Binary Search Algorithm

arr = [2, 4, 6, 9, 11, 12, 14, 20, 36, 48]
target = 36

- Binary search works on sorted array

=> 1. Find the middle element

=> 2. check :
if target > middle => search in right (start = mid+1)
else search in left side (end = mid-1)

=> 3. if target == middle => we found element

Example

In the above array

	0 (s)	1	2	3	4	5	6	7	8	9 (e)
arr =	2	4	6	9	11	12	14	20	36	48

target = 36

=> 1st -> find middle Element   s = 0, e = 9 (length)

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

	0 (s)	1	2	3	4 (m)	5	6	7	8	9 (e)
arr =	2	4	6	9	[11]	12	14	20	36	48

middle Element is arr[mid] = 36

:) 2nd -) Let's check:

target > middle :) 36 > 11 => yes

⇓

check in right side and skip left part from middle by start = mid + 1

[2, 4, 6, 9, 11, 12, 14, 20, 36, 48]
s → 2, m → 11, e → 48

⇓

start = mid + 1

⇓

now arr:

s → 12, e → 48

2, 4, 6, 9, 11 { 12, 14, 20, 36, 48 }
skiped because target > middle.

mid = (s + e)/2 = (s + 9)/2 = 7

let's check:

=> target > middle :) 36 > 20 :) yes
again skip left side part and search in right side from middle

start = mid + 1;

s → 8, e → 9

{ 36, 48 }

mid = (s + e)/2 = (8 + 9)/2 = 8

[36, 48]
s → 1, m → 1, e → 1

=) let's check:

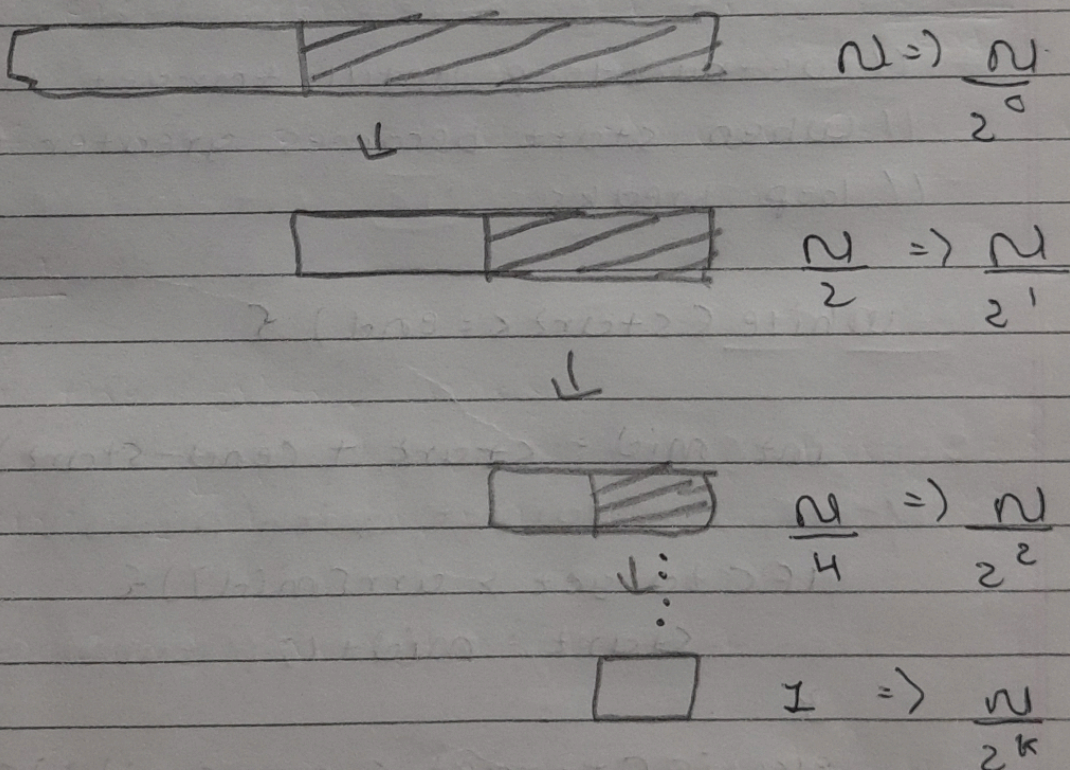
target == middle =) yes => we found target.

arr[5] = 36
ans

=) time Complexity:

=) Best case : O(1) => if mid is target (only 1 check)

=) Worst case : O(logn) => if element target not found



$\frac{N}{2^k} = 1$

$N = 2^k$

$\log(N) = \log(2^k)$

$\log(N) = k \log 2$

$k = \frac{\log N}{\log 2}$

worst case ↓ O(logn)

$k = \log_2 N$

code:

arr = [2, 4, 6, 9, 11, 12, 14, 20, 36, 46]
target = 36

Binary search method

```
binarySearch (int[] arr, int target){

  // first we initialize start and end point

    int start = 0;
    int end = arr.length -1;

  // we run loop until target is not found
  // when start becomes greater than end
  // loop breaks

    while (start <= end) {

      int mid = start + (end - start) /2;
  //compare target with middle
      if (target > arr[mid]){
          start = mid+1; // search in right
                                    side
      }
      else if (target < arr[mid]){
          end = mid-1; // search in left
                                    side
      }
      else{
          return mid; // target is found
      }
  }
  return -1; // taryet not found.
```