Prof. Friedrich Eisenbrand
Nicolai Hähnle

Discussions from: April 5, 2011

Computer Algebra

Spring 2011

Assignment Sheet 4

Warning: These are just notes and not necessarily full solutions for each exercise. Full solutions may require some additional details to be fleshed out. Please report any mistakes you may find.

Exercise 1

Prove that

$$\operatorname{ord}_p(n!) = \frac{n - S_p(n)}{p-1},$$

where $S_p(n)$ is the sum of the digits of n written in base p.

Let us write n in base p as

$$n = \sum_{j=0}^{\infty} a_j p^j,$$

where $a_j \in \{0, 1, \ldots, p-1\}$. Recall from the lecture that

$$\operatorname{ord}_p(n!) = \sum_{k=1}^{\infty} \lfloor n/p^k \rfloor$$

This allows us to compute $\operatorname{ord}_p(n!)$ in terms of the digits of n:

$$\operatorname{ord}_p(n!) = \sum_{k=1}^{\infty} \sum_{j=k}^{\infty} a_j p^{j-k},$$

and rearranging the order of summation, which is allowed because even though the sums appear to be infinite, there are in fact only finitely many non-zero summands:

$$\begin{aligned}
&= \sum_{j=1}^{\infty} a_j \sum_{i=0}^{j-1} p^i \\
&= \sum_{j=0}^{\infty} a_j \sum_{i=0}^{j-1} p^i \\
&= \sum_{j=0}^{\infty} a_j \cdot \frac{p^j - 1}{p-1} \\
&= \frac{\left(\sum_{j=0}^{\infty} a_j p^j\right) - \sum_{j=0}^{\infty} a_j}{p-1} = \frac{n - S_p(n)}{p-1}
\end{aligned}$$

Exercise 2 ($\star$)
Let us write $f = a_0 + a_1 x + \ldots + a_d x^d$, where $a_d \neq 0$ and d is odd. Assume that $a_d > 0$; otherwise, we can replace f by $-f$, which does not change the roots of the polynomial. Then we actually get $a_d \geq 1$, because the polynomial's coefficients are integer.

Set $M = \max\{|a_0|, \ldots, |a_d|\} + 1$. Then we can compute (using the formula for geometric series):

$$\begin{aligned} f(M) &= a_0 + a_1 M + \ldots + a_{d-1} M^{d-1} + a_d M^d \\ &\geq -(M-1) - (M-1)\cdot M - \ldots - (M-1)\cdot M^{d-1} + M^d = -(M-1)\cdot \frac{M^d - 1}{M-1} + M^d \geq 1 \end{aligned}$$

A similar calculation yields $f(-M) \leq -1$, because $(-M)^d = -M^d$ (recall that d is odd). By continuity of f, this implies that f has a root in the interval $[-M, M]$. Now consider the following algorithm:

FINDROOT$(f, [a,b], \varepsilon)$
1 **if** $b - a \leq \varepsilon$
2 **then return** $[a,b]$
3 $m \leftarrow \frac{a+b}{2}$
4 **if** $f(m) < 0$
5 **then return** FINDROOT$(f, [m,b], \varepsilon)$
6 **else return** FINDROOT$(f, [a,m], \varepsilon)$

We claim: Let $[a,b]$ contain a root of f and let $f(a) \leq 0$ and $f(b) \geq 0$, then the call FINDROOT$(f, [a,b], \varepsilon)$ returns an interval $[a', b']$ of length at most ε that contains a root of f.

For a fixed ε, the proof of this claim goes by "induction" on the length of $[a,b]$. It is clear that the claim is true whenever $b - a \leq \varepsilon$. Now suppose that the claim is true whenever $b - a \leq \ell$. We want to show that it is true for $b - a \leq 2\ell$. We do a case distinction. If $f(m) < 0$, then by continuity of f, the interval $[m,b]$ contains a root of f, and we have $f(m) \leq 0$ and $f(b) \geq 0$. Furthermore, $b - m = \frac{b-a}{2} \leq \ell$, so by "induction hypothesis", the recursive call returns an interval of length at most ε that contains a root of f. The case $f(m) \geq 0$ is analogous.

The above is not a classic case of induction, because we work on real (or rational) variables. In this particular case, to turn the argument into a formally correct induction, we would prove the claim for interval lengths up to $\varepsilon \cdot 2^k$ by induction on k.

We have established the claim, and so calling FINDROOT$(f, [-M, M], \varepsilon)$ will return the desired interval. It remains to verify that the running time of this algorithm is bounded by a polynomial in the input size.

First of all, M can clearly be computed in linear time in the input size, because we simply need to read all coefficients of the polynomial and remember the largest one in absolute value.

Second, each level of recursion can be performed in polynomial time, because a straight-

forward evaluation of the polynomial runs in polynomial time.[1] It remains to determine the depth D of the recursion.

Since the length of the interval is halved at each recursive step, we know that

$$(2M) \cdot 2^{-D} > \varepsilon$$

which implies

$$D < \log \frac{2M}{\varepsilon} = O(\log M + \log 1/\varepsilon)$$

So the recursion depth is bounded by a polynomial, and since we only have tail recursion, this implies together with the above observations that the total running time is bounded by a polynomial in the input size and $\log 1/\varepsilon$.

Exercise 3

Let $A \in \mathbb{Q}^{n \times n}$. Denote the columns of A by $a_1, \ldots, a_n$. Let B be an upper bound on the absolute values of entries in A.

1. Show the Hadamard bound $|\det(A)| \leq \prod_{j=1}^{n} |a_j|_2$, where $|\cdot|_2$ is the Euclidean norm.

   Recall the Gram-Schmidt orthogonalization process. It decomposes $A = B \cdot U$, where B is a matrix with pairwise orthogonal columns and U is a triangular matrix with ones on the diagonal.

   In slightly more detail, we successively define b_j to be the projection of a_j onto the orthogonal complement of the space spanned by $b_1, \ldots, b_{j-1}$, for $j = 1 \ldots n$. In other words, we decompose

   $$a_j = b_j + \sum_{i=1}^{j-1} u_{ij} b_i$$

   If we then set $u_{jj} = 1$, the matrix $U = (u_{ij})_{ij}$ is the desired matrix in the matrix equation above.

   Most importantly, we have

   $$\det(A) = \det(B) \cdot \det(U) = \det(B),$$

   because the determinant is a group homomorphism and U is triangular. Furthermore, we can compute

   $$\begin{aligned} \det(B)^2 &= \det(B^T) \cdot \det(B) = \det(B^T B) \\ &= \det \begin{pmatrix} b_1^T b_1 & \ldots & b_1^T b_n \\ \vdots & \ddots & \vdots \\ b_n^T b_1 & \ldots & b_n^T b_n \end{pmatrix} \\ &= \det \begin{pmatrix} |b_1|^2 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & |b_n|^2 \end{pmatrix} = \prod_{j=1}^{n} |b_j|^2 \end{aligned}$$

[1] To be totally precise, we should look at the size of the numbers involved. We eventually have to compute with rational numbers once the length of the interval becomes small enough, and so the size of the numbers does depend on the size of ε.

Taking the absolute value of the square root, we obtain $|\det(A)| = \prod_{j=1}^{n} |b_j|$. Furthermore, observe that

$$|a_j|^2 = |b_j + \sum_{i=1}^{j-1} u_{ij} b_i|^2 = |b_j|^2 + \sum_{i=1}^{j-1} |u_{ij} b_i|^2 \geq |b_j|^2$$

by the Pythagorean theorem (because the b_j are pairwise orthogonal!). This implies $|b_j| \leq |a_j|$,[2] which we can plug into the earlier result to get the desired inequality $|\det(A)| \leq \prod_{j=1}^{n} |a_j|$.

2. Simply compute

$$|\det(A)| \leq \prod_{j=1}^{n} |a_j| = \prod_{j=1}^{n} \sqrt{a_{j1}^2 + \ldots + a_{jn}^2} \leq \prod_{j=1}^{n} \sqrt{nB^2} = (nB^2)^{n/2} = n^{n/2} B^n$$

The Leibniz bound is $|\det(A)| \leq n! B^n$. Recall Stirling's approximation of the factorial:

$$n! \sim \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$$

For large values of n, we can clearly see that $n!$ is much bigger than $n^{n/2} = \sqrt{n}^n$, and so the Hadamard bound is better asymptotically.[3]

Exercise 5

We want to compute the determinant of

$$A = \begin{pmatrix} 1 & 0 & -2 \\ 2 & -1 & 1 \\ 0 & 2 & 2 \end{pmatrix}$$

1. Modulo 3, we get the matrix

$$\begin{pmatrix} 1 & 0 & 1 \\ 2 & 2 & 1 \\ 0 & 2 & 2 \end{pmatrix}$$

After Gauss elimination using elementary row operations, we obtain

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 2 & 2 \\ 0 & 0 & 0 \end{pmatrix}$$

so clearly $\det(A) = 0 \pmod 3$. Modulo 5, we have

$$A = \begin{pmatrix} 1 & 0 & 3 \\ 2 & 4 & 1 \\ 0 & 2 & 2 \end{pmatrix}$$

[2] Intuitively, this fact should be obvious: b_j is an orthogonal projection of a_j, so naturally it must be shorter.

[3] Note, however, that the difference is "only" a constant in the exponent of the n, so the difference is not important for many proofs and asymptotic considerations.

and after Gauss elimination, this becomes

$$\begin{pmatrix} 1 & 0 & 3 \\ 0 & 4 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

so from the diagonal we can read off that $\det(A) = 1 \cdot 4 \cdot 2 = 3 \pmod 5$. Finally, modulo 7 one has

$$A = \begin{pmatrix} 1 & 0 & 5 \\ 2 & 6 & 1 \\ 0 & 2 & 2 \end{pmatrix}$$

which becomes

$$\begin{pmatrix} 1 & 0 & 5 \\ 0 & 6 & 5 \\ 0 & 0 & 5 \end{pmatrix}$$

so from the diagonal we can see that $\det(A) = 2 \pmod 7$.

2. The Leibniz bound tells us that $|\det(A)| \leq n! \cdot B^n$ for an $n \times n$ square matrix with entries bounded by B in absolute value. Plugging in the values for the given matrix A, we get

$$|\det(A)| \leq 6 \cdot 2^3 = 48,$$

so $2|\det(A)| + 1 \leq 105$. In particular, there are at most 105 possible values for $\det(A)$ between -48 and 48. This means that the natural map $\{-48, \ldots, 48\} \to \mathbb{Z}_{105}$ is injective, and if we can determine $\det(A) \in \mathbb{Z}_{105}$, we can find its preimage in $\mathbb{Z}$.

Let us use the Chinese Remainder Theorem, using the fact that $105 = 3 \cdot 5 \cdot 7$, to find $x \in \mathbb{Z}_{105}$ such that

$$\begin{aligned} x &\equiv 0 \pmod 3 \\ x &\equiv 3 \pmod 5 \\ x &\equiv 2 \pmod 7 \end{aligned}$$

Let us first find the image of an element of the form $(0, x, 0) \in \mathbb{Z}_3 \times \mathbb{Z}_5 \times \mathbb{Z}_7$ in $\mathbb{Z}_{105}$. Since such an image has to be a multiple of both 3 and 7, simply consider $3 \cdot 7 = 21$. Observe that $21 = 1 \mod 5$, and so 21 is the image of $(0,1,0)$ under the CRT isomorphism. Similarly, it just so happens that $3 \cdot 5 = 15 = 1 \mod 7$[4], and so 15 is the image of $(0,0,1)$ under the isomorphism. Consequently, $x = 3 \cdot 21 + 2 \cdot 15 = 93$ satisfies the modular equation above (check this!). So $\det(A) = 93 \pmod{105}$.

Recall the natural map $\{-48, \ldots, 48\} \to \mathbb{Z}_{105}$ mentioned above. The preimage of 93 under that map is -12 (since $93 \equiv -12 \pmod{105}$), and so $\det(A) = -12$.

3. Using the Leibniz bound, compute an upper bound $M \geq 2|\det(A)| + 1$. In particular, we get $M = 2n! \cdot B^n + 1$, where B is a bound on the matrix entries. We want to find

[4] It is quite a lucky coincidence that we get 1 modulo the desired prime both times. In general, this is not the case, e.g. $5 \cdot 7 = 2 \mod 3$.

small prime numbers $p_1 < \ldots < p_k$ such that $\prod_{j=1}^{r} p_k \geq 2|\det(A)| + 1$. How many prime numbers do we need, and how can we find them?

Naturally, it makes sense to simply use the smallest k prime numbers, or alternatively all prime numbers smaller than some m. We then need to bound the product of those prime numbers:

$$\prod_{p \leq m, p \in \mathbb{P}} p \geq \prod_{\sqrt{m} < p \leq m, p \in \mathbb{P}} p \geq \prod_{\sqrt{m} < p \leq m, p \in \mathbb{P}} \sqrt{m} = \sqrt{m}^{\pi(m) - \pi(\sqrt{m})} \geq \sqrt{m}^{\pi(m) - \sqrt{m}},$$

where π is the prime counting function. So it suffices to find m large enough so that

$$\sqrt{m}^{\pi(m) - \sqrt{m}} \geq M$$

Taking logarithms on both sides, we get

$$(\pi(m) - \sqrt{m}) \cdot \frac{1}{2} \log m \geq \log M$$

Since we know that $\pi(m) \leq c \cdot \frac{m}{\log m}$ for a constant c, we can conclude that a feasible choice of m must satisfy:

$$\log M \leq (\pi(m) - \sqrt{m}) \cdot \frac{1}{2} \log m \leq c' \cdot m - \frac{\sqrt{m} \log m}{2}$$

or in other words, since the first term asymptotically dominates, $m = c'' \cdot \log M$ should be sufficient to get

$$\prod_{p \leq m, p \in \mathbb{P}} p \geq M$$

for a sufficiently large M.[5]

We can find all primes less than m using the Sieve of Eratosthenes. Note that $m = O(\log M)$, so the space and running time required to run the sieve is polynomial in the *encoding length* of the input (check this by plugging in the Leibniz formula for the bound M!).

Once the $\pi(m)$ many primes have been found, we compute $\det(A)$ modulo each of them and combine the result. The advantage is that each prime will fit into a single machine word on any reasonably-sized instance, and so native arithmetic can be used for computing the determinants. Arbitrary-length integers are then only used for the combination steps.

A lot of details are missing, including a proper analysis of the running time, but they should all be relatively straightforward.

Exercise 6

[5] To be precise, the above is not a proof of that fact, it's simply a calculation that gives us an idea how big m needs to be chosen. To complete the proof, one would have to fix a constant c'' and then substitute this choice of m to prove that the resulting product of primes really is bigger than M.

1. Let $T_1 = (V(T_1), E(T_1))$ and $T_2 = (V(T_2), E(T_2))$ be rooted trees with roots r_1 and r_2, respectively. Considering the nature of rooted trees, it makes sense to look for proofs by induction on the height of the trees involved. To facilitate this, I want to use the following fact.

   Proposition 1. *Let $N(r_1) = \{v_1, \dots, v_k\}$ be the set of children of r_1, and $N(r_2) = \{w_1, \dots, w_l\}$ the children of r_2. Then T_1 is isomorphic to T_2 if and only if there exists a bijection $\phi : N(r_1) \to N(r_2)$ such that for all $v \in N(r_1)$ the subtree rooted at v is isomorphic to the subtree rooted at $\phi(v)$.*

   The truth of this proposition is hopefully clear from an intuitive point of view, though proving particularly the "only if" direction in full formality is a rather annoying technical task. As a sub-lemma, one would prove that if $f : V(T_1) \to V(T_2)$ is an isomorphism of the two trees, then the subtree rooted at a vertex $v \in V(T_1)$ is always isomorphic to the subtree rooted at $f(v)$. The "if" direction is slightly simpler: here, all that one needs to do is to "stitch together" all the isomorphisms for the subtrees. However, let us just assume this proposition here without proof.

   Let us prove by induction on their height that if the two trees are isomorphic, the corresponding polynomials f_{r_1} and f_{r_2} are equal. If the height of the trees is 0 (that is, the trees contain only the root), $f_{r_1} = f_{r_2} = 1$. Otherwise, we know by the previous proposition that there is a bijection $\phi : N(r_1) \to N(r_2)$ such that the subtree rooted at v is isomorphic to the subtree rooted at $\phi(v)$ for all $v \in N(r_1)$. Since those subtrees have a lower height, we can use induction to argue that $f_v = f_{\phi(v)}$. Finally,

   $$f_{r_1} = (x_h - f_{v_1}) \cdots (x_h - f_{v_k}) = (x_h - f_{\phi(v_1)}) \cdots (x_h - f_{\phi(v_k)}) = (x_h - f_{w_1}) \cdots (x_h - f_{w_l}) = f_{r_2}$$

   For the reverse direction, let us first think of the polynomials as polynomials in $R[x_1, \dots, x_h]$, where R is an arbitrary integral domain. Now suppose that

   $$(x_h - f_{v_1}) \cdots (x_h - f_{v_k}) = (x_h - f_{w_1}) \cdots (x_h - f_{w_l})$$

   We can also think of those polynomials as univariate polynomials in $(R[x_1, \dots, x_{h-1}])[x_h]$. Since the polynomials are equal, they must have the same zeros, and since $R[x_1, \dots, x_{h-1}]$ is an integral domain,[6] the zeros are exactly $f_{v_1}, \dots, f_{v_k}$ for the left hand side and $f_{w_1}, \dots, f_{w_l}$ on the right hand side with corresponding multiplicities, and thus there exists a bijection $\phi : N(r_1) \to N(r_2)$ such that $f_v = f_{\phi(v)}$ for all $v \in N(r_1)$. Again, using induction on the height, we can conclude that the corresponding subtrees are isomorphic (because their associated polynomials are equal), and using the proposition above this implies that the trees T_1 and T_2 are isomorphic.

2. The obvious approach to testing whether the trees are isomorphic, given the previous part of this exercise, is to test the equality $f_{r_1} = f_{r_2}$. How big are those polynomials in terms of their computational representation?

[6] The polynomial ring $S[x]$ is a unique factorization domain when S is an integral domain. That is, every polynomial has a unique factorization into irreducibles (up to the obvious permutations or muliplication by units). Since we are given two factorizations of a polynomial into linear factors, it follows that the two factorization must in fact be equal.

Let us first look at the total degree of the polynomials. We see that

$$\deg(f_{r_1}) = \deg(f_{v_1}) + \deg(f_{v_2}) + \ldots + \deg(f_{v_k})$$

and by spinning this inequality inductively along the height of the tree, we see that the total degree is bounded by the number of leaves of the tree. Unfortunately, the polynomials are not univariate, but are in fact polynomials in h variables, where h is the height of the trees. A multivariate polynomial in h variables with total degree bounded by n can have up to $\binom{n+h}{h}$ coefficients when one expands it as a linear combination of monomials.[7] This quantity is not bounded by a polynomial in the size of the trees. Clearly, multiplying everything and then comparing the polynomials coefficient by coefficient appears to require too much space and take too long.[8]

However, we *can* quickly evaluate the polynomial at a given point by a simple recursive procedure following the structure of the tree:

```
EVALUATE(T, x_1, ..., x_h)
    ▷ T is a tree of height h
1   if h = 0
2     then return 1
3     else r ← 1
            ▷ Iterate over all children of the root r of T
4           for all v ∈ N(r)
5             do S ← subtree rooted at v
6                r ← r · (x_h − EVALUATE(S, x_1, ..., x_{h(S)}))
7           return r
```

It is easy to check that the number of ring operations performed by this function is linear in the size of the tree, and that it returns $f(x_1,\ldots,x_h)$, where f is the polynomial corresonding to the tree T.

So let us fix a finite field k of size m and let us choose the variables $x_1, \ldots, x_h \in k$ independently and uniformly at random. Then we check whether $f_{r_1}(x_1,\ldots,x_h) = f_{r_2}(x_1,\ldots,x_h)$ holds for the randomly chosen values. If it holds, we claim that the trees are isomorphic. If it doesn't hold, we claim that they are non-isomorphic. Of course the equality always holds if the trees are isomorphic. What is the probability that we are mislead into believing that the trees are isomorphic when they are not?

The Schwartz-Zippel lemma applied to the polynomial $f_{r_1} - f_{r_2}$ provides the answer: the probability is less than or equal to

$$\frac{\max\{\deg(f_{r_1}), \deg(f_{r_2})\}}{m}$$

[7] Where does the binomial coefficient $\binom{n+h}{h}$ come from? This is an easy exercise in combinatorics.

[8] This is not a rigorous argument since I did not actually give a lower bound on the number of coefficients and who knows–perhaps the number of coefficients miraculously remains polynomial in the size of the tree. Can you find an example of a tree whose associated polynomial has superpolynomially many coefficients?

We have already seen that the degree of the polynomials is bounded by the size of the trees, so we can easily choose m large enough[9] so that e.g. with probability at least $1/2$, we notice that the trees are non-isomorphic.

Observe that this test has a one-sided error. If the test claims that the trees are non-isomorphic, then we know for a fact that they are (we found a witness for the inequality of the polynomials). However, if the test claims that they are isomorphic, then this may be a mistake caused by an unlucky choice of the values of the x_j. Of course, we can boost the success probability of the algorithm by running it multiple times. If even one of the runs of the algorithm claims that the trees are non-isomorphic, this is enough to say that we know for a fact they are, and the probability that we get an unlucky choice of the x_j every time that we run the algorithm becomes exponentially small with the number of repetitions, as long as the random values of repeated runs are independent from each other.

[9] Recall that for all $m = p^r$ with p a prime there exists a (unique up to isomorphism) finite field of size m. In particular, one can easily do computations in $GF(2^r)$. Alternatively, one could use the variant of the Schwartz-Zippel lemma using a finite subset of an infinite field and do computations over the integers. However, the numbers involved might become quite large in that approach.