

# On Wagner’s $k$ -Tree Algorithm Over Integers

Haoxing Lin\*

Prashant Nalini Vasudevan†


October 10, 2024


## Abstract

The  $k$ -Tree algorithm [Wag02] is a non-trivial algorithm for the average-case  $k$ -SUM problem that has found widespread use in cryptanalysis. Its input consists of  $k$  lists, each containing  $n$  integers from a range of size  $m$ . Wagner’s original heuristic analysis [Wag02] suggested that this algorithm succeeds with constant probability if  $n \approx m^{1/(\log k+1)}$ , and that in this case it runs in time  $O(kn)$ . Subsequent rigorous analysis of the algorithm [Lyu05, Sha08, JKL24] has shown that it succeeds with high probability if the input list sizes are significantly larger than this.

We present a broader rigorous analysis of the  $k$ -Tree algorithm, showing upper and lower bounds on its success probability and complexity for any size of the input lists. Our results confirm Wagner’s heuristic conclusions, and also give meaningful bounds for a wide range of list sizes that are not covered by existing analyses. We present analytical bounds that are asymptotically tight, as well as an efficient algorithm that computes (provably correct) bounds for a wide range of concrete parameter settings. We also do the same for the  $k$ -Tree algorithm over  $\mathbb{Z}_m$ . Finally, we present experimental evaluation of the tightness of our results.

---

\*  National University of Singapore. Email: [haoxingl@comp.nus.edu.sg](mailto:haoxingl@comp.nus.edu.sg)

†  National University of Singapore. Email: [prashant@comp.nus.edu.sg](mailto:prashant@comp.nus.edu.sg)

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Our Results . . . . .	4
1.2	Technical Overview . . . . .	7
1.3	Related Work . . . . .	15
<b>2</b>	<b>Analysis</b>	<b>17</b>
2.1	Tools . . . . .	23
2.2	Proof of Proposition 2.3 . . . . .	27
2.3	Proof of Proposition 2.4 . . . . .	31
2.4	Proof of Proposition 2.5 . . . . .	40
2.5	$k$ -Tree over $\mathbb{Z}_m$ . . . . .	42
<b>3</b>	<b>Computed Bounds</b>	<b>43</b>
3.1	Algorithms . . . . .	44
3.2	Computed Bounds for $\mathbb{Z}_m$ . . . . .	55
3.3	Visualizations and Interpretations . . . . .	57
3.4	Practical Implications . . . . .	60
3.5	Limitations . . . . .	60
<b>4</b>	<b>Experiments</b>	<b>60</b>
4.1	Experimental Setup . . . . .	61
4.2	Success Probability . . . . .	62
4.3	Complexity . . . . .	64

# 1 Introduction

The (average-case)  $k$ -SUM problem is a basic computational problem that comes up relatively often in cryptanalysis [Sch01, Wag02, DEF<sup>+</sup>19, BLL<sup>+</sup>22, ...]. In this problem, given  $k$  lists of  $n$  integers each drawn uniformly at random from the range  $\{-\lfloor m/2 \rfloor, \dots, \lfloor m/2 \rfloor\}$ , the task is to find a set of  $k$  integers, one from each list, such that their sum is 0. Such sets exist with high probability once the list size  $n$  is larger than  $m^{1/k}$ ; and in this case there is a simple meet-in-the-middle algorithm that runs in time  $\tilde{O}(m^{1/2})$ . The only known algorithms that do significantly better than this are the  $k$ -Tree algorithm [Wag02] and its extensions [MS12, NS15, Din19] and variants [BKW03, Lyu05, JKL24].

The  $k$ -Tree algorithm, as formulated by Wagner [Wag02], is described in Figure 1.<sup>1</sup> Wagner provided heuristic arguments indicating that this algorithm succeeds with constant probability if the input lists are of size  $n \approx m^{1/(\log k+1)}$ , in which case it runs in time  $O(k \cdot m^{1/(\log k+1)})$ . Even for modest values of  $k$  (say 4 or 8), this runtime is significantly better than the earlier  $\tilde{O}(m^{1/2})$  for large values of  $m$ . For this reason, the  $k$ -Tree algorithm has been used repeatedly in the cryptanalysis of signatures [Wag02, DEF<sup>+</sup>19, BLL<sup>+</sup>22], hash functions [CJ04, BC22], identification schemes [LF06], symmetric-key encryption schemes [Jou03], etc., to provide simple attacks that perform significantly better than brute force.

**The  $k$ -Tree algorithm**

**Parameters:**  $k, n, m \in \mathbb{N}$ , with  $k$  being a power of 2

**Input:** Lists  $L_1, \dots, L_k$ , each consisting of  $n$  integers from  $\{-\lfloor m/2 \rfloor, \dots, \lfloor m/2 \rfloor\}$

**Output:** Indices  $\ell_1, \dots, \ell_k \in [n]$ , or a failure symbol  $\perp$

**Procedure:**

1. Initialization:
  - Set  $p = m^{1/(\log k+1)}$
  - For each  $i \in [k]$ , denote the list  $L_i$  by  $L_i^0$
  - Set  $\tau \leftarrow m/2$
2. For  $d$  from 1 to  $\log k$ :
  - Set  $\tau \leftarrow p \cdot \tau$
  - For  $i \in [\frac{k}{2^d}]$ :
    - compute  $L_i^d \leftarrow \left\{ (a+b) \mid a \in L_{2i-1}^{d-1} \wedge b \in L_{2i}^{d-1} \wedge |a+b| \leq \tau \right\}$
3. If  $L_1^{\log k}$  contains 0, output the indices in the input lists that led to this sum. Otherwise output  $\perp$ .

Figure 1: The  $k$ -Tree algorithm over Integers

<sup>1</sup>See Figure 5 in Section 2 for a more detailed presentation.

In spite of such widespread relevance, the complexity of the  $k$ -Tree algorithm is still not well understood. Wagner’s aforementioned heuristic analysis was based on assumptions that are not strictly true, though empirical evidence suggests that his conclusion is valid.<sup>2</sup> Rigorous analysis of the algorithm has since been carried out by Lyubashevsky [Lyu05], Shallue [Sha08], and Joux et al. [JKL24], but these have not been able to confirm this conclusion (see Section 1.3 for details). Further, we still do not have the means to provide good answers to basic questions like the following, even heuristically:

1. What is the probability that the algorithm will succeed if the lists are of size  $(10 \cdot m^{1/(\log k+1)})$  or  $(0.1 \cdot m^{1/(\log k+1)})$ ?
2. What size of lists is necessary or sufficient for the algorithm to succeed with probability 0.01?
3. What is the complexity of the algorithm for these list sizes?

Answers to these questions for some pairs of values  $(m, k)$  for which these list sizes are not too large – e.g.,  $(2^{64}, 4)$ ,  $(2^{256}, 128)$  – can be obtained empirically, but it is not clear how to extrapolate these to other settings that may come out of concrete parameter choices in constructions – e.g.,  $(2^{256}, 4)$ . Our objective in this work is to provide a tight analysis of the  $k$ -Tree algorithm that can help researchers answer questions like the ones above and thus understand more accurately the power of attacks that make use of the  $k$ -Tree algorithm.

**Paper Outline.** We summarize our results in Section 1.1, and provide an overview of our techniques in Section 1.2. In Section 1.3, we describe prior work on this topic, present comparisons to our results, and discuss other related work. In Section 2, we provide the complete proof of our main theorem regarding the behavior of the  $k$ -Tree algorithm. In Section 3, we present pseudo-code for computing tighter bounds on the behavior of the algorithm. In Section 4, we present experimental evaluation of our bounds for concrete parameters.

## 1.1 Our Results

Recall that we are interested in the performance of the  $k$ -Tree algorithm when given as input  $k$  lists of  $n$  numbers each drawn uniformly at random from  $\{-\lfloor m/2 \rfloor, \dots, \lfloor m/2 \rfloor\}$ . Here  $k$  is a power of 2. With these parameters, we define the *complexity* of the  $k$ -Tree algorithm to be the expected total size of all the lists that are generated in the course of its execution.<sup>3</sup>

**Analytical Bounds.** Our first set of results are analytical bounds on the success probability and complexity of the  $k$ -Tree algorithm for any list size.

**Theorem 1.** *Consider any  $k, n, m \in \mathbb{N}$ , where  $k \geq 4$  is a power of 2 and  $m > 30^{\log k+1}$ . Set  $p = m^{\frac{-1}{\log k+1}}$  and  $c = n/m^{\frac{1}{\log k+1}}$ . The probability of success of the  $k$ -Tree algorithm is bounded as:*

$$\frac{c^k}{1 + c^k \cdot \left(1 + \frac{k}{n}\right)^k} \cdot (1 - 150p)^k \leq \Pr[k\text{-Tree succeeds}] \leq c^k \cdot (1 + 37p)^k$$

<sup>2</sup>See, for instance, the results of our experiments presented in Section 4.

<sup>3</sup>See Remark 1.3 at the end of this subsection for a brief discussion of this complexity measure.

Its complexity is bounded as follows:

$$\text{Complexity of } k\text{-Tree} \in kn \cdot \left( 1 + \sum_{d \in [\log k]} \frac{c^{2^d - 1}}{2^d} \right) \cdot (1 \pm 37p)^{k-1}$$

Asymptotically, the following simpler bound can be inferred from the above theorem if  $k$  is not too large in relation to the other parameters. (For a more rigorous statement, see Corollary 2.2 in Section 2.)

**Corollary 1.1.** *Consider the same parameters as in Theorem 1, and in addition suppose that  $k = o(1/p)$  and  $k = o(n^{1/2})$ . Then we have:*

$$\frac{c^k}{1 + c^k} \cdot (1 - o(1)) \leq \Pr [k\text{-Tree succeeds}] \leq c^k \cdot (1 + o(1))$$

The setting  $c = 1$  corresponds to the choice of  $n = m^{1/(\log k + 1)}$  for the lists sizes suggested by Wagner’s heuristic argument [Wag02]. As long as  $k$  is not too large, the above results confirm that in this case the  $k$ -Tree algorithm succeeds with some probability roughly between 1/2 and 1. In addition, they also allow one to compute list sizes and the resulting complexity that is sufficient for the algorithm to succeed with probability, say, 0.01; or even the complexity that is *necessary* for the algorithm to succeed with probability at least 0.01. This enables answering the various kinds of questions about the algorithm presented earlier in this section.

**Computational Bounds.** There are still some meaningful concrete values of  $(m, k)$  (for example,  $(2^{256}, 1024)$ ), however, for which Theorem 1 does not give accurate bounds. This is due to two sources of inaccuracy in our results above: gaps that are inherent to our technique itself, and approximations that were made in the course of our proof to make intermediate expressions easier to handle analytically. Of these, the latter turn out to substantially dominate.

We eliminate this second source of inaccuracy by implementing our entire proof as a program that, instead of attempting to produce simple analytical expressions, explicitly computes the resulting probability and complexity bounds given values for the parameters  $k$ ,  $m$ , and  $n$ . The pseudo-code for the relevant algorithms are presented in Section 3, and our implementation of it is available in the associated repository [Lin24].

**Theorem 2.** *Consider any  $k, n, m \in \mathbb{N}$ , where  $k \geq 4$  is a power of 2, and let  $p = m^{-\frac{1}{\log k + 1}}$ . The pair of values output by `PROBBOUNDS`( $m, k, n, p$ ) bound the success probability of the  $k$ -Tree algorithm, and those output by `SIZEBOUNDS`( $m, k, n, p$ ) bound its complexity. Further, these algorithms run in time  $\text{polylog}(k, m, n)$ .*

We find that this tightening does significantly strengthen the resulting bounds for various concrete values of  $(m, k)$ . We plot the bounds on success probability from both these theorems for a couple of pairs of values of  $(m, k)$  in Figure 2 for illustration. For the rest of this section, we solely use the bounds from Theorem 2 as they are more accurate.

**Using Our Bounds.** We expect our results to be useful in a few different ways. In applications of the  $k$ -Tree algorithm where the size of the range  $m$  and the number of lists  $k$  are fixed and not in

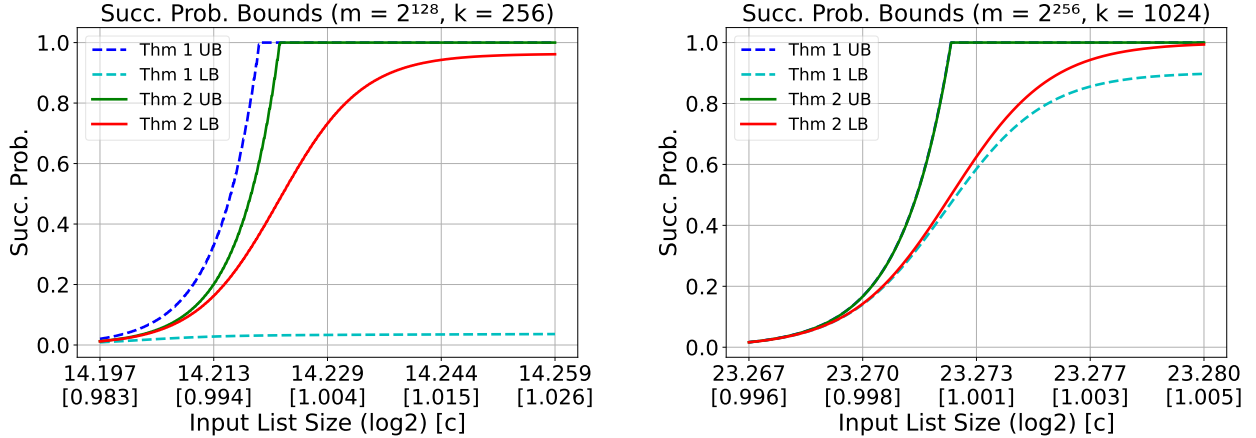


Figure 2: Plot of upper and lower bounds on success probabilities against the input list size  $n$ . Bounds from both Theorems 1 and 2 are represented. The quantity  $c$  is defined to be the ratio  $n/m^{1/(\log k+1)}$ . Labels on the x-axis are  $\log_2(n)$ , with the corresponding value of  $c$  in square brackets.

the control of the algorithm designer (e.g. the attack in [CJ04], solving the ROS problem [Wag02, BLL+22]), our bounds can be used to determine the list size (and the corresponding complexity) that is sufficient to achieve a specific desired probability of success. This may be done analytically using Theorem 1 if  $k$  is relatively small, or computationally using Theorem 2 (with a binary search for  $n$ ) for larger values of  $k$ .

In applications where the range  $m$  is fixed but  $k$  may be determined by the algorithm designer (e.g. attacks in [DEF+19, Jou03], the attacks on hash functions in [Wag02]), our bounds can also be used to select the best value of  $k$  together with the best value of  $n$  (as the complexity depends on both) for a desired probability of success. In Figure 3, we present some sample computations of this form, covering some of the parameter settings from the papers cited above as examples.

Finally, our upper bounds on the probability of success of the  $k$ -Tree algorithm can be used to obtain concrete lower bounds on the complexity of  $k$ -Tree-based attacks on candidate cryptographic constructions that achieve any given probability of success.

**Experimental Evaluation.** To understand the tightness of our upper and lower bounds, we empirically estimate the values of the quantities bounded in the theorems above for various values of the parameters  $k$ ,  $m$ , and  $n$ , and compare them to our predictions. We refer the reader to Section 4 for details of these experiments, and a summary of their results and implications.

**Remark 1.2** (Simple Optimizations). *There are some simple optimizations to the  $k$ -Tree algorithm that can significantly improve its behavior in some parameter regimes. For instance, removing duplicates when computing the intermediate lists  $L_i^d$  ensures that the complexity of the algorithm is at most  $O(k(n+m))$ , which is smaller than the bounds produced by Theorems 1 and 2 for very large values of  $c$ .*

*For large values of  $c$ , the lower bound on the success probability produced by these theorems is roughly  $(1 - c^{-k})$ . But for such  $c$ , simply breaking up the lists into  $\lfloor c \rfloor$  smaller lists with  $m^{1/(\log k+1)}$  elements each and running  $k$ -Tree on each separately (effectively with “ $c$ ” = 1) results in an amplified success probability of  $(1 - e^{-\Omega(c)})$ .*

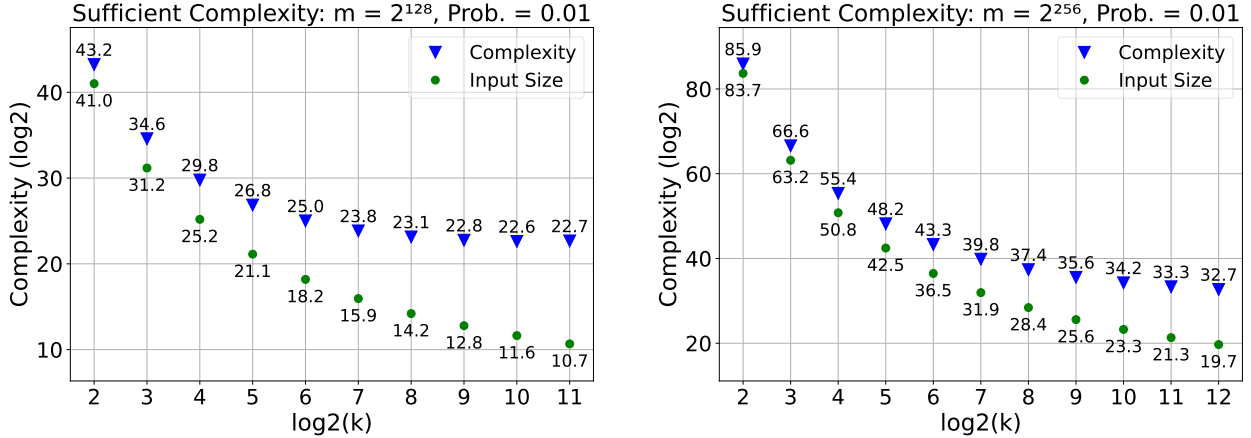


Figure 3: Plots of the complexity of the  $k$ -Tree algorithm that is sufficient to provably achieve success probability of 0.01, against  $k$ . Input list size in each case is chosen so that the lower bound on success probability is slightly larger than 0.01. Computed using Theorem 2.

**Remark 1.3** (Measure of Complexity). *Depending on various incidental factors such as the hardware being used, its word-size in relation to  $m$ , etc., the actual running time of the optimal implementation of the  $k$ -Tree algorithm might be anything from a constant factor to a logarithmic factor larger than the total list size. However, in all cases, the total list size is the central quantity that determines the running time. So for the sake of generality and simplicity, we directly use this as our measure of complexity.*

**Remark 1.4** (Integers vs.  $\mathbb{Z}_m$ ). *So far, we have been discussing the  $k$ -SUM problem where the addition is over integers. In reality, the problem that is most often useful in cryptanalysis, etc., is the problem where addition is done modulo  $m$  (that is, over  $\mathbb{Z}_m$ ), often for some prime number  $m$ . In this case, the  $k$ -Tree algorithm is modified to perform all its additions modulo  $m$  instead of over integers, with  $\mathbb{Z}_m$  being identified with the set  $\{-\lfloor m/2 \rfloor, \dots, \lfloor m/2 \rfloor\}$  in the natural manner. Prior analyses by Shallue [Sha08] and Joux et al. [JKL24] were also for  $k$ -Tree over  $\mathbb{Z}_m$ . However, as demonstrated in Section 2.5, the difference in the behavior of the  $k$ -Tree algorithm in these two cases is so small as to be beneath the precision with which we state our results in this section. So for simplicity, we continue to ignore this distinction in this section, though later sections will take it into account.*

## 1.2 Technical Overview

In this section, we present an overview of our proof of the bounds on the success probability of the  $k$ -Tree algorithm presented in Theorem 1. Our bounds on the complexity of the algorithm follow from similar arguments. The algorithms captured in Theorem 2 follow the same high-level approach as this proof, but involve some careful re-framing of its terms in order to enable efficient computation of some intermediate values. The entire rigorous proof is presented in Section 2.

Our approach to these bounds is quite elementary – we define a random variable corresponding to the number of occurrences of “0” in the final list computed by the  $k$ -Tree algorithm, compute bounds on its first and second moments, and then use standard second-moment-based tail bounds

to bound the probability that this variable has non-zero value. The challenge lies in computing these moments, as this variable is somewhat complex.

**Setup.** Recall that the input to the  $k$ -Tree algorithm is  $k$  lists of  $n$  integers, with each integer being drawn uniformly at random from the set  $\{-\lfloor m/2 \rfloor, \dots, \lfloor m/2 \rfloor\}$ . Given these parameters, define the symbols  $p = m^{-1/(\log k+1)}$ , which is the filtering parameter in the algorithm, and  $c = n \cdot p$ , which represents the size of the lists relative to the “default value” of  $m^{1/(\log k+1)}$  suggested by heuristic arguments. For any non-negative integer  $s$ , we will denote by  $\langle s \rangle$  such a set  $\{\lfloor s/2 \rfloor, \dots, \lfloor s/2 \rfloor\}$ .

For simplicity, we will fix the number of lists  $k$  to 4 in this overview (and so  $p = m^{-1/3}$ ). In this case, the  $k$ -Tree algorithm is given as inputs  $k = 4$  lists  $L_1, \dots, L_4$ , each of size  $n$ , and proceeds as follows:

1. Compute the following lists:

- $L_1^1 \leftarrow \{(a+b) \mid a \in L_1 \wedge b \in L_2 \wedge (a+b) \in \langle mp \rangle\}$
- $L_2^1 \leftarrow \{(a+b) \mid a \in L_3 \wedge b \in L_4 \wedge (a+b) \in \langle mp \rangle\}$
- $L_1^2 \leftarrow \{(a+b) \mid a \in L_1^1 \wedge b \in L_2^1 \wedge (a+b) \in \langle mp^2 \rangle\}$

2. Succeed if  $L_1^2$  contains a 0, fail otherwise.

Above, take the lists  $L_i^d$  to each be a multi-set instead of a set – for instance, if a specific value of  $(a+b) \in \langle mp \rangle$  occurs twice as the sum of elements in  $L_1$  and  $L_2$ , then two such values will be present in  $L_1^1$ . This makes the algorithm sub-optimal in terms of efficiency, but does not change its success probability and makes it easier to analyze. Define the following random variable as described earlier:

$$C = \text{number of occurrences of } 0 \text{ in the list } L_1^2$$

The algorithm succeeds exactly when the value of this non-negative variable  $C$  is at least 1. So the probability of this latter event is what we will be concerned with in our analysis.

For any set of input lists  $\{L_i\}$ , the value of  $C$  may be computed by iterating through every possible value of the indices  $\ell_1, \dots, \ell_4 \in [n]$ , and considering whether the tuple  $(L_1[\ell_1], \dots, L_4[\ell_4])$  passes all the “filters” of the algorithm and also sums to 0. For each such  $\bar{\ell} = (\ell_1, \dots, \ell_4)$ , we capture this by defining a variable  $C_{\bar{\ell}}$  that is 1 if all of the following events occur and is 0 otherwise:

$$\begin{aligned} E_1 &\equiv (L_1[\ell_1] + L_2[\ell_2] \in \langle mp \rangle) \\ E_2 &\equiv (L_3[\ell_3] + L_4[\ell_4] \in \langle mp \rangle) \\ E_3 &\equiv (L_1[\ell_1] + L_2[\ell_2] + L_3[\ell_3] + L_4[\ell_4] \in \langle mp^2 \rangle) \\ E_4 &\equiv (L_1[\ell_1] + L_2[\ell_2] + L_3[\ell_3] + L_4[\ell_4] = 0) \end{aligned}$$

Note that  $C_{\bar{\ell}}$  is a random variable, with randomness coming from the numbers in the input lists. The variable  $C$  can now be written as follows:

$$C = \sum_{\bar{\ell} \in [n]^4} C_{\bar{\ell}} \tag{1}$$

We will be interested in the moments of this random variable  $C$ , which are functions of  $m$  and  $n$  (since we have already fixed  $k$ ).



**Heuristic Analysis.** Wagner’s heuristic analysis [Wag02] of the  $k$ -Tree algorithm essentially relies on the following three assumptions (for any large enough integer  $s$ ):

- (a) When  $\mathbf{E}[C] = 1$ , the algorithm succeeds with constant probability.
- (b) When  $x$  and  $y$  are sampled uniformly at random from  $\langle s \rangle$ , their sum  $(x + y)$  is contained in  $\langle sp \rangle$  with probability  $p$ .
- (c) With  $x$  and  $y$  sampled uniformly at random from  $\langle s \rangle$ , the distribution of  $(x + y)$ , when conditioned on being contained in  $\langle sp \rangle$ , is uniform over  $\langle sp \rangle$ .

Fix some tuple of indices  $\bar{\ell} = (\ell_1, \dots, \ell_4)$ . Under assumptions (b) and (c), the expectation of  $C_{\bar{\ell}}$  can be computed by computing the probability of events  $E_1, \dots, E_4$  as follows:

- Assumption (b) implies that events  $E_1$  and  $E_2$  each happens with probability  $p$ .
- Conditioned on these happening, assumption (c) implies that the sums  $(L_1[\ell_1] + L_2[\ell_2])$  and  $(L_3[\ell_3] + L_4[\ell_4])$  are uniformly distributed over  $\langle mp \rangle$ .
- So by (b),  $E_3$  also happens with probability  $p$ .
- Finally, appealing to (c) again, conditioned on the first three events happening, the entire sum is distributed uniformly over  $\langle mp^2 \rangle$ , and so  $E_4$  happens with probability  $(mp^2)^{-1}$ .

Thus, we have:

$$\begin{aligned} \mathbf{E}[C_{\bar{\ell}}] &= \Pr[E_1 \wedge \dots \wedge E_4] \\ &= \Pr[E_1] \cdot \Pr[E_2] \cdot \Pr[E_3 \mid E_1 \wedge E_2] \cdot \Pr[E_4 \mid E_1 \wedge E_2 \wedge E_3] \\ &= p \cdot p \cdot p \cdot \frac{1}{mp^2} = \frac{p}{m} = p^4 \end{aligned} \tag{2}$$

The expected value of  $C$  is then:

$$\mathbf{E}[C] = \sum_{\bar{\ell} \in [n]^4} C_{\bar{\ell}} = n^4 \cdot p^4 = c^4 \tag{3}$$

Thus, when  $c = 1$  (that is,  $n = m^{1/3}$ ), this expectation is 1, and by assumption (a), the algorithm works with constant probability. This was the conclusion in [Wag02]. In providing a rigorous analysis of the  $k$ -Tree algorithm, our objective is to remove the above assumptions. We show that assumptions (b) and (c), while not strictly true, are close enough to being true. Further, we show that a generalization of assumption (a) is true, which allows us obtain bounds on the success probability of the algorithm for a wide range of values of  $c$  rather than just  $c = 1$ .

**Relaxing the Assumptions.** To compute the actual expectation of  $C_{\bar{\ell}}$ , we start by first showing that slightly relaxed versions of assumptions (b) and (c) above are true. Starting with assumption (b), it can be shown using some elementary computation (see Section 2.1) that:

$$\Pr_{x, y \leftarrow \langle s \rangle} [x + y \in \langle sp \rangle] \in \left( p - \frac{p^2}{4} \right) \pm O\left( \frac{1}{s} \right) \tag{4}$$

The right-hand side above is not exactly  $p$ , but for typical values of  $p$  and  $s$ , is quite close to it.

For assumption (c), we show that while the distribution of  $(x + y)$  as described there is not actually uniform, it is close to the uniform distribution. The specific notion of distance we use, which we refer to as the *Max-Ratio (MR) distance* from Uniform, is defined as follows<sup>4</sup> for a distribution  $D$  over a domain  $S$ :

$$\Delta_{\text{MR}}(U, D) = \frac{\max_{x \in S} D(x)}{\min_{x \in S} D(x)}$$

where  $D(x)$  is the probability mass placed on  $x$  by the distribution  $D$ . This distance is at least 1 (which is achieved if  $D$  is the uniform distribution over  $S$ ), and is potentially unbounded.

This notion of distance is particularly beneficial for our analysis for a couple of reasons. First, for any event  $E$  over the domain  $S$ , we can bound the probability of  $E$  happening under  $D$  in terms of the probability of  $E$  happening under the uniform distribution over  $S$  as follows:

$$\Delta_{\text{MR}}(U, D)^{-1} \cdot \Pr_{x \leftarrow S}[E(x)] \leq \Pr_{x \leftarrow D}[E(x)] \leq \Delta_{\text{MR}}(U, D) \cdot \Pr_{x \leftarrow S}[E(x)] \quad (5)$$

Second, this distance is easy to compute for distributions defined under conditioning, which can otherwise be difficult to reason about. In particular, suppose  $D$  is the distribution of  $(x + y)$  conditioned on being in  $\langle sp \rangle$  when  $x$  and  $y$  are uniformly drawn from  $\langle s \rangle$ . Then, this distance is as follows:

$$\begin{aligned} \Delta_{\text{MR}}(U, D) &= \frac{\max_{z \in \langle sp \rangle} \Pr_{x, y \leftarrow \langle s \rangle}[x + y = z \mid x + y \in \langle sp \rangle]}{\min_{z \in \langle sp \rangle} \Pr_{x, y \leftarrow \langle s \rangle}[x + y = z \mid x + y \in \langle sp \rangle]} \\ &= \frac{\max_{z \in \langle sp \rangle} \Pr_{x, y \leftarrow \langle s \rangle}[x + y = z] \cdot \Pr[x + y \in \langle sp \rangle]^{-1}}{\min_{z \in \langle sp \rangle} \Pr_{x, y \leftarrow \langle s \rangle}[x + y = z] \cdot \Pr[x + y \in \langle sp \rangle]^{-1}} \\ &= \frac{\max_{z \in \langle sp \rangle} \Pr_{x, y \leftarrow \langle s \rangle}[x + y = z]}{\min_{z \in \langle sp \rangle} \Pr_{x, y \leftarrow \langle s \rangle}[x + y = z]} \\ &\leq (1 + p) + O\left(\frac{1}{s}\right) \end{aligned} \quad (6)$$

where the second equality follows from the Bayes theorem, and the inequality again follows from elementary computations. Again, for typical values of  $p$  and  $s$ , the right-hand side above is quite close to 1, indicating that the distribution is close to uniform. Putting together (5) and (6), with some further computation, we get the following effective relaxation of assumption (c). For any event  $E(z)$  defined over domain  $\langle sp \rangle$ , we have:

$$\Pr_{x, y \leftarrow \langle s \rangle}[E(x + y) \mid (x + y) \in \langle sp \rangle] \in \Pr_{z \leftarrow \langle sp \rangle}[E(z)] \cdot \left[1 \pm \left(p + O\left(\frac{1}{s}\right)\right)\right] \quad (7)$$

For simplicity, in the rest of this overview we will ignore the  $O(1/s)$  parts in the expressions (4) and (7) above. It is not a crucial part of the big picture, and in most reasonable parameter settings is much smaller than  $p$  anyway.

---

<sup>4</sup>This is again a simplification. For the actual definition, see Section 2.1.

**Computing the Expectation.** With (4) and (7) in hand, we can now compute the expectation of  $C_{\bar{\ell}}$ . We essentially follow the earlier heuristic argument step-by-step, replacing the assumptions there with their true but relaxed versions.

Denote by  $\bar{x}$  the vector  $(x_1, \dots, x_4)$ , where  $x_i \in \langle m \rangle$  will later be identified with  $L_i[\ell_i]$ ; the variables  $x_1^1, x_2^1$ , and  $x_1^2$  below will similarly initially be identified with  $(x_1 + x_2), (x_3 + x_4)$ , and  $(x_1^1 + x_2^1)$ , respectively. We re-state the events in definition of  $C_{\bar{\ell}}$  to be parameterized as follows for ease of manipulation:

$$\begin{aligned} E_1(x_1, x_2) &\equiv (x_1 + x_2 \in \langle mp \rangle) \\ E_2(x_3, x_4) &\equiv (x_3 + x_4 \in \langle mp \rangle) \\ E_3(x_1^1, x_2^1) &\equiv (x_1^1 + x_2^1 \in \langle mp^2 \rangle) \\ E_4(x_1^2) &\equiv (x_1^2 = 0) \end{aligned}$$

Along the lines of (2), the expectation of  $C_{\bar{\ell}}$  for any  $\bar{\ell}$  can be written as follows:

$$\begin{aligned} \mathbf{E}[C_{\bar{\ell}}] &= \Pr_{x_1, \dots, x_4 \leftarrow \langle m \rangle} \left[ \begin{array}{c} E_1(x_1, x_2) \wedge E_2(x_3, x_4) \wedge \\ E_3(x_1 + x_2, x_3 + x_4) \wedge E_4(x_1 + \dots + x_4) \end{array} \right] \\ &= \Pr_{x_1, x_2 \leftarrow \langle m \rangle} [E_1(x_1, x_2)] \cdot \Pr_{x_3, x_4 \leftarrow \langle m \rangle} [E_2(x_3, x_4)] \\ &\quad \cdot \Pr_{x_1, \dots, x_4 \leftarrow \langle m \rangle} \left[ \begin{array}{c} E_3(x_1 + x_2, x_3 + x_4) \wedge E_4(x_1 + \dots + x_4) \\ | E_1(x_1, x_2) \wedge E_2(x_3, x_4) \end{array} \right] \end{aligned} \quad (8)$$

The first two terms in the product above can be bounded using (4) as follows:

$$\Pr_{x_1, x_2 \leftarrow \langle m \rangle} [E_1(x_1, x_2)] = \Pr_{x_3, x_4 \leftarrow \langle m \rangle} [E_2(x_3, x_4)] \approx \left( p - \frac{p^2}{4} \right) \quad (9)$$

The last term is more complex, but a useful observation here is that the events in the probability expression there only depend on the sums  $(x_1 + x_2)$  and  $(x_3 + x_4)$  rather than on the  $x_i$ 's directly. Further, these events are conditioned on these sums being contained in  $\langle mp \rangle$  (that is, conditioned on the events  $E_1$  and  $E_2$ ). If assumption (c) had been true, it would have implied that these sums are uniformly distributed over  $\langle mp \rangle$ , simplifying the probability expression considerably. We can do the same using (7), albeit with some loss as follows:

$$\begin{aligned} &\Pr_{x_1, x_2, x_3, x_4 \leftarrow \langle m \rangle} \left[ \begin{array}{c} E_3(x_1 + x_2, x_3 + x_4) \wedge E_4((x_1 + x_2) + (x_3 + x_4)) \\ | E_1(x_1, x_2) \wedge E_2(x_3, x_4) \end{array} \right] \\ &\in \Pr_{x_1^1, x_1^2 \leftarrow \langle mp \rangle} [E_3(x_1^1, x_2^1) \wedge E_4(x_1^1 + x_2^1)] \cdot (1 \pm p)^2 \end{aligned} \quad (10)$$

Putting together (8-10), we get:

$$\mathbf{E}[C_{\bar{\ell}}] \in p^2 \cdot (1 \pm p)^4 \cdot \Pr_{x_1^1, x_1^2 \leftarrow \langle mp \rangle} [E_3(x_1^1, x_2^1) \wedge E_4(x_1^1 + x_2^1)] \quad (11)$$

The probability expression above is similar to the one we started with in (8), except that it is for the case of  $k = 2$  and the range  $\langle mp \rangle$  instead of  $k = 4$  and range  $\langle m \rangle$ . For general  $k$ , we get a similar recursive expression in terms of  $k/2$  that enables us to compute the overall bound efficiently,

both analytically and computationally. In the present case, we simply have to apply (4) and (7) in turn once more to get the following final bound:

$$\mathbf{E}[C_{\bar{\ell}}] \in p^3 \cdot (1 \pm p)^6 \cdot \frac{1}{mp^2} \subseteq p^4 \cdot (1 \pm O(p)) \quad (12)$$

The above is again quite close to its heuristic evaluation in (2). The expectation of  $C$  can then be computed as in (3) to get:

$$\mathbf{E}[C] \in c^4 \cdot (1 \pm O(p)) \quad (13)$$

The upper bound on expectation above already gives us an upper bound on the success probability of the  $k$ -Tree algorithm using the Markov inequality. It remains, then, to show a corresponding lower bound.

**Concentration.** Next we move on to proving that (a generalization of) assumption (a) is valid. Our hope here is to show that when the expectation of  $C$  is significantly larger than 0, then with somewhat high probability, we have  $C \neq 0$  – that is, the algorithm succeeds. Clearly, this is not true of arbitrary random variables – if  $C$  was 0 with probability  $(1 - n^{-4})$  and  $n^4$  with probability  $n^{-4}$ , it would still have an expected value of 1. So to show this, we will need to rely on additional properties of  $C$ .

The property we will use is that  $C$  is the sum of the  $n^4$  identically distributed indicator random variables  $C_{\bar{\ell}}$ . If this set of random variables had been independent, we could have used a Chernoff-Hoeffding bound to show that  $C$  strongly concentrates around its expectation, and thus if its expectation is sufficiently larger than 0, then it will take non-zero values with large probability. However, these variables are not independent.

Consider, for example, tuples of indices  $\bar{\ell} = (\ell_1, \dots, \ell_4)$  and  $\bar{\ell}' = (\ell'_1, \dots, \ell'_4)$  that differ only on the value of their fourth element. If  $C_{\bar{\ell}} = 1$ , this implies that  $(L_1[\ell'_1] + L_2[\ell'_2]) = (L_1[\ell_1] + L_2[\ell_2]) \in \langle mp \rangle$ . Thus, the tuple  $(L_1[\ell'_1], \dots, L_4[\ell'_4])$  already passes one of the filters of the algorithm, and so  $C_{\bar{\ell}'}$  is now more likely to be 1 than it would have been without conditioning on  $C_{\bar{\ell}} = 1$ . In fact, this example illustrates that the set of random variables  $\{C_{\bar{\ell}}\}$  are not even pairwise independent. This lack of independence is perhaps the most significant challenge in analyzing the performance of the  $k$ -Tree algorithm.

Our way around this is the following set of observations. The degree of correlation between  $C_{\bar{\ell}}$  and  $C_{\bar{\ell}'}$  is proportional to the number of co-ordinates on which  $\bar{\ell}$  and  $\bar{\ell}'$  agree. For instance, if  $\bar{\ell}$  and  $\bar{\ell}'$  do not agree on any co-ordinate, then  $C_{\bar{\ell}}$  and  $C_{\bar{\ell}'}$  are, in fact, independent. Fortunately, for  $t \in [0, 4]$ , the number of pairs  $\bar{\ell}$  and  $\bar{\ell}'$  that agree on  $t$  co-ordinates decreases as  $t$  (and thus the correlation) increases. Out of the  $n^8$  possible pairs, only  $O(n^7)$  pairs agree on 1 co-ordinate,  $O(n^6)$  agree on 2, and  $O(n^5)$  agree on 3.

Taking advantage of this, we are able to carefully bound the second moment of  $C$ . Then, we use second-moment-based concentration bounds to show that if the expectation of  $C$  is large enough, it will indeed be non-zero with substantial probability.

**Computing the Second Moment.** The second moment of  $C$  can be written as follows:

$$\mathbf{E}[C^2] = \sum_{\bar{\ell}, \bar{\ell}' \in [n]^4} \mathbf{E}[C_{\bar{\ell}} \cdot C_{\bar{\ell}'}] = \sum_{\bar{\ell}, \bar{\ell}' \in [n]^4} \Pr[C_{\bar{\ell}} = 1 \wedge C_{\bar{\ell}'} = 1] \quad (14)$$

Due to symmetry, the term corresponding to any  $\bar{\ell}$  and  $\bar{\ell}'$  in the sum above is fully determined by the set of co-ordinates that  $\bar{\ell}$  and  $\bar{\ell}'$  agree on (that is, it does not matter what specific values the  $\ell_i$ 's and  $\ell'_i$ 's take once the set of  $i$ 's on which they agree is determined). To capture this, we define the string  $\delta(\bar{\ell}, \bar{\ell}') \in \{0, 1\}^4$  whose  $i^{\text{th}}$  co-ordinate is defined as follows:

$$(\delta(\bar{\ell}, \bar{\ell}'))_i = \begin{cases} 0 & \text{if } \ell_i = \ell'_i \\ 1 & \text{if } \ell_i \neq \ell'_i \end{cases}$$

We then segregate the pairs  $(\bar{\ell}, \bar{\ell}')$  in the sum in (14) according to the value of  $\delta(\bar{\ell}, \bar{\ell}')$ . For any  $s \in \{0, 1\}^4$ , the the number of such pairs with  $\delta(\bar{\ell}, \bar{\ell}') = s$  is  $n^4(n-1)^{wt(s)} \approx n^{4+wt(s)}$ , where  $wt(s)$  is the Hamming weight of  $s$ . We can then re-write the second moment as follows:

$$\begin{aligned} \mathbf{E}[C^2] &= \sum_{s \in \{0,1\}^4} \sum_{\substack{\bar{\ell}, \bar{\ell}' \in [n]^4 \\ \text{s.t. } \delta(\bar{\ell}, \bar{\ell}') = s}} \Pr[C_{\bar{\ell}} = 1 \wedge C_{\bar{\ell}'} = 1] \\ &\approx \sum_{s \in \{0,1\}^4} n^{4+wt(s)} \cdot f(s) \end{aligned} \quad (15)$$

where  $f(s)$  is defined to be the probability that  $C_{\bar{\ell}} = C_{\bar{\ell}'} = 1$  for any  $\bar{\ell}$  and  $\bar{\ell}'$  such that  $\delta(\bar{\ell}, \bar{\ell}') = s$ . In computing  $f(s)$ , our broad approach is similar to the one we took when computing the first moment – to repeatedly replace intermediate list elements with uniformly random values while computing the probabilities that the filters of the algorithm are passed. Only now, we need to do this simultaneously for two partially dependent executions of the algorithm.

We start first with the events  $E_1$  as defined above, which capture the first filter applied to the first two elements in the lists. The probability that this event happens in both executions is:

$$\Pr_{L_1, L_2} [(L_1[\ell_1] + L_2[\ell_2]) \in \langle mp \rangle \wedge (L_1[\ell'_1] + L_2[\ell'_2]) \in \langle mp \rangle] \quad (16)$$

Let  $s = \delta(\bar{\ell}, \bar{\ell}')$ , and denote its bits by  $s_1, \dots, s_4$ . If  $s_1 = s_2 = 0$ , then  $\ell_1 = \ell'_1$  and  $\ell_2 = \ell'_2$ , meaning the two events above are the same and so by (9) the above probability is  $\approx p$  (ignoring factors of  $(1 \pm O(p))$  for now). Similarly if  $s_1 = s_2 = 1$ , the two events are independent and the probability is  $\approx p^2$ . If exactly one of  $s_1$  and  $s_2$  is 1, then again the probability is roughly  $\leq p^2$ , due the following fact, which again follows from elementary computations:

$$\Pr_{w,x,y \leftarrow \langle s \rangle} [w+x \in \langle sp \rangle \wedge w+y \in \langle sp \rangle] \leq p^2 \cdot \left(1 + O\left(\frac{1}{sp}\right)\right)^2 \quad (17)$$

Overall, we have approximately the following (again ignoring  $(1 \pm O(p))$  factors):

$$\Pr_{L_1, L_2} [E_1(L_1[\ell_1], L_2[\ell_2]) \wedge E_1(L_1[\ell'_1], L_2[\ell'_2])] \leq p^{1+\max(s_0, s_1)} = p^{1+s_1^1} \quad (18)$$

where we define  $s_1^1$  to be  $\max(s_1, s_2)$ .

Next, we look at the joint distribution of the sums  $(L_1[\ell_1] + L_2[\ell_2], L_1[\ell'_1] + L_2[\ell'_2])$ . If  $s_1^1 = 0$ , then these two sums are equal, and by (6) are each  $(1 + O(p))$ -close to being uniformly distributed over  $\langle sp \rangle$ . If  $s_1 = s_2 = 1$ , then these are independent and so the joint distribution is now close to

being uniform over  $\langle mp \rangle \times \langle mp \rangle$ . We show by computing the MR distances explicitly that even in the case where exactly one of  $s_1$  and  $s_2$  is 1, this joint distribution is close to being uniform over  $\langle mp \rangle \times \langle mp \rangle$ .

So overall, if  $s_1^1 = 0$ , this pair of sums is equal with close-to-uniform marginals, and if  $s_1^1 = 1$ , the pair is close to being independently uniformly distributed over  $\langle mp \rangle$ . Notice that this is exactly the relationship that the pair  $(L_1[\ell_1], L_1[\ell'_1])$  had to  $s_1$ , or  $(L_2[\ell_2], L_2[\ell'_2])$  had to  $s_2$ , except that the marginal distributions there were over  $\langle m \rangle$ . The same sequence of arguments can be made with the events  $E_2$ , where in place of  $s_1^1$  we use  $s_2^1 = \max(s_3, s_4)$ . In effect, this lets us set up a recursive argument where we can replace the intermediate sums with uniform elements from  $\langle mp \rangle$ , and the string  $s$  with  $(s_1^1, s_2^1)$ .

We can then repeat the entire argument above once more (essentially with  $k = 2$  now), with  $s_1^2 = \max(s_1^1, s_2^1)$ , to show that the probability of each of the pairs of events corresponding to  $E_3$  and  $E_4$  happening is roughly bounded by  $p^{1+s_1^2}$ . Overall, we end up with the following bound:

$$\begin{aligned} \Pr [C_{\bar{\ell}} = 1 \wedge C_{\bar{\ell}'} = 1] &= f(s) \leq p^{1+s_1^1} \cdot p^{1+s_2^1} \cdot p^{1+s_1^2} \cdot p^{1+s_2^2} \cdot (1 + O(p)) \\ &= p^{4+s_1^1+s_2^1+2s_1^2} \cdot (1 + O(p)) \end{aligned} \quad (19)$$

Combining this bound with (15), we get:

$$\begin{aligned} \mathbf{E} [C^2] &\leq \sum_{s \in \{0,1\}^4} n^{4+wt(s)} \cdot p^{4+s_1^1+s_2^1+2s_1^2} \cdot (1 + O(p)) \\ &= c^4 \cdot (1 + O(p)) \cdot \sum_{s \in \{0,1\}^4} c^{wt(s)} \cdot p^{s_1^1+s_2^1+2s_1^2-wt(s)} \end{aligned} \quad (20)$$

Using graph-theoretic arguments, we can show that for any  $s \notin \{0^4, 1^4\}$ , the difference  $(s_1^1 + s_2^1 + 2s_1^2 - wt(s))$  is at least 1, and so  $p^{s_1^1+s_2^1+2s_1^2-wt(s)} \leq p$ . For  $s \in \{0^4, 1^4\}$ , this difference is 0. So we can bound the above sum as follows:

$$\begin{aligned} \sum_{s \in \{0,1\}^4} c^{wt(s)} \cdot p^{s_1^1+s_2^1+2s_1^2-wt(s)} &\leq 1 + c^4 + \sum_{s \in \{0,1\}^4 \setminus \{0^4, 1^4\}} c^{wt(s)} \cdot p \\ &\leq 1 + c^4 + p \cdot \sum_{s \in \{0,1\}^4} c^{wt(s)} \\ &= 1 + c^4 + p \cdot (1 + c)^4 \end{aligned} \quad (21)$$

Overall, from (20) and (21), we get the following bound on the second moment:

$$\mathbf{E} [C^2] \leq c^4 \cdot (1 + c^4 + p \cdot (1 + c)^4) \cdot (1 + O(p)) \quad (22)$$

**Computing the Lower Bound.** With the bounds on the first two moments from (13) and (22), the required lower bound can be obtained using the Paley-Zygmund inequality, which states that for any non-negative random variable  $Z$  and any  $\theta \in [0, 1]$ :

$$\Pr [Z > \theta \mathbf{E} [Z]] \geq (1 - \theta)^2 \frac{\mathbf{E} [Z]^2}{\mathbf{E} [Z^2]}$$

Applying this bound to  $C$  with  $\theta = 0$ , we get:

$$\begin{aligned} \Pr [k\text{-Tree succeeds}] &= \Pr [C > 0] \geq \frac{\mathbf{E}[C]^2}{\mathbf{E}[C^2]} \\ &\geq \frac{c^8 \cdot (1 - O(p))}{c^4(1 + c^4 + p(1 + c)^4)(1 + O(p))} \\ &\geq \frac{c^4}{1 + c^4 + p \cdot (1 + c)^4} \cdot (1 - O(p)) \end{aligned} \tag{23}$$

**Tightening the Bound.** The above approach naturally generalizes to any  $k$  that is a power of 2 to give the following lower bound:

$$\Pr [k\text{-Tree succeeds}] \geq \frac{c^k}{1 + c^k + p \cdot (1 + c)^k} \cdot (1 - O(kp))$$

While their asymptotics for any fixed  $k$  are the same, for moderately large values of  $k$  and some reasonable concrete values of  $m$  and  $n$ , the above bound ends up being much weaker than the lower bound actually stated in Theorem 1. This is because, even when  $c = 1$ , the  $p(1 + c)^k$  term in the denominator quickly starts to dominate. The bound stated in Theorem 1 remains meaningful for a wider range of concrete values of the parameters, and is obtained by performing a more careful analysis of the sum in (20), using a recursive argument to bound the entire sum together rather than each term separately.

The constants in the statement of Theorem 1 are obtained by carefully tracking the constants that come up in the course of the above analysis. Even so, the constants stated there are sub-optimal due to limits on human tolerance, and the algorithmic implementation of this proof (as captured in Theorem 2) improves upon them significantly.

### 1.3 Related Work

The worst-case version of the  $k$ -SUM problem has been studied extensively in the literature on algorithms [HS74, BDP08, DSW18, Cha20, ...], data structures [KP19, GGH<sup>+</sup>20, CL23, ...], and complexity theory [GO95, BHP01, SEO03, CGI<sup>+</sup>16, GS17, Eri95, AC05, Pat10, PW10, GP18, ABHS19, ...]. In the study of the fine-grained complexity of problems within  $\mathsf{P}$ , in particular, connections have been discovered between the complexity of this problem and those of various important problems from computational geometry, graph theory, data structures, etc. [GO95, BHP01, SEO03, Pat10, AW14, KPP16]. A simple meet-in-the-middle algorithm solves this problem in  $\tilde{O}(n^{\lceil k/2 \rceil})$  time [HS74]. The best algorithms known are faster than this by only a small polylog factor [BDP08, GP18, Cha20], or run in time  $\tilde{O}(m + n)$  [Bri17, JW19]. It has been conjectured that the best worst-case algorithm for this problem runs in time  $n^{\lceil k/2 \rceil - o(1)}$  [GO95, AL13].

For average-case  $k$ -SUM, non-trivial algorithms (beyond known worst-case algorithms) have so far been restricted to Wagner’s  $k$ -Tree algorithm [Wag02] and its extensions to using smaller lists (at the cost of running time) [MS12], to values of  $k$  that are not power of 2 [NS15, Din19], to generalizations with better time-memory tradeoffs [NS15, BK17, Din19], and to the quantum setting [GNS18]. These algorithms have found extensive use across cryptography and cryptanalysis [Sch01, Wag02, DEF<sup>+</sup>19, LS19, BLL<sup>+</sup>22, ...]. They are also closely related to some of the best algorithms for the Learning Parity with Noise problem [BKW03].

Lattice-based conditional lower bounds are known that indicate that the  $k$ -Tree algorithm for  $k$ -SUM is optimal in its asymptotic dependence on  $k$  [BSV21]. Some conditional lower bounds are also known for the complexity of the  $k$ -SUM problem given lists of size close to  $m^{1/k}$  [DKK21, ASS<sup>+</sup>24].

**Comparison with Existing Analyses.** As noted earlier, Wagner’s original argument [Wag02] of the correctness of the  $k$ -Tree algorithm was heuristic and relied on assumptions regarding the distribution and independence of elements in intermediate lists computed by the algorithm. The conclusion of this argument was that the algorithm works with constant success probability when given input lists of size  $m^{1/(\log k+1)}$ , and that in this case it runs in time  $O(k \cdot m^{1/(\log k+1)})$ .

This argument was made rigorous by Minder and Sinclair [MS12] for the variant of the  $k$ -Tree algorithm that solves the  $k$ -SUM problem over the group  $\mathbb{Z}_2^m$  (in which case it is referred to as the  $k$ -XOR problem). In this case, Wagner’s assumption regarding the uniform distribution of intermediate list elements is immediately seen to be true. Minder and Sinclair also follow the approach (described in Section 1.2) of then computing the first two moments of the random variable counting the number of solutions found, and then using these to prove concentration bounds. In the case of  $k$ -XOR, the process of computing these moments turns out to be much simpler due to the fact that the XOR of any arbitrary vector with a uniformly random vector results again in a uniformly random vector. Some of the questions that come up during this analysis of  $k$ -Tree over  $\mathbb{Z}_2^m$  also come up in the analysis of an approach to hashing called Simple Tabulation Hashing (see, for example, [PT12]), but the implications of the results there for Minder and Sinclair’s analysis are not immediately clear to us.

The first rigorous analysis of the  $k$ -Tree algorithm over integers<sup>5</sup> was by Lyubashevsky [Lyu05], who showed that the algorithm works with high probability if the input lists are of size at least  $\Omega(k^2 \log k \cdot m^{2/\log k})$ . This was improved by Shallue [Sha08] to lists of size at least  $\Omega(k \cdot m^{1/\log k})$ . Both of these actually study a variant of the  $k$ -Tree algorithm where the factor by which the permitted range shrinks at each level of the tree is set to  $p = m^{-1/\log k}$  instead of  $m^{-1/(\log k+1)}$ . In this case, the final list is only permitted to contain 0’s, and when the input list size is  $m^{1/\log k}$ , the expected number of solutions found by the algorithm is also  $\Theta(m^{1/\log k})$ . Both Lyubashevsky and Shallue were interested in using the  $k$ -Tree algorithm for very large values of  $k$ , and so the difference between  $\log k$  and  $(\log k + 1)$  in the exponent were not significant in their applications. It is possible that Shallue’s analysis can be made to work for the original  $k$ -Tree algorithm, but this is to be verified and even then it is not clear whether the bounds will work for list sizes smaller than  $k \cdot m^{1/(\log k+1)}$ .

Both of the above papers take similar approaches in their analysis, and we briefly describe Shallue’s here. Similar to us, Shallue starts by showing that the elements in the intermediate lists are close to uniform; the distance measure used there is also an element-wise bound on differences in probability mass, but the whereas MR distance measures relative difference, they measure absolute difference. We believe our choice is better for concrete parameters, but asymptotically these choices are likely equivalent. The larger difference is in how they deal with the dependence between list elements. They start with the observation that even though there are dependencies between list elements, it is possible to find a large enough subset of elements in each list such that all correlations among them are relatively small. They then use martingale-based concentration bounds

---

<sup>5</sup>We continue to ignore the distinction between  $k$ -Tree over integers and over  $\mathbb{Z}_m$ , following the discussion in Remark 1.4.



to recursively bound the size of each intermediate list.

Their utilization of correlation bounds on larger sets of variables eventually results in a better dependence of the probability bound on the list-size overhead. They show that if the input lists are of size  $c \cdot m^{1/\log k}$  for some  $c \geq k$ , then the probability of failure is roughly at most  $(m^{1/\log k} e^{-c/1024})$ , whereas our bound for lists of size  $c \cdot m^{1/(\log k+1)}$  is around  $c^{-k}$ . For very large values of  $c$ , the former will be much smaller, but for moderate values  $c = \Theta(1)$ , the latter bound is better.

A different approach to analyzing the  $k$ -Tree algorithm was taken by Joux, Kippen, and Loss [JKL24]. They get around the uniformity and independence issues by modifying the  $k$ -Tree algorithm in such a way that these properties actually hold as assumed by the heuristic analysis. This makes the algorithm slightly worse than the original, but easier to analyze. They then show that this modified algorithm works with probability roughly larger than  $(1 - e^{-\text{poly}(k)})$  if the input list sizes are  $n = \beta(k) \cdot m^{1/(\log k+1)}$ , where  $\beta(k) \approx (1.26)^{\frac{\ell-1}{\ell+1}} \cdot 6^{\frac{(\ell-1)(\ell+2)}{(\ell+1)}}$ , with  $\ell = \log k$ . Further, their modified algorithm runs in time  $\Theta(kn)$  in this case. This again results in very good bounds on the probability of success, but the bounds only work for large values of  $c$ , especially for larger values of  $k$  – for instance,  $\beta(4) \approx 3.563$  and  $\beta(1024) \approx 7964$ .

In summary, the following are the most significant points of comparison between our results and the existing analyses of Shallue [Sha08] and Joux et al. [JKL24] of the  $k$ -Tree algorithm over integers:

- The results in the other papers do not give meaningful bounds for  $c \leq 1$  – that is, when the size of the lists is  $n = m^{1/(\log k+1)}$ , as suggested by Wagner’s heuristic analysis, or smaller – while ours do. This is also true for some range of values of  $c$  larger than 1, with the range depending on  $k$ .
- For sufficiently larger list sizes ( $c \gg 1$ ), the approaches in the other papers can likely show lower bounds on the success probability that are much closer to 1 than our bounds can. This gap can be closed using certain optimizations to the  $k$ -Tree algorithm as described in Remark 1.2, though that does not vindicate our analysis itself.

We present comparisons of our results with these in Figure 4, making optimistic assumptions regarding the constants in the bounds of the other papers, as well as regarding the regime of their validity, wherever relevant.

## 2 Analysis

In this section, we present our analysis of Wagner’s  $k$ -Tree algorithm for the  $k$ -SUM problem over integers [Wag02]. We present the algorithm in full detail in Figure 5. It has four parameters – the number of lists ( $k$ ), the size of each list ( $n$ ), the range the numbers are drawn in ( $m$ ), and an internal filtering parameter  $p$ . We then state our main claim about its success probability in Theorem 2.1, state its significant corollaries, and then prove the theorem in the rest of the section.

**Notation.** For any  $m \in \mathbb{N}$ , we denote by  $[m]$  the set of natural numbers  $\{1, 2, \dots, m\}$ , and by  $\langle m \rangle$  the set of integers  $\{-\lfloor \frac{m}{2} \rfloor, \dots, \lfloor \frac{m}{2} \rfloor\}$ . Below, the term “list” simply refers to an array indexed by natural numbers in some range  $[n]$ . A list might contain duplicate entries, and in particular is to be distinguished from a set.

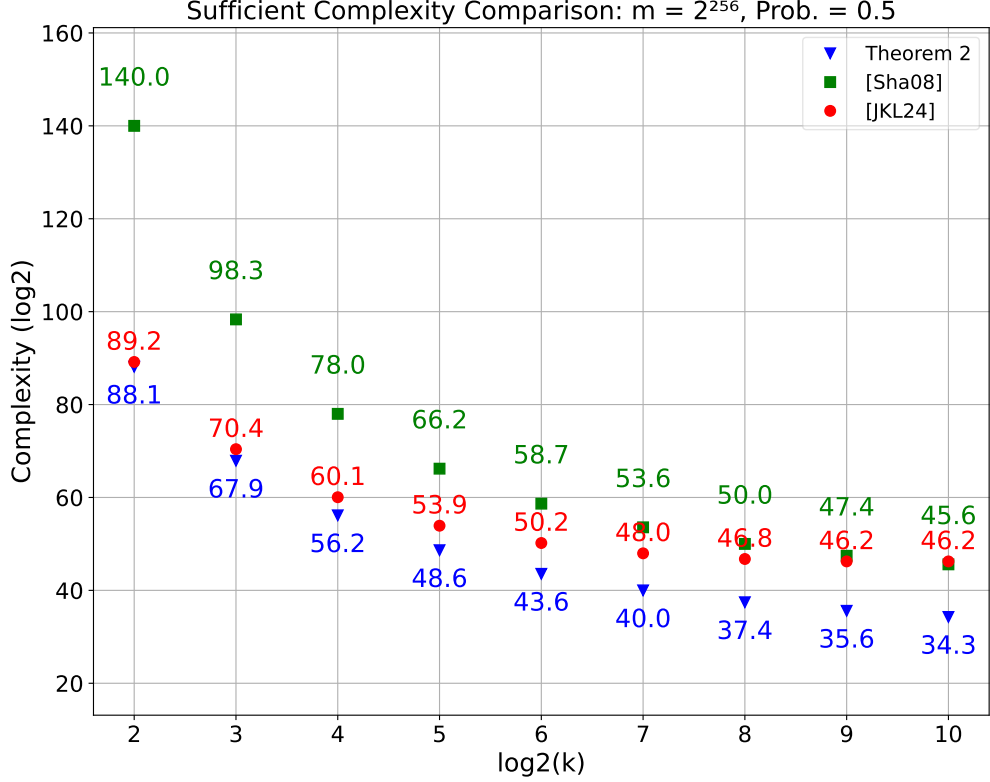


Figure 4: Comparison of bounds produced by Theorem 2 with those of [Sha08] and [JKL24]. For the other papers, we plot the minimum complexity of  $k$ -Tree for which they show non-trivial lower bounds on success probability (usually close to 1). In our case, we plot the smallest complexity of  $k$ -Tree for which the lower bound on success probability produced by Theorem 2 is at least 0.5.

**Theorem 2.1.** Consider any  $k, n, m \in \mathbb{N}$ , where  $k \geq 4$  is a power of 2 and  $m > 30^{\log k + 1}$ . Set  $p = m^{\frac{-1}{\log k + 1}}$  and  $c = p \cdot n$ . Consider  $k$  lists  $L_1, \dots, L_k$ , each consisting of  $n$  uniformly random integers from the range  $\langle m \rangle$ . The  $k$ -Tree algorithm (as in Figure 5) with these parameters, denoted by  $\text{kTree}$ , satisfies the following:

- **Success Probability.** Its probability of success is bounded as follows:

$$\frac{1}{c^{-k} + \left(1 + \frac{k}{n}\right)^k} \cdot (1 - 150p)^k \leq \Pr_{L_1, \dots, L_k} \left[ \begin{array}{l} \text{kTree}(L_1, \dots, L_k) \\ \text{outputs } (\ell_1, \dots, \ell_k) \\ \text{such that } \sum_i L_i[\ell_i] = 0 \end{array} \right] \leq c^k \cdot (1 + 37p)^k$$

- **Complexity.** Its expected complexity is bounded as follows:

$$\mathbf{E}_{L_1, \dots, L_k} \left[ \begin{array}{l} \text{Total size of all lists} \\ \text{involved in} \\ \text{kTree}(L_1, \dots, L_k) \end{array} \right] \in kn \cdot \left( 1 + \sum_{d \in [\log k]} \frac{c^{2^d - 1}}{2^d} \right) \cdot (1 \pm 37p)^{k-1}$$

**Corollary 2.2.** Consider functions  $k, n : \mathbb{N} \rightarrow \mathbb{N}$ , where for any  $m$ ,  $k(m) \geq 4$  is a power of 2. Set  $p(m) = m^{\frac{-1}{\log k(m) + 1}}$  and  $c(m) = p(m) \cdot n(m)$ . Further, suppose  $k = o(1/p)$  and  $k = o(n^{1/2})$ .

### The $k$ -Tree algorithm

**Parameters:**  $k, n, m \in \mathbb{N}$ ,  $p \in (0, 1]$ , with  $k$  being a power of 2

**Input:** Lists  $L_1, \dots, L_k$ , each consisting of  $n$  integers from  $\langle m \rangle$

**Output:** Indices  $\ell_1, \dots, \ell_k \in [n]$ , or a failure symbol  $\perp$

**Subroutines:**

- $\text{Merge}(L_a, L_b, \tau)$ : On input lists  $L_a$  and  $L_b$  of integers and threshold  $\tau \in \mathbb{R}$ , outputs a list consisting of all  $(a + b)$  such that  $a \in L_a$ ,  $b \in L_b$ , and  $|a + b| \leq \tau$ . This output maintains duplicates – if multiples pairs  $(a \in L_a, b \in L_b)$  have the same sum, a copy of that sum is included in the output list for each pair.

**Procedure:**

1. For each  $i \in [k]$ , denote the list  $L_i$  by  $L_i^0$
2. Set  $\tau \leftarrow m/2$
3. For  $d$  from 1 to  $\log k$ :
  - Set  $\tau \leftarrow p \cdot \tau$
  - For  $i \in [\frac{k}{2^d}]$ : set  $L_i^d \leftarrow \text{Merge}(L_{2i-1}^{d-1}, L_{2i}^{d-1}, \tau)$
4. If  $L_1^{\log k}$  contains 0, output the indices in the input lists that led to this sum. Otherwise output  $\perp$ .

**Remarks:**

- In order to perform its last step, the algorithm additionally needs to keep track of which elements of the input lists contribute to each sum in the intermediate lists. We leave this bookkeeping out of our description for simplicity.
- Note that if any list considered in the course of the algorithm is empty, the algorithm is eventually bound to output  $\perp$ .
- Above, we only describe the behavior of the Merge algorithm rather than specifying the procedure it follows because it can be implemented in various ways, with the most efficient choice depending on the parameters of the problem and the execution environment. Also see Remark 1.3.

Figure 5: The  $k$ -Tree algorithm over Integers

With  $k = k(m)$ ,  $n = n(m)$ ,  $m$ , and  $p(m)$  as its parameters, the probability of success of the  $k$ -Tree algorithm, denoted  $\text{kTree}$ , is bounded as follows:

$$\frac{c^k}{1 + c^k} \cdot (1 - o(1)) \leq \Pr[\text{kTree succeeds}] \leq c^k \cdot (1 + o(1))$$

Its complexity is bounded as follows:

$$\text{Complexity of kTree} \in kn \cdot \left( 1 + \sum_{d \in [\log k]} \frac{c^{2^d - 1}}{2^d} \right) \cdot (1 \pm o(1))$$

The rest of this section is a proof of Theorem 2.1. Our approach is non-trivial, but elementary. We simply compute the first two moments of the random variable that counts the number of occurrences of 0 in the final list produced by the algorithm, and then use standard tail bounds to bound the probability that this variable is non-zero.

Hereon, we adopt the context of the  $k$ -Tree algorithm from Figure 5, and use notation established in its description in the general text as well. For any set of parameters  $k, n, m, p$ , we define the aforementioned random variable as follows (with randomness coming from the choices of the  $L_i$ 's):

$$C_{k,n,m,p} = \text{number of occurrences of 0 in the list } L_1^{\log k}$$

We show the following properties of the lower moments of this random variable, use these to prove Theorem 2.1, and then later in the section prove the propositions themselves.

**Proposition 2.3.** *Consider any valid set of parameters  $k, n, m$ , and  $p$  such that  $m > 7k$ ,  $p \cdot k > (7/m)^{(k/2-1)}$ ,  $mp^{\log k} > 30$ , and  $p < 1/30$ . We have:*

$$\mathbf{E}[C_{k,n,m,p}] \in \frac{n^k p^k}{mp^{\log k + 1}} \cdot (1 \pm p)^{k-1} \left( 1 \pm \frac{35}{mp^{\log k}} \right)^{k-1}$$

The second moment of this variable is most conveniently bounded in terms of a function that is again recursively defined, and is a refinement of  $T_{\mu,n}$  defined above. For any  $\mu, \nu \in (0, 1)$  and  $n \in \mathbb{N}$ , the function  $T_{\mu,n,\nu} : \mathbb{N} \rightarrow \mathbb{R}$  is defined on inputs that are powers of 2. The base is its value on 1:

$$T_{\mu,n,\nu}(1) = n \cdot \mu \cdot \nu$$

For any  $k \geq 2$  that is a power of 2, it is defined as follows:

$$T_{\mu,n,\nu}(k) = T_{\mu,n,\nu}(k/2) \left( \frac{T_{\mu,n,\nu}(k/2)}{\nu} + 2\mu \right)$$

As the above is a quadratic recursion, there is no general closed-form expression for  $T_{\mu,n,\nu}(k)$ . We prove upper bounds on its value later in the course of the proof of Theorem 2.1.

**Proposition 2.4.** *Consider any valid set of parameters  $k, n, m$ , and  $p$  such that  $mp^{\log k - 1} > 30$  and  $p < 1/30$ . Define the following:*

$$\mu = p \cdot \left[ (1 - p)^{-1} \left( 1 + \frac{35}{mp^{\log k}} \right) \right] \quad \text{and} \quad \nu = \frac{1}{mp^{\log k + 1}}$$

Then, we have:

$$\mathbf{E}[C_{k,n,m,p}^2] \leq (n^k \mu^k \nu) \cdot (1 + T_{\mu,n,\nu}(k))$$

In order to discuss the complexity of the algorithm, for any set of parameters  $k, n, m, p$ , we define the following random variable that captures the total size of all the lists in an execution of the algorithm:

$$\Lambda_{k,n,m,p} = \sum_{d=0}^{\log k} |L_i^d|$$

**Proposition 2.5.** *Consider any valid set of parameters  $k, n, m$ , and  $p$  such that  $m > 7k$ ,  $p \cdot k > (7/m)^{(k/2-1)}$ ,  $mp^{\log k} > 30$ , and  $p < 1/30$ . We have:*

$$\mathbf{E}[\Lambda_{k,n,m,p}] \in kn \cdot \left(1 + \sum_{d \in [\log k]} \frac{(np)^{2^d-1}}{2^d}\right) \cdot (1 \pm p)^{k-1} \left(1 \pm \frac{35}{mp^{\log k}}\right)^{k-1}$$

With these propositions, we can now prove our main theorem.

*Proof of Theorem 2.1.* The lower bound on the success probability in Theorem 2.1 follows from applying a suitable second-moment-based tail-bound to the random variable  $C_{k,n,m,p}$ , and then appropriately bounding the function  $T_{\mu,n,\nu}$  defined above. The lower bound follows from a simple Markov bound, and the bounds on the complexity follow from Proposition 2.5.

**Lower bound.** For any  $k, n, m$ , and  $p$ , each entry in the list  $L_1^{\log k}$  is a sum of one entry each from the input lists  $L_1, \dots, L_k$ . So whenever the non-negative random variable  $C_{k,n,m,p}$  (denoted simply by  $C$  hereon) is non-zero, there is at least one entry in  $L_1^{\log k}$  that is 0, and the execution  $\text{kTree}_p(L_1, \dots, L_k)$  finds indices  $(\ell_1, \dots, \ell_k)$  such that  $\sum_i L_i[\ell_i] = 0$ . So all we need to do is to bound the probability that this random variable is non-zero. This we do using the Paley-Zygmund inequality.

**Lemma 2.6** (Paley-Zygmund Inequality, see e.g. [Roc24, Section 2.3]). *For any non-negative random variable  $Z$  and any  $\theta \in [0, 1]$ ,*

$$\Pr[Z > \theta \mathbf{E}[Z]] \geq (1 - \theta)^2 \frac{\mathbf{E}[Z]^2}{\mathbf{E}[Z^2]}$$

Substituting the values  $p = m^{\frac{-1}{\log k+1}}$  and  $n = c \cdot m^{\frac{1}{\log k+1}}$ , and observing that the conditions  $k \geq 4$  and  $m > 30^{\log k+1}$  guarantee the requirements in its hypotheses, we get the following bound from Proposition 2.3:

$$\mathbf{E}[C] \in c^k \cdot (1 \pm p)^k (1 \pm 35p)^k \tag{24}$$

For brevity, denote the function  $T_{\mu,n,1}$  that was defined above by  $T_{\mu,n}$ . We get the following from Proposition 2.4:

$$\mathbf{E}[C^2] \leq c^k \cdot (1 + T_{\mu,n}(k)) \cdot [(1 - p)^{-1} (1 + 35p)]^k \tag{25}$$

where  $\mu = p \cdot [(1 - p)^{-1} (1 + 35p)]$ . We bound this more concretely using the claim below, which we prove after the completion of the current proof.

**Claim 2.7.** For any  $\mu \in (0, 1)$  and  $n, k \in \mathbb{N}$ , we have:

$$T_{\mu, n}(k) \leq n^k \mu^k \cdot \left(1 + \frac{k}{n}\right)^k$$

Putting together (25) and Claim 2.7, we have:

$$\begin{aligned} \mathbf{E}[C^2] &\leq c^k \cdot \left(1 + c^k \cdot [(1-p)^{-1}(1+35p)]^k \cdot \left(1 + \frac{k}{n}\right)^k\right) \cdot [(1-p)^{-1}(1+35p)]^k \\ &\leq c^{2k} \left(c^{-k} + \left(1 + \frac{k}{n}\right)^k\right) [(1-p)^{-1}(1+35p)]^{2k} \end{aligned} \quad (26)$$

Applying Lemma 2.6 with  $\theta = 0$  and the bounds from (24) and (26), we get:

$$\begin{aligned} \Pr[C > 0] &\geq \frac{\mathbf{E}[C]^2}{\mathbf{E}[C^2]} \\ &\geq \frac{c^{2k} \cdot (1-p)^{2k} (1-35p)^{2k}}{c^{2k} \left(c^{-k} + \left(1 + \frac{k}{n}\right)^k\right) [(1-p)^{-1}(1+35p)]^{2k}} \\ &\geq \frac{1}{c^{-k} + \left(1 + \frac{k}{n}\right)^k} \cdot (1-p)^{4k} (1-35p)^{2k} (1+35p)^{-2k} \\ &\geq \frac{1}{c^{-k} + \left(1 + \frac{k}{n}\right)^k} \cdot (1-150p)^k \end{aligned}$$

as required.

**Upper bound.** To get the upper bound on the success probability, we apply the Markov bound using (24) as follows:

$$\Pr[C \geq 1] \leq \mathbf{E}[C] \leq c^k \cdot (1+37p)^k$$

**Complexity.** The bounds on the complexity of the algorithm follow directly from Proposition 2.5. By our setting of  $p$ , we have  $mp^{\log k+1} = 1$ , and using the fact that  $p < 1/30$ , we get:

$$\begin{aligned} \mathbf{E}[\Lambda_{k, n, m, p}] &\in kn \cdot \left(1 + \sum_{d \in [\log k]} \frac{(np)^{2^d-1}}{2^d}\right) \cdot (1 \pm p)^{k-1} (1 \pm 35p)^{k-1} \\ &\subseteq kn \cdot \left(1 + \sum_{d \in [\log k]} \frac{c^{2^d-1}}{2^d}\right) \cdot (1 \pm 37p)^{k-1} \end{aligned}$$

This proves the theorem. □

*Proof of Claim 2.7.* Fix some values of  $n$  and  $\mu$ , and denote  $T_{\mu,n}$  simply by  $T$ . Recall that  $T$  is defined as follows:

$$\begin{aligned} T(1) &= n \cdot \mu \\ T(k) &= T(k/2)(T(k/2) + 2\mu) \end{aligned}$$

The claim is that for every  $k$  (that is a power of 2),  $T(k) \leq (n\mu + k\mu)^k$ . This can be verified to be true for  $k = 1$  and 2. For some  $k \geq 4$ , suppose this is true for  $k/2$ . Then, we have:

$$\begin{aligned} T(k) &= T(k/2)(T(k/2) + 2\mu) \\ &\leq (n\mu + (k/2)\mu)^{k/2}(n\mu + (k/2)\mu + 2\mu)^{k/2} \\ &\leq (n\mu + k\mu)^{k/2}(n\mu + k\mu)^{k/2} \\ &= (n\mu + k\mu)^k \end{aligned}$$

which proves the claim. □

**Section Outline.** In Section 2.1, we define a useful notion of distance between probability distributions and establish some facts about distributions over bounded integer intervals. In Section 2.2, we compute the first moment of the above random variable to prove Proposition 2.3, and in Section 2.3 we compute its second moment to prove Proposition 2.4.

## 2.1 Tools

Here, we set up some notational conventions, definitions, and propositions that will be useful at multiple points in the proofs of Propositions 2.3 and 2.4.

**Distance between distributions.** For any distribution  $D$  over a domain  $\mathcal{X}$  and any  $x \in \mathcal{X}$  (or  $S \subseteq \mathcal{X}$ ), we denote by  $D(x)$  (resp.  $D(S)$ ) the probability mass placed by  $D$  on  $x$  (resp.  $S$ ). Given distributions  $D$  and  $\hat{D}$ , we denote by  $(D \otimes \hat{D})$  the (“direct product”) distribution of  $(x, y)$  where  $x \leftarrow D$  and  $y \leftarrow \hat{D}$  are sampled independently.

We will use the following notion of distance between probability distributions. Related notions of distance have found significant use in differential privacy [DMNS16], analysis of lattice algorithms [BLRL<sup>+</sup>18], etc.

**Definition 2.8** (Max-Ratio Distance). *Consider two distributions  $D_0$  and  $D_1$  over a finite domain  $\mathcal{X}$  that have the same support. The Max-Ratio (MR) distance between these, denoted by  $\Delta_{\text{MR}}(D_0, D_1)$ , is the smallest  $\lambda \geq 1$  such that for all  $x$ , we have  $D_1(x) \in [\lambda^{-1} \cdot D_0(x), \lambda \cdot D_0(x)]$ .*

Note that the above distance is symmetric, and is only defined if both distributions have the same support. All pairs of distributions we will consider in our work will indeed have the same support, and we will leave out stating this condition explicitly in the hypotheses of our statements below. The following are a few other easily observed facts about this distance that we will use.

**Fact 2.9.** *Consider any distributions  $D_0$  and  $D_1$  over  $\mathcal{X}$  with  $\Delta_{\text{MR}}(D_0, D_1) = \lambda$ . For any subset  $S \subseteq \mathcal{X}$ , we have  $D_1(S) \in [\lambda^{-1} \cdot D_0(S), \lambda \cdot D_0(S)]$ .*

**Fact 2.10.** Consider any distributions  $D_0, D_1, \hat{D}_0,$  and  $\hat{D}_1$  over appropriate domains. We have:

$$\Delta_{\text{MR}}(D_0 \otimes \hat{D}_0, D_1 \otimes \hat{D}_1) \leq \Delta_{\text{MR}}(D_0, D_1) \cdot \Delta_{\text{MR}}(\hat{D}_0, \hat{D}_1)$$

The following proposition will be especially useful for us since we will be repeatedly bounding the distance of distributions from the uniform distribution over their support.

**Proposition 2.11.** Consider any distribution  $D$ , denote its support by  $S$ , and let  $U$  be the uniform distribution over  $S$ . Then, we have:

$$\Delta_{\text{MR}}(U, D) \leq \frac{\max_{x \in S} D(x)}{\min_{x \in S} D(x)}$$

*Proof of Proposition 2.11.* Observe that for any  $x \in S$ , we necessarily have  $U(x) \geq \min_{y \in S} D(y)$  and  $U(x) \leq \max_{z \in S} D(z)$ . If either of these is not the case, the sum of the values of  $D(x)$  over all values of  $x \in S$  cannot be equal to 1. Thus, for any  $x \in S$ ,

$$D(x) = U(x) \cdot \frac{D(x)}{U(x)} \in \left[ U(x) \cdot \frac{\min_{y \in S} D(y)}{\max_{z \in S} D(z)}, U(x) \cdot \frac{\max_{z \in S} D(z)}{\min_{y \in S} D(y)} \right]$$

This proves the proposition. Note that above we rely on the fact that  $U(x)$  is non-zero for  $x \in S$ .  $\square$

**Distributions of Sums.** For any distribution  $D$  over integers (or reals), denote by  $(2 \cdot D)$  the distribution sampled by taking two independent samples  $x$  and  $y$  from  $D$  and outputting  $(x + y)$ . Denote by  $D|_{\langle s \rangle}$  the distribution  $D$  conditioned on the samples being contained in  $\langle s \rangle$ . For any  $s > 0$ , let  $U_s$  be the uniform distribution over  $\langle s \rangle$ . That is,  $U_s(x)$  is  $1/(2 \cdot \lfloor s/2 \rfloor + 1)$  if  $x \in \langle s \rangle$ , and is 0 otherwise.

We now look at what happens to the distributions of elements in the lists in a single merge step of the  $k$ -Tree algorithm. The distribution of the sum of two numbers uniform in some range is captured by the following.

**Claim 2.12.** For any  $s > 10$  and any integer  $z \in [-s, s]$ ,

$$\Pr_{x, y \leftarrow U_s} [x + y = z] \in \left( \frac{1}{s} - \frac{|z|}{s^2} \right) \pm \frac{5}{s^2}$$

*Proof of Claim 2.12.* This follows from the following calculation:

$$\begin{aligned} \Pr_{x, y \leftarrow U_s} [x + y = z] &= \sum_{x=-\lfloor s/2 \rfloor}^{\lfloor s/2 \rfloor} U_s(x) \cdot U_s(z - x) \\ &= \sum_{x=-\lfloor s/2 \rfloor + \max(0, z)}^{\lfloor s/2 \rfloor + \min(0, z)} \frac{1}{(2 \lfloor s/2 \rfloor + 1)^2} \\ &= (2 \lfloor s/2 \rfloor + 1 - |z|) \cdot \frac{1}{(2 \lfloor s/2 \rfloor + 1)^2} \\ &= \frac{1}{(2 \lfloor s/2 \rfloor + 1)} - \frac{|z|}{(2 \lfloor s/2 \rfloor + 1)^2} \end{aligned} \tag{27}$$



Using the fact that  $(2 \lfloor s/2 \rfloor + 1) \in s \pm 1$  and  $|z| \leq s$ , we can compute the “errors” in each term as follows:

$$\begin{aligned} \left| \frac{1}{s} - \frac{1}{2 \lfloor s/2 \rfloor + 1} \right| &\leq \frac{1}{s(s-1)} \\ \left| \frac{|z|}{s^2} - \frac{|z|}{(2 \lfloor s/2 \rfloor + 1)^2} \right| &\leq \frac{|z|(2s+1)}{s^2(s-1)^2} \leq \frac{3}{(s-1)^2} \end{aligned}$$

Putting these together with (27) and some simple approximations using the fact that  $s > 10$  gives the claim.  $\square$

**Proposition 2.13.** *For any  $s > 10$  and  $p \in [0, 1]$ ,*

$$\Pr_{x,y \leftarrow U_s} [x + y \in \langle sp \rangle] \in \left( p - \frac{p^2}{4} \right) \pm \frac{7}{s}$$

*Proof of Proposition 2.13.* Using Claim 2.12, we can calculate the relevant probability as follows:

$$\begin{aligned} \Pr_{x,y \leftarrow U_s} [x + y \in \langle sp \rangle] &= \sum_{z=-\lfloor sp/2 \rfloor}^{\lfloor sp/2 \rfloor} \Pr_{x,y \leftarrow U_s} [x + y = z] \\ &\in \sum_{z=-\lfloor sp/2 \rfloor}^{\lfloor sp/2 \rfloor} \frac{1}{s} - \frac{|z|}{s^2} \pm \frac{5}{s^2} \\ &= \frac{2 \lfloor sp/2 \rfloor + 1}{s} - \frac{1}{s^2} \cdot \lfloor \frac{sp}{2} \rfloor \left( \lfloor \frac{sp}{2} \rfloor + 1 \right) \pm \frac{5 \cdot (2 \lfloor sp/2 \rfloor + 1)}{s^2} \\ &\subseteq \left( p \pm \frac{1}{s} \right) - \left( \frac{p^2}{4} \pm \frac{p}{2s} \right) \pm \left( \frac{5p}{s} + \frac{5}{s^2} \right) \\ &\subseteq \left( p - \frac{p^2}{4} \right) \pm \frac{7}{s} \end{aligned}$$

which proves the proposition.  $\square$

**Proposition 2.14.** *For any  $s > 20$  and  $p \in [0, 1]$ ,*

$$\Delta_{\text{MR}}(U_{sp}, 2 \cdot U_s |_{\langle sp \rangle}) \leq \left( 1 - \frac{p}{2} \right)^{-1} \left( 1 + \frac{30}{s} \right)$$

*Proof of Proposition 2.14.* Following Proposition 2.11, as  $U_{sp}$  and  $(2 \cdot U_s |_{\langle sp \rangle})$  have the same support (that is,  $\langle sp \rangle$ ), it is sufficient to bound the ratio between the maximum and minimum probability masses of the latter to bound its distance from uniform. By Bayes’s theorem, the probability mass placed on any  $z \in \langle sp \rangle$  by this distribution is as follows:

$$(2 \cdot U_s |_{\langle sp \rangle})(z) = \Pr_{x,y \leftarrow U_s} [x + y = z \mid x + y \in \langle sp \rangle] = \frac{\Pr_{x,y \leftarrow U_s} [x + y = z]}{\Pr_{x,y \leftarrow U_s} [x + y \in \langle sp \rangle]}$$

As the normalizing factor in the denominator appears in all probability mass values, it is sufficient to bound the ratio between the maximum and minimum values of the numerator above. That is, Proposition 2.11 implies the following:

$$\Delta_{\text{MR}}(U_{sp}, 2 \cdot U_s |_{\langle sp \rangle}) \leq \frac{\max_{z \in \langle sp \rangle} \Pr_{x, y \leftarrow U_s}[x + y = z]}{\min_{z \in \langle sp \rangle} \Pr_{x, y \leftarrow U_s}[x + y = z]}$$

We can bound this ratio using Claim 2.12 as follows:

$$\begin{aligned} \frac{\max_{z \in \langle sp \rangle} \Pr_{x, y \leftarrow U_s}[x + y = z]}{\min_{z \in \langle sp \rangle} \Pr_{x, y \leftarrow U_s}[x + y = z]} &\leq \frac{1/s + 5/s^2}{1/s - p/2s - 5/s^2} \\ &\leq \left(1 - \frac{p}{2}\right)^{-1} \left(1 + \frac{5}{s}\right) \left(1 - \frac{10}{s}\right)^{-1} \\ &\leq \left(1 - \frac{p}{2}\right)^{-1} \left(1 + \frac{30}{s}\right) \end{aligned}$$

where the second inequality follows from the observation that  $(1 - p/2) \geq 1/2$ , and the third uses the hypothesis that  $s > 20$ . This proves the proposition.  $\square$

Next we show some bounds involving pairs of sums of dependent random variables that come in useful when computing second moments later. Just upper bounds turn out to be sufficient here since we are only interested in upper-bounding these second moments.

**Proposition 2.15.** *For any  $s > 10$  and  $p \in [0, 1]$ ,*

$$\Pr_{w, x, y \leftarrow U_s}[w + x \in \langle sp \rangle \wedge w + y \in \langle sp \rangle] \leq p^2 \cdot \left(1 + \frac{3}{sp}\right)^2$$

*Proof of Proposition 2.15.* We can write this probability as follows:

$$\begin{aligned} \Pr_{w, x, y \leftarrow U_s}[w + x \in \langle sp \rangle \wedge w + y \in \langle sp \rangle] &= \sum_{w = -\lfloor s/2 \rfloor}^{\lfloor s/2 \rfloor} U_s(w) \cdot \Pr_{x, y \leftarrow U_s}[w + x \in \langle sp \rangle \wedge w + y \in \langle sp \rangle] \\ &\leq \max_w \Pr_{x, y \leftarrow U_s}[w + x \in \langle sp \rangle \wedge w + y \in \langle sp \rangle] \\ &= \max_w \Pr_{x \leftarrow U_s}[w + x \in \langle sp \rangle] \cdot \Pr_{y \leftarrow U_s}[w + y \in \langle sp \rangle] \\ &= \max_w \Pr_{x \leftarrow U_s}[x \in -w + \langle sp \rangle] \cdot \Pr_{y \leftarrow U_s}[y \in -w + \langle sp \rangle] \\ &\leq \left(\frac{|\langle sp \rangle|}{|\langle s \rangle|}\right)^2 \\ &\leq \left(\frac{sp + 1}{s - 1}\right)^2 \\ &\leq p^2 \cdot \left(1 + \frac{3}{sp}\right)^2 \end{aligned} \tag{28}$$

where the second equality follows from the independence of  $x$  and  $y$ , and the last inequality uses the hypothesis that  $s > 10$  and the fact that  $p \leq 1$ .  $\square$

**Proposition 2.16.** Consider any  $s > 20$  and  $p \in [0, 1/2]$ , and let  $D$  be the distribution over  $\mathbb{Z} \times \mathbb{Z}$  sampled as follows: (Sample  $w, x, y \leftarrow U_s$  conditioned on  $((w + x), (w + y) \in \langle sp \rangle)$ , and output  $(w + x, w + y)$ ). Then,

$$\Delta_{\text{MR}}(U_{sp} \otimes U_{sp}, D) \leq (1 - p)^{-1} \left( 1 + \frac{4}{s} \right)$$

*Proof of Proposition 2.16.* Following Proposition 2.11, as  $U_{sp}^{\otimes 2}$  (denoting  $U_{sp} \otimes U_{sp}$ ) and  $D$  have the same support (that is,  $\langle sp \rangle \times \langle sp \rangle$ ), and  $U_{sp}^{\otimes 2}$  is uniform over this support, it is sufficient to bound the ratio between the maximum and minimum probability masses of  $D$  to bound this distance. We can bound this ratio as follows:

$$\Delta_{\text{MR}}(U_{sp}^{\otimes 2}, D) \leq \frac{\max_{z_1, z_2 \in \langle sp \rangle} \Pr_{w, x, y \leftarrow U_s} [w + x = z_1 \wedge w + y = z_2]}{\min_{z_1, z_2 \in \langle sp \rangle} \Pr_{w, x, y \leftarrow U_s} [w + x = z_1 \wedge w + y = z_2]}$$

where again we have ignored the Bayes normalizing factor as it appears in both the numerator and denominator. We can then express this as follows:

$$\begin{aligned} \frac{\max_{z_1, z_2 \in \langle sp \rangle} \Pr_{w, x, y \leftarrow U_s} [w + x = z_1 \wedge w + y = z_2]}{\min_{z_1, z_2 \in \langle sp \rangle} \Pr_{w, x, y \leftarrow U_s} [w + x = z_1 \wedge w + y = z_2]} &= \frac{\max_{z_1, z_2 \in \langle sp \rangle} \sum_{w \in \langle s \rangle} U_s(w) U_s(z_1 - w) U_s(z_2 - w)}{\min_{z_1, z_2 \in \langle sp \rangle} \sum_{w \in \langle s \rangle} U_s(w) U_s(z_1 - w) U_s(z_2 - w)} \\ &= \frac{\max_{z_1, z_2 \in \langle sp \rangle} |\langle s \rangle \cap (z_1 - \langle s \rangle) \cap (z_2 - \langle s \rangle)|}{\min_{z_1, z_2 \in \langle sp \rangle} |\langle s \rangle \cap (z_1 - \langle s \rangle) \cap (z_2 - \langle s \rangle)|} \\ &\leq \frac{|\langle s \rangle|}{|\langle s \rangle \cap (\lfloor sp/2 \rfloor - \langle s \rangle) \cap (-\lfloor sp/2 \rfloor - \langle s \rangle)|} \\ &\leq \frac{|\langle s \rangle|}{|\langle s \rangle| - 2 \lfloor sp/2 \rfloor} \\ &\leq \frac{(s - 1)}{(s - 1) - sp} \\ &= (1 - p)^{-1} \left( 1 - \frac{p}{(1 - p)(s - 1)} \right)^{-1} \\ &\leq (1 - p)^{-1} \left( 1 + \frac{4}{s} \right) \end{aligned}$$

where the second equality follows from the fact that the summand corresponding to each  $w$  is a fixed value when all of  $w$ ,  $(z_1 - w)$ , and  $(z_2 - w)$  are contained in  $\langle s \rangle$ , and is 0 otherwise. The first inequality follows by noting that the denominator is the size of the intersection of three intervals of integers, and is minimized when two of them are translated as far as possible in either direction. The last inequality uses the fact that  $p \leq 1/2$  and  $s > 20$ .  $\square$

## 2.2 Proof of Proposition 2.3

Fix any set of values for  $k$ ,  $n$ ,  $m$ , and  $p$  satisfying the hypothesis of Proposition 2.3. For each  $d \in [0, \log k]$  and  $i \in [k/2^d]$ , denote by  $L_i^d$  the corresponding list computed by the  $k$ -Tree algorithm as described in Figure 5, with  $L_1^0, \dots, L_k^0$  being the input lists, each of which contains  $n$  uniformly distributed integers from  $\langle m \rangle$ . Recall that we defined the variable  $C_{k,n,m,p}$  to be the number of occurrences of 0 in  $L_1^{\log k}$ ; denote this variable by  $C$  for brevity. We will write this  $C$  as a sum of

indicator variables that indicate whether each tuple of elements in the  $k$  input lists pass all the filters of the algorithm.

Each element of  $L_1^{\log k}$  corresponds to the sum  $(L_1^0[\ell_1] + \dots + L_k^0[\ell_k])$  for some  $\ell_1, \dots, \ell_k \in [n]$ . For each  $\ell_1, \dots, \ell_k \in [n]$ , denote the tuple  $(\ell_1, \dots, \ell_k)$  by  $\bar{\ell}$ , and define the following variable:

$$C_{\bar{\ell}} = \begin{cases} 1 & \text{if } (L_1^0[\ell_1] + \dots + L_k^0[\ell_k]) \text{ appears in } L_1^{\log k}, \text{ and this sum is } 0 \\ 0 & \text{otherwise} \end{cases}$$

Above, the phrase “ $(L_1^0[\ell_1] + \dots + L_k^0[\ell_k])$  appears in  $L_1^{\log k}$ ” is to be taken symbolically. That is, it means the following:

$$\forall d \in [\log k] \forall i \in \left[ \frac{k}{2^d} \right] : \left( \sum_{j=(i-1) \cdot 2^d + 1}^{i \cdot 2^d} L_j^0[\ell_j] \right) \in L_i^d$$

We can now write  $C$  as:

$$C = \sum_{\bar{\ell} \in [n]^k} C_{\bar{\ell}} \quad (29)$$

Computing the expectation of each  $C_{\bar{\ell}}$  then gives us the expectation of  $C$ . We do this as follows.

Fix any value of  $\bar{\ell} = (\ell_1, \dots, \ell_k)$ . For each  $i \in [k]$  denote by  $x_i^0$  the value of  $L_i^0[\ell_i]$ . When the input lists are uniformly random, each  $x_i^0$  is also uniformly random over  $\langle m \rangle$ . For each  $d \in [\log k]$  and  $i \in [k/2^d]$ , we also set up the following notation for the partial sums being considered at each step in the  $k$ -Tree algorithm:

$$x_i^d = x_{2i-1}^{d-1} + x_{2i}^{d-1}$$

We will later override some of these  $x_i^d$ 's by sampling them afresh rather than computing them as above, but for any  $x_i^d$  for which we do not explicitly say otherwise, the above is to be taken to be its definition. For each  $d \in [\log k]$ , define the following event that captures the set of checks made by the calls to the Merge function in the  $d^{\text{th}}$  iteration of the algorithm:

$$E_d(x_1, \dots, x_{k/2^{d-1}}) \equiv \left( \forall i \in \left[ \frac{k}{2^d} \right] : (x_{2i-1} + x_{2i}) \in \langle mp^d \rangle \right)$$

Finally, define the following event that captures the property we are interested in elements of the final list:

$$E_{\log k+1}(x) \equiv (x = 0)$$

We can now write the expectation of  $C_{\bar{\ell}}$  as follows:

$$\mathbf{E}[C_{\bar{\ell}}] = \Pr_{x_1^0, \dots, x_k^0 \leftarrow U_m} \left[ E_1(x_1^0, \dots, x_k^0) \wedge E_2(x_1^1, \dots, x_{\frac{k}{2}}^1) \wedge \dots \wedge E_{\log k}(x_1^{\log k-1}, x_2^{\log k-1}) \wedge E_{\log k+1}(x_1^{\log k}) \right]$$

We define the following sequence of probabilities of subsets of the events in the expression above that we then bound inductively to eventually arrive at a bound for the above expectation. For  $d \in [0, \log k]$ , define the following:

$$\zeta_d = \Pr_{x_1^d, \dots, x_{k/2^d}^d \leftarrow U_{mp^d}} \left[ E_{d+1}(x_1^d, \dots, x_{k/2^d}^d) \wedge E_{d+2}(x_1^{d+1}, \dots, x_{k/2^{d+1}}^{d+1}) \wedge \dots \wedge E_{\log k+1}(x_1^{\log k}) \right]$$

From the above two expressions, we have:

$$\mathbf{E}[C_{\bar{\ell}}] = \zeta_0 \quad (30)$$

The base of our induction is the following observation that follows from noting that for any  $z$ ,  $\lfloor z \rfloor \in (z \pm 1)$ :

$$\zeta_{\log k} = \Pr_{x_1^{\log k} \leftarrow U_{mp^{\log k}}} \left[ E_{\log k+1}(x_1^{\log k}) \right] = \Pr_{x \leftarrow U_{mp^{\log k}}} [x = 0] \in \frac{1}{mp^{\log k}} \left( 1 \pm \frac{1}{mp^{\log k} - 1} \right) \quad (31)$$

The inductive step uses the following claim.

**Claim 2.17.** For  $d \in [0, \log k - 1]$ , we have:  $\zeta_d \in \zeta_{d+1} \cdot \left[ p \cdot (1 \pm p) \left( 1 \pm \frac{30}{mp^{\log k}} \right) \right]^{k/2^{d+1}}$

We now complete the proof of Proposition 2.3 and then prove Claim 2.17 below.

*Proof of Proposition 2.3.* Inductively applying Claim 2.17 with  $d$  going from  $(\log k - 1)$  to 0, we get the following:

$$\zeta_0 \in \zeta_{\log k} \cdot \left[ p \cdot (1 \pm p) \left( 1 \pm \frac{30}{mp^{\log k}} \right) \right]^{\sum_{d=0}^{\log k-1} k/2^{d+1}} = \zeta_{\log k} \cdot \left[ p \cdot (1 \pm p) \left( 1 \pm \frac{30}{mp^{\log k}} \right) \right]^{k-1} \quad (32)$$

Putting together (30), (31) and (32), we get:

$$\begin{aligned} \mathbf{E}[C_{\bar{\ell}}] &= \zeta_0 \in \frac{p^k}{mp^{\log k+1}} \cdot (1 \pm p)^{k-1} \left( 1 \pm \frac{30}{mp^{\log k}} \right)^{k-1} \left( 1 \pm \frac{1}{mp^{\log k} - 1} \right) \\ &\subseteq \frac{p^k}{mp^{\log k+1}} \cdot (1 \pm p)^{k-1} \left( 1 \pm \frac{35}{mp^{\log k}} \right)^{k-1} \end{aligned} \quad (33)$$

Putting together (29) and (33), we get:

$$\mathbf{E}[C] = \sum_{\bar{\ell} \in [n]^k} \mathbf{E}[C_{\bar{\ell}}] \in \frac{n^k p^k}{mp^{\log k+1}} \cdot (1 \pm p)^{k-1} \left( 1 \pm \frac{35}{mp^{\log k}} \right)^{k-1} \quad (34)$$

as required in the statement of the proposition. Finally, we prove Claim 2.17, completing our proof of Proposition 2.3.  $\square$

*Proof of Claim 2.17.* Expanding the expression that defines  $\zeta_d$  using Bayes's theorem, we get:

$$\begin{aligned}
\zeta_d &= \Pr_{x_1^d, \dots, x_{k/2^d}^d \leftarrow U_{mp^d}} \left[ E_{d+1} \left( x_1^d, \dots, x_{k/2^d}^d \right) \wedge E_{d+2} \left( x_1^{d+1}, \dots, x_{k/2^{d+1}}^{d+1} \right) \wedge \dots \wedge E_{\log k+1} \left( x_1^{\log k} \right) \right] \\
&= \Pr_{x_1^d, \dots, x_{k/2^d}^d \leftarrow U_{mp^d}} \left[ E_{d+1} \left( x_1^d, \dots, x_{k/2^d}^d \right) \right] \cdot \\
&\quad \Pr_{x_1^{d+1}, \dots, x_{k/2^{d+1}}^{d+1} \leftarrow U_{mp^d}} \left[ E_{d+2} \left( x_1^{d+1}, \dots, x_{k/2^{d+1}}^{d+1} \right) \wedge \dots \mid E_{d+1} \left( x_1^d, \dots, x_{k/2^d}^d \right) \right] \\
&= \Pr_{x_1^d, \dots, x_{k/2^d}^d \leftarrow U_{mp^d}} \left[ E_{d+1} \left( x_1^0, \dots, x_{k/2^d}^0 \right) \right] \cdot \\
&\quad \Pr_{x_1^d, \dots, x_{k/2^d}^d \leftarrow U_{mp^d}} \left[ E_{d+2} \left( x_1^{d+1}, \dots, x_{k/2^{d+1}}^{d+1} \right) \wedge \dots \mid \forall i \in \left[ \frac{k}{2^{d+1}} \right] : (x_{2i-1}^d + x_{2i}^d) \in \langle mp^{d+1} \rangle \right] \\
&= \Pr_{x_1^d, \dots, x_{k/2^d}^d \leftarrow U_{mp^d}} \left[ E_{d+1} \left( x_1^0, \dots, x_{k/2^d}^0 \right) \right] \cdot \\
&\quad \Pr_{x_1^{d+1}, \dots, x_{k/2^{d+1}}^{d+1} \leftarrow 2 \cdot U_{mp^d}} \left[ E_{d+2} \left( x_1^{d+1}, \dots, x_{k/2^{d+1}}^{d+1} \right) \wedge \dots \mid \forall i \in \left[ \frac{k}{2^{d+1}} \right] : x_i^{d+1} \in \langle mp^{d+1} \rangle \right]
\end{aligned} \tag{35}$$

where the third equality follows from the definition of  $E_{d+1}$ , and the fourth from the definition of  $x_i^{d+1}$  as being  $(x_{2i-1}^d + x_{2i}^d)$ . We can bound the first term above directly using Proposition 2.13 as follows, noting that for the various values of  $i$ , the distributions of  $(x_{2i-1}^d + x_{2i}^d)$  are independent:

$$\begin{aligned}
\Pr_{x_1^d, \dots, x_{k/2^d}^d \leftarrow U_{mp^d}} \left[ E_{d+1} \left( x_1^0, \dots, x_k^0 \right) \right] &= \Pr_{x_1^d, \dots, x_{k/2^d}^d \leftarrow U_{mp^d}} \left[ \forall i \in \left[ \frac{k}{2^{d+1}} \right] : (x_{2i-1}^d + x_{2i}^d) \in \langle mp^{d+1} \rangle \right] \\
&\in \left( \left( p - \frac{p^2}{4} \right) \pm \frac{7}{mp^d} \right)^{k/2^{d+1}} \\
&\subseteq \left( p - \frac{p^2}{4} \right)^{k/2^{d+1}} \left( 1 \pm \frac{14}{mp^{d+1}} \right)^{k/2^{d+1}}
\end{aligned} \tag{36}$$

where the first containment follows from Proposition 2.13 (setting  $s$  there to be  $mp^d$ ), and the second from the fact that  $(p - p^2/4) > p/2$ .

The second term in (35) can be upper-bounded using Facts 2.9 and 2.10 and Proposition 2.14

(and the hypothesis that  $mp^d \geq mp^{\log k} > 30$ ) as follows:

$$\begin{aligned}
& \Pr_{x_1^{d+1}, \dots, x_{k/2^{d+1}}^{d+1} \leftarrow 2 \cdot U_{mp^d}} \left[ E_{d+2} \left( x_1^{d+1}, \dots, x_{k/2^{d+1}}^{d+1} \right) \wedge \dots \mid \forall i \in \left[ \frac{k}{2^{d+1}} \right] : x_i^{d+1} \in \langle mp^{d+1} \rangle \right] \\
&= \Pr_{x_1^{d+1}, \dots, x_{k/2^{d+1}}^{d+1} \leftarrow 2 \cdot U_{mp^d | \langle mp^{d+1} \rangle}} \left[ E_{d+2} \left( x_1^{d+1}, \dots, x_{k/2^{d+1}}^{d+1} \right) \wedge \dots \right] \\
&\leq \Delta_{\text{MR}} \left( U_{mp^{d+1}}, 2 \cdot U_{mp^d | \langle mp^{d+1} \rangle} \right)^{k/2^{d+1}} \cdot \Pr_{x_1^{d+1}, \dots, x_{k/2^{d+1}}^{d+1} \leftarrow U_{mp^{d+1}}} \left[ E_{d+2} \left( x_1^{d+1}, \dots, x_{k/2^{d+1}}^{d+1} \right) \wedge \dots \right] \\
&= \Delta_{\text{MR}} \left( U_{mp^{d+1}}, 2 \cdot U_{mp^d | \langle mp^{d+1} \rangle} \right)^{k/2^{d+1}} \cdot \zeta_{d+1} \\
&\leq \left[ \left( 1 - \frac{p}{2} \right)^{-1} \left( 1 + \frac{30}{mp^d} \right) \right]^{k/2^{d+1}} \cdot \zeta_{d+1} \tag{37}
\end{aligned}$$

We can similarly lower-bound it by:

$$\Delta_{\text{MR}} \left( U_{mp^{d+1}}, 2 \cdot U_{mp^d | \langle mp^{d+1} \rangle} \right)^{-k/2^{d+1}} \cdot \zeta_{d+1} \geq \left[ \left( 1 - \frac{p}{2} \right) \left( 1 + \frac{30}{mp^d} \right)^{-1} \right]^{k/2^{d+1}} \cdot \zeta_{d+1} \tag{38}$$

Putting together (35), (36), and (37), we get the following upper bound:

$$\begin{aligned}
\zeta_d &\leq \zeta_{d+1} \cdot \left[ \left( p - \frac{p^2}{4} \right) \left( 1 - \frac{p}{2} \right)^{-1} \left( 1 + \frac{14}{mp^{d+1}} \right) \left( 1 + \frac{30}{mp^d} \right) \right]^{k/2^{d+1}} \\
&\leq \zeta_{d+1} \cdot \left[ p(1+p) \left( 1 + \frac{29}{mp^{d+1}} \right) \right]^{k/2^{d+1}}
\end{aligned}$$

where the second inequality uses the fact that  $(1 - p/4)(1 - p/2)^{-1} < (1 + p)$ , and the hypotheses that  $mp^d \geq mp^{\log k} > 30$  and  $p < 1/30$ . Similarly, putting together (35), (36), and (38), we get:

$$\begin{aligned}
\zeta_d &\geq \zeta_{d+1} \cdot \left[ \left( p - \frac{p^2}{4} \right) \left( 1 - \frac{p}{2} \right) \left( 1 - \frac{14}{mp^{d+1}} \right) \left( 1 + \frac{30}{mp^d} \right)^{-1} \right]^{k/2^{d+1}} \\
&\geq \zeta_{d+1} \cdot \left[ p(1-p) \left( 1 - \frac{15}{mp^{d+1}} \right) \right]^{k/2^{d+1}}
\end{aligned}$$

where we use the fact that  $(1 - p/4)(1 - p/2) > (1 - p)$ , and the hypothesis that  $p < 1/30$ . The above two expressions, along with the fact that  $d$  is at most  $(\log k - 1)$ , now prove the claim.  $\square$

### 2.3 Proof of Proposition 2.4

Here, we prove Proposition 2.4, which bounds the second moment of the variable  $C_{k,n,m,p}$  defined earlier. Fix any set of values for  $k, n, m$ , and  $p$  satisfying the hypothesis of Proposition 2.4. We will continue to use the notation set up in Section 2.2, including denoting  $C_{k,n,m,p}$  by  $C$ . The second moment of  $C$  can be written as:

$$\mathbf{E} [C^2] = \mathbf{E} \left[ \left( \sum_{\bar{\ell} \in [n]^k} C_{\bar{\ell}} \right)^2 \right] = \sum_{\bar{\ell}, \bar{\ell}' \in [n]^k} \mathbf{E} [C_{\bar{\ell}} \cdot C_{\bar{\ell}'}]$$

For any tuples  $\bar{\ell}, \bar{\ell}' \in [n]^k$ , denote by  $\delta(\bar{\ell}, \bar{\ell}')$  the *Hamming difference* between them – that is,  $\delta : [n]^k \times [n]^k \rightarrow \{0, 1\}^k$  outputs a string of length  $k$  whose  $i^{\text{th}}$  bit is defined as follows:

$$(\delta(\bar{\ell}, \bar{\ell}'))_i = \begin{cases} 0 & \text{if } \ell_i = \ell'_i \\ 1 & \text{if } \ell_i \neq \ell'_i \end{cases}$$

Then, we can write the above expectation as:

$$\mathbf{E}[C^2] = \sum_{\bar{\ell}, \bar{\ell}' \in [n]^k} \mathbf{E}[C_{\bar{\ell}} \cdot C_{\bar{\ell}'}] = \sum_{s \in \{0, 1\}^k} \sum_{\bar{\ell}, \bar{\ell}' : \delta(\bar{\ell}, \bar{\ell}') = s} \mathbf{E}[C_{\bar{\ell}} \cdot C_{\bar{\ell}'}] \quad (39)$$

Our approach now is to first obtain symbolic bounds on the expectation in the sum for each value of  $\delta(\bar{\ell}, \bar{\ell}')$ , and then bound the sum itself inductively using some convenient structure that it possesses. The extremes have slightly different bounds from the general case, so we first separate them as follows.

**Claim 2.18.** *For any  $\bar{\ell}, \bar{\ell}' \in [n]^k$  such that  $\delta(\bar{\ell}, \bar{\ell}') = 0^k$ ,*

$$\mathbf{E}[C_{\bar{\ell}} \cdot C_{\bar{\ell}'}] = \mathbf{E}[C_{\bar{\ell}}] \in \frac{p^k}{mp^{\log k+1}} \cdot (1 \pm p)^{k-1} \left(1 \pm \frac{35}{mp^{\log k}}\right)^{k-1}$$

*Proof of Claim 2.18.* Because  $\bar{\ell} = \bar{\ell}'$  and  $C_{\bar{\ell}}$  and  $C_{\bar{\ell}'}$  are Boolean variables,  $C_{\bar{\ell}} \cdot C_{\bar{\ell}'} = C_{\bar{\ell}}$ , the expectation of which is bounded in (33) from Section 2.2.  $\square$

The bounds on the expectations in the general case are most conveniently expressed in terms of quantities that are defined on top of the difference  $\delta(\bar{\ell}, \bar{\ell}')$  in a tree structure that reflects the processing of the list elements by the algorithm. To capture these, we define a “tree extrapolation” function  $tex : \{0, 1\}^k \rightarrow \{0, 1\}^{k/2} \times \{0, 1\}^{k/4} \times \dots \times \{0, 1\}$  that on input an  $s^0 \in \{0, 1\}^k$ , outputs a tuple of strings  $(s^1, \dots, s^{\log k})$ , where for  $d \in [\log k]$ , the string  $s^d$  is contained in  $\{0, 1\}^{k/2^d}$ , and its  $i^{\text{th}}$  bit is defined as follows for  $i \in [k/2^d]$ :

$$s_i^d = \max(s_{2i-1}^{d-1}, s_{2i}^{d-1})$$

Further, we define the function  $stex : \{0, 1\}^k \rightarrow \mathbb{N}$  that on input  $s \in \{0, 1\}^k$ , computes the tree extrapolation  $(s^1, \dots, s^{\log k}) \leftarrow tex(s)$ , and then outputs the sum  $\sum_{d \in [\log k]} \sum_{i \in [k/2^d]} s_i^d$ .

**Claim 2.19.** *Consider any  $s \in \{0, 1\}^k \setminus \{0^k\}$ , and let  $\sigma \leftarrow stex(s)$ . For any  $\bar{\ell}, \bar{\ell}' \in [n]^k$  such that  $\delta(\bar{\ell}, \bar{\ell}') = s$ , we have:*

$$\mathbf{E}[C_{\bar{\ell}} \cdot C_{\bar{\ell}'}] \leq \frac{p^{k+\sigma+1}}{(mp^{\log k+1})^2} \cdot \left[ (1-p)^{-1} \left( 1 + \frac{30}{mp^{\log k}} \right) \right]^{(k+\sigma+1)}$$

We use these claims to prove Proposition 2.4, and then prove Claim 2.19.

*Proof of Proposition 2.4.* Fix any valid set of parameters  $k, n, m$ , and  $p$  that satisfy the hypothesis of the proposition. Define the following symbols for convenience:

$$\mu = p \cdot \left[ (1-p)^{-1} \left( 1 + \frac{35}{mp^{\log k}} \right) \right]$$

$$\nu = \frac{1}{mp^{\log k+1}}$$



For  $s \in \{0, 1\}^k$ , denote by  $(s^1, \dots, s^{\log k})$  the outputs of the tree extension function  $tex(s)$ . Note that the bit  $s_1^{\log k}$  is 0 if and only if  $s = 0^k$ . For any  $s$  and  $\bar{\ell}, \bar{\ell}'$  such that  $\delta(\bar{\ell}, \bar{\ell}') = s$ , we have the following by Claims 2.18 and 2.19 (and noting that  $(1+p) \leq (1-p)^{-1}$ ):

$$\mathbf{E}[C_{\bar{\ell}} \cdot C_{\bar{\ell}'}] \leq \mu^{k+stex(s)} \cdot \nu \cdot (\mu\nu)^{s_1^{\log k}} \quad (40)$$

For  $s \in \{0, 1\}^k$ , denote by  $wt(s)$  the Hamming weight of  $s$  – that is, the sum  $\sum_{i \in [k]} s_i$ . For any  $s \in \{0, 1\}^k$ , the number of pairs  $\bar{\ell}, \bar{\ell}' \in [n]^k$  such that  $\delta(\bar{\ell}, \bar{\ell}') = s$  is given by:

$$n^k \cdot (n-1)^{wt(s)} \leq n^{k+wt(s)} \quad (41)$$

Putting together (39, 40, 41), we have:

$$\begin{aligned} \mathbf{E}[C^2] &= \sum_{s \in \{0,1\}^k} \sum_{\bar{\ell}, \bar{\ell}': \delta(\bar{\ell}, \bar{\ell}')=s} \mathbf{E}[C_{\bar{\ell}} \cdot C_{\bar{\ell}'}] \\ &\leq \sum_{s \in \{0,1\}^k} n^{k+wt(s)} \cdot \mu^{k+stex(s)} \cdot \nu \cdot (\mu\nu)^{s_1^{\log k}} \\ &= (n^k \mu^k \nu) \cdot \sum_{s \in \{0,1\}^k} n^{wt(s)} \cdot \mu^{stex(s)} \cdot (\mu\nu)^{s_1^{\log k}} \end{aligned} \quad (42)$$

We show that the sum in the above expression can be simplified to a recursively defined function of  $k$ , and to do so we define the following function for every  $k \geq 2$  that is a power of 2:

$$T(k) = \sum_{s \in \{0,1\}^k} n^{wt(s)} \cdot \mu^{stex(s)} \cdot (\mu\nu)^{s_1^{\log k}} - 1 = (\mu\nu) \cdot \sum_{s \in \{0,1\}^k: s \neq 0^k} n^{wt(s)} \mu^{stex(s)}$$

where the second equality follows from the fact that  $wt(0^k) = stex(0^k) = (0^k)_1^{\log k} = 0$ . We also define  $T(1)$  to be  $(n \cdot \mu \cdot \nu)$ . For any  $k \geq 2$ , by (42) we have:

$$\mathbf{E}[C^2] \leq (n^k \mu^k \nu) \cdot (1 + T(k)) \quad (43)$$

It may be verified by computation that:

$$T(2) = 2n\mu^2\nu + n^2\mu^2\nu = T(1) \cdot \left( \frac{T(1)}{\nu} + 2\mu \right) \quad (44)$$

We show that a similar relation holds between  $T(k)$  and  $T(k/2)$  for other values of  $k$  as well. Generalize the definitions of  $wt$  and  $stex$  to input strings of length  $k/2$  in the natural manner. We

then do this as follows:

$$\begin{aligned}
T(k) &= \sum_{s \in \{0,1\}^k} n^{wt(s)} \cdot \mu^{stex(s)} \cdot (\mu\nu)^{s_1^{\log k}} - 1 \\
&= \sum_{s \in \{0,1\}^k: s \neq 0^k} n^{wt(s)} \cdot \mu^{stex(s)} \cdot (\mu\nu) \\
&= \sum_{q,r \in \{0,1\}^{k/2}: (q||r) \neq 0^k} n^{wt(q||r)} \cdot \mu^{stex(q||r)} \cdot (\mu\nu) \\
&= \sum_{q,r \in \{0,1\}^{k/2}: (q||r) \neq 0^k} n^{wt(q)+wt(r)} \cdot \mu^{stex(q)+stex(r)+1} \cdot (\mu\nu) \\
&= (\mu^2\nu) \cdot \sum_{q,r \in \{0,1\}^{k/2}: (q||r) \neq 0^k} n^{wt(q)+wt(r)} \cdot \mu^{stex(q)+stex(r)} \\
&= (\mu^2\nu) \cdot \left( \sum_{q,r \in \{0,1\}^{k/2}} n^{wt(q)+wt(r)} \cdot \mu^{stex(q)+stex(r)} - 1 \right) \\
&= (\mu^2\nu) \cdot \left( \left( \sum_{q \in \{0,1\}^{k/2}} n^{wt(q)} \cdot \mu^{stex(q)} \right)^2 - 1 \right) \\
&= (\mu^2\nu) \cdot \left( \left( \frac{T(k/2)}{\mu\nu} + 1 \right)^2 - 1 \right) \\
&= \mu \cdot T(k/2) \cdot \left( \frac{T(k/2)}{\mu\nu} + 2 \right) \\
&= T(k/2) \cdot \left( \frac{T(k/2)}{\nu} + 2\mu \right) \tag{45}
\end{aligned}$$

Above, in the second, sixth, and eighth equalities, we use the fact that  $wt(0^k) = stex(0^k) = 0$ . The fourth equality follows from the fact that  $wt$  is additive under concatenation of strings, and  $stex(q||r)$ , because of its tree-based definition, is equal to  $stex(q) + stex(r) + 1$  if  $(q||r)$  is not  $0^k$ . The rest are straightforward algebraic manipulations. The definition of  $T(1)$  and (44,45) imply that this function  $T$  is exactly the function  $T_{\mu,\nu,n}$  specified in the statement of the proposition. This observation together with (43) now proves Proposition 2.4.  $\square$

*Proof of Claim 2.19.* Fix any  $s \in \{0,1\}^k \setminus \{0^k\}$  with  $\sigma \leftarrow stex(s)$ , and tuples  $\bar{\ell}^x, \bar{\ell}^y \in [n]^k$  such that  $\delta(\bar{\ell}^x, \bar{\ell}^y) = s$ . We set up notation similar to that used in Section 2.2. For each  $i \in [k]$ , let:

$$\begin{aligned}
x_0^i &= L_i[\ell_i^x] \\
y_0^i &= L_i[\ell_i^y]
\end{aligned}$$

For each  $d \in [\log k]$  and  $i \in [k/2^d]$ , unless overridden, define the following:

$$\begin{aligned}
x_i^d &= x_{2i-1}^{d-1} + x_{2i}^{d-1} \\
y_i^d &= y_{2i-1}^{d-1} + y_{2i}^{d-1}
\end{aligned}$$

We sometimes denote by  $\bar{x}^d$  (similarly  $\bar{y}^d$ ) the tuple  $(x_1^d, \dots, x_{k/2^d}^d)$ . In our proof, we will rely on the fact that for  $i \in [k]$  on which  $\bar{\ell}^x$  and  $\bar{\ell}^y$  differ, the values of  $x_i^0 = L_i[\ell_i^x]$  and  $y_i^0 = L_i[\ell_i^y]$  are independent. To enable us to argue fluidly about this, we define the following indicator constants<sup>6</sup> for  $i \in [k]$ :

$$\text{Ind}_i^0 = \begin{cases} 1 & \text{if } \ell_i^x \neq \ell_i^y \\ 0 & \text{if } \ell_i^x = \ell_i^y \end{cases}$$

Similarly, to capture the independence of intermediate values computed in the course of the  $k$ -Tree algorithm, we extend this to define the following for each  $d \in [\log k]$  and  $i \in [k/2^d]$ :

$$\text{Ind}_i^d = \max(\text{Ind}_{2i-1}^{d-1}, \text{Ind}_{2i}^{d-1})$$

That is,  $\text{Ind}_i^d$  indicates whether the values computed at the  $i^{\text{th}}$  “node” at depth  $d$  of the  $k$ -Tree corresponding to the two tuples of indices  $\bar{\ell}^x$  and  $\bar{\ell}^y$  (that is,  $x_i^d$  and  $y_i^d$ ) are symbolically equal (in which case it is 0). Note that the  $\text{Ind}_i^d$ 's correspond directly to bits in the “tree extrapolation”  $\text{tex}(s)$  of  $s = \delta(\bar{\ell}^x, \bar{\ell}^y)$  as defined before the statement of Claim 2.19. In particular, we have the following:

$$\sum_{d \in [\log k]} \sum_{i \in [k/2^d]} \text{Ind}_i^d = \text{tex}(s) = \sigma \quad (46)$$

As in Section 2.2, for each  $d \in [\log k]$ , define the following event:

$$E_d(x_1, \dots, x_{k/2^{d-1}}) \equiv \left( \forall i \in \left[ \frac{k}{2^d} \right] : (x_{2i-1} + x_{2i}) \in \langle mp^d \rangle \right)$$

And the following event that captures the property we are interested in elements of the final list:

$$E_{\log k+1}(x) \equiv (x = 0)$$

Finally, we define the following events that enforce the conditions of independence captured by the  $\text{Ind}_i^d$ 's. For each  $d \in [0, \log k]$ :

$$V_d \equiv \left( \forall i \in \left[ \frac{k}{2^d} \right] \text{ such that } (\text{Ind}_i^d = 0) : x_i^d = y_i^d \right)$$

The expectation we care about can now be written as follows:

$$\mathbf{E}[C_{\bar{\ell}^x} \cdot C_{\bar{\ell}^y}] = \Pr_{\bar{x}^0, \bar{y}^0 \leftarrow (U_m)^k} \left[ (E_1(\bar{x}^0) \wedge E_1(\bar{y}^0)) \wedge \dots \wedge (E_{\log k+1}(\bar{x}^{\log k}) \wedge E_{\log k+1}(\bar{y}^{\log k})) \mid V_0 \right] \quad (47)$$

As in Section 2.2, we will bound this quantity inductively. For  $d \in [0, \log k]$ , define the following:

$$\chi_d = \Pr_{\bar{x}^d, \bar{y}^d \leftarrow (U_{mp^d})^{k/2^d}} \left[ (E_{d+1}(\bar{x}^d) \wedge E_{d+1}(\bar{y}^d)) \wedge (E_{d+2}(\bar{x}^{d+1}) \wedge E_{d+2}(\bar{y}^{d+1})) \wedge \dots \mid V_d \right]$$

---

<sup>6</sup>These are constants because we have already fixed  $\bar{\ell}^x$  and  $\bar{\ell}^y$ .

By definition, we have:

$$\mathbf{E} [C_{\bar{\ell}^x} \cdot C_{\bar{\ell}^y}] = \chi_0 \quad (48)$$

As the base for our induction, we will show a bound on  $\chi_{\log k}$ . Note that as  $\delta(\bar{\ell}^x, \bar{\ell}^y) \neq 0^k$ , the value of  $\text{Ind}_1^{\log k}$  is always 1, and so  $V_{\log k}$  is vacuously a tautology. So we have:

$$\begin{aligned} \chi_{\log k} &= \Pr_{x_1^{\log k}, y_1^{\log k} \leftarrow U_{mp^{\log k}}} \left[ E_{\log k+1}(x_1^{\log k}) \wedge E_{\log k+1}(y_1^{\log k}) \mid V_{\log k} \right] \\ &= \Pr_{x_1^{\log k}, y_1^{\log k} \leftarrow U_{mp^{\log k}}} \left[ E_{\log k+1}(x_1^{\log k}) \wedge E_{\log k+1}(y_1^{\log k}) \right] \\ &= \Pr_{x_1^{\log k} \leftarrow U_{mp^{\log k}}} \left[ E_{\log k+1}(x_1^{\log k}) \right] \cdot \Pr_{y_1^{\log k} \leftarrow U_{mp^{\log k}}} \left[ E_{\log k+1}(y_1^{\log k}) \right] \\ &\leq \frac{1}{(mp^{\log k} - 1)^2} = \frac{1}{(mp^{\log k})^2} \left( 1 + \frac{1}{mp^{\log k} - 1} \right)^2 \end{aligned} \quad (49)$$

The following claim enables our induction.

**Claim 2.20.** *For  $d \in [0, \log k - 1]$ , we have:*

$$\chi_d \leq \chi_{d+1} \cdot \left[ p(1-p)^{-1} \left( 1 + \frac{30}{mp^{\log k}} \right) \right]^{\binom{k}{2^{d+1}} + \sum_{i=1}^{k/2^{d+1}} \text{Ind}_i^{d+1}}$$

We continue with the proof of Claim 2.19 now, and then prove Claim 2.20 later below. Applying Claim 2.20 repeatedly, we get the following relationship:

$$\begin{aligned} \chi_0 &\leq \chi_{\log k} \cdot \left[ p(1-p)^{-1} \left( 1 + \frac{30}{mp^{\log k}} \right) \right]^{\sum_{d=0}^{\log k-1} \binom{k}{2^{d+1}} + \sum_{i=1}^{k/2^{d+1}} \text{Ind}_i^{d+1}} \\ &= \chi_{\log k} \cdot \left[ p(1-p)^{-1} \left( 1 + \frac{30}{mp^{\log k}} \right) \right]^{\binom{k-1}{2^1} + \sum_{d=0}^{\log k-1} \sum_{i=1}^{k/2^{d+1}} \text{Ind}_i^{d+1}} \\ &= \chi_{\log k} \cdot \left[ p(1-p)^{-1} \left( 1 + \frac{30}{mp^{\log k}} \right) \right]^{\binom{k-1}{2^0} + \sum_{d=1}^{\log k} \sum_{i=1}^{k/2^d} \text{Ind}_i^d} \\ &= \chi_{\log k} \cdot \left[ p(1-p)^{-1} \left( 1 + \frac{30}{mp^{\log k}} \right) \right]^{(k-1)+\sigma} \end{aligned} \quad (50)$$

where the last equality follows from (46). Putting together (49) and (50), we have:

$$\begin{aligned} \chi_0 &\leq \chi_{\log k} \cdot \left[ p(1-p)^{-1} \left( 1 + \frac{30}{mp^{\log k}} \right) \right]^{(k+\sigma-1)} \\ &\leq \frac{p^{k+\sigma-1}}{(mp^{\log k})^2} \cdot \left[ (1-p)^{-1} \left( 1 + \frac{30}{mp^{\log k}} \right) \right]^{(k+\sigma-1)} \left( 1 + \frac{1}{mp^{\log k} - 1} \right)^2 \\ &\leq \frac{p^{k+\sigma+1}}{(mp^{\log k+1})^2} \cdot \left[ (1-p)^{-1} \left( 1 + \frac{30}{mp^{\log k}} \right) \right]^{(k+\sigma+1)} \end{aligned}$$

Together with (48), this proves Claim 2.19.  $\square$

*Proof of Claim 2.20.* As in the proof of Claim 2.17, we can write  $\chi_d$  as follows:

$$\begin{aligned}
\chi_d &= \Pr_{\bar{x}^d, \bar{y}^d \leftarrow (U_{mp^d})^{k/2^d}} \left[ \left( E_{d+1}(\bar{x}^d) \wedge E_{d+1}(\bar{y}^d) \right) \wedge \left( E_{d+2}(\bar{x}^{d+1}) \wedge E_{d+2}(\bar{y}^{d+1}) \right) \wedge \cdots \mid V_d \right] \\
&= \Pr_{\bar{x}^d, \bar{y}^d \leftarrow (U_{mp^d})^{k/2^d}} \left[ E_{d+1}(\bar{x}^d) \wedge E_{d+1}(\bar{y}^d) \mid V_d \right] \cdot \\
&\quad \Pr_{\bar{x}^d, \bar{y}^d \leftarrow (U_{mp^d})^{k/2^d}} \left[ \left( E_{d+2}(\bar{x}^{d+1}) \wedge E_{d+2}(\bar{y}^{d+1}) \right) \wedge \cdots \mid V_d \wedge \left( E_{d+1}(\bar{x}^d) \wedge E_{d+1}(\bar{y}^d) \right) \right] \quad (51)
\end{aligned}$$

As in the proof of Claim 2.17, we would like to bound the two terms in the right-hand size of (51) separately. In each case, we do this by looking at the various nodes at level  $(d+1)$  of the  $k$ -Tree separately. As the values of  $x_i^{d+1}$  and  $x_{i'}^{d+1}$  are independent for distinct values of  $i$  and  $i'$  (and the same with the  $y$ 's), the first term can be written as the following product:

$$\begin{aligned}
&\Pr_{\bar{x}^d, \bar{y}^d \leftarrow (U_{mp^d})^{k/2^d}} \left[ E_{d+1}(\bar{x}^d) \wedge E_{d+1}(\bar{y}^d) \mid V_d \right] \\
&= \Pr_{\bar{x}^d, \bar{y}^d \leftarrow (U_{mp^d})^{k/2^d}} \left[ \forall i \in \left[ \frac{k}{2^{d+1}} \right] : (x_{2i-1}^d + x_{2i}^d) \in \langle mp^{d+1} \rangle \wedge (y_{2i-1}^d + y_{2i}^d) \in \langle mp^{d+1} \rangle \mid V_d \right] \\
&= \prod_{i=0}^{k/2^{d+1}} \Pr_{x_{2i-1}^d, x_{2i}^d, y_{2i-1}^d, y_{2i}^d \leftarrow U_{mp^d}} \left[ (x_{2i-1}^d + x_{2i}^d) \in \langle mp^{d+1} \rangle \wedge (y_{2i-1}^d + y_{2i}^d) \in \langle mp^{d+1} \rangle \mid V_d \right] \quad (52)
\end{aligned}$$

We bound the terms in the product in (52) separately. For  $i \in [k/2^{d+1}]$  such that  $\text{Ind}_i^{d+1} = 0$ , due to the conditioning on  $V_d$ , the events  $(x_{2i-1}^d + x_{2i}^d) \in \langle mp^{d+1} \rangle$  and  $(y_{2i-1}^d + y_{2i}^d) \in \langle mp^{d+1} \rangle$  are identical, and the probability them both occurring is bounded as follows:

$$\begin{aligned}
&\Pr_{x_{2i-1}^d, x_{2i}^d, y_{2i-1}^d, y_{2i}^d \leftarrow U_{mp^d}} \left[ (x_{2i-1}^d + x_{2i}^d) \in \langle mp^{d+1} \rangle \wedge (y_{2i-1}^d + y_{2i}^d) \in \langle mp^{d+1} \rangle \mid V_d \right] \\
&= \Pr_{x_{2i-1}^d, x_{2i}^d \leftarrow U_{mp^d}} \left[ (x_{2i-1}^d + x_{2i}^d) \in \langle mp^{d+1} \rangle \right] \\
&\leq \left( p - \frac{p^2}{4} \right) + \frac{7}{mp^d} \leq p \cdot \left( 1 + \frac{7}{mp^{d+1}} \right) \quad (53)
\end{aligned}$$

where the last inequality follows from Proposition 2.13. Next, consider the  $i$  such that  $\text{Ind}_i^{d+1} = \text{Ind}_{2i-1}^d = \text{Ind}_{2i}^d = 1$ . In this case, conditioning on  $V_d$  does not affect the independence of the  $x$  and  $y$  variables, and we can write the above probability as follows:

$$\begin{aligned}
&\Pr_{x_{2i-1}^d, x_{2i}^d, y_{2i-1}^d, y_{2i}^d \leftarrow U_{mp^d}} \left[ (x_{2i-1}^d + x_{2i}^d) \in \langle mp^{d+1} \rangle \wedge (y_{2i-1}^d + y_{2i}^d) \in \langle mp^{d+1} \rangle \mid V_d \right] \\
&= \Pr_{x_{2i-1}^d, x_{2i}^d \leftarrow U_{mp^d}} \left[ (x_{2i-1}^d + x_{2i}^d) \in \langle mp^{d+1} \rangle \right] \cdot \Pr_{y_{2i-1}^d, y_{2i}^d \leftarrow U_{mp^d}} \left[ (y_{2i-1}^d + y_{2i}^d) \in \langle mp^{d+1} \rangle \right] \\
&\leq p^2 \cdot \left( 1 + \frac{7}{mp^{d+1}} \right)^2 \quad (54)
\end{aligned}$$

where the last inequality is from Proposition 2.13. Finally, consider the  $i$ 's for which  $\text{Ind}_i^{d+1} = 1$  and exactly one of  $\text{Ind}_{2i-1}^d$  and  $\text{Ind}_{2i}^d$  is 1. Suppose, without loss of generality, that  $\text{Ind}_{2i-1}^d = 1$ . Then,

conditioning on  $V_d$ , the variables  $x_{2i-1}^d$  and  $y_{2i-1}^d$  are equal and uniformly distributed, whereas  $x_{2i}^d$  and  $y_{2i}^d$  are independent and uniformly distributed. Thus, in this case, we can use Proposition 2.15 to bound the probability as follows:

$$\begin{aligned}
& \Pr_{x_{2i-1}^d, x_{2i}^d, y_{2i-1}^d, y_{2i}^d \leftarrow U_{mp^d}} \left[ (x_{2i-1}^d + x_{2i}^d) \in \langle mp^{d+1} \rangle \wedge (y_{2i-1}^d + y_{2i}^d) \in \langle mp^{d+1} \rangle \mid V_d \right] \\
&= \Pr_{x_{2i-1}^d, x_{2i}^d, y_{2i}^d \leftarrow U_{mp^d}} \left[ (x_{2i-1}^d + x_{2i}^d) \in \langle mp^{d+1} \rangle \wedge (x_{2i-1}^d + y_{2i}^d) \in \langle mp^{d+1} \rangle \right] \\
&\leq p^2 \cdot \left( 1 + \frac{3}{mp^{d+1}} \right)^2 \leq p^2 \cdot \left( 1 + \frac{7}{mp^{d+1}} \right)^2
\end{aligned} \tag{55}$$

Putting together (52-55), we get:

$$\Pr_{\bar{x}^d, \bar{y}^d \leftarrow (U_{mp^d})^{k/2^d}} \left[ E_{d+1}(\bar{x}^d) \wedge E_{d+1}(\bar{y}^d) \mid V_d \right] \leq \left[ p \cdot \left( 1 + \frac{7}{mp^{d+1}} \right) \right]^{\left( \frac{k}{2^{d+1}} + \sum_{i=1}^{k/2^{d+1}} \text{Ind}_i^{d+1} \right)} \tag{56}$$

Next we bound the second term in the right-hand side of (51), which, to recall, is the following:

$$\Pr_{\bar{x}^d, \bar{y}^d \leftarrow (U_{mp^d})^{k/2^d}} \left[ \left( E_{d+2}(\bar{x}^{d+1}) \wedge E_{d+2}(\bar{y}^{d+1}) \right) \wedge \dots \mid V_d \wedge \left( E_{d+1}(\bar{x}^d) \wedge E_{d+1}(\bar{y}^d) \right) \right] \tag{57}$$

As in the proof of Claim 2.17, we bound this in terms of  $\chi_{d+1}$  and the distance between the distributions of  $x_i^{d+1}$ 's and  $y_i^{d+1}$ 's as sampled in the above expression and the corresponding uniform distributions. Note that  $\chi_{d+1}$  is the following:

$$\chi_{d+1} = \Pr_{\bar{x}^{d+1}, \bar{y}^{d+1} \leftarrow (U_{mp^{d+1}})^{k/2^{d+1}}} \left[ \left( E_{d+2}(\bar{x}^{d+1}) \wedge E_{d+2}(\bar{y}^{d+1}) \right) \wedge \dots \mid V_{d+1} \right] \tag{58}$$

By Fact 2.9, the (multiplicative) difference between (57) and  $\chi_{d+1}$  is bounded by the distance between the conditioned distributions from which  $(\bar{x}^{d+1}, \bar{y}^{d+1})$  is sampled in the two expressions. These distributions can be written in terms of the following two distributions defined for each  $i \in [k/2^{d+1}]$ :

$$\begin{aligned}
D_i^1 &\equiv \left[ \begin{array}{l} \text{Sample } x_{2i-1}^d, x_{2i}^d, y_{2i-1}^d, y_{2i}^d \leftarrow U_{mp^d} \\ \text{If } \text{Ind}_{2i-1}^d = 0, \text{ reassign } y_{2i-1}^d = x_{2i-1}^d \\ \text{If } \text{Ind}_{2i}^d = 0, \text{ reassign } y_{2i}^d = x_{2i}^d \\ \text{Set } x_i^{d+1} = (x_{2i-1}^d + x_{2i}^d) \text{ and } y_i^{d+1} = (y_{2i-1}^d + y_{2i}^d) \\ \text{Condition on } x_i^{d+1}, y_i^{d+1} \in \langle mp^{d+1} \rangle \\ \text{Output } (x_i^{d+1}, y_i^{d+1}) \end{array} \right] \\
D_i^0 &\equiv \left[ \begin{array}{l} \text{Sample } x_i^{d+1}, y_i^{d+1} \leftarrow U_{mp^{d+1}} \\ \text{If } \text{Ind}_i^{d+1} = 0, \text{ reassign } y_i^{d+1} = x_i^{d+1} \\ \text{Output } (x_i^{d+1}, y_i^{d+1}) \end{array} \right]
\end{aligned}$$

By the definitions of the  $E_d$ 's and  $V_d$ 's, it may be seen that the distribution of  $(\bar{x}^{d+1}, \bar{y}^{d+1})$  in (57) is essentially  $(D_1^1 \otimes \dots \otimes D_{k/2^{d+1}}^1)$ , while that in (58) is essentially  $(D_1^0 \otimes \dots \otimes D_{k/2^{d+1}}^0)$ . Thus, if

we bound the distance between  $D_i^0$  and  $D_i^1$  for each  $i$ , the multiplicative difference between these expressions, by Facts 2.9 and 2.10 will be bounded by the product of these distances.

Consider any  $i \in [k/2^{d+1}]$  for which  $\text{Ind}_i^{d+1} = 0$ , which implies that  $\text{Ind}_{2i-1}^d = \text{Ind}_{2i}^d = 0$ , and so the samples of both distributions are fully determined by the  $x$  variables. Then, by the definitions of the distributions, for such  $i$  we have:

$$\begin{aligned} \Delta_{\text{MR}}(D_i^0, D_i^1) &= \Delta_{\text{MR}}\left(U_{mp^{d+1}}, 2 \cdot U_{mp^d} |_{\langle mp^{d+1} \rangle}\right) \\ &\leq \left(1 - \frac{p}{2}\right)^{-1} \left(1 + \frac{30}{mp^d}\right) \\ &\leq (1-p)^{-1} \left(1 + \frac{30}{mp^d}\right) \end{aligned} \quad (59)$$

where the first inequality follows from Proposition 2.14. Next, consider an  $i$  such that  $\text{Ind}_i^{d+1} = \text{Ind}_{2i-1}^d = \text{Ind}_{2i}^d = 1$ . In this case, the  $x$  variables and the  $y$  variables are independent, and each behaves exactly like the  $x$  variables in the previous case. So, using Fact 2.10, for such  $i$  we have:

$$\begin{aligned} \Delta_{\text{MR}}(D_i^0, D_i^1) &= \Delta_{\text{MR}}\left(U_{mp^{d+1}}^{\otimes 2}, 2 \cdot U_{mp^d} |_{\langle mp^{d+1} \rangle}^{\otimes 2}\right) \\ &\leq \Delta_{\text{MR}}\left(U_{mp^{d+1}}, 2 \cdot U_{mp^d} |_{\langle mp^{d+1} \rangle}\right)^2 \\ &\leq \left[(1-p)^{-1} \left(1 + \frac{30}{mp^d}\right)\right]^2 \end{aligned} \quad (60)$$

Finally, consider  $i$  such that  $\text{Ind}_i^{d+1} = 1$  and exactly one of  $\text{Ind}_{2i-1}^d$  and  $\text{Ind}_{2i}^d$  is 1; without loss of generality, suppose  $\text{Ind}_{2i-1}^d = 0$ . Then, in  $D_i^1$ ,  $x_{2i-1}^d = y_{2i-1}^d$  is sampled from  $U_{mp^d}$ , and is added to  $x_{2i}^d$  and  $y_{2i}^d$ , which are sampled independently from  $U_{mp^d}$ , to get the outputs, whereas  $D_i^0$  simply outputs independent samples from  $U_{mp^{d+1}}$ . We can then use Proposition 2.16 to bound the distance between these as:

$$\Delta_{\text{MR}}(D_i^0, D_i^1) \leq (1-p)^{-1} \left(1 + \frac{4}{mp^d}\right) \leq \left[(1-p)^{-1} \left(1 + \frac{30}{mp^d}\right)\right]^2 \quad (61)$$

Putting together (58-61) and applying Facts 2.9 and 2.10, we get:

$$\begin{aligned} &\Pr_{\bar{x}^d, \bar{y}^d \leftarrow (U_{mp^d})^{k/2^d}} \left[ \left( E_{d+2}(\bar{x}^{d+1}) \wedge E_{d+2}(\bar{y}^{d+1}) \right) \wedge \dots \mid V_d \wedge \left( E_{d+1}(\bar{x}^d) \wedge E_{d+1}(\bar{y}^d) \right) \right] \\ &\leq \chi_{d+1} \cdot \left[ (1-p)^{-1} \left(1 + \frac{30}{mp^d}\right) \right]^{\binom{k}{2^{d+1}} + \sum_{i=1}^{k/2^{d+1}} \text{Ind}_i^{d+1}} \end{aligned} \quad (62)$$

Putting together (51), (56), and (62), we get:

$$\begin{aligned} \chi_d &\leq \chi_{d+1} \cdot \left[ p(1-p)^{-1} \left(1 + \frac{7}{mp^{d+1}}\right) \left(1 + \frac{30}{mp^d}\right) \right]^{\binom{k}{2^{d+1}} + \sum_{i=1}^{k/2^{d+1}} \text{Ind}_i^{d+1}} \\ &\leq \chi_{d+1} \cdot \left[ p(1-p)^{-1} \left(1 + \frac{30}{mp^{\log k}}\right) \right]^{\binom{k}{2^{d+1}} + \sum_{i=1}^{k/2^{d+1}} \text{Ind}_i^{d+1}} \end{aligned}$$

where the second inequality uses the hypotheses that  $mp^d \geq mp^{\log k} \geq 30$ , and  $p < 1/30$ . This proves the claim.  $\square$

## 2.4 Proof of Proposition 2.5

Much of this proof proceeds along the lines of that of Proposition 2.3, and parts of the proof below are reproduced from there nearly verbatim. Fix any set of values for  $k$ ,  $n$ ,  $m$ , and  $p$  satisfying the hypothesis of Proposition 2.5. For each  $d \in [0, \log k]$  and  $i \in [k/2^d]$ , denote by  $L_i^d$  the corresponding list computed by the  $k$ -Tree algorithm as described in Figure 5, with  $L_1^0, \dots, L_k^0$  being the input lists, each of which contains  $n$  uniformly distributed integers from  $\langle m \rangle$ . Recall that we defined the variable  $\Lambda_{k,n,m,p}$  to be the total size of all of these lists; denote this variable by  $\Lambda$  for brevity. For each  $d \in [0, \log k]$ , denote by  $\Lambda_d$  the size of the list  $L_1^d$ . For any  $d$ , due to symmetry, the expected size of all lists  $L_i^d$  is the same. Thus, we have:

$$\mathbf{E}[\Lambda] = \mathbf{E} \left[ \sum_{d \in [0, \log k]} \sum_{i \in [k/2^d]} |L_i^d| \right] = \sum_{d \in [0, \log k]} \sum_{i \in [k/2^d]} \mathbf{E} [|L_i^d|] = \sum_{d \in [0, \log k]} \frac{k}{2^d} \cdot \mathbf{E}[\Lambda_d] \quad (63)$$

We will then bound each  $\Lambda_d$  separately and use the above sum to bound  $\Lambda$ . By design, for  $d = 0$ , we have:

$$\mathbf{E}[\Lambda_0] = n \quad (64)$$

Fix any  $d \in [1, \log k]$ . As in the proof of Proposition 2.3, we will write  $\Lambda_d$  as a sum of variables that indicate whether each tuple of elements in the  $2^d$  input lists that contribute to the list  $L_1^d$  pass all the filters of the algorithm until that point. Each element of  $L_1^d$  corresponds to the sum  $(L_1^0[\ell_1] + \dots + L_{2^d}^0[\ell_{2^d}])$  for some  $\ell_1, \dots, \ell_{2^d} \in [n]$ . For each  $\ell_1, \dots, \ell_{2^d} \in [n]$ , denote the tuple  $(\ell_1, \dots, \ell_{2^d})$  by  $\bar{\ell}$ , and define the following variable:

$$\Lambda_{d, \bar{\ell}} = \begin{cases} 1 & \text{if } (L_1^0[\ell_1] + \dots + L_{2^d}^0[\ell_{2^d}]) \text{ appears in } L_1^d \\ 0 & \text{otherwise} \end{cases}$$

We can now write  $\Lambda_d$  as:

$$\Lambda_d = \sum_{\bar{\ell} \in [n]^{2^d}} \Lambda_{d, \bar{\ell}} \quad (65)$$

Computing the expectation of each  $\Lambda_{d, \bar{\ell}}$  then gives us the expectation of  $\Lambda_d$ . We do this as follows.

Fix any value of  $\bar{\ell} = (\ell_1, \dots, \ell_{2^d})$ . For each  $i \in [2^d]$  denote by  $x_i^0$  the value of  $L_i^0[\ell_i]$ . When the input lists are uniformly random, each  $x_i^0$  is also uniformly random over  $\langle m \rangle$ . For each  $t \in [d]$  and  $i \in [2^{d-t}]$ , we also set up the following notation for the partial sums being considered at each step in the  $k$ -Tree algorithm:

$$x_i^t = x_{2i-1}^{t-1} + x_{2i}^{t-1}$$

We will later override some of these  $x_i^t$ 's by sampling them afresh rather than computing them as above, but for any  $x_i^t$  for which we do not explicitly say otherwise, the above is to be taken to be its definition. For each  $t \in [d]$ , define the following event that captures the set of checks made by the calls to the Merge function in the  $t^{\text{th}}$  iteration of the algorithm:

$$E_t(x_1, \dots, x_{2^{d-t+1}}) \equiv \left( \forall i \in [2^{d-t}] : (x_{2i-1} + x_{2i}) \in \langle mp^t \rangle \right)$$



We can now write the expectation of  $\Lambda_{d,\bar{\ell}}$  as follows:

$$\mathbf{E} [\Lambda_{d,\bar{\ell}}] = \Pr_{x_1^0, \dots, x_{2^d}^0 \leftarrow U_m} \left[ E_1(x_1^0, \dots, x_{2^d}^0) \wedge E_2(x_1^1, \dots, x_{2^{d-1}}^1) \wedge \dots \wedge E_d(x_1^{d-1}, x_2^{d-1}) \right]$$

We define the following sequence of probabilities of subsets of the events in the expression above that we then bound inductively to eventually arrive at a bound for the above expectation. For  $t \in [0, d-1]$ , define the following:

$$\xi_t = \Pr_{x_1^t, \dots, x_{2^{d-t}}^t \leftarrow U_{mp^t}} \left[ E_{t+1}(x_1^t, \dots, x_{2^{d-t}}^t) \wedge E_{t+2}(x_1^{t+1}, \dots, x_{2^{d-t-1}}^{t+1}) \wedge \dots \wedge E_d(x_1^{d-1}, x_2^{d-1}) \right]$$

From the above two expressions, we have:

$$\mathbf{E} [\Lambda_{d,\bar{\ell}}] = \xi_0 \tag{66}$$

The base of our induction is the following bound that follows from Proposition 2.13:

$$\begin{aligned} \xi_{d-1} &= \Pr_{x_1^{d-1}, x_2^{d-1} \leftarrow U_{mp^{d-1}}} \left[ E_d(x_1^{d-1}, x_2^{d-1}) \right] \\ &= \Pr_{x_1, x_2 \leftarrow U_{mp^{d-1}}} \left[ x_1 + x_2 \in \langle mp^d \rangle \right] \\ &\in \left( p - \frac{p^2}{4} \right) \pm \frac{7}{mp^{d-1}} \end{aligned} \tag{67}$$

The inductive step uses the following claim.

**Claim 2.21.** For  $t \in [0, d-2]$ , we have:  $\xi_t \in \xi_{t+1} \cdot \left[ p \cdot (1 \pm p) \left( 1 \pm \frac{30}{mp^{\log k}} \right) \right]^{2^{d-t-1}}$

The proof of Claim 2.21 is identical to that of Claim 2.17, so we leave it out and refer the reader to the earlier proof. If  $d = 1$ , we already have the following from (66,67):

$$\mathbf{E} [\Lambda_{1,\bar{\ell}}] \in \left( p - \frac{p^2}{4} \right) \pm \frac{7}{m} \subseteq p \cdot (1 \pm p) \left( 1 \pm \frac{35}{mp^{\log k}} \right) \tag{68}$$

where in the last bound we use the hypothesis that  $mp^{\log k} > 30$  and  $p \leq 1$ . Suppose  $d \in [2, \log k]$ . Inductively applying Claim 2.21 with  $t$  going from  $(d-2)$  to 0, we get the following:

$$\xi_0 \in \xi_{d-1} \cdot \left[ p \cdot (1 \pm p) \left( 1 \pm \frac{30}{mp^{\log k}} \right) \right]^{\sum_{i=0}^{d-2} 2^{d-i-1}} = \xi_{d-1} \cdot \left[ p \cdot (1 \pm p) \left( 1 \pm \frac{30}{mp^{\log k}} \right) \right]^{2^{d-2}} \tag{69}$$

Putting together (66), (67) and (69), we get:

$$\begin{aligned} \mathbf{E} [\Lambda_{d,\bar{\ell}}] &= \xi_0 \in \left( p - \frac{p^2}{4} \pm \frac{7}{mp^{d-1}} \right) \left( p \cdot (1 \pm p) \left( 1 \pm \frac{30}{mp^{\log k}} \right) \right)^{2^{d-2}} \\ &\subseteq p^{2^{d-1}} \cdot (1 \pm p)^{2^{d-1}} \left( 1 \pm \frac{35}{mp^{\log k}} \right)^{2^{d-1}} \end{aligned} \tag{70}$$

Putting together (65,68,70), and using the shorthand  $\alpha = (1 \pm p) \left(1 \pm \frac{35}{mp^{\log k}}\right)$  we get the following for any  $d \in [\log k]$ :

$$\mathbf{E}[\Lambda_d] = \sum_{\bar{\ell} \in [n]^{2^d}} \mathbf{E}[\Lambda_{d,\bar{\ell}}] \in n^{2^d} p^{2^d-1} \alpha^{2^d-1} = n \cdot (np)^{2^d-1} \cdot \alpha^{2^d-1} \quad (71)$$

Putting together (63,64,71), we get:

$$\begin{aligned} \mathbf{E}[\Lambda] &= \sum_{d \in [0, \log k]} \frac{k}{2^d} \cdot \mathbf{E}[\Lambda_d] \in k \cdot n + \sum_{d \in [\log k]} \frac{k}{2^d} \cdot n \cdot (np)^{2^d-1} \alpha^{2^d-1} \\ &= kn \cdot \sum_{d \in [0, \log k]} \frac{(np)^{2^d-1} \alpha^{2^d-1}}{2^d} \\ &\subseteq kn \cdot \alpha^{k-1} \sum_{d \in [0, \log k]} \frac{(np)^{2^d-1}}{2^d} \end{aligned} \quad (72)$$

which proves the proposition.

## 2.5 $k$ -Tree over $\mathbb{Z}_m$

In this section, we present our analysis of the  $k$ -Tree algorithm (from Figure 5) for the  $k$ -SUM problem over  $\mathbb{Z}_m$  rather than over the integers. For convenience, we will restrict  $m$  to be odd. We will identify the elements of  $\mathbb{Z}_m$  with integers in the range  $\langle m \rangle = \{-\lfloor \frac{m}{2} \rfloor, \dots, \lfloor \frac{m}{2} \rfloor\}$  in the natural manner.

The only modification to the  $k$ -Tree algorithm itself is that the addition operation (as used by the Merge procedure) is addition modulo  $m$  rather than addition over integers. A key observation here is that if  $p < 1/2$ , then this difference in the addition operation is only relevant in the first iteration of the algorithm's loop in step 3 – that is, only when the initial input lists are merged. Thereafter, all the numbers in all the lists have absolute value at most  $mp$ , and thus adding them will never cause a “wrap-around” where the modulus operation with  $m$  actually makes a difference. This lets us adapt our earlier analysis to this case with minimal modification. This is captured by the following theorem. As its proof is almost the same as that of Theorem 2.1, we only provide a sketch that highlights and follows the differences between the proofs.

**Theorem 2.22.** *Consider any  $k, n, m \in \mathbb{N}$ , where  $k \geq 4$  is a power of 2 and  $m > 30^{\log k+1}$  is an odd number. Set  $p = m^{\frac{-1}{\log k+1}}$  and  $c = p \cdot n$ . Consider  $k$  lists  $L_1, \dots, L_k$ , each consisting of  $n$  uniformly random elements from the range  $\langle m \rangle$ . The  $k$ -Tree algorithm (as in Figure 5) over  $\mathbb{Z}_m$  with these parameters, denoted by  $\text{kTree}_m$ , satisfies the following:*

- **Success Probability.** *Its probability of success is bounded as follows:*

$$\frac{1}{c^{-k} + \left(1 + \frac{k}{n}\right)^k} \cdot (1 - 150p)^k \leq \Pr_{L_1, \dots, L_k} \left[ \begin{array}{l} \text{kTree}(L_1, \dots, L_k) \\ \text{outputs } (\ell_1, \dots, \ell_k) \\ \text{such that } \sum_i L_i[\ell_i] = 0 \end{array} \right] \leq c^k \cdot (1 + 37p)^k$$

- **Complexity.** *Its expected complexity is bounded as follows:*

$$\mathbf{E}_{L_1, \dots, L_k} \left[ \begin{array}{l} \text{Total size of all lists} \\ \text{involved in} \\ \text{kTree}(L_1, \dots, L_k) \end{array} \right] \in kn \cdot \left( 1 + \sum_{d \in [\log k]} \frac{c^{2^d-1}}{2^d} \right) \cdot (1 \pm 37p)^{k-1}$$

*Proof Sketch of Theorem 2.22.* The proofs of Propositions 2.3 to 2.5 (which together imply the bounds in Theorem 2.1) are built solely on top of Propositions 2.13 to 2.16, following by appropriate manipulations of the bounds provided by the latter. As observed above, looking at the iterations in step 3 of the  $k$ -Tree algorithm (as in Figure 5), all iterations from  $d = 2$  onwards are identical for the algorithm over integers and over  $\mathbb{Z}_m$ , as there is no wrap-around in the addition operations involved, and that part of the analysis from the proof of Theorem 2.1 can be used as is.

For the first iteration, the primary differentiating factor in the analysis of the algorithms in the two cases is that addition modulo  $m$  of two uniformly random numbers from  $\langle m \rangle$  is slightly more likely to lead to an element in  $\langle mp \rangle$  than addition of such numbers over the integers. So the statements of Propositions 2.13 to 2.16 might no longer be true. To redeem the proof, we instead provide the following alternatives to each of the propositions that may be used instead in the analysis of the first iteration. It may be verified that the bounds they provide suffice to make the proofs of Propositions 2.3 to 2.5 continue to work even in the case of addition modulo  $m$ , which can then be used to prove Theorem 2.22 in the same manner as Theorem 2.1.

*Replacing Proposition 2.13:* Note that the sum of two uniformly random elements of  $\mathbb{Z}_m$  is also a uniformly random element of  $\mathbb{Z}_m$ . This lets us replace the use of Proposition 2.13 in the analysis of the first iteration with the following statement for any  $m$  and  $p \in [0, 1]$ :

$$\Pr_{x,y \leftarrow \langle m \rangle} [(x+y) \pmod m \in \langle mp \rangle] = \frac{|\langle mp \rangle|}{|\langle m \rangle|} = \frac{2 \lfloor mp/2 \rfloor + 1}{m} \in \left( p \pm \frac{1}{m} \right) \quad (73)$$

*Replacing Proposition 2.14:* As noted above, the sum of two uniformly random elements of  $\mathbb{Z}_m$  is also a uniformly random element of  $\mathbb{Z}_m$ , and conditioning on this sum being in  $\langle mp \rangle$  results in a uniform distribution over  $\langle mp \rangle$ . Thus the MR distance between this conditioned sum distribution and the uniform distribution over  $\langle mp \rangle$  is actually *equal to 1*, which also happens to satisfy the bound in Proposition 2.14.

*Replacing Proposition 2.15:* If  $w$ ,  $x$ , and  $y$  are sampled uniformly at random from  $\mathbb{Z}_m$ , the sums  $(w+x)$  and  $(w+y)$  are, in fact, independent and uniformly random over  $\mathbb{Z}_m$ . Thus, we have the following using (73):

$$\begin{aligned} & \Pr_{w,x,y \leftarrow \langle m \rangle} [(w+x) \pmod m \in \langle mp \rangle \wedge (w+y) \pmod m \in \langle mp \rangle] \\ &= \Pr_{w,x \leftarrow \langle m \rangle} [(w+x) \pmod m \in \langle mp \rangle] \cdot \Pr_{w,y \leftarrow \langle m \rangle} [(w+y) \pmod m \in \langle mp \rangle] \\ &\leq p^2 \cdot \left( 1 + \frac{1}{mp} \right)^2 \end{aligned}$$

which also satisfies the bound in Proposition 2.15.

*Replacing Proposition 2.16:* Following the arguments above, the values  $(w+x)$  and  $(w+y)$  conditioned on both being in  $\langle mp \rangle$  are also independent and uniformly distributed over  $\langle mp \rangle$ . Thus the MR distance between this joint distribution and the uniform distribution over  $\langle mp \rangle \times \langle mp \rangle$  is again equal to 1, which also satisfies the bound in Proposition 2.16.  $\square$

### 3 Computed Bounds

The theoretical analysis presented in the previous sections provides asymptotically tight bounds for the  $k$ -Tree algorithm. However, for some practical parameter values that modern computers

can handle, these bounds may not be as precise as one might desire due to the approximations used in the proofs. While these approximations do not affect the asymptotic tightness, they can have significant impact on the bounds for the parameter ranges we wish to evaluate empirically.

To address this limitation, we have implemented a set of computer programs that compute these bounds without the approximations used in the theoretical analysis. This approach allows us to obtain much tighter bounds for smaller parameter values, which are particularly relevant for practical applications and our experimental evaluations.

In this section, we present the computational methods used to calculate these precise bounds. We provide pseudo-code for each key function, along with corresponding theorem statements that relate these computations to our theoretical results and the  $k$ -Tree algorithm. By removing the approximations used in the manual proofs, these computational methods provide a bridge between our asymptotic analysis and the actual performance of the algorithm. The bounds computed by these programs serve two crucial purposes:

- Offer more accurate predictions of the algorithm’s behavior for practical parameter ranges.
- Provide a means to validate our theoretical predictions against experimental measurements.

In the following subsections, we will detail each computational method, its relationship to the theoretical claims, propositions and theorems, and how it contributes to our understanding of the  $k$ -Tree algorithm’s performance.

## Notations

To ensure consistency throughout this section, we use the following notations and definitions:

- $k$ : The number of input lists (always a power of 2).
- $n$ : The size of each input list.
- $m$ : The range from which the numbers in the lists are drawn.
- $p$ : The  $p$  parameter in the statement of Theorem 1, computed as  $p = m^{-\frac{1}{\log k+1}}$ .

## 3.1 Algorithms

Here we present the algorithms that implement our approach to proving bounds on the  $k$ -Tree algorithm. We start with sub-routines that compute basic probabilities and distances before proceeding to the ones that actually implement the proof using them. The final bounds on the success probability are computed by the function `PROBBOUNDS` (Algorithm 11), and the bounds on the complexity by `SIZEBOUNDS` (Algorithm 13). In each case, we state a claim about the behavior of the sub-routine, and if applicable refer to the corresponding analogue in Section 2 in this statement. We provide proof sketches for these claims except in cases where the implementation departs significantly from the proof in Section 2, in which case we provide more detailed proofs.

**Claim 3.1** (Claim 2.12). *Given parameters  $s$  and an integer  $z \in [-s, s]$  as inputs, `PROBSUMTOZ` (Algorithm 1) outputs the exact probability  $\Pr_{x,y \leftarrow U_s}[x+y = z]$ , where  $U_s$  is the uniform distribution over  $\langle s \rangle$ . And the running time of the algorithm is  $O(\text{polylog}(s))$ .*

---

**Algorithm 1** PROBSUMTOZ

---

**Input:**  $s, z$  ▷ Range size and target sum

**Output:** Probability of the sum of two random variables in  $\langle s \rangle$  equaling  $z$

1:  $d \leftarrow 2 \cdot \lfloor s/2 \rfloor + 1$

2: **return**  $\frac{1}{d} - \frac{|z|}{d^2}$

---

*Proof Sketch.* The computation is exactly the result of the proof for Claim 2.12 up to (27):

$$\Pr_{x,y \leftarrow U_s} [x + y = z] = \frac{1}{(2 \lfloor s/2 \rfloor + 1)} - \frac{|z|}{(2 \lfloor s/2 \rfloor + 1)^2}$$

□

---

**Algorithm 2** PROBSUMINRANGE

---

**Input:**  $s, p$  ▷ Range size and  $p$

**Output:** Probability of sum being in specified range

1:  $t \leftarrow \lfloor (sp)/2 \rfloor$

2:  $d \leftarrow 2 \cdot \lfloor s/2 \rfloor + 1$

3:  $r \leftarrow \frac{2t+1}{d} - \frac{t^2+t}{d^2}$

4: **return**  $r$

---

**Proposition 3.2** (Proposition 2.13). *For any  $s$  and  $p \in [0, 1]$  as inputs, PROBSUMINRANGE (Algorithm 2) computes the probability  $\Pr_{x,y \leftarrow U_s} [x+y \in \langle sp \rangle]$  with running time being  $O(\text{polylog}(s))$ .*

*Proof Sketch.* The function calculates this probability by summing the exact probabilities of each possible sum within the specified range as in Proposition 2.13. We simplify it a bit by directly calculating the closed-form sum of  $\Pr_{x,y \leftarrow U_s} [x + y = z]$  over  $z$  instead of iteratively calling Algorithm 1:

$$\Pr_{x,y \leftarrow U_s} [x + y \in \langle sp \rangle] = \sum_{z=-\lfloor sp/2 \rfloor}^{\lfloor sp/2 \rfloor} \Pr_{x,y \leftarrow U_s} [x + y = z] = \frac{(2 \lfloor sp/2 \rfloor + 1)}{(2 \lfloor s/2 \rfloor + 1)} - \frac{\lfloor sp/2 \rfloor (\lfloor sp/2 \rfloor + 1)}{(2 \lfloor s/2 \rfloor + 1)^2}.$$

□

---

**Algorithm 3** MRDISTFROMUNIF

---

**Input:**  $s, p$  ▷ Range size and probability

**Output:** Max-ratio distance from uniform distribution

1:  $s' \leftarrow sp$

2:  $\alpha \leftarrow \text{PROBSUMTOZ}(s, 0) / \text{PROBSUMINRANGE}(s, p)$

3:  $\beta \leftarrow \text{PROBSUMTOZ}(s, \lfloor s'/2 \rfloor) / \text{PROBSUMINRANGE}(s, p)$

4:  $u \leftarrow \frac{1}{2 \cdot \lfloor s'/2 \rfloor + 1}$

5: **return**  $\max(\alpha/u, u/\beta)$

---

**Proposition 3.3** (Proposition 2.14). *For any  $s$  and  $p \in [0, 1]$  as inputs, MRDISTFROMUNIF (Algorithm 3) computes  $\Delta_{MR}(U_{sp}, 2 \cdot U_s|_{\langle sp \rangle})$  – the max-ratio distance between the uniform distribution over  $\langle sp \rangle$  and the distribution of the sum of two uniform random variables in  $\langle s \rangle$  conditioned on being in  $\langle sp \rangle$ . And the running time of the algorithm is  $O(\text{polylog}(s))$ .*

*Proof Sketch.* The computation follows the exact definition of Max-Ratio Distance:

$$\Delta_{MR}(U_{sp}, 2 \cdot U_s|_{\langle sp \rangle}) = \max \left( \frac{\max_{z \in \langle sp \rangle} 2 \cdot U_s|_{\langle sp \rangle}(z)}{U_{sp}}, \frac{U_{sp}}{\min_{z \in \langle sp \rangle} 2 \cdot U_s|_{\langle sp \rangle}(z)} \right).$$

And one can verify that  $2 \cdot U_s|_{\langle sp \rangle}(z)$  is maximized when  $z = 0$  and minimized when  $z = \pm \lfloor sp \rfloor$ .  $\square$

---

**Algorithm 4** PROBSUMWITHTWO RV IN RANGE

---

**Input:**  $s, p$

$\triangleright$  Range size and probability

**Output:** Probability of summing one r.v. with two other r.v.'s being in specified range respectively

**Require:**  $0 \leq p \leq 1$

- 1:  $\alpha = 2 \cdot (\lfloor s/2 \rfloor - \lfloor sp/2 \rfloor) \cdot \frac{|\langle sp \rangle|^2}{|\langle s \rangle|^3}$
  - 2:  $x = |\langle sp \rangle| - 1$
  - 3:  $y = |\langle sp \rangle| - 1 - \lfloor sp/2 \rfloor$
  - 4:  $\beta = \frac{2}{|\langle s \rangle|^3} (x(x+1)(2x+1) - y(y+1)(2y+1)) / 6$
  - 5: **return**  $\alpha + \beta$
- 

**Proposition 3.4** (Proposition 2.15). *For any  $s$  and  $p \in [0, 1]$  as inputs, PROBSUMWITHTWO RV IN RANGE (Algorithm 4) computes the probability  $\Pr_{w,x,y \leftarrow U_s} [w+x \in \langle sp \rangle \wedge w+y \in \langle sp \rangle]$ . And the running time of the algorithm is  $O(\text{polylog}(s))$ .*

*Proof Sketch.* The algorithm computes the probability following the first equality of (28):

$$\Pr_{w,x,y \leftarrow U_s} [w+x \in \langle sp \rangle \wedge w+y \in \langle sp \rangle] = \sum_{w=-\lfloor sp/2 \rfloor}^{\lfloor sp/2 \rfloor} U_s(w) \cdot \Pr_{x,y \leftarrow U_s} [w+x \in \langle sp \rangle \wedge w+y \in \langle sp \rangle].$$

The calculation considers two cases:

1. When  $w \in [-\lfloor s/2 \rfloor + \lfloor sp/2 \rfloor, \lfloor s/2 \rfloor - \lfloor sp/2 \rfloor]$ , each term on the RHS equals

$$U_s(w) \cdot \Pr_{x,y \leftarrow U_s} [w+x \in \langle sp \rangle \wedge w+y \in \langle sp \rangle] = \frac{|\langle sp \rangle|^2}{|\langle s \rangle|^3},$$

and  $\alpha$  is summing this value over all such  $w$ .

2. When  $w \in [-\lfloor s/2 \rfloor, -\lfloor s/2 \rfloor + \lfloor sp/2 \rfloor - 1]$  or  $w \in [\lfloor s/2 \rfloor - \lfloor sp/2 \rfloor + 1, \lfloor s/2 \rfloor]$ , let  $i = |w| - \lfloor s/2 \rfloor + \lfloor sp/2 \rfloor$ . Then the probability on the RHS is

$$\Pr_{x,y \leftarrow U_s} [w+x \in \langle sp \rangle \wedge w+y \in \langle sp \rangle] = \left( \frac{|\langle sp \rangle| - i}{|\langle s \rangle|} \right)^2.$$

Summing all these values over  $w$  results in a sum of some squares, which is computed as  $\beta$  in the algorithm.

---

**Algorithm 5** MRDISTFROMPAIRUNIF

---

**Input:**  $s, p$ 

▷ Range size and probability

**Output:** Max-ratio distance from uniform distribution for pairs

- 1:  $\alpha \leftarrow \frac{1}{(|\langle s \rangle|)^2 \cdot \text{PROBSUMWITHTWO RV IN RANGE}(s, p)}$
  - 2:  $\beta \leftarrow \frac{2(\lfloor s/2 \rfloor - \lfloor sp/2 \rfloor)}{(|\langle s \rangle|)^3 \cdot \text{PROBSUMWITHTWO RV IN RANGE}(s, p)}$
  - 3:  $u \leftarrow \frac{1}{(|\langle sp \rangle|)^2}$
  - 4: **return**  $\max(\alpha/u, u/\beta)$
- 

□

**Proposition 3.5** (Proposition 2.16). *For any  $s$  and  $p \in [0, 0.5]$  as inputs, MRDISTFROMPAIRUNIF (Algorithm 5) computes  $\Delta_{\text{MR}}(U_{sp} \otimes U_{sp}, D)$  as defined in Proposition 2.16. And the running time of the algorithm is  $O(\text{polylog}(s))$ .*

*Proof Sketch.* The computation follows the definition of Max-Ratio Distance:

$$\Delta_{\text{MR}}(U_{sp} \otimes U_{sp}, D) = \max \left( \frac{\max_{(z_1, z_2) \in \langle sp \rangle^2} D(z_1, z_2)}{U_{sp} \otimes U_{sp}}, \frac{U_{sp} \otimes U_{sp}}{\min_{(z_1, z_2) \in \langle sp \rangle^2} D(z_1, z_2)} \right).$$

The algorithm computes:

1. The maximum probability  $\max_{(z_1, z_2) \in \langle sp \rangle^2} D(z_1, z_2)$  ( $\alpha$ ) occurs when  $(z_1, z_2) = (0, 0)$ .
2. The minimum probability  $\min_{(z_1, z_2) \in \langle sp \rangle^2} D(z_1, z_2)$  ( $\beta$ ) occurs when
$$(z_1, z_2) = (-\lfloor sp/2 \rfloor, \lfloor sp/2 \rfloor) \text{ or } (\lfloor sp/2 \rfloor, -\lfloor sp/2 \rfloor).$$
3. The uniform probability ( $u$ ) over  $\langle sp \rangle^2$ .

It then calculates the results by comparing these terms. The use of PROBSUMWITHTWO RV IN RANGE in the denominator of  $\alpha$  and  $\beta$  accounts for the conditioning on the sums being in  $\langle sp \rangle$ . □

---

**Algorithm 6** FIRSTMOMENTINDUCTFACTORS

---

**Input:**  $m, k, p, d$ 

▷ Input parameters

**Output:** UB, LB

▷ Upper and lower bounds

- 1:  $s \leftarrow m \cdot p^d$
  - 2:  $\alpha \leftarrow (\text{PROBSUMINRANGE}(s, p))^{\frac{k}{2^{d+1}}}$
  - 3:  $\beta \leftarrow (\text{MRDISTFROMUNIF}(s, p))^{\frac{k}{2^{d+1}}}$
  - 4:  $\text{UB} \leftarrow \alpha \cdot \beta$
  - 5:  $\text{LB} \leftarrow \alpha/\beta$
  - 6: **return** UB, LB
-

**Claim 3.6** (Claim 2.17). For any  $m$  and  $k$  being powers of 2,  $p \in [0, 1]$  and  $d \in [0, \log k - 1]$  as inputs, `FIRSTMOMENTINDUCTFACTORS` (Algorithm 6) computes factors  $UB = \alpha \cdot \beta$  and  $LB = \alpha / \beta$  such that  $\zeta_{d+1} \cdot LB \leq \zeta_d \leq \zeta_{d+1} \cdot UB$ , where:

$$\alpha = \Pr_{x_1^d, \dots, x_{k/2^d}^d \leftarrow U_{mp^d}} \left[ \forall i \in \left[ \frac{k}{2^{d+1}} \right] : (x_{2i-1}^d + x_{2i}^d) \in \langle mp^{d+1} \rangle \right],$$

$$\beta = \Delta_{MR}(U_{mp^{d+1}}, 2 \cdot U_{mp^d} |_{\langle mp^{d+1} \rangle})^{\frac{k}{2^{d+1}}},$$

and  $\zeta_d$  is as defined in the proof of Proposition 2.3. And the algorithm runs in  $O(\text{polylog}(m, k))$ .

*Proof Sketch.* Algorithm 6 computes the exact values of the factors described in the proof of Proposition 2.3. It calls `PROBSUMINRANGE` to compute  $\alpha$  as in (36), and `MRDISTFROMUNIF` to compute  $\beta$  as in (37). Then it computes the upper and lower bound factors accordingly.  $\square$

---

**Algorithm 7** `FIRSTMOMENTBOUNDS`

---

**Input:**  $m, k, n, p$

$\triangleright$  Input parameters

**Output:**  $UB, LB$

$\triangleright$  Upper and lower bounds

- 1:  $UB \leftarrow n^k \cdot \frac{1}{2^{\lfloor mp^{\log k} \rfloor - 1}}$
  - 2:  $LB \leftarrow UB$
  - 3: **for**  $d \leftarrow 0$  to  $\log k - 1$  **do**
  - 4:    $\alpha_u, \alpha_l \leftarrow \text{FIRSTMOMENTINDUCTFACTORS}(m, k, p, d)$
  - 5:    $UB \leftarrow \alpha_u \cdot UB$
  - 6:    $LB \leftarrow \alpha_l \cdot LB$
  - 7: **return**  $UB, LB$
- 

**Proposition 3.7** (Proposition 2.3). Given parameters  $m, k, n$  and  $p$  satisfying the hypotheses of Proposition 2.3 as inputs, `FIRSTMOMENTBOUNDS` (Algorithm 7) computes upper and lower bounds  $UB$  and  $LB$  such that:

$$LB \leq \mathbb{E}[C] \leq UB,$$

where  $C$  is the number of  $k$ -tuples  $(l_1, \dots, l_k) \in [n]^k$  such that  $\sum_{i=1}^k L_i[l_i] = 0$ , and  $L_1, \dots, L_k$  are the uniformly random input lists. And  $UB, LB$  are at least as tight as the bounds in Proposition 2.3. And the running time of the algorithm is  $O(\text{polylog}(m, k, n))$ .

*Proof Sketch.* The algorithm first computes the base values of the upper and lower bounds as in (31) and multiplies them with the scaling factor  $n^k$  as in (34). Then it iteratively computes the induction factors using `FIRSTMOMENTINDUCTFACTORS` as in (32) and (33). As the computations in the subroutines have no approximations involved, the bounds computed by Algorithm 7 are at least as tight as the ones in Proposition 2.3.  $\square$

**Proposition 3.8** (Proposition 2.4). Given parameters  $m, k, n$  and  $p$  satisfying the hypotheses of Proposition 2.4 as inputs, `SECONDMOMENTUB` (Algorithm 8) computes an upper bound on the second moment of the number of solutions, such that:

$$\mathbb{E}[C^2] \leq UB$$



---

**Algorithm 8** SECONDMOMENTUB

---

**Input:**  $m, k, n, p$ 

▷ Input parameters

**Output:** Recursive result of the second moment bound

```
1: if  $k = 1$  then
2:    $u \leftarrow \frac{1}{2 \cdot \lfloor m \rfloor + 1}$ 
3:   return  $(n \cdot u) \cdot (n \cdot u + 1)$ 
4: else
5:    $m' = m \cdot p$ 
6:    $k' = k/2$ 
7:    $n' = ((\text{PROBUNIFXMRDIST}(m, p))^2 \cdot n^4 + 2 \cdot (\text{PROBUNIFXMRDISTPAIR}(m, p)) \cdot n^3)^{\frac{1}{2}}$ 
8:   return SECONDMOMENTUB( $m', k', n', p$ )
```

---

---

**Algorithm 9** PROBUNIFXMRDIST

---

**Input:**  $m, p$ 

▷ Input parameters

**Output:** Product of uniform probability and max-ratio distance

```
1: return PROBSUMINRANGE( $m, p$ ) · MRDISTFROMUNIF( $m, p$ )
```

---

---

**Algorithm 10** PROBUNIFXMRDISTPAIR

---

**Input:**  $m, p$ 

▷ Input parameters

**Output:** Product of uniform probability and max-ratio distance for pairs

```
1: return PROBSUMWITHTWOVRINRANGE( $m, p$ ) · MRDISTFROMPAIRUNIF( $m, p$ )
```

---

and UB is at least as tight as the bound in Proposition 2.4. And the running time of the algorithm is  $O(\text{polylog}(m, k, n))$ .

*Proof.* Algorithm 8 employs a recursive variant of the calculation presented in the proof of Proposition 2.4 to compute the upper bound on the second moment. For consistency, we adhere to the notations defined in the proof of Proposition 2.4.

We begin by refining several expressions from the proof of Proposition 2.4 to enable recursive calculation. Recall the expression we want to calculate in (39):

$$\mathbf{E}[C^2] = \sum_{\bar{\ell}, \bar{\ell}' \in [n]^k} \mathbf{E}[C_{\bar{\ell}} \cdot C_{\bar{\ell}'}] = \sum_{s \in \{0,1\}^k} \sum_{\bar{\ell}, \bar{\ell}': \delta(\bar{\ell}, \bar{\ell}') = s} \mathbf{E}[C_{\bar{\ell}} \cdot C_{\bar{\ell}'}].$$

We continue to use the notations  $(s^1, \dots, s^{\log k}) \leftarrow \text{tex}(s)$  defined before Claim 2.19. Unless overridden in the context, we write  $s^d$  by assuming it is one of the corresponding outputs generated by  $\text{tex}(s^0)$  for some  $s^0$  either implicitly or explicitly. We further define  $ex: \{0,1\}^{2^*} \rightarrow \{0,1\}^{2^*/2}$  on input  $s \in \{0,1\}^{2^*}$  that is of length power of 2, outputs  $s' \in \{0,1\}^{2^*/2}$  with half the length, where its  $i^{\text{th}}$  bit is defined as follows, for  $i \in [k/2^d]$ :

$$s'_i = \max(s_{2i-1}, s_{2i}).$$

Then we refine (48) and provide its extended notation  $\chi_d(s)$  with  $s \in \{0,1\}^{\frac{k}{2^d}}$  as input, where for  $\chi_0$ , the input  $s$  corresponds to  $\delta(\bar{\ell}, \bar{\ell}')$ :

$$\mathbf{E}[C_{\bar{\ell}} \cdot C_{\bar{\ell}'}] = \chi_0(\delta(\bar{\ell}, \bar{\ell}')).$$

The inclusion of input  $s$  is by revisiting the definition of  $\chi_d$  in the proof of Proposition 2.4 and verifying that in each step of the recursive calculations, the corresponding  $s^d$  is sufficient and necessary to determine the value of  $\chi_d$ . Putting this back to (39), we have the following expression:

$$\mathbf{E} [C^2] = \sum_{s \in \{0,1\}^k} \sum_{\bar{\ell}, \bar{\ell}': \delta(\bar{\ell}, \bar{\ell}')=s} \chi_0(s). \quad (74)$$

Then we recall the inequality stated by Claim 2.20 and its precise form:

$$\begin{aligned} \chi_d(s^d) \leq & \chi_{d+1}(s^{d+1}) \cdot (\mathbf{A}_{mp^d,p} \mathbf{B}_{mp^d,p})^{\sum_{i \in \lfloor \frac{k}{2^{d+1}} \rfloor} \mathbb{1}[s_{2i-1}^d = s_{2i}^d = 0]} \\ & \cdot (\mathbf{A}_{mp^d,p} \mathbf{B}_{mp^d,p})^{2 \cdot \sum_{i \in \lfloor \frac{k}{2^{d+1}} \rfloor} \mathbb{1}[s_{2i-1}^d = s_{2i}^d = 1]} \cdot (\mathbf{C}_{mp^d,p} \mathbf{D}_{mp^d,p})^{\sum_{i \in \lfloor \frac{k}{2^{d+1}} \rfloor} \mathbb{1}[s_{2i-1}^d \neq s_{2i}^d]}, \end{aligned} \quad (75)$$

where  $s^{d+1} = \text{ex}(s^d)$  and the notations  $\mathbf{A}_{mp^d,p}$ ,  $\mathbf{B}_{mp^d,p}$ ,  $\mathbf{C}_{mp^d,p}$ , and  $\mathbf{D}_{mp^d,p}$  are defined as the shorthands for the following values:

$$\begin{aligned} \mathbf{A}_{mp^d,p} &= \Pr_{x,y \leftarrow U_{mp^d}} \left[ (x+y) \in \langle mp^{d+1} \rangle \right], \\ \mathbf{B}_{mp^d,p} &= \Delta_{\text{MR}} \left( U_{mp^{d+1}}, 2 \cdot U_{mp^d} |_{\langle mp^{d+1} \rangle} \right), \\ \mathbf{C}_{mp^d,p} &= \Pr_{w,x,y \leftarrow U_{mp^d}} \left[ (w+x) \in \langle mp^{d+1} \rangle \wedge (w+y) \in \langle mp^{d+1} \rangle \right], \\ \mathbf{D}_{mp^d,p} &= \Delta_{\text{MR}}(D^0, D^1) \text{ in (61)}. \end{aligned}$$

Recall that in the proof of Claim 2.20, we consider three cases regarding the values of  $s_{2i-1}^d$  and  $s_{2i}^d$ :  $s_{2i-1}^d = s_{2i}^d = 0$ ,  $s_{2i-1}^d = s_{2i}^d = 1$  and  $s_{2i-1}^d \neq s_{2i}^d$ . And in (75), the base terms corresponding to the three cases are by putting together (53) and (59), (54) and (60), (55) and (61) respectively. For simplification, we define the following functions:

$$\begin{aligned} \gamma_0(s) &= \sum_{i \in \lfloor |s|/2 \rfloor} \mathbb{1}[s_{2i-1} = s_{2i} = 0], \\ \gamma_1(s) &= \sum_{i \in \lfloor |s|/2 \rfloor} \mathbb{1}[s_{2i-1} = s_{2i} = 1], \\ \gamma_{\neq}(s) &= \sum_{i \in \lfloor |s|/2 \rfloor} \mathbb{1}[s_{2i-1} \neq s_{2i}]. \end{aligned}$$

We would then rewrite the recursive upper bound (75) in terms of the following function  $g$ :

$$g(s, m, p) = \begin{cases} \left( \Pr_{x \leftarrow U_{mp^{\log k}}} [x = 0] \right)^{1+wt(s)} & \text{if } \text{length}(s) = 1, \\ g(\text{ex}(s), m \cdot p, p) \cdot \delta_{s,m,p} & \text{otherwise,} \end{cases} \quad (76)$$

where  $\delta_{s,m,p} = (\mathbf{A}_{m,p} \mathbf{B}_{m,p})^{\gamma_0(s)} \cdot (\mathbf{A}_{m,p} \mathbf{B}_{m,p})^{2 \cdot \gamma_1(s)} \cdot (\mathbf{C}_{m,p} \mathbf{D}_{m,p})^{\gamma_{\neq}(s)}$ .

It is easy to see that for  $\chi_0(s^0)$ , the  $\delta_{s^0,m,p}$  in  $g(s^0, m, p)$  is exactly the RHS of (75) with  $d = 0$ . And in the subsequent  $g(s^1, m \cdot p, p)$  (i.e.,  $\chi_1(\text{ex}(s^0))$ ) where  $s^1 \leftarrow \text{ex}(s^0)$ ,  $\delta_{s^1,mp,p}$  equals

$$(\mathbf{A}_{mp,p} \mathbf{B}_{mp,p})^{\gamma_0(s^1)} \cdot (\mathbf{A}_{mp,p} \mathbf{B}_{mp,p})^{2 \cdot \gamma_1(s^1)} \cdot (\mathbf{C}_{mp,p} \mathbf{D}_{mp,p})^{\gamma_{\neq}(s^1)}.$$

It is evident that the calculations align with (75) in each inductive step. The base case when  $\text{length}(s) = 1$  is derived from (49), where the value of  $\chi_{\log k}(s)$  can be computed precisely as

$$\Pr_{x \leftarrow U_{m p^{\log k}}} [x = 0] \cdot \Pr_{y \leftarrow U_{m p^{\log k}}} [y = 0]$$

if  $s = 1$  and

$$\Pr_{y \leftarrow U_{m p^{\log k}}} [y = 0]$$

otherwise.

Now we substitute  $\chi_0(s^0)$  in (74) with the recursive upper bound  $g(s^0, m, p)$  to obtain

$$\begin{aligned} \mathbf{E}[C^2] &\leq \sum_{s^0 \in \{0,1\}^k} \sum_{\bar{\ell}, \bar{\ell}': \delta(\bar{\ell}, \bar{\ell}') = s^0} g(s^0, m, p) \\ &= \sum_{s^0 \in \{0,1\}^k} n^{(k - \text{wt}(s^0))} \cdot (n(n-1))^{\text{wt}(s^0)} \cdot g(s^0, m, p) \\ &\leq \sum_{s^0 \in \{0,1\}^k} n^{(k - \text{wt}(s^0))} \cdot (n^2)^{\text{wt}(s^0)} \cdot g(s^0, m, p). \end{aligned} \quad (77)$$

To conclude our computation of the second moment upper bound recursively, we define a function  $f(m, k, n, p)$  to capture (77) more generally:

$$f(m, k, n, p) = \sum_{s \in \{0,1\}^k} n^{(k - \text{wt}(s))} \cdot (n^2)^{\text{wt}(s)} \cdot g(s, m, p). \quad (78)$$

Corresponding to the base case of  $g$ , the base case of  $f$  is when  $k = 1$ :

$$\begin{aligned} f(m, 1, n, p) &= n \cdot g(0, m, p) + n^2 \cdot g(1, m, p) \\ &= \frac{n}{2 \cdot \lfloor m/2 \rfloor + 1} \left( \frac{n}{2 \cdot \lfloor m/2 \rfloor + 1} + 1 \right). \end{aligned}$$

We now rewrite (77) in a slightly different form:

$$\begin{aligned} f(m, k, n, p) &= \sum_{s \in \{0,1\}^k} \sum_{\bar{\ell}, \bar{\ell}' \in [n]^k: \delta(\bar{\ell}, \bar{\ell}') = s} g(s, m, p) \\ &= \sum_{s' \in \{0,1\}^{\frac{k}{2}}} \sum_{s: \text{ex}(s) = s'} \sum_{\bar{\ell}, \bar{\ell}': \delta(\bar{\ell}, \bar{\ell}') = s} g(s, m, p) \\ &= \sum_{s' \in \{0,1\}^{\frac{k}{2}}} \sum_{s: \text{ex}(s) = s'} \sum_{\bar{\ell}, \bar{\ell}': \delta(\bar{\ell}, \bar{\ell}') = s} g(s', m, p) \cdot \delta_{s, m, p} \\ &= \sum_{s' \in \{0,1\}^{\frac{k}{2}}} g(s', m, p) \cdot \sum_{s: \text{ex}(s) = s'} \sum_{\bar{\ell}, \bar{\ell}': \delta(\bar{\ell}, \bar{\ell}') = s} \delta_{s, m, p} \end{aligned} \quad (79)$$

(80)

In the sum  $\sum_{s:ex(s)=s'} \sum_{\bar{\ell}, \bar{\ell}': \delta(\bar{\ell}, \bar{\ell}')=s} \delta_{s,m,p}$ , the value of  $\delta_{s,m,p}$  is determined by the values of  $s$ ,  $m$  and  $p$ . We then count the total number of  $\bar{\ell}$  and  $\bar{\ell}'$  that satisfy  $\delta(\bar{\ell}, \bar{\ell}') = s$  such that  $ex(s) = s'$  then multiply the result by  $\delta_{s,m,p}$ :

$$\sum_{s:ex(s)=s'} \delta_{s,m,p} \sum_{\bar{\ell}, \bar{\ell}': \delta(\bar{\ell}, \bar{\ell}')=s} 1 = \sum_{s:ex(s)=s'} \delta_{s,m,p} \cdot (n^2)^{\gamma_0(s)} (n^2(n-1)^2)^{\gamma_1(s)} (n^2(n-1))^{\gamma_{\neq}(s)} \quad (81)$$

$$\leq \sum_{s:ex(s)=s'} (\mathbf{A}_{m,p} \mathbf{B}_{m,p} \cdot n^2)^{\gamma_0(s)} \cdot (\mathbf{A}_{m,p} \mathbf{B}_{m,p} \cdot n^2)^{2 \cdot \gamma_1(s)} \cdot (\mathbf{C}_{m,p} \mathbf{D}_{m,p} \cdot n^3)^{\gamma_{\neq}(s)} \quad (82)$$

$$= \sum_{i=0}^{wt(s')} \binom{wt(s')}{i} \cdot (\mathbf{A}_{m,p} \mathbf{B}_{m,p} \cdot n^2)^{k/2 - wt(s')} \cdot (\mathbf{A}_{m,p} \mathbf{B}_{m,p} \cdot n^2)^{2 \cdot i} \cdot (2 \cdot \mathbf{C}_{m,p} \mathbf{D}_{m,p} \cdot n^3)^{wt(s') - i} \quad (83)$$

$$= (\mathbf{A}_{m,p} \mathbf{B}_{m,p} \cdot n^2)^{k/2 - wt(s')} \cdot ((\mathbf{A}_{m,p} \mathbf{B}_{m,p} \cdot n^2)^2 + 2 \cdot \mathbf{C}_{m,p} \mathbf{D}_{m,p} \cdot n^3)^{wt(s')}.$$

The first equality is by counting the number of  $\bar{\ell}$  and  $\bar{\ell}'$  that satisfy  $\delta(\bar{\ell}, \bar{\ell}') = s$  for a fixed  $s$  and the following inequality is by upper bounding  $n-1$  by  $n$ . The equality in (83) is by counting the number of possible  $s$  given  $s'$ , where  $s'_i = 0$  implies  $s_{2i-1} = s_{2i} = 0$  ( $\gamma_0(s)$ ) and  $s'_i = 1$  implies  $s_{2i-1} = s_{2i} = 1$  ( $\gamma_1(s)$ ) or  $s_{2i-1} \neq s_{2i}$  ( $\gamma_{\neq}(s)$ ). Then we denote by  $i$  the number of case  $s_{2i-1} = s_{2i} = 1$  and sum over all possible  $i \in [0, wt(s')]$ . Finally, the last equality is by summing the binomial coefficients with the last two terms then applying Binomial Theorem. Combining the result with (79), we have:

$$f(m, k, n, p) \leq \sum_{s' \in \{0,1\}^{\frac{k}{2}}} g(s', mp, p) \cdot (\mathbf{A}_{m,p} \mathbf{B}_{m,p} \cdot n^2)^{k/2 - wt(s')} \cdot ((\mathbf{A}_{m,p} \mathbf{B}_{m,p} \cdot n^2)^2 + 2 \cdot \mathbf{C}_{m,p} \mathbf{D}_{m,p} \cdot n^3)^{wt(s')} \leq f(mp, k/2, ((\mathbf{A}_{m,p} \mathbf{B}_{m,p} \cdot n^2)^2 + 2 \cdot \mathbf{C}_{m,p} \mathbf{D}_{m,p} \cdot n^3)^{1/2}, p). \quad (84)$$

Finally, we note that Algorithm 9 and Algorithm 10 computes the products of  $\mathbf{A}_{m,p}$  and  $\mathbf{B}_{m,p}$ , and  $\mathbf{C}_{m,p}$  and  $\mathbf{D}_{m,p}$ . Combining all these elements, the function  $f$  precisely follows the procedure defined in SECONDMOMENTUB (Algorithm 8). Since in each recursive call, the total running time of all operations is  $O(\text{polylog}(m, k, n))$ , the running time of the algorithm is determined by multiplying the number of total recursive calls, which gives  $O(\log k \cdot \text{polylog}(m, k, n)) = O(\text{polylog}(m, k, n))$ .  $\square$

**Theorem 3.9.** *Given parameters  $m, k, n, p$  satisfying the hypotheses in Theorem 2.1 as inputs, PROBBOUNDS (Algorithm 11) computes upper and lower bounds  $UB$  and  $LB$  on the success probability of the kTree algorithm. And the running time of the algorithm is  $O(\text{polylog}(m, k, n))$ .*

*Proof Sketch.* PROBBOUNDS (Algorithm 11) combines the results from FIRSTMOMENTBOUNDS and SECONDMOMENTUB to compute the final bounds by first computing the upper and lower bounds  $UB_1$  and  $UB_2$  of the first moment and the upper bound  $UB_2$  of the second moment. Then applying the Paley-Zygmund inequality to obtain  $LB = \Pr[C > 0] \geq \frac{\mathbf{E}[C]^2}{\mathbf{E}[C^2]} \geq \frac{LB_1^2}{UB_2}$  and Markov's inequality to obtain  $UB = \Pr[C \geq 1] \leq \mathbf{E}[C] \leq UB_1$ .  $\square$

---

**Algorithm 11** PROBBOUNDS

---

**Input:**  $m, k, n, p$ **Output:** Bounds on the success probabilities

- 1:  $\alpha_u, \alpha_l \leftarrow \text{FIRSTMOMENTBOUNDS}(m, k, n, p)$
  - 2:  $\beta \leftarrow \text{SECONDMOMENTUB}(m, k, n, p)$
  - 3:  $\text{UB} \leftarrow \min(\alpha_u, 1)$
  - 4:  $\text{LB} \leftarrow \frac{\alpha_l^2}{\beta}$
  - 5: **return**  $\text{UB}, \text{LB}$
- 

---

**Algorithm 12** SIZEBOUNDINDUCTFACTORS

---

**Input:**  $m, k, p, d, t$  $\triangleright$  Input parameters**Output:**  $\text{UB}, \text{LB}$  $\triangleright$  Upper and lower bounds

- 1:  $s \leftarrow mp^t$
  - 2:  $\alpha \leftarrow \text{PROBSUMINRANGE}(s, p)$
  - 3:  $\beta \leftarrow \text{MRDISTFROMUNIF}(s, p)$
  - 4:  $(\alpha \cdot \beta)^{2^{d-t-1}}$
  - 5:  $(\alpha/\beta)^{2^{d-t-1}}$
  - 6: **return**  $\text{UB}, \text{LB}$
- 

---

**Algorithm 13** SIZEBOUNDS

---

**Input:**  $m, k, n, p$  $\triangleright$  Input parameters**Output:**  $\text{UB}, \text{LB}$  $\triangleright$  Upper and lower bounds

- 1:  $\text{UB} \leftarrow 0, \text{LB} \leftarrow 0$
  - 2: **for**  $d \leftarrow 0$  to  $\log k$  **do**
  - 3:     **if**  $d = 0$  **then**
  - 4:          $\text{UB} \leftarrow \text{UB} + k \cdot n$
  - 5:          $\text{LB} \leftarrow \text{LB} + k \cdot n$
  - 6:     **else if**  $d = 1$  **then**
  - 7:          $\alpha \leftarrow \frac{k}{2} \cdot \text{PROBSUMINRANGE}(m, p) \cdot n^2$
  - 8:          $\text{UB} \leftarrow \text{UB} + \alpha$
  - 9:          $\text{LB} \leftarrow \text{LB} + \alpha$
  - 10:    **else**
  - 11:        $s' \leftarrow mp^{d-1}$
  - 12:        $\gamma_u \leftarrow \text{PROBSUMINRANGE}(s', p), \gamma_l \leftarrow \text{PROBSUMINRANGE}(s', p)$
  - 13:       **for**  $t \leftarrow 0$  to  $d - 2$  **do**
  - 14:            $\beta_u, \beta_l \leftarrow \text{SIZEBOUNDINDUCTFACTORS}(m, k, p, d, t)$
  - 15:            $\gamma_u \leftarrow \gamma_u \cdot \beta_u$
  - 16:            $\gamma_l \leftarrow \gamma_l \cdot \beta_l$
  - 17:          $\text{UB} \leftarrow \text{UB} + \frac{k}{2^d} \cdot n^{2^d} \cdot \gamma_u$
  - 18:          $\text{LB} \leftarrow \text{LB} + \frac{k}{2^d} \cdot n^{2^d} \cdot \gamma_l$
  - 19: **return**  $\text{UB}, \text{LB}$
-

**Proposition 3.10** (Proposition 2.5). *Given parameters  $m, k, n$  and  $p$  satisfying the hypotheses in Proposition 2.5 as inputs, SIZEBOUNDS (Algorithm 13) computes upper and lower bounds  $UB$  and  $LB$  on the expected total size of all lists involved in the execution of the  $k$ -Tree algorithm, such that  $LB \leq \mathbb{E}[\Lambda] \leq UB$  and*

$$\begin{aligned} UB &= nk + \frac{kn^2}{2} \cdot \Pr_{x_1, x_2 \leftarrow U_m} [x_1 + x_2 \in \langle mp \rangle] \\ &\quad + \sum_{d \in [2, \log k]} \frac{k}{2^d} \cdot n^{2^d} \cdot \Pr_{x_1, x_2 \leftarrow U_{mp^{d-1}}} [x_1 + x_2 \in \langle mp^d \rangle] \cdot [\alpha \cdot \beta]^{\sum_{t=0}^{d-2} 2^{d-t-1}}, \\ LB &= nk + \frac{kn^2}{2} \cdot \Pr_{x_1, x_2 \leftarrow U_m} [x_1 + x_2 \in \langle mp \rangle] \\ &\quad + \sum_{d \in [2, \log k]} \frac{k}{2^d} \cdot n^{2^d} \cdot \Pr_{x_1, x_2 \leftarrow U_{mp^{d-1}}} [x_1 + x_2 \in \langle mp^d \rangle] \cdot [\alpha/\beta]^{\sum_{t=0}^{d-2} 2^{d-t-1}}, \end{aligned}$$

where

$$\begin{aligned} \alpha &= \Pr_{x_1^d, \dots, x_{2^{d-t}}^d \leftarrow U_{mp^t}} \left[ \forall i \in [2^{d-t-1}] : (x_{2i-1}^d + x_{2i}^d) \in \langle mp^{t+1} \rangle \right], \\ \beta &= \Delta_{MR}(U_{mp^{t+1}}, 2 \cdot U_{mp^t} |_{\langle mp^{t+1} \rangle})^{2^{d-t-1}}. \end{aligned}$$

$\Lambda$  is the sum of the sizes of all lists generated during the algorithm's execution (as defined above in Proposition 2.5). And the running time of the algorithm is  $O(\text{polylog}(m, k, n))$ .

*Proof Sketch.* Similar to the proof of Proposition 2.5, the proof of the property of SIZEBOUNDINDUCTFACTORS (Algorithm 13) can be referred to the proof sketch of Proposition 3.7. In Algorithm SIZEBOUNDS, the total size of all lists is computed by summing the expected sizes of the lists generated at each level of the  $k$ -Tree algorithm. For  $d = 0$ , the size is  $k \cdot n$  as each list has  $n$  elements as in (64). For  $d = 1$ , the size is computed based on the probability of the sum being in the specified range as in (68). For  $d > 1$ , the size is calculated using the induction factors computed by SIZEBOUNDINDUCTFACTORS, with the initial value  $\gamma_u, \gamma_l$  ( $\xi_{d-1}$ ) calculated by PROBSUMINRANGE following (67). Then the results are scaled according to (71) and (72).  $\square$

---

**Algorithm 14** MAINTHEOREM

---

**Input:**  $m, k, n$

**Output:** Bounds on the probabilities and sizes

- 1:  $p \leftarrow m^{-\frac{1}{\log k+1}}$
  - 2:  $UB_{\text{success}}, LB_{\text{success}} \leftarrow \text{PROBBOUNDS}(m, k, n, p)$
  - 3:  $UB_{\text{size}}, LB_{\text{size}} \leftarrow \text{SIZEBOUNDS}(m, k, n, p)$
  - 4: **return**  $UB_{\text{success}}, LB_{\text{success}}, UB_{\text{size}}, LB_{\text{size}}$
- 

The properties of MAINTHEOREM (Algorithm 14) follows directly from Theorem 3.9 and Proposition 3.10, and the proof is omitted for brevity.

### 3.2 Computed Bounds for $\mathbb{Z}_m$

To compute bounds for the  $k$ -SUM problem over  $\mathbb{Z}_m$ , we provide several slightly modified subroutines to account for the differences in probability calculations when working with modular arithmetic. The key modifications are based on the observations in Section 2.5, particularly for the first iteration of the algorithm.

Notice that in the computed bounds, the only subroutines affected are PROBSUMINRANGE (Algorithm 2), MRDISTFROMUNIF (Algorithm 3), PROBSUMWITHTWOVRINRANGE (Algorithm 4) and MRDISTFROMPAIRUNIF (Algorithm 5) during the first iteration of the algorithm. We first introduce the following modifications to PROBSUMINRANGE:

---

#### Algorithm 15 PROBSUMMODSINRANGE

---

**Input:**  $s, p$  ▷ Modulus and  $p$   
**Output:** Probability of sum being in specified range  
1: **return**  $\frac{2 \cdot \lfloor sp/2 \rfloor + 1}{s}$

---

PROBSUMMODSINRANGE (Algorithm 15) follows direct from (73) in Section 2.5, and we will omit a formal statement of its correctness here. Regarding PROBSUMWITHTWOVRINRANGE (Algorithm 4), its  $\mathbb{Z}_m$  calculation is simply the square of PROBSUMMODSINRANGE. And for MRDISTFROMUNIF (Algorithm 3) and MRDISTFROMPAIRUNIF (Algorithm 5), we relies on the conclusion from Section 2.5 that the max-ratio distance from the uniform distribution is always 1. Then we can simply replace any calls to MRDISTFROMUNIF and MRDISTFROMPAIRUNIF with constant 1 in the first iteration of the algorithm.

To realize the modified calculations during the first iteration, we introduce the following modification to FIRSTMOMENTINDUCTFACTORS, SECONDMOMENTUB, PROBUNIFXMRDIST, SIZEBOUNDINDUCTFACTORS and SIZEBOUNDS.

---

#### Algorithm 16 FIRSTMOMENTINDUCTFACTORSMOD

---

**Input:**  $m, k, p, d$  ▷ Input parameters  
**Output:** UB, LB ▷ Upper and lower bounds  
1:  $s \leftarrow m \cdot p^d$   
2: **if**  $d = 0$  **then**  
3:      $\alpha \leftarrow (\text{PROBSUMMODSINRANGE}(s, p))^{\frac{k}{2^{d+1}}}$   
4:      $\beta \leftarrow 1$   
5: **else**  
6:      $\alpha \leftarrow (\text{PROBSUMINRANGE}(s, p))^{\frac{k}{2^{d+1}}}$   
7:      $\beta \leftarrow (\text{MRDISTFROMUNIF}(s, p))^{\frac{k}{2^{d+1}}}$   
8:     UB  $\leftarrow \alpha \cdot \beta$   
9:     LB  $\leftarrow \alpha / \beta$   
10: **return** UB, LB

---

The modification in SECONDMOMENTUBMOD is realize by passing a flag  $b$  to indicate the first iteration of the algorithm. Only the initial call to SECONDMOMENTUB will be given  $b = 1$ , and it will remain 0 in all the subsequent recursive calls.

These modifications address the calculation of PROBSUMINRANGE and MRDISTFROMUNIF

---

**Algorithm 17** SECONDMOMENTUBMOD

---

```
1: if  $k = 1$  then
2:    $u \leftarrow \frac{1}{2 \cdot \lfloor m \rfloor + 1}$ 
3:   return  $(n \cdot u) \cdot (n \cdot u + 1)$ 
4: else
5:    $m' = m \cdot p$ 
6:    $k' = k/2$ 
7:    $n' = ((\text{PROBUNIFXMRDIST}(m, p, b))^2 \cdot n^4 + 2 \cdot (\text{PROBUNIFXMRDISTPAIR}(m, p, b)) \cdot n^3)^{\frac{1}{2}}$ 
8:   return SECONDMOMENTUB( $m', k', n', p, 0$ )
```

---

---

**Algorithm 18** PROBUNIFMODMXMRDIST

---

**Input:**  $m, p, b$  ▷ Input parameters  
**Output:** Product of uniform probability and max-ratio distance

```
1: if  $b = 0$  then
2:   return PROBSUMINRANGE( $m, p$ ) · MRDISTFROMUNIF( $m, p$ )
3: else
4:   return PROBSUMMODSINRANGE( $m, p$ )
```

---

---

**Algorithm 19** PROBUNIFXMRDISTPAIR

---

**Input:**  $m, p, d$  ▷ Input parameters  
**Output:** Product of uniform probability and max-ratio distance for pairs

```
1: if  $b = 0$  then
2:   return PROBSUMWITHTWO RVINRANGE( $m, p$ ) · MRDISTFROMPAIRUNIF( $m, p$ )
3: else
4:   return (PROBSUMMODSINRANGE( $m, p$ ))2
```

---

---

**Algorithm 20** SIZEBOUNDINDUCTFACTORSMOD

---

**Input:**  $m, k, p, d, t$  ▷ Input parameters  
**Output:** UB, LB ▷ Upper and lower bounds

```
1:  $s \leftarrow mp^t$ 
2: if  $t = 0$  then
3:    $\alpha \leftarrow (\text{PROBSUMMODSINRANGE}(s, p))$ 
4:    $\beta \leftarrow 1$ 
5: else
6:    $\alpha \leftarrow \text{PROBSUMINRANGE}(s, p)$ 
7:    $\beta \leftarrow \text{MRDISTFROMUNIF}(s, p)$ 
8:  $(\alpha \cdot \beta)^{2^{d-t-1}}$ 
9:  $(\alpha/\beta)^{2^{d-t-1}}$ 
10: return UB, LB
```

---

when the input  $s = m$ , corresponding to the first iteration of the algorithm over  $\mathbb{Z}_m$ . They allow us to compute bounds for the  $k$ -SUM problem over  $\mathbb{Z}_m$  that share the same tightness as the bounds for the  $k$ -SUM problem over integers.



---

**Algorithm 21** SIZEBOUNDSMOD

---

**Input:**  $m, k, n, p$ 

▷ Input parameters

**Output:** UB, LB

▷ Upper and lower bounds

```
1: UB  $\leftarrow$  0, LB  $\leftarrow$  0
2: for  $d \leftarrow 0$  to  $\log k$  do
3:   if  $d = 0$  then
4:     UB  $\leftarrow$  UB +  $k \cdot n$ 
5:     LB  $\leftarrow$  LB +  $k \cdot n$ 
6:   else if  $d = 1$  then
7:      $\alpha \leftarrow \frac{k}{2} \cdot \text{PROBSUMMODSINRANGE}(m, p) \cdot n^2$            ▷ Modified for  $\mathbb{Z}_m$ 
8:     UB  $\leftarrow$  UB +  $\alpha$ 
9:     LB  $\leftarrow$  LB +  $\alpha$ 
10:  else
11:     $s' \leftarrow mp^{d-1}$ 
12:     $\gamma_u \leftarrow \text{PROBSUMINRANGE}(s', p)$ ,  $\gamma_l \leftarrow \text{PROBSUMINRANGE}(s', p)$ 
13:    for  $t \leftarrow 0$  to  $d - 2$  do
14:       $\beta_u, \beta_l \leftarrow \text{SIZEBOUNDINDUCTFACTORSMOD}(m, k, p, d, t)$ 
15:       $\gamma_u \leftarrow \gamma_u \cdot \beta_u$ 
16:       $\gamma_l \leftarrow \gamma_l \cdot \beta_l$ 
17:      UB  $\leftarrow$  UB +  $\frac{k}{2^d} \cdot n^{2^d} \cdot \gamma_u$ 
18:      LB  $\leftarrow$  LB +  $\frac{k}{2^d} \cdot n^{2^d} \cdot \gamma_l$ 
19: return UB, LB
```

---

### 3.3 Visualizations and Interpretations

In this section, we present examples of our computing bounds for various parameter settings. These visualizations help us better understand the tightness of the upper and lower bounds of the success probability as key parameters, such as  $m$  and  $k$ , change. We will also directly compare our bounds with our experimental results to validate the trends we observe in this section (see Section 4 for details).

**Success Probability Bounds.** In Figures 6, the most notable observation is that for a fixed  $m$ , the lower bounds on the success probability become looser as  $k$  increases. This trend will be further illustrated when we later compare the bound with our empirical measurements of success probabilities. Such phenomenon is clearly visualized in the case where  $m = 2^{64}$  and  $k = 256$  (see Figure 6b). In this example, the lower bound flattens early and is far from 1, indicating a quick relaxation of the bound as  $k$  increases. This behavior shows that when  $k$  is large, the success probability bounds are less effective for moderately small values of  $m$ .

However, as  $m$  increases, the bounds become significantly tighter for all values of  $k$ . For instance, in the case where  $m = 2^{256}$ , the bounds  $k = 1024$  (Figures 6d) exhibits much better tightness compared to when  $m = 2^{64}$  and  $k = 128$ . This improvement in the tightness of the bounds align with our conclusion in Section 2 that our bounds become asymptotically tight as  $m$  approaches infinity.

Overall, the visualizations indicate that larger values of  $m$  lead to more refined bounds on success probability. And the impact of increasing  $k$  is more pronounced at smaller values of  $m$ ,

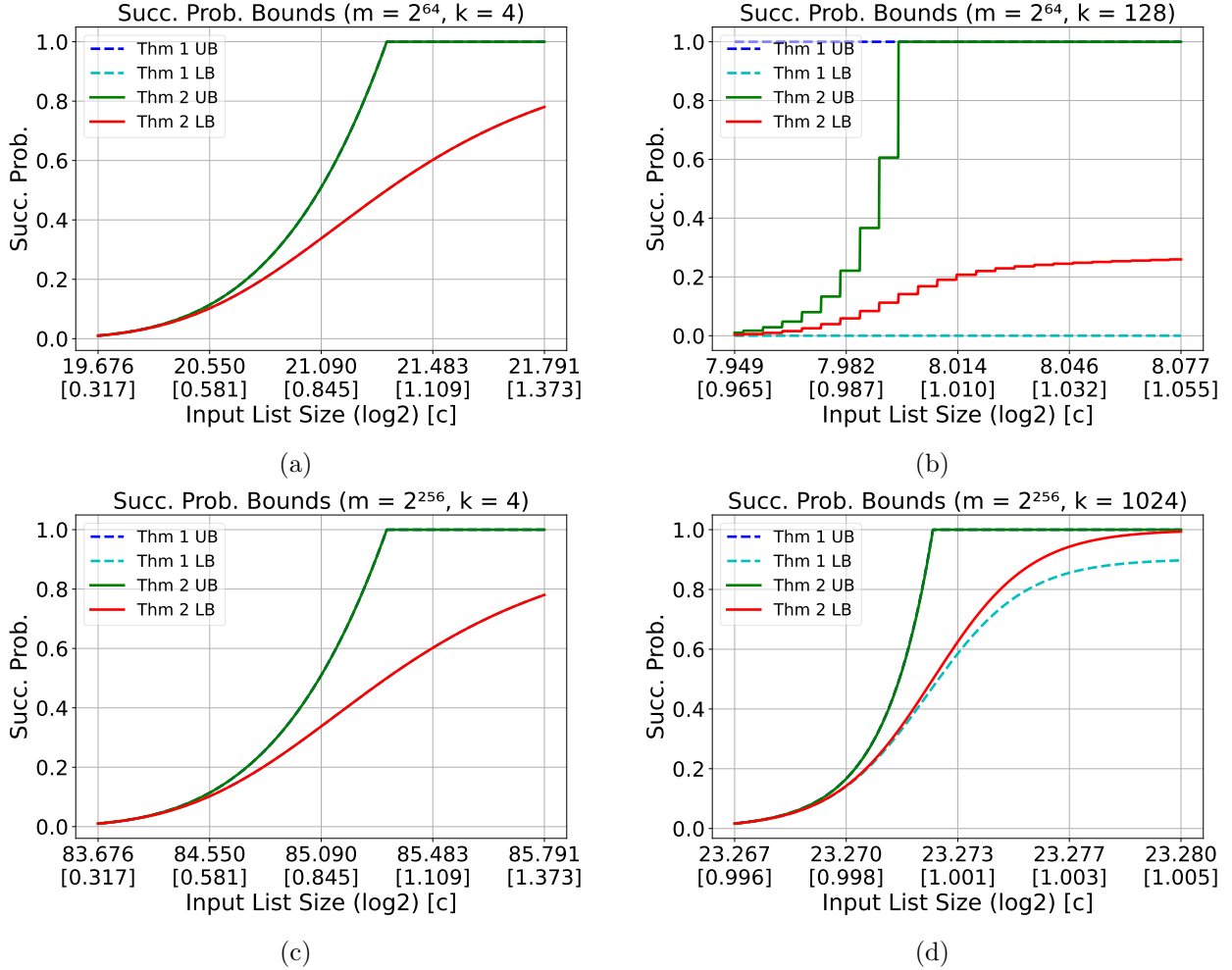
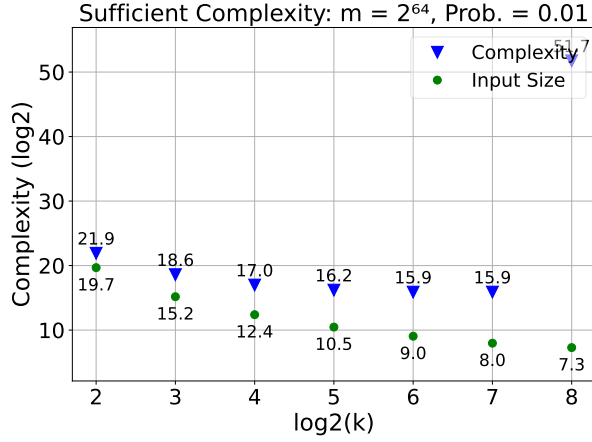


Figure 6: Success probability upper and lower bounds when varying the input list size under fixed  $m$ 's and  $k$ 's. The values in the square brackets report  $c = n/m^{1/(\log k+1)}$  as described in Theorem 1. Note that the cascading patterns of the bounds for certain values of  $m$  and  $k$  are due to rounding the input list size to the nearest integer.

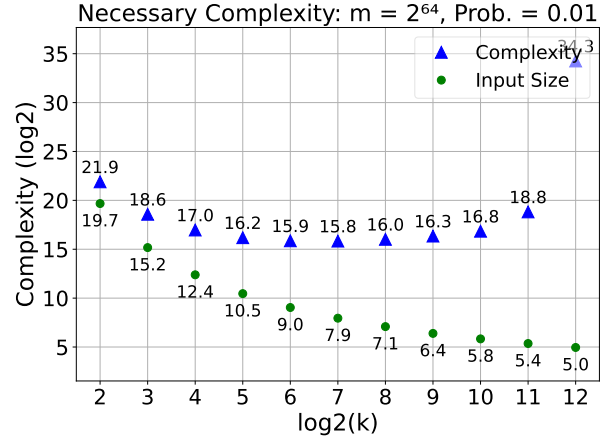
while the bounds become more stabilized as  $m$  increases.

**Complexity Bounds.** In the examples shown in Figure 7, we present the sufficient and necessary complexities (total list sizes) for achieving a target success probability. For the sufficient complexity, given a fixed  $m$ , we compute the minimum input list size via binary search such that our success probability lower bound equals or slightly larger than the specified probability threshold. We then use this minimum input list size as an input to compute our upper bound on the expected total size of all lists. For the necessary complexity, we first search for the minimum input list size such that our success probability upper bound equals or slightly smaller than the specified probability threshold. We then use this minimum input list size to compute our lower bound on the expected total size of all lists.

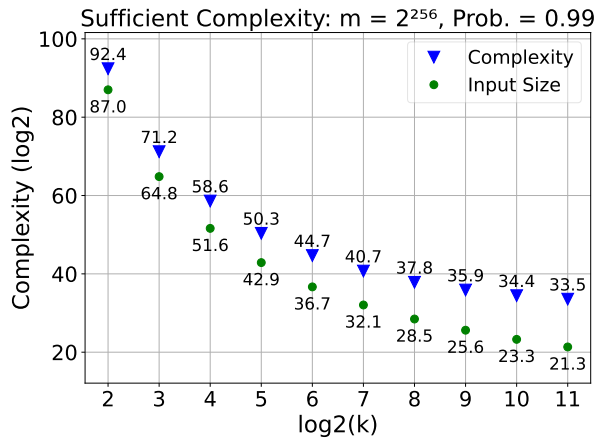
As observed in these figures, when  $k$  increases, the upper bound on the total list size first



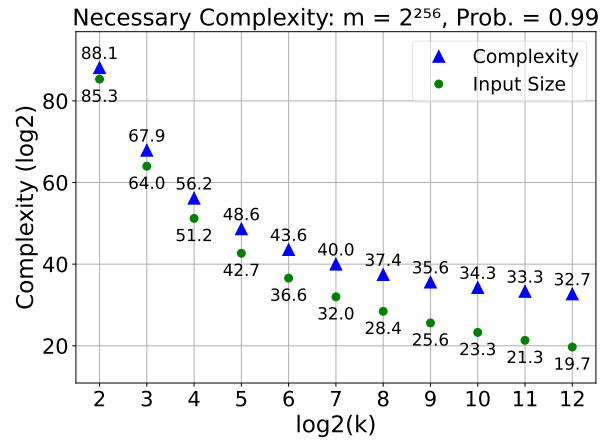
(a)



(b)



(c)



(d)

Figure 7: Size upper and lower bounds when varying  $k$  for fixed  $m$ 's (the fewer amount of  $k$  in some of the Sufficient Complexity plots is due to the lower bound cannot reach the probability threshold even for very large  $c$ ).

decreases accordingly then increases again as  $k$  continues to grow. This trend is more pronounced for small  $m$ 's, as shown in Figures 7a. This behavior is consistent with the results shown in the Figures 6, where the lower bounds on the success probability become looser as  $k$  increases. The immediate consequence is that the input list size has to be substantially larger to achieve the same success probability when  $k$  is large. That accounts for the surge of total list size observed in Figure 7a. As  $m$  increases, the total list sizes become more stable. In Section 4, we see a similar trend of decrease-then-increase of the complexities in our empirical measurements.

Overall, these visualizations provide a straightforward understanding of the efficacy of our method in estimating the computational resource for achieving a target success probability. We will further validate these results through empirical evaluation in Section 4.

### 3.4 Practical Implications

The precise bounds computed by our implementation have several significant implications for the practical use of the kTree algorithm:

1. **Accurate Performance Prediction:** The tighter bounds allow for more accurate predictions of the algorithm’s performance on real-world problem sizes. This is particularly crucial when deciding whether to employ the kTree algorithm for a specific task, as it provides a more reliable estimate of the required computational resources and expected success probability.
2. **Optimal Parameter Selection:** With our precise bounds, practitioners can make better-informed decisions when selecting algorithm parameters. For instance, the optimal choice of the number of lists ( $k$ ) can be more accurately determined to achieve desired success probabilities without actually running the algorithm for various  $k$ .
3. **Confidence in Cryptographic Applications:** In cryptographic settings, where the kTree algorithm might be used for attacks or analysis, having precise bounds allows for more accurate estimation of the computational effort required for potential attacks, leading to better-informed decisions about key sizes and security parameters.

In summary, these computing bounds enhance our understanding of the kTree algorithm’s behavior in practical settings and provide a crucial bridge between theoretical analysis and real-world application. As we move forward to the evaluation of the kTree algorithm, these computed bounds will serve as a valuable reference point and will be further validated against empirical observations.

### 3.5 Limitations

For small values of  $m$ , the tightness of our computed bounds degrade noticeably even when  $k$  is in a reasonable range, despite they are still much tighter than the analytical bounds. This limitation can also be inferred from our analytical results, where our bounds loosen considerably after the point around  $k > m^{1/(\log k+1)}$ .

Besides, for very large values of  $m$  (e.g.,  $2^{512}$ ), our current Python implementation may encounter numerical stability issues due to overflow or underflow. To address this limitation, one can update our implementation by using Python Decimal library that handles large numbers with arbitrary precision with reasonable overhead.

## 4 Experiments

The theoretical analysis and bounds computed in the previous sections are to offer insights into the expected performance of the kTree algorithm. However, empirical experiments are crucial to understand the algorithm’s behavior under practical conditions and to validate these theoretical predictions. In this section, our primary objectives are as follows:

1. Evaluate the behavior of the kTree algorithm under various parameter configurations;
2. Compare the empirical results with our computed bounds.

The results we demonstrate in this section provide empirical evidence on how different parameters influence the algorithm’s success probability, time, and space complexity.

## 4.1 Experimental Setup

In discussions below, we use symbols defined in the context of the description of the kTree algorithm in Figure 5.

**Algorithm Implementation.** We implemented the kTree algorithm in C++, closely following the description provided in Figure 5. The Merge subroutine was implemented by conducting a binary search over the sorted second list for each element in the first list.

**Parameter Configurations.** We investigate the impact of varying the following parameters:

- $m$ : taking values from  $\{2^{64}, 2^{96}, 2^{128}\}$ . Due to limitations in computational resources, we were unable to explore larger values of  $m$ .
- $k$ : The number of input lists, with values ranging from 4 to 1024, varied in powers of 2. Note that not all  $k$ 's are feasible for all  $m$ 's. For example, when  $m = 2^{128}$ , any  $k$  smaller than 512 results in a large input list size that exceeds the computational resources available. We report all the feasible results we have obtained.
- $n$ : The size of each input list, dynamically adjusted via binary search to achieve target success probabilities varied from 0.01 to 1.0. Notice that not all the target probabilities are feasible due to either the nature of the problem (e.g., when  $m$  is relatively small and  $k$  is large, increasing the input list size by 1 will increase the success probability from below 0.2 to above 0.8) or the limit of computational resources. We report all the feasible results we have obtained.

### Evaluation Metrics.

- **Success Probability:** The fraction of trials where the algorithm successfully identifies a set of indices such that the indexed elements of the  $k$  input lists sum to 0.
- **Total Size:** The sum of the sizes of all lists generated when running the kTree algorithm, measured in terms of the number of elements:

$$\sum_{d=0}^{\log k} \sum_{i \in \left[\frac{k}{2^d}\right]} |L_i^d|.$$

We use this measurement as a proxy for the running time of the algorithm (see Remark 1.3).

- **Max Level Size:** The maximum sum of the sizes of all lists generated at one level when running the kTree algorithm:

$$\max_d \sum_{i \in \left[\frac{k}{2^d}\right]} |L_i^d|.$$

This measurement corresponds to the space complexity of the algorithm.

**Hardware and Software Environment.** All experiments were conducted on a machine running Ubuntu 24.04 LTS (64-bit) system on an Intel Core i7-12700 CPU @ 4.90GHz with 16 GB of RAM. The kTree algorithm was implemented in C++ 17, and the experimental results were managed and analyzed using Python 3.9 with libraries such as NumPy and Matplotlib for plotting results.

## 4.2 Success Probability

**Objectives.** This section is to explore and measure the effect of varying the input list size on the success probability of the kTree algorithm under different configurations of  $m$  and  $k$ .

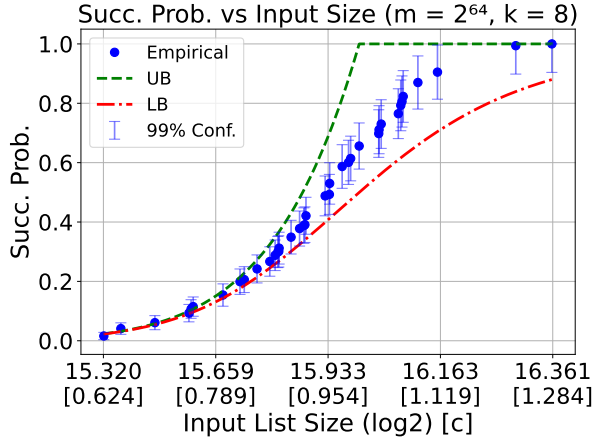
**Methodology.** For each combination of  $m$  and  $k$ , we conducted a series of trials with varying input list sizes. We started with a list size corresponding to  $c = 1$ , where  $c = n/m^{1/(\log k+1)}$  as defined in Theorem 1. We then used binary search to iteratively adjust the list size, exploring a range of  $c$  values that span a typical set of success probabilities (i.e., percentage points). For each parameter configuration, we performed 1000 independent trials to calculate the empirical success probability. Then we further compare the empirical results with our theoretical bounds.

**Results.** Figure 8 presents the results of our experiments. The blue dots represents the measured success probability, while the dashed red and green lines represent the upper and lower bounds derived from our theoretical analysis, respectively. The error bars around the empirical measurements indicates the 99% confidence interval, calculated using the Chernoff inequality. The parameter  $c$ , as defined earlier, serves as a normalized measure of the input list size relative to  $m$  and  $k$ . It allows us to compare results across different parameter configurations and relates directly to our theoretical bounds. These plots enable us to visualize how the success probability transitions from near 0 to near 1 as the input list size increases, and how this transition varies for different values of  $m$  and  $k$ . They also allow us to assess the tightness of our theoretical bounds under various conditions.

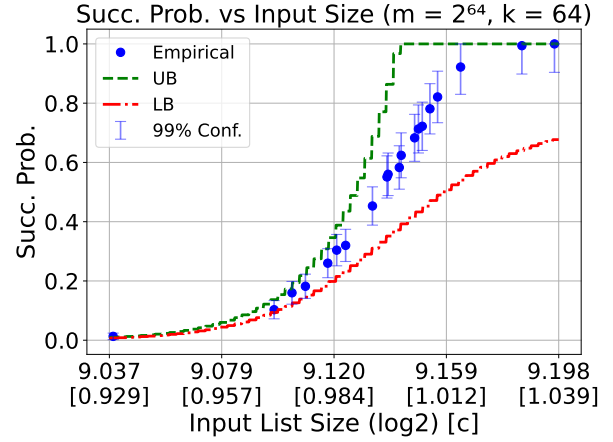
The results shown in Figure 8 illustrate several key trends in the success probability of the kTree algorithm as the input list size varies. We report both the actual input list size  $n$  and the corresponding  $c$  value. We make the following general observations:

- Generally, after fixing reasonable values for  $m$  and  $k$ , the actual success probability approaches 0 and 1 at the left and right extremes when the  $c$  value is slightly below or above 1.
- For a fixed  $m$ , as  $k$  increases, the success probability converges to 0 and 1 more rapidly as  $c$  is decreased or increased linearly.
- In terms of the bounds we provided, for a fixed  $m$ , the tightness of the bounds improves as  $k$  decreases. Conversely, for a fixed  $k$ , increasing  $m$  significantly improves the tightness of our bounds, which aligns with the earlier conclusion that our bounds are asymptotically tight.

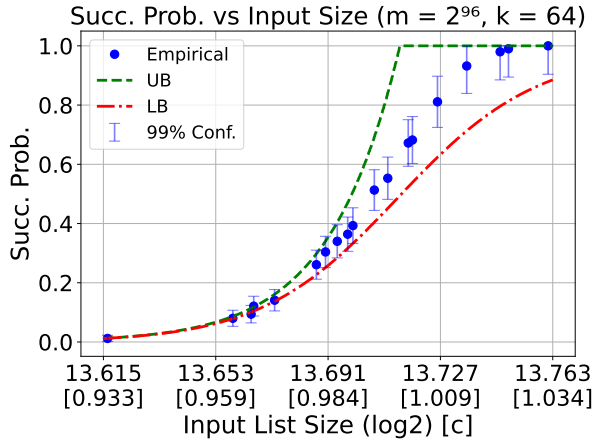
As expected from our theoretical model, the  $c$  value has a direct and significant impact on the success probability. When  $c$  is below 1, the success probability approaches 0 gradually. Conversely, when  $c$  exceeds 1, the success probability converges to 1. When  $m$  is fixed, increasing  $k$  accelerates the convergence of the success probability toward 0 and 1. Larger  $k$  increases the rate at which the success probability transitions between failure and success. This faster convergence for larger



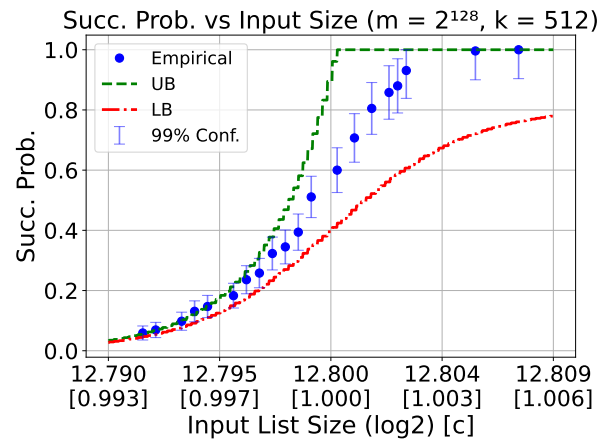
(a)



(b)



(c)

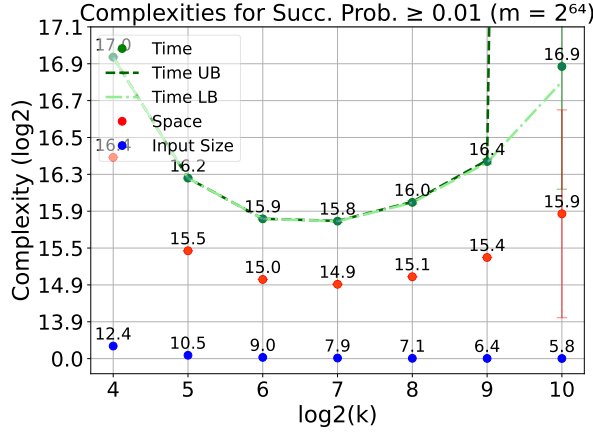


(d)

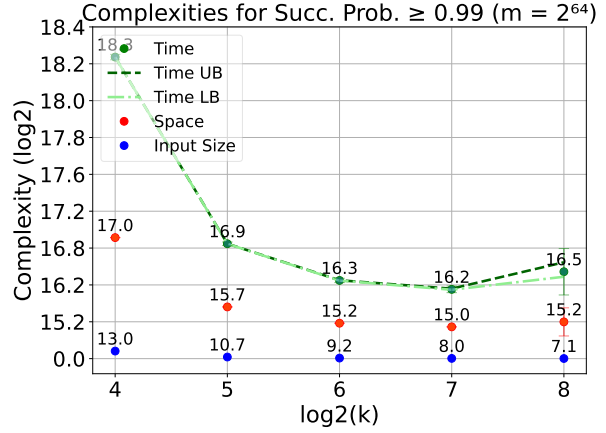
Figure 8: Success probability when varying the input list size under fixed  $m$ 's and  $k$ 's. The values in the square brackets report  $c = n/m^{1/(\log k+1)}$  as described in Theorem 1.

$k$  highlights a sharper threshold between the regions where the algorithm succeeds and fails, which could be beneficial in applications where precise control over success rates is desired.

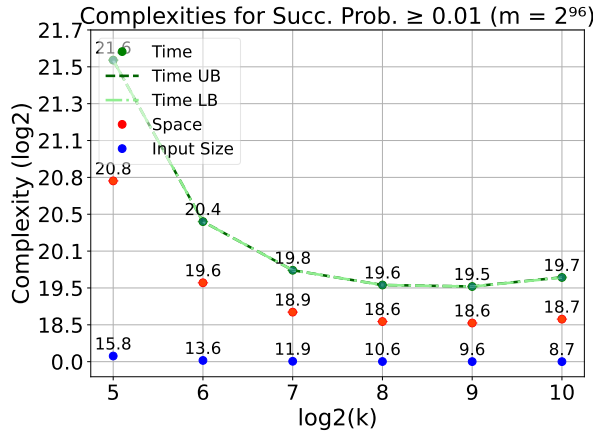
The empirical results closely align with our theoretical predictions regarding the behavior of the kTree algorithm. Specifically, the success probability's dependence on the  $c$  value and the role of  $k$  in determining the speed of convergence are both well-captured by the theoretical bounds. The observation that the bounds are tighter for smaller  $k$  (for fixed  $m$ ) and for larger  $m$  (for fixed  $k$ ) is consistent with the asymptotic analysis provided earlier. Moreover, the empirical data validates the hypothesis that our bounds are asymptotically tight: as  $m$  grows relative to  $k$ , the bounds become increasingly accurate, which confirms that the algorithm's performance can be effectively predicted using the theoretical framework developed.



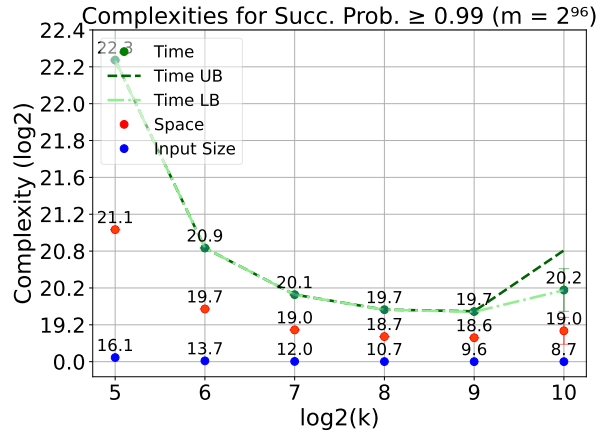
(a)



(b)



(c)



(d)

Figure 9: Measurements of  $k$ Tree’s complexities for different  $k$  under fixed  $m$ ’s. Due to the limitation of computational resources, we were only able to measure the  $k = 512$  and  $k = 1024$  cases for  $m = 128$ . Therefore its plot is not included.

### 4.3 Complexity

**Objectives.** The primary objectives of this section are to measure and analyze the time and space complexity of the  $k$ Tree algorithm under various parameter configurations. Specifically, we aim to:

- Observe how the total list size (a proxy for time complexity) and maximum level size (space complexity) vary with different values of  $k$  for fixed  $m$  and success probability.
- Investigate the impact of different success probability thresholds on the algorithm’s complexity.
- Identify optimal parameter configurations that minimize time and space usage.



**Methodology.** For each  $m$  ( $2^{64}$  and  $2^{96}$ ) and success probability threshold (0.01 and 0.99), we varied  $k$  from 4 to 1024 in the powers of 2. For each configuration:

1. We used binary search to find the minimum input list size  $n$  that achieves a success probability above the specified threshold.
2. We measured the total size and the maximum level size to represent the time and space complexity.
3. We conduct 1000 trials and report the average and standard deviation to ensure reliability.

**Results.** Figure 9 provide a detailed analysis of the algorithm’s complexities. For fixed  $m$  and a given probability threshold, we observe that both time and space complexities initially decrease and then increase as  $k$  grows. This reflects a trade-off: from the results we observed in Figure 8, increasing  $k$  boosts the value of  $p = m^{\frac{-1}{\log k+1}}$  and significantly decrease the threshold  $c = np$  that achieve success probability 1. This further leads to a drastically decrease in the input list size. Nevertheless, at the same time, a larger  $k$  increases the number of total lists and the number of levels in the kTree algorithm. This indicates the importance of finding an optimal  $k$  value that minimizes complexity.

The impact of the probability threshold (0.01 vs 0.99) on the optimal complexity is relatively small, particularly for larger  $k$  values. This suggests that achieving a constantly higher success probability doesn’t necessarily incur a significant additional cost. This can also be inferred from the analytical bounds, where the success probability changes exponentially (where  $k$  is the exponent) as  $n$  vary.

We also observe that as  $m$  increases, the minimum complexity (in both time and space) occurs at a higher value of  $k$ . This indicates that for larger  $m$ , the optimal choice of  $k$ —in terms of minimizing both time and space complexity—shifts upwards. This implies that for larger problem sizes, using more input lists can be beneficial up to a certain point. And the insight we obtained here allows users of the algorithm to select the most efficient parameter configuration using our bounds.

## Acknowledgements

This work was supported by the National Research Foundation, Singapore, under its NRF Fellowship programme, award no. NRF-NRFF14-2022-0010.

## References

- [ABHS19] Amir Abboud, Karl Bringmann, Danny Hermelin, and Dvir Shabtay. Seth-based lower bounds for subset sum and bicriteria path. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 41–57. SIAM, 2019.
- [AC05] Nir Ailon and Bernard Chazelle. Lower bounds for linear degeneracy testing. *J. ACM*, 52(2):157–171, 2005.

- [AL13] Amir Abboud and Kevin Lewi. Exact weight subgraphs and the  $k$ -sum conjecture. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, volume 7965 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2013.
- [ASS<sup>+</sup>24] Shweta Agrawal, Sagnik Saha, Nikolaj I. Schwartzbach, Akhil Vanukuri, and Prashant Nalini Vasudevan.  $k$ -sum in the sparse regime: Complexity and applications. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part II*, volume 14921 of *Lecture Notes in Computer Science*, pages 315–351. Springer, 2024.
- [AW14] Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 434–443, 2014.
- [BC22] Chris Brzuska and Geoffroy Couteau. On building fine-grained one-way functions from strong average-case hardness. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 584–613. Springer, 2022.
- [BDP08] Ilya Baran, Erik D. Demaine, and Mihai Pătraşcu. Subquadratic algorithms for 3sum. *Algorithmica*, 50(4):584–596, Apr 2008.
- [BHP01] Gill Barequet and Sarel Har-Peled. Polygon containment and translational min-hausdorff-distance between segment sets are 3sum-hard. *Int. J. Comput. Geometry Appl.*, 11:465–474, 08 2001.
- [BK17] Alex Biryukov and Dmitry Khovratovich. Equihash: Asymmetric proof-of-work based on the generalized birthday problem. *Ledger*, 2:1–30, 2017.
- [BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, jul 2003.
- [BLL<sup>+</sup>22] Fabrice Benhamouda, Tancrede Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova. On the (in)security of ROS. *J. Cryptol.*, 35(4):25, 2022.
- [BLRL<sup>+</sup>18] Shi Bai, Tancrede Lepoint, Adeline Roux-Langlois, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: Using the rényi divergence rather than the statistical distance. *Journal of Cryptology*, 31(2):610–640, Apr 2018.
- [Bri17] Karl Bringmann. A near-linear pseudopolynomial time algorithm for subset sum. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1073–1084. SIAM, 2017.

- [BSV21] Zvika Brakerski, Noah Stephens-Davidowitz, and Vinod Vaikuntanathan. On the hardness of average-case k-sum. In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference)*, volume 207 of *LIPICs*, pages 29:1–29:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
- [CGI<sup>+</sup>16] Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, ITCS '16*, page 261–270, New York, NY, USA, 2016. Association for Computing Machinery.
- [Cha20] Timothy M. Chan. More logarithmic-factor speedups for 3sum, (median, +)-convolution, and some geometric 3sum-hard problems. *ACM Trans. Algorithms*, 16(1):7:1–7:23, 2020.
- [CJ04] Jean-Sébastien Coron and Antoine Joux. Cryptanalysis of a provably secure cryptographic hash function. *IACR Cryptol. ePrint Arch.*, page 13, 2004.
- [CL23] Eldon Chung and Kasper Green Larsen. Stronger 3sum-indexing lower bounds. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 444–455. SIAM, 2023.
- [DEF<sup>+</sup>19] Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igor Stepanovs. On the security of two-round multi-signatures. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 1084–1101. IEEE, 2019.
- [Din19] Itai Dinur. An algorithmic framework for the generalized birthday problem. *Des. Codes Cryptogr.*, 87(8):1897–1926, 2019.
- [DKK21] Itai Dinur, Nathan Keller, and Ohad Klein. Fine-grained cryptanalysis: Tight conditional bounds for dense k-sum and k-xor. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 80–91. IEEE, 2021.
- [DMNS16] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. *J. Priv. Confidentiality*, 7(3):17–51, 2016.
- [DSW18] Martin Dietzfelbinger, Philipp Schlag, and Stefan Walzer. A subquadratic algorithm for 3xor. In Igor Potapov, Paul G. Spirakis, and James Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, volume 117 of *LIPICs*, pages 59:1–59:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
- [Eri95] Jeff Erickson. Lower bounds for linear satisfiability problems. In Kenneth L. Clarkson, editor, *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete*

*Algorithms, 22-24 January 1995. San Francisco, California, USA*, pages 388–395. ACM/SIAM, 1995.

- [GGH<sup>+</sup>20] Alexander Golovnev, Siyao Guo, Thibaut Horel, Sunoo Park, and Vinod Vaikuntanathan. Data structures meet cryptography: 3sum with preprocessing. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 294–307. ACM, 2020.
- [GNS18] Lorenzo Grassi, María Naya-Plasencia, and André Schrottenloher. Quantum algorithms for the  $k$ -xor problem. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part I*, volume 11272 of *Lecture Notes in Computer Science*, pages 527–559. Springer, 2018.
- [GO95] Anka Gajentaan and Mark H Overmars. On a class of  $O(n^2)$  problems in computational geometry. *Computational Geometry*, 5(3):165–185, 1995.
- [GP18] Allan Grønlund and Seth Pettie. Threesomes, degenerates, and love triangles. *J. ACM*, 65(4), apr 2018.
- [GS17] Omer Gold and Micha Sharir. Improved Bounds for 3SUM,  $k$ -SUM, and Linear Degeneracy. In Kirk Pruhs and Christian Sohler, editors, *25th Annual European Symposium on Algorithms (ESA 2017)*, volume 87 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 42:1–42:13, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- [HS74] Ellis Horowitz and Sartaj Sahni. Computing partitions with applications to the knapsack problem. *J. ACM*, 21(2):277–292, 1974.
- [JKL24] Antoine Joux, Hunter Kippen, and Julian Loss. A concrete analysis of wagner’s  $k$ -list algorithm over  $F_p$ . *IACR Cryptol. ePrint Arch.*, page 282, 2024.
- [Jou03] Antoine Joux. Cryptanalysis of the EMD mode of operation. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2003.
- [JW19] Ce Jin and Hongxun Wu. A simple near-linear pseudopolynomial time randomized algorithm for subset sum. In Jeremy T. Fineman and Michael Mitzenmacher, editors, *2nd Symposium on Simplicity in Algorithms, SOSA 2019, January 8-9, 2019, San Diego, CA, USA*, volume 69 of *OASiCs*, pages 17:1–17:6. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.
- [KP19] Tsvi Kopelowitz and Ely Porat. The strong 3sum-indexing conjecture is false. *CoRR*, abs/1907.11206, 2019.

- [KPP16] Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3sum conjecture. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, page 1272–1287, USA, 2016. Society for Industrial and Applied Mathematics.
- [LF06] Éric Leveil and Pierre-Alain Fouque. An improved LPN algorithm. In Roberto De Prisco and Moti Yung, editors, *Security and Cryptography for Networks, 5th International Conference, SCN 2006, Maiori, Italy, September 6-8, 2006, Proceedings*, volume 4116 of *Lecture Notes in Computer Science*, pages 348–359. Springer, 2006.
- [Lin24] Haoxing Lin. On Wagner’s k-Tree Algorithm Over Integers. <https://github.com/haoxingl/kTreeInt>, 2024.
- [LS19] Gaëtan Leurent and Ferdinand Sibleyras. Low-memory attacks against two-round even-mansour using the 3-xor problem. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 210–235. Springer, 2019.
- [Lyu05] Vadim Lyubashevsky. The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In Chandra Chekuri, Klaus Jansen, José D. P. Rolim, and Luca Trevisan, editors, *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th International Workshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings*, volume 3624 of *Lecture Notes in Computer Science*, pages 378–389. Springer, 2005.
- [MS12] Lorenz Minder and Alistair Sinclair. The extended k-tree algorithm. *J. Cryptol.*, 25(2):349–382, 2012.
- [NS15] Ivica Nikolic and Yu Sasaki. Refinements of the k-tree algorithm for the generalized birthday problem. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 683–703. Springer, 2015.
- [Pat10] Mihai Patrascu. Towards polynomial lower bounds for dynamic problems. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, STOC '10, page 603–610, New York, NY, USA, 2010. Association for Computing Machinery.
- [PT12] Mihai Patrascu and Mikkel Thorup. The power of simple tabulation hashing. *J. ACM*, 59(3):14:1–14:50, 2012.
- [PW10] Mihai Patrascu and Ryan Williams. On the possibility of faster SAT algorithms. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1065–1075. SIAM, 2010.

- [Roc24] Sebastien Roch. *Modern Discrete Probability: An Essential Toolkit*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2024.
- [Sch01] Claus-Peter Schnorr. Security of blind discrete log signatures against interactive attacks. In Sihan Qing, Tatsuaki Okamoto, and Jianying Zhou, editors, *Information and Communications Security, Third International Conference, ICICS 2001, Xian, China, November 13-16, 2001*, volume 2229 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2001.
- [SEO03] Michael Soss, Jeff Erickson, and Mark Overmars. Preprocessing chains for fast dihedral rotations is hard or even impossible. *Computational Geometry*, 26(3):235–246, 2003.
- [Sha08] Andrew Shallue. An improved multi-set algorithm for the dense subset sum problem. In Alfred J. van der Poorten and Andreas Stein, editors, *Algorithmic Number Theory, 8th International Symposium, ANTS-VIII, Banff, Canada, May 17-22, 2008, Proceedings*, volume 5011 of *Lecture Notes in Computer Science*, pages 416–429. Springer, 2008.
- [Wag02] David A. Wagner. A generalized birthday problem. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303. Springer, 2002.