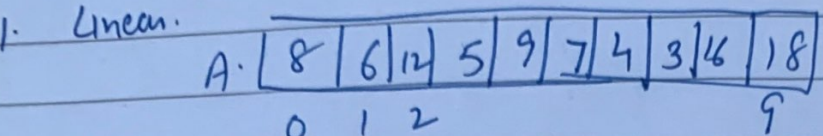
Best, Worst and Average case.

1. Linear Search
2. Binary Search tree

1. Linear.



1) Key = 7 (item to search)
comparison = 6
2) Key = 20. (not present)

Best case :- K=8 (if searching for first index element)
time → constant - O(1).
↳ B(n) = O(1)

worst case :- K=18 (last element as key element)
W(n) = n = O(n)
↓
no. of comparison.

Average case :- $\frac{\text{all possible case time}}{\text{no. of cases.}}$ (not possible to calculate for all.)

$= \frac{1+2+3+\dots+n}{n}$ → (no of comparison) function.

$A(n) = \frac{n+1}{2}$ [No. of comparisons are used to calculate time required]

(log2basic2.com)

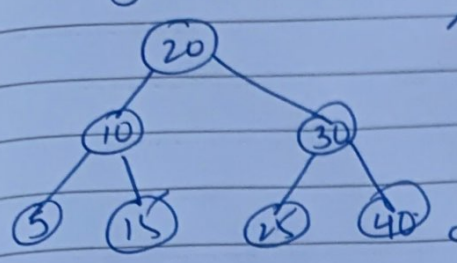
cases ≠ notations.

B(n) = 1 W(n) = n

B(n) = O(1) | Big O = O(n)
B(n) = Ω(1) | Omega = Ω(n)
B(n) = Θ(1) | theta. = Θ(n)

Any notation can be used.

2) Binary search tree.



height of tree = $\log n$. $n = 8$

$\log 8 = 3$.

1) Best case → search for element at root

$B(n) = 1$

2) worst case → search for leaf element i.e. 5, 15, 25, 40.

(depends on height)

$W(n) = \log n$. → min $W(n) = \log n$.

max $W(n) = n$.

If one sided tree.