Binary Search

To search the data using binary search array should be sorted.

Let's take an example to understand the working principle of binary search.

$n = 10$
search data $= 59$

	0	1	2	3	4	5	6	7	8	9
a	5	9	17	23	25	45	59	63	71	89

Binary search uses divide & conquer approach to it recursively divides the array.

First find middle element of an array.

$l = 0$
$r = 9$

$$mid = \frac{l+r}{2}$$ (take floor value)

$$mid = \frac{0+9}{2} = 4$$

~~As mi~~ After finding mid there are 3 cases:-
1) Data to search is equal to mid element.
2) Data to search is less than mid element
3) Data to search is greater than mid element

case I: data $= a[mid]$
case II: data $> a[mid]$
case III: data $< a[mid]$

In our case $a[mid] = a[4] = 25$

data $>$ $a[mid]$ case II
$59 > 25$

As the array is is sorted data is present on the right side of mid

After 1st comparision our sample space divided into half
earier we had 10 nos to search & now we need to find between 6 nos.

Now $l = mid+1 = 5$
$r = 9$
$mid = \frac{5+9}{2} = 7$
$a[mid] = a[7] = 63$

data $< a[mid]$
$59 < 63$ Case III

Now $l = 5$
$r = mid-1 = 7-1 = 6$
$mid = \frac{l+r}{2} = \frac{5+6}{2} = \frac{11}{2} = 5$
$a[mid] = a[5] = 45$

data > a[mid] case III
59 > 45

we need to check to the right side of mid.

Now, $l = mid+1 = 6$
$r = 6$

$mid = \frac{6+6}{2} = 6$

$a[mid] = a[6] = 59$

data = a[mid] case I
59 = 59

and the data is found. return is the index 6 at which 59 is present.

If the data is not present then how binary search works

l	r	mid	
0	9	4	Iteration 1
5	9	7	Iteration 2
5	6	5	Iteration 3
6	6	6	Iteration 4
7	6		Iteration 5

We need to stop searching at Iteration 5 and concuble that the element is present in the array.

So the stopping condition for binary search is $l > r$.

if $(l > r)$ then data is not present

```
BinarySearch ( a , n , data )
{
    l=0 , r=n-1
  while( l<r )
  {  mid = (l+r)/2

     if (data == a[mid] )
           return mid;
     else if (data < a[mid] )
           r = mid-1;
     else
           l = mid+1;
  }
  return -1;
}
```

Time complexity

As our sample space ~~div~~ reduced to half after every comparision ~~as~~, binary search has worst case time complexity as $O(\log n)$

In best case the time complexity of binary search is O(1).