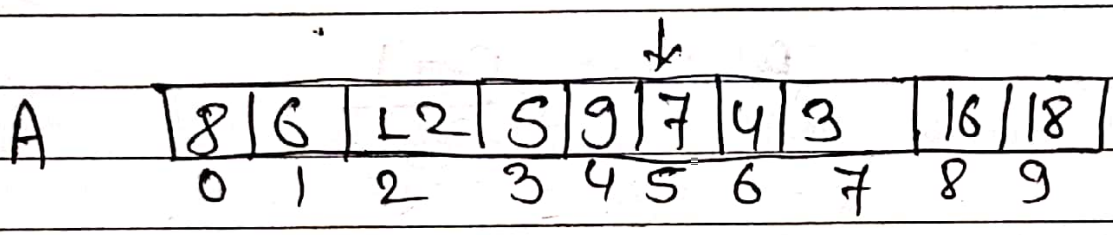
Best, worst and Average Case Analysis

- linear search
- Binary search Tree

linear search



Key = 7
Key = 20

Best Case — Searching key element present at first index.

Best Case Time — 1 $O(1)$

$B(n) = O(1)$

worst Case — Searching a key at last index

worst Case Time — n

$w(n) = n$

$O(n)$

Average case – $\frac{\text{all possible case time}}{\text{no. of cases}}$

$$= \frac{1+2+3+\cdots+n}{n}$$

$$= \frac{\frac{n(n+1)}{2}}{n} = \frac{n+1}{2}$$

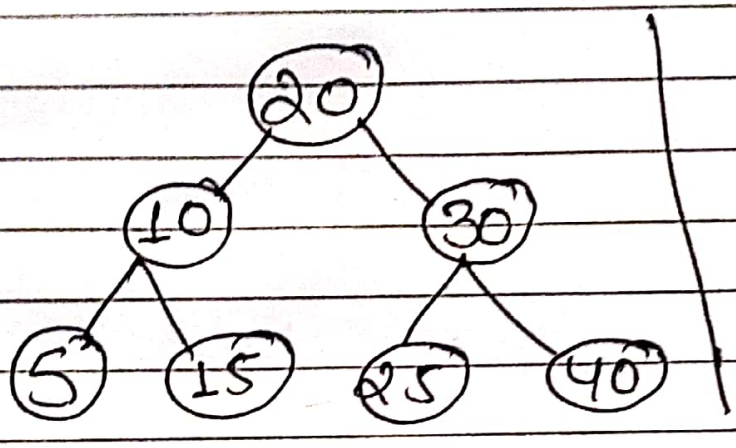
$$A(n) = \frac{n+1}{2}$$

$B(n) = 1$
$B(n) = O(1)$
$B(n) = \Omega(1)$
$B(n) = \Theta(1)$

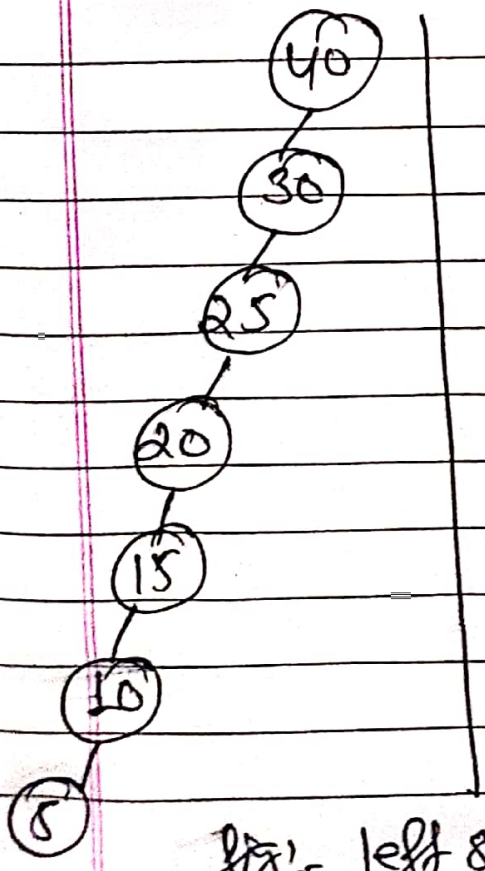
$w(n) = n$
$w(n) = O(n)$
$w(n) = \Omega(n)$
$w(n) = \Theta(n)$

We can write O, Ω & Θ for any notation (i.e. worst, best avg.)

Binary Search Tree (BST)



Best Case — Search root element
Best Case Time — $B(n) = 1$
Worst Case — Search for leaf element.
Worst Case Time — $W(n) = \log n$



$\min W(n) = \log n = h$
$\max . W(n) = n$

$h \rightarrow$ height of BST

Fig:- left skewed BST