Universidad Panamericana

Compilers
Computer Graphics and Systems Engineering

Dispersion Tables (Hash Tables)

semester

2022-2

Students:

(0197495@up.edu.mx) Mariana Ávalos Arce

Professor:

Blanca Estela Villalvazo Flores

monday, may 2nd, 2022
Guadalajara, Jalisco

1 Introduction

The following document presents a brief dive into what a **Dispersion Table** is, also known as **Hash Table**. It covers the main questions like: what is a Dispersion Table?, what are its uses and applications? and a description of an algorithm that implements it.

2 What is a Dispersion Table?

A **Dispersion Table or Hash Tables** are a data structure with the purpose of implementing a technique to insert, delete and search data in an average time complexity known as O(1), that is, constant time.

A hash table is a data structure that is used to store keys/value pairs. It uses a **hash function** to compute an index into an array in which an element will be inserted or searched. By using a good hash function, hashing can work well.

2.1 Hashing

But what is *hashing*? In hashing, large keys are converted into small keys by using hash functions. The values are then stored i a data structure called hash table. The idea of hashing is to distribute entries (key/value pairs) uniformly across an array. Each element is assigned a key (converted key). By using that key you can access the element in O(1) time. Using the key, the algorithm (hash function) computes an index that suggests where an entry can be found or inserted.

Hashing is implemented in two steps:

1. An element is converted into an integer by using a hash function. This element can be used as an index to store the original element, which falls into the hash table.
2. The element is stored in the hash table where it can be quickly retrieved using hashed key.

```
hash_val = hashfunc(key)
index = hash_val % array_size
```

2.2 Hash Functions

A hash function is any function that can be used to map a data set of an arbitrary size to a data set of a fixed size, which falls into the hash table. The values returned by a hash function are called hash values, hash codes, hash sums, or simply hashes.

3 Hash Algorithm: Cichelli's Method

Uniform distribution of the hash values is a fundamental requirement of a hash function. A non-uniform distribution increases the number of collisions and the cost of resolving them. Uniformity is sometimes difficult to ensure by design, but may be evaluated empirically using statistical tests. A **perfect hash function** is one that maps the set of actual key values to the table without any collisions.

If the set of keys IS known in advance, it is possible to construct a specialized hash function that is perfect, perhaps even minimal perfect. The algorithm presented here is one of the most used: **Cichelli's Method**. This method was **published in January of 1980 by R.J. Cichelli, through the ACM**.

This method is used primarily when it is necessary to hash a relatively small collection of keys, such as the set of reserved words for a programming language.

The formula is:

```
h(S) = S.length() + g(S[0]) + g(S[S.length()-1])
```

Where `g()` is constructed using Cichelli's algorithm so that `h()` will return a different hash value for each word in the set (no collisions). The algorithm has three steps:

1. Computation of the letter frequencies in the words.
2. Ordering the words.
3. Searching

The following subsections will describe and explain with images an overview of Cichelli's algorithm for hashing and storing keys for a Dispersion Tables.

3.1 Step 1

Suppose we need to hash the words in the list below:

```
calliope
clio
erato
euterpe
melpomene
polyhymnia
terpsichore
thalia
urania
```

Figure 1: Sample keys

We determine the frequency with which each first and last letter of each word occurs, which is shown in Figure 2.:

letter:	e	a	c	o	t	m	p	u
freq:	6	3	2	2	2	1	1	1

Figure 2: Frequency table

3.2 Step 2

After computing the frequency table, we need to score the words by summing the frequencies of their first and last letters, and then sort them in that order, which is shown in Figure 3. In this way, the highest scoring word is *euterpe* with a count of 12.

calliope	8
clio	4
erato	8
euterpe	12
melpomene	7
polyhymnia	4
terpsichore	8
thalia	5
urania	4

euterpe
calliope
erato
terpsichore
melpomene
thalia
clio
polyhymnia
urania

Figure 3: Score and sort

3.3 Step 3

Finally, consider the words in order and define `g(x)` for each possible first and last letter in such a way that each of the words will have a distinct hash value, which is shown in Figure 4. Thus, as shown in the picture, the hash of *euterpe* is 7: therefore, in the final array with size `MAX_T`, the slot where the value for euterpe key will be stored is slot 7.

If there is the same hash output for two words, as the case with the key *urania*, there is a collision and therefore the value is rejected and computed again until there is one unique for each key.

word	g_value assigned	h(word)	table slot
euterpe	e-->0	7	7 ok
calliope	c-->0	8	8 ok
erato	o-->0	5	5 ok
terpsichore	t-->0	11	2 ok
melpomene	m-->0	9	0 ok
thalia	a-->0	6	6 ok
clio	none	4	4 ok
polyhymnia	p-->0	10	1 ok
urania	u-->0	6	6 reject
	u-->1	7	7 reject
	u-->2	8	8 reject
	u-->3	9	0 reject
	u-->4	10	1 reject

Figure 4: Hashing last step

4 Where are Hash Tables used? (Applications)

Since a Hash Table optimizes the code to a large degree by transforming an O(N) search to O(1), it is one of the most critical things that any programmer and developer should know. Also, the hash algorithm that the Dispersion Table uses often is **secret** and thus the information becomes more secure.

The most common and demanded applications of Hash Tables are:

- To implement programming languages: more specifically, in grammar theory of compilers.
- To implement file systems: the operating system often uses hash tables for its API function access.
- To implement pattern searching software: **RegEx** APIs for most programming languages.
- Distributed key-value storage in distributed systems.
- Cryptography: encoding and decoding messages through networks.
- Password verification and storage: for security purposes, in large and public Databases hash tables are used to avoid attacks and user's privacy issues.

5 Conclusions

A hash table is honestly one of the most valuable data structures, since the search time goes from O(N) complexity using another structure, to O(1), which drastically improves the performance of the program. Also, it is very user-friendly to use a Hash Table given that the key automatically gives the program the value stored. Therefore, it becomes always an option for storing dictionaries or sets where the key and value are discernible.

References

[1] R. Charles Bell. *A MONTE CARLO STUDY OF CICHELLI HASH-FUNCTION SOLVABILITY.* URL: https://dl.acm.org/doi/pdf/10.1145/182.358446.

[2] Prateek Garg. *Basics of Hash Tables.* URL: https://www.hackerearth.com/practice/data-structures/hash-tables/basics-of-hash-tables/tutorial/.

[3] Teach Computer Science. *Application of Hashing Tables.* URL: https://teachcomputerscience.com/application-of-hashing/.

[4] Virginia Tech University. *Perfect Hash Functions.* URL: https://dl.acm.org/doi/pdf/10.1145/182.358446.