Eric Timmerman, Stephanie Ison, Geoffrey Corey
October 15, 2013
CS 325 Proj 1

1 Run-time Analysis

1.0.1 Enumeration 1

Let 0 to end be n
Let the index be k
First loop = n amount of work
the second for loop = $n - k$ amount of work
Which = $n(n-k)$
$n(n-k) = n^2 - nk$
$\lim_{n\to\infty} n(n-k) = n^2 - nk \leq n^2$
Therefore, Enumeration 1 is $O(n^2)$

1.0.2 Enumeration 2

Let 0 to end be n
Let *index* be j
The first for loop = m amount of work
The second for loop = $\frac{(m-j)}{2}$ amount of work
the third for loop = $\frac{(m-j)}{2}$ amount of work
This is close to $m(m-j)$
$m(m-j) = m^2 - mj$
$\lim_{n\to\infty} m(m-j) = m^2 - mk \leq m^2$
Therefore, Enumeration 2 is $O(m^2)$

1.1 Pseudo-code

1.1.1 Enumeration 1

```
enumeration1(list)
    $minimum = \infty$
    $total = 0$
    for $i$ in $list$
        $total = 0$
        for $j$ from $i$ to end of $list$
            $total += list[j]$
            if |total| $< minimum$
                $minimum =$ |total|
                $left = i$
                $right = j$
    return $i, j$
```

1.1.2 Enumeration 2

```
enumeration2(list)
    $minimum = \infty$
    $total = 0$
    $dir = right$
    for $i$ in $list$
    if $dir = right$
        $total = 0$
        for $j$ from $i$ to end of $list$
            $total += list[j]$
            if |total| $< min$
                $min =$ |total|
                left $= i$
                right $= j$
    if $dir = left$
        for $j$ from end of $list$ to $i$
            $total += list[j]$
            if |total| $< min$
                $min =$ |total|
                left $= i$
                right $= j$
    $dir = !dir$
return $i, j$
```

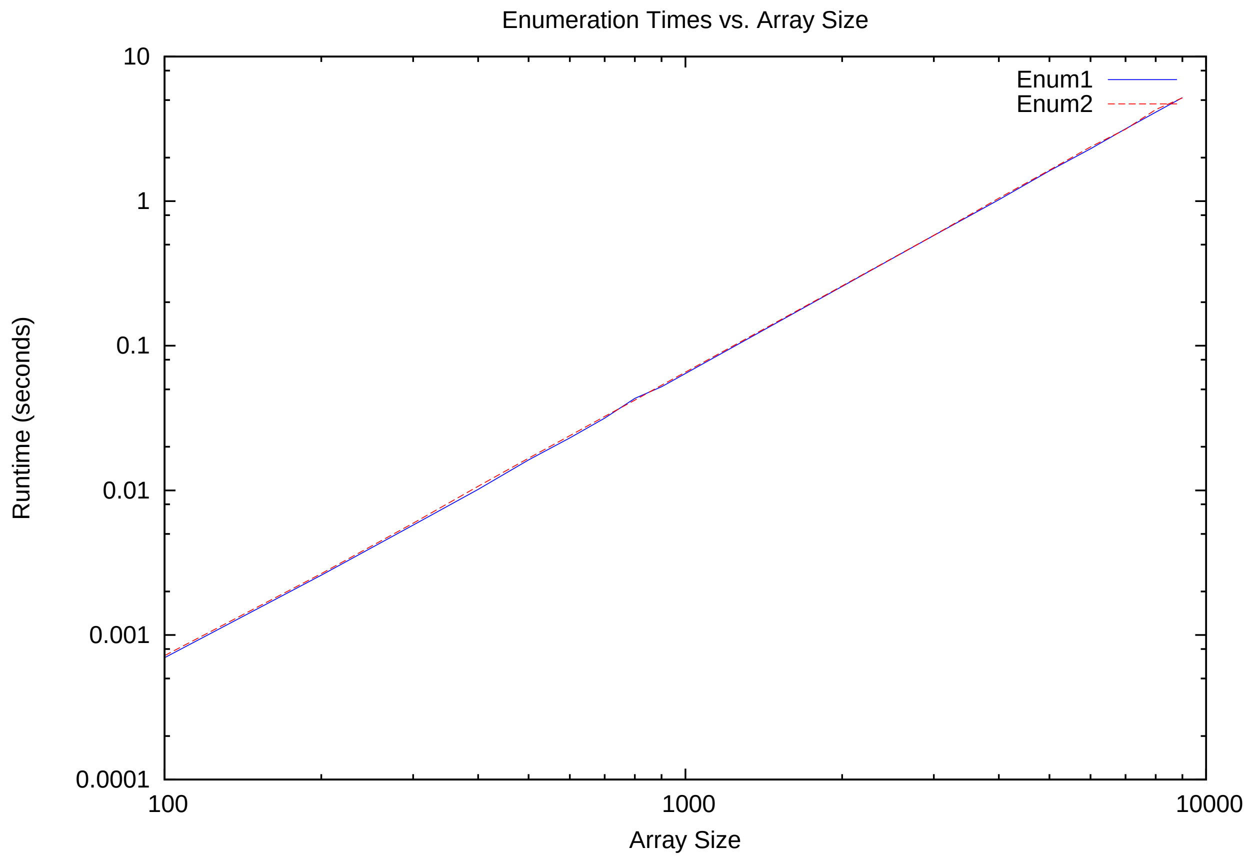
1.2 Asymptotic Analysis

1.2.1 Enumeration1

1.2.2 Enumeration2

2 Experimental Analysis

```
Size    Enumeration1            Enumaeration2
---------------------------------------------
100     0.000696                0.000719
200     0.002591                0.002654
300     0.005745                0.005936
400     0.010156                0.010666
500     0.016255                0.016689
600     0.023073                0.023882
700     0.031572                0.032457
800     0.043199                0.042115
900     0.052044                0.053318
1000    0.064217                0.065591
2000    0.257333                0.259802
3000    0.578875                0.581656
4000    1.021853                1.047431
5000    1.620242                1.640189
6000    2.302268                2.375077
7000    3.166515                3.149312
8000    4.118694                4.301560
9000    5.197848                5.182940
```



3 Extrapolation and interpretation

3.1 Length to Process for 1 Hour?

Using

$$Slope = m = \frac{log(\frac{F_1}{F_0})}{log(\frac{x_1}{x_0})} \tag{1}$$

Using (1) to get the slope of enum1, I get 1.96504. Plugging this value back into the slope equation and using 3600 for one of the x values,

$$1.96504 = \frac{log(\frac{0.000696}{3600})}{log(\frac{100}{x})}$$

and solving for the only x value shows that and array of length $260,958$ should require 1 hour to process. And since both the enum1 and enum2 plots are very similar, the length of an array to require a processing time of 1 hour for enum2 would also be extremely close to $260,958$.

*used `http://en.wikipedia.org/wiki/Log-log_plot` for help understanding log slope.

3.2 Discrepancies Discussion

3.2.1 Log Plots

Using (1) I get that the slope of enum1 is 1.96504.

$$F(x) = C * x^m, C = constant \tag{2}$$

After finding the slope, enum1 is $C * x^{1.96504}$ from using (2), which roughly matches with the $O(n^2)$.

3.2.2 Discrepancies

Since both enumeration methods had extremely similar results, the expected asymptotic analysis for the intent of the 2nd enumeration method is different than what we found.

Also, there is a slight discrepancy between the asymptotic analysis of the first enumeration method and the results from the experiment ($1.96504 \neq 2$) because of slight differences in the programming language as well as difference in how and what the computer was running at the time of the tests. However, this was minimized by taking the average of 10 runs for each array size to account for this fluctuation.