The Power of Two Choices with Simple Tabulation

Søren Dahlgaard, Mathias Bæk Tejs Knudsen, Eva Rotenberg, and Mikkel Thorup

University of Copenhagen, 
{soerend,knudsen,roden,mthorup}@di.ku.dk

Abstract

The power of two choices is a classic paradigm used for assigning \( m \) balls to \( n \) bins. When placing a ball we pick two bins according to some hash functions \( h_0 \) and \( h_1 \), and place the ball in the least full bin. It was shown by Azar et al. [STOC’94] that for \( m = O(n) \) with perfectly random hash functions this scheme yields a maximum load of \( \lg \lg n + O(1) \) with high probability. The two-choice paradigm has many applications in e.g. hash tables and on-line assignment of tasks to servers.

In this paper we investigate the two-choice paradigm using the very efficient simple tabulation hashing scheme. This scheme dates back to Zobrist in 1970, and has since been studied by Pătrașcu and Thorup [STOC’11]. Pătrașcu and Thorup claimed without proof that simple tabulation gives an expected maximum load of \( O(\log \log n) \). We improve their result in two ways. We show that the expected maximum load, when placing \( m = O(n) \) balls into two tables of \( n \) bins, is at most \( \lg \lg n + O(1) \). Furthermore, unlike with fully random hashing, we show that with simple tabulation hashing, the maximum load is not bounded by \( \lg \lg n + O(1) \), or even \((1 + o(1)) \lg \lg n\) with high probability. However, we do show that it is bounded by \( O(\log \log n) \) with high probability, which is only a constant factor worse than the fully random case. Previously, such bounds have required \( \Omega(\log n) \) independent hashing, or other methods that require \( \omega(1) \) computation time.

Our analysis relies on classifying the dependencies between keys when using simple tabulation. We show, as a corollary, that these methods can be used to give good bounds for any constant moment.

*Research partly supported by Thorup’s Advanced Grant from the Danish Council for Independent Research under the Sapere Aude research carrier programme.
†Research partly supported by the FNU project AlgoDisc - Discrete Mathematics, Algorithms, and Data Structures.
1 Introduction

Consider the problem of placing $n$ balls into $n$ bins. If the balls are placed independently and uniformly at random it is well known that the maximum load of any bin is $\Theta(\log n / \log \log n)$ with high probability (whp) [4]. An alternative variant chooses $d$ possible bins per ball independently and uniformly at random, placing the ball in the bin with the lowest load (breaking ties arbitrarily). It was shown by Azar, et al. [1] that with this scheme the maximum load, surprisingly, drops to $\log \log n / \log d + O(1)$ whp. When $d = 2$, this is known as the power of two choices paradigm and is well-studied in the literature [7, 6].

The power of two choices paradigm is useful in several applications. In particular, it can be used to reduce the maximum time required to search a hash table. In the classic hash table by chaining (see e.g. [5]) keys are inserted into a table using a hash function to decide the location and collisions are handled by making a linked list of all keys in the bin. If we insert $n$ keys into a table of size $n$ and the hash function used is perfectly random, then the longest chain has length $\Theta(\log n / \log \log n)$ whp. If we instead use the two-choice paradigm and search both chains pointed to by the two hash functions, then the maximum time to search for a key will be $\Theta(\log \log n)$ whp. Another application where the two-choice paradigm has proven useful is the problem of assigning tasks to servers. In this problem tasks arrive in an online fashion and have to be assigned to a server. We are interested in assigning tasks to the least loaded servers, but in practice it may be expensive to obtain the information of which server has the smallest load. Instead we may query two random servers for their load and assign the task to the least loaded of the two. In this case the maximum load on $n$ servers is only $\lg \lg n + O(1)$ whp. For a survey on the two-choice paradigm and more applications, see [7].

In this paper we consider the two-choice paradigm using the very practical simple tabulation hashing dating back to Zobrist [15]. In simple tabulation hashing, the hash value is computed by looking up $c = O(1)$ characters in tables of size $u^{1/c}$ and XORing the results. Pătraşcu and Thorup [11] have shown that simple tabulation, which is not even 4-independent in the classic notion of Carter and Wegman [2], has many desirable properties such as random graph properties necessary for cuckoo hashing, but with failure probability $O(n^{-1/3})$ opposed to the $O(n^{-1})$ in the fully random case. They also claim, without proof, that their techniques give a $O(\log \log n)$ bound on the expected maximum load when using the two-choice paradigm for assigning keys.
1.1 Our results

In this paper we show that simple tabulation is almost as good fully random hash functions when using the two-choice paradigm. Similar to [11] we consider the bipartite case, where \( h_0 \) and \( h_1 \) hash to different tables. More precisely we show the following two theorems:

**Theorem 1.** Let \( h_0 \) and \( h_1 \) be two independent random simple tabulation hash functions. If \( m = O(n) \) balls are put into two tables of \( n \) bins sequentially using the two-choice paradigm with \( h_0 \) and \( h_1 \), then the expected maximum load is at most \( \lg \lg n + O(1) \).

**Theorem 2.** Let \( h_0 \) and \( h_1 \) be two independent random simple tabulation hash functions. If \( m = O(n) \) balls are put into two tables of \( n \) bins sequentially using the two-choice paradigm with \( h_0 \) and \( h_1 \), then for any constant \( \gamma > 0 \) the maximum load of any bin is \( O(\log \log n) \) with probability \( 1 - n^{-\gamma} \).

Theorem 1 matches the fully random case and improves the bound claimed without proof in [11]. The maximum load in Theorem 2 is within a constant factor from that of fully random hash functions (with the same probability). It was recently shown by Reingold et al. [12] how to guarantee a maximum load of \( O(\log \log n) \) whp. using the hash functions of [3]. These functions use a seed of \( O(\log n \log \log n) \) random bits and can be evaluated in \( O((\log \log n)^2) \) time. Similar bounds can also be obtained using \( \Omega(\log n) \)-independence, which can be obtained efficiently using the hash function of [13]. Simple tabulation is, however, significantly faster than both methods.

In contrast to the positive results, we also show that for any \( k > 0 \) there exists a set of keys such that the maximum load is \( \geq k \lg \lg n \) with probability \( 1 - \Omega(n^{-\gamma}) \) for some \( \gamma > 0 \). This shows that the results are asymptotically tight and that unlike the fully random case, \( \lg \lg n + O(1) \) is not the right bound for the maximum load.

It remains a major open problem what happens for \( m \gg n \) balls, and it does not seem like current techniques alone generalize to this case without the assumption that the hash functions are fully random. We do not know of any practical hash functions that guarantee that the difference between the maximum and the average load is \( \lg \lg n + O(1) \) with high probability. Not even \( \log n \)-independence suffices for this case.

We believe that the techniques employed in our proofs are of independent interest, and provide a new fundamental understanding of simple tabulation. In Section 4 these techniques are employed to show that simple tabulation guarantees good bounds for any constant moment \( k \) generalizing the 4th moment bounds of Pătraşcu and Thorup [11]. Similar to [11] we consider
both the standard moment bounds, and the case when the query depends on the hash value of a key \( q \). The bounds we obtain are within a constant factor of those obtained with \( k \)-independence even though simple tabulation is not even 4-independent. More precisely we show the following two theorems:

**Theorem 3.** Let \( h : [u] \to \mathcal{R} \) be a simple tabulation hash function on \( c \) characters into some output range \( \mathcal{R} \). Let \( X = (x_0, \ldots, x_{m-1}) \) be an \( m \)-tuple of distinct keys from \([u]\). Let \( Y_0, \ldots, Y_{m-1} \) be random variables such that \( Y_i \in [0, 1] \) is a function of \( h(x_i) \) and \( \mathbb{E}[Y_i] = p \) for all \( i \in [m] \). Define \( Y = \sum_{i \in [m]} Y_i \) and \( \mu = \mathbb{E}[Y] = mp \). Then for any constant integer \( k \geq 1 \):

\[
\mathbb{E}[(Y - \mu)^{2k}] \leq O\left(\sum_{j=1}^{k} \mu^j\right),
\]

where the constant in the \( O \)-notation is dependent on \( k \) and \( c \).

**Theorem 4.** Let \( h : [u] \to \mathcal{R} \) be a simple tabulation hash function on \( c \) characters into some output range \( \mathcal{R} \). Let \( X = (x_0, \ldots, x_{m-1}) \) be an \( m \)-tuple of distinct keys from \([u]\), and \( q \in [u] \) a query key. Let \( Y_0, \ldots, Y_{m-1} \) be random variables such that \( Y_i \in [0, 1] \) is a function of \( (h(x_i), h(q)) \) and for all \( r \in \mathcal{R} \), \( \mathbb{E}[Y_i | h(q) = r] = p \) for all \( i \in [m] \). Define \( Y = \sum_{i \in [m]} Y_i \) and \( \mu = \mathbb{E}[Y] = mp \). Then for any constant integer \( k \geq 1 \):

\[
\mathbb{E}[(Y - \mu)^{2k}] \leq O\left(\sum_{j=1}^{k} \mu^j\right),
\]

where the constant in the \( O \)-notation is dependent on \( k \) and \( c \).

## 2 Preliminaries

### 2.1 Simple Tabulation

Let us briefly review simple tabulation hashing. The goal is to hash keys from some universe \([u] = \{0, \ldots, u - 1\}\) into some range \( \mathcal{R} = [2^r] \) (i.e. hash values are \( r \) bit numbers for convenience). In tabulation hashing we view a key \( x \in [u] \) as a vector of \( c > 1 \) characters from the alphabet \( \Sigma = [u^{1/c}] \), i.e. \( x = (x_0, \ldots, x_{c-1}) \in \Sigma^c \). We generally assume \( c \) to be a small constant.

In simple tabulation hashing we initialize \( c \) independent random tables \( T_0, \ldots, T_{c-1} : \Sigma \to \mathcal{R} \). The hash value \( h(x) \) is then computed as

\[
h(x) = \bigoplus_{i \in [c]} T_i[x_i],
\]
where \( \oplus \) denotes the bitwise XOR operation. This is a well known scheme dating back to Zobrist [15]. Simple tabulation is known to be just 3-independent, but it was shown in [11] to have much more powerful properties than this suggests. This includes fourth moment bounds, Chernoff bounds when distributing balls into many bins, and random graph properties necessary for cuckoo hashing.

2.1.1 Notation

We will now recall some of the notation used in [11]. Let \( S \subseteq [u] \) be a set of keys. Denote by \( \pi(S, i) \) the projection of \( S \) on the \( i \)th character, i.e. \( \pi(S, i) = \{ x_i | x \in S \} \). We also use this notation for keys, so \( \pi((x_0, \ldots, x_{c-1}), i) = x_i \). A position character is an element of \([c] \times \Sigma\). Under this definition a key \( x \in [u] \) can be viewed as a set of \( c \) position characters \( \{(0, x_0), \ldots, (c-1, x_{c-1})\} \).

Furthermore we assume that \( h \) is defined on position characters as \( h((i, \alpha)) = T_i[\alpha] \). This definitions extends to sets of position characters in a natural way by taking the XOR over the hash of each position character.

2.1.2 Dependent keys

That simple tabulation is not 4-independent implies that there exists keys \( x_1, x_2, x_3, x_4 \) such that for any choice of \( h \), \( h(x_1) \) is dependent on \( h(x_2), h(x_3), h(x_4) \). However, contrary to e.g. polynomial hashing this is not the case for all 4-tuples. Such key dependences in simple tabulation can be completely classified. We will state this as the following lemma, first observed in [14].

Lemma 1 (Thorup and Zhang). Let \( x_1, \ldots, x_k \) be keys in \([u]^k\). If \( x_1, \ldots, x_k \) are dependent, then there exists an \( I \subseteq \{1, \ldots, k\} \) such that each position character of \( (x_i)_{i \in I} \) appears an even number of times.

Conversely, if each position character of \( x_1, \ldots, x_k \) appears an even number of times, then \( x_1, \ldots, x_k \) are dependent and, for any \( h \),

\[
\bigoplus_{i=1}^{k} h(x_i) = 0.
\]

This means that if a set of keys \( (x_i)_{i \in I} \) has symmetric difference \( \emptyset \), then it is dependent. Throughout the paper, we will denote the symmetric difference between the position characters of \( \{x_i\}_{i \in I} \) as \( \bigoplus_{i \in I} x_i \).

2.2 Two Choices

In the two-choice paradigm we are distributing \( m \) balls (keys) into \( n \) bins. The keys arrive sequentially and we associate with each key \( x \) two random
bins $h_0(x)$ and $h_1(x)$ according to hash functions $h_0$ and $h_1$. When placing
a ball we pick the bin with the fewest balls in it breaking ties arbitrarily. If
$h_0$ and $h_1$ are perfectly random hash functions the maximum load of any bin
is known to be $\log \log n + O(1)$ whp. if $m = O(n)$ [1].

Definition 1. Given hash functions $h_0, h_1$ (as above), let the hash graph
denote the graph with bins as vertices and an edge between $(h_0(x), h_1(x))$
for each $x \in S$.

In this paper we assume that $h_0$ and $h_1$ map to two disjoint tables, and the
graph can thus be assumed to be bipartite. This is a standard assumption,
see e.g. [11], and is actually preferable in the distributed setting. We note
that the proofs can easily be changed such that they also hold when $h_0$ and
$h_1$ map to the same table.

Definition 2. The hash-graph $G_m$ may be decomposed into a series
of nested subgraphs $G_0, G_1, \ldots, G_j, \ldots, G_m$ with edge-set $\emptyset \subseteq \ldots \subseteq \{(h_0(x_i), h_1(x_i))\}_{i \in j} \subseteq \ldots \subseteq \{(h_0(x_i), h_1(x_i))\}_{x_i \in S}$, which we will call the
hash-graph at the time $0, \ldots, m$. Similarly, the load of a vertex at the time
$j$ is well-defined.

Similar to the power of 2 choice hashing, another scheme using two hash
functions is cuckoo hashing [10]. In cuckoo hashing we wish to place each
ball in one of two random locations without collisions. It was shown in [11]
that simple tabulation has the random graph properties necessary for cuckoo
hashing. We will use the following theorem [11, Thm. 5] in our analysis of
two-choice hashing

Theorem 5 (Pătraşcu and Thorup). Any set of $m$ keys can be placed into
two tables of size $n = (1 + \varepsilon)m$ by cuckoo hashing and simple tabulation with
probability $1 - O(n^{-1/3})$.

In other words: In the hash graph for cuckoo hashing, any connected
component contains at most one cycle with probability $1 - O(n^{-1/3})$.

2.3 Graph terminology

The binomial tree $B_0$ of order 0 is a single node. The binomial tree $B_k$ of
order $k$ is a root node, which children are binomial trees of order $B_0, \ldots, B_{k-1}$.
A binomial tree of order $k$ has $2^k$ nodes and height $k$.

The arboricity of a graph $G$ is the minimum number of spanning forests
needed to cover all the edges of the graph. As shown by Nash-Williams [9, 8],
the arboricity of a graph $G$ equals

$$\max \left\{ \left| \frac{|E_s|}{|V_s| - 1} \right| \mid (V_s, E_s) \text{ is a subgraph of } G \right\}$$
3 Classifying dependent keys

As mentioned in Lemma 1 the key dependences in simple tabulation are well understood. It was shown in [11], that for every subset \( X \subseteq [u] \) with \( |X| = n \) there are at most \( O(n^2) \) 4-tuples \((x_1, x_2, x_3, x_4) \in X^4\) such that \( x_4 \) is dependent on \( x_1, x_2, x_3 \). Note that this implies that there are at most \( O(n^2) \) 3-tuples \((x_1, x_2, x_3)\) for which there exists \( x_4 \) such that \( x_1, \ldots, x_4 \) are dependent.

In this section we prove several lemmas about the dependencies of keys, which will be key in proving the main theorems. We believe that these lemmas are of independent interest.

We know from Lemma 1 that if the keys \( x_1, \ldots, x_k \) are dependent, then there exists a non-empty subset \( I \subseteq \{1, \ldots, k\} \) such that

\[
\bigoplus_{i \in I} x_i = \emptyset
\]

A key in our analysis is to count how many of these zero-sums there exists. This is done in the following lemma:

**Lemma 2.** Let \( X \subseteq U \) be a subset with \( n \) elements. The number of \( 2t \)-tuples \((x_1, \ldots, x_{2t}) \in X^{2t}\) such that

\[
x_1 \oplus \cdots \oplus x_{2t} = \emptyset
\]

is at most \( ((2t-1)!!)^c n^t \). (Where \( (2t-1)!! = (2t-1)(2t-3) \cdots 3 \cdot 1 \))

It turns out that it is more useful to prove a generalised version which will also be useful later on:

**Lemma 3.** Let \( A_1, \ldots, A_{2t} \subseteq U \) be subsets of \( U \). The number of \( 2t \)-tuples \((x_1, \ldots, x_{2t}) \in A_1 \times \cdots \times A_{2t}\) such that

\[
x_1 \oplus \cdots \oplus x_{2t} = \emptyset
\]

is at most \( ((2t-1)!!)^c \prod_{i=1}^{2t} \sqrt{|A_i|} \). (Where \( (2t-1)!! = (2t-1)(2t-3) \cdots 3 \cdot 1 \))

**Proof.** Let \((x_1, \ldots, x_{2t})\) be such a \( 2t \)-tuple. Equation (2) implies that the number of times each position character appears is an even number. Hence we can partition \((x_1, \ldots, x_{2t})\) into \( t \) pairs \((x_{i_1}, x_{j_1}), \ldots, (x_{i_t}, x_{j_t})\) such that \( \pi(x_{i_k}, c-1) = \pi(x_{j_k}, c-1) \) for \( k = 1, \ldots, t \). Note that there are \( (2t-1)!! \) ways to partition the elements in such a way.

We now prove the claim by induction on \( c \). First assume that \( c = 1 \). We fix some partition \((x_{i_1}, x_{j_1}), \ldots, (x_{i_t}, x_{j_t})\) and count the number of \( 2t \)-tuples which fulfil \( \pi(x_{i_k}, c-1) = \pi(x_{j_k}, c-1) \) for \( k = 1, \ldots, t \). Since \( c = 1 \) we have
$x_{i_k}, x_{j_k} \in A_{i_k} \cap A_{j_k}$. The number of ways to choose such a 2t-tuple is thus bounded by:

$$\prod_{k=1}^{t} |A_{i_k} \cap A_{j_k}| \leq \prod_{k=1}^{t} \min \{|A_{i_k}|, |A_{j_k}|\} \leq \prod_{k=1}^{t} \sqrt{|A_{i_k}||A_{j_k}|} = \prod_{k=1}^{2t} \sqrt{|A_k|}$$

And since there are $(2t-1)!!$ such partitions the case $c = 1$ is finished.

Now assume that the lemma holds when the keys have $< c$ characters. As before, we fix some partition $(x_{i_1}, x_{j_1}), \ldots, (x_{i_t}, x_{j_t})$ and count the number of 2t-tuples which satisfy $\pi(x_{i_k}, c - 1) = \pi(x_{j_k}, c - 1)$ for all $k = 1, \ldots, t$. Fix the last position character $(a_k, c - 1) = \pi(x_{i_k}, c - 1) = \pi(x_{j_k}, c - 1)$ for $k = 1, \ldots, t$, $a_k \in \Sigma$. The rest of the position characters from $x_{i_k}$ is then from the set

$$A_{i_k}[a_k] = \{x|(a_k, c - 1) | (a_k, c - 1) \in x, x \in A_{i_k}\}$$

By the induction hypothesis the number of ways to choose $x_1, \ldots, x_{2t}$ with this choice of $a_1, \ldots, a_t$ is then at most:

$$( (2t - 1)!!)^{c-1} \prod_{k=1}^{t} \sqrt{|A_{i_k}[a_k]| |A_{j_k}[a_k]|}$$

Summing over all choices of $a_1, \ldots, a_t$ this is bounded by:

$$( (2t - 1)!!)^{c-1} \sum_{a_1, \ldots, a_t \in \Sigma} \prod_{k=1}^{t} \sqrt{|A_{i_k}[a_k]| |A_{j_k}[a_k]|}$$

$$= ( (2t - 1)!!)^{c-1} \prod_{k=1}^{t} \sum_{a_k \in \Sigma} \sqrt{|A_{i_k}[a_k]| |A_{j_k}[a_k]|}$$

$$\leq ( (2t - 1)!!)^{c-1} \prod_{k=1}^{t} \sqrt{\sum_{a_k \in \Sigma} |A_{i_k}[a_k]|} \sqrt{\sum_{a_k \in \Sigma} |A_{j_k}[a_k]|}$$

$$= ( (2t - 1)!!)^{c-1} \prod_{k=1}^{t} \sqrt{|A_{i_k}|} \sqrt{|A_{j_k}|} = ( (2t - 1)!!)^{c-1} \prod_{k=1}^{2t} \sqrt{|A_k|}$$

(3)

Here (3) is an application of Cauchy-Schwartz’s inequality. Since there are $(2t - 1)!!$ such partitions the conclusion follows.

**Lemma 4.** Let $X \subseteq U$ be a subset with $n$ elements and fix $s$ such that $s^c \leq \frac{3}{2}n$. The number of $s$-tuples $(x_1, \ldots, x_s), x_i \in X$ for which there exists $y \in X, y \neq x_1, \ldots, x_s$ such that $h(y)$ is dependent of $h(x_1), \ldots, h(x_s)$ is no more than:

$$s^4 \frac{3^c}{6} n^{s-1} \leq s^{O(1)} n^{s-1}$$
Proof. Since $h(y)$ is dependent of $h(x_1), \ldots, h(x_s)$ there exists a subset $I \subseteq \{1, \ldots, s\}$ such that for all choices of $h$:

$$\bigoplus_{i \in I} x_i = y$$

Fix $|I|$ and note that $|I| \geq 3$ (by 3-independence). There are $\binom{s}{|I|}$ ways to choose $I$. Note that $(x_i)_{i \in \{1, \ldots, s\} \setminus I}$ can be chosen in at most $n^{s-|I|}$ ways and by Lemma 2 $(x_i)_{i \in I}$ can be chosen in at most $((|I|)!!)^{c_n(|I|+1)/2}$ ways. I.e. for a fixed value of $|I|$ an upper bound is:

$$\left(\binom{s}{|I|}\right)^{c_n^{s-|I|/2+1/2}} \cdot \left(\binom{|I|+2}{|I|+2}\right)^{c_n^{s-|I|+2/2+1/2}}$$

(4)

We can show that this upper bound is maximal when $|I| = 3$. Since $|I|$ is odd it suffices to show that the value decreases when $|I|$ increases by 2 as long as $|I| + 2 \leq s$. Consider the following fraction:

$$\frac{\left(\binom{s}{|I|}\right)^{c_n^{s-|I|/2+1/2}}}{\left(\binom{|I|+2}{|I|+2}\right)^{c_n^{s-|I|+2/2+1/2}}} = \frac{(|I|+1)(|I|+2)n}{(s-|I|)(s-|I|-1)(|I|+2)^c} \geq \frac{\frac{4}{3}n}{s^c}$$

By the assumption this fraction is at least 1, and hence the upper bound decreases with $|I|$. Therefore, as $|I|$ grows there are fewer ways to describe $(x_1, \ldots, x_s)$. Since $3 \leq |I| \leq s$, the number of ways to choose $(x_1, \ldots, x_s)$ is upper bounded by:

$$s \cdot \left(\binom{s}{3}\right)^{(3!!)^{c_n^{s-3/2+1/2}}} \leq s \cdot \frac{3^c}{6} n^{s-1}$$

\[\square\]

Lemma 5. Let $X \subseteq U$ be a subset with $n$ elements and fix $s$ such that $s^c \leq \frac{1}{4}n$. The number of $s$-tuples $(x_1, \ldots, x_s), x_i \in X$ for which there exists distinct $y_1, \ldots, y_k \in X \setminus \{x_1, \ldots, x_s\}$, which are dependent on $x_1, \ldots, x_s$ and $k \geq \max \{s-1, 5\}$ is no more than:

$$s^6 \frac{15^c}{120} n^{s-2} + s^6 \frac{9^c}{36} n^{s-2} + s^5 \frac{9^c}{4} n^{s-3/2} \leq s^{O(1)} n^{s-3/2}$$

Proof. For each $j = 1, \ldots, k$ let $I_j \subseteq \{1, \ldots, s\}$ be such that $y_j = \bigoplus_{i \in I_j} x_i$ for all choices of $h$. All the tuples for which $|I_j| > 3$ for some $j$ can be bounded easily using the same idea as in Lemma 4: The upper bound decreases as $|I_j|$ increases and since $|I_j| \geq 5$ we can use (4) to get an upper bound on these $s$-tuples which is:

$$s \cdot \left(\binom{s}{5}\right)^{(5!!)^{c_n^{s-5/2+1/2}}} \leq s^6 \frac{15^c}{120} n^{s-2}$$

8
Now assume that $|I_j| = 3$ for $j = 1, \ldots, k$. Note that the sets $I_j$ must be distinct and since $I_j \subseteq \{1, \ldots, s\}$ and $k \geq \max\{5, s-1\}$ there must exist $j, l \in \{1, \ldots, k\}$ such that $|I_j \cap I_l| \leq 1$.

Case $|I_j \cap I_l| = 0$: In this case the number of possible values for $(x_i)_{i \in I_j}$, $(x_i)_{i \in I_l}$, $I_j$, and $I_l$ is, by Lemma 2, no more than:

$$\binom{s}{3} \binom{3!}{n^2}$$

and the remaining $x_i$’s can be chosen in at most $n^{s-6}$ ways giving an upper bound of:

$$\frac{s^6}{3!} \binom{3!}{n^2}^{s-6} = s^6 \frac{9c}{n^{s-2}}$$

Case $|I_j \cap I_l| = 1$: $I_j$ and $I_l$ can be chosen in $\binom{s}{3} \cdot 3$ ways. By Lemma 2 $(x_i)_{i \in I_j}$ can be chosen in $(3!)^{n^2}$ ways. The number of ways to choose $(x_i)_{i \in I_l}$ once $(x_i)_{i \in I_j}$ is then by Lemma 3 no more than $(3!)^{n^3/2}$ since we choose one of the $A_i$’s to be a singleton. The remaining $x_i$’s can be chosen in at most $n^{s-5}$ ways giving a total upper bound of:

$$\binom{s}{3, 2} \cdot 3 \cdot (3^2 n^2) \cdot (3^3 n^{3/2}) \cdot n^{s-5} \leq s^5 \frac{9c}{4} n^{s-3/2}$$

Which concludes the proof.

4 Constant moment bounds

In this section we will see how the lemmas of Section 3 imply that simple tabulation guarantees good bounds on any constant moment as stated in Theorem 3 and Theorem 4.

Recall the definitions of Theorem 3. Let $k = O(1)$ be fixed. For $i \in [m]$ let $Z_i = Y_i - p$ and define $Z = \sum_{i \in [m]} Z_i$. Note that we want to bound $\mathbb{E}[Z^{2k}]$ and by linearity of expectation this equals:

$$\mathbb{E}[Z^{2k}] = \sum_{r_0, \ldots, r_{2k-1} \in [m]^{2k}} \mathbb{E}[Z_{r_0} \cdots Z_{r_{2k-1}}]$$

Fix some $2k$-tuple $r = (r_0, \ldots, r_{2k-1}) \in [m]^{2k}$ and define $V(r) = \mathbb{E}[Z_{r_0} \cdots Z_{r_{2k-1}}]$. First note if there exists $i \in [2k]$ such that $x_{r_i}$ is independent of $(x_{r_j})_{j \neq i}$ then

$$V(r) = \mathbb{E}[Z_{r_0} \cdots Z_{r_{2k-1}}] = \mathbb{E}[Z_{r_i}] \mathbb{E}\left[\prod_{j \neq i} Z_{r_j}\right] = 0$$

Consider now the following lemma.
Lemma 6. The number of 2k-tuples \( r \) s.t. \( V(r) \neq 0 \) is \( O(m^k) \).

Proof. Fix \( r \in [m]^{2k} \) and let \( T_0, \ldots, T_{s-1} \) be all subsets of \( [2k] \) such that \( \bigoplus_{i \in T_j} x_{r_i} = \emptyset \) for \( j \in [s] \). If \( \bigcup_{j \in [s]} T_j \neq [2k] \) then there exists \( x_{r_i} \) that is independent of \( (x_{r_j})_{j \neq i} \) which implies that \( V(r) = 0 \).

Now for a fixed \( T_0, \ldots, T_{s-1} \) such that \( \bigcup_{j \in [s]} T_j = [2k] \) we want to count the number of ways to choose \( r \in [m]^{2k} \) such that \( \bigoplus_{i \in T_j} x_{r_i} = \emptyset \) for all \( j \in [s] \). The goal is to prove that the number of 2k-tuples is \( O(m^k) \). Since \( k \) is constant, there are \( O(1) \) ways to choose \( T_0, \ldots, T_{s-1} \) this will imply that the number of 2k-tuples \( r \) such that \( V(r) \neq 0 \) is \( O(m^k) \). Let \( A_i = \bigcup_{j < i} T_i \) and \( B_i = T_i \backslash A_i \) for \( i \in [s] \). We will choose \( r \) by choosing \( (x_{r_i})_{i \in A_1} \), then \( (x_{r_i})_{i \in A_2} \), and so on up to \( (x_{r_i})_{i \in A_{s-1}} \). When we choose \( (x_{r_i})_{i \in A_j} \) we have already chosen \( (x_{r_i})_{i \in B_j} \) and by Lemma 3 the number of ways to choose \( (x_{r_i})_{i \in A_j} \) is bounded by:

\[
(|T_j| - 1)!! m^{|A_j|/2} = O \left( m^{|A_j|/2} \right)
\]

Since \( s = O(1) \) we see that the number of ways to choose \( (x_{r_i})_{i \in [2k]} \) and hence \( r \in [m]^{2k} \) is \( O(m^k) \) if we require that \( V(r) \neq 0 \). \( \Box \)

We note that since \( |V(r)| \leq 1 \) this already proves that:

\[
E[Z^{2k}] \leq O(m^k)
\]

For any \( r \in [m]^{2k} \) let \( f(r) \) denote the size of the largest subset \( U \subseteq [2k] \) of independent keys \( (x_{r_i})_{i \in U} \). Then:

\[
E \left[ \prod_{i \in [2k]} Z_{r_i} \right] \leq E \left[ \prod_{i \in [2k]} Z_{r_i} \right] \leq E \left[ \prod_{i \in U} Z_{r_i} \right] \leq O \left( p^{f(r)} \right)
\]

Now we fix \( s \in \{1, \ldots, 2k\} \) and count the number of \( r \in [m]^{2k} \) such that \( f(r) = s \). We can bound this by first choosing the \( s \) independent keys of \( U \) in at most \( m^s \) ways, and for each of these choices we can pick the dependent keys in at most \( O(1) \) ways since each dependent key can be written as the sum of a subset of keys from \( (x_{r_i})_{i \in U} \). Since \( s \leq 2k \) is a constant this can be done in at most \( O(1) \) ways. Thus for each \( s \in \{1, \ldots, 2k\} \) there are at most \( O(m^s) \) such sets \( r \in [m]^{2k} \).

Now consider the \( O(m^k) \) 2k-tuples \( r \in [m]^{2k} \) such that \( V(r) \neq 0 \). For each \( s \in \{1, \ldots, 2k\} \) there is \( O(m^\min(k,s)) \) ways to choose \( r \) such that \( f(r) = s \). All these choices of \( r \) satisfy \( V(r) \leq O(p^s) \). Hence:

\[
E[Z^{2k}] = \sum_{r \in [m]^{2k}} V(r) \leq \sum_{s=1}^{2k} O(m^\min(k,s)) \cdot O(p^s) = O \left( \sum_{s=1}^{k} (pm)^s \right).
\]
This finishes the proof of Theorem 3. A similar argument can be used to show Theorem 4.

5 Bounding the expected maximum load

This section is dedicated to proving Theorem 1. The main idea is to bound the probability that a big binomial tree (cf. Section 2.3) appears in the hash graph using the results of Section 3. A crucial point of the proof is to consider a subtree of the binomial tree which is chosen such that the number of leaves are much larger than the number of internal nodes.

Proof of Theorem 1. First of all note that by Theorem 2, the probability that the maximum load is more than \(k_0 \cdot \lg \lg n\) is \(O(n^{-1})\) for some constant \(k_0 > 1\). Hence it suffices to prove that the probability that the maximum load is larger than \(\lg \lg n\) is at most \(O(p \lg \lg n)\) for some constant \(r\) depending on \(m/n\) and \(c\).

Observation 1. If there exists a bin with load at least \(k + 1\) then either there is a component with more edges than nodes or the binomial tree \(B_k\) is a subgraph of the hash graph (see Definition 1).

Proof. Assume no component has more edges than nodes. Then, removing at most one edge from each component yields a forest. One edge per component will at most increase the load by 1, so consider the remaining forest. The statement is seen by induction over time (see Definition 2), where the induction hypothesis \(I_{j-1}\) is that the statement holds for all \(k\) for the graph \(G_j\), in such a way that the node corresponding to the bin with a load of \(k + 1\) is the root of the binomial tree \(B_k\).

The induction start is trivial as \(G_0\) has a max load of 0. Assume the statement holds for \(G_0 \ldots G_{j-1}\). Consider now \(G_j\) and assume that a bin \(v\) has load \(l\) in \(G_j\). Then, for each \(1 \leq i < l\) there exists an edge \((v, u_i)\) in \(G_j\), where \(u_i\) had load \(\geq i\) when the edge was added. By our induction hypothesis \(u_i\) is the root of \(B_i\), and since the graph is acyclic none of the \(B_i\)'s share an edge.

If \(m(1 + \varepsilon) < n\) we know from Theorem 5 that no component of the hash graph contains a double cycle with probability \(O(n^{-1/3})\). Looking into the proof we see that there doesn’t exist a double cycle consisting of at most \(s\) edges with probability \((O(m/n))^s n^{-1/3}\) even when \(m > n\). In the terminology of [11] \(\lg(n/m)\) bits per edge is saved in the encoding of the hash-values. But when \(\lg(n/m) < 0\) we add \(\lg(m/n)\) bits to the encoding instead. If the double cycle consist of \(s\) edges this is \(s \lg(m/n)\) extra bits
in the encoding, i.e. that the bound on the probability is multiplied with \((O(m/n))^s\). This means that we only need to bound the probability that there exists a binomial tree \(B_k, k = [\log \log n + r]\), because any bin with load \(k+1\) will either imply the existence of \(B_k\) in the hash graph or the existence of a double cycle consisting of \(\leq 4k = O(\log \log n)\) edges, and the latter happens with probability \((\log n)^{O(1)} n^{-1/3} = O(n^{-1/4})\).

Say that the hash graph contains a binomial tree \(B_k\). Consider the subtree \(T_{k,d}\) defined by removing the children of all nodes that have less than \(d\) children, where \(d \leq k\) is some constant to be defined (see Figure 1). Note that \(T_{k,d}\) has \(O(d^k)\) nodes, because any bin with load \(k+1\) will either imply the existence of \(B_k\) in the hash graph or the existence of a double cycle consisting of \(\leq 4k = O(\log \log n)\) edges, and the latter happens with probability \((\log n)^{O(1)} n^{-1/3} = O(n^{-1/4})\).

A visualization of the set \(S\) can be seen in Figure 1. We will think of \(S\) as a set of edges, but also as a set of independent keys. The idea is to bound the probability that we could find such a set \(S\). We will split the proof into four cases depending on \(S\), and each will end with a \(\triangleright\).

**Case 1:** \(s := |S| = (d + 1)2^{k-d} - 1\): In this case every edge of the tree is independent, and there are at most \(m^s\) different ways to choose the ordered set \(S\). Note that there are \(2^{k-d}\) groups of \(d\) leaves which have the same parent. The set \(S\) corresponds to the same subgraph of the hash graph regardless of the ordering of these leaves. Since we only want to bound the probability that we can find such a set \(S\), we can thus chose the edges of \(S\) in at most \(m^s (\frac{1}{d!})^{2^{k-d}}\) ways. For a given choice of \(S\) there are \(s - 1\) equations \(h_k(x) = h_k(y)\) which must be fulfilled where \(k \in \{1, 2\}\) and \(x, y\) are keys in \(S\). Since the keys in \(S\) are independent, the probability that this happens for a given \(S\) is at most \(2n^{-(s-1)}\). By a union bound on all the choices of \(S\) the probability that such an \(S\) exists is at most:

\[
m^s \left( \frac{1}{d!} \right)^{2^{k-d}} (2n^{-(s-1)}) \leq 2m^s \left( \frac{1}{d!} \right)^{2^{k-d}(d+1)} n^{-(s-1)} \leq 2n \cdot \left( \frac{m}{n d!} \right)^s
\]

We assume that \(d\) and \(r\) are chosen such that \(\frac{m}{n d!} < \frac{1}{2}\) and \(r \geq d + 1\). Then \(s \geq 2\log n\) and the probability is bounded by \(2n^{-1}\). \(\triangleright\)
Figure 1: Example of $T_{5,2}$ and the corresponding set $S$. The dashed edges correspond to key dependencies at the time the edge is considered in the order. This example would correspond to case 2.

**Case 2:** All the edges incident to the root lie in $S$: Let $S'$ be defined in a similar manner as $S$: Order the edges in increasing distance from the root and on each level from left to right as before. Traverse the edges in this order, and add the edges to $S'$ if the corresponding key is independent of the keys in $S$. However, stop this traversal the first time a dependent key occurs. In this way $S'$ will be an ordered subset of $S$ and the tree-structure will only depend on $s' = |S'|$. Fix this value $s'$. Since there is a key which is dependent on the keys in $S'$ there are at most $s'^{O(1)} m^{s' - 1}$ ways to choose $S'$ by Lemma 4 assuming that $s'^c \leq \frac{4}{5} m$, i.e. assuming that $n$ is larger than some constant depending on $c$.

Every internal node of $T_{k,d}$ has exactly $d$ children that are leaves. Therefore, there can be at most one node in $S'$ having less than $d$ children that are leaves and belong to $S'$. Let $v_1, \ldots, v_l$ denote the internal nodes in $S'$, where $l$ is the number of internal nodes. Let $w_i$ denote the number of children of $v_i$ that are leaves. Similar to case 1, the structure of $S'$ is independent of the order of the leaves with the same parent. Therefore $S'$ can be chosen in at most $s'^{O(1)} m^{s' - 1} \prod_{i=1}^l \frac{1}{w_i}$ ways. Since $w_i! \geq \left( \frac{w_i}{e} \right)^{w_i}$ we see that:

$$\prod_{i=1}^l \frac{1}{w_i!} \leq \prod_{i=1}^l \left( \frac{e}{w_i} \right)^{w_i}$$

Letting $w = \sum_{i=1}^l w_i$ the concavity of $x \to x \log(e/x)$ combined with Jensen’s
inequality yields:

\[ \prod_{i=1}^{l} \left( \frac{e}{w_i} \right)^{w_i} \leq \left( \frac{le}{w} \right)^{w} \]

At most one of the \( w_i \)'s can be smaller than \( d \), so wlog. assume that \( w_1, \ldots, w_{l-1} \geq d \). The total number of nodes must be at least \( l + d(l - 1) \), i.e. \( s' \geq l + d(l - 1) \) giving \( l \leq \frac{s' + d}{d+1} \). Since \( l + w = s' \) we see that:

\[ \frac{l}{w} \leq \frac{s' + d}{d+1} = \frac{1}{d} \frac{s' + d}{s' - 1} \leq \frac{2}{d} \]

Where the last inequality holds assuming that \( n \) (and hence \( s' \geq \lg \lg n \)) is larger than a constant. Since \( w \geq (s - 1) \frac{d}{d+1} \) we see that:

\[ \prod_{i=1}^{l} \frac{1}{w_i!} \leq \left( \frac{2e}{d} \right)^{\frac{d}{d+1}} \]

Assume that \( d \) is chosen such that \( \left( \frac{2e}{d} \right)^{\frac{d}{d+1}} \leq \frac{n}{2m} \). The number of cases that we need to consider is then at most:

\[ s'^{O(1)} m^{s' - 1} \prod_{i=1}^{l} \frac{1}{w_i!} \leq s'^{O(1)} \left( \frac{n}{2} \right)^{s' - 1} \]

Since \( S' \) is a tree there are \( s' - 1 \) equalities on the form \( h_k(x) = h_k(y) \) where \( k \in \{1, 2\} \), \( x, y \in S' \) that must be satisfied if \( S' \) occurs. Since we know the tree structure from knowing \( s' \) there are at most two ways two choose these equalities. This means that the probability that a specific \( S' \) occurs is bounded by \( 2n^{-(s' - 1)} \). For a fixed \( |S'| = s' \) the probability that there exists \( S' \) with \( s' \) elements is therefore bounded by:

\[ 2 s'^{O(1)} \left( \frac{n}{2} \right)^{s' - 1} n^{-(s' - 1)} = 2 s'^{O(1)} 2^{-s' + 1} \]

A union bound over all \( s' \geq \lg \lg n \) now yields the desired upper bound:

\[ \sum_{s' \geq \lg \lg n} 2 s'^{O(1)} 2^{-s' + 1} \leq 2^{-\lg \lg n + 3} \sum_{k \geq 1} (k + \lceil \lg \lg n \rceil - 1)^{O(1)} 2^{-k} \]

\[ \leq \frac{8}{\lg n} \lceil \lg \lg n \rceil^{O(1)} \sum_{k \geq 1} k^{O(1)} 2^{-k} \]

\[ = \frac{(\lg \lg n)^{O(1)}}{\lg n} \]

\( \diamond \)
Case 3: Not all, but at least \((\lg \lg n)/2\) edges incident to the root lie in \(S\): Let \(S' \subseteq S\) be the set of independent keys adjacent to the root, and set \(s' = |S'|\). By Lemma 4, \(S'\) can be chosen in no more than \(s^{O(1)} m^{s'-1} / s'!\) ways since there must exist a key (corresponding to an edge incident to the root) which is dependent on the keys in \(S'\) and the order of the keys are irrelevant. Since all the keys in \(S'\) are independent, the probability that \(h_0(x)\) or \(h_1(x)\) are the same for all the keys \(x \in S'\) is at most \(2n^{-s' - 1}\). So the probability that such a \(S'\) can be found is at most:

\[
\frac{s^{O(1)} m^{s'-1}}{s'!} \cdot (2n^{-s' - 1}) = 2s^{O(1)} \left( \frac{m}{n} \right)^{s'-1} \leq 2s^{O(1)} \left( \frac{me}{ns} \right)^{s'-1} = O((\lg \lg n)^{-1})
\]




Case 4: There are less than \((\lg \lg n)/2\) edges incident to the root in \(S\): Let \(S' \subseteq S\) be the set of keys corresponding to the edges from \(S\) incident to the root and let \(s' = |S'|\). Since the other keys incident to the root must be dependent on the keys from \(S'\), Lemma 5 states that \(S'\) can be chosen in at most \(s^{O(1)} m^{s'-3/2}\) ways. Since all the keys in \(S'\) are independent the probability that \(h_0(x)\) or \(h_1(x)\) are the same for all the keys \(x \in S'\) is at most \(2n^{-(s'-1)}\). Thus, the probability of such a set \(S'\) occurring is bounded by:

\[
s^{O(1)} m^{s'-3/2} \cdot (2n^{-(s'-1)}) \leq s^{O(1)} 2n^{-1/2} \left( \frac{m}{n} \right)^{s'-3/2} = (\log n)^{O(1)} n^{-1/2}
\]

Consider the case of distributing \(m\) balls into \(n\) bins. Note that the proof actually gives an expected maximum load of \(O(m/n) + \lg \lg n + O(1)\) if \(m/n = o((\lg n)/((\lg \lg n)))\). However, this only matches the behaviour of truly random hash functions under the assumption that \(m = O(n)\).

The same techniques can be used to show that \(\Omega \left( \frac{m}{n} \log n \right)\)-independent hash functions yield a maximum load of \(O(m/n) + \lg \lg n + O(1)\) with high probability (this is essentially case 1 in the proof). This implies that \(\Omega (\lg n)\)-independence hashing is sufficient to give the same theoretical guarantees as truly random hash functions in the context of the power of two choices when \(m = O(n)\).

6 \(O(\lg \lg n)\) whp

This section is dedicated to proving Theorem 2. The main idea of the proof is to show, that a hash graph resulting in high load must either have a huge
component, or a component with high arboricity. We show that both cases are very unlikely.

As a negative result, we will first observe, that we cannot prove that the maximum load is $\lg \lg n + O(1)$ or even $(1 + o(1)) \lg \lg n$ whp. when using simple tabulation.

**Observation 2.** Given $k = O(1)$, there exists an ordered set $S$ consisting of $n$ keys, such that when they are distributed into $n$ bins using hash values from simple tabulation the max load is $\geq \left \lfloor k^{c-1}/2 \right \rfloor \lg \lg n - O(1)$ with probability $\Omega(n^{-2(k-1)(c-1)})$.

**Proof.** Consider now the set of keys $[n/k^{c-1}] \times [k]^{c-1}$ consisting of $n$ keys. For each of the positions $i = 1, \ldots, c - 1$ the probability that all the position characters on position $i$ hash to the same value is $n^{-k+1}$. So with probability $n^{-(k-1)(c-1)}$ this happens for all positions $i = 1, \ldots, c - 1$. This happens for both hash functions with probability $n^{-2(k-1)(c-1)}$. In this case $h_l(x) = h_l(x_0x_1 \ldots x_{c-1}) = h_l(x_{c-1}) \oplus h_l(x_0 \ldots x_{c-2})$ is only dependent on $h_l(x_{c-1}), l \in \{0, 1\}$. Order the keys lexicographically and insert them into the bins. If $n/k^{c-1} = \Omega(n)$ balls are distributed independently and uniformly at random to $n$ bins the maximum load would be $\lg \lg n - O(1)$ with probability $\Omega(1)$. (This can be proved along the lines of [1, Thm. 3.2].) If we had exactly $2 \left \lfloor k^{c-1}/2 \right \rfloor$ copies of independent and random keys the maximum load would be at least $\left \lfloor k^{c-1}/2 \right \rfloor$ times larger than if we had had $n/k^{c-1}$ independent and random keys. The latter is at least $\lg \lg n - O(1)$ with probability $\Omega(1)$.

Since there are $k^{c-1} \geq 2 \left \lfloor k^{c-1}/2 \right \rfloor$ copies of independent and uniformly random hash values we conclude that the maximum load is at least $\left \lfloor k^{c-1}/2 \right \rfloor (\lg \lg n - O(1)) = \left \lfloor k^{c-1}/2 \right \rfloor \lg \lg n - O(1)$ with probability $\Omega(1)$ under the assumption that $h_l(x_0 \ldots x_{c-2})$ is constant for any $(x_0, \ldots, x_{c-2}) \in [k]^{c-1}, l \in \{0, 1\}$. Since the latter happens with probability $1/n^{2(k-1)(c-1)}$ the proof is finished.

We will now show that a series of insertions inducing a hash graph with low arboricity and small components cannot cause a too big maximum load. Note that this is the case for any hash functions and not just simple tabulation.

**Lemma 7.** Consider the process of placing some balls into bins with the two choice paradigm, and assume that some bin gets load $k$. Then there exists a connected component in the corresponding hash graph with $x$ nodes and arboricity $a$ such that:

$$a \lg x \geq k$$

16
Proof. Let \( v \) be the node in the hash graph corresponding to the bin with load \( k \). Let \( V_k = \{v\}, E_k = \emptyset \) and define \( V_l, E_l \) for \( l, 0 \leq l < k \) in the following way: For each bin of \( V_{l+1} \), add the edge corresponding to the \( l + 1 \)th ball landing in the bin to the set \( E_l \). Define \( V_l \) to be the endpoints of the edges in \( E_l \) (see Figure 2 for a visualization).

Figure 2: A visualisation of the sets \( V_0, \ldots, V_k \).

It is clear, that each bin of \( V_l \) must have a load of at least \( l \). Note that the definition implies that \( |E_l| = |V_{l+1}| \) and \( V_k \subseteq V_{k-1} \subseteq \ldots \subseteq V_0 \). For each \( l \in [k] \) consider the subgraph \( (V_l, E_l \cup E_{l+1} \cup \ldots \cup E_{k-1}) \) and let \( a_l \) be defined as the following lower bound on the arboricity of this subgraph:

\[
a_l = \left\lfloor \frac{|E_l| + \ldots + |E_{k-1}|}{|V_l| - 1} \right\rfloor
\]

Let \( a = \max_{l \in [k]} a_l \), then \( a \) is a lower bound on the arboricity of \( (V_0, E_0 \cup \ldots \cup E_{k-1}) \). Now note that for each \( l \in [k] \):

\[
\frac{|E_l| + \ldots + |E_{k-1}|}{|V_l| - 1} \leq a
\]

Since \( |E_l| = |V_{l+1}| \) for each \( l \in [k] \) this means that:

\[
|V_l| - 1 \geq \frac{|V_{l+1}| + \ldots + |V_k|}{a}
\]

By an easy induction \( |V_l| \geq (1 + \frac{1}{a})^{k-l} \), and therefore \( |V_0| \geq (1 + \frac{1}{a})^k \). The connected component that contains \( v \) contains at least \( |V_0| \) nodes, has arboricity \( \geq a \), and:

\[
a \log |V_0| \geq a \log \left(1 + \frac{1}{a}\right)^k = k \log \left(1 + \frac{1}{a}\right)^a \geq k
\]

In order to show that components cannot be too large or have too big arboricity, we will need to generalize some of the lemmas from Section 3. We will need the following combinatorial lemma.

17
Lemma 8. Let $s, k, c \geq 1$ be integers and $A_1, \ldots, A_{(2k)^c s + 1}$ be non-empty subsets of $\{1, \ldots, s\}$, such that for every $B \subseteq \{1, \ldots, s\}$:

$$|\{A_i \mid A_i \subseteq B\}| \leq |B|^c$$

Then there exists $I \subseteq \{1, \ldots, (2k)^c s + 1\}$ such that $|I| \leq k$ and

$$f(I) \overset{\text{def}}{=} \left| \bigcup_{i \in I} A_i \right| - |I| \geq k$$

Proof. Let $I \subseteq \{1, \ldots, (2k)^c s + 1\}$ be such that $|\cup_{i \in I} A_i| < 2k$. We want to show that there exists $J = I \cup \{r\}$ for some $r \in \{1, \ldots, (2k)^c s + 1\}$ such that $f(J) > f(I)$. Let $A = \cup_{i \in I} A_i$ and assume for the sake of contradiction that no such $r$ exists. This implies that $|A_r \setminus A| \leq 1$ for all $r \in \{1, \ldots, (2k)^c s + 1\}$. I.e. that each $A_r$ is contained in one of the sets

$$(A \cup \{1\}), (A \cup \{2\}), \ldots, (A \cup \{s\})$$

By assumption, each of these sets contains no more than $(|A| + 1)^c s$ sets $A_r$, and thus they contain at most $(|A| + 1)^c s$ sets combined. This means that

$$(2k)^c s + 1 \leq (|A| + 1)^c s \leq (2k)^c s,$$

which is a contradiction. Thus there must exists an $r$ such that $f(I \cup \{r\}) > f(I)$.

Now consider the following greedy algorithm: Let $I := \emptyset$ and iteratively set $I := I \cup \{r\}$ for such an $r$ until $|\cup_{i \in I} A_i| \geq 2k$. Since $f(I)$ increases in each step, the algorithm stops after at most $k$ steps. This implies that $f(I) \geq 2k - k = k$ and $|I| \leq k$ as desired. \hfill \Box

We can use Lemma 8 to show a more general version of Lemma 5.

Lemma 9. Let $X \subseteq U$ be a subset with $n$ elements and fix $k = O(1)$ and $s$ such that $ks^2 c \leq \sqrt{n}$. The number of $s$-tuples $(x_1, \ldots, x_s) \in X^s$ for which there exists distinct $y_1, \ldots, y_{(2k)^c s + 1} \in X$, which are dependent on $x_1, \ldots, x_s$ is no more than:

$$n^{s-k/2} O(1)$$

where the constant in the $O$-notation is dependent on $k$.

Proof. For each $i \in \{1, \ldots, (2k)^c s + 1\}$ let $A_i \subseteq \{1, \ldots, s\}$ be such that:

$$\bigoplus_{j \in A_i} x_j = y_i$$

By Lemma 8 there exists $I \subseteq \{1, \ldots, (2k)^c + 1\}$ such that for $A := \cup_{i \in I} A_i$, $|A| - |I| \geq k$, $|I| \leq k$. 18
It is enough to show the lemma for a fixed $|A|$ and $|I|$ as these can be chosen in at most $ks = O(s)$. Fix $|A| = a$ and $|I| = r$.

Let $I = \{v_1, \ldots, v_r\}$ and for each $j \in \{1, \ldots, r\}$ define $B_j$ as:

$$B_j = A_{v_j} \setminus \left( \bigcup_{i < j} A_{v_i} \right)$$

Wlog. assume that $a = \sum_{j < r} |B_j| \leq 2k$. (Otherwise there exists a smaller set $I$) The number of ways to choose $(B_j)_{1 \leq j \leq r}$ is at most \binom{s}{a} r^a: There are \binom{s}{a} ways to choose $A$ and $r^a$ ways to partition $A$ into $B_1, \ldots, B_r$.

Now, fix the choice of $B_1, \ldots, B_r$. We will bound the number of ways to choose $(x_i)_{i \in B_j}$ given that $(x_i)_{i \in B_1}, \ldots, (x_i)_{i \in B_{j-1}}$ are chosen. The number of ways to choose $A_j$ is at most $2^{2k}$ for $j \in I$. For a fixed choice of $A_j$ the number of ways to choose $(x_i)_{i \in B_j}$ is at most $\binom{|A_j|!}{r} r^{(|B_j|+1)/2}$ by Lemma 3. Hence, the number of ways to choose $(x_i)_{i \in A}$ is at most:

$$\prod_{j=1}^{r} \left( 2^{2k}(|A_j|!) r^{(|B_j|+1)/2} \right) \leq 2^{2kr} (a!) r^{c n^{(a+r)/2}} \leq 2^{2k^2} (a!) r^{kc n^{(a+r)/2}}$$

The number of ways to choose the remaining $(x_i)_{i \notin A}$ is trivially bounded by $n^{s-a}$ giving a total upper bound on the number of ways to choose $(x_i)_{i \in \{1, \ldots, s\}}$ of:

$$\binom{s}{a} k^a 2^{2k^2} (a!) r^{kc n^{s-a/2+r/2}}$$

Now note that if $a < s$:

$$\frac{(s-a+1) k^{s-a+1} 2^{2k^2} ((a+1)!) r^{kc n^{s-(a+1)/2+r/2}}}{(s-a) k^a 2^{2k^2} (a!) r^{kc n^{s-a/2+r/2}}} = \frac{(s-a) k(a+1)^k c}{(a+1)n^{1/2}} < 1$$

This implies that the upper bound is biggest when $a$ is smallest, i.e. when $a = r + k$. In this case the upper bound is:

$$\binom{s}{k} k^k 2^{2k^2} ((r+k)!) r^{kc n^{s-k/2}} \leq \binom{s}{k} k^k 2^{2k^2} ((2k)!) r^{kc n^{s-k/2}} = s^{O(1)} n^{s-k/2}$$

which concludes the proof. \qed

**Lemma 10.** Let $X \subseteq [u]$ with $|X| = m$, and let $h_0, h_1 : [u] \to [n]$ be two independent simple tabulation hash functions. Fix some integer $k$. If $m < n/(2^8 (4k)^c)$, then the maximum load of any bin when assigning keys using the two-choice paradigm is $O(\log \log n)$ with probability $1 - O(n^{-k+2})$. 

19
Proof. Fix the hash values of all the keys and consider the hash graph. Note that there is a one-to-one correspondence between the edges and the keys and we will not distinguish between the two in this proof. Consider any connected subgraph $C$ in the hash graph. We wish to argue that $C$ cannot be too big or have too high arboricity. In order to do this, we construct a set $S$ of independent edges contained in $C$. Initially let $S = \{e\}$ for some edge in $e$ in $C$. At all times we maintain the set $Y = Y(S)$ of keys which are dependent on the keys in $S$. Note that $S \subseteq Y$. The set $S$ is constructed iteratively in the following way: If there exists an edge $e \in C \setminus Y$ that is incident to an edge in $S$ add $e$ to $S$. Otherwise, if there exists an edge $e \in C \setminus Y$, which is incident to an edge in $Y$, add $e$ to $S$. If neither type of edge exists we do not add more edges to $S$. Note that in this case $C = Y$.

At any point we can partition the edges of $S$ into connected components $C_1, \ldots, C_t$, such that $C_1$ is the component of the initial edge of $S$. For each $i > 1$ we let $b_i \in Y \setminus S$ be an edge incident to $C_i$. Order the components $C_2, \ldots, C_t$ such that $b_2 < \ldots < b_t$. For a visualisation of $S$ Figure 3 can be consulted.

![Figure 3: A visualization of the process. $C_1, \ldots, C_t$ correspond to components, and the red, dashed lines correspond to edges $b_2, \ldots, b_t \in Y$.](image)

We stop the algorithm when either $|S| \geq k \lg n$ or $|Y| > (4k)^c |S|$. We will show that the probability that this can happen in the hash graph is bounded by $O(n^{-k+2})$. The two cases are described below and the proof of each case is ended with a $\diamond$.

**The algorithm stops because $|Y| > (4k)^c |S|$:** In this case we know that $|S| \leq k \lg n$ since the algorithm has not stopped earlier. Fix the size $|S| = s$ and the number of components $t$. First we bound the number of ways we can choose the subgraphs $C_1, \ldots, C_t$. Let $a_i$ be the number of nodes in the subgraph $C_i$. We can choose the structure of a spanning tree in each of $C_1, \ldots, C_t$ in no more than $2^{2(a_1-1)+\ldots+2(a_t-1)} \leq 2^{2s}$ ways. Let $a = \sum_i a_i$ be the total number of nodes. Then it remains to place $s - a + t$ edges which can be done in at most $s^{2(s-a+t)}$ ways. The number of ways that the nodes can
be chosen is at most \( n^{a-t+1} 2^{t-1} \binom{(4k)cs}{t-1} \) by arguing in the following manner:

For each component \( C_i \) we can describe one node by referring to \( b_i \) and which endpoint the node is at. Thus we can describe \( t-1 \) of the nodes in at most \( 2^{t-1} \binom{|Y'|}{t-1} \) ways, where \( Y' \) was the set \( Y \) before the addition of the last edge, so \(|Y'| \leq (4k)cs\). The \( t-1 \) nodes can be picked in at most \( 2^{2s} \) ways, since there are at most \( 2s \) nodes in \( C_1, \ldots, C_t \). The remaining \( a-t+1 \) nodes can be chosen in no more than \( n^{a-t+1} \) ways. Assuming that \( n \) is larger than a constant we know by Lemma 9 that the number of ways to choose the keys in \( S \) (including the order in which they were added) is bounded by \( s^{O(1)} m^{s-k} \).

Hence for a fixed \( a \) the total number of ways to choose \( S \) is at most:

\[
2^{4s} \cdot s^{2(s-a+t) + O(1)} n^{a-t+1} \cdot 2^{t-1} \binom{(4k)cs}{t-1} \cdot s^{O(1)} m^{s-k}
\]

For each of the \( s \) independent keys we fix 2 hash values, so the probability that those values occur is at most \( n^{-2s} \). Thus the total probability that we can find such \( S \) for fixed values of \( s, a, t \) is at most:

\[
2^{4s} \cdot s^{2(s-a+t) + O(1)} n^{a-t+1-2s} \cdot 2^{t-1} \binom{(4k)cs}{t-1} m^{s-k}
\]

\[
\leq n2^{5s} \left( \frac{s^2}{n} \right)^{s-a+t} s^{O(1)} \left( \frac{e(4k)cs}{t-1} \right)^{t-1} \left( \frac{m}{n} \right)^s m^{-k}
\]

\[
\leq ns^{O(1)} \left( \frac{2^5 e(4k)cm}{n} \right)^s m^{-k} \leq s^{O(1)} m^{-k} \leq n(\lg n)^{O(1)} m^{-k}
\]

Since there are at most \( (2k \lg n)^3 = (\lg n)^{O(1)} \) ways to choose \( s, a, t \) we can bound the probability by a union bound and get \( n(\lg n)^{O(1)} m^{-k} = O(n^{-k+2}) \).

**The algorithm stops because** \( |S| \geq k \lg n \): Let \( s, a, t \) have the same meaning as before. In the same way we can show (without using Lemma 9) that the number of ways to choose \( S \) is bounded by

\[
ns^{O(1)} \left( \frac{s^2}{n} \right)^{s-a+t} \left( \frac{2^5 e(4k)cm}{n} \right)^s \leq ns^{O(1)} 2^{-s}
\]

Since \( s = \lfloor k \lg n \rfloor \) we know that \( 2^{-s} \leq n^{-k} \) and a union bound over all choices of \( a, t \) will suffice.

Along the same lines we can show that \( s-a+t \leq k \) with probability \( 1 - O(n^{-k+2}) \). The idea here is that we need to place \( s-a+t \) additional keys when the spanning trees are fixed. Such a key and placement can be chosen in at most \( ms^2 \) ways, but it happens with probability at most \( 1/n^2 \) due to the independence of the keys.
Assume there exists a component with arboricity \( \alpha \geq 2(k + 2)(4k)^c \) and choose a subgraph \( H \) such that \( |E(H)| \geq \alpha |V(H)| - 1 \). Consider the algorithm constructing \( S \) restricted to \( H \). If the algorithm is not stopped early we know that \( Y \) contains exactly the edges of \( H \), so \( |Y| \geq |V(H)| \cdot (k + 2)(4k)^c \) and thus \( |S| \geq (k + 2)|V(A)| \). This implies that \( s - a + t \geq (k + 1)|V(A)| \geq k + 1 \), i.e. every component has arboricity \( \leq 2(k + 2)(4k)^c \) with probability \( 1 - O(n^{-k+2}) \).

From the analysis above we get that there exists no component with more than \( (4k)^c k \lg n \) nodes with probability \( 1 - O(n^{-k+2}) \). Combining this with Lemma 7 we now conclude that with probability \( 1 - O(n^{-k+2}) \) the maximum load is upper bounded by:

\[
2(k + 2)(4k)^c \cdot \lg ((4k)^c k \lg n) = O(\lg \lg n)
\]

The proof of Theorem 2 is now straightforward, since we just need to apply Lemma 10 \( 2^5(4 \lceil \gamma + 2 \rceil)^c \frac{m}{n} = O(1) \) times and take a union bound.

References

[1] Yossi Azar, Andrei Z. Broder, Anna R. Karlin, and Eli Upfal, *Balanced allocations*, SIAM Journal on Computing **29** (1999), no. 1, 180–200, See also STOC’94.

[2] Larry Carter and Mark N. Wegman, *Universal classes of hash functions*, Journal of Computer and System Sciences **18** (1979), no. 2, 143–154, See also STOC’77.

[3] Laura Elisa Celis, Omer Reingold, Gil Segev, and Udi Wieder, *Balls and bins: Smaller hash families and faster evaluation*, FOCS, IEEE, 2011.

[4] Gaston H. Gonnet, *Expected length of the longest probe sequence in hash code searching*, Journal of the ACM **28** (1981), no. 2, 289–304.

[5] Donald E. Knuth, *The art of computer programming, volume III: Sorting and searching*, Addison-Wesley, 1973.

[6] Michael Mitzenmacher, *The power of two choices in randomized load balancing*, IEEE Trans. Parallel Distrib. Syst. **12** (2001), no. 10, 1094–1104.

[7] Michael Mitzenmacher, Andréa W. Richa, and Ramesh Sitaraman, *The power of two random choices: A survey of techniques and results*, Handbook of Randomized Computing: volume 1, 2001, pp. 255–312.
[8] C. St. J. A. Nash-Williams, Edge-disjoint spanning trees of finite graphs, Journal of London Mathematical Society 36 (1961).

[9] C. St. J. A. Nash-Williams, Decomposition of finite graphs into forests, Journal of London Mathematical Society - Second Series 39 (1964).

[10] Rasmus Pagh and Flemming Friche Rodler, Cuckoo hashing, Journal of Algorithms 51 (2004), no. 2, 122–144, See also ESA’01.

[11] Mihai Pătraşcu and Mikkel Thorup, The power of simple tabulation-based hashing, Journal of the ACM 59 (2012), no. 3, Article 14, Announced at STOC’11.

[12] Omer Reingold, Ron D. Rothblum, and Udi Wieder, Pseudorandom graphs in data structures, Proc. 41st International Colloquium on Automata, Languages and Programming (ICALP), 2014, pp. 943–954.

[13] Mikkel Thorup, Simple tabulation, fast expanders, double tabulation, and high independence, FOCS, 2013, pp. 90–99.

[14] Mikkel Thorup and Yin Zhang, Tabulation-based 5-independent hashing with applications to linear probing and second moment estimation, SIAM Journal on Computing 41 (2012), no. 2, 293–331, Announced at SODA’04 and ALENEX’10.

[15] Albert Lindsey Zobrist, A new hashing method with application for game playing, Tech. Report 88, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, 1970.