Drew Pulliam – DTP180003

CS3345.0U1

Assignment 2

1. N >= n0 is needed because Big-O does not work accurately for “small” values of N. This means that N must be larger than a certain value for Big-O to apply

------------------------------------------------------------------------------------------------------------------------------------------

1. Big-O notation does not care about constant multipliers, so 3N and 2N are both simply O(N)

------------------------------------------------------------------------------------------------------------------------------------------

1. F1(10) = 20, F1(20) = 40, F2(10) = 30, F2(20) = 60 --- Result doubles when N doubles

F1(10) = 100, F1(20) = 400, F2(10) = 200, F2(20) = 800 --- Result quadruples when N doubles

------------------------------------------------------------------------------------------------------------------------------------------

1. Big-O helps to define how any program will perform; it gives us the worst-case time scenario. This can be used to compare two different approaches to the same problem, the approach with the better Big-O is the faster approach.

------------------------------------------------------------------------------------------------------------------------------------------

1. N! grows faster than 2^N. This is because factorial grows faster than any exponential function.

------------------------------------------------------------------------------------------------------------------------------------------

1. A. O(n^5)

B. O(5^n)

C. O(n)

D. O(n\*log(n))

E. O(n)

------------------------------------------------------------------------------------------------------------------------------------------

1. O(N), the loop runs N times, and executes one println command each time

------------------------------------------------------------------------------------------------------------------------------------------

1. O(N^2), loop 1 runs N times, and each iteration of loop 1 contains N iterations of loop 2. N\*N = N^2

------------------------------------------------------------------------------------------------------------------------------------------

1. O(N^2), loop1 (N+1) \* loop2 (2N) = 2N^2+2N = O(N^2)

------------------------------------------------------------------------------------------------------------------------------------------

1. O(N), if num > numItems then the loop is never entered and the time is O(1), but because big-O is concerned with worst case, we only look at when the loop is entered, resulting in O(N)

------------------------------------------------------------------------------------------------------------------------------------------

1. O(log(n)), this while loop runs log(n) times, where log is base 2 because we successively divide i by 2.

------------------------------------------------------------------------------------------------------------------------------------------

1. O(log(n)), this recursion runs log(n) times, where log is base 2 because we successively divide numItems by 2.