CSCI-2270
Recitation 9
Instructor: Boese

Hash Tables

Objectives

- Learn about hash tables and hash functions and how records are inserted into, retrieved from, and deleted from hash tables
- Write code to handle inserting into and searching through a hash table

Hash Tables

A hash table, also known as a hash map, is a data structure that stores data using a parameter in the data, called a key, to map the data to an index in an array. The data is also called a record, and the array where records are stored is called a hash table.

Two necessary components in a hash table are the array where the records are stored and a hash function that generates the mapping to an array index. Consider a hash table that is used to store records of movies. Each movie record contains the title, ranking, and year of the movie. Each movie record could be defined with a struct as follows:

```
struct movie {
        string title;
        int ranking;
        int year;
}
```

Figure 1 shows the process of how individual movie records are stored in a hash table. Each movie is a record that contains the properties of the movie. The key for the record, which in this case is the title, is input to a hash function, shown in Figure 1 as h(Title). The hash function uses the ASCII characters in the title and generates a unique integer value for that title. That integer value is the index in the hash table array where the movie record is then stored. For example, if the hash function returns a value of 2, then the movie would be stored at index = 2 in the hash table array. Each index in the hash table array can "hold" a linked list so items with the same unique integer value generated by the hash function can all reside in the same index of the hash table array.

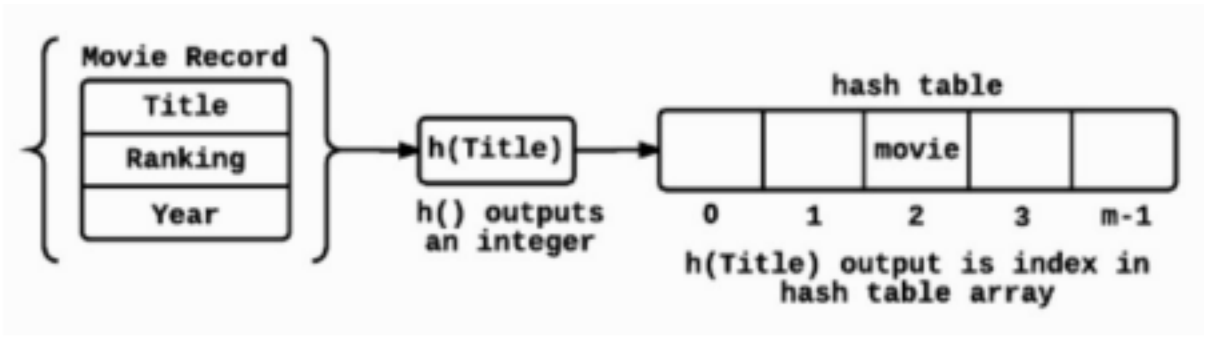

Figure 1. Process showing how individual movie records are stored in a hash table. The movie Title is the input to a hash function, which outputs an integer that serves as the index for the movie in an array. (Excerpt From: Rhonda Hoenigman. "Visualizing Data Structures." iBooks.)

Hash Functions

Hash functions convert the key into an integer index to store the record in the hash table. One of the simplest hash functions first converts a string to an integer by summing the ASCII values of all letters in the string and then takes the modulo of the sum by the array size. The modulo operation ensures that the integer is within the bounds of the array. The pseudocode for this hash function is shown below. The algorithm takes the key and the size of the hash table as arguments and returns the index where the record is to be stored in the hash table.

```
Function hashSum(key, tableSize)
where key is a string or character array and tableSize is the size of
the array
   -  initialize sum = 0
   -  iterate through the key, summing the ASCII values of the letters
      in the key
   -  perform a modulo operation on the sum using the size of the table
      and return this value
```

Inserting

To insert a new record into a hash table, first use the hash function to determine which in which index the item will reside. If there is nothing in that array index, start a new list with the new item the only thing in that list. If a list exists, traverse the list until you find the location where the new item would be in order. For example, items that are sorted with words should be placed in alphabetical order. You should also check if the item you are trying to insert already exists in the hash table. How you handle duplicates is dependent on the particular requirements of your hash table. Your function can return TRUE of the insertion operation is successful, or FALSE if it is not.

```
Function insert(itemToInsert, hashTableArray, tableSize)
   -  Using the hash function, calculate the index where the new item
      will go in the hash table
   - If there is no list at this index, start a new one with the index
      as the first node in the list.  If there are items in the list,
      traverse the list until you find the location where the new item
      will be inserted into the list in the proper order and insert it
Return TRUE or FALSE as appropriate
```

Searching

The search algorithm uses the hash function to determine at which index in the array the search key would be stored. It then traverses the list at that index in an attempt to locate the key. If it finds the key, the function should return the node where the key is stored. Otherwise, it should return NULL. It is important to note that unused indices in the hash table array should be set to NULL before the array is searched.

```
Function search(key, hashTableArray, tableSize)

   - Calculate the index using the hash function
   - Walk through the list at the index calculated from the hash
     function, checking for the key you are looking for
   - If the key is found, return the node where it resides, otherwise
     return NULL
```

Deleting

To delete a record from a hash table, first find the item in question and then simply remove it from its list. The delete function can return a Boolean that identifies whether or not the operation was successful.

```
Function delete (key, hashTableArray, tableSize)
   - Search for the item in the hash table array
   - If the item is not located return FALSE
   - If the item is located, delete the node as you would from a
     linked list and return TRUE
```

Lab Exercise

For this exercise, you will write C++ code to build a hash table using the hashfunction described above. You will also write code to handle inserting and searching for items in the hash table. Your hash table should contain movie structs as described at the beginning of this document.