Binary Search

It is used for Sorted arrays

arr = [2, 4, 9, 10, 12, 14, 18, 19]
→ ascending

arr 2 = [19, 12, 6, 5, 3, 2, -8, -16]
→ descending

In Linear Search max comparisons : N ⇒ No. of element

s = 0, m = 4, e = 9

indices: 0 1 2 3 4 5 6 7 8 9

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

target = 36

① Find the middle Element $\frac{s+e}{2}$

② target > mid ⇒ search in right
else search in left

③ if middle element == target //ans

s = 0, m = 4, e = 9

indices: 0 1 2 3 4 5 6 7 8 9

2, 4, 6, 9, (11), [12, 14, 20, 36, 48]

↓

$\frac{5+9}{2} = 7$

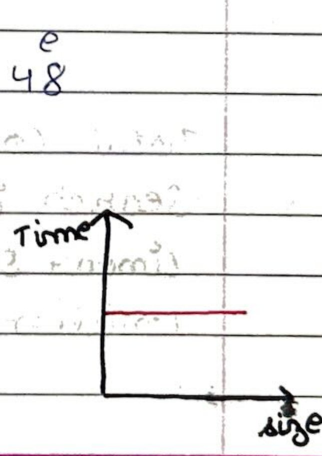
s, m, e: 12, 14, 20, [36, 48]

↓

s, e: (36), 48

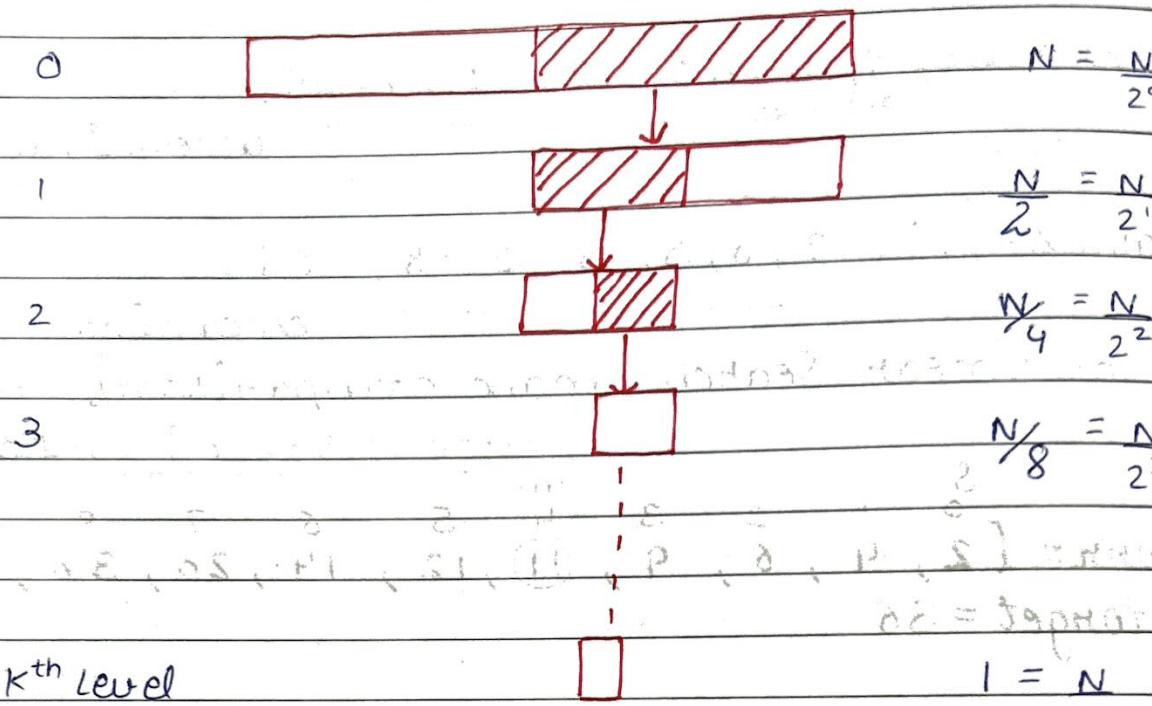
if s > e : element not found

Best case
O(1)



Why Binary Search?

→ Find the max numbers of such comparison in worst case?



$\frac{N}{2^K} = 1 \Rightarrow N = 2^K$

$\log N = K \log 2$

$K = \frac{\log N}{\log 2}$ ⇒ $O(\log N)$

$K = \log_2 N$

Total comparisons in the worst case = $\log_2 N$

Search in 1000,000

Linear Search	Binary Search
1 million comparisons	20 comparisons