An Empirical Study of Three Primality Algorithms
James Murphy & Guolong Luo
Team stayHome
CSCI 323 - Analysis of Algorithms
Semester: Spring 2020
Output :

BRUTE FORCE
RANDOM INTEGERS
After 10 iterations of the brute force algorithm on the data set : Random Integers, it took 4800 nanoseconds and 134 total comparisons.
After 100 iterations of the brute force algorithm on the data set : Random Integers, it took 1015800 nanoseconds and 76321 total comparisons.
After 1000 iterations of the brute force algorithm on the data set : Random Integers, it took 2968800 nanoseconds and 597357 total comparisons.
After 10000 iterations of the brute force algorithm on the data set : Random Integers, it took 14418500 nanoseconds and 5847631 total comparisons.

EVEN INTEGERS
After 10 iterations of the brute force algorithm on the data set : Even Integers, it took 37900 nanoseconds and 19 total comparisons.
After 100 iterations of the brute force algorithm on the data set : Even Integers, it took 11600 nanoseconds and 199 total comparisons.
After 1000 iterations of the brute force algorithm on the data set : Even Integers, it took 44800 nanoseconds and 1999 total comparisons.
After 10000 iterations of the brute force algorithm on the data set : Even Integers, it took 312300 nanoseconds and 19999 total comparisons.

PRIME INTEGERS
After 10 iterations of the brute force algorithm on the data set : Prime Integers, it took 1000 nanoseconds and 119 total comparisons.
After 100 iterations of the brute force algorithm on the data set : Prime Integers, it took 125100 nanoseconds and 24033 total comparisons.
After 1000 iterations of the brute force algorithm on the data set : Prime Integers, it took 9074300 nanoseconds and 3681913 total comparisons.
After 10000 iterations of the brute force algorithm on the data set : Prime Integers, it took 1135205600 nanoseconds and 496155411 total comparisons.

SIEVE OF ERATOSTHENES
RANDOM INTEGERS
After 10 iterations of the Sieve Of Eratosthenes algorithm on the data set : Random Integers, it took 929800 nanoseconds and 159 total comparisons.
After 100 iterations of the Sieve Of Eratosthenes algorithm on the data set : Random Integers, it took 2312000 nanoseconds and 1820 total comparisons.
After 1000 iterations of the Sieve Of Eratosthenes algorithm on the data set : Random Integers, it took 8588700 nanoseconds and 18357 total comparisons.
After 10000 iterations of the Sieve Of Eratosthenes algorithm on the data set : Random Integers, it took 66123800 nanoseconds and 184364 total comparisons.

EVEN INTEGERS
After 10 iterations of the Sieve Of Eratosthenes algorithm on the data set : Even Integers, it took 153700 nanoseconds and 254 total comparisons.
After 100 iterations of the Sieve Of Eratosthenes algorithm on the data set : Even Integers, it took 1352600 nanoseconds and 2467 total comparisons.
After 1000 iterations of the Sieve Of Eratosthenes algorithm on the data set : Even Integers, it took 13945700 nanoseconds and 24800 total comparisons.
After 10000 iterations of the Sieve Of Eratosthenes algorithm on the data set : Even Integers, it took 93962500 nanoseconds and 243659 total comparisons.

PRIME INTEGERS
After 10 iterations of the Sieve Of Eratosthenes algorithm on the data set : Prime Integers, it took 1400 nanoseconds and 15 total comparisons.
After 100 iterations of the Sieve Of Eratosthenes algorithm on the data set : Prime Integers, it took 26900 nanoseconds and 569 total comparisons.
After 1000 iterations of the Sieve Of Eratosthenes algorithm on the data set : Prime Integers, it took 3346600 nanoseconds and 16166 total comparisons.
After 10000 iterations of the Sieve Of Eratosthenes algorithm on the data set : Prime Integers, it took 553446700 nanoseconds and 455284 total comparisons.

FERMAT'S LITTLE THEOREM
RANDOM INTEGERS
After 10 iterations of the Fermat's Little Theorem algorithm on the data set : Random Integers, it took 87900 nanoseconds and 192 total comparisons.
After 100 iterations of the Fermat's Little Theorem algorithm on the data set : Random Integers, it took 3023800 nanoseconds and 339524 total comparisons.
After 1000 iterations of the Fermat's Little Theorem algorithm on the data set : Random Integers, it took 17698600 nanoseconds and 2511361 total comparisons.
After 10000 iterations of the Fermat's Little Theorem algorithm on the data set : Random Integers, it took 170766800 nanoseconds and 23999292 total comparisons.

EVEN INTEGERS
After 10 iterations of the Fermat's Little Theorem algorithm on the data set : Even Integers, it took 29800 nanoseconds and 211 total comparisons.
After 100 iterations of the Fermat's Little Theorem algorithm on the data set : Even Integers, it took 20300 nanoseconds and 2277 total comparisons.
After 1000 iterations of the Fermat's Little Theorem algorithm on the data set : Even Integers, it took 210900 nanoseconds and 22945 total comparisons.
After 10000 iterations of the Fermat's Little Theorem algorithm on the data set : Even Integers, it took 1723400 nanoseconds and 228514 total comparisons.

PRIME INTEGERS
After 10 iterations of the Fermat's Little Theorem algorithm on the data set : Prime Integers, it took 2300 nanoseconds and 167 total comparisons.
After 100 iterations of the Fermat's Little Theorem algorithm on the data set : Prime Integers, it took 520900 nanoseconds and 67560 total comparisons.

After 1000 iterations of the Fermat's Little Theorem algorithm on the data set : Prime Integers, it took 103521000 nanoseconds and 14673568 total comparisons.
After 10000 iterations of the Fermat's Little Theorem algorithm on the data set : Prime Integers, it took 4169466000 nanoseconds and 609110991 total comparisons.