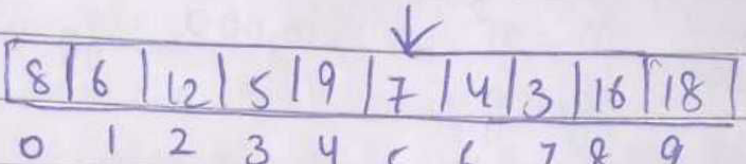
→ Best, Worst and Average case Analysis

1. Linear search
2. Binary search Tree

→ Linear search

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




key = 7 → Total Comparison = 6
key = 20 → Total comparison = 10

- Best element → 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 - $\frac{\text{all possible case time}}{\text{no. of cases}}$

- Average case time - $\frac{1+2+3+\cdots+n}{n} = \frac{n(n+1)/2}{n} = \frac{n+1}{2}$

→ Asymptotic Notation of Best, Worst and Average Case Time:

- Best case time:

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

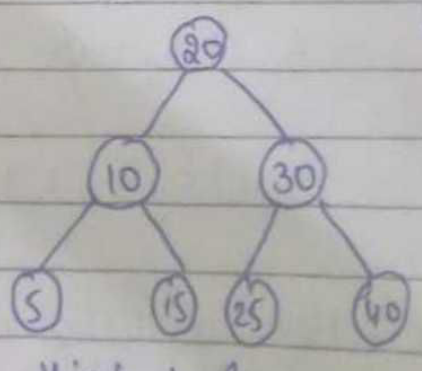
- Worst case time:

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

---

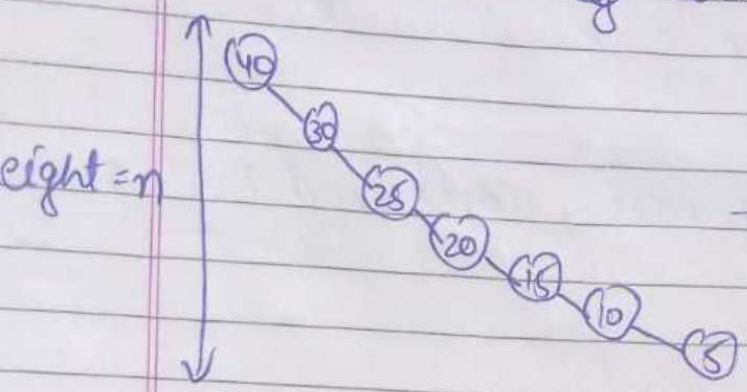
→ Binary search Tree

Key = 15 - Total comparison = 3
$\log 8 = 3$




High-balanced Binary search Tree

Best case - searching root element
Best case time - $B(n) = 1$
Worst case - searching for leaf element
eg. 5, 15, 25, 40
Worst case time - $W(n) = \log(n)$ = height of tree



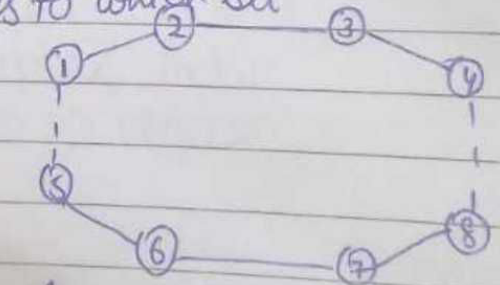

→ Left skew binary search Tree

$\therefore$ min $W(n) = \log n$
max $W(n) = n$

---

→ Disjoint sets ~~Data Structure~~

1. Find → element belongs to which set
2. Union




$S_1 = \{1, 2, 3, 4\}$

$S_2 = \{5, 6, 7, 8\}$ ∪ ∨ ∨ ∨ ∨
(4, 8) (1, 5)

$S_1 \cap S_2 = \phi$ → Intersection of two set

$S_1 \cup S_2 = \{1, 2, 3, 4, 5, 6, 7, 8\} = S_3$

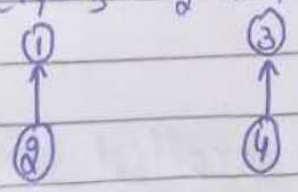
Take a edge if both vertices belongs to same set then there is a cycle in a graph.

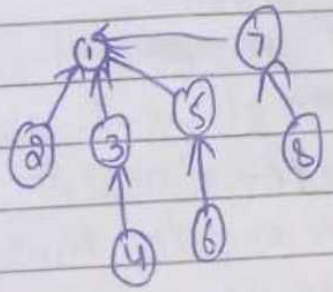
Date
Page

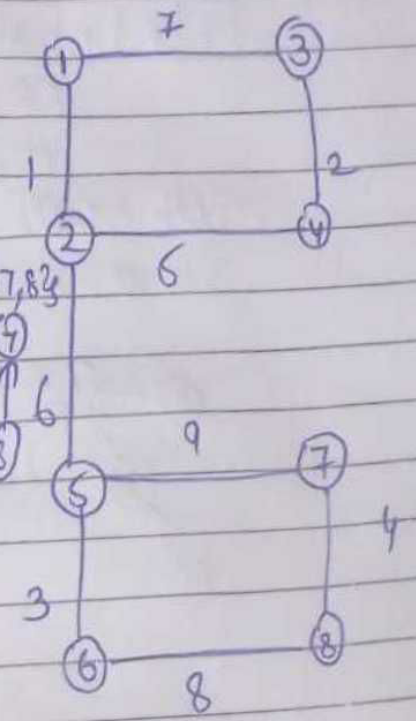
eg. Graphical representation

$U = \{1, 2, 3, 4, 5, 6, 7, 8\}$
(1) (2) (3) (4) (5) (6) (7) (8)

$S_1 = \{1, 2\}$ $S_2 = \{3, 4\}$ $S_3 = \{5, 6\}$ $S_4 = \{7, 8\}$








Same graph

- Array representation

$U = \{1, 2, 3, 4, 5, 6, 7, 8\}$
① ② ③ ④ ⑤ ⑥ ⑦ ⑧

| -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Parent

| -2 | 1 | -2 | 3 | -2 | 5 | -2 | 7 |
|---|---|---|---|---|---|---|---|

(1,2) (3,4) (5,6) (7,8)

| -4 | 1 | 1 | 3 | -2 | 5 | -2 | 7 |
|---|---|---|---|---|---|---|---|

(2,4) (2,8)

| -6 | 1 | 1 | 3 | 1 | 5 | -2 | 7 |
|---|---|---|---|---|---|---|---|

(2,6)

→

| -6 | 1 | 1 | 3 | 1 | 5 | -2 | 7 |
|---|---|---|---|---|---|---|---|

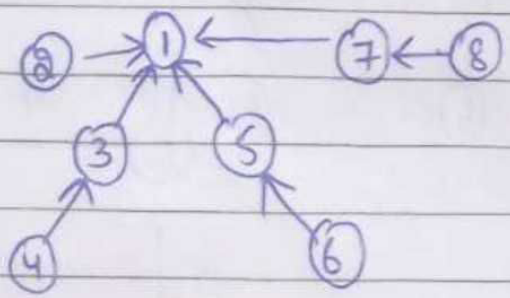
(1,3)

Both parents are same
This will form a cycle

| -8 | 1 | 1 | 3 | 1 | 5 | 1 | 7 |
|---|---|---|---|---|---|---|---|

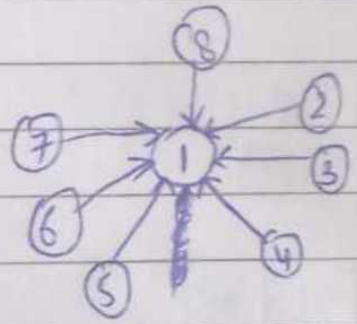
(6, 8)
(5, 7)

Both exists in same set
This forms a cycle




→ This is called weighted Union
That is set with more elements always become parent

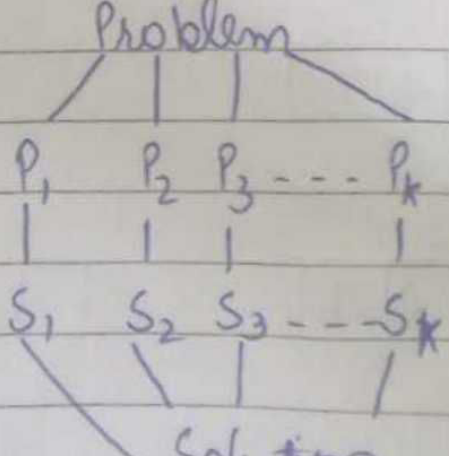
Collapsing find → We can reduce time to find value




---

→ Divide and Conquer

- Recursive in nature




---

```
DAC(P)
{
    if (small(P))
    {
        S(P);
    }
    else
    {
        divide P into P1, P2, P3 -- Pk
        Apply DAC(P1), DAC(P2), - - -
        Combine (DAC(P1), DAC(P2), - - -)
    }
}
```

-- * ---- * ---- * ---- * ---- * ---- * ---- * ---- * -