CSC 3110
May Wandyez
Gq5426
2/27/2024
Homework 3

CSC 3110

Algorithm Design and Analysis

(30 points)

Due: 02/28/2024 11:59 PM

Note: Submit answers in PDF document format. Please read the submission format for appropriate file naming conventions.

1) Exercises 5.1:
   a. Problem 1 (parts a - d) (2 points)

1. a. Write pseudocode for a divide-and-conquer algorithm for finding the position of the largest element in an array of n numbers.

b. What will be your algorithm's output for arrays with several elements of the largest value?

c. Set up and solve a recurrence relation for the number of key comparisons made by your algorithm.

d. How does this algorithm compare with the brute-force algorithm for this problem?

a.) Too bad we can't assume n is sorted. Let's use merge sort
Pseudocode:
Recursively divide array into two halves
Find largest element in each half
Merge two halves together starting from bottom of recursion

b.) It will still work, the one that will be returned will be the one from the leftmost index.

c.)Starting to realize my pseudo code sucked. T(n) = 2T(n/2) +n seems to be the standard number of key comparisons for merge sort. Merge sort has an efficiency of nlog(n)

d.) This algorithm is less efficient than the brute force mechanism, brute force only requires that O(n) take place where every element is compared, this requires dividing everything which takes MORE time because inherently for each division an additional basic operation is required. Merge sort's efficiency is O(n)=nlog(n)

CSC 3110
May Wandyez
Gq5426
2/27/2024
Homework 3

b. Problem 5 (parts a - c) (2 points)

5. Find the order of growth for solutions of the following recurrences.

a. $T(n) = 4T(n/2) + n,\ T(1) = 1$

b. $T(n) = 4T(n/2) + n^2,\ T(1) = 1$

c. $T(n) = 4T(n/2) + n^3,\ T(1) = 1$

Oh hey look it's master theorem in action.

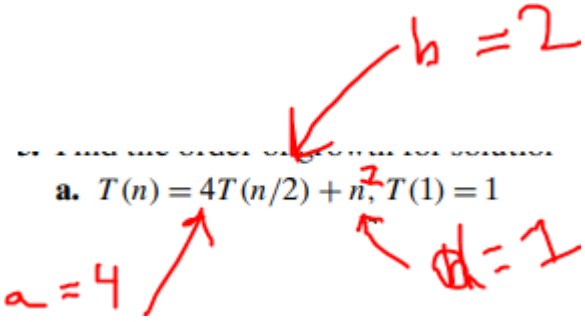
Recurrence: Master Theorem

$T(n) = aT(n/b) + f(n)$ where $f(n)=cn^k$

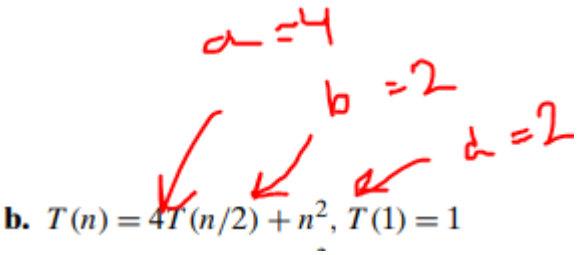
1. $a < b^k$ $\quad T(n) \sim n^k$
2. $a = b^k$ $\quad T(n) \sim n^k \log_b n$
3. $a > b^k$ $\quad T(n) \sim n^{\log_b a}$

Here is what master theorem looks (though this one I googled uses k instead of d)like so I can stop looking it up, now we just need to identify a,d,and b

a.)



4 > 2^1 ------> T(n) = N^(log2(4))-> O(n)=n^2



b.)

2^2 = 4 ------> T(n) = n^2log(n)

c.)

CSC 3110
May Wandyez
Gq5426
2/27/2024
Homework 3

c. $T(n) = 4T(n/2) + n^3, T(1) = 1$

4 < 2^3 -----> T(n) = n^3

2) Exercises 5.2:

a. Problem 8 (2 points)

8. Design an algorithm to rearrange elements of a given array of n real numbers so that all its negative elements precede all its positive elements. Your algorithm should be both time efficient and space efficient.

Too bad we haven't learned bucket sort yet, it doesn't say that the array has to be sorted, just that negative has to be on one side, and positive on the other.
1.) Define a variable for the left index, L with the initial value equaling to zero, define a variable for the right index, R, the with the initial value equally to one.
2.) Check if the value at L is negative, if it is, leave it, increase L by one.
3.) Check if the value at L is positive, if it is, begin increasing the value of R by one until a negative number is found. When a negative number is found, swap the positions of the positive number at L and the negative number at R. Increase L by one.
4.)Repeat steps 2-3 until R = (n-1)

This algorithm is pretty simple, the basic idea is put all of the negative numbers on the left side, so we simply start at the left side and reach for the first negative number we find and put it there. Theoretically this algorithm has an efficiency of O(n) because the right index never scans more than the entirety of the array, and the number of swaps never exceeds the number of negative numbers in the array.

b. Problem 10 (2 points)

10. Implement quicksort in the language of your choice. Run your program on a sample of inputs to verify the theoretical assertions about the algorithm's efficiency.

Python has quicksort built in as its default sorting algorithm. I will thereby demonstrate how to use the python implementation of quicksort in python: List.sort()

Wow. What a sort.

CSC 3110
May Wandyez
Gq5426
2/27/2024
Homework 3

3) Exercises 5.3:

a. Problem 2 (2 points)

algorithm?

2. The following algorithm seeks to compute the number of leaves in a binary tree.

ALGORITHM *LeafCounter(T)*

//Computes recursively the number of leaves in a binary tree
//Input: A binary tree T
//Output: The number of leaves in T
if $T = \varnothing$ **return** 0
else return $LeafCounter(T_{left}) + LeafCounter(T_{right})$

Is this algorithm correct? If it is, prove it; if it is not, make an appropriate correction.

This is a recursive algorithm that starts at the root of a binary tree, and call itself for the left and right branches of that binary tree. However, when the recursion reaches an empty leaf, it returns 0, which makes sense, as empty leaves are not leaves and are the end of the line, however, there is no recursive call for if the leaf is the last leaf and has a value, so as a result the recursion returns 0 for everything.

Here is the fix:
If T=NULL return 0 //not a leaf, not anything
elseif TL = NULL and TR = NULL return 1 //case where the leaf has a value, but //its children don't
else return LeafCounter(TL)+(LeafCounter(TR) //recursion call

b. Problem 5 (parts a - c) (2 points)

CSC 3110
May Wandyez
Gq5426
2/27/2024
Homework 3

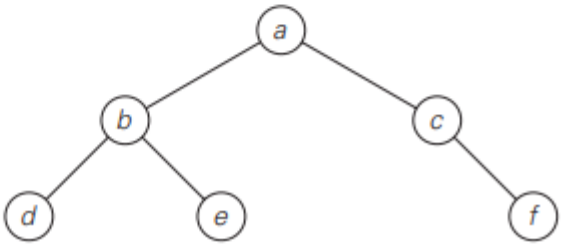
5. Traverse the following binary tree

a. in preorder.

b. in inorder.

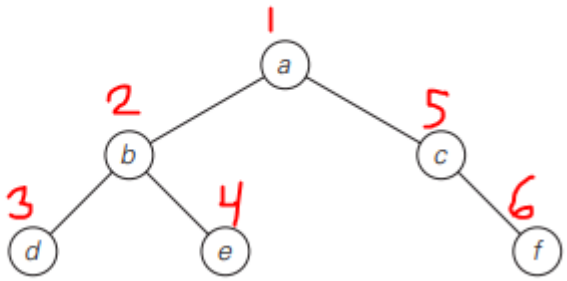
c. in postorder.

Divide-and-Conquer



PreOrder: Root, Left, Right

Divide-and-Conquer

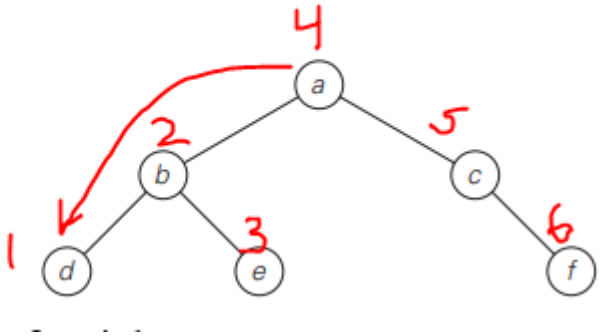


a,b,d,e,c,f

InOrder: Left,Root,Right

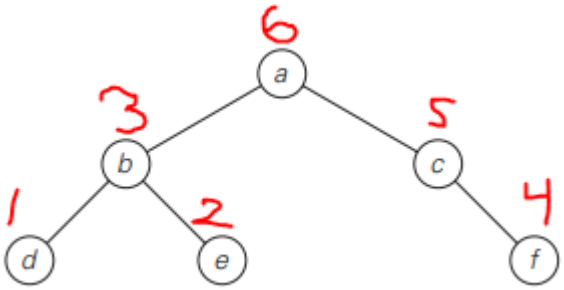
CSC 3110
May Wandyez
Gq5426
2/27/2024
Homework 3

Divide-and-Conquer



d,b,e,a,c,f

PostOrder:Left,Right,Root



D,e,b,f,c,a

4) Exercises 5.4:

a. Problem 3 (part a & b) (2 points)

3. a. Prove the equality $a^{\log_b c} = c^{\log_b a}$, which was used in Section 5.4.

b. Why is $n^{\log_2 3}$ better than $3^{\log_2 n}$ as a closed-form formula for $M(n)$?

b. Problem 5 (2 points)

5) Exercises 6.1

a. Problem 1 (part a & b) (2 points)

b. Problem 2 (part a & b) (2 points)

5. How many one-digit additions are made by the pen-and-pencil algorithm in multiplying two n-digit integers? You may disregard potential carries.

6) Exercises 6.3

a. Problem 4 (parts a- c) (2 points)

4. For each of the following lists, construct an AVL tree by inserting their elements successively, starting with the empty tree.

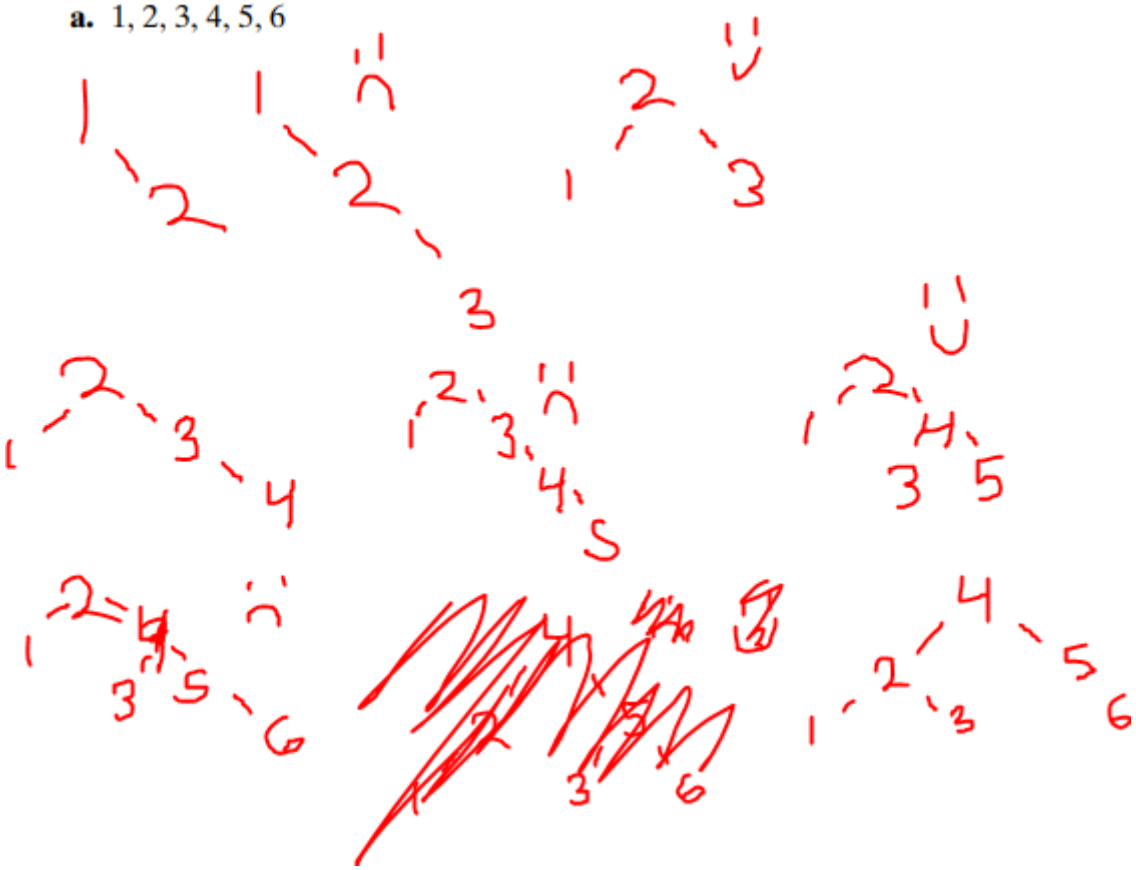
a. 1, 2, 3, 4, 5, 6

b. 6, 5, 4, 3, 2, 1

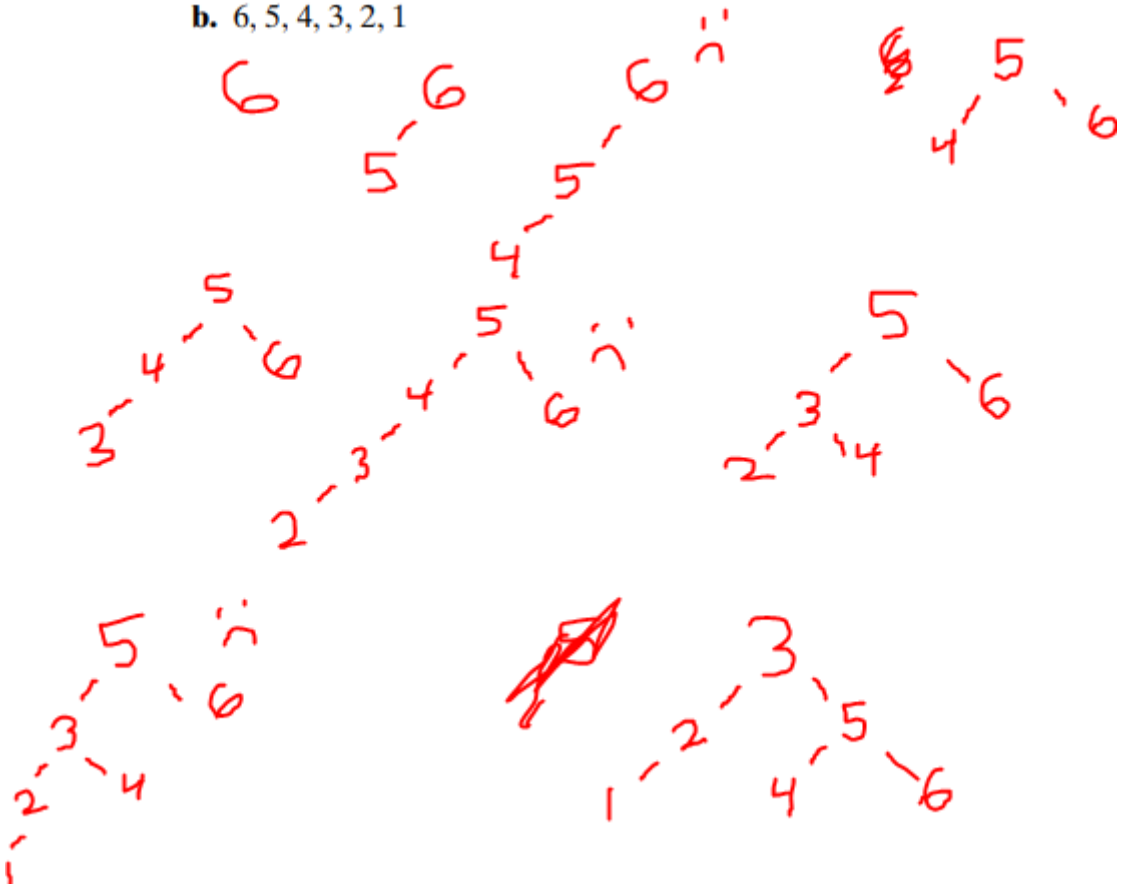
c. 3, 6, 5, 1, 2, 4

CSC 3110
May Wandyez
Gq5426
2/27/2024
Homework 3

a. 1, 2, 3, 4, 5, 6

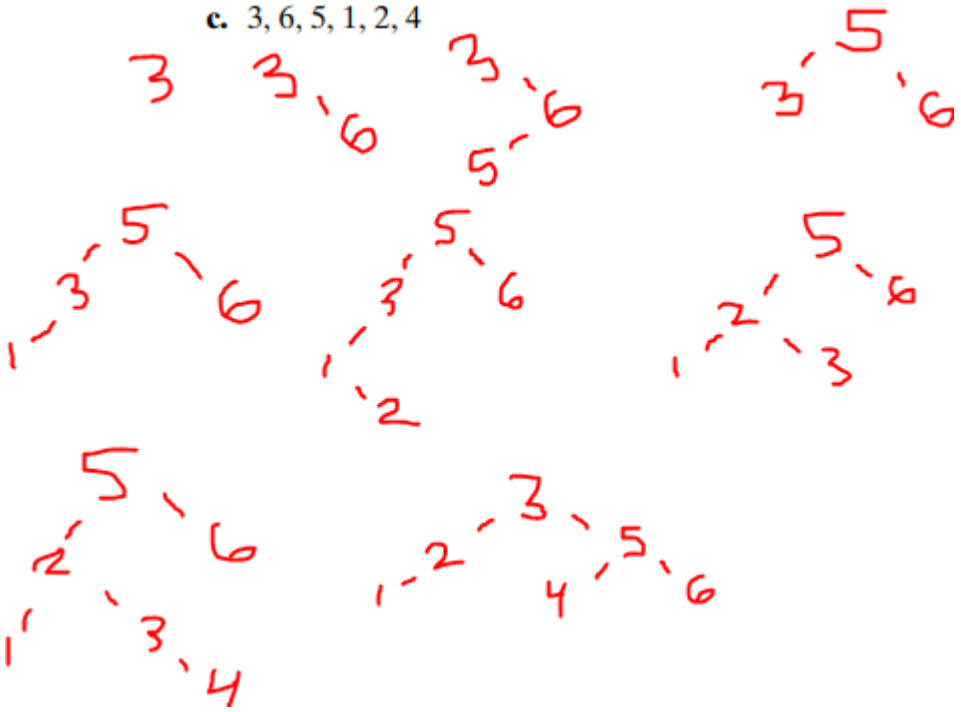


b. 6, 5, 4, 3, 2, 1



CSC 3110
May Wandyez
Gq5426
2/27/2024
Homework 3

c. 3, 6, 5, 1, 2, 4



b.

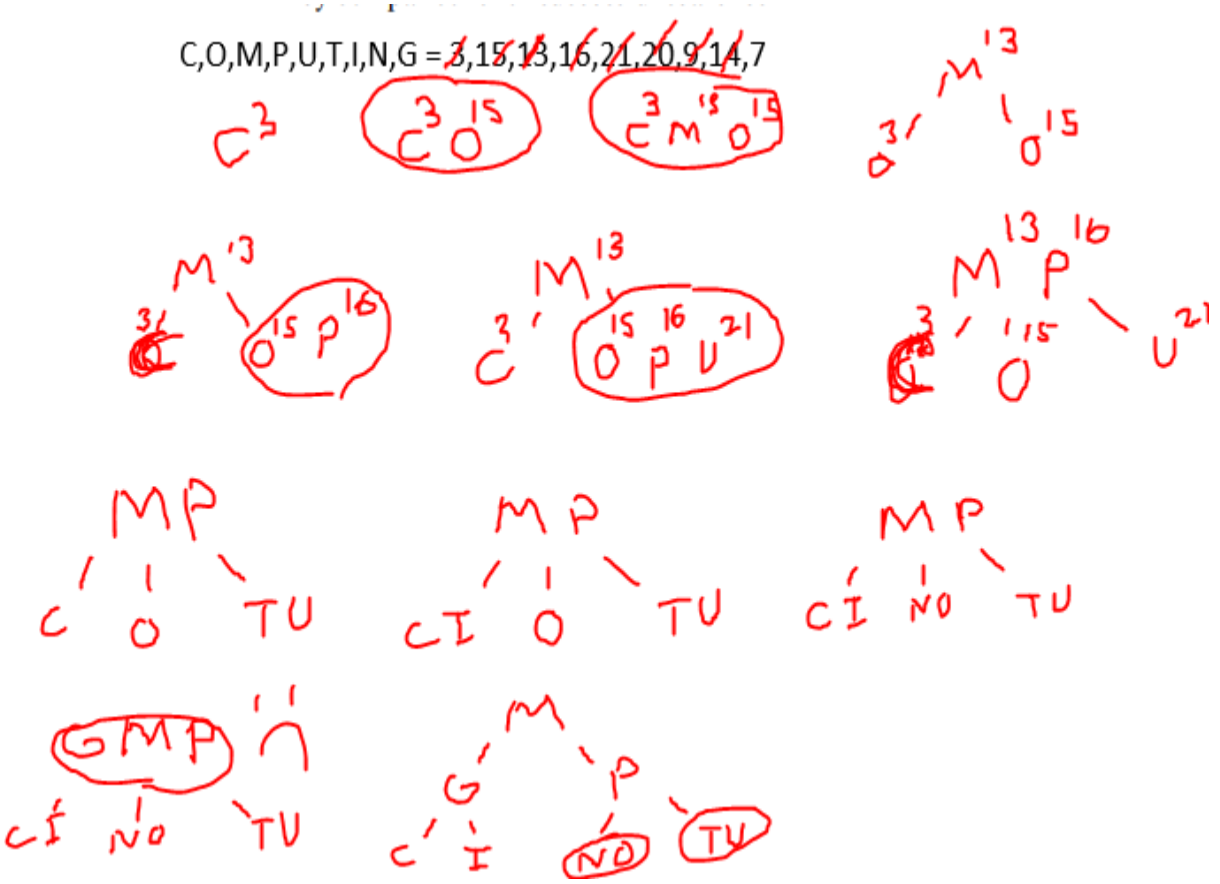
c. Problem 7 (part a & b) (2 points)

7. a. Construct a 2-3 tree for the list C, O, M, P, U, T, I, N, G. Use the alphabetical order of the letters and insert them successively starting with the empty tree.

b. Assuming that the probabilities of searching for each of the keys (i.e., the letters) are the same, find the largest number and the average number of key comparisons for successful searches in this tree.

CSC 3110
May Wandyez
Gq5426
2/27/2024
Homework 3

C,O,M,P,U,T,I,N,G = 3,15,13,16,21,20,9,14,7



b.) The largest number of comparison will require 4 comparison, this is through the path MPNO or MPTU. The average case efficiency of a 2-3 tree is log(n) for a search, so it is likely that the case efficiency would be log3(9) or around ~2. More specifically it would be the average of (1,2,2,3,3,3,3,4,4) ~2.77778

7) Exercises 6.4

a. Problem 1 (parts a- c) (2 points)

1. a. Construct a heap for the list 1, 8, 6, 5, 3, 7, 4 by the bottom-up algorithm.

b. Construct a heap for the list 1, 8, 6, 5, 3, 7, 4 by successive key insertions (top-down algorithm).

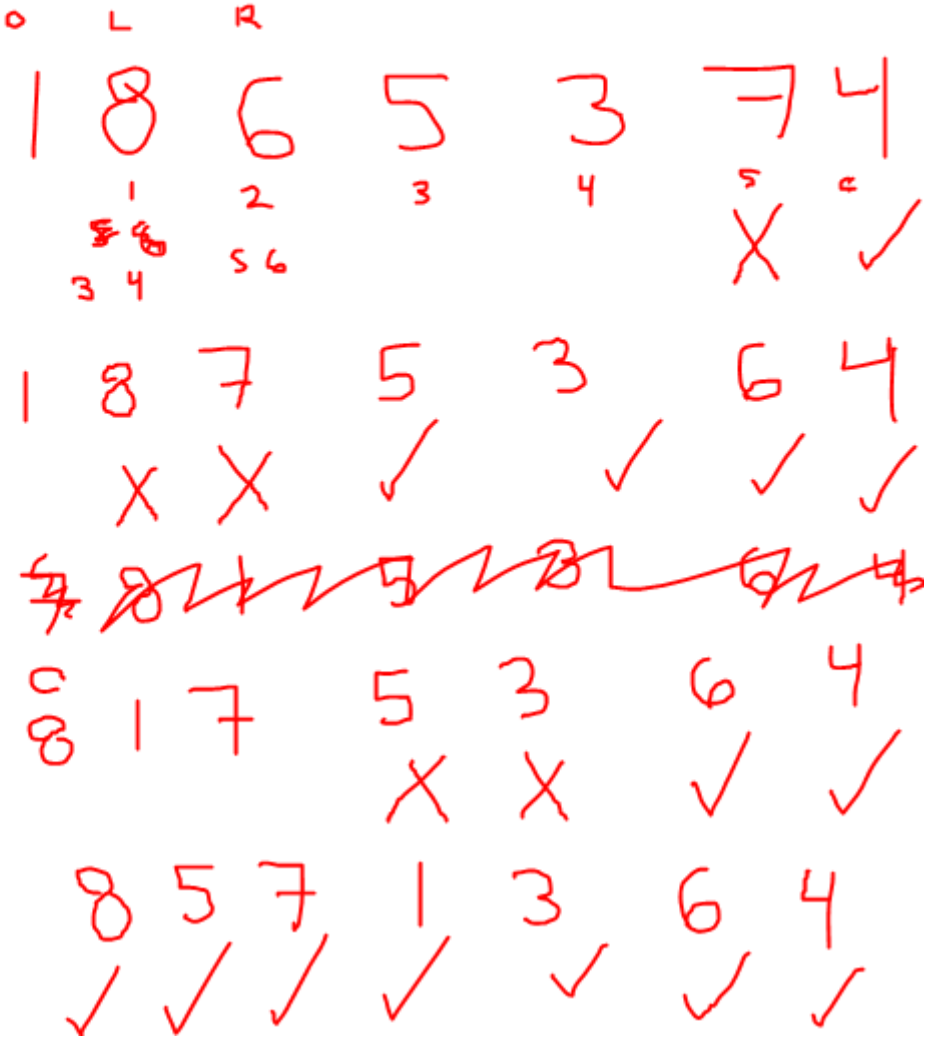
c. Is it always true that the bottom-up and top-down algorithms yield the same heap for the same input?

a.)

CSC 3110
May Wandyez
Gq5426
2/27/2024
Homework 3

2i + 1 = left
2i + 2 = right

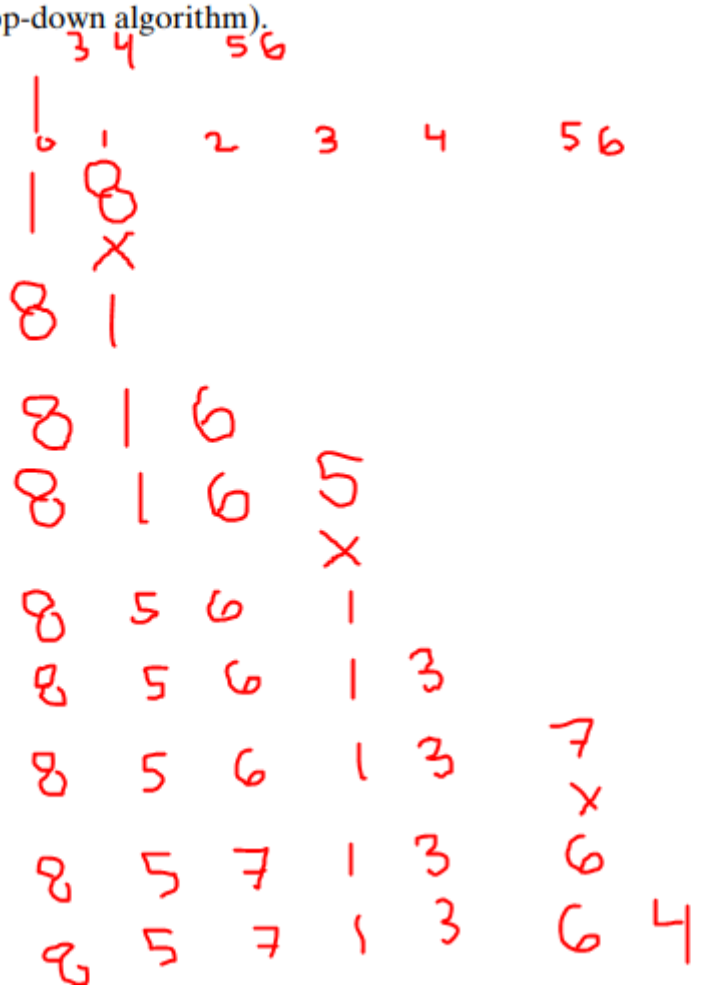
. **a.** Construct a heap for the list 1, 8, 6, 5, 3, 7, 4 by the bottom-up algorithm.



b.)

CSC 3110
May Wandyez
Gq5426
2/27/2024
Homework 3

b. Construct a heap for the list 1, 8, 6, 5, 3, 7, 4 by successive key insertions (top-down algorithm).



c.) No, bottom up and top down do not yield the same results always. There are multiple valid ways to build a heap, so the insertion order matters in how the heap is constructed.

b. Problem 5 (part a & b) (2 points)

5. a. Design an efficient algorithm for finding and deleting an element of the smallest value in a heap and determine its time efficiency.

b. Design an efficient algorithm for finding and deleting an element of a given value v in a heap H and determine its time efficiency.

a.) The smallest elements of a heap are at its leaves. All leaves in a heap are found in the second half of the heap (n/2). Therefore we just have to iterate through the second half of the heap until the end of the heap and take the lowest value. This value could then be deleted. The efficiency would be O(n/2), as deleting an element at the end of a heap does not require adjusting the elements above it.

b.) Unfortunately, an element of a given value can be anywhere in the heap because the element could be the largest or the smallest. The only thing that is certain is that the parent is smaller than its children. So one can simply scan

from left to right from 0 to n-1 to find the element that one is searching for. However, once the element is found, and is deleted, the heap must be rebuilt, this takes log(n) for binary heaps. Therefore the time taken would be o(n)+log(n).

c. Problem 7 (parts a- c) (2 points)

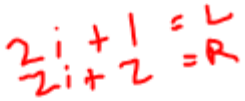
7. Sort the following lists by heapsort by using the array representation of heaps.

a. 1, 2, 3, 4, 5 (in increasing order)

b. 5, 4, 3, 2, 1 (in increasing order)

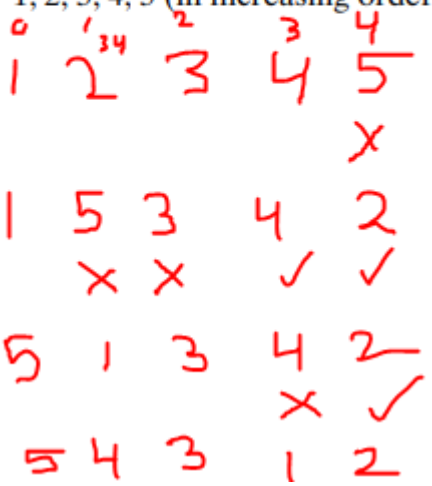
c. S, O, R, T, I, N, G (in alphabetical order)

S,O,R,T,I,N,G = 19,15,18,20,9,14,7

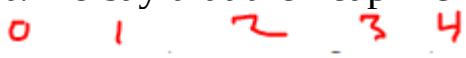


7. Sort the following lists by heapsort by

a. 1, 2, 3, 4, 5 (in increasing order)



By increasing order you mean you want the largest number on top right? To say that the heap increases from bottom to top?



b. 5, 4, 3, 2, 1 (in increasing order)

CSC 3110
May Wandyez
Gq5426
2/27/2024
Homework 3

19 15

c. S, O, R, T, I, N, G (in alphabetical order)

0 1 2 3 4 5 6

19 15 18 20 9 14 7

X ✓ ✓ ✓

19 20 18 15 9 14 7

X ✓ ✓ ✓ ✓ ✓

20 19 18 15 9 14 7

T S R O I N G