Matthew Planchard msp2377 3 Feb 2014 Novak CS314

############## Question One ##################

Test One: Factorials of Decimals?

```
# Although taking a factorial of a decimal is a weird thing to do,
# casting to int allowed numbers with nonzero decimals to yield the
# correct answer. Answers with decimals consistently yielded odd and
# substntially divergent answers:
#
#       Factorial of 10.0        Factorial of 10.1                Factorial of 10.2
#       int: 3628800             int: 3628800                     int: 3628800
#       long: 3628800            long: 3628800                    long: 3628800
#       Integer: 3628800 Integer: 3628800         Integer: 3628800
#       float: 3628800.0 float: 482798.38         float: 1267540.8
#       double: 3628800.0        double: 482796.0133…             double: 1267542.623…
```

Test Two: When Do Answers Become Inaccurate?

```
# Tests were performed by casting ints to various forms and then running the
# provided factorial methods.
#
# The first divergence between the answers occurs for the factorial of 13.
# 13 as an int and an Integer yielded incorrect answers. It should be noted
# that the reason for this is because the answer is above the maximum possible
# value for an int.
#
# Factorials of int 13
# int: 1932053504
# long: 6227020800
# Integer: 1932053504
# float: 6.2270208E9
# double: 6.2270208E9
#
# The first divergence between the remaining types occurs at 14, where the
```

```
# decimal values for floats and doubles begin to diverge slightly. Here the
# float value is incorrect, while long and double continue to produce the
# correct result. However, although float is incorrect, the percent difference
# between float and double is only 2.5e-6 %.
#
# Factorials of int 14
# int: 1278945280
# long: 87178291200
# Integer: 1278945280
# float: 8.7178289E10
# double: 8.71782912E10
#
# Everything goes along nicely until 21. At this point, long ceases to produce
# meaningful output. The difference between the float and double answers has been
# increasing slowly. At this point it is at 4.25e-6 %.
#
# Factorials of int 21
# int: -1195114496
# long: -4249290049419214848
# Integer: -1195114496
# float: 5.109094E19
# double: 5.109094217170944E19
#
# Starting at 34, wahtever random garbage int and Integer were spitting out is
# replaced by a simple zero. Float is still relatively accurate, with a difference
# of only -7e-6 % compared to double.
#
# Factorials of int 34
# int: 0
# long: 4926277576697053184
# Integer: 0
# float: 2.9523282E38
# double: 2.9523279903960412E38
#
# Immediately thereafter, at 35, float starts to report the answer to be Infinity.
#
```

```
# Factorials of int 35
# int: 0
# long: 6399018521010896896
# Integer: 0
# float: Infinity
# double: 1.0333147966386144E40
#
# Long reports an answer of 0 from 66 onward.
#
# Factorials of int 66
# int: 0
# long: 0
# Integer: 0
# float: Infinity
# double: 5.443449390774431E92
#
# Finally, at 171, double joins the other data types in reporting the answer to be
# Infinity.
#
# Factorials of int 171
# int: 0
# long: 0
# Integer: 0
# float: Infinity
# double: Infinity
#
# Around 9000 (on my system), java begins to yield a stack overflow error. However,
# prior to that, there was no indication other than odd results that the values
# being produced were, in fact, garbage.



############## Question Two ##################

# Donald was too fast to time (max. time was 1.3e-5 s).
# Gyro was also too fist to time (max time was 7.7 e -6 s).
```

```
# All others are plotted on the graph.

Estimation of Big O Based on Graph:

#       Function:                       Big O:
#       Daffy                           nlog(n)
#       Minnie                          n^2
#       Goofy                           n^2
#       Pluto                           n
#       Mickey                          n

Ratio calculations for Mickey, Minnie, Goofy, and Pluto:

#       Function:       n:              n Ratio:Time (s):       T Ratio:Big O:
# ──────────────────────────────────────────────────────────────────────────────
#       Mickey          64k             n/a             3.60e-5         n/a
#                       128k            2               6.33e-5         1.75
#                       256k            2               1.19e-4         1.90
#                       512k            2               2.26e-4         1.90
#                       1.024m          2               4.71e-4         2.08
#                       2.048m          2               8.56e-4         1.82
#                       4.096m          2               1.65e-3         1.93
#                       8.192m          2               3.37e-3         2.04            n
#
#       Minnie          2k              n/a             1.15e-3         n/a
#                       4k              2               4.15e-3         3.61
#                       8k              2               1.51e-2         3.63
#                       16k             2               5.90e-2         3.91
#                       32k             2               2.32e-1         3.93
#                       64k             2               9.19e-1         3.96
#                       128k            2               3.65            3.97
#                       256k            2               15.5            4.25            n^2
#
#       Goofy           2k              n/a             6.22e-4         n/a
#                       4k              2               2.42e-3         3.89
#                       8k              2               9.68e-3         4.00
```

```
#                   16k             2               3.88e-2         4.01
#                   32k             2               1.62e-1         4.18
#                   64k             2               6.16e-1         3.80
#                   128k            2               2.43            3.94
#                   256k            2               10.7            4.40            n^2
#
#       Pluto       8k              n/a             5.10e-4         n/a
#                   16k             2               1.08e-3         2.12
#                   32k             2               2.26e-3         2.09
#                   64k             2               4.72e-3         2.09
#                   128k            2               9.91e-3         2.10
#                   256k            2               2.06e-2         2.08            n
```