RAIRO-Theor. Inf. Appl. **42** (2007) 309–322
DOI: 10.1051/ita:2007024

Available online at:
www.rairo-ita.org

CENSUS ALGORITHMS FOR CHINESE REMAINDER PSEUDORANK

David Laing[1] and Bruce Litow[1]

Abstract. We investigate the density and distribution behaviors of the chinese remainder representation pseudorank. We give a very strong approximation to density, and derive two efficient algorithms to carry out an exact count (census) of the bad pseudorank integers. One of these algorithms has been implemented, giving results in excellent agreement with our density analysis out to 5189-bit integers.

Mathematics Subject Classification. 11Y99, 68R99.

1. Introduction

Motivation

CRR, chinese remainder representation, has been used as an important tool in parallel arithmetic complexity research. CRRS, CRR systems, have been used in studying parallel complexity of integer division, and also in connection with certain arithmetic circuit complexity problems. See [2,3,10]. A key concept, the CRR pseudorank, introduced in [3], has proved to be very useful in developing $\mathbf{NC}^1$ algorithms for integer comparison. Unlike integer comparison using radix notation, which can be implemented on a DFA (deterministic finite automaton), comparison in CRR is difficult since there is no evident correspondence between ordering and the notation.

Despite its importance in CRR-related research, there appears to be little information on how the pseudorank behaves over a given CRRS. The pseudorank of integer x is known to be either equal to or one less than the rank of x. Rank is a basic CRR function which we will define later on. Knowledge of the rank makes

Keywords and phrases. Chinese remainder representation, rank, pseudorank, pseudorank census algorithms.

[1] School of Information Technology, James Cook University, Townsville, Qld. 4811, Australia; bruce@cs.jcu.edu.au, david.laing1@jcu.edu.au

Article published by EDP Sciences

CRR-intrinsic comparison quite simple. Unfortunately, no efficient CRR-intrinsic method of computing rank is known. This paper gives the first clear indication of how pseudorank compares with rank over a CRRS. In particular, we give a sharp estimate of the density of integers for which rank and pseudorank differ, and show that there are subintervals of a CRRS where either all rank and pseudorank match, or differ. This is the first step in determining how rank-pseudorank mismatches are distributed in a CRRS. We also describe two efficient algorithms for exactly counting the number of rank-pseudorank mismatches. One of these algorithms has been implemented, and results on density based on running it are presented for various CRRS, up to one capable of representing 5189-bit integers. These experimental results are in excellent agreement with our density estimate. Interestingly, our results and algorithms make no essential use of number theory. We hope that the more difficult problem of characterizing the distribution of rank-pseudorank mismatches, which is likely to require number theoretic and other deeper considerations, will be taken up by others. The significance of distribution information will be discussed in Section 2.

This work grows out of an earlier effort, the results of which are summarized in [8]. The algorithm described in that earlier work only gives an estimate of the number of rank-pseudorank msimatches, and although running in polynomial time is more complex and much slower than the newer algorithms described in this paper. However, elements of the original approach can still be found in the implemented algorithm (Algorithm 2) of this paper. The convolution approach to iterated polynomial multiplication that forms the basis of the original algorithm may be of independent interest to some readers. The source for this original implementation is available on request.

Structure of the paper

Section 2 defines CRR and pseudorank, and concludes with some results on rank-pseudorank match and mismatch distribution and density. Section 3 contains the descriptions and analysis of two new census algorithms, and includes tabulated experimental results from an implementation of one of them.

2. CRR and pseudorank

2.1. CRR basic facts

A CRRS can be specified by a single positive integer $P = p_1 \cdots p_r$, where $p_1, \ldots, p_r$ are consecutive odd primes. We will fix $p_1 = 3$. We will simply refer to P as a CRRS, also using CRRS to mean the integers $x < P$. The CRR of an integer x (integer will mean nonnegative integer) is the list $|x|_{p_1}, \ldots, |x|_{p_r}$, where $|x|_z = y$ means that y is the least integer such that $x \equiv y \bmod z$. The weak chinese remainder theorem asserts that each $0 \leq x < P$ is uniquely identified by its CRR. We write $|x|_P$ to indicate that the integer x is given in CRR, rather than in radix notation. On the other hand, $|x|_{p_i}$ indicates that radix notation is used.

The reason for the difference is that by Lemma 2.1, $|x|_{p_i} < \log P$, so its radix notation involves at most about $\log \log P$ bits. Notice that log is the base 2 logarithm, and ln the natural logarithm.

The following two results give some quantitative information about CRRS.

Lemma 2.1. *If $n > 2.89 \times 10^7$, and r is the largest integer such that $p_r < n$, then $r = \Theta(n/\log n)$, and $\exp(n + .001 \times n) > P > 2^n$.*

Proof. The claim on r follows from the prime number theorem. See [5]. The claim for P follows from the bound ($n > 2.89 \times 10^7$)

$$\left| \sum_{i=1}^{r} \ln p_i - n \right| \leq .0068 \cdot n/\ln n.$$

See [4]. From this bound we get

$$\exp(n + .0068n/(1.44 \cdot \log n)) > P > \exp(n - .0068n/(1.44 \cdot \log n)),$$

since $\log n > 1.44 \cdot \ln n$. It is straightforward that $\exp(n + .001 \times n) > P > 2^n$. □

From this point we take the view that we are interested in choosing a CRR that can represent all integers below 2^n. That is, we use n bits as the problem size. By Lemma 2.1, we have that $\log P = \Theta n$ can be imposed, and Θ-notation implies a small range for constants. We could actually obtain smaller values for r yielding $P > 2^n$, and while in practice this could be significant, for our study such a savings is irrelevant. All of our computations are understood to be in the Turing model. By polynomial time, then, we mean $n^{O(1)}$ time on a Turing machine, which is equivalent by Lemma 2.1 to $\log^{O(1)} P$ Turing machine time.

Lemma 2.2.

$$\sum_{i=1}^{r} 1/p_i = O(\log \log n).$$

Proof. See exercise II.9.d in [11]. □

2.2. An observation about parity and comparison in CRR

We first note that given the CRR for $x, y < P$, it is trivial to obtain $|x \diamond y|_P$, where $\diamond$ is addition, subtraction or multiplication. This follows from noting that $|x \diamond y|_{p_1}, \ldots, |x \diamond y|_{p_r}$ is the CRR of $|x \diamond y|_P$. Indeed, this is the chief reason why CRR plays an important role in analyzing parallel complexity of arithmetic operations.

Observe that knowing $|x|_2$ generally makes possible very efficient integer comparison algorithms. First, note that if $x < y < P$, then $|x - y|_P = x - y + P$. This is the "wrap" effect that complicates doing arithmetic in CRR. For example,

if $x = 1$ and $y = 2$, we have $|1 - 2|_P = P - 1$. Of course, if $y < x < P$, then $|x - y|_P = x - y$. Next, we have that if $x < y$,

$$||x - y|_P|_2 \equiv |x - y + P|_2 \equiv |x|_2 + |y|_2 + 1 \bmod 2,$$

since $|P|_2 = 1$. This is why in our research, $p_1 = 3$ and not $p_1 = 2$. On the other hand, if $y < x$,

$$||x - y|_P|_2 = |x - y|_2 \equiv |x|_2 + |y|_2 \bmod 2.$$

Thus we can conclude for $x, y < P$, that $x > y$ iff $||x - y|_P|_2 \equiv |x|_2 + |y|_2 \bmod 2$. If parity were easy to compute in CRR (i.e. without some conversion into radix), CRR-intrinsic comparison would also be easy. There appear to be serious obstacles to CRR-intrinsic comparison. For example, see Theorem S, p. 255 in [6].

The next theorem is a strong form of the chinese remainder theorem. Recently, it has been rediscovered in [1]. It is the starting point for the parallel arithmetic work in [2,3].

Theorem 2.3. *There exists a unique integer, $q(x) < r$, such that*

$$x + q(x) \cdot P = \sum_{i=1}^{r} |x \cdot (P/p_i)^{p_i - 2}|_{p_i} \cdot P/p_i \,. \tag{1}$$

Proof. The RHS of equation (1) and x give equal residues $\bmod p_i$, for $i = 1, \ldots, r$. By the chinese remainder theorem we have that the RHS and x differ by a factor of P. Dividing the RHS by P we see that each resulting summand is less than 1, thus the RHS is less than $r \cdot P$. We can conclude that the RHS can be written as $x + q \cdot P$, and letting $q(x) = q$, which is clearly unique, we have the theorem. □

The integer $q(x)$ is called the rank of x. Given the rank, it is easy to compute parity from CRR data. Observe that from equation (1) we have

$$|x|_2 = \left| \sum_{i=1}^{r} x_i + q(x) \right|_2 , \tag{2}$$

where

$$x_i = |x \cdot (P/p_i)^{p_i - 2}|_{p_i},$$

and we have used $|P|_2 = 1$. The integers x_i are all available via CRR, so to compute $|x|_2$ it suffices to know $|q(x)|_2$. However, there is no known way to efficiently compute $|q(x)|_2$ in a CRR intrinsic manner. This difficulty is the motivation for introducing the pseudorank of x. Pseudorank will be defined in such a way that it is easy to compure it directly from CRR data. In fact, pseudorank can be computed by a finite automaton that can be constructed in polynomial time. This will be brought out in Section 3.

2.3. The pseudorank

Dividing through by P, from Theorem 2.3, we can write

$$x/P + q(x) = \sum_{i=1}^{r} x_i/p_i. \tag{3}$$

Let g be the least integer such that $2^g > 4r$. We define integers $\alpha(x), \beta(x)$ by $\alpha(x) < 2^g$ such that

$$2^g \cdot \beta(x) + \alpha(x) = \sum_{i=1}^{r} \lfloor 2^g \cdot x_i/p_i \rfloor. \tag{4}$$

Observe that

$$0 \le x_i/p_i - \lfloor 2^g \cdot x_i/p_i \rfloor / 2^g < 1/2^g.$$

Thus, to g bits precision, $\beta(x) + \alpha(x)/2^g$ mimics $q(x) + x/P$. We call $\beta(x)$ the pseudorank of x.

The values $\lfloor 2^g \cdot x_i/p_i \rfloor$ can be precomputed, based solely on n, and the consequent choice of P, and stored in a table of diemensions $L \times 2^g$, where $L = \sum_{i=1}^{r} p_i$. We regard this table as indexed for each $i = 1, \ldots, r$ by x_i. We will see in Section 3 that a polynomial time constructible finite automaton can use CRR as input, and properly index into this table. Since $L < r \cdot p_r < r \cdot n$, and $2^g < 8r$, our table has size less than $8r^2 \cdot n \cdot O(\log n) = o(n^3)$ by Lemma 2.1, and the fact that an entry has size at most $g = O(\log r) = O(\log n)$ bits. There is nothing remotely like this economy of data known for computing the rank.

The next result is proved in [3,10].

Theorem 2.4. *If $x > P/4$, then $\beta(x) = q(x)$. If $\beta(x) \neq q(x)$, then $\beta(x) = q(x) - 1$.*

The integers below $P/4$ comprise the critical region of the CRRS. The integer x is said to be good if $q(x) = \beta(x)$, otherwise it is said to be bad. By Theorem 2.5, all bad integers are in the critical region.

We point out that the choice of $4r < 2^g < 8r$ is somewhat arbitrary, in that similar results concerning good and bad integers and the corresponding critical region could be obtained provided:

- $2^g > 2r$ (if $2^g < 2r$ the error in pseudorank compared to rank is too large to be of use), and
- $g = n^{O(1)}$ (otherwise we lose the polynomial size of the table, and other polynomial time bounds for our finite automaton constructions).

Theorem 2.6 shows that the concept of bad density makes sense for any choice $2^\ell \cdot r < 2^g < 2^{\ell+1} \cdot r$, subject to the two items just mentioned. First, it is straightforward to generalize Theorem 2.5 to show that if $2^\ell \cdot r < 2^g < 2^{\ell+1} \cdot r$, then all bad integers are below $P/2^\ell$. Next, the proof of Theorem 2.6 tells us that the ratio of census of bad integers to P is essentially $r/2^{g+1}$. From this and the

upper bound of $P/2^\ell$ on bad integers it follows that bad density is always in the range between $1/4$ and $1/2$.

Our choice $4r < 2^g < 8r$ to leads to some technical simplifications over the "minimal" choice of $2r < 2^g < 4r$ in applications not covered in this paper. The reader may want to consult [1] on the choice $2r < 2^g < 4r$.

To see the relationship between rank and pseudorank for our choice of g, consider a small CRR given by $P = 3 \cdot 5 \cdot 7 = 105$. Here $r = 3$, so $g = 4$ will do since $2^4 = 16 > 4 \cdot 3 = 12$. First choose $x = 14$, by direct calculation of equation (1), we get

$$\sum_{i=1}^{3} (14)_i \cdot 105/p_i = 17 = 14 + q(14) \cdot 105.$$

From this we get $q(14) = 1$. Next, we compute $\beta(14)$ *via*

$$\sum_{i=1}^{3} \lfloor 16 \cdot (14)_i/p_i \rfloor = 16 \cdot \beta(14) + \alpha = 17.$$

It follows that $\beta(14) = 1 = q(14)$, so 14 is good. On the other hand, we get $q(2) = 1$, but

$$\sum_{i=1}^{3} \lfloor 16 \cdot (2)_i/p_i \rfloor = 16 \cdot \beta(2) + \alpha = 15.$$

This means that $\beta(2) = 0$, so 2 is bad.

Let B denote the bad integers. We write $\sum_x$ for $\sum_{x=0}^{P-1}$. The (bad) census is just the indicator sum $\sum_{x \in \mathcal{B}} 1$.

2.4. Distribution and density results

We can say something about the distribution of good and bad integers in extreme parts of the critical region.

Theorem 2.5. *If $x < \frac{r \cdot P}{n \cdot 2^g}$, and x and P are coprime, then x is bad. If $x > P/4 - P/2^g$, then x is good.*

Proof. We treat the bad integer case. We derive a contradiction by assuming $x < \frac{r \cdot P}{n \cdot 2^g}$ and x is good. Let a be the integer for which $x/P = a/2^g + \epsilon$ such that $0 \le \epsilon < 1/2^g$. Since

$$x/P < \frac{r}{n \cdot 2^g} < 1/n < 1/2^g,$$

we have that

$$\epsilon = x/P < \frac{r}{n \cdot 2^g},$$

and $a = 0$. Since x and P are coprime, $x_i \neq 0$ for $i = 1, \ldots, r$. Let ϵ_i be

$$x_i/p_i - \lfloor 2^g \cdot x_i/p_i \rfloor / 2^g.$$

Note that $0 \leq \epsilon_i$. Also by the upper bound on ϵ and nonnegativity of the ϵ_i, for at least one i,

$$\epsilon_i < \frac{1}{n \cdot 2^g}. \tag{5}$$

It is clear that $x_i \neq 0$ implies that

$$0 < \epsilon_i \cdot p_i \cdot 2^g = x_i \cdot 2^g - \lfloor 2^g \cdot x_i / p_i \rfloor \cdot p_i.$$

On the other hand, if ϵ_i satisfies equation (5), then since $p_i \leq p_r < n$ by Lemma 2.1,

$$0 < x_i \cdot 2^g - \lfloor 2^g \cdot x_i / p_i \rfloor \cdot p_i < 1,$$

which is impossible.

We treat the good integer case, also by contradiction. This result is really a slight refinement of Theorem 2.4. Since by Theorem 2.4, $x > P/4$ implies that x is good, we can assume that $P/4 > x > P/4 - P/2^g$, and that x is bad. By equation (4), Theorem 2.4 and the fact that

$$0 \leq \epsilon_i = x_i/p_i - \lfloor 2^g \cdot x_i / p_i \rfloor / 2^g < 1/2^g,$$

we have

$$1 + x/P - \sum_{i=1}^{r} \epsilon_i = \alpha(x)/2^g.$$

But, this is impossible because $\alpha(x)$ is an integer less than 2^g while

$$\sum_{i=1}^{r} \epsilon_i < r/2^g < 1/4,$$

and $1/4 - 1/2^g < x/P < 1/4$. □

The density of bad integers is defined to be the ratio of the number of bad integers to $P/4$.

Theorem 2.6. *The density of bad integers is*

$$r/2^{g-1} - O(\log^2(n)/2^g).$$

Proof. By Theorem 2.4, the bad integer census, written explicitly as the difference of two sums is

$$\sum_x q(x) - \sum_x \beta(x). \tag{6}$$

We develop expressions for the two sums in equation (6). From equation (3),

$$\sum_x q(x) = \sum_x \sum_{i=1}^{r} x_i/p_i - \sum_x x/P. \tag{7}$$

We write, using twice that for integers a, b, $\lfloor a/b \rfloor = a/b - |a|_b/b$,

$$\beta(x) = \left(\sum_{i=1}^{r} (2^g \cdot x_i - |2^g \cdot x_i|_{p_i})/p_i - \alpha(x) \right) /2^g.$$

Since 2^g and p_i are coprime, multiplying each element of $\{0, 1, \ldots, p_i - 1\}$ by 2^g and reducing modp_i results in $\{0, 1, \ldots, p_i - 1\}$, we have

$$\sum_x \beta(x) = \sum_x \sum_{i=1}^{r} x_i/p_i - \frac{1}{2^g} \sum_x \sum_{i=1}^{r} x_i/p_i - \sum_x \alpha(x)/2^g \ .$$

Thus, the census of bad integers can be written as

$$\frac{1}{2^g} \sum_x \sum_{i=1}^{r} x_i/p_i + \frac{1}{2^g} \sum_x \alpha(x) - \sum_x x/P. \tag{8}$$

Let

$$2^g \cdot x_i/p_i = \lfloor 2^g \cdot x_i/p_i \rfloor + \delta_i(x).$$

We can then write

$$\sum_{i=1}^{r} \lfloor 2^g \cdot x_i/p_i \rfloor = \sum_{i=1}^{r} 2^g \cdot x_i/p_i - \sum_{i=1}^{r} \delta_i(x),$$

and using equation (3) we get

$$\sum_{i=1}^{r} \lfloor 2^g \cdot x_i/p_i \rfloor = 2^g \cdot x/P + 2^g \cdot q(x) - \sum_{i=1}^{r} \delta_i(x).$$

Since

$$\alpha(x) = \left| \sum_{i=1}^{r} \lfloor 2^g \cdot x_i/p_i \rfloor \right|_{2^g},$$

we get

$$\alpha(x) = 2^g \cdot x/P - \sum_{i=1}^{r} \delta_i(x). \tag{9}$$

From equations (8) and (9) we can express the bad integer census as

$$\frac{1}{2^g} \sum_x \sum_{i=1}^{r} x_i/p_i - \frac{1}{2^g} \sum_x \sum_{i=1}^{r} \delta_i(x). \tag{10}$$

We analyze the error term

$$\Delta = \sum_x \sum_{i=1}^{r} \delta_i(x)$$

by first estimating $\Delta_i = \sum_x \delta_i(x)$. The possible values for x_i as x ranges from 0 to $P-1$ are, of course, $0, \ldots, p_i - 1$. Each value occurs P/p_i times. Consider the values $k, k+1, \ldots, k+h$ such that

$$\ell < 2^g \cdot k/p_i < \cdots < 2^g \cdot (k+h)/p_i < \ell + 1.$$

An upper bound on the sum of the fractional parts of these fractions is clearly

$$\sum_{j=1}^{h+1} j/p_i.$$

Thus, $\Delta_i < \frac{P}{p_i} \cdot \sum_{j=1}^{h+1} j/p_i$. Also, we have that

$$h < p_i/2^g = O(\log n).$$

It follows that

$$\Delta_i = O(P \cdot \log^2 n/p_i^2).$$

From this we have

$$\Delta = \sum_{i=1}^{r} \Delta_i = O(P \cdot \log^2(n)),$$

since $\sum_{k=1}^{\infty} 1/k^2 = \Theta(1)$.

It is straightforward to show that

$$\sum_x x_i/p_i = 1/2 \cdot (P - P/p_i).$$

Note that $\sum_{i=1}^{r} P/p_i = O(P \cdot \log\log n)$ by Lemma 2.2. Combining this, equation (10), the upper bound on Δ and dividing by $P/4$ to obtain a density, we have the theorem. □

We make two observations about Theorem 2.6. First, the error bound goes to zero rapidly since $2^g = \Theta(n/\log n)$. Second, asymptotically, the density of bad integers oscillates in sawtooth fashion between 1/4 at r a power of 2, and 1/2 at r one less than a power of 2.

3. Two pseudorank census algorithms

3.1. WFA preliminaries

We use weighted finite automata (WFA) to describe the algorithms of this section. WFA is a huge topic. Examples of directions in WFA research are presented in [7,9]. The first general algebraic treatment of WFA appears to be due to Schützenberger. We deal with only those aspects that are relevant to our algorithms in this section.

We work over CRRS P as before. For our CRR oriented purposes a WFA, F, is a tuple Σ, U, V, X_σ, where Σ is a finite alphabet, U is a $1 \times h$ matrix, V is a $h \times 1$ matrix, and for each $\sigma \in \Sigma$, X_σ is a $h \times h$ matrix. All matrix entries are rational coefficient polynomials in the variable t. We regard the indices of these matrices as states. The integer h is the referred to as the number of states. In all of our WFA, h will be bounded above as $n^{O(1)}$. It is sometimes useful to think of the nonzero entries of U as "start" states, and the nonzero entries of V as "final" states. Σ is partitioned as $\Sigma_1 \cup \cdots \cup \Sigma_r$, where the symbols of Σ_i are in bijective correspondence with $0, 1, \ldots p_i - 1$. The r-behavior of F, denoted by $\langle F \rangle_r$ is defined to be

$$U \cdot \left(\sum_{\sigma \in \Sigma} X_\sigma \right)^r \cdot V.$$

We can write $\langle F \rangle_r$ as

$$\sum_{\sigma_{i_1} \cdots \sigma_{i_r}} U \cdot X_{\sigma_{i_1}} \cdots X_{\sigma_{i_r}} \cdot V,$$

and a summand is denoted by $\langle F \rangle_r(\sigma_{i_1} \cdots \sigma_{i_r})$.

All of the square matrices X_σ will be structured so that unless $j = i + 1$, where $\sigma_i \in \Sigma_i$ and $\sigma_j \in \Sigma_j$, then $X_{\sigma_i} \cdot X_{\sigma_j} = 0$. That is we will enforce $\langle F \rangle_r(\sigma_{i_1} \cdots \sigma_{i_r}) = 0$ unless $\sigma_{i_j} \in \Sigma_j$ for $j = 1, \ldots, r$. This means that if $\sigma_{i_1} \cdots \sigma_{i_r}$ does not correspond to a CRR, any WFA that we construct will generate a zero summand for that string. This structure will emerge naturally in each of our constructions.

3.2. Algorithm 1

Recall that all matrix entries are rational coefficient polynomials in the variable t.

Lemma 3.1. *A WFA, F, can be computed in $n^{O(1)}$ time such that*

$$\langle F \rangle_r = \sum_x (1 - \beta(x) \cdot t) \cdot \left(\prod_{i=1}^{r} (1 + x_i \cdot t/p_i) \right).$$

Proof. The states are ordered pairs of integers (k, f), where $k = 0, \ldots, r$, and $f = 0, \ldots, 2^g \cdot r - 1$. For $\sigma \in \Sigma_i$, a nonzero entry of X_σ is indexed by a row, column pair of states

$$(i - 1, f), (i, f + \lfloor 2^g \cdot |\sigma \cdot (P/p_i)^{p_i - 2}|_{p_i} / p_i) \rfloor).$$

The entry at this location is

$$1 + |\sigma \cdot (P/p_i)^{p_i - 2}|_{p_i} \cdot t/p_i.$$

The only nonzero entry of U has index $1, (0, 0)$ and the entry there is 1. A nonzero entry of V is indexed by $(r, 2^g \cdot \beta + \alpha), 1$, and the entry there is $1 - \beta \cdot t$.

Recalling equation (4), it is clear that a nonzero entry of $(\sum_{\sigma\in\Sigma} X_\sigma)^k$ is indexed by

$$(0,0),\left(k,\sum_{i=1}^{k}\lfloor 2^g\cdot\sigma_i\cdot(P/p_i)^{p_i-2}|_{p_i}/p_i\rfloor\right),$$

where $\sigma_i\in\Sigma_i$. The entry at this location is

$$\prod_{i=1}^{k}1+|\sigma_i\cdot(P/p_i)^{p_i-2}|_{p_i}\cdot t.$$

The claim for $\langle F\rangle_r$ follows from this, the chinese remainder theorem, and the definition of V.
The time bound for computing F is clear from the description of its construction. □

Let $B=\langle F\rangle_r$, where F is the WFA in Lemma 3.1, and $B'=\frac{dB}{dt}|_{t=0}$. Here is algorithm 1: compute $B'-(P-1)/2$.

Theorem 3.2. *Algorithm 1 computes the bad census in polynomial time.*

Proof. By Lemma 3.1, equation (3), and calculus,

$$B'=\sum_x\left(-\beta(x)+\sum_{i=1}^{r}x_i/p_i\right)=\sum_x\left(x/P+q(x)-\beta(x)\right).$$

By equation (6),

$$B'=\sum_{x\text{ bad}}1+\sum_x x/P,$$

but $\sum_x x/P=(P-1)/2$. The time bound follows from the fact that B is a polynomial in t which is computable in polynomial time, so B' is computable in polynomial time. □

3.3. Algorithm 2

We have implemented and run Algorithm 2. Before describing the main steps of the algorithm, which again counts the bad integers we need to cover some preliminaries. We define a kind of polynomial multiplication, denoted by $\odot$ as follows. Let P,Q be polynomials in the variable z, then $P\odot Q$ is the polynomial obtained from $P\cdot Q$ (ordinary Cauchy product) by reducing all exponents of z $\bmod 2^g$, and collecting like terms. Thus, $P\odot Q$ has degree at most 2^g-1. The r-fold $\odot$ product of $P_1,\ldots,P_r$ is denoted by $\bigodot_{i=1}^{r}P_i$. The evaluation of a polynomial P at $z=a$ is denoted by $P(a)$. For $i=1,\ldots,r$ let

$$\mu_i(x)=\lfloor 2^g\cdot x_i/p_i\rfloor.$$

Here is Algorithm 2.

(1) For $i = 1, \ldots, r$, construct the polynomial C_i in the variable z,

$$\sum_{j=0}^{p_i - 1} z^{\mu_i(j)}.$$

(2) Compute $C = \bigodot_{i=1}^{r} C_i$.
(3) Compute $C' = \frac{d}{dz} C$.
(4) Compute $C'(1)$.
(5) Compute $D = C'(1)/2^g + (r/2^{g+1}) \cdot P - (r/2^{g+1}) \cdot \sum_{i=1}^{r} P/p_i - (P-1)/2$.

We claim that $D = \sum_{x \in \mathcal{B}} 1$.

Lemma 3.3. *Algorithm 2 computes the bad integer census.*

Proof. Start from equation (8). Calculation shows that

$$\sum_{i=1}^{r} \sum_{x} x_i/p_i = (1/2) \left(P - \sum_{i=1}^{r} P/p_i \right),$$

and $\sum_x x/P = (P-1)/2$. It remains to show that $C'(1) = \sum_x \alpha(x)$.

Consider the WFA, F whose states are pairs of integers of the form $(i, \sum_{j=1}^{i} \mu_j(x))$ for $i = 1, \ldots, r$. The start state is $(0, 0)$. We regard $(0, \sum_{j=1}^{0} \mu_j(x))$ as $(0, 0)$. The only nonzero entries of $X_{|x|_{p_i}}$ are indexed by the state transition pairs

$$\left(i-1, \sum_{j=1}^{i-1} \mu_j(x) \right), \left(i, \sum_{j=1}^{i} \mu_j(x) \right).$$

The common nonzero entry value is 1. The only nonzero entry of matrix $X_{|x|_{p_1}} \cdots X_{|x|_{p_r}}$ will be indexed by

$$(0, 0), \left(r, \sum_{j=1}^{r} \mu_j(x) \right).$$

Now, $\sum_{j=1}^{r} \mu_j(x) = a + 2^g \rho(x)$, where $a < 2^g$ is an integer. V has nonzero entries at states $(r, a + 2^g \cdot b)$, where $b < r$. The entry at such a state is a. We can see that the r-output of F is $\alpha(x)$, and hence $\langle F \rangle_r = \sum_x \alpha(x)$.

We now show that $C'(1) = \langle F \rangle_r$. The idea here recasts the original census approximation algorithm described in [8]. Write the polynomial C in step 2 of the algorithm as $C = \sum_{i=0}^{2^g - 1} a_i \cdot z^i$. In step 4 we obtain $\sum_{i=0}^{2^g - 1} i \cdot a_i$. We claim that

$$\sum_{i=0}^{2^g - 1} a_i \cdot i = \sum_{x} \alpha(x).$$

The coefficient a_i is just the number of $x < P$ such that $\alpha(x) = i$. Indeed, this is just a partial summand computed in the r-behavior of the WFA F described in the previous paragraph, and the lemma follows. □

Theorem 3.4. *Algorithm 2 computes the census of bad integers in*

$$O(n^2 \cdot \log\log(n) \cdot \mathcal{A}(n)/\log n)$$

bitwise arithmetic operations, where $\mathcal{A}(n)$ is the time to multiply two n-bit integers.

Proof. Lemma 3.3 establishes that the algorithm computes the bad census. We proceed to the time complexity. Multiplication of the r polynomials C_i, with reduction of all intermediate polynomial degrees modulo 2^g can be done in time dominated by the product of two degree 2^g polynomials with integer coefficients no larger than P. By Lemma 2.1, $P < 2^{2n}$, so coefficient multiplications involve at most $\mathcal{A}(2n)$ bitwise arithmetic operations. Since multiplication is no worse than quadratic time, we can express this cost as $O(\mathcal{A}(n))$. Using FFT, or similar convolution transform methods, the $O(r)$ pairwise polynomial products can each be done using $2^g \cdot g \cdot \log(g) \cdot \mathcal{A}(n)$ operations. By Lemma 2.1, $r = O(n/\log n)$, and since $2^g < 8r$, this cost is bounded above by $O(n \cdot \log\log(n) \cdot \mathcal{A}(n))$. The overall time follows by multiplying this bound by r. □

3.4. Experimental results

We tabulate representative densities calculated from actual counts for the bad integers over a range of values for r. We point out that on a dedicated 2.66 gHz computer with a 3 gB real address space the running time for $r = 511$ is roughly 70 minutes. For $r = 511$, $\lfloor \log P \rfloor = 5189$. In the interest of sanity, we illustrate the actual output only for $r = 127$ at the end of the paper.

```
r               density

8:              .18753305742133311028
15:             .40060892370174538968
16:             .21566463134239888256
24:             .33901848123361139064
31:             .44741712193144528296
32:             .23146403541827736960
40:             .29357038284339453948
63:             .47250524865958437592
64:             .24014639429784329084
127:            .48573776789735434444
128:            .24481932239434431308
511:            .49622202812798288372
```

The reader can check that as r increases, so does agreement with the asymptotic density expression $r/2^{g-1}$. Ultimately, the graph of density versus r is a "sawtooth" oscillating between $1/4$ and $1/2$.

For $r = 127$, P is

49620530728628930118156860850549 43
99585766193441465439326955956110026846
84338790529969965791243468218008024643
04723640429503137601278290422409955273 1270
96762893555100070212926092147189104881 3
74461018100187690751198809547084086962840
13642605698852198723139366300922347816 49
52125897464044412149392265

and the bad census is exactly

30128207297507874338819999414696658 31
41174530307576420814353435762841884
36106719059952854371696497034820435136
04877924724924946075379009091559973290 5296
66167484995354199796344101320542934876274
25754374668624388887175146205277926910 3
82434101790344495375748616805666567744 33
02058533986599226626870 39

References

[1] D.J. Bernstein and J. Sorenson, Modular exponentiation *via* the explicit chinese remainder theorem. *Math. Comp.* **76** (2007) 443–454.

[2] A. Chiu, G. Davida and B. Litow, Division in logspace-uniform NC^1. *RAIRO-Theor. Inf. Appl.* **35** (2001) 259–275.

[3] G. Davida and B. Litow, Fast parallel arithmetic via modular representation. *SIAM J. Comput.* **20** (1991) 756–765.

[4] P. Dusart, The kth prime is greater than $k(\ln k - \ln\ln k - 1)$ for $k \geq 2$. *Math. Comp.* **68** (1999) 411–415.

[5] G.H. Hardy and E.M.Wright, *An Introduction to the Theory of Numbers*. Oxford Press, USA (1979).

[6] D. Knuth, *The Art of Computer Programming, Vol. II*. Addison-Wesley (1969).

[7] W. Kuich and A. Salomaa, *Semirings, Automata, Languages*. Springer-Verlag (1986).

[8] B. Litow and D. Laing, A census algorithm for chinese remainder pseudorank with experimental results. Technical Report. http://www.it.jcu.edu.au/ftp/pub/techreports/2005-3.pdf

[9] A. Salomaa and S. Soittola, *Automata Theoretic Aspects of Formal Power Series*. Springer-Verlag (1978).

[10] S.P. Tarasov and M.N. Vyalyi, *Semidefinite programming and arithmetic circuit evaluation*. Technical report, arXiv:cs.CC/0512035 v1 9 Dec 2005 (2005).

[11] I.M. Vinogradov, *Elements of Number Theory*. Dover (1954).

Communicated by C. Choffrut.
Received February 17, 2006. Accepted June 18, 2007.