Tri Dang
trisotall@csu.fullerton.edu
Due: Wednesday November 1, 2017

CS335 Project 2: Empirical Analysis

Is there a noticeable difference in the performance of the two algorithms? Which is faster, and by how much? Does this surprise you?

There is definitely a noticeable difference between the two algorithms. Algorithm 1 (greedy_max_protein) was much faster than algorithm 2 (exhaustive_max_protein). According to my mathematical analysis, if we let n = size(todo) and m = size(foods), where n < m, the function for algorithm 1 is nlog(n) + n + m, or O(m). This is because we are creating a subset of food objects that will meet our requirement, so the input will always be greater than the output of our program. Algorithm 2, on the other hand, has efficiency of $O(2^n)$, because we are generating all subsets of our input set, making it an algorithm that performs in exponential times.

Are you empirical analyses consistent with your mathematical analyses? Justify your answer.
Yes, the empirical results are consistent with my mathematical analyses. The growth of algorithm 1's time as input increases has a linear relationship. I was able to test inputs in the thousands without even breaking 1/10 of a second. Algorithm 2, on the other hand, broke 1/10 of a second when n = 15. This is a huge difference in performance, because as I tried to test n = 30, it simply took too long and I had to stop the test.

Is this evidence consistent or inconsistent with hypothesis 1? Justify your answer.
This evidence is inconsistent with hypothesis 1. While exhaustive search can produce correct outputs, they are not feasible to implement due to how long it takes to compute the output as the size of n grows. It can be simple to code and implement, but the time it takes to process even a small amount of input really cripples the entire program.

Is this evidence consistent or inconsistent with hypothesis 2? Justify your answer.
The evidence is consistent with hypothesis 2, because exponential running times are too slow to be of practical use. For a database with few elements, yes, an exponential algorithm can be useful, but in reality, databases often store millions of elements of data. Producing subsets for those millions of elements would take an extremely long time.

Tri Dang
trisotall@csu.fullerton.edu
Due: Wednesday November 1, 2017