Math 146 - Calculus II Name________________________

How fast can you work? [1]

For this activity we are going to look at a measurement of how computers work to do a given task - computer scientists call these *fundamental operations*. Measuring the number of fundamental computer operations is important in our world of big data!

Computer scientists measure the efficiency of a computer program (what can be called an algorithm) by the number of fundamental operations (such as adding, comparing, multiplying, or dividing numbers) it takes to complete a task. In most cases the cumulative number of operations is a function of the input size n. For example, let's say your algorithm was to square a number n. It would take more fundamental operations to compute a large n, such as 347634^2 versus 5^2. One way computer scientists denote the efficiency of this algorithm is with "big O" notation. Algorithms may have a different efficiency, such as $O(n^2)$ or $O(n \ln n)$, which approximates the number of fundamental operations needed to complete the calculation for a given value of n. For this activity we will focus on comparing different functions $f(n)$ in $O(f(n))$.[2] An efficient algorithm needs fewer operations.

1. You are considering four different algorithm speeds:

   A: $n \ln(n)$

   B: $n(\ln(n))^2$

   C: n^2

   D: $e^{\ln(2)n}$ (This is really just 2^n, but we are going to write it this way (see footnote 2).

   Complete the following table, rounding to the nearest whole number.

n	1	10	100	1000	10000	10^7
A						
B						
C						
D						

2. Based on the evidence in the table, which algorithm do you think is the most efficient as n increases?

---

[1] This example is based on problem 4.7.110 in your textbook

[2] Computer scientists also use logarithms of base 2, i.e. $\log_2(n)$ or $\lg(n)$. For ease of use we will just stick with the natural logarithm. For large n the distinctions between $\lg(n)$ and $\ln(n)$ are not as important.

3. Let's try to confirm our results using a graph. Even though we typically think of n as a discrete variable ($n = 1, 2, 3...$), let's make a continuous plot of each of the algorithms. Choose your x-axis to span from 0 to 100 and your y axis from 0 to 1000 and sketch and label your graph below. Which algorithm seems to be most efficient?

4. Another way we could investigate efficiency is to compare the ratio of these algorithms and evaluate the limit as n grows large. For example if we compare Algorithm A to Algorithm B we have:

$$\lim_{n\to\infty} \frac{\text{Algorithm A:}}{\text{Algorithm B:}} = \lim_{n\to\infty} \frac{n\ln(n)}{n(\ln(n))^2}$$

   (a) Simplify this expression, and then evaluate the limit as $n \to \infty$, using your graph and table as a guide. Write your answer below.

   (b) Based on the graph you made and the conclusion you found from the limit, which Algorithm (A or B) is more efficient?

5. Notice that you had some nice algebraic simplification in the last problem.

   (a) Explain why if we were to repeat this analysis for Algorithm C and Algorithm D this would not be the case.

   (b) Our tables and graphs seem to suggest that Algorithm C is more efficient. Another way we could evaluate this efficiency is to investigate the ratio of the rates of change of each algorithm, in the hopes of some nice simplification. If $f(n) = n^2$ and $g(n) = e^{\ln(2)n}$, develop expressions for the ratio of the rates of change:

   $$\lim_{n\to\infty} \frac{f'(n)}{g'(n)} =$$

   (c) You may notice that the ratio of the rates does not simplify algebraically. Both of these algorithms are increasing as n increases. Another comparison we could make is in regards how fast the rate of change is increasing (i.e. the second derivative):

   $$\lim_{n\to\infty} \frac{f''(n)}{g''(n)} =$$

   What is the result of this limit calculation? Which algorithm (C or D) has the smaller rate of increase, and therefore is more efficient?