The array is iteratively split into half till the search element is found or till the low index goes beyond the high index. In the latter case, the search element is assumed not to be present in the array and the search operation is unsuccessful. The algorithm looks like below

```
Low ← 1, High ← N, S ← Item to be searched
while (High >= Low)
 mid ← (High + Low) / 2
 if (Array[mid] = S
  return mid
 else
   if Array[mid} < S
    low ← mid + 1
  else
    high ← mid - 1
end while
return -1 (i.e. item not found)
```

FIGURE 1: BINARY SEARCH

In the worst-case scenario, the total number of steps required to search for a given element is equal to the maximum number of times the array is split and scanned. To continue the search operation, the array is split till the size becomes one.

Let us now derive the number of splits required for an 'n' sized array.

Array Size	Number of Times Split
3	1
4,5,6,7	2
8	3

FIGURE 2

Generalizing the data from the above table, we see that for an array size of 'n', the array is split *log n* times.