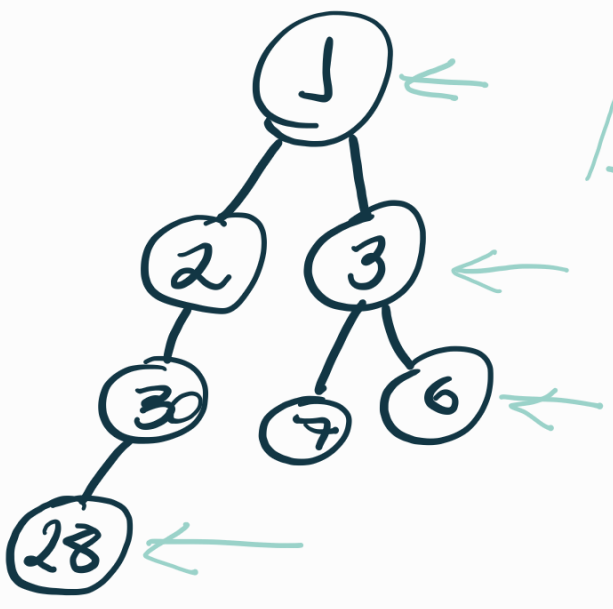
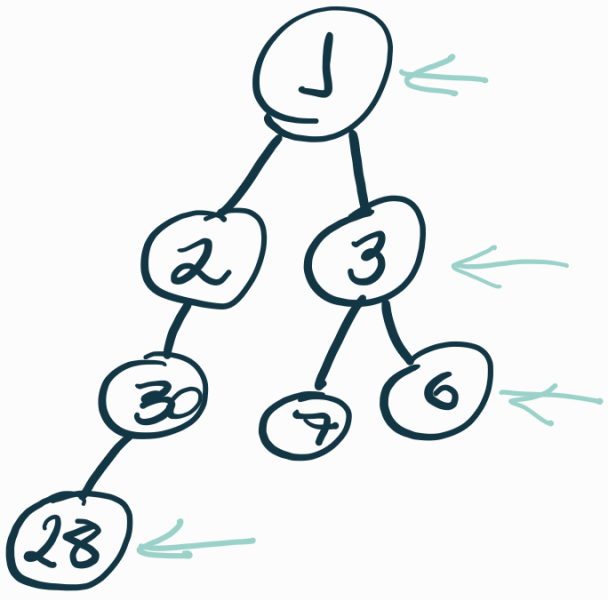
- Given the root of a BT
  1. list only the values of the nodes that are on the outer right side of it
  e.g:



Answer: [1, 3, 6, 28]

obs: final list has to be ordered top to bottom.

- Root also counts

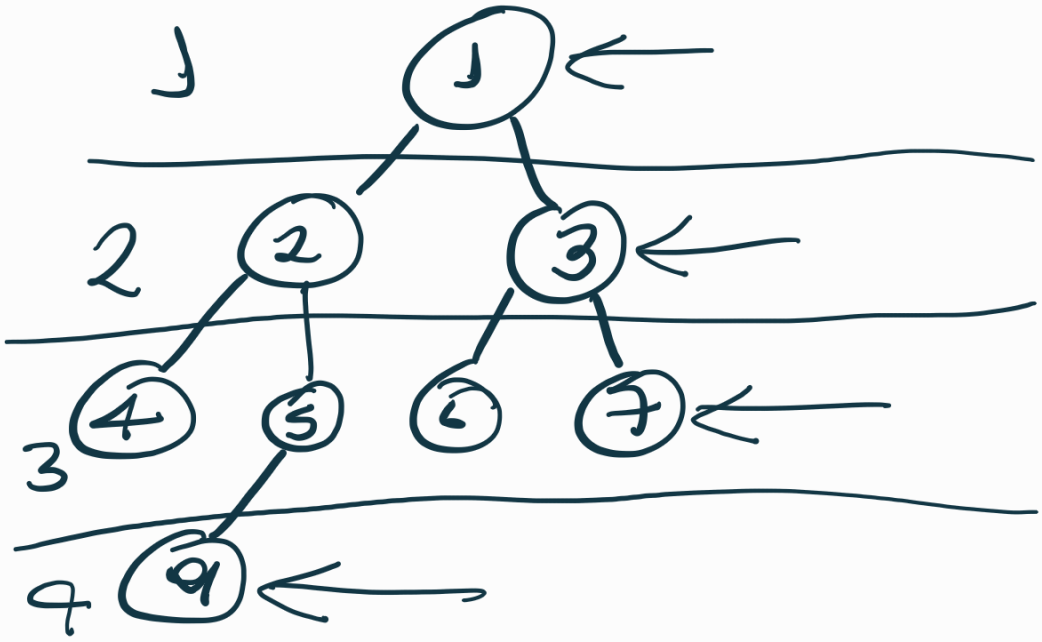


→ As right-most items matter the most, focus them first before checking left

→ traverse right sides and add them to the list

→ Every recursion should should know the number of its current layer.

↳ So as you do a deep dive pass it as an incremental parameter.



→ each level can have only one valid value on the result list.

→ given that layer 0 (root) will always add one element to the result on the following recursion

- if the current size of my result array equals my current layer, then i KNOW the current layer im in hasnt added an value yet.
  - and thus has to be added.

→ for this to work root (Source) has to be handled as layer 0 instead of layer 1.

→ with this approach, if left side & root has more layers than right, then they will all be accounted for.

└→ As result will only account for layers that haven't had values accounted until its traversal.

Alternate Solution:

↓ ↓ ↓

→ I could also think of it in terms of layers (levels)

- each layer has an right-most node

→ Create an storage <K, V>

- Run through the whole right side and add values to its respective layers.

→ Obs: if a layer already has

an value in storage then
You cant replace it

→ After visiting the inside right side, do visit left side and add the extra layers that were not present on the right

↳ finally after having traversed it all

↳ create the <K, V> item in storage and create

an list from it.

L> then return that list.