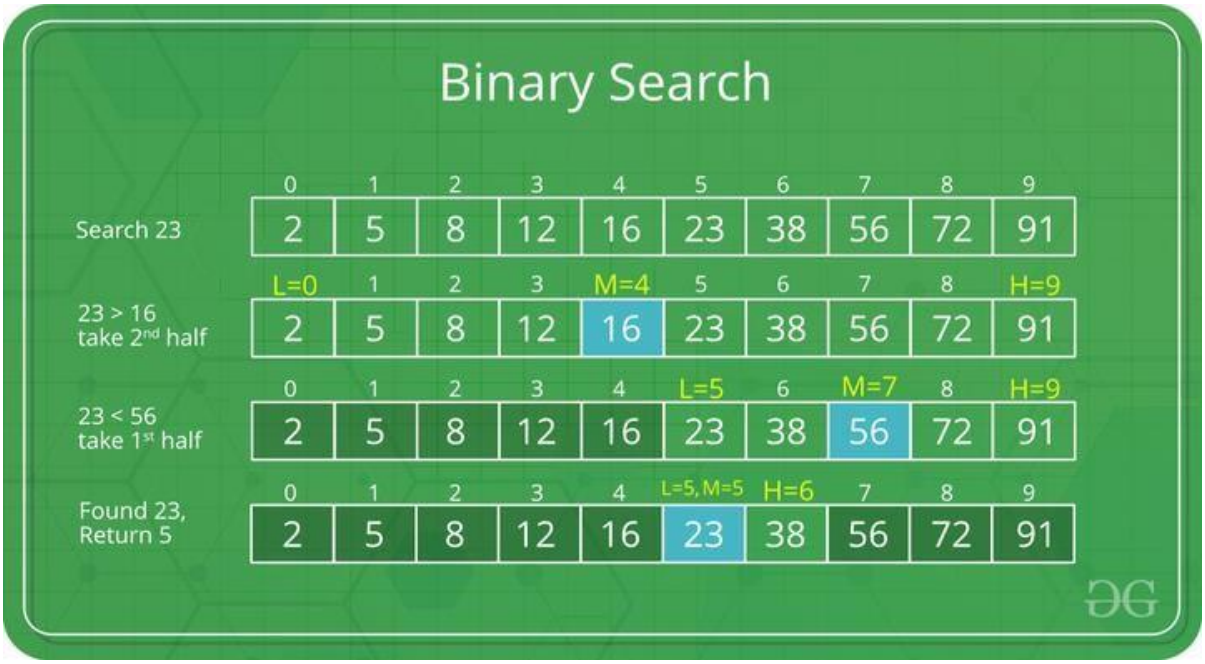
Binary Search

-Binary Search:

is defined as a searching algorithm used in a sorted array by repeatedly dividing the search interval in half. The idea of binary search is to use the information that the array is sorted and reduce the time complexity to O(log N).



-Conditions for when to apply Binary Search in a Data Structure:

To apply Binary Search algorithm:

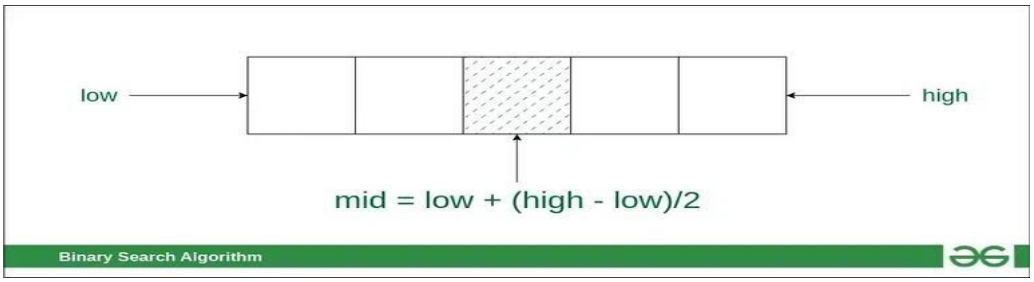
The data structure must be sorted.

Access to any element of the data structure takes constant time**.**

Binary Search Algorithm:

In this algorithm,

Divide the search space into two halves by finding the middle index “mid”**.**



- Compare the middle element of the search space with the key.
- If the key is found at middle element, the process is terminated.
- If the key is not found at middle element, choose which half will be used as the next search space.
- If the key is smaller than the middle element, then the left side is used for next search. ○ If the key is larger than the middle element, then the right side is used for next search.
- This process is continued until the key is found or the total search space is exhausted.

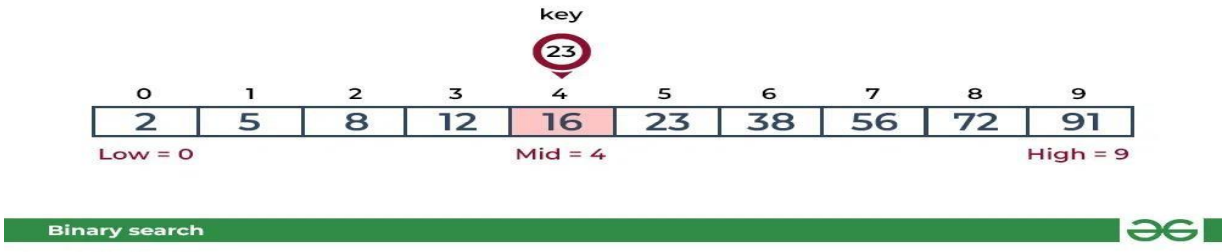
How does Binary Search work?

To understand the working of binary search, consider the following illustration:

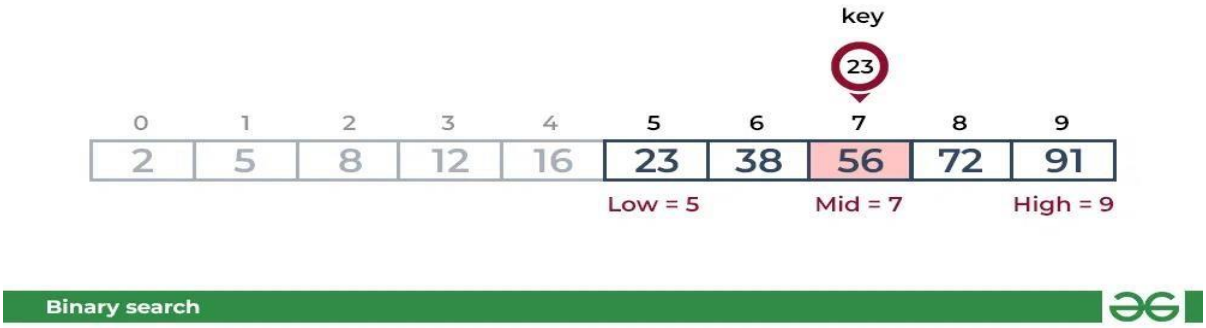
Consider an array ***arr[] = {2, 5, 8, 12, 16, 23, 38, 56, 72, 91}***, and the ***target = 23***.

-First Step: Calculate the mid and compare the mid element with the key. If the key is less than mid element, move to left and if it is greater than the mid then move search space to the right.

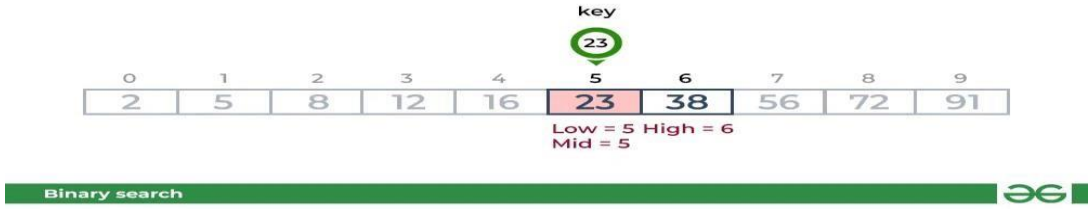
Key (i.e., 23) is greater than current mid element (i.e., 16). The search space moves to the right.



Key is less than the current mid 56. The search space moves to the left.



-Second Step: If the key matches the value of the mid element, the element is found and stop search.



-How to Implement Binary Search?

The Binary Search Algorithm can be implemented in the following two ways

Iterative Binary Search Algorithm

Recursive Binary Search Algorithm

Given below are the pseudocodes for the approaches.

1. Iterative Binary Search Algorithm:

Here we use a while loop to continue the process of comparing the key and splitting the search space in two halves.

2. Recursive Binary Search Algorithm:

Create a recursive function and compare the mid of the search space with the key. And based on the result either return the index where the key is found or call the recursive function for the next search space.

Advantages of Binary Search:

- Binary search is faster than linear search, especially for large arrays.
- More efficient than other searching algorithms with a similar time complexity, such as interpolation search or exponential search. ○ Binary search is well-suited for searching large datasets that are stored in external memory, such as on a hard drive or in the cloud.

Drawbacks of Binary Search:

- The array should be sorted.
- Binary search requires that the data structure being searched be stored in contiguous memory locations. ○ Binary search requires that the elements of the array be comparable, meaning that they must be able to be ordered.

Applications of Binary Search:

- Binary search can be used as a building block for more complex algorithms used in machine learning, such as algorithms for training neural networks or finding the optimal hyperparameters for a model.
- It can be used for searching in computer graphics such as algorithms for ray tracing or texture mapping.
- It can be used for searching a database.