The quadratic hash method when the table size is a power of 2

F. R. A. Hopgood and J. Davenport

Atlas Computer Laboratory, Chilton, Didcot, Berkshire

A number of recent papers have considered the quadratic hash method when the table size is a prime number. This paper shows that, contrary to what is normally assumed, the method can be used for tables whose size is a power of 2 without the usual drawback that the period of search is significantly less than the table size.

(Received October 1971)

To access or enter a key K into a hash table of length M, a mapping function $I(K)$ is defined where $1 \leqslant I(K) \leqslant M$ for all keys K in the population. If the position $I(K)$ in the table is empty or contains the key K then the table search is concluded. However, if this entry contains some other key, then a systematic method of trying additional positions in the table must be defined (Morris, 1968; Maurer, 1968). The simplest is the *linear search* method where the positions immediately following the initial position are tried in turn. This can be expressed as follows:

1. Calculate $k = I(K)$.
2. If the kth position is empty or contains K then the search is concluded.
3. Otherwise set $k = k + 1$ (modulo the table size) and repeat 2.

The quadratic hash method

In the simple linear search method, the ith position tried after the initial one is $k + i$ (in future it will be assumed that this is modulo the table size). An obvious extension is to define the ith entry as $k + ai$ where a is coprime with the length of the table, M. The major fault with the linear search is that if two sequences originating from k_1 and k_2 come together so that

$$k_1 + ai = k_2 + aj$$

then all subsequent entries in one sequence also appear in the other. Once the table starts filling up therefore, sequences tend to join together producing clusters. These clusters themselves join with other clusters so that, when the table is nearly full, the search length is much higher than would be expected if the entries had been added randomly.

The quadratic hash method avoids this clustering by defining the ith position in a sequence as

$$k + ai + bi^2$$

The next position in a sequence now depends on the length of the sequence so that random collisions of two sequences do not cause them to combine. The computation required for the quadratic search method can be reduced by defining $R = a + b$ and $Q = 2b$ so that the algorithm is:

1. Calculate $k = I(K), j = R$.
2. If the kth position is empty or contains K then the search is concluded.
3. Otherwise set $k = k + j, j = j + Q$ and repeat 2.

Frequently the method is used with $Q = 1$. This means that on most computers the updating of j is a single machine order.

Table size M = Prime number

The number of entries that appear in a sequence from a particular initial position before an entry is encountered twice is called the *period of search*. The period of search should be as large as possible and preferably the same as the table size. Otherwise the table can appear to be full when there is still space available. The condition that a and M are coprime ensures that the period of search for the linear hash method is M, the table size. If M is a prime number then the period of search for the quadratic hash method is $M/2$. Although the table would most likely be nearly full (at say 90%) before the maximum search length became as high as $M/2$, it is possible that exceptional cases could arise.

Radke (1970) and Day (1970) have shown that it is possible to combine two quadratic sequences with disjoint members so that, at the expense of having a more complex function for defining the next in a sequence, it is possible to produce a method having M as the period of search.

Table size M = Power of 2

Maurer (1968) states that, in general, when M is a power of 2 the period of a quadratic search is usually too small for effective use. However, in the special case $Q = 1$, the period of search is $M - R + 1$ (see Appendix). Consequently with $R = 1$, the complete table is searched. The period of search in this case is considerably better than the case where M is a prime number. Also, as M is a power of 2, additions modulo the table size can be achieved in most computers by masking off the desired number of bits. The algorithm therefore takes the simple form:

1. Calculate $k = I(K), j = R$.
2. If the kth position is empty or contains K then the search is concluded.
3. Otherwise set $k = k + j, j = j + 1$ and repeat 2.

An unusual property of the method is that the period of search is reached when the ith and $i + 1$th entries are the same. That is the sequence repeats when $j = M$. For example, the entries tried for a table of length 8 starting at $k = 1, R = 1$ are 1, 2, 4, 7, 3, 8, 6, 5, 5 . . . This is often a more convenient test for the table being full than keeping a count of the entries in the table.

Comparison of results

A measure of the efficiency of a table search is the average length of search assuming that each entry in the table is accessed as frequently as any other. If p is the fraction that the table is full then the average length of search for the linear hash method is approximately $(2 - p)/(2 - 2p)$ assuming that the keys are chosen randomly (Hopgood, 1969). If we assume that the quadratic method eliminates clustering then the average length of search is approximately $-(1/p) \log (1 - p)$ (Morris, 1968). The accuracy of these formulae are shown in **Table 1** where the results achieved for a table of length 2048 using random data are compared with the values that should be obtained according to the formulae.

These results are not usually achieved in practice due to keys

Table 1 **Comparison of formulae for linear and quadratic hash methods with results achieved with random data. Table size = 2048**

P	$\frac{2-p}{2-2p}$	LINEAR $a = 1$	$-\frac{1}{p}\log(1-p)$	QUADRATIC $R=7$, $Q=1$
0·1	1·056	1·076	1·054	1·064
0·2	1·125	1·135	1·116	1·123
0·3	1·214	1·212	1·189	1·207
0·4	1·333	1·312	1·277	1·316
0·5	1·500	1·492	1·386	1·441
0·6	1·750	1·733	1·527	1·605
0·7	2·167	2·127	1·720	1·819
0·8	3·000	2·956	2·012	2·187
0·9	5·500	5·579	2·558	2·818

Table 2 **Results obtained with a table of length 2048 and non-random data**

P	LINEAR $a = 1$	LINEAR $a = 173$	QUADRATIC $R=1$, $Q=1$	QUADRATIC $R = 7$, $Q = 1$
0·1	1·160	1·165	1·165	1·180
0·2	1·417	1·396	1·400	1·381
0·3	1·727	1·657	1·641	1·597
0·4	2·216	1·981	1·938	1·871
0·5	2·748	2·689	2·322	2·234
0·6	3·762	3·199	2·744	2·648
0·7	5·338	4·529	3·218	3·042
0·8	8·728	6·302	3·917	3·647
0·9	16·674	8·927	4·957	4·542

normally not being random. These tests were therefore repeated with non-random data. As a frequent use of hash tables is the Symbol Table of a compiler, the identifiers declared in the Algorithms published in the Communications of the Association for Computing Machinery were used. These were hashed into a table of length 2048. The hash function used depended on the first three characters of the identifier. The results obtained are shown in **Table 2**. The value $a = 173$ in the linear case was the best result obtained in a set of 10 runs. It can be seen that the quadratic method is considerably better than the linear hash method even though the extra amount of work involved in incrementing j is usually only a single computer instruction per entry examined.

The value $R = 7$ in the quadratic case was a typical result from a set of R values chosen between 1 and 23. The best result for the average length of search at 0·8 full was 3·538. The case $R = 1$ gave the worst result in this set.

Appendix

Consider the sequence $a_i = iR + \frac{1}{2}i(i-1)$ for $i = 0, 1, 2, \ldots$

If $b_i = a_i \pmod M$ where $M = 2^t$ for some positive t then we require to show that the b_i's for $i = 0$ to $M - R$ are distinct.

Suppose $b_{i+j} = b_i$ for some i, j and $i + j \leqslant M - R$ then

$$0 \equiv b_{i+j} - b_i \equiv a_{i+j} - a_i \pmod{2^t}$$

but

$$a_{i+j} - a_i = jR + \tfrac{1}{2}j(j + 2i - 1)$$

therefore

$$jR + \tfrac{1}{2}j(j + 2i - 1) \equiv 0 \pmod{2^t}$$

If j is even, $= 2^T r$, where r is odd and we know $T < t$ then dividing by $\frac{1}{2}j$ gives:

$$2R + j + 2i - 1 \equiv 0 \pmod{2^{t-T+1}}$$

hence

$$2(R + i) + (j - 1) \equiv 0 \pmod{2^{t-T+1}}$$

This implies $j - 1$ is even which is a contradiction.

If j is odd then dividing by j gives:

$$R + i + \tfrac{1}{2}(j - 1) \equiv 0 \pmod{2^t}$$

We can assume that $i = 0$ and $j = 1$ are not both true as then $R = 0$ or $\lambda 2^t$.

Since

$$0 < i + j \leqslant 2^t - R$$

then

$$0 < i + \tfrac{1}{2}(j - 1) < 2^t - R$$

but

$$i + \tfrac{1}{2}(j - 1) = \lambda 2^t - R \text{ for some } \lambda$$

and again there is a contradiction.

References

Day, J. C. (1970). Full table quadratic searching for scatter storage, *CACM*, Vol. 13, No. 8, pp. 481-482.

Hopgood, F. R. A. (1969). *Compiling Techniques*, London: Macdonald.

Maurer, W. P. (1968). An improved hash code for scatter-storage, *CACM*, Vol. 11, No. 1, pp. 35-38.

Morris, R. (1968). Scatter Storage Techniques, *CACM*, Vol. 11, No. 1, pp. 38-44.

Radke, C. E. (1970). The use of quadratic residue research, *CACM*, Vol. 13, No. 2, pp. 103-107.