Binary Search

- Efficient searching method, which consumes less time
- The essential thing here is the array should be sorted first.
- Steps for binary search

Let the input array be A[0 - - n-1]
Key is the element to be searched
A[m] is the mid element of array A.
Then,

1. Check is A[m] = Key
2. If A[m] > Key, then go to left Sub list.
3. If A[m] < Key, then go to right sublist.

Example

Input array → 0 5 10 15 20 25 30 35 40

0	1	2	3	4	5	6	7	8
0	5	10	15	20	25	30	35	40
↑ Low								↑ High

Let Key = 5 (no. to be searched)

Now find the middle element

$$m = (low + high)/2$$
$$= (0 + 8)/2$$
$$= 4$$

$\therefore A[m] = 20$

Check if $A[m] = Key$

Here, $A[m] = 20$
$Key = 5$

$\therefore A[m] \neq Key$

Now, $Key < A[m]$

$\therefore$ we have to search the left sub-list

0	1	2	3
0	5	10	15

↑ Low ↑ High

$m = (Low + high)/2$
$= (0 + 3)/2$
$= 1$

Now $Key = 5$
$A[m] = 5$

$\therefore Key = A[m]$

Thus, the no. is present in the list.

Algorithm for binary search

1. Initialize array $A[0 \dots n-1]$
2. $low \leftarrow 0$
3. $high \leftarrow n-1$
4. Do $((low + high)/2) \rightarrow m$
5. if $(key = A[m])$ then

   return m;

6. if $(key < A[m])$ then

   $high \leftarrow m-1$ [Search left sub tree]

7. if $(key > A[m])$ then

   $low \leftarrow m+1$ [Search right sub tree]

8. return -1 (if element is not present)