**Report 1**

**Computer Science II**

When I refer to Algorithm #1,2,3 these are corresponding to problems 11.4, 11.5, and 11.10.

Algorithm #1 -> 11.4

Algorithm #2 -> 11.5

Algorithm #3 -> 11.10

*Prediction*

Theoretical Formula

Below are the theoretical formulas for the three Algorithms. These are important to display as a sort of “proof” that displays what kind of function appears for each algorithm and distinguishes which is the most useful for what certain type of situation.

Notation

* For the ease of typing and reading, I displayed the lines that contained just constants that are only read through once in the runtime of the program with the variable **C**. Examples of this can be allocating the registers, doing load instances, reading input values, etc.
* The values that are constants or functions that are consistently being looped through were denoted with the variable **D**. This includes functions such as sne, seq, mul, jeqz, j etc.
* The variable **n** represents the input specifically for the exponent variable since that is the number of times the loop within the algorithm is iterated.

Algorithm #1 formula: 9C + n(4D) + 2D + 3C

Algorithm #2 formula: 12C + (3D + 4D + 3D)logn\* + 4D + 3D + 3D

\*Notice the log function in this formula. This function is curtesy of the fact that every time the exponent went through the even function (because the exponent in this case started out as even), it was divided by 2.

Ex: (N, N/2, N/2/2, …, 0) This is an example of a logarithmic function.

Note about Algorithm #2: This formula for Algorithm #2 is for an exponent of power 2 to simplify not having to write two different types of formulas for the input of an odd and an even.

Algorithm #3 formula: 13C + n-1(9D) + 2D + 3D + n-1(9D) + 3D

Logic

Algorithm #1 and Algorithm #3’s formula’s display that they are both linear functions because their values for n have no function attached to them meaning that for each increase in n, there will be a constant, steady increase in “y” which is the number for the counter.

Algorithm #2’s formula displays that it is a logarithmic function. I deduced this below in the (\*) note.

The next portion of this write-up is to display the comparison of the theoretical formulas to the actual values that are displayed when using a constant increase in the exponent after being run through the algorithms.

*Verification*

One way to verify the results of this is to have a steady increase in the exponent being input and have a constant value for 5 through all of the trials. Multiple runs through all three algorithms can be seen below in Table 1.

Quantitative Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **b** | **e** | **Algorithm #1** | **Algorithm #2** | **Algorithm #3** |
| 5 | 5 | 33 | 46 | 82 |
| 5 | 15 | 83 | 72 | 222 |
| 5 | 25 | 133 | 71 | 358 |
| 5 | 35 | 183 | 79 | 498 |
| 5 | 45 | 233 | 88 | 638 |
| 5 | 55 | 283 | 97 | 782 |
| 5 | 65 | 333 | 78 | 922 |
| 5 | 75 | 383 | 96 | 1062 |
| 5 | 85 | 433 | 96 | 1198 |

*Table 1:* This table displays the quantitative values for the counter, which is the number of lines that the algorithms had to iterate through, that were found when running each of the algorithms with controlled variables b and e. The results are displayed below in section Logic/Analyze.

Visual Representation

*Visual 1:* This is a graph displaying the three algorithms in a line-plot with markers with the values for the counts on the y-axis and the number of rows in the table on the x-axis.

Logic/Analysis

As seen in Visual 1, the graph shows what we displayed as well in the logic statement in the theoretical section. We can see a constant linear progression from both Algorithm 1 and Algorithm 3 with Algorithm 2 tapering off. This “tapering off” description is exactly defining what a logarithmic function does. We see an initial spike from e being 5 to 15, but then as the values for e get larger, the slope starts slowing down eventually coming to a steady, constant number that in this case appears to be around the mid-nineties; however, I am sure further testing can help deduce a more precise calculation.

Conclusion

As a final statement, it is important to acknowledge the reasoning behind these analyses. The conclusions displaying that Algorithms #1 and 3 graph a linear function and Algorithm #2 graphs a logarithmic function are all tied to the idea of what the value of n is. If the value of n is larger perhaps in the 80 or higher region, it is reasonable to use Algorithm #2 due to the fact that the “running time” or the counter amount is significantly lower than that of Algorithm #1 or 3. However, if the value of n is smaller, it may make more sense to use Algorithm #1 or 3 due to the fact that initially, they are very equal, and perhaps if you do not have an n that is going to go high enough to make it worth it to construct a log function, you could save time and register space.