Software Engineering Lab – Profiling In Visual Studio

I have written methods for 3 separate algorithms that return the remainder of integer division (or modulo) of two unsigned integers. They are...

- Modulo using recursion, **modR()**.
- Modulo using the bitwise solution from Hacker's Delight, **modB()**.
- Modulo using arithmetic with C#'s built in division operator, **modD()**.
- and for consistency a fourth that uses C#'s built in modulus operator, **modO()**.

*Compilers are smart enough these days that their built-in operators include our bitwise optimization, along with many others.

Your task is to profile these methods.
Provide a meaningful comparison of their relative performances using Visual Studio's (or your preferred ide's) built in cpu usage tools.

Memory does not vary in any real way. Focus on cpu usage (expressed in time) under different conditions, and relative to one another. Think of it as an experiment that you must design, and then take and report the data over multiple trials and under different conditions.

You will deliver a short report, 1-2 pages, consisting of screen shots, descriptive statistics, and short summaries of what you see, as well as a short description of the code that you chose to write for the conditions which you chose to characterize these algorithms. This is open-ended. You will have to make decisions about what to do and what is important to share.

The project and Visual Studio 19 solution (HD_Warren_(403Final)) with all code in Program.cs is available at https://bit.ly/35kLLP5.
I recommend VS, but am agnostic to IDE, though you must perform this work and backup your observations using a tool that observes the heap and call stack, and profiles performance in real time. I have included some notes on the VS profiler below as well as a walkthrough video on Moodle.

Some VS profile tool tips...

- Ctrl+Alt+F2, or Debug-> Windows-> Show Diagnostic Tools.

- You need to break the code somewhere before the block you want to profile in order to verify the tool is recording the cpu profile when that code executes.

- Step over calls to the methods you want to profile while cpu profiling is being recorded.

- Use the timeline (the top graph in the window) to set bookends on the time you want to view. You can zoom in/zoom out. Take snapshots to place markers on the timeline, and use them to help you set your view ranges. Then click Open Details.

- The profiler in VS records and reports on calling functions (eg. Program.main), so you'll need to look a little closer to see the performance of the specific methods you call.

- "Load symbols" in function view is your best friend. (the most helpful tip by far)

There are two learning objectives --

(1) Spend some time in a profiling tool, clanking around and learning to set ranges and look at performance.

(2) Learn about the inconsistent results that profiling provides, and design an experiment to account for that. It is taking samples once every ms, and if a method is running it gets recorded. It is not a deterministic exercise, and it is normal to get inconsistent results -- especially at small scales. If a method executes between samples, for example, it may not be recorded at all. Consider writing separate calling functions outside of main to tuck any iteration you might want to add into one place.

Think of the exercise more like a science experiment (data collected over multiple trials under changing real world conditions) than a program (executed once in a deterministic environment), and that should help you plan your investigation and frame your report.