

University of Bath

**DEPARTMENT OF COMPUTER SCIENCE
EXAMINATION**

CM30070: Computer Algebra
(with *answers* and **markers comments**)

As a general point, it is important to read the paper and answer the whole question. One candidate, for example, gave a perfect answer to the first two parts of question 2, but totally failed to answer the third, as if it never been there

No calculators may be brought in and used.

Full marks will be given for correct answers to **THREE** questions.
Only the best three answers will contribute towards the assessment.

Examiners will attach importance to the number of
well-answered questions.

1. (a) What are meant by ‘normal’ and ‘canonical’ representations? [2]
A Normal: 0 has a unique representation. Canonical: every object has a unique representation.
- (b) What are meant by ‘sparse’ and ‘dense’ in computer algebra? [2]
A Sparse means that only non-zero items are stored, e.g. x^2+1 rather than $x^2+0\cdot x+1$, whereas dense stored the zeros as well.
- (c) How are parts (a) and (b) related? [4]
A To implement something sparsely, you have to recognise that an item is zero, which is much easier if the underlying representation is normal.
C Some people did state this point. There was, unfortunately a lot of rubbish also written here, such as “only dense representations can be canonical”.
- (d) Assuming that you have a *normal* representation of the entries, outline a normal sparse representation of $n \times n$ square matrices. [4]
*A Store a list of triples $(i, j, a_{i,j})$, with $a_{i,j} \neq 0$, as can be told by normality. Can't have two entries with same (i, j) : **several students missed this point.** Should also store n for an $n \times n$ matrix, otherwise can't tell (1) from $\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$. Normal since an $n \times n$ zero matrix is just stored as n , or (N . NIL) if one prefers.*
- (e) What more would you need to do to make the representaton *canonical*? [4]
A Two things:
*(i) the entries themselves would have to be canonical: **several students missed this point;***
*(ii) we need to impose some order on the triples, say purely lex, otherwise both $((1,1,1)(2,2,1))$ and $((2,2,1)(1,1,1))$ would represent the 2×2 identity: **several students missed this point.***
- (f) Outline an algorithm for multiplying two such canonical matrices. If they each have kn non-zero entries, for small fixed k , how does the complexity depend on n ? [4]
A “transpose B then zip each row in $A \cdot B^T$ ” is the obvious one, with complexity $O(kn \log kn)$ for the transpose (assuming a merge sort), and $O(n \cdot (kn))$ for the zip itself, totally $O(kn^2)$. There are probably better ways.
C This part was not done well: several students forgot their representation and told me how to multiply dense matrices. One student told me how to add matrices.

2. In genera, the first two parts were done extremely well, but the last part, which required understanding what a Gröbner base was doing, much more poorly.

Give a computationally testable definition of what it means for a set of polynomials (with coefficients from a field) to be a Gröbner base, defining any terms you use which are specific to Gröbner base theory. [4]

(This is the only constructive definition I gave: the rest all rely on ideals, which aren't testable without GB.).

G is a Gröbner base with respect to a particular admissible order $>$ on monomials if and only iff

$$\forall f, g \in G \quad S(f, g) \xrightarrow{*G} 0 \quad \text{where}$$

$$S(f, g) \text{ is } \frac{\text{lt}(g)}{\text{gcd}(\text{lm}(f), \text{lm}(g))} f - \frac{\text{lt}(f)}{\text{gcd}(\text{lm}(f), \text{lm}(g))} g$$

$\xrightarrow{*G}$ is the result of carrying out \rightarrow^G until it is no longer possible

$f_1 \rightarrow^G f_2$ is f_2 being the result of subtracting $\frac{\text{lt}(f_1)}{\text{lt}g} g$ from f_1 where g is any element of G with $\text{lm}(g)$ dividing $\text{lm}(f_1)$.

C the next two are arguably not specific to Gröbner base theory, so I didn't deduct marks for not giving them.

$\text{lt}(f)$ is the leading term of f (i.e. power product and coefficient)

$\text{lm}(f)$ is the leading monomial of f (i.e. power product only)

Describe Buchberger's algorithm, and show how your testable definition proves that, if the algorithm terminates, the result is a Gröbner basis. [8]

If any $S(f, g) \xrightarrow{*G} h \neq 0$, then $h \in (G)$, so add h to G and keep testing. If this terminates, the resulting G is a Gröbner base by the test above. This is Buchberger's algorithm.

What does Buchberger's algorithm reduce to in the following special cases (you should state any additional results required)? [8]

- All the polynomials are in the same one variable only, i.e. in $K[x]$.
- All the polynomials are linear.

A (Marks split 3 and 5)

- Since reduction of one univariate by another is computation of the remainder, this reduces to Euclid's algorithm. We therefore get the g.c.d.
- We need the g.c.d. criterion, that if $\text{gcd}(\text{lm}(f), \text{lm}(g)) = 1$ we don't need to compute the S -polynomial. But for linear polynomials, if the g.c.d. is not 1, it must be the common leading variable. Hence the S -polynomial is just scaled subtraction, and we have Gaussian elimination.

C It is rather worrying that one candidate wrote "I do not know what linear means".

3. **The first part was generally well done. People lost marks for not explaining the Landau–Mignotte bound, or the results on good/bad reduction based on the degrees of the gcd. Those who attempted the second part generally did well.**

Explain how modular methods (Chinese Remainder Theorem) can be used to avoid intermediate expression swell in the computation of greatest common divisors of polynomials in $\mathbf{Z}[x]$ (i.e. univariate polynomials over the integers). *State carefully any results you use.* [12]

Bookwork. Need to state

1. *Landau–Mignotte bound*
2. *If p does not divide the leading coefficients (in fact, it is sufficient for it not to divide one of them, but this refinement isn't necessary), then $\deg \gcd(f_p, g_p) \geq \deg \gcd(f, g)$. Hence a test for relative luckiness.*
3. *Chinese Remainder Theorem to combine information from p and q .*

Computation of Gröbner bases also suffers from intermediate expression swell. What obstacles stand in the way of using modular methods here? [8]

Various points might come out. These two should be mentioned, and would get full marks.

1. *Lack of an equivalent to the Landau–Mignotte bound (partially solved by just trying, and probably using early abort, i.e. if the result doesn't change when we add a further prime, let's hope it's right).*
2. *If the g.c.d. case, if the degrees differed, we knew the p with the lower degree was 'less unlucky' — here if we get different-shaped results it's not so obvious (partially solved by Hilbert luckiness, see [Arn03]). Hence we have a more complicated test for relative luckiness.*

One could also mention (no-one did) that testing for correctness of the result isn't so obvious (again see [Arn03]).

4. Throughout this question, you should assume that factoring a univariate polynomial modulo a prime p is a *solved* problem, e.g. by Cantor–Zassenhaus.

Given this, why are modular (Chinese Remainder Theorem) methods *not* used for polynomial factorization? [4]

Because, if $f \pmod{p}$ factors as $f_1^{(p)} f_2^{(p)}$ of the same degree, and if $f \pmod{q}$ factors as $f_1^{(q)} f_2^{(q)}$ of the same degree, should we pair $f_1^{(p)}$ with $f_1^{(q)}$ (and therefore $f_2^{(p)}$ with $f_2^{(q)}$), or the other way round. In general, if there were n factors and k primes, we would have $n!^{k-1}$ possible pairings.

C **Almost no-one got this right, and generally there was much waffle.**

Given an *arbitrary* polynomial in $\mathbf{Z}[x]$, outline a method of factorizing it based on the assumption above. Be sure to indicate what the constraints on p are. [12]

Bookwork, but requires pulling a few bits together.

1. If f is not square-free, do a square-free decomposition, and work on each component of this.
2. Choose p not dividing $\text{lc}(f)$ and such that $f^{(p)}$ is still square-free (3 marks here)
3. Factor $f^{(p)}$ as per assumption
4. Linear Hensel lift (as in book) to a factorization modulo p^n , greater than twice Landau–Mignotte bound
5. Check this is a factorization over \mathbf{Z} , and if not try recombinations, remembering to try subsets of T before T itself.

C **Those who did it generally did quite well, though some forgot the first step, and the square-free check in the second.**

What might you do to make this algorithm more efficient (brief sketches are all that is required)? [4]

Various possibilities.

- factor $f^{(p)}$ for several p and do a Musser-style compatibility test, choosing the best p .
- Twice Landau–Mignotte bound is probably overkill, so check early.
- Quadratic lifting (this has been mentioned, but not taught in detail, so all that is needed is “better lifting” or words to that effect)

C **Not particular well done — some people hinted at the second point.**

References

- [Arn03] E.A. Arnold. Modular algorithms for computing Gröbner bases. *J. Symbolic Comp.*, 35:403–419, 2003.