Factoring Algorithms over Finite Fields

Details:
Email: r.howell-peak@warwick.ac.uk
FLINT website: http://flintlib.org/
Supervisor: Bill Hart

THE UNIVERSITY OF WARWICK

URSS

Introduction:

This project was part of the FLINT – Fast Library for Number Theory – project which is a highly optimised C library used for doing number theory calculations. The goal of my summer project was to implement a factoring algorithm for polynomials with coefficients from a finite field. This involves splitting the polynomial into irreducible factors, the product of which is the polynomial.
For example: $x^2+3x+2=(x+1)(x+2)$ is a polynomial factorisation.
Currently the Magma software is believed to be the fastest in the world for this so it serves as the benchmark for most of the code written.

The Algorithms:

The algorithms are not new, but this is the first time anything like this has been implemented in FLINT. The work done this summer will be built upon in the future to do factorisation over **Z** and **Q.**

Irreducibility test – (Rabin)
I did this one first since it was relatively easy and also very useful, since it can be used to determine whether a factorisation was successful or not.

Square-Free – (Musser)
Most factoring algorithms require a square-free input which simplifies the factoring greatly. It is also a relatively cheap algorithm and partially factors the polynomial speeding up the overall factoring time.

Factoring – (Berlekamp) Taking a square-free input, this algorithm factors the polynomial by finding congruences of $g^p=g \bmod f$. It can be shown that for such a g,

$$f=\prod_{a\in F_p} gcd(f, g-a)$$

Finding g involves computing the kernel of the map $a\to a^p$ by putting the coefficients in a matrix and using Gaussian elimination to find a basis of the kernel. It is then a matter of randomly combining elements of the basis to see if we get a non-trivial factor, then using recursion on this factor and the quotient to completely factor the polynomial.

Testing – The first test was simply to multiply the factors together and check if it was the original polynomial. This would ensure that the factorisation was correct. Secondly I checked the factors using the irreducibility test to ensure it was a complete factorisation.

Profiling – This involved timing the implementation of the algorithm, and comparing it to other implementations in different software packages.

Results:

These are the results of the benchmarking tests done against Magma, I plotted average time against length of polynomial. There are some anomalous results where maybe there was a particularly difficult polynomial to factorise, but overall the FLINT implementation was around 1.5 – 2 times faster than Magma