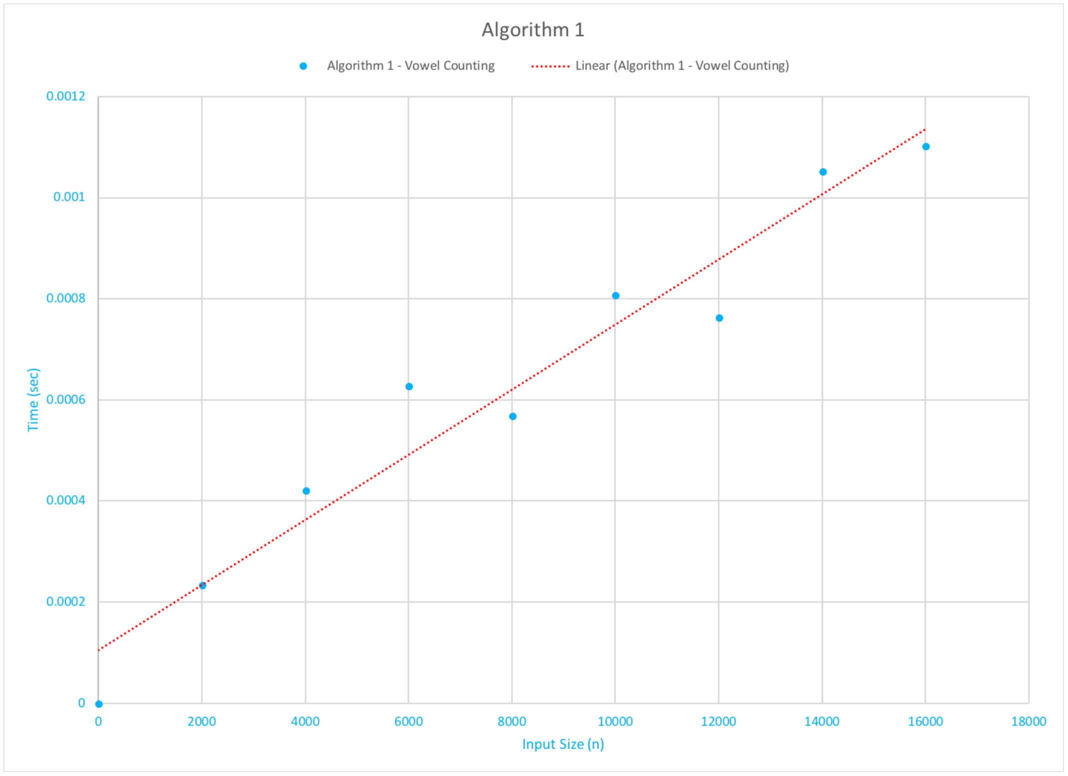
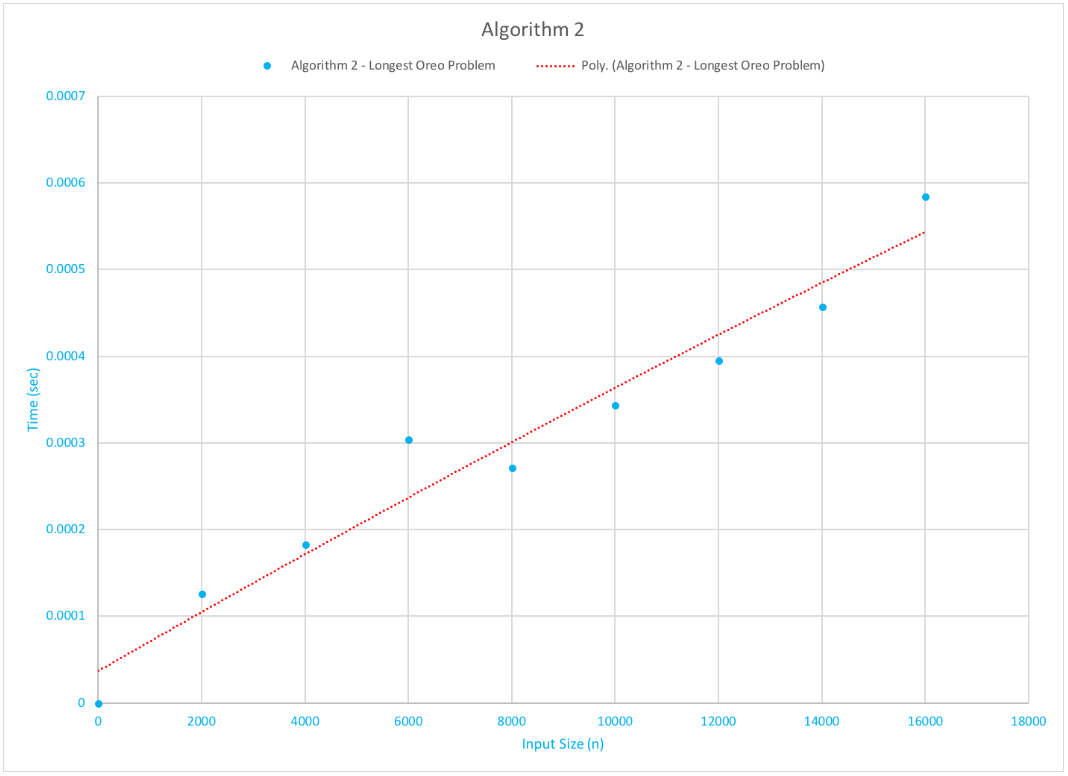
CPSC 335

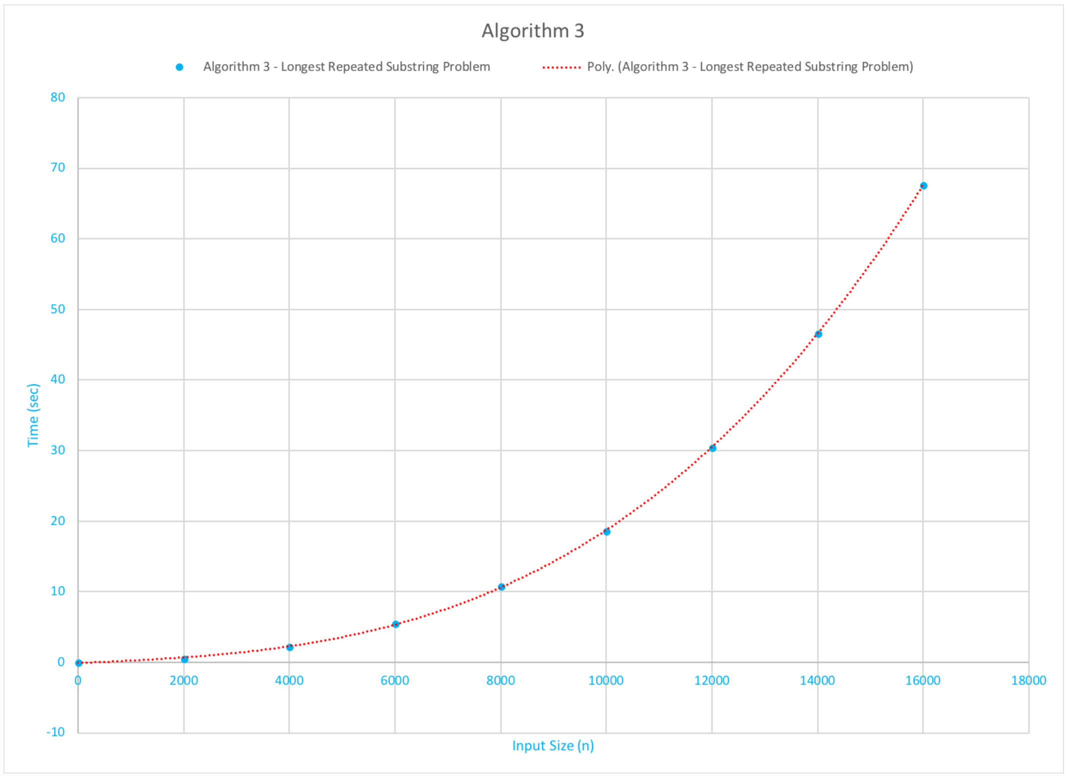
Project 2: Empirical Analysis

Devontae Reid devontae.reid@csu.fullerton.edu

Scatter Plots






Algorithm 3
Algorithm 3 - Longest Repeated Substring Problem
Poly. (Algorithm 3 - Longest Repeated Substring Problem)
Time (sec)
80
70
60
50
40
30
20
10
0
-10
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
Input Size (n)

Questions

1. Provide pseudocode for your three algorithms.

   a. Vowel Counter

```
vowel_count(s):
        set vowel_count to 0                        1
        for i from 0 through n-1                           |
                if s[i] is a vowel                  1      | n – 1
                        increment vowel_count by 1  1      |
        return vowel_count                          1
```

   b. Longest Oreo

```
longest_oreo(s):
        if s is empty
                return ""                           1
        set result to ""                            1
        for i from 0 through n-1                                   |
                if s[i] is equal to s[n-1]          1              |
                        for j from i through n-1           |       | n – 1
                                add character to result 1  | n – 1 |
                        return result               1              |
        return result                               1
```

   c. Longest Repeated Substring

```
longest_repeated_substring(s):
        if s is empty
                return ""                           1
        set n to s.length                           1
        set result to ""                            1
for i from 0 through n-1                                           |
        for j from i+1 through n-1                         |       |
                set lcp to longest_common_prefix    2n     |       | n – 1
                if lcp.length > result.length       1      | n – 1 |
                        set result to lcp           1      |       |
return result                                       1

longest_common_prefix(s,t):
        set n to minimum length of the two strings s,t  1
        for i from 0 through n-1                           |
                if s[i] is equal to s[n-1]          1      | n – 1
                        return s.substr(0,i)        1      |
        return s.substr(0,n)                        1
```

2. What is the efficiency class of each of your algorithms, according to your own mathematical analysis? (You are not required to include all your math work, just state the classes you derived and proved.)

   a. Vowel Counter

   $$\begin{aligned} T(n) &= 1 + (n - 1) * (1 + 1) + 1 \\ &= 2n - 2 + 2 \\ &= 2n \therefore O(n) \end{aligned}$$

   b. Longest Oreo

   $$\begin{aligned} T(n) &= 1 + 1 + (n - 1) * (1 + (n - 1) * (1) + 1) + 1 \\ &= (n - 1) * (n + 1) + 3 \\ &= n^2 + 2 \therefore O(n^2) \end{aligned}$$

   c. Longest Repeated Substring

      i. $$\begin{aligned} T(n_1) &= 1 + (n - 1) * (1 + 1) + 1 \\ &= 2n - 2 + 2 \\ &= 2n \therefore O(n) \end{aligned}$$

      $$\begin{aligned} T(n_2) &= 1 + 1 + 1 + (n - 1) * ((n - 1) * (2n + 1 + 1)) + 1 \\ &= (n - 1) * (2n^2 - 2) + 4 \\ &= 2n^3 - 2n^2 - 2n + 6 \therefore O(n^3) \end{aligned}$$

3. Is there a noticeable difference in the running speed of the three algorithms? Which is faster, and by how much? Does this surprise you?

   a. Yes, I notice that there is a big difference in performance from the first two algorithms to the third algorithm. The second algorithm seems to be the fastest out of all of them.

4. Are the fit lines on your scatter plots consistent with these efficiency classes? Justify your answer.

   a. Yes, the fit lines are consistent with the efficiency classes that were chosen. Such as for the third algorithm the best fit line seemed to be the polynomial line to the third order. Also, the first and second fit similar to their efficiency class.

5. Is this evidence consistent or inconsistent with the hypothesis stated on the first page? Justify your answer.

   a. Yes, this evidence is consistent with the hypothesis stating that "For large values of n, the mathematically-derived efficiency class of an algorithm accurately predicts the observed running time of an implementation of that algorithm." I notice that the larger values on the graph were closer to the fit line.