CS-300

Code Reflection Flowchart 4-2

Hash Table Algorithms

Scott Dixon

The Hash algorithm is a powerful data structure that allows for efficient storage and retrieval of key-value pairs. I need to study more about it and become more familiar writing this type of code! It works by using a hash function to map keys to indexes in an array. When a value is inserted into the table, its key is hashed to determine the index where it will be stored. Then, the value is placed at that index in the array. When a value is retrieved from the table, its key is hashed to find the corresponding index in the array, and the value stored at that index is returned.

One of the main advantages of using a hash table is its constant-time performance for both inserting and retrieving values. This makes it a popular choice for many applications, including databases, caching systems, and compiler symbol tables. However, hash tables can have collisions, where two different keys map to the same index. To handle collisions, many hash table implementations use a technique called chaining, where each index in the array contains a linked list of values that hashed to that index.

I think a good analogy for this algorithm would be that the Hash Table is like a large box of drawers, each with a label on the outside. You can put items inside each drawer and find them quickly by looking at the label on the outside.

Just like the label on a drawer, each piece of data is associated with a key. The Hash Table uses a special function to generate an index from the key, which determines the location in the table where the data is stored.

This algorithm searches and retrieves data very quickly, even if you have a large amount of data to work with. Hash Tables are commonly used in computer programming to store things like usernames and passwords, or to keep track of which items are in a shopping cart on a website.

Hash Table Pseudocode:

initialize a hash table with a fixed size

for each item in the unsorted list:

hash the item to determine its index in the table

if the table[index] is empty:

add the item to the table at index

else:

resolve any collision by finding the next available empty slot

add the item to the table at the empty slot

create an empty list to hold the sorted items

for each item in the hash table:

if the item is not empty:

add the item to the sorted list

return the sorted list