Hunter memonon

CIS 313 Written Assignment 3

1. merging Lists using heaps : TB 6.5-11

- give an O(n Log K) time Algorithm to merge K Sorted List into one sorted List where n is the total # of elements in all input List

```
List_Combine(Lists)
        Result_index = 0
        heap_sorter = minheap(K)
        result = [] = []
                                O(K)        O(Log K)
        for sort in K
Adds top note of  ->    heap_sorter.min_heap_insert(Lists[0][K], K)
K[i] to heap
O(n)  ->  while heap_sorter.empty() != True:                O(Log K)
                to_add = heap_sorter.delete_min()
                result[Resut_index] = to_add[
                result_index ++
If index != List Length, ->     if to_add.index() != to_add.Length()
Replace the value in the                heap_sorter.min_heap.insert(to_add.next(), K)
heap
```

O(Log K)

$$O(K * \text{Log}(K) + n * 2\text{Log } K)$$

$$=$$

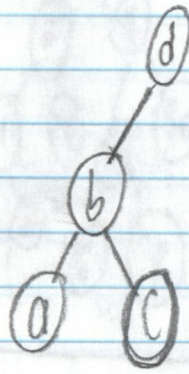
$$O(K * \log K + \text{Log } K^{2n})$$ Just a constant

$$O(n\text{Log}(K) - O(2n\text{Log}(K^2)) \quad O(2n\text{Log } K \circ K^{2n})$$

2. draw the Binary tree w/ inorder traversal: a b c d e f g h
3 postorder traversal: a c b e g f h d

Root

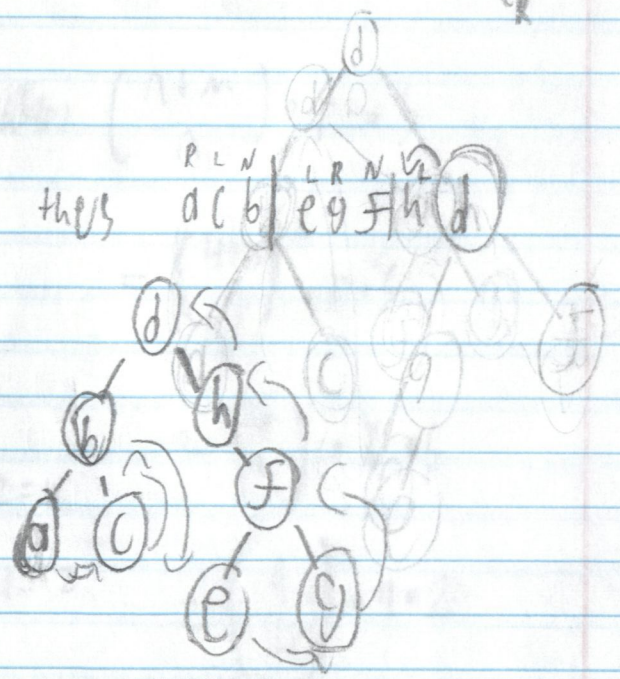
from in order + Root

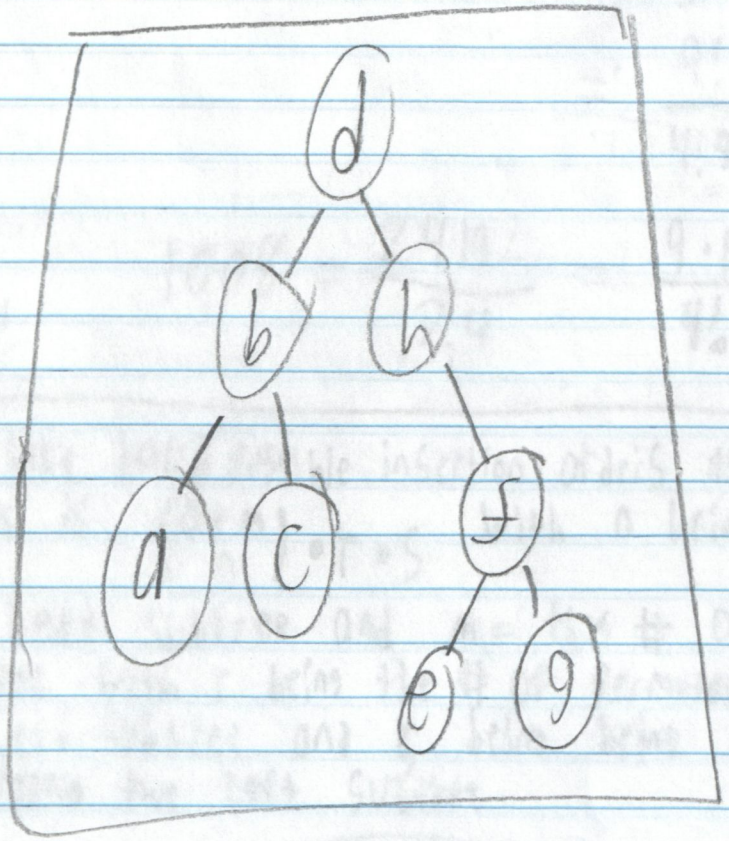


?

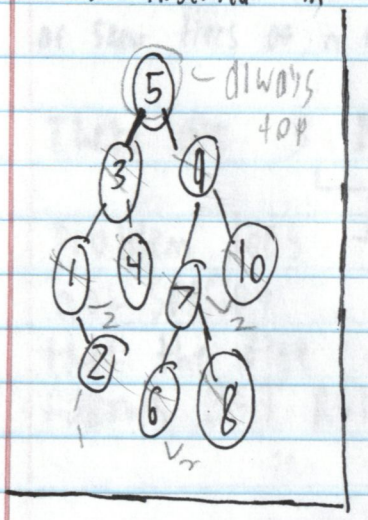
R L N L R N

thus a c b e g f h d





3. Consider the tree, How many permutations of the values 1 through 10, when inserted in that order will yield this particular tree? explain



$n=4$
$m=5$

$$\binom{n+m}{n} \cdot r \cdot s$$

$$= \binom{4+5}{4} \cdot r \cdot s$$

$$= \binom{9}{4} \cdot r \cdot s$$

$s = 2 \cdot 2 = 4$

$r = 2 \cdot 1 = 2$

$$= \binom{9}{4} \cdot 4 \cdot 2$$

$$= \frac{9!}{4! \cdot 9-4!} \cdot 4 \cdot 2$$

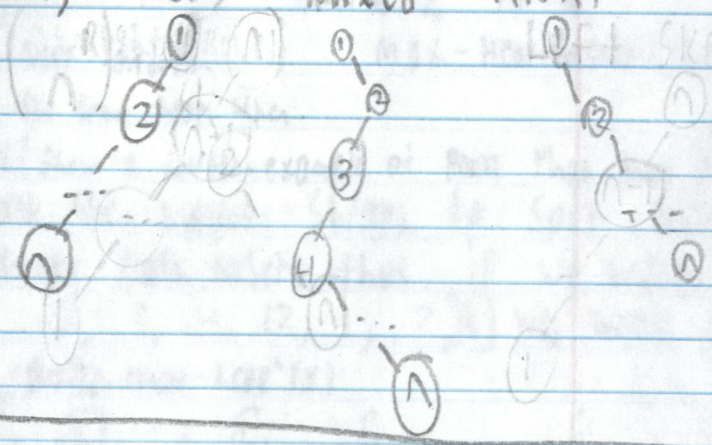
$$1008 = \frac{24192}{24} = \frac{9 \cdot 8 \cdot 7 \cdot \cancel{6} \cdot \cancel{5} \cdot \cancel{4}}{4! \cdot \cancel{5!}}$$

We have 1,008 possible insertion orders as the # of insertion orders is $\binom{n+m}{n} \cdot r \cdot s$ with n being the # of nodes in the Left subtree and m = the # of nodes in the right subtree with r being the # of permutations needed to generate the Left subtree and s being being the # of permutations generating the Left subtree

4. How many permutations of 1, 2, ... n yield a skew tree (tree of height n-1)? * Since a skew tree is gen by a single permutation, this Q is asking for the # of skew trees of n nodes) explain your formula

There are 3 possibilities; Left Mixed Right

Problem does not specify that the tree follows BST Rules



We know there's only 1 # that can be the root and for each value, it can be 1 of two options, either its a Left child or a Right child of the prior node thus;

$$\underbrace{1}_{\text{\# of permutations 4 Root}} \times \underbrace{2}_{\text{\# of side choices for 2}} \times \underbrace{2}_{\text{\# of side choices for 3}} \ldots \underbrace{2}_{n} = 1 \cdot 2^{n-1} = 2^{n-1}$$

possible Binary Skew trees

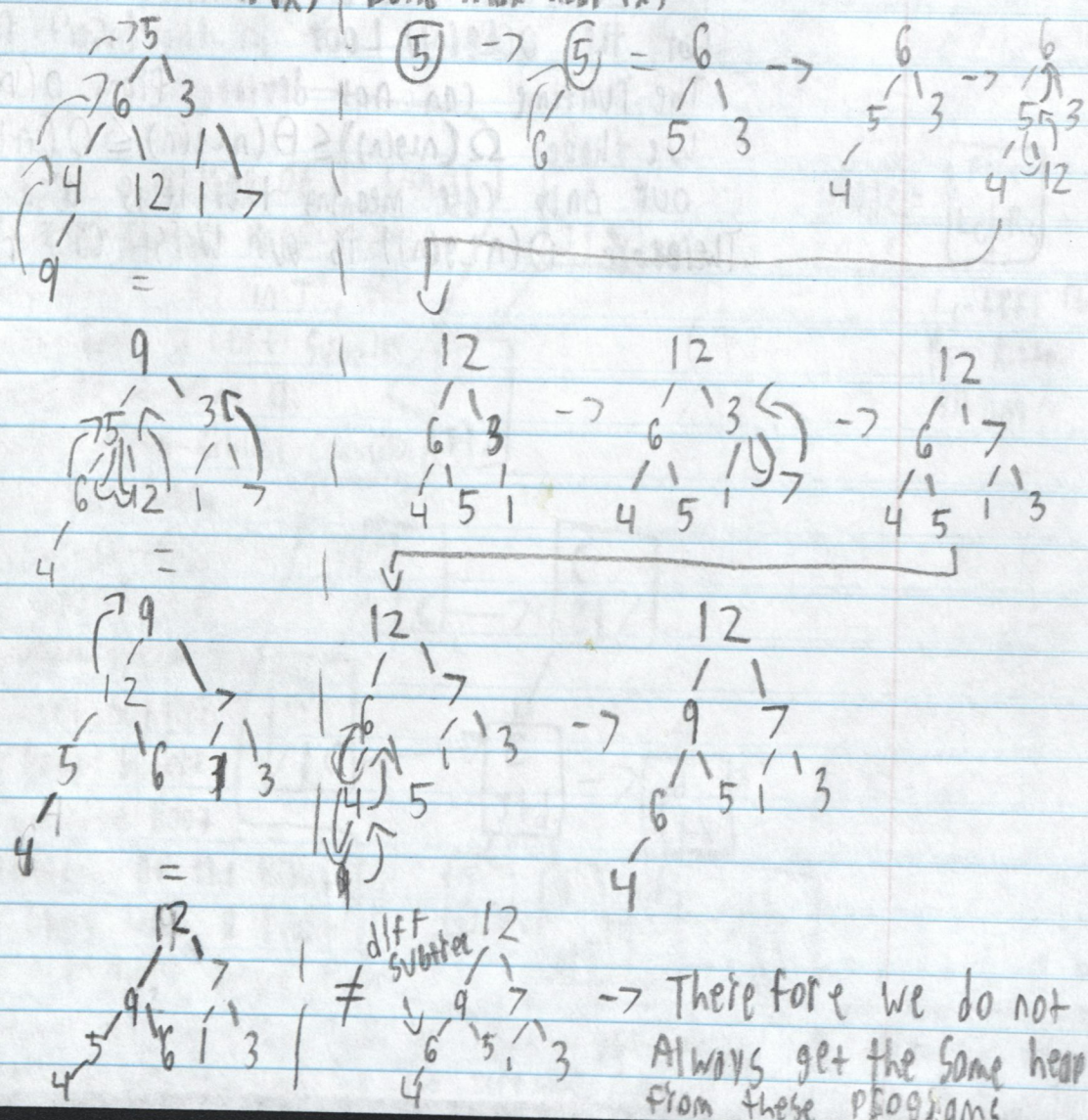
5. TB 6-1; one way to build a heap is by repeatedly calling Max-Heap-Insert to insert the elements into the heap. Consider;

```
build-max-heap'(A, n)
    A.heap-size = 1
    for i = 2 to n
        Max-Heap-Insert(A, A[i], n)
```

a. do the procedures build-max-heap and build-max-heap' always create the same heap when ran on the same input array? Show a counter-example or prove that they do

Build-max-heap inserts all then makes swaps to sort while build-max-heap' sorts after each insert thus if we were to run both on $x = [5, 6, 3, 4, 12, 1, 7, 9]$ we would have;

Build-max-heap(x) | Build-max-heap'(x)



→ Therefore we do not always get the same heap from these programs

b. Show that Build-max-heap' has a worst case $\Theta(n \lg n)$ time to build an n element heap

Build-max-heap'

A.heap-size = 1

LOOPS → for i = 2 to n

for $O(n)$ max-heap-insert(A, A[i], n)

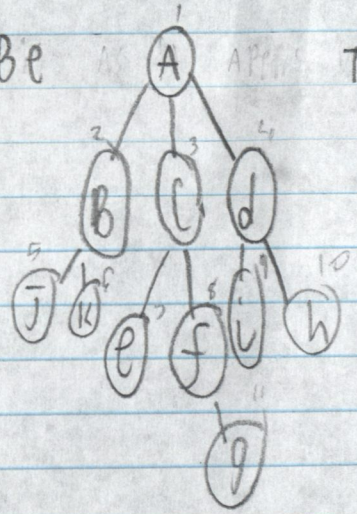
from TB, Run time of insert is $O \lg(n)$

Therefore we get $O(n) \cdot O(\lg(n) = O(n \lg(n))$. In addition to this, we also know that there are no conditionals that can end the Loop early before it Runs n times (and the same goes for the $O(\lg(n))$ Loop in the insert function). Therefore The runtime can not deviate from $O(n \lg n)$ which means we have $\Omega(n \lg(n)) \leq \Theta(n \lg(n) \leq O(n \lg(n))$ for our only case meaning that it is also our worst-case

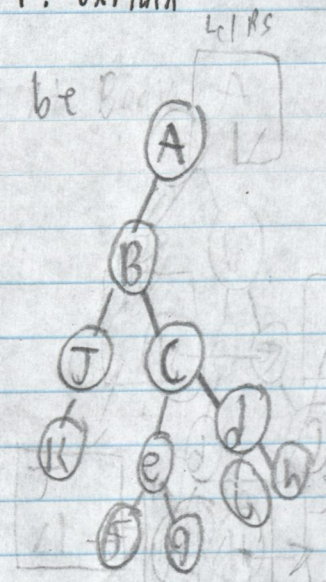
Therefore $\Theta(n \lg(n))$ is our worst case run-time

6. Consider ordered tree T & binary tree T' representing it. Using the first-child next-sibling Representation (sect 10.5), an in order traversal of T' is ≡ to what kind of traversal of T? explain

Let T Be



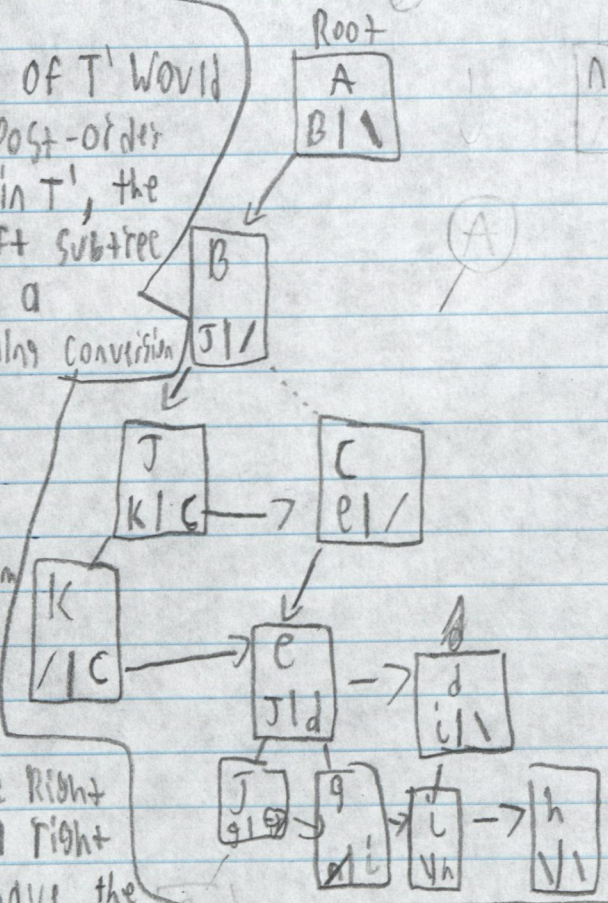
T' would be



≡ null

node = [node | L_c | R_s]

L_c = Left Child
R_s = Right Sibling



An in order traversal of T' would be the same as a post-order traversal of T as in T', the Root only has a Left subtree by nature of it being a first-child next-sibling conversion to a Binary tree. Thus when we do the in-order traversal of T', we traverse from the bottom of the Left Subtree to the Root before going from the Root to the bottom of the Right Subtree thus w/o a right subtree, we only have the Left subtree meaning the root is the Last value we traverse thus Its a post order traversal as we list Left sub, Root.