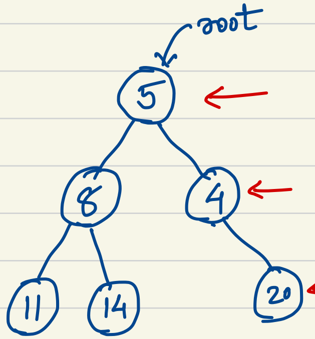
RIGHT SIDE VIEW - BINARY TREE



LEVEL ORDER

→ [5]

→ [8, 4]

→ [11, 14, 20]

Observe: we just need rightmost

OUTPUT:

```
[
  [5],
  [8, 4],
  [11, 14, 20]
]
```

IDEA - BFS TRAVERSAL

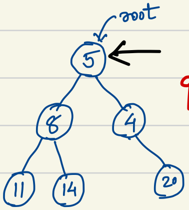
Node Count at each level

KEY POINTS:

i ← 0 to nodeCount-1
when i = = nodeCount -1
its rightmost node
in that level

L.L. : LinkedList

- → QUEUE IS PROCESSED LEVEL BY LEVEL.
- → EACH NODE ONCE PROCESSED, PUTS ITS CHILDREN IN QUEUE.



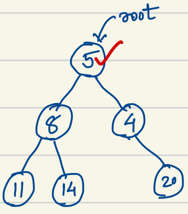
q.add(source)

nodecount = 1

q | 5 |

i = 0 = nodecount - 1

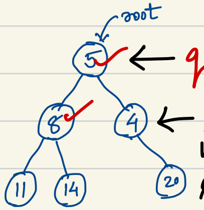
result = [5]



nodecount = 2

q | 8 | 4 |

i = 0



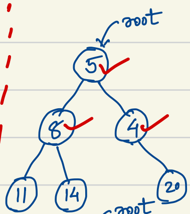
q | 4 | 11 | 14 |

i = 1

i = nodecount - 1

result

[5, 4]

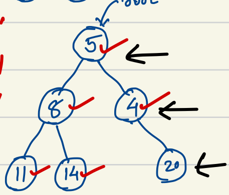


nodecount = 3

q | 11 | 14 | 20 |

i = 0

i = 1

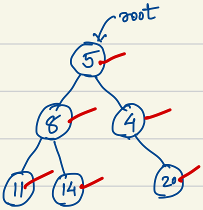


q | 20 |

i = 2 = nodecount - 1

result = [5, 4, 20]

q | |
empty



ALGO:

```
result ← new LL();
q ← new LL();
if root == NULL RET result
q.add(root)
while(!q.isEmpty()){
  nodeCount ← q.size();
  for i ← 0 to nodecount-1
     current ← q.poll();
     if i == nodecount -1
        result.add(current)
     if current.left != null
        q.add(current.left)
     if current.right != null
        q.add(current.right)
}
```

YASH PRADHAN
STUDY NOTES