Reduce Exercise
(25% of total marks)

Notes by James H. Davenport

4 September 2018

1 Aim

The aim of this exercise is to implement various g.c.d. algorithms in Reduce. You are writing in ordinary Reduce (*not* the "symbolic mode described in Chapter 17 of [HS18]. You should implement functions for g.c.d. of polynomials in several variables, using, recursively, these algorithms as defined in Lecture 2. Note that these algorithms (slide 11 etc.) fulfil the role of "Some PseudoEuclid" in the Gauss algorithm on slide 8. To do g.c.d. of polynomials in several variables (say x, y, z) you let them be in $\mathbf{Z}[z][y][x]$ and use the Gauss algorithm in $R[x]$, where $R = \mathbf{Z}[z][y]$ etc. A lot of this infrastructure will be common to all implementations, and I expect you to cut and paste it from one file to the next.

2 Research Questions

Reduce has two g.c.d. algorithms already programmed:

Native This is almost the algorithm described as "Full" in [Hea79];

EZGCD This is the [MN81] version of the [MY73] algorithm, obtained after you say `on ezgcd`. See [HS18, p. 131].

So comparing your implementations with the built-in ones, using the sort of polynomials described in the appendix of [Hea79] gives these questions.

RQ1 How much faster is Reduce's built-in implementation than your `HearnFull`?

RQ2 How do your five implementations compare with each other?

3 Submission

You must submit, by e-mail to `J.H.Davenport@bath.ac.uk` by 18:00 on Saturday 8 September, a single zip file, called 3160567890.zip (if your student ID is 3160567890 - use YOUR OWN student number) containing these files.

Primgcd.red Implements a function `Primgcd` to take the g.c.d. of two polynomials using the algorithm on slide 11. **Marks: 10%**.

HearnBasic.red Implements a function `HearnBasicgcd` to take the g.c.d. of two polynomials using the algorithm on slide 15. **Marks: 10%**.

HearnFull.red Implements a function `HearnFullgcd` to take the g.c.d. of two polynomials using the algorithm called "Hearn Primitive" on slide 16. **Marks: 10%**.

Subresgcd.red Implements a function `Subresgcd` to take the g.c.d. of two polynomials using the algorithm on slide 13. **Marks: 10%**.

JHD.red Implements a function `JHDgcd` to take the g.c.d. of two polynomials using the ideas on slide 16. I haven't given you an algorithm — you have to work out what you think "better management of l" could mean. **Marks: 20%**.

Report.pdf A report with your investigation of the research questions stated above. **Marks: 40%**.

4 A note on Testing

I will test your problems by running a test script against them. Obviously, I can't give you a complete script, but the file at `http://staff.bath.ac.uk/masjhd/Zhejiang/PrimgcdTests.red` gives you an example.

For the Reduce functions, at least 25% of the marks will be for problems in one variable only, called x, and at least 25% for problems in two variables only, called x and y. So even if your solutions are not perfect, you will still get marks. In particular, you will get 40% if you can pass this test script.

5 Questions and Answers

There will undoubtedly be questions, and if they are general ones, I will put the questions and answers here. If it seems important, I will tell QQ that I have updated this file on my website `http://staff.bath.ac.uk/masjhd/Zhejiang`.

Q1 So the testfile means that we need to suubmit various functionswhich should be declated in format like PrimGCD(A,B).

A1 Yes. And that function, PrimGCD, will probably do some sorting out of the variables, choosing a main variable x and therefore all other variables would be in R, then be an implementation of the Algorithm I called Gauss, which would call PrimitiveGCD as its pseudoEuclid *and* would call itself recursively for the g.c.d. in R implied in Gauss. **Note that you are not allowed to use Reduce's own gcd for this, either directly or via Reduce's 'content' function**.

Q2 It says “comparing” — does that mean time?

A2 Yes. The easiest was to get the time in Reduce is to say `on time`; Then it prints the time, then after each commnd how long it took, and how much extra system overhead time there was (it’s called GC time, but includes the time takes to load a package, so is often greater the first time you try something).

Q3 So our function has to work in one or two variables?

A3 Or more (but you do get some marks if it only works in one or two). Note the `MAINVAR` function in Reduce will tell you the main variable of an expression.

Q4 My Hearn seems to be returning quite large numbers: larger than SubRes.

A4 That’s possible, but you may have missed the clarification note to Slide 15 of Lecture 2 (new!).

Q5 Is JHD based on HearnBasic or HearnFull?

A5 Good question. Could be either. But since the theory is that JHD cancels more than HearnBasic, it’s probably better to base it on HearnBasic.

Q6 For JHD, is it smallest size or fastest running time that you want?

A6 The two tend to be connected. But given a choice, fastest running time on big examples is what we want. You should describe your choice in the comments, or in the report if it is long.

References

[Hea79] A.C. Hearn. Non-Modular Computation of Polynomial Gcd Using Trial Division. In *Proceedings EUROSAM 79*, pages 227–239, 1979.

[HS18] A.C. Hearn and R. Schöpf. REDUCE User’s Manual (Free Version; June 8, 2018). `http://reduce-algebra.sourceforge.net/`, 2018.

[MN81] P.M.A. Moore and A.C. Norman. Implementing a Polynomial Factorization and GCD Package. In *Proceedings SYMSAC 81*, pages 109–116, 1981.

[MY73] J. Moses and D.Y.Y. Yun. The EZ GCD Algorithm. In *Proceedings ACM 73*, pages 159–166, 1973.