ON TESTING THE DIVISIBILITY
OF LACUNARY POLYNOMIALS BY CYCLOTOMIC
POLYNOMIALS

MICHAEL FILASETA AND ANDRZEJ SCHINZEL

ABSTRACT. An algorithm is described that determines whether a given poly-
nomial with integer coefficients has a cyclotomic factor. The algorithm is
intended to be used for sparse polynomials given as a sequence of coefficient-
exponent pairs. A running analysis shows that, for a fixed number of nonzero
terms, the algorithm runs in polynomial time.

1. INTRODUCTION AND THE MAIN THEOREMS

This paper describes an algorithm for determining whether a given polynomial
$f(x) \in \mathbb{Z}[x]$ has a cyclotomic divisor. In particular, the algorithm works well when
the number of nonzero terms is small compared to the degree of $f(x)$. This work
is based on a paper by J. H. Conway and A. J. Jones [1]. The specific result we
establish is the following.

Theorem 1. *There is an algorithm that has the following property: given $f(x) =$
$\sum_{j=1}^{N} a_j x^{d_j} \in \mathbb{Z}[x]$ with $N > 1$ and $\deg f = n > 1$, the algorithm determines
whether $f(x)$ has a cyclotomic factor and the algorithm runs in time*

$$(1) \qquad \ll \exp\left((2 + o(1))\sqrt{N/\log N}(\log N + \log\log n)\right)\log(H + 1)$$

as N tends to infinity, where $H = \max_{1 \le j \le N}\{|a_j|\}$.

In the above theorem, we view the input as consisting of a list of N coefficient-
exponent pairs (a_j, d_j) sorted in increasing order by the values of d_j. Observe that
for fixed N, the algorithm runs in polynomial time (time that is polynomial in
$\log n$ and $\log(H + 1)$). In the case that a cyclotomic factor exists, the algorithm
can be made to output a positive integer m for which $\Phi_m(x)$, the mth cyclotomic
polynomial, divides $f(x)$ without affecting the bound given for the running time.

For m a positive integer, let $\zeta_m = e^{2\pi i/m}$. For integers a, b, and m with $m > 0$,
we write $a \equiv b \pmod{m}$ if $m|(a-b)$ and we use the notation $a \bmod m$ to represent
the unique $b \equiv a \pmod{m}$ such that $0 \le b < m$. For $f(x)$, $g(x)$, and $w(x)$ in $\mathbb{Q}[x]$
with $\deg w(x) \ge 1$, we write $f(x) \equiv g(x) \pmod{w(x)}$ if $w(x)|(f(x) - g(x))$, and
we use the notation $f(x) \bmod w(x)$ to denote the unique polynomial $g(x) \equiv f(x)$
$\pmod{w(x)}$ with either $g(x) \equiv 0$ or $0 \le \deg g(x) < \deg w(x)$. If S is a set, we will
denote by $|S|$ the cardinality of S.

Theorem 2. *Let $f(x) \in \mathbb{Z}[x]$ have N nonzero terms. Suppose n is a positive integer such that $\Phi_n(x)|f(x)$. Suppose further that $p_1, p_2, \ldots, p_k$ are distinct primes satisfying*

$$2 + \sum_{j=1}^{k}(p_j - 2) > N.$$

Let e_j be the nonnegative integer for which $p_j^{e_j}||n$. Then for at least one $j \in \{1, 2, \ldots, k\}$, we have $\Phi_m(x)|f(x)$ where $m = n/p_j^{e_j}$.

Proof. We first describe and then make use of Theorem 5 from [1]. For r a positive integer, define $\gamma(r) = 2 + \sum_{p|r}(p - 2)$. Following [1], we call a vanishing sum S minimal if no proper subsum of S vanishes. We will be interested in sums $S = \sum_{j=1}^{t} a_j\omega_j$ where t is a positive integer, each a_j is a nonzero rational number and each ω_j is a root of unity. We refer to the reduced exponent of such an S as the least positive integer r for which $(\omega_i/\omega_1)^r = 1$ for all $i \in \{1, 2, \ldots, t\}$. Theorem 5 of [1] asserts then that if $S = \sum_{j=1}^{t} a_j\omega_j$ is a minimal vanishing sum, then $t \geq \gamma(r)$ where r is the reduced exponent of S. (Also, note that Theorem 5 of [1] implies that the reduced exponent r of a minimal vanishing sum is necessarily squarefree.)

To prove Theorem 2, we suppose as we may that $e_j > 0$ for each $j \in \{1, 2, \ldots, k\}$. We write $f(x) = \sum_{i=1}^{s} f_i(x)$ where each $f_i(x)$ is a nonzero polynomial divisible by $\Phi_n(x)$, no two $f_i(x)$ have terms involving x to the same power, and s is maximal. Thus, each $f_i(\zeta_n)$ is a minimal vanishing sum. For each $i \in \{1, 2, \ldots, s\}$, we write $f_i(x) = x^{b_i}g_i(x^{d_i})$ where b_i and d_i are nonnegative integers chosen so that $g_i(0) \neq 0$ and the greatest common divisor of the exponents appearing in $g_i(x)$ is 1. Then $g_i(\zeta_n^{d_i})$ is a minimal vanishing sum with reduced exponent $n/\gcd(n, d_i)$. If t_i denotes the number of nonzero terms of $g_i(x)$, we deduce from Theorem 5 of [1] that

$$N = \sum_{i=1}^{s} t_i \geq \sum_{i=1}^{s} \gamma\left(\frac{n}{\gcd(n, d_i)}\right)$$

$$\geq 2s + \sum_{j=1}^{k}(p_j - 2)\left|\left\{1 \leq i \leq s : p_j \text{ divides } \frac{n}{\gcd(n, d_i)}\right\}\right|.$$

The inequality in Theorem 2 implies that at least one of the expressions $|\{1 \leq i \leq s : p_j|(n/\gcd(n, d_i))\}|$ is zero. In other words, for some $j \in \{1, 2, \ldots, k\}$ and every $i \in \{1, 2, \ldots, s\}$, we have $p_j^{e_j}|d_i$. Setting $m = n/p_j^{e_j}$ and $d_i' = d_i/p_j^{e_j}$, we obtain that $g_i(\zeta_m^{d_i'}) = 0$. Since $\gcd(m, p_j) = 1$, $\zeta_m^{p_j^{e_j}}$ is a primitive mth root of unity and we deduce $g_i(\zeta_m^{d_i}) = 0$. As this is true for every $i \in \{1, 2, \ldots, s\}$, we conclude $f(\zeta_m) = 0$, establishing the theorem. $\square$

Corollary 1. *Let $f(x) \in \mathbb{Z}[x]$ have N nonzero terms. If $f(x)$ is divisible by a cyclotomic polynomial, then there is a positive integer m such that*

$$2 + \sum_{p|m}(p - 2) \leq N \quad and \quad \Phi_m(x)|f(x).$$

The above is a direct consequence of Theorem 2. Observe that it follows easily from Corollary 1 that if $f(x)$ is divisible by a cyclotomic polynomial, then there is a positive integer m such that every prime divisor of m is $\leq N$ and $\Phi_m(x)|f(x)$.

2. THE PROOF OF THEOREM 1

For the proof of Theorem 1, we will make use of the following preliminary result of independent interest.

Theorem 3. *There is an algorithm that has the following property: given $f(x) = \sum_{j=1}^{N} a_j x^{d_j} \in \mathbb{Z}[x]$ with $\deg f = n > 1$ and a positive integer m together with its factorization $m = \prod_{j=1}^{s} p_j^{e_j}$ where $s \geq 1$, the p_j are distinct primes $\leq M$, and the e_j are positive integers $\leq E$, the algorithm determines whether $\Phi_m(x)$ divides $f(x)$ and the algorithm runs in time*

$$\ll (\log n)^{2+o(1)} + N(\log n)^{1+o(1)} + s\big(\log M + \log(E+1)\big)$$
$$+ N \log N \big(\log m + \log N\big) + 2^s N \big(\log n + \log(H+1)\big),$$

as N tends to infinity, where $H = \max_{1 \leq j \leq N}\{|a_j|\}$.

In the above theorem, we view the input as consisting of a list of coefficient-exponent pairs (a_j, d_j) sorted in increasing order by the values of d_j and a list of prime-exponent pairs (p_j, e_j) sorted in increasing order by the values of p_j. Before giving the proof, we note that the factor $\log m + \log N$ that appears in the fourth term above may be replaced by $\log n$ (which, depending on the size of m, could be an improvement).

Proof of Theorem 3. We suppose as we may throughout that $N > 1$. We begin with the algorithm and then we justify that the algorithm works and has the indicated bound for its running time.

For a polynomial $g(x) \in \mathbb{Z}[x]$, we define $\omega\big(g(x)\big)$ as $g(x) \bmod (x^m - 1)$. We can view $\omega\big(g(x)\big)$ as the polynomial obtained by reducing the exponents of $g(x)$ modulo m and combining the terms with like exponents. In other words, if $g(x) = \sum_{j=1}^{T} u_j x^{v_j}$, then

$$\omega\big(g(x)\big) = \sum_{j=1}^{T} u_j x^{(v_j \bmod m)} = \sum_{j=1}^{T'} u_j' x^{v_j'}$$

where $0 \leq v_1' < v_2' < \cdots < v_{T'}' < m$, $0 \leq T' \leq T$, and each u_j' is a sum of one or more u_j.

For the moment, suppose that m has already been computed from the prime-exponent pairs (p_j, e_j); the running time for computing m will be discussed later. For the time being, we view our asymptotic estimates as holding for T tending to infinity. Later, we will consider values of T which are $\geq N$ so that T will tend to infinity with N. Given $g(x)$ as an ordered list of coefficient-exponent pairs (u_j, v_j), ordered in increasing order by the size of v_j, we compute $\omega\big(g(x)\big)$ as follows. We compute the complete list of pairs $(u_j, v_j \bmod m)$ in

$$\ll T(\log U) + T\big(\log m + \log V\big)^{1+o(1)}$$

binary operations, where $U = 1 + \max_{1 \leq j \leq T}\{|u_j|\}$ and $V = 1 + \max_{1 \leq j \leq T}\{v_j\}$. Next, we sort the pairs in increasing order according to the values of $v_j \bmod m$ and we combine terms of like exponents to form a sorted list of coefficient-exponent pairs (u_j', v_j') associated with $\omega\big(g(x)\big)$; this requires

$$\ll T(\log U) + T(\log T)(\log m + \log T)$$

binary operations. Thus, the total number of binary operations to compute $\omega\big(g(x)\big)$ is bounded by

$$\ll T(\log U) + T\big(\log m + \log V\big)^{1+o(1)} + T(\log T)(\log m + \log T).$$

This analysis will be used only for an initial computation of $\omega\big(f(x)\big)$ in our algorithm.

Our main interest will be in computing $\omega\big(g(x)(x^{m/p} - 1)\big)$ for a given prime p dividing m where $\deg g < m$ and $g(x)$ is given as an ordered list of coefficient-exponent pairs (u_j, v_j), ordered in increasing order by the size of v_j. We suppose that both m and m/p have already been computed. Using a bisection approach, we determine the positive integer k for which $v_k < m - (m/p) \le v_{k+1}$. (If every $v_k < m - (m/p)$, then we take $k = T$.) This computation involves $\ll \log T$ comparisons with the total running time being

$$\ll \log T\big(\log m + \log T\big),$$

where we have used now that $V = 1 + \max_{1 \le j \le T}\{v_j\} \le m$. We first compute $g(x)x^{m/p}$ by forming the sequence of tuples $(u_j, v_j + (m/p) - m)$ for $k+1 \le j \le T$ followed by the tuples $(u_j, v_j + m/p)$ for $1 \le j \le k$. Observe that, in this order, the T tuples for $g(x)x^{m/p}$ are increasing with respect to their second components. The running time for this part is bounded by

$$\ll T\big(\log m + \log U + \log T\big).$$

We now combine this sequence of tuples with the sequence of tuples for $-g(x)$, namely $(-u_j, v_j)$ for $1 \le j \le T$. We do this by forming a new list of ordered pairs that starts with the smaller (with respect to second components) of $(-u_1, v_1)$ and $(u_{k+1}, v_{k+1} + (m/p) - m)$ (and simply $(-u_1, v_1)$ if $k = T$) or, in the case that $v_1 = v_{k+1} + (m/p) - m$, with $(-u_1 + u_{k+1}, v_1)$. Similarly, the new list of ordered pairs will end with the larger of $(-u_T, v_T)$ and $(u_k, v_k + m/p)$ or, in the case that $v_T = v_k + m/p$, with $(-u_T + u_k, v_T)$. This can be accomplished by passing through each element of both sequences of tuples in increasing order with running time

$$\ll T\big(\log m + \log U + \log T\big).$$

We deduce that computing $\omega\big(g(x)(x^{m/p} - 1)\big)$ can be accomplished in

$$\ll T\big(\log m + \log U + \log T\big)$$

binary operations. We use the above to describe and justify the running time of the following algorithm.

Algorithm A: *(Specific cyclotomic factor test):* Given $f(x) = \sum_{j=1}^{N} a_j x^{d_j} \in \mathbb{Z}[x]$ with $\deg f = n > 1$ and $m = \prod_{j=1}^{s} p_j^{e_j}$ as in the statement of the theorem, determine whether $\Phi_m(x)$ divides $f(x)$.

Step A1: *Check the size of $\phi(m)$.* Check whether

$$\prod_{j=1}^{s} p_j^{e_j - 1}(p_j - 1) \le n.$$

If the inequality holds, then proceed to Step A2. Otherwise, output that $\Phi_m(x)$ does not divide $f(x)$.

Step A2: *Reduce exponents of f modulo m.* Compute

$$f_0(x) = \omega\big(f(x)\big)$$

where the function ω is defined above.

Step A3: *Multiply by $x^{m/p} - 1$ for each p dividing m.* For $j \in \{1, 2, \ldots, s\}$, recursively define

$$f_j(x) = \omega\big(f_{j-1}(x)\big(x^{m/p_j} - 1\big)\big).$$

Step A4: *Check whether the final result is zero.* Check whether $f_s(x) \equiv 0$. If $f_s(x) \equiv 0$, then output that $\Phi_m(x)$ divides $f(x)$. Otherwise, output that $\Phi_m(x)$ does not divide $f(x)$.

We justify now the correctness of the algorithm. Observe that the degree of $\Phi_m(x)$ is

$$\phi(m) = \prod_{j=1}^{s} p_j^{e_j-1}(p_j - 1).$$

Thus, if $\phi(m) > n$, then $\Phi_m(x)$ does not divide $f(x)$. It remains to consider then the case that $\phi(m) \leq n$. We use that the factorization of $x^m - 1$ into irreducible polynomials over the rationals is given by

$$x^m - 1 = \prod_{d|m} \Phi_d(x).$$

Observe that every divisor d of m with $d \neq m$ divides m/p_j for some $j \in \{1, 2, \ldots, s\}$. Thus, for each such d, the polynomial $\Phi_d(x)$ divides

$$h(x) = \prod_{j=1}^{s} \big(x^{m/p_j} - 1\big).$$

On the other hand, $\Phi_m(x)$ does not divide $h(x)$. We deduce that $h(x)$ is not divisible by $x^m - 1$, but $\Phi_m(x)h(x)$ is. This implies that $\Phi_m(x)$ divides $f(x)$ if and only if $x^m - 1$ divides $f(x)h(x)$. From Steps A2 and A3, we see that $f_s(x)$ is $f(x)h(x) \bmod (x^m - 1)$. Therefore, $\Phi_m(x)$ divides $f(x)$ if and only if $f_s(x) \equiv 0$. Step A4 and the correctness of the algorithm are justified.

For an upper bound on the running time of the algorithm, we return now to asymptotic estimates that hold for N tending to infinity. We begin in Step A1 by reading the data and checking the sizes of s, the primes p_j, and the exponents e_j to determine if the inequality in Step A1 does not hold due to any of these being too large. The inequality can only hold if

$$s \leq 1 + \lg n, \quad p_j \leq n + 1, \quad \text{and} \quad e_j \leq 1 + \lg n,$$

where $\lg x$ denotes the logarithm function to the base 2 and the bounds involving j are for every $j \in \{1, 2, \ldots, s\}$. These bounds are weaker than they need to be, but they will serve for our purposes. To read the data and to check these bounds requires

$$\ll \big(\log(H + 1) + \log n\big)N + \big(\log M + \log(E + 1) + \log n\big)s$$

binary operations. If any of the bounds does not hold, neither does the inequality in Step A1 (and so an output would be given that $\Phi_m(x)$ does not divide $f(x)$ and no further time would be required). We assume therefore that the above bounds hold for s and each p_j and e_j. In particular, we consider from this point on only the case that $s \leq 1 + \lg n$, $M \leq n + 1$ and $E \leq 1 + \lg n$.

We continue to estimate the amount of time needed to check the inequality in Step A1. We view this as being done as follows. Set a variable, say A, to be 1. Consider in turn each p_j beginning with $j = 1$ and ending with $j = s$. For each such j, replace the value of A with the value of $A \times p_j$ and do this $e_j - 1$ times. Then replace the value of A with the value of $A \times (p_j - 1)$ before continuing to the next value of j. After each multiplication, check if $A \leq n$. Each multiplication will take $\mathcal{O}\big((\log n + \log M)^{1+o(1)}\big)$ binary operations. At most $\mathcal{O}(\log n)$ multiplications are necessary (before obtaining $A > n$). Hence, completing Step A1 requires

$$\ll (\log n + \log M)^{2+o(1)} \ll (\log n)^{2+o(1)}$$

binary operations. Observe that further steps in the algorithm are only considered if $\phi(m) \leq n$. In particular, $m \leq 2\phi(m)^2 \leq 2n^2$ implies $\log m \leq 3 \log n$. We therefore suppose this holds in discussing the remaining steps of the algorithm.

A procedure analogous to that just described for determining whether the inequality in Step A1 holds can be used for computing m (as well as m/p_j) from the list of exponent pairs (p_j, e_j). Given that now $m \leq 2n^2$, computing m (or m/p_j) requires

$$\ll (\log n + \log M)^{2+o(1)} \ll (\log n)^{2+o(1)}$$

binary operations.

Given the running time analysis for computing $\omega\big(g(x)\big)$, Step A2 takes

$$\ll N \log(H + 1) + N(\log n)^{1+o(1)} + N \log N \big(\log m + \log N\big)$$

binary operations. In Step A3, for $j \in \{1, 2, \ldots, s\}$, we now have a polynomial $f_{j-1}(x)$ which is of degree $\leq m$ and are wanting to obtain the value of $\omega\big(f_{j-1}(x)(x^{m/p_j} - 1)\big)$. Inductively, we obtain that $f_j(x)$ has $\leq 2^j N$ nonzero terms, with each coefficient $\leq 2^j NH$ and each exponent $\leq m$. By our earlier analysis for computing $\omega\big(g(x)(x^{m/p} - 1)\big)$, computing $\omega\big(f_{j-1}(x)\big)$ takes

$$\ll 2^j N \big(j + \log n + \log N + \log(H + 1)\big)$$

binary operations. Summing over j gives an upper bound of

$$\ll 2^s N \big(s + \log n + \log N + \log(H + 1)\big)$$

for the total number of binary operations to perform Step A3. On the other hand, the number of nonzero terms of $f(x)$ cannot exceed one more than its degree so that $N \leq n + 1$. Recalling that $s \ll \log n$, Step A3 takes

$$\ll 2^s N \big(\log n + \log(H + 1)\big)$$

binary operations.

The check in Step A4 to determine if $f_s(x)$ is an empty coefficient-exponent pair list can be completed as the computation for $f_s(x)$ is done in Step A3. Thus, no further work is needed in Step A4 (except to output the result).

The result of the theorem follows.   □

Proof of Theorem 1. Corollary 1 implies that we need only consider the possibility that $\Phi_m(x)|f(x)$ where each prime divisor of m is $\leq N$, the number of nonzero terms of $f(x)$. For each prime $p \leq N$, we consider

$$(2) \qquad\qquad r(p) = \left[\frac{\log \deg f}{\log p}\right] + 1.$$

Observe that if $p^e | m$, then

$$\deg f(x) \geq \deg \Phi_m(x) = \phi(m) \geq \phi(p^e) = p^{e-1}(p - 1) \geq p^{e-1}$$

so that $e \leq r(p)$. We make use of this notation for describing an algorithm for proving Theorem 1.

> **Algorithm B:** *(General cyclotomic factor test):* Given $f(x) = \sum_{j=1}^{N} a_j x^{d_j} \in \mathbb{Z}[x]$ with $\deg f = n$, determine whether there is at least one m such that $\Phi_m(x)$ divides $f(x)$.
>
> **Step B1:** *Determine relevant primes.* Compute the set $P = \{p_1, p_2, \ldots, p_r\}$ of all primes $\leq N$.
>
> **Step B2:** *Obtain exponent bounds.* Compute $B_j = r(p_j)$ (defined by (2)) for $1 \leq j \leq r$.
>
> **Step B3:** *Determine relevant elements.* Compute the subsets $\{q_1, q_2, \ldots, q_s\}$ of P for which

$$2 + \sum_{j=1}^{s} (q_j - 2) \leq N. \tag{3}$$

> Let $\mathcal{Q}$ denote the set of all such subsets.
>
> **Step B4:** *Compute possible cyclotomic factors.* Construct a list of tuples

$$\big((q_1, e_1), (q_2, e_2), \ldots, (q_s, e_s)\big)$$

> where $\{q_1, q_2, \ldots, q_s\} \in \mathcal{Q}$ and if $q_i = p_j$, then $1 \leq e_i \leq B_j$.
>
> **Step B5:** *Check divisibility.* For each $\big((q_1, e_1), \ldots, (q_s, e_s)\big)$ in Step B4, apply Algorithm A with $m = q_1^{e_1} q_2^{e_2} \cdots q_s^{e_s}$ to determine whether $\Phi_m(x)$ divides $f(x)$. If at least one such m exists, indicate that $f(x)$ has a cyclotomic factor. Otherwise, indicate that $f(x)$ has no cyclotomic factor.

Given the algorithm above, we need to justify the correctness of the algorithm and the appropriate running time. Corollary 1 implies that $f(x)$ is divisible by a cyclotomic polynomial if and only if $\Phi_m(x) | f(x)$ for some positive integer m such that the complete set of prime divisors of m is an element of $\mathcal{Q}$. As indicated after (2), if $p_j^e || m$, then $e \leq r(p_j) = B_j$. It follows that $f(x)$ is divisible by a cyclotomic polynomial if and only if $\Phi_m(x) | f(x)$ for some positive integer m considered in Step B5. Thus, the correctness of the algorithm is justified.

Next, we justify the bound on the running time indicated for the algorithm. The first two steps indicated in the algorithm can be estimated rather poorly without affecting this bound. In Step B1, the primes that are $\leq N$ are determined. A crude upper bound on the number of binary operations for this step is $\mathcal{O}(N^2)$. Note that r is determined by the condition $p_r \leq N < p_{r+1}$. Hence, by the Prime Number Theorem, $r \sim N / \log N$. Computing any particular B_j takes no more than $\mathcal{O}(\log^3 n)$ binary operations so that the running time for Step B2 is $\ll r \log^3 n \ll N \log^3 n / \log N$.

The running time for the algorithm is affected largely by the number of elements of $\mathcal{Q}$. From (3) and $r \sim N / \log N$, we deduce that

$$\sum_{j=1}^{s} q_j \leq (1 + o(1)) N \tag{4}$$

as N tends to infinity. Using that $\sum_{p \leq z} p \sim z^2/(2 \log z)$, we obtain from (4) that

$$s \leq (2 + o(1))\sqrt{N/\log N}$$

(take $z = (1 + \varepsilon)\sqrt{N \log N}$ and use that $\pi(z) \sim z/\log z$). Let K denote this bound on s. Since there are r choices for each prime q_j, we deduce that

$$|\mathcal{Q}| \leq r^K \leq N^K = \exp\left((2 + o(1))\sqrt{N \log N}\right).$$

In Step B3, the elements of $\mathcal{Q}$ are determined. This can be done by considering increasing values of s beginning with $s = 1$ and determining those subsets of P of size s that belong to $\mathcal{Q}$. For each of the $\leq r^s$ subsets of P of size s, checking (3) takes $\ll s \log N$ binary operations. Once a value of s is obtained for which no subsets of P of size s belong to $\mathcal{Q}$, the set $\mathcal{Q}$ will be determined and Step B3 ends. By the definition of K, there are no subsets of P of size $K + 1$ in $\mathcal{Q}$. It follows that the number of binary operations needed for Step B3 is

$$\ll \sum_{s=1}^{K+1} r^s s \log N \ll r^{K+1} K \log N \ll \exp\left((2 + o(1))\sqrt{N \log N}\right).$$

For Step B4, we observe that $B_1 \leq 1 + 2 \log n$ and $B_j \leq 1 + \log n$ for $j > 1$. For each of the $\leq r^K$ elements $\{q_1, q_2, \ldots, q_s\}$ of $\mathcal{Q}$, we therefore form at most $2(1 + \log n)^s \leq 2(1 + \log n)^K$ tuples $\big((q_1, e_1), \ldots, (q_s, e_s)\big)$. It follows that there are

$$\ll r^K (1 + \log n)^K \ll \exp\left(K(\log r + \log(1 + \log n))\right)$$

such tuples. Note that we are interested in asymptotics as N (and, hence, n) tends to infinity so that, in particular, $\log(1 + \log n) \sim \log \log n$. Forming each tuple $\big((q_1, e_1), \ldots, (q_s, e_s)\big)$ takes

$$\ll K(\log N + \log \log n)$$

binary operations. Therefore, we can use

$$\mathcal{O}\big(\exp\left((1 + o(1))K(\log r + \log \log n)\right)\big)$$

as an upper bound on the running time for Step B4.

The running time for Step B5 is determined from the running time for Algorithm A as given by Theorem 3. Observe that we can take $M = N$ in Theorem 3 as N serves as a bound for every q_j. Also, $M = N \leq n$, $E \ll \log n$, and $s \leq K \ll \sqrt{N}$. Given the number of tuples considered in Step B4, the running time for Step B5 is

$$\ll 2^K N \big(\log^{2+o(1)} n + \log(H + 1)\big) \exp\left((1 + o(1))K(\log r + \log \log n)\right).$$

Using $r \sim N/\log N$ and $K = (2 + o(1))\sqrt{N/\log N}$, the running time in Step B5 is bounded by

$$\ll \exp\left((2 + o(1))\sqrt{N/\log N}(\log N + \log \log n)\right) \log(H + 1)$$

as N tends to infinity. As this exceeds the bounds obtained for the running times in the previous steps of the algorithm, it also serves as a bound for the order of magnitude of the running time of the entire algorithm, completing the proof. $\square$

References

[1] J. H. Conway and A. J. Jones, *Trigonometric diophantine equations (On vanishing sums of roots of unity)*, Acta Arith. **30** (1976), 229–240. MR **54:**10141

Department of Mathematics, University of South Carolina, Columbia, South Carolina 29218

E-mail address: `filaseta@math.sc.edu`

Institute of Mathematics of the Polish Academy of Sciences, P.O. Box 137, ul. Śniadeckich 8, 00-950 Warszawa 10, Poland

E-mail address: `a.schinzel@impan.gov.pl`