Quicksort, Largest Bucket, and Min-Wise Hashing with Limited Independence

Mathias Bæk Tejs Knudsen[1] * and Morten Stöckel[2] **

[1] University of Copenhagen
knudsen@di.ku.dk
[2] IT University of Copenhagen
mstc@itu.dk

Abstract. Randomized algorithms and data structures are often analyzed under the assumption of access to a perfect source of randomness. The most fundamental metric used to measure how "random" a hash function or a random number generator is, is its *independence*: a sequence of random variables is said to be k-independent if every variable is uniform and every size k subset is independent.

In this paper we consider three classic algorithms under limited independence. Besides the theoretical interest in removing the unrealistic assumption of full independence, the work is motivated by lower independence being more practical. We provide new bounds for randomized quicksort, min-wise hashing and largest bucket size under limited independence. Our results can be summarized as follows.

- *Randomized quicksort.* When pivot elements are computed using a 5-independent hash function, Karloff and Raghavan, J.ACM'93 showed $\mathcal{O}(n \log n)$ expected worst-case running time for a special version of quicksort. We improve upon this, showing that the same running time is achieved with only 4-independence.
- *Min-wise hashing.* For a set A, consider the probability of a particular element being mapped to the smallest hash value. It is known that 5-independence implies the optimal probability $\mathcal{O}(1/n)$. Broder et al., STOC'98 showed that 2-independence implies it is $\mathcal{O}(1/\sqrt{|A|})$. We show a matching lower bound as well as new tight bounds for 3- and 4-independent hash functions.
- *Largest bucket.* We consider the case where n balls are distributed to n buckets using a k -independent hash function and analyze the largest bucket size. Alon et. al, STOC'97 showed that there exists a 2-independent hash function implying a bucket of size $\Omega(n^{1/2})$. We generalize the bound, providing a k-independent family of functions that imply size $\Omega(n^{1/k})$.

* Research partly supported by Mikkel Thorup's Advanced Grant from the Danish Council for Independent Research under the Sapere Aude programme and the FNU project AlgoDisc - Discrete Mathematics, Algorithms, and Data Structures.
** This author is supported by the Danish National Research Foundation under the Sapere Aude program.

1 Introduction

A unifying metric of strength of hash functions and pseudorandom number generators is the *independence* of the function. We say that a sequence of random variables is k-independent if every random variable is uniform and every size k subset is independent. A question of theoretical interest is, regarding each algorithmic application, *how much independence is required?*. With the standard implementation of a random generator or hash function via a $k-$ degree polynomial k determines both the space used and the amount of randomness provided. A typical assumption when performing algorithmic analysis is to just assume full independence, i.e., that for input size n then the hash function is n-independent. Besides the interest from a theoretic perspective, the question of how much independence is required is in fact interesting from a practical perspective: hash functions and generators with lower independence are as a rule of thumb faster in practice than those with higher independence, hence if it is proven that the algorithmic application needs only k-independence to work, then it can provide a speedup for an implementation to specifically pick a fast construction that provides the required k-independence. In this paper we consider three fundamental applications of random hashing, where we provide new bounds for limited independence.

Min-wise hashing. We consider the commonly used scheme *min-wise hashing*, which was first introduced by Broder [2] and has several well-founded applications (see Section 2). Here we study families of hash functions, where a function h is picked uniformly at random from the family and applied to all elements of a set A. For any element $x \in A$ we say that h is min-wise independent if $\mathbf{Pr}(\min h(A) = x) = 1/|A|$ and ε-min-wise if $\mathbf{Pr}(\min h(A) = x) = (1+\varepsilon)/|A|$. For this problem we show new tight bounds for $k = 2, 3, 4$ of $\varepsilon = \Theta(\sqrt{n}), \Theta(\log n), \Theta(\log n)$ respectively and for $k = 5$ it is folklore that $O(1)$-min-wise ($\varepsilon = \mathcal{O}(1)$) can be achieved. Since tight bounds for $k \geq 5$ exist (see Section 2) , our contribution closes the problem.

Randomized quicksort. Next we consider a classic sorting algorithm presented in many randomized algorithms books, e.g. already on page three of Motwani-Raghavan [12]. The classic analysis of quicksort in Motwani-Raghavan uses crucially the probability of a particular element being mapped to the smallest hash value out of all the elements: the expected worst-case running time in this analysis is $\mathcal{O}(n \log n \cdot \mathbf{Pr}(\min h(A) = x))$, where A is the set of n elements to be sorted and $x \in A$. It follows directly from our new tight min-wise bounds that this analysis cannot be improved further. A special version of randomized quicksort was showed by Karloff and Raghavan to use expected worst-case time $\mathcal{O}(n \log n)$ when the pivot elements are chosen using a 5-independent hash function [11]. Our main result is a new general bound for the number of comparisons performed under limited independence, which applies to several settings of quicksort, including the setting of Karloff-Raghavan where we show the same running time using only 4-independence. Furthermore, we show that $k = 2$ and $k = 3$ can imply expected worst-case time $\Omega\left(n \log^2 n\right)$. An interesting observation is that our new bounds for $k = 4$ and $k = 2$ shows that the classic analysis using min-

wise hashing is not tight, as we go below those bounds by a factor $\log n$ for $k = 4$ and a factor $\sqrt{n}/\log n$ for $k = 2$. Our findings imply that a faster 4-independent hash function can be used to guarantee the optimal running time for randomized quicksort, which could potentially be of practical interest. Interestingly, our new bounds on the number of performed comparisons under limited independence has implications on classic algorithms for binary planar partitions and treaps. For binary planar partitions our results imply expected partition size $\mathcal{O}(n \log n)$ for the classic randomized algorithm for computing binary planar partitions [12, Page 10] under 4-independence. For randomized treaps [12, Page 201] our new results imply $\mathcal{O}(\log n)$ worst-case depth for 4-independence.

Larget bucket size. The last setting we consider is throwing n balls into n buckets using an k-independent hash function and analyzing the size of the largest bucket. This can be regarded as a load balancing as the balls can represent "tasks" and the buckets represent processing units. Our main result is a family of k-independent hash functions, which when used in this setting implies largest bucket size $\Omega(n^{1/k})$ with constant probability. This result was previously known only for $k = 2$ due to Alon et al. [1] and our result is a generalization of their bound. As an example of the usefulness of such bucket size bounds, consider the fundamental data structure; the dictionary. Widely used algorithms books such as Cormen et al. [7] teaches as the standard method to implement a dictionary to use an array with *chaining*. Chaining here simply means that for each key, corresponding to an entry in the array, we have a linked list (chain) and when a new key-value pair is inserted, it is inserted at the end of the linked list. Clearly then, searching for a particular key-value pair takes worst-case time proportional to the size of the largest chain. Hence, if one is interested in worst-case lookup time guarantees then the expected largest bucket size formed by the keys in the dictionary is of great importance.

2 Relation to previous work

We will briefly review related work on the topic of bounding the independence used as well as mention some of the popular hash function constructions.

The line of research that considers the amount of independence required is substantial. As examples, Pagh et al. [13] showed that linear probing works with 5-independence. For the case of ε-min-wise hashing ("almost" min-wise-hashing as used e.g. in [9]) Indyk showed that $\mathcal{O}(\log \frac{1}{\varepsilon})$-independence is sufficient. For both of the above problems Thorup and Pătraşcu [15] showed optimality: They show existence of explicit families of hash functions that for linear probing is 4-independent leading to $\Omega(\log n)$ probes and for ε-min-wise hashing is $\Omega(\log \frac{1}{\varepsilon})$-independent that implies (2ε)-min-wise hashing. Additionally, they show that the popular multiply-shift hashing scheme by Dietzfelbinger et al. [8] is not sufficient for linear probing and ε-min-wise hashing. In terms of lower bounds, it was shown by Broder et al.[3] that $k = 2$ implies $\mathbf{Pr}(\min h(A) = x) = 1/\sqrt{|A|}$. We provide a matching lower bound and new tight bounds for $k = 3, 4$. Additionally we review a folklore $\mathcal{O}(1/n)$ upper bound for $k = 5$. Our lower bound proofs for

min-wise hashing (see Table 1) for $k = 3, 4$ are surprisingly similar to those of Thorup and Pătraşcu for linear probing, in fact we use the same "bad" families of hash functions but with a different analysis. Further the same families imply the same multiplicative factors relative to the optimal. Our new tight bounds together with the bounds for $k \geq 5$ due to [9,15] provide the full picture of how min-wise hashing behaves under limited independence.

Randomized quicksort[12] is well known to sort n elements in expected time $\mathcal{O}(n \log n)$ under full independence. Given that pivot elements are picked by having n random variables with outcomes $0, \ldots, n-1$ and the outcome of variable i in the sequence determines the ith pivot element, then running time $\mathcal{O}(n \log n)$ has been shown[11] for $k = 5$. We improve this and show $\mathcal{O}(n \log n)$ time for $k = 4$ in the same setting. To the knowledge of the authors, it is still an open problem to analyze the version of randomized quicksort under limited independence as presented by e.g. Motwani-Raghavan. The analysis of both the randomized binary planar partition algorithm and the randomized treap in Motwani-Raghavan is done using the exact same argument as for quicksort, namely using min-wise hashing which we show cannot be improved further and is not tight. Our new quicksort bounds directly translates to improvements for these two applications. The randomized binary planar partition algorithm is hence improved to be of expected size $\mathcal{O}(n \log^2 n)$ for $k = 2$ and $\mathcal{O}(n \log n)$ for $k = 4$, and the expected worst case depth of any node in a randomized treap is improved to be $\mathcal{O}(\log^2 n)$ for $k = 2$ and $\mathcal{O}(\log n)$ for $k = 4$.

As briefly mentioned earlier, our largest bucket size result is related to the generalization of Alon et al., STOC'97, specifically [1, Theorem 2]. They show that for a (perfect square) field $\mathbb{F}$ then the class $\mathcal{H}$ of all linear transformations between $\mathbb{F}^2$ and $\mathbb{F}$ has the property that when a hash function is picked uniformly at random from $h \in \mathcal{H}$ then an input set of size n exists so that the largest bucket has size at least $\sqrt{n}$. In terms of upper bounds for largest bucket size, remember that a family $\mathcal{H}_u$ of hash functions that map from $\mathcal{U}$ to $[n]$ is *universal* [4] if for a h picked uniformly from $\mathcal{H}_u$ it holds

$$\forall x \neq y \in \mathcal{U} : \mathbf{Pr}(h(x) = h(y)) \leq 1/n.$$

Universal hash functions are known to have expected largest bucket size at most $\sqrt{n} + 1/2$, hence essentially tight compared to the bound $\sqrt{n}$ lower bound of Alon et al. On the other end of the spectrum, full independence is known to give expected largest bucket size $\Theta(\log n / \log\log n)$ due to a standard application of Chernoff bounds. This bound was proven to hold for $\Theta(\log n / \log\log n)$-independence as well [16]. In Section 7.1 we additionally review a folklore upper bound coinciding with our new $\Omega(n^{1/k})$ lower bound.

Since the question of how much independence is needed from a practical perspective often could be rephrased "how fast a hash function can I use and maintain algorithmic guarantees?" we will briefly recap some used hash functions and pseudorandom generators. Functions with lower independence are typically faster in practice than functions with higher. The formalization of this is due to Siegel's lower bound [17] where he shows that in the cell probe model, to achieve

k-independence and number of probes $t < k$ then you need space $k(n/k)^{1/t}$. Since space usage scales with the independence k then for high k the effects of the memory hierarchy will mean that even if the time is held constant the practical time will scale with k as cache effects etc. impact the running time.

The most used hashing scheme in practice is, as mentioned, the 2-independent multiply-shift by Dietzfelbinger et al. [8], which can be twice as fast [19] compared to even the simplest linear transformation $x \mapsto (ax + b) \mod p$. For 3-independence we have due to (analysis by) Thorup and Pătraşcu the simple tabulation scheme [14], which can be altered to give 5-universality [20]. For general k-independent hash functions the standard solution is degree $k-1$ polynomials, however especially for low k these are known to run slowly, e.g. for $k = 5$ then polynomial hashing is 5 times slower than the tabulation based solution of [20]. Alternatively for high independence the double tabulation scheme by Thorup[18], which builds on Siegels result [17], can potentially be practical. On smaller universes Thorup gives explicit and practical parameters for 100-independence. Also for high independence, the nearly optimal hash function of Christiani et al.[6] should be practical. For generating k-independent variables then Christiani and Pagh's constant time generator [5] performs well - their method is at an order of magnitude faster than evaluating a polynomial using fast fourier transform. We note that even though constant time generators as the above exist, the practical evaluation will actually scale with the independence, as the memory usage of the methods depend on the independence and so the effects of the underlying memory hierarchy comes to effect.

Finally, we would like to note that the paradigm of independence has its limitations in the sense that even though one can prove that k-independence by itself does not imply certain algorithmic guarantees, it can not be ruled out that k-independent hash functions exist that do. That is, lower bound proofs typically construct artificial families to provide counter examples, which in practice would not come into play. As an example, consider that linear probing needs 5-independence to work as mentioned above but it has been proven to work with simple tabulation hashing [14], which only has 3-independence.

3 Our results

With regard to min-wise hashing, we close this version of the problem by providing new and tight bounds for $k = 2, 3, 4$. We consider the following setting: let A bet a set of size n and let $\mathcal{H}$ be a k-independent family of hash functions. We examine the probability of any element $x \in A$ receiving the smallest hash value $h(x)$ out of all elements in A when $h \in \mathcal{H}$ is picked uniformly at random. For the case of $k = 2, 3, 4$-independent families we provide new bounds as shown in Table 1, which provides a full understanding of the parameter space as a tight bound of $\mathbf{Pr}(\min h(A) = x) = \mathcal{O}(1/n)$ is known for $k \geq 5$ due to Indyk[9]. We make note that our lower bound proofs, which work by providing explicit "bad" families of functions, share similarity with Thorup and Pătraşcu's [15, Table 1] proof of linear probing. In fact, our bad families of functions used are exactly the

	$k=2$	$k=3$	$k=4$	$k \geq 5$
Upper bound	$\mathcal{O}(\sqrt{n}/n)$	$\mathcal{O}((\log n)/n)^*$	$\mathcal{O}((\log n)/n)^*$	$\mathcal{O}(1/n)$
Lower bound	$\Omega(\sqrt{n}/n)^*$	$\Omega((\log n)/n)^*$	$\Omega((\log n)/n)^*$	$\Omega(1/n)$

Table 1. Result overview for min-wise hashing. Results in this paper are marked with *. For a set A of size n and an element $x \in A$ the cells correspond the probability $\mathbf{Pr}(\min h(A) = x)$ for a hash function h picked uniformly at random from a k-independent family $\mathcal{H}$.

same, while the analysis is different. Surprisingly, the constructions imply the same factor relative to optimal as in linear probing, for every examined value of k.

Next, we consider randomized quicksort under limited independence. In the same setting as Karloff and Raghavan [11] our main result is that 4-independence is sufficient for the optimal $\mathcal{O}(n \log n)$ expected worst-case running time. The setting is essentially that pivot elements are picked from a sequence of k-independent random variables that are pre-computed. Our results apply to a related setting of quicksort as well as to the analysis of binary planar partitions and randomized treaps. Our results are summarized in Table 2.

	$k=2$	$k=3$	$k=4$	$k \geq 5$
Upper bound	$\mathcal{O}(n \log^2 n)^*$	$\mathcal{O}(n \log^2 n)^*$	$\mathcal{O}(n \log n)^*$	$\mathcal{O}(n \log n)$
Lower bound	$\Omega(n \log n)$	$\Omega(n \log n)$	$\Omega(n \log n)$	$\Omega(n \log n)$

Table 2. Result overview for randomized quicksort. Results in this paper are marked with *. When our hash function h is picked uniformly from k-independent family $\mathcal{H}$ then the cells in the table denote the expected running time to sort n distinct elements. The 5-independent upper bound is from Karloff-Raghavan[11].

Finally for the fundamental case of throwing n balls into n buckets. The main result is a simple k-independent family of functions which when used to throw the balls imply that with constant probability the largest bucket has $\Omega(n^{1/k})$ balls. We show the theorem below.

Theorem 1. *Consider the setting where n balls are distributed among n buckets using a random hash function h. For $m \leq n$ and any $k \in \mathbb{N}$ such that $k < n^{1/k}$ and $m^k \geq n$ a k-independent distribution over hash functions exists such that the largest bucket size is $\Omega(m)$ with probability $\Omega\left(\frac{n}{m^k}\right)$ when h is chosen according to this distribution.*

An implication of Theorem 1 is that we now have the full understanding of the parameter space for this problem, as it was well known that independence

$k = \mathcal{O}(\log n/\text{loglog}n)$ implied $\Theta(\log n/\text{loglog}n)$ balls in the largest bucket. We summarize with the corollary below.

Corollary 1. *Consider the setting where n balls are distributed among n buckets using a random hash function h. Given an integer k a distribution over hash functions exists such that if h is chosen according to this distribution then with L being the size of the largest bucket*

(a) if $k \leq n^{1/k}$ then $L = \Omega\left(n^{1/k}\right)$ with probability $\Omega(1)$.
(b) if $k > n^{1/k}$ then $L = \Omega\left(\log n/\log\log n\right)$ with probability $\Omega(1)$.

We note that the result of Theorem 1 is not quite the generalization of the lower bound of Alon et al. since they show $\Omega(n^{1/2})$ largest bucket size for any linear transformation while our result provides an explicit worst-case k-independent scheme to achieve largest bucket size $\Omega(n^{1/k})$. However, as is evident from the proof in the next section, our scheme is not that artificial: In fact it is "nearly" standard polynomial hashing, providing hope that the true generalization of Alon et al. can be shown.

4 Preliminaries

We will introduce some notation and fundamentals used in the paper. For an integer n we let $[n]$ denote $\{0, \ldots, n-1\}$. For an event E we let $[E]$ be the variable that is 1 if E occurs and 0 otherwise. Unless explicitly stated otherwise, the $\log n$ refers to the base 2 logarithm of n. For a real number x and a non-negative integer k we define $x^{\underline{k}}$ as $x(x-1)\ldots(x-(k-1))$.

The paper is about application bounds when the independence of the random variables used is limited. We define independence of a hash function formally below.

Definition 1. *Let $h : \mathcal{U} \mapsto V$ be a random hash function, $k \in \mathbb{N}$ and let $u_1, \ldots, u_k$ be any distinct k elements from $\mathcal{U}$ and $v_1, \ldots, v_k$ be any k elements from V.*
Then h is k-independent if it holds that

$$\mathbf{Pr}\left(h(u_1) = v_1 \wedge \ldots \wedge h(u_k) = v_k\right) = \frac{1}{|V|^k}.$$

Note that an equivalent definition for a sequence of random variables hold: they are k-independent if any element is uniformly distributed and every k-tuple of them is independent.

5 Min-wise hashing

In this section we show the bounds that can be seen in Table 1. As mentioned earlier, there is a close relationship between the worst case query time of an

element in linear probing and min-wise hashing when analysed under the assumption of hash functions with limited independence. Intuitively, long query time for linear probing is caused by many hash values being "close" to the hash value of the query element. On the other hand a hash value is likely to be the minimum if it is "far away" from the other hash values. So intuitvely, min-wise hashing and linear probing are related by the fact that good guarantees require a "sharp" concentration on how close to the hash value of the query element the other hash values are.

5.1 Upper bounds

We show the following theorem which results in the upper bounds shown in Table 1. Note that the bound for 4-independence follows trivially from the bound for 3-independence and that the 5-independence bound is folklore but included for completeness.

Theorem 2. *Let $X = \{x_0, x_1, \ldots, x_n\}$ and $h : X \to (0,1)$ be a hash function. If h is 3-independent*

$$\mathbf{Pr}\left(h(x_0) < \min_{i\in\{1,\ldots,n\}} \{h(x_i)\}\right) = \mathcal{O}\left(\frac{\log(n+1)}{n+1}\right)$$

If h is 5-independent

$$\mathbf{Pr}\left(h(x_0) < \min_{i\in\{1,\ldots,n\}} \{h(x_i)\}\right) = \mathcal{O}\left(\frac{1}{n+1}\right)$$

Proof. For notational convenience let E denote the event $\left(h(x_0) < \min_{i\in\{1,\ldots,n\}} \{h(x_i)\}\right)$. First assume that h is 3-independent. Fix $h(x_0) = \alpha \in (0,1)$. Then h is 2-independent on the remaining keys. Let $Z = \sum_{i=1}^{n} [h(x_1) \leq \alpha]$. Then under the assumption $h(x_0) = \alpha$:

$$\mathbf{Pr}\left(E \mid h(x_0) = \alpha\right) = \mathbf{Pr}\left(Z = 0 \mid h(x_0) = \alpha\right) \leq \mathbf{Pr}\left(|Z - \mathbb{E}Z| \geq \mathbb{E}Z \mid h(x_0) = \alpha\right)$$

Now since h is 2-independent on the remaining keys we see that $\mathbf{Pr}\left(E \mid h(x_0) = \alpha\right)$ is upper bounded by (using Fact 6):

$$\begin{aligned}\mathbf{Pr}\left(|Z - \mathbb{E}Z| \geq \mathbb{E}Z \mid h(x_0) = \alpha\right) &\leq \frac{\mathbb{E}\left((Z - \mathbb{E}Z)^2\right)}{(\mathbb{E}Z)^2} = \mathcal{O}\left(\frac{1}{\mathbb{E}Z}\right) \\ &= \mathcal{O}\left(\frac{1}{n\alpha}\right) \qquad (1)\end{aligned}$$

Hence:

$$\begin{aligned}\mathbf{Pr}\left(E \mid h(x_0) = \alpha\right) &= \int_0^1 \mathbf{Pr}\left(E \mid h(x_0) = \alpha\right) d\alpha \\ &\leq \frac{1}{n} + \int_{1/n}^1 \mathcal{O}\left(\frac{1}{n\alpha}\right) = \mathcal{O}\left(\frac{\log(n+1)}{n+1}\right) \qquad (2)\end{aligned}$$

This proves the first part of the theorem. Now assume that h is 5-independent and define Z in the same way as before. In the same manner as we established the upper bound for $\mathbf{Pr}\left(E \mid h(x_0) = \alpha\right)$ in (1) we see that it is now upper bounded by (using Fact 6):

$$\begin{aligned}\mathbf{Pr}\left(|Z - \mathbb{E}Z| \geq \mathbb{E}Z \mid h(x_0) = \alpha\right) &\leq \frac{\mathbb{E}\left((Z - \mathbb{E}Z)^4\right)}{(\mathbb{E}Z)^4} \\ &= \mathcal{O}\left(\frac{1}{(\mathbb{E}Z)^2}\right) = \mathcal{O}\left(\frac{1}{(n\alpha)^2}\right)\end{aligned}$$

In the same manner as in (2) we now see that

$$\mathbf{Pr}\left(E\right) = \int_0^1 \mathbf{Pr}\left(E \mid h(x_0) = \alpha\right) d\alpha \leq \frac{1}{n} + \int_{1/n}^1 \mathcal{O}\left(\frac{1}{(n\alpha)^2}\right) = \mathcal{O}\left(\frac{1}{n+1}\right)$$

■

5.2 Lower bounds

We first show the $k = 4$ lower bound seen in Table 1. As mentioned earlier, the argument follows from the same "bad" distrubition as Thorup and Pătraşcu[15], but with a different analysis.

Theorem 3. *For any key set $X = \{x_0, x_1, \ldots, x_n\}$ there exists a random hash function $h : X \to (0, 1)$ that is 4-independent such that*

$$\mathbf{Pr}\left(h(x_0) < \min\left\{h(x_1), \ldots, h(x_n)\right\}\right) = \Omega\left(\frac{\log(n+1)}{n+1}\right) \tag{3}$$

Proof. We consider the strategy from Thorup and Pătraşcu [15, Section 2.3] where we hash X into $[t]$, where t power of 2 such that $t = \Theta(n)$. We use the strategy to determine the first $\log t$ bits of the values of h and let the remaining bits be chosen independently and uniformly at random. The strategy ensures that for every $\ell \in \left[\frac{2}{3}\log t, \frac{5}{6}\log t\right]$ with probability $\Theta(2^\ell/n)$ there exists an interval I of size $\Theta(2^{-\ell})$ such that $h(x_0)$ is uniformly distributed in I and I contains at least $\frac{t}{2^\ell} \cdot (1 + \Omega(1))$ keys from X. Furthermore these events are disjoint. From the definition of the algorithm we see that for every $\ell \in \left[\frac{2}{3}\log t, \frac{5}{6}\log t\right]$ with probability $\Theta(2^\ell/n)$ there exists an interval I of size $\Theta(2^{-\ell})$ such that $h(x_0)$ is uniformly distributed in I and I contains no other element than $h(x_0)$. Let y be the maximal value of all of $h(x_1), \ldots, h(x_n)$ which are smaller than $h(x_0)$ and 0 if all hash values are greater than $h(x_0)$. Then we know that:

$$\mathbb{E}(h(x_0) - y) \geq \sum_{\ell \in \left[\frac{2}{3}\log t, \frac{5}{6}\log t\right]} \Theta\left(\frac{2^\ell}{n}\right) \cdot \Theta(2^{-\ell}) = \Theta\left(\frac{\log n}{n}\right)$$

We know define the hash function $h' : X \to (0,1)$ by $h'(x) = (h(x) - z) \bmod 1$ where $z \in (0,1)$ is chosen uniformly at random. Now fix the choice of h. Then $h'(x_0)$ is smaller than $\min\{h'(x_1), \ldots, h'(x_n)\}$ if $z \in (y, h(x_0))$. Hence for this fixed choice of h:

$$\mathbf{Pr}\left(h'(x_0) < \min\{h'(x_1), \ldots, h'(x_n)\} \mid h\right) \geq h(x_0) - y$$

Therefore

$$\begin{aligned}\mathbf{Pr}\left(h'(x_0) < \min\{h'(x_1), \ldots, h'(x_n)\}\right) &\geq \mathbb{E}\left(h(x_0) - y\right) \\ &= \Omega\left(\frac{\log n}{n}\right) = \Omega\left(\frac{\log(n+1)}{n+1}\right)\end{aligned}$$

and h satisfies (3) ■

The lower bound for $k = 2$ seen in Table 1 is shown in the following theorem, using a probabilistic mix between distribution strategies as the main ingredient.

Theorem 4. *For any key set $X = \{x_0, x_1, \ldots, x_n\}$ there exists a random hash function $h : X \to [0,1)$ that is 2-independent such that*

$$\mathbf{Pr}\left(h(x_0) < \min_{i \in \{1,\ldots,n\}} \{h(x_i)\}\right) = \Omega\left(\frac{1}{\sqrt{n}}\right)$$

Proof. Since we are only interested in proving the asymptotic result, and have no intentions of optimizing the constant we can wlog. assume that $10\sqrt{n}$ is an integer that divides n. To shorten notation we let $\ell = 10\sqrt{n}$.

We will now consider four different strategies for assigning h, and they will choose a hash function $g : X \to [\ell + 1]$. Then we let $(U_x)_{x \in X}$ be a family of independent random variables uniformly distributed in $(0,1)$ and define $h(x) = \frac{g(x)+U_x}{\ell+1}$. The high-level approach is to define distribution strategies such that some have too high pair-collision probability, some have too low and likewise for the probability of hashing to the same value as x_0. Then we mix over the strategies with probabilities such that in expectation we get the correct number of collisions but we maintain and increased probability of x_0 hashing to a smaller value than the rest of the keys. We will now describe the four strategies for choosing g.

- **Strategy S_1:** $g(x_0)$ is uniformly chosen. Then $(g(x))_{x \neq x_0}$ is chosen uniformly at random such that $g(x) \neq g(x_0)$ and for each $y \neq g(x_0)$ there are exactly $\frac{n}{\ell}$ hash values equal to y.
- **Strategy S_2:** $g(x_0)$ is uniformly chosen, and y_1 is uniformly chosen such that $y_1 \neq g(x_0)$. For each $x \in X \setminus \{x_0\}$ we define $g(x) = y_1$.
- **Strategy S_3:** $g(x_0)$ is uniformly chosen. Then $Z \subseteq X$ is chosen uniformly at random such that $|Z| = \frac{\sqrt{n}}{5}$. We define $g(z) = g(x_0)$ for every $z \in Z$. Then $(g(x))_{x \neq x_0, x \notin Z}$ is chosen uniformly at random under the constraint that $g(x) \neq g(x_0)$ and for each $y \neq g(x_0)$ there are at most $\frac{n}{\ell}$ hash values equal to y.

– **Strategy** S_4**:** $y \in [\ell + 1]$ is uniformly chosen and $g(x) = y$ for each $x \in X$.

For each of the four strategies we compute the probability that $g(x_0) = g(x)$ and $g(x) = g(x')$ for each $x, x' \in X \setminus \{x_0\}$. Because of symmetry the answer is independent of the choice of x and x'. This is a trivial exercise and the results are summarized in table 3.

Strategy	$\mathbf{Pr}_{S_i}(g(x_0) = g(x))$	$\mathbf{Pr}_{S_i}(g(x) = g(x'))$
S_1	0	$\frac{\frac{n}{\ell}-1}{n-1}\left(< \frac{1}{\ell+1}\right)$
S_2	0	1
S_3	$\frac{1}{5\sqrt{n}}$	$\leq \frac{\frac{n}{\ell}-1}{n-1} + \frac{\frac{\sqrt{n}}{5}\left(\frac{\sqrt{n}}{5}-1\right)}{n(n-1)}\left(< \frac{1}{\ell+1}\right)$
S_4	1	1

Table 3. Strategies for choosing function h and their collision probabilities for $x, x' \in X \setminus \{x_0\}$. The main idea is that there are two strategies with too low probability and two with too high probability, for both types of collisions. However, we can mix probabilistically over the strategies to achieve the theorem.

For event E and strategy S let $\mathbf{Pr}_S(E)$ be the probability of E under strategy S. First we define the strategy T_1 that chooses strategy S_1 with probability p_1 and strategy S_2 with probability $1 - p_1$. We choose p_1 such that $\mathbf{Pr}_{T_1}(g(x) = g(x')) = \frac{1}{\ell+1}$. Then $p_1 > 1 - \frac{1}{\ell+1}$. Likewise we define the strategy T_2 that chooses strategy S_3 with probability p_2 and strategy S_4 with probability $1 - p_2$ such that $\mathbf{Pr}_{T_2}(g(x) = g(x')) = \frac{1}{\ell+1}$. Then $p_2 > 1 - \frac{1}{\ell+1}$ as well. Then:

$$\mathbf{Pr}_{T_1}(g(x) = g(x_0)) = 0 < \frac{1}{\ell+1} < \frac{2}{\ell} = \frac{1}{5\sqrt{n}} \leq \mathbf{Pr}_{T_2}(g(x) = g(x_0))$$

Now we define strategy T^* that chooses strategy T_1 with probability q and T_2 with probability $1 - q$. We choose q such that $\mathbf{Pr}_{T_2}(g(x) = g(x_0)) = \frac{1}{\ell+1}$. Then $q \geq 1 - \frac{\frac{1}{\ell+1}}{\frac{2}{\ell}} \geq \frac{1}{2}$. Hence T^* chooses strategy S_1 with probability $\geq \frac{1}{2}\left(1 - \frac{1}{\ell+1}\right) = \Omega(1)$.

The strategy T^* implies a 2-independent g, since due to the the mix of strategies the pairs of keys collide with the correct probability, that is, the same probability as under full independence. Further, with constant probability $g(x_0)$ is unique. Hence with probability $\Omega\left(\frac{1}{\ell+1}\right) = \Omega\left(\frac{1}{\sqrt{n}}\right)$, $g(x_0) = 0$ and $g(x_0)$ is unique. In this case $h(x_0)$ is the minimum of of all $h(x), x \in X$ which concludes the proof. ■

6 Quicksort

The textbook version of the quicksort algorithm, as explained in [12], is the following. As input we are given a set of n numbers $S = \{x_0, \ldots, x_{n-1}\}$ and we

uniformly at random choose a pivot element x_i. We then compare each element in S with x_i and determine the sets S_1 and S_2 which consist of the elements that are smaller and greater than x_i respectively. Then we recursively call the procedure on S_1 and S_2 and output the sorted sequence S_1 followed by x_i and S_2. For this setting there are to the knowledge of the authors no known bounds under limited independence.

We consider two different settings where our results seen in Table 2 apply.

Setting 1. Firstly, we consider the same setting as in [11]. Let the input again be $S = \{x_0, \ldots, x_{n-1}\}$. The pivot elements are pre-computed the following way: let random variables $Y_1, \ldots, Y_n$ be k-independent and each Y_i is uniform over $[n]$. The ith pivot element is chosen to be x_{Y_i}. Note that the sequence of Y_i's is not always a permutation, hence a "cleanup" phase is necessary afterwards in order to ensure pivots have been performed on all elements.

Setting 2. The second setting we consider is the following. Let $Z = Z_1, \ldots, Z_n$ be a sequence of k-independent random variables that are uniform over the interval $(0, 1)$. Let $\min(j, Z)$ denote the index i of the j'th smallest Z_i. We choose pivot element number j to be $x_{\min(j,Z)}$. Note that the sequence Z here defines a permutation with high probability and so we can simply repeat the random experiments if any Z_i collide.

In this section we show the results of Table 2 in Setting 1. We refer to Appendix A.1 for proofs for Setting 2 and note that the same bounds apply to both settings.

Recall, that we can use the results on min-wise hashing to show upper bounds on the running time. The key to sharpening this analysis is to consider a problem related to that of min-wise hashing. In Lemma 1 we show that for two sets A, B satisfying $|A| \leq |B|$ there are only $O(1)$ pivot elements chosen from A before the first element is chosen from B. We could use a min-wise type of argument to show that a single element $a \in A$ is chosen as a pivot element before the first pivot element is chosen from B with probability at most $\mathcal{O}\left(\frac{\log n}{|B|}\right)$. However, this would only gives us an upper bound of $\mathcal{O}(\log n)$ and not $\mathcal{O}(1)$.

Lemma 1. *Let $h : [n] \to [n]$ be a 4-independent hash function and let $A, B \subseteq [n]$ be disjoint sets such that $|A| \leq |B|$. Let $j \in [n]$ be the smallest value such that $h(j) \in B$, and $j = n$ if no such j exist. Then let C be the number of $i \in [j]$ such that $h(i) \in A$, i.e.*

$$C = |\{i \in [n] \mid h(i) \in A,\ h(0), \ldots, h(i-1) \notin B\}|$$

Then $\mathbb{E}(C) = O(1)$.

Before we prove Lemma 1 we first show how to apply it to guarantee that quicksort only makes $\mathcal{O}(n \log n)$ comparisons.

Theorem 5. *Consider quicksort in Setting 1 where we sort a set $S = \{x_0, \ldots, x_{n-1}\}$ and pivot elements are chosen using a 4-independent hash function. For any i the expected number of times x_i is compared with another element $x_j \in S \setminus \{x_i\}$ when x_j is chosen as a pivot element is $\mathcal{O}(\log n)$. In particular the expected running time is $\mathcal{O}(n \log n)$.*

Proof. Let $\pi : [n] \to [n]$ be a permutation of $[n]$ such that $x_{\pi(0)}, \ldots, x_{\pi(n-1)}$ is sorted ascendingly. Then $\pi \circ h$ is a k-independent function as well, and therefore wlog. we assume that $x_0, \ldots, x_{n-1}$ is sorted ascendingly.

Fix $i \in [n]$ and let $X = \{x_{i+1}, \ldots, x_{n-1}\}$. First we will upper bound the expected number of comparisons x_i makes with elements from X when an element of X is chosen as pivot. We let A_ℓ and B_ℓ be the sets defined by

$$A_\ell = \left\{x_j \mid j \in \left[i, i + 2^{\ell-1}\right) \cap [n]\right\} B_\ell = \left\{x_j \mid j \in \left[i + 2^{\ell-1}, i + 2^\ell\right) \cap [n]\right\}$$

For any $x_j \in A_\ell$, x_j is compared with x_i only if it is chosen as a pivot element before any element of B_ℓ is chosen as a pivot element. By Lemma 1 the expected number of times this happens is $\mathcal{O}(1)$ for a fixed ℓ since $|B_\ell| \geq |A_\ell|$. Since A_ℓ is empty when $\ell > 1 + \log n$ we see that x_i is in msexpectation only compared $\mathcal{O}(\log n)$ times to the elements of X. We use an analogous argument to count the number of comparisons between x_i and $x_0, x_1, \ldots, x_{i-1}$ and so we have that every element makes in expectation $\mathcal{O}(\log n)$ comparisons. As we have n elements it follows directly from linearity of expectation that the total number of comparisons made is in expectation $\mathcal{O}(n \log n)$. The last minor ingredient is the running time of the cleanup phase of Setting 1. We show in Lemma 2 that this uses expected time $\mathcal{O}(n \log n)$ for $k = 2$, hence the stated running time of the theorem follows. ∎

We now show Lemma 1, which was a crucial ingredient in the above proof.

Proof (of Lemma 1). Wlog. assume that $|A| = |B|$ and let m the size of A and B. Let $\alpha = \frac{m}{n}$.

For each non-negative integer $\ell \geq 0$ let $C_\ell = \left\{i \in [n] \mid i < 2^\ell \mid h(i) \in A\right\}$. Let E_ℓ be the event that $h(j) \notin B$ for all $j \in [n]$ such that $j < 2^\ell$. It is now easy to see that if $i \in C$ then for some integer $\ell \leq 1 + \lg n$, $i \in C_\ell$ and $E_{\ell-1}$ occurs. Hence:

$$\mathbb{E}(C) \leq \sum_{\ell=0}^{\lfloor \lg n \rfloor + 1} \mathbb{E}\left(|C_\ell| \cdot [E_{\ell-1}]\right) \tag{4}$$

Now we note that

$$\mathbb{E}\left(|C_\ell|\,[E_{\ell-1}]\right) \leq \mathbb{E}\left(\left(|C_\ell| - \alpha 2^{\ell+1}\right)^+\right) + \mathbb{E}\left(\alpha 2^{\ell+1} \cdot [E_{\ell-1}]\right) \tag{5}$$

where x^+ is defined as $\max\{x, 0\}$.

First we will bound $\mathbb{E}\left(\left(|C_\ell| - \alpha 2^{\ell+1}\right)^+\right)$ when $\alpha 2^\ell \geq 1$. Note that for any $r \in \mathbb{N}$:

$$\mathbf{Pr}\left((|C_\ell| - \alpha 2^{\ell+1})^+ \geq r\right) = \mathbf{Pr}\left(|C_\ell| - \mathbb{E}(|C_\ell|) \geq \alpha 2^\ell + r\right) \tag{6}$$

$$\leq \frac{\mathbb{E}\left(|C_\ell| - E\,|C_\ell|\right)^4}{(\alpha 2^\ell + r)^4} \tag{7}$$

Now consider Facts 6 and 9 which we will use together with (15).

Fact 6 *Let $X = \sum_{i=1}^{n} X_i$ where $X_1, \ldots, X_i$ are k-independent random variables in $[0,1]$ for some even constant $k \geq 2$. Then*

$$\mathbb{E}\left((X - \mathbb{E}X)^k\right) = \mathcal{O}\left((\mathbb{E}X) + (\mathbb{E}X)^{k/2}\right)$$

Fact 7 *Let $r, l \in \mathbb{R}$. It holds that*

$$\sum_{l \geq 1} \frac{1}{(r+l)^4} \leq \frac{1}{r^3}.$$

Proof. We have

$$\sum_{l \geq 1} \frac{1}{(r+l)^4} \leq \int_0^{\infty} \frac{1}{(r+x)^4} \mathrm{d}x = \left[-\frac{1}{3}\frac{1}{(r+x)^3}\right]_0^{\infty} \leq \frac{1}{r^3}.$$

■

Note that whether each element $i \in [n], i < 2^k$ is lies in C_ℓ is only dependent on $h(i)$. Hence $|C_\ell| = \sum_{i \in [n], i < 2^k} [h(i) \in A]$ is the sum of 4-independent variables with with values in $[0,1]$ and hence we can use Fact 6 to give an upper bound on (15). Combining Facts 6 and 9 and (15) we see that:

$$\begin{aligned}
\mathbb{E}\left(\left(|C_\ell| - \alpha 2^{\ell+1}\right)^+\right) &= \sum_{r \geq 1} \mathbf{Pr}\left((|C_\ell| - \alpha 2^{\ell+1})^+ \geq r\right) \\
&\leq \sum_{r \geq 1} \frac{\mathbb{E}\left(|C_\ell| - E\,|C_\ell|\right)^4}{(\alpha 2^\ell + r)^4} \\
&= \mathcal{O}\left(\frac{\left(\alpha 2^\ell\right)^2}{\left(\alpha 2^\ell\right)^3}\right) = \mathcal{O}\left(\frac{1}{\alpha 2^\ell}\right) \qquad (8)
\end{aligned}$$

We we will bound $\mathbb{E}\left(\alpha 2^{\ell+1} \cdot [E_{\ell-1}]\right)$ (the second term of (14)) in a similar fashion still assuming that $\alpha 2^\ell \geq 1$. For each $i \in [n]$ such that $i < 2^{\ell-1}$ let $Z_i = 1$ if $h(i) \in B$ and $Z_i = 0$ otherwise. Let Z be the sum of these 4-independent variables, then E_k is equivalent to $Z = 0$. By Fact 6

$$\mathbb{E}\left([E_{\ell-1}]\right) = \mathbf{Pr}\left(Z = 0\right) \leq \mathbf{Pr}\left(|Z - \mathbb{E}Z| \geq \mathbb{E}Z\right) \leq \frac{E(Z - \mathbb{E}Z)^4}{(\mathbb{E}Z)^4} = \mathcal{O}\left(\frac{1}{(\mathbb{E}Z)^2}\right)$$

Since $\mathbb{E}(Z) = \alpha \left\lceil 2^{\ell-1} \right\rceil$ we see that

$$\alpha 2^{\ell+1} \cdot \mathbb{E}\left([E_k]\right) = \mathcal{O}\left(\frac{1}{\alpha 2^\ell}\right) \qquad (9)$$

By combining (8), (9) and (14) we see that for any ℓ such that $\alpha 2^\ell \geq 1$:

$$\mathbb{E}\left(|C_\ell|\,[E_{\ell-1}]\right) \leq \mathcal{O}\left(\frac{1}{\alpha 2^\ell}\right) \qquad (10)$$

Furthermore, for any ℓ such that $\alpha 2^\ell \leq 1$ we trivially get:

$$\mathbb{E}\left(|C_\ell|\left[E_{\ell-1}\right]\right) \leq \mathbb{E}\left(|C_\ell|\right) \leq 2^\ell \alpha \tag{11}$$

To conclude we combine (4), (10) and (11) and finish the proof

$$\mathbb{E}\left(C\right) \leq \mathcal{O}\left(\sum_{\ell,\alpha 2^\ell \geq 1} \frac{1}{\alpha 2^\ell}\right) + \mathcal{O}\left(\sum_{\ell,\alpha 2^\ell \leq 1} \alpha 2^\ell\right) = \mathcal{O}(1)$$

■

We now show that the cleanup phase as described by Setting 1 takes $\mathcal{O}(n \log n)$ for $k = 2$, which means it makes no difference to asymptotic running time of quicksort.

Lemma 2. *Consider quicksort in Setting 1 where we sort a set $S = \{x_0, \ldots, x_{n-1}\}$ with a 2-independent hash function. The cleanup phase takes $\mathcal{O}\left(n \log n\right)$ time.*

Proof. Assume wlog. that n is a power of 2. For each $\ell \in \{0, 1, \ldots, \lg n\}$ let A_ℓ be the set of dyadic intervals of size 2^ℓ, i.e.

$$A_\ell = \left\{ \left[i2^\ell, (i+1)2^\ell\right) \cap [n] \mid i \in \left[n2^{-\ell}\right] \right\}$$

For any consecutive list of s elements $x_i, \ldots, x_{i+s-1}$ such that none of them are chosen as pivot elements, there exist a dyadic interval I of size $\Omega(s)$ such that none of $x_j, j \in I$ are chosen as pivot elements. Hence we only need to consider the time it takes to sort elements corresponding to dyadic intervals. Let P_ℓ be an upper bound on the probability that no element from $\left[0, 2^\ell\right)$ is chosen as a pivot element. Then the total running time of the cleanup phase is bounded by:

$$\mathcal{O}\left(\sum_{\ell=0}^{\lg n} |A_\ell| \, P_\ell 2^{2\ell}\right) = \mathcal{O}\left(n \sum_{\ell=0}^{\lg n} 2^\ell \mathbf{P}_\ell\right) \tag{12}$$

Fix ℓ and let $X = \sum_{i=0}^{n-1} \left[h(i) \in \left[0, 2^\ell\right)\right]$. Then by $\mathbb{E}(X) = 2^\ell$, so by Markov's inequality

$$\begin{aligned}
\mathbf{Pr}\left(X = 0\right) &\leq \mathbf{Pr}\left((X - \mathbb{E}(X))^2 \geq (\mathbb{E}(X))^2\right) \\
&\leq \frac{\mathbb{E}\left((X - \mathbb{E}(X))^2\right)}{(\mathbb{E}(X))^2} = \mathcal{O}\left(\frac{1}{\mathbb{E}(X)}\right) = \mathcal{O}\left(2^{-\ell}\right)
\end{aligned}$$

Plugging this into (12) shows that the running time is bounded by $\mathcal{O}\left(n \log n\right)$. ■

Finally we show the new 2-independent bound. The argument follows as the 4-independent argument, except with 2nd moment bounds instead of 4th moment bounds.

Theorem 8. *Consider quicksort in Setting 1 where we sort a set* $S = \{x_0, \ldots, x_{n-1}\}$ *and pivot elements are chosen using a* 2*-independent hash function. For any* i *the expected number of times* x_i *is compared with another element* $x_j \in S \setminus \{x_i\}$ *when* x_j *is chosen as a pivot element is* $\mathcal{O}\left(\log^2 n\right)$*. In particular the expected running time is* $\mathcal{O}\left(n \log^2 n\right)$*.*

Proof. The proof for $\mathcal{O}(n \log^2 n)$ expected running time follows from an analogous argument as Theorem 5. The main difference being that the analogous lemma to Lemma 1 yields $\mathbb{E}(C) = \mathcal{O}(\log n)$ instead of $\mathbb{E}(C) = \mathcal{O}(1)$, which implies the stated running time. This is due to the fact that as we have 2-independence we must use the weaker 2nd moment bounds instead of 4th moment bounds as used e.g. in (7). Since the cleanup phase takes time $\mathcal{O}(n \log n)$ time even for $k = 2$ due to Lemma 2 the stated time holds. Otherwise the proof follows analogously and we omit the full argument due to repetetiveness. ■

6.1 Binary planar partitions and randomized treaps

The result for quicksort shown in Theorem 5 has direct implications for two classic randomized algorithms. Both algorithms are explained in common text books, e.g. Motwani-Raghavan.

A straightforward analysis of randomized algorithm[12, Page 12] for construction binary planar bipartitions simply uses min-wise hashing to analyze the expected size of the partition. In the analysis the size of the constructed partition depends on the probability of the event happening that a line segment u comes before a line segment v in the random permutation $u, \ldots, u_i, v$. Using the the min-wise probabilities of Table 1 directly we get the same bounds on the partition size as running times on quicksort using the min-wise analysis. This analysis is tightened through Theorem 5 for both $k = 2$ and $k = 4$.

By an analogous argument, the randomized treap data structure of [12, Page 201] gets using the min-wise bounds expected node depth $\mathcal{O}(\log n)$ when a treap is built over a size n set. Under limited independence using the min-wise analysis, the bounds achieved are then $\{\mathcal{O}(\sqrt{n}), \mathcal{O}(\log^2 n), \mathcal{O}(\log^2 n), \mathcal{O}(\log n)\}$ for $k = \{2, 3, 4, 5\}$ respectively. By Theorem 5 we get $\mathcal{O}(\log^2 n)$ for $k = 2$ and $\mathcal{O}(\log n)$ for $k = 4$.

7 Largest bucket size

We explore the standard case of throwing n balls into n buckets using a random hash function. We are interested in analyzing the bucket that has the largest number of balls mapped to it. Particularly, for this problem our main contribution is an explicit family of hash functions that are k-independent (remember Definition 1) and where the largest bucket size is $\Omega\left(n^{1/k}\right)$. However we start by stating the matching upper bound.

7.1 Upper bound

We will briefly show the upper bound that matches our lower bound presented in the next section. We are unaware of literature that includes the upper bound, but note that it follows from a standard argument and is included for the sake of completeness.

Lemma 3. *Consider the setting where n balls are distributed among n buckets using a random hash function h. For $m = \Omega\left(\frac{\log n}{\log\log n}\right)$ and any $k \in \mathbb{N}$ such that $k < n^{1/k}$ then if h is k-independent the largest bucket size is $\mathcal{O}(m)$ with probability at least $1 - \frac{n}{m^k}$.*

Proof. Consider random variables $B_1, \ldots, B_n$, where B_i denotes the number of balls that are distributed to bin i. By definition, the largest bucket size is $\max_i B_i$ Since $(\max_i B_i)^{\underline{k}} \leq \sum_i (B_i)^{\underline{k}}$ for any threshold t we see that

$$\mathbf{Pr}(\max_i B_i \geq t) = \mathbf{Pr}\left((\max_i B_i)^{\underline{k}} \geq t^{\underline{k}}\right) \leq \mathbf{Pr}\left(\sum_i (B_i)^{\underline{k}} \geq t^{\underline{k}}\right).$$

Since $\sum_i (B_i)^{\underline{k}}$ is exactly the number of ordered k-tuples being assigned to the same bucket we see that $\mathbb{E}\left(\sum_i (B_i)^{\underline{k}}\right) = n^{\underline{k}} \cdot \frac{1}{n^{k-1}}$, because there are exactly $n^{\underline{k}}$ ordered k-tuples. Hence we can apply Markov's inequality

$$\mathbf{Pr}\left(\sum_i (B_i)^{\underline{k}} \geq t^{\underline{k}}\right) \leq \frac{\mathbb{E}\left(\sum_i (B_i)^{\underline{k}}\right)}{t^{\underline{k}}} = \frac{n^{\underline{k}}}{n^k} \cdot \frac{n}{t^{\underline{k}}} \leq \frac{n}{t^{\underline{k}}}.$$

Since $k < n^{1/k}$ implies $k = \mathcal{O}\left(\frac{\log n}{\log\log n}\right)$ we see that $k + m = \Theta(m)$. Letting $t = k + m$ we get the desired upper bound $\frac{n}{m^k}$ on the probability that $\max_i B_i \geq m + k$ since $(m+k)^{\underline{k}} > m^k$. ∎

7.2 Lower bound

At a high level, our hashing scheme is to divide the buckets into sets of size p and in each set polynomial hashing is used on the keys that do not "fill" the set. The crucial point is then to see that for polynomial hashing, the probability that a particular polynomial hashes a set of keys to the same value can be bounded by the probability of all coefficients of the polynomial being zero. Having a bound on this probability, the set size can be picked such that with constant probability the coefficients of one of the polynomials is zero, resulting in a large bucket.

Proof. (of Theorem 1) Fix n, m, and k. We will give a scheme to randomly choose a vector $x = (x_0, \ldots, x_{n-1}) \in [n]^n$ such that the entries are k-independent.

First we choose some prime $p \in \left[\frac{1}{4}m, \frac{1}{2}m\right]$. This is possible by Bertrand's postulate.

Let $t = \left\lfloor \frac{n}{p} \right\rfloor$ and partition $[n]$ into $t+1$ disjoint sets $S_0, S_1, \ldots, S_t$, such that $|S_i| = p$ when $i < t$ and $|S_t| = n - pt = (n \bmod p)$. Note that S_t is empty if p divides n.

The scheme is the following:

- First we pick t polynomial hash function $h_0, h_1, \ldots, h_{t-1} : [p] \to [p]$ of degree k, i.e. $h_i(x) = a_{i,k-1}x^{k-1} + \ldots + a_{i,0} \bmod p$ where $a_{i,j} \in [p]$ is chosen uniformly at random from $[p]$.
- For each x_i we choose which of the events $(x_i \in S_0), \ldots, (x_i \in S_t)$ are true such that $P(x_i \in S_j) = \frac{|S_j|}{n}$. This is done independently for each x_i.
- For each $j = 0, \ldots, t-1$ we let $Y_j = \{x_i \mid x_i \in S_j\}$ be the set of all x_i contained in S_j. If $|Y_j| > p$ we let $Z_j \subseteq Y_j$ be a subset with p elements and $Z_j = Y_j$ otherwise. We write $Z_j = \{x'_0, \ldots, x'_{r-1}\}$ and $S_j = \{s_0, \ldots, s_{p-1}\}$. Then we let $x'_\ell = s_{h_i(\ell)}, \ell \in [r]$. The values for $Y_j \backslash Z_j$ are chosen uniformly in S_j and independently.
- For all x_i such that $(x_i \in S_t)$ we uniformly at random and independently choose $s \in S_t$ such that $x_i = s$.

This scheme is clearly k-independent. The at most p elements in Y_j we distribute using a $k-1$ degree polynomial are distributed k-independently as degree $k-1$ polynomials over p are known to be k-independent (see e.g. [10]). The remaining elements are distributed fully independently.

We can write $|S_i| = \sum_{j=0}^{n-1} [x_j \in S_i]$ and therefore $|S_i|$ is the sum of independent variables from $\{0,1\}$. Since $\mathbb{E}(|S_i|) = p = \omega(1)$ a standard Chernoff bound gives us that

$$\mathbf{Pr}\left(|S_i| \leq \left(1 - \frac{1}{2}\right)p\right) \leq e^{-\Omega(p)} = o(1) \ . \tag{13}$$

For $i \in [t]$ let X_i be 1 if S_i consists of at least $p/2$ elements and 0 otherwise. In other words $X_i = [|S_i| \geq p/2]$. By (13) we see that $\mathbb{E}(X_i) = 1 - o(1)$. Let $X = \sum_{i=0}^{t-1} X_i$. Then $\mathbb{E}(X) = t(1 - o(1))$, so we can apply Markov's inequality to obtain

$$\mathbf{Pr}\left(X \leq \frac{1}{2}t\right) = \mathbf{Pr}\left(t - X \geq \frac{1}{2}t\right) \leq \frac{\mathbb{E}(t-X)}{\frac{1}{2}t} = o(1) \ .$$

So with probability $1 - o(1)$ at least half of the sets $S_i, i \in [t]$ contain at least $p/2$ elements. Assume that this happens after we for every x_i fix the choice of S_j such that $x_i \in S_j$, i.e. assume $X \geq t/2$. Wlog. assume that $S_0, \ldots, S_{\lceil t/2 \rceil - 1}$ contain at least $p/2$ elements. For each $j \in [\lceil t/2 \rceil]$ let Y_j be 1 if h_j is constant and 0 otherwise. That is, $Y_j = [a_{i,k-1} = \ldots = a_{i,1} = 0]$. We note that Y_j is 1 with probability $\frac{1}{p^{k-1}}$. Since $Y_0, \ldots, Y_{\lceil t/2 \rceil - 1}$ are independent we see that

$$\begin{aligned}\mathbf{Pr}\left(Y_0 + \ldots + Y_{\lceil t/2 \rceil - 1} > 0\right) &= 1 - \left(1 - \frac{1}{p^{k-1}}\right)^{\lceil t/2 \rceil} \\ &\geq 1 - e^{-\frac{\lceil t/2 \rceil}{p^{k-1}}} = 1 - e^{-\Theta(n/p^k)}\end{aligned}$$

Since $p \leq m$ we see that $e^{-\Theta(n/p^k)} \leq e^{-\Theta(n/m^k)}$ furthermore $n/m^k \leq 1$ by assumption and so $e^{-\Theta(n/m^k)} = 1 - \Theta\left(\frac{n}{m^k}\right)$. This proves that at least one $h_i, j \in [\lceil t/2 \rceil]$ is constant with probability $\Omega\left(\frac{n}{m^k}\right)$. And if that is the case at least on bucket has size $\geq p/2 = \Omega(m)$. This proves the theorem under the assumption that $X \geq t/2$. Since $X \geq t/2$ happens with probability $1 - o(1)$ this finishes the proof.

■

Since it is well known that using $\mathcal{O}(\log n/\text{loglog}n)$-independent hash function to distribute the balls will imply largest bucket size $\Omega\left(\log n/\log\log n\right)$, Corollary 1 provides the full understanding of the largest bucket size.

Proof. (of Corollary 1) Part (a) follows directly from Theorem 1. Part (b) follows since $k > n^{1/k}$ implies $k > \log n/\log\log n$ and so we apply the $\Omega\left(\log n/\log\log n\right)$ bound from [16]. ■

References

1. Noga Alon, Martin Dietzfelbinger, Peter Bro Miltersen, Erez Petrank, and Gábor Tardos, *Is linear hashing good?*, Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '97, ACM, 1997, pp. 465–474.
2. Andrei Z. Broder, *On the resemblance and containment of documents*, In Compression and Complexity of Sequences (SEQUENCES), 1997, pp. 21–29.
3. Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher, *Min-wise independent permutations*, Journal of Computer and System Sciences **60** (1998), 327–336.
4. J.Lawrence Carter and Mark N. Wegman, *Universal classes of hash functions*, Journal of Computer and System Sciences **18** (1979), no. 2, 143 – 154.
5. T. Christiani and R. Pagh, *Generating k-independent variables in constant time*, Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on, Oct 2014, pp. 196–205.
6. T. Christiani, R. Pagh, and M. Thorup, *From independence to expansion and back again*, Forthcoming, STOC'15, 2015.
7. Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson, *Introduction to algorithms*, 2nd ed., McGraw-Hill Higher Education, 2001.
8. Martin Dietzfelbinger, Torben Hagerup, Jyrki Katajainen, and Martti Penttonen, *A reliable randomized algorithm for the closest-pair problem*, Journal of Algorithms **25** (1997), no. 1, 19 – 51.
9. Piotr Indyk, *A small approximately min-wise independent family of hash functions*, Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms (Philadelphia, PA, USA), SODA '99, Society for Industrial and Applied Mathematics, 1999, pp. 454–456.
10. A. Joffe, *On a set of almost deterministic k-independent random variables*, Ann. Probab. **2** (1974), no. 1, 161–162.
11. Howard Karloff and Prabhakar Raghavan, *Randomized algorithms and pseudorandom numbers*, Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '88, ACM, 1988, pp. 310–321.

12. Rajeev Motwani and Prabhakar Raghavan, *Randomized algorithms*, Cambridge University Press, New York, NY, USA, 1995.
13. Anna Pagh, Rasmus Pagh, and Milan Ruzic, *Linear probing with constant independence*, Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '07, ACM, 2007, pp. 318–327.
14. Mihai Patrascu and Mikkel Thorup, *The power of simple tabulation hashing*, Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '11, ACM, 2011, pp. 1–10.
15. Mihai Pătraşcu and Mikkel Thorup, *On the k-independence required by linear probing and minwise independence*, Automata, Languages and Programming (Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and PaulG. Spirakis, eds.), Lecture Notes in Computer Science, vol. 6198, Springer Berlin Heidelberg, 2010, pp. 715–726 (English).
16. Jeanette P. Schmidt, Alan Siegel, and Aravind Srinivasan, *Chernoff-hoeffding bounds for applications with limited independence*, SIAM J. Discret. Math. **8** (1995), no. 2, 223–250.
17. A. Siegel, *On universal classes of extremely random constant-time hash functions*, SIAM J. Comput. **33** (2004), no. 3, 505–543.
18. M. Thorup, *Simple tabulation, fast expanders, double tabulation, and high independence*, Proc. FOCS'13, 2013, pp. 90–99.
19. Mikkel Thorup, *Even strongly universal hashing is pretty fast*, Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (Philadelphia, PA, USA), SODA '00, Society for Industrial and Applied Mathematics, 2000, pp. 496–497.
20. Mikkel Thorup and Yin Zhang, *Tabulation based 5-universal hashing and linear probing*, ALENEX'10, 2010, pp. 62–76.

A Appendix

A.1 Quicksort in Setting 2

The analog to Lemma 1 that we need in order to prove that quicksort in Setting 2 using a 4-independent hash function runs in expected $\mathcal{O}(n \log n)$ time is proved below.

Lemma 4. *Let $h : X \to (0,1)$ be a 4-independent hash function and $A, B \subseteq X$ disjoint sets such that $|A| \leq |B|$. Then*

$$\mathbb{E}\left(\left|\left\{a \in A \mid h(a) < \min_{b \in B} h(b)\right\}\right|\right) = O(1)$$

Proof. Wlog assume that $|A| = |B| = n$. Let Y be defined by

$$Y = \left\{a \in A \mid h(a) < \min_{b \in B} h(b)\right\}.$$

If $a \in Y$ then either $h(a) < \frac{1}{n}$ or there exists $k \in \mathbb{N}$ such that $h(a) \leq 2^{-k+1}$ and $\min_{b \in B} h(b) \geq 2^{-k}$, where we can choose k such that $2^{-k+1} > \frac{1}{n}$, i.e. $2^k < 2n$.

Let Y_k be the set of all keys $a \in A$ satisfying $h(a) \leq 2^{-k+1}$ and let E_k be the event that $\min_{b \in B} h(b) \geq 2^{-k}$. Also let 1_{E_k} denote the indicator variable defined as being 1 when event E_k occurs and 0 otherwise. Since the expected number of keys in A hashing below $\frac{1}{n}$ is 1 we see that:

$$\mathbb{E}(|Y|) \leq 1 + \sum_{k=1}^{\lfloor \lg n \rfloor + 1} \mathbb{E}(|Y_k| \, 1_{E_k})$$

Now note that:

$$\mathbb{E}(|Y_k| \, 1_{E_k}) \leq \mathbb{E}(|Y_k| - 2^{-k+2}n)^+ + 2^{-k+2}n \cdot \mathbb{E}(1_{E_k}) \tag{14}$$

where x^+ is defined as $\max\{x, 0\}$.

First we will bound $\mathbb{E}(|Y_k| - 2^{-k+2}n)^+$. Note that for any $\ell \in \mathbb{N}$:

$$\begin{aligned}\mathbf{Pr}\left((|Y_k| - 2^{-k+2}n)^+ \geq \ell\right) &= \mathbf{Pr}\left(|Y_k| - \mathbb{E}(|Y_k|) \geq 2^{-k+1}n + \ell\right) \\ &\leq \frac{\mathbb{E}\left(|Y_k| - E\,|Y_k|\right)^4}{(2^{-k+1}n + \ell)^4}\end{aligned} \tag{15}$$

Remember that we consider a 4-independent hash function h. Next we wish to upper bound $\mathbb{E}\left(|Y_k| - \mathbb{E}(|Y_k|)\right)^4$ (the numerator of (15)). Consider indicator variables X_a for all $a \in A$ such that $X_a = 1$ if $a \in Y_k$ and 0 otherwise. By the definition of Y_k we have $|Y_k| = \sum_{a \in A} X_a$ and $\mathbb{E}(\sum_{a \in A} X_a) = \mathcal{O}(2^{-k+1})$.

$$\begin{aligned}\mathbb{E}\left(|Y_k| - \mathbb{E}(|Y_k|)\right)^4 &= \mathbb{E}\left(\sum_{a \in A} X_a - \mathbb{E}(X_a)\right)^4 \\ &= \mathcal{O}\left(n\mathbb{E}(X_a - \mathbb{E}(X_a))^4 + n^2\mathbb{E}((X_a - \mathbb{E}(X_a))^2)^2\right) \\ &= \mathcal{O}\big(\mathbb{E}\big(\sum_{a \in A} X_a\big)^2\big) = \mathcal{O}((2^{-k}n)^2)\end{aligned} \tag{16}$$

Consider now the following fact, which we will use to bound a particular type of sum.

Fact 9 *Let $r, l \in \mathbb{R}$. It holds that*

$$\sum_{l \geq 1} \frac{1}{(r+l)^4} \leq \frac{1}{r^3}.$$

Proof. We have

$$\sum_{l \geq 1} \frac{1}{(r+l)^4} \leq \int_0^\infty \frac{1}{(r+x)^4} \mathrm{d}x = \left[-\frac{1}{3} \frac{1}{(r+x)^3} \right]_0^\infty \leq \frac{1}{r^3}.$$

■

By application of Fact 9 and using our bound from (16) we can finish the upper bound on (15):

$$\begin{aligned}
\mathbb{E}(|Y_k| - 2^{-k+2}n)^+ &= \sum_{\ell \geq 1} \mathbf{Pr}\left((|Y_k| - 2^{-k+2}n)^+ \geq \ell\right) \\
&= \mathcal{O}\left((2^{-k}n)^2 \sum_{\ell \geq 1} \frac{1}{(2^{-k+1}n + \ell)^4} \right) \\
&= \mathcal{O}\left(\frac{1}{2^{-k}n} \right) = \mathcal{O}\left(\frac{2^k}{n} \right)
\end{aligned}$$

We only need to bound $2^{-k+2}n \cdot \mathbb{E}(1_{E_k})$ (the second term of (14)) in order to finish the proof. For each $b \in B$ let $Z_b = 1$ if $h(b) \leq 2^{-k}$ and $Z_b = 0$ otherwise. Then E_k implies that $\sum_{b \in B} Z_b = 0$. Let $Z = \sum_{b \in B} Z_b$. Then by an equivalent argument as used for (16):

$$\mathbb{E}(1_{E_k}) = \mathbf{Pr}\left(Z = 0\right) \leq \mathbf{Pr}\left(|Z - \mathbb{E}Z| \geq \mathbb{E}Z\right) \leq \frac{E(Z - \mathbb{E}Z)^4}{(\mathbb{E}Z)^4} = \mathcal{O}\left(\frac{1}{(\mathbb{E}Z)^2} \right)$$

Since $\mathbb{E}(Z) = 2^{-k}n$ we see that

$$2^{-k+2}n \cdot \mathbb{E}(1_{E_k}) = \mathcal{O}\left(\frac{1}{2^{-k}n} \right) = \mathcal{O}\left(\frac{2^k}{n} \right)$$

To conclude, we insert our bounds on the two terms of (14), which completes the proof.

$$\begin{aligned}
\mathbb{E}(|Y|) &\leq 1 + \sum_{k=1}^{\lfloor \lg n \rfloor + 1} \mathbb{E}(|Y_k| - 2^{-k+2}n)^+ + 2^{-k+2}n \cdot \mathbb{E}(1_{E_k}) \\
&= 1 + \sum_{k=1}^{\lfloor \lg n \rfloor + 1} \mathcal{O}\left(\frac{2^k}{n} \right) = \mathcal{O}(1)
\end{aligned}$$

■