Linear Search in Java

Searching

- Process of finding a given value position in a list of values.

Linear / Sequential Search

- basic and simple algorithm
- target value is compared with all elements in the list.

```
arr = [18, 21, 35, 77, 64, 23]
```
unsorted array

target = 77.

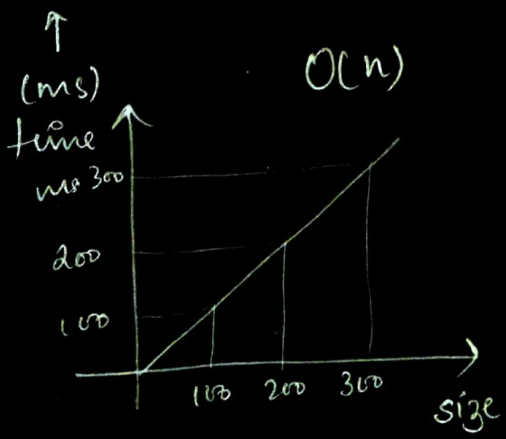
Time Complexity

Best case: O(1)

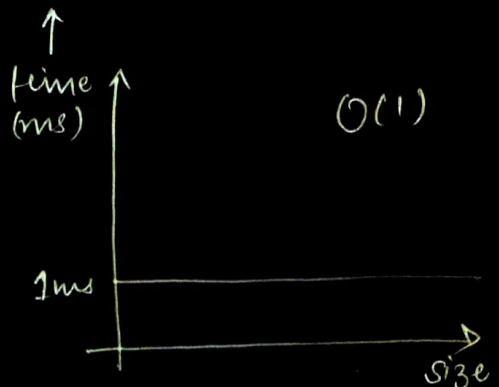
- target found at 0th index

Worst case: O(n)

- target not present in array.



O(n)



O(1)

* Note :-

```
char ch= sc.next().charAt(0);
```

↑ this is the correct syntax to take a character input.

Binary Search

Algorithm

sorted array.

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

ascending order.

target 36

1. Find middle element.
2. Check:
   if target > middle → search in right. (s = m+1)
   else → search in left. (e = m-1)
3. if target == middle ⇒ we found the element.

1st → first middle element.

$$mid = \frac{start + end}{2} = \frac{0 + 8}{2} = 4$$

2nd → target > mid

36 > 11 ⇒ yes → check in right side

3rd → at index 8,

target == middle

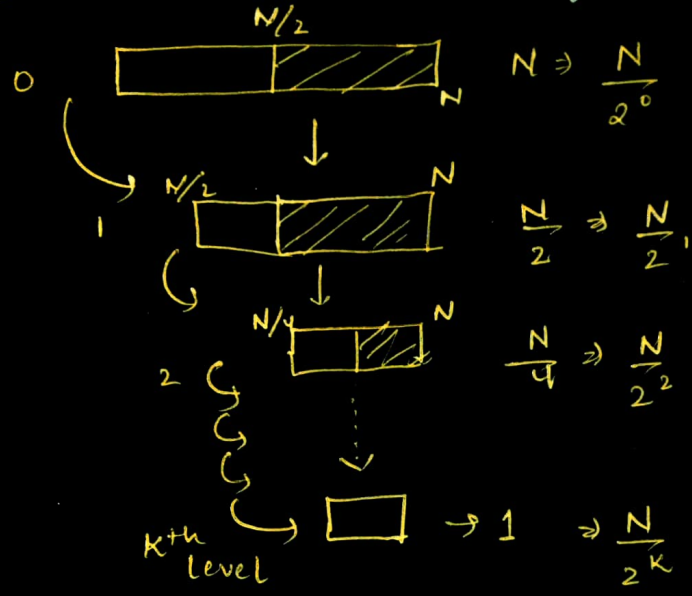
Time Complexity

Best Case : $O(1)$

Worst case : $O(\log n)$

explanation

↳ find the maximum no. of comparisons



at the end

only one element is left

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

$$N = 2^K$$

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

$$\log N = K \log 2$$

$$K = \log_2 N \longrightarrow \text{size of array}$$

↳ total no. of comparisons in worst case.

Order agnostic binary search

- let's say if the array is sorted but in which order, we don't know.

just check if start > end → descending order
else
→ ascending order.

also for descending.

if target > middle
↳ search in left.
(e = m - 1)

and if target < middle
↳ search in right.
(e = m + 1)