CIS 121

# Homework 08

Nathan Baker

Recitation 201

Collaborators: None

---

**Q1.**

```
dictionary.txt
--------------
10 iterations complete.
20 iterations complete.
30 iterations complete.
HashMap: Average running time: 79865 microseconds
HashMap: Average memory usage: 30484 kilobytes
10 iterations complete.
20 iterations complete.
30 iterations complete.
TrieMap: Average running time: 146936 microseconds
TrieMap: Average memory usage: 65806 kilobytes
10 iterations complete.
20 iterations complete.
30 iterations complete.
java.util.HashMap: Average running time: 52327 microseconds
java.util.HashMap: Average memory usage: 33218 kilobytes
10 iterations complete.
20 iterations complete.
30 iterations complete.
java.util.TreeMap: Average running time: 362057 microseconds
java.util.TreeMap: Average memory usage: 33903 kilobytes

phonenumbers.txt
-------------
10 iterations complete.
20 iterations complete.
30 iterations complete.
HashMap: Average running time: 641 microseconds
HashMap: Average memory usage: 761 kilobytes
10 iterations complete.
20 iterations complete.
30 iterations complete.
TrieMap: Average running time: 711 microseconds
TrieMap: Average memory usage: 331 kilobytes
10 iterations complete.
20 iterations complete.
30 iterations complete.
java.util.HashMap: Average running time: 516 microseconds
```

```
java.util.HashMap: Average memory usage: 824 kilobytes
10 iterations complete.
20 iterations complete.
30 iterations complete.
java.util.TreeMap: Average running time: 2108 microseconds
java.util.TreeMap: Average memory usage: 824 kilobytes
```

**Q2.** For 'dictionary.txt', my implementation of Hashmap had better space usage and run time than my implementation of TrieMap. However, for 'phonenumbers.txt', my implementation of HashMap had a slightly better runtime, but my implementation of TrieMap has a better memory usage. This was not what I expected. I expexcted the results to be the same in both scenarios. However, it makes sense that the space usage of TrieMaps work better in 'PhoneNumbers.txt' because the Trie data structure takes advantage of common prefix encodings, which are more frequent in phone numbers than words.

**Q3.** The compression of common prefixes for the TrieMap implementation was better for 'phonenumbers.txt' than 'dictionary.txt'. This is because phone numbers contain common prefixes more often than words in the dictionary. That is, there is more variation in 'dictionary.txt'. One way to be more space efficent would be to compress long single child chains into one final leaf node. This way, we don't need to store as many nodes to represent the same key.

**Q4.** In both scenarios, TrieMap uses O(n) space where n is the DATA set, but a lot less space is used to store the 'phonenumbers.txt' big-Oh notation gives us a solid upper bound but it does not tell us the full story. Understanding the algorithm and data structure well are imperative to knowing how things will actually play out.

**Q5.** For 'dictionary.txt': TrieMap: Average memory usage: 87431 kilobytes
**Absolute Savings:** 21,625KB
**Relative Savings:** 1.32x

For 'phonenumbers.txt': TrieMap: Average memory usage: 1282 kilobytes
**Absolute Savings:** 951KB
**Relative Savings:** 3.87x

In my opinion using this optimization saves a ton of space and is totally worth it, I would definitely recommend using it. These relative numbers demonstrate that at high numbrers the optimization can make a big difference in terms of saving space. This matters in the context of saving large clusters of DATA.

**Q6.** The JAVA 8 version of a HashMap utilizes a slower runtime, but a higher space usage than my implmentation of a HashMap for both 'dictionary.txt' and 'phonenumbers.txt'. This is due to the optimization as mentioned toward the begginging of the assignment. The store more in terms of a binary tree, so that accessing keys takes less time (lg(n) rather than n).

**Q7.** My TrieMap had almost double the space usage but half the runtime of the Java TreeMap. This makes sense as the balanced binary tree implementation probably uses less nodes in total to store the keys, but the height of the tree is longer and therefore, takes more time to traverse.