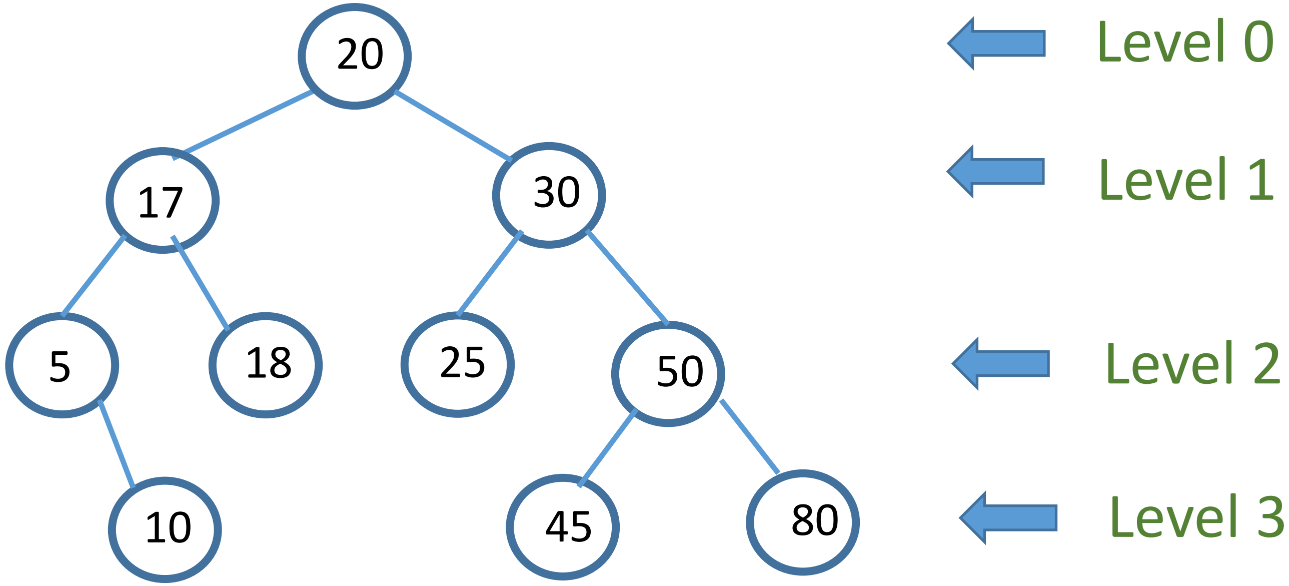
COMP 2611, DATA STRUCTURES
LECTURE 11

RETURN TO BINARY TREES

- **Performing a Level Order Traversal**
- **Types of Binary Trees**

Level Order Traversal of a Binary Tree



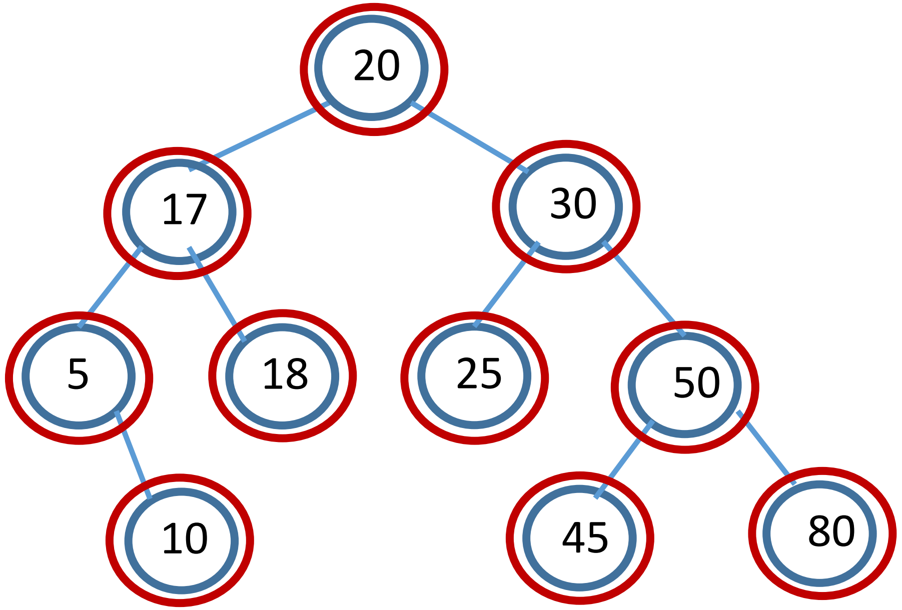
Level 0

Level 1

Level 2

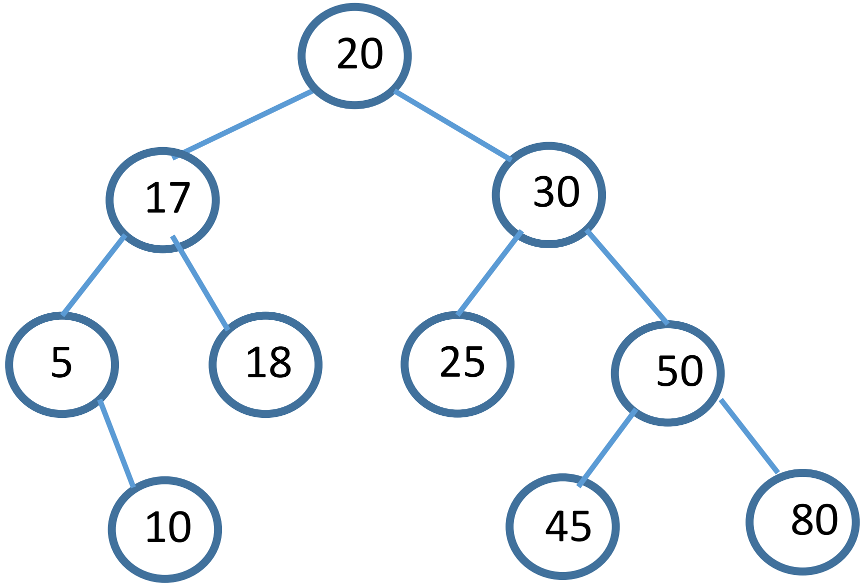
Level 3

Level Order Traversal of a Binary Tree



```
initialize a queue, Q
insert the root of the BT in Q
while (Q is not empty) {
        set p = dequeue (Q)
        visit p
        if left (p) is not null
                insert left(p) in Q
        if right (p) is not null
                insert right (p) in Q
}
```

Level Order Traversal of a Binary Tree

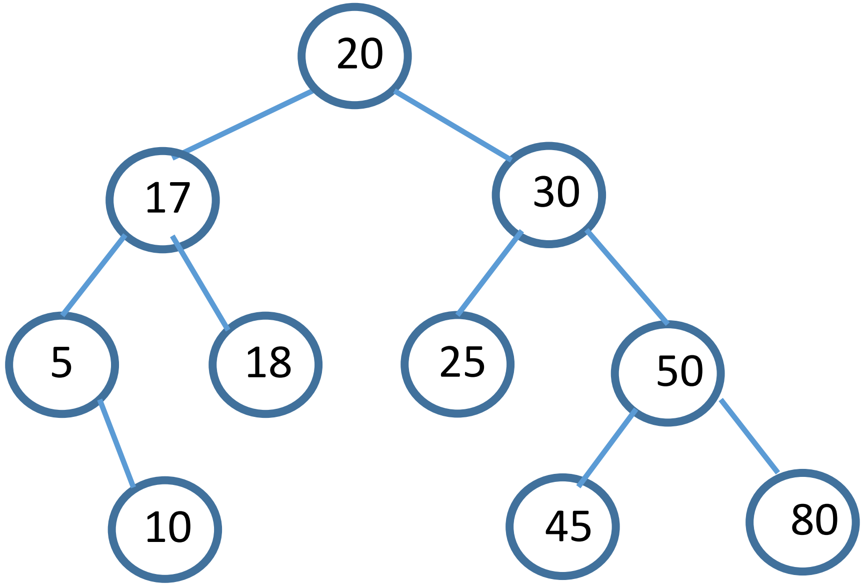


```
initialize a queue, Q
insert the root of the BT in Q
while (Q is not empty) {
        set p = dequeue (Q)
        visit p
        if left (p) is not null
                insert left(p) in Q
        if right (p) is not null
                insert right (p) in Q
}
```

Queue:

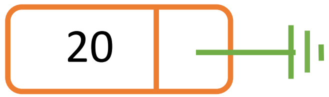
Output:

Level Order Traversal of a Binary Tree



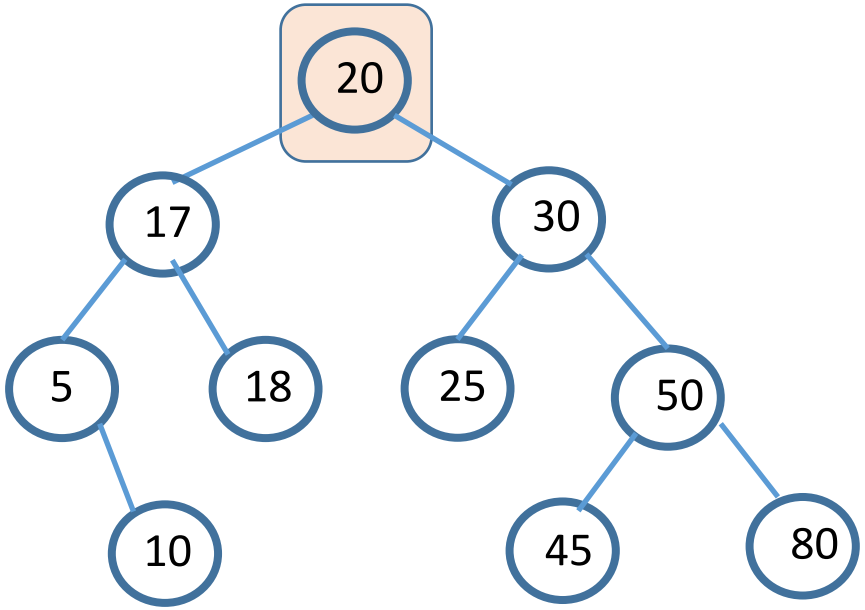
```
initialize a queue, Q
insert the root of the BT in Q
while (Q is not empty) {
        set p = dequeue (Q)
        visit p
        if left (p) is not null
                insert left(p) in Q
        if right (p) is not null
                insert right (p) in Q
}
```

Queue:



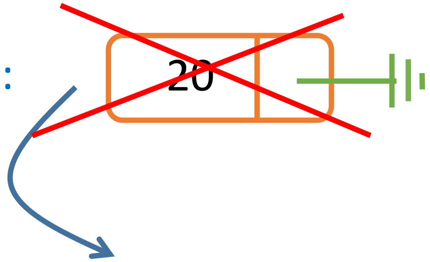
Output:

Level Order Traversal of a Binary Tree



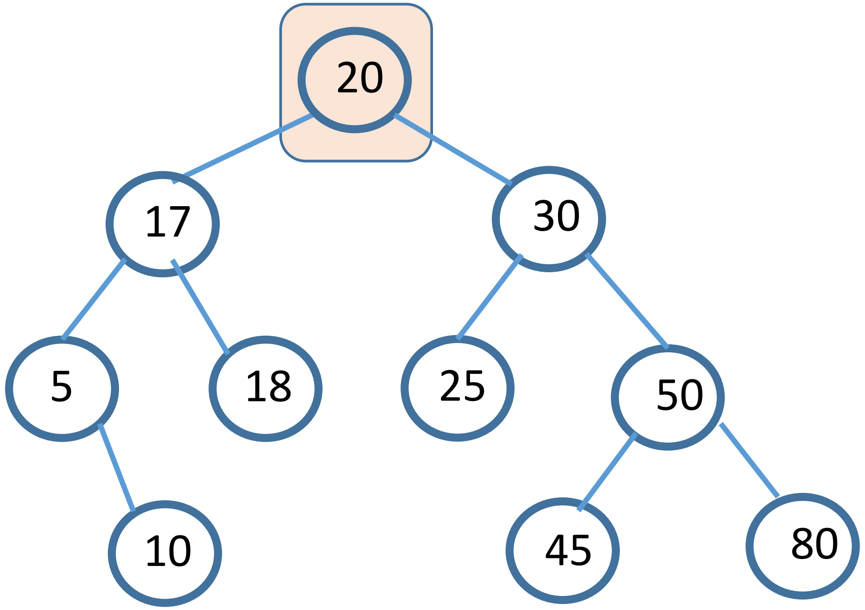
```
initialize a queue, Q
insert the root of the BT in Q
while (Q is not empty) {
      set p = dequeue (Q)
      visit p
      if left (p) is not null
            insert left(p) in Q
      if right (p) is not null
            insert right (p) in Q
}
```

Queue:



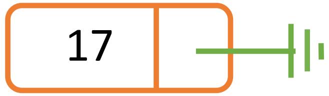
Output: 20

Level Order Traversal of a Binary Tree



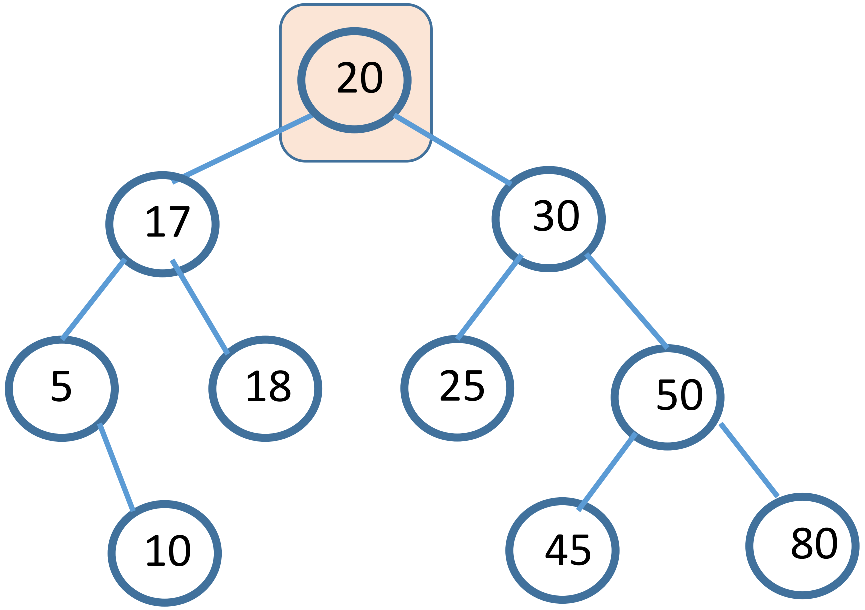
```
initialize a queue, Q
insert the root of the BT in Q
while (Q is not empty) {
        set p = dequeue (Q)
        visit p
        if left (p) is not null
                insert left(p) in Q
        if right (p) is not null
                insert right (p) in Q
}
```

Queue:



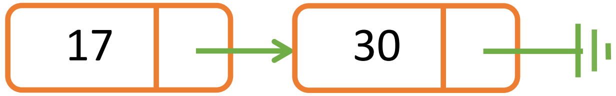
Output: 20

Level Order Traversal of a Binary Tree



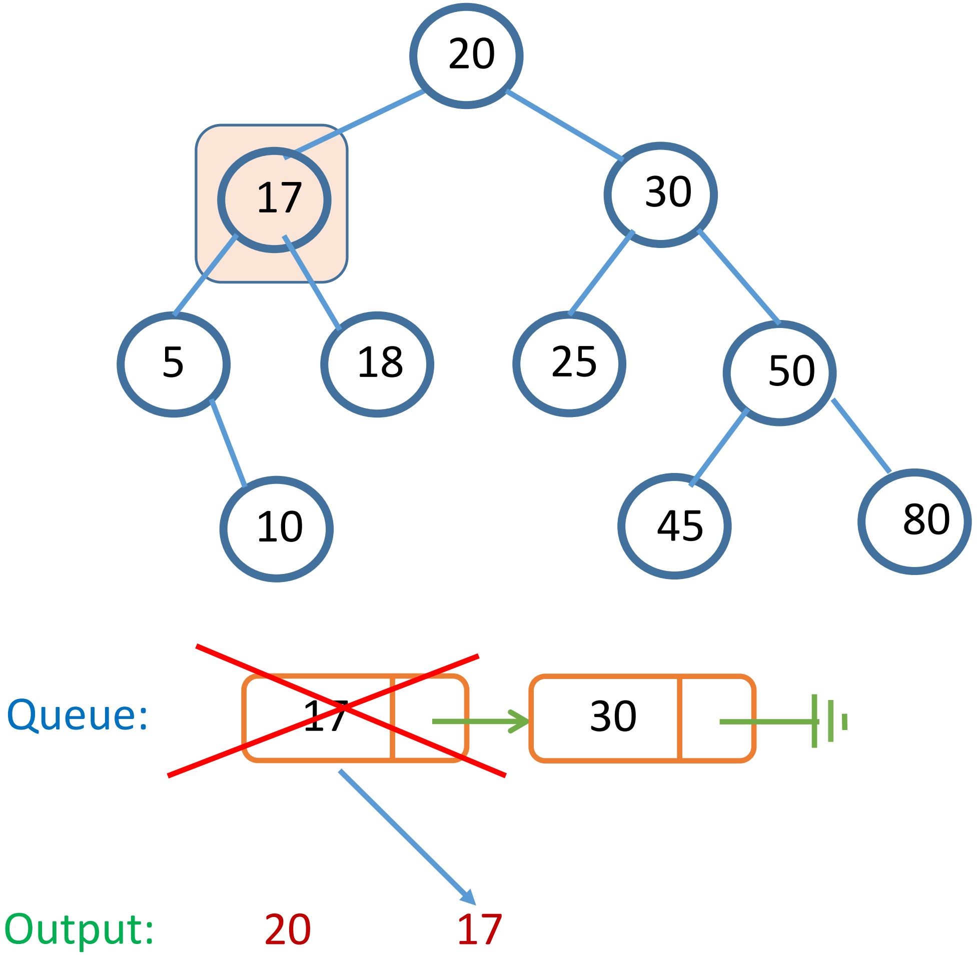
```
initialize a queue, Q
insert the root of the BT in Q
while (Q is not empty) {
        set p = dequeue (Q)
        visit p
        if left (p) is not null
                insert left(p) in Q
        if right (p) is not null
                insert right (p) in Q
}
```

Queue:



Output: 20

Level Order Traversal of a Binary Tree

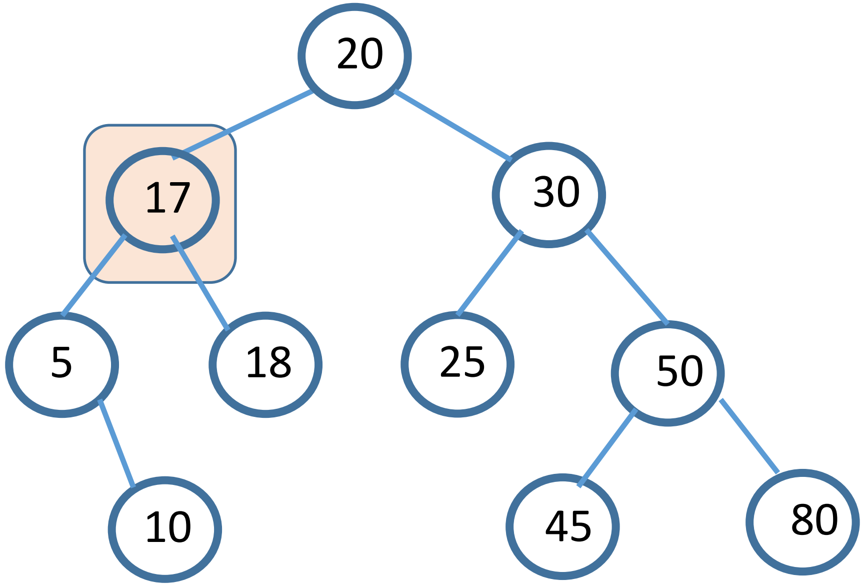


Queue: ~~17~~ 30

Output: 20 17

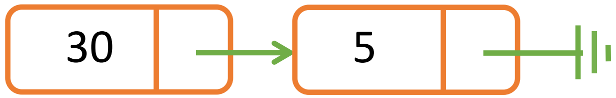
```
initialize a queue, Q
insert the root of the BT in Q
while (Q is not empty) {
        set p = dequeue (Q)
        visit p
        if left (p) is not null
                insert left(p) in Q
        if right (p) is not null
                insert right (p) in Q
}
```

Level Order Traversal of a Binary Tree



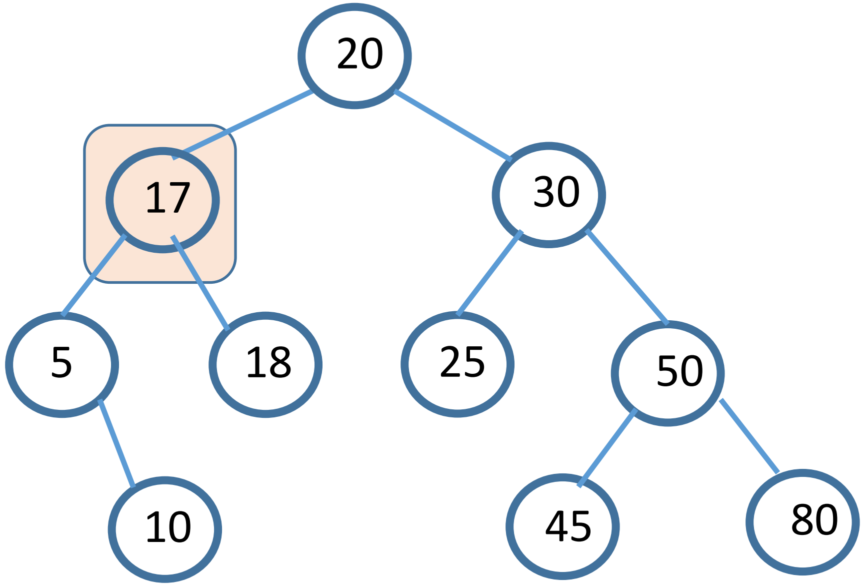
```
initialize a queue, Q
insert the root of the BT in Q
while (Q is not empty) {
      set p = dequeue (Q)
      visit p
      if left (p) is not null
            insert left(p) in Q
      if right (p) is not null
            insert right (p) in Q
}
```

Queue:



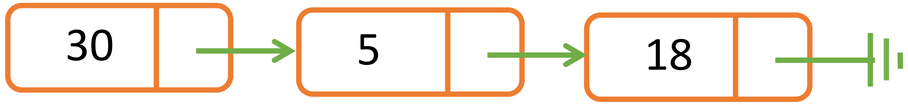
Output: 20 17

Level Order Traversal of a Binary Tree



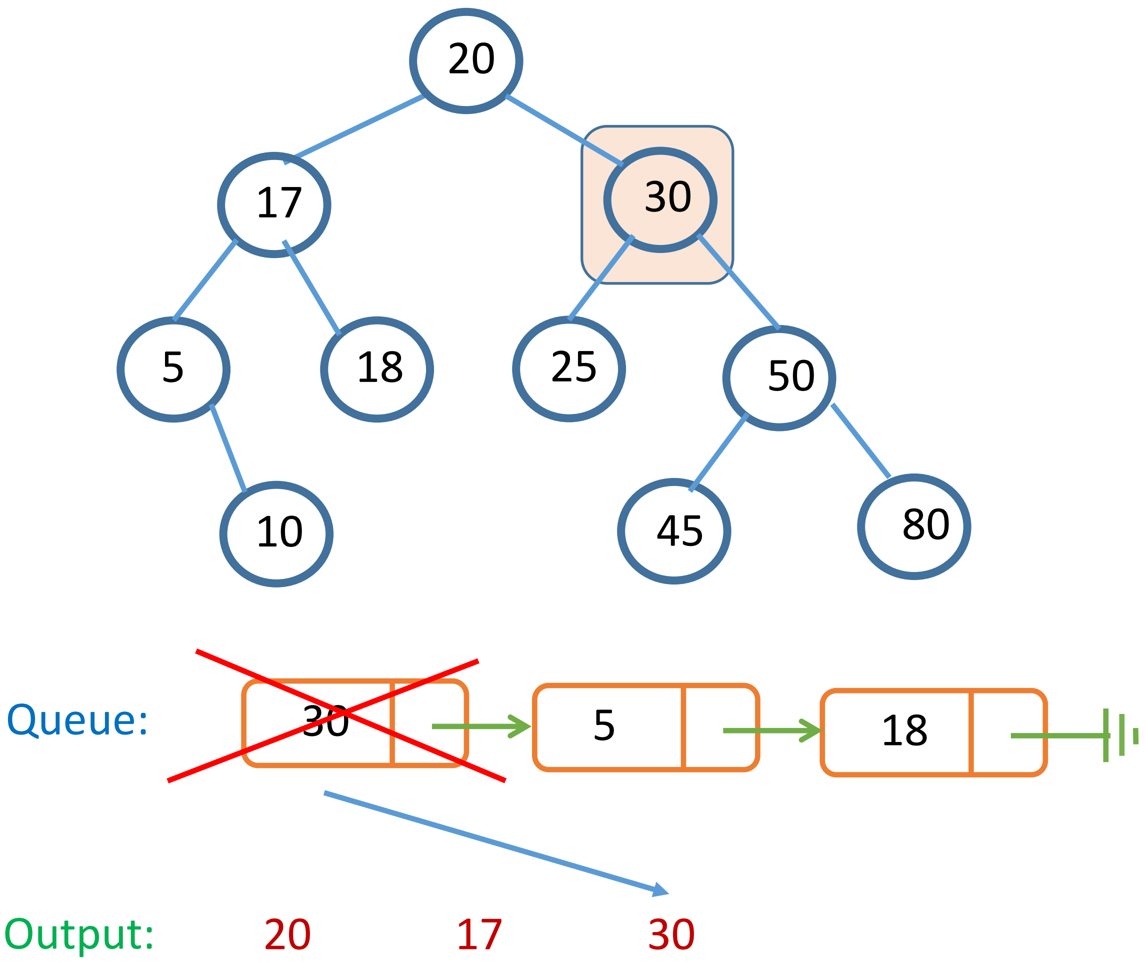
```
initialize a queue, Q
insert the root of the BT in Q
while (Q is not empty) {
        set p = dequeue (Q)
        visit p
        if left (p) is not null
                insert left(p) in Q
        if right (p) is not null
                insert right (p) in Q
}
```

Queue:



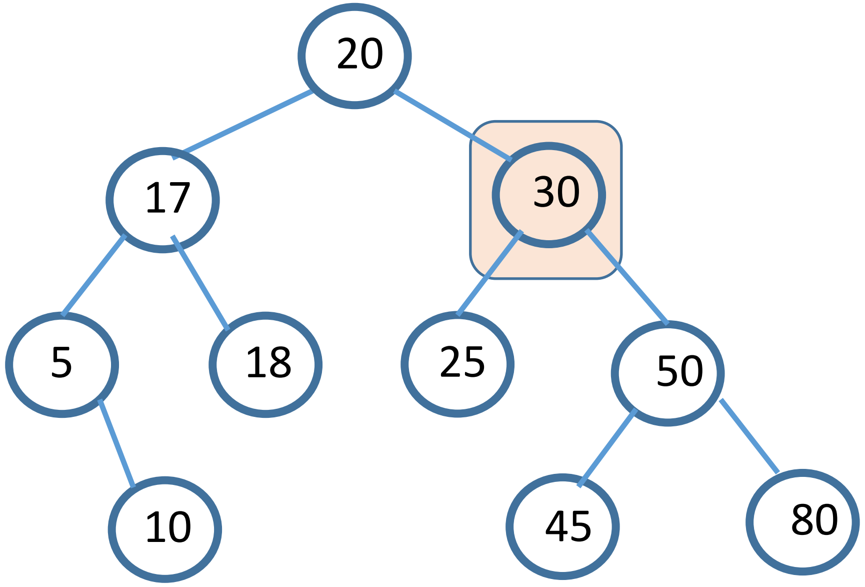
Output: 20 17

Level Order Traversal of a Binary Tree



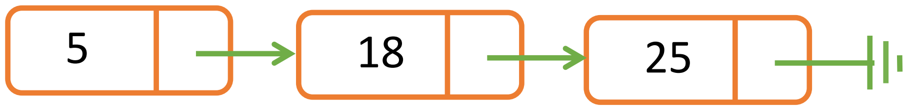
```
initialize a queue, Q
insert the root of the BT in Q
while (Q is not empty) {
        set p = dequeue (Q)
        visit p
        if left (p) is not null
                insert left(p) in Q
        if right (p) is not null
                insert right (p) in Q
}
```

Level Order Traversal of a Binary Tree



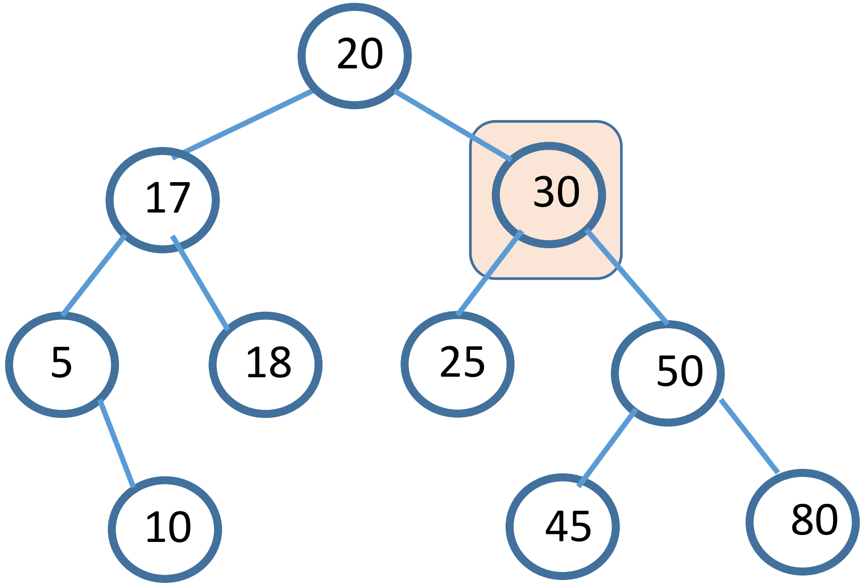
```
initialize a queue, Q
insert the root of the BT in Q
while (Q is not empty) {
        set p = dequeue (Q)
        visit p
        if left (p) is not null
                insert left(p) in Q
        if right (p) is not null
                insert right (p) in Q
}
```

Queue:



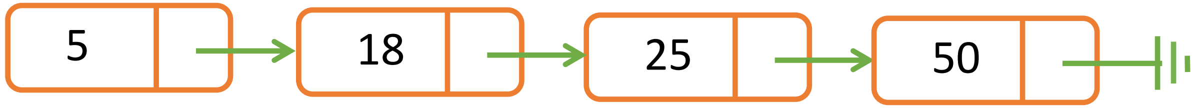
Output: 20 17 30

Level Order Traversal of a Binary Tree



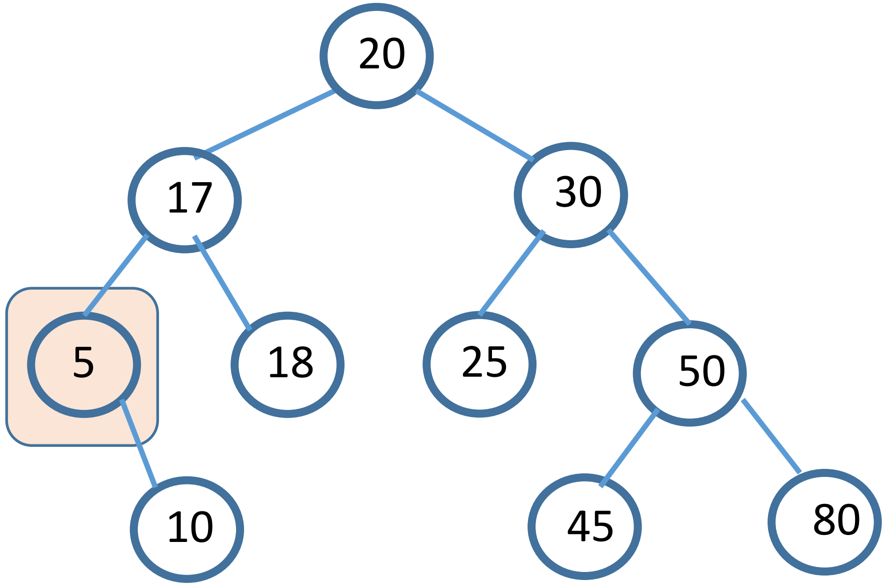
```
initialize a queue, Q
insert the root of the BT in Q
while (Q is not empty) {
        set p = dequeue (Q)
        visit p
        if left (p) is not null
                insert left(p) in Q
        if right (p) is not null
                insert right (p) in Q
}
```

Queue:



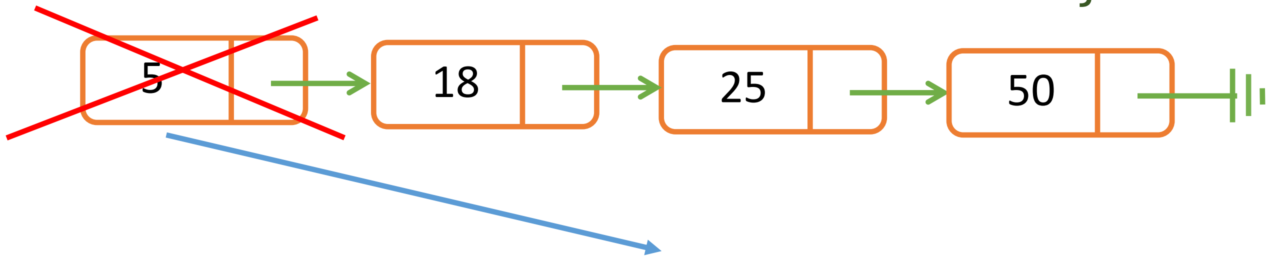
Output: 20 17 30

Level Order Traversal of a Binary Tree



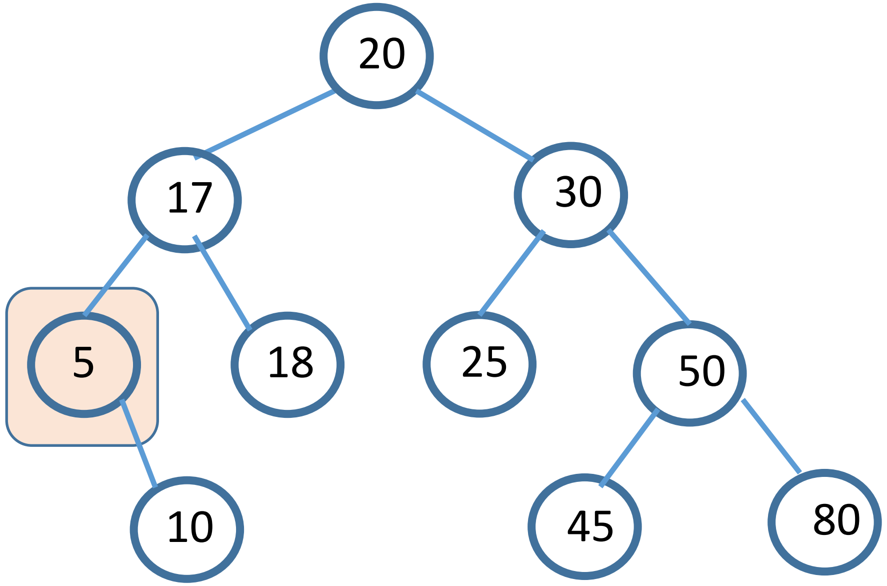
```
initialize a queue, Q
insert the root of the BT in Q
while (Q is not empty) {
        set p = dequeue (Q)
        visit p
        if left (p) is not null
                insert left(p) in Q
        if right (p) is not null
                insert right (p) in Q
}
```

Queue:



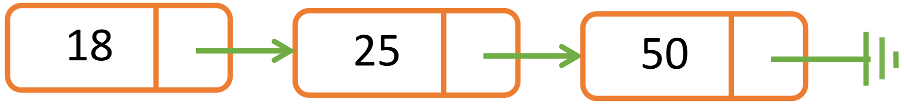
Output: 20 17 30 5

Level Order Traversal of a Binary Tree



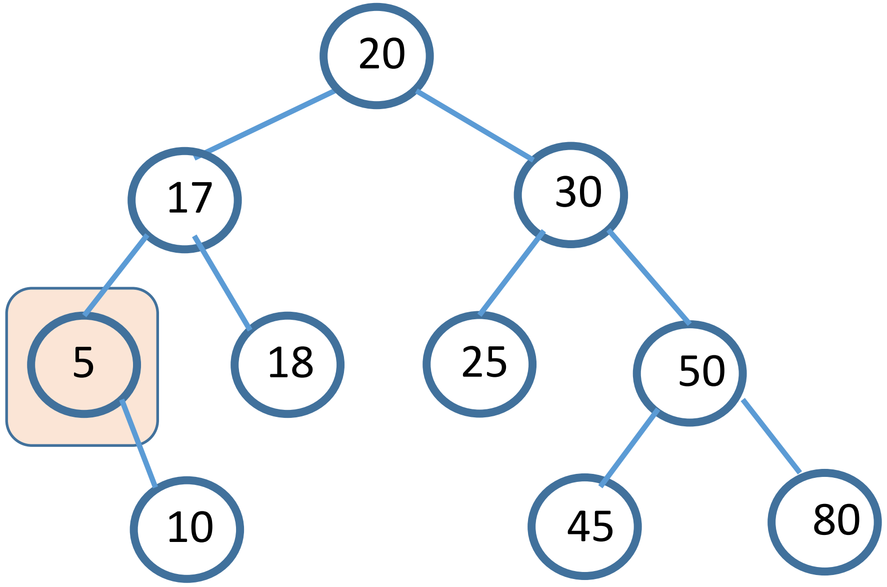
```
initialize a queue, Q
insert the root of the BT in Q
while (Q is not empty) {
        set p = dequeue (Q)
        visit p
        if left (p) is not null
                insert left(p) in Q
        if right (p) is not null
                insert right (p) in Q
}
```

Queue:



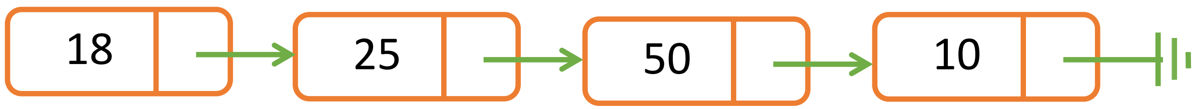
Output: 20 17 30 5

Level Order Traversal of a Binary Tree



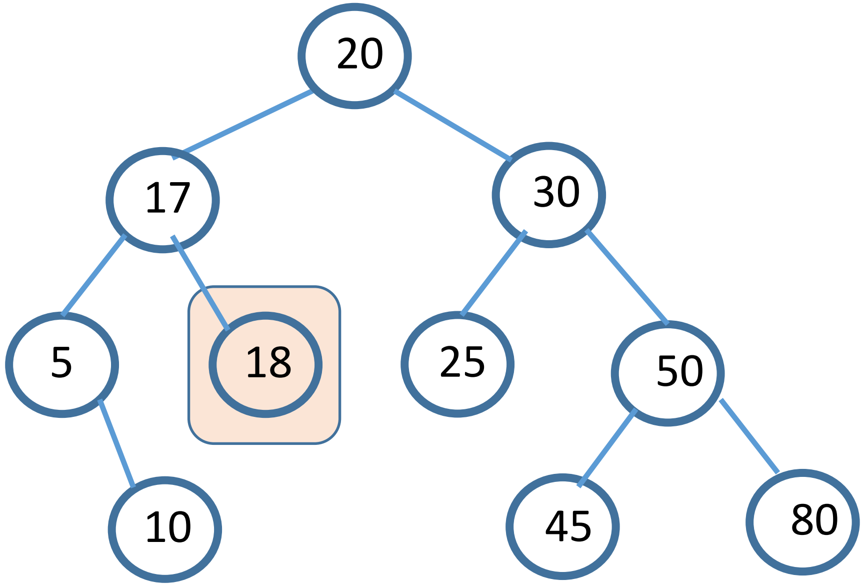
```
initialize a queue, Q
insert the root of the BT in Q
while (Q is not empty) {
        set p = dequeue (Q)
        visit p
        if left (p) is not null
                insert left(p) in Q
        if right (p) is not null
                insert right (p) in Q
}
```

Queue:



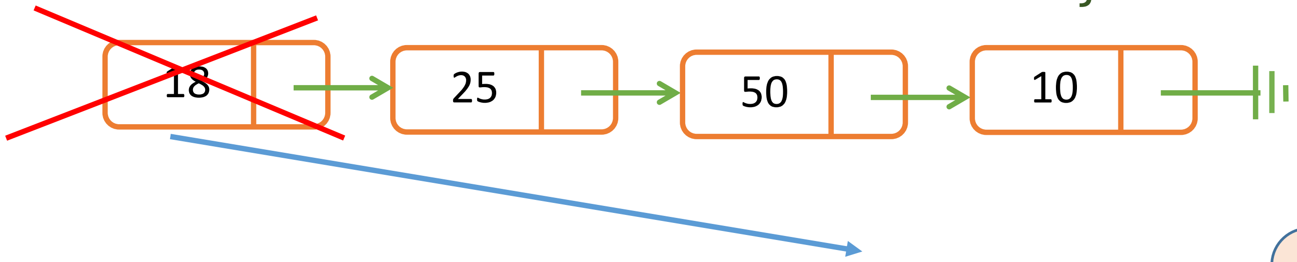
Output: 20 17 30 5

Level Order Traversal of a Binary Tree



```
initialize a queue, Q
insert the root of the BT in Q
while (Q is not empty) {
      set p = dequeue (Q)
      visit p
      if left (p) is not null
            insert left(p) in Q
      if right (p) is not null
            insert right (p) in Q
}
```

Queue:



Output: 20 17 30 5 18

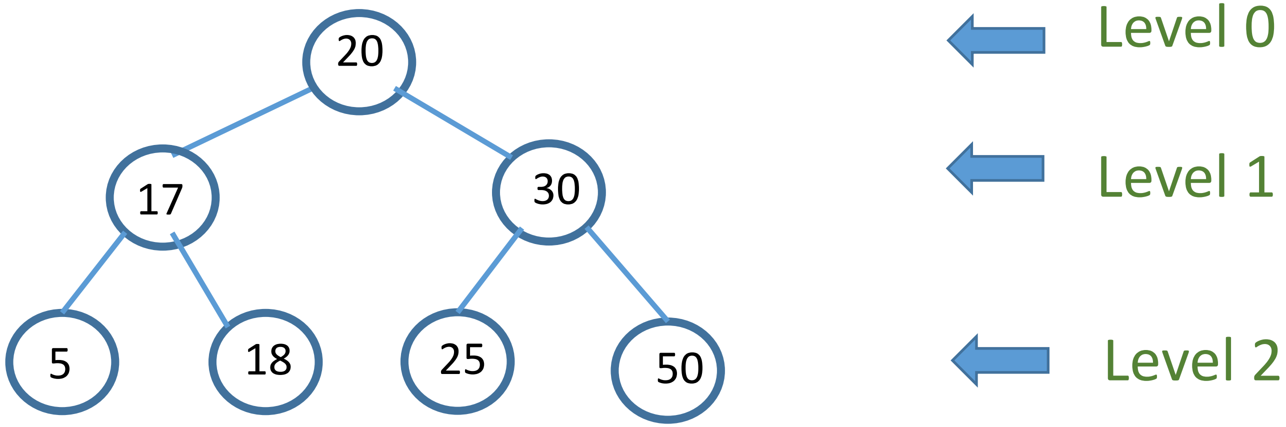
Process continues until the queue is empty – at which point, the level-order traversal is complete.

Types of Binary Trees

1. Complete
2. Almost complete
3. Full

Complete Binary Tree

- A *complete* binary tree is one in which:
  - ✓ Every non-leaf node has two non-empty subtrees, and
  - ✓ All leaves are at the same level.

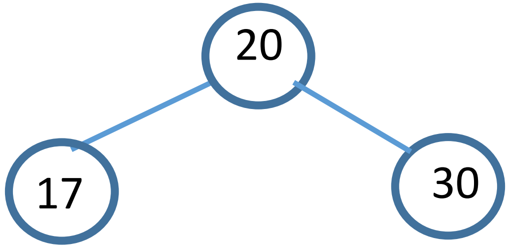


Complete Binary Tree

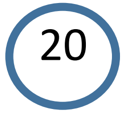
- Which of the following binary trees are complete?

- ✓ Every non-leaf node has two non-empty subtrees, and
- ✓ All leaves are at the same level.

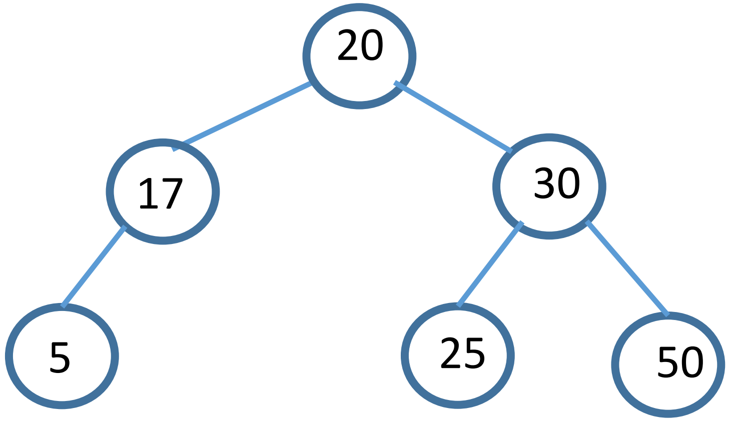
(a)



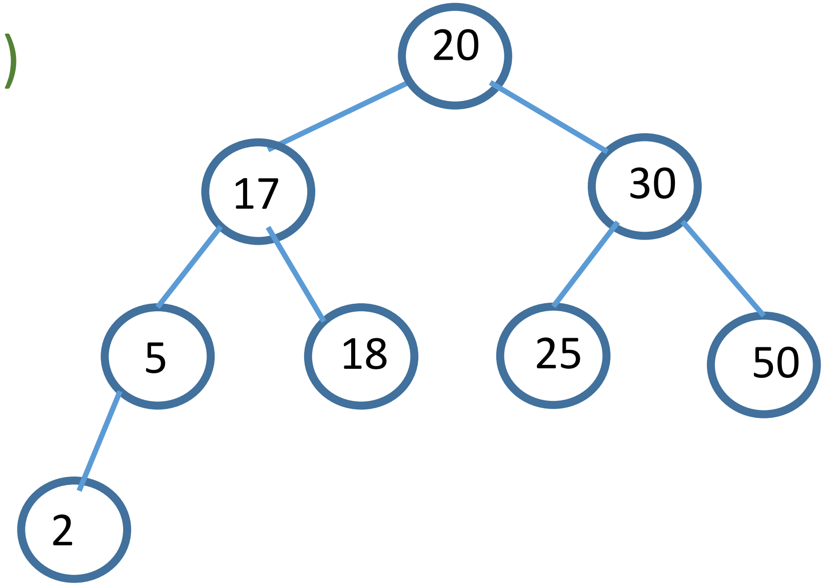
(b)



(c)

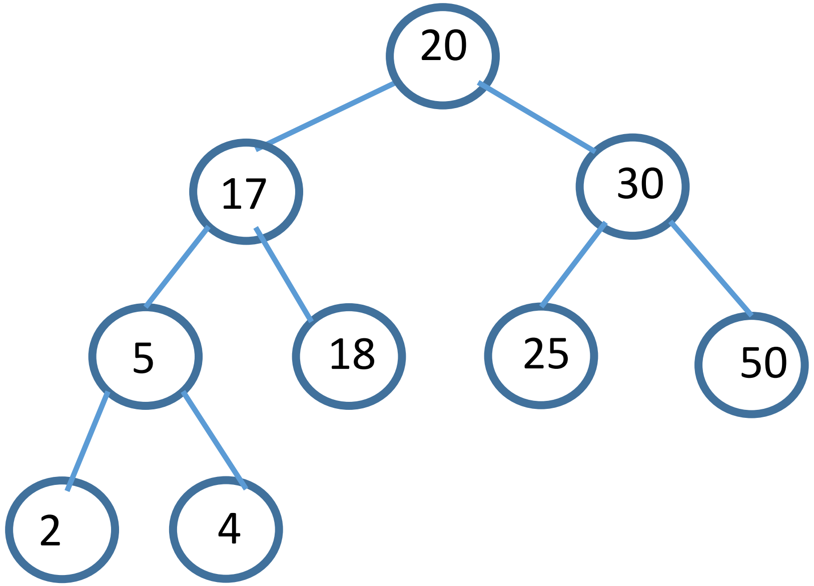


(d)



Almost Complete Binary Tree

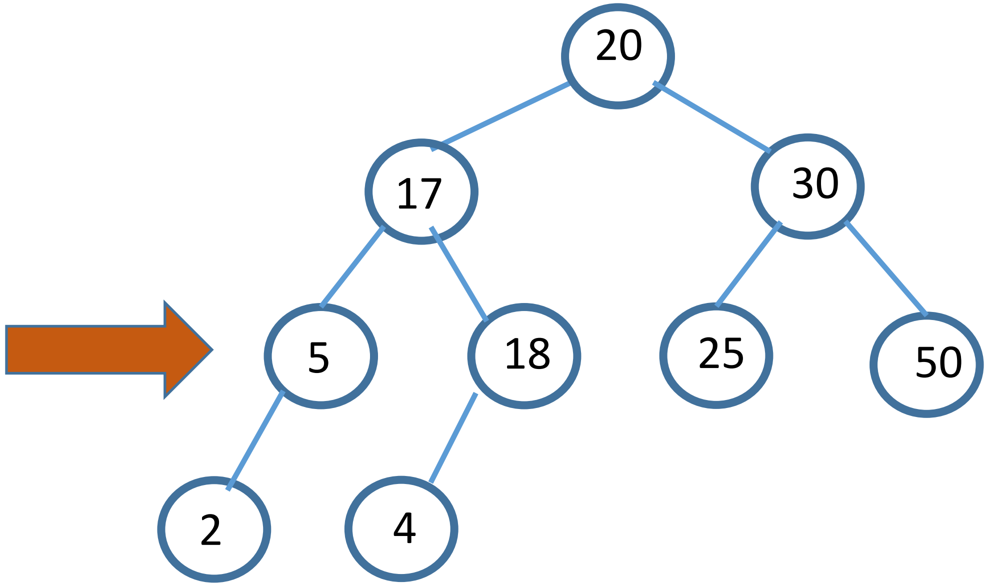
- An *almost complete* binary tree is one in which:
  - ✓ All levels, except possibly the lowest, are completely filled.
  - ✓ The nodes at the lowest level (all leaves) are as far left as possible.



Almost Complete Binary Tree

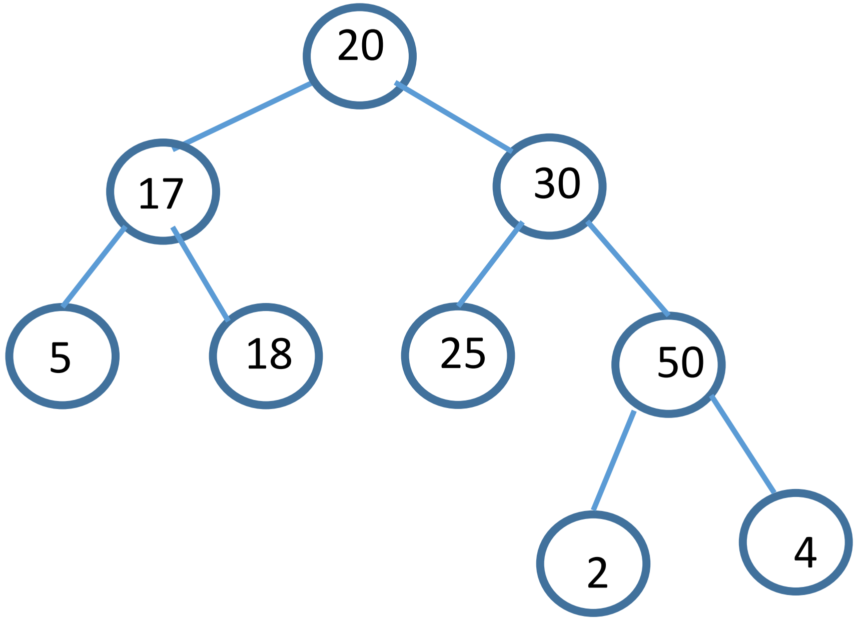
- Is the following binary tree almost complete?

Problem here!



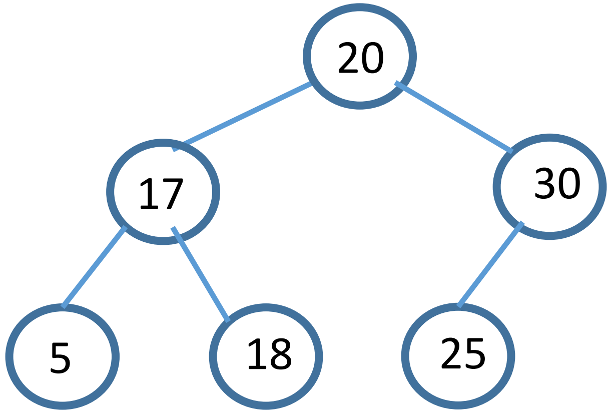
Full Binary Tree

- A *full* binary tree is one in which:
  - ✓ Every non-leaf node has exactly two non-empty subtrees.

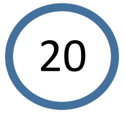


Full Binary Tree

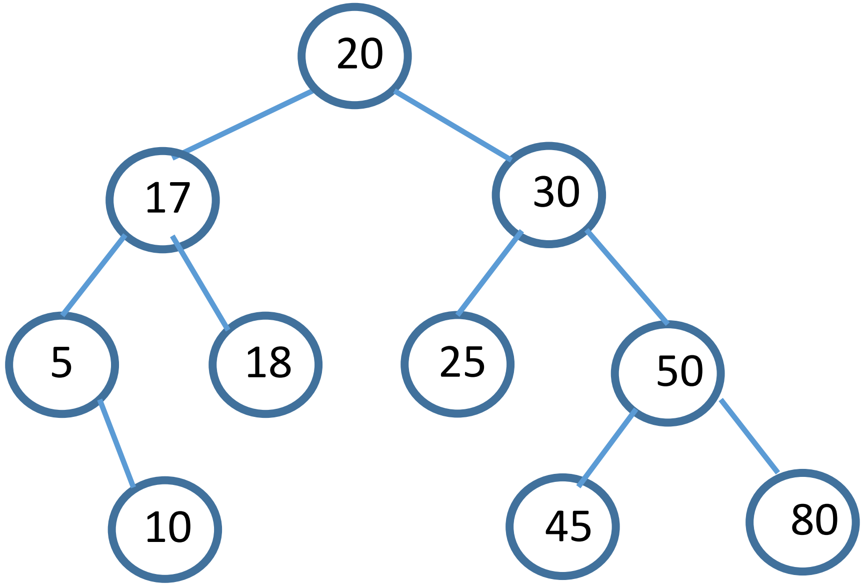
- Is the following binary tree full?



- Is the following binary tree full?



Performance Analysis of a Binary Search Tree



- How does it compare to linked list?
- How does it compare to array?