Hi everyone,

As you code project # 0, be sure to meticulously follow the specifications that I have outlined on the handout. For the analysis to make any sense, you must not only submit working code but also code that follows the specifications. Here are some details, among others, that you need to consider:

1. Implement the power algorithm exactly as outlined in the table on the handout. Handle all the trivial cases using 'if' statements with the relevant Boolean expressions. Avoid unnecessary evaluation of Boolean expressions when implementing this algorithm: this may require the use of nested 'if' statements when handling the trivial cases. Also, some kind of loop will be required to handle the non-trivial cases, the last two rows of the table. You will need a loop for the repeated multiplication of the base to determine the power.

2. Do not use the standard math library pow function/method in your code. It has some optimization that will skew the analysis if it is used.

3. 'fastLn' should not use the 'iPow' function or a nested loop. You will need to generate each succeeding term from its predecessor and accumulate all the terms to determine the sum. This requires some kind of loop to do the accumulation.

4. 'naiveLn' calls the 'iPow' function and it calculates each term independently and then sums those terms. This requires some kind of loop to do the accumulation.

5. You might want to save (x-1)/(x+1) to a variable, say alpha, so that you don't have to repeatedly do the +, - and / operations in the expression.

6. In 'naiveLn' you might want to save alpha squared to a variable, say alphaSqr, so that you don't have to multiply alpha*alpha each time you need the square of alpha. You should just calculate it once and retrieve alphaSqr from memory anytime you need it.

7. Do not add any additional functions/methods to the code other than those mentioned in the specifications. Complete the functions in NaturalLogger where indicated. NaturalLogProfiler, should only consist of one function/method, the main.

8. Before attempting to generate run times for your code, the table at the end of the output, verify that naiveLn(double, int) and fastLn(double, int) are working correctly and are giving you the correct values as shown in the sample run. Use additional test cases. The approximations may not be exact but they shouldn't be way off either.

Most of these are already outlined in the handout but emphasizing them here doesn't hurt. If you need clarifications on any of the algorithms, take advantage of my office hours. You can also see Rohan if you cannot make my office hours.

William E. Duncan, PhD
School of Electrical Engineering and Computer Science
Louisiana State University
3270C Patrick F. Taylor Hall, Baton Rouge LA 70803
Phone: 225-578-eighty-nine forty-one
Fax: 225-578-fourteen sixty-eight
Webpage: https://csc.lsu.edu/~duncan
