value stored at location 7 is not a match, rather it is less than what we are looking for. So, the value must be in the lower part from this location.



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.

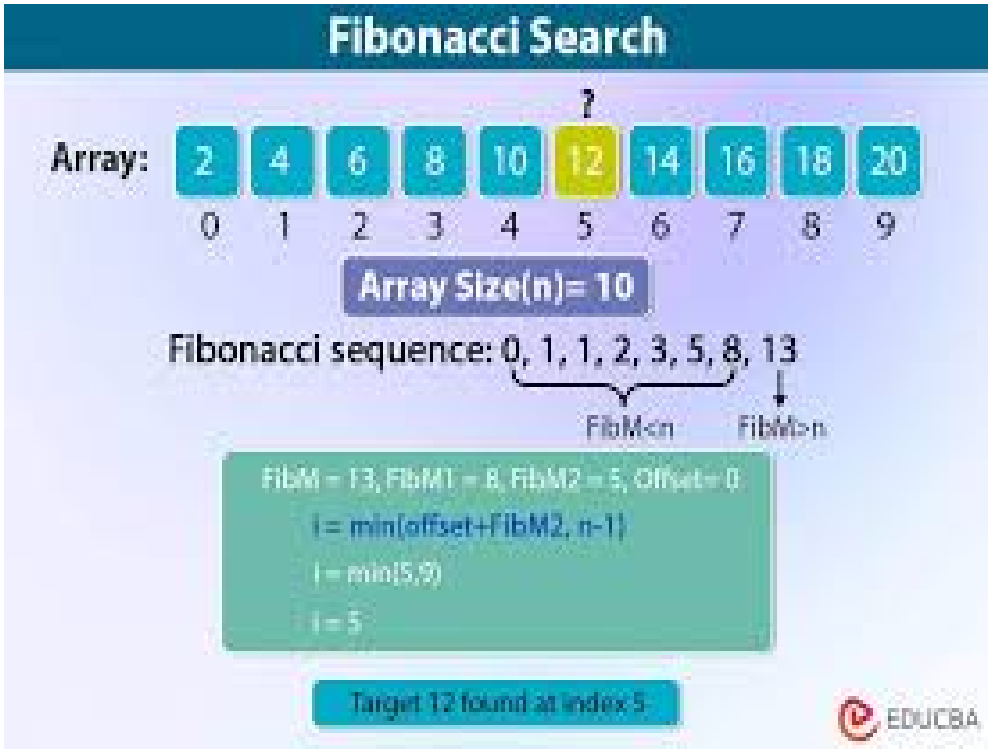
Binary search halves the searchable items and thus reduces the count of comparisons to be made to very less numbers.

Fibonacci Search:

- “Means searching an element in a array using fibonacci series”
- Array should be sorted then it is only works

Unit 5

Searching, Sorting and Hashing

- 

The Fibonacci search algorithm is a comparison-based search algorithm designed to find a specific element within a sorted array. It utilizes the properties of the Fibonacci sequence to efficiently narrow down the search space.

It is based on divide and conquer strategy.

Time Complexity:
The time complexity of Fibonacci search is O(logn), similar to binary search, as it also effectively halves the search space in each iteration

Advantages:

- No Division Operator: Unlike binary search, Fibonacci search avoids the division operator, which can be computationally expensive on some architectures. It primarily uses addition and subtraction.

Unit 5

Searching, Sorting and Hashing

Algorithm:

1) Find f(k) where {kth fibonacci number}.which is greater than or equal to n(size of array).
2) If f(k)=0 then stop and print the message as an element not found.
3) offset=-1
4) Find i=min(offset+f(k-2),n-1)
5) If S==A[i]
   Return i and stop the search
   If S>A[i]
   k=k-1, offset=i and repeat steps 4,5
   If S<A[i]
   k=k-2 repeat steps 4,5