**Binary Search: Explained**

380

This is the very first article of the Binary Search series. Until now, we have learned the [linear search](https://takeuforward.org/data-structure/linear-search-in-c/) algorithm. Now, in this article, we will discuss another search algorithm i.e. the Binary Search algorithm. The flow of this article will be the following:

* **A real-life example of Binary Search**
* **Coding problem example**
* **Iterative code implementation of Binary Search**
* **Recursive code implementation of Binary Search**
* **Time complexity**
* **Overflow case**

**A real-life example of Binary Search:**

**Problem statement:**Assume there is a dictionary and we have to find the word “raj”.

**Method 1:**One of the many ways is to check every possible page of the entire dictionary and see if we can find the word “raj”. This technique is known as [linear search](https://takeuforward.org/data-structure/linear-search-in-c/). *Basically, we can traverse from the first till the end to find the target value in the search space i.e. the entire dictionary in our example.*

**Method 2:**In this case, we will optimize our search by using the property of a dictionary i.e. a dictionary is always in the sorted order.

* We will first try to open the dictionary in such a way that it is roughly divided into two parts. Then, we will check the left page. Now, assume the words on the left page starts with ‘s’. We can certainly say that our target word i.e. “raj” definitely comes before the words start with ‘s’. So, now, we need not search in the entire dictionary rather we will only search in the left half.
* Now, we will do the same thing with the left half. First, we will divide it into 2 halves and then try to locate which half contains the word “raj”. Eventually, after certain steps, we will end up finding the word “raj”.

This is a typical real-life example of binary search.

**Note:**

* Binary search is only applicable in a sorted search space. The sorted search space does not necessarily have to be a sorted array. It can be anything but the search space must be sorted.
* In binary search, we generally divide the search space into two equal halves and then try to locate which half contains the target. According to that, we shrink the search space size.

**Coding problem example:**

**Problem statement:** You are given a sorted array of integers and a target, your task is to search for the target in the given array. Assume the given array does not contain any duplicate numbers.

Let’s say the given array is = {3, 4, 6, 7, 9, 12, 16, 17} and target = 6.