COMP202 Project 3:

Odysseas Stavrou – Project 3

COMP202 – Technical University of Crete

June 2020

Methodology:

Data Structure:

After careful consideration on which combination of Java's Data structures would be useful for this project, I ended up with using Vectors. For the simple reasons that they are very agile and provide much more functionality from other Data structures like Arrays or Array Lists. Also being generic, helps us with having a place to store our **Buckets** and manipulate them with ease.

The Hash Table is comprised by the Object itself and many Bucket Objects that form a Vector as mentioned above. Along with complimentary methods it can achieve the following:

- Insertion of a Random Key
- Search of Random Key
- Deletion of Random Key
- Split Buckets and Extend the Table to accommodate more Keys
- Merge bloated Buckets to save Size
- Bucket Overflow if multiple Keys Hash to the same Bucket

Splitting:

The principle of splitting it's rather quite simple. There is a pointer on a bucket and when it's time to split, the contents of that Bucket (and its overflows) are rehashed and split between the existing Bucket and the newly added one. This is achieved quite easily. All Keys are pulled from the designated Bucket and then the Bucket is reinitialized. After that all the items are reinserted, either on their old Bucket or the new one, using the help of the Hash2 Function which hashes keys with twice the length of the tree. As soon as the table has reached double its length then the pointer gets reset and also the N variable gets doubled, effectively changing the hash functions. (Hash2 becomes Hash1 and Hash2 hashes with double that)

Merging:

Merging is the reversed process described above. First a check is performed to see if the table is at double length. If that's the case then it gets halved and the pointer get placed in the correct place (just before splitting for the last time). Else it just reduces the pointer by 1. Just like splitting, all the keys from the LAST bucket are pulled and then the last Bucket is destroyed. Then the keys are reinserted to their origin bucket (the bucket that split last) using the Hash1 function. Note that merge cannot happen below the starting length of the table.

Inserting:

For inserting any random number first, the load factor of the table is calculated (including the new element) and if necessary, a split it performed. After that the new element is inserted in the proper index of the table depending on the output of the Hash functions and the position of the pointer p. Insertion is handled by the Bucket Object which creates any overflows needed.

Searching:

For searching we just need to use the proper hash function and call the search method of the specified Bucket. It searches in the Bucket including its overflows for appearances of the Key.

Deletion:

Just like Searching, firstly we need to find the index using the proper hash function. After that the remove method of that Bucket is called, which pulls all the keys from the Bucket and its overflows, removes the Key if existing and then reinserts them to the Bucket. Returns negative number if the Key wasn't found or positive if Key was found, so that the calling method knows whether to reduce the element counter. After deletion a Load Factor check is performed if the LF falls below 50% then a merge is performed.

Gathering Data and Producing Results:

For each test (search/deletion) 50 new random numbers are picked for each test for each data structure. Then the specific test is ran with the random numbers.

Exporting / Printing:

After running and gathering all the required results the program exports the results into a "graphs.m" script with all proper functions to plot on 3 different figures (BST, LF>50%, LF>80%) 3 different graphs (insertion, searching, deletion) using MatLab. I used Java's formatted output to print a tabulated output.

Usage:

```
java Main.class [filename]
```

Software Used:

- IntelliJ IDEA Ultimate (Student's License)
- Python 3.8 (for the test file generation)

Java Vs Python:

This is what I learned from programming in Python and later in Java

- Java compared to Python is stupidly fast. Seeing how much faster is Java I will probably keep it in mind for other Projects to come.
- Since I was more fluent in Python, I knew exactly how to approach each task rather than searching the web for how to do simple things in Java like file reading, random integers etc.
- Python is ready for programming out of the box, no main methods, no objects no nothing.
- Java is much more complicated that Python. Whether it's the Static Typing or the Visibility Modifiers, it felt that with Python less code was needed for the same functionality.