# Data Structure and Algorithms Binary Search

A quick search algorithm with run-time complexity of O is binary search (log n). Divide and conquer is the guiding philosophy behind this search algorithm. The data collection must be in sorted form for this algorithm to function correctly.

Binary search compares the collection's middle item in an effort to find a specific item. If a match occurs, then the index of item is returned. The item is searched in the sub-array to the left of the middle item if the middle item is greater than the item. If not, the sub-array to the right of the middle item is searched for the item. This procedure is repeated on the sub-array as well until the sub-size array's reaches zero.



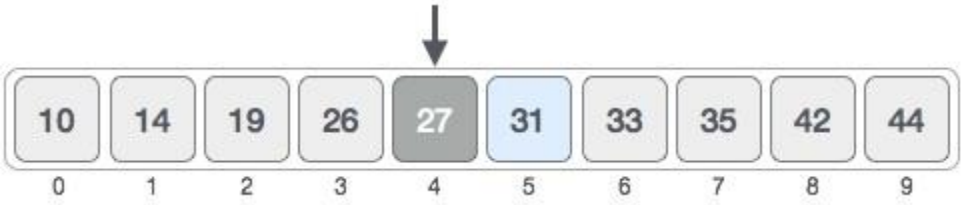
How Binary Search Works?

The target array must be sorted in order for a binary search to function. We will use a visual illustration to teach us how binary search works. Here is our sorted array, and let's imagine that we need to use binary search to locate the value 31.


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.

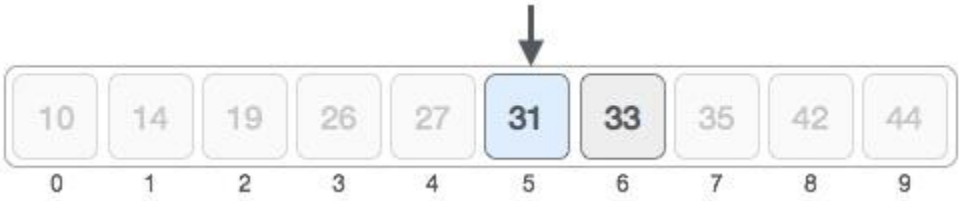

We change our low to mid + 1 and find the new mid value again.

```
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 more 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.

Pseudocode

The pseudocode of binary search algorithms should look like this –

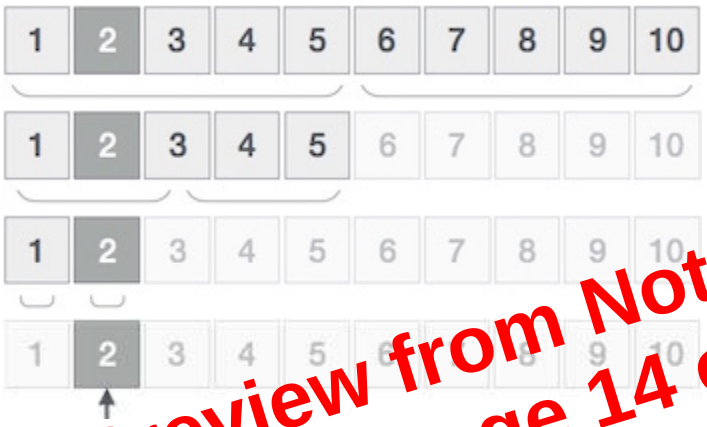


```
Procedure binary_search
  A ← sorted array
  n ← size of array
  x ← value to be searched

  Set lowerBound = 1
  Set upperBound = n

  while x not found
    if upperBound < lowerBound
      EXIT: x does not exists.
```

want to search the telephone number of Morphius. Here, linear search and even binary search will seem slow as we can directly jump to memory space where the names start from 'M' are stored.

Positioning in Binary Search

In binary search, if the desired data is not found then the rest of the list is divided in two parts, lower and higher. The search is carried out in either of them.






Even when the data is sorted, binary search does not take advantage to probe the position of the desired data.

Position Probing in Interpolation Search

Interpolation search finds a particular item by computing the probe position. Initially, the probe position is the position of the middle most item of the collection.


If a match occurs, then the index of the item is returned. To split the list into two parts, we use the following method –