CS 335 Project 1
Empirical Analysis

9 October 2017

Arshdeep Singh (Arshi95@csu.fullerton.edu)
Elizabeth Tsan (ElizabethChen@csu.fullerton.edu)
Steven Kha (SKha@csu.fullerton.edu)

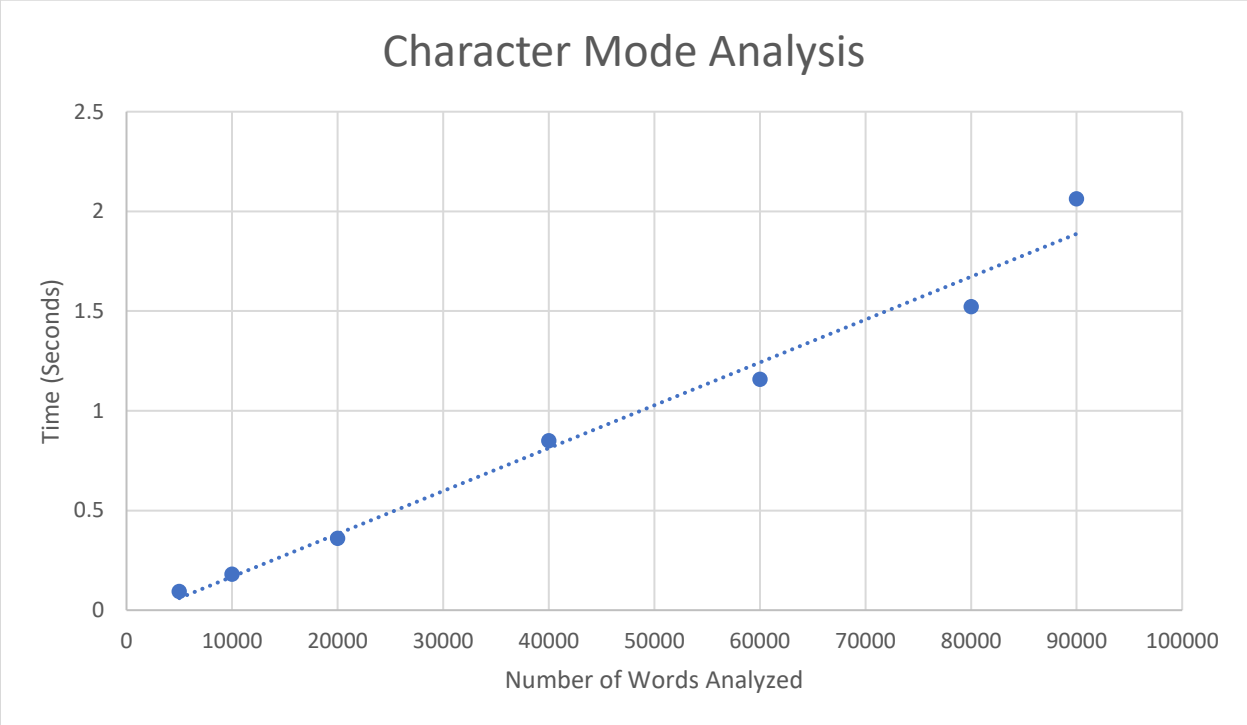
Character Mode Problem

```
def character_mode(V):
    counts = Array(256, 0)                                          1
    for string in V:                                                1
        for character in string:                                    1
            counts[character]++                                     1
    c = <find the character c with the greatest value of counts[c]> 1
    return c                                                        1
```

}c }n

An initial glance may lead one to believe that the analysis will be close to T(n) = 3n(2c) +3; however, as it was indicated in the explanation for this assignment that *"We will treat the length of words as constant, […] . Therefore, for our purposes the input size will refer to the number of words, not the number of characters, in a vector of strings."* This makes the analysis T(n) = 3n +3.

Theorem: if T and F are complexity functions and f(n) > 0 and lim (n -> ∞) T(n)/f(n) = L where L is a defined non-negative constant then T(n) ∈ O(f(n)).
Proof: lim(n -> ∞) [(3n+3)/n] = 3
∴ 3n + 3 ∈ O(n) ■



As can be seen from the scatterplot the values are roughly linear. This is consistent with what was seen in the time analysis of the pseudocode.

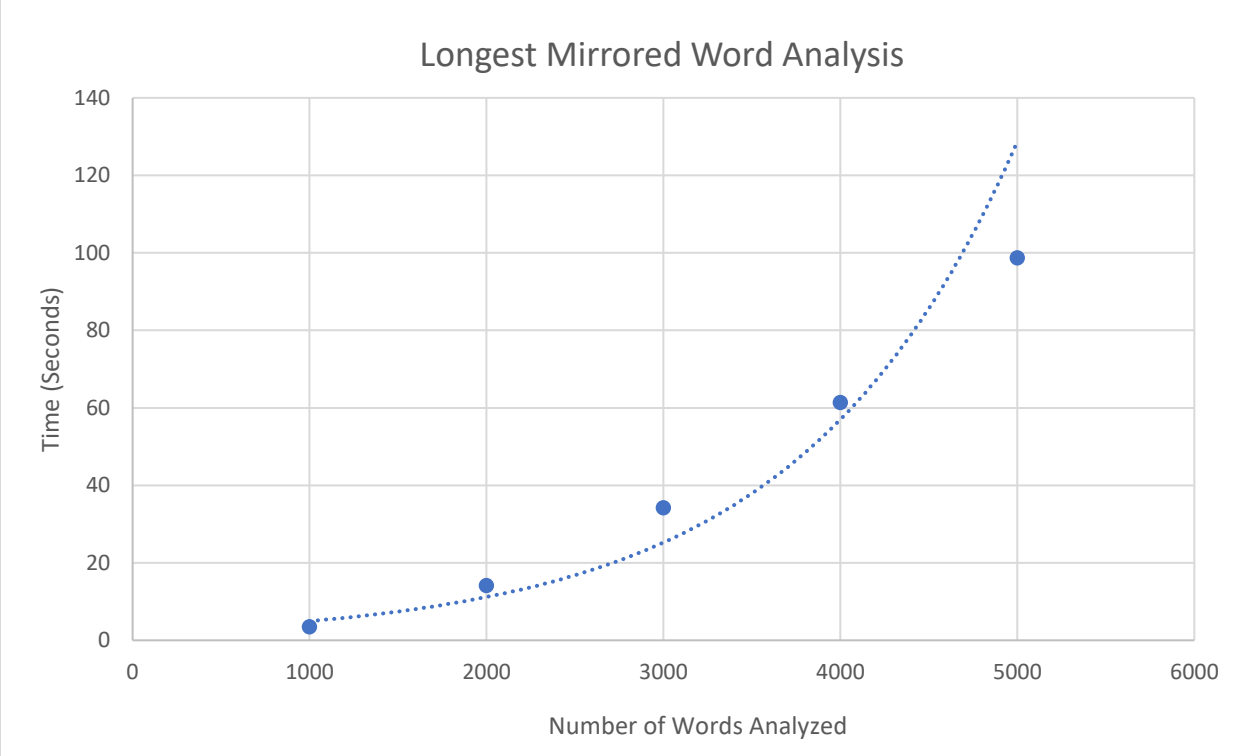
Longest Mirrored String Problem

```
def longest_mirrored_string(V):
        best = “”                                                   #1
        for a in V:                                                 #n 1
                for b in V:                                         #n 1
                        If a is the mirror of b and len(a) > len(best):   #1
                                best = a                            #1
        return best                                                 #1
                = (1+(n(1 + n(3))+1)
                = (1 + n + 3n^2 + 1)
           T(n) = 3n^2 + n +2
```

Theorem: if T and F are complexity functions and f(n) > 0 and lim (n -> ∞) T(n)/f(n) = L where L is a defined non-negative constant then T(n) ∈ O(f(n)).

Proof: lim(n -> ∞) [(3n^2 + n +2)/n^2] = 3

∴ 3n^2 + n +2 ∈ O(n^2) ■



Judging from this graph, the observed analysis of this function is running in quadratic time. This is consistent with my analysis of the function’s pseudocode which was O(n^2).

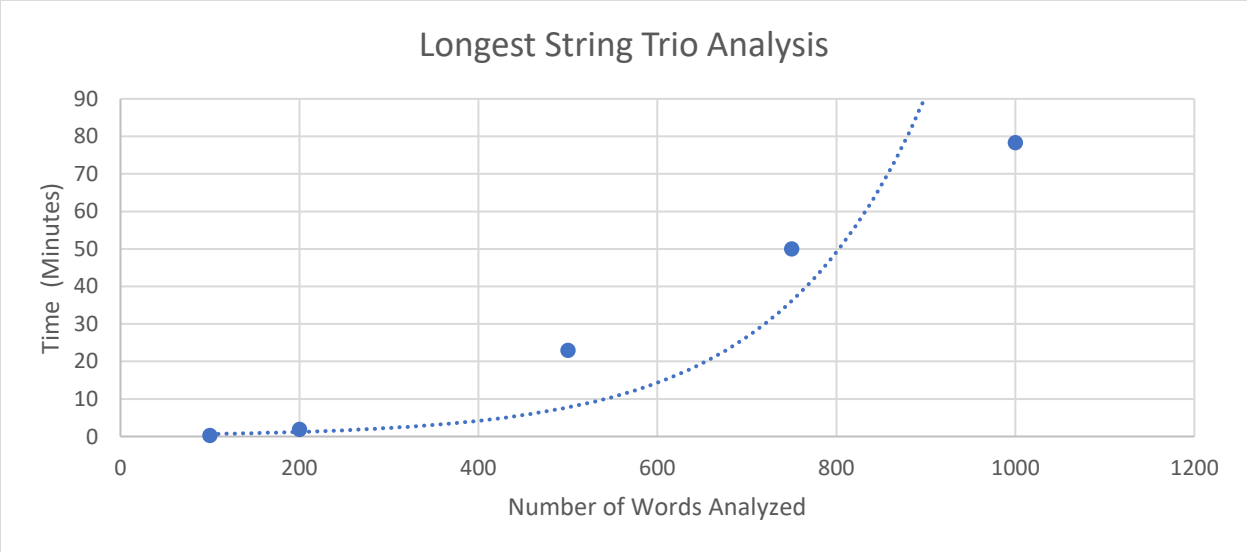
Longest Subset Trio Problem

```
def longest_subset_trio(V):
        best_length = 0                                          #1
        trio = Vector(3, “”)                                     #1
        for a in V:                                              #n
            for b in V:                                          #n
                for c in V:                                      #n
                    abc_length = |a|+|b|+|c|                     #1
                    if <a, b, c are a subset trio> and
                      abc_length > best_length:                  #1
                        best_length = abc_length                 #1
                        trio[0] = a                              #1
                        trio[1] = b                              #1
                        trio[2] = c                              #1
        return trio                                              #1
```

$= (1 + 1 + n(1+n(1+n(6)))+1)$
$= (2+ n(1+n+6n)+ 1$
$= 2 + n + n2 + 6n^3 + 1$
$T(n) = 6n^3 + n^2 + n + 3$

Theorem: if T and F are complexity functions and f(n) > 0 and lim (n -> ∞) T(n)/f(n) = L where L is a defined non-negative constant then T(n) ∈ O(f(n)).

Proof: $\lim(n \to \infty) [(6n^3 + n^2 + n + 3)/ n^3] = 1$
$\therefore 6n^3 + n^2 + n + 3 \in O(n^3)$



Looking at this graph, the observed analysis of the function Longest_Substring_Trio is running in cubic time. This is reasonably consistent with my analysis of the function’s pseudocode which was $O(n^3)$.

Final Analysis

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

   There is an extreme difference between the three functions. The fastest, was Character Mode, in the middle was Longest Mirrored String and the slowest was Longest String Trio. It was to be expected after the analysis that Longest String Trio with n^3 time it would be slow but doing more than 1000 words was almost ridiculously untenable with over an hour of run time. It was surprising that even a very fast computer had trouble completing this.

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.)

   We proved that Character Mode was linear, Longest Mirrored String was quadratic, and Longest String Trio was cubic.

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

   The efficiency classes for Character Mode and Longest Mirrored String have the best fit lines but Longest String Trio does not. This is partly a fault of using Excel to make the graphs which has a trendline option which includes exponential but is not very specific. Also, it should be noted that because the mathematically-derived classes are both worst-case and machine dependent they are a guide, not a rule.

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

   The hypothesis was
   "*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.*"
   We believe that our observations are consistent with the hypothesis. The time mathematically derived is worst case and can vary by machine; but certainly, the functions ran fastest when provided slower-growing functions.