# Binary Search Algorithm: Function, Benefits, Time & Space Complexity

## **Introduction**

In any computational system, the search is one of the most critical functionalities to develop. Search techniques are used in file retrievals, indexing, and many other applications. There are many search techniques available. One of which is the binary search technique.

A **binary search algorithm** works on the idea of neglecting half of the list on every iteration. It keeps on splitting the list until it finds the value it is looking for in a given list. A **binary search algorithm** is a quick upgrade to a simple linear search algorithm.

## **Working of a Binary Search Algorithm**

The first thing to note is that a **binary search algorithm** always works on a sorted list. Hence the first logical step is to sort the list provided. After sorting, the median of the list is checked with the desired value.

* If the desired value is equal to the central index’s worth, then the index is returned as an answer.
* If the target value is lower than the central index’s deal of the list, then the list’s right side is ignored.
* If the desired value is greater than the central index’s value, then the left half is discarded.
* The process is then repeated on shorted lists until the target value is found.

**Example #1**

Let us look at the algorithm with an example. Assume there is a list with the following numbers:

1, 15, 23, 7, 6, 14, 8, 3, 27

Let us take the desired value as 27. The total number of elements in the list is 9.

**The first step is to sort the list. After sorting, the list would look something like this:**

1, 3, 6, 7, 8, 14, 15, 23, 27

As the number of elements in the list is nine, the central index would be at five. The value at index five is 8. The desired value, 27, is compared with the value 8. First, check whether the value is equal to 8 or not. If yes, return index and exit.

As 27 is greater than 8, we would ignore the left part and only traverse the list’s right side. The new list to traverse is:

14, 15, 23, 27

**Note:** In practice, the list is not truncated. Only the observation is narrowed. So, the “new list” should not be confused as making a new list or shortening the original one. Although it could be implemented with a new list, there are two problems. First, there will be a memory overhead. Each new list will increase the space complexity. And second, the original indexes need to be tracked on each iteration.

The new central index can be taken as the second or third element, depending on the implementation. Here, we will consider the third element as central. The value 23 is compared with value 27. As the value is greater than the central value, we will discard the left half.

**The list to traverse is:**

27

As the list contains only a single element, it is considered to be the central element. Hence, we compare the desired value with 27. As they match, we return the index value of 27 in the original list.

**Example #2**

In the same list, let us assume the desired value to be 2.

First, the central value eight is compared with 2. As the desired value is smaller than the central value, we narrow our focus down to the list’s left-hand side.

**The new traversal will consist of:**

1, 3, 6, 7

Let us take the central element as the second element. The desired value two is compared with 3. As the value is still smaller, we again narrow the focus down to the list’s left-hand side.

**The new traversal will consist of:**

1

As the traversing list has only one element, the value is directly compared to the remaining element. We see that the values do not match. Hence, we break out of the loop with an error message: value not found.

## **Time and Space complexity**

The time complexity of the**binary search algorithm** is O(log n). The best-case time complexity would be O(1) when the central index would directly match the desired value. The worst-case scenario could be the values at either extremity of the list or values not in the list.

The space complexity of the **binary search algorithm** depends on the implementation of the algorithm. There are two ways of implementing it:

1. Iterative method
2. Recursive method

Both methods are quite the same, with two differences in implementation. First, there is no loop in the recursive method. Second, rather than passing the new values to the next iteration of the loop, it passes them to the next recursion. In the iterative method, the iterations can be controlled through the looping conditions, while in the recursive method, the maximum and minimum are used as the boundary condition.

In the iterative method, the space complexity would be O(1). While in the recursive method, the space complexity would be O(log n).

## **Benefits**

* A **binary search algorithm** is a fairly simple search algorithm to implement.
* It is a significant improvement over linear search and performs almost the same in comparison to some of the harder to implement search algorithms.
* The **binary search algorithm** breaks the list down in half on every iteration, rather than sequentially combing through the list. On large lists, this method can be really useful.

**Checkout:**[Decision Tree Classification: Everything You Need to Know](https://www.upgrad.com/blog/decision-tree-classification-everything-you-need-to-know/)

## **Conclusion**

A **binary search algorithm** is a widely used algorithm in the computational domain. It is a fat and accurate search algorithm that can work well on both big and small datasets. A **binary search algorithm** is a simple and reliable algorithm to implement. With time and space analysis, the benefits of using this particular technique are evident.

If you are curious to learn about data science, check out IIIT-B & upGrad’s [PG Diploma in Data Science](https://www.upgrad.com/data-science-pgd-iiitb/?utm_source=BLOG&utm_medium=BODY&utm_campaign=DV_DA_PGD_BLOG_BODY_88588) which is created for working professionals and offers 10+ case studies & projects, practical hands-on workshops, mentorship with industry experts, 1-on-1 with industry mentors, 400+ hours of learning and job assistance with top firms.

## Is it true that linear search is superior to binary search?

If you just need to search once, linear search will surely be faster than sorting followed by binary search if the data is originally unsorted. Binary search, on the other hand, is recognized to be a considerably quicker method of searching than linear search. Binary search allows you to remove half of the remaining items at a time, whereas linear search would go through each element one by one.

## What distinguishes interpolation search from binary search?

Interpolation search is a binary search-like technique for finding a specified target value in a sorted array. It's similar to how people search through a phone book for a certain name, with the target value used to sort the book's contents. To check, binary search always travels to the center element. Interpolation searching, on the other hand, may lead to various places depending on the value of the key being searched for. If the key's value is closer to the final element, for example, interpolation search is more likely to begin at the end.