Alberto Rocha
June 21, 2018
Project 3

```
detect_subsequence(a,b)
      subsets = subset(b)                          2^n * n
      for sub in subsets                                n
       if a == b                                        1
            return true                                              1
return false                                          1
```

$T(n) = 2^n * n + n(2) + 1 \rightarrow T(n) = 2^n * n + 2n + 1$

$2^n * n + 2n + 1 \; \varepsilon \; O(2^n * n + 2n + 1)$ *trivial*

$O(2^n * n + 2n + 1) = O(2^n * n + n + 1)$ *multiplicative*

$O(2^n * n + n + 1) = O(2^n * n + n)$ *additive*

$O(2^n * n + n) = O(max(2^n * n, n))$ *dominate term*

$= O(2^n * n)$

Time Complexities:

Exhaustive_longest_substring:

$T(n) = n^3 + 5n^2 + 5$

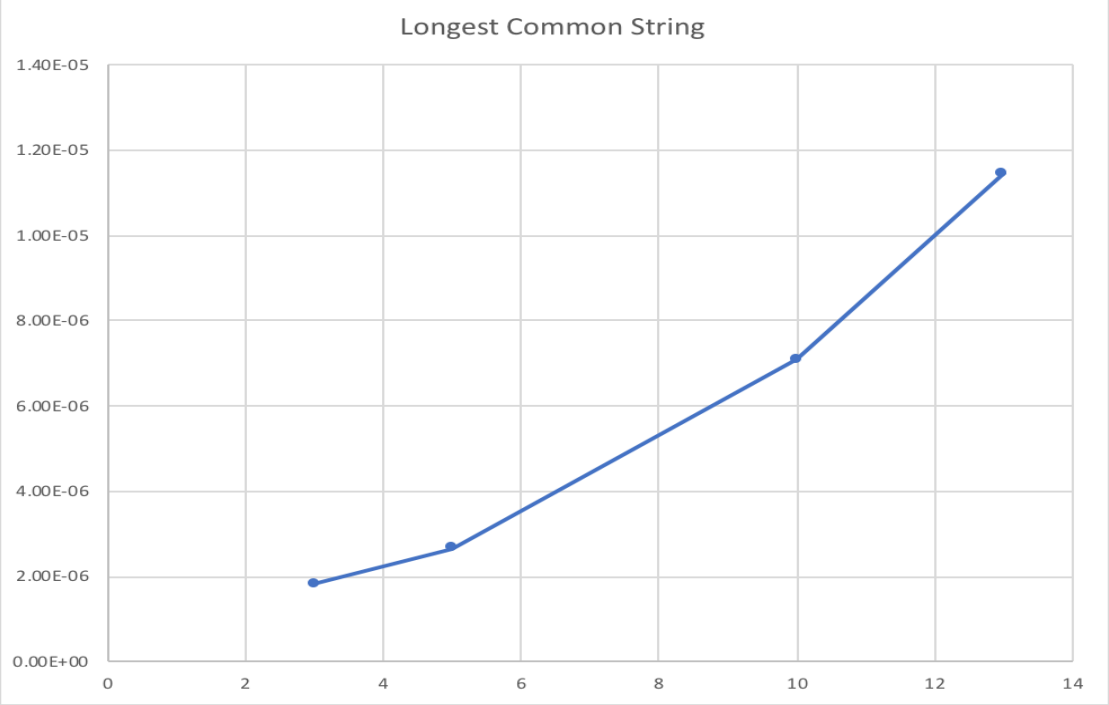
$n^3 + 5n^2 + 5 \; \varepsilon \; O(n^3 + 5n^2 + 5)$ *trivial*

$O(n^3 + 5n^2 + 5) = O(n^3 + n^2 + 5)$ *multiplicative*

$O(n^3 + n^2 + 5) = O(n^3 + n^2)$ *additive*

$O(n^3 + n^2) = O(max(n^3, n^2))$ *dominate term*

$= O(n^3)$



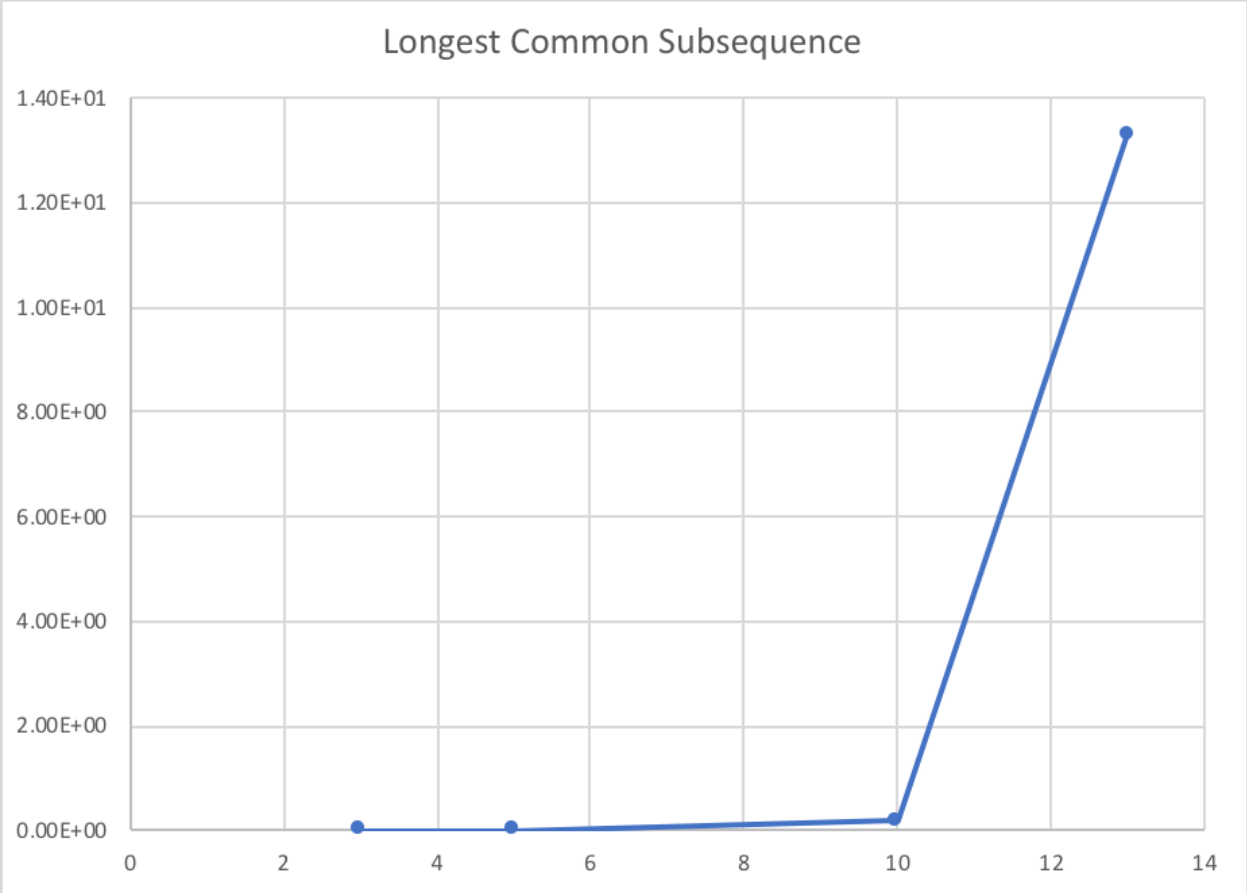
Exhaustive_longest_substring:

$T(n) = 9 + 2^n * n$

$9 + 2^n * n \, \varepsilon \, O(9 + 2^n * n)$ *trivial*

$O(9 + 2^n * n) = O(2^n * n)$ *additive*

$= O(2^n * n)$



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

A: Yes, there is a huge difference between the time complexity between the two functions. The Exhaustive_longest_substring is a lot faster than Exhaustive_longest_substring. It was not surprising given that the first function was n^3 and the second function $2^n * n$. I was surprised however, when I started to test the could and when I set the n value to 20 it took a long time.

Q: Are your empirical analyses consistent with your mathematical analyses? Justify your answer.

A:Given the two time complexities and the scatters plot charts I can conclude that the mathematically-derived big-O classes are consistent because chart one follows the graph of a function that is n^3. The same can be said about the second graph, which follows that of an exponential graph.

The Hypotheses:

1. Exhaustive search algorithms are feasible to implement, and produce correct outputs.

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

A:The evidence is consistent with hypotheses one because it states that it was feasible to implement. The whole function was able to be implemented with about 10 lines of code. The hardest part of the coding process was figuring out how find() method worked. It also produced the correct code. It was tested using the conditions provided by the professor.

2. Algorithms with exponential running times are extremely slow, probably too slow to be of practical use.

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

A:The evidence is consistent with hypotheses two because it states the exponential algorithms are too slow. It would get to the point that when n>15 it was taking more than a minute to compute. I also tried 20 when by that point I force the program to terminate.