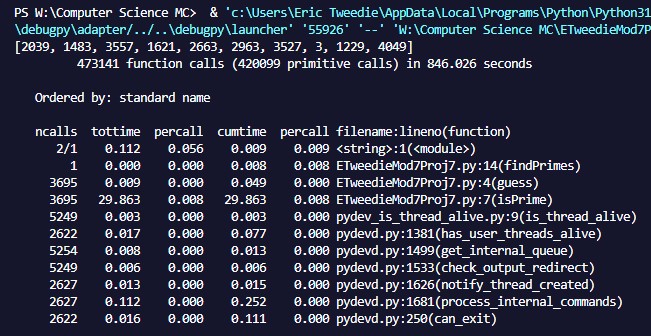
Eric Tweedie

CSC 6301 Module 7 Project 7

Code analysis report

Version 1:

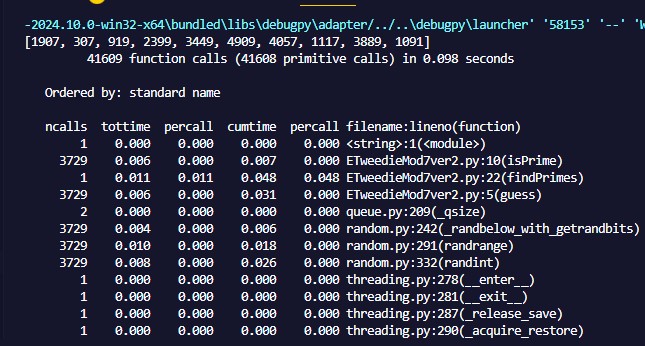




The first version of the code was finding the prime numbers within a random range and then checking every option possible within that range by seeing if a number within that range multiplied by another number equals the number used for the range. It is not a very efficient method because it will take longer when the range is a large number. If time is not a concern, then sure it will be an effective method to use since it will give the answer. The function that is causing the slow running time is the isPrime() function because it is iterating over every element in the range twice with the 2 loops being used to see if any of those elements multiplied together give you the number x to decide if it is a prime or not. The inefficiency of this function is mainly because it is having to iterate over every element twice and then check to see if the multiplication is equal to the number used for the range.

Version 2:





Version 2 of the code to find the prime numbers has been made more optimal and is running faster than the original version of the code. What has been done to make it more efficient is having the isPrime() function check up to the square root of x so as not to check a greater number twice and skipping even numbers except 2 since all even numbers greater than 2 are not prime numbers. This modification of the code will cut down on the total amount of numbers the function will have to check in order to determine if it is a prime number. The 2nd version allows for a faster check of the random number being checked for what is prime, and is more efficient over all because of it. Viewing the cProfile output it is showing a significant decrease in the amount of time the function is taking overall all the calls being done on it. Where as in the first version the function was taking much longer due to how many checks the function has to do in order to determine what is a prime number with all the multiplication.