Optimal search on navigation equipment troubleshooting database

Ming-hai Li, Tian-wei Li, Hai-rui Ma, Su Wang, Yi-li Liu and Ding-ding Guo
Department of navigation, Dalian Naval Academy
Dalian, Liaoning, 116018, China
tianweilee@sina.com

***Abstract*—In order to diagnose and resolve issues occurred on navigation equipment quickly and accurately, an optimal search algorithm, Hash table and Hash function on a troubleshooting database of navigation equipment, are presented. The algorithm can directly find the issue reason and solution by searching Hash address calculated from Hash function. The average search time does not increase when the number of issue record increases. This paper shows how the algorithm is used to find the issue reason and solution in a troubleshooting database of navigation equipment. Thus, it can be widely used in many fields.**

Keywords—Hash table; Hash function; Hash address; load factor; troubleshooting

I. INTRODUCTION

Navigation equipment guides the direction of vessel sailing. If the navigation equipment of a sailing ship is malfunctioning, the ship will be out of track and get lost. Therefore, it is important to trouble shoot rapidly and accurately when an issue happens, so that the ship can be recovered quickly and be brought back on track. Fault Tree Analysis is a powerful tool for troubleshooting because it is good at reliability and safety analysis of large and complex systems. The navigation equipment troubleshooting knowledge database is accumulated based on the system structure, function principle and large amount of user's experience. It involves tables or flow charts with complex logical relations. Therefore, it has significantly practical meaning to the study the optimal search of the navigation equipment troubleshooting database based on Fault Tree Analysis.

II. THE PRINCIPLE AND METHOD OF OPTIMAL SEARCH

A. The concept

Hash function and hash table: Binary Search and Binary Tree have high average cost because the average lookup number is high which determines search efficiency. Ideally, the target value or record can be found during data query without any lookup. A hash table is a structure that can map a key K to a value. A hash table uses a hash function f to compute an index $f(k)$ into an array of buckets or slots, from which the desired value can be found.

Work partially supported by the Dalian naval academy of Science Foundation.

Hash collisions: different keys that are assigned by the hash function to the same bucket. That is,

For $key1 \neq key2$, $f(key1) = f(key2)$

B. Choosing a good hash function

There are many algorithms for a hash function. However, a good hash function and implementation algorithm is essential for good hash table performance. A basic requirement is that the function should provide a uniform distribution of hash values, which means, for any give key, the corresponding index is equally distributed in the index range. Thus the number of hash collision can be reduced. The common hash function algorithms are: direct address method, digit-extraction method, mid-square method, folding method, division remainder method, Pseudorandom generation method etc.

Mid-square hashing is suitable in our practical application considering computing cost, key length, hash table size, key length and distribution, and value search frequency. In mid-square hashing, the key is squared and the address is selected from the middle of the square number. The key size is determined by the hash table. The digits can be chosen based on the practical situation. Mid-square hashing is a commonly used method. For example, we can use two octal digits to present letters and integers (see Figure1), the corresponding hash key and index are shown in Table1.

A	B	C	…	Z	0	1	2	…	9
01	02	03		32	60	61	62		71

Figure 1 - Letters and numbers represented by octal digit

Table **1 Hash index calculated from mid-square hashing**

Record	Keywords	(Keywords) ^{2}	Hash index
A	0100	0010000	010
I	1100	1210000	210
P1	2061	4310541	310
Q2	2162	4741304	741

C. Hash collision resolution

The concept of hash collision is introduced in chapter 1.1. The collision should be reduced but is practically unavoidable.

Resolving collision is to find the next bucket of the same index for the value of the given key. You may find the next bucket is also taken, then you need to keep looking for the next bucket until you find one that is available. The common methods include open addressing, separate chaining, build public overflow area etc. We choose separate chaining because it does not have secondary clustering which is, two records do only have the same collision chain if their initial position is the same. In addition, separate chaining does not increase computing time.

In separate chaining, each bucket is independent, and has an array of entries with the same index. Suppose the index range [0…m-1] is generated from some hash function, an array with pointer can be created:

chainhash: ARRAY[0…m-1] of pointer;

The pointer of each item is initialized as null pointer. Each record with index 'i' is inserted to the list with head pointer pointing to chainhash[i]. The record can be inserted to the head or the tail of the list. It can also be inserted between the head and the tail so records can be ordered in the list based on the key (see Figure 2).

During record search, the bucket is computed from the hash function. Then the entries of the selected bucket are scanned for the desired key.

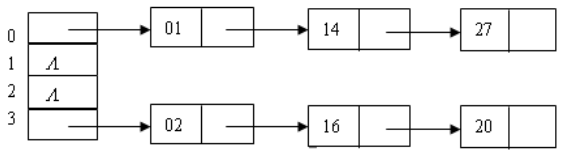


Figure 2 Example of separate chaining for hash collision

（Keys on the list of the same index in order of increasing）

D. Conclusion

It can be concluded during hash table search that,

1) Although a hash map can be built between the key and index of a record, the unavoidable hash collision make the search become a comparison between a given value and a key. Thus, the average search cost is still needed to measure the performance hash table search.
2) The number of the key compared to the given value during the search is determined by the following three items: hash function, collision resolution and the load factor of the hash table.

In general, for a hash table with the same collision solution, the average cost depends on load factor. The load factor of α hash table is defined as:

$$\alpha = \frac{\text{The number of records in hash table}}{\text{hash table length}} \qquad (1)$$

Therefore, we can approve that

The average cost of separate chaining method of hash table when target value search is successful is:

$$S_{nc} \approx 1 + \frac{\alpha}{2} \qquad (2)$$

From the above formula we can get the conclusion that the average cost of hash table is only a function of α, not a function of record number n. In other words, regardless of the size of n, the average search cost can be limited to a range by choosing a reasonable load factor. Thus the search cost will not increase because the length of table record is increased.

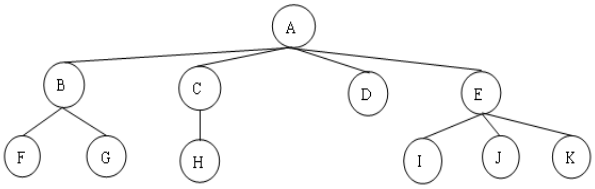
Sometime a hash function without collision can be found depending on the practical situation.

III. Optimal Search Application in Fault Tree Analysis

When troubleshooting expert system is implemented, the key words of the issue can be used to build hash function and hash table using above optimal search method. Thus the issue reason and solution can be directly found. However, there may be more than one reason and solution for an issue. A Fault tree is needed in this case, see Figure 3. The hash index computed from the hash function using the key words in the issue is the top event of issue reason and solution fault tree.

The knowledge database structure is in Figure 4, and the logic flow chart is in Figure 5.

Figure 3 Fault tree structure



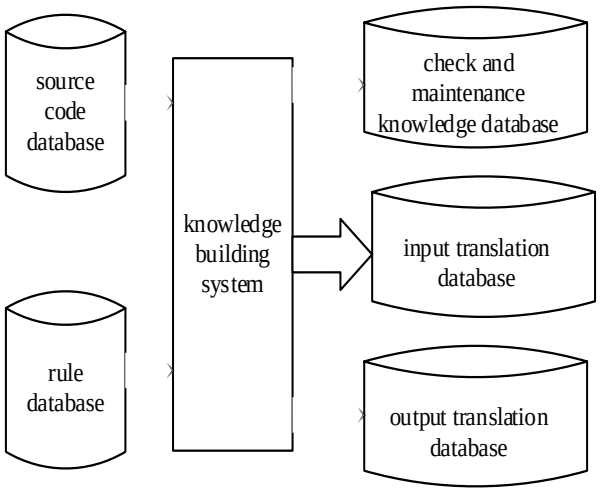


Figure 4 Figure knowledge database structure

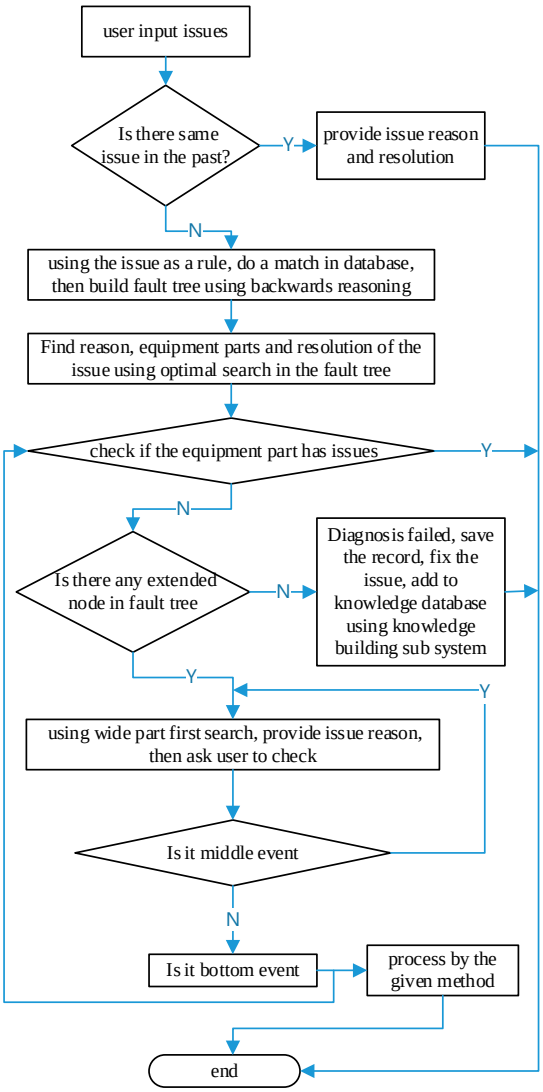


Figure 5 Flow chart of the logic

The system uses meta control strategy. Meta control strategy displays the metadata and build into knowledge database. The metadata reasoning system uses metadata to direct target reasoning system to solve the problem. Meta control strategy is a powerful control method which can increase the flexibility and diagnosis ability of the diagnosis system.

Backwards reasoning in fault tree analysis generates many reasons for top and middle events. The optimal search can quickly obtain the most likely reason event.

IV. Conclusions

The article presented the hash table application in navigation equipment troubleshooting database. The algorithm can compute the hash index based on hash key and performs a direct search. It avoids the multiple lookup and comparison during the search as in traditional search methods. Thus it can search the desired record quickly and accurately. In addition, the average search cost does not increase when the record length of the hash table increases. Although this algorithm uses large memory to speed up the search process, it is not a real problem for a modern computer system.

References

[1] Giarrantano J, Riley G. “Expert Systems Principles and Programming ”[M]. PWS Publishing Company,1998.

[2] Wei-min Yan,Wei-min Wu. “data structure ”G. Eason, B. Noble, and I.N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529-551, April 1955. (*references*)

[3] E.Horowitz and S.Sahni, Fundamentals of Data Structures, by Pitmen Publishing Limited, 1976.

[4] D.E.Knuth, The Art of Computer Programming, volume 1/Foundamental Algorithms; volume3/Sorting and Searching, by Addison—Wesley Publishing Company, Inc.,1973.