Modeling recursive code and the Tree Method

Due date: Friday April 24, 2020 at 11:59 pm

Instructions:

Submit your responses to the "Exercise 2" assignment on Gradescope here: https://www.gradescope.com/courses/97095. Make sure to log in to your Gradescope account using your UW email to access our course.

These problems are meant to be done **individually**. If you do want to discuss problems with a partner or group, make sure that you're writing your answers individually later on. Check our course's collaboration policy if you have questions.

1. Modeling recursive code

For the following problems, Give a recurrence formula for the running time of this code. Do not worry about finding the exact constants for the non-recursive term (for example if the running time is A(n) = 4A(n/3) + 25n, you need to get the 4 and the 3 right but you don't have to worry about getting the 25 right).

(a) Write a recurrence representing the runtime of the private method `printABCDStrings(String soFar, int sizeLeft)` in terms of `sizeLeft`. Please also assume that any + operations (String concatenation included) and `System.out.println()` calls take constant runtime. The public method is provided for context.

```java
/*
* Prints every string of length 'resultSize' composed of 0 or more
* a's, b's, c's, and d's.
*
* Note: If you took CSE 143 here, you might recognize that this is some
* recursive code that does "recursive backtracking". Recursive backtracking
* often has a corresponding decision tree that map out all possible states -
* these decision trees and their branching factors fit nicely into the
* framework of tree method analysis and modeling code as recurrences!
*/
public static void printABCDStrings(int resultSize) {
    printABCDStrings("", resultSize);
}

private static void printABCDStrings(String soFar, int sizeLeft) {
    if (sizeLeft == 0) {
        System.out.println(soFar);
    } else {
        printABCDStrings(soFar + "a", sizeLeft - 1);
        printABCDStrings(soFar + "b", sizeLeft - 1);
        printABCDStrings(soFar + "c", sizeLeft - 1);
        printABCDStrings(soFar + "d", sizeLeft - 1);
    }
}
```

2. The tree method

Consider the following recurrence:

$$A(n) = \begin{cases} 5 & \text{if } n = 1 \\ 4A(n/2) + n^3 & \text{otherwise} \end{cases}$$

We want to find an *exact* form of this equation by using the tree method.

(a) Draw your recurrence tree. Your drawing must include the **top three levels of the tree**, as well as a portion of the final level. It should be in the same style as the Section 3 slides Tree (link: https://tinyurl.com/y22b83zd). In particular, write both the **work** and **input size** for each node (if you draw nodes too small to fit both of these labels inside, you can put the labels nearby, but make it clear what the label is for each node). **Label** i for each level except the last one. Don't worry about explicitly drawing all the nodes as the levels increase; just showing the general pattern is good enough, as this portion is mostly just for your own understanding.

For this drawing portion, upload a pdf or image of your work. You can either scan or photograph a hand-written drawing or use an online drawing site like https://awwapp.com/ and export your drawing to pdf or an image format.

(b) What is the size of the **input** to each node at level i? As in class, we call the root level $i = 0$. This means that at $i = 0$, your expression for the input should equal n.

(c) What is the amount of **work** done by a single node at the i-th recursive level?

(d) What is the total number of nodes at level i? As in class, we call the root level $i = 0$. This means that at $i = 0$, your expression for the total number of nodes should equal 1.

(e) What is the total work done across the i-th *recursive* level? Be sure to show your work.

(f) What value of i does the last level of the tree occur at? Be sure to show your work.

(g) What is the total work done across the base case level of the tree (i.e. the last level)? Be sure to show your work.

(h) Combine your answers from previous parts to get an expression for the total work. You do not have to simplify the summation, just combine your previous answers up to this point into one expression.

Since this will require a summation in gradescope, you can copy paste the following: `$$\sum_{i=0}^{n - 1} i$$`. This will convert to

$$\sum_{i=0}^{n-1} i$$

in Gradescope, and you can change the bounds or the inner expression as necessary.