CPSC 221
Theory Assignment 2

Brian Taylor
52311859
z2b0b
Lab L0D

1. A number of different sized pancakes are stacked. A sorted pancake stack is defined as having the smallest pancake on top, the second smallest pancake under the smallest pancake, etc. The only tool provided is a spatula that will flip any top partition of the stack. There only ever exists one stack of pancakes, not multiple smaller, independent stacks. If every flip took one unit of time to complete, exactly how many flips or units of time are required in the worst-case (i.e., for the worst-case arrangement of pancakes) to sort the stack? Aim for a good algorithm. Express your answer, T(n), as a function of the number of pancakes. Then, give a Big-Θ estimate. What is the minimum number of flips needed to sort a worst-case arrangement of 4 pancakes? (Note that your algorithm may not necessarily give the true minimum number of flips.)

My algorithm starts by locating the largest "out of order" pancake (an "out of order" pancake is a pancake with smaller pancakes below it). Let's say it's the *k-th* largest pancake. We then perform two flips to get it into its proper place. The partial stack with the *k-th* largest pancake on the bottom is flipped first, putting the *k-th* largest pancake on top. Then the top *k* pancakes are flipped. In this way the *k-th* largest pancake is in its correct position and all pancakes below it are also in their correct position.

The following is an example of the worst-case arrangement of 4 pancakes, with 1 representing the smallest pancake and 4 representing the largest pancake. With 4 pancakes the worst-case arrangement requires 5 flips. The largest "out of order" pancake is the 4. So, we flip the top two pancakes and then flip all four pancakes. The 4 is now in its correct position. Then the largest "out of order" pancake is the 3. So, we flip the top two pancakes and then flip the top three pancakes. At this point only a single flip is required to get the two smallest pancakes in order.

2	4	1	3	2	1
4	2	3	1	1	2
3	3	2	2	3	3
1	1	4	4	4	4

With 4 pancakes the worst-case arrangement requires 5 flips.

In the worst case each of the n-2 largest pancakes will be out of order when we need to deal with them and will require two flips to get into their proper place. When all but the two smallest pancakes are in order the worst case only requires a single flip to get those two pancakes in order. Thus the number of flips in the worst-case is $T(n) = 2(n-2) + 1 = 2n - 3$.

T(n) = 2n – 3 is in O(n) and in Ω(n) so therefore in Θ(n).

Show that $T(n) = 2n - 3$ is O(n):

$$let\ c = 2\text{: } 2n \geq 2n - 3\ \forall n > n_0 = 1$$

Show that $T(n) = 2n - 3$ is Ω(n):

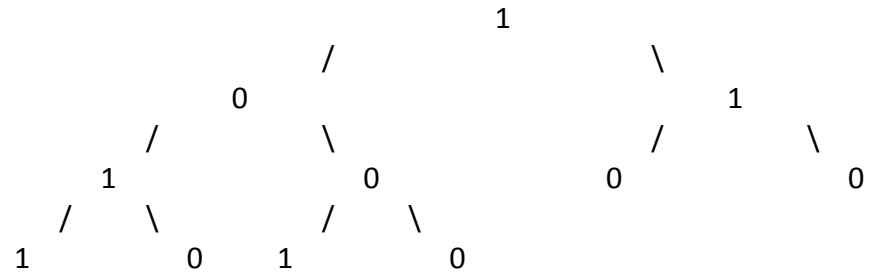
$$let\ c = 1\text{: } n \leq 2n - 3\ \forall n > n_0 = 3$$

Therefore $T(n) = 2n - 3\ is\ in\ \Theta(n)$.

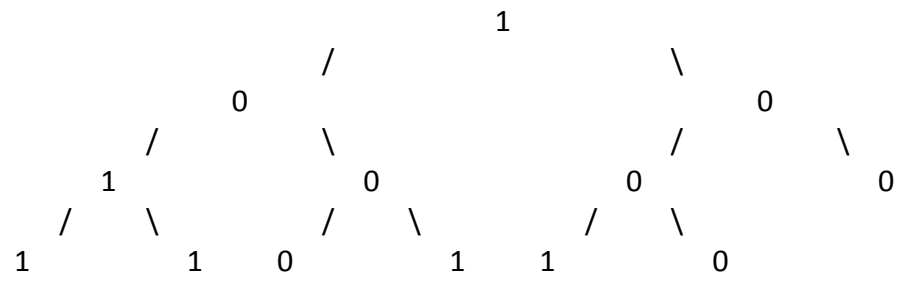
2. Let T be a full binary tree, and assume that the nodes of T are ordered by a preorder traversal. This traversal assigns the label 1 to all internal nodes, and the label 0 to each leaf (a node is internal if it is not a leaf). The sequence of 0's and 1's that results from the preorder traversal of T is called the tree's characteristic sequence.

   a) Determine the full binary trees for the characteristic sequences:

      i. 10110010100



      ii. 1011100100100

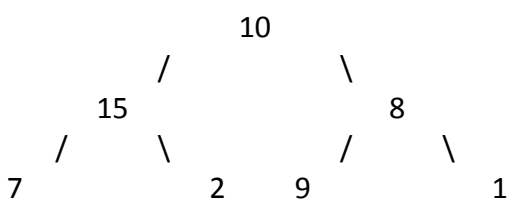


3. In class we employed a version of the Heapify algorithm in which an unordered sequence of elements was converted to a heap by the repeated insertion of the elements into a heap (beginning with the empty heap). After each insertion, the ReheapUp algorithm was applied to the new element to reheapify the tree structure.

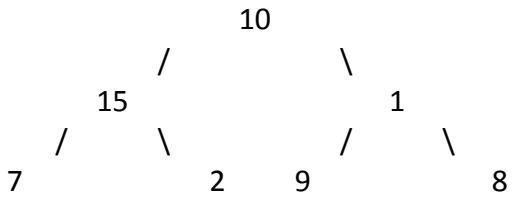
   Here's another way to Heapify a sequence of elements. First, just put all the elements into a complete binary tree, ignoring any notion of "heapness". Apply the ReheapDown algorithm to the roots of the smallest subtrees, then to the roots of the subtrees whose left and right subtrees have been heapified, and so on up the tree.

   Use this new version of Heapify to convert the following trees to heaps, and then sort them using Heapsort (either ascending or descending order for the results is fine). Clearly show each step.

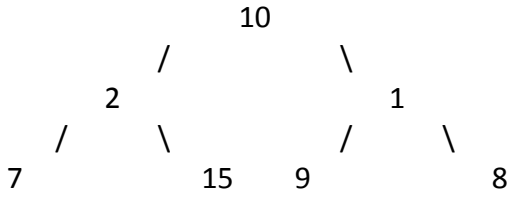
a) Create a minimum heap:



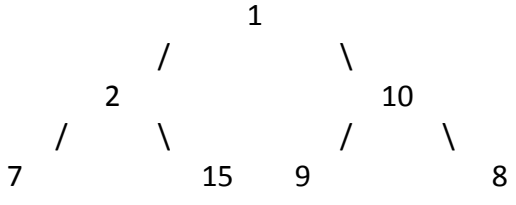
10	15	8	7	2	9	1



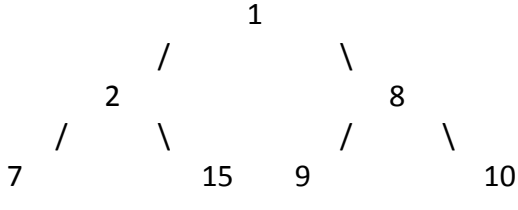
10	15	1	7	2	9	8



10	2	1	7	15	9	8

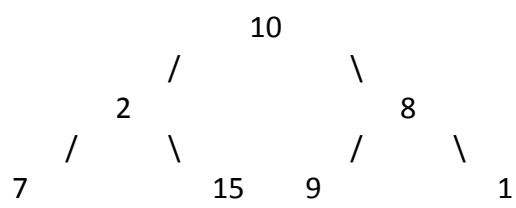


1	2	10	7	15	9	8

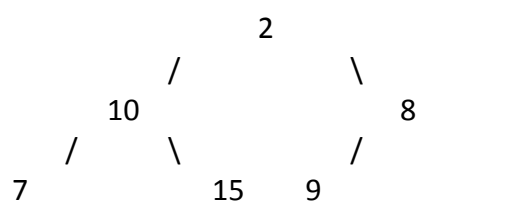


1	2	8	7	15	9	10

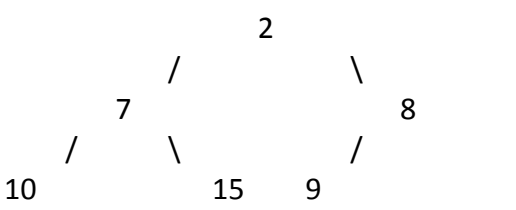
(then sort):



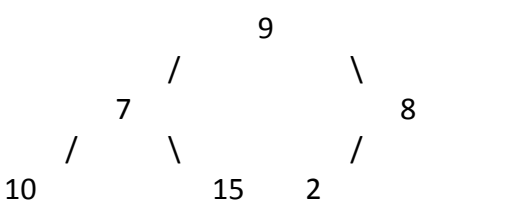
10	2	8	7	15	9	1



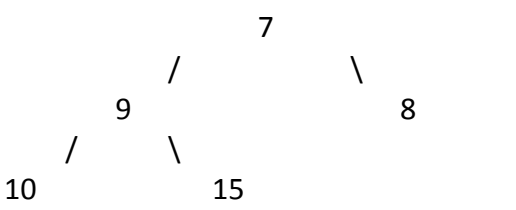
2	10	8	7	15	9	1



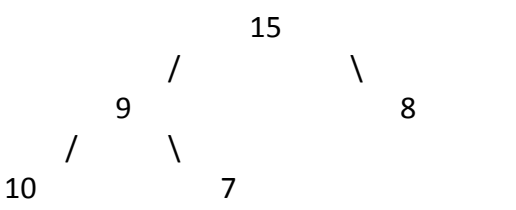
2	7	8	10	15	9	1



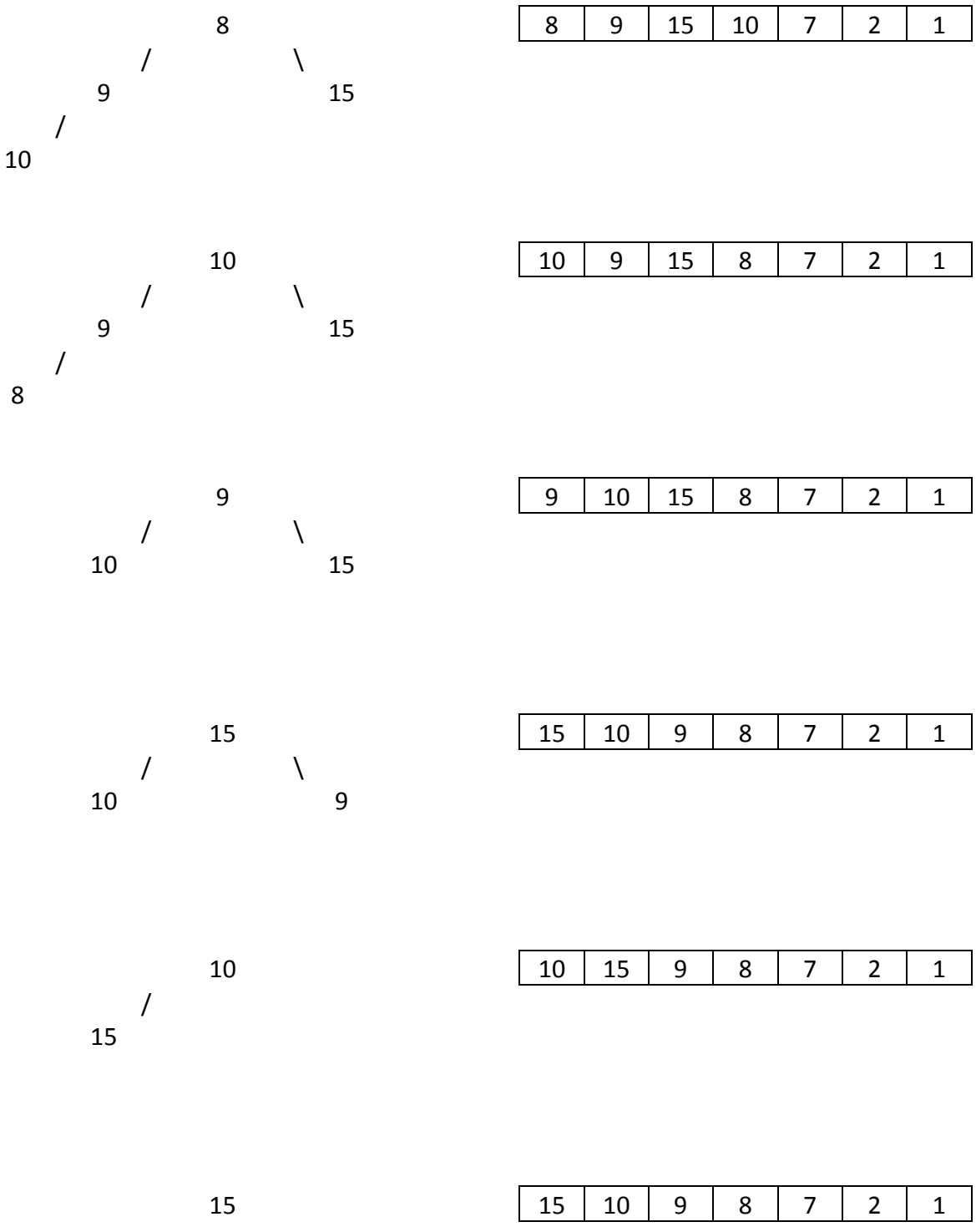
9	7	8	10	15	2	1



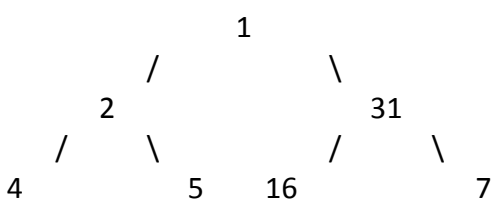
7	9	8	10	15	2	1



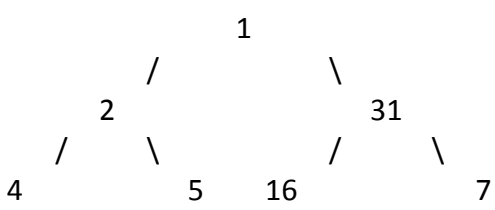
15	9	8	10	7	2	1


8
9
15
10
8 9 15 10 7 2 1
10
9
15
8
10 9 15 8 7 2 1
9
10
15
9 10 15 8 7 2 1
15
10
9
15 10 9 8 7 2 1
10
15
10 15 9 8 7 2 1
15
15 10 9 8 7 2 1

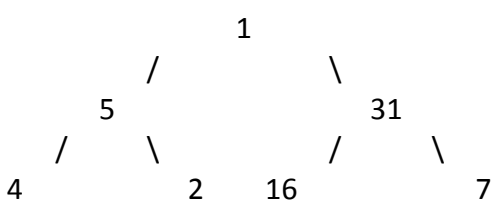
b) Create a maximum heap:



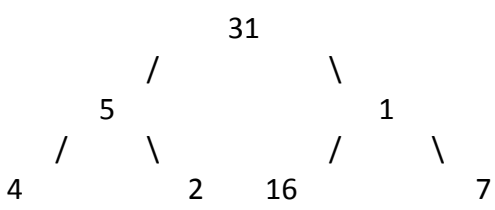
1	2	31	4	5	16	7



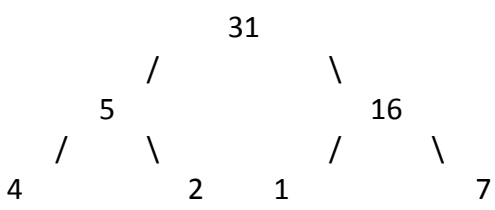
1	2	31	4	5	16	7



1	5	31	4	2	16	7

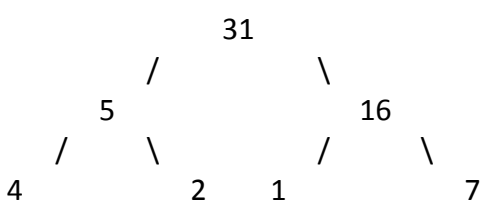


31	5	1	4	2	16	7

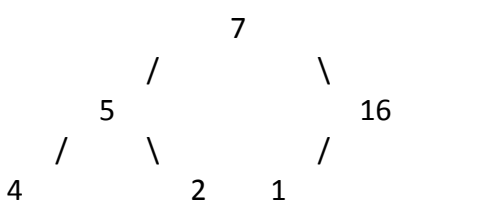


31	5	16	4	2	1	7

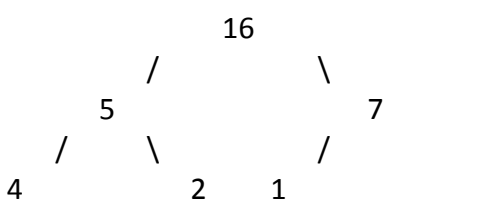
(then sort):



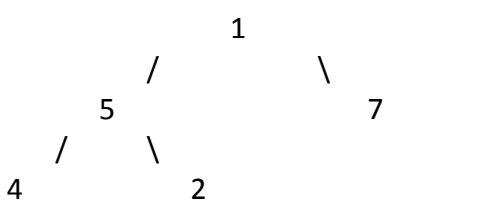
31	5	16	4	2	1	7



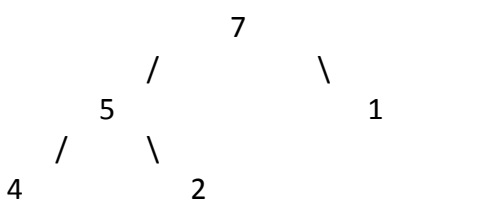
7	5	16	4	2	1	31



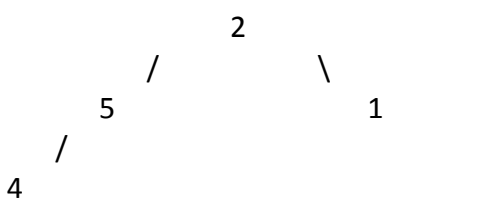
16	5	7	4	2	1	31



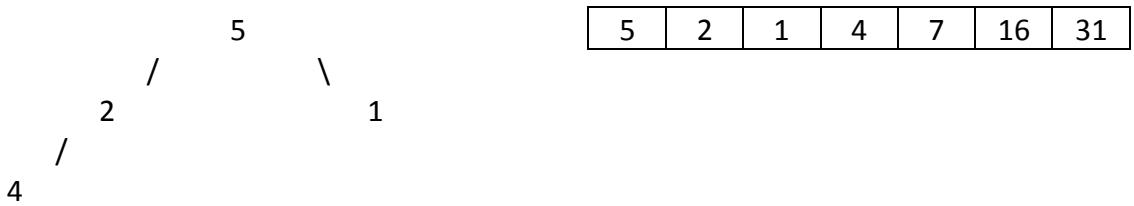
1	5	7	4	2	16	31

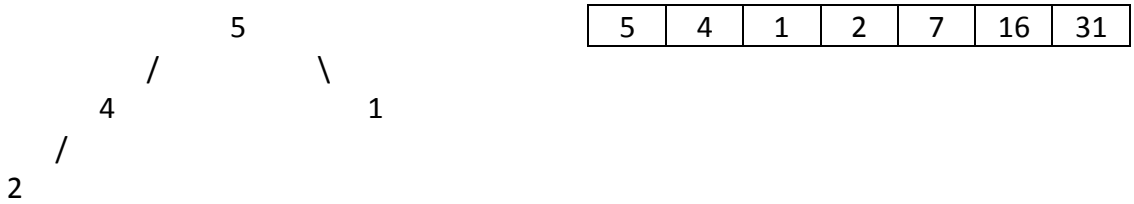


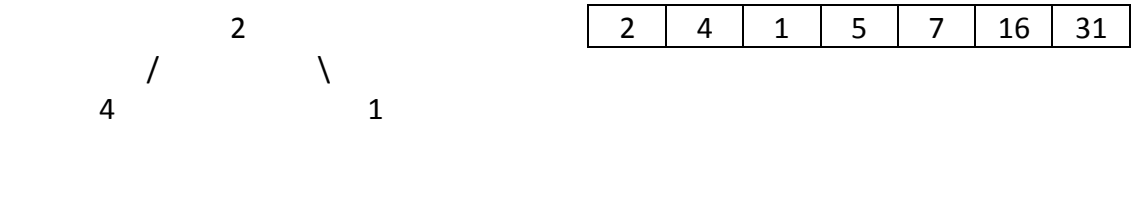
7	5	1	4	2	16	31

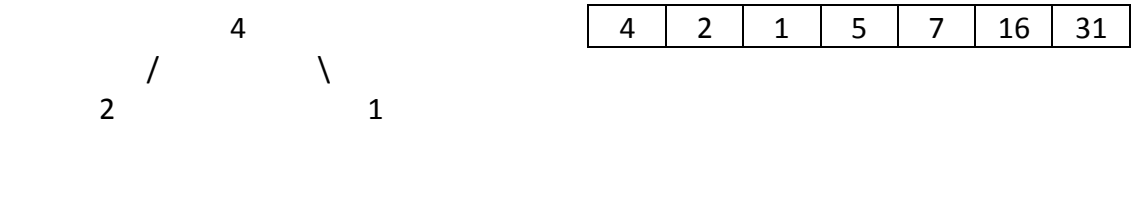


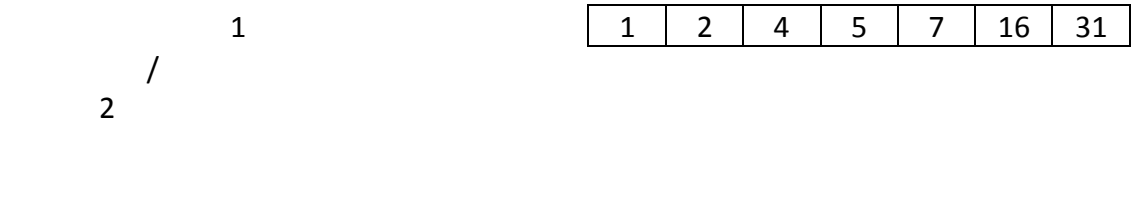
2	5	1	4	7	16	31

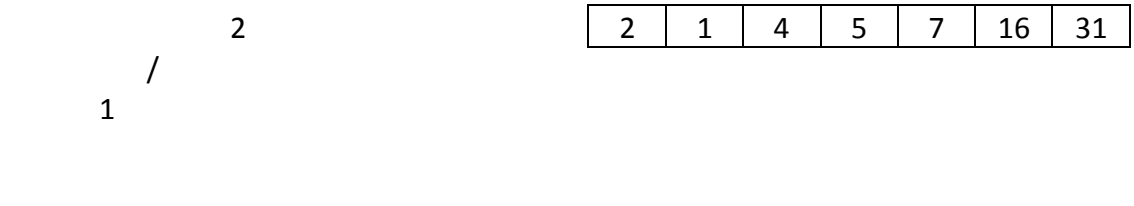

5
/ \
2 1
/
4
5 2 1 4 7 16 31

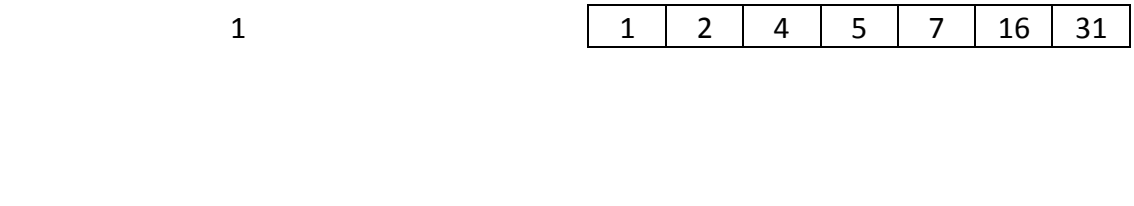

5
/ \
4 1
/
2
5 4 1 2 7 16 31


2
/ \
4 1
2 4 1 5 7 16 31


4
/ \
2 1
4 2 1 5 7 16 31


1
/
2
1 2 4 5 7 16 31


2
/
1
2 1 4 5 7 16 31


1
1 2 4 5 7 16 31

4. The degree of a node in a tree refers to the number of edges incident on it (that is, the number of edges that connect to it). In the case of a binary tree, the maximum degree of any node is 3. The total degree refers to the sum total of every node's degree. For the following questions, either draw a tree with the given specification or explain why no such tree exists:

    a) Tree, 12 vertices, 15 edges

This is not possible. With a single vertex there are no edges. With each addition of one vertex a single edge is added. So, the number of vertices is always one more than the number of edges.

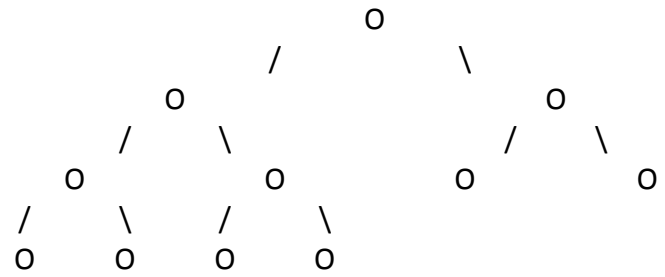
$V = E + 1$

    b) Tree, 5 vertices, total degree 10

This is not possible. Each edge is connected to exactly two vertices. Thus, each edge contributes 2 to the total degree. A tree with 5 vertices has 4 edges and thus a total degree of 8.

Total Degree $= 2E = 2(V-1) = 2V - 2$

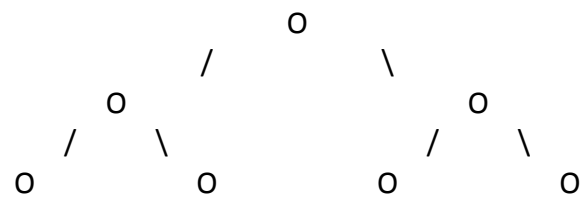
    c) Full binary tree, 5 internal (non-leaf) vertices



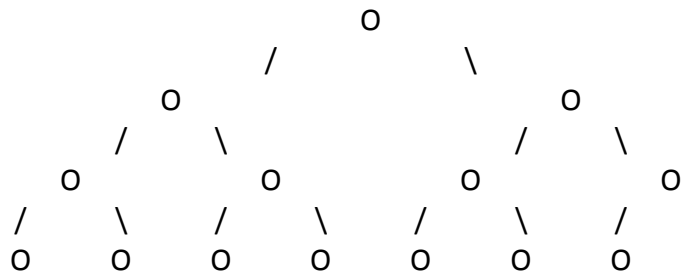
    d) Full binary tree, 8 internal (non-leaf) vertices, 7 leaves

This is not possible. In a full binary tree each non-leaf vertex (parent) has exactly two leaves (children). There is always one more leaf than non-leaf vertices. A tree with a single vertex has no non-leaf vertices and one leaf. Every time two leaves are added one of the leaves becomes a parent (adding one to the total number of non-leaf vertices) and two leaves are added, but one leaf has been re-classified as a parent (adding one to the total number of leaf vertices). Thus the relationship is maintained and there is always one more leaf than non-leaf vertices.

    e) Complete binary tree, 4 leaves



f) Nearly complete binary tree, 7 leaves



5. What is the height of the tallest possible nearly complete binary tree with 240 leaves? (Do not draw the tree – instead show your calculation.)

A nearly complete binary tree of height 1 has $2^1 - 1 = 1$ leaf.
A nearly complete binary tree of height 2 has at most $2^2 - 1 = 3$ leaves.
A nearly complete binary tree of height 3 has at most $2^3 - 1 = 7$ leaves.

A nearly complete binary tree of height h has at most $2^h - 1$ leaves.

$$2^7 - 1 = 127$$
$$2^8 - 1 = 255$$

Thus, a nearly complete binary tree with 240 leaves would have a height of 8.