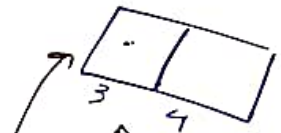
BINARY SEARCH

ITERATIVE

```
f( arr, n, target ) {
    low = 0, high = n - 1;
    while(low ≤ high) {
        mid = (low + high) / 2;
        if (arr[mid] == target) return mid;
        else if (target > arr[mid]) low = mid + 1;
        else high = mid - 1;
    }
return -1;
```



do main mid hamesha pehle jaha pe hoge. if not found then single element

RECURSIVE

```
f(arr, low, high, target) {
  if (low > high) return -1;
  if(arr[ma mid = (low + high)/2;
  if (a[mid] == target) return mid;
else if (target > arr[mid]) return f(arr, mid+1, high, target)
else return f(arr, low, mid - 1, target);
}
```

complexity - (32) → ①

→ (16) → ②

somewhere about half

8 → ③

4 → ④

2 → ⑤

1 → ⑥

$\log_2(32) = 6$

$\therefore \log_2(n)$;

overflow case when

0 ..... INTMAX

assume that low and high becomes equal to max index

0 - - - - Int_Max

then $\frac{(low + high)}{2}$ ← overflow

∴ take long datatype for high and low

or

use this → low mid = low + $\frac{(high - low)}{2}$

↓

eventually

$\frac{2low + high - low}{2}$ = $\frac{low + high}{2}$

LOWER & UPPER BOUND

smallest index such that

arr[ind] >= n

arr = [3, 5, 8, 15, 19] ⇒ n = 5

n = 8

then answer ind = 2

n = 4, ind = 3

if n = 20

ind = 5 (last as not one > found)

arr = [3, 5, 8, 15, 19, 19, 19]

n = 19

then ind = (4) as smallest

→ answer = 10; (arr size)

```
if (arr, target, n) {
    low = 0, high = n-1;
    ans = n;          // when none can exists
    while (low <= high) {
    mid = (low + high)/2;
    if (arr[mid] >= target) {
                ans = mid;
                high = mid - 1;
            }
    else low = mid + 1;
      }
return ans;
```

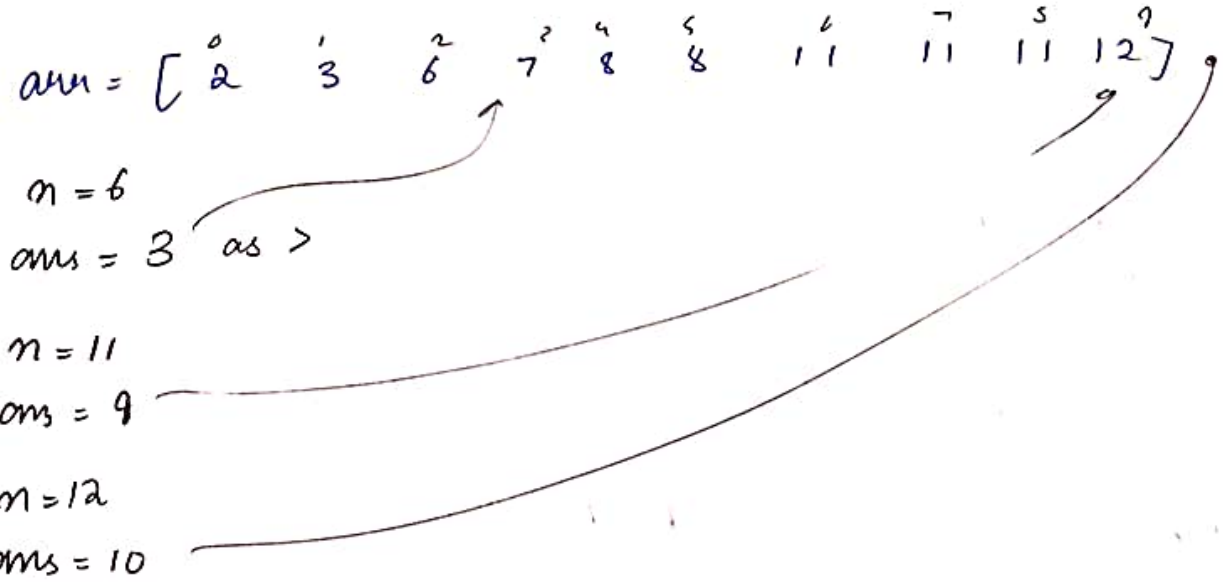
can be an answer as we are looking for an element i.e arr[mid] ≥ target but not sure if it the smallest greater answer the question.

no chance of answer

time → $\log(n)$

UPPER BOUND →

smallest index such that arr[ind] > n;



previous know (arr[mid] >= n) {

ans = mid;

high = mid - 1;

}

but now if (arr[mid] > n) {

ans = mid;

high = mid - 1;

}

$\log(n)$