

University of Bath

**DEPARTMENT OF COMPUTER SCIENCE
EXAMINATION**

CM30070: Computer Algebra

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. What are meant by ‘normal’ and ‘canonical’ representations? [2]

Normal: 0 has a unique representation. Canonical: every object has a unique representation.

Describe a *factored* representation of polynomials. [4]

JHD described one in which f is stored as $c \prod f_i^{n_i}$ where c is a content, the f_i are primitive square-free, relatively prime, and stored as $\sum a_i m^{e_i}$, where the a_i are (non-zero, therefore the representation is sparse) coefficients and the m_i are monomials in some admissible ordering. Zero is stored specially (e.g. as NULL).

To what extent is your representation normal, or canonical? How would you test for equality? [6]

Certainly normal. Not canonical, though if items have a different shape in terms of a different set (not multiset) of n_i they are certainly different. To test for equality, for a given value v we need to check whether $\prod_{n_i=v} f_i = \prod_{\hat{n}_i=v} \hat{f}_i$.

To what extent does your representation rely on g.c.d. computations; what properties of the g.c.d. process do you need for a reasonably efficient representation, and how might they be achieved? [8]

When multiplying polynomials, we need to check for common factors, e.g. $f \times g$, where in fact $f = h\hat{f}$ and $g = h\hat{g}$, will be $h^2\hat{f}\hat{g}$. After we’ve added polynomials, we need to perform a square-free decomposition. In both cases, we want “there is only a trivial g.c.d.” to be returned quickly, and a modular method will do this most of the time, which is sufficient.

2. State the Bareiss–Dodgson Theorem, as it applies to fraction-free Gaussian elimination in matrices with coefficients in an integral domain. [4]

Bookwork.

The subresultant algorithm for the g.c.d. of two polynomials f and g with coefficients in an integral domain is given as follows.

```

i := 1;
if deg(f) < deg(g)
  then  $a_0 := \text{pp}(g); a_1 := \text{pp}(f);$ 
  else  $a_0 := \text{pp}(f); a_1 := \text{pp}(g);$ 
 $\delta_0 := \text{deg}(a_0) - \text{deg}(a_1);$ 
 $\beta_2 := (-1)^{\delta_0+1};$ 
 $\psi_2 := -1;$ 
while  $a_i \neq 0$  do
   $a_{i+1} = \text{prem}(a_{i-1}, a_i) / \beta_{i+1};$ 
  # $q_i :=$ the corresponding quotient:  $a_{i+1} = a_{i-1} - q_i a_i$ 
   $\delta_i := \text{deg}(a_i) - \text{deg}(a_{i+1});$ 
   $i := i + 1;$ 
   $\psi_{i+1} := -\text{lc}(a_{i-1})^{\delta_{i-2}} \psi_i^{1-\delta_{i-2}};$ 
   $\beta_{i+1} := -\text{lc}(a_{i-1}) \psi_{i+1}^{\delta_{i-1}};$ 
return  $\text{pp}(a_{i-1});$ 

```

Considering only the *regular* case, i.e. $\delta_0 = 0$ and all other $\delta_i = 1$, sketch how the Bareiss–Dodgson Theorem implies that the divisions involved are exact. [8]

In the regular case, $\psi_{i+1} := \pm \text{lc}(a_{i-1})$ and $\beta_{i+1} := -\text{lc}(a_{i-1})^2$, and the only division is by β_{i+1} after the prem, and then only when $i > 1$, since $\beta_2 = -1$

It has been suggested that, rather than calculate all the ψ and β , we should simply remember all the previous $\text{lc}(a_i)$, and divide through after each step (i.e. after the prem) by all of those that divide exactly, as often as possible (starting with the most recent leading coefficient). Show that, in the *regular* case, this has the same effect as the subresultant algorithm. What might go wrong in general? [8]

This is Hearn's 1979 paper. In the regular case, as shown above, the β_i are merely (powers of) leading coefficients in the subresultant algorithm, so we will indeed divide out β_i . We may divide out more, which will make the next leading coefficients merely factors of the subresultant ones, but the cofactors will also be previous leading coefficient, so will still be divided out.

The β_i are now quotients of the leading coefficients by previous ψ (ultimately by leading coefficients), so they may cancel whereas the whole leading coefficient might not.

3. What is an *admissible* ordering on monomials? [1]

1 is the least monomial, and $a < b \Rightarrow ac < bc$.

Describe *two* common orderings used in the solution of systems of polynomial equations. [2]

For example these.

- **plex**(x, y, \dots): *the one with the higher power of x wins; if equal then take the one with the higher power of y ;*
- **tdeg**(x, y, \dots): *the one with the higher total degree wins; if the total degrees are the same, the one which wins lexicographically loses. (Maple reverses the variable order as well, which might or might not be stated, but has to be consistent with the next part.)*

What is the most general form of admissible ordering, and how can the orderings you have given be expressed in terms of it? [3]

Matrix orderings: "Let \mathbf{M} be a fixed $n \times n$ matrix of non-negative reals, and regard the exponents of A as an n -vector \mathbf{a} . Then we compare A and B by computing the two vectors $\mathbf{M}\mathbf{a}$ and $\mathbf{M}\mathbf{b}$, and comparing these lexicographically." For **plex**, \mathbf{M} is

*the identity matrix. For **tdeg** it is $\mathbf{M} = \begin{pmatrix} 1 & 1 & \dots & 1 & 1 \\ -1 & 0 & \dots & 0 & 0 \\ 0 & \ddots & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & -1 & 0 \end{pmatrix}$, or the reflection*

of this if we do the Maple-like inversion of variables.

Given a Gröbner base for an ideal with respect to *any* ordering, how can you check whether the corresponding system of equations has a *finite* number of solutions, and, if so, determine how many there are? Describe how your method handles multiplicities of solutions? [5]

To check that there are only a finite number of solutions, we need to know that each variable occurs alone, to some power, as the leading monomial of some polynomial in the GB. The number of solutions (with multiplicity) is the number of irreducible monomials. Trying to count without multiplicity is a much harder question.

The attached Maple worksheet computes two Gröbner bases **m** and **mm**. For each of them, what can you deduce about the number of solutions? Note that you are *not* required to deduce the solutions themselves. [5]

m *d does not occur alone, so there are infinitely many solutions.*

mm *Each variable does occur alone, and the irreducible monomials are $1, c, c^2, b$, so there are four solutions.*

4. Describe *two* of these algorithms, stating: [10 each]

the purpose of the algorithm;

the inputs and outputs;

if there are any pre-conditions on the outputs, how they might be achieved;

the general methodology of the algorithm.

(for the subresultant algorithm in question 2, the general methodology might be “we perform a fraction-free version of Euclid’s algorithm, but keep track of the leading coefficients by which we have multiplied to ensure fraction-free, since Bareiss–Dodgson guarantees that suitable multiples of these can be cancelled.)

1. The Cantor–Zassenhaus algorithm.
2. The Hermite algorithm for rational function integration.
3. The Trager–Rothstein algorithm.

Largely bookwork, but reasonably intelligent summarisation required.