Algorithm for Solving the Collisions in Uniform Simple Hashing with Combined Linked Ordered Lists

Tudor Nicoleta Liviana
Department of Computer Science
Petroleum-Gas University of Ploiesti,
39 Bucuresti Avenue, 100680, Ploiesti, Romania
tudorl@upg-ploiesti.ro

Abstract- **This article addresses the problem of solving the collisions through chaining in the simple uniform hashing. The personal contribution consists of the improvement of the algorithm of dispersion through linking, by modifying the linking schema of the elements of the hash table. A contribution presents solving the collisions by creating combined linked ordered lists for storing the elements that have the same hash value. The code returned by the second hash function applied to the key of each element, is used for ordering the elements of the combined linked list. The proof of the performance of the proposed algorithm consists in the computing of the time needed for the search with and without success and of the comparative analysis of the time of execution of the algorithms.**

Keywords: algorithm dispersion, hash table, collision, combined linked ordered lists, simple uniform hashing, linking.

I. Introduction

This article presents the way of solving the collisions through linkage, in uniform, simple dispersion and presents several improvements. First of all, we are introducing the notion of combined linked ordered lists (CLOL) and we are presenting the algorithm of searching in a hash table that uses CLOL. For comparing two elements, we propose the usage of the code generated by the second hash function. Second, we are proving the fact that CLOL offers a better search time than the linked lists. Third, we are analyzing the statistics of the execution times for the presented algorithms that are dependent of the load factor of the hash table.

The paper is organized as follows:

- the section The Hashing with Chaining addresses the issue of the hashing of keys from a domain of possible keys to pseudo-random locations and the way of avoiding the collisions through linkage in the case of simple, uniform hashing;
- the section Solving the Collisions using Combined Linked Ordered Lists presents a way of optimizing the hashing with chaining algorithm by modifying the linking schema of the elements of the hash table. The idea is to create combined linked ordered lists for storing the elements that have the same hash value. The proof of the efficiency of the CLOL algorithm is based on computing the time needed for a search, with and without success (when the value exists and when it does not exist in the collection).
- the section Application: Processes Monitoring on the Computer describes an application of the Hash tables using CLOL.
- the section The Comparative Analysis of the Execution of Algorithms is dedicated to interpretation of the statistics regarding the execution time of the algorithms presented in the previous sections.

T. Sobh (ed.), Innovations and Advanced Techniques in Computer and Information Sciences and Engineering, 363–368.

II. THE HASHING WITH CHAINING

A hash table [1] is an efficient data structure that can store numerical data, strings or complex objects in pseudo-random locations, usually by using static structures of type array, or dynamic structures as linked lists. Each element of the array is called slot (or bucket) and can contain a pair key-value or a record.

For defining the hashing [2], we consider D the domain of the possible keys and E the set of the keys that are actually stored. The hashing means that an element having the key k is stored at location h(k), where h is a function of hashing defined as follows [3]:

h: D → { 0, 1, …, n-1}, |T| = n
k→ h[k], where h[k] is an index in the hash table T[0 … n-1].

If there are two keys for which the same index is allocated, than the corresponding records can not be stored at the same location. There are lots of techniques for avoiding the collision (the repetition of indexes) and among those, the most popular are the linkage and the direct addressing.

Next, we are going to present the way of avoiding the collisions through linkage in the case of simple, uniform hashing.

Each slot of the hash table points to a linked list of inserted records that point to the same element (figure 1) or is NULL, if it does not contain any elements [4].

The base operations supported by the hash tables, in the case of linkage, are:

- Inserting – each record (or pair key – value) can be inserted in a free slot of the table or in a slot that points to a linked list of records;
- Searching (key) that returns true/false if the key is found/not found in the table;
- Deleting (key).

The algorithm that creates and displays a hash table using the linkage, contains the following procedures and functions for inserting, searching and deleting:

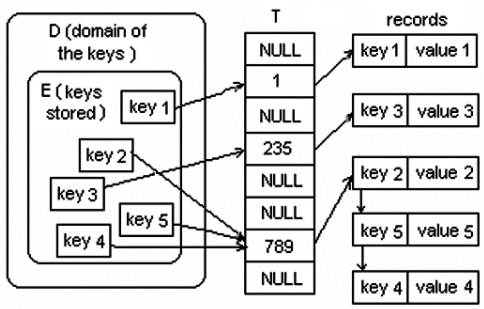

Fig. 1. Hash table with linked lists

```
procedure inserting ( x, k )
    // k = h(key[x])
    Insert x in Hash table at position k, at the
    begining of the list
end // procedure

function exist(k, c)
   if Hash table is empty (at position k)
          return(false)
       else
          q ← element k of the Hash table
          while (q <> NULL)
                if key[q] = c
                      return (true)
                 endif
                q ← next[q]
           repeat
           return(false)
    endif
    end // function

procedure delete( x)
    Delete x from list, from position h(key[x])
end
```

For the analysis of the time needed for the hashing of the elements in hash tables, one needs to find the load factor of the table which will be defined as follows:

Definition 1: Given a hash table T with n elements (T[n]), that stores m elements, the load factor α for the table T is defined as follows: $\alpha = m/n$ and represents the average number of elements stored in a linkage.

In what follows, we are presenting several base definitions of the used notions [5].

Definition 2: *The simple, uniform hashing* defines the hypothesis according to which each element can be hashed in any of the n slots of the table with the same probability regardless of the slots where the other elements have been hashed.

Definition 3: (Asymptotic notation θ) Given a function g(n), we mark with $\theta(g(n))$, the set of functions [6]:

$$\theta(g(n)) = \{ f / (\exists)\ c_1, c_2, n_0 > 0, \text{ so that } 0 <= c_1 g(n) <= f(n) <= c_2 g(n), (\forall)\ n >= n_0 \}. \quad (1)$$

In order to find the average number of elements examined by the search algorithm, we are considering two cases: when the search is unsuccessful (no element from the table has the searched key) and the case when the search is successful, so that it finds an element with the given key. Cormen [7] has proven the following theorem: