Binary Search-

- Binary Search is one of the fastest searching algorithms.
  - It is used for finding the location of an element in a linear array.
  - It works on the principle of divide and conquer technique.

Binary Search Algorithm can be applied only on **Sorted arrays**.

So, the elements must be arranged in-

- Either ascending order if the elements are numbers.
- Or dictionary order if the elements are strings.

To apply binary search on an unsorted array,

- First, sort the array using some sorting technique.
- Then, use binary search algorithm.

How Binary Search Works?

For a binary search to work, it is mandatory mandatory for the target array to be sorted. sorted. We shall learn the

Process of binary search with a pictorial pictoriCal example. 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)) // 22

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 location 4 is 27, which is not a match. As the value is greater 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 ++ 11

mid == low ++ ((high -- loww)) // 22

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

The value stored at location location 7 is not a match, rather it is more than what we are looking 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 searchable items and thus reduces reduces the count of comparisons comparisons to be made to very less numbers

PSEUDOCODE

```
int binarySearch(int arr[], int l, int r, int x)
{
   while (l <= r) {
      int m = l + (r - l) / 2;
```

```
        // Check if x is present at mid
        if (arr[m] == x)
            return m;

        // If x greater, ignore left half
        if (arr[m] < x)
            l = m + 1;

        // If x is smaller, ignore right half
        else
            r = m - 1;
    }

    // If we reach here, then element was not present
    return -1;
}
```