Ratio 1 and Ratio 2

Richard Adhika

1. Introduction

For reasons coming from number theory, we are interested in computing the following two kinds of (closely related) ratios, for different primes p: Let t and D be integers. The first ratio is

$$\nu_{p,n}(t,d) = \frac{\#\{M \in \mathbf{GL}_2(\mathbb{Z}/p^n\mathbb{Z}) | M \text{ has trace } t \text{ and } \det D \bmod p^n\}}{p^{2n-2}(p^2-1)}.$$

This ratio is really interesting only when the *discriminant* of the characteristic polynomial, i.e., $t^2 - 4D$, is divisible by p (and they get more interesting as the power of p dividing the discriminant gets larger). This stabilizes when n gets large (if p doesn't divide the discriminant, they stabilize right away). However, we understand these quite well. We are more interested in the ones that come not from counting solutions modp^n, but from counting the *reductions* modp^n *of solutions in* $\mathbb{Z}_p$. More precisely, we define

$$\mu_{p,n}(t,d) = \frac{\#\{M \in \mathbf{GL}_2(\mathbb{Z}/p^n\mathbb{Z}) | \exists M' \in \mathbf{GL}_2(\mathbb{Z}_p) : M' \text{ has trace } t \text{ and } \det D, \text{ and } M' \bmod p^n = M\}}{p^{2n-2}(p^2-1)}.$$

Since we cannot really handle full p-adic series in a computer code, we introduce another parameter, "kbig" or something (also user-defined), and compute

$$\mu_{p,n}(t,d) = \frac{\#\{M \in \mathbf{GL}_2(\mathbb{Z}/p^n\mathbb{Z}) | \exists M' \in \mathbf{GL}_2(\mathbb{Z}/p^{kbig}\mathbb{Z}) : M' \text{ has trace } t \text{ and } \det D, \text{ and } M' \bmod p^n = M\}}{p^{2n-2}(p^2-1)}.$$

Again this stabilizes eventually (as n gets large, but we need $kbig >> n$), and again it is only interesting when a power of p divides the discriminant (when it doesn't, this also stabilizes at $n = 1$ and are equal to ν_p). We want to know these numbers for various values of t, d, and p. The problem is that even if you let $p = 5$ and $kbig = 6$, it is already taking forever to compute (using brute force method).

2. Ratio 1

We first want to try to generate all matrices of the form $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ with $d \equiv t - a \bmod p^n$ and $ad - bc \equiv D \bmod p^n$ for $a, b, c, d \in \mathbb{Z}/p^n\mathbb{Z}$. Note that throughout this file, this notation will be used when referring to a general 2×2 matrix. Here, we have 3 free parameters a, b, and c since d depends solely on a for the given t. The naive brute force method is to loop through $[0, p^n - 1]$ for each a, b, c and then check the determinant condition. This takes $\mathcal{O}(p^{3n})$, which is very slow. Hence, the function `build_matrices1` in the file *ratio1_and_ratio2.ipynb* (as well as all other code discussed in this file) aims to generate the matrices more efficiently.

For the parameter a, we will still loop through $[0, p^n - 1]$. First, fix an a. Let rhs$=$ $ad - D \bmod p^n$. Then, for any $b \in [0, p^n - 1]$, the problem turns into solving the equation (for c)

$$bc \equiv rhs \bmod p^n. \tag{1}$$

If $p \nmid b$, then b^{-1} exists and thus we have a unique solution $c \equiv rhs \cdot b^{-1} \bmod p^n$ (line 6). We don't need to loop through c for these b. If $p \mid b$, then we have three cases as follows,

1. $p \nmid rhs$. This immediately tell us that (1) can't have any solution if $p \mid b$. This corresponds to the fact that the loop in line 9 won't execute.

2. $v_p(rhs) \geq 1$. If $v_p(b) > v_p(rhs)$, then (1) also can't have any solution. Hence, it suffices to check b with $1 \leq v_p(b) \leq v_p(rhs)$ (line 10 with $v_p(b) = i$). Then, for each i, let $b = p^i j$ so that we can rewrite (1) as
$$p^i j \cdot c \equiv rhs \bmod p^n,$$
We then get $c \equiv (rhs/p^i) \cdot j^{-1} \bmod p^{n-i}$, for which we lift c to modulo p^n. Also $v_p(rhs) < n$ so the modulo is well-defined.

3. $rhs \equiv 0 \bmod p^n$. Then, we can set $b = 0$ so that all $c \in [0, p^n - 1]$ satisfy (1) (line 14-16). Furthermore, for any b, all c with $v_p(c) = n - v_p(b)$ satisfy (1) (line 17 - 19).

Algorithm 1 build_matrices1

1: **function** BUILD_MATRICES1(n, p, t, D)
2: $\quad res \leftarrow [\,]$, $R_n \leftarrow \mathbb{Z}/p^n\mathbb{Z}$
3: $\quad$ **for all** $a \in R_n$ **do**
4: $\quad\quad rhs \leftarrow R_n(D - a(t - a))$
5: $\quad\quad$ Add all tuples (a, b, c, d) to res such that $c \equiv rhs \cdot b^{-1} \bmod p^n$ with $p \nmid b$.
6:
7: $\quad\quad$ **if** rhs $\neq 0$ **then**
8: $\quad\quad\quad$ **for** $i = 1$ to $v_p(rhs)$ **do**
9: $\quad\quad\quad\quad$ **for** $b \in R_n$ with $b = p^i j$ with $p \nmid j$ **do**
10: $\quad\quad\quad\quad\quad$ Add (a, b, c, d) to res with $c \in Rn$ satisfying $c \equiv rhs \cdot j^{-1} \bmod p^{n-i}$
11: $\quad\quad\quad\quad$ **end for**
12: $\quad\quad\quad$ **end for**
13: $\quad\quad$ **else**
14: $\quad\quad\quad$ **for all** $c \in R$ **do**
15: $\quad\quad\quad\quad$ Add $(a, 0, c, d)$ to res
16: $\quad\quad\quad$ **end for**
17: $\quad\quad\quad$ **for** $b \in R_n$ with $v_p(b) = i \geq 1$ **do**
18: $\quad\quad\quad\quad$ Add (a, b, c, d) to res with $c \in R_n$ satisfying $v_p(c) = n - i$
19: $\quad\quad\quad$ **end for**
20: $\quad\quad$ **end if**
21: $\quad$ **end for**
22: $\quad$ **return** res
23: **end function**

Assuming n is not large, this algorithm costs $\mathcal{O}(p^{2n})$, which is already quite an improvement from the brute force algorithm. Notice that in the second case, the number of matrices with $b = b'$ satisfying $v_p(b') = v$ is the same as that of $b = p^v$ as both corresponding loops in line 9-11 have the same step size.

Now, we can modify this algorithm a bit if we only want to *count* the number of matrices satisfying the numerator condition. The few points that we obtained so far are:

- For each a, there are $p^{n-1}(p-1)$ b's that are relatively prime to p. Each of these b produces exactly one matrix. Hence, we have $p^{2n-1}(p-1)$ matrices in total with $p \nmid b$.
- For each a, if rhs has p-valuation k, then there are

$$\sum_{i=1}^{k} p^{n-i-1} \cdot (p-1) \cdot p^i = kp^{n-1}(p-1)$$

  many solutions with $1 \leq v_p(b) \leq k$. The term $p^{n-i-1}(p-1)$ comes from the number of b with $v_p(b) = i$ and the term p^i comes from lifting the modulo from p^{n-i} to p^n.
- For each a, if rhs $\equiv 0 \bmod p^n$, then there are

$$p^n + \sum_{i=1}^{n-1} p^i(p^{n-i} - p^{n-i-1}) = p^n + (n-1)(p^n - p^{n-1})$$

  many solutions with $b = 0$ and $v_p(b) \geq 1$.

With these in mind, we then write a function `ratio1` that count the number of matrices satisfying the numerator condition in $\mathcal{O}(p^n)$. The pseudocode is as follows,

Algorithm 2 ratio1

function RATIO1(n, t, D)
    $\mathit{count} \leftarrow p^n \cdot p^{n-1} \cdot (p-1)$, $R_n \leftarrow \text{Integers}(p^n)$
    for all $a \in R$ **do**
        $rhs \leftarrow R_n(a(t-a) - D)$
        if $rhs \neq 0$ **then**
            $\mathit{count} \leftarrow \mathit{count} + v_p(b) \cdot p^{n-1} \cdot (p-1)$
        else
            $\mathit{count} \leftarrow \mathit{count} + p^n + (n-1) \cdot (p^n - p^{n-1})$
        end if
    end for
    return *count*
end function

3. Ratio 2

For the second ratio, however, we still need to generate the matrices as the reduction from $kbig$ to n may not include every matrix that satisfies the constraint in $\mathbb{Z}/p^n\mathbb{Z}$ (which is true if $p^l | t^2 - 4D$ for some $l \in \mathbb{N}$). The problem now is that generating and storing all such matrices in $\mathbb{Z}/p^{kbig}\mathbb{Z}$ and then reduce each one to mod p^n takes up a lot of time and space (e.g. for $p = 5$ and $kbig = 5$, we have around 11 million matrices and for $kbig = 6$, CoCalc runs out of space).

What if we don't actually need to generate all the matrices? The first observation is that

Lemma 1. *For any matrix* $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathbf{GL}_2(\mathbb{Z}/p^{kbig}\mathbb{Z})$, *if* $p \nmid b$, *we only need to consider* $a, b \in [0, p^n - 1]$.

Proof. Consider two matrices $\begin{pmatrix} a_1 & b_1 \\ c_1 & d_1 \end{pmatrix}$ and $\begin{pmatrix} a_2 & b_2 \\ c_2 & d_2 \end{pmatrix}$ with $p \nmid b_1, b_2$, $b_2 \equiv b_1 \bmod p^n$, and $a_2 \equiv a_1 \bmod p^n$. Let $rhs = a_1 d_1 - D \bmod p^{kbig}$. Then, the modular equation $b_1 c_1 \equiv rhs \bmod p^{kbig}$ has a unique solution $c_1 \equiv rhs \cdot b_1^{-1} \bmod p^{kbig}$. We therefore have

$$d_2 \equiv t - a_2 \equiv t - a_1 \equiv d_1 \bmod p^n$$

and

$$c_2 \equiv (a_2 d_2 - D) b_2^{-1} \equiv (a_1 d_1 - D) b_1^{-1} \equiv c_1 \bmod p^n.$$

Hence all $a, b \geq p^n$ with $p \nmid b$ will not produce any new reduced matrix. $\square$

This means that the reduction does not affect the existence and uniqueness of matrices for b with $p \nmid b$. Therefore, just as the first ratio, we don't need to consider these b's and that the count for these matrices is $p^{2n-1}(p - 1)$. Next, we have

Theorem 1. *Let A be the set of all $M \in \mathbf{GL}_2(\mathbb{Z}/p^n\mathbb{Z})$ such that there exists $M' \in \mathbf{GL}_2(\mathbb{Z}/p^{kbig}\mathbb{Z})$ having trace t and determinant D, and that $M \equiv M' \bmod p^n$ (the numerator condition). Let $S \subset A$ with $b = p^v$ and $S' \subset A$ with $b = p^v u$ for some fixed u with $p \nmid u$ and $v < n$. Then, $|S| = |S'|$.*

Proof. Let $v \in [1, n-1]$ and $M = \begin{pmatrix} a & p^v \\ c & d \end{pmatrix} \in S$. Define a map $\phi : S \to S'$ by

$$\phi(M) = N = \begin{pmatrix} a & p^v u \\ c u^{-1} & d \end{pmatrix}.$$

First, we want to show that this mapping is well-defined by showing that $N \in S'$. By the definition of M, there exists $M' \in \mathbf{GL}_2(\mathbb{Z}/p^{kbig}\mathbb{Z})$ of the form $M' = \begin{pmatrix} a' & b' \\ c' & d' \end{pmatrix}$ having trace t and determinant D and that $M' \equiv M \mod p^n$. Let

$$U = \begin{pmatrix} 1 & 0 \\ 0 & u \end{pmatrix}, \; U^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & u^{-1} \end{pmatrix}.$$

Let $N' \in \mathbf{GL}_2(\mathbb{Z}/p^{kbig}\mathbb{Z})$ be defined by

$$N' = U^{-1}MU = \begin{pmatrix} a' & b'u \\ c'u^{-1} & d' \end{pmatrix}.$$

Then, N' has trace t and determinant D, and that $b'u \equiv p^v u \bmod p^n$. Hence, $N = N' \bmod p^n \in S'$.

We then want to show that this mapping is injective. Let $M_1, M_2 \in S$ of the form

$$M_1 = \begin{pmatrix} a_1 & p^v \\ c_1 & d_1 \end{pmatrix}, M_2 = \begin{pmatrix} a_2 & p^v \\ c_2 & d_2 \end{pmatrix}.$$

Assume that $\phi(M_1) = \phi(M_2)$. Hence, we have

$$\begin{pmatrix} a_1 & p^v u \\ c_1 u^{-1} & d_1 \end{pmatrix} = \begin{pmatrix} a_2 & p^v u \\ c_2 u^{-1} & d_2 \end{pmatrix}$$

Since $p \nmid u$ and $p \nmid u^{-1}$, the equality $M_1 = M_2$ follows.

Finally, we want to show that the map is surjective. Let $N = \begin{pmatrix} a & p^v u \\ cu^{-1} & d \end{pmatrix} \in S'$ (any matrix in S' can be written in this form). Let $M = \begin{pmatrix} a & p^v \\ c & d \end{pmatrix}$. If N' is the lift of N, then we define $M' = U^{-1}N'U$ and a similar argument as before shows that $M \equiv M' \bmod p^n$. Hence $M \in S$ and that $\phi(M) = N$ so that ϕ is surjective. Finally, we conclude that ϕ is a well-defined bijection between S and S' so that $|S| = |S'|$. □

By Theorem 1, for ratio 2, we *only* need to consider the reduced matrices with $b = p^v$ for $1 \leq v < kbig$ and $b = 0$. There are still a few points that can be optimized especially with the heavy loops of the parameters b and c, which is what we are trying to do next.

Theorem 2. *Let $S \subset \mathbf{GL}_2(\mathbb{Z}/p^{kbig}\mathbb{Z})$ be the set of all matrices that we have encountered so far in the loop, having trace t and determinant D. Let $M = \begin{pmatrix} a_1 & b_1 \\ c_1 & d_1 \end{pmatrix} \in S$. Furthermore, let $N = \begin{pmatrix} a_2 & b_2 \\ c_2 & d_2 \end{pmatrix} \in \mathbf{GL}_2(\mathbb{Z}/p^{kbig}\mathbb{Z})$ be the matrix that we are considering with $N \equiv M \bmod p^n$ and $v_p(b) = v_p(b') = i$. Then, we don't need to consider all matrices of the form $N' = \begin{pmatrix} a_2 & b_2' \\ c_2' & d_2 \end{pmatrix}$ for*

- *$b_2' = b_2$ and $c_2' > c_2$ satisfying the determinant constraint, or*
- *$b_2' > b_2$ with $v_p(b_2') = i$ and for any c_2' satisfying the determinant constraint.*

Proof. To show the first point we want to show that we don't need to consider further c values once we encounter such N. For any $s \in \mathbb{N}$ and for any matrix $N' = \begin{pmatrix} a_2 & b_2 \\ c_2 + sp^{kbig-i} & d_2 \end{pmatrix}$, we have

$N' \equiv M' \bmod p^n$ where $M' = \begin{pmatrix} a_1 & b_1 \\ c_1 + sp^{kbig-i} & d_1 \end{pmatrix}$. If either $a_2 > a_1$ or $b_2 > b_1$, it is clear that $M' \in S$. Otherwise, $c_2 > c_1$ so M' must have also been considered earlier (in the code itself, this case won't happen since it is enough to consider all matrices with $c \in [0, p^n - 1]$).

Next, we want to show that if $a_2 > a_1$, then the assumption also holds with $b_2 = p^i$ and that we don't need to consider any matrix of the form $N' = \begin{pmatrix} a_2 & sp^i \\ c_2' & d_2 \end{pmatrix}$ with $p \nmid s, s \geq 2$, and for any c_2' satisfying the determinant constraint. Let $b_2 = p^i u$ with $p \nmid u$. We can then write $b_1 = p^i \beta' u$, $c_1 = \gamma' u^{-1}$, and $c_1 = \gamma u^{-1}$ so that we have

$$\begin{pmatrix} a_2 & p^i u \\ \gamma u^{-1} & d_2 \end{pmatrix} \equiv \begin{pmatrix} a_1 & p^i \beta' u \\ \gamma' u^{-1} & d_1 \end{pmatrix} \bmod p^n.$$

Since $p \nmid u, u^{-1}$, the equation above is equivalent to

$$\begin{pmatrix} a_2 & p^i \\ \gamma & d_2 \end{pmatrix} \equiv \begin{pmatrix} a_1 & p^i \beta' \\ \gamma' & d_2 1 \end{pmatrix} \bmod p^n.$$

Since the right matrix is in S, the left matrix doesn't give any new reduced matrix so that the assumption also holds for $b_2 = p^i$. Then, for any $s \in \mathbb{N}$ with $p \nmid s$, we have

$$\begin{pmatrix} a_2 & p^j s \\ \gamma s^{-1} & d_2 \end{pmatrix} \equiv \begin{pmatrix} a_1 & p^i \beta' s \\ \gamma' s^{-1} & d_1 \end{pmatrix} \bmod p^n,$$

for which the right matrix is also in S. Together with the first point, this means that N' have the same reduced form as one of the matrices in S.

Otherwise, assume that $a_2 = a_1$. Then $b_2 > b_1$, so let $b_2 = b_1 + sp^i$ for some $s \in \mathbb{N}$. Let $\alpha = \frac{a_2 d_2 - D}{p^i}$. Then we have

$$c_2 \equiv \alpha (b_1/p^i + s)^{-1} \bmod p^{kbig-i} \quad \text{and} \quad c_1 \equiv \alpha (b_1/p^i)^{-1} \bmod p^{kbig-i}.$$

Since $c_2 \equiv c_1 \bmod p^n$, we have

$$\alpha (b_1/p^i + s)^{-1} \equiv \alpha (b_1/p^i)^{-1} \bmod p^n.$$

Note that $(b_1/p^i + s)^{-1}$ exists since $v_p(b_2) = i$ by assumption. Let $b_2' = b_1 + (s + s')p^i$ with $s' \in \mathbb{N}$. First, assume that $v_p(\alpha) < n$ so that $s \equiv 0 \bmod p$. We then have

$$c_2' \equiv \alpha (b_1/p^i + (s + s'))^{-1} \equiv \alpha (b_1/p^i + s')^{-1} \bmod p^n.$$

Note that $(b_1/p^i + s')^{-1}$ exists since $b_1/p^i + s' \equiv b_1/p^i + s' + s' \not\equiv 0 \bmod p$. Hence, we have

$$\begin{pmatrix} a_2 & b_1 + (s + s')p^i \\ c_2' & d_2 \end{pmatrix} \equiv \begin{pmatrix} a_1 & b_1 + s'p^i \\ c_1' & d_1 \end{pmatrix} \bmod p^n, \tag{2}$$

where $c_1' \equiv \alpha (b_1/p^i + s')^{-1} \bmod p^n$. If $v_p(\alpha) \geq n$, then (2) also holds with $c_2' \equiv c_1' \equiv 0 \bmod p^n$. In both cases, we get that the reduced form of N' has been considered earlier. Hence combined with the first point, we don't need to consider all such N'.

$\square$

By Theorem 2, for a fixed a, once we encounter such N in the code, we can directly move on to another value of a. Before going to the main function `ratio2`, we will take a look more closely at two helper functions: `ratio2_adder` and `ratio2_adder_with_check`, which reduce a lot of repetitions (of b and c) to make the code more efficient by utilizing Theorem 2. Below is the pseudocode for the function `ratio2_adder`.

Algorithm 3 Helper Function 1

function RATIO2_ADDER($p, n, kbig$, projset$, a, c_s, exp, R_n$)
    if $\big(R_n(a),\ R_n(c_{\text{start}})\big) \in$ projset **then**
        return false
    end if
    Add $\big(R_n(a),\ R_n(c_{\text{start}})\big)$ to projset
    for c from $c_{\text{start}} + p^{k_{\max}-\text{exp}}$ to $p^n - 1$ step $p^{k_{\max}-\text{exp}}$ **do**
        Add $\big(R_n(a),\ R_n(c)\big)$ to projset
    end for
    return true
end function

Note that we use this function only for $v_p(b) < n$ as the b entry in all matrices in $projset$ has the same p-valuation. The parameter $projset$ keeps track of the reduced matrices we encounter so far, c_s denotes the smallest solution of c when solving the determinant constraint, and exp denotes $v_p(b)$. This function returns false if we encounter one reduced matrix that has been considered earlier so that the function `ratio2` can break out of the loop. For $v_p(b) \geq n$, we have the following similar helper function instead,

Algorithm 4 Helper Function 2

function RATIO2_ADDER_WITH_CHECK($p, n, kbig$, lift_dict$, a, b, c_s, exp, R_n, R_{kbig}$)
    if lift_dict has key $\big(R_n(a),\ R_n(c_s)\big)$ **then**
        if the corresponding lifted matrix has $v_p(b) = exp$ or $b \equiv 0 \bmod p^{kbig}$ **then**
            return false
        end if
    end if
    Add $\big(R_n(a),\ R_n(c_s)\big)$ as key and $\big(R_{kbig}(a),\ R_{kbig}(b),\ R_{kbig}(c_s)\big)$ to lift_dict
    for c from $c_s + p^{kbig-exp}$ to $p^n - 1$ step $p^{kbig-exp}$ **do**
        Add $\big(R_n(a),\ R_n(c)\big)$ as key and $\big(R_{kbig}(a),\ R_{kbig}(b),\ R_{kbig}(c)\big)$ to lift_dict
    end for
    return true
end function

The only different thing here is that we now have a dictionary called $lift\_dict$ that keeps track on the pairing between the reduced matrix (entries in $\mathbb{Z}/p^n\mathbb{Z}$) and the original matrix (entries in $\mathbb{Z}/p^{kbig}\mathbb{Z}$). This is done to make sure that Theorem 2 applies as we require the assumption that the matrices compared have the same p-valuation of their corresponding b entry. Furthermore, if $b \equiv 0 \bmod p^{kbig}$, this means that we have covered all possible reduced form of c.

The function `ratio2` uses these two helper functions to efficiently count the number of matrices satisfying the numerator of the second ratio. For $1 \leq v_p(b) \leq n-1$, the function uses `ratio2_adder` and only uses set to keep track of the matrices encountered (only store the reduced matrices), which can be seen in line 3-17. On the other hand, for $v_p(b) \geq n$, the function uses `ratio2_adder_with_check` and uses dictionary to keep track of both the reduced and original matrices. It is not easy to assess the asymptotic runtime of this function, but it takes only a few seconds to compute ratio2 for $p = 5, kmax = 9$. The pseudocode is as follows,

Algorithm 5 Ratio2

1: **function** RATIO2$(p, kbig, n, t, D)$
2: $count \leftarrow p^{2n-1}(p-1)$, $R_n \leftarrow \mathbb{Z}/p^n\mathbb{Z}$, $R_{kbig} \leftarrow \mathbb{Z}/p^{kbig}\mathbb{Z}$
3: **for** $\exp = 1$ **to** $n-1$ **do**
4: $projset \leftarrow \emptyset$
5: **for** $a = 0$ **to** $p^{kbig} - 1$ **do**
6: $rhs \leftarrow R_{kbig}(a \cdot (t-a) - D)$
7: **if** $rhs \neq 0$ **then**
8: **if** $v_p(rhs) \geq exp$ **then**
9: for all $b = p^{exp}u$ with $p \nmid u$, let $c_s \equiv (rhs/p^{exp})u^{-1} \bmod p^{kbig-exp}$
10: if ratio2_adder$(p, n, kbig, projset, a, c_s, \exp, R_n)$ is **false**, stop checking other b's
11: **end if**
12: **else**
13: ratio2_adder$(p, n, kbig, projset, a, 0, \exp, R_n)$
14: **end if**
15: **end for**
16: $count \leftarrow count + |projset| \cdot \left(p^{n-\exp} - p^{n-\exp-1}\right)$
17: **end for**
18:
19: $lift\_dict \leftarrow \emptyset$
20: **for** $a = 0$ **to** $p^{kbig} - 1$ **do**
21: $rhs \leftarrow R_{kbig}(a \cdot (t-a) - D)$
22: **if** $rhs \neq 0$ **then**
23: **for** $\exp = n$ **to** $v_p(rhs)$ **do**
24: for all $b = p^{exp}u$ with $p \nmid u$, let $c_s \equiv (rhs/p^{exp})u^{-1} \bmod p^{kbig-exp}$
25: if ratio2_adder_with_check$(p, n, kbig, lift_dict, a, b, c_s, \exp, R_n, R_{kbig})$ is **false**, stop checking other b's
26: **end for**
27: **else**
28: ratio2_adder_with_check$(p, n, kbig, lift_dict, a, 0, 0, kbig, R_n, R_{kbig})$
29: **end if**
30: **end for**
31: **return** $count + |\text{lift\_dict}|$
32: **end function**