Raising Polynomials to Powers

Alex Pan

April 2024

Proofs for correctness of computed numbers:

Because the GPU doesn't report when an integer datatype overflows, we need to verify that the numbers the algorithm returns are actually correct, and less than 2^{64}.

Theorem: Let f be a k-variate polynomial with homogeneous degree n. Let M be the coefficient of f^p. Then, the maximum coefficient of f^p satisfies $M' \leq M^p \cdot \binom{n+k-1}{k-1}^{p-1}$

Proof: Say we have a k-variate polynomial f with homogeneous degree n. Each term of f will look like $cx_1^{d_1}x_2^{d_2}...x_k^{d_k}$, with $\sum_1^k d_i = n$. To make things easier, let $(d_1, d_2, ..., d_k)$ denote the degree of a term from now on.

Let $(d'_1, d'_2, ..., d'_k)$ denote the degree of a term of f^2. To find all unreduced terms that contribute to it, we look at pairs of terms of f with the original degrees $(d'_1 - a_1, d'_2 - a_2, d'_3 - a_3, ..., d'_k - a_k)$ and $(a_1, a_2, a_3, ..., a_k)$, where $a_i > 0$ and $\sum_1^k = n$. The number of these terms is bounded above by the number of weak integer compositions of n into k parts, which is given by $\binom{n+k-1}{k-1}$. The coefficients of each of these resulting terms is bounded above by M^2, and because the number of these is bounded above by $\binom{n+k-1}{k-1}$, we know that the maximum reduced coefficient of f^2 is $M^2 \cdot \binom{n+k-1}{k-1}$

The inductive step is similar. Let $(d'_1, d'_2, ..., d'_k)$ denote the degree of a term of f^p. The pairs of terms that contribute to it come respectively from f^{p-1} and f, so we look again at a term of f^{p-1} with degree $(d'_1 - a_1, d'_2 - a_2, d'_3 - a_3, ..., d'_k - a_k)$, and at a term of f with degree $(a_1, a_2, a_3, ..., a_k)$. Again, there are at most $\binom{n+k-1}{k-1}$ of these, and because the maximum unreduced coefficient is bounded above by $\left(M^{p-1} \cdot \binom{n+k-1}{k-1}^{p-2}\right) \cdot (M)$, we have an upper bound of $M^p \cdot \binom{n+k-1}{k-1}^{p-1}$ for the coefficients. QED

We are interested in two steps: raising $g = f^{p-1} \mod p$ for primes p, and raising g^p for the same p.

- $g = f^{p-1} \mod p$

  If f is a polynomial of 4 variables, then we have an upper bound of $(p-1)^{p-1} \cdot \binom{4+4-1}{4-1}^{p-1}$. This is less than 2^{64} for primes 5 and 7, so we can reduce mod p at the end of our computation without overflowing. (In the raise to $p-1$ case, the OEIS sequence gives a better bound, but I don't know how to prove that the problems are equivalent yet. That bound is actually obtainable)

  If f is a polynomial of 5 variables, then primes 5 and 7 still work for this first step.

- g^p

  If f is a polynomial of 4 variables, and it has been raised to the $p-1$ and taken mod p, then the greatest coefficient of g is $p-1$, and g is homogeneous with degree $4 * p$. This gives the upper bound $(p-1)^p \cdot \binom{4p+4-1}{4-1}^p$. For $p = 5$, this evaluates to $1.00733564 \cdot 10^{17}$, which is less than $2^{64} - 1$, or $1.84467441 \cdot 10^{19}$. So, for the case of 4 variables, and $p = 5$, Int64's can be used without having to worry about overflowing.

Using FFT to raise polynomials to powers

DFT works for multiplying polynomials by evaluating both polynomials at n points, represented by the output vectors $\hat{f}$ and $\hat{g}$. Component-wise multiplication is then performed on $\hat{f}$ and $\hat{g}$, representing the product polynomial evaluated at n points, and IDFT is performed to give the actual coefficients of the polynomial.

If we want to raise a polynomial to a power, we can simply DFT to get $\hat{f}$, then raise each element of $\hat{f}$ to the p-th power, and that will represent f^p evaluated at n points, and IDFT to get the coefficients of f^p.

This should be much faster than the repeated squaring method; this method only requires one DFT to be evaluated, and raising each component to a power in parallel should be effectively instant. In contrast to $log_2(p)$ different DFTs