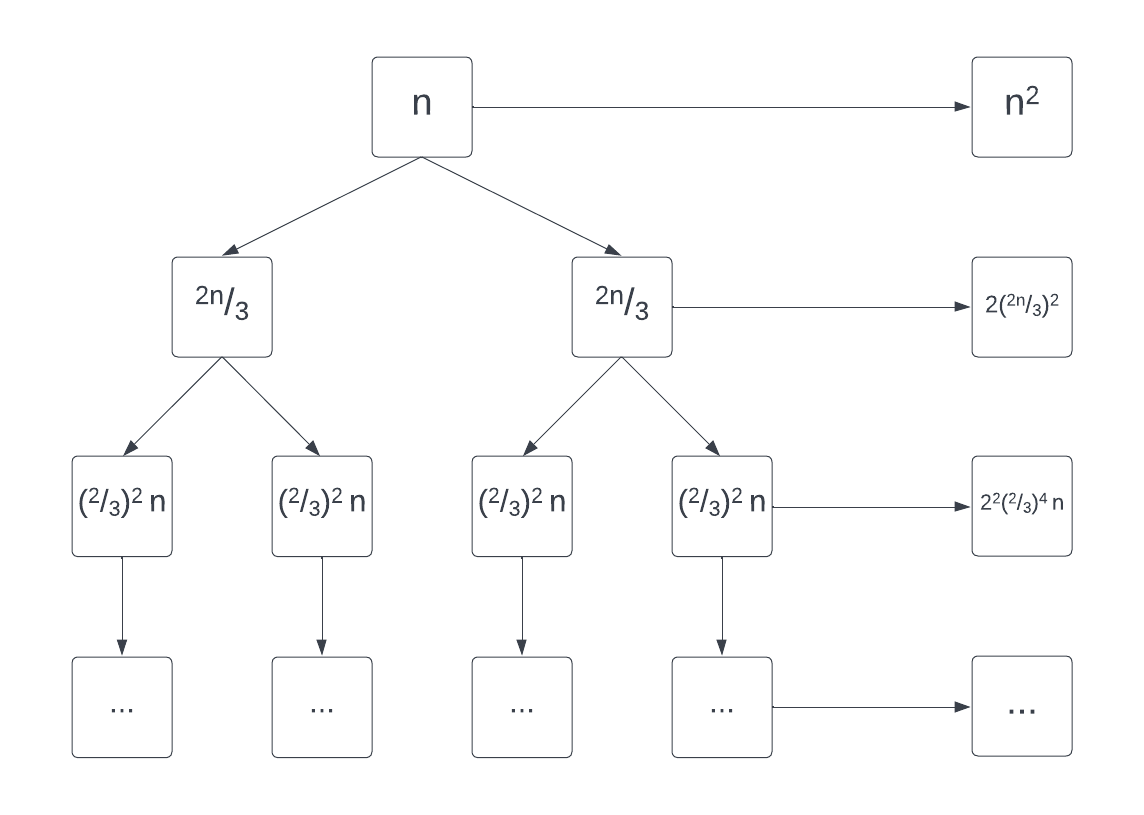
**Problem 1 (5 points):**

**Solve the recurrence  first by using a recursion tree and then using the Master theorem. Show work.**

Recursion Tree:

First, we calculate cost at each level:



Assuming k levels, at the kth level we see:

Finally, identify asymptotic bound:

Master Theorem:

**Problem 2 (5 points):**

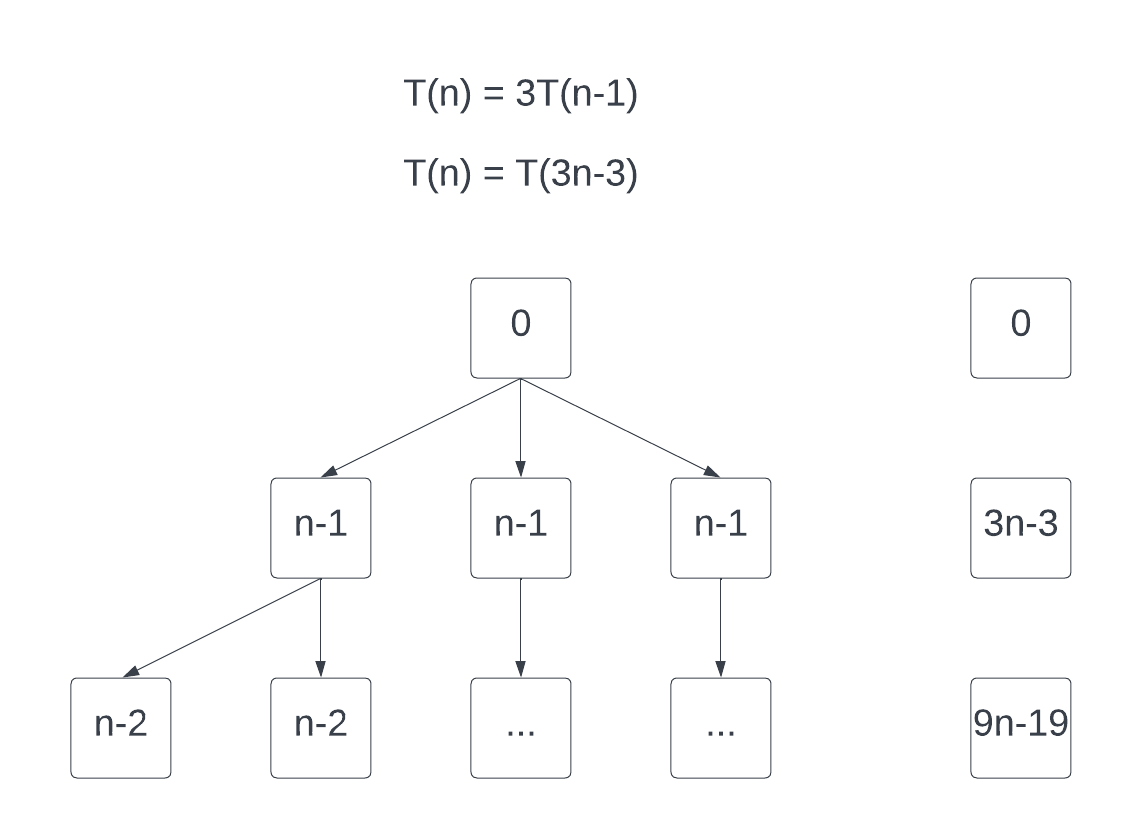
**Give asymptotic upper and lower bounds for the recurrence . Justify your answer.**

< n

**Problem 3 (10 = 5 + 5 points):**

**Give asymptotic upper and lower bounds for each of the following recurrences. Justify your answer.  
(a)**

**(b)**



**Problem 4: Programming Assignment (10 points):**

* **Maximum Product Subarray**
* **Given an integer array nums, find a contiguous non-empty subarray within the array that has the largest product, and return *the product*.**
* **The test cases are generated so that the answer will fit in a 32-bit integer.**
* **A subarray is a contiguous subsequence of the array.**

Please find a zip file attached with the code provided in the form of a Jupyter notebook. Using **Python3**, you can install Jupyter Notebook via ‘pip install jupyter notebook’. Once installed, you can run the commands you will see below.

The following figure shows the function **maximumSubArrayProductFinder** which takes one argument, an array **nums**. The function will check for an empty or too large of an array, before going through the given array to find the largest product of a sub array.



The following figure show 11 test cases developed to demonstrate the both the examples included in the assignments specification, as well as a few others. Each test case has an **input\_array\_n**, the **function** being executed with that array, as well as the **output**. These test cases show the utility of the application to handle the ability of finding sub array products, as well as ensuring that edge cases are handled properly.

