Binary Search:

In normal array if we want to search any element if its there or not, worst scenario would be o(n) as not fidning it.

In binary search, for example lets take a sorted array.

* Find the middle element
* If target elem>middle elem, then search in right, otherwise in left.
* If target elem==middle elem. Then that’s the answer.

Now,

0 level = N/2^0= N/1 = N

1 level = N/2^1 = N/2

2 level = N/2^2 = N/4

…..

Kth level = N/2^k

N/2^k = 1 => N/2^k

log(N) =log(2^k)

log(N)= klog2

k = log(N)/long2;

k= logN

so……Remember:

Total number of worst case comparison = logN

So if a linear array is element of 1 million, worst case comparison would be 1 million

But for Binary search, log(1million) with base 2 would be, 19   
night and day difference.

#Ceiling of a number

Smallest number greater than equal to target number

Start <= end

So when while loop breaks, start will be eequal to end+1.

Next big numbers when no answer found will be equal to start element. that means next starting will be our answer.

So that number will be GREATER or EQUAL to Target and that is also the smallest number as the starting position is now changed and starts from just right on the target element.

So when the answer is not found in start and end, and when the while loop is violated, hence the answer is not in between start and end. So now next the start as start has been increamented, and that’s the next greater element.

All of this above works and ascending order sorted array.

#Floor of a number

greatest number smaller than equal to target number.

So when the condition is violated as Start increamented and cross the end. That means end is our target element or the answer as per as the condition. Which is biggest number which is smaller than the equal to target number.

So above 2 problem ceiling and flooring.

Example suppose we want 15 in an aarray of 14, 16, 18   
then for ceiling: smallest number greater then equal to 15 which answer would be 16, start index.

Then for flooring: greates number smaller then equal to 15 which is 14, end index.

#Smallest letter check

* Exactly as ceiling problem
* Ignore the target= what we are looking for. (smallest number greater than target number no need to check equality.)
* Arr [c,d,f,j ] if target is ‘j’ , then there is no larger than j so the answer will be ‘c’. as start will be end+1 so loop will be break. So now the start is arr.length. so if we just modulus the start by array size we get ZERO. That’s the position of ‘c’. simple.

#finding first and last position of element.