Problem Set 5

Task 1

The graphs are arranged from slowest to fastest as shown below:

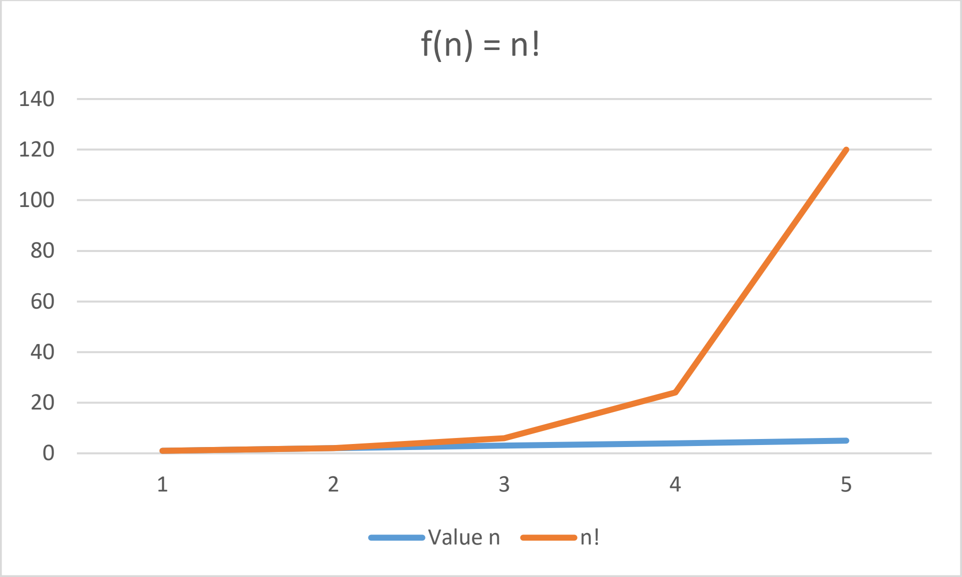

Figure 1: Graph of f(n) = n!

Expression in Big O: O (n!)

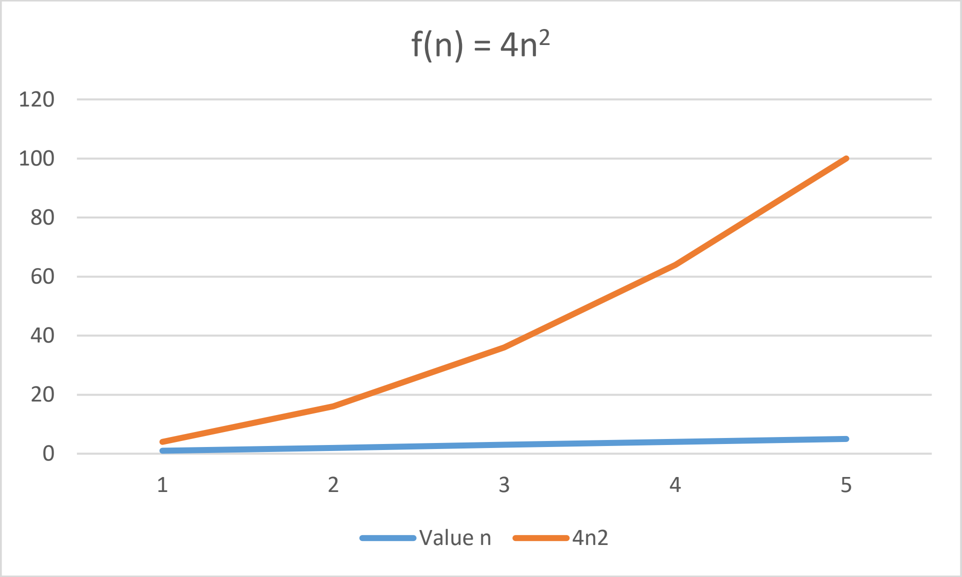

Figure 2: Graph of f(n) = 4n^2

Expression in Big O: O(n2)

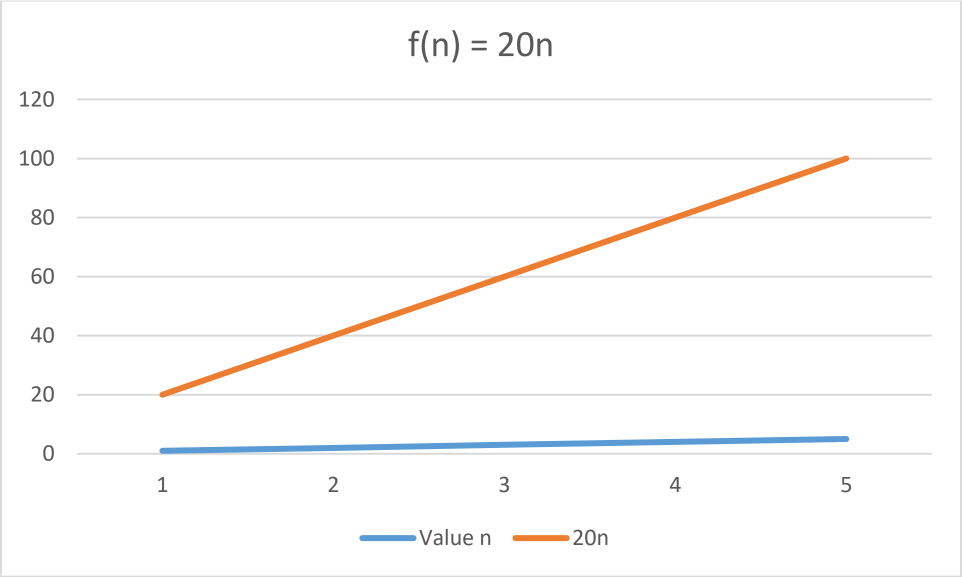

Figure 3: Graph of f(n) = 20n

Expression in Big O: O (n)

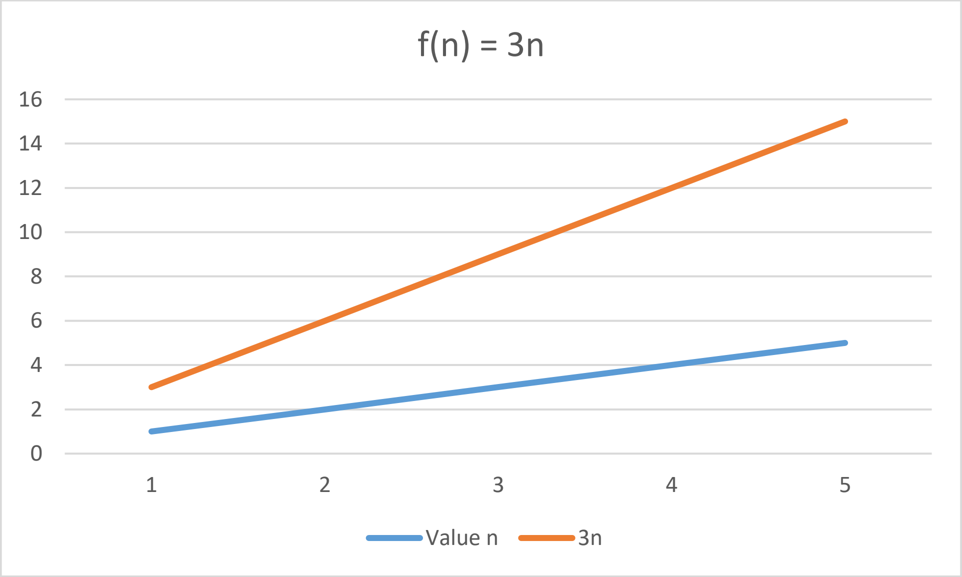

Figure 4: Graph of f(n) = 3n

Expression in Big O: O (n)

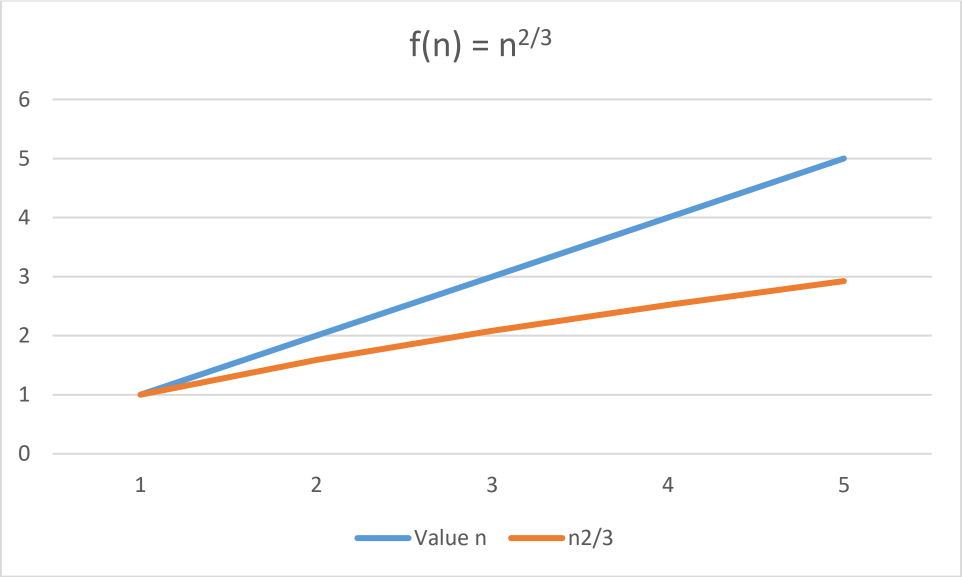

Figure 5: Graph of f(n) = $n^{2/3}$

Expression in Big O: O (n)

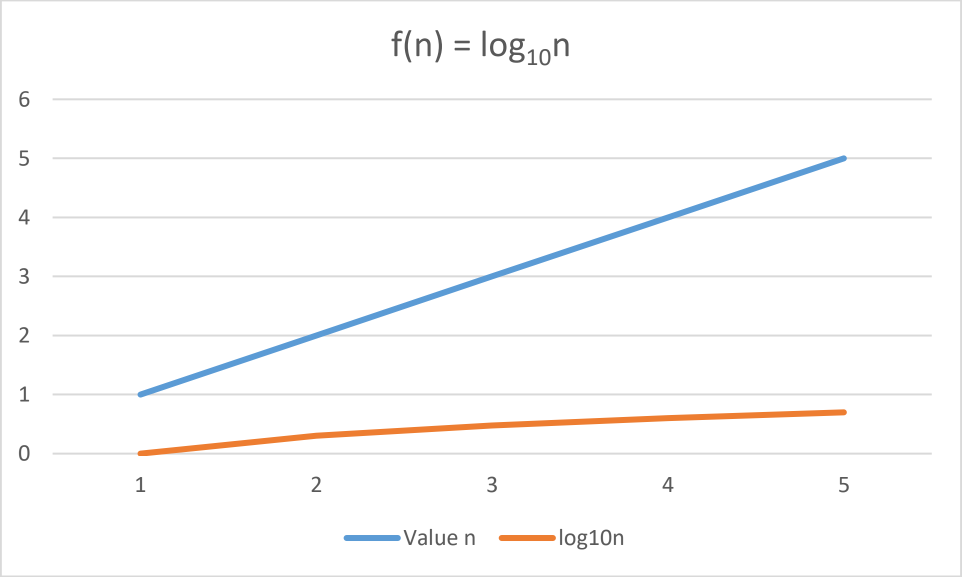

Figure 6: Graph of f(n) = $\log_{10} n$

Expression in Big O: O (log n)

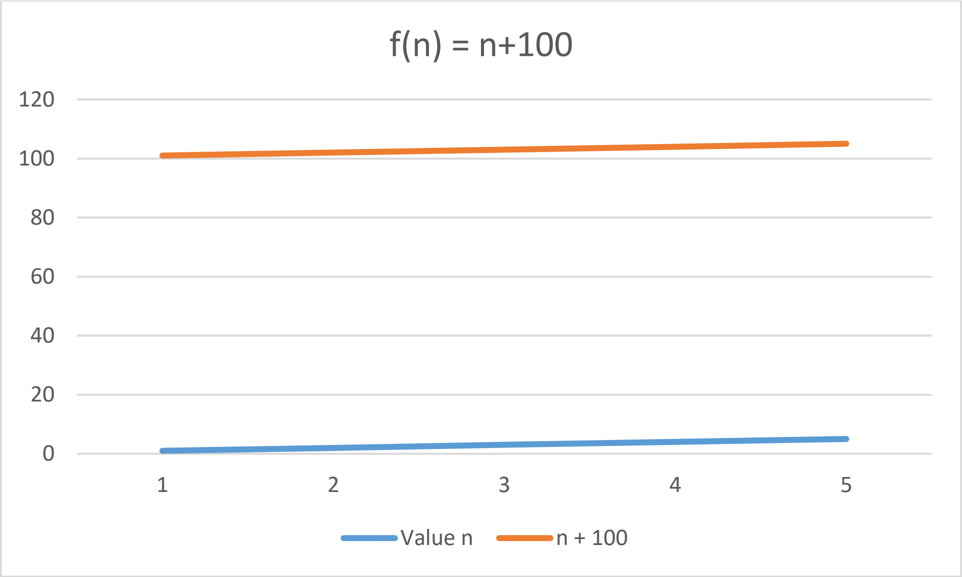

Figure 7: Graph of f(n) = n+100

Expression in Big O: O (1)

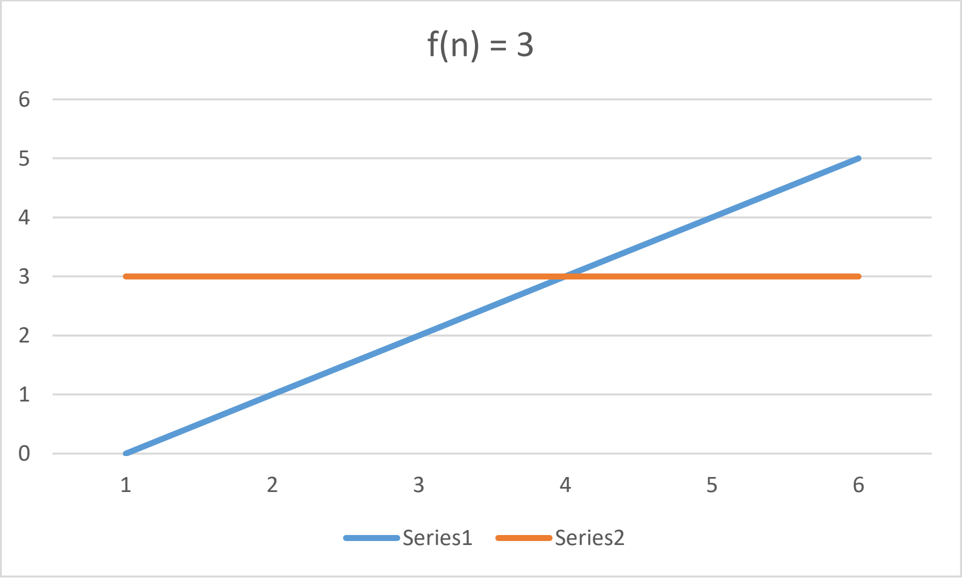

Figure 8: Graph of f(n) = 3

Expression in Big O: O (1)

Task 2

The Big O notations for the 2 algorithms are different as stated below:

Big O notation for Algorithm A: O (n)

Big O notation for Algorithm B: O (1)

This is because there is a loop function inside algorithm A, which will prolong the running time depending on the number of inputs or the value of “n”. This makes the running time to be O (n). On the other hand, algorithm B only contains an if statement, where when the worse case happens, the if statement will run. However, inside the if statement only contains a return statement which is only executed once, therefore the running time will be O (n).

Task 3

The Big O notation for the algorithm is O (n^2). This is because the algorithm contains 2 nested loop statements. The “While” statement in pseudocode represents a loop in programming language. As there is another “While” statement within a “While” statement, there is a nested loop. The Big O notation would be 2O (n^2), but constant coefficients are ignored, and algorithms with “higher growth rate” dominate others, therefore the Big O notation is O (n^2).

Task 4

The Big O notation for this algorithm is O (1). This is because there is only one if statement, and as Big O considers worst case scenario, therefore the code inside the if statement will run. However, the code inside the if statement is just a return statement. Therefore, this will only run once. So, the Big O notation would be O (1).