CSCI 654-01—Foundations of Parallel Computing
Programming Project 3

Prof. Alan Kaminsky—Fall Semester 2018
Rochester Institute of Technology—Department of Computer Science

Overview

Write a GPU parallel program, with the main program in Java using the Parallel Java 2 Library and the computational kernel in C using CUDA. Run the program on a GPU accelerated computer to learn about GPU parallel programming.

Help with your project: I am willing to help you with the design of your project. I am willing to help you debug your project if the code isn't working. However, for help with design or debugging issues *you must come see me in person.* Either visit me during office hours or make an appointment. I will not help you with design or debugging issues via email. If it's the evening of the project deadline and I have gone home, you are on your own. Plan and work ahead so there will be plenty of time for me to help you if necessary.

Breaking RSA

Here is a simplified description of the RSA public key encryption algorithm. The public encryption key consists of two integers (e, n), where e is an exponent and n is a modulus. For this project, $e = 3$ and n is either a prime or the product of two primes. To encrypt a plaintext message m, where m is an integer in the range 0 through $n-1$, compute the following formula yielding the ciphertext c, where c is also an integer in the range 0 through $n-1$:

$$c = m^3 \pmod{n}$$

To decrypt the ciphertext, normally you have to know the private decryption key that corresponds to the public encryption key. However, you can also decrypt the ciphertext without knowing the private key by computing a modular cube root:

$$m = c^{1/3} \pmod{n}$$

If n is a large integer, e.g. a 2048-bit integer, no one knows an efficient algorithm to compute a modular cube root. But if n is small, it becomes possible to compute the modular cube root by brute force search: Given a ciphertext c, compute $m^3 \pmod{n}$ for every m, $0 \le m \le n-1$, until the answer equals c.

Depending on the particular values of c and n, c might have anywhere from zero to three modular cube roots—that is, zero to three values of m such that $c = m^3 \pmod{n}$.

You will write a program to compute modular cube root(s) in parallel on the GPU using a brute force search.

Design Hints

1. When computing the formula $m^3 \pmod{n}$, use 64-bit integers (type `long` in Java). If you use 32-bit integers (type `int` in Java), the calculation might overflow and yield the wrong result.
2. When computing the formula $m^3 \pmod{n}$, do the $\pmod{n}$ after every operation. If you do the $\pmod{n}$ just once at the end, the calculation might overflow and yield the wrong result.
3. Study the CUDA documentation on atomic functions and consider which of these you might be able to use.

Program Input and Output

The program's command line arguments are the ciphertext c and the modulus n.

The program finds ***all*** the cube roots of $c \pmod{n}$.

If there are no cube roots, the program prints one line as shown in the examples below. If there are one or more cube roots, the program prints one line for each cube root as shown in the examples below; the cube roots are printed in ascending order.

```
$ java pj2 ModCubeRoot 2 1033
No cube roots of 2 (mod 1033)
$ java pj2 ModCubeRoot 2 1031
798^3 = 2 (mod 1031)
$ java pj2 ModCubeRoot 6 1033
56^3 = 6 (mod 1033)
387^3 = 6 (mod 1033)
```

```
590^3 = 6 (mod 1033)
$ java pj2 ModCubeRoot 43781972 124822021
142857^3 = 43781972 (mod 124822021)
$ java pj2 ModCubeRoot 46054145 124822069
142857^3 = 46054145 (mod 124822069)
27549958^3 = 46054145 (mod 124822069)
97129254^3 = 46054145 (mod 124822069)
```

Compiling and Running Your Program

Your Java main program must be in a file named `ModCubeRoot.java`. Your C/CUDA kernel must be in a file named `ModCubeRoot.cu`. To compile and run your program:

1. Log into the `kraken` computer.
2. Set the `CLASSPATH`, `PATH`, and `LD_LIBRARY_PATH` variables as described here.
3. Compile the Java main program using this command:

4.
   ```
   $ javac ModCubeRoot.java
   ```
5. Compile the CUDA kernel using this command:

6.
   ```
   $ nvcc -ptx --compiler-bindir=/usr/bin/gcc-4.8 -arch compute_20 -o
   ModCubeRoot.ptx ModCubeRoot.cu
   ```
7. Run the program using this command (substituting the proper command line arguments):

8.
   ```
   $ java pj2 ModCubeRoot <c> <n>
   ```

Note: The compiled CUDA module is stored in a `.ptx` file, not a `.cubin` file like the examples in the textbook.

Software Requirements

1. The program must be run by typing this command line:

2.
   ```
   java pj2 ModCubeRoot <c> <n>
   ```
   a. `<c>` is the number whose modular cube root(s) are to be found; it must be a decimal integer (type `int`) in the range $0 \le c \le n{-}1$.
   b. `<n>` is the modulus; it must be a decimal integer (type `int`) ≥ 2.

   Note: This means that the program's class must be named `ModCubeRoot`, this class must not be in a package, and this class must extend class edu.rit.pj2.Task.

   Note: You can assume that *n* is either a prime or the product of two primes. You do not have to verify this.

3. If the command line does not have the required number of arguments, if any argument is erroneous, or if any other error occurs, the program must print an error message on the console and must exit. The error message must describe the problem. The wording of the error message is up to you.

4. If there are no cube roots of *c* (mod *n*), the program must print the following line on the console:
5. `No cube roots of <c> (mod <p>)`

   where `<c>` is replaced with the value of *c* and `<p>` is replaced with the value of *p*. There must be no spaces at the beginning or end of the line. The line must be terminated with a newline.

6. If there are one or more cube roots of *c* (mod *n*), the program must print the following line on the console for each cube root *m*:
7. `<m>^3 = <c> (mod <p>)`

   where `<m>` is replaced with the value of *m*, `<c>` is replaced with the value of *c*, and `<p>` is replaced with the value of *p*. There must be no spaces at the beginning or end of the line. The line must be terminated with a newline. The lines must be printed in ascending order of *m*.

   Note: If your program's output does not conform ***exactly*** *to Software Requirements 1 through 4,* ***you will lose credit*** *on your project.* See the Grading Criteria below.

Software Design Criteria

1. The modular cube root computations must be performed in parallel on one GPU.
2. The program must follow the GPU parallel programming patterns studied in class.
3. The program must be designed using object oriented design principles as appropriate.
4. The program must make use of reusable software components as appropriate.
5. Each class or interface must include a Javadoc comment describing the overall class or interface.
6. Each constructor and method within each class or interface must include a Javadoc comment describing the overall constructor or method, the arguments if any, the return value if any, and the exceptions thrown if any.

   Note: See my Java source files which we studied in class for the style of Javadoc comments I'm looking for.

   Note: If your program's design does not conform to Software Design Criteria 1 through 6, ***you will lose credit*** *on your project.* See the Grading Criteria below.

Submission Requirements

Your project submission will consist of a ZIP file containing the Java source file for every class and interface in your project, as well as the `ModCubeRoot.cu` source file. Put all the source files into a ZIP file named `"<username>.zip"`, replacing `<username>` with the user name from your Computer Science Department account. On a Linux system, the command for creating a ZIP file is:

```
zip <username>.zip *.java ModCubeRoot.cu
```

DO NOT include the Parallel Java 2 Library in your ZIP file!

Send your ZIP file to me by email at ark@cs.rit.edu. Include your full name and your computer account name in the email message, and include the ZIP file as an attachment.

When I get your email message, I will extract the contents of your ZIP file into a directory. I will set my Java class path to include the directory where I extracted your files, plus the Parallel Java 2 Library. I will compile all the Java source files using the JDK 1.8 compiler. I will compile the `ModCubeRoot.cu` source file with the CUDA compiler. I will then send you a reply message acknowledging I received your project and stating whether I was able to compile all the source files. If you have not received a reply within one business day (i.e., not counting weekends), please contact me. Your project is not successfully submitted until I have sent you an acknowledgment stating I was able to compile all the source files.

The submission deadline is Monday, November 12, 2018 at 11:59pm. The date/time at which your email message arrives in my inbox will determine whether your project meets the deadline.

You may submit your project multiple times up until the deadline. I will keep and grade only the most recent successful submission. There is no penalty for multiple submissions.

If you submit your project before the deadline, but I do not accept it (e.g. I can't compile all the source files), and you cannot or do not submit your project again before the deadline, the project will be late (see below). ***I STRONGLY advise you to submit the project several days BEFORE the deadline, so there will be time to deal with any problems that may arise in the submission process.***

Grading Criteria

I will grade your project by:

- (10 points) Evaluating the design of your program, as documented in the Javadoc and as implemented in the source code.
  - All of the Software Design Criteria are fully met: 10 points.
  - Some of the Software Design Criteria are not fully met: 0 points.
- (20 points) Running your program. There will be twenty test cases, each worth 1 point. For each test case, if the program runs using the command line in Requirement 1 and the program produces the correct output, the test case will get 1 point, otherwise the test case

will get 0 points. "Correct output" means "output fulfills *all* the Software Requirements *exactly*."
- (30 points) Total.

When I run your program, the Java class path will point to the directory with your compiled class files, plus the Parallel Java 2 Library. I will use JDK 1.8 to run your program.

I will grade the test cases based *solely* on whether your program produces the correct output as specified in the above Software Requirements. *Any* deviation from the requirements will result in a grade of 0 for the test case. This includes errors in the formatting (such as extra spaces), output lines not terminated with a newline, and extraneous output not called for in the requirements. The requirements state *exactly* what the output is supposed to be, and there is no excuse for outputting anything different. If any requirement is unclear, please ask for clarification.

If there is a defect in your program and that same defect causes multiple test cases to fail, I will deduct points for *every* failing test case. The number of points deducted does *not* depend on the size of the defect; I will deduct the same number of points whether the defect is 1 line, 10 lines, 100 lines, or whatever.

After grading your project I will put your grade and any comments I have in your encrypted grade file. For further information, see the Course Grading and Policies and the Encrypted Grades.

Test Cases

The grading test case commands and correct printouts were as follows. For test cases 1–3, the wording of the error message was up to you, as long as it described the problem.

IMPORTANT NOTE
Running on the `kraken` machine, each test case 4–20 should have taken about 2 seconds. I set a 10-second timeout on each test case. If the program timed out, it indicates a bad design.

IMPORTANT NOTE
I have accurately recorded in your grade file what I observed during my testing. If you run the test cases and do not get the same results as I, then there is a flaw in your design somewhere that causes inconsistent results (perhaps due to incorrect parallelization), and I am not going to change your grade for the original submission. You may still resubmit your project.

```
1.  $ java pj2 timelimit=10 ModCubeRoot
    Usage: java pj2 ModCubeRoot <a> <p>
2.  $ java pj2 timelimit=10 ModCubeRoot 2 xyz
    ModCubeRoot: p must be an integer
    Usage: java pj2 ModCubeRoot <a> <p>
3.  $ java pj2 timelimit=10 ModCubeRoot 1032 1031
    ModCubeRoot: a must be >= 0 and < p
    Usage: java pj2 ModCubeRoot <a> <p>
4.  $ java pj2 timelimit=10 ModCubeRoot 12345 60139
    No cube roots of 12345 (mod 60139)
```

```
5.  $ java pj2 timelimit=10 ModCubeRoot 35308 60139
    432^3 = 35308 (mod 60139)
    20835^3 = 35308 (mod 60139)
    38872^3 = 35308 (mod 60139)
6.  $ java pj2 timelimit=10 ModCubeRoot 0 60139
    0^3 = 0 (mod 60139)
7.  $ java pj2 timelimit=10 ModCubeRoot 12345 122636057
    27224487^3 = 12345 (mod 122636057)
    64935236^3 = 12345 (mod 122636057)
    83241604^3 = 12345 (mod 122636057)
8.  $ java pj2 timelimit=10 ModCubeRoot 9227 122636057
    No cube roots of 9227 (mod 122636057)
9.  $ java pj2 timelimit=10 ModCubeRoot 80621568 122636057
    432^3 = 80621568 (mod 122636057)
    1163034^3 = 80621568 (mod 122636057)
    95887416^3 = 80621568 (mod 122636057)
10. $ java pj2 timelimit=10 ModCubeRoot 12345 233549
    205676^3 = 12345 (mod 233549)
11. $ java pj2 timelimit=10 ModCubeRoot 47165 233549
    192459^3 = 47165 (mod 233549)
12. $ java pj2 timelimit=10 ModCubeRoot 1 233549
    1^3 = 1 (mod 233549)
13. $ java pj2 timelimit=10 ModCubeRoot 12345 581289391
    314279690^3 = 12345 (mod 581289391)
14. $ java pj2 timelimit=10 ModCubeRoot 12345 646121461
    84076831^3 = 12345 (mod 646121461)
    155049960^3 = 12345 (mod 646121461)
    406994670^3 = 12345 (mod 646121461)
15. $ java pj2 timelimit=10 ModCubeRoot 12347 646121461
    No cube roots of 12347 (mod 646121461)
16. $ java pj2 timelimit=10 ModCubeRoot 12345 6791167
    No cube roots of 12345 (mod 6791167)
17. $ java pj2 timelimit=10 ModCubeRoot 12345 510226727
    356821774^3 = 12345 (mod 510226727)
18. $ java pj2 timelimit=10 ModCubeRoot 123 3547
    No cube roots of 123 (mod 3547)
19. $ java pj2 timelimit=10 ModCubeRoot 1140 3547
    234^3 = 1140 (mod 3547)
    977^3 = 1140 (mod 3547)
    2336^3 = 1140 (mod 3547)
20. $ java pj2 timelimit=10 ModCubeRoot 0 3547
    0^3 = 0 (mod 3547)
```

Late Projects

If I have not received a successful submission of your project by the deadline, your project will be late and will receive a grade of zero. You may request an extension for the project. There is no penalty for an extension. See the Course Policies for my policy on extensions.

Plagiarism

Programming Project 3 must be entirely your own individual work. I will not tolerate plagiarism. If in my judgment the project is not entirely your own work, you will automatically receive, as a minimum, a grade of zero for the assignment. See the Course Policies for my policy on plagiarism.

Resubmission

If you so choose, you may submit a revised version of your project after you have received the grade for the original version. However, if the original project was not successfully submitted by the (possibly extended) deadline or was not entirely your own work (i.e., plagiarized), you are not allowed to submit a revised version. Submit the revised version via email in the same way as the original version. I will accept a resubmission up until 11:59pm Monday 26-Nov-2018. You may resubmit your project multiple times up until the deadline; I will keep and grade only the most recent successful resubmission; there is no penalty for multiple resubmissions. I will grade the revised version using the same criteria as the original version, then I will subtract 3 points (10% of the maximum possible points) as a resubmission penalty. The revised grade will replace the original grade, even if the revised grade is less than the original grade.