International Journal of Computer and Information Sciences, Vol. 3, No. 2, 1974

A Full Table Quadratic Search Method Eliminating Secondary Clustering

Seiichi Nishihara[1] and Hiroshi Hagiwara[2]

Received July 1973; revised January 1974

A quadratic search method for hash tables whose size is a power of 2 is presented. It allows full table searching, and yet eliminates secondary clustering, which occurs when different keys hashed initially to the same location proceed to trace through the same sequence of locations. Comparisons with other methods are made experimentally.

1. INTRODUCTION

Several papers[1-3] have considered the quadratic search method when the table size is a prime number. In a recent paper, however, Hopgood and Davenport[4] have shown that, contrary to what is normally assumed, the quadratic search 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.

To access or enter a key K into a hash table of length M, a hashing function $h(K)$ which generates an initial hash address is defined, where $0 \leqslant h(K) \leqslant M - 1$ for all keys. If the $h(K)$ th location in the table is empty or contains the key K, then the table search is concluded. However, if this entry contains some other key, an additional location must be searched for. The quadratic search method is a method to generate such subsequent locations as

$$[h(K) + ai + bi^2]_{\mathrm{mod}M}, \qquad i = 0, 1,...$$

In the following, additions are done modulo the table size M. Letting

[1] Data Processing Center, Kyoto University, Kyoto, Japan.
[2] Faculty of Engineering, Kyoto University, Kyoto, Japan.

$R = a + b$ and $Q = 2b$, the algorithm for this quadratic search method takes the following form[4]:

Step 1. Calculate $k = h(K), j = R$.

Step 2. If the kth location is empty or contains K, then the search is concluded.

Step 3. Otherwise, set $k = k + j$ and $j = j + Q$ and repeat step 2.

The number of entries that appear in a sequence from a particular initial position until an entry is encountered twice is called the period of search. Hopgood and Davenport[4] have shown that in the special case when M is a power of 2 and $Q = 1$ the period of search is $M - R + 1$.

2. A NEW SEARCH METHOD

In this section a new quadratic search method for tables whose size M is a power of 2 (put $M = 2^n$) is presented.

Note that if d is an odd number and $R = Q = d$, then the period of search by the algorithm given in the previous section is equal to the table size M (see Appendix A). Furthermore, as shown in Appendix A, if $d_1 \equiv d_2 (\text{mod } M)$, then the same sequence of locations is generated in both cases.

Let $h(K)$ be the hashing function, where $0 \leqslant h(K) \leqslant M - 1$ for all keys K in the population. Now, in order to generate odd numbers d mentioned above, we introduce a function $g(K)$ independent of h, where

$$0 \leqslant g(K) \leqslant \tfrac{1}{2}M - 1.$$

In fact, these functions h and g can easily be defined by $2n - 1$ of the bits which construct a key, that is, n bits to define h and $n - 1$ bits to define g.

By letting $d = 2g(K) + 1$, the algorithm generates a sequence of locations given by

$$h(K) + \frac{2g(K) + 1}{2} i + \frac{2g(K) + 1}{2} i^2, \qquad i = 0, 1, 2, \ldots$$

whose period of search is equal to the table size M.

Moreover, the above sequence eliminates secondary clustering.[3] In fact, as long as $g(K_1) \neq g(K_2)$ for any keys K_1 and K_2, even if $h(K_1) = h(K_2)$, the keys trace out two different sequences of locations.

The algorithm is rewritten as follows:

Step 1. Calculate $k = h(K), d = 2g(K) + 1, j = d$.

Step 2. If the kth location is empty or contains K, then the search is concluded.

Step 3. Otherwise, set $k = k + j$ and $j = j + d$ and repeat step 2.

An unusual property of the method is that the period of search is reached when the ith and $(i + 1)$ th locations are the same, i.e., $j \equiv 0 \pmod{M}$, as was pointed out by Hopgood and Davenport.[4]

Compared with other quadratic search methods for tables whose size is a prime number, the above method is simple and rapid, because the modulo M operation and the calculation of $h(K)$ and $g(K)$ are easily achieved by masking or shifting a proper number of bits (see Appendix B).

Especially when the table size varies during the course of computation, can a new table size be found easily by doubling or halving the former table size.

3. EXPERIMENTAL VERIFICATION

The efficiency of a table search is best expressed in terms of the average number E of probes assuming that each entry in the table is accessed as frequently as any other. Probing occurs in step 2. The number E depends on the fraction p of the table that is occupied, but not on the size of the table.

If we assume that the quadratic search method eliminates all clusterings, then E is approximately $-(1/p)\log(1 - p)$.[5] But the assumption of absence of secondary clustering does not hold for usual methods. The additional cost of this clustering has been estimated by Bell.[3]

Simulations were programmed and run on FACOM 230-60[3] for both the present method and the method of Hopgood and Davenport[4] in the case of $R = 7$ and $Q = 1$. In both cases the table size was taken as 2048 and random data were used as keys. The simulation was repeated ten times for each method. In Table I the results of the simulations are compared with the values obtained according to the above formulas. It is seen that the present method gives results very close to the theoretical values.

4. CONCLUSION

We have presented a quadratic search method whose period of search is equal to the table size M, where M is a power of 2. It has been verified experimentally that the method eliminates secondary clustering.

The method is especially suited for computers whose fixed-point division is either nonexistent or very slow, because such operation is not needed. It is also suited when the table is in some slow access external medium.

[3] Fujitsu Co., Japan.

Table I. Average Number E of Probes Necessary to Search an Item in the Table

p	Bell's formula[3]	Method of Hopgood and Davenport[4] $R = 7, Q = 1$	$-(1/p)\log(1-p)$	Present method
0.1	—	1.051	1.054	1.050
0.2	—	1.135	1.116	1.122
0.3	—	1.215	1.189	1.191
0.4	—	1.315	1.277	1.284
0.5	1.44	1.443	1.386	1.384
0.6	1.61	1.613	1.527	1.526
0.7	1.84	1.829	1.720	1.717
0.8	2.18	2.156	2.012	2.014
0.9	2.79	2.731	2.558	2.533

APPENDIX A

Let $M = 2^n$ and d be an odd number. Consider the sequence

$$a_i = \tfrac{1}{2}di + \tfrac{1}{2}di^2 \qquad \text{for} \quad i = 0, 1, \ldots, 2^n - 1$$

Let $b_i = [a_i]_{\text{mod}M}$ for each i. If $d_1 \equiv d_2 \pmod{M}$, then

$$\tfrac{1}{2}d_1 i + \tfrac{1}{2}d_1 i^2 = \tfrac{1}{2}d_1 i(i+1) \equiv \tfrac{1}{2}d_2 i + \tfrac{1}{2}d_2 i^2 \pmod{M}$$

Therefore from the definition of b_i the same sequence of b_i is generated, where $0 \leqslant i \leqslant 2^n - 1$.

Next, we show that the b_i are distinct. Suppose $b_{i+j} = b_i$ for some i, j and $0 < (i+j) < 2^n$; then from the definition of b_i

$$a_{i+j} - a_i \equiv 0 \pmod{2^n}$$

But

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

Therefore

$$\tfrac{1}{2}d_j(2i + j + 1) \equiv 0 (\text{mod } 2^n)$$

By definition, d is coprime with 2^n. Therefore, dividing by $d/2$ gives

$$j(2i + j + 1) \equiv 0 (\text{mod } 2^{n+1})$$

1. If j is even, i.e., $j = 2^t r$, where $1 \leqslant t < n$ and r is odd, then dividing by j gives

$$2i + j + 1 \equiv 0 (\text{mod } 2^{n-t+1})$$

Hence $j + 1$ is even, which is a contradiction.

2. If j is odd, then dividing by j gives

$$2i + j + 1 \equiv 0 (\text{mod } 2^{n+1}) \tag{A.1}$$

But $0 < (i + j) < 2^n$; therefore

$$0 < [i + \tfrac{1}{2}(j + 1)] < 2^n \qquad \text{for} \quad j \geqslant 1$$

Thus multiplying by two gives

$$0 < (2i + j + 1) < 2^{n+1}$$

which contradicts Eq. A.1).

APPENDIX B

For a practically efficient algorithm, the present method is restated in a pattern similar to that given by Bell[3]:

1. Set $k = h(K)$.
2. If location k is successful (matches K or empty), we are done.
3. Set $d = 2g(K) + 1$ and $j = d$.
4. If location $(k + j)_{\text{mod}M}$ is successful, we are done.
5. If $j \equiv 0 (\text{mod } M)$, search is a failure and we are done.
6. Increment k by j.
7. Increment j by d.
8. Go back to step 4.

Make a timing comparison with the algorithm of Bell[3], whose table size must be a prime number. Steps 1 and 4 (corresponding to step 5 of Bell's algorithm) take less time, because Bell's algorithm contains fixed-point divisions. Only step 3 may take longer, because it contains masking, shifting, and addition as extra instructions. However, when step 3 is executed, step 4 is also inevitably executed one or more times. The execution of division usually takes five or more times longer than that of masking, shifting, or addition, so the extra cost of step 3 will be compensated by the division instruction in step 5 of Bell's algorithm. The final test (step 5) is also simple and rapid.

REFERENCES

1. W. D. Maurer, "An improved hash code for scatter storage," *Comm. ACM* **11**(1):35–38 (1968).
2. C. E. Radke, "The use of quadratic residue research," *Comm. ACM* **13**(2):103–105 (1970).
3. J. R. Bell, "The quadratic quotient method: a hash code eliminating secondary clustering," *Comm. ACM* **13**(2):107–109 (1970).
4. F. R. A. Hopgood and J. Davenport, "The quadratic hash method when the table size is a power of 2," *Comput. J.* **15**(4):314–315 (1972).
5. R. Morris, "Scatter storage techniques," *Comm. ACM* **11**(1):38–43 (1968).