Deterministic Polynomial Factorisation Over a Finite Field

Kundan Kumar
Under the mentorship of *Dr. Nitin Saxena*

November 14, 2014

1 Abstract

The problem of univariate polynomial factorization is known to have a number of polynomial time randomized algorithms such as Berlekamp, Rabin, Cantor & Zassenhaus, von zur Gathen & Shoup, Kaltofen, etc. But, till date, there is no known polynomial time deterministic algorithm for factoring a general uni-variate polynomial over a finite field. In this project, we present a deterministic algorithm to find a factor of a univariate polynomial over a finite field.The algorithm has been implemented on MuPAD (MATLAB mathematics tool) and tested for thousands of primes. We present various data to support that this algorithm factors polynomial in deterministic time.

2 Introduction

The problem can be formally defined in the following manner. Here, F_q represents finite field of size q.

Definition 1 (Formal definition of the problem)**.** *Given a monic univariate polynomial* $f \in F_q[x]$*, find the complete factorization* $f = f_1^{e1} f_2^{e_2} .....f_k^{e_k}$ *where* f_i*'s are distinct monic polynomials* $\in F_q$ *and* e_i*'s are postive integers.*

3 Pre-Requisites

Lemma 1. *There exist efficient algorithms for square-free factorisation(SFF) and distinct degree factorisation(DDF) which reduces the problem into factorisation of polynomial which only has square-free equi-degree irreducible factors. [3]*

Lemma 2. *Factorisation over* F_q *can be reduced efficiently to* F_p *using Berlecamp's algorithm [1] where* $q = p^m$*,* p *is prime and* m *is a positive integer.*

Lemma 3. *It's enough to have an algorithm for factoring square-free equi-degree polynomials over* F_p *to factor a general polynomials over* F_q*.*

Proof. This follows from lemma 1 and 2. □

Lemma 4. *Probability that a randomly chosen non-zero element in F_p is quadratic residue (or a quadratic non-residue) is nearly $\frac{1}{2}$. Along with this, all quadratic residues are root of the equation*

$$x^{\frac{p-1}{2}} - 1 \equiv 0 \tag{1}$$

while all non-residues are root of the equation.

$$x^{\frac{p-1}{2}} + 1 \equiv 0 \tag{2}$$

Proof. Since F_q* is a cyclic group, with generator (let's say) g, then all odd powers of g are squares and others are non-squares. Hence, half of the elements are square and half are not. Thus, probability of finding a quadratic residue is nearly half (if we include 0).

All elements of F_q* are root of the equation

$$x^{q-1} - 1 \equiv 0 \tag{3}$$

Also, the following equations hold in F_q

$$x^{q-1} - 1 = (x^{\frac{q-1}{2}} + 1) * (x^{\frac{q-1}{2}} - 1) \tag{4}$$

$$(g^{2i})^{\frac{q-1}{2}} = 1 \tag{5}$$

$$(g^{2i+1})^{\frac{q-1}{2}} = -1 \tag{6}$$

Hence, the lemma follows. □

Lemma 5. *If h is an irreducible of degree d over F_p, then h divides $x^{p^d} - x$.*

Proof. If h is irreducible, then $F_p/ < h >$ is a field isomorphic to F_{p^d}. Hence, the following equation holds

$$x^{p^d} - x \equiv 0 \pmod{h} \tag{7}$$

Thus, h divides $x^{p^d} - x$ over F_p □

3.1 Cantor & Zassenhaus randomized algorithm(CZ algorithm) [2]

Cantor & Zassenhaus presented a polynomial time randomized algorithm for factoring polynomial over a finite field. They shift the formal variable in a polynomial by random value (say α) in F_p enabling roots to behave randomly with respect to their quadratic residuosity. Using lemma 4, we can use following equations to seperate factors containing quadratic residues and non-residues.

$$f_1^i = gcd(f, x^{\frac{p^i-1}{2}} - 1) f_2^i = gcd(f, x^{\frac{p^i-1}{2}} + 1) \tag{8}$$

where f_1^i will be product of ith degree factors of f corresponding to quadratic residues and f_2^i is corresponding polynomial for quadratic non-residues. Using lemma 5, we can know the degree of factors (which are equal from our assumption). Then, suppose that degree is i. Then chance of failure of this algorithm is nearly $\frac{1}{2}$ using lemma **??**. Therefore, using this algorithm repeatedly choosing α randomly each time reduces the chances of failure to negligible value.

4 Basic Approach

We'll modify Cantor & Zassenhaus randomized algorithm to make it deterministic. The polynomial used as a test case in this project is $f(x) = x^2 - a$ where $a \in F_p^2$.

Claim 1. *If we have a CZ like algorithm for factoring p, then that can be extended easily to factor general univariates over F_p.*

Proof. This follows from the observation that the only difference in applying CZ like algorithm for higher degree case is that we take use

$$gcd(f, x^{\frac{p^i-1}{2}} - 1) \tag{9}$$

instead of

$$gcd(f, x^{\frac{p-1}{2}} - 1) \tag{10}$$

□

Using the above claim, we can say that it suffices to give an algorithm for factoring f over F_p to have an algorithm for factoring general polynomials over F_q.

Conjecture 1. *The distribution of roots of unity over F_q and distribution of quadratic residues are independent i.e. roots of unity behaves as nearly random elements for qudratic residues.*

Lemma 6. *If $ord_r(p) = r - 1$, then $h = \frac{y^r-1}{y-1}$ is irreducible i.e. $F_p / < h >$ is a field of size p^{r-1}.*

Conjecture 2. *Using the conjecture 1, we give an algorithm to factor f over F_p.Shift the polynomial by a root of unity and hope that this shift would change the quadratic residuosity of the roots of newly obtained polynomial i.e. we now look at factoring $p(x - y)$. We do this shifting by doing all the computation in ring $R \equiv$ polynomial ring over $F_p[y]/ < y^r - 1 >$.*

4.1 Algorithm

Input : f and p

Output : Factors of f

Step 1 : $r = 2$;
Step 2 : $h = \frac{y^r - 1}{y-1}$; $R = F_p[y]/ < h >$;
Step 3 : Apply CZ algorithm with modification that we shift by y (done implicitly by doing all computation in R) instead of α
Step 4 : If we get factors, shift them back by y and return them.
Step 5 : If step 3 fails, then $r = r + 1$. Go to step 2 and repeat.

Conjecture 3. *The above algorithms returns factors in $O(log(p))$ iterations.*

5 Observations

Observation 1. *We observed that due to flipping of residuosity of roots over different component of the ring R[x], we were not getting absolute factors i.e. we got factors of the kind*

$$x - q(y) \tag{11}$$

where q is a polynomial in y and y is a non-trivial root of the equation

$$y^r - 1 \equiv 0 \tag{12}$$

. Since we don't know factors of $y^r - 1$, we can't get absolute factors from the factors of the kind $x - q(y)$. When we implement our algorithm, to keep the intermediate polynomials monic(while gcd computation) we need to perform divisions modulo h. If we are not successful in divisions, then we get some factor of h and recursively compute factors of f(x) modulo the factors of h.

Observation 2. *To avoid this flipping, we felt that it may be interesting to look at the cases where there is exactly 1 component of R (as defined in the algorithm). Using lemma 6, we can found out such good r. Using density of prime numbers, we can say that they are quite abundant.*

6 Results

6.1 Checking conjecture 3

When we chose such good r(as defined in Observation 2) we found out that we quite often get absolute factors of $f(x)$. We randomly generated primes and tested the algorithm. Table 6.1 gives the minimum value of r (lets say this value is r*) for which we got absolute factors for given f and p (here we are factoring $f = x^2 - a$ over F_p).

Table 1: Minimum value of r i.e. r* for which we get absolute factors given p and a. p and a have been generated randomly

p	a	r*
107897	83458	7
108293	61596	5
114781	35551	7
110477	90762	7
192173221	32342986	2
182158409	29617101	9
195248057	44040277	3
180133673	175430357	3
188698553	185209056	5
188139541	94225242	7
186947689	72493289	3
185269297	52453857	5
191022809	157277712	2
189122881	14325146	2
196584257	112679783	5
187856969	30238769	2
192093637	119286441	15
193800713	32913580	3
181495709	50957795	3
196084033	49493777	9
192296861	85819729	2
2222105477	326069981	3
2059175449	534600332	2
2143375301	1973524134	2
2132716609	652247966	5
2190848129	1719748497	2
2141699801	969983963	2
2114656769	969074251	2
2199143101	1027256781	7
2100078521	1050192805	7
2199429613	1926325142	13
2166553861	1513615265	5
2162781889	509762812	2
2129227549	1830209562	5
2095037801	1610267164	2
2054970677	1380913385	5
2158401793	22589838	5
2230740097	550996218	9

Table 2: Distribution of $|F_p*|/|C|$ w.r.t p and r

p	r	$\|F_p*\|$	$\|F_p*\|/\|C\|$	Is C subset of F_p^2* ?
7	2	6	1	No
11	2	10	2	Yes
11	3	120	2	Yes
13	2	12	1	No
17	2	16	2	Yes
17	3	288	3	No
19	2	18	2	Yes
23	2	22	1	No
23	3	528	4	Yes
29	2	28	1	No
29	3	840	5	No
31	2	30	3	No
37	2	36	1	No
37	5	1874160	137	No
41	2	40	2	Yes
41	3	1680	35	No
43	2	42	6	Yes
43	5	3418800	185	No
47	2	46	1	No
47	3	2208	8	Yes
47	5	4879680	663	No

6.2 Proving the conjecture 3

We tried to prove the time bound by looking at the cardinality of the set $S \equiv \{y-1, y^2-1, y^3-1, ..., y^{r-1}-1\}$ when we work mod irreducible $h(y) = \frac{y^r-1}{y-1}$. C is the subgroup generated by S.

Claim 2. *If cardinality of the set C > cardinality of the set of quadratic residues, then the shifted root will be a non-quadratic residue in at least one of the attempts of r from 1 to nearly log(p) and our algorithm will factor the polynomial.*

The index of this set S in F_p^2 i.e. F_p^2/S is given in the table 6.2.

7 Conclusions

Factoring of polynomials over F_p is indeed very efficient when we do all computation in $R \equiv$ *polynomial ring over* $F_p[y]/<f(y)>$ where $f(y) = \frac{y^r-1}{y-1}$, in particular, when f is irreducible.

All codes and raw data can be found at this github repository. Documentation will be available soon.

References

[1] Elwyn R Berlekamp. Factoring polynomials over large finite fields. *Mathematics of Computation*, 24(111):713–735, 1970.

[2] David G Cantor and Hans Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, pages 587–592, 1981.

[3] Joachim von zur Gathen and Daniel Panario. Factoring polynomials over finite fields: A survey. *Journal of Symbolic Computation*, 31(12):3 – 17, 2001.