Project 3: Polynomial vs Exponential Time

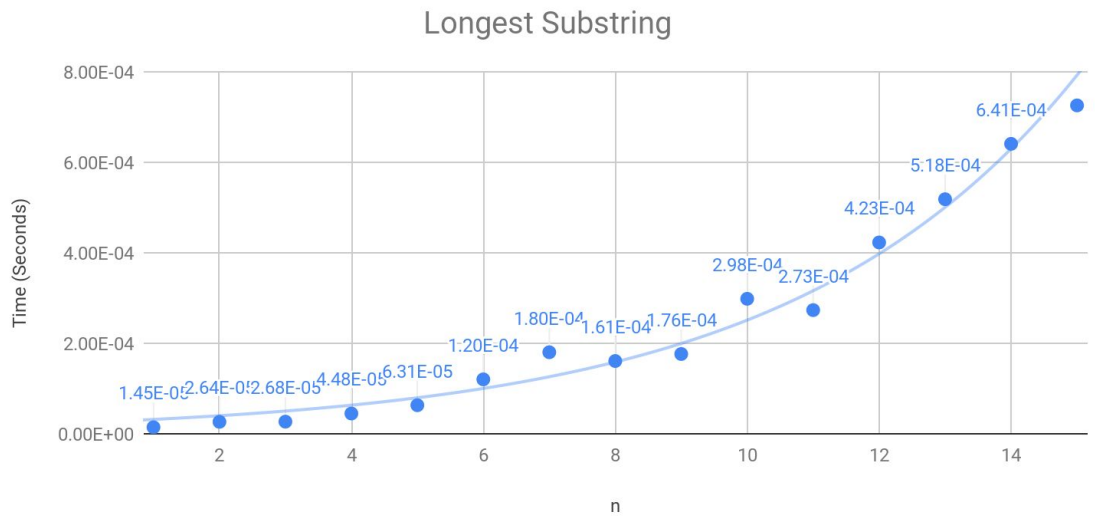
Team Members

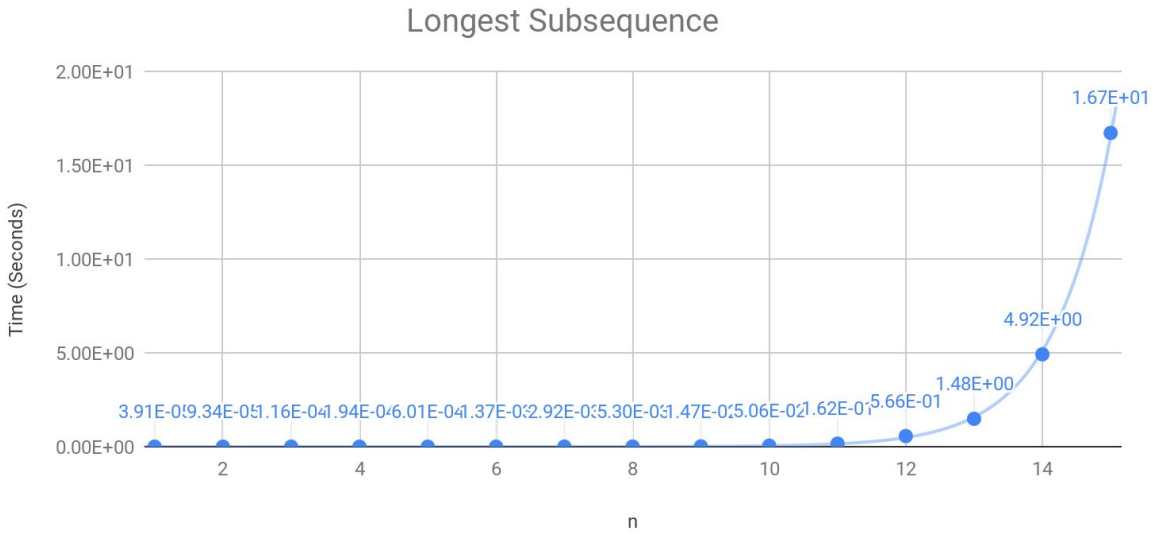
Jasper Chen jsprchn2@csu.fullerton.edu
Alex Ma alexjmma@csu.fullerton.edu
William Bui wqbui3@csu.fullerton.edu

Hypothesis

1. *Exhaustive search algorithms are feasible to implement, and produce correct outputs.*
2. *Algorithms with exponential running times are **extremely** slow, probably **too slow** to be of practical use.*

Graphs





Questions

1. Write your pseudocode for subsequence detection.

```
detect_subsequence(sub, super)
	leftover = 0

	if (length(sub) > length(super))
		return false
	else if (sub is empty)
		return true
	else if (super is empty)
		return false

	for i in sub
		for k in super
			 if sub[i] == super[k]
				i++
		leftover = length(sub) - i
		if leftover > 0
			return false
	return true
```

2. Analyze your subsequence detection algorithm; state and justify a time efficiency class.
   $T(n) = 2^n(n(3) + 1) + 5$
   $= O(2^n n)$

   We conclude that this is our efficiency class because of generating the subsets for the shorter string. The amount of subsets is 2 to nth length of the string then afterwards is sent into a for loop to look through every possible subset that is a subsequence of the longer string, excluding any duplicates.

3. 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 indeed a noticeable difference in performance between the two algorithms. Since longest substring takes n3 time and longest subsequence takes $2^n n$ time. We justify this by running our timing algorithm is there is a significant time difference between the two. When we reach n = 15 the time for longest substring is 0.000725687 compared to 16.7151. This is mostly because finding the subsequence we have to run through detect. This does not surprise us since it takes a lot more time to find a search for which character to remove to obtain the longest subsequence.

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

   Our empirical analysis is consistent with our mathematical analysis because we know that if an algorithm runs in exponential time it is usually very slow. We proved this by running our timing test and it displaying extremely slow times. It is preferred that algorithms run in constant or linear time. Exponential times are indeed usable but in practicality it is not.

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

   It is evident that the hypothesis is correct in some ways. The algorithm is easy to implement but it takes way too much steps to compute it. We have found that it would be much faster and efficient to use dynamic programming or use a suffix tree. However those two are definitely not feasible and require much more thinking. We agree that using an exhaustive search is much easier and we also get the correct output. It may sometimes however take more time to run as prove by our longest subsequence algorithm.

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

   Yes, the evidence from getting the longest subsequence is consistent to the hypothesis which becomes impractical to use if "n" is to large. Given that the efficiency class of the detect subsequence is $O(2^n n)$, the exponent rate from how long the string is justifies the amount of subsets and how long it will take to find a single correct subsequence to another string, if the length gets as large as 14 - 16.