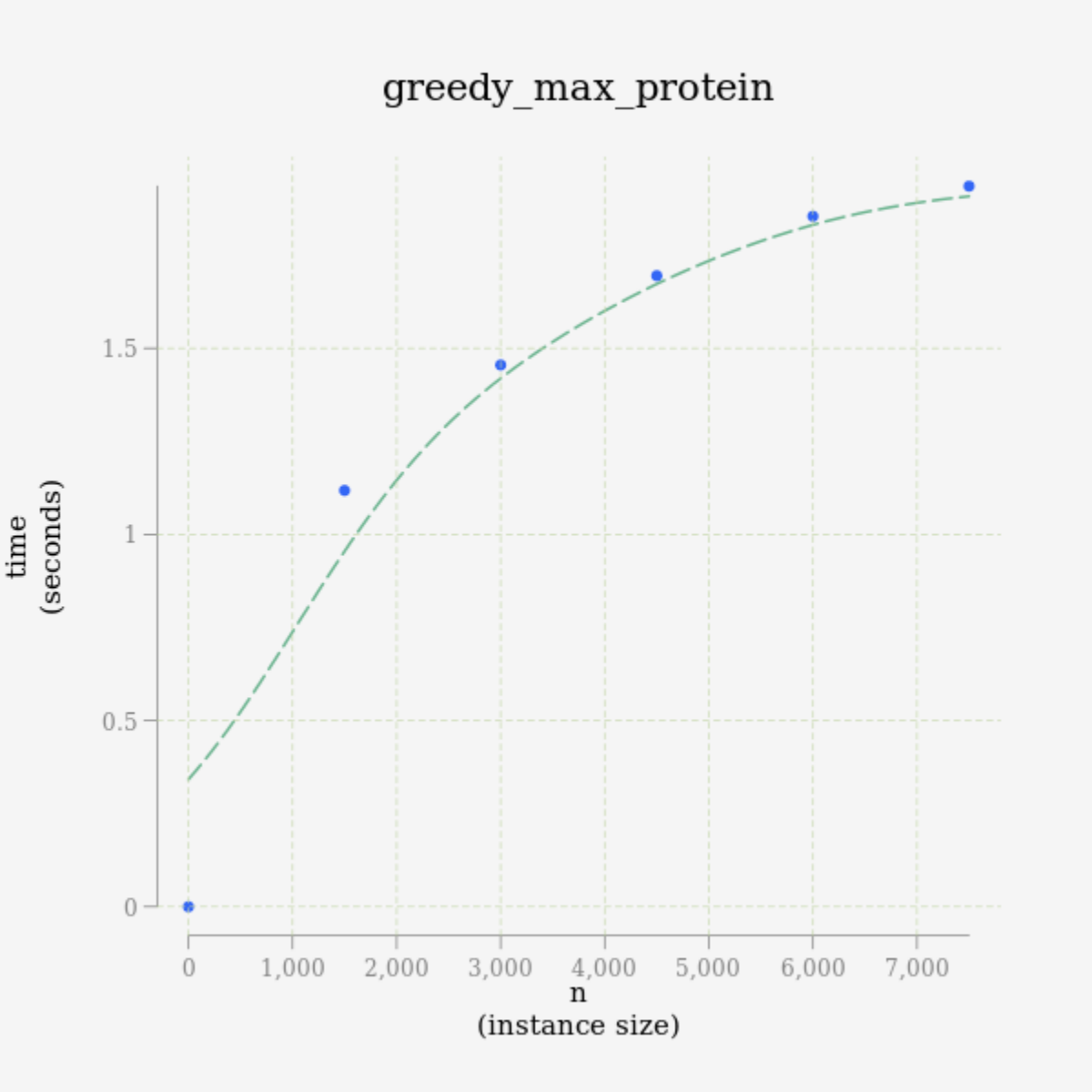
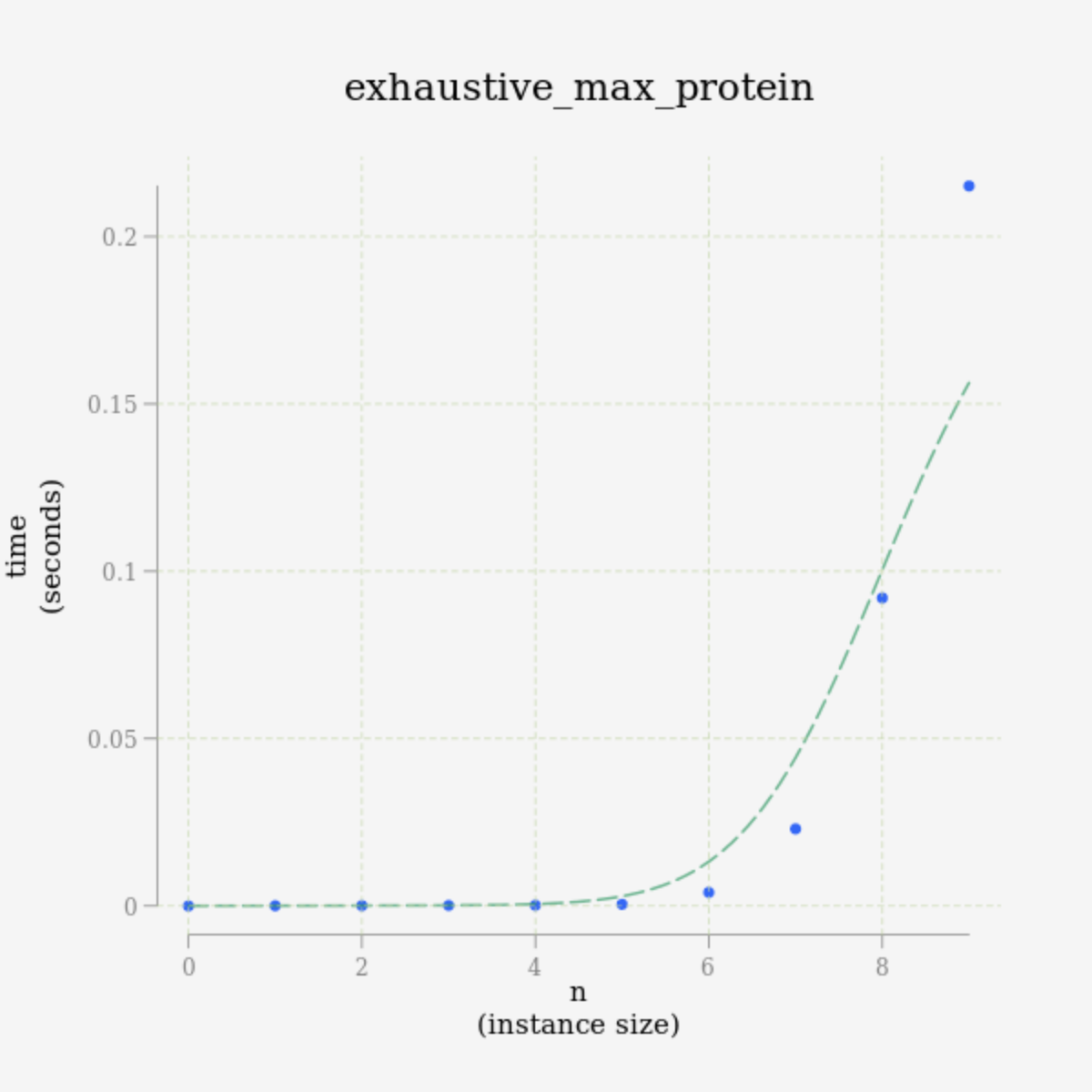
This is a submission for project 2

Austin Draper austindraper@csu.fullerton.edu





Q1. 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 a very large difference in the performance of the two algorithms. The greedy function runs significantly faster than the exhaustive search. While the greedy runs at O(n log(n)) time, the exhaustive runs at O(2^n * n) which is absurdly worse! This isn't very surprising since greedy (after sorting) just looks for the best in a greedy fashion (hence the name) while the exhaustive search is using subsets in order to find the best possible solution. Since subsets cause a run time of O(2^n) it is pretty obvious without even running the code that greedy would run much faster.

Q2. Are your empirical analyses consistent with your mathematical analyses? Justify your answer.

Yes, my empirical analysis for greedy came out to be O(n log(n)). Our hypothesis stated that without a sort it would run at O(n^2) and with a sorting algorithm could get it down to O(n log(n)). The timings I got in my empirical analysis of exhaustive definitely shows a rate of O(2^n * n).

Q3. Is this evidence consistent or inconsistent with hypothesis 1? Justify your answer.

This evidence is consistent with hypothesis 1. The exhaustive search algorithm I implemented was feasible and it also produces the correct outputs. It's really slow, but it wasn't too hard to code.

Q4. Is this evidence consistent or inconsistent with hypothesis 2? Justify your answer.

The evidence is consistent with the hypothesis 2. The code runs increasingly slow (exponentially) with each element added to the vector. It is most likely too slow to be of real practical use; however, it is feasible to use as long as small amounts of elements are sent to it.