Comment on Brent's Scatter Storage Algorithm

Jerome A. Feldman and James R. Low
Stanford University

Key Words and Phrases: hashing, information storage and retrieval, scatter storage, searching, symbol table
CR Categories: 3.7, 3.73, 3.74, 4.1, 4.9

R.P. Brent in his presentation of a modification to the linear quotient algorithm [1] shares the common misconception that dynamic chaining requires larger table entries because of the space required for link fields.

With dynamic chaining we do not need to store the entire key with each entry, but just enough information to distinguish between entries which have the same the initial hash ($r(K) = K \bmod n$). Thus we need to store only abbreviated keys [2] with each table entry ($a(K) =$ integer part of K/n). We also need a tag bit per entry to indicate if the entry is the head of a conflict list. We may represent the link field by integers in the range 0 to $n - 1$. This has been understood since at least 1965 [3]. See [2] for the complete algorithm.

Now we will demonstrate that this does not need significantly more space than required for the full key used by rehashing techniques. For simplicity, let us assume that all keys are positive and that the value of the largest key is K. We may represent the abbreviated key and link in a field large enough to represent $(a(K)+1)n-1$. The field we would need for a full key would have to be large enough to represent a number $(a(K)n + r)$ where $0 \leq r \leq n - 1$. Thus, we find in the worst case that the minimum field needed to represent the tag bit, the abbreviated key, and the link field is only 2 bits wider than the minimum field needed to represent a full key. Other hashes such as taking m bits (tablesize $= 2 \uparrow m$) from the key will require only the extra tag bit.

Authors' address: Computer Science Department, Stanford University, Stanford, CA 94305.

Deletions do not cause any problem with dynamic chaining and do not clutter up the table with "deleted" entries which slow down searches using rehashing. Dynamic chaining takes very little extra space, is easy to program, and according to Brent requires on the average less probes per search. With this knowledge, we find ourselves wondering when rehashing should be used instead of dynamic chaining.

Received March 1973

References

1. Brent, R.P. Reducing the retrieval time of scatter storage techniques. *Comm. ACM 16*, 2 (Feb. 1973), 105–109.
2. Knuth, D.E. *The Art of Computer Programming: Searching and Sorting. Vol III.* Addison-Wesley, Reading, Mass., 1973, sec. 6.4, exercise 13 p. 543.
3. Feldman, J.A. Aspects of associative processing. Tech. Note 1965–13, MIT Lincoln Laboratory, Lexington, Mass., 1965.

Reply by Richard P. Brent

It is true that an abbreviated key and tag bit may be stored in place of the full key when direct chaining is used. However, two points should be considered.

First, in applications it often happens that the key, and associated information, requires an integral number of words (or bytes) of storage. Also, for programming convenience and execution speed the table entries must occupy an integral number of words (or bytes). Hence, the requirement for even one extra bit per entry may mean a significant increase in the table size. Space could be saved by storing the tag bits in a separate bit table, but this would increase the time required to make a probe.

Second, even assuming that the extra space required for direct chaining is negligible, the expected number of probes per retrieval is essentially the same as for our method over a wide range of load factors (see Figure 2 of [1]). Hence, the choice of method depends mainly on the time required to make a probe, and this depends on the machine, language, hashing method, and the implementation of the algorithm.

The conclusion of [1] is that, in applications where most entries are looked up several times and deletions are rare, our method is preferable to the linear quotient method, and at least competitive with direct chaining. The observations of Feldman and Low do not invalidate this conclusion.