Concordance Hash Table

Summary

Hash table data structure consists of an array like container with a hash function. An element is given a hash value based on the contents of the element. The hash function assigns a unique hash value or index to an element. The hashed index is the used to store the element in the array structure. Although a good hash function will try to create random and evenly distributable hash index values, collisions are inevitable in large data sets. That is why collision handling method is just as important as hash functions. There are many ways of handling collisions such as chaining or probing. The method we used for the assignment was quadratic probing. When a collision occurs, quadratic probing looks for an empty index based on squaring the collision count of the current hashing. Probing occurs until an empty index is found.

Hash Table vs Balanced BST

The biggest advantage hash table has against any data structure is the fact that average insertion and retrieval time of any element can be considered O(c). A good hash function and a decent collision handling method is all that is required. On the other hand a balanced tree is limited to O(log n) on those operations. One of the downside of a hash table is that the table requires a set number of indexes which means that a rehash operation is required to increase the size of the table. Rehash operation is very costly as all the previous elements will need to be rehashed into the new hash table. The balanced tree has an advantage of being able to keep appending elements without the need to resize.

Another advantage of the tree structure is that accessing elements in order is already built into the system. Printing out elements in order in a tree data structure only requires a simple choice of traversal options. Hash table does not store elements in order. This is can be an advantage and a disadvantage. Distributing the elements evenly allows for O(c) on operations but it makes it hard to access elements in order. That is why the words were not displayed in alphabetic order. A sort function that can sort parallel arrays is required in order to display the words alphabetically.

About Rehashing

A hash table using probing method will have a load factor. A load factor shows how full the hash table is. If the load factor exceeds certain percentage (usually around 50% ~ 60%) the time complexity of hash table cannot stay near O(c). A resize must occur in order to maintain the constant time advantage. In our assignment we used quadratic probing method for collision handling. Hash table size must be a prime number for such method to perform properly. Constraining the hash table size to prime numbers ensures that quadratic probing will find empty indexes more easily.

Operations for Hash Tables

Insert() O(c)

incCount() O(c)

rehash() O(n)

getMaxWord() O(n)

Good Hash Function vs Bad Hash Function

The concordance program took 1187 milliseconds to process 2,014,878 words using the ‘Jenkins one at a time’ hash functions. The concordance program took 6733 milliseconds to process the same amount of words when using a bad hash function. The bad hash function just added up the int values of each characters in the words using ACII values. The difference in runtime is over 500%.