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 to compute the sum where where and are a real numbers with .
2. Collect the execution time T(n) of algorithm A as a function of n
3. Plot the functions , , and 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[[1]](#footnote-1)\*** language an algorithm A to compute the sum where and are a real numbers (). 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 of Algorithm A as a function of n and plot , , and on different graphs. Finally, discuss your results: use the plots you will build to determine and justify the time complexity of .

**Algorithm A**

**ComputeSumPowers(x,n)**

**inputs:**  is a real number with . is a real number. is an integer()

**output:** a real number equal to

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**

These questions are meant as hints to analyze, predict, determine, and/or justify the shape of time complexities

**Insert your answers in THIS file after each question**

**a)** Suppose that for very large, where is a constant.

i) **(0.5 point)** What would then the values of , , and be, respectively? (just replace T(n) with

and simplify the expression you get).

.... answer here

ii) **(1.5 points)** Based on the expressions obtained in the previous question, what would then the **shapes** of the plots , , and be, respectively?

.... answer here

**b)** Suppose that for very large, where is a constant.

i) **(0.5 point)** What would then the values of , , and be, respectively? (just plug in T(n)

and simplify the expression you get).

.... answer here

ii) **(1.5 points)** Based on the expressions obtained in the previous question, what would then the **shapes** of the plots , , and be, respectively?

.... answer here

**c)** Suppose that for very large, where is a constant.

i) **(0.5 point)** What would then the values of , , and be, respectively? (just plug in T(n)

and simplify the expression you get).

.... answer here

ii) **(1.5 points)** Based on the expressions obtained in the previous question, what would then the **shapes** of the plots , , and be, respectively?

.... answer here

**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 |  |  |
| Additions (line 6) |  |  |
| Multiplications |  |  |

**Program to implement (28 points)**

.... answer here

**State here whether your implementation worked and produced data.**

**Provide here the instructions to compile and execute your program on a 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 // Note current time **start**

ComputeSumPowers(0.25,n)

Stop Timing // T(n) = Current Time - **start**

Store the value n and the values T(n)/, T(n)/n2, and T(n)/n.ln(n) in a file **F** where T(n) is the execution time.

// Pay attention do not use n^2. 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 , , and 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.

.... answer here

**Improve Algorithm A**

a) (**12 points**) Propose a more efficient algorithm to compute the sum such that the time complexity **grows as n**. **Use pseudocode to describe it**.

.... answer here

B) (**8 points**) Propose a more efficient algorithm to compute the sum such that the time complexity is **constant** (independent of ). **Use pseudocode to describe it**.

.... answer here

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 data you produce , , and
* Electronic copy of the report (including your answers) (standalone). Submit the file as a Microsoft Word or PDF file.

**Grading**

* Each question shows the number of points for it

**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.

1. \* You can use any language as long as it is already installed on Engineering Unix Tux machines. [↑](#footnote-ref-1)