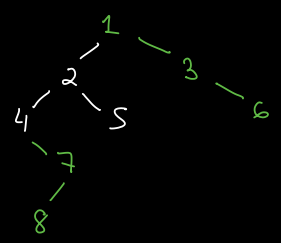
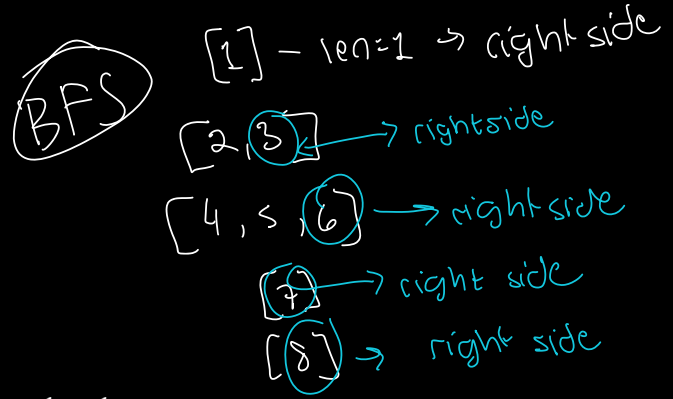
Problem: Given a binary tree, imagine standing to right side of tree return array of values seen top to bottom

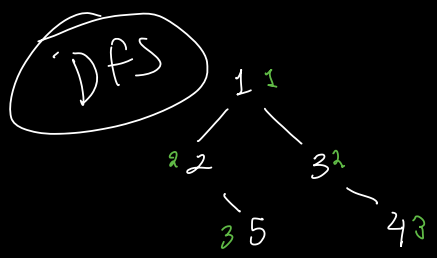


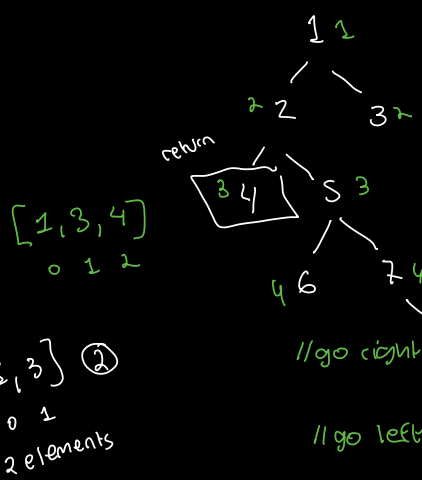
=> another bfs problem
level order traversal

Intro:
- Verify Constraints
- Create Testcases

Brute Force:
- Brainstorming & Pattern Observations







- Pseudocode
- Write code
- Run through testcases
- Analyze time and space complexity
  - Time: O(n)
  - Space: O(n)

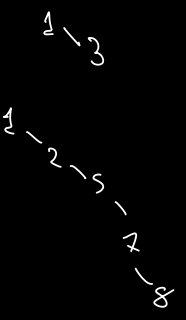
BFS: worst: full + complete tree (widest part of tree)
DFS: worst: skewed tree

space changes: height of tree is worst case: O(h)
queue size of level, width of complete tree: O(w)

[1,3,2,4]

right_height = 2
right_list = [1,3]

left_height = 3
left_list = [1,2,4]

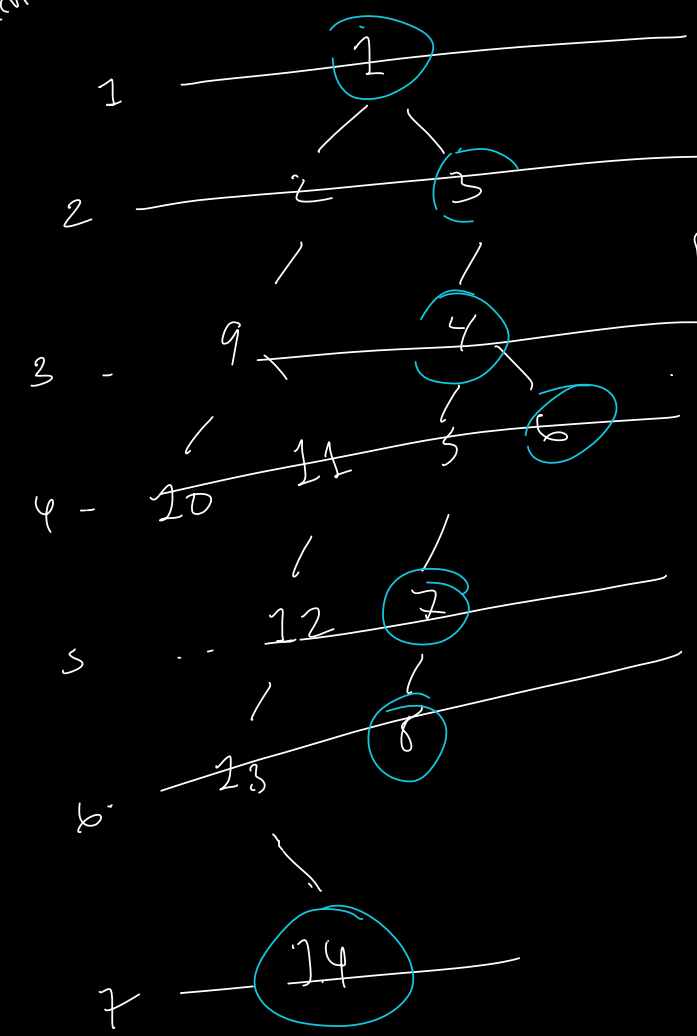


node
traverse (right) ★ ← prioritize searching rightmost side
traverse (left)

Basically
@ each level
we want (1) element

so

we only want
to add an element
if there isn't an
element for the
corresponding height
⇒ add the element



height
=
len(list)

[1, 3, 4, 6, 5]

length = 5
height = 4

checking if element empty

if list[height-1] != None:
index in list

remove last element

[1, 3, 4, 6, 7, 8, 2]

height = 2

if list[height-1] != None

remove last element

[1, 0]

height += 1

if height - 1 == len(list):

add element

list so far since we have not added anything to list yet

height == len(list) - 1