Expected Worst-case Performance of Hash Files

Per-Åke Larson
Department of Information Processing, Åbo Akademi, Fänriksgatan 3, SF-20500 ÅBO 50, Finland

The following problem is studied: consider a hash file and the longest probe sequence that occurs when retrieving a record. How long is this probe sequence expected to be? The approach taken differs from traditional worst-case considerations, which consider only the longest probe sequence of the worst possible file instance. Three overflow handling schemes are analysed: uniform hashing (random probing), linear probing and separate chaining. The numerical results show that the worst-case performance is expected to be quite reasonable. Provided that the hashing functions used are well-behaved, extremely long probe sequences are very unlikely to occur.

INTRODUCTION

It is well-known that the worst-case performance of a hash file is $0(n)$, where n is the number of records stored. This is true for any of the basic operations: retrieval, insertion or deletion of a record. This result is seen immediately by considering the longest probe sequence of the worst possible file instance. However, the probability that the worst possible file instance will actually occur is incredibly small. It is an extremely pessimistic view which does not give any information about the expected worst-case performance of a 'normal, well-behaved' hash file.

In this paper the worst-case performance of a hash file is approached from a different angle. The problem studied can be stated as follows: consider a certain hash file and the longest of the probe sequences required to retrieve a record stored in the file. What is the expected length of this probe sequence? In other words, not only the longest probe sequence of the worst possible file instance is of interest, but the whole distribution of the longest probe sequence over all file instances. The term 'expected worst-case' will be used to refer to the latter type of analysis. The former, traditional approach might perhaps be called the worst-of-the-worst approach.

The analysis reported here is restricted to the retrieval performance of externally stored hash files. The length of a probe sequence, or simply the search length, is measured in the number of accesses to secondary storage. Three different overflow handling schemes are studied: uniform hashing (random probing) (section two), linear probing (section three) and separate chaining (section four). The bucket size is one or larger than one. Only the case of initial loading is considered, i.e. no deletions are allowed.

So far only two analyses of the expected worst-case performance of hashing have been reported: one by Gonnet[1] and one by Larson.[2] Gonnet studied internally stored hash tables (bucket size one) assuming that overflow records are handled by random probing or by separate chaining. He showed that the expected length of the longest probe sequence grows logarithmically in the case of random probing, and sublogarithmically in the case of separate chaining. Larson analysed a new hashing scheme called Linear Hashing with Partial Expansions. Gonnet was able to obtain simple approximate formulae for the schemes considered. The modelling of the schemes considered in this paper is considerably more complex mathematically, and therefore we will have to be content with numerical results.

From elementary probability theory we recall the following useful theorem. In a random sample $x_1, x_2, \ldots, x_n$ the probability that the largest observed value is less than or equal to a fixed value x_0 equals the product of the probabilities that each observed value is less than or equal to x_0, i.e. $P(\max(x_i) \leq x_0) = P(x_1 \leq x_0)\, P(x_2 \leq x_0) \ldots P(x_n \leq x_0)$. This is true if all the random variables are mutually independent.

The analysis of each scheme follows the same basic pattern. First, the probability distribution of the length of a probe sequence (successful search) is derived. Thereafter, the probability distribution of the length of the longest probe sequence, hereafter abbreviated to llps, in a sample of n sequences is readily found by applying the above theorem. Finally the expected value of the llps is numerically computed and tabulated. The results are approximate because the random sample is considered to be drawn from an infinite file. This simplification was necessary because, for the schemes considered, either the distribution of the length of a probe sequence in a finite file is not known or, when known, the numerical computations involved are far too time-consuming. The results given here overestimate the expected length of the longest probe sequence. For moderately large files the error is no more than a few per cent.

To facilitate comparisons the expected lengths of both successful and unsuccessful searches have also been computed and compiled into an appendix. Note that these expected values are for an arbitrary successful or unsuccessful search, not the longest one.

UNIFORM HASHING

Consider a hash file consisting of m buckets, each bucket having a capacity of b records, $b \geq 1$. There are n records stored in the file. The storage utilization, denoted by α, is then $\alpha = n/(mb)$. We shall assume in this section that overflow records are handled by uniform hashing (random probing, rehashing); i.e. each overflowing record is rehashed randomly until a non-full bucket is found (see Ref. 3).

Let $P(L = k)$, $k = 1, 2, \ldots$, denote the probability that retrieval of a record requires k accesses. For finite m this

CCC-0010-4620/82/0025-0347 $03.00

distribution is known only for the case $b = 1$.[3] For $b \geq 1$, the asymptotic distribution, obtained by letting $m \rightarrow \infty$ keeping the storage utilization constant, was derived by Larson.[4] It can be computed as

$$P(L = k) = (1/\alpha)\int_0^{x(\alpha)} P_b(y)^{k-1}(1 - P_b(y))^2 dy$$

where

$$P_b(y) = 1 - \exp(-yb)\sum_{j=0}^{b-1}(yb)^j/j!$$

and where the function $x(\alpha)$ is (implicitly) defined by Eq. (1).

$$\alpha = 1 - \exp(-xb)(1 - x)\sum_{k=0}^{b-2}(xb)^k/k! - \exp(-xb)(xb)^{b-1}/(b-1)! \quad (1)$$

Let Q_k denote the probability that retrieval of a record requires at most k accesses, i.e. $Q_k = P(L \leq k)$. This probability is

$$Q_k = \sum_{j=1}^{k}(1/\alpha)\int_0^{x(\alpha)} P_b(y)^{j-1}(1 - P_b(y))^2 dy$$

which can be reduced to

$$Q_k = 1 - (1/\alpha)\int_0^{x(\alpha)} P_b(y)^k(1 - P_b(y))dy$$

The integral must be computed numerically. The value $x(\alpha)$ must also be computed numerically from Eqn (1).

Consider a sample of n records from an infinite file with bucket size b and storage utilization α. By applying the theorem mentioned above, we immediately find that the probability that the longest probe sequence encountered in the sample is at most k accesses equals Q_k^n. The expected length of the longest probe sequence is thus

$$R_n = \sum_{k=1}^{\infty} k(Q_k^n - Q_{k-1}^n) = 1 + \sum_{k=1}^{\infty}(1 - Q_k^n)$$

The distribution Q_k^n has been plotted in Fig. 1 for $n = 1000$, $\alpha = 0.8$ and $b = 5, 10$. We see that with a probability greater than 90%, the length of the longest probe sequence will not exceed 9 when $b = 5$ and 6 when $b = 10$. The length of the longest probe sequence is very stable; the probability that it will be either 4 or 5 is as high as 73.3 per cent when $b = 10$.

Table 1 shows the expected length of the longest probe sequence for a few parameter combinations. The values are surprisingly low, especially for larger buckets. The longest probe sequence in a file with $b = 10$, $\alpha = 0.8$ and storing one million records is expected to be only 10.9 accesses, for example. For expected search lengths, see the Appendix.

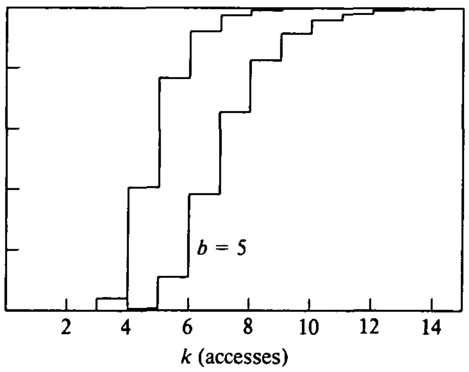

Figure 1. The probability that llps $\leq k$ for uniform hashing, $n = 1000$, $\alpha = 0.8$, $b = 5, 10$.

LINEAR PROBING

Linear probing is one of the earliest and most widely used techniques for handling overflow records. The search path of a record hashing to bucket k is $k, k + 1, \ldots, m - 1, 0, 1, \ldots, k - 1$ and the record is inserted into the first non-full bucket encountered on this path. The performance of linear probing has been analysed by several authors, but either only the case $b = 1$ or only the expected performance is considered,[3,5,6,7] The first analysis of the case $b > 1$, which includes the whole probability distribution, not merely the expected value, of the search length, was published in 1978 by Blake and Konheim.[8] The analysis below is based on their model, but the formulae are rewritten in a form better suited for numerical computations.

Assuming an infinite file, the probability that k accesses will be required to retrieve a record can be computed as

$$P(L = k) = (1/\alpha)\int_0^{\alpha}\{\varphi_b(x)\exp(-xb) \sum_{j=k-1}^{\infty}(xe^{1-x})^{bj}\sum_{s=0}^{b-1}(xe)^s V_{bj+s}\}dx$$

The integral must be computed numerically. The numbers $\{V_{bj+s}\}, j \geq 0, 0 \leq s \leq b - 1$, occurring in the formula do not depend on the storage utilization x. They can be computed using the recurrence

$$V_{bj+s} = \begin{cases} (1/ej)\sum_{i=1}^{j} iV_{bi-1}V_{b(j-1)} & \text{for } s = 0 \\ (b/e(bj+s))\{(j+1)V_{bj+s-1} + \sum_{i=1}^{j} iV_{bi-1}V_{b(j-i)+s}\} & \text{for } s = 1, 2, \ldots, b-1 \end{cases}$$

Table 1. Expected llps for uniform hashing

No. of records	10^3	10^4	10^5	10^6	10^3	10^4	10^5	10^6	10^3	10^4	10^5	10^6	10^3	10^4	10^5	10^6
Bucket size, b			$\alpha = 0.6$				$\alpha = 0.7$				$\alpha = 0.8$				$\alpha = 0.9$	
1	10.5	14.4	18.4	22.5	14.0	19.6	25.3	31.2	20.5	29.2	38.3	47.7	37.4	55.4	74.4	94.0
2	6.6	8.9	11.4	13.9	8.7	12.1	15.5	19.1	12.6	17.9	23.4	29.0	22.6	33.3	44.6	56.3
3	5.1	6.9	8.8	10.7	6.8	9.3	12.0	14.7	9.7	13.7	17.9	22.2	17.2	25.4	34.0	42.8
4	4.3	5.8	7.3	8.9	5.7	7.8	10.0	12.3	8.2	11.5	15.0	18.6	14.4	21.2	28.3	35.6
5	3.8	5.1	6.4	7.8	5.0	6.9	8.8	10.7	7.2	10.1	13.1	16.2	12.6	18.5	24.7	31.1
10	2.5	3.4	4.3	5.2	3.4	4.6	5.9	7.2	4.9	6.8	8.8	10.9	8.5	12.4	16.5	20.8
15	2.1	2.7	3.3	4.1	2.7	3.7	4.7	5.7	4.0	5.5	7.1	8.7	6.9	10.0	13.3	16.7
20	1.9	2.2	2.9	3.3	2.3	3.2	4.0	4.8	3.4	4.7	6.1	7.5	6.0	8.6	11.4	14.3

where the boundary values for $j = 0$ are given by $V_s = (b/e)^s/s!$, $s = 0, 1, \ldots, b-1$. The numbers $\{V_{bj+s}\}$ are related to the numbers $\{T_{b,bj+s}\}$ defined in Ref. 8 by $V_{bj+s} = T_{b,bj+s}(b/e)^{bj+s}/(bj+s)!$. The change has been made purely for numerical reasons; the numbers $\{T_{b,bj+s}\}$ grow extremely rapidly.

The function $\varphi_b(x)$ is a normalization factor which ensures that the probabilities sum to one. It is given by

$$\varphi_b(x) = b(1-x)/\prod_{j=1}^{b-1}(1-\omega^j\exp(-x)\theta(\omega^j x\exp(-x)))$$

where ω is a primitive bth root of unity; $\omega = \cos(2\pi/b) + i\sin(2\pi/b)$, for example. The complex-valued function $\theta(z)$, $z \in \mathbf{C}$, is defined by the series

$$\theta(z) = \sum_{r=0}^{\infty}(r+1)^{r-1}z^r/r!, \quad |z| < 1/e$$

The series converges slowly, however, and numerically the function $\theta(z)$ is most easily computed from the functional equation $\theta(z) = \exp(z\theta(z))$ using Newton–Raphson iteration. A first approximation is readily found by computing a few terms of the series.

The rest is then straightforward. The probability that retrieval of a record requires at most k accesses is

$$Q_k = \sum_{j=1}^{k} P(L = j)$$

and the probability that the longest probe sequence occurring in a sample of size n is of length k or less is Q_k^n. The expected llps is then given by $R_n = 1 + \sum_{k=1}(1 - Q_k^n)$.

Exactly as for uniform hashing, the analysis of linear probing is also approximate and slightly overestimates the expected value. The error is thus 'on the safe side'. Compared with uniform hashing the numerical computations are considerably more complex.

Figure 2 shows the cumulative probability distribution Q_k^n for two different bucket sizes. Compared with uniform

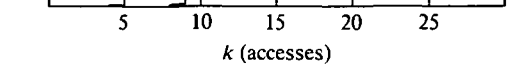

Figure 2. The probability that llps $\leq$ k when using linear probing, $n = 1000, \alpha = 0.8, b = 5, 10$.

hashing, the probabilities grow very slowly; the risk of long probe sequences occurring is much greater. The expected values given in Table 2 are considerably larger than the corresponding values for uniform hashing. This poor worst-case performance is not unexpected if the tendency of linear probing to create long clusters of full buckets is borne in mind. For expected search lengths, see the Appendix.

SEPARATE CHAINING

When using separate chaining, overflow records are stored by linking one or more secondary pages from a separate storage area to the overflowing (primary) page.[3,9] Each secondary page holds records from only one chain, i.e. chains are not allowed to coalesce. The secondary page size, denoted by c, $c \geq 1$, may differ from the primary page size b.

Consider a file where the expected number of records per bucket is zb, $z \geq 0$. The parameter z denotes the load factor. Note that the load factor is not the same as storage utilization and that the load factor may be larger than one. For a large file the probability that j, $j = 0, 1, \ldots,$ records hash to a bucket can be approximated by the Poisson probability $\exp(-zb)(zb)^j/j!$. Hence the probability that a chain of pages will be of length k or less, $k \geq 1$, (the primary page plus $k-1$ secondary pages) is

$$Q_k(z) = \exp(-zb)\sum_{j=0}^{b+(k-1)c}(zb)^j/j!, \quad k = 1, 2, \ldots$$

If a file consisting of N primary pages is considered to have been created by sampling from an infinite file, the probability that the longest chain will be of length k or less is $Q_k(z)^N$. This is at the same time the probability that the longest probe sequence will be of length k or fewer accesses. Observe that we now have the number of primary pages in the exponent, not the number of records as in the two previous cases.

Separate chaining is a totally different type of overflow handling scheme from uniform hashing and linear probing. Performance figures for separate chaining cannot be compared with those of the two other techniques unless the following two conditions are fulfilled. (1) Storage utilization is the same for all three methods, and (2) files of the same size, i.e. the same number of records, are considered. These conditions can be fulfilled by adjusting the parameters z and N respectively.

In order to be able to compute the storage utilization, we must make two simplifying assumptions. The space occupied by pointers, (one per page) and overflow space

Table 2. Expected llps for linear probing

No. of records	10^3	10^4	10^5	10^6	10^3	10^4	10^5	10^6	10^3	10^4	10^5	10^6	10^3	10^4	10^5	10^6
Bucket size, b			$\alpha = 0.6$				$\alpha = 0.7$				$\alpha = 0.8$				$\alpha = 0.9$	
1	23.6	36.7	51.4	67.1	40.8	65.6	93.7	124	85.5	144	211	284	288	521	798	1106
2	12.2	18.7	26.0	33.9	20.7	33.2	47.2	62.4	43.1	72.2	106	142	144	261	400	553
3	8.3	12.7	17.6	22.9	14.1	22.3	31.7	41.8	29.0	48.4	70.8	95.1	96.4	174	267	369
4	6.4	9.7	13.4	17.3	10.7	16.9	24.0	31.6	21.9	36.5	53.2	71.5	72.4	131	200	277
5	5.3	7.9	10.9	14.0	8.7	13.7	19.3	25.4	17.7	29.3	42.7	57.4	58.1	105	161	222
10	3.0	4.3	5.8	7.4	4.7	7.2	10.0	13.1	9.2	15.0	21.8	29.1	29.4	52.8	80.5	111
15	2.3	3.2	4.1	5.2	3.4	5.1	7.0	9.0	6.4	10.3	14.8	19.6	19.9	35.4	53.9	74.4
20	1.9	2.5	3.3	4.1	2.7	4.0	5.4	6.9	5.0	7.9	11.3	14.9	15.1	26.8	40.6	56.0

allocated but not in use, i.e. empty overflow pages, is ignored. They both depend on the details of the implementation. Under these assumptions the load factor z required to achieve storage utilization α can be computed numerically from Eqn (2).

$$zb/\alpha = b + c\exp(-zb) \sum_{k=0}^{\infty} (k+1) \sum_{i=1}^{c} (zb)^{b+kc+i}/(b+kc+i)! \quad (2)$$

This equation defines z as a function of α and this function will be denoted by $z(\alpha)$.

The expected number of records per bucket is $z(\alpha)b$. To store n records in the file, the number of buckets must be approximately

$$N(\alpha) = n/(z(\alpha)b). \quad (3)$$

Finally the expected length of the longest probe sequence can be computed as

$$R_n = 1 + \sum_{k=1}^{\infty} (1 - Q_k(z(\alpha))^{N(\alpha)})$$

To sum up: given storage utilization α and page size b and c, the required load factor $z(\alpha)$ is computed numerically from Eqn (2). Thereafter the number of pages $N(\alpha)$ is given by Eqn (3) and finally the expected value can be computed.

Figure 3 shows the cumulative probability distribution of the llps for $\alpha = 0.8$, $n = 1000$, $c = 1$ and $b = 1, 5, 10$. The computed load factor is $z(0.8) = 1.202$ for $b = 1$, $z(0.8) = 0.899$ for $b = 5$ and $z(0.8) = 0.845$ for $b = 10$. Observe that separate chaining behaves differently from the two other techniques: the probability of long probe sequences *increases* with increasing primary page size. The same behaviour can be observed in Table 3 for higher storage utilizations. For expected search lengths, see the Appendix.

The tables lists numerical results for the case $c = 1$ only, the secondary page size most often used in practice.

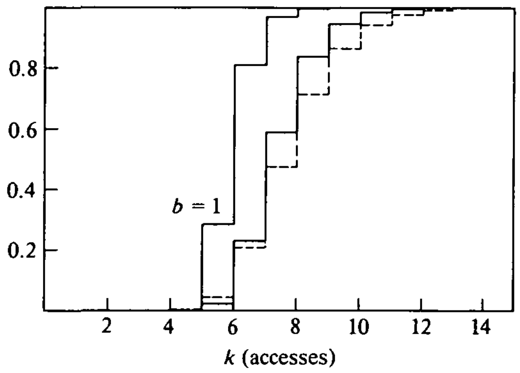

Figure 3. The probability that llps $\leq$ k when using separate chaining, $n = 1000, \alpha = 0.8, c = 1, b = 1, 5, 10$. (---) denotes $b = 10$.

CONCLUSIONS

The expected worst-case retrieval performance of three hashing schemes, uniform hashing (uh), linear probing (lp) and separate chaining (sc) has been analysed. Figure 4 is an attempt to summarize the major results of this analysis.

The analysis reveals that the expected worst-case performance is not very bad. With the exception of linear probing using small buckets, the longest probe sequence occurring in a file is expected to be of quite reasonable length. At least to the author, this was a surprise. The expected length of the longest probe sequence increases with the file size. For uniform hashing the growth is logarithmic, and for the two others nearly logarithmic. It is not easily seen from Fig. 4, but the growth is

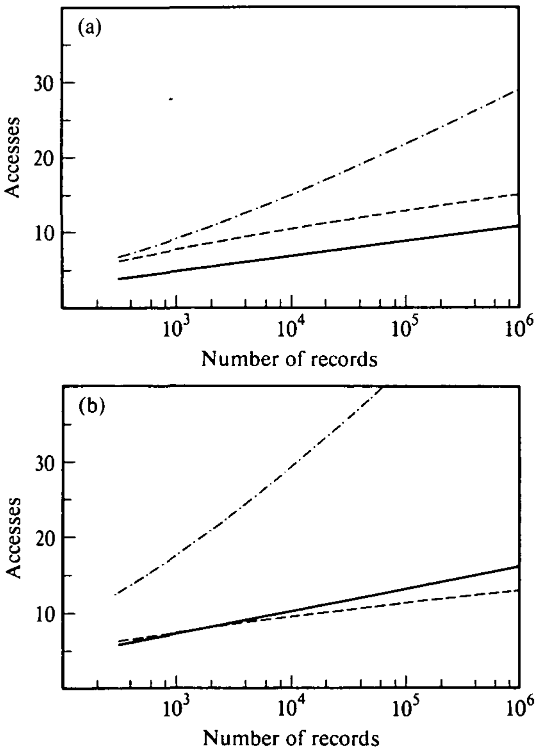

Figure 4. Expected length of the longest probe sequence as a function of file size, $\alpha = 0.8$. (a) For $b = 10$; (b) For $b = 5$ (——) uh, (----) sc, (-·-·-) lp.

Table 3. Expected llps for separate chaining with $c = 1$

No. of records	10^3	10^4	10^5	10^6	10^3	10^4	10^5	10^6	10^3	10^4	10^5	10^6	10^3	10^4	10^5	10^6
Bucket size, b			$\alpha = 0.6$				$\alpha = 0.7$				$\alpha = 0.8$				$\alpha = 0.9$	
1	4.9	5.9	6.9	7.9	5.4	6.5	7.5	8.5	5.9	7.2	8.3	9.4	6.9	8.3	9.6	10.8
2	5.2	6.5	7.7	8.8	5.8	7.2	8.5	9.7	6.6	8.1	9.5	10.7	7.7	9.4	10.9	12.4
3	5.3	6.8	8.1	9.4	6.0	7.6	9.1	10.4	7.0	8.7	10.2	11.7	8.3	10.2	11.9	13.5
4	5.3	6.9	8.4	9.8	6.2	7.9	9.5	11.0	7.2	9.1	10.8	12.4	8.7	10.8	12.7	14.4
5	5.3	7.0	8.6	10.1	6.2	8.2	9.9	11.5	7.4	9.5	11.3	13.0	9.0	11.3	13.3	15.2
10	4.6	6.9	9.0	10.9	6.1	8.6	10.8	12.9	7.8	10.5	12.9	15.1	10.1	13.1	15.7	18.0
15	3.5	6.3	8.8	11.0	5.5	8.6	11.2	13.6	7.7	11.0	13.8	16.4	10.6	14.2	17.2	20.0
20	2.5	5.5	8.3	10.8	4.8	8.3	11.3	13.9	7.5	11.2	14.4	17.3	10.9	15.0	18.4	21.5

sublogarithmic for separate chaining and slightly faster than logarithmic for linear probing.

The worst-case performance of uniform hashing is considerably better than that of linear probing. This was to be expected because linear probing tends to create long clusters of full buckets. For small bucket sizes the performance of separate chaining is better than that of uniform hashing, but for large buckets the order is reversed.

The main result of all the mathematics above is perhaps simply this: it offers a theoretical explanation for the empirical observation that very long probe sequences are unlikely to occur in a well-designed hash file.

A few comments on the results listed in the appendix are in order. The expected search lengths of uniform hashing is always lower than those of linear probing. This comes as no surprise. However, for large buckets the search lengths for both uniform hashing and linear probing are lower than those of separate chaining with $c = 1$. The explanation for this somewhat surprising fact is quite simple: When the primary page size is increased, keeping the total size of the primary storage area constant, the total number of overflow records decreases, but so does the number of overflow chains. So even though there are, in total, fewer overflow records, there may be more of them *per chain*, resulting in longer chains. This problem can be overcome by using either larger overflow pages or several overflow chains per primary page, selecting one of the chains by hashing.

Acknowledgement

The help of B. Qvist in solving some of the computational problems was imperative and is gratefully acknowledged.

REFERENCES

1. G. H. Gonnet, Expected length of the longest probe sequence in hash code searching. *Journal of the ACM* **28** (No. 2) (1981).
2. P.-Å. Larson, Performance analysis of linear hashing with partial expansions. *ACM Transactions on Database Systems* (to appear).
3. D. E. Knuth, The Art of Computer Programming, Vol. 3: Sorting and Searching. Addison Wesley, Reading, Massachusetts (1973).
4. P.-Å. Larson, Analysis of uniform hashing. *Åbo Akademi, Reports on Computer Science & Mathematics Ser. A*, No. 1 (1978).
5. P.-Å. Larson, Frequency loading and linear probing, *BIT*, Bind **19** (No. 2), 223–228 (1979).
6. G. Schay and W. G. Spruth, Analysis of a file addressing method. *Communications of the ACM* **5** (No. 2), 459–462 (1962).
7. G. Tainiter, Addressing for random-access storage with multiple bucket capacities. *Journal of the ACM* **10** (No. 2), 307–315 (1963).
8. I. F. Blake and A. G. Konheim, Big buckets are (are not) better! *Journal of the ACM* **24** (No. 4), 591–606 (1977).
9. J. A. van der Pool, Optimum storage allocation for initial loading of a file. *IBM Journal of Research and Development* **16** (No. 6), 579–586 (1972).

Received June 1981

APPENDIX

Expected length of successful and unsuccessful searches

The tables below list the expected length of an arbitrary (not the longest) successful and unsuccessful search for uniform hashing, linear probing and separate chaining. The results are asymptotic, i.e. for an infinite file, and slightly overestimate the search lengths for a corresponding finite file.

Table A1. Expected search lengths for uniform hashing[4]

Bucket size	Successful search				Unsuccessful search			
			α				α	
	0.6	0.7	0.8	0.9	0.6	0.7	0.8	0.9
1	1.527	1.720	2.012	2.558	2.500	3.333	5.000	10.000
2	1.222	1.330	1.498	1.818	1.757	2.249	3.236	6.171
3	1.126	1.202	1.325	1.561	1.494	1.860	2.602	4.803
4	1.081	1.140	1.237	1.429	1.357	1.654	2.264	4.077
5	1.056	1.103	1.184	1.348	1.272	1.524	2.049	3.616
10	1.014	1.035	1.079	1.177	1.099	1.243	1.572	2.591
15	1.005	1.016	1.045	1.117	1.045	1.141	1.386	2.186
20	1.002	1.009	1.029	1.086	1.022	1.089	1.284	1.957

Table A2. Expected search lengths for linear probing[3, 5, 8]

Bucket size	Successful search				Unsuccessful search			
			α				α	
	0.6	0.7	0.8	0.9	0.6	0.7	0.8	0.9
1	1.750	2.167	3.000	5.500	3.625	6.056	13.000	50.50
2	1.293	1.494	1.903	3.147	2.181	3.387	6.850	25.59
3	1.158	1.286	1.554	2.378	1.714	2.509	4.810	17.30
4	1.098	1.190	1.386	2.000	1.487	2.075	3.794	13.15
5	1.066	1.136	1.289	1.777	1.356	1.819	3.187	10.67
10	1.015	1.042	1.110	1.345	1.114	1.323	1.987	5.71
15	1.005	1.019	1.058	1.209	1.049	1.172	1.597	4.06
20	1.002	1.010	1.036	1.144	1.023	1.103	1.407	3.24

Table A3. Expected search lengths for separating chaining ($c = 1$)[3, 6]

Bucket size	Successful search				Unsuccessful search			
			α[a]				α	
	0.6	0.7	0.8	0.9	0.6	0.7	0.8	0.9
1	1.363	1.463	1.601	1.840	1.210	1.322	1.503	1.866
2	1.217	1.313	1.455	1.713	1.209	1.348	1.580	2.050
3	1.146	1.232	1.369	1.629	1.191	1.344	1.608	2.154
4	1.104	1.182	1.310	1.567	1.171	1.331	1.617	2.224
5	1.078	1.146	1.267	1.518	1.152	1.314	1.617	2.274
10	1.024	1.064	1.152	1.372	1.082	1.230	1.568	2.392
15	1.009	1.033	1.100	1.293	1.044	1.166	1.501	2.421
20	1.004	1.019	1.071	1.242	1.024	1.119	1.437	2.416

[a] α denotes the storage utilization. The corresponding load factor is computed from Eqn (2).