**DISTRIBUTION OF THE NUMBER OF DIFFERENT PRIME FACTORS OF NON-PRIME NUMBERS**

*CS 501: Introduction to JAVA Programming*

Instructor: Peter Jurkat

Prepared by: Siddharth Pansuria, Daksh Bhuva and Arpit Patel

* Introduction:

This project revolves around the subject of distribution of the number of unique prime factors of non-prime numbers. For a number n, **we find all the non-prime numbers and the number of unique prime factors each of them has**. Then **we create a distribution which shows the number of numbers having *x unique prime factors* where *x can be 1, 2, 3 and so on***. Furthermore, we show a bar chart showing this distribution.

* Summary:

For this project, we have created a user interface, where a user can input a number *n*, and the program will show the bar chat of distribution of the number of unique prime factors of all the non-prime numbers smaller than or equal to n and greater than 2. There are a lot of methods to find prime numbers. For this project, we have used the ***Sieve of Eratosthenes***. So basically, what Sieve does, is that it creates and empty Boolean array equal to the size of user’s input + 1, i.e. (*n + 1)*. The goal is to modify this array such that accessing the *ith* element returns a Boolean value that shows whether the number *i* is prime. Now initially we set all the values to *true*. After that, we know that 0 and 1 are not prime, hence we set it to *false*. Now starting from 2, we check whether the current number is prime. If so, we just set all the multiples of 2 which are greater than or equal to 22 and smaller than or equal to *n* as false. However, if a given number is not prime, we just skip that number and move on to the next number. This process continues till we reach *,* since all the non-prime numbers greater than are already covered. The time complexity of this algorithm is O(n\*log(log(n))) and space complexity is O(n). We find all the primes using this and then for every non-prime number we find all its unique prime factors by starting from the smallest prime factor and if the given number is divisible by the chosen prime number, we just keep on dividing the number by the chosen prime number, unless the number is no longer divisible by the current number or the number becomes prime or the number becomes one. After this is done, we can just create the final distribution using a HashMap. Below is a simple visualization on how Sieve[(link)](https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes) works.

Table

Description automatically generated

* Outputs:

1. For n = 100:

Chart, waterfall chart

Description automatically generated

1. For n = 1000:

Chart, bar chart

Description automatically generated

1. For n = 10000:

Chart, bar chart, histogram

Description automatically generated

1. For n = 100000:

Chart, bar chart

Description automatically generated

1. For n = 1000000:

Chart, bar chart

Description automatically generated

* Conclusion:

Chart, line chart

Description automatically generated

We plotted a line graph comparing all the outputs and the result is show below. The graph starts to flatten on the borders, and it starts to smoothen as the n increases. Since we do not have enough space to calculate for larger n values, we cannot say what happens, but as we can clearly see, as n increases the distribution starts becoming more and more uniform. Hence we can say that, as n will increase, the percentage of numbers with lesser prime factors will keep decreasing while the percentage of numbers with more prime factors starts to increase slowly and then after a point it will achieve a maxima and then start to decrease.