Objectives of this assignment:

- to explore time complexity and "real time"
- to "dust off" programming skills

What you need to do:

1. Implement a silly (naïve and inefficient) algorithm A to compute the sum $\sum_{i=1}^{n} x^i$ where x is a real number with $0 \le x \le 1$.
2. Collect the execution time T(n) of algorithm A as a function of n
3. Plot the functions T(n)/n, T(n)/n^2, and T(n)/n^3 on separate graphs.
4. Refer to the analysis of the time complexity your performed for your Module 1 and discuss it in light of the plots you plotted above.

Objective:

The objective of this programming assignment is to implement in your **preferred**[*] language an algorithm A to compute the sum $\sum_{i=1}^{n} x^i$ where x is a real number $(0 \le x \le 1)$. We are interested in exploring the relationship between the time complexity and the "real time" (wall time). For this exploration, you will collect the execution time $T(n)$ of Algorithm A as a function of n and plot $\frac{T(n)}{n}$, $\frac{T(n)}{n^2}$,, and $\frac{T(n)}{n^3}$on different graphs. Finally, discuss your results: use the plots you will build to determine the time complexity of $T(n)$ and justify.

Algorithm A

```
ComputeSumPowers(x,n)
    inputs: $x$ is a real number with $0 \le x \le 1$. $n$ is an integer ($n \ge 1$)
    output: a real number equal to $\sum_{i=1}^{n} x^i$
    sum = 0
    for i = 1 to n
        prod = 1
        for j = 1 to i
            prod = prod * x
        sum = sum +prod
    return sum
```

Answer These Questions (to put you on the right path to analyze the results)

Insert your answers in THIS file after each question

a) Suppose that for n very large, $T(n) \approx Kn$ where K is a constant.

i) **(0.5 point)** What would then the values of T(n)/n, T(n)/n^2, and T(n)/n^3 be, respectively?

T(n)/n = Kn/n = K
T(n)/n^2 = Kn/n^2 = K/n
T(n)/n^3 = Kn/n^3 = K/n^2

ii) **(1.5 points)** What would then the **shapes** of the plots T(n)/n, T(n)/n^2, and T(n)/n^3 be, respectively?

T(n)/n would be flat, and grow as a constant ***K***, making the time complexity of this function ***n***. ***T(n)/n²*** and ***T(n)/n³***, would both appear flat on a graph as well, but below the constant ***K*** (because they are fractions), meaning that the time complexity cannot be ***n²*** or ***n³***.

[*] You can use any language as long as it is already installed on Engineering Unix Tux machines.

b) Suppose that for n very large, $T(n) \approx Kn^2$ where K is a constant.

i) **(0.5 point)** What would then the values of T(n)/n, T(n)/n², and T(n)/n³ be, respectively?

$T(n)/n = Kn^2/n = Kn$
$T(n)/n^2 = Kn^2/n^2 = K$
$T(n)/n^3 = Kn^2/n^3 = K/n$

ii) **(1.5 points)** What would then the **shapes** of the plots T(n)/n, T(n)/n², and T(n)/n³ be, respectively?

T(n)/n would be growing in an upward trend, meaning that the time complexity of ***T(n)*** is greater than ***n***, meaning it will be n^x where $x > 1$. ***T(n)/n²*** would appear linear at constant ***K***, confirming that the time complexity is ***n²***. ***T(n)/n³***, would appear flat on a graph as well, but below the constant ***K*** (because it is a fraction), meaning that the time complexity cannot be ***n³***.

c) Suppose that for n very large, $T(n) \approx Kn^3$ where K is a constant.

i) **(0.5 point)** What would then the values of T(n)/n, T(n)/n², and T(n)/n³ be, respectively?

$T(n)/n = Kn^3/n = Kn^2$
$T(n)/n^2 = Kn^3/n^2 = Kn$
$T(n)/n^3 = Kn^3/n^3 = K$

ii) **(1.5 points)** What would then the **shapes** of the plots T(n)/n, T(n)/n², and T(n)/n³ be, respectively?

T(n)/n would be growing in an upward trend dwarfing the other 2 functions, meaning that the time complexity of ***T(n)*** is greater than ***n***, meaning it will be n^x where $x > 1$. ***T(n)/n²*** would appear linear because of how large ***T(n)/n*** is, but it would not be linear at constant ***K***, meaning that T(n) is greater than ***n²***, meaning it will be n^x where $x > 2$. ***T(n)/n³***, would appear flat on a graph as well, but at the constant ***K***, meaning that the time complexity is ***n³***.

d) (4 points) Time complexity of Algorithm A:

Report on this table the time complexity you obtained for the silly algorithm in M1:Homework (Refer to your homework)

Operations	Total Operations	Grows as
Comparisons	$f_c(n) = \frac{1}{2}n^2 + \frac{5}{2}n + 1$	n^2
Additions (line 6)	$f_a(n) = a_1 + a_2 + a_3 \ldots + a_n$	n
Multiplications	$f_m(n) = \frac{n^2}{2} + \frac{n}{2}$	n^2

Program to implement (28 points)

State here whether your implementation worked and produced data.

The implementation for program CollectData.java worked perfectly and produced data. Data for n, ***T(n)/n***, ***T(n)/n²***, and ***T(n)/n³*** were collected in a .csv file called collectData.csv.

Provide here the instructions to compile and execute your program on a Tux machine.

Directions for **MAC:**

Transferring file onto Tux machine:

1. Open XQuartz or Mac Terminal
2. Find the directory that the program is on (in your local machine) – *for example, type cd Documents if the `CollectData.java` file is in the Documents folder*
3. Type `sftp username@sftp.eng.auburn.edu` – *username is your auburn username*
4. Enter `password` – *your Auburn password*
5. Create a directory on the remote machine to put the program file in by typing `mkdir directoryName` – *directoryName is the folder you put the program file into*
6. Type `cd directoryName` – *opens the directory you just created*
7. Type `put CollectData.java` – this will add the program file into the directory you just created.

Compiling & executing the program on a Tux machine:

1. Open a new XQuartx or Mac Terminal
2. Type `ssh username@gate.eng.auburn.edu`
3. Enter `password`
4. When it asks `Please enter the name of an Engineering host <anywhere>:` just press Enter
5. Type `yes` when it asks you if you want to continue
6. Enter `password` again
7. Type `cd directoryName`
8. Type `ls` – *You will now see the file `CollectData.java`*
9. Type `javac CollectData.java` – *this will compile the program*
10. Type `java CollectData` and this will execute the program – *it will take some time to execute*
11. Type `ls` again once the program is done executing and you should see the files `CollectData.java, CollectData.class,` and `collectData.csv`

Once you see the 3 files in the directory you created, it means that the program successfully compiled and executed on the Tux machine.

```
collectData()
        for n = 100 to L (with step 100)// L should be as large as your machine
                                        // and your available time allow
            Start timing // We time the transposition of Matrix A
            ComputeSumPowers(0.5,n)
            Store the value n and the values T(n)/n, T(n)/n², and T(n)/n³ in a
            file F where T(n) is the execution time.
            // Pay attention do not use n^2 or n^3. The ^ operator is often not the exponentiation. Rather, it
            // is the exclusive OR (XOR)
```

Data Analysis (42 points)

(3*7 points per plot) Use any plotting software (e.g., Excel) to plot the values T(n)/n, T(n)/n^2, and T(n)/n^3 in File F as a function of n (on different graphs). File F is the file produced by the program you implemented. Discuss your results based on the plots you obtain (3*7 points per plot discussion). Do not list here data as tables. Only plots are expected.

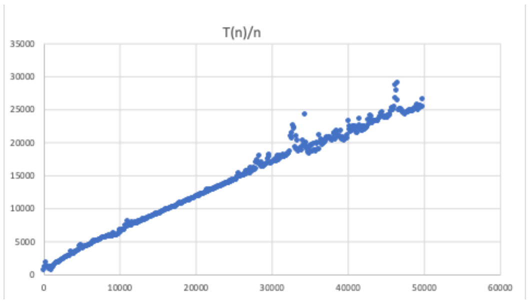

Figure 1

Above is **Figure 1,** the graph for $f_1(n)$ = $T(n)/n$. The graph for $f_1(n)$ grows in an upward trend, and is the highest and fastest growing function out of the 3. This graph exemplifies that $T(n)$ grows faster than n, meaning that the time complexity does **not** grow as n, but has to be n^x where x > 1.

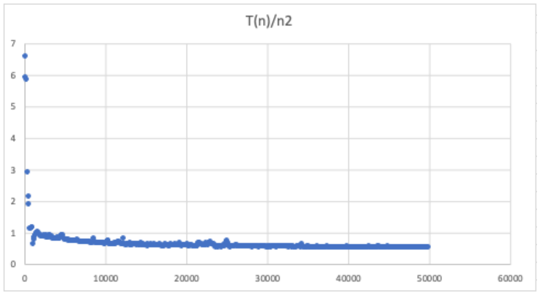

Figure 2

Above is **Figure 2,** the graph for $f_2(n)$ = $T(n)/n^2$. The graph for $f_2(n)$ appears flat, making it constant. This graph exemplifies that $T(n)$ grows at a constant rate around 1 when compared to n^2, meaning that the time complexity is n^2 because $T(n)$ scales at the same rate as n^2.

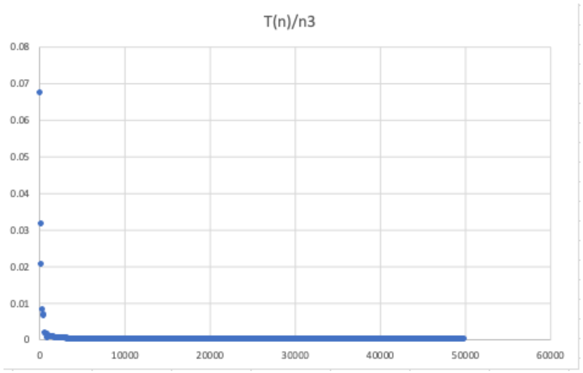

Figure 3

Above is **Figure 3,** the graph for $f_3(n)$ = $T(n)/n^3$. The graph for $f_3(n)$ also appears flat, but closer to the value of 0 than 1. This graph exemplifies that $T(n)$ grows at a rate <1 when compared to n^3, meaning that the time complexity does not grow at n^3, making the time complexity between n and n^3, n^2.

Improve Algorithm A

a) (**12 points**) Propose a more efficient algorithm to compute the sum $\sum_{i=1}^{n} x^i$ where x is a real number with $0 \le x \le 1$ such that the time complexity **grows as n**. **Use pseudocode to describe it**.

```
ComputeSumPowers($x$, $n$)
sum = 0
for $i$ = 1 to $n$
    sum = sum + $x$^$i$
return (sum)
```

This algorithm grows as n. We eliminated one for loop, making the algorithm more efficient.

B) (**8 points**) Propose a more efficient algorithm to compute the sum $\sum_{i=1}^{n} x^i$ where x is a real number with $0 \le x \le 1$ such that the time complexity is **constant** (independent of n). **Use pseudocode to describe it**.

```
ComputeSumPowers($x$, $n$)
sum = 0
sum = (x(1 – x^n)) / (1 – x)
return (sum)
```

This algorithm grows in constant time. Since we have formulas to compute summations, we are able to make this algorithm more efficient by eliminating any loops and calculate it using a summation formula. For $\sum_{i=1}^{n} x^i$ the formula is $\frac{x(1-x^n)}{1-x}$

ASK on Piazza for precisions if you have any doubts, concerns, or issues.
Let us know if you need help to work on Tux machines. (See at the end about how to log on Tux machines)

How to Plot?

I suggest to store the values in File F following the csv format used by Excel. Once the file F is in csv format, you can use Excel to plot.

If you do not know the csv format, google "csv format". Do not hesitate to ask for help if you need any.

Report

- Write a report using this file to insert your answers (Do not delete anything from this original file)
- Good writing is expected.
- Recall that answers must be well written, documented, justified, and presented to get full credit.
- Make sure that the TA has complete instructions/directions to compile and execute your program on Tux machines.

What you need to turn in:

- Electronic copy of your source program (**collectData**)
- Electronic copy of the report (including your answers) (standalone). Submit the file as a Microsoft Word or PDF file.

Grading

- Program is worth 28% if it works and provides data to analyze (Recall that your program must compile and execute on Engineering Unix Tux machines)
- Quality of the report is worth 72% distributed as described in the rubric per question/task above.

Login on Engineering Unix Machines,

Log in remotely on the Engineering Tux machines to implement, compile and execute. To log in remotely, you must use an **ssh** client such as SecureCRT (Windows).

On Windows 10, you may use from the command prompt the following command (if ssh is available):

```
ssh username@gate.eng.auburn.edu
```

where username is your Auburn University username (**without** @auburn.edu).

On Mac or any Unix machine (Ubuntu...), use the same command (see above) on a terminal.