EXPLICIT BOUNDS AND PARALLEL ALGORITHMS FOR COUNTING MULTIPLY GLEEFUL NUMBERS

SARA MOORE AND JONATHAN P. SORENSON

Abstract. Let $k \geq 1$ be an integer. A positive integer n is *k-gleeful* if n can be represented as the sum of kth powers of consecutive primes. For example, $35 = 2^3 + 3^3$ is a 3-gleeful number, and $195 = 5^2 + 7^2 + 11^2$ is 2-gleeful. In this paper, we present some new results on k-gleeful numbers for $k > 1$.

First, we extend previous analytical work. For given values of x and k, we give explicit upper and lower bounds on the number of k-gleeful representations of integers $n \leq x$.

Second, we describe and analyze two new, efficient parallel algorithms, one theoretical and one practical, to generate all k-gleeful representations up to a bound x.

Third, we study integers that are *multiply* gleeful, that is, integers with more than one representation as a sum of powers of consecutive primes, including both the same or different values of k. We give a simple heuristic model for estimating the density of multiply-gleeful numbers, we present empirical data in support of our heuristics, and offer some new conjectures.

1. Introduction

Let $k \geq 1$ be an integer. We say a positive integer n is *k-gleeful* if n can be written as the sum of kth powers of consecutive primes. For example, $35 = 2^3 + 3^3$ is a 3-gleeful number, and $195 = 5^2 + 7^2 + 11^2$ is 2-gleeful. Let $f_k(n)$ denote the number of representations a positive integer n has as a k-gleeful number, and let

$$s_k(x) = \sum_{n=1}^{x} f_k(n),$$

the total number of k-gleeful representations up to x. In this paper, we address two questions about gleeful numbers and their representations:

- Can we give explicit upper and lower bounds on $s_k(x)$?
- What can we say about integers where $f_k(n) > 1$?

Before we state our results, we give some background on what is already known.

Date: July 16, 2025.

1.1. **Previous Work.** Moser [?] proved that $s_1(x) \sim x \log 2$. He also posed several interesting questions on the behavior of $f_1(n)$. See also [?]. In this paper, we only look at k-gleeful numbers for integers $k > 1$.

Tongsomporn, Wananiyakul, and Steuding [?] proved that

$$s_2(x) < 10.9558 \frac{x^{2/3}}{(\log x)^{4/3}}.$$

They also computed a list of 2-gleeful numbers up to 2000.

In [?] it was proved that for every integer $k > 1$,

$$s_k(x) \geq \frac{(k+1)^2}{2} \cdot \frac{x^{2/(k+1)}}{(\log x)^{2k/(k+1)}} \cdot (1 + o(1)) \tag{1.1}$$

and for $c_k = (k^2/(k-1))(k+1)^{1-1/k}$,

$$s_k(x) \leq c_k \cdot \frac{x^{2/(k+1)}}{(\log x)^{2k/(k+1)}} (1 + o(1)). \tag{1.2}$$

Note that $(k+1)^2 > c_k > (k+1)^2/2$. They also gave two efficient sequential algorithms: one to enumerate k-gleeful representations and one to compute the exact value of $s_k(x)$, and they presented numerical data supporting their analytical results.

1.2. **New Results and Paper Outline.** In this paper, we continue the work from [?].

In §**??**, for given values of k and x, we give explicit upper and lower bounds on $s_k(x)$.

Our particular interest was to learn more about multiply-gleeful numbers or *duplicates*, that is, integers n with either $f_k(n) > 1$ or both $f_k(n) > 0$ and $f_{k'}(n) > 0$ for $k \neq k'$.

We found the enumeration algorithm from [?] did not work well for finding duplicates, as it requires too much memory. In §**??**, we describe and analyze two parallel algorithms for finding k-gleeful numbers – one practical, the other theoretical. The practical algorithm is based on a sequential routine that finds all k-gleeful numbers in a short interval. The results from that interval are then sorted to detect values of n with $f_k(n) > 1$. This parallelizes nicely by simply processing short intervals concurrently. To detect duplicates with differing k values, the algorithm is run twice on the same interval, once for each k value, and again, the interval's results are sorted to detect the duplicates. Our theoretical algorithm shows this problem is in $\mathcal{NC}$ [?].

Then, in §**??**, we describe a heuristic model that predicts how many duplicates we expect to find up to x. We then evaluated our model using data generated by our new parallel algorithm. We state some conjectures consistent with these results.

Our code and data can be found at `https://github.com/sorenson64/sopp`.

2. Explicit Bounds

In this section, we prove the following theorems, which are explicit versions of Theorem 1 from [**?**].

Let p_n denote the nth prime with $p_1 = 2$. The number of primes up to x is given by $\pi(x)$. For fixed x and $k \geq 2$ an integer, let $M := M(x,k)$ be the maximum length of any representation of any integer up to x, so that

$$2^k + 3^k + \cdots + p_M^k \leq x < 2^k + 3^k + \cdots + p_M^k + p_{M+1}^k. \tag{2.1}$$

Observe that for a given k, any bound on M implies a bound on x. Let M_0 be a fixed integer, at least 6. Our results depend on $M(x,k)$ being larger than M_0. Choosing M_0 to be larger gives better explicit constants.

We define the following functions on y and k. We will be plugging M_0 in for y.

$$\begin{aligned}
c_k &= \left(\frac{k^2}{k-1}\right) \cdot (k+1)^{(k-1)/k} \qquad \text{from equation (\textbf{??})},\\
A(y) &:= \frac{\log(y/2)}{\log y},\\
B(y,k) &:= \frac{\log(y+1)}{\log y} + \frac{\log\log(y+1)^2}{\log y}\frac{k}{k+1},\\
C(y,k) &:= \left(\frac{y}{y-1}\right)^{1/(k+1)} B(y,k)^{k/(k+1)},\\
D(y,k) &:= \left(\frac{y}{y+3}\right)\left(\frac{\log(y/2)}{\log y + 2\log\log(y+2)}\right)^{k/(k+1)},\\
E(y,k) &:= 1 + \frac{1}{(k+1)A(y) - 1},\\
F(y,k) &:= \left(\frac{y+1}{y}\right)^{(k-1)/k} \cdot 4^{(k-1)/(k(k+1))} \cdot C(y,k)^{(k-1)/k} \cdot E(y,k),\\
U(y,k) &:= 1.25506 \cdot F(y,k),\\
L(y,k) &:= \left(\frac{y-1}{y}\right) \cdot D(y,k)^2.
\end{aligned}$$

Note that $A(y), B(y,k), C(y,k), D(y,k) \to 1$ for large y. $E(y,k) \to 1 + 1/k$ for large y.

Theorem 2.1. *For $k > 1$ and $M \geq M_0 \geq 6$, we have*

$$s_k(x) \leq c_k \cdot \frac{x^{2/(k+1)}}{(\log x)^{k/(k+1)}} \cdot U(M_0, k).$$

For $k = 2$, in [**?**] they give 10.9558. Here, for $k = 2$, we get the weaker bound 14.2423 for $M \geq M_0 = 6$. The results in [**?**] give a constant of $c_2 = 4 \cdot \sqrt{3} \approx 6.928$, for large x

Theorem 2.2. *For $k > 1$ and $M \geq M_0 \geq 6$, we have*

$$s_k(x) \geq \frac{(k+1)^2}{2} \cdot \frac{x^{2/(k+1)}}{(\log x)^{k/(k+1)}} \cdot L(M_0, k).$$

See Table **??**; we show two numbers for each combination of M_0 and k: the lower bound constant followed by the upper bound constant. These constants include everything except the main term $x^{2/(k+1)}/(\log x)^{k/(k+1)}$.

$M_0 =$	6	100	10000	1000000
$k = 2$	0.391504, 14.2423	1.71182, 12.1097	2.39745, 11.6778	2.7343, 11.5116
$k = 3$	0.580731, 23.4232	2.72032, 18.7705	3.93987, 17.7299	4.5675, 17.3147
$k = 5$	1.09023, 63.156	5.47127, 48.0799	8.19445, 44.4013	9.6564, 42.8989
$k = 10$	3.10821, 249.625	16.6068, 182.224	25.6426, 164.599	30.6698, 157.311
$k = 20$	10.3113, 1009.68	57.0995, 720.629	89.7176, 642.315	108.222, 609.797

TABLE 1. Constants for lower and upper bounds on $s_k(x)$

2.1. **Setup.** Let us define $f_{k,m}(n)$ to be the number of representations of n as a sum of exactly m kth powers of consecutive primes. Observe that $f_{k,m}(n) = 0$ or 1 and that

$$f_k(n) = \sum_{m=1}^{M} f_{k,m}(n).$$

Let $s_{k,m}(x)$ be the number of positive integers n such that

$$p_n^k + p_{n+1}^k + \cdots + p_{n+m-1}^k \leq x.$$

Then

$$s_{k,m}(x) = \sum_{n=1}^{x} f_{k,m}(n)$$

so that

$$s_k(x) = \sum_{n=1}^{x} f_k(n) = \sum_{n=1}^{x} \sum_{m=1}^{M} f_{k,m}(n) = \sum_{m=1}^{M} s_{k,m}(x).$$

This counts the number of representations of k-gleeful numbers $\leq x$.

We will make use of the following results due to Rosser and Schonfeld [**?**] and Rosser [**?**].

$$n \log n < p_n \qquad \text{for } n \geq 1 \tag{2.2}$$

$$p_n < n \log n + 2n \log \log n \qquad \text{for } n \geq 3 \tag{2.3}$$

$$p_n < 2n \log n \qquad \text{for } n \geq 3 \tag{2.4}$$

$$\pi(x) < 1.25506 \frac{x}{\log x} \qquad \text{for } x \geq 2 \tag{2.5}$$

We will need upper and lower bounds on M.

Lemma 2.3. *If* $M \geq M_0 \geq 6$ *then*

$$M < 4^{1/(k+1)} \cdot \frac{(k+1)x^{1/(k+1)}}{(\log x)^{k/(k+1)}} \cdot C(M_0, k).$$

Lemma 2.4. *If* $M \geq M_0 \geq 6$ *then*

$$M \geq (k+1) \cdot \frac{x^{1/(k+1)}}{(\log x)^{k/(k+1)}} \cdot D(M_0, k).$$

For large M, from [**?**] we expect $M \sim (k+1) \cdot x^{1/(k+1)}/(\log x)^{k/(k+1)}$. See Table **??** for some exact values of $M(x, k)$.

x	$M(x,2)$	$M(x,3)$	$M(x,5)$	$M(x,10)$	$M(x,20)$
10^3	7	4	2	0	0
10^4	14	7	3	1	0
10^5	28	11	4	2	0
10^6	54	18	6	2	0
10^7	105	29	8	3	1
10^8	207	47	11	3	1
10^9	411	77	15	4	1
10^{10}	822	126	21	4	2
10^{11}	1656	209	30	5	2
10^{12}	3356	348	40	6	2
10^{13}	6834	581	55	8	2
10^{14}	13975	974	76	9	3
10^{15}	28682	1640	106	10	3
10^{16}	59066	2771	148	12	3
10^{17}	121987	4695	206	15	4
10^{18}	252574	7977	288	17	4
10^{19}	524136	13589	403	20	4
10^{20}	1089888	23201	566	24	4

TABLE 2. Exact values of $M(x, k)$ for various x, k

As stepping stones, the proofs of these two lemmas utilize easier-to-prove upper and lower bounds on $\log M$ in terms of x and k.

Lemma 2.5. *If* $M \geq M_0 \geq 6$ *then we have*

$$\begin{aligned} \log M &< \frac{\log x}{k+1} \cdot \frac{1}{A(M_0)} \quad \textit{and} \\ \log M &\geq \frac{\log x}{k+1} \cdot \frac{1}{B(M_0, k)}. \end{aligned}$$

As we let M_0 get larger, we get the expected $(k+1)\log M \sim \log x$.

Proof. Since $M \geq 6$, we can bound the sum on the left of (**??**) from below by $(M/2)p_{M/2}^k$. Taking $(M/2)\log(M/2) < p_{M/2}$ from (**??**), we have

$$(M/2)^{k+1}(\log(M/2))^k < x.$$

Because $M \geq 6 > 2e$, we can drop the $(\log(M/2))^k$ term since it exceeds 1. Taking logarithms of both sides then gives

$$\begin{aligned}\log x > (k+1)\log(M/2) &= (k+1)(\log M)\left(1 - \frac{\log 2}{\log M}\right) \\ &\geq (k+1)(\log M)\left(1 - \frac{\log 2}{\log M_0}\right).\end{aligned}$$

The sum on the right of (**??**) is bounded above by $(M+1)p_{M+1}^k$. Using (**??**) gives

$$\log x < (k+1)\log(M+1) + k\log(2\log(M+1)).$$

With $M \geq M_0$, we know that

$$\log(M+1) = \frac{\log(M+1)}{\log M}\log M < \frac{\log(M_0+1)}{\log M_0}\log M. \tag{2.6}$$

We also know that when $M \geq M_0 \geq 6$,

$$\frac{\log(2\log(M+1))}{\log(M+1)}$$

is maximized when $M = M_0$. This then gives

$$\begin{aligned}\frac{\log x}{k+1} &\leq (\log M)\left(1 + \frac{\log\log(M_0+1)^2}{\log(M_0+1)}\frac{k}{k+1}\right)\frac{\log(M_0+1)}{\log M_0} \\ &= (\log M)\left(\frac{\log(M_0+1)}{\log M_0} + \frac{\log\log(M_0+1)^2}{\log M_0}\frac{k}{k+1}\right).\end{aligned}$$

Plugging in the definitions of $A(M_0), B(M_0,k)$ completes the proof. $\square$

2.2. **Proof of Lemma ??.** Next, we bound x from below in terms of M and k. Using (**??**) and (**??**), We have

$$\begin{aligned}x &\geq p_1^k + p_2^k + \cdots + p_M^k \\ &> \sum_{n=1}^{M}(n\log n)^k \quad \geq \sum_{n=M^{1-1/k}}^{M}(n\log n)^k \\ &\geq (\log M^{1-1/k})^k \sum_{n=M^{1-1/k}}^{M} n^k \\ &= (1-1/k)^k(\log M)^k \sum_{n=M^{1-1/k}}^{M} n^k.\end{aligned}$$

We can bound the sum on n^k with an integral to get

$$\begin{aligned}\sum_{n=M^{1-1/k}}^{M} n^k &\geq \int_{M^{1-1/k}}^{M} t^k dt = \frac{M^{k+1}-M^{(k+1)(1-1/k)}}{k+1} \\ &= \frac{M^{k+1}-M^{k-1/k}}{k+1} = \frac{M^{k+1}}{k+1}\left(1-\frac{1}{M^{1+1/k}}\right) \\ &> \frac{M^{k+1}}{k+1}\cdot\frac{M_0-1}{M_0}\end{aligned}$$

if $M \geq M_0$. We also have $(1-1/k)^k \geq 1/4$ for $k \geq 2$. Pulling this together gives

$$x \geq \frac{M^{k+1}(\log M)^k}{4(k+1)}\cdot\frac{M_0-1}{M_0},$$

which is valid when $M \geq M_0$. This directly gives

$$M^{k+1} \leq \frac{4(k+1)x}{(\log M)^k}\cdot\frac{M_0}{M_0-1}.$$

Applying Lemma **??** then gives

$$M^{k+1} < \frac{4\cdot(k+1)^{k+1}x}{(\log x)^k}\cdot\frac{M_0}{M_0-1}B(M_0,k)^k$$

and

$$M < 4^{1/(k+1)}\frac{(k+1)x^{1/(k+1)}}{(\log x)^{k/(k+1)}}\cdot\left(\frac{M_0}{M_0-1}\right)^{1/(k+1)} B(M_0,k)^{k/(k+1)}.$$

This completes the proof of Lemma **??**.

2.3. **Proof of Lemma ??.** Again, starting from (**??**), observing that $2^k + 3^k < 5^k < p_{M+2}^k$, and applying (**??**), we have

$$\begin{aligned}x &\leq p_1^k + \cdots + p_{M+1}^k \leq p_3^k + \cdots + p_{M+2}^k \\ &\leq \sum_{n=3\rfloor}^{M+2} n^k(\log n + 2\log\log n)^k \\ &\leq (\log(M+2) + 2\log\log(M+2))^k \sum_{n=3}^{M+2} n^k \\ &\leq (\log(M+2) + 2\log\log(M+2))^k \frac{(M+3)^{k+1}}{k+1} \\ &\leq (\log M)^k\left(\frac{\log(M_0+2)}{\log M_0} + \frac{2\log\log(M_0+2)}{\log M_0}\right)^k \frac{(M+3)^{k+1}}{k+1}.\end{aligned}$$

Here we bounded the sum $\sum_{n=3}^{M+2} n^k$ with an integral to get $\frac{(M+3)^{k+1}}{k+1}$. Using (**??**), we have

$$x \leq (\log M)^k \frac{((M_0+3)/M_0) \cdot M)^{k+1}}{k+1} \left(\frac{\log(M_0+2)}{\log M_0} + \frac{2 \log\log(M_0+2)}{\log M_0} \right)^k,$$

or

$$(k+1) \frac{x}{(\log M)^k} \frac{(M_0/(M_0+3))^{k+1}}{\left(\frac{\log(M_0+2)}{\log M_0} + \frac{2 \log\log(M_0+2)}{\log M_0} \right)^k} \leq M^{k+1}.$$

We apply Lemma **??** and simplify a bit to obtain

$$\frac{(k+1)x^{1/(k+1)}}{(\log x)^{k/(k+1)}} \cdot \left(\frac{M_0}{M_0+3} \right) \left(\frac{\log(M_0/2)}{\log(M_0+2) + 2\log\log(M_0+2)} \right)^{k/(k+1)} \leq M.$$

This completes the proof.

2.4. **Proof of Theorem ??.** For positive integers n, m, we refer to any sum of the form $p_n^k + p_{n+1}^k + \cdots + p_{n-1+m}^k$ as a *chain* of length m. Recall that

$$s_{k,m}(x) = \#\{n : p_n^k + p_{n+1}^k + \cdots + p_{n-1+m}^k \leq x\},$$

the number of k-gleeful representations of integers $\leq x$ of length m, or the number of chains of length m whose sums are bounded by x.

For the moment, fix a chain length m. Choose n, the starting point of the chain, as large as possible so that we have

$$m \cdot p_n^k \leq p_n^k + p_{n+1}^k + \cdots + p_{n-1+m}^k \leq x \leq p_{n+1}^k + p_{n+2}^k + \cdots + p_{n+m}^k.$$

This gives then

$$\begin{array}{rcl} mp_n^k & \leq & x, \\ p_n & \leq & (x/m)^{1/k}, \\ n & \leq & \pi((x/m)^{1/k}). \end{array}$$

Observe that $s_{k,m}(x) = n$ here. (See Lemma 1 from [**?**].) Thus,

$$s_k(x) = \sum_{m=1}^{M} s_{k,m}(x) \leq \sum_{m=1}^{M} \pi((x/m)^{1/k}).$$

Observe that $(x/m)^{1/k} \geq (p_M^k/M)^{1/k} > 2$ for $M \geq 6$. From (**??**) we have

$$\pi(t) \leq 1.25506 \cdot t/\log t$$

when $t \geq 2$. This gives

$$\begin{aligned} s_k(x) &\leq \sum_{m=1}^{M} \pi((x/m)^{1/k}) \\ &\leq 1.25506 \cdot \sum_{m=1}^{M} \frac{k(x/m)^{1/k}}{\log(x/m)} \\ &\leq 1.25506 \cdot \frac{kx^{1/k}}{\log(x/M)} \sum_{m=1}^{M} \frac{1}{m^{1/k}}. \end{aligned}$$

Focusing first on the logarithm in the denominator, we have

$$\begin{aligned} \log(x/M) &\geq \log x - \frac{\log x}{(k+1)A(M_0)} \qquad \text{or} \\ \frac{1}{\log(x/M)} &\leq \frac{1}{\log x}\left(1 + \frac{1}{(k+1)A(M_0) - 1}\right) \\ &= \frac{1}{\log x} \cdot E(M_0, k) \end{aligned}$$

using Lemma **??**. Next, we estimate the sum:

$$\begin{aligned} \sum_{m=1}^{M} \frac{1}{m^{1/k}} &\leq \int_{1}^{M+1} t^{-1/k} dt \\ &\leq \frac{(M+1)^{1-1/k}}{1 - 1/k}. \end{aligned}$$

Pulling this together, we have

$$\begin{aligned} s_k(x) &\leq 1.25506 \cdot E(M_0, k) \cdot \frac{kx^{1/k}}{\log x} \frac{(M+1)^{1-1/k}}{1-1/k} \\ &\leq 1.25506 \cdot E(M_0, k) \cdot \left(\frac{M_0+1}{M_0}\right)^{(k-1)/k} \left(\frac{k^2}{k-1}\right) \frac{x^{1/k}}{\log x} \cdot M^{(k-1)/k}, \end{aligned}$$

since $M \geq M_0$. Next, we plug in our upper bound on M from Lemma **??**. We have

$$M^{(k-1)/k} < 4^{(k-1)/(k(k+1))} \cdot (k+1)^{(k-1)/k} \cdot C(M_0, k)^{(k-1)/k} \cdot \frac{x^{(k-1)/(k(k+1))}}{(\log x)^{(k-1)/(k+1)}}.$$

Plugging this in, we obtain

$$s_k(x) \leq 1.25506 \cdot c_k \cdot \frac{x^{2/(k+1)}}{(\log x)^{2k/(k+1)}} \cdot F(M_0, y),$$

and the result follows.

2.5. **Proof of Theorem ??.** Any subsequence sum of the maximum chain length M represents a k-gleeful number $\leq x$. Thus, the number of i, j pairs such that $1 \leq i \leq j \leq M$, or $\binom{M}{2}$, is a lower bound for $s_k(x)$. Thus,

$$s_k(x) \geq \binom{M}{2} = \frac{M(M-1)}{2} \geq \frac{M_0 - 1}{2M_0} \cdot M^2$$

since we are assuming $M \geq M_0$. Simply apply Lemma **??** and a bit of algebra and the result follows.

3. Two Parallel Algorithms

We begin with a straightforward adaptation of the enumeration algorithm from [**?**] to work on an interval.

3.1. **An Algorithm to Enumerate Representations on an Interval.** Here we describe an algorithm that generates all integers n with $f_k(n) > 0$, where $x_1 \leq n < x_2$ for inputs k, x_1, x_2. We obtain the original algorithm from [**?**] by setting $(x_1, x_2) = (1, x)$.

Let x be the largest value of x_2 we plan to use in any application of this algorithm. As a preprocessing step, we find all primes up to $x^{1/k}$ and compute the prefix array $r[]$, where $r[0] = 0$ and $r[j] = r[j-1] + p_j^k$ where p_j is the jth prime with $p_1 = 2$.

For a particular value of n with $f_k(n) > 0$, we write $n = r[t] - r[b]$, a difference of prefix sum values, which gives its representation as $p_{b+1}^k + \cdots + p_t^k$. The trick is to generate exactly the correct values of b and t to ensure that $x_1 \leq n < x_2$. The outer loop iterates through all possible b values, and the inner loop iterates through the correct t values. Let t_s indicate the smallest t for a given b. Observe that as b increases, t_s is non-decreasing and $t_s > b$. See Algorithm **??**.

The running time of this algorithm is bounded by a constant times the number of times through the while loops and the inner for loop. Observe that the while loops increment t_s, which is bounded by $\ell = \pi(x^{1/k})$, so the total number of while-loop iterations is bounded by ℓ. The number of times we iterate through the inner for-loop is bounded by a constant times the number of times the output (n, p_{b+1}) statement executes, which in turn is $s_k(x_2) - s_k(x_1)$.

We have proven the following:

Theorem 3.1. *Given integers $k > 1$ and $x_1 < x_2 \leq x$, Algorithm **??** will list all integers n with $f_k(n) > 0$ and $x_1 \leq n < x_2$. The number of arithmetic operations used by the algorithm is at most $O(x^{1/k}/\log\log x + (s_k(x_2) - s_k(x_1)))$.*

In addition, for every representation of n as a k-gleeful number, the first prime in that representation is also given.

We have a few comments:

Algorithm 3.1 Enumerate integers n with $x_1 \le n < x_2$ and $f_k(n) > 0$

Require: Integers $k > 1$, $x_1 < x_2$, a list of all primes up to $x_2^{1/k}$, and the prefix sum array $r[\,]$
  $t_s \leftarrow 1$
  $\ell \leftarrow \pi(x_2^{1/k})$
  for $b \leftarrow 0$ to ℓ **do**
    while $t_s \le \ell$ and $t_s \le b$ **do**
      $t_s \leftarrow t_s + 1$
    end while
    while $t_s \le \ell$ and $r[t_s] - r[b] < x_1$ **do**
      $t_s \leftarrow t_s + 1$
    end while
    for $t \leftarrow t_s$ to ℓ **do**
      $n \leftarrow r[t] - r[b]$
      if $x_1 \le n < x_2$ **then**
        output (n, p_{b+1})
      else if $n \ge x_2$ **then**
        break the inner for-loop (on t)
      end if
    end for
  end for

- The understanding is that x is an upper bound on the application of the algorithm, and that it may be used on multiple intervals $[x_1, x_2)$ with $x_1 < x_2 \le x$ We are also assuming here that k is fixed, but x (and x_1, x_2) are large.
- The $x^{1/k}/\log\log x$ term is the time to compute the list of primes up to $x^{1/k}$ using, say, the Atkin-Bernstein algorithm [**?**]. If the primes are already available, this term changes to $\pi(x_2^{1/k}) = O(kx^{1/k}/\log x)$.
- In practice, we make the interval length $x_2 - x_1$ large enough so that we expect $s_k(x_2) - s_k(x_1) \gg x^{1/k}$, thereby ensuring that the cost of managing the list of primes becomes negligible.
- We obtain a practical parallel algorithm by dividing the range $(1, x)$ into equal-sized intervals of length $\Delta = x_2 - x_1$. We than assign one processor to each interval. Thus, x/Δ processors can run in parallel with little communication overhead, and the list of primes and the prefix sum array $r[\,]$ can be shared.
- When searching for duplicates, which are integers n with either $f_k(n) > 1$ or both $f_k(n) > 0$ and $f_{k'}(n) > 0$ for $k \ne k'$, the interval of size Δ should be sorted to look for matches. Note that a list of integers in a limited range can be sorted in linear time using a radix or bucket-style sort.

In practice, we found quicksort [**?**] was good enough. See [**?**, §5.2.5].

3.2. A Theoretical Parallel Algorithm.

We describe the steps and analyze the algorithm as we go. We assume an EREW PRAM parallel model with arithmetic operations on integers with $O(\log x)$ bits taking constant time. Note that $\pi(x^{1/k}) = O(kx^{1/k}/\log x)$ by the prime number theorem. As above, we use ℓ for the number of such primes.

(1) To find the primes up to $x^{1/k}$, we use the algorithm from [**?**]. This takes $O((1/k)\log x)$ time and $O(kx^{1/k}/(\log x \log\log x))$ processors.
(2) To compute the kth powers of all the primes, we use a sequential binary exponentiation algorithm that takes $O(\log k) = O(\log\log x)$ time, since we can assume $k = O(\log x)$ here. We apply this to all primes in parallel, taking $\ell = O(kx^{1/k}/\log x)$ processors.
(3) The prefix sum array $r[\,]$ can be computed in $O(\log \ell)$ time using $O(\ell/\log \ell)$ processors, or $O((1/k)\log x)$ time and $O(k^2x^{1/k}/(\log x)^2)$ processors.
(4) To start, we assign one processor to each b value from 0 to ℓ. Then, for each b in parallel, we perform a binary search on the $r[\,]$ array to find the correct start and stop t values, $(t_1(b), t_2(b))$ so that for every t with $t_1(b) \le t \le t_2(b)$, we have $0 < r[t] - r[b] < x$. This takes $O(\log \ell)$ time using $O(\ell)$ processors since this is how many b values there are.
(5) For each b, we allocate $(t_2(b) - t_1(b))/\log \ell$ additional processors. This is a total of $O(\ell + s_k(x)/\log \ell)$ processors overall.
(6) For every b, in parallel we compute $n = r[t] - r[b]$ for every $t_1(b) \le t \le t_2(b)$ and output (p_{b+1}, n). This uses the processors allocated in the previous step. Each processor may have to do up to $O(\log \ell)$ such computations, but they take constant time each. This takes $O(\log \ell)$ time using $O(\ell + s_k(x)/\log \ell)$ processors.

We have proven the following.

Theorem 3.2. *There is an EREW PRAM algorithm to find all integers $n \le x$ with $f_k(n) > 0$ that uses at most $O(\log \ell)$ time and $O(\ell + s_k(x)/\log \ell)$ processors, where $\ell := \pi(x^{1/k}) = O(kx^{1/k}/\log x)$.*

Note that this algorithm is work-optimal, and proves that computing $s_k(x)$ is in the complexity class $\mathcal{NC}$.

4. Duplicates

In this section, we examine, heuristically, the distribution of duplicates, which come in two varieties:

(1) Integers n with $f_k(n) > 1$, or
(2) Integers n with both $f_k(n) > 0$ and $f_{k'}(n) > 0$ for $k \ne k'$.

WLOG, in the second type we shall henceforth assume $k < k'$.

In the spirit of Cramér's model, we assume that an integer $n \le x$ is k-gleeful with probability given by

$$s_k(x)/x. \tag{4.1}$$

4.1. **Duplicates for $f_k(n) > 1$.** Assuming (**??**), a first try at estimating the probability an integer $n \le x$ is a duplicate with $f_k(n) > 1$ would be simply

$$\begin{aligned}\left(\frac{s_k(x)}{x}\right)^2 &\approx \frac{k^4 x^{4/(k+1)}}{x^2(\log x)^{2k/(k+1)}} \\ &= x^{4/(k+1)-2} \cdot \frac{k^4}{(\log x)^{2k/(k+1)}}.\end{aligned}$$

This probability is $o(1/x)$, unless $k < 3$. When $k = 2$ we might expect the density of duplicates to be around $x^{1/3}/(\log x)^{4/3}$, which implies there are infinitely many examples.

The only potential flaw in our logic is that two different gleeful representations for n with the same value of k must be of different lengths. Recall that

$$s_k(x) = \sum_{m=1}^{M(x,k)} s_{k,m}(x).$$

Thus, a finer estimate for the probability of a duplicate is

$$\begin{aligned}&\frac{1}{x^2}\sum_{m_1=1}^{M(x,k)} s_{m_1,k}(x) \sum_{m_2<m_1} s_{m_2,k}(x) \\ = &\frac{1}{x^2}\sum_{m_1=1}^{M(x,k)} \sum_{m_2<m_1} s_{m_1,k}(x) s_{m_2,k}(x) \\ \approx &\frac{1}{x^2}\sum_{m_1=1}^{M(x,k)} \sum_{m_2<m_1} \pi((x/m_1)^{1/k}) \cdot \pi((x/m_2)^{1/k}).\end{aligned}$$

By the prime number theorem, this is asymptotic to

$$
\begin{aligned}
&\sim \frac{1}{x^2}\sum_{m_1=1}^{M(x,k)}\sum_{m_2<m_1}\frac{k^2x^{2/k}}{(m_1m_2)^{1/k}\log(x^2/(m_1m_2))}\\
&\approx \frac{k^2x^{2/k-2}}{2\log(x/M)}\sum_{m_1=1}^{M(x,k)} m_1^{-1/k}\sum_{m_2<m_1} m_2^{-1/k}\\
&\approx \frac{k^2x^{2/k-2}}{2\log(x/M)}\sum_{m_1=1}^{M(x,k)} m_1^{-1/k}\cdot\frac{m_1^{1-1/k}}{1-1/k}\\
&\approx \frac{k^3x^{2/k-2}}{2(k-1)\log(x/M)}\sum_{m_1=1}^{M(x,k)} m_1^{1-2/k}.
\end{aligned}
$$

If $k=2$, the sum is just $M(x,2)\sim 3x^{1/3}/(\log x)^{2/3}$, which gives a probability of

$$\frac{4M}{x\log(x/M)}\sim\frac{18}{x^{2/3}(\log x)^{5/3}},$$

smaller than our first estimate by a factor of roughly $(\log x)^{1/3}$, but still enough that we expect infinitely many integers n with $f_2(n)>1$.

If $k\ge 3$, we end up with the following probability:

$$\frac{k^3x^{2/k}}{2(k-1)x^2\log(x/M)}\cdot\frac{M^{2-2/k}}{2-2/k}.$$

Plugging in our estimate that $M\approx(k+1)x^{1/(k+1)}/(\log x)^{k/(k+1)}$, we obtain the probability

$$\frac{k^3(k+1)}{2(k-1)}\cdot\frac{x^{4/(k+1)-2}}{(\log x)^{(3k-1)/(k+1)}}.$$

For $k>3$ this is clearly $o(1/x)$, as the exponent on x is less than -1. For $k=3$ the exponent on x is exactly -1, but the log factor in the denominator still gives us $o(1/x)$.

This leads us to the following conjectures.

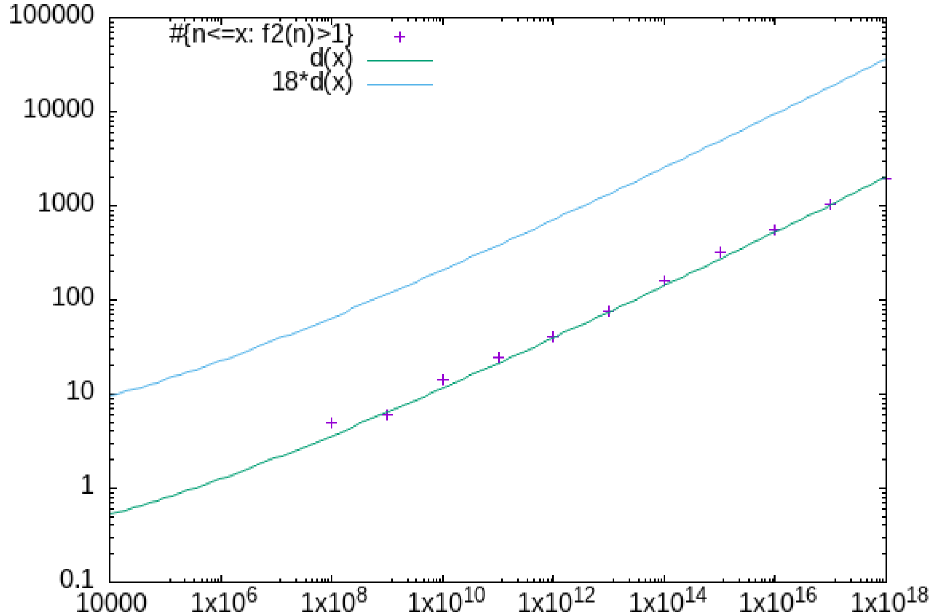
Conjecture 4.1. *There are infinitely many integers n with $f_2(n)>1$.*

We found 1950 integers $n\le 10^{18}$ with $f_2(n)>1$. Let us set $d(x):=x^{1/3}/(\log x)^{5/3}$, the number of integers n below x we expect to find with $f_2(n)>1$, with the constant factor 18 dropped. As you can see in Figure **??**, $d(x)$ lines up very nicely with our data. We currently have no explanation for why our prediction above is off by a factor of 18.

Conjecture 4.2. *For each integer $k\ge 3$, there are finitely many integers n with $f_k(n)>1$.*

With a bit more work, our heuristics also lead to this much stronger conjecture:

FIGURE 1. Comparing $d(x)$ to the number of $n \le x$ with $f_2(n) > 1$.



Conjecture 4.3. *There are finitely many integers n with $f_k(n) > 1$ for any $k \ge 3$.*

We have not found any examples n with $f_k(n) > 1, k > 2$.

Our code and data are available on the second author's github repository at `https://github.com/sorenson64/sopp`.

4.2. **Duplicates for $f_k(n) > 0$ and $f_{k'}(n) > 0$ with $k < k'$.** We continue to assume $k < k'$. The heuristic probability that a randomly chosen integer $n \le x$ has both $f_k(n), f_{k'}(n) > 0$ is at most

$$\frac{s_k(x)s_{k'}(x)}{x^2} \approx (kk')^2 \frac{x^{2/(k+1)+2/(k'+1)-2}}{(\log x)^{2k/(k+1)+2k'/(k'+1)}}.$$

With the log factors in the denominator, we can expect infinitely many examples if the exponent on x is strictly greater than -1, that is, $2/(k+1) + 2/(k'+1) - 2 > -1$, or

$$\frac{2}{k+1} + \frac{2}{k'+1} > 1.$$

With $k' > k$, this is not true when $k \ge 3$. With $k = 2$, we then require $2/(k'+1) > 1/3$. This gives $k' = 3$ or 4.

Conjecture 4.4. *For $k' = 3$ or 4, there are infinitely many integers n with both $f_2(n) > 1$ and $f_{k'}(n) > 1$.*

We have 3 examples of 2-3 duplicates up to 10^{18}:

23939
432958700126053
137610738498311684

We found no 2-4 duplicates below 10^{18}. We are hopeful that more examples will eventually be found.

Conjecture 4.5. *For $k < k'$, if $k \geq 3$ or $k' \geq 5$ then there are finitely many integers n with both $f_k(n) > 1$ and $f_{k'}(n) > 1$.*

5. Acknowledgements

This work was supported in part by the Mathematics Research Camp at Butler University in August 2023, and by a grant from the Holcomb Awards Committee.

Thanks to Frank Levinson for supporting Butler's research computing infrastructure.

This work began when the first author was an undergraduate student at Butler University.

References

[1] The On-Line Encyclopedia of Integer Sequences. `https://oeis.org`.
[2] A. O. L. Atkin and D. J. Bernstein. Prime sieves using binary quadratic forms. *Math. Comp.*, 73(246):1023–1030 (electronic), 2004.
[3] R. Greenlaw, H. J. Hoover, and W. L. Ruzzo. *Limits to Parallel Computation*. Oxford University Press, 1995.
[4] Richard K. Guy. *Unsolved problems in number theory*. Problem Books in Mathematics. Springer-Verlag, New York, third edition, 2004.
[5] C. A. R. Hoare. Quicksort. *The Computer Journal*, 5:10–15, 1962.
[6] D. E. Knuth. *The Art of Computer Programming: Sorting and Searching*, volume 3. Addison-Wesley, Reading, Mass., 3rd edition, 1998.
[7] L. Moser. Notes on number theory. III. On the sum of consecutive primes. *Canad. Math. Bull.*, 6:159–161, 1963.
[8] Cathal O'Sullivan, Jonathan P. Sorenson, and Aryn Stahl. Algorithms and bounds on the sums of powers of consecutive primes. *Integers*, 24:Paper No. A4, 14, 2024.
[9] J. H. Reif, editor. *Synthesis of Parallel Algorithms*. Morgan Kaufman, San Mateo, California, 1993.
[10] Barkley Rosser. The n-th Prime is Greater than nlogn. *Proc. London Math. Soc. (2)*, 45(1):21–44, 1939.
[11] J. B. Rosser and L. Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois Journal of Mathematics*, 6:64–94, 1962.
[12] Jonathan Sorenson and Ian Parberry. Two fast parallel prime number sieves. *Inform. and Comput.*, 114(1):115–130, 1994.
[13] Janyarak Tongsomporn, Saeree Wananiyakul, and Jörn Steuding. Sums of consecutive prime squares. *Integers*, 22, 2022. A9.

Department of Physics & Astronomy, University of Rochester, Rochester, New York 14627
Email address: `smoore55@u.rochester.edu`

Computer Science and Software Engineering Department, Butler University, Indianapolis, Indiana 46208
Email address: `jsorenso@butler.edu`