Matthew Noack matthewnoack@csu.fullerton.edu
<center>Project 3</center>

1) Analyze the efficiency of the exhaustive algorithm mathematically to determine its big-O efficiency class, probably similar to the exhaustive algorithm in Project 2 .

The big O of the Exhaustive Method is $2^{(n+m)}$. This is due to number of iterations that the generate_all_subsequences() generates has a for loop of two to the power of the size of the first set of subsequences and a for loop of two to the power of second set of subsequences.

2)Analyze the efficiency of the dynamic programming algorithm mathematically to determine its big-O efficiency class .
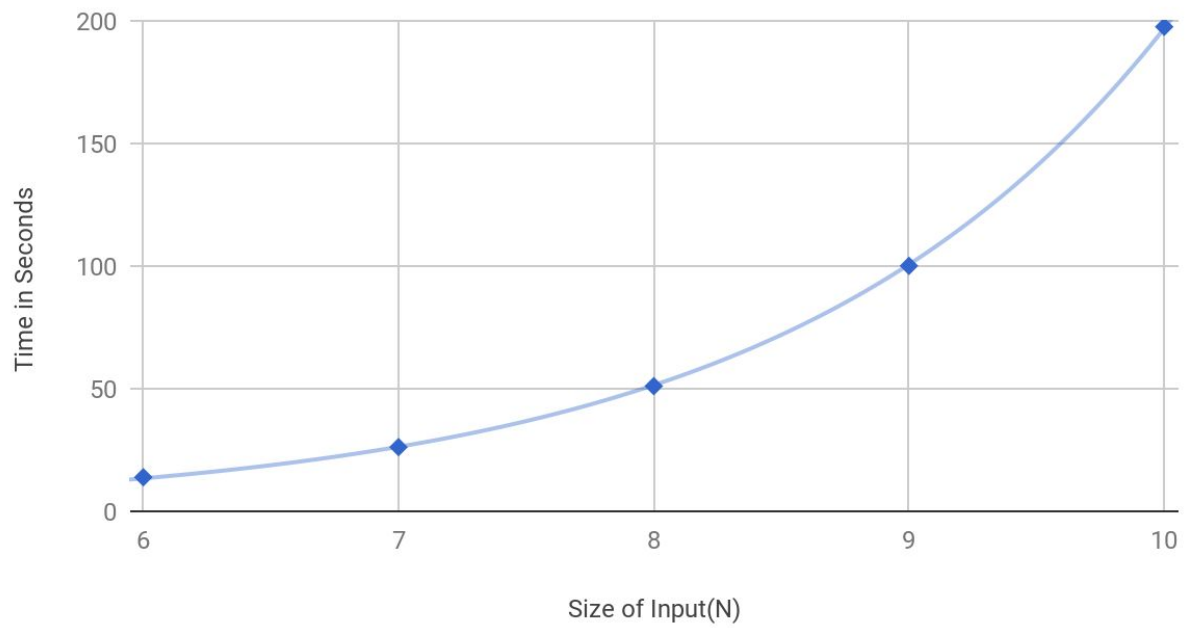
The big O of the Dynamic Programming is $O(n*m)$. This is due to a nested for loop iterating over the size of n and m.

3a) In both the exhaustive and dynamic methods, the fourth string matches to a different protein strain than sp|P32469|DPH5_YEAST Diphthine synthase. These results are still correct due to the time it took to complete the methods being within expected results predicted for each method.

4) The empirical data matches the mathematically derived big O for both the Exhaustive Method and Dynamic Programming. Since the only change that occurs with each program call is the String to Match(n), the list of proteins(m), is a constant. Therefore, the Exhaustive Method grows at a $O(2^n)$ rate, as shown in the Exhaustive Method graph, and the Dynamic programming grows at a $O(n)$ rate, as shown in the Dynamic Programming graph.

Exhastive Method

Dynamic Programming