★★★ Binary search in Java ★★★

★ Binary Search :-

⇒ It is the most important Algorithm.

⇒ It is used for Sorted array. (either in ascending order or descending)

★ Algorithm :- (When in ascending order)

steps :-

① Find the middle element (in array)

② target > mid ⇒ search in the right.
else ⇒ Search in the left.

③ if middle element == target. // answer.

★ Example :-

arr = [2, 4, 6, 9, 11, 12, 14, 20, 36, 48]
0 1 2 3 4 5 6 7 8 9

target = 36.

① 1st step is to found middle element.

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

★ $\frac{0+9}{2}$ = ④ Middle element ⇓ Mid + 1

② nd step, if target > M. ⇒ search in right.

∴ 20, 14, 20, 36, 48
5 6 7 8 9

$\frac{9+5}{2}$ = ⑦ → 20 < 36

36, 48

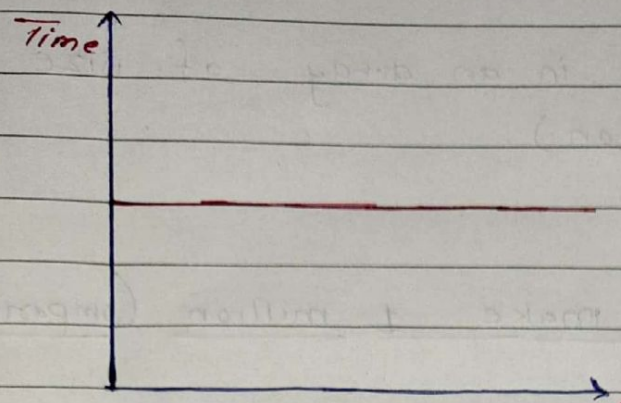
Middle :

$\frac{9+8}{2}$ = ⑧

target == 36

★ // Ans.

* Best Case.



Best case. O(1)

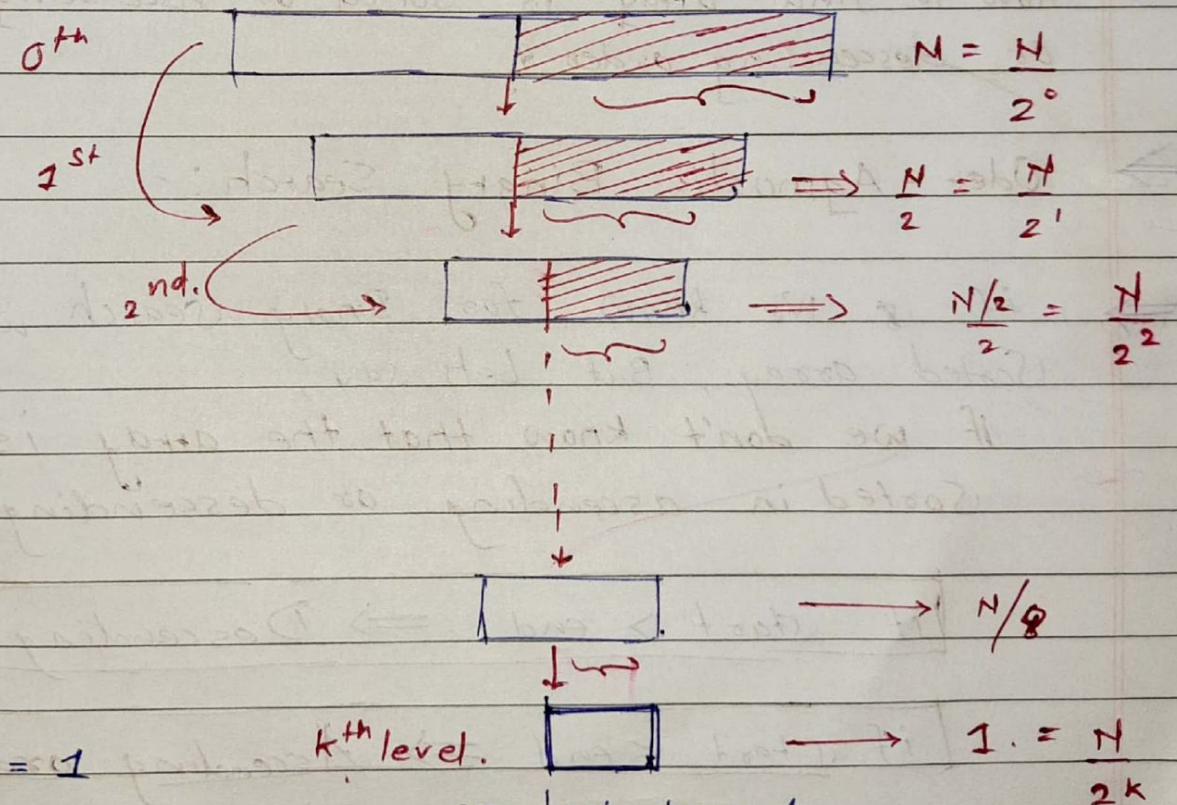
* Worst case:-

O(log N)

// n → size of array

* Why Binary Search :?

Q. Find the maximum numbers of Comparison in worst case.....



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

one last element there will only one this is ans//.

$\Longrightarrow N = 2^k$

$\longrightarrow \log N = k \log 2.$

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

$\Longrightarrow \boxed{k = \log_2 N}$ → size of array

} Total numbeo of Comparisons in worst case.

+ Why Binary Search ?

Q. Search an element in an array of size 1,000,000 (1 million).

⇒ In worst case,
Linear search will make 1 million comparisons.

⇒ In worst case for same array,
Binary search will make log(1,000,000)
i.e. 20 comparisons only

+ How to find array is sorted or Ascending order or descending order ?

⇒ Oder - Agnostic Binary Search :-

⇒ We know, for Binary Search we need Sorted array, But Let's say,
if we don't know that the array is sorted in ascending or descending order.

if start > end ⇒ Descending order;

if start < end ⇒ Ascending order.