Paige Thomas

Senior Capstone in Computer Science

February 21st, 2017

The Functionality of Hash Tables

A hash function, an algorithm that maps data to specific memory addresses according to a mathematical formula. The hash table is created by mapping the data to a database or other means of storage, commonly a table. These functions organize data to be easily accessed, changed, or observed as needed.

Each item in the hash table is given a unique identifier when inserted into the hash function and can be easily located each time access is needed. It also creates easy data management as most algorithms will check for duplicate data before inserting a copy into the table, preventing the use from wasting valuable space. In a simple hash function used to sort string, we can use the total decimal ASCII value of the letters and dived them by the number of items in the table (see example).

A hash function has the following properties:

- Determinism
    - A given input value must always output the same hash value
- Uniformity
    - The function should map input evenly over the output range
- Defined Range
    - If the range of outputs is defined, the hash values will fit inside of that range
- Variable Range (Dynamic)
    - If the range of outputs is variable, the hash function shall grow and shrink as needed
- Data Normalization
    - Two inputs that are considered equal must yield the same hash value, such as comparing upper and lower case letters
- Continuity
    - The functions should map similar data as closely together as possible
- Non-Invertible
    - For cryptographic purposes the hash should be non-reversible

In this example we would create a for loop to run through each character in the string, add the ASCII value of each number, and then mod by three to find an address location for each number. However, this is a very basic example to demonstrate how a hash function works and would not be an effective means in real programming.

An effective hash function is a function that can be used with the data set in order to minimalism search time and has the least amount of collisions; preferably the function causes no collisions. Perfect hashing is a hash function that contains no collisions within its' data set. However, a minimalistic perfect hash does this in the least amount of space possible. A collision is created when two different data inputs are mapped to the same location in memory. This obviously cause a problem because we cannot store both of pieces of data at the same location without causing look-up conflict. At this point in the function we move on to giving the most recent data input a new location.

The collisions are solved used chaining or open addressing, also known as probing. Both of these methods have several approaches. Chaining generally consist of creating a linked list at the address location that points to items in order of insertion time. These list items are called buckets and the more buckets per item the less effective the has function. The alternative is open addressing which moves the most recent item in conflict to the next available space in memory, or further if using quadratic probing.

Other means of collision resolution include coalesced hashing and cuckoo hashing. Coalesced hashing is a combination of separate chaining and opening addressing. If collision occurs, the new item is placed in memory with a pointer from the location the hash originally gave. In Cuckoo hashing a combination of two hashes are found, and if an empty memory location is not found it then uses the next open space in memory.

Hash tables are used primarily to store information in an organized manner that creates an easy look up method. Hash tables are used in the following methods:

- Associative Arrays
- Cache
- Database indexing
- Object Representation
- Sets
- Unique Data Representation

These structures use hash tables to reduce network traffic, sort information for retrieval, and hold objects and variable data in a designated space.

Hash tables are also used in cryptography to store username-password combinations. Because hash functions are designed to only work one way, they cannot easily be decrypted by reverse look up. Similar to private keys, the user must have the original value to find the hashed result. Information can be stored through a cryptographic hash function in order to encrypt the data you are entering into the table.