hello

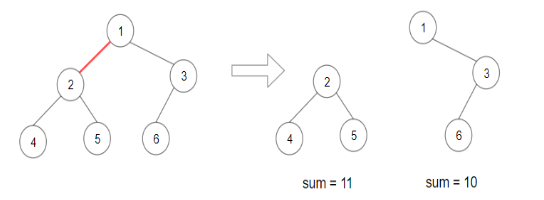
Ok

let’s talk by hangouts

ok

i am calling you

tree root



separate the tree into 2 subtree

make the production max of subtree sum

input: root , define tree-API, tree node value is repeated? Y order? N, value range? positive number of node? <1000

output: int

corner case: None? N Only one node? Y

1. Idea: brute force - dfs - scan #define the function of the method

Method:

* 1. using dfs to scan all node, and calculate and save all subtree sum in a list [subtreeSum1, subtreeSum2]
     1. in-order dfs 4 - 2 -5 -1-6 -3 scan, calculate the sum of every cut.
        1. 4 - 2 - 5 - 1 - 6 - 3
        2. 4 -2 - 5, cut, save sum of them in list
        3. my question: if you cut the edge between 2 - 5. how could you get the two sums for this inorder dfs traverse
     2. find the max production combination

1. brute force - BFS

method:

* 1. scan all node and get sum of all node by BFS - tO(N) s best O(1) averageO(N) time Complexity O(N)
     1. 1 -6 = 1+2+3+4+5+6 = 21
  2. scan all node again, save all leaf or subtree sum in a list t(N) space O(N)
     1. [4, 5, 6, 2 -4 -5, 3 -6] = [4, 5,6, 11, 9]
  3. use the sum of the tree and the list to find best combination tO(N) spaceO(N)
     1. scan list and get the pair value
        1. 4 - 17 = 4 \*17 = ..
        2. 5 - 16 = .

dfs to find the sum of tree, calc the

time tO(N) spaceO(N)

class mySolution:

def maxProdcution(self, root): # input: tree root, output: int

if not root.left and not root.right: # corner case

return root.val

sumTree = self.sumTreeRoot(root) # get the sum of nodes of tree

pair = []

pair = self.findPair(root, pair) # get all leaf and subtree sum

res = float(‘-inf’)

for elem in pair: # find all combinations

production = (sumTree - elem) \* elem

res.append(production ) # add all production into res

res = max(res, ..)

return max(res)

from collecitons import deque

def sumTreeRoot(self, root): # input: root, output: int

if not root.left and not root.right: # corner case

return root.val

res = 0

queue = deque()

queue.append(root)

while queue: # BFS method to get sum

node = queue.popleft()

res += node.val

if node.left:

queue.append(node.left)

if node.right:

queue.append(node.right)

return res

from collections import deque

def findPair(self, root, pair): # input: root, output: list

queue = deque()

queue.append(root)

while queue: # BFS scan

node = queue.popleft()

if not node.left and not node.right: # node is leaf

pair.append(node.val)

if node.left and node.right: # node is subtree root

pair.append(self.sumTreeRoot(node))

if node.left:

queue.append(node.left)

if node.right:

queue.append(node.right)

return pair