Grace Portelance

Grace.portelance

431602

Lab 2 README

The first thing I did was create and implement my hash functions. I did this in the way suggested in class, using two irrational numbers. I knew I would be doubling my table size eventually, indicating that it would have an even number of slots after the first doubling, so I knew my function needed to be relatively prime to that. I ensured that STEP would always be odd in order to improve my hashing.

I then implemented FIND and INSERT with nearly identical code—I tracked the elements the same way, and wither returned that they were inserted or returned the record. REMOVE was simple, as I simply used my find operation to locate the record, and changed the key to a placeholder string.

To implement DOUBLESIZE, I created a void method that copied the elements of the current array into a temporary array, doubled the original array, and copied the elements back in using insert. I put a conditional at the top of my insert method, indicating it should not execute unless the load (1/4) in this case, was not yet reached. If it was exceeded, the method would be called and the table doubled.

Finally, I added a piece to the Record class that stored the HashValue of the key. I followed the advice on piazza and simply created a reference to the toHashKey method provided, being sure to update it when I inserted an element. Then, in find, I simply compared the toHashValue of the record to the toHashKey of the given string. If this was not the case, I knew for sure that the key was not at that index.

If they were the same, I then checked if the string “keys” were equal. By doing this, I was able to compare int values for the vast majority of checks, saving strong comparisons only for when the ints were equal. This saves time.